Deep Learning-based Underbody Vehicle Wireless Automated Inspection System

LEE TIAN HAU

BACHELOR OF MECHATRONICS ENGINEERING (HONS)

UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH

UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	LEE TIAN HAU
Date of Birth	:	18 NOVEMBER 2000
Title	:	DEEP LEARNING-BASED UNDERBODY VEHICLE WIRELESS AUTOMATED INSPECTION SYSTEM
Academic Session	:	SEM 2 2024

I declare that this thesis is classified as:

☐	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☒	OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

001118101275
New IC/Passport Number
Date: 16 June 2024

(Supervisor's Signature)

Dr Mohd Azraai bin Mohd Razman
Name of Supervisor
Date: 16 June 2024

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER

Librarian,
Universiti Malaysia Pahang Al-Sultan Abdullah, Lebuh Persiaran Tun Khalil Yaakob,
26300, Gambang, Kuantan, Pahang.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name	Lee Tian Hau
Thesis Title	Deep Learning-based Underbody Vehicle Wireless Automated Inspection System
Reasons	(i) The information and technical support sponsored by Volkswagen Group Malaysia Sdn.Bhd. are confidential (ii) The thesis author holds the agreement with the company which shall hold and maintain the confidential information in strictest confidence. (iii) The thesis author has to compensate the company for any leakage of information coming from this thesis.

Thank you.

Yours faithfully,

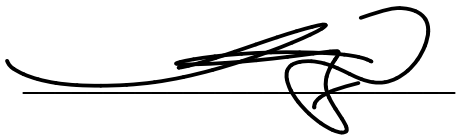
(UMPSA Supervisor's Signature)
Date: 16 June 2024
Stamp:

Note: This letter should be written by the supervisor and addressed to the Librarian, *Universiti Malaysia Pahang Al-Sultan Abdullah* with its copy attached to the thesis.

اونيۏرسيتي مليسيا قهڠ السلطان عبد الله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

SUPERVISOR'S DECLARATION

I/We* hereby declare that I/We have checked this thesis/project and in my/our opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of Bachelor in Mechatronics Engineering in cooperation of Universiti Malaysia Pahang Al-Sultan Abdullah.



(Supervisor's Signature)

Full Name : DR. MOHD AZRAAI BIN MOHD RAZMAN

Position : Senior Lecturer

Date : 16 June 2024

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang Al-Sultan Abdullah or any other institutions.

(Student's Signature)

Full Name : LEE TIAN HAU

Student ID : FB20053

Date : 16 June 2024

DEEP LEARNING-BASED UNDERBODY VEHICLE WIRELESS AUTOMATED INSPECTION SYSTEM

LEE TIAN HAU

Thesis submitted in fulfillment of the requirementsfor
the award of the degree of
Bachelor's Degree of Mechatronic Engineering (Hons)

Faculty of Manufacturing & Mechatronics Engineering Technology

UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH

2023/2024

ACKNOWLEDGEMENTS

To begin, I would like to offer my most sincere appreciation to my supervisor, Dr. Mohd Azraai B. Mohd Razman, for placing his steadfast confidence in me and allowing me the opportunity to investigate and develop my own thoughts and concepts. His academic advice and technological experience were extremely helpful in determining the path that this research would go and the findings that it would provide. I am also thankful to him for giving me the required resources, which were absolutely necessary in order to carry out the experiments and carry out the study.

I would like to express my sincere gratitude to my industry supervisors at Volkswagen Group Malaysia Sdn Bhd., namely Faiz bin Muhammad Mohd and Mohd Syazrulazwan bin Mohd Zin. Their assistance, confidence and the chances to cooperate on practical problems have enhanced my research experience.

In addition, I would want to express my gratitude to every single member of the IMAMS Lab for their contributions. This study endeavor has been significantly enriched as a result of their perceptive conversations, constructive criticism, and spirit of collaboration.

Next, I want to express my profound appreciation to my friends and family members for the continuous support and encouragement they have provided during this journey. Their compassion, tolerance, and unwavering faith in my abilities have been an endless well of inspiration for me.

Lastly, I would want to offer my gratitude to everyone who has helped in any way to bring this project to a successful conclusion, whether it was through their direct or indirect involvement. I am indebted to you for your help due to the fact that the accomplishment of this research would not have been possible without the support and encouragement that you provided throughout the process.

ABSTRAK

Memandangkan industri automotif memainkan peranan penting di Malaysia, adalah penting bagi negara untuk mewujudkan kaedah pemeriksaan yang tepat untuk kenderaan. Ini berikutan bilangan kenderaan di jalan raya semakin bertambah, seterusnya membawa kepada peningkatan jumlah kemalangan yang berlaku di jalan raya secara keseluruhan. Melalui pembentangan sistem pemeriksaan kenderaan bawah badan yang inovatif yang telah dicipta khusus untuk model Tiguan Allspace Highline, penyelidikan ini menangani keperluan mendesak yang telah dikenal pasti. Kaedah YOLO digunakan bersama dengan pendekatan pembelajaran mendalam untuk mencapai matlamat sistem. Objektif utama adalah untuk meningkatkan keselamatan keseluruhan, mengurangkan bilangan ralat yang disebabkan oleh manusia, memaksimumkan keberkesanan pentadbiran data, dan meningkatkan keberkesanan pemeriksaan manual di mana-mana sahaja yang boleh dilakukan. Penyelidikan ini termasuk aplikasi komprehensif algoritma YOLO pada set data yang telah dikumpulkan secara khusus, penciptaan antara muka yang mesra pengguna melalui penggunaan Streamlit, kemudahan penangkapan imej tanpa wayar untuk pengesanan objek yang tepat, dan penggabungan pangkalan data MySQL yang boleh dipercayai untuk pentadbiran data dengan cara yang lancar. Data empirikal adalah sangat baik, dengan YOLOv9c mencapai ketepatan terbaik (0.983) dan ingat semula (0.961), diikuti rapat oleh YOLOv7 dengan ketepatan 0.977 dan ingatan 0.950. Kedua-dua nilai ketepatan dan ingatan ini agak menggalakkan. Sebaliknya, walaupun pada hakikatnya ia berkesan, YOLOv8l menunjukkan tahap ketepatan yang agak rendah (0.959) dan ingat semula (0.958). Penyelidikan ini memberikan sumbangan besar kepada industri automobil dengan mencadangkan pendekatan tanpa wayar untuk pemeriksaan kereta bawah badan yang sangat cekap, automatik dan menjimatkan masa. Akibat daripada ini, ia menandakan piawaian baharu untuk ketepatan dan kebolehpercayaan dalam penyelenggaraan kenderaan dan pelaksanaan langkah keselamatan.

ABSTRACT

Due to the fact that the automotive industry plays a vital role in Malaysia, it is essential for the country to establish precise inspection methods for vehicles. This is due to the fact that the number of vehicles on the road is growing, which in turn is leading to an increase in the number of accidents that take place on the roads overall. Through the presentation of an innovative underbody vehicle inspection system that has been created specifically for the Tiguan Allspace Highline model, this research addresses the urgent need that has been identified. The YOLO method is utilised in conjunction with deep learning approaches in order to accomplish the system's goals. The primary objectives are to improve the overall safety, reduce the number of errors that are caused by humans, maximise the effectiveness of data administration, and improve the effectiveness of manual inspections wherever it is practicable to do so. The research includes the comprehensive application of the YOLO algorithm on a dataset that has been specifically gathered, the creation of an interface that is user-friendly through the utilisation of Streamlit, the facilitation of wireless image capture for accurate object detection, and the incorporation of a reliable MySQL database for the administration of data in a seamless manner. The empirical data are excellent, with YOLOv9c achieving the best precision (0.983) and recall (0.961), followed closely by YOLOv7 with a precision of 0.977 and memory of 0.950. Both of these precision and recall values are quite favourable. On the other hand, despite the fact that it is effective, YOLOv8l demonstrates a somewhat lower level of precision (0.959) and recall (0.958). This research provides a big contribution to the automobile industry by proposing a wireless approach for underbody car inspections that is exceptionally efficient, automated, and saves time. As a consequence of this, it marks a new standard for precision and dependability in the maintenance of vehicles and the implementation of safety measures.

Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUC	Area under the curve
CA	Classification Accuracy
CPU	Central Processing Unit
DSSD	Deconvolutional Single Shot Detector
Eq	Equation
GPU	Graphical Processing Unit
MCC	Matthews Correlation Coefficient
mAP	Mean Average Precision
OS	Operating System
PCB	Printed Circuit Board
Prec	Precision
RAM	Random Access Memory
VS	Virtual Studio
YOLO	You Only Look Once

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The purpose of this chapter is to provide a brief summary of the conceptual and theoretical basis of this thesis. This project primarily aims to employ deep learning-based image processing algorithms to examine underbody of vehicles. In the beginning of this chapter, we will provide a detailed explanation of the research environment, problem statement, research project objective and research project scope.

Frequently, vehicle inspection systems are utilized by manufacturers of cars, providers of maintenance services, and authorities in charge of traffic regulation in order to assess the level of safety and emissions produced by vehicles [1]. When it comes to assuring the safety, dependability, and longevity of the vehicle, the examination of the underbody of the vehicle is an essential component that plays a major role.

Due to the fact that they are frequently exposed to a wide variety of road conditions, the underbodies of automobiles are vulnerable to damage because of their close proximity to the road. The gas tank, suspension and exhaust system are frequently affected by common faults. These defects are generally ascribed to factors such as the usage of road salt for melting snow and ice, worn-out shock absorbers, damage caused by mud and collisions from debris on roadways [4]. When left unchecked, these variables have the potential to cause significant damage to the underbody of the car, which can result in costly repairs and put the vehicle's safety at risk.

The current method of inspecting underbody cars depends on visual inspection, which results in different levels of reliability depending on factors such as the amount of time spent examining the vehicle and the inspector's competence and training level [5]. The first approach that was introduced to improve the inspection of underbody cars featured the utilization of a mirror that was affixed to the tip of a rod [6], as depicted in Figure 1.1. This technology is mostly employed at security checkpoints to facilitate efficient and rapid detection of irregularities. Nevertheless, it is accompanied by intrinsic limits. Initially, if the inspector exhibits sluggishness in identifying explosives, there persists a susceptibility to detonation prior to the completion of the examination. This worry holds significant importance due to the

requirement for the inspector to maintain close physical proximity to the vehicle. Additionally, the mirror's physical limitations lead to a coverage of around 40-50% of the vehicle's underbody. The task of accessing the central axis of the vehicle presents difficulties, resulting in the presence of oblique viewing angles. In addition, although the mirror approach is cost-effective, it is restricted to visual examination and does not have aspects that are suitable for automation [6].

Figure 1.1 Inspection of underbody vehicle using mirror method

1.2 Research Background

The automobile sector in Malaysia holds the position of being the third largest among other ASEAN countries, with a sales volume of approximately 500 thousand motor vehicles [2]. Nevertheless, the proliferation of automobiles on roadways has led to a notable apprehension over road accidents. According to the Transport Ministry's report, the number of accidents in 2021 amounted to 370,286, while in 2022, the number of accidents increased to 545,588, resulting in 4,539 and 6,080 fatalities, respectively. According to a study undertaken by the Malaysian Institute of Road Safety Research, it has been determined that road accidents are mostly caused by human behaviour. Additionally, the design and condition of road infrastructure, as well as the state of vehicles, have been recognised as significant factors [3]. Hence, it is imperative to address the deficiency of precise vehicle inspection systems in Malaysia in order to mitigate the probability of defective underbody components leading to road accidents.

After the explanation of mirror method for vehicle inspection in section 1.1, apparently that mirror method shows that it is inefficient and consists of limitations like difficulty in inspecting deeply or requires manpower. Now, the integration of contemporary breakthroughs into the assessment of underbody cars represents a notable progression in augmenting their efficacy. An illustrative instance is the incorporation of sensors [6]. The proposed methodology entails underground placement of cameras in order to capture images as the target vehicle moves across the sensor suite. In a similar vein, a mobile robotics platform known as SafeBot was developed with the purpose of navigating beneath a vehicle as shown in Figure 1.2 which is the utilisation of autonomous control for under-vehicle inspection with SafeBot. The four tyres symbolise a random arrangement of a vehicle. The SafeBot utilises a 3D range brick to accurately determine its position and subsequently devises a trajectory that encompasses the entirety of the vehicle. Both of these solutions offer the benefits of remote inspection and the automation's flexibility. However, these methods are still inadequate in adequately capturing the entirety of the vehicle's underbody, as the camera's view is limited to a tiny portion of the underbody due to the car's low ground clearance [6].

Figure 1.2 Autonomous mobile robotics platform, SafeBot to capture images based on video frames

Given the aforementioned issues, a novel system employing 3D scanning technology is proposed as a means to overcome the constraints associated with the aforementioned approach [6]. The aforementioned method is capable of producing a comprehensive composite image of the underbody of automobiles [5]. By doing this, it provides a more comprehensive and meticulous examination, improving the precision and efficiency of the overall evaluation of the

underbody. Although this system has made significant progress, it still necessitates the involvement of an inspector to visually inspect and manually detect anomalies in the generated images [5]. However, there are proponents who contend that capturing images of an object, particularly larger items such as cars, is considered to be a faster and more cost-effective alternative to the process of 3D scanning [7].

The introduction of multiple algorithms capable of effectively identifying and categorising structural elements can be attributed to recent advancements in deep learning. The advancements in technology have enabled the automation of processes, eliminating the need for manual comparison of processed picture outcomes with a presumed constant baseline. Deep learning has demonstrated promising results in various domains, including autonomous driving [8], [9], data mining [10], literary translation [11], and medical applications [12], [13], [14]. Image categorization is a prevalent application in the domain of deep learning, serving a pivotal function in the detection of flaws in automotive components. There are two primary methodologies employed in the field of image classification: Support Vector Machine (SVM), which incorporates machine learning techniques, and Convolutional Neural Network (CNN), which relies on deep learning principles [15]. Multiple tests and research have consistently shown that Support Vector Machines (SVM) excel at acquiring knowledge from small sample sizes, producing impressive outcomes. However, Convolutional Neural Networks (CNNs) are particularly suitable for training with large datasets, since they effectively extract data features and achieve precise and efficient classification.

Unlike image classification that pertains to single-object image, object detection involves classification and location of multiple objects within an image [16]. It applies classification to distinct objects and uses bounding boxes [17]. While image classification focuses on classifying an entire image by object category, object detection involves tasks such as foreground extraction, static region analysis, and blob classification, demonstrating the multifaceted nature of object detection [18]. Accuracy assessment is crucial in both image classification and object detection, with the former aiming to classify unknown images by object category and the latter focusing on identifying and localizing objects within images [19]. Other than expediting processes by giving computers the ability to learn and make predictions based on the data and information that is fed to it and also through real-world interactions and observations [20], object detection, as exemplified in maritime ship inspections, also streamlines remote assessments by predicting and tracking potential issues from videos, enhancing efficiency and minimizing manual inspection risks [21]. On top of that, object

detection is a fundamental step for automated video analysis in various vision applications, showcasing its potential for fully automated processes [22].

This project aims to improve the inspection process of underbody cars by incorporating deep learning, notably utilising the You Only Look Once (YOLO) algorithm, to overcome the issues associated with current inspection procedures. The goal is to enhance uniformity and precision by implementing automation in the inspection procedure. The effective data feature extraction and speedy, exact classification capabilities of YOLO have the potential to greatly improve inspection efficiency, especially in situations with a large number of cars.

The YOLO method facilitates the detection of objects in real-time, hence offering the potential for remote examination. Streamlit has been chosen as the designated user interface (UI) in this particular scenario to provide smooth interaction with the underbody inspection system. The objective of integrating YOLO and Streamlit is to optimise the user experience and enhance the efficiency of inspection data processing.

This research utilises a MySQL database to store essential information, such as inspection findings, timestamps, and other pertinent data, in order to ensure thorough data management and integrity. The implementation of this methodical database strategy ensures both the dependability of data and the establishment of a structured framework for the methodical retrieval and administration of inspection records. The primary objective of this study is to enhance the approaches used for inspecting underbody cars by strategically using deep learning technology, real-time object identification, and efficient data management processes.

1.3 Problem Statement

Numerous urgent problems with manual inspection techniques highlight the need to reinvent car underbody inspection procedures. While useful in some situations, traditional approaches have numerous disadvantages that affect productivity, safety, and dependability in the automobile sector.

Firstly, the process of manual underbody vehicle inspection highly consumed large amounts of time with more assistance required in manpower. Followed by the study conducted for defect inspection of crankshaft, it is highly time consuming the fact that the workers spent approximately 45 seconds on each single piece for manual inspection, not to mention that the workers will eventually be tired and took longer time than 45 seconds to check each single

piece [23]. Other than that, it is formidable to undergo manual inspection if professional skill is not high enough, this situation eventually leads to poor efficiency and effectiveness in inspection process due to the reliance on professional with expertise level to do manual inspection [24]. Over dependent on manual inspection can cause delays or slowing the overall production rate of vehicles and it required higher cost due to hiring professionals with expertise in manual vehicle checking.

Secondly, it is undeniable that manual inspection can cause human error due to various reasons like distraction, tiredness, over work load, etc [25]. It can bring false alarms towards the maintenance team which unnecessary repair and extra cost are given for repairing what is not broken along with the possibility of overlooking critical issues. Fatal consequences could occur ranging from financial lost due to damage car to human safety risk like road accidents, that is why that a more effective and reliable approach are needed to rectify potential risk that can occur if underbody vehicle inspection are not efficiently conducted.

Next, security issues are still another important factor that is impossible to emphasize. Workers who perform manual underbody inspections are subject to a number of risks, such as the possibility of accidents and long-term health problems. One study supports this by highlighting the fact that employing unmanned aerial vehicles (UAVs) to assess cracks in bridge piers based on images removes the risk associated with manual inspection [26]. Thus, during underbody examinations, it is imperative to create and implement automated visual inspections that minimize human engagement with potential dangers.

Not to mention, there are many difficulties with the conventional method of keeping handwritten data records for inspection results. This approach not only increases the risk of data loss due to physical record degradation or misplacement, but it also makes it more difficult to analyze and follow inspection data over time [27]. In the absence of an efficient data management system, identifying patterns or recurring problems becomes an enormous undertaking, resulting in reactive rather than preventative maintenance plans.

Ultimately, the existing situation, marked by laborious procedures, human mistakes and insufficient data handling, clearly need a complete revamp of the vehicle underbody inspection technique. Through the utilisation of technological advancements, the sector has the potential to improve the precision and effectiveness of inspections, while also effectively reducing the dangers and constraints associated with manual approaches.

1.4 Aims and Objectives

The objective of this study is to develop a system that use a deep learning methodology to identify irregularities in the undercarriage of an automobile. To achieve the desired outcome, a set of distinct objectives has been formulated.

Several objectives are:

1. To identify defect type of vehicle model, collect dataset, pre-process and train a model for underbody vehicle inspection.
2. To develop a constructive system website with step-by step instruction for underbody vehicle inspection
3. To enhance the system by upgrading object detection to wireless defect inspection
4. To design a digital database for data management and display.

1.5 Research Scope and Delimitation

Research Scope

The objective of this project is to address the difficulties related to car underbody inspections by creating a sophisticated anomaly detecting system. The scope of the research comprises a number of primary aims.

This project attempts to accomplish precise and effective defect identification by detecting several kinds of defects and using pre-processed data to train YOLO algorithms. A strong inspection system is developed on the basis of the identification and classification of these problems. The system's ability to precisely identify underbody car issues, through the use of YOLO algorithms, overcomes the drawbacks of manual inspections.

A key aspect of the project is the development of a real-time wireless defect inspection graphical user interface (GUI) using Streamlit. This GUI not only enhances the user experience by providing step-by-step instructions for underbody inspections but also facilitates real-time monitoring and analysis. The wireless capability ensures that inspections can be conducted without the constraints of physical connections, increasing the flexibility and convenience of the inspection process.

Another essential element is effective data management, which is accomplished by including a MySQL database. The systematic management, retrieval, and storing of inspection

data is made possible by this database, guaranteeing that all data is readily available and properly arranged. The data management system is more scalable and reliable when MySQL is used, which makes it appropriate for managing massive amounts of inspection data.

The project's final goal is to integrate the data management and user interface (UI) components for practical system deployment. This integration guarantees that the system is practical and easy to use in addition to being technically solid. With a streamlined user interface, strong data management, and the powerful detection capabilities of the YOLO algorithms combined, the system provides a complete underbody car inspection solution that is ready for real-world use.

Delimitation

The only underbody inspection of the Tiguan Allspace Highline model is the subject of this study, which can restrict the applicability of the conclusions to other car models. Although the particular model selected guarantees a controlled environment for system development and testing, it might not take into consideration changes in other car designs.

The geographic region in which the dataset used to train the YOLO algorithms was gathered may not have included all conditions or defect kinds that could exist worldwide. This geographical restriction may have an impact on how well the model performs in various settings or circumstances.

The graphical user interface (GUI) developed with Streamlit is customized for this particular application, and although Streamlit offers a reliable platform for monitoring in real-time, it might not offer as much customization as other platforms. While functionality and usability are balanced, this decision may limit some of the more sophisticated capabilities offered by competing GUI frameworks.

For this specific project, the integration of a MySQL database was selected due to its effectiveness and dependability in handling data. This decision, however, leaves out alternative database systems that might provide distinct advantages, including NoSQL databases for unstructured data or alternative SQL databases with various performance features.

The possible effects of wireless signal problems or network delay on the inspection system's real-time performance are not examined in this study. Although wireless capacity increases flexibility, it does so under the assumption of a stable and dependable network—a requirement that may not always hold true in practical situations.

Lastly, the system is intended to be deployed in settings with easily accessible digital infrastructure and support. While this emphasis guarantees top performance, it might not adequately handle issues in places with little access to technology or assistance, which could reduce the system's usefulness in certain situations.

1.6 Thesis Outline

Chapter 1 functions as an initial exposition, offering a comprehensive understanding of the research's context, elucidating the research topic, and outlining the study's aims and scope. This chapter establishes the groundwork for the future investigation by providing a thorough comprehension of the setting in which the study is conducted.

Chapter 2 provides an in-depth examination and integration of pertinent prior research, theories, and concepts that are relevant to the subject of study. This chapter provides a thorough analysis of the available literature to clarify the theoretical framework and place the findings within the wider scholarly conversation. Chapter 2 makes a valuable contribution to the establishment of a strong theoretical framework for the study by integrating a range of views.

Chapter 3 provides a comprehensive account of the research design, offering a thorough explanation of the techniques, data gathering methods, and analytical procedures utilised in the study. This chapter elucidates the methodical methodology employed to tackle the study objectives. Chapter 3 provides an analysis of the research design, so enhancing the understanding of the study's methodological rigour and the validity of its findings.

Chapter 4 provides a comprehensive overview of the research endeavours, highlighting the outcomes obtained through the analysis, interpretation, and deliberation of the data. This chapter elucidates significant discoveries and patterns revealed during the study process by presenting empirical evidence. In addition, Chapter 4 provides a thorough analysis of the consequences of the findings in relation to the research problem, therefore adding to the progress of knowledge in the field.

Chapter 5 functions as the apex of the research expedition, encompassing the principal discoveries, formulating conclusions, and providing suggestions for forthcoming study undertakings. This chapter consolidates the knowledge obtained from the previous chapters, offering a thorough summary of the study's contributions and consequences. Chapter 5 enhances comprehension of the research's importance and provides opportunities for additional investigation by examining the research procedure and results.

1.7 Summary

The research background highlights the disadvantages of old ways of checking underbody vehicles with the increment of vehicles production and accidents among ASEAN countries which draws to a conclusion saying that inspection of underbody vehicles has to be enhance to overcome the challenges. There are several issues which also mentioned in the problem statement which highlights the reason why we are trying to develop an anomaly detection using deep learning for checking the underbody of vehicles which is our main objective of this project also.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

From chapter 2, it gives an examination of the fundamental principles employed in the research. This chapter will primarily examine the specifications, parameters, methodologies, and studies conducted by previous researchers, with the aim of addressing existing gaps in knowledge. Moreover, the primary objective of this chapter is to provide a comprehensive examination of fundamental algorithms, while also delving into their practical implementations in other domains. This research aims to acquire insights into the ways in which sophisticated computational techniques contribute to notable enhancements in efficiency and accuracy within various domains. Additionally, it will address potential problems and constraints encountered during the implementation process.

2.2 Defect Detection

2.2.1 Defect Detection with Traditional Image Processing Techniques

Within the domain of image processing and feature extraction, numerous innovative methodologies have significantly influenced the development of anomaly detection systems. An exemplary method in this category is the Snake algorithm, which was released in 1987. It brought about a significant change in picture segmentation by utilising energy-minimizing splines that are regulated by a combination of external constraint forces and internal image forces. The aforementioned methodology demonstrated a notable enhancement in the localization of features, specifically lines and edges, resulting in an enhanced accuracy in detecting anomalies [28]. Furthermore, a comparative research conducted in the field investigated four texture algorithms that were developed in the 1980s. Among these algorithms, the spatial grey level dependence technique (SGLDM) stood out due to its strong and reliable texture analysis capabilities [29-33]. In the 1990s, the development of discrete and continuous wavelet transforms was a significant breakthrough in feature extraction technology. These transforms enabled detailed image analysis using efficient computational methods such as Fast Fourier Transforms (FFT) and fast finite impulse response (FIR) algorithms [34-35]. The lasting significance and flexibility of wavelet

transforms remained in 2009, highlighting their ongoing use in contemporary image processing [36].

In recent times, there has been a significant emphasis on the utilisation of artificial intelligence methodologies for the purpose of defect identification. An example of this is a defect detection system designed for textile fabric that uses multi-resolution decomposition in conjunction with an Artificial Neural Network (ANN) to classify faults in real-time [37]. This device demonstrated efficacy, even in industrial environments, in detecting flaws with low contrast. In contrast, the utilisation of pulsed thermography and perceptron neural networks was employed for the purpose of detecting corrosion in aluminium. This approach achieved a noteworthy accuracy rate of 99% under controlled experimental conditions [38]. Furthermore, there is an ongoing investigation into optimisation techniques for artificial neural networks (ANNs), including the substitution of back-propagation with population-based search algorithms such as the bees algorithm [39]. The aforementioned approach exhibited a good level of competence in identifying defects in wood. However, it is important to note that additional study is required to properly use genetic programming techniques for accurate assessment and optimisation in this particular domain [40].

It is noteworthy to emphasise that the aforementioned advanced genetic optimisation strategies did not exhibit superior performance compared to the conventional back-propagation method, hence raising concerns over their practical efficacy [40]. In addition, a study employed a cepstral technique, which utilises either a discrete wavelet transform (DWT) or a discrete cosine transform (DCT), to examine radiography pictures for the presence of welding errors. However, this method was only evaluated on a limited sample size of 16 images. The limited sample size necessitates caution when making generalisations, despite the method's ability to achieve a maximum accuracy of 100% in spotting faults [41].

In addition, another study endeavour utilised mathematical methodologies to reveal concealed intricacies in ancient manuscripts through diverse imaging techniques such as thermography and X-ray [42]. This approach demonstrated potential in multiple case studies. A distinct experiment was conducted with the objective of detecting flaws in apples by employing a shallow MLP-Neural Networks. The experiment involved the analysis of many variables, including colour and texture [43]. In addition, the utilisation of principal component analysis (PCA) was employed to reduce dimensionality, resulting in a significant accuracy rate of 89.9% on a restricted dataset employed for technique evaluation

[44]. A study utilising thermal imaging was conducted to detect material faults and quantify their depth, employing similar methodologies. This also entailed a rudimentary neural network and dimensionality reduction techniques such as Principal Component Analysis (PCA).

The aforementioned conventional image processing techniques predominantly utilised shallow networks and did not incorporate deep learning technologies. This highlights a deficiency in both accuracy and efficiency, which necessitates the adoption of deep learning approaches to enhance anomaly detection with more precision. Nevertheless, the comparison between conventional methods and newer deep learning algorithms continues to face hurdles, principally stemming from the lack of standardised benchmarking datasets such as COCO or ImageNet for traditional methods. Furthermore, it should be noted that classical classification algorithms are not universally assessed using contemporary benchmarking datasets. Therefore, in order to make an accurate comparison of various methodologies without any benchmark findings, it is important to conduct individual experiments and benchmarking.

2.3 Deep Learning

The establishment of deep learning concepts in conjunction with neural networks occurred some decades ago. The five-layer classifier, LeNet5, was first developed by researcher LeCun and colleagues in 1998 using Convolutional Neural Network (CNN) technology [45]. In recent years, deep learning has made considerable gains because to large improvements in processing power and the fast growth of big data. This methodology is dependent on comprehensive datasets that have been gathered within a particular domain, where the availability of substantial quantities of information holds significance.

In the realm of neural networks, the term "deep" pertains to the inclusion of multiple layers that are specifically meant to replicate the functions of the human brain. The advent of high-performance Graphics Processing Units (GPU), Application-Specific Integrated Circuits (ASIC) accelerators, cloud storage solutions, and advanced computing infrastructures has enabled the collection, organisation, and analysis of substantial amounts of data. This is essential because large datasets are required to successfully tackle overfitting problems in deep learning systems. Moreover, the enhanced processing capabilities greatly decrease the time needed for the strenuous training stages.

To promote deep learning in the field of computer vision, it is crucial to have a large collection of pre-existing image examples. ImageNet is a prime example [47]. This research has made a significant contribution by developing a comprehensive database that includes fish species found in marine waters. The purpose of this database is to improve both the training and testing procedures. Nevertheless, the importance of utilising a proficient learning algorithm should not be underestimated. In contrast to traditional approaches in the fields of computer vision and image processing, which have encountered challenges in accurately extracting features, deep learning techniques provide a significant benefit by enhancing existing techniques using sophisticated neural networks [47].

Deep learning approaches are increasingly being incorporated into numerous fields, providing significant advantages over traditional algorithms in areas like computer vision and object detection. The introduction of deep learning techniques has significantly improved the functionality of various robotic systems. AlphaGo, developed by Google, exemplifies a notable case when it examined human learning patterns and later engaged in competition with esteemed Go players [46].

2.3.1 Hardware

The exceptional ability of deep learning to handle and analyse large datasets, particularly for applications like automobile underbody inspection systems, relies heavily on the computational strength of the hardware that powers these artificial intelligence models. The performance of deep learning frameworks is significantly impacted by the underlying hardware configuration, including GPUs, TPUs, and CPUs [54]. These components play a crucial role in enhancing the computational capabilities required for processing intricate neural network algorithms.

The utilisation of Graphics Processing Units (GPUs) has played a pivotal role in the progression of deep learning hardware, principally owing to its capacity to concurrently do several elementary computations. GPUs were first developed for the purpose of producing images in video games. However, they have exhibited remarkable efficacy in performing matrix and vector operations, which are fundamental elements of deep learning algorithms. The capacity to analyse data in parallel greatly speeds up the training process of deep learning models [55]. NVIDIA, a prominent supplier of GPUs for deep learning, offers specialised modules such as the Tesla and Titan series, specifically designed to augment

computational capabilities for AI applications. The utilisation of GPUs in Deep Convolutional Neural Networks (DCNNs) offers significant benefits in terms of parallelism and efficient acceleration, hence playing a pivotal role in the advancement of deep learning. Furthermore, the extensive acceptance and success of Convolutional Neural Networks (CNNs) within the academic community can be primarily ascribed to the availability of high-performance computing hardware, such as Graphics Processing Units (GPUs), as well as a wide range of user-friendly open-source frameworks like Caffe, Torch, and Theano, which have been specifically designed to maximise GPU utilisation [56-57].

Nevertheless, GPUs possess certain constraints, namely in relation to memory, despite their noteworthy contributions. The constraints of GPU memory provide challenges in training extensive models on a single GPU. Furthermore, in order to optimise the use of all GPUs, it is necessary for systems to augment the batch size, hence impeding statistical efficiency. Furthermore, due to the rapid advancement in GPU processing capabilities, the transmission of data across GPUs has emerged as a major hindrance to the overall efficiency of training.

Tensor Processing Units (TPUs), created by Google, have had a substantial influence on the domain of deep learning. TPUs are specialised microchips meant to enhance the speed of deep learning tasks. They are specifically designed to support TensorFlow, an open-source deep learning framework developed by Google. These chips are optimised for high-throughput operations during both the training and inference stages of deep learning models [58]. TPUs provide exceptional performance in activities that necessitate a substantial amount of low-precision calculations, such as training neural networks. They provide notable enhancements in processing speed and energy efficiency compared to traditional CPUs and GPUs. The utilisation of TPUs has been seen to exhibit an average speed that is 15 to 30 times greater than that of modern GPUs and CPUs, so underscoring their substantial influence on the efficacy of deep learning models [59].

Central Processing Units (CPUs) are of utmost importance in the deep learning framework, especially for jobs that do not necessitate parallel processing or for smaller models where the additional resources required for GPUs or TPUs may not be warranted [60]. In certain scenarios, central processing units (CPUs) may be more appropriate than GPUs/accelerators as the primary processor for processing deep neural networks (DNNs), despite the fact that GPUs have greatly decreased training time and have been the dominant

choice for deep learning applications [61]. Furthermore, it has been observed that current deep learning models, which are extensively utilised, are unable to effectively handle precise real-time forecasting tasks. This limitation arises from the existence of inconsistent and nonlinear workloads within cloud computing systems. Consequently, there is a requirement for CPU-based solutions in specific circumstances [62].

2.3.2 Libraries and Tools

Specialised software known as deep learning tools and libraries are specifically developed to streamline and accelerate the process of creating deep learning models. These tools offer a structured approach that enables developers and academics to construct intricate neural networks without delving into the mathematical complexities. Libraries like TensorFlow, PyTorch, Keras, and others have become closely associated with the development of deep learning. They provide a combination of user-friendly features, adaptability, and strong capabilities [48].

TensorFlow, created by Google, is renowned for its robust computing capabilities and scalability, rendering it appropriate for both research endeavours and operational systems [49]. In contrast, PyTorch is renowned for its dynamic computing networks that enable streamlined modelling and efficient iteration, making it particularly attractive in the realm of research [50]. Keras, an API for neural networks at a high level, is designed to enhance the accessibility and use of TensorFlow, while maintaining its power and capability [51].

Furthermore, there exist libraries that concentrate on particular domains of deep learning applications. OpenCV provides real-time computer vision techniques [52]. An additional illustration is NLP libraries like as NLTK and spaCy, which specifically concentrate on human language data. These libraries offer resources for constructing programmes capable of interacting with both text and speech [53].

2.4 You Only Look Once (YOLO)

The identification of objects in real-time has emerged as a crucial component in various domains, such as autonomous cars, robotics, video surveillance, and augmented reality. The YOLO algorithm, created by Joseph Redmon and his team, was introduced at CVPR 2016 [63]. The technology known as YOLO, which stands for "You Only Look Once," is a notable departure from previous methodologies as it enables detection with a solitary

network pass. This stands in stark contrast to previous approaches that utilised sliding windows in conjunction with classifiers that were executed hundreds to thousands of times per image or more advanced two-step procedures. In the former, the initial step involved identifying regions that potentially contained objects, followed by classification in the second step. In contrast to Fast R-CNN [64], YOLO adopted a more straightforward methodology by relying exclusively on regression to forecast detection outputs as shown in figure 2.1. Fast R-CNN utilised two separate outputs, one for classification probabilities and another for the bounding box coordinates.

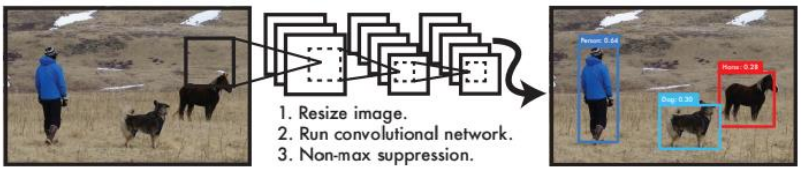


Figure 2.1 System of YOLO Detection

Various approaches have been developed in the field of object identification and picture analysis, in connection with the YOLO architecture as shown in figure 2.2. The utilisation of methodologies like as R-CNN (Region-based Convolutional Neural Networks) [66], together with its subsequent advancements, Fast R-CNN [64] and Faster R-CNN [67], has significantly contributed to the improvement of accuracy in object detection tasks. These two methodologies have significantly improved the accuracy of object detection. Their operation involves a two-step procedure: first, a technique of selective search generates propositions for specific locations, and then convolutional neural networks are employed to classify and enhance these areas.

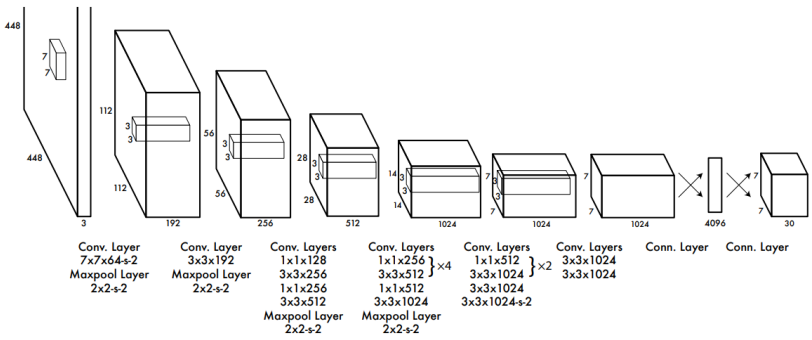


Figure 2.2 Architecture of YOLO

The YOLO framework stands out among the various object identification algorithms due to its notable combination of speed and accuracy, which enables efficient and reliable

object recognition in photos. Since its inception in the field, subsequent versions of the YOLO architecture have gradually enhanced their predecessors by addressing limitations and enhancing overall effectiveness (refer to Figure 2.3).

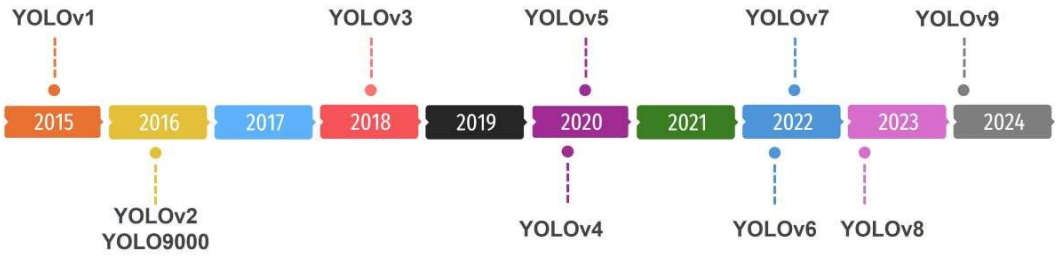


Figure 2.3 Timeline of YOLO versions throughout the years

YOLO algorithm is considered for the used of object detection in the inspection of underbody verhicle due to its marvellous accuracy and speed. We believe that YOLO could have the fastest speed in majority of object detection projects although there are other models that shown almost similar accuracy on different datasets.

2.4.1 Various Application of YOLO

The real-time object detection skills of YOLO play a crucial role in the development of self-driving automobile technologies. These capabilities enable the prompt identification and surveillance of various things, such as cars, pedestrians, bicycles, and others [72, 73, 74, 75]. These properties have been widely utilised in various domains, including the identification of actions in video material [76] for security purposes [77], the analysis of sports performances [78], and the improvement of human-computer interactions [79].

YOLO models have been utilised in the manufacturing sector for the purpose of surface inspection, with the aim of detecting flaws and anomalies. This application has resulted in enhanced quality assurance within production workflows [90], [91], [92]. These models have proven to be useful in several traffic-related applications, such as licence plate identification [93] and the recognition of traffic signs [94]. As a result, they have contributed to the advancement of intelligent transportation systems and the improvement of traffic regulation techniques.

YOLO models have been employed in the field of remote sensing to perform object recognition and classification on satellite and aerial imagery. These models have made significant contributions to many applications such as land use mapping, urban development

planning, and environmental surveillance [86], [87]. The integration of YOLO models into security frameworks enables the real-time monitoring and analysis of video feeds, hence expediting the detection of atypical behavioural patterns [88], implementation of social distancing protocols, and identification of face mask utilisation [89].

Within the field of agriculture, these technologies have been employed to identify and categorise crops, pests, and illnesses, hence improving precision agriculture methods and streamlining farming operations [80], [81]. YOLO designs have been instrumental in the healthcare industry for tasks like as cancer detection [82], skin region segmentation [83], and pill recognition [84]. This has improved the accuracy of diagnoses and simplified the process of treatment. Moreover, the aforementioned models have been suitably adapted for the purpose of identifying individuals in biometrics, security frameworks, and facial recognition technologies [85].

Moreover, YOLO algorithms have a crucial function that extends beyond the detection of objects in typical scenarios. Wildlife conservation relies heavily on their important role. By leveraging the detection and monitoring functionalities of these models, scholars have successfully identified and monitored endangered species with precision, thereby playing a crucial role in the conservation of biodiversity and the facilitation of efficient ecological management [95]. Furthermore, the utilisation of these models extends to the field of robotics, where they make a substantial contribution to the improvement of robotic vision systems, enabling more effective interaction with their environment [96], [97]. Furthermore, there has been widespread implementation of YOLO-powered technology in aerial drone surveillance operations to accurately and efficiently detect things from above [98], [99].

2.4.2 Comparisons of YOLOv7, YOLOv8 and YOLOv9

Every iteration of the YOLO algorithm incorporates distinct characteristics and enhancements. The objective of this study is to conduct a comparative analysis of three distinct iterations within the YOLO family, namely YOLOv7, YOLOv8, and the recently introduced YOLOv9.

There are five scaled variations of YOLOvX with " X " is the version number, namely YOLOvXn (nano), YOLOvXs (small), YOLOvXm (medium), YOLOvXl (large), and YOLOvXx (extra-large). The convolution modules in these versions exhibit variations in terms of their width and depth, which are tailored to suit certain applications and hardware

prerequisites. For instance, YOLOv5n and YOLOv4s have been specifically developed to cater to low-resource devices, whilst YOLOv5x has been specifically intended to achieve high performance, albeit at the expense of slower speed.

2.4.2.1 YOLOv7

The creation of YOLOv7 [103] by Chien-Yao Wang, Hong-Yuan Mark Liao, and Alexey Bochkovskiy is a noteworthy development in the field of real-time object detection. A "trainable bag-of-freebies," or collection of methods that improve model performance without raising the cost of inference, is introduced in this edition. With these improvements, YOLOv7 is able to achieve previously unheard-of levels of accuracy and speed, creating a new standard for real-time object detectors. As a result of both architectural and training process optimization, YOLOv7 outperforms its predecessors in terms of precision and recall rates, giving it a cutting-edge option for a range of applications needing quick and precise object detection. This innovation highlights how YOLOv7 can revolutionize real-time object detection in domains including industrial automation, autonomous driving, and surveillance. The comprehensive architectural design of YOLOv7 is presented in Appendix B.

YOLOv7 Architecture

YOLOv7's architecture incorporates a number of cutting-edge features and optimizations to improve real-time object detection's accuracy and performance. The YOLO (You Only Look Once) framework, which reframes object identification as a single regression problem and predicts bounding boxes and class probabilities directly from the full image in a single assessment, is the foundation of YOLOv7. This design is appropriate for applications needing quick and accurate detections because it strikes a compromise between speed and accuracy.

The backbone network of YOLOv7 is in charge of retrieving crucial characteristics from the input image. This network uses sophisticated convolutional layers that are intended to effectively collect features at multiple scales. YOLOv7 divides feature maps into two halves using CSP (Cross Stage Partial) connections in its backbone, which enhance gradient flow and lessen computing load. The other component is concatenated directly, while the first part is processed through a deep layer of convolutional processes. This architecture maintains a high processing speed while simultaneously increasing learning efficiency.

Numerous feature pyramid networks (FPN) and path aggregation networks (PAN) make up YOLOv7's neck. These networks are essential for combining and honing features at various scales. The integration of multi-scale features is essential for identifying objects with different sizes and forms present in the same image. Anchor boxes and regression layers are incorporated into the head of YOLOv7, which predicts bounding boxes and class scores. These elements are tailored to precisely forecast object boundary coordinates and the classes that correspond with them. YOLOv7 is able to detect both small and large objects with greater precision by utilizing multi-scale predictions.

YOLOv7's "trainable bag-of-freebies," a set of methods that improve model performance without raising the inference cost, is one of its best features. These methods include data augmentation procedures that enhance the training dataset and increase the resilience of the model, such as mosaic and mix-up. Moreover, YOLOv7 uses classification loss functions and sophisticated label smoothing to enhance generalization and minimize overfitting during training. The reason these freebies are "trainable" is because they are incorporated into the model's learning process, which improves efficiency without adding more complexity to the runtime.

YOLOv7's architecture is further optimized through meticulous training procedures and hyperparameter tuning. Techniques like adaptive learning rate schedules, gradient clipping, and regularization are employed to stabilize and expedite the training process. These optimizations ensure that YOLOv7 converges faster and achieves higher accuracy compared to its predecessors. Moreover, the architecture is designed to be scalable, allowing for adjustments in model size to cater to different computational constraints and application requirements.

2.4.2.2 YOLOv8

Ultralytics unveiled YOLOv8 [117] in January 2023, expanding upon their prior efforts with YOLOv5. The latest iteration of the model is available in five distinct scaled forms, namely YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large). In addition to object detection, YOLOV8 has a wide range of skills in the field of vision, including segmentation, posture estimation, tracking, and classification.

