RADIATOR COUNT MONITORING SYSTEM FOR LOGISTIC MANAGEMENT SYSTEM USING YOLOV8.

MUHAMMAD SYUKRI BIN AHMAD ADNAN

BACHELOR OF MECHATRONICS ENGINEERING WITH HONOURS

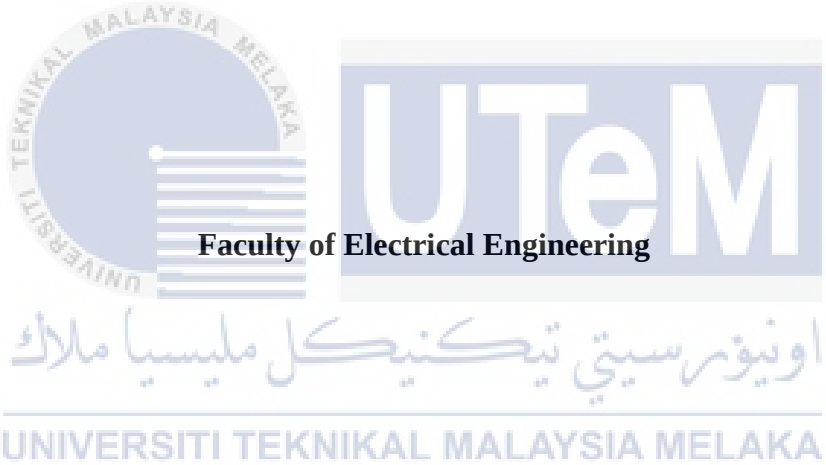
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

RADIATOR COUNT MONITORING SYSTEM FOR LOGISTIC MANAGEMENT SYSTEM USING YOLOV8.

MUHAMMAD SYUKRI BIN AHMAD ADNAN

A report submitted
in partial fulfilment of the requirements for the degree of
Bachelor of Mechatronics Engineering with Honours



Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DECLARATION

I declare that this thesis entitled ": RADIATOR COUNT MONITORING SYSTEM FOR LOGISTIC MANAGEMENT SYSTEM USING YOLOV8" is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.

Signature : Syukri

Name : MUHAMMAD SYUKRI BIN AHMAD ADNAN

Date : 17/6/2024

APPROVAL

I hereby declare that I have checked this report entitled "RADIATOR COUNT MONITORING SYSTEM FOR LOGISTIC MANAGEMENT SYSTEM USING YOLOV8", and in my opinion, this thesis fulfils the partial requirement to be awarded the degree of Bachelor of Mechatronics Engineering with Honours

Signature : .....

Supervisor Name : IR. DR. ZAMANI BIN MD. SANI

Date : 17/6/2024 .....

DEDICATIONS

To my beloved mother and father

ACKNOWLEDGEMENTS

In preparing this report, I communicated with a great deal of academics, practitioners, and researchers while writing this study. My comprehension and thinking have been enhanced by them. I wish to express my sincere appreciation to my project supervisor, Ir. Dr. ZAMANI BIN MD. SANI, for encouragement, guidance critics and friendship. This project would not have become what it is now without his unwavering support.

Additionally, I would like to extend my sincerest gratitude to SIME DARBY AUTO SDN. BHD. for the opportunity to explore this project. This honor helps to motivate me to continue pushing boundaries and striving to complete this project. I am truly grateful for their generosity in promoting academic growth and innovation among students like me.

I also want to express my gratitude for the help and inspiration that I have received from the Faculty of Electrical Engineering faculty at Universiti Teknikal Malaysia Melaka. I would especially want to thank my panelists AHMAD ZAKI BIN SHUKOR and FADILAH BINTI ABDUL AZIZ for their insightful criticism that helped me better understand my idea. This project would not have been the same without their ongoing attention and support.

Finally, I should also thank my family and other undergraduate students for their support. My deep gratitude also goes out to all of the individuals who have provided help on many occasions. Their opinions and advice are very helpful. Unfortunately, it is not possible to list all of them in this limited space.

ABSTRACT

This project focuses on the development and implementation of an object detection system utilizing the YOLOv8 (You Only Look Once version 8) deep learning architecture for accurate identification of car radiators in the context of assembly line and logistics. The automotive assembly industry relies heavily on manual labor for part supply monitoring. Traditional methods for radiator detection often fall short in terms of speed and accuracy. Using the capabilities of YOLOv8, this approach aims to enhance real-time detection of car radiators, enabling swift and precise identification for logistic management system for parts replenish applications to assembly lines. By training the model on a comprehensive dataset of annotated car radiator images, the aim is to fine-tune the YOLOv8 architecture, specifically to recognize the complex features and variations associated with different radiator designs. The project's key objectives include optimizing the detection accuracy, minimizing false positives, and ensuring real-time processing speed to meet the rigorous requirements of automotive applications. This includes providing information on the quantity of logistic parts supply, specifically car radiators, within the field of view. The study demonstrates the progressive improvement of YOLOv8's performance metrics with increasing epochs. At 10 epochs, YOLOv8 achieves moderate precision (0.59862), recall (0.59321), and an F1 score of 0.59, which improves significantly at 25 epochs with precision (0.92441), recall (0.8916), and an F1 score of 0.91. By 50 epochs, YOLOv8 further enhances its performance, achieving a precision of 0.95921, recall of 0.95593, and an F1 score of 0.96. At 75 epochs, the model maintains high precision (0.96098), recall (0.96816), and an F1 score of 0.96, ultimately reaching outstanding precision (0.97493), recall (0.97277), and an F1 score of 0.97 at 100 epochs. Comparatively, YOLOv9 shows potential with higher initial precision (0.67825) and recall (0.6316) at 10 epochs but requires longer training times (6.651 hours) and further optimization, with an F1 score of 0.67. The shorter training time of YOLOv8, requiring only 1.664 hours for 10 epochs, makes it advantageous for rapid prototyping and iterative model refinement. YOLOv8's successful recognition and localization of objects signify its potential contribution to automating logistic part supply chains, offering real-time insights for streamlined operations.

ABSTRAK

Projek ini meumpukan kepada pembangunan dan pelaksanaan sistem pengesanan objek menggunakan seni bina “deep learning” YOLOv8 (You Only Look Once version 8) untuk mengesan dengan tepat radiator kereta dalam konteks kawasan pemasangan dan logistik. Industri pemasangan automotif banyak bergantung kepada buruh manual untuk pemantauan bahagian bekalan alat pemasangan. Kaedah tradisional untuk pengesanan radiator mempunyai kekurangan dari segi kelajuan dan ketepatan. Menggunakan keupayaan YOLOv8, dapat meningkatkan pengesanan dunia nyata radiator kereta, membolehkan pengesanan pantas dan tepat untuk sistem pengurusan logistik bagi aplikasi penambahan alat pemasangan pada barisan pemasangan. Dengan melatih model pada set data komprehensif, imej radiator kereta beranotasi, bertujuan untuk memngajar seni bina YOLOv8, khususnya untuk mengenali ciri dan variasi kompleks yang dikaitkan dengan reka bentuk radiator yang berbeza. Objektif utama projek termasuk mengoptimumkan ketepatan pengesanan, meminimumkan positif palsu, dan memastikan kelajuan pemprosesan masa nyata untuk memenuhi keperluan aplikasi automotif. Ini termasuk menyediakan maklumat tentang kuantiti bekalan alat ganti logistik, khususnya radiator kereta, dalam sudut pandangan. Kajian ini menunjukkan peningkatan progresif dalam metrik prestasi YOLOv8 dengan peningkatan bilangan epoch. Pada 10 epoch, YOLOv8 mencapai ketepatan sederhana (0.59862), ingatan (0.59321), dan skor F1 sebanyak 0.59, yang meningkat dengan ketara pada 25 epoch dengan ketepatan (0.92441), ingatan (0.8916), dan skor F1 sebanyak 0.91. Pada 50 epoch, YOLOv8 terus meningkatkan prestasinya, mencapai ketepatan sebanyak 0.95921, ingatan sebanyak 0.95593, dan skor F1 sebanyak 0.96. Pada 75 epoch, model mengekalkan ketepatan tinggi (0.96098), ingatan (0.96816), dan skor F1 sebanyak 0.96, dan akhirnya mencapai ketepatan cemerlang (0.97493), ingatan (0.97277), dan skor F1 sebanyak 0.97 pada 100 epoch. Sebaliknya, YOLOv9 menunjukkan potensi dengan ketepatan awal yang lebih tinggi (0.67825) dan ingatan (0.6316) pada 10 epoch, tetapi memerlukan masa latihan yang lebih lama (6.651 jam) dan pengoptimuman selanjutnya, dengan skor F1 sebanyak 0.67. Masa latihan yang lebih pendek bagi YOLOv8, yang memerlukan hanya 1.664 jam untuk 10 epoch, menjadikannya lebih sesuai untuk prototaip cepat dan penghalusan model berulang. YOLOv8 dalam mengenali dan menempatkan objek menunjukkan potensinya untuk menyumbang secara signifikan kepada automasi rantaian bekalan alat logistik, menawarkan wawasan masa nyata untuk operasi yang lebih lancar.

TABLE OF CONTENTS

		PAGE
DECLARATION		
APPROVAL		
DEDICATIONS		
ACKNOWLEDGEMENTS		**2**
ABSTRACT		**3**
ABSTRAK		**4**
TABLE OF CONTENTS		**5**
LIST OF TABLES		**7**
LIST OF FIGURES		**9**
LIST OF SYMBOLS AND ABBREVIATIONS		**12**
LIST OF APPENDICES		**13**
CHAPTER 1	**INTRODUCTION**	**14**
1.1	Background	14
1.2	Motivation	15
1.3	Problem Statement	16
1.4	Objective	17
1.5	Scope	17
1.6	Limitation	18
CHAPTER 2	**LITERATURE REVIEW**	**19**
2.1	Introduction	19
2.2	Manual Monitoring Approach	20
2.3	Vision Based Monitoring Approach.	21
2.3.1	Machine Vision	21
2.4	Machine Learning	23
2.4.1	Unsupervised Machine Learning	23
2.4.2	Deep Learning	23
2.4.3	Comparison Between Unsupervised Learning and Deep Learning.	24
2.5	Object Detection	26
2.5.1	Two-stage Detectors	26
2.5.2	Single-stage Detectors.	26
2.6	Data Acquisition	29
2.6.1	Roboflow	29
2.6.2	CVAT (Computer Vision Annotation Tool)	30
2.7	Training Algorithm	31
2.7.1	YOLO-v6.	31

2.7.2 YOLO-v7. 33
2.7.3 YOLO-v8. 36
2.7.4 Single-Shot MultiBox Detector 39
2.7.5 Regional Convolutional Neural Network (R-CNN) 40
2.7.6 Comparison of Faster-RCNN, YOLO, and SSD. 41
2.8 Summary of Past Research 43
2.9 Conclusion 47

CHAPTER 3 METHODOLOGY 48
3.1 Introduction 48
3.2 Project Overview 48
3.2.1 Training Algorithm for Object Recognition and Classification 49
3.2.1.1 Experimental Setup 50
3.2.1.2 Experimental Datasets 50
3.2.1.3 Data Labelling and Annotation. 52
3.3 Experimental Environment 53
3.4 Training Model 53
3.5 Evaluation Indicators of Model 55

CHAPTER 4 RESULTS AND DISCUSSIONS 56
4.1 Training Result 56
4.2 Confusion Matrix 57
4.2.1 Comparison for YOLOV8 between 25, 50, 75 and 100 Epochs. 57
4.2.2 Comparison between YOLOV8 and YOLOV9 with 10 Epochs. 63
4.3 F1 Confidence Curve 67
4.3.1 Comparison for YOLOV8 between 25, 50, 75 and 100 Epochs. 67
4.3.2 Comparison between YOLOV8 and YOLOV9 with 10 Epochs. 70
4.4 Precision-Recall (PR) Curve 72
4.4.1 Comparison for YOLOV8 between 25, 50, 75 and 100 Epochs 72
4.4.2 Comparison between YOLOV8 and YOLOV9 with 10 Epochs. 75
4.5 Training Loss 77
4.5.1 Comparison for YOLOV8 between 25, 50, 75 and 100 Epoch 78
4.5.2 Comparison between YOLOV8 and YOLOV9 with 10 Epochs. 81
4.6 Testing Result 83

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS 86
5.1 Conclusion 86
5.2 Future Works 88

REFERENCES 89

APPENDICES 94

LIST OF TABLES

Table 2.1 Difference between Unsupervised Learning and Deep Learning 24

Table 2.2 Comparison of YOLOv6 variant 31

Table 2.3 Comparison of YOLOv7 Variant 35

Table 2.4 Performance on Validation and Test Datasets [49] 38

Table 2.5 Differences Between YOLO and SSD 39

Table 2.6 Comparison Between Object Detection Models 40

Table 2.7 Deep Learning Models Performance [52] 41

Table 2.8 Past Research Training Algorithm 43

Table 3.1 Experiment Dataset 51

Table 3.2 Experimental Setup Specification Hardware Specification 53

Table 4.1 Summary of the Training Parameters and Results for Yolov8 56

Table 4.2 Summary of the Training Parameters and Results for Yolov9 56

Table 4.3 Confusion matrix values for 25 epoch 58

Table 4.4 Confusion matrix values for 50 epoch 59

Table 4.5 Confusion matrix values for 75 epoch 61

Table 4.6 Confusion matrix values for 100 epoch 62

Table 4.7 Confusion matrix values of Yolov8 for 10 epoch 64

Table 4.8 Confusion matrix values of Yolov9 with 10 epoch 65

Table 4.9 Last 5 Batch of 25 epoch 79

Table 4.10 Last 5 Batch of 50 epoch 79

Table 4.11 Last 5 Batch of 75 epoch 79

Table 4.12 Last 5 Batch of 100 epoch 79

Table 4.13 Last 5 Batch of 10 epoch with Yolov8 81

Table 4.14 Last 5 Batch of 10 epoch with Yolov9 82

LIST OF FIGURES

Figure 1.1 Manual Monitoring of Parts Replenish. 14

Figure 1.2 Low Number of Radiation 15

Figure 1.3 Camera Placement Illustration. 17

Figure 2.1 K-Chart 19

Figure 2.2 Introduction Principle of Machine Vision Technology [2] 22

Figure 2.3 Structure of object detection. 27

Figure 2.4 Evolution of YOLO. 28

Figure 2.5 Basic Architecture of YOLOv6. 32

Figure 2.6 YOLOv5 vs YOLOv6. 33

Figure 2.7 Compound Scaling of YOLOv7. 34

Figure 2.8 YOLOv7 vs alternative object detectors [41]. 35

Figure 2.9 YOLOv8 vs alternative YOLO version. 37

Figure 2.10 Faster R-CNN Pipeline 41

Figure 2.11 Performance comparison 42

Figure 3.1 Project Flowchart 48

Figure 3.2 Recognition and Classification using Artificial Intelligence Flowchart 49

Figure 3.3 Experimental setup 50

Figure 3.4 Image Datasets Overview 51

Figure 3.5 Computer Vision Annotation Tool (CVAT) Software 52

Figure 4.1 Validation image overview 56

Figure 4.2 Confusion Matrix 25 epoch 57

Figure 4.3 Confusion Matrix for 50 epoch 59

Figure 4.4 Confusion Matrix for 75 epoch 60

Figure 4.5 Confusion Matrix for 100 epoch 62

Figure 4.6 Confusion Matrix of Yolov8 for 10 epoch 63

Figure 4.7 Confusion Matrix of Yolov9 with 10 epoch 65

Figure 4.8 F1 Confidence Curve for 25 epoch 67

Figure 4.9 F1 Confidence Curve for 50 epoch 67

Figure 4.10 F1 Confidence Curve for 75 epoch 68

Figure 4.11 F1 Confidence Curve for 100 epoch 68

Figure 4.12 F1 Confidence Curve for 10 epoch with Yolov8 70

Figure 4.13 F1 Confidence Curve for 10 epoch with Yolov9 70

Figure 4.19 Precision-Recall (PR) curve for 25 Epoch 72

Figure 4.20 Precision-Recall (PR) curve for 50 Epoch 72

Figure 4.21 Precision-Recall (PR) curve for 75 Epoch 73

Figure 4.22 Precision-Recall (PR) curve for 100 Epoch 73

Figure 4.22 Precision-Recall (PR) curve for 15 Epoch using YOLOV8 75

Figure 4.23 Precision-Recall (PR) curve for 10 Epoch using YOLOV9 75

Figure 4.21 Training loss, Validation loss for 25 epoch. 78

Figure 4.22 Training loss, Validation loss for 50 epoch. 78

Figure 4.23 Training loss, Validation loss for 75 epoch. 78

Figure 4.24 Training loss, Validation loss for 100 epoch. 78

Figure 4.20 Training loss, Validation loss Precision and Recall for 10 epoch with Yolov8. 81