YOLOv8 Architecture

The comprehensive architectural design of YOLOv8 is presented in Appendix C. YOLOv8 utilises a comparable foundational framework to YOLOv5, but modifies the CSPLayer, which is now known as the C2f module. In order to improve the accuracy of detection, the C2f module, which stands for cross-stage partial bottleneck with two convolutions, combines high-level characteristics with contextual information.

In contrast to its previous iterations, YOLOv8 utilises an anchor-free architecture including a detached head, enabling autonomous handling of objectness, classification, and regression tasks. The implementation of this streamlined strategy improves the accuracy of the model by allowing each branch to concentrate on its designated duty. YOLOv8 employs the sigmoid function in the output layer to compute the objectness score, which indicates the probability that an object is present in a bounding box. Additionally, the softmax function is used to calculate class probabilities, which represent the likelihood of objects belonging to their respective classes. Furthermore, YOLOv8 incorporates enhanced loss functions, namely CIoU and DFL, which are utilised for bounding-box and classification losses, respectively. The aforementioned improvements have demonstrated enhanced efficacy, namely in the identification of diminutive entities during the process of object detection [104, 118].

The YOLOv8 framework presents a semantic segmentation model called YOLOv8-Seg, which incorporates a CSPDarknet53 backbone and a C2f module as opposed to the conventional YOLO neck approach. Following this module, there are two segmentation heads that are responsible for predicting semantic segmentation masks for the input images. The proposed model incorporates detection heads that bear resemblance to those found in YOLOv8, with five detection modules and a prediction layer. The YOLOv8-Seg model has demonstrated exceptional performance in object detection and semantic segmentation benchmarks, showcasing its exceptional speed and efficiency.

Additionally, the software can be executed using the command line interface (CLI) or implemented as a PIP package, providing various integrations for the purposes of labelling, training, and deployment. When evaluating performance on the MS COCO dataset test-dev 2017, YOLOv8x achieved an AP (Area Precision) of 53.9%. This was better than YOLOv5's AP of 50.7% at the same input size. Additionally, YOLOv8x achieved speeds of up to 280 FPS on an NVIDIA A100 with TensorRT integration.

2.4.2.3 YOLOv9

YOLOv9 [109] was introduced in 2024 which present a novel approach to the improvement of deep learning model YOLO. In order to solve the problem of information loss during layer-by-layer feature extraction and spatial transformation in deep networks, YOLOv9 presents the idea of Programmable Gradient Information (PGI). YOLOv9 guarantees that all input data is retained for the intended purpose by integrating PGI, which improves gradient updates and model performance. Additionally, the researchers created the Generalized Efficient Layer Aggregation Network (GELAN), a novel lightweight network architecture that maximizes parameter use using gradient path planning. When tested against state-of-the-art techniques on the MS COCO dataset, YOLOv9 produced better results and achieved improved parameter efficiency. With this breakthrough, YOLOv9 represents a substantial advancement in object detection as models built from scratch can outperform pre-trained models on big datasets. The comprehensive architectural design of YOLOv9 is presented in Appendix D.

YOLOv9 Architecture

YOLOv9's architecture is intended to lessen the problems caused by inefficiencies and information loss in deep learning models. The Programmable Gradient Information (PGI) mechanism, which makes sure that important data is not lost as it moves through different network layers, lies at the heart of this system. With this method, the model can calculate gradients and update weights accurately, retaining and using all available input data to produce more accurate and consistent predictions.

YOLOv9 presents the Generalized Efficient Layer Aggregation Network (GELAN), a network architecture that is lightweight and efficient, with an emphasis on optimizing parameter consumption. Unlike other lightweight models, which often use depth-wise convolutions, GELAN uses ordinary convolution operators. With this design decision, GELAN is able to efficiently aggregate and analyze characteristics at many sizes, resulting in higher performance. Because GELAN's architecture is designed to optimize gradient information flow, it can achieve excellent accuracy and efficiency even with lightweight models.

PGI, a fundamental innovation of YOLOv9, improves learning by giving the network a complete gradient path. In contrast to conventional techniques that experience information bottlenecks, PGI allows the model to retain and make use of all of the input data during the training phase. This leads to enhanced model performance and more efficient gradient updates.

PGI is a useful technique for optimizing deep learning models for a variety of applications since it is adaptable and can be used with models of varying sizes, from small to large.

Performance and efficiency are significantly increased in YOLOv9 when PGI and GELAN are combined. The architecture was evaluated using the MS COCO dataset, outperforming state-of-the-art techniques in terms of accuracy and parameter consumption. The usefulness of the YOLOv9 architecture is highlighted by the ability to train models from scratch and obtain improved outcomes without depending on big pre-trained datasets. Because of this, YOLOv9 is an effective real-time object identification tool that provides improved precision, effectiveness, and versatility for a range of uses.

2.4.3 Performance Evaluation

The Average Precision, also referred to as the Mean Average Precision (mAP), is a commonly used metric for evaluating the performance of object identification algorithms. The calculation determines the mean precision across all categories, providing a unified statistic for comparing different models. The PASCAL VOC 2007 and VOC 2012 datasets were used for training and evaluation in YOLOv1 and YOLOv2 [100]. Starting with the release of YOLOv3 and subsequent iterations, Microsoft's COCO dataset emerged as the prevailing option [101]. It is of significance to note that the computation of AP exhibits variation across these data sources.

The problem of object detection models involves the identification and localization of many object categories within an image. The Mean Average Precision (mAP) metric addresses this difficulty by determining the average precision for each category individually and subsequently calculating the mean of these averages across all categories. This approach guarantees a comprehensive assessment of the model's performance on specific categories, providing a more intricate perspective on its overall efficacy.

Precision and recall are two key metrics used to assess the quality of a model in making positive predictions. Precision measures the ratio of correctly identified actual positive predictions to the total number of positive predictions made by the model. A common occurrence is the presence of a trade-off between precision and recall. For example, when object detection is enhanced to achieve higher recall, it may result in an elevation of false positives, hence reducing precision. In order to achieve this equilibrium, the Average Precision metric use the precision-recall curve, which delineates the measure of precision and recall at

different levels of confidence. The aforementioned metric provides a full assessment of precision and recall by analysing the area beneath the curve.

The AP measure employs precision-recall metrics, effectively handles various object categories, and creates a positive prediction using Intersection over Union (IoU) [102]. Intersection over Union: The objective of object detection is to accurately identify items in photographs by estimating the boundaries of the objects. The accuracy of the projected bounding boxes is successfully evaluated using the Intersection over Union measure in the AP metric. The area of overlap (IoU) is determined by dividing the total area covered by the predicted and actual ground truth bounding box by the area of overlap, as shown in Figure 2.4. This metric offers a measure of the degree of agreement between the predicted and actual bounding boxes. Furthermore, the utilisation of different IoU thresholds enables the measurement of performance on various scales of localization precision, which is significantly helped by the implementation of COCO benchmarks.

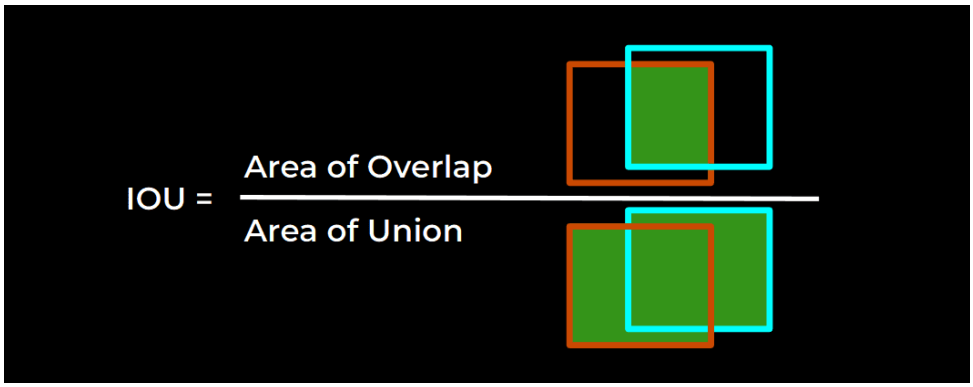


Figure 2.4 IoU formula

2.4.4 Performance Evaluation of YOLO with Benchmark Datasets

The YOLO measure is assessed in a study that examines benchmark datasets, encompassing iterations ranging from the initial YOLO model to the most recent YOLO-NAS [102]. The YOLO and YOLOv2 models were trained using the VOC2007 dataset, whereas the succeeding models were trained using the COCO2017 dataset. Furthermore, it is worth mentioning that the NAS-YOLO model functions with a precision of 16 bits.

Table 2.1 Summary of YOLO performance metrics

Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	DarkNet	DarkNet24	63.4

YOLOv2	2016	Yes	DarkNet	DarkNet24	78.6
YOLOv3	2018	Yes	DarkNet	DarkNet53	33.0
YOLOv4	2020	Yes	DarkNet	CSP DarkNet53	43.5
YOLOv5	2020	Yes	PyTorch	YOLOv5CSPDarkNet	55.8
PP-YOLO	2020	Yes	PaddlePaddle	ResNet50-vd	45.2
Scaled-YOLOv4	2021	Yes	PyTorch	CSP-Darknet	56.0
PP-YOLOv2	2021	Yes	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	Yes	PyTorch	CSP-Darknet	55.4
YOLOX	2021	No	PyTorch	YOLOXCSPDarknet	51.2
PP-YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	PyTorch	EfficientRep	52.5
YOLOv7	2022	No	PyTorch	YOLOv7Backbone	56.8
DAMO-YOLO	2022	No	PyTorch	MAE-NAS	50.0
YOLOv8	2023	No	PyTorch	YOLOv8CSPDarknet	53.9
YOLO-NAS	2023	No	PyTorch	NAS	52.2

Anchors

The initial YOLO model exhibited a relatively uncomplicated design and did not incorporate anchors, whereas the other versions during that era heavily depended on two-stage detectors incorporating anchors. The use of anchors in YOLOv2 led to improved accuracy in the prediction of bounding boxes. Over a span of around five years, this methodology persisted until YOLOX introduced an anchorless technique that yielded exceptional outcomes. Later iterations of YOLO have subsequently shifted their focus away from utilising anchors.

Framework

The Darknet infrastructure was initially employed in the development of YOLO, and this approach has been maintained in future iterations. Nevertheless, upon Ultralytics' adaptation of YOLOv3 to PyTorch, the subsequent iterations were also created using PyTorch, leading to notable enhancements. PaddlePaddle, an open-source framework developed by Baidu, is another deep learning language that is used.

Backbone

Significant advancements have been made in the structural underpinnings of YOLO models. The Darknet architecture was initially employed, incorporating basic convolutional

and max pooling layers. Subsequent updates of the architecture incorporated cross stage partial connections in YOLOv4, reparameterization in YOLOv6 and YOLOv7, and neural architecture search in DAMO-YOLO and YOLO Neural Architecture Search (NAS).

Average Precision Performance

The YOLO models have demonstrated a steady enhancement in their overall performance over time. However, it is crucial to acknowledge that these models frequently prioritise the attainment of a trade-off between speed and accuracy, rather than exclusively emphasising precision. The trade-off mentioned is a crucial aspect of the framework's architecture, since it facilitates the ability to recognise objects in real-time across a wide range of applications.

2.5 Summary

Chapter 2 literature review explores key factors essential for comprehending and applying defect detection systems, with a specific emphasis on the shift from conventional image processing methods to deep learning approaches. The investigation commences by providing a comprehensive examination of defect detection methodologies, emphasising the constraints associated with conventional image processing approaches in effectively detecting anomalies. The subsequent analysis focuses on the rise of deep learning as a potent instrument in defect detection, comprising deliberations on the necessary hardware prerequisites and the wide array of libraries and tools accessible for the application of deep learning. The review primarily focuses on You Only Look Once (YOLO), a widely used object detection method. It explores its applicability in many fields and compares several versions of YOLO, such as YOLOv7, YOLOv8, and YOLOv9. Furthermore, the paper critically examines the performance of YOLO-based systems, placing significant emphasis on the necessity of conducting thorough evaluations utilising benchmark datasets in order to guarantee the dependability and efficacy of defect detection models. In summary, the literature analysis establishes a thorough basis for the subsequent creation and assessment of defect detection systems, providing valuable information on both conventional and cutting-edge methods in the field.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter talks about the process flow of our targeted method to achieve the objectives of this research which includes the data acquisition, types of data, image process flow, deep learning architecture and hardware setup. It is crucial to have a plan prepared beforehand so that we will have all the necessary preparation ready and minimize any unnecessary errors or mistakes.

The literature review discussed previously is undertaken with the purpose of collecting pre-existing knowledge and ideas related to this project, hence facilitating the development of the inspection system. The subsequent sections will be explaining about all the steps and methods that will be compare and select for this project based on software, electrical & electronics and mechanical. Furthermore, all the necessary equipment that required in this project will be sum up in bill of materials. Finally, we will be explaining the whole working process of this inspection system with a diagram and a flowchart.

3.2 Concept Selection

This section is going to show how we design our system based on four concept designs, each of the design has its own unique methods and tools which will be going to discuss in the next few sections to decide which concept we are going to select as our main design of this project.

The four concept designs will be shown in the next section Concept Generation where we can view our rough design idea tabulated in a table. Then we will be comparing the design concepts with each other to see the difference among them. After that, we will do concept screening to list out the requirements of this project and check whether our design fulfilled the requirements. Other than that, we will be ranking the importance of the criteria to see how important each of our requirements is. Finally, we will do concept scoring to decide which concept we will choose to design and use it in our project.

3.2.1 Concept Generation

Appendix: D

3.2.2 Concept Comparison

In this section, we will separate the concept into three categories which are based on software, electrical & electronic and mechanical. All the comparisons are tabulated in tables for a better visual of what functions and benefits from each concept provides which we can have a better understanding and do selections later on.

3.2.2.1 Software Concept Comparison

Table 3.1 Concept Comparison: Data Labelling

Concept	Tools	Advantages
1	LabelMe	Specialised with segmentation and polygon annotation tools Straightforward tools but lack advanced features Limited export options compared to CVAT and Roboflow
2	Labellmg	Simple and light tool specialised for marking up bounding boxes Straightforward tools but lack advanced features Limited export options compared to CVAT and Roboflow
3	Roboflow	Easy-to-use interface and supports different types of annotations User-friendly interfaces Support flexible export options
4	CVAT	Wide range of marking tools and allow add notes to videos Straightforward and User-friendly interfaces Supports various exporting formats (YOLO, COCO, etc) Active community support

Table 3.2 Concept Comparison: Deep Learning Framework

Specifications	Concept 1 Concept 2	Concept 3 Concept 4
Framework	Tensorflow	PyTorch
Developer	Google Brain	Facebook's AI Research lab (FAIR)
Strengths	High-level and low-level APIs Good production deployment due to support for Tensorflow	Dynamic computation graph Support mobile deployment

	Strong ecosystem for production use	Extensive support for computer vision and NLP (Natural Language Processing)
	Large Community and well-documented	Popular research community
	Good for large-scale distributed computing and deployment	Suitable for quick prototyping

Table 3.3 Concept Comparison: Deep Learning Model

Specifications	**Concept 1**	**Concept 3**	**Concept 2** **Concept 4**
Model Name	Faster R-CNN	DSSD	YOLO
Architecture Type	Two-stage object detection model	One-stage object detection model	One-stage object detection model
Real-time Performances	Poor	Good	Good
Mean Average Precision on VOC2007 Dataset	73.20%	81.50%	-
Mean Average Precision on COCO Dataset	37.40%	33.20%	57.2%

Table 3.4 Concept Comparison: Model Training Platform

Specs	**Concept 1**	**Concept 2**	**Concept 3**	**Concept 4**
Feature	Jupyter	Google Colab	VSCode	Pycharm
Platform	Web-based	Web-based	Desktop	Desktop
Usage	Data science, academic research	Data science, machine learning	General-purpose coding, data science	General-purpose coding, data science

Language	Primarily Python (via kernels)	Primarily Python (via kernels)	Multiple (Python, JavaScript, C++, etc.)	Primarily Python, but others supported
Code Execution	Cell-based execution	Cell-based execution	Script/module execution	Script/module execution
GPU Enabled	CUDA Enabled	Limited GPU Access (Paid)	CUDA Enabled	CUDA Enabled

Table 3.5 Concept Comparison: Data Transmission

Specifications	**Concept 1**	**Concept 3**	**Concept 2** **Concept 4**
Transmission Name	LoRa	Bluetooth	WiFi
Transmit Distance	7-10 km	up to 100 m	30-100 m
Power Consumption	Low power consumption	Power efficient than WiFi	Consume more power than LoRa and Bluetooth
Data Transmission Speed	Low data rates	Medium data rates	High data rates

Table 3.6 Concept Comparison: Data Storage

Specifications	**Concept 1**	**Concept 3**	**Concept 2** **Concept 4**
Database Name	MariaDB	NoSQL Real-time Database	MySQL
Database Type	Relational DBMS	-	Relational DBMS
Query Language	SQL	Firebase Realtime Database API query language	SQL

Deployment	Self-hosted, Cloud	Cloud-based (Google Cloud)	Self-hosted, Cloud
Community & Document	Strong community & well-documented	Firebase community and Google support	Strong community & well-documented
Licensing Information	Open source (GNU GPL)	Proprietary (Google Cloud pricing)	Dual-licensing (Community/Enterprise)

Table 3.7 Concept Comparison: Data Visualization

Specs	Concept 1	Concept 2	Concept 3	Concept 4
Software	Gradio	Dash	Thingsboard	Streamlit
Strength	Easy to use Multiple input types Integration with ML libraries Rapid prototyping	Data visualization Real-time updates Community and ecosystem	Scalability Complexity	Simplicity Rapid prototyping Extensive library integration Automatic UI generation
Limits	Limited scalability Limited customization Limited complexity	Performance issues Limited customization	-	Limited customization Limited for complex web apps

3.2.2.2 Electrical & Electronic Concept Comparison

Table 3.8 Concept Comparison: Camera

Specifications	Concept 1 Concept 2	Concept 3	Concept 4

Model Name	Atlas IMX545	OAK-1 PoE	Hikvision Web Camera DS-U02
Camera Resolution	12.3 MP Camera	12.3 MP Camera	2 MP CMOS
Image Resolution FPS	4096x3000 px 22.6 FPS	4096x3000 px 22.6 FPS	1920x1080 px 30 FPS
Image Sensor Type	Sony IMX304 sensor	Sony IMX304 sensor	AGC for self-adaptive brightness
Power Over Ethernet	PoE connection	PoE connection	Type A interface USB 2.0 protocol
Weight	90g	90g	89g

Table 3.9 Concept Comparison: Single-board Computer for Camera

Specs	**Concept 1**	**Concept 2**	**Concept 3**	**Concept 4**
Board	Raspberry Pi Zero	Raspberry Pi 2 W	Raspberry Pi Zero W	Raspberry Pi Zero WH
Processor	1GHz, Single-core ARM1176JZF-S	1GHz, Quad-core Cortex-A53	1GHz, Single-core ARM1176JZF-S	1GHz, Single-core ARM1176JZF-S
RAM	512MB	512MB	512MB	512MB
Wireless	None	802.11 b/g/n Wi-Fi, Bluetooth 4.2	802.11 b/g/n Wi-Fi, Bluetooth 4.1	802.11 b/g/n Wi-Fi, Bluetooth 4.1
Bluetooth	None	Bluetooth 4.2	Bluetooth 4.1	Bluetooth 4.1
USB Ports	1 micro USB (OTG)	1 micro USB (OTG)	1 micro USB (OTG)	1 micro USB (OTG)
GPIO Pins	40 (unpopulated)	40 (unpopulated)	40 (unpopulated)	40 (pre-soldered)
Camera Interface	CSI (requires additional adapter)	CSI-2	CSI (requires additional adapter)	CSI (requires additional adapter)

Power Consumption	Low	Low	Low	Low

Table 3.10 Concept Comparison: Single-board Computer for Object Detection

Specifications	**Concept 1**	**Concept 2**	**Concept 3**	**Concept 4**
Model Name	Coral Dev Board	Jetson Orin Nano	Raspberry Pi 4 Model B	Jetson Orin Nano
RAM	4GB LPDDR4	8GB LPDDR5	4GB LPDDR4	8GB LPDDR5
CPU	NXP i.MX 8M SoC (quad Cortex-A53, Cortex-M4F)	6-core Arm Cortex-A78AE v8.2	Broadcom BCM2711, 64-bit ARM Cortex-A72 (ARMv8), Quad core	6-core Arm Cortex-A78AE v8.2
GPU	Integrated GC7000 Lite Graphics	NVIDIA AmpereTM architecture	28nm Processor SoC	NVIDIA AmpereTM architecture
Specialisation	AI applications	AI applications	Lesser AI capabilities	AI applications
Operating System	Ubuntu Linux 4 Tegra	Ubuntu Linux 4 Tegra	Raspberry Pi OS, Debian Buster, Ubuntu, LibreELEC, RetroPie, etc	Ubuntu Linux 4 Tegra
Price	Cheap	Expansive	Cheap	Expensive

3.2.2.3 Mechanical Concept Comparison

Table 3.11 Concept Comparison: Camera Mount Design

Specification	**Concept 1**	**Concept 2**	**Concept 3**	**Concept 4**
Clamps	G clamp	Clip	G clamp	Clip

Movement	Rotating camera orientation	Adjustable camera angle Adjustable camera height	Adjustable camera height	Adjustable camera angle Rotating camera orientation

Table 3.12 Concept Comparison: Mounting Manufacture

Specs	**Concept 1** **Concept 3**	**Concept 2** **Concept 4**
Methods	3D Print	Purchase
Strength	Design modification freely Cost saving	Time saving Design and material well-made
Limits	3D print time consuming Redesign and reprint if not suitable Only one type of material for 3D print	Design fixed Cost consuming

3.2.3 Concept Screening

Table 3.13 Concept Screening Table

Selection criteria	**Concept 1**	**Concept 2**	**Concept 3**	**Concept 4**	**Ref**
Processing power	0	+	-	+	0
Real-time data	-	0	0	+	0
Durability	-	+	0	+	0
Reliability	-	+	0	+	0
Cost	-	-	+	+	0
Accessibility for maintenance and repair	+	+	0	+	0
Camera angle adjustability	-	0	-	+	0

Detection accuracy	-	0	+	+	
Environmental Reistance	-	+	-	+	
Ease of integration	+	+	+	+	
Update and upgrade capability	0	0	-	+	
Compliance with industry standards	-	+	0	+	0
Net	-6	7	-1	12	
Rank	4	2	3	1	
Continue?	No	Yes	No	Yes	

3.2.4 Criteria Importance Ranking

Table 3.14 below is the annotations for criteria selection

1. A = Processing power
2. B = Real-time data
3. C = Durability
4. D = Reliability
5. E = Cost
6. F = Accessibility for maintenance and repair
7. G = Camera angle adjustability
8. H = Detection accuracy
9. I = Environmental Resistance
10. J = Ease of integration
11. K = Update and upgrade capability
12. L = Compliance with industry standards

Table 3.14 Criteria Importance Ranking Table

Criteria Selection	A	B	C	D	E	F	G	H	I	J	K	L	Total "+"	Weight Factor
A		+	+	-	+	-	-	0	0	+	+	0	5	0.081
B	+		0	+	0	+	0	+	0	+	0	+	6	0.097
C	+	0		+	+	+	+	0	+	0	-	+	7	0.11
D	-	+	+		+	0	0	+	0	0	0	+	5	0.081
E	+	0	+	+		+	0	-	-	0	0	-	4	0.065
F	-	+	+	0	+		-	-	0	+	+	+	6	0.097
G	-	0	+	0	0	-		+	0	0	0	+	3	0.048
H	0	+	0	+	-	-	+		0	0	0	+	4	0.065
I	+	0	+	+	0	+	-	-		+	0	+	6	0.097
J	+	+	-	+	-	0	-	+	-		+	+	6	0.097
K	-	+	-	+	0	0	0	+	0	+		+	5	0.081
L	0	+	+	+	-	-	0	+	+	0	0		5	0.081
Total													62	1.00

3.2.5 Concept Scoring

After we have done the concept scoring as shown in table 3.15, we found out that concept 4 has the highest ranking when compared to concept 2. So, we selected concept 4 as our main design. All the methods, design flows and process are all about developing our system based on concept 4 after this.

Table 3.15 Concept Scoring Table

Criteria Selection	Weighted score, w	Concept 2		Concept 4	
		Rating, R	R*w	Rating, R	R*w
Processing power	0.081	4	0.32	5	0.41
Real-time data	0.097	2	0.19	4	0.39
Durability	0.11	3	0.33	3	0.33
Reliability	0.081	2	0.16	5	0.41
Cost	0.065	4	0.26	1	0.065
Accessibility for maintenance and repair	0.097	3	0.29	3	0.29

Camera angle adjustability	0.048	3	0.14	3	0.14
Detection accuracy	0.065	2	0.13	4	0.26
Environmental Resistance	0.097	5	0.49	5	0.49
Ease of integration	0.097	2	0.20	4	0.39
Update and upgrade capability	0.081	3	0.24	3	0.24
Compliance with industry standards	0.081	2	0.16	5	0.41
Total	1.00	35	2.90	45	3.83
Rank		2		1	

3.3 Software

This section talks about the process flow of the software part in this project, mainly we separate it into a few categories which are chapter 1, chapter 2, data pre-processing, model training & selection, model testing, implementation process and documentation.

The next step is to collect data, which means getting relevant information from a number of different sources. After this, the data is pre-processed so that it can be used for training the model. Data annotation will be proceed to box the defect part from the image. Then, data augmentation methods are used to make the information more diverse and help the model work better in real life situations.

Picking the right deep learning model design is the most important part of the process. Once the right model has been chosen, it is trained using a dataset that has already been prepared and then its success is evaluated. Once the model performs well enough, it moves on to the implementation phase, where we will create our graphical user interface, databases, setup of camera and web application to implement with our trained model.

After that, the model goes through testing and evaluation steps to make sure it is reliable and no errors or bug. If optimization methods are thought to be necessary to make the model work better, they may be used. Finally, the whole process and its results are written down, documentations will be made for future users and engineers to use and address any issues. Figure 3.1 shows the whole process flow of software part in this project.

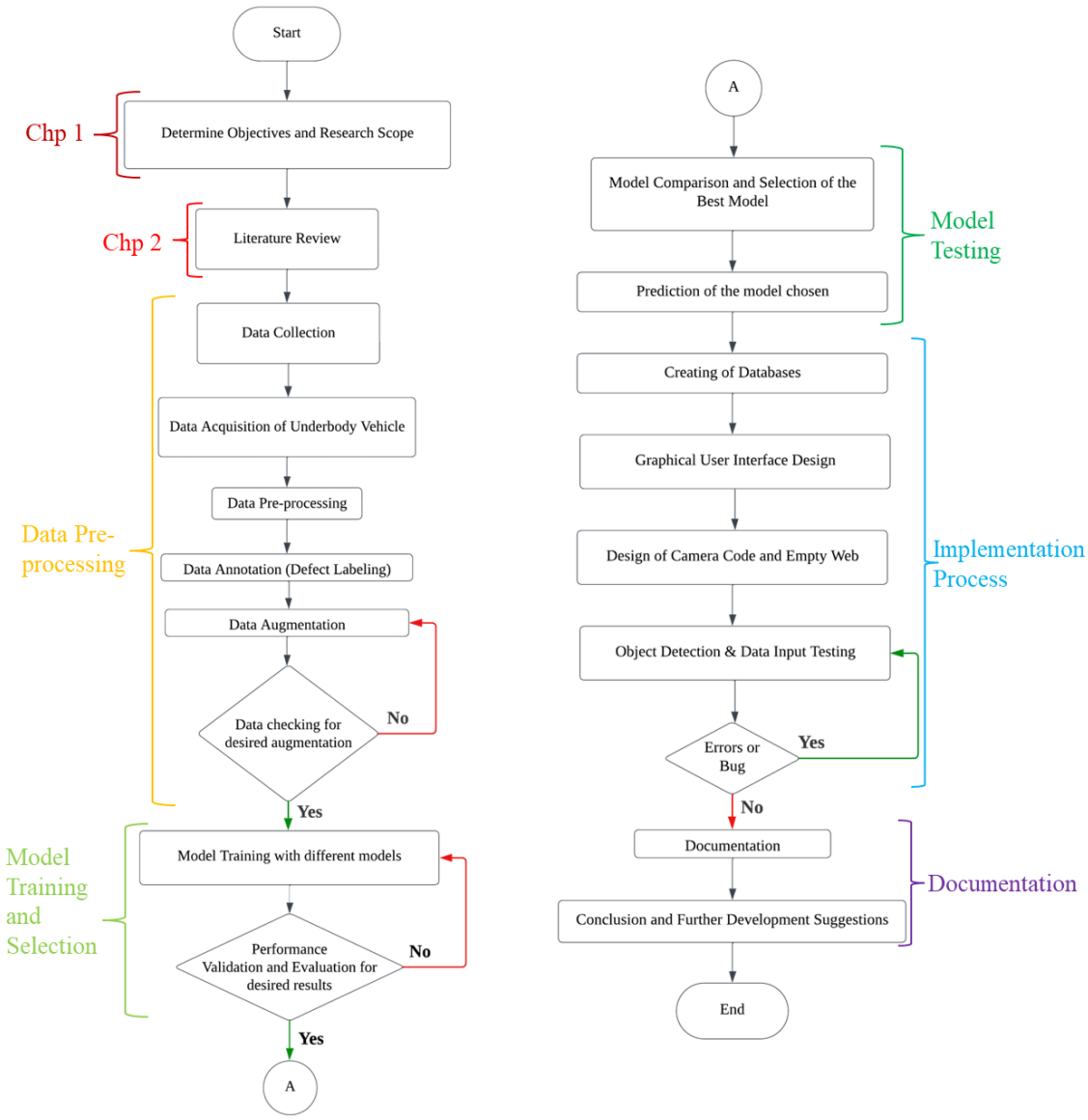


Figure 3.1: Software flowchart

3.3.1 Data Collection

This sub-section talks about what kinds of defect we should be focusing on from which type of Volkswagen car model. Basically we should focus more on the car model which has the highest percentage of defect type. In order to do that, we will be recording the amount of car defects and amount of defect types for a few months in an inspection logbook for us to do an analysis.

We will use the data collected in the inspection logbook to do some analysis towards the model car defects and defects types to find out which car and which defect type we should be focusing more on the next few sections.

3.3.1.1 Inspection Logbook

The car types and defect types are recorded in the inspection logbook as shown in figure 3.2. The contents from the table of the logbook are as follows:

1. Number of items: The total count of items inspected during the process.
2. Date: The date on which the inspection was conducted.
3. Car Model: Identification of the inspected vehicle model.
4. Vehicle Identification Number (VIN): Keeps track of the distinctive number connected to every car.
5. Sequence Number: Stands for the distinct sequence number that is given to every inspection entry.
6. Colour: The colour of the vehicle being inspected.
7. Time In: Records the time at which the vehicle entered the inspection area.
8. Time Out: shows the time at which the vehicle's examination was completed.
9. Condition Assessment: Describes the underbody's general state as determined by the inspection.
10. Findings: Provide a thorough explanation of any abnormalities or flaws found during the inspection.

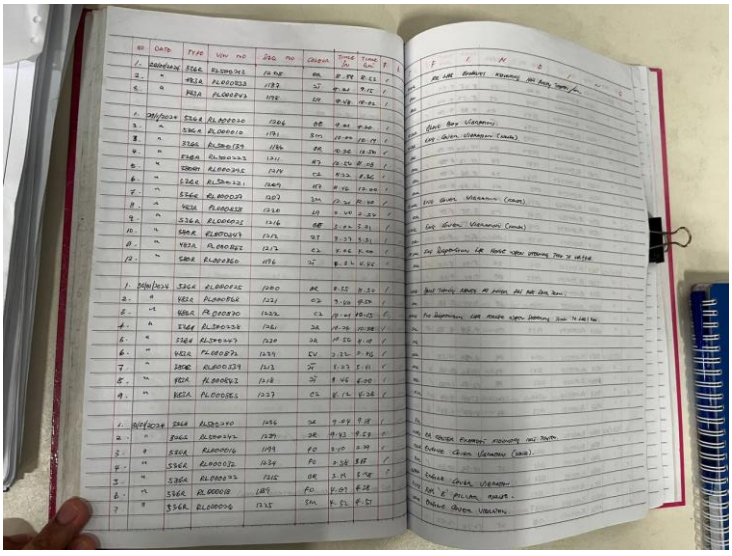


Figure 3.2 Handwritten Inspection Logbook

3.3.1.2 Vehicle Type Defect Analysis

Based on the results from the inspection logbook, the defect percentage of each vehicle model has been calculated over a specific period of time (months). The number of vehicle

models manufactured and the number of defects from each vehicle type both from May to August are being arranged in table 3.16 and table 3.17 below.

Table 3.16 Quantity of Vehicle Models Manufactured from May to August 2023

Car Model	Code	Months				Sum
		May	June	July	August	
Tiguan Allspace Life	VW326S	14	5	4	1	24
Tiguan Allspace Highline	VW326H	90	113	131	156	490
Tiguan Allspace R-Line	VW326R	111	38	104	61	314
Arteon R-Line 4MOTION	VW483R	24	79	47	43	193
Golf GTI	VW380GTI	50	40	8	66	164
Sum		289	275	294	327	1185

Table 3.17 Number of Defects Under the Vehicle from May to August 2023

Car Model	Code	Months				Sum
		May	June	July	August	
Tiguan Allspace Life	VW326S	0	0	0	0	0
Tiguan Allspace Highline	VW326H	3	10	6	1	19
Tiguan Allspace R-Line	VW326R	4	5	6	1	17
Arteon R-Line 4MOTION	VW483R	2	0	0	0	2
Golf GTI	VW380GTI	0	1	0	0	1
Sum		9	16	12	2	39

The data from table 3.17 will be used to plot a bar chart graph to have a better visual representation for us to see how much the percentage of defects in each Volkswagen vehicle models. We can use the following formula to calculate the percentage of monthly underbody vehicle defects from each vehicle models.

$$\text{Monthly Defect Percentage of a Car Model, \%} = \frac{Defects\ of\ Each\ Model}{Quantity\ of\ Each\ Model}\ x\ 100\%$$

Based on the graph plotted in figure 3.3, the graph shows us that Tiguan Allspace Life (VW326S) has no defects in this four months period. After that, the vehicle models that have only one month consists of percentage of defects are Arteon R-Line 4MOTION (VW483R)

and Golf GTI (VW380GTI) with 8.3% of defects in May and 2.5% of defects in June respectively but apparently Arteon R-Line 4MOTION (VW483R) has a higher percentage of defects. Furthermore, we can also see that Tiguan Allspace Highline (VW326H) has defects in four months with each of the percentage is lower than Tiguan Allspace R-Line (VW326R). Finally, it also shows us that Tiguan Allspace R-Line (VW326R) achieved the highest percentage of defects in these four months when comparing with other vehicle models.

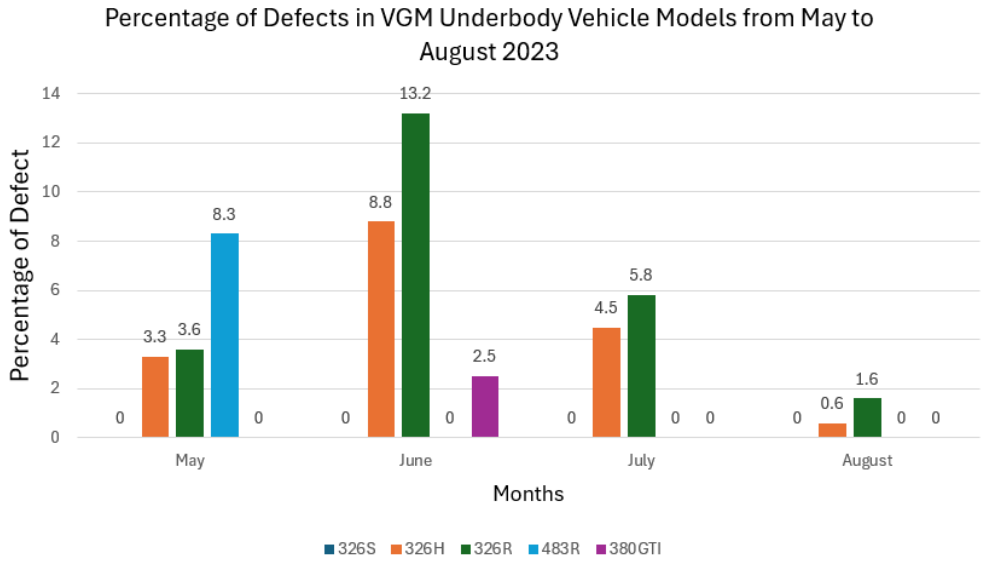


Figure 3.3 Pie Chart for Percentage of Defects under VGM Vehicle Models from May to August 2023

To summarize which vehicle model has the highest percentage of defects in these four months, we can use another formula as below to calculate the total percentage of defects throughout these four months. The results calculated are being tabulated in table 3.18. Based on table 3.18, it showed us that Arteon R-Line 4MOTION (VW483R) has the highest defect percentage when compared to other vehicle models.

$$\text{Total Monthly Defect Percentage of a Car Model, \%} = \frac{Total\ Defects\ of\ Each\ Model}{Total\ Quantity\ of\ Each\ Model}\ x\ 100\%$$

Table 3.18 Total Defect Percentage of the Car Models

Car Models	Code	Defect Percentage (%)
Tiguan Allspace Life	VW326S	-
Tiguan Allspace Highline	VW326H	4.8
Tiguan Allspace R-Line	VW326R	5.1
Arteon R-Line 4MOTION	VW483R	1

Golf GTI	VW380GTI	0.61

3.3.1.3 Defect Type Analysis

After collecting the amount of defect vehicle models, we will focus on the type of defects in each vehicle model in this section. The defect types are recorded in the inspection logbook also and the results are tabulated for us to visualize which defect types occurs most frequently in few table below which are table 3.19, table 3.20, table 3.21 and table 3.22.

Based on the four tables below, it showed us that the highest amount of defect types that occurred in each month is the loose bolt on exhaust mounting. Other than that, the other two defect types that occurred a lot are the missing screw under the vehicle and missing plug grommet.

Table 3.19 Defect Types in May

May						
Defect Types	**326S**	**326H**	**326R**	**483R**	**380GTI**	**SUM**
Heat shield deform			1			1
Excess grease on drive shaft			1			1
Missing screw on front bumper			1			1
Loose bolt on exhaust mounting		2	1			3
Missing screw on wheel housing				1		1
Missing plug grommet				1		1
Missing plug on rear bumper		1				1
SUM	0	3	4	2	0	9

Table 3.20 Defect Types in June

June						
Defect Types	**326S**	**326H**	**326R**	**483R**	**380GTI**	**SUM**
Excess grease on drive shaft			1			1
Loose bolt on exhaust mounting		10	3		1	14
Loose bolt on fuel canister			1			1
SUM	0	10	5	0	1	16

Table 3.21 Defect Types in July

July						
Defect Types	**326S**	**326H**	**326R**	**483R**	**380GTI**	**SUM**
Excess grease on drive shaft			1			1
Missing screw on underbody cover			2			2
Loose bolt on exhaust mounting		4	2			6
Missing plug grommet		1	1			2
Missing bolt on heat shield cover		1				1
SUM	0	6	6	0	0	12

Table 3.22 Defect Types in August

August						
Defect Types	**326S**	**326H**	**326R**	**483R**	**380GTI**	**SUM**
Excess grease on drive shaft			1			1
Loose bolt on underbody cover		1				1
SUM	0	1	1	0	0	2

To summarize everything from these four months into one table for a better data visualization, we will be using the formula below to find the defect percentage of each defect type for each particular vehicle model as illustrated in table 3.23 below. Apparently from that table, we know that the **Tiguan Allspace Highline (VW326H)** with defect type **loose bolt on exhaust mounting** achieved the highest percentage of defect in these four months among these vehicle models.

$$\text{Total Monthly Defect Percentage of a Defect Type, \%} = \frac{Total\ Defect\ Types\ of\ Each\ Model}{Total\ Quantity\ of\ Each\ Model}\ x\ 100\%$$

Table 3.23 Defect Type Percentage of each Vehicle Model

No	Defect Types	Defect Type Percentage of each Car Model (%)				
		326S	326H	326R	483R	380GTI
1	Heat shield deform			0.32		
2	Excess grease on drive shaft			1.27		
3	Missing screw on front bumper			0.32		
4	Loose bolt on exhaust mounting		3.27	1.91	0.61	
5	Missing screw on wheel housing					0.52
6	Missing plug grommet		0.20	0.32		0.52
7	Missing plug on rear bumper		0.20			
8	Loose bolt on fuel canister			0.32		
9	Missing screw on underbody cover			0.64		
10	Missing bolt on heat shield cover		0.20			
11	Loose bolt on underbody cover		0.20			

3.3.2 Data Acquisition

This section talks about how we obtain the data images of the bolt-on exhaust mounting under the body of Tiguan Highline to create our own dataset for the model training later. Factors that might affect the quality of the data were under our consideration to ensure that we have an abundant and a high quality of dataset. Smartphone with a better camera specification as shown in table 3.24 was used in this data acquisition process to capture the image of bolt on exhaust mounting under the vehicle, optimization towards the lightning conditions were made for us to minimize the shadows so that we would have a uniform illumination for all the images taken under the vehicle.

Table 3.24 Specifications of Smartphone camera

Parameters	Value
Size	3024 × 4032 pixels
Vertical resolution	72 dpi
Horizontal resolution	72 dpi
Focal length	5.1 mm
Bit depth	24

Aperture	f/1.6

Other than that, different kinds of camera distance and camera angle with a certain range were under our considerations also to capture the image of the exhaust mounting form different perspectives, covering all the necessary angles and distance for the detection to be more accurate after training. The resolution of the camera was also adjusted to be as high as possible to ensure that the picture of the mounting components are clear and detailed. Apparently we found out that the amount of data collected in the dataset were still lesser than what we expected, so we used video footage to record the exhaust mounting then we extract each particular image from the video footage frame per second. The amount of images taken and extracted from the video frame are illustrated as table 3.25 below.

Table 3.25 Total Image Taken and Extracted from the Smartphone Camera

Dataset	Images	Video Frames	Total
Bolt on exhaust mounting	350	216	566

3.3.3 Data Annotation

This section talks about how we annotate or label the defect parts of the component from the image. Image labelling plays an important role in deep learning object detection field as it is the part where we tell the model which part of the image it should be focusing and learning on. This procedure involves labelling or annotating particular items in an image along with providing details about their location and class. This kind of annotations are essential for precisely training deep learning models to locate and identify items in photos.

We need to identify and label the objects of interest in each image in our dataset, this is where a method called bounding box annotation comes in handy as it is a popular annotation technique that entails drawing bounding boxes surrounding the objects. These boxes indicate where the object is located in the pictures.

Before we do the annotation, we have to know how should we actually define the bolt to be loose or fit just by looking at the image without moving the bolt ourselves. There are two parts of the image we should be focusing on which is the bracket and the washer. First. we look at the alignment of the bracket. If the bracket is slanted in a certain degree without horizontal line then it means that the bolt is loose causing the bracket not tight enough to hold in one place

as shown in figure 3.4. Secondly, we can look at the washer friction of the bolt there. If the bolt is loose, it should have a marking or scar of the previous washer position meaning that the washer and the bolt moves due to loose bolt but we have to keep in mind that if we saw there is a hole beside the washer without the marking or scar of the washer, it does not mean that the bolt and the washer are loose causing it to move as shown in figure 3.5.

Figure 3.4 Bracket angle image data

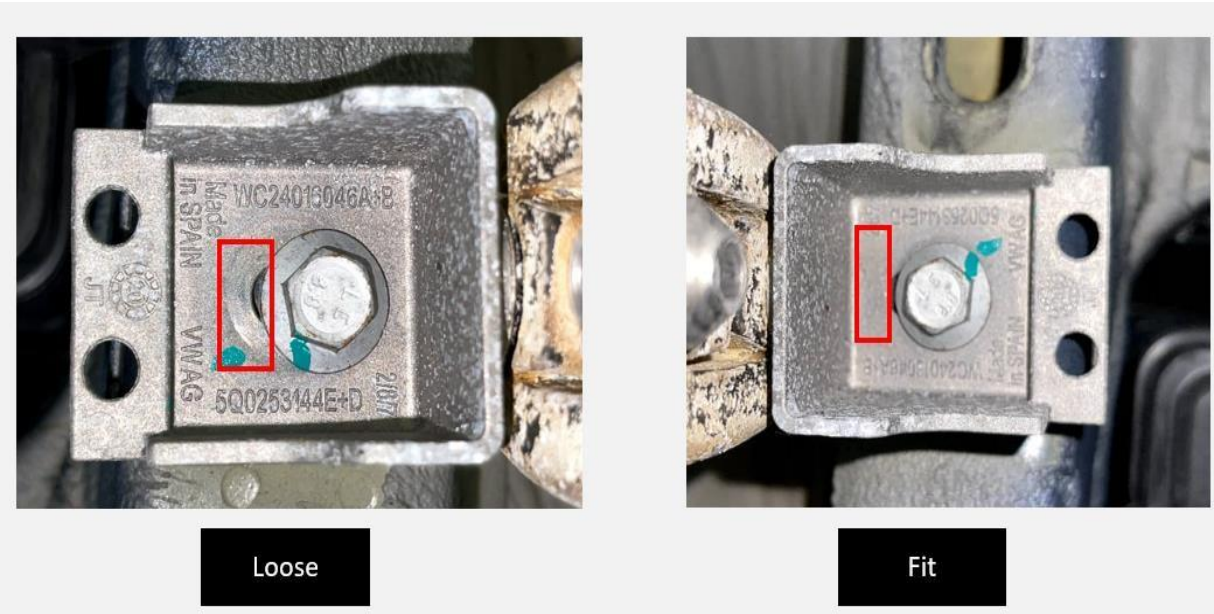


Figure 3.5 Washer friction image data

3.3.3.1 Data Labelling & Classes

There are two components from each of the data image we should be labelling which are the bracket and the washer, we can do that by using an annotation tool called Computer

Vision Annotation Tool also known as CVAT to help us annotate the image dataset. CVAT is an effective and user-friendly tool for annotating picture datasets which is a necessary step in the training of deep learning models. Like Roboflow, CVAT is adaptable to various annotation requirements since it provides a range of annotation types, including bounding boxes, polygons and points.

In order to label the images using CVAT tool, the data images are uploaded to the platform and there will be a total of four classes we will be creating as shown in figure 3.6 which are straight bracket, crooked bracket, no washer friction and washer friction. An exhaust mounting with a straight bracket and no washer friction indicate that the bolt is fit like figure 3.7 while crooked bracket and washer friction indicate that the bolt is loose like figure 3.8.

We could select various annotation tools like bounding boxes, polygons and key points to label the areas of interest in each image as shown in figure 3.7 and figure 3.8. After that, we can follow the export options provided by CVAT which the annotated images can be converted into text files that is compatible with YOLO format. The downloaded export text file as shown in figure 3.9 consist of labels which includes the class label, object coordinates and bounding box sizes.

Figure 3.6 Classes for the dataset created using CVAT

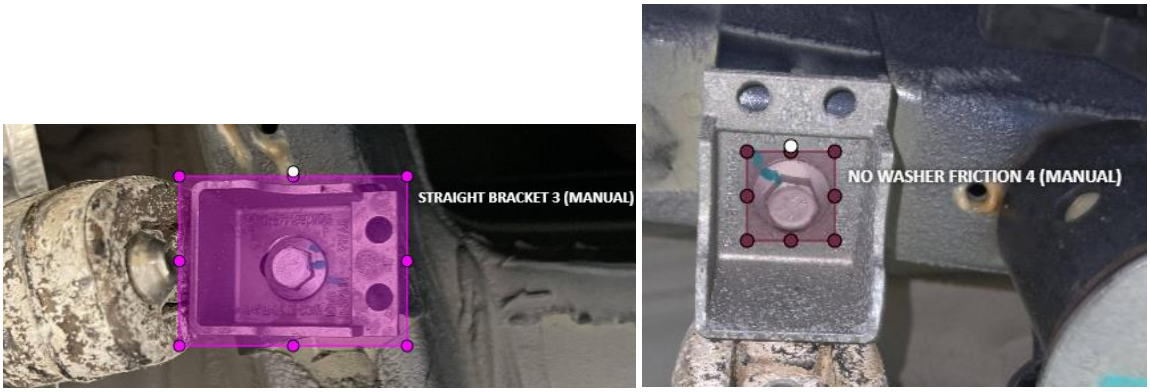


Figure 3.7 Bounding box for straight bracket and no washer friction

Figure 3.8 Bounding box for crooked bracket and washer friction

```
0 0.5120502645502646 0.4627108134920635 0.5535648148148149 0.5352827380952382
1 0.5026818783068783 0.43467757936507934 0.20158399470899477 0.13430059523809523
```

Figure 3.9 Label text file in YOLO format after export form CVAT

3.3.4 Data Augmentation

Due to the fact the amount of images in the dataset is still very less and too uniform, a process called data augmentation is taken to increase the amount of data images with a certain modification of the image brightness thereby creating a different kind of data variation in our training dataset, this can increase the detection range of our model in different brightness. At the end, our dataset will be doubled.

In the field of industry, for an object detection system to be a prototype, the minimum requirement of dataset should consist of at least 1,000 data images only it fits the purpose of being a prototype, future development will add even more data to our dataset. The dataset we augmented will have a total of 1185 data images.

After we augmented the dataset, it will be separated into three categories which 80% of the dataset will be used for training process, 10% of the dataset will be used for testing process and 10% of the dataset will be used for validation process which the amount of dataset in these three categories are tabulated in table 3.26.

Table 3.26 Dataset allocation of dataset for train, test and validation

Dataset	Training images	Validating images	Testing images	Total
Bolt on exhaust mounting	948	120	117	1185

3.3.5 Model Training

From the concept selection, Pytorch, YOLO and PyCharm will be used in our project to train the model for object detection. We are going to pick three YOLO versions which are YOLOv7, YOLOv8l and YOLOv9c to train with the dataset and pick the best among them as our model for object detection. With the benefits of PyCharm, we created folders with different environments each to run each of the YOLO versions.

For the process of training to run faster and efficiently, we have to activate our Compute Unified Device Architecture (CUDA) on our device to enable our GPU to run the training process instead of CPU. CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA to allows us to use CUDA-enabled GPU for running our application, in our project which is deep learning training.

First of all, we have to know which CUDA version is supported by our GPU on our device. In our case, we are using a MSI laptop with specification as stated in table 3.27. Then we can head over to NVIDIA CUDA toolkit downloader website to the download the suitable toolkit as shown in figure 3.10, in our case is 12.1 CUDA version then we run the installer after downloading. Since we are using Pytorch we CUDA support, we can head over their website to download the suitable Pytorch version that is compatible with CUDA 12.1 support as shown in figure 3.11, just select the options and copy the command the website gave and paste it in the terminal of you PyCharm project then it will do the rest.

Table 3.27 MSI Laptop Specifications

Parameters	MSI Katana GF66 12UD

GPU	NVIDIA® GeForce RTX™ 3070 Ti Laptop GPU 8GB GDDR6 Up to 1460MHz Boost Clock 140W
CPU	12th Gen Intel® Core™ i7 Processor
Memory	16 GB 2GB
Storage	1TB SSD
CUDA	12.1 Supported

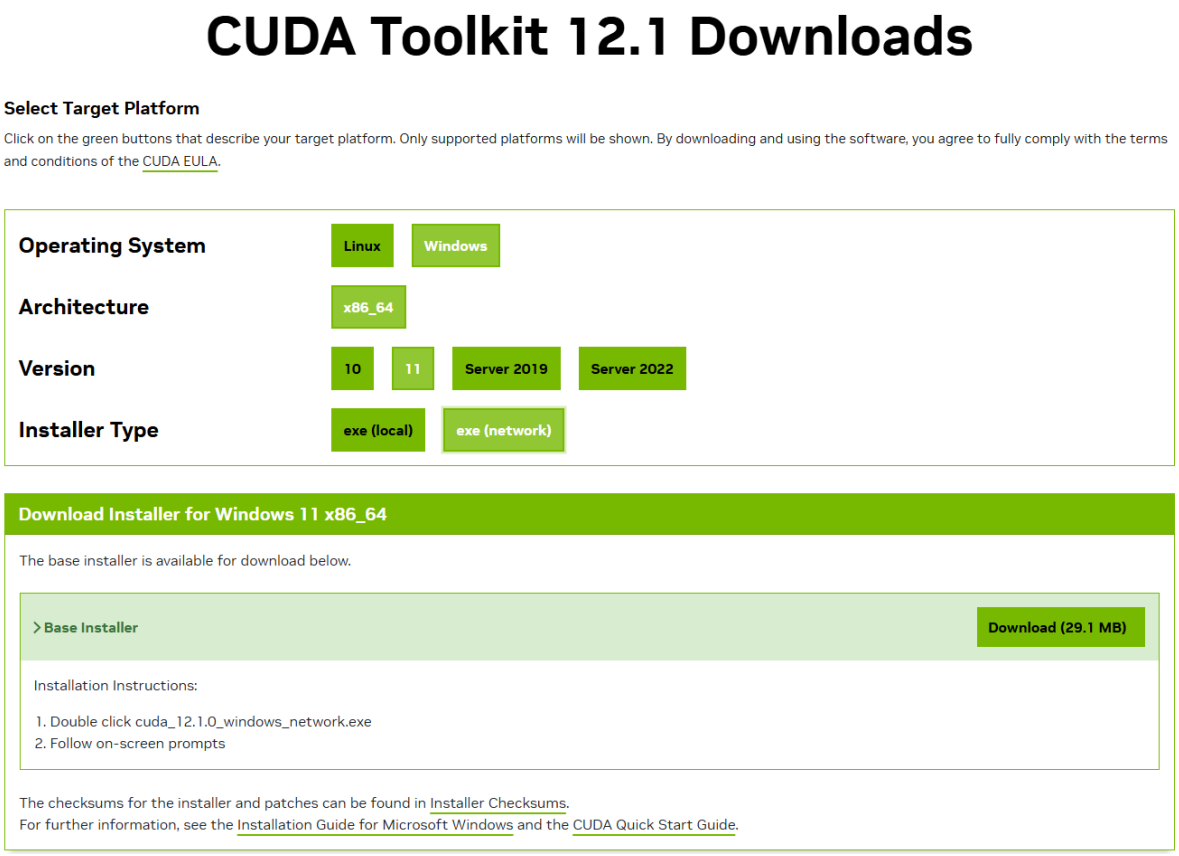


Figure 3.10 CUDA Toolkit 12.1 Downloader Website

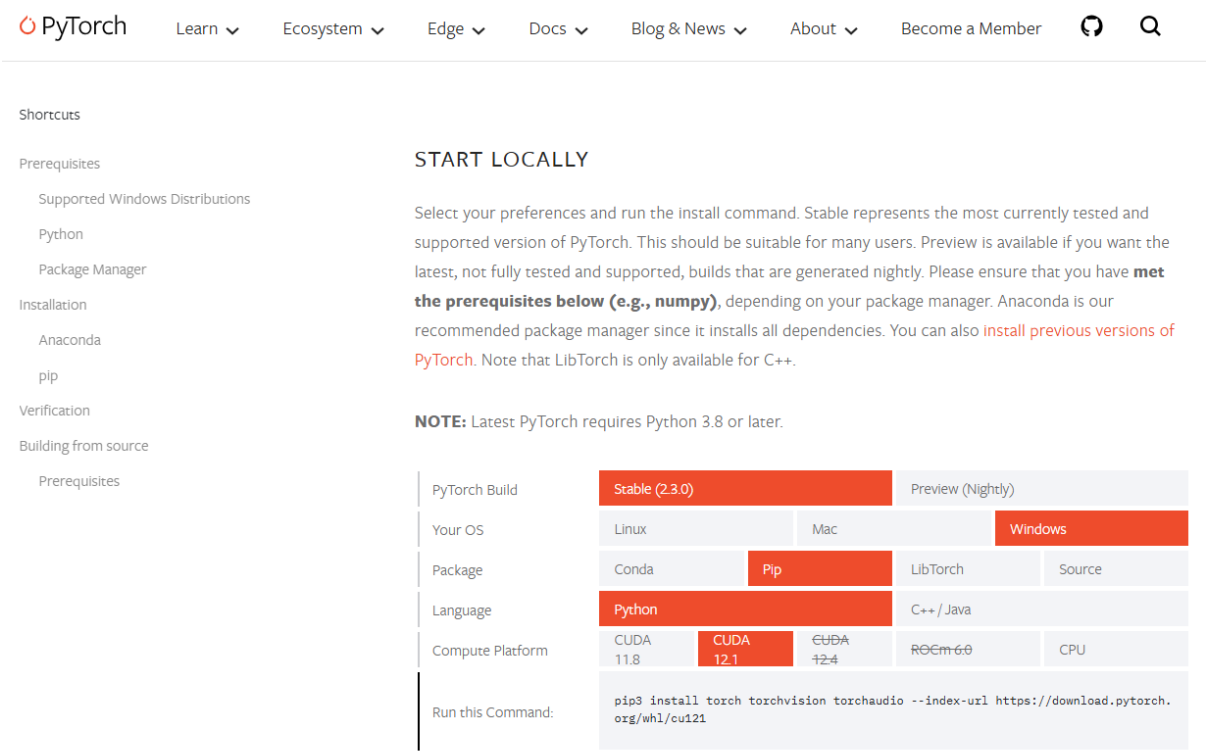


Figure 3.11 Pytorch with 12.1 CUDA support downloader website

After the installation of the Pytorch is completed, we can create a python file for CUDA testing by typing the codes as shown in figure 3.12. If CUDA is enabled and up to running then it will pop out the results as shown in figure 3.13 saying that the it is true that CUDA is available with version 12.1 and the GPU of our laptop device.

```
1 import torch
2
3 print(torch.cuda.is_available())
4 print(torch.cuda_version)
5 print(torch.cuda.get_device_name(0))
```

Figure 3.12 Codes for CUDA testing

```
True
12.1
NVIDIA GeForce RTX 3050 Ti Laptop GPU
```

Figure 3.13 CUDA testing results

After the CUDA is up and running, we can begin the next part of our training process which is to arrange the dataset folder accordingly. We can create a data folder which consists of three sub-folders train, test and valid. Each of these sub-folders should have its own sub-folder which consists of image and label folders. The folder structure look like in figure 3.14 and make sure the folder structure is in desired directory, in our case is in our PyCharm project folder. In the next few sections, we will be talking about the running of each YOLO versions for training process due to different approaches to run each version to train.

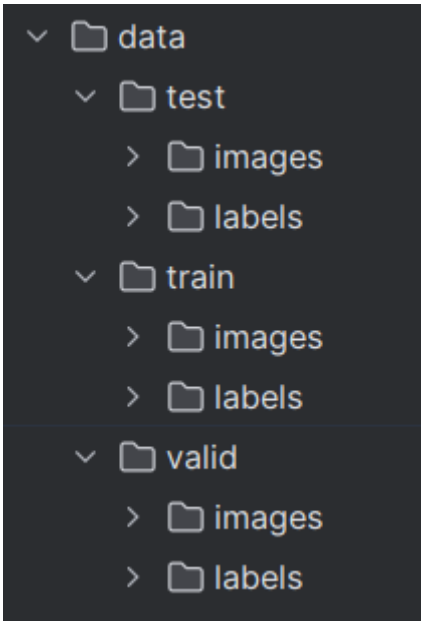


Figure 3.14 Data folder structure

3.3.5.1 YOLOv7

For YOLOv7, we will begin by downloading the YOLOv7 zip folder and YOLOv7.pt file provided by Wong Kin You from the GitHub. After that, we extract the folder and move the folder to our PyCharm project folder renaming as “yolov7-custom” while the YOLOv7.pt file we move it into “yolov7-custom” folder with the folder structure illustrated in figure 3.15.

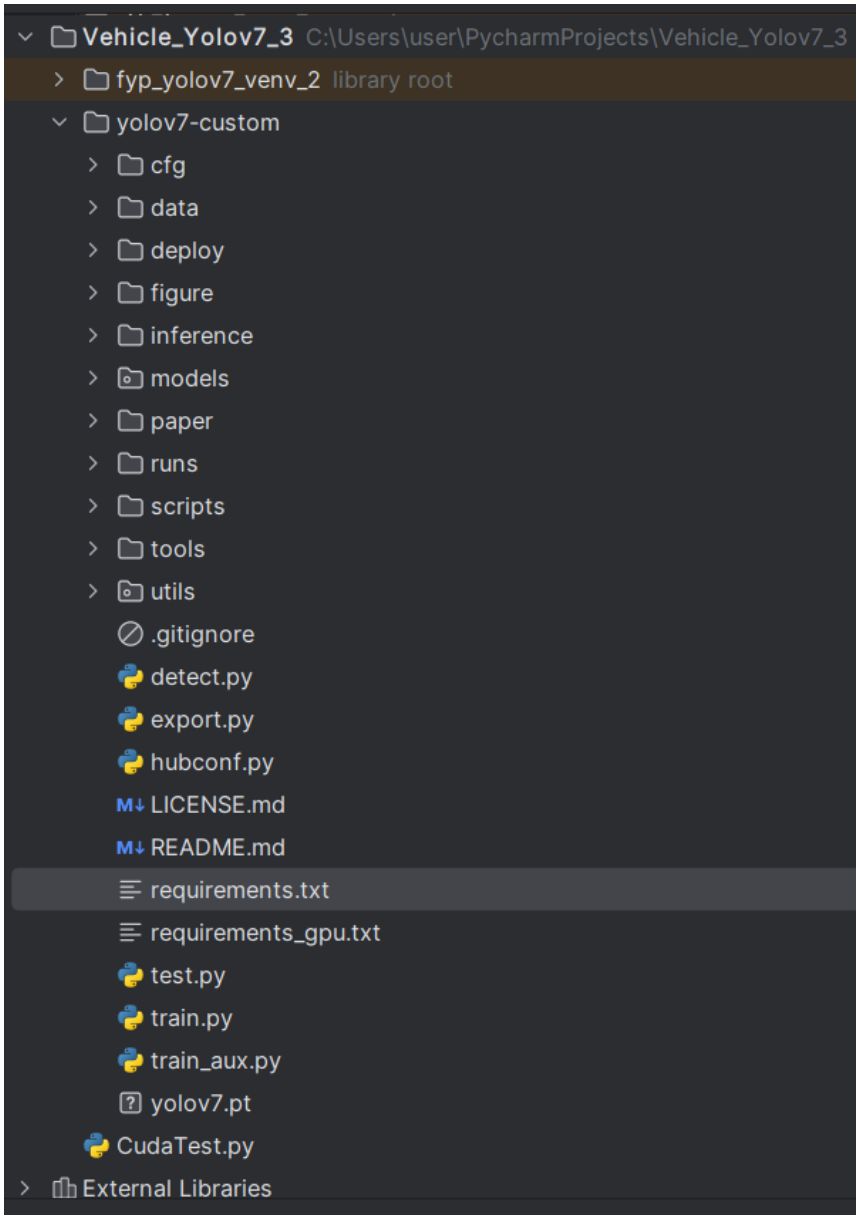


Figure 3.15 YOLOv7 project folder structure

Since we already have a CUDA up and running, we opened the requirement.txt file and remove the code at line 11 and line 12 which indicating torch and torchvision because these two code of lines will interrupt our CUDA. Then we run copy the command from the first line of the code and run in the terminal of the PyCharm project, it will install all the necessary libraries for running YOLOv7 later.

In the next step, we move our train, test and valid folders which we created in the previous section to the data folder of "yolov7-custom" folder as illustrated in figure 3.16. We create another .yaml file with the name "custom_data.yaml" and type the codes as shown in figure 3.17, this will help the system to find the data folder.

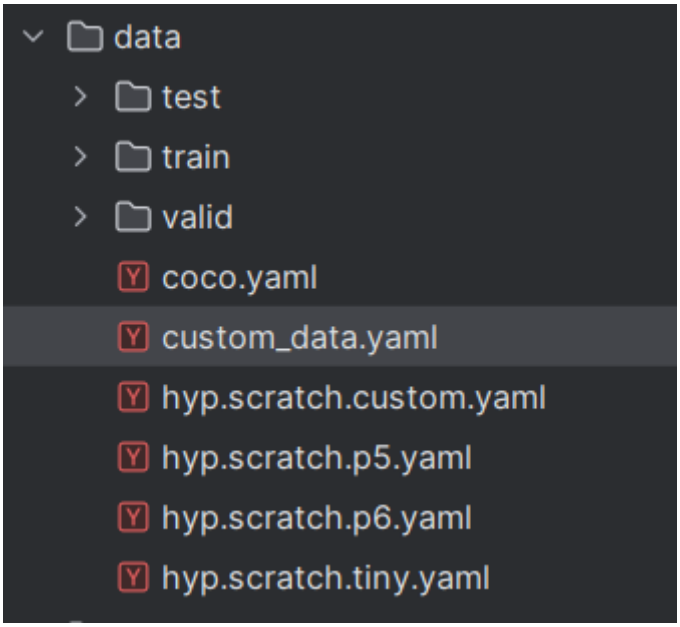


Figure 3.16 Data folder structure in “yolov7-custom”

```yaml
path: C:\Users\user\PycharmProjects\Vehicle_Yolov9_2\data  # dataset root dir
train: train/images
val: valid/images
test: test/images

# Classes
nc: 4
names: ['crooked bracket', 'no washer friction', 'straight bracket', 'washer friction']
```

Figure 3.17 Codes for “custom_data.yaml”

In the last step before we begin our training, we head into our “training” folder which is in “cfg” folder then we duplicate “yolov7.yaml” with the name “yolov7-custom.yaml”. Then we set nc from 80 to 1 as shown in figure 3.18.

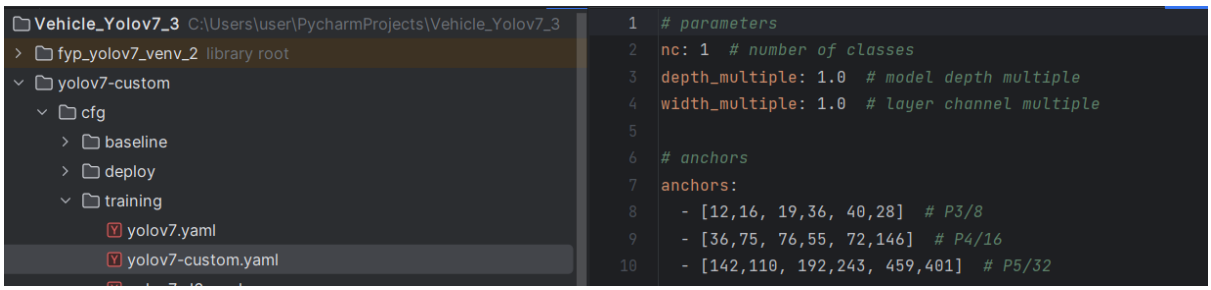


Figure 3.18 Codes and folder structure for “yolov7-custom.yaml”

Finally, we run the following command in the terminal to begin the training process.

```
python train.py --workers 8 --device 0 --batch-size 4 --epochs 300 --img 640 640 --data data/custom_data.yaml --hyp
data/hyp.scratch.custom.yaml --cfg cfg\training\yolov7-custom.yaml --name yolov7-custom --weights yolov7.pt
```

Figure 3.19 Codes for begin the training process

python train.py

The Python script train.py is run in this part. It's possible that the script is in charge of the training.

--workers 8

This argument tells the computer how many worker processes to use to load data during training. If we set it to 8, 8 processes will load data at the same time. If we have a CPU with multiple cores, this can speed up training.

--device 0

This input tells the programme which device to train on. It is set to 0 in this case, which usually means the first GPU if one is available. We can set it to -1 if you don't have a GPU or want to use the CPU instead.

--batch-size 4

This argument tells the training algorithm how many pictures to process in each round. Every time the model is trained, it will look at four pictures at a time if the batch size is 4.

--epochs 300

This input sets the number of epochs, which means how many times the whole dataset will be fed to the model while it is being trained. When the model has been trained on every sample in the collection once, an epoch is over.

--img 640 640

This argument tells the model what size pictures to take in. It's set to 640x640 pixels in this case.

--data data/custom_data.yaml

We can use this argument to tell the programme where to find a YAML file with details about the training dataset. It most likely has information like the links to image files, classes, and annotations.

--hyp data/hyp.scratch.custom.yaml
This argument tells the programme where to find a YAML file with training settings for the model. Some examples of these hyperparameters are learning rate, momentum, weight loss, and so on.

--cfg cfg\training\yolov7-custom.yaml
This argument tells the programme where to find the model settings in a YAML file. It probably has information about how the network is set up, like how many levels, filters, etc. there are.

--name yolov7-custom
The name of the model that is being learned is given by this argument. During and after training, it will be used to find the model.

--weights yolov7.pt
We can set the model's settings before training by using this argument to tell it where to find the weights file. The weights used in this case are named yolov7.pt and seem to have already been learned. These weights may have been trained on a different dataset or model architecture.

3.3.5.2 YOLOv8

There are four things that we have to do if we want use YOLOv8 to train which are moving data folders, download libraries, create python file for code to run the training and create a .yaml file. First of all, we move our data folder which contains of train, test and valid folders into our main directory of YOLOv8 PyCharm project folder, the folder structure looks like in figure 3.20.

The Next thing we have to do is to install Ultralytics library in our PyCharm project by running “pip install ultralytics”. After that, we create a .yaml file with the name “config.yaml” and type the codes as illustrated in figure 3.21. Finally, we create a python file or use the “main.py” file to type the codes as shown in figure 3.22. At the end, all we have to do is to run our “main.py” file to start the training process.

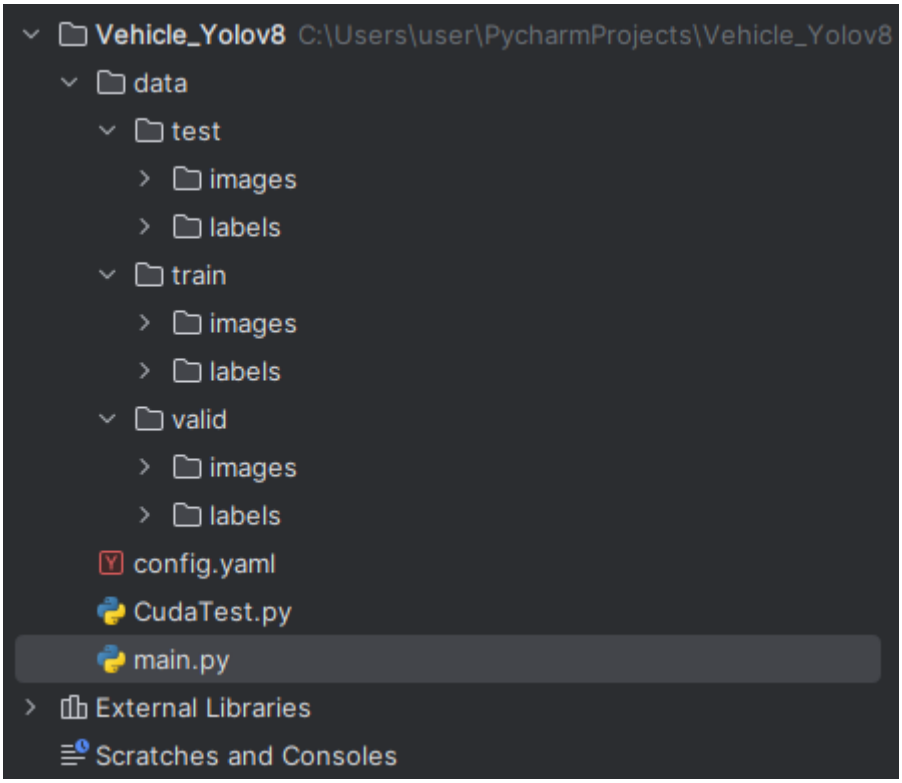


Figure 3.20 Folder structure of YOLOv8 PyCharm Project

```yaml
path: C:\Users\user\PycharmProjects\Vehicle_Yolov9_2\data  # dataset root dir
train: train/images
val: valid/images
test: test/images

# Classes
nc: 4
names: ['crooked bracket', 'no washer friction', 'straight bracket', 'washer friction']
```

Figure 3.21 Codes for “config.yaml”

```python
import torch
from ultralytics import YOLO

def main():
    print(torch.cuda.is_available())
    print(torch.cuda.get_device_name(0))
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    model = YOLO('yolov8l.pt')          # load a pretrained model (recommended for training)

    try:
        # For more: https://docs.ultralytics.com/usage/cfg/#train-settings
        # Use the model
        model.train(device=device,
                    data='config.yaml',
                    epochs=300,
                    batch=-1
                    )  # train the model
    except Exception as e:
        print("Error: ", str(e))
        raise


if __name__ == "__main__":
    main()
```

Figure 3.22 Codes for main.py

3.3.5.3 YOLOv9

For YOLOv9, it is pretty much the same procedure as in YOLOv8. There are only two different things we need to do which is to modified the file path in “config.yaml” file and the code in “main.py” file as shown in figure 3.23 and figure 3.24.

Figure 3.23 Code in “config.yaml” for YOLOv9

```python
import torch
from ultralytics import YOLO


def main():
    print(torch.cuda.is_available())
    print(torch.cuda.get_device_name(0))
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    model = YOLO('yolov9c.pt')          # load a pretrained model (recommended for training)

    try:
        # model.train(device=device, data='config.yaml', epochs=325, batch=-1)  # train the model
        model.train(device=device, data='config.yaml', epochs=100, batch=-1)
    except Exception as e:
        print("Error: ", str(e))
        raise


if __name__ == "__main__":
    main()

```

Figure 3.24 Codes in “main.py” for YOLOv9

3.3.6 Database

Based on concept selection, we used MySQL and PhpMyAdmin as our data storage for the detection result of the system. Structured Query Language (SQL) is used to control and change databases in MySQL, it is an open-source relational database management system (RDBMS). MySQL is an important part of many web services, especially those that use the LAMP stack (Linux, Apache, MySQL, PHP/Python/Perl). It is known for being fast, scalable, and widely used. It can be used on multiple platforms, is good at managing big databases and a lot of users and can be used for anything from small websites to large enterprise applications.

PhpMyAdmin is a free, open-source web-based tool for handling MySQL databases through a graphical user interface (GUI). PhpMyAdmin makes it easier to do things like create and change databases and tables, run SQL queries and manage user rights, all without having to use command-line tools. It can be accessed through a web browser. It has tools for managing users, maintaining databases, and importing and exporting data. This makes it an accessible and easy-to-use option for both new and experienced MySQL database administrators.

3.3.6.1 MySQL

This section explains about how we start from installing MySQL to creating our own user. The steps and explanation are tabulated in table 3.28, all we have to do is just follow the steps and paste the command in our terminal or linux.

Table 3.28 Steps for creating user in MySQL

Steps	Command	Remarks
1	sudo apt update	
2	sudo apt install mysql-server	Type and enter Y for confirm installing
3	sudo mysql	
4	alter user 'root'@'localhost' identified with mysql_native_pasword by 'yourpassword';	Change 'yourpassword' to preferred password
5	exit	
6	mysql_secure_installation	Enter password from step 4 Follow the instruction and select prefer options In our case: yes > 0 > no > yes > yes > yes > yes
7	Mysql -u root -p	Enter password from step 4
8	create database data1;	Change 'data1 to preferred name
9	show schemas;	To see database it created or not
10	Create user 'abstract-programmer'@'localhost' identified with mysql_native_password by 'yourpassword2';	Change 'abstract-programmer' and 'yourpassword2' to preferred username and password respectively.
11	use mysql	
12	select user from user;	Check whether user is created
13	Grant all on data1.* to 'abstract-programmer'@'localhost';	Change 'data1' and 'abstract-programmer' to database and username from step 8 and step 10 respectively.
14	exit	

3.3.6.2 PhpMyAdmin

The installation process for PhpMyAdmin are stated in table 3.29 below. After that we can head to PhpMyAdmin local webpage by this URL link http://localhost/phpmyadmin . The login page of PhpMyAdmin should pop out as in figure 3.25. We can login by entering the username and password we created, in our case are 'abstract-programmer' and 'yourpassword2' respectively.

After that, we can click the plus sign of our 'data1' which is the database we created then it will extend out. We can create our data table by clicking the new, then we can follow figure 3.26 to fill up the name and necessary slots with the information we need. In our project, we are going to need five data tables to store our data results due to the designs we are going to create later on which are 'data_bulk_image', 'data_capture', 'data_image', 'data_video' and 'data_live' as shown in figure 3.27.

Table 3.29 Installation process for PhpMyAdmin

Steps	Commands	Remarks
1	sudo apt install php	Press Y and enter for installation confirm
2	Sudo apt install phpmyadmin	Press Y and enter for installation confirm Choose No Choose Apache2
3	sudo apt install apache2	
4	sudo systemctl start apache2	Start apache2
5	sudo systemctl enable apache2	Start apache2 every time device boot
6	sudo systemctl status apache2	Check whether apache2 is started

Figure 3.25 Login page of PhpMyAdmin

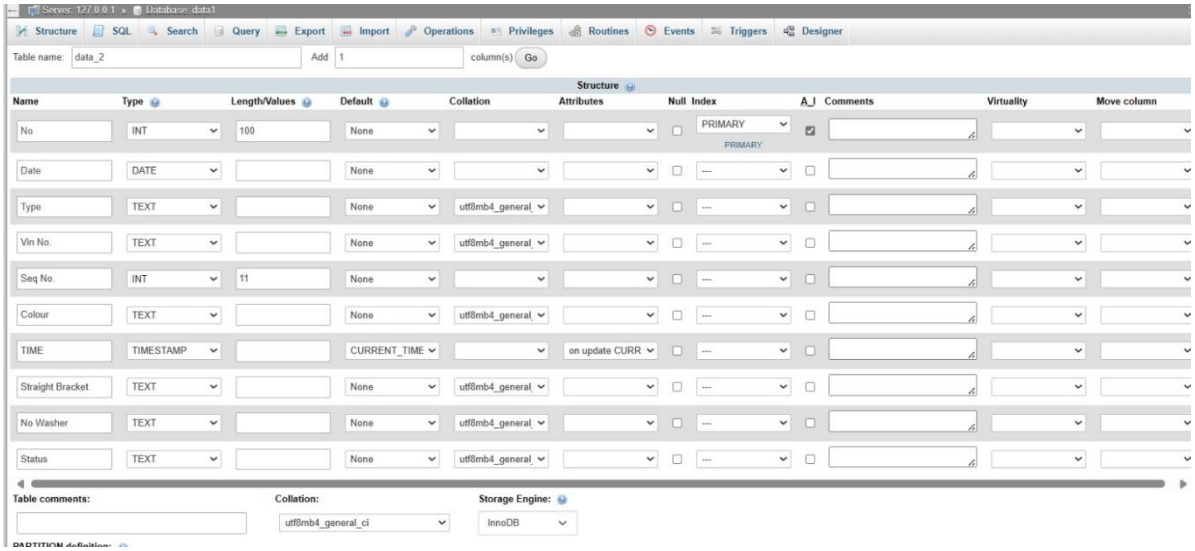


Figure 3.26 Table create in database ‘data1’

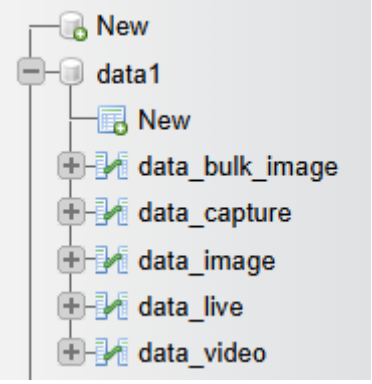


Figure 3.27 Data structure of database 'data1' PhpMyAdmin

3.3.7 Graphical User Interface

In the case of concept selection, we are going to used Streamlit as the platform for data visualization. Streamlit is an open-source Python tool that makes it easier to make web-based apps that are interactive for data science and machine learning projects. Developers and data scientists can quickly turn data scripts into shared web apps without having to know a lot about web development.

For our design, we are planning to create a total of five webpages. The first page will be the project explanation. The second page will be showing us the data results collected with bar chart displaying the vehicle results based on months, we can also insert certain parameters at the sidebar for data query. The third page will be for single image object detection where we can upload our image to do object detection and it also shows the data stored with a bar chart, we can also insert data for data storing in our database.

The fourth page is going to be a place where we can upload a bunch of images to do object detection and allow us to download the detected result images in a zip file. The fifth page will allow us to upload a video and do object detection on it. The sixth page will be the page where it will send a signal to the camera to take picture and send it back to do object detection which this is the main page we are going to use all the time as it is real-time wireless data transferring and there will be a pop out info telling us that whether the bolt is loose or not. The last page is where we can open our webcam to test the real-time object detection.

Before we run all these files, we have to install all the necessary libraries as below:

1. pip install ultralytics
2. pip install streamlit

3. pip install pandas
4. pip install mysql.connector
5. pip install datetime
6. pip install matplotlib
7. pip install pillow

After that, we can start designing the webpages in our PyCharm project and the folder structure is illustrated in figure 3.28. The 'CodeTest' folder is where we store the code for testing purposes where the 'images' folder is where we stored our demo images. The 'pages' folder is where we stored the pages other than homepage and the 'video' folder is where we stored the demo videos.

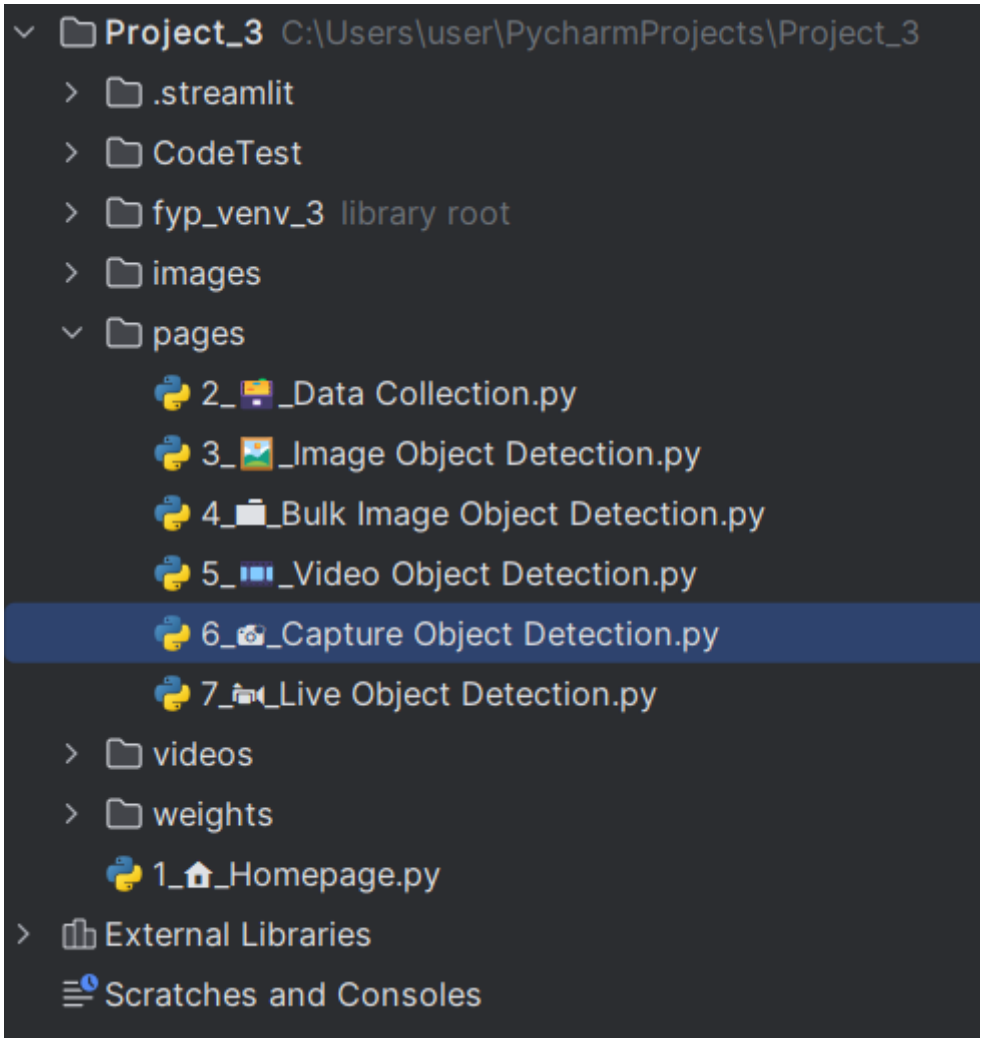


Figure 3.28 Folder structure of Streamlit project

The codes for each of the webpages are as below:

Appendix F: Homepage

Appendix G: Data Collection

Appendix H: Image Object Detection

Appendix I: Bulk Image Object Detection

Appendix J: Video Object Detection

Appendix K: Capture Object Detection

Appendix L: Live Object Detection

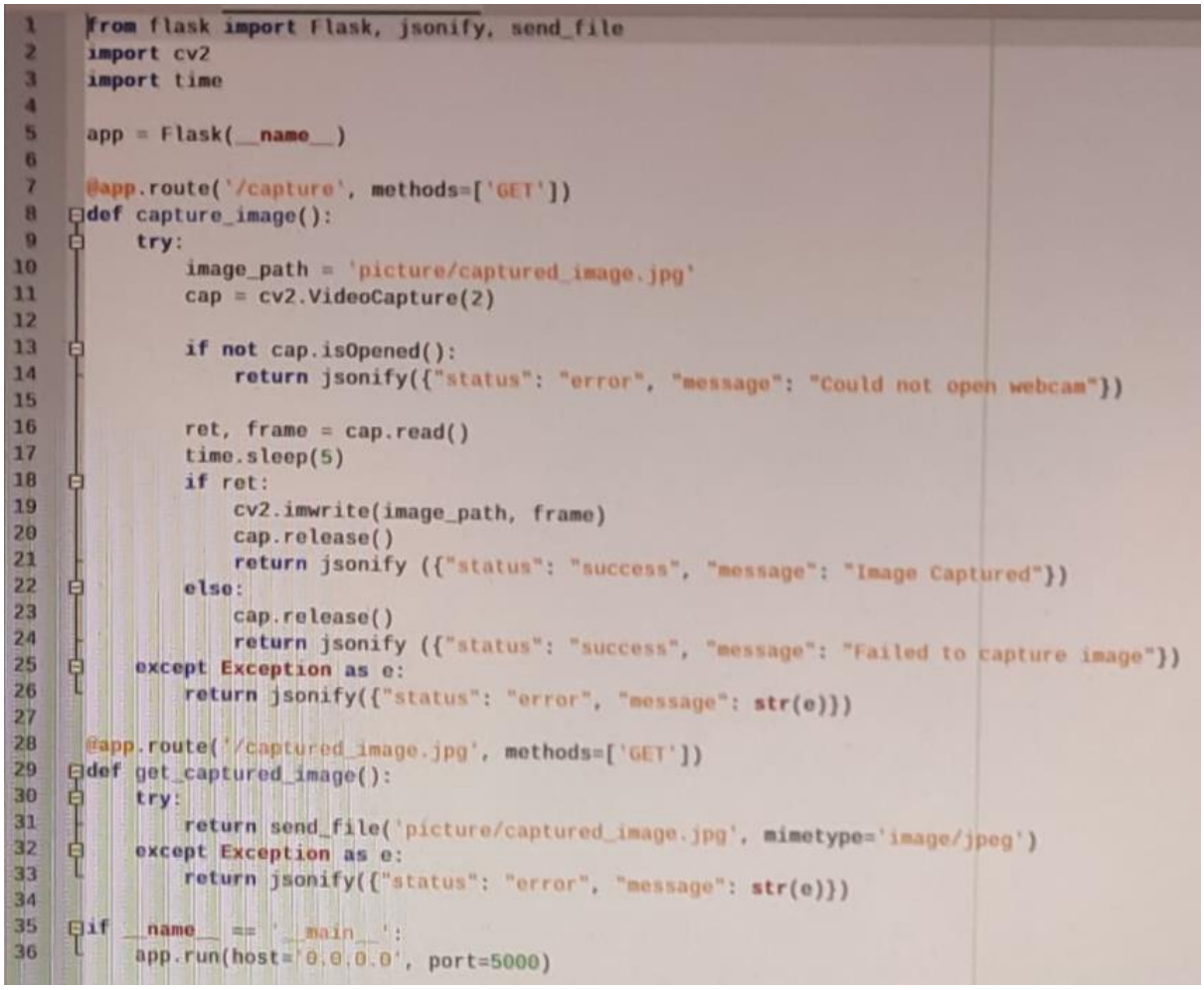
3.3.8 Pi Zero WH & Camera

This section talks about how we plan to make Pi Zero WH to accept signal if triggered from Streamlit and snap the picture then send it back to Streamlit to do object detection. The first thing we need to do is to setup our Pi Zero WH OS system which will be explained in the next section.

In order to allow the signal to transfer and work, we planned to use Flask server library. Flask is a lightweight and adaptable Python web framework that makes it simple for programmers to create online apps and APIs. The simplicity and minimalism of Flask are among its primary characteristics. It offers all of the necessary components and tools to build a web server, including routing, templating, and request processing, without adding extra dependencies or imposing a certain project structure.

Our plan is simple, create a web server between Streamlit and Pi Zero WH then Streamlit send signal to that web server and Pi Zero WH will read it from there. If there is a signal received by Pi Zero WH then it will run a python file which will tell the camera to take the picture and send it back to the web server.