Figure 4.25 Training loss, Validation loss Precision and Recall for 10 epoch with Yolov9. 81

Figure 4.26 Result of Video Testing 1. 83

Figure 4.27 Result of Video Testing 2. 84

Figure 4.28 Result of Video Testing 3. 85

LIST OF SYMBOLS AND ABBREVIATIONS

ms	-	Millisecond
AI	-	Artificial Intelligence
ANN	-	Artificial Neural Networks
CV	-	Computer Vision
CNN	-	Convolutional Neural Networks
QI	-	Quality Inspection
VGG	-	Visual Geometry Group
FPS	-	Frames Per Second
SSD	-	Single Shot Detector
YOLO	-	You Only Look Once
DFL	-	Distribution Focal Loss
VFL	-	Varifocal Loss
NMS	-	Non-Maximum Suppression
mAP	-	Mean Average Precision
SOTA	-	State-Of-The-Art
FPN	-	Feature Pyramid Network
PANet	-	Path Aggregation Network
IoU	-	Intersection Over Union
GPU	-	Graphic Processing Unit
CVAT	-	Computer Vision Annotation Tool

LIST OF APPENDICES

APPENDIX A 94

CHAPTER 1

INTRODUCTION

1.1 Background

Time is important for automotive assembly lines, dictating efficiency, productivity, and profitability. Every second needs to be accurately planned and optimized to ensure a seamless flow of operations. In this high-paced environment, the assembly line must operate perfectly to avoid disruptions that will impact production output, thus affecting the number of vehicles manufactured within a given period.

For years, the replenishment of parts (radiators) from the logistics supply chain to the assembly lines has relied heavily on manual monitoring by human operators as illustrated in Figure 1.1. This traditional approach, however, has been consistently prone to errors and delay. After all, humans' errors or delays along the line can cascade, causing bottlenecks that ripple through the entire manufacturing process, leading to increased costs, missed deadlines, and potential production halts. Moreover, in the automotive industry, time holds immense financial consequences.

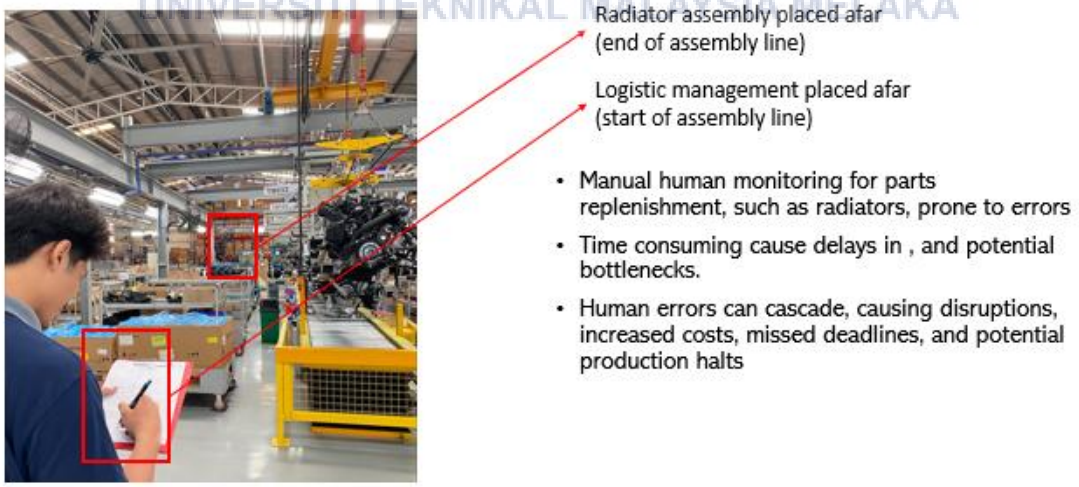


Figure 1.1 Manual Monitoring of Parts Replenish.

Hence, introducing computer vision with the help of AI algorithm for automated part detection to replenish the parts (radiators) into the assembly lines whenever the number is low, as shown in Figure 1.2. This idea can eliminate humans' error and downtime that tend to happen traditionally (manual human interaction)

Figure 1.2 Low Number of Radiation

1.2 Motivation

Improving error management in logistic systems within automotive assembly lines is crucial to sustain its efficiency and quality for a big company like SIME DARBY MOTORS SDN BHD. Addressing errors within logistic management systems holds the key to rectify the operations and enhancing overall productivity. By minimizing errors, such as inventory inconsistencies, delayed deliveries, or inaccurate parts allocation, the assembly line can function flawlessly, meeting production schedules and maintaining high-quality standards. This improvement not only reduces downtime and associated costs but also enhances customer satisfaction by delivering products on time with fewer defects.

Furthermore, a robust error management system fosters a culture of continuous improvement. It encourages proactive problem-solving approaches such as using computer vision for automated part detection using AI. This commitment to refining logistic management systems serves as a catalyst for progress, pushing the industry toward higher levels of precision, reliability, and competitiveness in the global market. Plus, Reduced errors mean fewer resources wasted, optimized processes, and improved resource allocation. This contributes positively to a greener and more sustainable production model.

1.3 Problem Statement

In the current operational landscape, the logistics supply chain to the assembly lines has relied heavily on manual monitoring to assess the need for radiators replenish. The manual nature of monitoring activities introduces inconsistencies that can lead to inaccuracies of notifying the logistic management system. Since the monitoring operator are sometime notify at the last-minute manners or too early. This inconsistency not only compromises the reliability of the information but also challenges the integrity of decision-making processes of the need to supply the radiator.

Moreover, the manual monitoring approach contributes to a higher likelihood of errors, as human operators may unintentionally overlook to notify logistic management system for the radiator replenish. This scenario will affect the operator's decision making and increase the probability of supplying the wrong radiator unit. Additionally, the reliance on manual monitoring has proven to be a significant cause of delays in the workflows. The time-consuming nature of human-centric monitoring activities hampers the agility and responsiveness of the operations. As the scenario mentioned before, whenever the operator overlooks to notify the logistic management system for supply parts, delayed information causes the bottleneck inside the assembly line.

1.4 Objective

i. To develop a computer vision using YOLOv8 that can detect and classify logistic parts supply (car radiator).

ii. To analyse the performance of the detection using YOLOv8 algorithm with different epoch (10, 25, 50, 75, and 100).

iii. To train and compare the performance with other algorithms (YOLOv9) and decide on the best algorithm for this type of project.

1.5 Scope

i. Detect and recognize BMW car radiator for every car model (28 variant) assembled in the same assembly line.

ii. Detect and determine the number of car radiators left within the assembly line and notify the logistic management system to replenish the assembly parts if it is low.

iii. Detection from above (1.2 meters from the radiator) during good lighting environment, where the car radiator packaging was consistently placed for every replenish as shown in Figure 1.3.

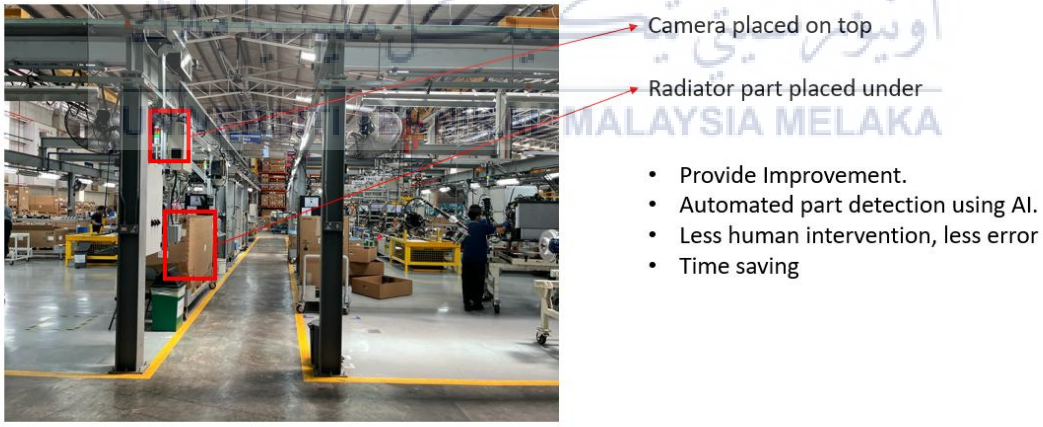


Figure 1.3 Camera Placement Illustration.