We need to create a folder named ‘example’ in our Pi Zero WH and create a python file inside that folder with the following code typed in it as shown in figure. We have to create a folder named “picture” inside the ‘example’ folder to store the image taken by the camera. If we run the python file, it will showed an IP address in an URL link format where we will be going to use that URL link to access the webserver either sending signal or retrieving the image.



```
from flask import Flask, jsonify, send_file
import cv2
import time

app = Flask(__name__)

@app.route('/capture', methods=['GET'])
def capture_image():
    try:
        image_path = 'picture/captured_image.jpg'
        cap = cv2.VideoCapture(2)

        if not cap.isOpened():
            return jsonify({"status": "error", "message": "Could not open webcam"})

        ret, frame = cap.read()
        time.sleep(5)
        if ret:
            cv2.imwrite(image_path, frame)
            cap.release()
            return jsonify ({"status": "success", "message": "Image Captured"})
        else:
            cap.release()
            return jsonify ({"status": "success", "message": "Failed to capture image"})
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)})

@app.route('/captured_image.jpg', methods=['GET'])
def get_captured_image():
    try:
        return send_file('picture/captured_image.jpg', mimetype='image/jpeg')
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Figure 3.29 Codes for receive signal to take picture and send it back

After we have created the python file for receiving signal and send the image back, now we have to make sure that every time the Pi Zero WH is turned on, the python file will run itself. We can do this by accessing to the Systemd Service file in the OS system of Pi Zero WH, the steps are tabulated in table 3.30.

Table 3.30 Steps for adding configuration to the service file

Steps	Commands	Remarks
1	sudo nano /etc/systemd/system/flask_server.service	
2	[Unit] Description=Flask Server for Camera Capture After=network.target [Service]	Type the following codes inside Ctrl + x to exit then press Y then press enter

	ExecStart=/usr/bin/python3 /home/pi/app.py WorkingDirectory=/home/pi/ StandardOutput=inherit StandardError=inherit Restart=always User=pi [Install] WantedBy=multi-user.target
3	sudo systemctl enable flask_server.service
4	sudo systemctl start flask_server.service
5	sudo systemctl status flask_server.service

3.4 Electrical & Electronics

This section talks about the electrical components used and their specifications. Furthermore, we will be explaining about how should we setup the hardware especially Jetson Orin Nano and Pi Zero WH.

3.4.1 Hardware Specifications

NVidia Jetson Orin Nano 8GB Dev Kit

Figure 3.30 NVidia Jetson Orin Nano 8GB Dev Kit

Source: NVidia Jetson Orin Nano 8GB Dev Kit [119].

Table 3.31 Specifications of NVidia Jetson Orin Nano 8GB Dev Kit

Specification	Detail

Processor	Integrated with NVIDIA's advanced GPU architecture
Memory	8GB LPDDR4
Storage	MicroSD card slot for OS and storage
Connectivity	Gigabit Ethernet, USB ports, and wireless network capabilities with an additional dongle
I/O interfaces	Multiple GPIO, I2C, SPI interfaces for peripherals
Size	Compact form factor suitable for embedded applications

Raspberry Pi Zero WH (Wireless with Header)

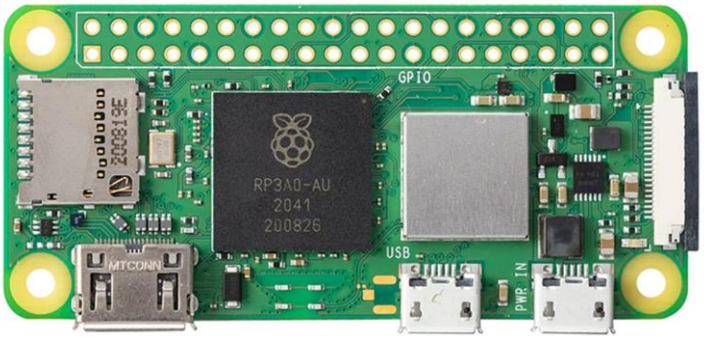


Figure 3.31 Raspberry Pi Zero WH (Wireless with Header)

Source: Raspberry Pi Zero WH (with Header) [121].

Table 3.32 Specifications of Raspberry Pi Zero WH (Wireless with Header)

Specification	**Detail**
Processor	Single-core CPU
Memory	512MB RAM
Connectivity	Wi-Fi 802.11 b/g/n, Bluetooth 4.1, BLE (Bluetooth Low Energy)
I/O ports	Mini HDMI, USB On-The-Go port, Micro USB power
GPIO	40-pin GPIO header pre-soldered

Hikvision 1080P HD Webcam DS-U02

Figure 3.32 HIKVISION Web Camera

Table 3.33 Specifications of Hikvision Web Camera

Specification	Detail
Frane Rate	30 fps
Resolution	1080P (1920 x 1080)
Field of View	95°
Connection Interface	USB 2.0
Mounting Options	Universal clip for laptops, LCDs, or monitors

RPi Approved MakerDisk C10 A1 uSD with RPi OS - 32GB

Figure 3.33 RPi Approved MakerDisk C10 A1 uSD with RPi OS - 32GB

Source: 32GB Raspberry Pi Approved MakerDisk uSD with RPi OS [125].

Table 3.34 Specifications of Jetson NANO USB Wireless Network Dongle

Specification	Detail

Capacity	32GB
Class	Class 10, A1 rating for faster application performance
Preloaded	Comes with Raspberry Pi OS pre-installed
Compatibility	Designed for use with Raspberry Pi devices

Official RPi 12.5W PSU microB UK Plug

Figure 3.34 Official RPi 12.5W PSU microB UK Plug

Source: 32GB Raspberry Pi Approved MakerDisk uSD with RPi OS [125].

Table 3.35 Specifications of Official RPi 12.5W PSU microB UK Plug

Specification	**Detail**
Output	5.1V / 2.5A DC output
Connector	Micro USB B
Power	12.5W, suitable for powering Raspberry Pi devices
Plug type	UK plug with fuse

Mini HDMI Adapter

Figure 3.35 Mini HDMI Adapter

Source: Mini HDMI Adapter [127].

Table 3.36 Specifications of Mini HDMI Adapter

Specification	**Detail**
Interface	Converts from mini HDMI to standard HDMI
Compatibility	Designed for devices with mini HDMI ports, like Raspberry Pi Zero

Official RPi micro USB adapter

Figure 3.36 Official RPi micro USB adapter

Source: Official RPi micro USB Adapter [128].

Table 3.37 Specifications of Official RPi micro USB adapter

Specification	**Detail**
Type	Micro USB to USB adapter
Functionality	Enables connection of USB devices to micro USB ports

USB2.0 Cytron microSD Card Reader/Writer Adapter

Figure 3.37 USB2.0 Cytron microSD Card Reader/Writer Adapter

Source: USB2.0 Cytron microSD Card Reader/Writer [129].

Table 3.38 Specifications of USB2.0 Cytron microSD Card Reader/Writer Adapter

Specification	Detail
Interface	USB 2.0 for fast data transfer
Compatibility	Designed for reading and writing to microSD cards

3.4.2 Hardware Setup

In this section, we are going to talk about how we setup the Jetson Orin Nano and Pi Zero WH. Both of them are single-board computer but with different specifications as specified in previous section. Jetson Orin Nano board is going to be placed at the production line where the operator can view and do object detection from there whereas Pi Zero WH is going to be fixed under the station under the vehicle along with the camera to standby for taking the picture of the underbody vehicle.

Operators or future engineer can add other features or add modifications to the system by modifying the Jetson Orin Nano which will be talked about in chapter 5 later about future improvements.

3.4.2.1 Jetson Orin Nano

To setup Jetson Orin Nano, there are two ways of doing it which is either with the help of SDK manger or shell script but there is a requirement for running both which is it needs Linux system to run the program to setup Jetson Orin Nano. Since SDK manger is much user friendly, we are going to use that to help us setup the Jetson so we are going to need a Linux

based laptop to install SDK manger, virtual machine is unavailable because it cannot flash Jetson board due to the USB plug connected to it. Personal guess would be the port from the laptop does not directly link to virtual machines which might be the reason causing it to fail when we flash our Jetson from virtual machine Linux. There are two ways to use SDK manager which are either we used another Linux based laptop or we dual-boot our own laptop to have both Windows and Linux OS system in it.

After we have our Linux system up and running, we can start installing SDK manager from NVIDIA website and run the application. Once the installation is complete, it will ask us to login before starting the first step of setting up our Jetson board. After the login is completed, the first step page will pop out as illustrated in figure 3.38 and we can change the options if required. After we pressed continue, step 2 allowed us to determine which Jetson software we are going to install and we can change the download folder if needed.

In step 3, there are two ways for us to setup the Jetson. If the Jetson has been flash before with exist username and password, then we can used the automatic setup as shown in figure 3.40. If we are flashing the Jetson board for the first time, we can do the manual setup, select the necessary options for manual setup as shown in figure 3.41 then we connect the GPIO pin 9 and pin 10 together through a jumper as shown in figure 3.42 to force the Jetson to be in a recovery mode then we turned on the switch and plug the USB cable of Jetson to our device.

After the Jetson has been detected, we click flash then we wait until another pop-up window show up as shown in figure 3.43. After the window as shown in figure 3.43 has pop-up, we have configure the Jetson before proceeding to the next step. To do that, we have to used a HDMI cable to connect the Jetson board to a monitor or screen and we also need to plug in a keyboard and a mouse to the Jetson board for us to configure the Jetson. We just need to follow the instructions shown by the Jetson on the monitor until the Jetson reboot itself, the most important part is setting up an username and a password. After that, we can proceed we the SDK manager (figure 3.43) to insert the username and password that we just created in Jetson Linux and click install then process will run until step 4 which is completed.

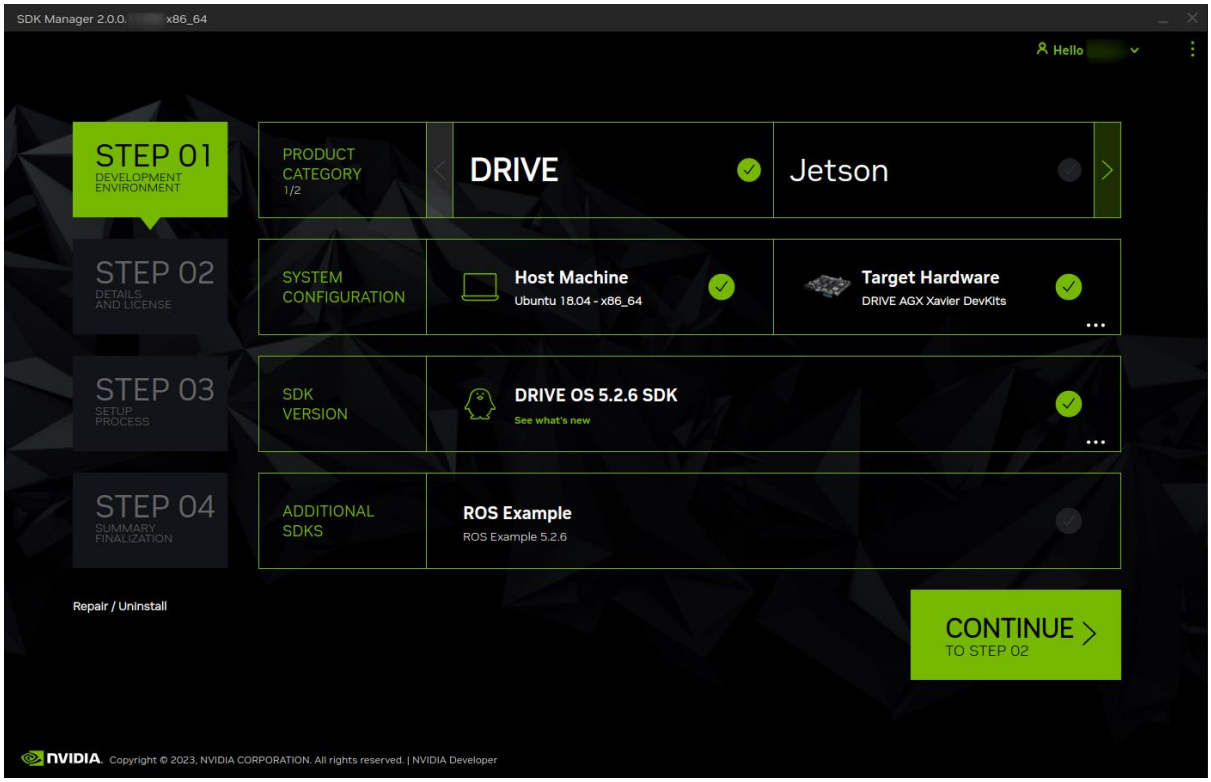


Figure 3.38 Step 1 of SDK manager

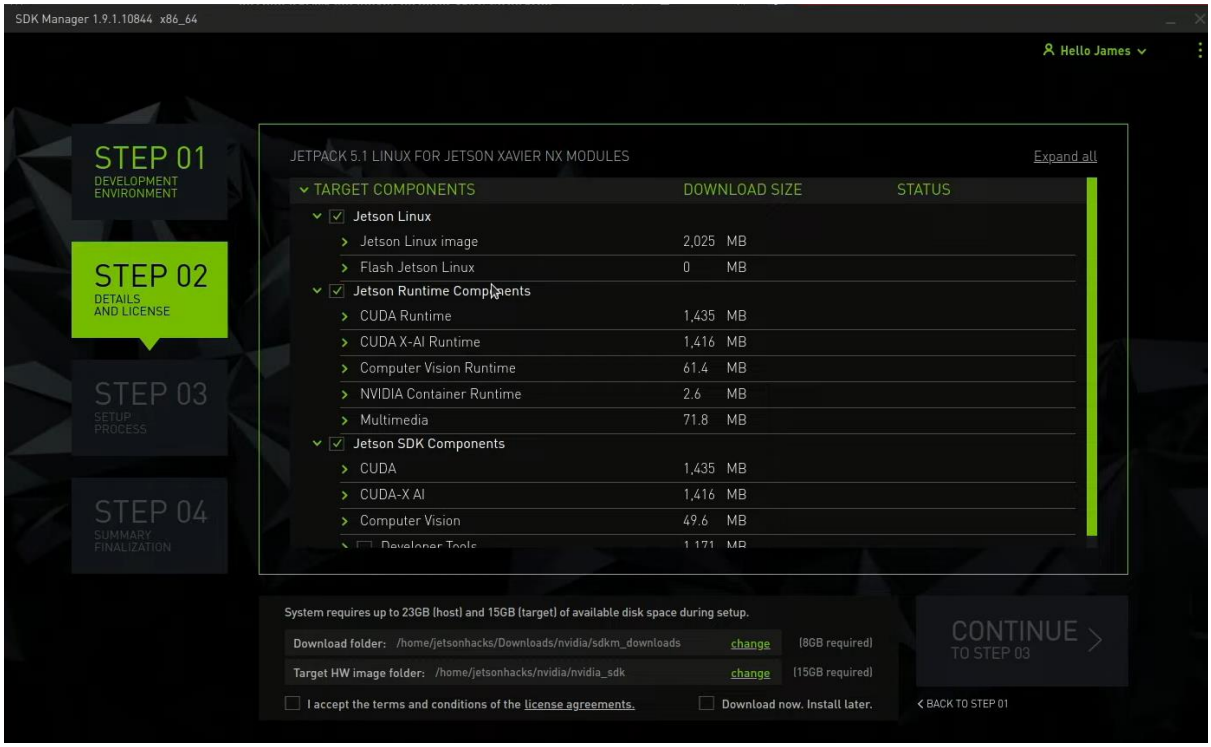


Figure 3.39 Step 2 of SDK manager

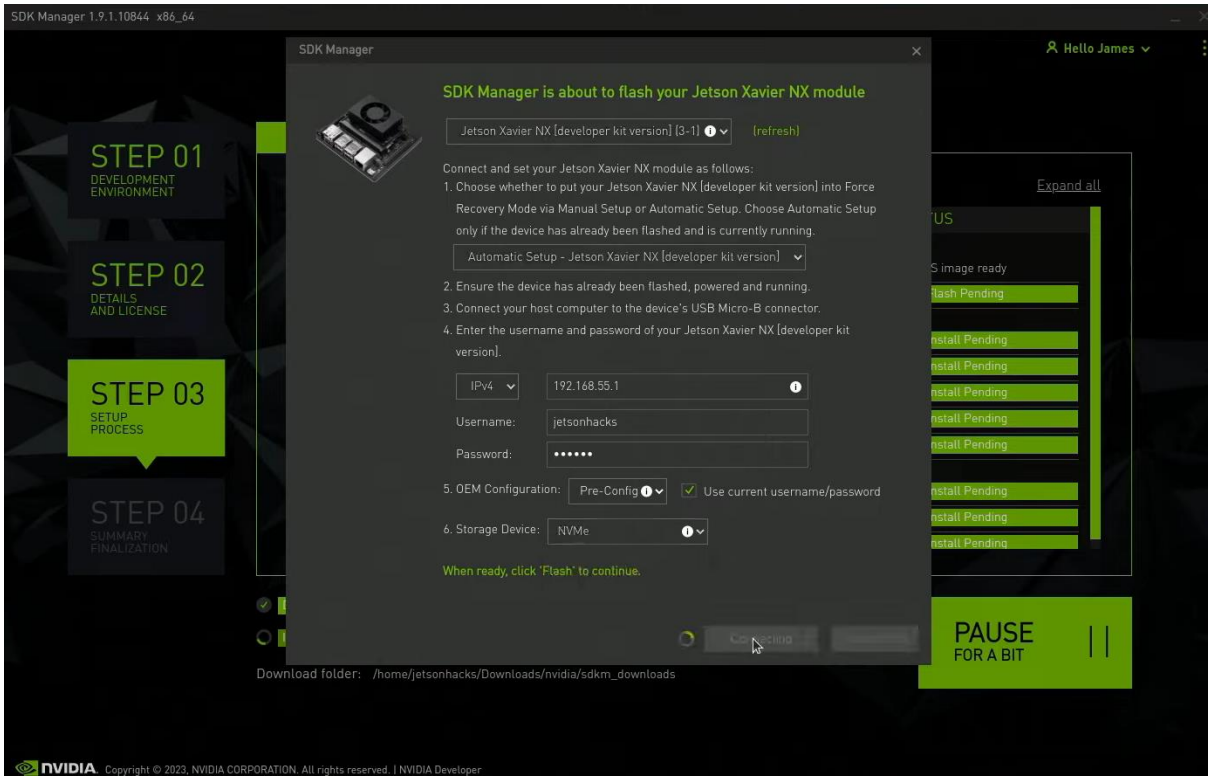


Figure 3.40 Automatic setup in step 3 of SDK manager

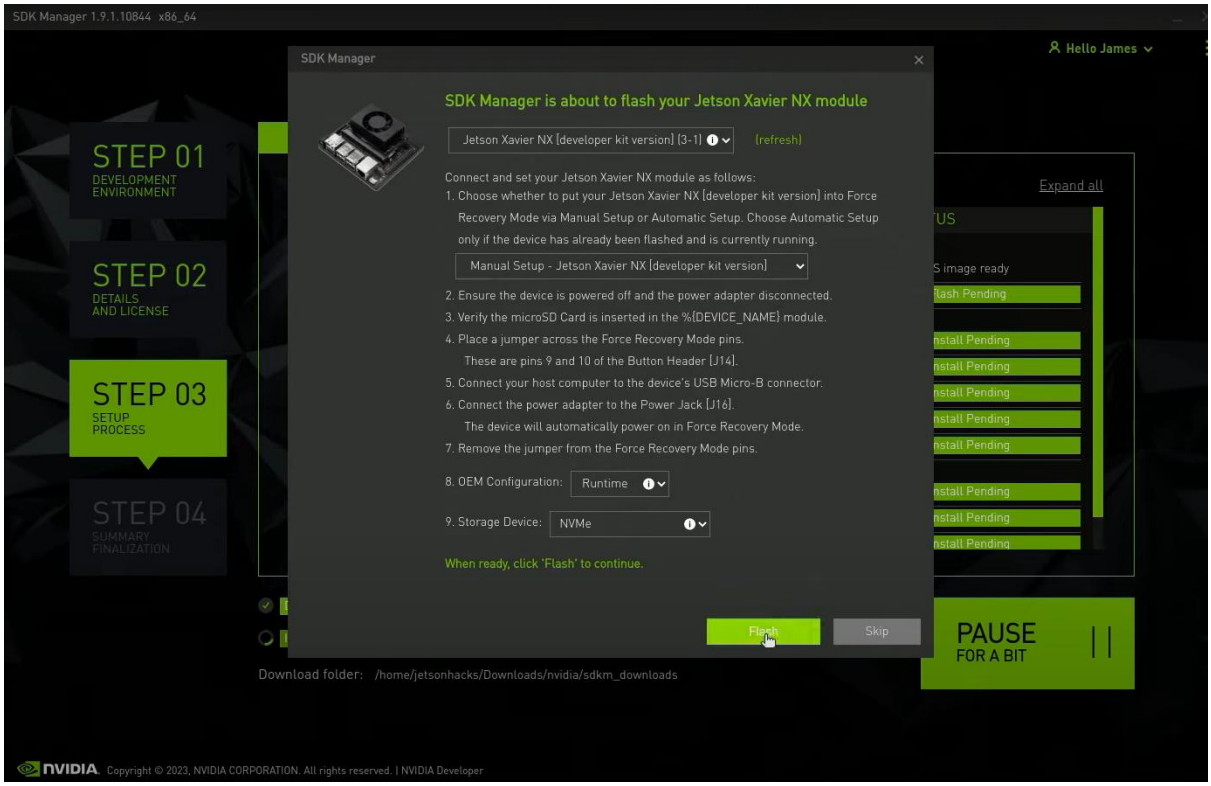


Figure 3.41 Manual setup in step 3 of SDK manager

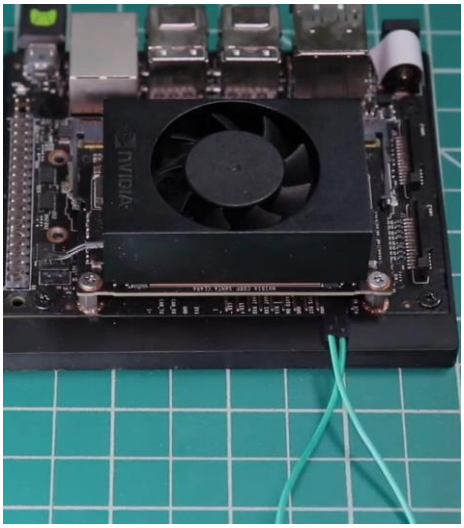


Figure 3.42 Connecting pins through jumper for force recovery mode

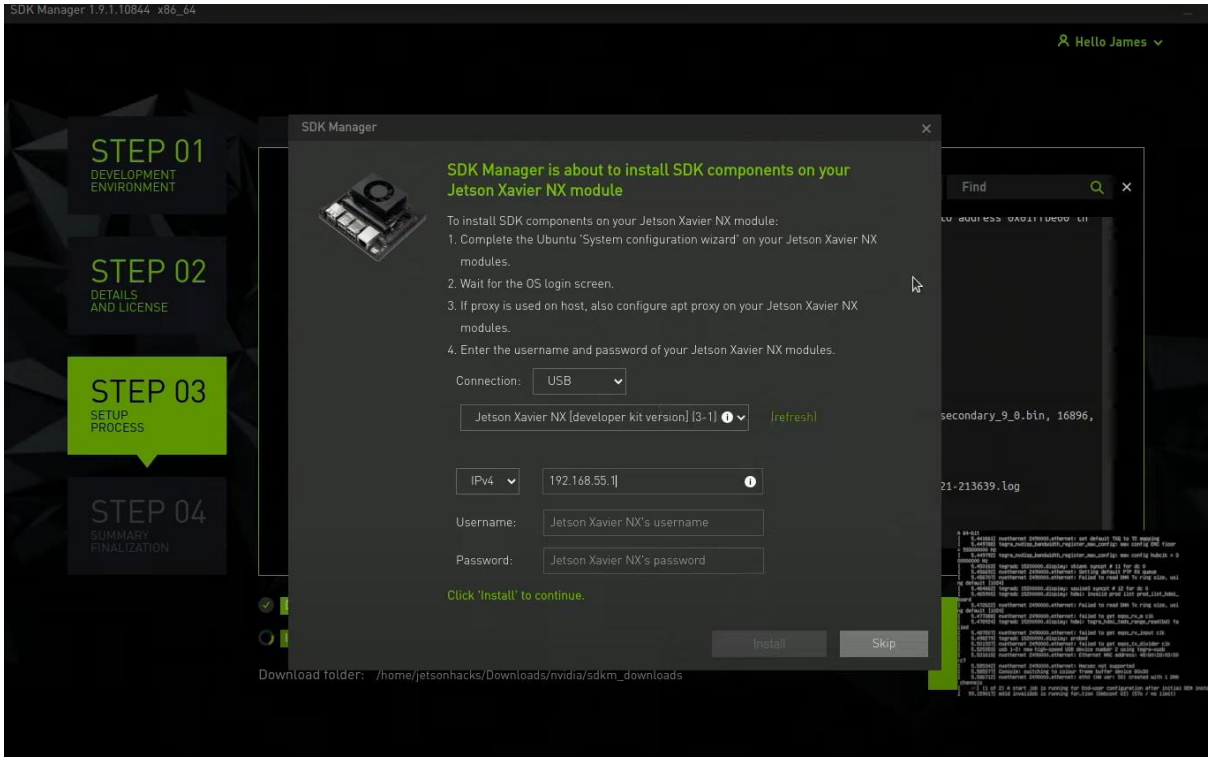


Figure 3.43 SDK components installation menu

3.4.2.2 Pi Zero WH

To setup Pi Zero WH with Raspberry Pi OS system, the first thing we need to do is to install Raspberry Pi Imager from its official website and run the Raspberry Pi Imager until the window pop-up as shown in figure 3.44. The second thing we need to do is to insert the SD card into the microSD USB card reader and plug it into our laptop. We have to format the SD

card before we install the Raspberry Pi OS system inside because it might cause some issue or errors if we did not format it.

For the “CHOOSE DEVICE” option, we have to select Raspberry Pi Zero WH from it and select “Erase” from the “CHOOSE OS” option then select our SD card from the “CHOOSE STORAGE” option, press Next and wait for the process to completed.

After the process is completed, eject and plug in back the SD card reader then this time we pick Raspberry Pi OS (32-bit) from the “CHOOSE OS” option then press Next. If there is a pop-window shown, we can select the edit settings to set our username, password, Wi-Fi and other configurations as shown in figure 3.45. After that, just wait for the process to complete and insert the SD card to our Pi Zero WH. If we plug the Pi Zero WH with a screen, there should be a Raspberry Pi logo screen displayed it out then it means we successfully setup the OS system in Pi Zero WH.

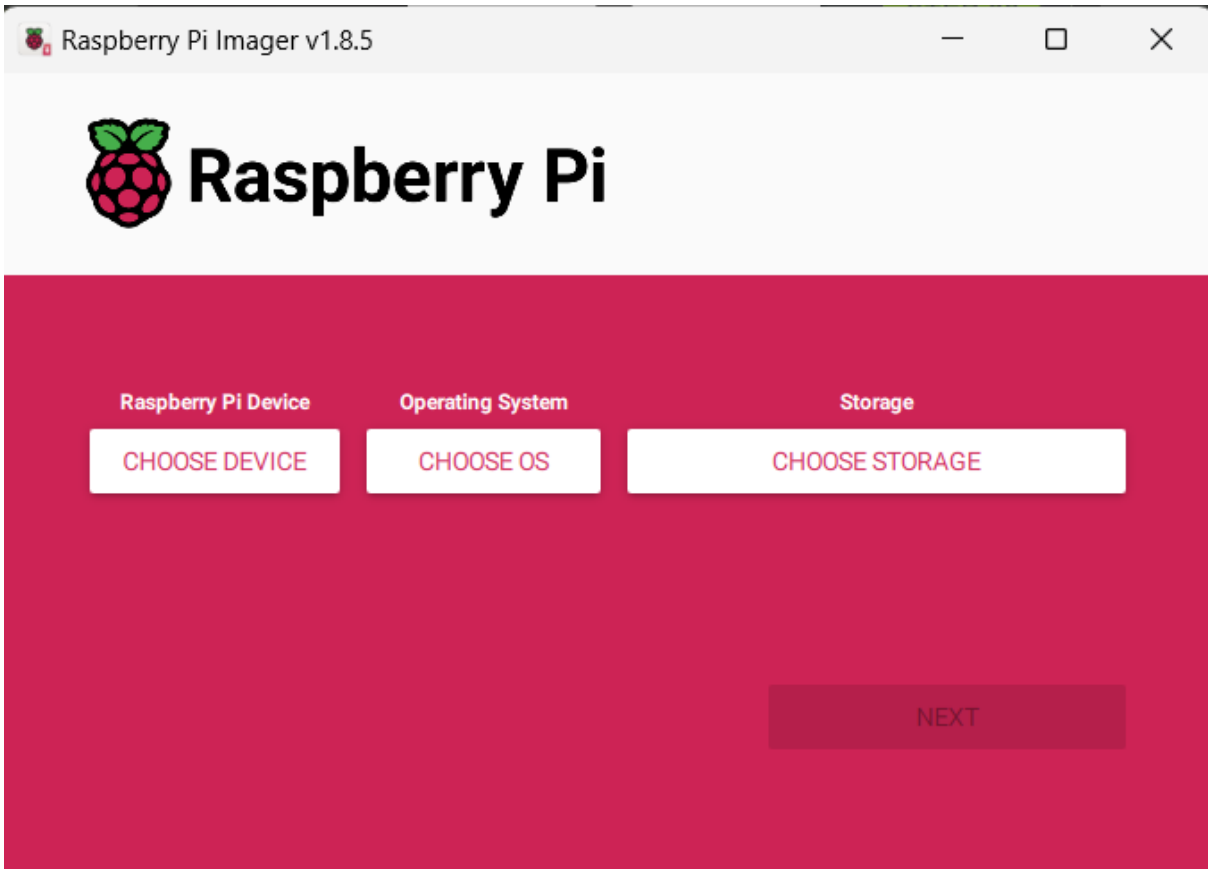


Figure 3.44 Raspberry Pi Imager pop-up window

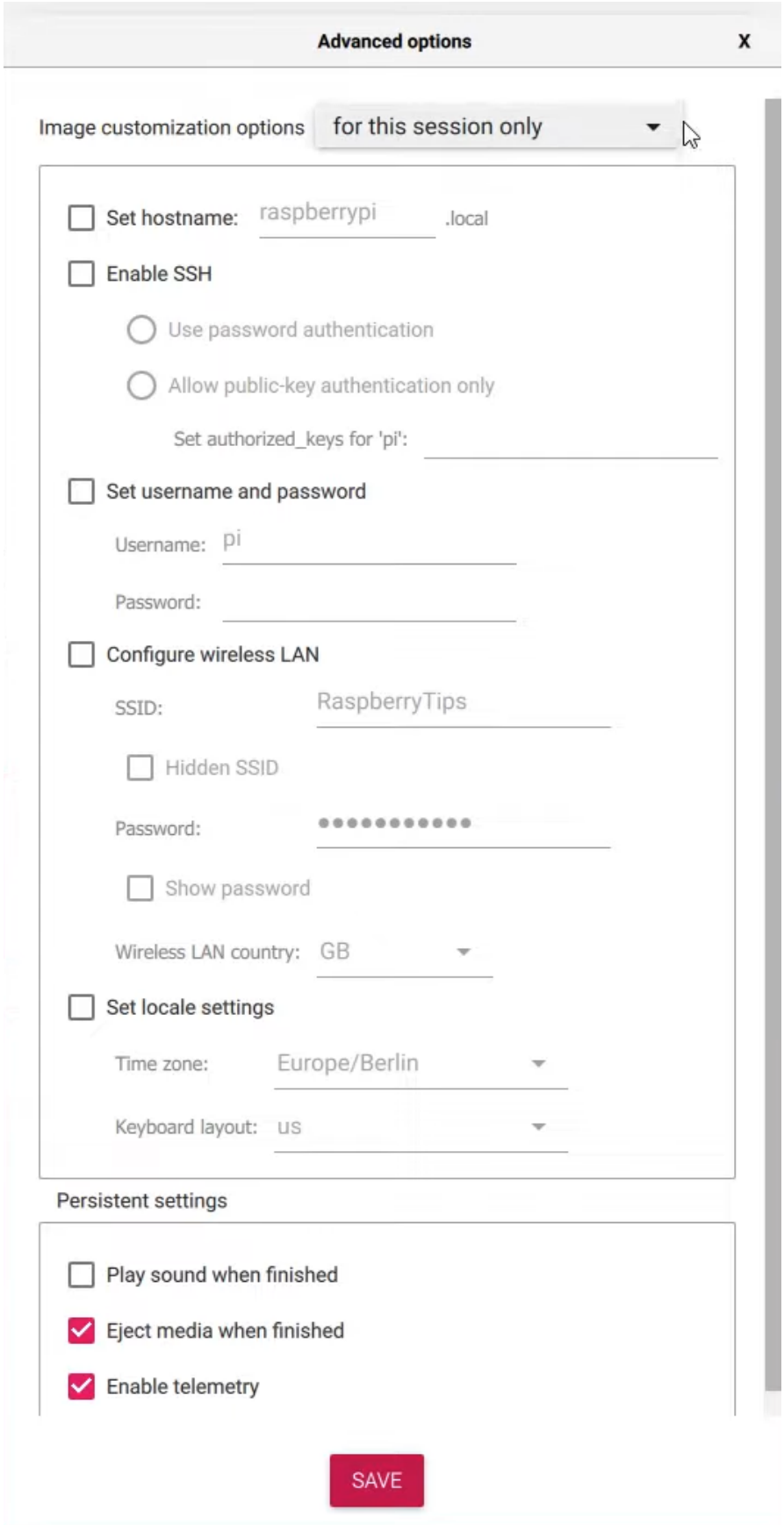


Figure 3.45 Advanced options for the configurations before setting up OS system

3.5 Mechanical

This section mainly talks about the mechanical part of this system which is the enclosure of Raspberry Pi Zero WH and Camera mounting design. Figure 3.46 shows the casing developed for storing Pi Zero WH inside it with different kinds of cover, we will be using the casing to enclose out Pi Zero WH.

Figure 3.46 Enclosure for Pi Zero WH

Source: Raspberry Pi Zero Official Case [133].

In order for the camera to be fixed below the vehicle of the station, we used the clip pattern camera mounting design as discussed in concept selection as shown in figure 3.47 for holding our camera and clip at the station as shown in figure 3.48.

Figure 3.47 Camera mounting design

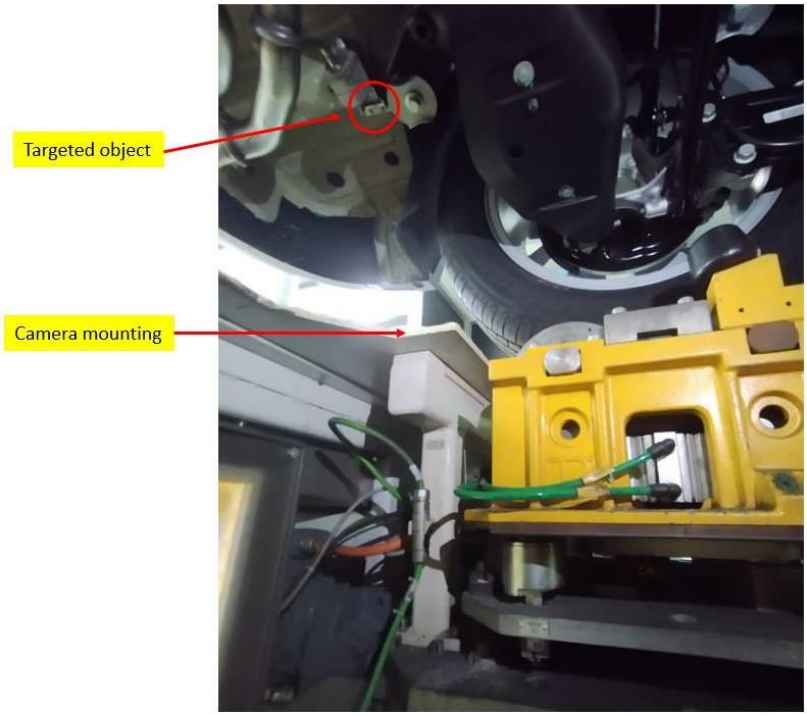


Figure 3.48 Implementation area at Wheel and Headlamp Alignment (WAHA) station

For the enclosure for Jetson Orin Nano, we will be design it on our own and 3D print it out. We designed the casing for the Jetson using SolidWorks as shown in figure 3.49.

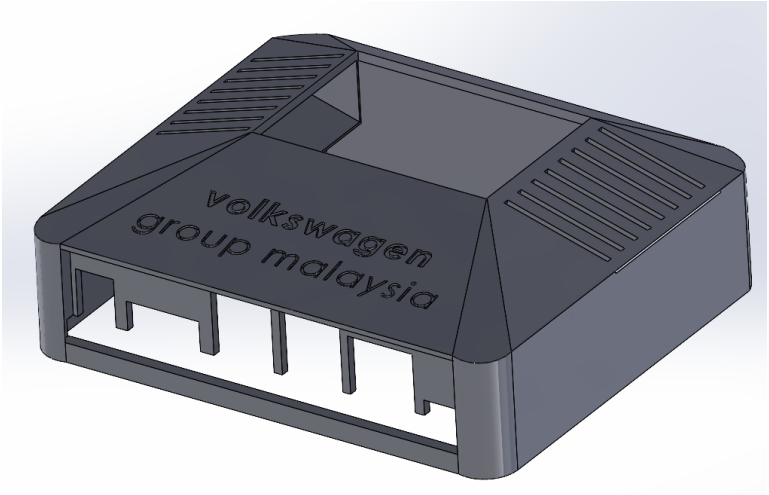


Figure 3.49 Casing for Jetson Orin Nano

3.6 Bill of Materials

Table 3.39 Bill of Materials List

No	Items	Unit	Price/Unit (RM)	Total (RM)
1	NVidia Jetson Orin Nano 8GB Dev Kit	1	2,799.00	2,799.00
2	128GB uSD with JetPack for Jetson Orin Nano	1	96.00	96.00
3	Hikvision 1080P HD Web Camera	2	100.00	200.00
4	Raspberry Pi Zero WH (Wireless with Header	2	80.00	160.00
5	Official Raspberry Pi Zero Case	2	25.00	25.00
6	RPi Approved MakerDisk C10 A1 uSD with RPi OS -32GB	2	44.00	88.00
7	Official RPi 12.5W PSU microB UK White	2	42.00	84.00
8	Mini HDMI Adapter	2	6.00	12.00
9	Official RPi micro USB adapter	2	6.00	6.00
Total				3,470

3.7 Overall System Diagram & Flowchart

This sections is the summarize of all the parts discussed previously into one system, we will be explain about the whole system process or how our system works and communicates between each other. We will explain it throught the blackbox, system flowchart and deep learning flowchart.

3.7.1 Blackbox

A blackbox refers to a conceptual or methodological framework in which the internal mechanisms are not openly understood or elucidated, instead emphasising the input and output aspects. The blackbox for the inspection system is depicted in Figure 3.2.

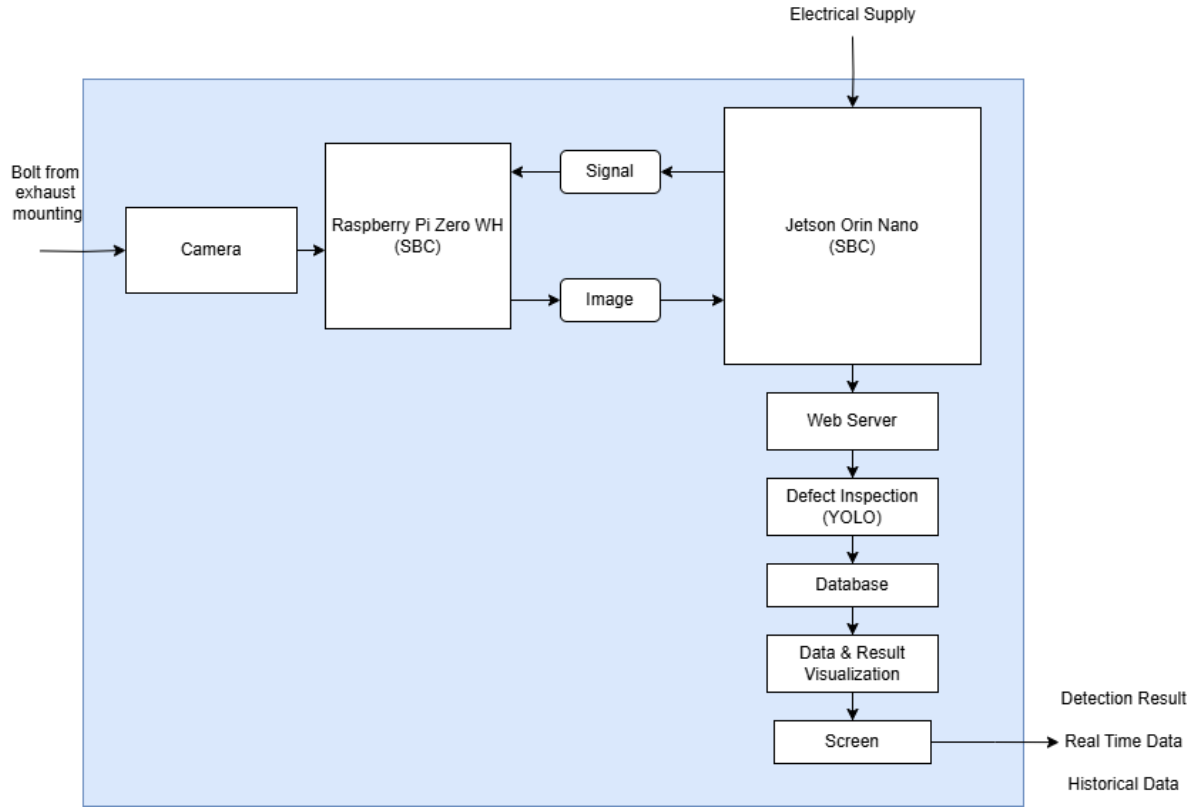


Figure 3.50: Blackbox for the whole system

Basically figure 3.50 shows the communication of our system, we can trigger a signal from Jetson Orin Nano to tell the Raspberry Pi Zero WH to take the picture of the bolt from exhaust mounting, then the Raspberry Pi Zero WH will send the image back to Jetson Orin Nano. After that, Jetson Orin Nano will take the picture and put it at the web server where the YOLO trained model weight exist to do defect inspection. After the defect inspection, we can insert the data to store the data collected from the defect inspection into our database and display the data and the result out in our data visualization which is Streamlit webpage we designed.

3.7.2 System Flowchart

Figure 3.51 below shows the system flowchart of our system. We should be undergoing system initialization at the beginning which is the meaning of turning on all the system which include the Jetson Orin Nano, Pi Zero WH and the Wi-Fi. After that, the button for capturing the image will be standing by waiting for be pressed and send a signal to Pi Zero WH. If the button of capturing the image is pressed, then it will send a signal to Pi Zero WH which tell it to take a picture. After that, the system will undergo a error handling process which means if

the image does not capture successfully then it will send a error message and go back to standy by mode of capture image button. Once the image has been captured successfully, it will send the image back to the Jetson and Jetson will pre-process the image into trained YOLO and activate Pytorch to run YOLO for the defect inspection. After that, the result detected will be send into Streamlit which is the GUI for data visualiztion then the detected result will be displayed there. Finally, we can insert the result into our database to store the data and plot bar chart with it for better analysis.

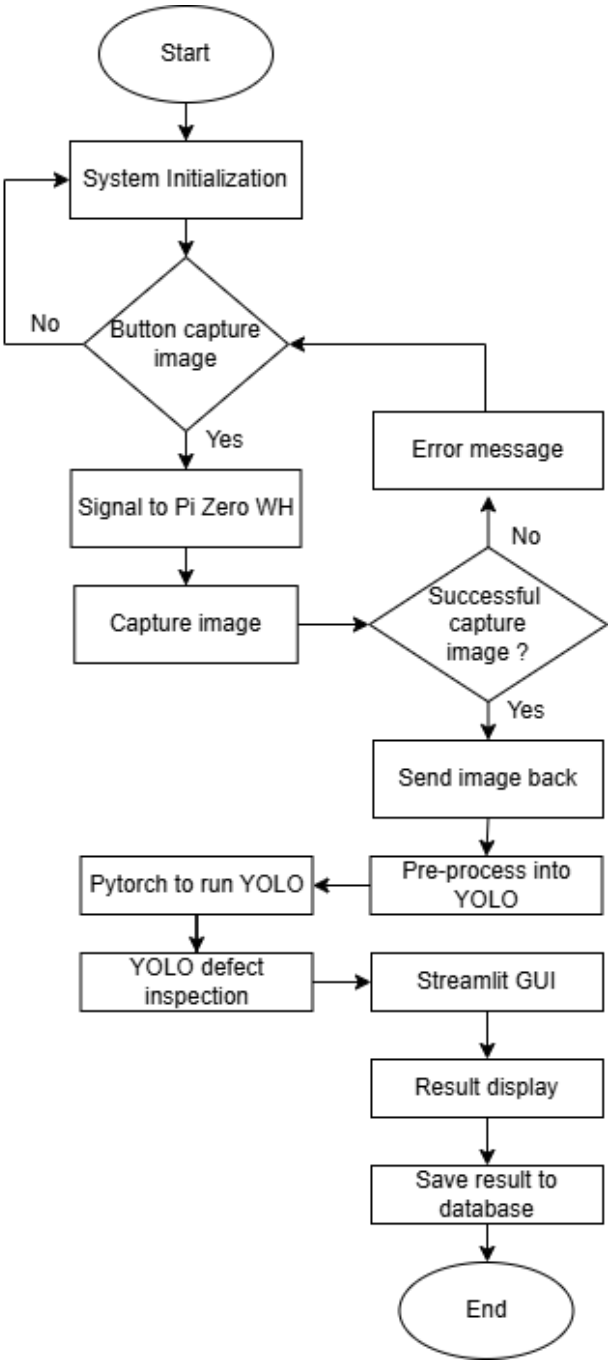


Figure 3.51 System flowchart of the system

3.7.3 Deep Learning Flowchart

This section involves the explanation of how we pre-process our data and do defect inspection with it. Based on figure 3.52, it provides a visual representation of the procedures and stages involved in an AI model—specifically, YOLO networks—based defect inspection system. The two primary portions of the flowchart are "Data Pre-processing" and "Defect Inspection." From data preparation to model training and defect inspection, including data distribution, augmentation, training, testing, validation and final inspection results, each section describes the procedures involved.

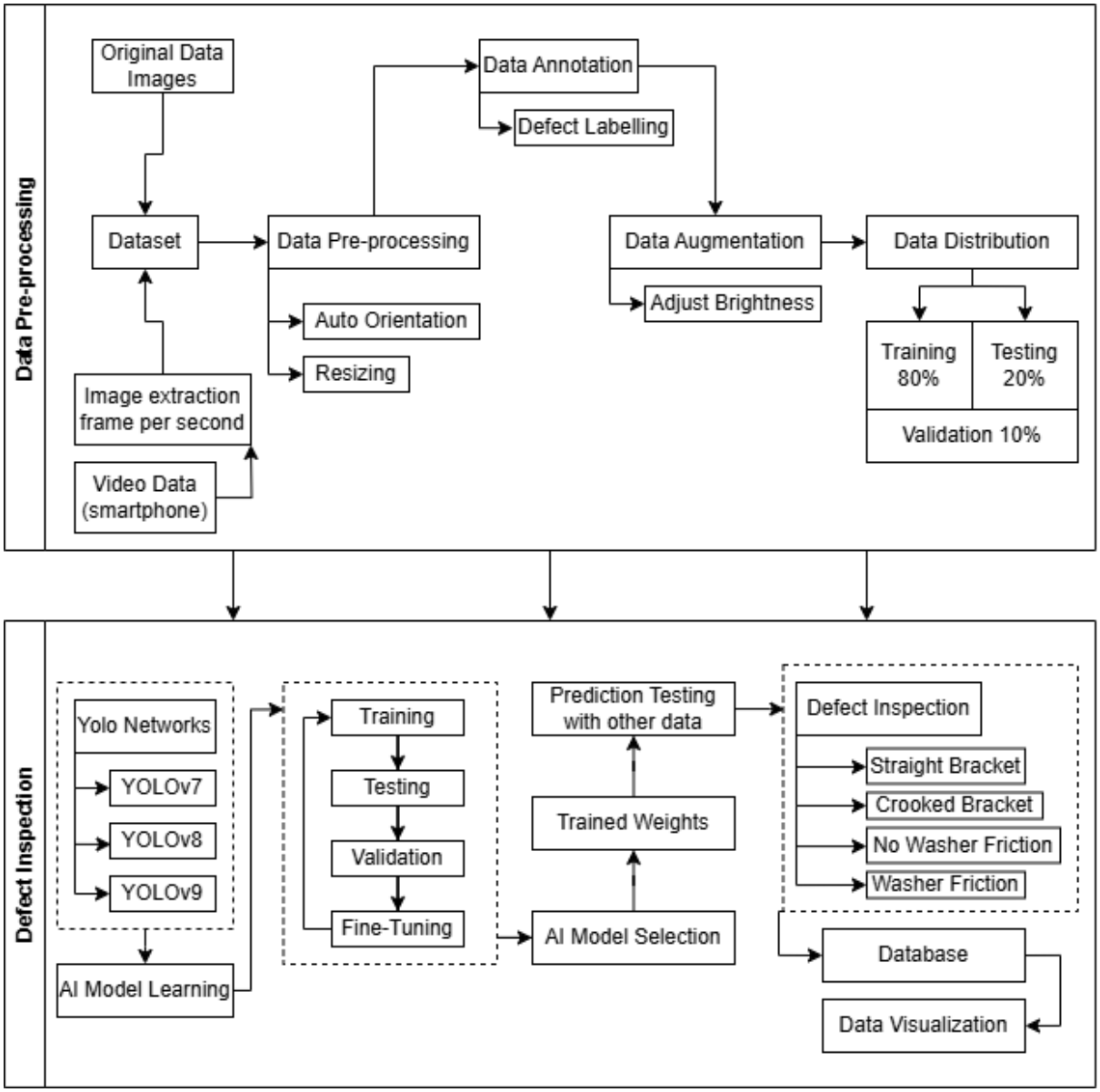


Figure 3.51 Deep learning flowchart

3.8 Summary

This section provided a concise overview of the approach, outlining the specific processes undertaken to carry out the research. Initially, a custom dataset was gathered and constructed for the specific classes of interest. Subsequently, an image pre-processing technique was implemented through picture annotation, segmentation and augmentation. Subsequently, the dataset was allocated into separate sets for training, validation and testing. In addition, the training configuration was examined and tailored to train the models. Finally, the performance evaluation approach including the comparison metric was explained to be conducted in chapter 4 result and discussion.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

Chapter four demonstrate all the results obtained from the system project developed, but we will explain about the metrics we used to evaluate the model performance results. In this study, we mainly focus on the newly developed YOLOv9 but we will also used YOLOv8 and YOLOv7 for comparison purposes. This section begins with the explanation of evaluation metrics, then the performance result obtained from these YOLO versions. We will also be looking at the confusion matrix, train/validation curve graphs and overall detection performance result of these YOLO versions. Furthermore, webpage developed using Streamlit and the database created will be shown in this section. At the end, we will analyze the system and identify the limitations and errors that we might have missed when developing this system.

4.1.1 Evaluation Metrics

Confusion Matrix

The confusion matrix is one of the most important tools for testing classification models. It shows a breakdown of the predictions made on a classification problem. There are numbers of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) in the matrix as shown in table 4.1. To give you an example, the confusion matrix clearly shows how many positive instances were correctly identified (TP), how many negative instances were correctly identified (TN), how many positive instances were wrongly identified as negative (FN) and how many negative instances were wrongly identified as positive (FP). This in-depth breakdown helps you understand not only how accurate the model is generally, but also what kinds of mistakes it is making. The result of confusion matrix for our model will be shown in the next few sections.

Table 4.1 Confusion matrix of real and predicted in binary classification

	Predicted Positive (P)	**Predicted Negative (N)**
Actual Positive (P)	True Positive (TP)	False Negative (FN)
Actual Negative (N)	False Positive (FP)	True Negative (TN)

Train/Validation Cure Graph

A vitally important visualization tool in the process of model training is the train/validation curve graph graph. Additionally, it depicts the performance metrics of the model, which are typically loss and accuracy, on both the training dataset and the validation dataset over the course of epochs. When it comes to diagnosing overfitting and underfitting, these curves are helpful. Typically, it includes:

- Training Loss: Shows the degree to which the model is learning from the training set.
- Validation Loss: Shows the model's effectiveness using unseen validation data.
- Training Accuracy: Demonstrates the evolution of the training set's accuracy over time.
- Validation Accuracy: Demonstrates the evolution of the validation set's correctness over time.

For example, overfitting happens when the training accuracy keeps getting better while the confirmation accuracy stays the same or gets worse. If, on the other hand, both the training and confirmation accuracies are low, the model may not be fitting well enough. You can change your model design, data preprocessing or training process to make it work better by looking at these curves. The result of train/validation graph will be shown in the next section.

Accuracy, Precision and Recall

A commonly used metric that shows the percentage of properly classified instances out of all instances is called accuracy. It is found by adding up all the true positives and false negatives and dividing that number by the total number of cases as shown in Eq 4.1. Even though accuracy is helpful, it can be wrong when datasets aren't balanced and the number of instances in each class changes a lot.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(Eq 4.1)}$$

Precision and recall are more detailed measures than accuracy, especially when datasets aren't balanced. Precision estimates how many true positive predictions there are out of all positive predictions as shown in Eq 4.2. This shows how good the positive predictions are. If the model is very precise, it means that it doesn't give many fake positives. Recall, on the other hand, counts how many true positive predictions there are out of all real positives as shown in

Eq 4.3. This shows how well the model can catch all the important cases. The model has a low rate of false negatives if it has a high recall.

$$Precision = \frac{TP}{TP + FP} \quad \text{(Eq 4.2)}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{(Eq 4.3)}$$

F1-Score

The F1 Score is the harmonic mean of accuracy and memory, giving us a single measure that is equal for both. This method works especially well when we need to find the best mix between accuracy and recall, especially when we can't pick one over the other. The F1 Score is only high if both accuracy and recall are high. This makes it a good way to test models that have to deal with datasets that aren't balanced. In order to achieve a compromise between precision and recall rates, the F1 score is employed to calculate the weighted average as shown in Eq 4.4. The algorithm considers both the rates of true positives and false negatives, with a focus on recognising objects that occur most frequently due to difficulties in accurately assessing their detection.

$$F_1 = \frac{Precision\ x\ Recall}{Precision + Recall} \quad \text{(Eq 4.2)}$$

or

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Mean Average Precision (mAP)

Mean average precision is a broad measure that is often used to judge how well object detection models work. It takes precision and recall into account to give a single number that shows how accurate the model is across different classes and levels of confidence. The first thing we have to understand is the Average Precision (AP) which is found for every class and shows the area under the Precision-Recall (PR) curve as shown in Eq 4.3. The PR curve is a graph that shows how accuracy changes with recall at different threshold levels.

$$AP = \int_0^1 Precision\,(Recall) d(Recall) \quad \text{(Eq 4.3)}$$

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

where R_n and R_{n-1} are recall values at thresholds n and $n-1$, P_n is the precision at threshold n.

The PR curve is made by figuring out precision and recall at different levels in order to find AP. The area under this slope is then the AP. To get a rough idea of this, numerical integration methods like the trapezoidal rule can be used. So, the average AP score for all classes is given by mAP as shown in Eq 4.4. It gives a single quality score for all types of objects in a collection.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad \text{(Eq 4.4)}$$

where N is the number of classes and AP_i is the Average Precision for class i.

4.2 Training Performance Results

Detection Performance

There will be three tables (table 4.2, table 4.3 and table 4.4) which demonstrate the results of training performance from each YOLO versions (YOLOv7, YOLOv8l and YOLOv9c), the evaluation metrics in the results includes precision value, recall value and mean average precision. Each of the table will also show the detection result of each classes which we can know the performance of the model towards the classes. Finally, the overall result of these three model will be tabulated in table 4.5.

YOLOv9c has the maximum precision (0.983), therefore its predictions for specific classes, such as straight brackets, are extremely accurate. Minimizing false positives is critical, and YOLOv9c excels at that. YOLOv7 has good precision (0.977), but it is somewhat less sensitive than YOLOv9c, which means it may produce more false positives. YOLOv8l has the lowest precision (0.959), which results in more false positives and less accurate predictions.

YOLOv9c achieves a recall rate of 0.961, indicating that it correctly identifies the majority of cases in each class. YOLOv8l has a strong recall (0.958), but significantly lower than YOLOv9c, implying more class misclassifications. YOLOv7 has the lowest recall (0.950), indicating that it may miss more cases in the sample.

In terms of mean Average Precision (mAP), YOLOv9c has the best score of 0.968, indicating a strong balance between precision and recall. YOLOv8l has a reasonable mAP (0.864), but it is much lower than YOLOv9c, implying a trade-off. YOLOv7 has the lowest mAP (0.947), indicating lower overall performance than the other models.

In conclusion, YOLOv9c surpasses YOLOv7 and YOLOv8l in precision, recall, and mAP, making it the optimal choice for class classification. YOLOv7 is an excellent choice for balancing performance and processing resources, whereas YOLOv8l's lower ratings indicate that it requires additional refinement.

Table 4.2 YOLOv7 Trained Results

Class	Images	Instance	P	R	mAP50	mAP50-95
All	120	237	0.977	0.950	0.973	0.947
Crooked bracket	120	92	0.951	0.989	0.980	0.980
No washer friction	120	94	0.998	0.979	0.998	0.955
Straight bracket	120	25	0.957	0.92	0.953	0.953
Washer friction	120	26	1	0.913	0.962	0.9

Table 4.3 YOLOv8l Trained Results

Class	Images	Instance	P	R	mAP50	mAP50-95
All	120	237	0.959	0.958	0.985	0.964
Crooked bracket	120	92	0.923	0.989	0.992	0.989
No washer friction	120	94	0.995	1	1	0.948
Straight bracket	120	25	0.92	0.88	0.88	0.973
Washer friction	120	26	0.998	0.962	0.962	0.944

Table 4.4 YOLOv9c Trained Results

Class	Images	Instance	P	R	mAP50	mAP50-95
All	120	237	0.983	0.961	0.99	0.968
Crooked bracket	120	92	0.945	1	0.992	0.963
No washer friction	120	94	0.998	1	0.995	0.963
Straight bracket	120	25	1	0.884	0.985	0.982
Washer friction	120	26	0.989	0.962	0.989	0.938

Table 4.5 Results Comparison of YOLOv7, v8 and v9

Model	**Precision Value**	**Recall Value**	**mAP50-95**
YOLOv7	0.977	0.950	0.947
YOLOv8l	0.959	0.958	0.964
YOLOv9c	0.983	0.961	0.968

Confusion Matrix

The confusion matrix as shown in figure 4.1 displays how well the YOLOv9 model performs in terms of correctly and wrongly categorizing distinct groups. The matrix is normalized, which means that the values are shown in proportion to the total number of forecasts for each class.

The values are represented in the confusion matrix using various hues of blue, where deeper shades correspond to higher values and lighter shades correspond to lower values. The utilisation of colour coding in this context offers a visual depiction of the model's performance, facilitating the identification of regions where the model demonstrates strong performance (represented by deeper colours) and regions that may necessitate enhancement (represented by lighter shades).

- Crooked Bracket: Crooked brackets are classified perfectly by the model with a precision of 1.00. However, the "no washer friction" class has a misclassification rate of 0.20, while the "straight bracket" class has a misclassification rate of 0.04. In addition, the "background" class has a 0.29 misclassification rate.

- No Washer Friction: With a precision of 1.00, the model likewise accurately defines the "no washer friction" class.

- Straight Bracket: The accuracy of the model's categorization of straight brackets is 0.80; however, the "washer friction" and "background" classes have misclassification rates of 0.57 and 0.96, respectively.

- Washer Friction: With a minor misclassification rate of 0.14 for the "background" class, the precision for the "washer friction" class is 0.96.

- Background: The background is accurately detected with an accuracy of 0.57 for the class "straight bracket" and a misclassification rate of 0.14 for the class "washer friction."

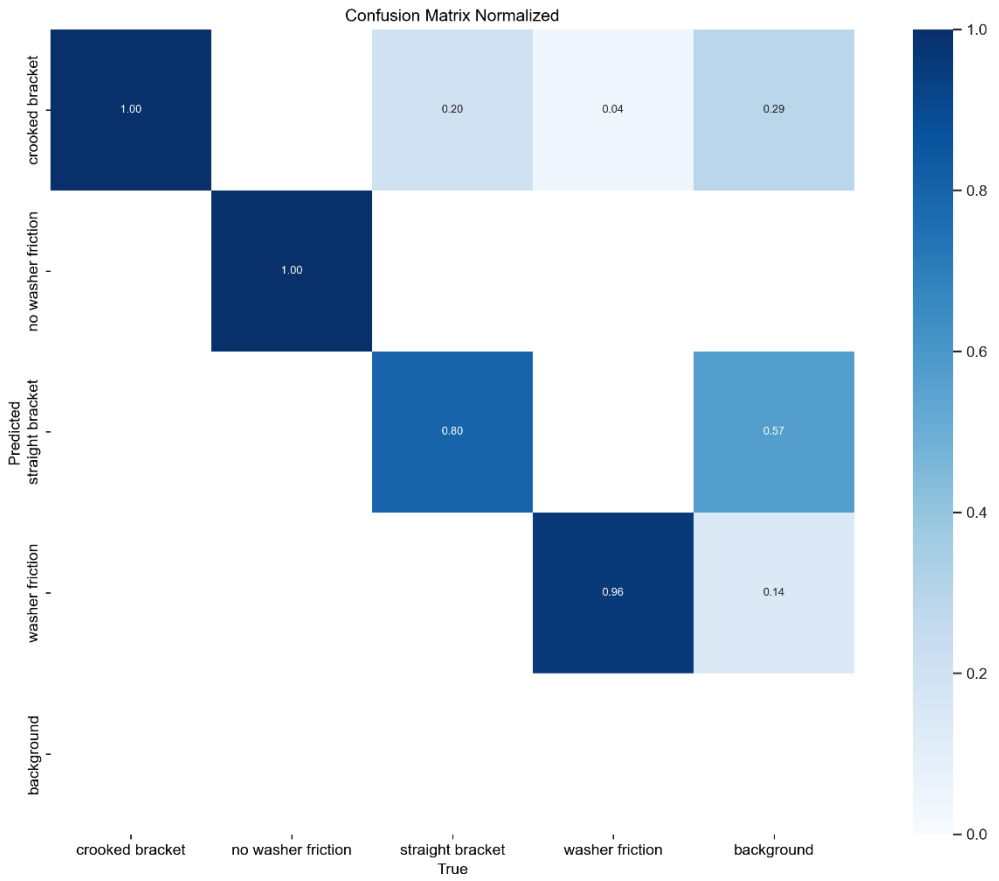


Figure 4.1 Normalized Confusion Matrix for YOLOv9c

Training and Validation Performance Metrics of YOlOv8

The ten different line graphs as shown in figure 4.2 are used to depict the results, each of which shows a different set of measures and losses related to the model's performance. Each graph's x-axes, which span from 0 to about 300, indicate an epoch or iteration that occurred during the training process.

i. Train/Box Loss: Begins at about 0.8 and exhibits a distinct downward trend, suggesting that the model gets better with time at predicting bounding boxes.

ii. Train/Cls Loss: Starts at 1.2 and falls off quickly, indicating that as the model gets better at classifying objects, the classification loss lowers.

iii. Train/Dfl Loss: Begins around 1.2 and exhibits a steep fall as well, suggesting that the differentiation loss of the model is getting smaller over time.

iv. Metrics/Precision(B): Starts at 0.7 and increases steadily from there, suggesting that the model's accuracy is getting better while it is being trained.

v. Metrics/Recall(B): As with precision, this measure begins at 0.7 and rises, suggesting that the model's recall is likewise getting better.

vi. Val/Box Loss: Starts at 0.7 and drops off quickly, much like the training box loss, suggesting better validation performance as well.

vii. Val/Cls Loss: Starts at a number that is close to an outlier, about 1.2, and then sharply decreases to about 0.2, suggesting a substantial decrease in the validation classification loss.

viii. Val/Dfl Loss: Identical to the training differentiation loss, it begins somewhat above 1 and falls off quickly, showing steady model improvement.

ix. Metrics/mAP50(B): Starts at 0.6 and rises gradually, suggesting that the model's mean Average Precision has improved at IoU=0.5.

x. Metrics/mAP50-95(B): Shows an improvement in the model's mean Average Precision over various IoU thresholds (0.5 to 0.95), starting at approximately 0.6 and increasing.

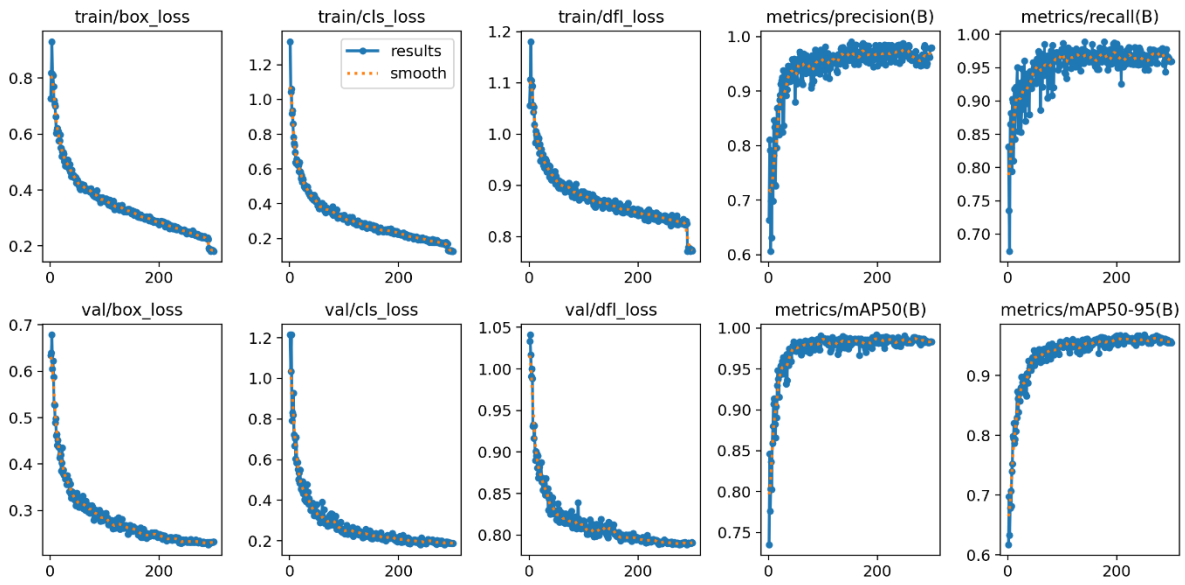


Figure 4.2 Performance metrices of Train and Validation

The findings are visually shown using eight separate line graphs, each representing various measures and losses linked to the performance of the algorithm. In the training phase, the x-axis of all graphs correspond to epochs or iterations, which span from 0 to 50.

In general, the visualisations shown indicate that as the training process advances, there is a drop in both training and validation losses, accompanied by improvements in precision and recall metrics. This demonstrates the YOLOv8 model's successful acquisition of knowledge across consecutive iterations. The use of dotted lines in the graphs serves to enhance the visual representation of the overall trend by smoothing the values.

4.2.1 Detection Results

Based on the detection results from figure 4.3 it shows the ability of the model to detect each kinds of class. For an example, the model is able to detect the bracket is a straight bracket while the washer has no friction with confidence level of 0.97 (97%) and 0.92 (92%) respectively. The result shown on the detected images indicate how confidence the model is to identify the part of the image as part of the class, like the model detects crooked bracket but with how much confidence does the model confirm that the bracket is crooked or has washer friction. Once the model detected a class, it will show how much it confidence that the part of the image is from the class.

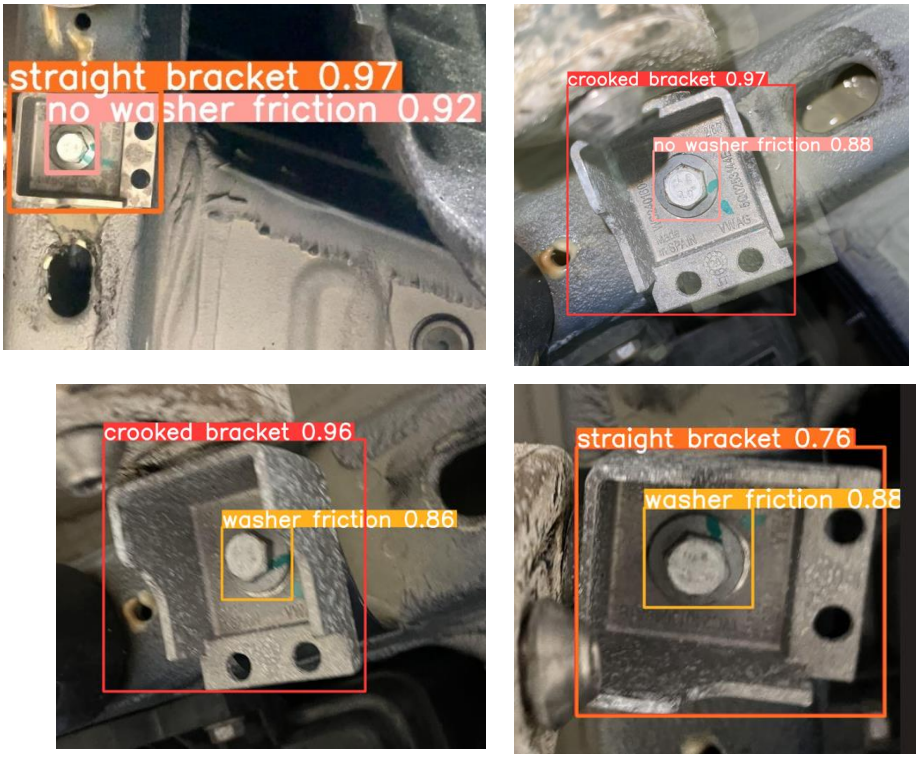


Figure 4.3 Detection Results of all the possible classes using YOLOv9c

4.3 UI Design

Streamlit's user interface design for the automotive underbody inspection system provides a simple and interactive platform for detecting and analyzing anomalies. The integration of MySQL data into Streamlit allows for more efficient data management and retrieval inside the user interface. Furthermore, Streamlit may combine real-time data from the camera while also showing pre-processed data and results. Users can view a live feed of the car's underbody during the inspection process, allowing for real-time visualisation.

The result of Streamlit webpage has been shown in figures according to figure 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10. The homepage (figure 4.4) is the main page that discussed and explained the purpose and information of this system. The data collection (figure 4.5) webpage is where we displayed all the data collected from the detection with the help of bar chart based on months to provide a better visualization for the company to view the conditions of all the vehicles.

The bulk image object detection webpage comes in handy when operator has a lot of data image that needs to be detected in an instance, all the operator has to do is to drag all the images there and pressed the detect image button then the results will be displayed and the right

side with the an addition of download button where we can download all the detected images in a zip file.

The video object detection is where the operator can upload the video recorded of the data footage and begin the detection process. The camera capture object detection is where real-time defect inspection process start with a push of a button to detect image after we toggle the capture image switch, the webpage will send a signal to the Pi Zero WH to take a picture through the camera and send the image capture back to the webpage to do defect inspection.

Finally, the live object detection is where we can turn on our webcam or maybe wireless streaming camera in the future to do our real-time streaming defect inspection. All the webpage consists of sidebar with required parameters need to input before begin defect inspection, we can also query the data at the side bar by insert the parameters according to the vehicle characteristics and also insert data to store in the database with a press of a submit button.

Homepage

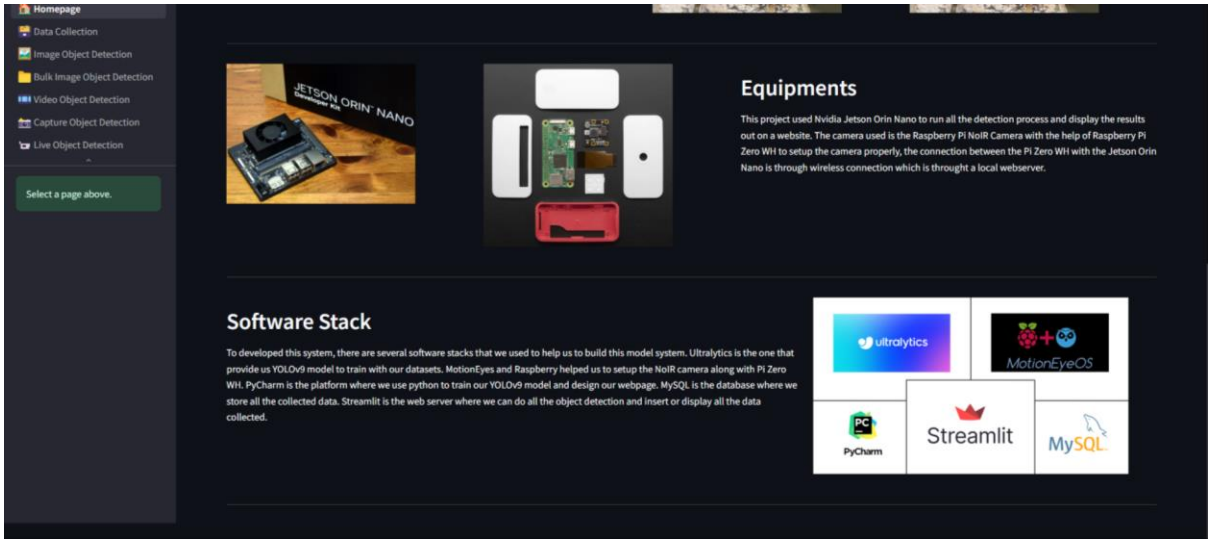
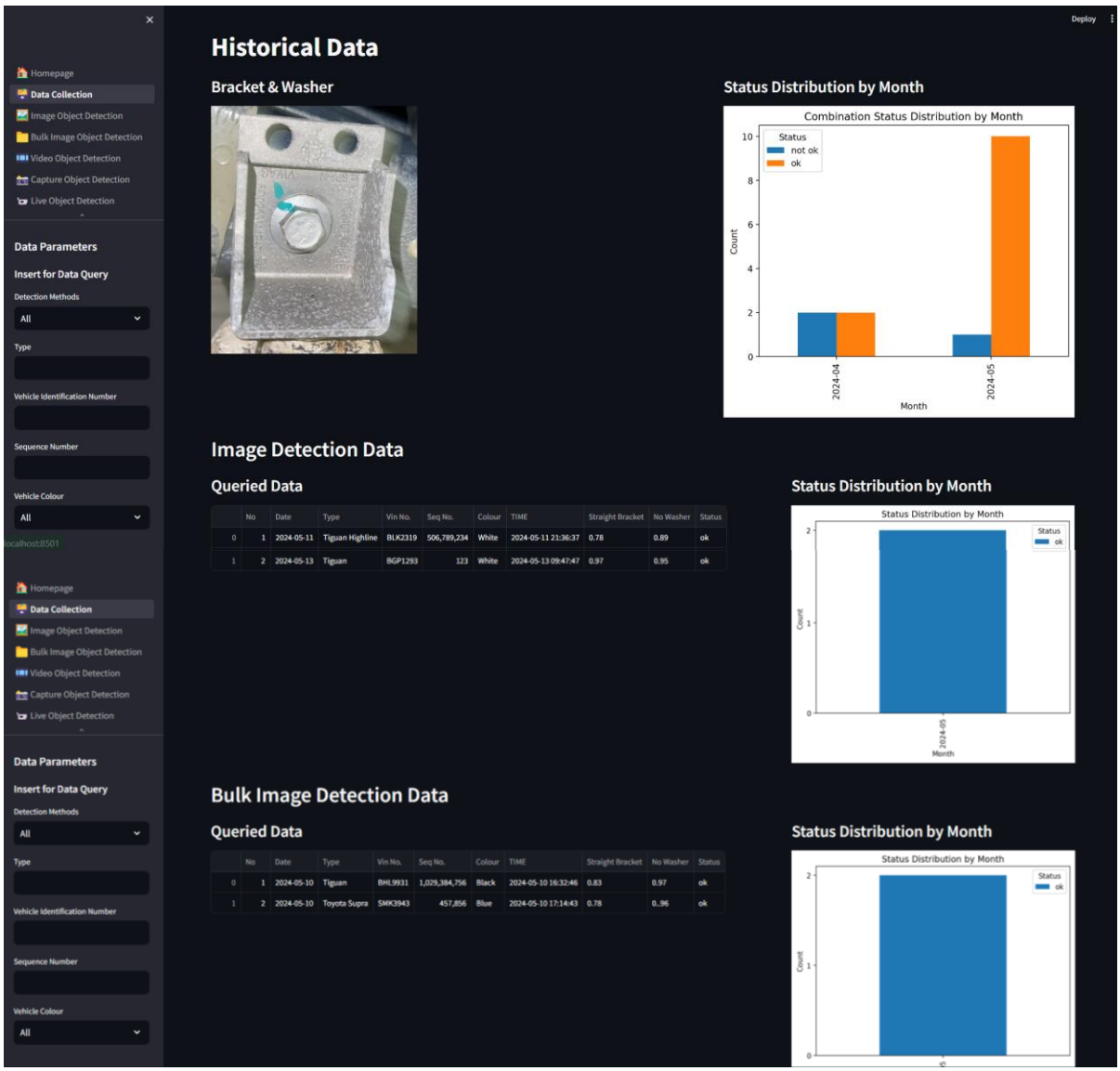


Figure 4.4 Streamlit webpage for homepage

Data Collection



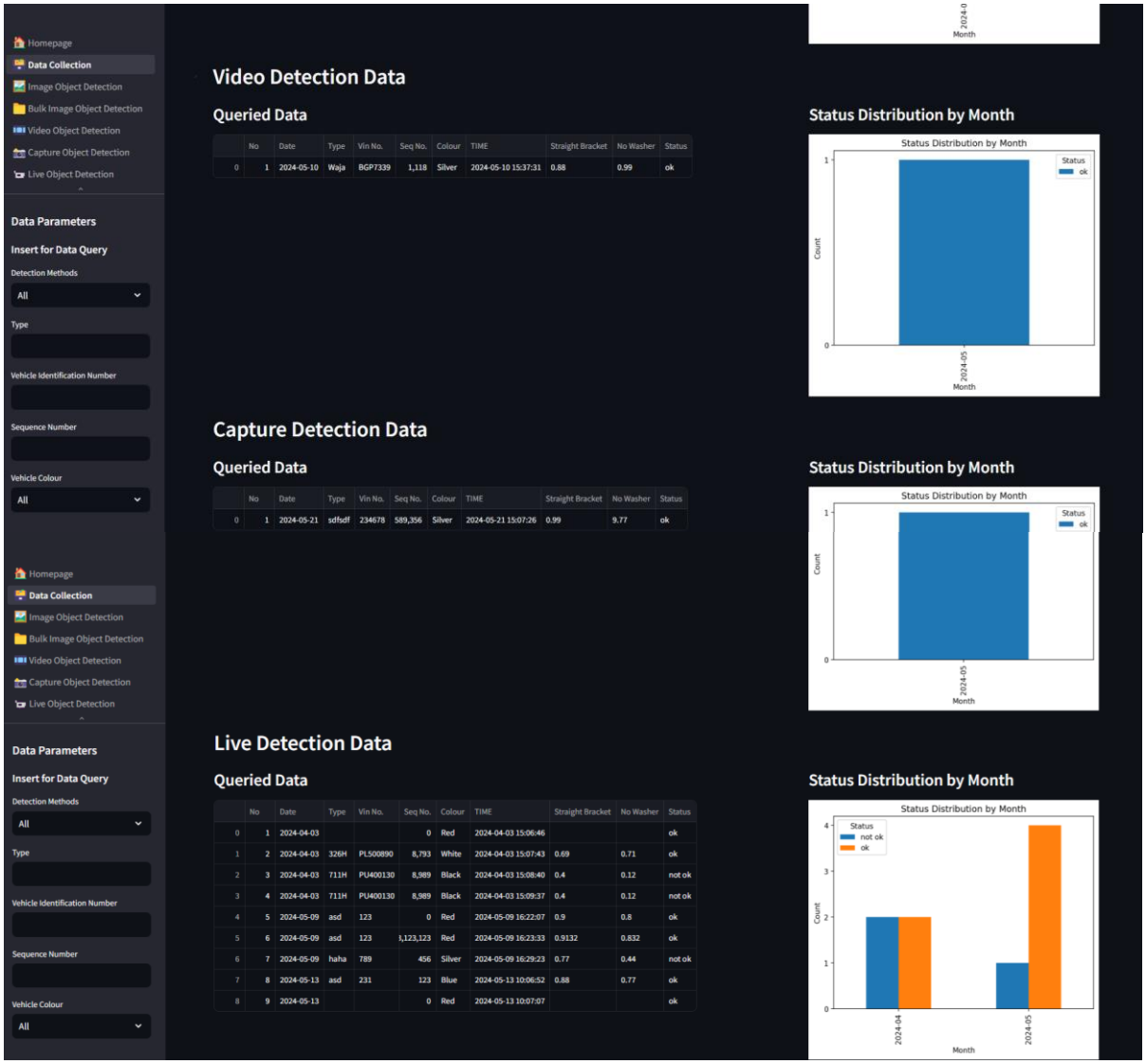


Figure 4.5 Streamlit webpage for data collection

Image Object Detection

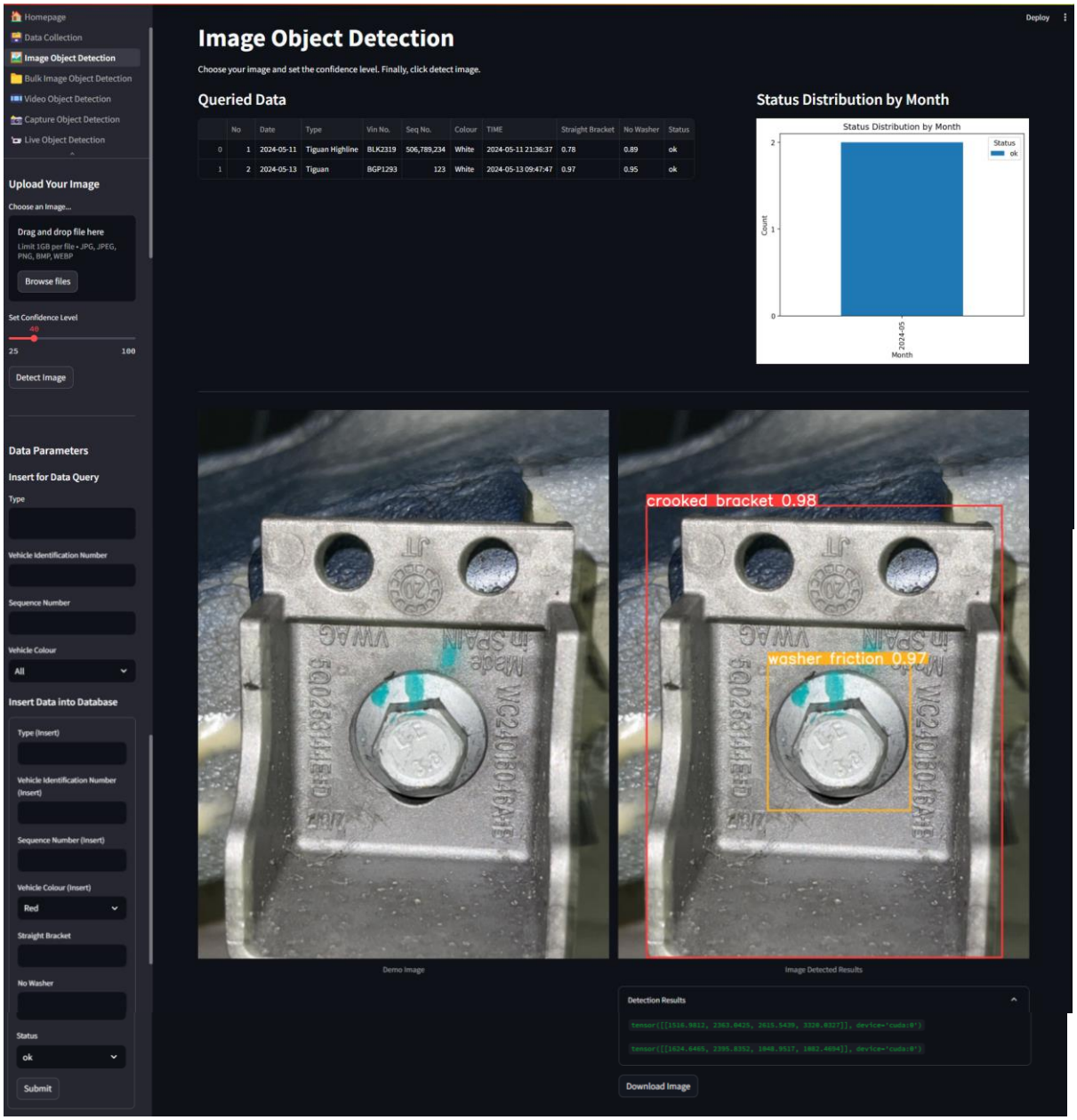


Figure 4.6 Streamlit webpage for image object detection

Bulk Image Object Detection

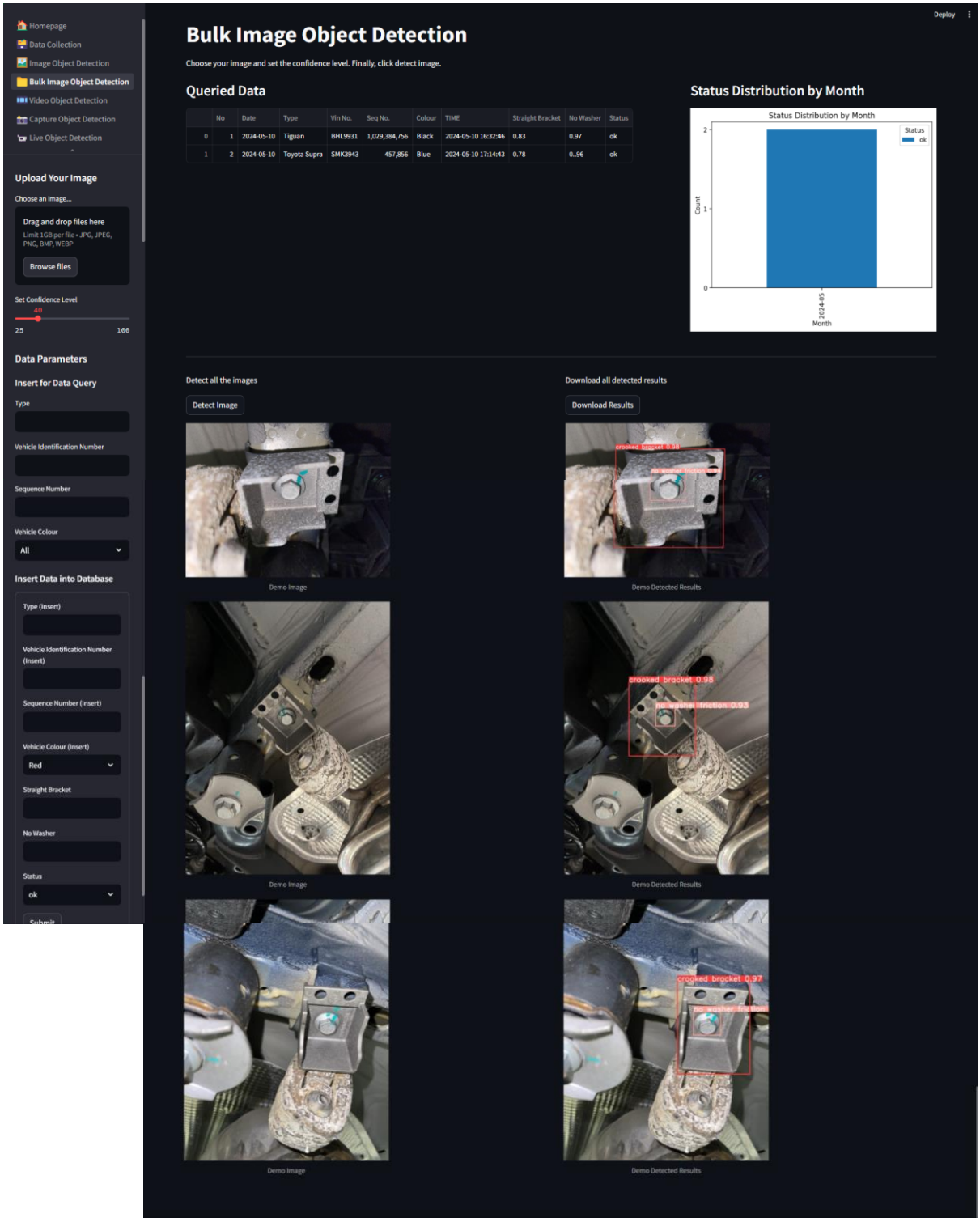


Figure 4.7 Streamlit webpage for bulk image object detection

Video Object Detection

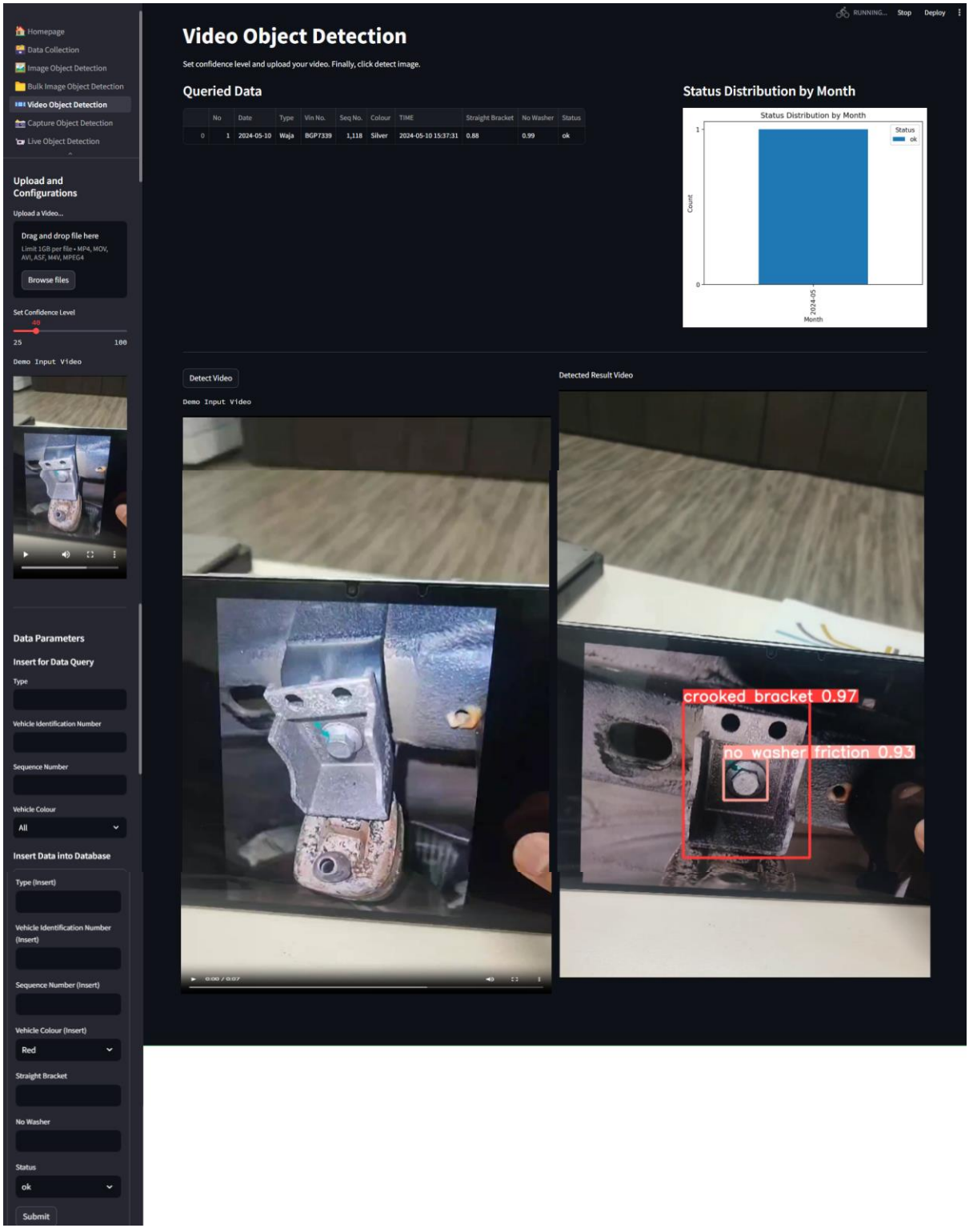
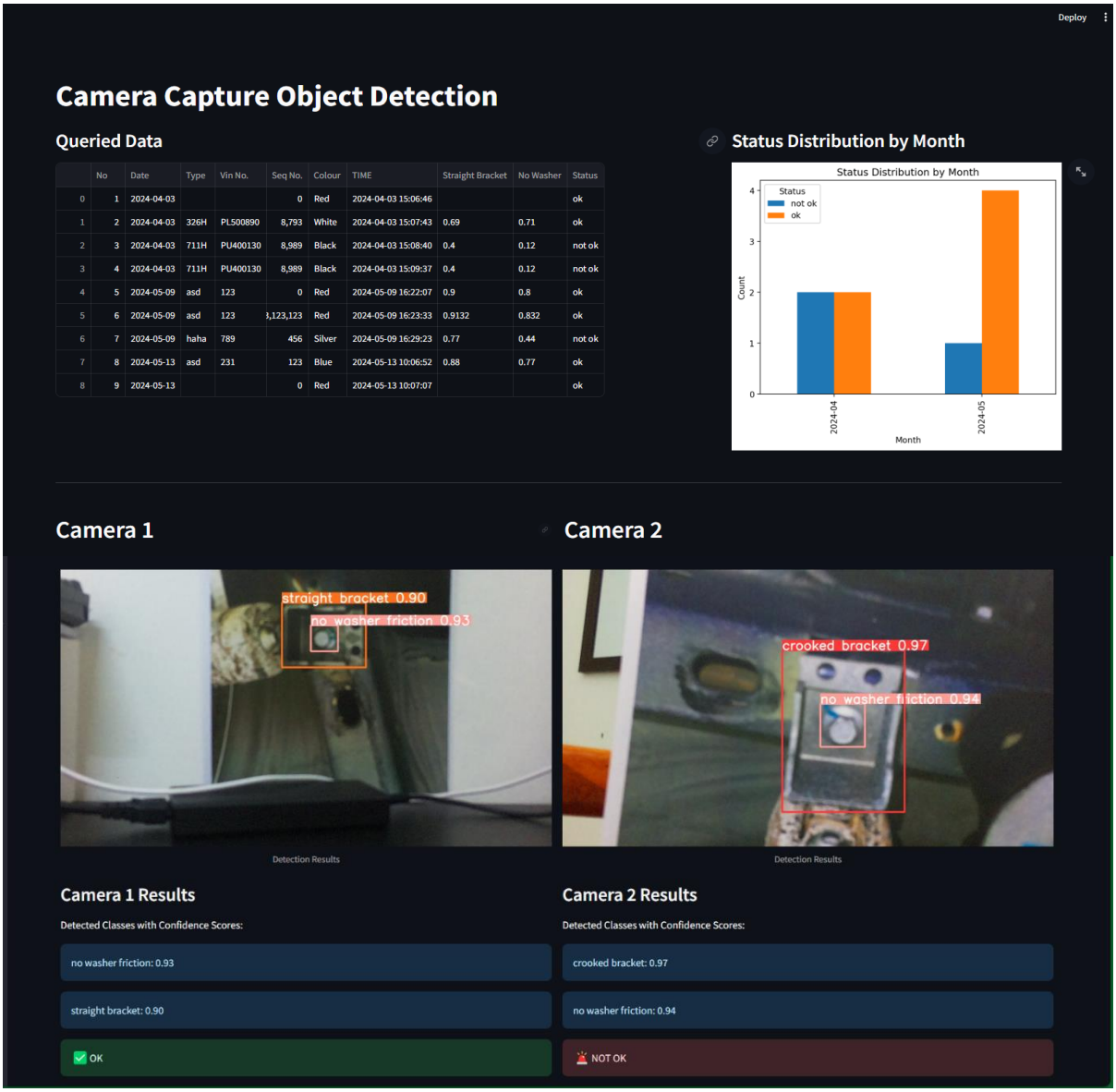


Figure 4.8 Streamlit webpage for Video Object Detection

Capture Object Detection



	No	Date	Type	Vin No.	Seq No.	Colour	TIME	Straight Bracket	No Washer	Status
0	1	2024-04-03			0	Red	2024-04-03 15:06:46			ok
1	2	2024-04-03	326H	PL500890	8,793	White	2024-04-03 15:07:43	0.69	0.71	ok
2	3	2024-04-03	711H	PU400130	8,989	Black	2024-04-03 15:08:40	0.4	0.12	not ok
3	4	2024-04-03	711H	PU400130	8,989	Black	2024-04-03 15:09:37	0.4	0.12	not ok
4	5	2024-05-09	asd	123	0	Red	2024-05-09 16:22:07	0.9	0.8	ok
5	6	2024-05-09	asd	123	3,123,123	Red	2024-05-09 16:23:33	0.9132	0.832	ok
6	7	2024-05-09	haha	789	456	Silver	2024-05-09 16:29:23	0.77	0.44	not ok
7	8	2024-05-13	asd	231	123	Blue	2024-05-13 10:06:52	0.88	0.77	ok
8	9	2024-05-13			0	Red	2024-05-13 10:07:07			ok

Figure 4.9 Streamlit webpage for camera capture object detection

Live Object Detection

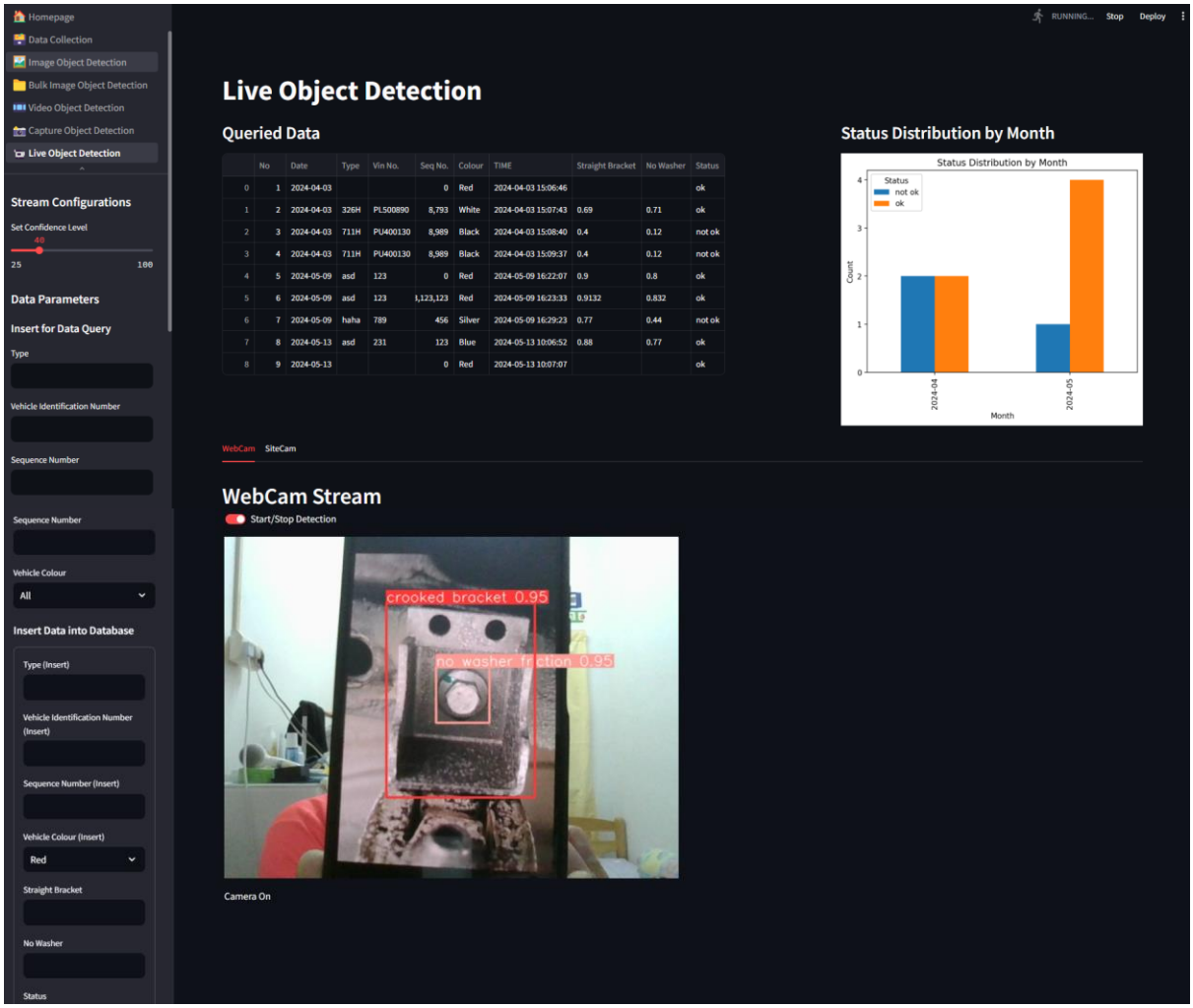


Figure 4.10 Streamlit webpage for live object detection

4.4 MySQL Database

The created data storage in the database consists five data table (figure 4.11, 4.12, 4.13, 4.14 and 4.15) which are "data_image", "data_bulk_image", "data_video", "data_capture" and "data_live". Each of these table has successfully received and stored data, all the data can be even being sent to the webpage for displayed and analysis purposes.

The detailed information regarding the status of the vehicle are explained as follows:

i. Number (No.): Identifier for each entry.

ii. Date: Date of the data recorded.

iii. Type: The type of vehicle being inspected.

iv. Vehicle Number: VIN entered by the user.

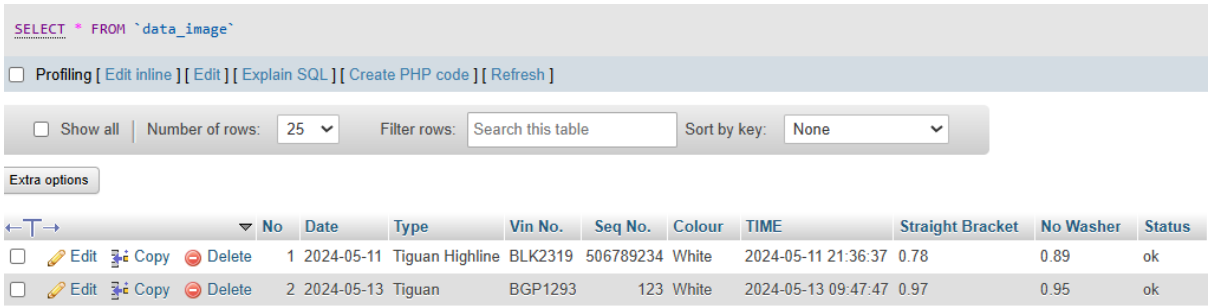
v. Sequence Number: Inspection sequence tracking.

vi. Colour Code: Vehicle colour code

vii. Time: Time when data is recorded

viii. Bracket & Washer: Value for describe vehicle condition, with values ranging from 0 (loss bolt) to 1 (tight bolt). It is crucial for assessing the vehicle's underbody parts' condition.

ix. Status: Determines if the car is "OK" or "Not OK" by summarizing its overall condition based on the Bracket and Washer values.



SELECT * FROM `data_image`

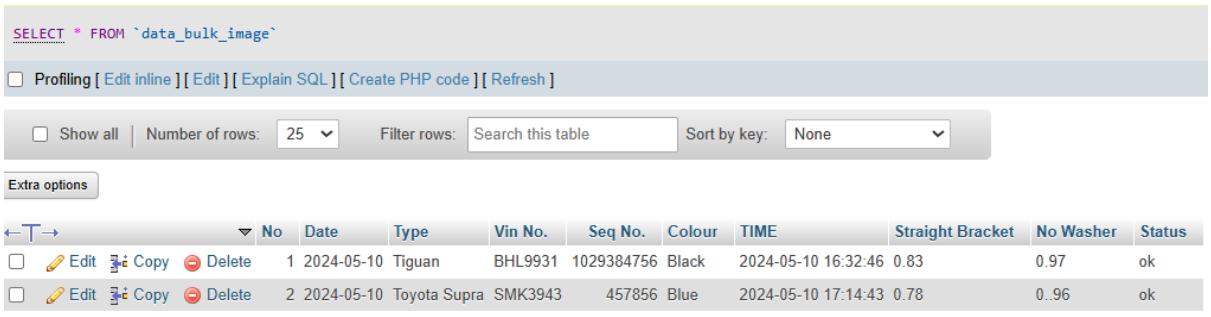
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	No	Date	Type	Vin No.	Seq No.	Colour	TIME	Straight Bracket	No Washer	Status
Edit Copy Delete	1	2024-05-11	Tiguan Highline	BLK2319	506789234	White	2024-05-11 21:36:37	0.78	0.89	ok
Edit Copy Delete	2	2024-05-13	Tiguan	BGP1293	123	White	2024-05-13 09:47:47	0.97	0.95	ok

Figure 4.11 Data storage for “data_image”



SELECT * FROM `data_bulk_image`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	No	Date	Type	Vin No.	Seq No.	Colour	TIME	Straight Bracket	No Washer	Status
Edit Copy Delete	1	2024-05-10	Tiguan	BHL9931	1029384756	Black	2024-05-10 16:32:46	0.83	0.97	ok
Edit Copy Delete	2	2024-05-10	Toyota Supra	SMK3943	457856	Blue	2024-05-10 17:14:43	0.78	0..96	ok

Figure 4.12 Data storage for “data_bulk_image”

SELECT * FROM `data_video`

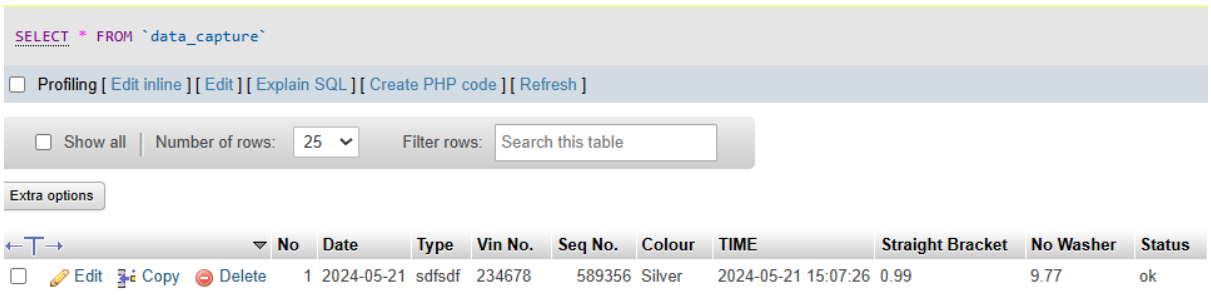
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	No	Date	Type	Vin No.	Seq No.	Colour	TIME	Straight Bracket	No Washer	Status
Edit Copy Delete	1	2024-05-10	Waja	BGP7339	1118	Silver	2024-05-10 15:37:31	0.88	0.99	ok

Figure 4.13 Data storage “data_video”



SELECT * FROM `data_capture`

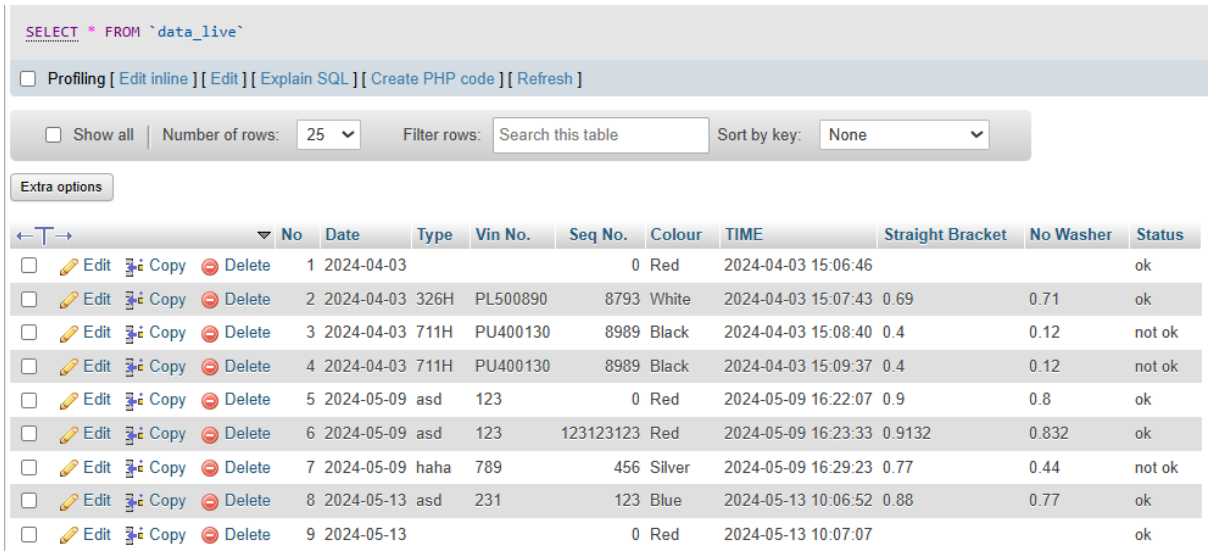
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	No	Date	Type	Vin No.	Seq No.	Colour	TIME	Straight Bracket	No Washer	Status
Edit Copy Delete	1	2024-05-21	sdfsdf	234678	589356	Silver	2024-05-21 15:07:26	0.99	9.77	ok

Figure 4.14 Data storage “data_capture”



SELECT * FROM `data_live`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	No	Date	Type	Vin No.	Seq No.	Colour	TIME	Straight Bracket	No Washer	Status
Edit Copy Delete	1	2024-04-03			0	Red	2024-04-03 15:06:46			ok
Edit Copy Delete	2	2024-04-03	326H	PL500890	8793	White	2024-04-03 15:07:43	0.69	0.71	ok
Edit Copy Delete	3	2024-04-03	711H	PU400130	8989	Black	2024-04-03 15:08:40	0.4	0.12	not ok
Edit Copy Delete	4	2024-04-03	711H	PU400130	8989	Black	2024-04-03 15:09:37	0.4	0.12	not ok
Edit Copy Delete	5	2024-05-09	asd	123	0	Red	2024-05-09 16:22:07	0.9	0.8	ok
Edit Copy Delete	6	2024-05-09	asd	123	123123123	Red	2024-05-09 16:23:33	0.9132	0.832	ok
Edit Copy Delete	7	2024-05-09	haha	789	456	Silver	2024-05-09 16:29:23	0.77	0.44	not ok
Edit Copy Delete	8	2024-05-13	asd	231	123	Blue	2024-05-13 10:06:52	0.88	0.77	ok
Edit Copy Delete	9	2024-05-13			0	Red	2024-05-13 10:07:07			ok

Figure 4.15 Data storage “data_live”

4.5 System Setup

Figure 4.16 shows the system setup for the inspection process where we can setup our Jetson Orin Nano at the desired place we want to do defect inspection while the camera will be fixed

under the vehicle to capture the image under the body of the vehicle. A demo system setup is shown in figure 4.17 where at the right table is the Jetson Orin Nano while the left table is the Hikvision web camera mounted on the table connecting with Raspberry Pi Zero WH.

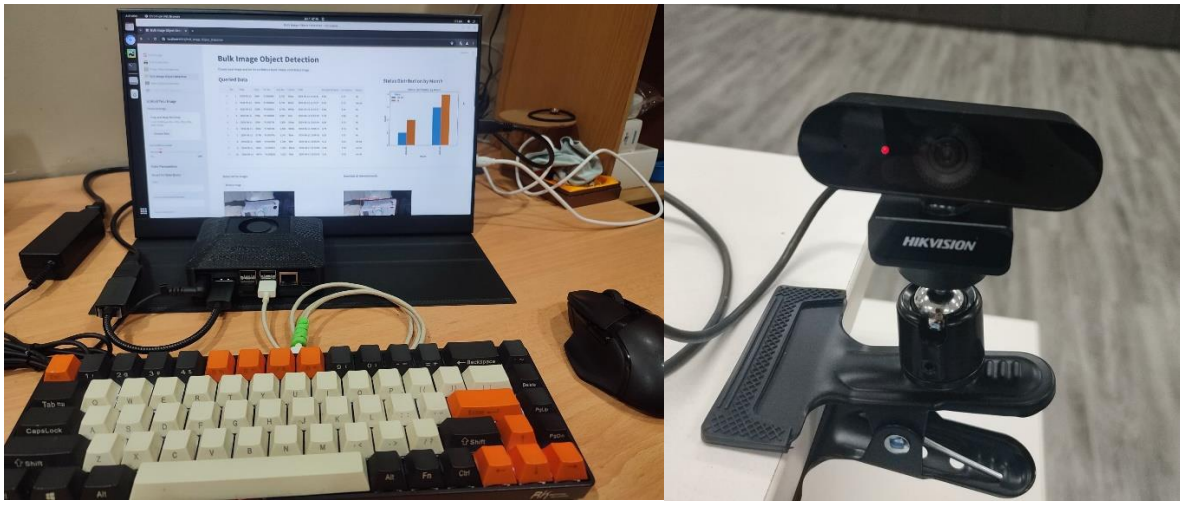


Figure 4.16 Defect inspection system setup

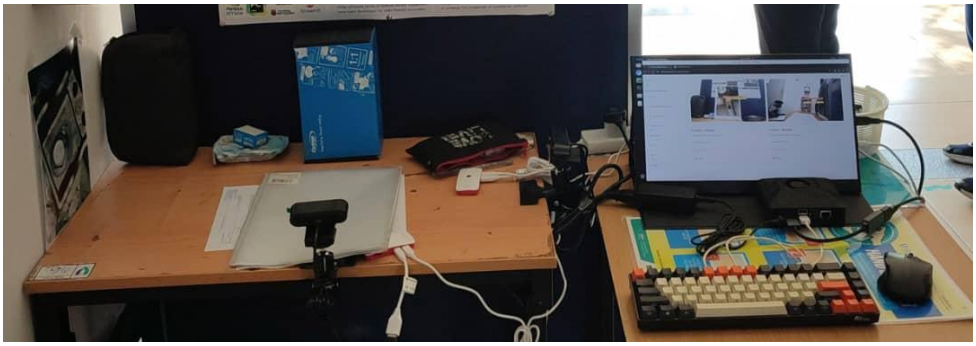


Figure 4.17 Demo system setup

4.6 Error Analysis

In this section, we will be talking about the error that we are having in this defect inspection system. Currently, there are two major errors that we come across which as follows:

i. False Positives: Situation when part of the image is identified as a defect by the model when there is nothing to be detect as defect. This could be due to the poor lightning conditions, objects resembling defect characteristics or poor camera quality.

ii. False Negative: Situation when the object with actual defect in the image failed to be detect as defect. These errors mostly occur when there are subtle defects that compromise into the background or the camera failed to focus on the object.

Cause of Errors

After analysing and researching the root causes of these errors, we found out there are three cause of errors that direct leads to the failure of defect detection.

i. Insufficient amount of training dataset: Special kind of defect with lack of proper dataset amount to be train for the model to recognize the defect might led the model to face the challenges in model generalization.

ii. Highly complex background and poor lightning conditions: The fact that the view from the exhaust mounting under the vehicle has a very complex background and low lightning quality causing the model unable to focus and dissect the image properly to detect the defects.

iii. Similarity of defect and non-defect characteristics: Sometimes the characteristic of loose bolt resembled the characteristic of tight bolt that will cause confusion to the model and lead to misclassifications.

Implications of Errors

If the errors persists with no further improvement and corrections, we predicted that there are some major consequences that might happen.

i. False Positives will eventually lead to unnecessary further inspection or intervention which will potentially cause the increment of maintenance cost, lower vehicle production rate, longer inspection time and lower company profit.

ii. False Negatives might lead to higher potential of significant risk like vehicle damage or road accidents due to actual defect exhaust mounting does not being detected.

Improvement Strategies

After the discussion and analysis of the errors, there are plans or suggestions that might be useful to solve the errors or to minimize the occurrence of the errors as much as possible.

i. Enhancement of dataset range: Analyse the characteristics of the loose bolt on exhaust mounting with special defect type then take more data images specific similar to that

kind of defect type so that the model can learn more and able to identify special defect type of loose bolt.

ii. Image pre-processing techniques: Implement different kinds of image pre-processing techniques like background subtraction, image segmentation, colour space conversion, etc. to rectify image with sophisticated background for a better object detection by the model.

iii. Modification of model architecture: Fine-tuning of YOLOv9 architectures and create sensitivity thresholds for loose bolt with similar characteristics as tight bolt.

iv. Camera upgrade: Upgrading of camera with a higher resolution, higher frame per second and processing power to capture image with a higher clearance and sharping colour for a better detection accuracy.

4.7 Limitations

One of the largest restrictions in this study is the limited specifications level of the laptop used for training process. The fear of damaging the laptop due to overheat leads to inefficient training process by reducing training epochs and lower fine-tuning. Furthermore, limited power specifications like GPU leads to out of memory for a higher performance model like YOLOv9e or YOLOv8x when training starts. Other than that, the time taken during each training session is very long with approximately 6 hours for 300 epochs, not to mention that it took 16 hours for training with YOLOv7 in 300 epochs.

Secondly, the lack of processing power, speed and RAM of Pi Zero WH limits the streaming features for real-time defect inspection through live footage due to lower frame per second. Furthermore, it limits the capturing speed of the camera when signal is received from the webpage causing slower image received by the webpage to undergo defect inspection.

Thirdly, camera specifications are limited which leads to certain limitations of image quality when image captured for defect inspection. The lack of focus from the camera causes the model unable to detect the object properly which is they misclassification occurred. Furthermore, the angle and distance from the exhaust mounting to the camera are limited that leads to certain area of the exhaust mounting not being capture by the camera.

4.8 Summary

Based on the results shown previously, we can see that the model is detecting well and showing a very good result on the UI created by Streamlit. Furthermore, it is also capable of storing the data into the database of MySQL. Unpredicted errors that occurred in the system have been analyse and discuss to come out with a proper solutions to rectify the mistake that might be made during the development of the system. Further development will be made to create a product which we can mount the camera and the system below the production line of vehicle to do inspection.

CHAPTER 5

CONCLUSION

5.1 Introduction

This section provides a concise and detailed summary and analysis of the achievement of each target. Automating automobile underbody inspections is a big step toward increasing road safety and vehicle reliability in addition to being a technological advancement in vehicle maintenance. This study deployed a comprehensive data management system and systematically developed, trained, and evaluated several iterations of the You Only Look Once (YOLO) algorithm, setting a milestone in the field of automated vehicle inspections.

The framework for a thorough examination of the theoretical and practical implications of our findings is established in this introduction. It identifies opportunities for further investigation and highlights the contributions of the study. The project's completion is not considered as its conclusion, but rather as a step toward continued breakthroughs in vehicle inspection technology.

This study provided insightful information about the accuracy and efficiency that may be attained with data management and deep learning models. The significance of these results is emphasized in this chapter, along with potential avenues for future investigation in this ever-evolving field.

5.1 Practical Effect

Vehicle Underbody Inspection Efficiency Enhancement

An important practical advantage of this discovery is a considerable increase in inspection efficiency. The time needed for each inspection can be significantly decreased by analyzing car underbodies quickly and precisely with YOLO-based deep learning algorithms for defect identification. Throughput rates at maintenance facilities and inspection centers may change as a result of this enhanced efficiency, which speeds up the inspection procedure and allows for the inspection of more cars in the same amount of time.

Accuracy and Reliability Enhancement

Compared to conventional manual inspection techniques, the accuracy of YOLO models in identifying and classifying underbody problems represents a significant improvement. The automated method reaches unparalleled levels of consistency and reliability by reducing subjectivity and human error. This precision is essential to maintaining road safety since it guarantees that any possible problems are quickly found and fixed, improving road safety in general.

Cost Saving

Vehicle owners and repair facilities can save a lot of money by automating the inspection process. Labor costs can be reduced due to the shorter inspection times, and unnecessary repairs and maintenance can be avoided thanks to the precision of the system. Early problem detection can also prevent future damage that is more costly and severe, which has long-term financial benefits.

Scalability and Adaptability

The predictive maintenance benefits of the system's capacity to evaluate historical inspection data and identify trends are substantial. It is feasible to optimize maintenance schedules and proactively handle prospective issues before they escalate by utilizing a robust data management system. This proactive approach increases overall driving safety while also extending the life of automobiles.

Predictive Occurrence and Repair

Predictive occurrence and repair gains new prospects with the backing of a strong data management system and the capacity to examine inspection data from the past. Maintenance plans can be optimized and any problems can be handled before they worsen by seeing patterns and trends in underbody flaws in vehicles. In addition to increasing vehicle longevity, this preventative maintenance strategy improves driving safety.

5.1 Recommendation

Based on the experiment did, there are several suggestions can be given upon enhancing the model performances and overall system efficiency.

Firstly is to improve the final prediction's accuracy and robustness, ensemble learning integrates predictions from several models. Ensemble learning improves the overall

performance and generalization of deep learning models by the training and merging of several models, each capturing distinct features of the input.

Secondly, upgrading of Pi Zero WH with a higher processing power and speed single-board computer for a better reaction time during real-time defect inspection. Image can be captured and send back to the webpage quickly and defect inspection for streaming feature can be conducted due to higher frame per second or frame rate.

Thirdly, integration of other types of defect inspection into the system to undergo inspection for all in one go to reduce total inspection time for a vehicle. This can give benefits to the production rate of vehicles and cost-saving. Data management of each vehicle can be detailed and systematic due to inspection of all defects in one time and results stored.

The process of deep learning training is complex, and these extra suggestions are meant to make the training even more engaging. These suggestions would surely enhance the entire training experience and result in a more efficient vehicle underbody inspection system for anomaly identification when incorporated into the deep learning training process.

REFERENCES

[1] L. Yang, Y. An, N. Wu, P. Chen, H. Cheng, and X. Zhao, “A petri net-based heuristic algorithm for short-term vehicle scheduling in a vehicle inspection system,” IEEE Access, vol. 7, pp. 138442–138460, 2019, doi: 10.1109/ACCESS.2019.2942851.

[2] Statista Search Department, “Automotive Industry in Malaysia - Statistics & Facts,” Statista. Accessed: Feb. 01, 2024. [Online]. Available: https://www.statista.com/topics/5040/automotive-industry-in-malaysia/

[3] G. Gimino, R. Rahim, R. Vethasalam, and T. A. Yusof, “915,874 road accidents recorded throughout 2021 and 2022, says Transport Ministry,” The Star, Jun. 13, 2023. Accessed: Feb. 01, 2024. [Online]. Available: https://www.thestar.com.my/news/nation/2023/06/13/915874-road-accidents-recorded- in-2021-and-2022-says-transport-ministry

[4] Sudden Impact Auto, “4 Common Undercarriage Damages We Tend To Miss,” Sudden Impact Auto. Accessed: Feb. 05, 2024. [Online]. Available: https://www.suddenimpactauto.com/4-common-undercarriage-damages-we-tend-to- miss/

[5] E. E. Ruiz and K. L. Head, “Use of an automatic under-vehicle inspection system as a tool to streamline vehicle screening at ports of entry and security checkpoints,” in Proceedings - 2012 European Intelligence and Security Informatics Conference, EISIC 2012, 2012, pp. 329–333. doi: 10.1109/EISIC.2012.64.

[6] S. R. Sukumar, D. L. Page, A. F. Koschan, and M. A. Abidi, “Under Vehicle Inspection with 3d Imaging,” in 3D Imaging for Safety and Security, Springer Netherlands, 2007, pp. 249–278. doi: 10.1007/978-1-4020-6182-0_11.

[7] D. W. James, F. Belblidia, J. E. Eckermann, and J. Sienz, “An innovative photogrammetry color segmentation based technique as an alternative approach to 3D scanning for reverse engineering design,” Comput Aided Des Appl, vol. 14, no. 1, pp. 1– 16, Jan. 2017, doi: 10.1080/16864360.2016.1199751.

[8] M. Bojarski et al., “End to End Learning for Self-Driving Cars,” Apr. 2016, [Online]. Available: http://arxiv.org/abs/1604.07316

[9] M. Liu, J. Niu, and X. Wang, “An Autopilot System Based on ROS Distributed Architecture and Deep Learning,” 2017.

[10] P. Wlodarczak, J. Soar, and M. Ally, “Multimedia data mining using deep learning,” in 2015 5th International Conference on Digital Information Processing and Communications, ICDIPC 2015, Institute of Electrical and Electronics Engineers Inc., Nov. 2015, pp. 190–196. doi: 10.1109/ICDIPC.2015.7323027.

[11] D. Castelvecchi, “Deep learning boosts Google Translate tool,” Nature. Nature Research, 2020. doi: 10.1038/nature.2016.20696.

[12] R. Li et al., “Deep learning based imaging data completion for improved brain disease diagnosis,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2014, pp. 305–312. doi: 10.1007/978-3-319-10443-0_39.

[13] S. Fathi, M. Ahmadi, and A. Dehnad, “Early diagnosis of Alzheimer’s disease based on deep learning: A systematic review,” Computers in Biology and Medicine, vol. 146. Elsevier Ltd, Jul. 01, 2022. doi: 10.1016/j.compbiomed.2022.105634.

[14] L. Nie, M. Wang, L. Zhang, S. Yan, B. Zhang, and T. S. Chua, “Disease Inference from Health-Related Questions via Sparse Deep Learning,” IEEE Trans Knowl Data Eng, vol. 27, no. 8, pp. 2107–2119, Aug. 2015, doi: 10.1109/TKDE.2015.2399298.

[15] F. Y. Zhang and R. J. Liu, “Study on the Parts Surface Defect Detection Method Based on Modifed SVM Algorithm,” Applied Mechanics and Materials, vol. 541–542, pp. 1447–1451, Mar. 2014, doi: 10.4028/www.scientific.net/AMM.541-542.1447.

[16] B. Salehi, Y. Zhang, M. Zhong, and V. Dey, “Object-Based Classification of Urban Areas Using VHR Imagery and Height Points Ancillary Data,” Remote Sens (Basel), vol. 4, no. 8, pp. 2256–2276, Aug. 2012, doi: 10.3390/rs4082256.

[17] Qualcomm Developer Network, “Classification, Object Detection and Image Segmentation,” Qualcomm. Accessed: Feb. 03, 2024. [Online]. Available: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation

[18] L. Caro Campos, J. C. SanMiguel, and J. M. Martinez, “Discrimination of abandoned and stolen object based on active contours,” in 2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), IEEE, Aug. 2011, pp. 101–106. doi: 10.1109/AVSS.2011.6027302.

[19] G. Shao, L. Tang, and H. Zhang, “Introducing Image Classification Efficacies,” IEEE Access, vol. 9, pp. 134809–134816, 2021, doi: 10.1109/ACCESS.2021.3116526.

[20] P. Vadapalli, “Ultimate Guide to Object Detection Using Deep Learning [2024],” upGrad. Accessed: Feb. 05, 2024. [Online]. Available: https://www.upgrad.com/blog/ultimate-guide-to-object-detection-using-deep-learning/

[21] J. Xie, E. Stensrud, and T. Skramstad, “Detection-Based Object Tracking Applied to Remote Ship Inspection,” Sensors, vol. 21, no. 3, p. 761, Jan. 2021, doi: 10.3390/s21030761.

[22] W. B. Verschoof‐van der Vaart and K. Lambers, “Applying automated object detection in archaeological practice: A case study from the southern Netherlands,” Archaeol Prospect, vol. 29, no. 1, pp. 15–31, Jan. 2022, doi: 10.1002/arp.1833.

[23] B. Remeseiro, J. Tarrío-Saavedra, M. Francisco-Fernández, M. G. Penedo, S. Naya, and R. Cao, “Automatic detection of defective crankshafts by image analysis and supervised classification,” International Journal of Advanced Manufacturing Technology, vol. 105, no. 9, pp. 3761–3777, Dec. 2019, doi: 10.1007/s00170-019-03819-7.

[24] K. Wu, S. A. Prieto, and B. García de Soto, “Measuring The Spacing of Formwork System Members Using 3D Point Clouds,” Periodica Polytechnica Budapest University of Technology and Economics, 2022, pp. 396–404. doi: 10.3311/ccc2022-049.

[25] J. Cemenska, T. Rudberg, and M. Henscheid, “Automated In-Process Inspection System for AFP Machines,” SAE International Journal of Aerospace, vol. 8, no. 2, pp. 2015-01– 2608, Sep. 2015, doi: 10.4271/2015-01-2608.

[26] Y. Liu, X. Nie, J. Fan, and X. Liu, “Image‐based crack assessment of bridge piers using unmanned aerial vehicles and three‐dimensional scene reconstruction,” Computer-

Aided Civil and Infrastructure Engineering, vol. 35, no. 5, pp. 511–529, May 2020, doi: 10.1111/mice.12501.

[27] R. Webb, “12 Challenges of Data Analytics and How to Fix Them,” ClearRisk. Accessed: Feb. 05, 2024. [Online]. Available: https://www.clearrisk.com/risk- management-blog/challenges-of-data-analytics-0

[28] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” Int J Comput Vis, vol. 1, no. 4, pp. 321–331, Jan. 1988, doi: 10.1007/BF00133570.

[29] R. W. Conners and C. A. Harlow, “A Theoretical Comparison of Texture Algorithms,” IEEE Trans Pattern Anal Mach Intell, vol. PAMI-2, no. 3, pp. 204–222, May 1980, doi: 10.1109/TPAMI.1980.4767008.

[30] O. Ben Sassi, “Improved Spatial Gray Level Dependence Matrices for Texture Analysis,” International Journal of Computer Science and Information Technology, vol. 4, no. 6, pp. 209–219, Dec. 2012, doi: 10.5121/ijcsit.2012.4615.

[31] W. Xinlily, F. Albregtsen, and B. Foyn, “Texture Features from Gray Level Gap Length Matrix,” 1994.