1.6 Limitation

The challenges associated with obtaining datasets, primarily due to the considerable distance of the research sites and the demanding schedule constraints. The necessity for on-site visits to capture the dataset is hindered by the packed schedule. To mitigate these limitations, visiting the site during the holiday period to collect more datasets is required. Plus, applying data augmentation techniques to artificially expand the diversity of the dataset is also be considered.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Figure 2.1 breakdown the general approach, processes, techniques, and architecture of Logistic Management System for parts replenishment related to this project.

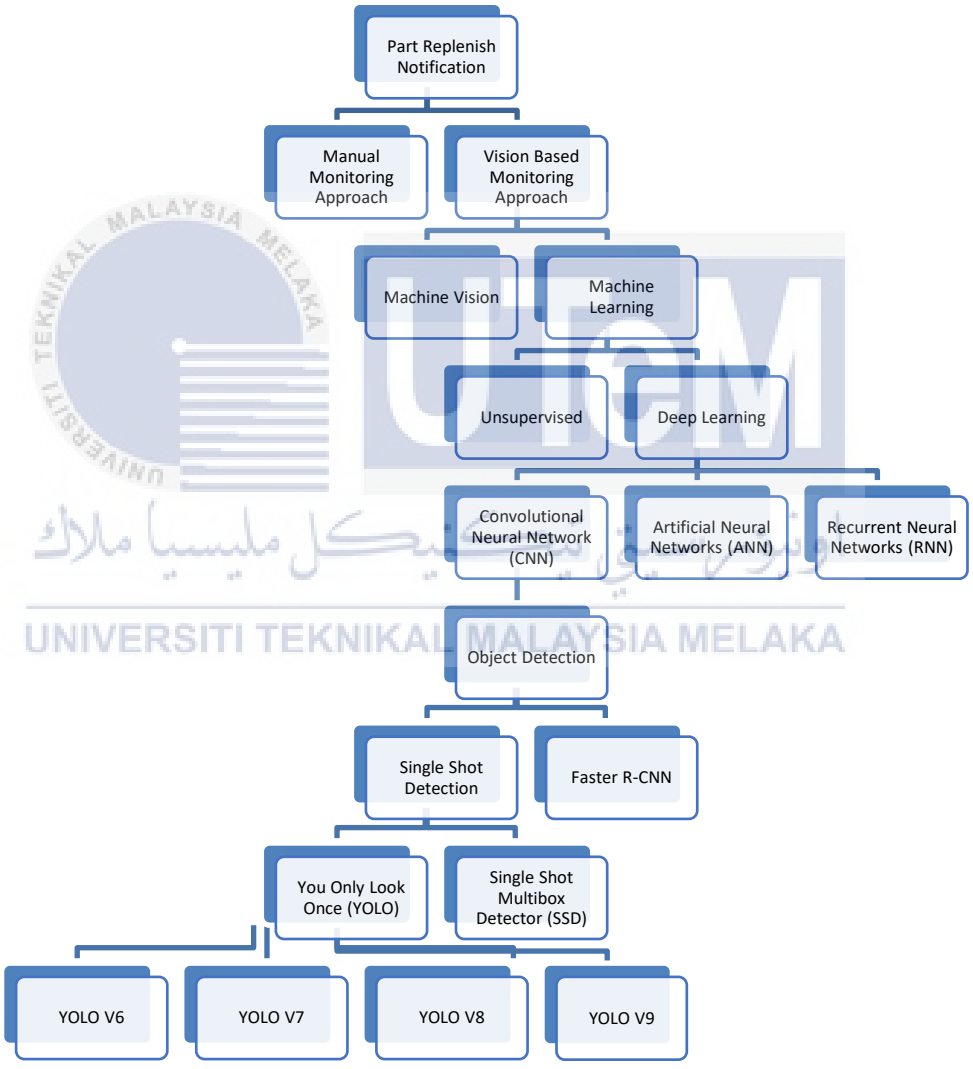


Figure 2.1 K-Chart

2.2 Manual Monitoring Approach

Manual human monitoring in assembly part allocation refers to the process of overseeing and managing the distribution of components or parts within an assembly line or production system using human intervention rather than automated systems. In this context, assembly part allocation involves assigning and distributing the necessary parts to different stations or workstations along the assembly line where they are needed for the manufacturing or assembly of a final product.

One aspect of manual human monitoring in assembly part allocation involves individuals physically observing the production line, checking the availability of parts, and ensuring that each station has the necessary components to carry out its specific tasks. This process often relies on human judgement, memory, and communication skills to coordinate and allocate parts effectively.

However, this manual approach has inherent challenges. Firstly, it is susceptible to inconsistency, as different human operators may interpret the requirements differently or make decisions based on varying criteria. This can lead to uneven distribution of parts, causing delays, disruptions, or errors in the assembly process. Additionally, manual monitoring is prone to human error, as individuals may overlook critical details, misinterpret information, or make mistakes in part allocation. These errors can result in defective products, increased rework, or the need for costly corrections.

Lastly, the reliance on manual human monitoring for assembly part allocation can lead to delays in production. Human operators may struggle to keep pace with the speed required in modern manufacturing environments, causing bottlenecks, and reducing overall efficiency. This delay can have a cascading effect on the entire production schedule, potentially impacting deadlines and customer satisfaction. To address these challenges, organizations often seek to implement automated systems for part allocation, leveraging technology to enhance accuracy, consistency, and speed in the assembly process.

2.3 Vision Based Monitoring Approach.

Automated vision-based monitoring in assembly part allocation involves the use of machine vision or machine learning with computer vision technology to oversee and manage the distribution of components or parts within an assembly line or production system. In this context, the automated system relies on cameras and image processing algorithms to identify, track, and allocate parts to different stations along the assembly line where they are needed for the manufacturing or assembly of a final product. The process begins with cameras strategically placed along the assembly line, capturing real-time images or video footage of the production environment. These cameras feed the visual data to a computer system equipped with advanced image processing and computer vision algorithms.

2.3.1 Machine Vision

Machine vision technology is a disciplinary field that identifies and verifies components based on visual characteristics such as shape, color, size, or unique markings. It utilizes computer simulation video screens to extract and recognize targets, offering the benefit of achieving high accuracy [1]. Machine vision technology utilizes industrial Charge Coupled Device (CCD) cameras for capturing video screen targets, enabling it to preprocess them. [1] [2]. Then, data acquisition cards are utilized for the processing and transmission of video screen targets. Subsequently, computer terminals are employed to analyze and assess the processed targets. Following this, the video screen target detection is accomplished based on the computer's output result. The detection and tracking of video objects incorporate the application of the video processing module of machine vision technology [1] [3]. Figure 2.2 shows the flow of machine vision principle.

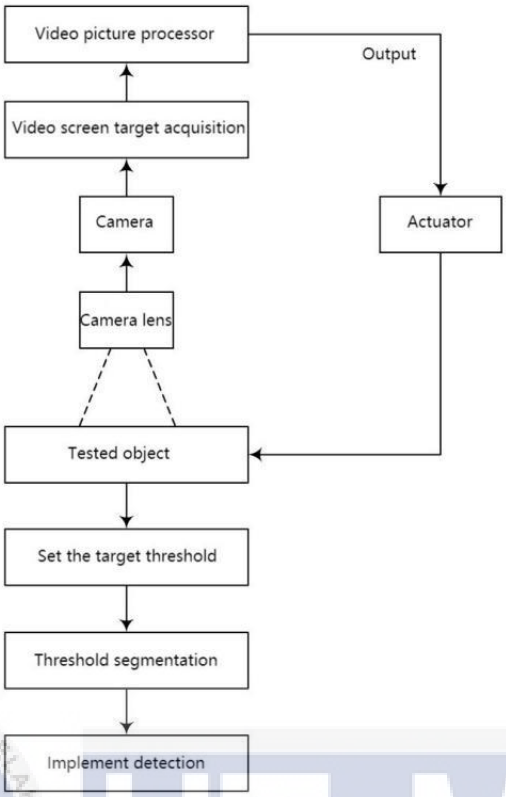


Figure 2.2 Introduction Principle of Machine Vision Technology [2]

However, machine vision systems are designed for specific tasks and lack adaptability to inconsistent datasets. While machine vision excels at recognizing patterns and features, it may lack the contextual understanding that humans possess. Understanding the broader context of a scene, considering the relationships between objects, or interpreting complex scenarios may still be a hurdle for machine vision systems. struggle to adapt to variations in part appearance, especially when dealing with diverse or irregularly shaped objects. This gives rise to Machine Learning applications.