[32] J. Yang and J. Guo, “Image Texture Feature Extraction Method Based on Regional Average Binary Gray Level Difference Co-occurrence Matrix,” in 2011 International Conference on Virtual Reality and Visualization, IEEE, Nov. 2011, pp. 239–242. doi: 10.1109/ICVRV.2011.20.

[33] A. M. Shabat and J.-R. Tapamo, “A comparative study of the use of local directional pattern for texture-based informal settlement classification,” Journal of Applied Research and Technology, vol. 15, no. 3, pp. 250–258, Jun. 2017, doi: 10.1016/j.jart.2016.12.009.

[34] O. Rioul and P. Duhamel, “Fast algorithms for discrete and continuous wavelet transforms,” IEEE Trans Inf Theory, vol. 38, no. 2, pp. 569–586, Mar. 1992, doi: 10.1109/18.119724.

[35] U. Oberst, “The Fast Fourier Transform,” SIAM J Control Optim, vol. 46, no. 2, pp. 496–540, Jan. 2007, doi: 10.1137/060658242.

[36] M. Ghazvini, S. A. Monadjemi, N. Movahhedinia, and K. Jamshidi, “Defect Detection of Tiles Using 2D Wavelet Transform and Statistical Features,” International Journal of Electrical and Computer Engineering, vol. 3, no. 1, 2009.

[37] Y. A. Karayiannis et al., “Defect detection and classification on web textile fabric using multiresolution decomposition and neural networks,” in ICECS’99. Proceedings of ICECS ’99. 6th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.99EX357), IEEE, pp. 765–768. doi: 10.1109/ICECS.1999.813221.

[38] S. Vallerand and X. Maldague, “Defect characterization in pulsed thermography: a statistical method compared with Kohonen and Perceptron neural networks,” NDT & E International, vol. 33, no. 5, pp. 307–315, Jul. 2000, doi: 10.1016/S0963-8695(99)00056-0.

[39] D. t Pham, A. Soroka, A. Ghanbarzadeh, E. Koc, S. Otri, and M. Packianather, “Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm,” in 2006 IEEE International Conference on Industrial Informatics, IEEE, Aug. 2006, pp. 1346–1351. doi: 10.1109/INDIN.2006.275855.

[40] W. Gao, X. Chen, and D. Chen, “Genetic programming approach for predicting service life of tunnel structures subject to chloride-induced corrosion,” J Adv Res, vol. 20, pp. 141–152, Nov. 2019, doi: 10.1016/j.jare.2019.07.001.

[41] H. Kasban, O. Zahran, H. Arafa, M. El-Kordy, S. M. S. Elaraby, and F. E. Abd El-Samie, “Welding defect detection from radiography images with a cepstral approach,” NDT & E International, vol. 44, no. 2, pp. 226–231, Mar. 2011, doi: 10.1016/j.ndteint.2010.10.005.

[42] A. Tonazzini et al., “Analytical and mathematical methods for revealing hidden details in ancient manuscripts and paintings: A review,” J Adv Res, vol. 17, pp. 31–42, May 2019, doi: 10.1016/j.jare.2019.01.003.

[43] B. Gosselin and D. Unay, “Apple defect detection and quality classification with MLP- Neural networks,” 2004. [Online]. Available: https://www.researchgate.net/publication/228952374

[44] S. Mishra et al., “Principal Component Analysis,” International Journal of Livestock Research, p. 1, 2017, doi: 10.5455/ijlr.20170415115235.

[45] Y. Lecun, L. Eon Bottou, Y. Bengio, and P. H. Abstract|, “Gradient-Based Learning Applied to Document Recognition.”

[46] BBC News, “Artificial intelligence: Google’s AlphaGo beats Go master Lee Se-dol.” Accessed: Feb. 05, 2024. [Online]. Available: https://www.bbc.com/news/technology-35785875

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[48] R. Munjal, S. Arif, F. Wendler, and O. Kanoun, “Comparative Study of Machine-Learning Frameworks for the Elaboration of Feed-Forward Neural Networks by Varying the Complexity of Impedimetric Datasets Synthesized Using Eddy Current Sensors for the Characterization of Bi-Metallic Coins,” Sensors, vol. 22, no. 4, p. 1312, Feb. 2022, doi: 10.3390/s22041312.

[49] B. Pang, E. Nijkamp, and Y. N. Wu, “Deep Learning With TensorFlow: A Review,” Journal of Educational and Behavioral Statistics, vol. 45, no. 2, pp. 227–248, Apr. 2020, doi: 10.3102/1076998619872761.

[50] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.01703

[51] H. Jin, Q. Song, and X. Hu, “Auto-Keras: An Efficient Neural Architecture Search System,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA: ACM, Jul. 2019, pp. 1946–1956. doi: 10.1145/3292500.3330648.

[52] Z. Z, W. W, S. W, K. J, C. KJ, and T. PH, “Real-time envelope detection of ultrasound radiofrequency signals using OpenCV GPU framework,” Medical Devices and Diagnostic Engineering, vol. 2, no. 2, 2017, doi: 10.15761/MDDE.1000121.

[53] B. Kevin, B. Vikin, and M. Nair, “Building A Chatbot for Healthcare using NLP”, doi: 10.36227/techrxiv.22578472.v1.

[54] Y. E. Wang, G.-Y. Wei, and D. Brooks, “Benchmarking TPU, GPU, and CPU Platforms for Deep Learning,” Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.10701

[55] H. Zhang et al., “Poseidon: A System Architecture for Efficient GPU-based Deep Learning on Multiple Machines,” Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.06216

[56] W. Rahmaniar and A. Hernawan, “Real-Time Human Detection Using Deep Learning on Embedded Platforms: A Review,” Journal of Robotics and Control (JRC), vol. 2, no. 6, 2021, doi: 10.18196/jrc.26123.

[57] S. Hong, H. Cho, and J.-S. Kim, “CitiusSynapse: A Deep Learning Framework for Embedded Systems,” Applied Sciences, vol. 11, no. 23, p. 11570, Dec. 2021, doi: 10.3390/app112311570.

[58] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” May 2016, [Online]. Available: http://arxiv.org/abs/1605.08695

[59] F. Huot, Y.-F. Chen, R. Clapp, C. Boneti, and J. Anderson, “High-resolution imaging on TPUs,” Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.08063

[60] S. Shi, Q. Wang, P. Xu, and X. Chu, “Benchmarking State-of-the-Art Deep Learning Software Tools,” in 2016 7th International Conference on Cloud Computing and Big Data (CCBD), IEEE, Nov. 2016, pp. 99–104. doi: 10.1109/CCBD.2016.029.

[61] G. Jeong et al., “VEGETA: Vertically-Integrated Extensions for Sparse/Dense GEMM Tile Acceleration on CPUs,” Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.08687

[62] M. S. Al-Asaly, M. A. Bencherif, A. Alsanad, and M. M. Hassan, “A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment,” Neural Comput Appl, vol. 34, no. 13, pp. 10211–10228, Jul. 2022, doi: 10.1007/s00521-021-06665-5.

[63] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[64] R. Girshick, “Fast R-CNN,” in 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, Dec. 2015, pp. 1440–1448. doi: 10.1109/ICCV.2015.169.

[65] Y. Zhang, “Stall Number Detection of Cow Teats Key Frames,” Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.10444

[66] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” Nov. 2013, [Online]. Available: http://arxiv.org/abs/1311.2524

[67] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Trans Pattern Anal Mach Intell, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[68] W. Liu et al., “SSD: Single Shot MultiBox Detector,” 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[69] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, Oct. 2017, pp. 2980–2988. doi: 10.1109/ICCV.2017.322.

[70] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.02002

[71] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and Efficient Object Detection,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2020, pp. 10778–10787. doi: 10.1109/CVPR42600.2020.01079.

[72] W. Lan, J. Dang, Y. Wang, and S. Wang, “Pedestrian Detection Based on YOLO Network Model,” in 2018 IEEE International Conference on Mechatronics and Automation (ICMA), IEEE, Aug. 2018, pp. 1547–1551. doi: 10.1109/ICMA.2018.8484698.

[73] A. Benjumea, I. Teeti, F. Cuzzolin, and A. Bradley, “YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles,” Dec. 2021, [Online]. Available: http://arxiv.org/abs/2112.11798

[74] Y. Li, J. Wang, J. Huang, and Y. Li, “Research on Deep Learning Automatic Vehicle Recognition Algorithm Based on RES-YOLO Model,” Sensors, vol. 22, no. 10, p. 3783, May 2022, doi: 10.3390/s22103783.

[75] S. Liang et al., “Edge YOLO: Real-Time Intelligent Object Detection System Based on Edge-Cloud Cooperation in Autonomous Vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 12, pp. 25345–25360, Dec. 2022, doi: 10.1109/TITS.2022.3158253.

[76] S. Shinde, A. Kothari, and V. Gupta, “YOLO based Human Action Recognition and Localization,” Procedia Comput Sci, vol. 133, pp. 831–838, 2018, doi: 10.1016/j.procs.2018.07.112.

[77] A. Hanan Ashraf et al., “Weapons Detection for Security and Video Surveillance Using CNN and YOLO-V5s,” Computers, Materials & Continua, vol. 70, no. 2, pp. 2761–2775, 2022, doi: 10.32604/cmc.2022.018785.

[78] Y. Zheng and H. Zhang, “Video Analysis in Sports by Lightweight Object Detection Network under the Background of Sports Industry Development,” Comput Intell Neurosci, vol. 2022, pp. 1–10, Aug. 2022, doi: 10.1155/2022/3844770.

[79] A. N. Arun, P. Maheswaravenkatesh, and T. Jayasankar, “Facial Micro Emotion Detection and Classification Using Swarm Intelligence based Modified Convolutional Network,” Expert Syst Appl, vol. 233, p. 120947, Dec. 2023, doi: 10.1016/j.eswa.2023.120947.

[80] M. Lippi, N. Bonucci, R. F. Carpio, M. Contarini, S. Speranza, and A. Gasparri, “A YOLO-Based Pest Detection System for Precision Agriculture,” in 2021 29th Mediterranean Conference on Control and Automation (MED), IEEE, Jun. 2021, pp. 342–347. doi: 10.1109/MED51440.2021.9480344.

[81] D. Wu, S. Lv, M. Jiang, and H. Song, “Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments,” Comput Electron Agric, vol. 178, p. 105742, Nov. 2020, doi: 10.1016/j.compag.2020.105742.

[82] Y. Nie, P. Sommella, M. O'Nils, C. Liguori, and J. Lundgren, "Automatic Detection of Melanoma with Yolo Deep Convolutional Neural Networks," in 2019 E-Health and Bioengineering Conference (EHB), IEEE, Nov. 2019, pp. 1–4. doi: 10.1109/EHB47216.2019.8970033.

[83] H. M. Ünver and E. Ayan, "Skin Lesion Segmentation in Dermoscopic Images with Combination of YOLO and GrabCut Algorithm," Diagnostics, vol. 9, no. 3, p. 72, Jul. 2019, doi: 10.3390/diagnostics9030072.

[84] L. Tan, T. Huangfu, L. Wu, and W. Chen, "Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification," BMC Med Inform Decis Mak, vol. 21, no. 1, p. 324, Dec. 2021, doi: 10.1186/s12911-021-01691-8.

[85] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-face: a real-time face detector," Visual Computer, vol. 37, no. 4, pp. 805–813, Apr. 2021, doi: 10.1007/s00371-020-01831-7.

[86] Z. Zakria, J. Deng, R. Kumar, M. S. Khokhar, J. Cai, and J. Kumar, "Multiscale and Direction Target Detecting in Remote Sensing Images via Modified YOLO-v4," IEEE J Sel Top Appl Earth Obs Remote Sens, vol. 15, pp. 1039–1048, 2022, doi: 10.1109/JSTARS.2022.3140776.

[87] L. Cheng, J. Li, P. Duan, and M. Wang, "A small attentional YOLO model for landslide detection from satellite remote sensing images," Landslides, vol. 18, no. 8, pp. 2751– 2765, Aug. 2021, doi: 10.1007/s10346-021-01694-6.

[88] P. Kumar, S. Narasimha Swamy, P. Kumar, G. Purohit, and K. S. Raju, "Real-Time, YOLO-Based Intelligent Surveillance and Monitoring System Using Jetson TX2," 2021, pp.461–471. doi: 10.1007/978-981-15-8335-3_35.

[89] K. Bhambani, T. Jain, and K. A. Sultanpure, "Real-time Face Mask and Social Distancing Violation Detection System using YOLO," in 2020 IEEE Bangalore Humanitarian Technology Conference (B-HTC), IEEE, Oct. 2020, pp. 1–6. doi: 10.1109/B-HTC50970.2020.9297902.

[90] Y. Du, N. Pan, Z. Xu, F. Deng, Y. Shen, and H. Kang, “Pavement distress detection and classification based on YOLO network,” International Journal of Pavement Engineering, vol. 22, no. 13, pp. 1659–1672, Nov. 2021, doi: 10.1080/10298436.2020.1714047.

[91] E. N. Ukhwah, E. M. Yuniarno, and Y. K. Suprapto, “Asphalt Pavement Pothole Detection using Deep learning method based on YOLO Neural Network,” in 2019 International Seminar on Intelligent Technology and Its Applications (ISITIA), IEEE, Aug. 2019, pp. 35–40. doi: 10.1109/ISITIA.2019.8937176.

[92] J. Li, Z. Su, J. Geng, and Y. Yin, “Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network,” IFAC-PapersOnLine, vol. 51, no. 21, pp. 76–81, 2018, doi: 10.1016/j.ifacol.2018.09.412.

[93] Hendry and R.-C. Chen, “Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning,” Image Vis Comput, vol. 87, pp. 47–56, Jul. 2019, doi: 10.1016/j.imavis.2019.04.007.

[94] C. Dewi, R.-C. Chen, X. Jiang, and H. Yu, “Deep convolutional neural network for enhancing traffic sign recognition developed on Yolo V4,” Multimed Tools Appl, vol. 81, no. 26, pp. 37821–37845, Nov. 2022, doi: 10.1007/s11042-022-12962-5.

[95] A. M. Roy, J. Bhaduri, T. Kumar, and K. Raj, “WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection,” Ecol Inform, vol. 75, p. 101919, Jul. 2023, doi: 10.1016/j.ecoinf.2022.101919.

[96] D. H. Dos Reis, D. Welfer, M. A. De Souza Leite Cuadros, and D. F. T. Gamarra, “Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm,” Applied Artificial Intelligence, vol. 33, no. 14, pp. 1290– 1305, Dec. 2019, doi: 10.1080/08839514.2019.1684778.

[97] S. D. Kulik and A. N. Shtanko, “Experiments with Neural Net Object Detection System YOLO on Small Training Datasets for Intelligent Robotics,” 2020, pp. 157–162. doi: 10.1007/978-3-030-33491-8_19.

[98] O. Sahin and S. Ozer, “YOLODrone: Improved YOLO Architecture for Object Detection in Drone Images,” in 2021 44th International Conference on Telecommunications

and Signal Processing (TSP), IEEE, Jul. 2021, pp. 361–365. doi: 10.1109/TSP52935.2021.9522653.

[99] C. Chen et al., “YOLO-Based UAV Technology: A Review of the Research and Its Applications,” Drones, vol. 7, no. 3, p. 190, Mar. 2023, doi: 10.3390/drones7030190.

[100] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” Int J Comput Vis, vol. 88, no. 2, pp. 303–338, Jun. 2010, doi: 10.1007/s11263-009-0275-4.

[101] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” May 2014, [Online]. Available: http://arxiv.org/abs/1405.0312

[102] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” Mach Learn Knowl Extr, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.

[103] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 7464–7475. https://doi.org/10.1109/cvpr52729.2023.00721

[104] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 12993–13000, Apr. 2020, doi: 10.1609/aaai.v34i07.6999.

[105] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs),” Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.08415

[106] G. Ghiasi et al., “Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation,” Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.07177

[107] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.09412

[108] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and Flexible Image Augmentations,” Information, vol. 11, no. 2, p. 125, Feb. 2020, doi: 10.3390/info11020125.

[109] Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. https://arxiv.org/abs/2402.13616v2

[110] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, “RepVGG: Making VGG-style ConvNets Great Again,” in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2021, pp. 13728–13737. doi: 10.1109/CVPR46437.2021.01352.

[111] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, “TOOD: Task-aligned One-stage Object Detection,” Aug. 2021, [Online]. Available: http://arxiv.org/abs/2108.07755

[112] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf, “VarifocalNet: An IoU-aware Dense Object Detector,” Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.13367

[113] Z. Gevorgyan, “SIOU LOSS: MORE POWERFUL LEARNING FOR BOUNDING BOX REGRESSION 1 SIoU Loss: More Powerful Learning for Bounding Box Regression SIOU LOSS: MORE POWERFUL LEARNING FOR BOUNDING BOX REGRESSION 2.”

[114] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression,” in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Jun. 2019, pp. 658–666. doi: 10.1109/CVPR.2019.00075.

[115] X. Ding, H. Chen, X. Zhang, K. Huang, J. Han, and G. Ding, “Re-parameterizing Your Optimizers rather than Architectures,” May 2022, [Online]. Available: http://arxiv.org/abs/2205.15242

APPENDICES

Appendix A: Gantt Chart

#	Task Description	Start Date	End Date	Duration (Days)		Week 1	2	3	4	5	Midterm Break	6	7	8	9	10	11	12	13	14
1	**Contacting of supervisor in charge** Description: - Supervision confirmation with the supervisor selected - arranging of apointment with supervisor to have the first meeting	9-Oct-23	13-Oct-23	5	PLANNED															
					ACTUAL															
2	**Attending of FYP Briefing** Description: - Overview of the course outcome and the outline of events in this semester - To have a rough idea and expectation of the work should be done	18-Oct-23	18-Oct-23	1	PLANNED															
					ACTUAL															
3	**First meeting with supervisor** Description: - Discussion and selection of the title given by the supervisor - Introduction and explanation of work flow under the supervision of the in charge supervisor	20-Oct-23	20-Oct-23	1	PLANNED															
					ACTUAL															
4	**Investigation of researches regarding Literature Review** Description: - Searching case study, articles, research paper, etc. which are related to the topic give. - Study the knowledge of deep learning which needs to be used for defect inspection. - Study the working principle and the flow of 3D analysing inspection for defect wire bond.	16-Oct-23	25-Oct-23	9	PLANNED															
					ACTUAL															
5	**Second meeting with supervisor** Description: - Discuss and understand to kickstart of our FYP - Introduction and installation of Mendeley for the next meeting - Guidance and ways given by supervisor to find related researches for literature review and gain our understanding more	23-Oct-23	23-Oct-23	1	PLANNED															
					ACTUAL															
6	**Third meeting with supervisor** Description: - Learning of using Mendeley for citation purposes from supervisor - Discussion with supervisor to check and guide me through the articles or research papers found which related to the topicDiscuss what I found and what I known - Discuss in details on the top 3 articles that the most related to the topica - Obtained and understand the dataset and the overall needs for this topic given by the supervisor.	30-Oct-23	30-Oct-23	1	PLANNED															
					ACTUAL															
7	**Listing of the research papers in chapter 2** Description: - This is to gain more understanding and knowledge towards the topic we are going to investigate so that we will know how to have a better explanation and describe for the research background, aims, problem statement and objectives. - Research and make comparison in details on the methods available through the article found from literature review	26-Oct-23	17-Nov-23	18	PLANNED															
					ACTUAL															
8	**Attending of Profesor talk series in FYP Briefing** Description: - Participate in a talk by profesor from industry in FYP briefing	1-Nov-23	1-Nov-23	1	PLANNED															
					ACTUAL															
9	**Forth meeting with supervisor** Description: - Discussion for the contents of Chapter 1 - Asking questions related to the format of chapter 1 and categorization of chapter 2	8-Nov-23	8-Nov-23	1	PLANNED															
					ACTUAL															
10	**Attending of FYP Seminar for chapter 1 and 2** Description: - Explanation of the contents requirement for chapter 1 and 2 - Overview the the needs for the submission of chapter 1 - To update and remind us the latest assessment schedule and upcoming events	8-Nov-23	8-Nov-23	1	PLANNED															
					ACTUAL															

11	**Writing of chapter 1 based on the research papers collcted** Description: - Writing of introduction, research background, problem statement, aims and objectives, research scope and thesis outline. - Continue of finding any related research paper	15-Nov-23	20-Nov-23	5	PLANNED ACTUAL	
12	**Fifth meeting with supervisor** Description: - Discussion of the contents for Chapter 1 - Correct any unecessary mistake or contens after checking by supervisor	21-Nov-23	21-Nov-23	1	PLANNED ACTUAL	
13	**Complete Chapter 1 for mid-sem evaluation**	22-Nov-23	22-Nov-23	1	PLANNED ACTUAL	
14	**Attending industrial talk** Description: - Design and product development in automotive industry by Puan Jehan Adnan	29-Nov-23	29-Nov-23	1	PLANNED ACTUAL	
15	**Sixth meeting with supervisor** Description: - Discussion in details for Chapter 2 - Discussion on what I should do for Chapter 3	30-Nov-23	30-Nov-23	1	PLANNED ACTUAL	

16	**Writing of literature review of chapter 2** Description: - Writing of all the defect inspection methods I found during research of the articles - Writing discussion of deep learning models with their concepts	30-Nov-23	5-Dec-23	6	PLANNED ACTUAL	
17	**Getting started with chapter 3 methodology** Description: - Writing the contents of methodology according to chapter 2, objectives and problem statement.	4-Dec-23	6-Dec-23	3	PLANNED ACTUAL	
18	**Attending industrial talk** Description: - Technical Presentation And Report by Associate Lt. Col. (CD) Ts. Zulhaidi Mohd Jawi	6-Dec-23	6-Dec-23	1	PLANNED ACTUAL	
19	**Seventh meeting with supervisor** Description: - Discussion on what should I buy for deep learning training - Asking questions I gained from chapter 3 - Updating chapter 2 and chapter 3 with supervisor to check	6-Dec-23	6-Dec-23	1	PLANNED ACTUAL	
20	**Correction and continue writing of chapter 2** Description: - Explanation of deep learning models that will be used - correction for the methods of writing paragraphs in chapter 2	7-Dec-23	18-Dec-23	12	PLANNED ACTUAL	
21	**Writing of methodology for chapter 3** Description: - explanation of the process flow of this research	8-Dec-23	14-Dec-23	7	PLANNED ACTUAL	

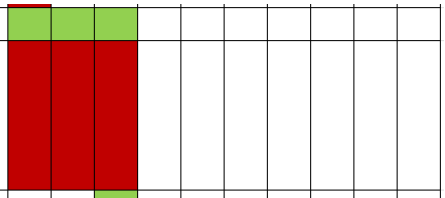
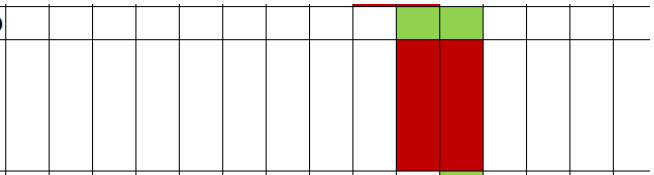
21	**Writing of methodology for chapter 3** Description: - explanation of the process flow of this research	8-Dec-23	14-Dec-23	7	PLANNED ACTUAL	
22	**Attending last seminar held for FYP talk** Description: - Checking of the contents needed for chapter 3 - Discussiuon of the ontents needed for final presentation - Rubrics for panels evaluation	13-Dec-23	13-Dec-23	1	PLANNED ACTUAL	
23	**Eighth meeting with supervisor** Description: - Recommendation of Orange Data Mining by supervisor - Examples and tutorial taught by supervisor on using Orange - Chapter 3 methodology update with supervisor	14-Dec-23	14-Dec-23	1	PLANNED ACTUAL	
24	**Model training for preliminary results** Description: - output generation from Orange Data Mining - training of deep learning model for defect inspection using YOLOv8	14-Dec-23	28-Dec-23	15	PLANNED ACTUAL	
25	**Correction and continue writing of chapter 3**	14-Dec-23	2-Jan-24	18	PLANNED ACTUAL	

26	**Nineth meeting with supervisor** Description: - Explanation of flowchart by supervisor - Final evaluation on chapter 2 - Explanation of results needed for preliminary results	18-Dec-23	18-Dec-23	1	PLANNED ACTUAL	
27	**Tenth meeting with supervisor** Description: - Checking of flowchart to supervisor - Demonstrating preliminary results generated from trained model - Discussion of the preliminary results	28-Dec-23	28-Dec-23	1	PLANNED ACTUAL	
28	**Continue of model training for preliminary results**	28-Dec-23	7-Jan-24	11	PLANNED ACTUAL	
29	**Preparation of presentation slides** Description: - Writing of the process flow, methods, problem statement, objectives, preliminary results and discussion	3-Jan-24	8-Jan-24	6	PLANNED ACTUAL	
30	**Eleventh meeting with supervisor** Description: - Checking of presentation slides - Checking of preliminary resutls	5-Jan-24	5-Jan-24	1	PLANNED ACTUAL	
31	**Final presentation**	10-Jan-24	10-Jan-24	1	PLANNED ACTUAL	

APPENDICES

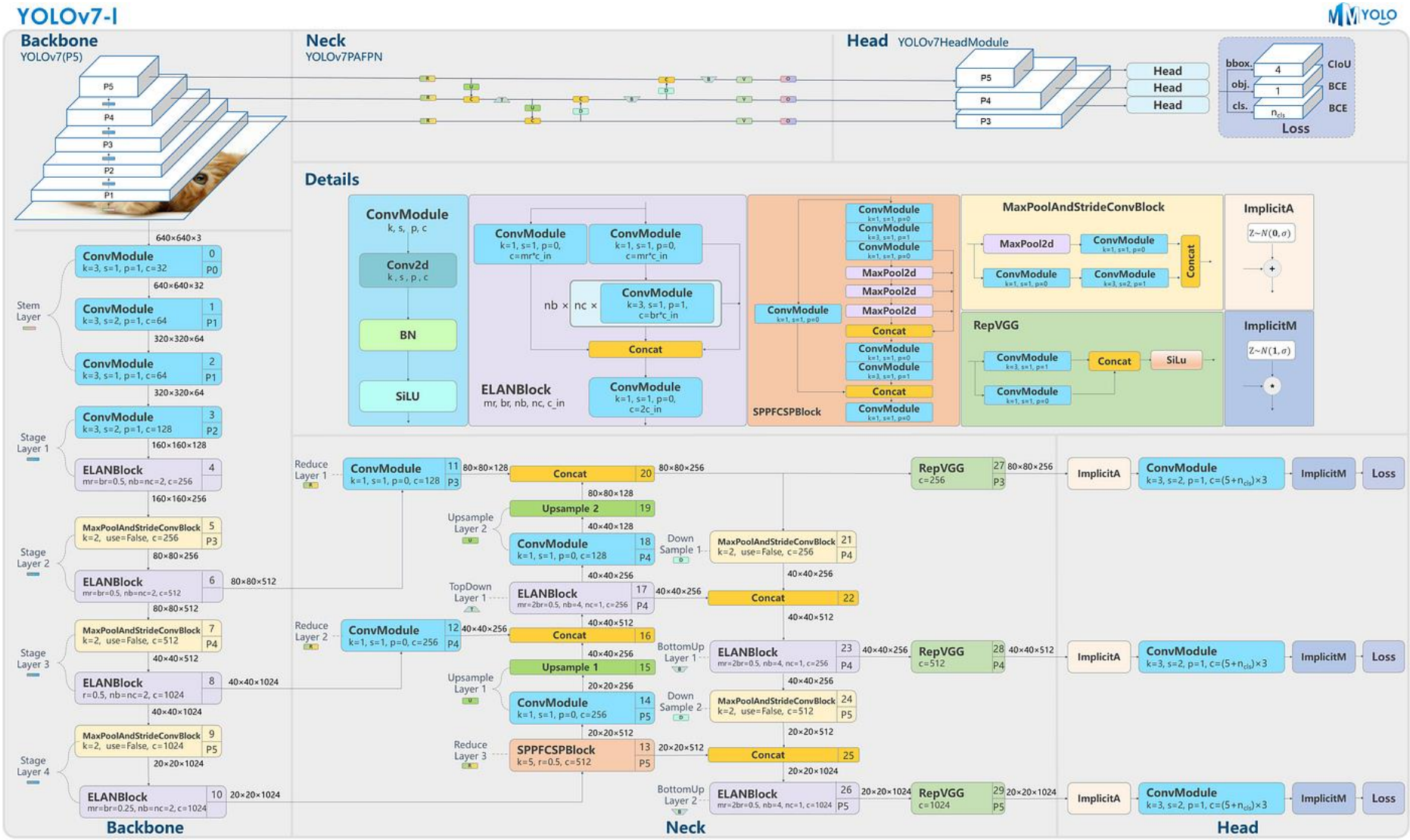
Appendix A: Gantt Chart

#	Task Description	Start Date	End Date	Duration (Days)		Week													
						1	2	3	4	5	MB	6	7	8	9	10	11	12	13	14
1	**Meeting with supervisor** Description: - Discussion regarding project with Volkswagen Group Malaysia - Meeting with Volkswagen Group to discuss the problem statement and requirements regarding the project - Create startup plan for the project	5/3/2024	9/3/2024	5	PLANNED														
					ACTUAL														
2	**Software: Data Collection** Description: - Data collection and analysis to find defect type - Data acquisition through smartphone camera capture - Data acquisition through video footage	10/3/2024	16/3/2024	6	PLANNED														
					ACTUAL														
3	**Software: Data Pre-processing** Description: - Data annotation - Data augmentation - Data distribution	17/3/2024	23/3/2024	7	PLANNED														
					ACTUAL														
4	**Electrical: Raspberry Pi Zero WH and Camera** Description: - Setup of OS system in Raspberry Pi Zero WH - Code designed for Pi Zero to received signal and take picture using the Hikvision web camera then sent the camera back	22/3/2023	28/3/2024	7	PLANNED														
					ACTUAL														
5	**Software: Model Setup and Training** Description: - Setting up of environment for YOLOv7, YOLOv8 and YOLOv9 - Training of these three models with different parameters, fine-tuning and dataset adjustments.	26/3/2024	26/4/2024	32	PLANNED														
					ACTUAL														
6	**Software: Installation of CUDA and Continue Training** Description: - Install and use CUDA to speed up the training process - Continue of training process to obtain best result - Performance analysis and model selection	7/4/2024	8/4/2024	2	PLANNED														
					ACTUAL														

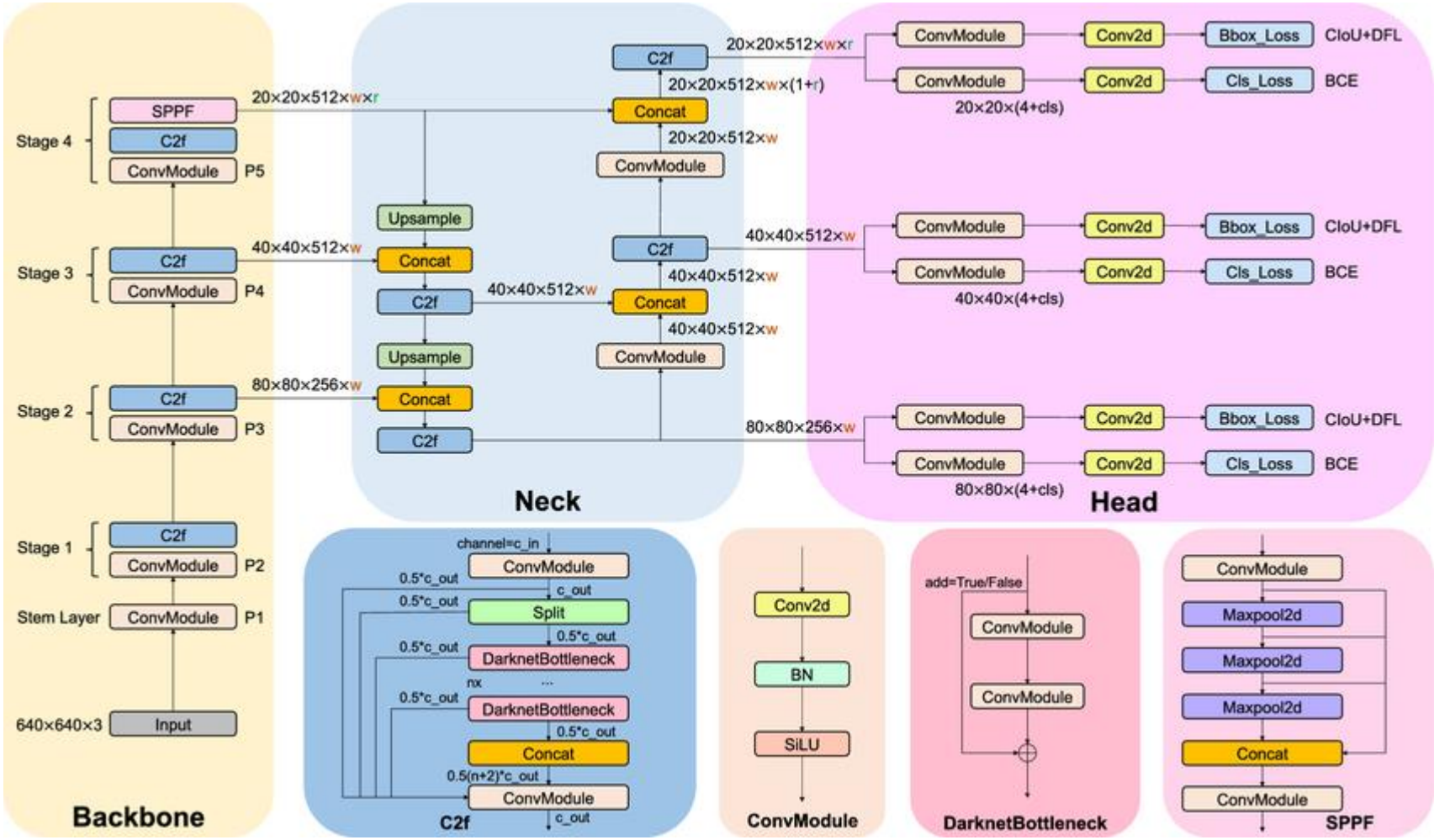
7	**Software: Webpage Design and Create Database** Description: - Designing of webpages using Streamlit with the implementation of model selected - Create database for data management using MySQL and PhpMyAdmin	8/4/2024	26/4/2024	19	PLANNED ACTUAL	
8	**Meeting with supervisor** Description: - Progress update and discussion to further improve the project	26/4/2024	26/4/2024	1	PLANNED ACTUAL	
9	**Midterm Evaluation Submission**	27/4/2024	27/4/2024	1	PLANNED ACTUAL	
10	**Electical: Jetson Orin Nano Setup** Description: - Setting up of OS system in Jetson Orin Nano using SDK manager with linux-based laptop - Installation of necessary software and libraries in Jetson board for further implementation	24/4/2024	7/5/2024	14	PLANNED ACTUAL	
11	**Electrical: Jetson Orin Nano Implementation** Description: - Implement the webpage designed, database and YOLO model into Jetson Orin Nano	7/5/2024	20/5/2024	14	PLANNED ACTUAL	
12	**Electrical: System Testing** Description: - Testing of defect inpsection in Jetson Orin Nano with Raspberry Pi Zero WH and Hikvision camera - Further debug and error solving	20/5/2024	22/5/2024	2	PLANNED ACTUAL	
13	**Meeting with supervisor** Description: - Progress update and system demonstration	22/5/2024	22/5/2024	1	PLANNED ACTUAL	
14	**Mechanical: 3D Design of Jetson Cover and Finding Suitable Camera Mount** Description: - 3D design of Jetson cover and 3D print it out - Surverying suitable camera mount and purchase - Final Testing the whole system	22/5/2024	29/5/2024	8	PLANNED ACTUAL	

15	**Completion of thesis report** Description: - Finish all the contents required in the report	28/5/2024	10/6/2024	14	PLANNED																	
					ACTUAL																	
16	**Completion of presentation poster** Description: - Complete thesis report with results and conclusion	4/6/2024	11/6/2024	8	PLANNED																	
					ACTUAL																	
17	**Completion of article writing** Description: - Writing of article to summarize project done	4/6/2024	13/6/2024	9	PLANNED																	
					ACTUAL																	
18	**FYP2 Poster and Project Presentation**	12/6/2024	12/6/2024	1	PLANNED																	
					ACTUAL																	
19	**Further update of thesis report and article** Description: - Update any information acquired from the project and panel	12/6/2024	14/6/2024	3	PLANNED																	
					ACTUAL																	
20	**Submission of thesis report and article**	16/6/2024	16/6/2024	1	PLANNED																	
					ACTUAL																	

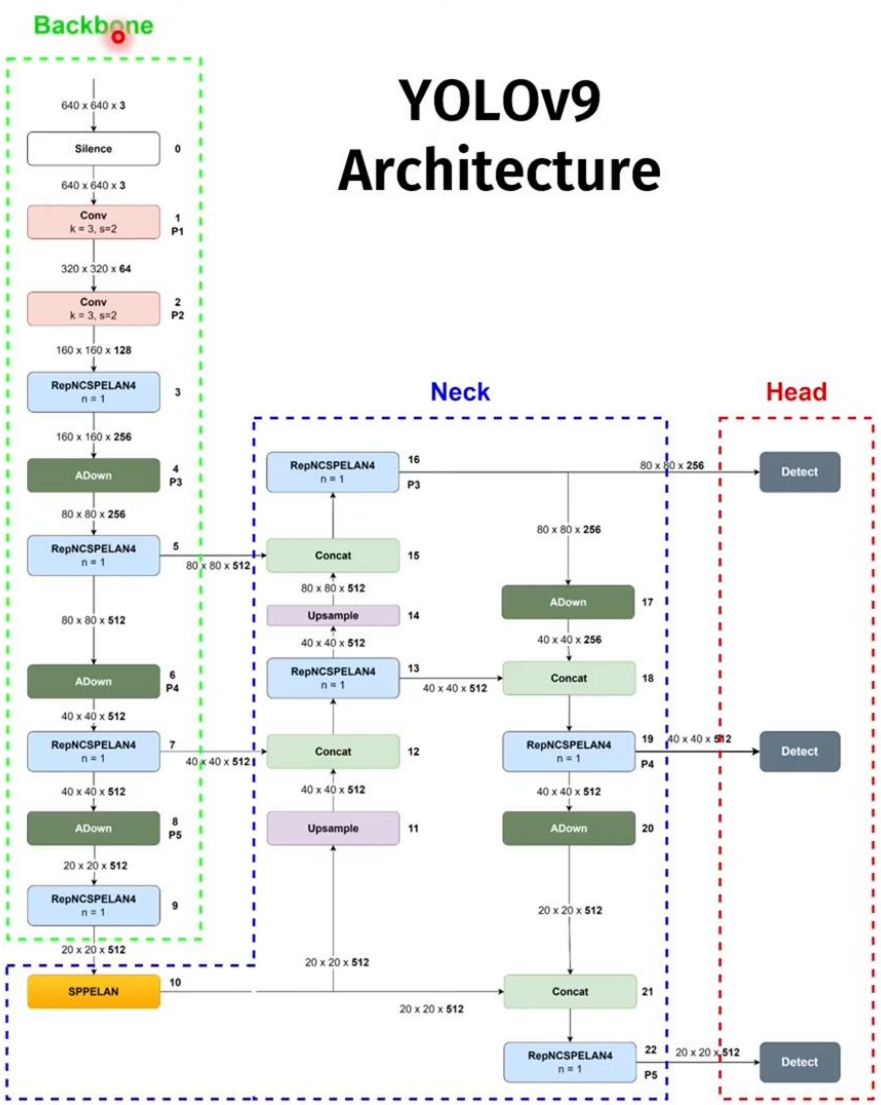
Appendix B: YOLOv7 Architecture



Appendix C: YOLOv8 Architecture



Appendix D: YOLOv9 Architecture



Appendix E: Concept Generation

No	Function	Concept 1	Concept 2	Concept 3	Concept 4
1	Data Labeling	LabelMe	Labellmg	Roboflow	CVAT
2	Deep Learning Framework	Tensorflow	Tensorflow	PyTorch	PyTorch
3	Deep Learning Model	Faster R-CNN	YOLO	DSSD	YOLO
4	Model Training Platform	Jupyter	Google Colab	VSCode	PyCharm
5	Data Transmission	LoRa	Bluetooth	WiFi	WiFi
6	Data Storage	MariaDB	MySQL	NoSQL Real-time Database	MySQL
7	Data Visualization	Gradio	Dash	Thingsboard	Streamlit
8	Camera	Atlas IMX545	Atlas IMX545	OAK-1 PoE	HikVision Web Camera DS-U02
9	Single-board Computer (Camera)	Raspberry Pi Zero	Raspberry Pi 2 W	Raspberry Pi Zero W	Raspberry Pi Zero WH
10	Single-board Computer (Detection)	Coral Dev Board	Jetson Orin Nano	Raspberry Pi 4 Model B	Jetson Orin Nano

11	Camera Mount Design	G clamp	Clip	G clamp	Clip
12	Mounting Manufacture	3D Print	Purchase	3D Print	Purchase

Appendix F: Homepage

```python
import streamlit as st

st.set_page_config(
    page_title="Home",
    page_icon='🏠',
    layout="wide",
    initial_sidebar_state="expanded"
)

st.sidebar.success("Select a page above.")

st.image("images/logo.png", width=150)
st.title("Project Explanation")

c1, c2 = st.columns([0.3, 0.5])
st.markdown("---")
col1, col2, col3 = st.columns([0.5, 0.3, 0.3])
st.markdown("---")
colu1, colu2, colu3 = st.columns([0.3, 0.3, 0.5])
st.markdown("---")
colum1, colum2 = st.columns([0.5, 0.3])
st.markdown("---")
```

```
23
24  with c1:
25      st.image("images/model.jpg", width=500)
26  with c2:
27      st.header("Enhanced Inspection System")
28      st.subheader("Deep Learning for Anomaly Detection of Underbody Vehicles")
29      proj_exp = "This project is developed to undergo car inspection with the help of Artificial Intelligence which" \
30                 "is YOLOv9 from Ultralytics. With the help of AI, detection process becomes more easier and " \
31                 "convinient. The AI will automatically help us to scan the image/video input from camera then process " \
32                 "it and displayed out a detected result for us to see."
33      page_exp = "There are total of six pages including the homepage. The Data Collection page will allowed you to " \
34                 "access and inspect all the data from different detection categories. The Image Object Detection page " \
35                 "allows you to upload one image and detect it. The Bulk Image Object Detection page allows you to " \
36                 "upload a folder of images and detect all. The Video Object Detection page allows you to upload a " \
37                 "video and detects the video. The Live Object Detection allows you to stream through your webcam or " \
38                 "wireless camera to do live detection."
39      st.write(proj_exp)
40      st.write(page_exp)
41
42  with col1:
43      st.header("Detection")
44      st.subheader("Bracket & Washer")
45      detect_exp = "The system developed is used to detect the bracket and washer on the exhaust mounting underneath the" \
46                   " vehicle. There will be a total of four classes which are 'Straight Bracket', 'Crooked Bracket', " \
47                   "'Washer Friction' and 'No Washer Friction'."
48      st.write(detect_exp)
49  with col2:
50      st.image("images/demo4.jpg", width=300)
51  with col3:
52      st.image("images/demo5.jpg", width=300)
53
```

```
with colu1:
    st.image("images/demo6.png", width=300)
with colu2:
    st.image("images/demo7.png", width=300)
with colu3:
    st.header("Equipments")
    equip_exp = "This project used Nvidia Jetson Orin Nano to run all the detection process and display the results " \
                "out on a website. The camera used is the Raspberry Pi NoIR Camera with the help of " \
                "Raspberry Pi Zero WH to setup the camera properly, the connection between the Pi Zero WH with the " \
                "Jetson Orin Nano is through wireless connection which is throught a local webserver."
    st.write(equip_exp)

with colum1:
    st.header("Software Stack")
    software_exp = "To developed this system, there are several software stacks that we used to help us to build this" \
                   " model system. Ultralytics is the one that provide us YOLOv9 model to train with our datasets. " \
                   "MotionEyes and Raspberry helped us to setup the NoIR camera along with Pi Zero WH. PyCharm is " \
                   "the platform where we use python to train our YOLOv9 model and design our webpage. MySQL is the" \
                   " database where we store all the collected data. Streamlit is the web server where we can do " \
                   "all the object detection and insert or display all the data collected."
    st.write(software_exp)
with colum2:
    st.image("images/demo8.png", width=500)
```

Appendix G: Data Collection

```python
import pandas as pd
import streamlit as st
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

import mysql.connector
from mysql.connector import Error

st.set_page_config(
    page_title="Data Collection",
    page_icon='📅',
    layout="wide",
    initial_sidebar_state="expanded"
)

# Fetching Data based on data table name
def fetch_data(table_name, vin, sequence_number, colour, type):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            query = f"SELECT No, Date, Type, `Vin No.`, `Seq No.`, Colour, TIME, `Straight Bracket`, `No Washer`, Status FROM {table_name}"
            conditions = []
            params = []
            if type:
```

```
            params = []
            if type:
                conditions.append("Type = %s")
                params.append(type)
            if vin:
                conditions.append("`Vin No.` = %s")
                params.append(vin)
            if sequence_number:
                conditions.append("`Seq No.` = %s")
                params.append(sequence_number)
            if colour and colour != 'All':
                conditions.append("Colour = %s")
                params.append(colour)
            if conditions:
                query += ' WHERE ' + ' AND '.join(conditions)
            cursor = db_conn.cursor()
            cursor.execute(query, params)
            df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
            cursor.close()
            db_conn.close()
            return df
    except Error as e:
        st.error(f"Error connecting to the database: {e}")
        return pd.DataFrame()

with st.sidebar:
    # Sidebar for user inputs to query the database
    st.header("Data Parameters")
    st.subheader("Insert for Data Query")
    detections = ['All', 'Image', 'Bulk Image', 'Video', 'Capture', 'Live']
```

```
    selected_detection = st.selectbox('Detection Methods', detections)
    detection_type = {'All': None, 'Image': 'Image', 'Bulk Image': 'Bulk Image', 'Video': 'Video',
                      'Capture': 'Capture', 'Live': 'Live'}.get(selected_detection)
    type = st.text_input('Type')
    vehicle_identification_number = st.text_input('Vehicle Identification Number')
    sequence_number = st.text_input('Sequence Number')
    colour = ['All', 'Red', 'Blue', 'Silver', 'White', 'Black']
    selected_colour = st.selectbox('Vehicle Colour', colour)

st.title("Historical Data")
clm1, clm2 = st.columns([0.5,0.35])

with clm1:
    st.subheader("Bracket & Washer")
    st.image("images/demo4.jpg", width=350)

data={}

data_image = fetch_data("data_image", vehicle_identification_number, sequence_number, selected_colour, type)
data_bulk_image = fetch_data("data_bulk_image", vehicle_identification_number, sequence_number, selected_colour, type)
data_video = fetch_data("data_video", vehicle_identification_number, sequence_number, selected_colour, type)
data_capture = fetch_data("data_capture", vehicle_identification_number, sequence_number, selected_colour, type)
data_live = fetch_data("data_live", vehicle_identification_number, sequence_number, selected_colour, type)

if selected_detection == "All":
    # Create column 1 for data and column 2 for pie chart
    st.header("Image Detection Data")
    c1, c2 = st.columns([0.7,0.35])
    st.header("Bulk Image Detection Data")
    co1, co2 = st.columns([0.7,0.35])
    st.header("Video Detection Data")
```

```
    col1, col2 = st.columns([0.7,0.35])
    st.header("Capture Detection Data")
    colum1, colum2 = st.columns([0.7, 0.35])
    st.header("Live Detection Data")
    column1, column2 = st.columns([0.7,0.35])

    ##############################     Data Image Object Detection     ##############################
    with c1:
        st.subheader('Queried Data')
        if not data_image.empty:
            st.dataframe(data_image)
        else:
            st.write("No data available based on the input parameters.")

    # Pie chart section for 'Status' distribution
    with c2:
        if not data_image.empty and 'Status' in data_image.columns and 'Date' in data_image.columns:
            # Parse the date column to extract months
            data_image['Month'] = pd.to_datetime(data_image['Date']).dt.strftime('%Y-%m')
            # Aggregate the data to count statuses by month
            status_counts_by_month = data_image.groupby(['Month', 'Status']).size().unstack(fill_value=0)
            # Plot the data as a bar chart
            fig, ax = plt.subplots()
            status_counts_by_month.plot(kind='bar', ax=ax)
            ax.set_ylabel('Count')
            ax.set_xlabel('Month')
            ax.set_title('Status Distribution by Month')
            ax.legend(title='Status')
            # Set yaxis whole number
            ax.yaxis.set_major_locator(MaxNLocator(integer=True))
            st.subheader('Status Distribution by Month')
```

```
            st.pyplot(fig)
        else:
            st.write("Data is empty or required columns are missing for the bar chart.")
    ##############################     (END) Data Image Object Detection (END)      ##############################

    ##############################     Data Bulk Image Object Detection     ##############################
    with co1:
        st.subheader('Queried Data')
        if not data_bulk_image.empty:
            st.dataframe(data_bulk_image)
        else:
            st.write("No data available based on the input parameters.")

    # Pie chart section for 'Status' distribution
    with co2:
        if not data_bulk_image.empty and 'Status' in data_bulk_image.columns and 'Date' in data_bulk_image.columns:
            # Parse the date column to extract months
            data_bulk_image['Month'] = pd.to_datetime(data_bulk_image['Date']).dt.strftime('%Y-%m')

            # Aggregate the data to count statuses by month
            status_counts_by_month = data_bulk_image.groupby(['Month', 'Status']).size().unstack(fill_value=0)

            # Plot the data as a bar chart
            fig, ax = plt.subplots()
            status_counts_by_month.plot(kind='bar', ax=ax)
            ax.set_ylabel('Count')
            ax.set_xlabel('Month')
            ax.set_title('Status Distribution by Month')
            ax.legend(title='Status')

            # Set yaxis whole number
```

```
            ax.yaxis.set_major_locator(MaxNLocator(integer=True))

            st.subheader('Status Distribution by Month')
            st.pyplot(fig)
        else:
            st.write("Data is empty or required columns are missing for the bar chart.")
    ##############################     (END) Data Bulk Image Object Detection (END)      ##############################

    ##############################     Data Video Object Detection     ##############################
    with col1:
        st.subheader('Queried Data')
        if not data_video.empty:
            st.dataframe(data_video)
        else:
            st.write("No data available based on the input parameters.")

    # Pie chart section for 'Status' distribution
    with col2:
        if not data_video.empty and 'Status' in data_video.columns and 'Date' in data_video.columns:
            # Parse the date column to extract months
            data_video['Month'] = pd.to_datetime(data_video['Date']).dt.strftime('%Y-%m')

            # Aggregate the data to count statuses by month
            status_counts_by_month = data_video.groupby(['Month', 'Status']).size().unstack(fill_value=0)

            # Plot the data as a bar chart
            fig, ax = plt.subplots()
            status_counts_by_month.plot(kind='bar', ax=ax)
            ax.set_ylabel('Count')
            ax.set_xlabel('Month')
            ax.set_title('Status Distribution by Month')
```

```
            ax.legend(title='Status')

            # Set yaxis whole number
            ax.yaxis.set_major_locator(MaxNLocator(integer=True))

            st.subheader('Status Distribution by Month')
            st.pyplot(fig)
        else:
            st.write("Data is empty or required columns are missing for the bar chart.")
    ##############################     (END) Data Video Object Detection (END)      ##############################

    ##############################     Data Capture Object Detection     ##############################
    with colum1:
        st.subheader('Queried Data')
        if not data_capture.empty:
            st.dataframe(data_capture)
        else:
            st.write("No data available based on the input parameters.")