2.4 Machine Learning

Machine learning is a branch of computer science and artificial intelligence that uses data and algorithms to imitate on how human learns by learning from and expanding upon past experiences [4]. There are several well-established algorithms for prediction and analysis, such as unsupervised learning and deep learning. The Popular libraries for image segmentation include SciPy, Scikit, OpenCV, Matplotlib, and Keras [4]

2.4.1 Unsupervised Machine Learning

In this approach, algorithms are trained using unlabeled data to extract features, identify crucial patterns and structures, and link related objects. These techniques serve practical purposes by organizing information, recognizing patterns, and facilitating efficient systems. This method doesn't rely on labeled data for training, enabling algorithms to uncover insights and relationships independently [4][5]. However, Unsupervised learning encounters certain challenges. Duplicating samples accurately poses a significant difficulty, impeding the generation of diverse and representative datasets crucial for robust model training. Additionally, managing the semantic distributions of collected data are complicated, as ensuring a balanced and comprehensive representation of various semantic aspects within the dataset is crucial for the model's understanding and generalization abilities. These hurdles underscore the complexities inherent in deploying unsupervised learning methods effectively.[4]

2.4.2 Deep Learning

People use visual cortex, the essential cortical part of the brain that accountable for managing visual information [6], which observes, recognizes [7], and differentiates among objects instantly [8]. Research into the deep mechanism of the visual cortex in the brain open the way for ANN (artificial neural networks) [9] together with many other computational architectures that fall into the deep learning category. For the past few year, due to fast and revolutionary developments in the deep learning field [10], researchers worked hard on providing effective computers simulation of the human visual system. Enabling computers to recognize desired objects within images and video [11]. This field of study are called as computer vision (CV) [12]. CV comprised of subfields involving, object detection [13], image classification, and object segmentation [14]. These fields shared an architectural theme, which the manipulation of CNN (convolutional neural networks) [15]. CNN is used when tackling image data. Compared with other conventional image processing, CNN use multiple

convolutional layers which is convolution, pooling, and fully connected layers structures to extract deep semantic features masked within the image pixels [16]. The industrial sector presents a promising opportunity for artificial intelligence (AI) due to its potential for massive automation through CV. especially in terms of QI (quality inspection) [17]. But in order to be successful, CV-based systems have to meet a strict set of deployment specifications that might differ depending on the manufacturing industry [18].

2.4.3 Comparison Between Unsupervised Learning and Deep Learning.

Deep learning indeed holds several advantages over unsupervised learning, primarily in delivering more accurate classification results due to its reliance on labeled datasets. The presence of clear labels allows the model to learn patterns and relationships effectively, leading to precise predictions or classifications. Moreover, the simplicity of training and testing models with labeled data streamlines the process, enabling straightforward evaluation and refinement [4]. The comparison between these two approaches is shown in Table 2.1.

Table 2.1 Difference between Unsupervised Learning and Deep Learning

Type	**Unsupervised Learning**	**Deep Learning.**
Input data	Unlabeled	Labelled
Training process	Model receives only input data without ground-truth label during training	Model receives input data and ground-truth label during training
General purpose	Gain insight from the data	Predict an outcome
Computational Complexity	More computationally demanding	Computationally demanding
Time Complexity	Less time consuming	More time consuming
Performance	Less accurate	More accurate
Number of Classes	Unknown, the results can be arbitrary	Known in advance

Deep learning involves training a model on a labelled dataset, where the input data is paired with corresponding output labels. The algorithm learns to map the input to the output by generalizing patterns from the labelled examples, making it suitable for tasks like classification and regression. On the other hand, unsupervised learning deals with unlabeled data, aiming to uncover hidden patterns or structures within the dataset. Algorithms in this category, such as clustering or association, don't rely on labelled outputs. Instead, they identify inherent relationships or groupings within the data, aiding tasks like customer segmentation or anomaly detection. Thus, deep learning is more preferred for application of car radiator detection.

2.5 Object Detection

CNN are classified under feed forward neural networks [19]. Multiple convolutional layers are applied to the input layer to obtain a larger collection of feature maps on a smaller size. After further processing, these features mapping were converted into one-dimensional vectors feature. Subsequently, the vectors are utilized as the input for the fully connected layers. To achieve deeper feature maps, multiple convolutional and pooling layers may be stacked during the feature extraction and manipulation processes, which are crucial to the network's overall accuracy.

Widely used architectures for extracting features involve AlexNet [20], VGGNet [21], GoogleNet [22], and ResNet [23]. AlexNet developed in 2012 which comprises of 5 convolutional, 3 fully connected layers, and 3 pooling. VGGNet aims for performance improvement by deepening the architecture, introducing versions like VGG-16/19 with increased layer complexity. In contrast, GoogLeNet innovated with cascading 'inception' modules to capture information at various scales efficiently. On the other hand, ResNet introduced skip-connections, preserving, and conveying information from earlier to later layers for enhanced model training and performance.

The primary goal of an object detector is to determine whether the desired objects exist within an image or frame of a video. In the event of their presence, the detector provides information concerning the class and location dimensions of the identified object. There are two category of architectures in the object detection domain which is two-stage and single-stage detectors. [24].

2.5.1 Two-stage Detectors

The detection process are divided for Two-stage detectors: the first stage involves feature extraction and proposal, while the second stage encompasses regression and classification [25]. While offering elevated accuracy, this approach entails an extensive computational requirement, making it impractical for real-time implementation on limited edge devices. Two-stage detectors involves well-known R-CNN [26] variants, for example Fast R-CNN [27] and Faster R-CNN [28].

2.5.2 Single-stage Detectors.

Conversely, single-stage detectors merge the classification and regression processes into a single stage, markedly diminishing its computational requirements. This renders them

more suitable for deployment in production scenarios Several single-stage detectors are SSD (single shot detector), D-SSD (deconvolutional single shot detector) [29], RetinaNet [30], and the YOLO (You Only Look Once) [31] architectures that is highly compatible with industrial needs, due to its lightweight, accuracy, and edge-friendly deployment conditions. The basic structure was illustrated in Figure 2.3.

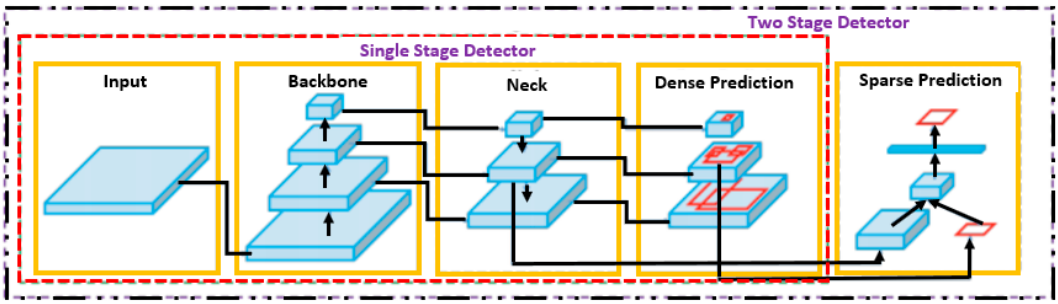


Figure 2.3 Structure of Object Detection.

The introduction of YOLO occurred through the paper 'You Only Look Once: Unified, Real-Time Object Detection' authored by Joseph Redmon et al. [31] that release in 2015. This paper redefined the landscape of object detection, outlining it effectively in a single pass. It began with pixels of image and progressed to predict bounding boxes and class of probabilities. The 'unified' approach, as the core of the methodology, allows for the prediction of numerous class probabilities and bounding boxes simultaneously, improving accuracy and speed. From the time of its launch in 2016 to the present (2023), the YOLO family has seen constant change. Despite the fact that the original inventor, Joseph Redmon, discontinued his work in the computer vision field at YOLO-v3 [32]. Many writers have developed further on the usefulness and potential of the 'unified' concept fundamental. The chronology of the YOLO development is shown in Figure 2.4.