    # Pie chart section for 'Status' distribution
    with colum2:
        if not data_capture.empty and 'Status' in data_capture.columns and 'Date' in data_capture.columns:
            # Parse the date column to extract months
            data_capture['Month'] = pd.to_datetime(data_capture['Date']).dt.strftime('%Y-%m')

            # Aggregate the data to count statuses by month
            status_counts_by_month = data_capture.groupby(['Month', 'Status']).size().unstack(fill_value=0)

            # Plot the data as a bar chart
            fig, ax = plt.subplots()
```

```
            status_counts_by_month.plot(kind='bar', ax=ax)
            ax.set_ylabel('Count')
            ax.set_xlabel('Month')
            ax.set_title('Status Distribution by Month')
            ax.legend(title='Status')

            # Set yaxis whole number
            ax.yaxis.set_major_locator(MaxNLocator(integer=True))

            st.subheader('Status Distribution by Month')
            st.pyplot(fig)
        else:
            st.write("Data is empty or required columns are missing for the bar chart.")
    ##############################     (END) Data Capture Object Detection (END)      ##############################

    ##############################     Live Object Detection     ##############################
    with column1:
        st.subheader('Queried Data')
        if not data_live.empty:
            st.dataframe(data_live)
        else:
            st.write("No data available based on the input parameters.")

    # Pie chart section for 'Status' distribution
    with column2:
        if not data_live.empty and 'Status' in data_live.columns and 'Date' in data_live.columns:
            # Parse the date column to extract months
            data_live['Month'] = pd.to_datetime(data_live['Date']).dt.strftime('%Y-%m')

            # Aggregate the data to count statuses by month
            status_counts_by_month = data_live.groupby(['Month', 'Status']).size().unstack(fill_value=0)
```

```
245
246             # Plot the data as a bar chart
247             fig, ax = plt.subplots()
248             status_counts_by_month.plot(kind='bar', ax=ax)
249             ax.set_ylabel('Count')
250             ax.set_xlabel('Month')
251             ax.set_title('Status Distribution by Month')
252             ax.legend(title='Status')
253
254             # Set yaxis whole number
255             ax.yaxis.set_major_locator(MaxNLocator(integer=True))
256
257             st.subheader('Status Distribution by Month')
258             st.pyplot(fig)
259         else:
260             st.write("Data is empty or required columns are missing for the bar chart.")
261     ##############################     (END) Live Object Detection (END)     ##############################
262
263 else:
264     data[selected_detection] = fetch_data(f"data_{selected_detection.lower().replace(' ', '_')}",
265                                           vehicle_identification_number, sequence_number, selected_colour, type)
266     for detection, df in data.items():
267         st.header(f"{detection} Detection Data")
268         c1, c2 = st.columns([0.7, 0.35])
269
270         with c1:
271             st.subheader('Queried Data')
272             if not df.empty:
273                 st.dataframe(df)
274             else:
```

```
                st.write("No data available based on the input parameters.")

        with c2:
            if not df.empty and 'Status' in df.columns and 'Date' in df.columns:
                # Parse the date column to extract months
                df['Month'] = pd.to_datetime(df['Date']).dt.strftime('%Y-%m')

                # Aggregate the data to count statuses by month
                status_counts_by_month = df.groupby(['Month', 'Status']).size().unstack(fill_value=0)

                # Plot the data as a bar chart
                fig, ax = plt.subplots()
                status_counts_by_month.plot(kind='bar', ax=ax)
                ax.set_ylabel('Count')
                ax.set_xlabel('Month')
                ax.set_title('Status Distribution by Month')
                ax.legend(title='Status')

                # Set yaxis whole number
                ax.yaxis.set_major_locator(MaxNLocator(integer=True))

                st.subheader('Status Distribution by Month')
                st.pyplot(fig)
            else:
                st.write("Data is empty or required columns are missing for the bar chart.")

# First, combine the data from all tables into a single DataFrame
combined_data = pd.concat([data_image, data_bulk_image, data_video, data_capture, data_live], ignore_index=True)

with clm2:
    if not combined_data.empty and 'Status' in combined_data.columns and 'Date' in combined_data.columns:
```

```
        # Parse the date column to extract months
        combined_data['Month'] = pd.to_datetime(combined_data['Date']).dt.strftime('%Y-%m')
        # Aggregate the data to count statuses by month
        status_counts_by_month = combined_data.groupby(['Month', 'Status']).size().unstack(fill_value=0)
        # Plot the data as a bar chart
        fig, ax = plt.subplots()
        status_counts_by_month.plot(kind='bar', ax=ax)
        ax.set_ylabel('Count')
        ax.set_xlabel('Month')
        ax.set_title('Combination Status Distribution by Month')
        ax.legend(title='Status')

        # Set yaxis whole number
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))

        st.subheader('Status Distribution by Month')
        st.pyplot(fig)
    else:
        st.write("Data is empty or required columns are missing for the bar chart.")
```

Appendix H: Image Object Detection

```python
import os
import zipfile
import tempfile
import pandas as pd
from PIL import Image
import streamlit as st
from ultralytics import YOLO
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

import mysql.connector
from mysql.connector import Error

# Specified Model Path
model_path = "weights/best.pt"

# Loading of Trained YOLOv9 Model
try:
    model = YOLO(model_path)
except Exception as ex:
    st.error(f"Unable to load YOLO model. Check Model Path: {model_path}")
    st.error(ex)

# Page Layout Setting
st.set_page_config(
    page_title="Image Object Detection",
    page_icon='🖼️',
    layout="wide",
    initial_sidebar_state="expanded"
)
```

```python

# Function to fetch data of data_image from MySQL database
def fetch_data(vin, sequence_number, colour, type):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            # Change query "data_image" to your table name
            query = "SELECT No, Date, Type, `Vin No.`, `Seq No.`, Colour, TIME, `Straight Bracket`, `No Washer`, Status FROM data_image"
            conditions = []
            params = []
            if type:
                conditions.append("Type = %s")
                params.append(type)
            if vin:
                conditions.append("`Vin No.` = %s")
                params.append(vin)
            if sequence_number:
                conditions.append("`Seq No.` = %s")
                params.append(sequence_number)
            if colour and colour != 'All':
                conditions.append("Colour = %s")
                params.append(colour)
            if conditions:
                query += ' WHERE ' + ' AND '.join(conditions)
```

```
            cursor = db_conn.cursor()
            cursor.execute(query, params)
            df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
            cursor.close()
            db_conn.close()
            return df
    except Error as e:
        st.error(f"Error connecting to the database: {e}")
        return pd.DataFrame()

# Function to insert data of data_image to MySQL database
def insert_data(type, vin, seq_no, colour, bracket, washer, status):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        # with mysql.connector.connect(**db_config) as db_conn:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            query = """
                    INSERT INTO data_image (Type, `Vin No.`, `Seq No.`, Colour, `Straight Bracket`, `No Washer`, Status, `Date`)
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                    """
            values = (type, vin, seq_no, colour, bracket, washer, status, datetime.now())
            cursor = db_conn.cursor()
            cursor.execute(query, values)
            db_conn.commit()
```

```
            st.success("Data inserted successfully!")
            cursor.close()
            db_conn.close()
    except Error as e:
        st.error(f"Error connecting to the database: {e}")

# Create Side Bar
with st.sidebar:
    st.header("Upload Your Image")
    source_img = st.file_uploader("Choose an Image...", type=("jpg", "jpeg", "png", 'bmp', 'webp'))
    DEMO_IMAGE = 'images/demo.jpg'
    confidence = float(st.slider("Set Confidence Level", 25, 100, 40)) / 100 # Model Confidence Option
    detect_button = st.button('Detect Image')
    st.markdown("---")

    # Sidebar for user inputs to query the database
    st.header("Data Parameters")
    st.subheader("Insert for Data Query")
    type = st.text_input('Type')
    vehicle_identification_number = st.text_input('Vehicle Identification Number')
    sequence_number = st.text_input('Sequence Number')
    colour = ['All', 'Red', 'Blue', 'Silver', 'White', 'Black']
    selected_colour = st.selectbox('Vehicle Colour', colour)

    st.subheader("Insert Data into Database")
    with st.form(key='insert form'):
        type_input = st.text_input('Type (Insert)')
        vin_input = st.text_input('Vehicle Identification Number (Insert)')
        seq_no_input = st.text_input('Sequence Number (Insert)')
        colour_input = st.selectbox('Vehicle Colour (Insert)', ['Red', 'Blue', 'Silver', 'White', 'Black'])
        bracket_input = st.text_input('Straight Bracket')
```

```
        washer_input = st.text_input('No Washer')
        status_input = st.selectbox('Status', ['ok', 'not ok'])

        submit_button = st.form_submit_button('Submit')
        if submit_button:
            insert_data(type_input, vin_input, seq_no_input, colour_input, bracket_input, washer_input, status_input)

# Main Page
st.title("Image Object Detection")
st.write("Choose your image and set the confidence level. Finally, click detect image.")

# Fetch data based on inputs
data = fetch_data(vehicle_identification_number, sequence_number, selected_colour, type)

# Create column 1 for data and column 2 for pie chart
column1, column2 = st.columns([0.7,0.35])
st.markdown("---")

# Creating two columns on the main page
col1, col2 = st.columns(2)

# Display the fetched data
with column1:
    st.subheader('Queried Data')
    if not data.empty:
        st.dataframe(data)
    else:
        st.write("No data available based on the input parameters.")

```

```
# Pie chart section for 'Status' distribution
with column2:
    if not data.empty and 'Status' in data.columns and 'Date' in data.columns:
        # Parse the date column to extract months
        data['Month'] = pd.to_datetime(data['Date']).dt.strftime('%Y-%m')

        # Aggregate the data to count statuses by month
        status_counts_by_month = data.groupby(['Month', 'Status']).size().unstack(fill_value=0)

        # Plot the data as a bar chart
        fig, ax = plt.subplots()
        status_counts_by_month.plot(kind='bar', ax=ax)
        ax.set_ylabel('Count')
        ax.set_xlabel('Month')
        ax.set_title('Status Distribution by Month')
        ax.legend(title='Status')

        # Set yaxis whole number
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))

        st.subheader('Status Distribution by Month')
        st.pyplot(fig)
    else:
        st.write("Data is empty or required columns are missing for the bar chart.")

# Creating two columns on the main page
with col1:
    if not source_img:
        uploaded_image = Image.open(DEMO_IMAGE)
        st.image(DEMO_IMAGE, caption="Demo Image", use_column_width=True)

```

```
    if source_img:
        uploaded_image = Image.open(source_img)
        st.image(source_img, caption="Image Uploaded", use_column_width=True)

# Detect image and display at column 2 after Detect Image is pressed
if detect_button:
    res = model.predict(uploaded_image, conf=confidence)
    boxes = res[0].boxes
    res_plotted = res[0].plot()[:, :, ::-1]
    with col2:
        st.image(res_plotted, caption='Image Detected Results', use_column_width=True)
        try:
            with st.expander("Detection Results"):
                for box in boxes:
                    st.write(box.xywh)
        except Exception as ex:
            st.write("No image is uploaded yet!")

        with tempfile.NamedTemporaryFile(suffix='.jpg', delete=False) as temp_file:
            temp_file_path = temp_file.name
            Image.fromarray(res_plotted).save(temp_file_path)
            # Provide a download button for the temporary file
            with open(temp_file_path, "rb") as file:
                btn = st.download_button(
                    label="Download Image",
                    data=file,
                    file_name="detected_img.jpg",
                    mime="image/jpg"
                )
        # Delete the temporary file after the download button is clicked
        os.unlink(temp_file_path)
```

Appendix I: Bulk Image Object Detection

```python
import os
import zipfile
import tempfile
import pandas as pd
from PIL import Image
import streamlit as st
from ultralytics import YOLO
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

import mysql.connector
from mysql.connector import Error

# Specified Model Path
model_path = "weights/best.pt"

# Loading of Trained YOLOv9 Model
try:
    model = YOLO(model_path)
except Exception as ex:
    st.error(f"Unable to load YOLO model. Check Model Path: {model_path}")
    st.error(ex)

# Page Layout Setting
st.set_page_config(
    page_title="Bulk Image Object Detection",
    page_icon='🖼️',
    layout="wide",
    initial_sidebar_state="expanded"
)
```

```
32
33  # Function to fetch data of data_bulk_image from MySQL database
34  def fetch_data(vin, sequence_number, colour, type):
35      db_config = {
36          'host': 'localhost',    # dont change
37          'user': 'root',         # change user
38          'password': '',         # put password
39          'database': 'data1'     # database name
40      }
41      try:
42          db_conn = mysql.connector.connect(**db_config)
43          if db_conn.is_connected():
44              # Change query "data_bulk_image" to your table name
45              query = "SELECT No, Date, Type, `Vin No.`, `Seq No.`, Colour, TIME, `Straight Bracket`, `No Washer`, Status FROM data_bulk_image"
46              conditions = []
47              params = []
48              if type:
49                  conditions.append("Type = %s")
50                  params.append(type)
51              if vin:
52                  conditions.append("`Vin No.` = %s")
53                  params.append(vin)
54              if sequence_number:
55                  conditions.append("`Seq No.` = %s")
56                  params.append(sequence_number)
57              if colour and colour != 'All':
58                  conditions.append("Colour = %s")
59                  params.append(colour)
60              if conditions:
61                  query += ' WHERE ' + ' AND '.join(conditions)
62              cursor = db_conn.cursor()
```

```
            cursor.execute(query, params)
            df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
            cursor.close()
            db_conn.close()
            return df
    except Error as e:
        st.error(f"Error connecting to the database: {e}")
        return pd.DataFrame()

# Function to insert data of data_bulk_image to MySQL database
def insert_data(type, vin, seq_no, colour, bracket, washer, status):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        # with mysql.connector.connect(**db_config) as db_conn:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            query = """
                    INSERT INTO data_bulk_image (Type, `Vin No.`, `Seq No.`, Colour, `Straight Bracket`, `No Washer`, Status, `Date`)
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                    """
            values = (type, vin, seq_no, colour, bracket, washer, status, datetime.now())
            cursor = db_conn.cursor()
            cursor.execute(query, values)
            db_conn.commit()
            st.success("Data inserted successfully!")
            cursor.close()
```

```
            db_conn.close()
    except Error as e:
        st.error(f"Error connecting to the database: {e}")

# Create Side Bar
with st.sidebar:
    st.header("Upload Your Image")
    source_imgs = st.file_uploader("Choose an Image...", type=("jpg", "jpeg", "png", 'bmp', 'webp'), accept_multiple_files=True)
    DEMO1_PATH = 'images/demo1.jpg'
    DEMO2_PATH = 'images/demo2.jpg'
    DEMO3_PATH = 'images/demo3.jpg'
    confidence = float(st.slider("Set Confidence Level", 25, 100, 40)) / 100  # Model Confidence Option

    # Sidebar for user inputs to query the database
    st.header("Data Parameters")
    st.subheader("Insert for Data Query")
    type = st.text_input('Type')
    vehicle_identification_number = st.text_input('Vehicle Identification Number')
    sequence_number = st.text_input('Sequence Number')
    colour = ['All', 'Red', 'Blue', 'Silver', 'White', 'Black']
    selected_colour = st.selectbox('Vehicle Colour', colour)

    st.subheader("Insert Data into Database")
    with st.form(key='insert form'):
        type_input = st.text_input('Type (Insert)')
        vin_input = st.text_input('Vehicle Identification Number (Insert)')
        seq_no_input = st.text_input('Sequence Number (Insert)')
        colour_input = st.selectbox('Vehicle Colour (Insert)', ['Red', 'Blue', 'Silver', 'White', 'Black'])
        bracket_input = st.text_input('Straight Bracket')
        washer_input = st.text_input('No Washer')
        status_input = st.selectbox('Status', ['ok', 'not ok'])
```

```
125
126         submit_button = st.form_submit_button('Submit')
127         if submit_button:
128             insert_data(type_input, vin_input, seq_no_input, colour_input, bracket_input, washer_input, status_input)
129
130 # Main Page
131 st.title("Bulk Image Object Detection")
132 st.write("Choose your image and set the confidence level. Finally, click detect image.")
133
134 # List to store many paths of detected images
135 detected_image_paths = []
136
137 # Fetch data based on inputs
138 data = fetch_data(vehicle_identification_number, sequence_number, selected_colour, type)
139
140 # Create column 1 for data and column 2 for pie chart
141 column1, column2 = st.columns([0.7,0.35])
142 st.markdown("---")
143
144 # Creating 1 column for detect image button and 1 column for download zip
145 c1, c2 = st.columns(2)
146
147 # Creating 1 column for ori image and 1 column for detected image
148 col1, col2 = st.columns(2)
149
150 # Display the fetched data
151 with column1:
152     st.subheader('Queried Data')
153     if not data.empty:
154         st.dataframe(data)
```

```
    else:
        st.write("No data available based on the input parameters.")

# Pie chart section for 'Status' distribution
with column2:
    if not data.empty and 'Status' in data.columns and 'Date' in data.columns:
        # Parse the date column to extract months
        data['Month'] = pd.to_datetime(data['Date']).dt.strftime('%Y-%m')

        # Aggregate the data to count statuses by month
        status_counts_by_month = data.groupby(['Month', 'Status']).size().unstack(fill_value=0)

        # Plot the data as a bar chart
        fig, ax = plt.subplots()
        status_counts_by_month.plot(kind='bar', ax=ax)
        ax.set_ylabel('Count')
        ax.set_xlabel('Month')
        ax.set_title('Status Distribution by Month')
        ax.legend(title='Status')

        # Set yaxis whole number
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))

        st.subheader('Status Distribution by Month')
        st.pyplot(fig)
    else:
        st.write("Data is empty or required columns are missing for the bar chart.")

# Column 1 for uploading demo images or uploaded images
with col1:
    if not source_imgs:
```

```
        demo_image1 = Image.open(DEMO1_PATH)
        demo_image2 = Image.open(DEMO2_PATH)
        demo_image3 = Image.open(DEMO3_PATH)
        st.image(DEMO1_PATH, caption="Demo Image", width = 400)
        st.image(DEMO2_PATH, caption="Demo Image", width=400)
        st.image(DEMO3_PATH, caption="Demo Image", width=400)
        demo_file_paths = [DEMO1_PATH, DEMO2_PATH, DEMO3_PATH]
    else:
        for source_img in source_imgs:
            uploaded_image = Image.open(source_img)
            st.image(uploaded_image, caption="Image Uploaded", width = 400)

# Put detect image button at c1
with c1:
    st.write("Detect all the images")
    detect_button = st.button('Detect Image')

# Do object Detection on demo images or uploaded images and display them one by one at column 2.
# Create temp path of each image and add them into one file paths for zip download
if detect_button:
    for source_img in source_imgs:
        uploaded_image = Image.open(source_img)
        res = model.predict(uploaded_image, conf = confidence)
        boxes = res[0].boxes
        res_plotted = res[0].plot()[:, :, ::-1]
        with col2:
            st.image(res_plotted, caption = 'Image Detected Results', width = 400)
            with tempfile.NamedTemporaryFile(suffix='.jpg', delete=False) as temp_file:
                temp_file_path = temp_file.name
                Image.fromarray(res_plotted).save(temp_file_path)
                detected_image_paths.append(temp_file_path)
```

```
    if not source_imgs:
        res1 = model.predict(demo_image1, conf=confidence)
        res2 = model.predict(demo_image2, conf=confidence)
        res3 = model.predict(demo_image3, conf=confidence)
        boxes1 = res1[0].boxes
        boxes2 = res2[0].boxes
        boxes3 = res3[0].boxes
        res_plotted1 = res1[0].plot()[:, :, ::-1]
        res_plotted2 = res2[0].plot()[:, :, ::-1]
        res_plotted3 = res3[0].plot()[:, :, ::-1]
        with col2:
            st.image(res_plotted1, caption='Demo Detected Results', width = 400)
            st.image(res_plotted2, caption='Demo Detected Results', width=400)
            st.image(res_plotted3, caption='Demo Detected Results', width=400)
        with tempfile.NamedTemporaryFile(suffix='.jpg', delete=False) as temp_file1, \
                tempfile.NamedTemporaryFile(suffix='.jpg', delete=False) as temp_file2, \
                tempfile.NamedTemporaryFile(suffix='.jpg', delete=False) as temp_file3:
            temp_file_path1 = temp_file1.name
            temp_file_path2 = temp_file2.name
            temp_file_path3 = temp_file3.name
            Image.fromarray(res_plotted1).save(temp_file_path1)
            Image.fromarray(res_plotted2).save(temp_file_path2)
            Image.fromarray(res_plotted3).save(temp_file_path3)
            detected_image_paths.extend([temp_file_path1, temp_file_path2, temp_file_path3])

# If there is a detection of paths and after the detection and display finish,
# it will allow us to download the zip file with the button appear.
if detected_image_paths:
    with c2:
        st.write("Download all detected results")
        with tempfile.TemporaryDirectory() as temp_dir:
```

```
248             zip_file_path = os.path.join(temp_dir, "detected_images.zip")
249             with zipfile.ZipFile(zip_file_path, "w") as zipf:
250                 for img_path in detected_image_paths:
251                     zipf.write(img_path, os.path.basename(img_path))
252                     os.unlink(img_path)  # Delete the temporary file
253             with open(zip_file_path, "rb") as file:
254                 st.download_button(
255                     label="Download Results",
256                     data=file,
257                     file_name="detected_images.zip",
258                     mime="application/zip"
259                 )
260
261
```

Appendix J: Video Object Detection

```python
import cv2
import tempfile
import pandas as pd
import streamlit as st
from ultralytics import YOLO
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

import mysql.connector
from mysql.connector import Error

# Specified Model Path
model_path = "weights/best.pt"

# Loading of Trained YOLOv9 Model
try:
    model = YOLO(model_path)
except Exception as ex:
    st.error(f"Unable to load YOLO model. Check Model Path: {model_path}")
    st.error(ex)

# Page Layout Setting
st.set_page_config(
    page_title="Video Object Detection",
    page_icon='',
    layout="wide",
    initial_sidebar_state="expanded"
)

# Function for drawing the frame if detected anything
```

```python
def draw_boxes(frame, boxes, threshold):
    # Implementation to draw bounding boxes on the image
    # You need to implement this function
    return frame

# Function for using the model to detect
def detect_objects(image):
    # Detection implementation using YOLO model
    detections = model(image, conf = confidence)
    # Format detections if necessary
    return detections

# Function to fetch data of data_video from MySQL database
def fetch_data(vin, sequence_number, colour, type):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            # Change query "data_video" to your table name
            query = "SELECT No, Date, Type, `Vin No.`, `Seq No.`, Colour, TIME, `Straight Bracket`, `No Washer`, Status FROM data_video"
            conditions = []
            params = []
            if type:
                conditions.append("Type = %s")
                params.append(type)
            if vin:
```

```
                conditions.append("`Vin No.` = %s")
                params.append(vin)
            if sequence_number:
                conditions.append("`Seq No.` = %s")
                params.append(sequence_number)
            if colour and colour != 'All':
                conditions.append("Colour = %s")
                params.append(colour)
            if conditions:
                query += ' WHERE ' + ' AND '.join(conditions)
            cursor = db_conn.cursor()
            cursor.execute(query, params)
            df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
            cursor.close()
            db_conn.close()
            return df
    except Error as e:
        st.error(f"Error connecting to the database: {e}")
        return pd.DataFrame()

# Function to insert data of data_video to MySQL database
def insert_data(type, vin, seq_no, colour, bracket, washer, status):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        # with mysql.connector.connect(**db_config) as db_conn:
        db_conn = mysql.connector.connect(**db_config)
```

```
        if db_conn.is_connected():
            query = """
                    INSERT INTO data_video (Type, `Vin No.`, `Seq No.`, Colour, `Straight Bracket`, `No Washer`, Status, `Date`)
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                    """
            values = (type, vin, seq_no, colour, bracket, washer, status, datetime.now())
            cursor = db_conn.cursor()
            cursor.execute(query, values)
            db_conn.commit()
            st.success("Data inserted successfully!")
            cursor.close()
            db_conn.close()
    except Error as e:
        st.error(f"Error connecting to the database: {e}")

# Create Side Bar
with st.sidebar:
    st.header("Upload and Configurations")
    source_vid = st.file_uploader("Upload a Video...", type=["mp4", "mov", 'avi', 'asf', 'm4v'])
    DEMO_VIDEO = 'videos/demo.mp4'
    tffile = tempfile.NamedTemporaryFile(suffix='.mp4', delete=False)
    confidence = float(st.slider("Set Confidence Level", 25, 100, 40)) / 100  # Model Confidence Option

    # Upload Demo/Uploaded Video
    if not source_vid:
        vid = cv2.VideoCapture(DEMO_VIDEO)
        tffile.name = DEMO_VIDEO

        st.sidebar.text('Demo Input Video')
        st.sidebar.video(tffile.name)
    else:
```

```
        tffile.write(source_vid.read())
        vid = cv2.VideoCapture(tffile.name)

        st.sidebar.text('Your Input Video')
        st.sidebar.video(tffile.name)
    st.markdown("---")

    # Sidebar for user inputs to query the database
    st.header("Data Parameters")
    st.subheader("Insert for Data Query")
    type = st.text_input('Type')
    vehicle_identification_number = st.text_input('Vehicle Identification Number')
    sequence_number = st.text_input('Sequence Number')
    colour = ['All', 'Red', 'Blue', 'Silver', 'White', 'Black']
    selected_colour = st.selectbox('Vehicle Colour', colour)

    st.subheader("Insert Data into Database")
    with st.form(key='insert form'):
        type_input = st.text_input('Type (Insert)')
        vin_input = st.text_input('Vehicle Identification Number (Insert)')
        seq_no_input = st.text_input('Sequence Number (Insert)')
        colour_input = st.selectbox('Vehicle Colour (Insert)', ['Red', 'Blue', 'Silver', 'White', 'Black'])
        bracket_input = st.text_input('Straight Bracket')
        washer_input = st.text_input('No Washer')
        status_input = st.selectbox('Status', ['ok', 'not ok'])

        submit_button = st.form_submit_button('Submit')
        if submit_button:
            insert_data(type_input, vin_input, seq_no_input, colour_input, bracket_input, washer_input, status_input)

# Main Page
```

```
st.title("Video Object Detection")
st.write("Set confidence level and upload your video. Finally, click detect image.")

# Fetch data based on inputs
data = fetch_data(vehicle_identification_number, sequence_number, selected_colour, type)

# Create column 1 for data and column 2 for pie chart
column1, column2 = st.columns([0.7, 0.35])

# Display the fetched data
with column1:
    st.subheader('Queried Data')
    if not data.empty:
        st.dataframe(data)
    else:
        st.write("No data available based on the input parameters.")

# Pie chart section for 'Status' distribution
with column2:
    if not data.empty and 'Status' in data.columns and 'Date' in data.columns:
        # Parse the date column to extract months
        data['Month'] = pd.to_datetime(data['Date']).dt.strftime('%Y-%m')

        # Aggregate the data to count statuses by month
        status_counts_by_month = data.groupby(['Month', 'Status']).size().unstack(fill_value=0)

        # Plot the data as a bar chart
        fig, ax = plt.subplots()
        status_counts_by_month.plot(kind='bar', ax=ax)
        ax.set_ylabel('Count')
        ax.set_xlabel('Month')
```

```
        ax.set_title('Status Distribution by Month')
        ax.legend(title='Status')

        # Set yaxis whole number
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))

        st.subheader('Status Distribution by Month')
        st.pyplot(fig)
    else:
        st.write("Data is empty or required columns are missing for the bar chart.")

st.markdown("---")

# Create column 1 for data and column 2 for pie chart
placeholder = st.columns(2)

with placeholder[0]:
    detect_button = st.button('Detect Video')
    if not source_vid:
        demo_text = st.text("Demo Input Video")
    else:
        input_text = st.text("Your Input Video")

with placeholder[1]:
    st.write("Detected Result Video")

# Assign st.empty to place hold at col1 and col2
col1 = placeholder[0].empty()
col2 = placeholder[1].empty()

# Upload Demo/Uploaded Video
```

```
if not source_vid:
    vid = cv2.VideoCapture(DEMO_VIDEO)
    tffile.name = DEMO_VIDEO

    if demo_text:
        col1.video(tffile.name)
else:
    tffile.write(source_vid.read())
    vid = cv2.VideoCapture(tffile.name)

    if input_text:
        col1.video(tffile.name)

# Upload video to column 1 to fix the display size
if detect_button:
    while True:
        ret, img = vid.read()
        if ret:
            detections = detect_objects(img)
            res_plotted = detections[0].plot()
            output_frame = cv2.cvtColor(res_plotted, cv2.COLOR_BGR2RGB)
            col2.image(res_plotted, channels="BGR", use_column_width=True)
```

Appendix K: Capture Object Detection

```python
import streamlit as st
import requests
from PIL import Image
from io import BytesIO
import pandas as pd
from ultralytics import YOLO
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

import mysql.connector
from mysql.connector import Error

# Specified Model Path
model_path = "weights/best.pt"

# Loading of Trained YOLOv9 Model
try:
    model = YOLO(model_path)
except Exception as ex:
    st.error(f"Unable to load YOLO model. Check Model Path: {model_path}")
    st.error(ex)

# Page Layout Setting
st.set_page_config(
    page_title="NoIR Camera Object Detection",
    page_icon='📷',
    layout="wide",
    initial_sidebar_state="expanded"
)

```

```python
st.title("Camera Capture Object Detection")
column1, column2 = st.columns([0.7,0.35])
st.markdown("---")
co1, co2 = st.columns(2)
c1, c2 = st.columns(2)
col1, col2 = st.columns(2)

with co1:
    st.header("Camera 1")
with co2:
    st.header("Camera 2")


# Function to fetch data of data_capture from MySQL database
def fetch_data(vin, sequence_number, colour, type):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            # Change query "data_image" to your table name
            query = "SELECT No, Date, Type, `Vin No.`, `Seq No.`, Colour, TIME, `Straight Bracket`, `No Washer`, Status FROM data_capture"
            conditions = []
            params = []
            if type:
                conditions.append("Type = %s")
                params.append(type)
```

```
            if vin:
                conditions.append("`Vin No.` = %s")
                params.append(vin)
            if sequence_number:
                conditions.append("`Seq No.` = %s")
                params.append(sequence_number)
            if colour and colour != 'All':
                conditions.append("Colour = %s")
                params.append(colour)
            if conditions:
                query += ' WHERE ' + ' AND '.join(conditions)
            cursor = db_conn.cursor()
            cursor.execute(query, params)
            df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
            cursor.close()
            db_conn.close()
            return df
    except Error as e:
        st.error(f"Error connecting to the database: {e}")
        return pd.DataFrame()

# Function to insert data of data_capture to MySQL database
def insert_data(type, vin, seq_no, colour, bracket, washer, status):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        # with mysql.connector.connect(**db_config) as db_conn:
```

```
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            query = """
                    INSERT INTO data_capture (Type, `Vin No.`, `Seq No.`, Colour, `Straight Bracket`, `No Washer`, Status, `Date`)
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                    """
            values = (type, vin, seq_no, colour, bracket, washer, status, datetime.now())
            cursor = db_conn.cursor()
            cursor.execute(query, values)
            db_conn.commit()
            st.success("Data inserted successfully!")
            cursor.close()
            db_conn.close()
    except Error as e:
        st.error(f"Error connecting to the database: {e}")

# Function to send signal to cam 1 to capture image
def cam_1_capture():
    try:
        # Send request to Flask server on Raspberry Pi to capture image
        response = requests.get('http://192.168.217.201:5000/capture')
        if response.status_code == 200 and response.json()['status'] == 'success':
            with c1:
                st.success("Image captured successfully!")
            # Fetch the captured image
            image_response = requests.get('http://192.168.217.201:5000/captured_image.jpg', stream=True)
            img = Image.open(BytesIO(image_response.content))
            return img
        else:
            st.error("Failed to capture image!")
    except Exception as e:
```

```
        st.error(f"Error: {e}")

# Function to send signal to cam 2 to capture image
def cam_2_capture():
    try:
        # Send request to Flask server on Raspberry Pi to capture image
        response = requests.get('http://192.168.217.210:5010/capture')
        if response.status_code == 200 and response.json()['status'] == 'success':
            with c2:
                st.success("Image captured successfully!")
            # Fetch the captured image
            image_response = requests.get('http://192.168.217.210:5010/captured_image.jpg', stream=True)
            img = Image.open(BytesIO(image_response.content))
            return img
        else:
            st.error("Failed to capture image!")
    except Exception as e:
        st.error(f"Error: {e}")

# Function to detect the image and display the image and detected results
def display_detection_results(image, cam_results):
    res = model.predict(image, conf=confidence)
    boxes = res[0].boxes
    res_plotted = res[0].plot()[:, :, ::-1]
    st.image(res_plotted, caption='Detection Results', use_column_width=True)
    detected_classes = [(model.names[int(c)], conf) for r in res for c, conf in zip(r.boxes.cls, r.boxes.conf)]
    st.subheader(cam_results)
    st.write("Detected Classes with Confidence Scores:")
    for cls, conf in detected_classes:
        st.info(f"{cls}: {conf:.2f}")

```

```
    # Check conditions for OK or NOT OK
    detected_classes_set = set(cls for cls, _ in detected_classes)
    # classy = set(["straight bracket", "no washer friction"])  # Method 1 test the below statement
    # classy= {"straight bracket", "no washer friction"}        # Method 2 test the below statement

    if "crooked bracket" in detected_classes_set or "washer friction" in detected_classes_set:
        st.error("  NOT OK", icon="🚨")
    elif detected_classes_set == {"straight bracket", "no washer friction"}:
        st.success("    OK", icon="✅")
    else:
        st.warning("Unknown", icon="⚠️")

with st.sidebar:
    st.header("Insert Parameters")
    # Model Confidence Option
    confidence = float(st.slider("Set Confidence Level", 25, 100, 40)) / 100
    capture_button = st.toggle('Capture Image')
    detect_button = st.sidebar.button("Detect Images")
    st.sidebar.markdown("---")
    st.header("Data Parameters")
    st.subheader("Insert for Data Query")
    type = st.text_input('Type')
    vehicle_identification_number = st.text_input('Vehicle Identification Number')
    sequence_number = st.text_input('Sequence Number')
    colour = ['All', 'Red', 'Blue', 'Silver', 'White', 'Black']
    selected_colour = st.selectbox('Vehicle Colour', colour)

    st.subheader("Insert Data into Database")
    with st.form(key='insert form'):
        type_input = st.text_input('Type (Insert)')
```

```
        vin_input = st.text_input('Vehicle Identification Number (Insert)')
        seq_no_input = st.text_input('Sequence Number (Insert)')
        colour_input = st.selectbox('Vehicle Colour (Insert)', ['Red', 'Blue', 'Silver', 'White', 'Black'])
        bracket_input = st.text_input('Straight Bracket')
        washer_input = st.text_input('No Washer')
        status_input = st.selectbox('Status', ['ok', 'not ok'])

        submit_button = st.form_submit_button('Submit')
        if submit_button:
            insert_data(type_input, vin_input, seq_no_input, colour_input, bracket_input, washer_input, status_input)

# If button capture then go to display_detection_results
if capture_button:
    img1 = cam_1_capture()
    img2 = cam_2_capture()
    if img1 and img2:
        with col1:
            st.image(img1, caption='Captured Image', use_column_width=True)
        with col2:
            st.image(img2, caption='Captured Image', use_column_width=True)

    if detect_button:
        cam_results1 = "Camera 1 Results"
        cam_results2 = "Camera 2 Results"
        with col1:
            display_detection_results(img1, cam_results1)
        with col2:
            display_detection_results(img2, cam_results2)

# Fetch data based on inputs
data = fetch_data(vehicle_identification_number, sequence_number, selected_colour, type)
```

```
217
218 # Display the fetched data
219 with column1:
220     st.subheader('Queried Data')
221     if not data.empty:
222         st.dataframe(data)
223     else:
224         st.write("No data available based on the input parameters.")
225
226 # Pie chart section for 'Status' distribution
227 with column2:
228     if not data.empty and 'Status' in data.columns and 'Date' in data.columns:
229         # Parse the date column to extract months
230         data['Month'] = pd.to_datetime(data['Date']).dt.strftime('%Y-%m')
231
232         # Aggregate the data to count statuses by month
233         status_counts_by_month = data.groupby(['Month', 'Status']).size().unstack(fill_value=0)
234
235         # Plot the data as a bar chart
236         fig, ax = plt.subplots()
237         status_counts_by_month.plot(kind='bar', ax=ax)
238         ax.set_ylabel('Count')
239         ax.set_xlabel('Month')
240         ax.set_title('Status Distribution by Month')
241         ax.legend(title='Status')
242         # Set yaxis whole number
243         ax.yaxis.set_major_locator(MaxNLocator(integer=True))
244         st.subheader('Status Distribution by Month')
245         st.pyplot(fig)
246     else:
247         st.write("Data is empty or required columns are missing for the bar chart.")
```

Appendix L: Live Object Detection

```python
import cv2
import pandas as pd
import streamlit as st
from ultralytics import YOLO
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

import mysql.connector
from mysql.connector import Error

# Specified Model Path
model_path = "weights/best.pt"

# Loading of Trained YOLOv9 Model
try:
    model = YOLO(model_path)
except Exception as ex:
    st.error(f"Unable to load YOLO model. Check Model Path: {model_path}")
    st.error(ex)

# Page Layout Setting
st.set_page_config(
    page_title="Live Object Detection",
    page_icon='📽',
    layout="wide",
    initial_sidebar_state="expanded"
)

# Function for drawing frame purposes
def draw_boxes(frame, boxes, threshold):
```

```
    # Implementation to draw bounding boxes on the image
    return frame

# Function for WebCam and SiteCam1 to detect
def detect_objects(image):
    # Detection implementation using YOLO model
    detections = model(image, conf = confidence)
    # Format detections if necessary
    return detections

# Function for SiteCam2 to detect
def detect_objects2(image):
    # Detection implementation using YOLO model
    detections2 = model(image, conf = confidence)
    # Format detections if necessary
    return detections2

# Function to fetch data from MySQL database
def fetch_data(vin, sequence_number, colour, type):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            # Change query "data_live" to your table name
            query = "SELECT No, Date, Type, `Vin No.`, `Seq No.`, Colour, TIME, `Straight Bracket`, `No Washer`, Status FROM data_live"
            conditions = []
```

```
            conditions = []
            params = []
            if type:
                conditions.append("Type = %s")
                params.append(type)
            if vin:
                conditions.append("`Vin No.` = %s")
                params.append(vin)
            if sequence_number:
                conditions.append("`Seq No.` = %s")
                params.append(sequence_number)
            if colour and colour != 'All':
                conditions.append("Colour = %s")
                params.append(colour)
            if conditions:
                query += ' WHERE ' + ' AND '.join(conditions)
            cursor = db_conn.cursor()
            cursor.execute(query, params)
            df = pd.DataFrame(cursor.fetchall(), columns=[i[0] for i in cursor.description])
            cursor.close()
            db_conn.close()
            return df
    except Error as e:
        st.error(f"Error connecting to the database: {e}")
        return pd.DataFrame()

# Function to insert data to MySQL database
def insert_data(type, vin, seq_no, colour, bracket, washer, status):
    db_config = {
        'host': 'localhost',    # dont change
        'user': 'root',         # change user
```

```
        'password': '',         # put password
        'database': 'data1'     # database name
    }
    try:
        # with mysql.connector.connect(**db_config) as db_conn:
        db_conn = mysql.connector.connect(**db_config)
        if db_conn.is_connected():
            # Change the name in the query below ("data_live"
            query = """
                    INSERT INTO data_live (Type, `Vin No.`, `Seq No.`, Colour, `Straight Bracket`, `No Washer`, Status, `Date`)
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
                    """
            values = (type, vin, seq_no, colour, bracket, washer, status, datetime.now())
            cursor = db_conn.cursor()
            cursor.execute(query, values)
            db_conn.commit()
            st.success("Data inserted successfully!")
            cursor.close()
            db_conn.close()
    except Error as e:
        st.error(f"Error connecting to the database: {e}")

# Create Side Bar
with st.sidebar:
    st.header("Stream Configurations")
    confidence = float(st.slider("Set Confidence Level", 25, 100, 40)) / 100  # Model Confidence Option

    # Sidebar for user inputs to query the database
    st.header("Data Parameters")
    st.subheader("Insert for Data Query")
```

```
    type = st.text_input('Type')
    vehicle_identification_number = st.text_input('Vehicle Identification Number')
    sequence_number = st.text_input('Sequence Number')
    colour = ['All', 'Red', 'Blue', 'Silver', 'White', 'Black']
    selected_colour = st.selectbox('Vehicle Colour', colour)

    st.subheader("Insert Data into Database")
    with st.form(key='insert form'):
        type_input = st.text_input('Type (Insert)')
        vin_input = st.text_input('Vehicle Identification Number (Insert)')
        seq_no_input = st.text_input('Sequence Number (Insert)')
        colour_input = st.selectbox('Vehicle Colour (Insert)', ['Red', 'Blue', 'Silver', 'White', 'Black'])
        bracket_input = st.text_input('Straight Bracket')
        washer_input = st.text_input('No Washer')
        status_input = st.selectbox('Status', ['ok', 'not ok'])

        submit_button = st.form_submit_button('Submit')
        if submit_button:
            insert_data(type_input, vin_input, seq_no_input, colour_input, bracket_input, washer_input, status_input)

# Main Page
st.title("Live Object Detection")

# Fetch data based on inputs
data = fetch_data(vehicle_identification_number, sequence_number, selected_colour, type)

column1, column2 = st.columns([0.7,0.35])
# Display the fetched data
with column1:
    st.subheader('Queried Data')
```

```
    if not data.empty:
        st.dataframe(data)
    else:
        st.write("No data available based on the input parameters.")

# Pie chart section for 'Status' distribution
with column2:
    if not data.empty and 'Status' in data.columns and 'Date' in data.columns:
        # Parse the date column to extract months
        data['Month'] = pd.to_datetime(data['Date']).dt.strftime('%Y-%m')

        # Aggregate the data to count statuses by month
        status_counts_by_month = data.groupby(['Month', 'Status']).size().unstack(fill_value=0)

        # Plot the data as a bar chart
        fig, ax = plt.subplots()
        status_counts_by_month.plot(kind='bar', ax=ax)
        ax.set_ylabel('Count')
        ax.set_xlabel('Month')
        ax.set_title('Status Distribution by Month')
        ax.legend(title='Status')

        # Set yaxis whole number
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))

        st.subheader('Status Distribution by Month')
        st.pyplot(fig)
    else:
        st.write("Data is empty or required columns are missing for the bar chart.")

```

asdasdasda

```python
# Create two tabs for Webcam and SiteCam
tab1, tab2 = st.tabs(["WebCam", "SiteCam"])

# Tab 1 for Webcam
with tab1:
    column1, column2 = st.columns([4, 4])
    with column1:
        st.header("WebCam Stream")
        webcam_switch = st.toggle("Start/Stop Detection")
        webcam = cv2.VideoCapture(0)
        display_webcam = st.empty()
        if webcam_switch:
            st.write("Camera On")
            webcam_OnOff = True
        else:
            st.write("Camera Off")
            webcam_OnOff = False
        while webcam_OnOff:
            ret, img = webcam.read()
            if ret:
                detections = detect_objects(img)
                res_plotted = detections[0].plot()
                output_frame = cv2.cvtColor(res_plotted, cv2.COLOR_BGR2RGB)
                display_webcam.image(res_plotted, channels="BGR", use_column_width=True)

# Tab 2 for SiteCam 1 and SiteCam 2
with tab2:
    placeholder = st.columns(2)
    st.header("SiteCam Stream")
    sitecam_switch = st.toggle("Start/Stop Detection All SiteCam")
```

```
    sitecam_1 = cv2.VideoCapture("http://192.168.1.158:8081") # Site Cam 1http://192.168.1.137:8081
    sitecam_2 = cv2.VideoCapture("http://192.168.1.152:8081")  # SiteCam 2 http://192.168.1.125:8081
    col1 = placeholder[0].empty()
    col2 = placeholder[1].empty()
    if sitecam_switch:
        st.write("SiteCam On")
        sitecam_OnOff = True
    else:
        st.write("SiteCam Off")
        sitecam_OnOff = False
    while sitecam_OnOff:
        ret1, img1 = sitecam_1.read()
        ret2, img2 = sitecam_2.read()
        if ret1 | ret2:
            detections = detect_objects(img1)
            detections2 = detect_objects2(img2)
            res_plotted1 = detections[0].plot()
            res_plotted2 = detections2[0].plot()
            output_frame1 = cv2.cvtColor(res_plotted1, cv2.COLOR_BGR2RGB)
            output_frame2 = cv2.cvtColor(res_plotted2, cv2.COLOR_BGR2RGB)
            col1.image(res_plotted1, channels="BGR", use_column_width=True)
            col2.image(res_plotted2, channels="BGR", use_column_width=True)
```

END

FYP2_Report_FB20053_removed

ORIGINALITY REPORT

10%	8%	5%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www2.mdpi.com Internet Source	1%
2	m.moam.info Internet Source	<1%
3	www.progress.com.mx Internet Source	<1%
4	epdf.pub Internet Source	<1%
5	Submitted to SRH Holding Student Paper	<1%
6	lup.lub.lu.se Internet Source	<1%
7	Submitted to University of College Cork Student Paper	<1%
8	Submitted to Universiti Teknologi MARA Student Paper	<1%
9	ijrpr.com Internet Source	<1%