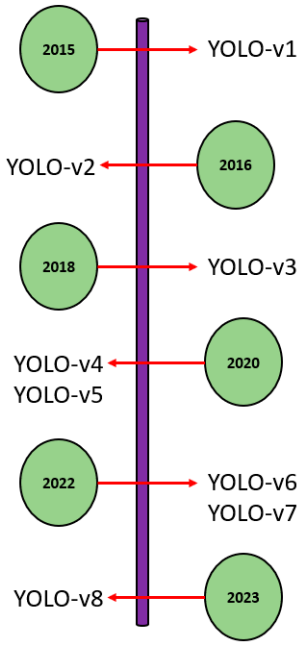


Figure 2.4 Evolution of YOLO.

2.6 Data Acquisition

In the case of common object detection, the usage of existing datasets can be optional. Depending on the specific use case, existing datasets might need leverage that is relevant to application. Few popular datasets like COCO (Common Objects in Context) [33] or Pascal VOC [34] provide annotated images for a variety of object classes.

However, for specific object detection like car radiators, data acquisition involves collecting and preparing a dataset that will be used to train, validate, and test an object detection model are needed. The process typically includes a diverse set of images collection that represent the scenarios in which the object detection model will operate. These images should cover different environments, and variations in object appearances. Additionally, data augmentation can also be used to increase the diversity of your dataset. Common augmentations include rotation, flipping, scaling, changes in brightness, and variations in contrast. Data augmentation helps the model generalize better to different scenarios. Consequently, images annotations are needed where are annotated with bounding boxes around the objects of interest. Each bounding box is labeled with the corresponding object class. This annotation process is crucial for training the model to recognize and localize objects. Several tools and frameworks can be employed for data acquisition in object detection. These tools aid in the annotation and organization of images, making the data suitable for training object detection models.

2.6.1 Roboflow

Roboflow is a comprehensive platform that simplifies and streamlines the entire process of preparing, managing, and deploying computer vision models. Offering a diverse set of tools and workflows, Roboflow addresses various tasks in the realm of computer vision. One of its notable features is its robust image annotation capabilities, supporting tasks like object detection, image segmentation, and keypoint annotation with versatile annotation types such as bounding boxes, polygons, and segmentation masks. The platform goes beyond simple annotation by providing powerful data augmentation capabilities, empowering users to effortlessly apply transformations like rotation, flipping, scaling, and brightness adjustments, thereby enhancing dataset diversity.

Roboflow stands out for its versatility in format conversion, allowing datasets to seamlessly transition between different annotation formats, including COCO JSON, Pascal VOC XML, YOLO, and TensorFlow Record. Version control for datasets enables efficient tracking of modifications and facilitates easy reverting to previous versions, promoting a clear and organized history of dataset changes.

The platform integrates seamlessly with popular deep learning frameworks such as TensorFlow, PyTorch, and Keras. It not only assists in generating code snippets and configurations but also provides preprocessing capabilities for standardizing and optimizing images before model training. While Roboflow's primary focus lies in data preparation, it offers integration with external model training services like Google Colab, Kaggle, and custom GPU servers, allowing users to connect datasets directly to their preferred training platforms.

2.6.2 CVAT (Computer Vision Annotation Tool)

CVAT is an open-source, web-based platform designed to facilitate the annotation of images and videos for computer vision tasks. Developed by the Computer Vision team at Intel, CVAT serves as a versatile tool for creating high-quality annotated datasets, particularly for object detection, image segmentation, and other computer vision applications. One of CVAT's key strengths lies in its user-friendly interface, making it accessible to both individual users and collaborative teams involved in machine learning projects. The platform supports various annotation types, including bounding boxes, polygons, polylines, and segmentation masks, providing flexibility for different annotation needs. CVAT offers robust features for both image and video annotation, allowing users to define and label objects within frames accurately Additionally, CVAT provides tools for handling complex scenarios, such as tracking objects across multiple frames in video annotation projects. The tool's extensibility allows for integration with different deep learning frameworks, facilitating a seamless workflow from annotation to model training. Annotations created in CVAT can be exported in various formats compatible with popular deep learning frameworks. Common export formats include PASCAL VOC, COCO JSON, and YOLO format.

2.7 Training Algorithm

Training an object detection model involves using a specific algorithm to learn patterns and features that allow it to recognize and locate objects within images or videos. Several popular algorithms and architectures have their own approach. For instance, YOLO employs a single-stage approach, dividing the image into a grid and predicting bounding boxes and class probabilities simultaneously. Moreover**,** SSD is also a single-stage approach, but predicts multiple bounding boxes at different scales for each object. However, Faster R-CNN utilizes a two-stage approach with region proposals and object classification.

2.7.1 YOLO-v6.

Meituan Technical Team based in China published the YOLO-v6 [35] codebase in June 2022. The goal of the authors' design approach was to create an object detector that was targeted at the industry. The architecture needs to be extremely efficient on a variety of hardware configurations, while retaining high speed and precision, in order to fulfill the needs of real-world applications. Table 2.2 illustrates the many variations of YOLO-v6 that are available to meet the requirements of a wide range of industrial applications. YOLO-v6-nano is the quickest variant with the fewest parameters, while YOLO-v6-large offers great accuracy at the cost of speed.

Table 2.2 Comparison of YOLOv6 Variant

Variant	**mAP 0.5:0.95 (COCO-val)**	**FPS Tesla T4**	**Parameters (Million)**
YOLO-v6-N	35.9 (300 epochs)	802	4.3
YOLO-v6-T	40.3 (300 epochs)	449	15.0
YOLO-v6-RepOpt	43.3 (300 epochs)	596	17.2
YOLO-v6-S	43.5 (300 epochs)	495	17.2
YOLO-v6-M	49.7	233	34.3

Numerous advancements incorporated into the YOLO-v6 design are responsible for the outstanding performance seen in Table 2.1. Few points can be used to summarize the major contributions. Firstly, YOLO-v6 chooses an anchor-free technique instead of an anchor-based one, which makes it 51% quicker than its predecessors.

Second, the authors presented a redesigned reparametrized neck and backbone that they called the Rep-PAN neck and the EfficientRep backbone [36]. That is to say, the regression and classification heads were identical up to and including YOLO-v5. Figure 2.3 illustrates

how YOLO-v6 implements the decoupled head, which differs from convention. Because of this, the design includes extra layers that divide features from the final head, has been methodically demonstrated to enhance performance.

Third, a two-loss function is required by YOLO-v6. The classification loss is distribution focal loss (DFL) [37], and the regression loss is SIoU/GIoU [38], using Varifocal loss (VFL) [39]. As a focal loss derivative, VFL balances the learning signals from both types of data by assigning different weights to positive and negative samples. Box regression in the medium and large forms of YOLO-v6 is implemented using DFL, which treats the continuous distribution of the box locations as a discretized probability distribution. This approach has been demonstrated to be especially effective in cases when the ground truth box borders are hazy. The basic architecture of YOLOv6 was illustrated in Figure 2.5.

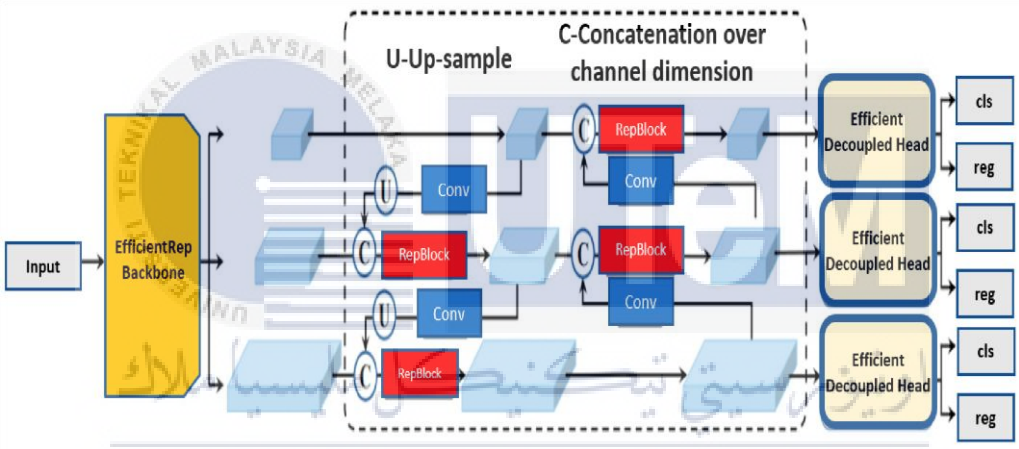


Figure 2.5 Basic Architecture of YOLOv6.

Further advancements focused toward industrial usage include knowledge distillation [40], which involves using a teacher model to train a student model in which the teacher model's predictions serve as soft labels in addition to the ground truth for the student model's training. Since the main goal is to train a smaller (student) model to mimic the high performance of the bigger (teacher) model, this is accomplished without increasing the computing cost. It is evident by comparing YOLO-v6's performance with that of its other past version, YOLO-v5 included, on the benchmark COCO dataset in Figure 2.6. YOLO-v6 accomplishes a higher mAP at different FPS.

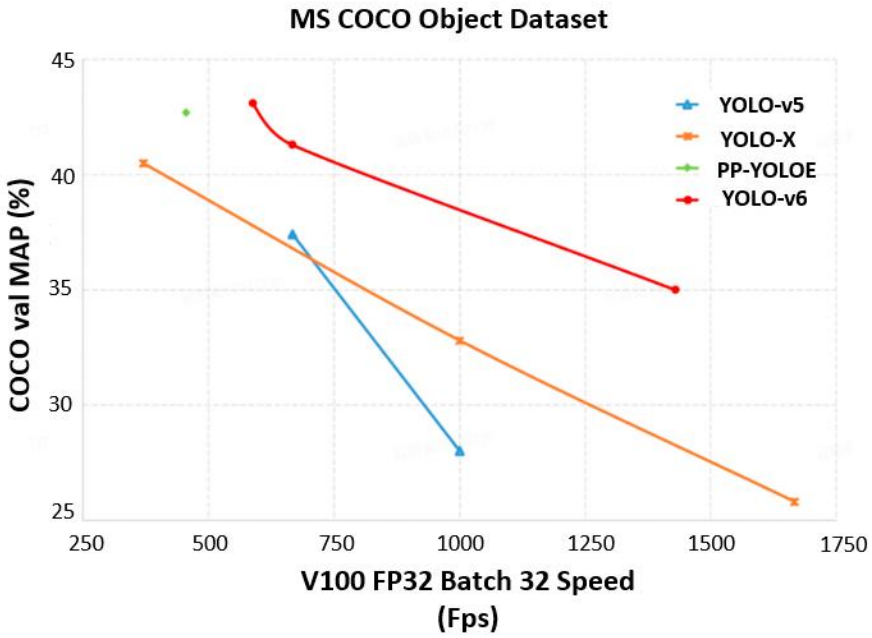


Figure 2.6 YOLOv5 vs YOLOv6.

2.7.2 YOLO-v7.

After YOLO-v6 was launched, YOLO-v7 was released the month after that [41]. Other versions, such as YOLO-X [42] and YOLO-R [43], were published in the meantime, however they were mostly concentrated on improving GPU speed for inferencing. In order to retain fast detection speeds and increase accuracy, YOLO-v7 suggests a number of architectural changes. Two types may be distinguished from the suggested changes: Trainable BoF (bag-of-freebies) and architectural reforms. Inspired by research breakthroughs in network efficiency, architectural improvements included the deployment of the E-ELAN (extended efficient layer aggregation network) [44] in the YOLO-v7 backbone. The study of variables including memory access cost, input/output channel ratio, and gradient path that affect accuracy and speed served as the basis for the creation of the E-ELAN.

Figure 2.7 depicts the second architectural reform, known as compound model scaling. The goal was to provide a broader range of application needs. For instance, some applications could value speed above accuracy, while others would emphasize both. For parameter-specific scaling to discover the optimal factors, NAS (network architecture search) [45] can be used, although the scaling factors are independent [46]. On the other hand, the concatenation-based networks' width and depth may be scaled coherently using the compound-scaling process, preserving the best possible network design even at varying sizes.

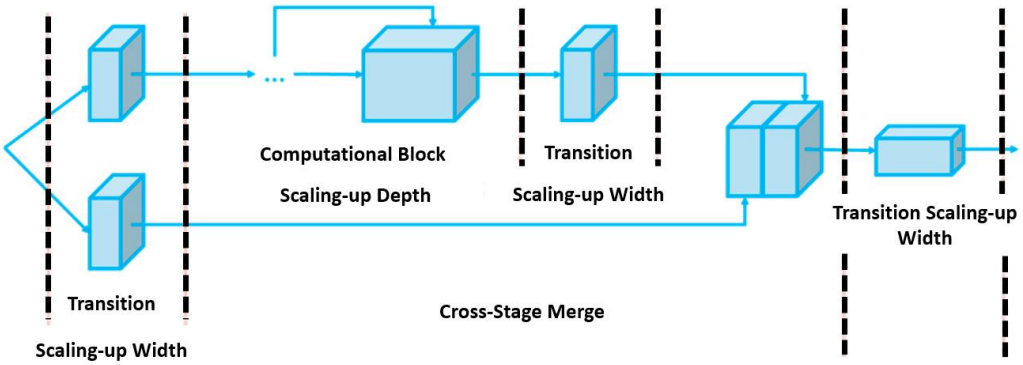


Figure 2.7 Compound Scaling of YOLOv7.

To create a more robust network, re-parameterization planning is predicated on averaging a set of model weights [47][48]. Module level re-parameterization goes even farther by allowing individual network segments to control how they parameterize themselves. Gradient flow propagation processes are employed by YOLO-v7 in order to observe which internal network modules need to implement re-parameterization procedures.

A performance comparison between YOLO-v7 and the previous YOLO versions on the MS COCO dataset is shown in Figure 2.8, demonstrate that all YOLO-v7 variations outperformed the compared object detectors in terms of accuracy and speed within the 5–160 FPS range. But as the YOLO-v7 authors point out, it's crucial to remember that none of the YOLO-v7 variations are intended for CPU-based mobile device deployment. The YOLOv7-tiny/v7/W6 variants are optimized for cloud, consumer, and edge GPUs, respectively. However, only high-end cloud GPUs are intended for use with YOLO-v7-E6/D6/E6E.

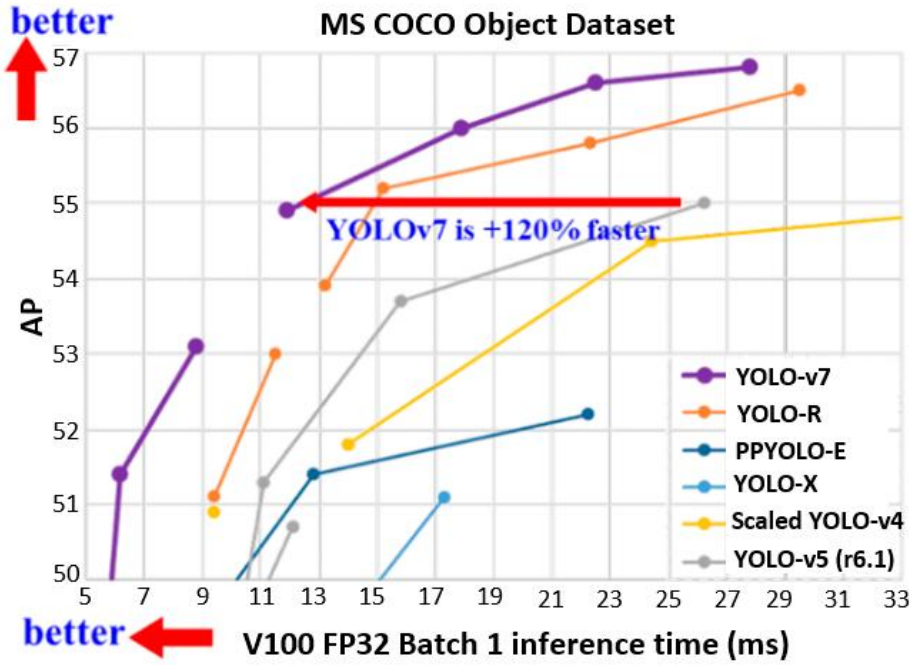


Figure 2.8 YOLOv7 vs Alternative Object Detectors [41].

Table 2.3 displays the YOLO-v7 internal variation comparison. It is clear that YOLO-v7-tiny performs significantly worse than the computationally intensive YOLO-v7-D6, at least in terms of mAP. On the other hand, it would not be appropriate for edge deployment onto a device with limited processing power.

Table 2.3 Comparison of YOLOv7 Variant

Model	Size (Pixels)	mAP (@50)	Parameters	FLOPs
YOLO-v7-tiny	640	52.8%	6.2 M	5.8G
YOLO-v7	640	69.7%	36.9 M	104.7G
YOLO-v7-X	640	71.1%	71.3 M	189.9G
YOLO-v7-E6	1280	73.5%	97.2 M	515.2G
YOLO-v7-D6	1280	73.8%	154.7 M	806.8G

2.7.3 YOLO-v8.

In January 2023, Ultralytics reported the introduction of YOLO-v8, the most recent addition to the YOLO family. YOLOv8 represents the pinnacle of YOLO models for tasks such as object detection, image classification, and instance segmentation. Its design not only enhances the ease of use for developers but also streamlines bounding box predictions through its anchor-free approach, enabling quicker non-max suppression (NMS). Drawing inspiration from YOLOv5, the YOLOv8 architecture introduces alterations in the convolution layers [49].

To lower the number of channels, the initial 6x6 convolution layer in the model's stem has been replaced by a 3x3 layer, and the 3x3 convolution layer in the bottleneck has been replaced by a 1x1 layer. The efficacy and efficiency of YOLOv8 training may be attributed to the use of mosaic augmentation, which combines 4 photos in each epoch to enable the model to learn about objects in varied locations while also dealing with partial obstructions and diverse surrounding pixels. However, in the last 10 epochs of training, mosaic augmentation is halted in order to prevent any decline in performance when evaluating the model on validation and test datasets [49].

Initial comparisons between the YOLOv8 and its version show its supremacy. Figure 2.9 shows that, when YOLO-v8 is compared to YOLO-v5 and YOLOv6 trained on 640 image resolution, all YOLO-v8 versions provide higher throughput with the same number of parameters, demonstrating hardware-efficient architectural changes. Ultralytics has presented YOLO-v8 and YOLO-v5, with YOLO-v5 providing notable real-time performance, and based on Ultralytics' initial benchmarking results, it is broadly assumed that YOLO-v8 will focus on controlled edge device deployment at high-inference speed.

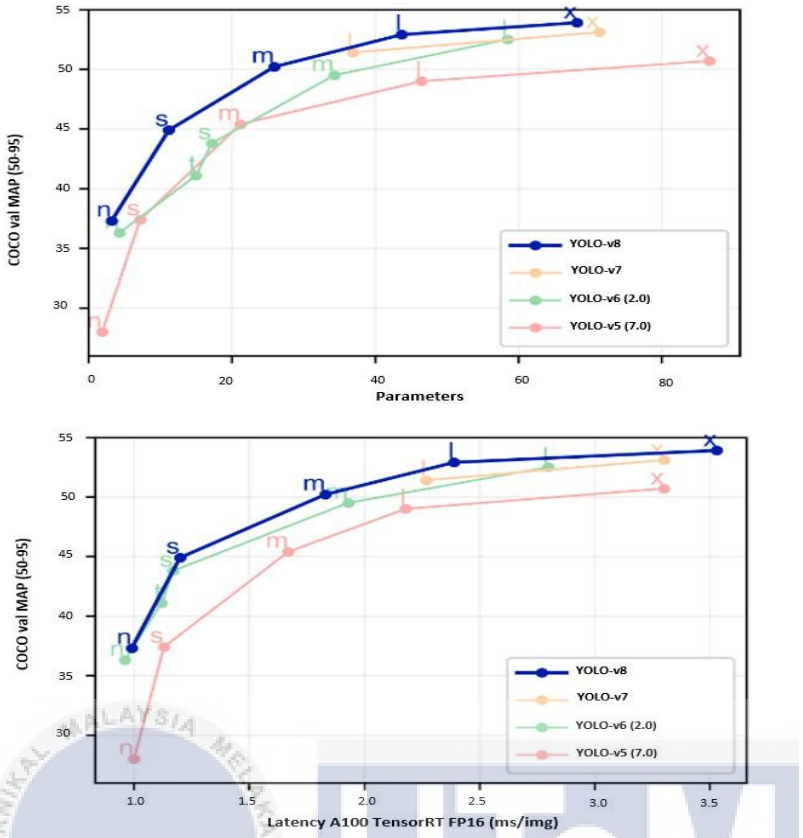


Figure 2.9 YOLOv8 vs Alternative YOLO Version.

Z. Mahboob et al. [49] study, trained all five models of YOLOv8 with YOLOv7 and YOLOv5. A fair comparison of the performance of the models undergone 50 epochs consistently on the same dataset. Although YOLOv8 have a relatively smaller number of parameters because of the network in network [50] architecture, it achieved a mAP of over 90%. The performance of models on test data were highlighted in Table 2.4, where YOLOv8m as the superior performer with a mean average precision of 96.4% on Tobset dataset, excelling other models.v

Table 2.4 Performance on Validation and Test Datasets [49]

Model	**Parameters Million (M)**	**Inference Time Milliseconds (ms)**	**mAP@0.5**
YOLOv5n	1.9M	7.5	88.3
YOLOv5s	7.2M	10.1	90
YOLOv5m	21.2M	15.5	91.2
YOLOv5l	46.5M	20.8	88.7
YOLOv5x	86.7M	39	92.2
YOLOv7	36.5M	16.8	94.5
YOLOv7x	37.19M	18	76
YOLOv8n	3.2M	9.7	93.5
YOLOv8s	11.2M	14	91.6
YOLOv8m	25.9M	19.5	96.4
YOLOv8l	43.7M	28	90. 6
YOLOv8x	68.2M	44	95.6

2.7.4 Single-Shot MultiBox Detector

Single-Shot MultiBox Detector is one of the deep learning models for object detection which detects objects from an image by using a single forward pass. Different from YOLO, SSD consists of two main components: multi-scale feature maps and convolutional predictor. Multi scale feature maps are a pre-trained model (VGG-16) that will be used to classify images. Next, SSD applies a 3x3 matrix convolution filter to each cell to make prediction.

Single-Shot MultiBox Detector has a different approach when dealing with multiple bounding boxes Single-Shot MultiBox Detector uses anchor box, a pre-trained fixed sized boxes for IOU approach when the score is more than 0.5 and based on that the initial course for bounding-box regression is set. Table 2.5 below shows a few of the differences between YOLO and SSD.

Table 2.5 Differences Between YOLO and SSD

You Only Look Once (YOLO)	**Single-Shot Multibox Detector (SSD)**
Consists of 3 steps	Consist of 2 steps
Higher speed with accuracy trades off	Slower speed but higher accuracy
Used grid-based approach for object detection	Uses anchor boxes
Able to detect small object with accuracy	Decrease in performance when detecting small object

2.7.5 Regional Convolutional Neural Network (R-CNN)

A study by Niu et al. [51] describes a deep convolutional neural network (CNN) approach to detecting traffic lights in real-time for use in self-driving vehicles. The author proposed a method which utilizes CNN to analyze images and video camera mounted on the vehicle and detect traffic lights in the scene, with an emphasis on real-time performance. The proposed method, which is the combination of CNN classifier model and ROI candidate detection algorithm will then be compared to a few of the existing object detection models to evaluate the performance. Table 2.6 below shows the result obtained from the experiment.

Table 2.6 Comparison Between Object Detection Models

Model	**IOU**	**Fps**	**Recall**
YOLOv2	27.21%	4.7%	15.15%
YOLOv2-tiny	27.65%	3.9%	19.03%
YOLOv3	19.9%	1.96%	6.9%
YOLOv3-tiny	38.13%	5.2%	32.47%
SSD	11.37%	11.37%	3.3%
Faster R-CNN	14.77%	3.5%	7.3%
Proposed Method	40.45%	10.6%	31.4%

The third version of R-CNN, Faster R-CNN were published in 2015 by Girshick et al. [28]. By incorporating a Region Proposal Network (RPN) that simultaneously predicts object bounds and objectiveness scores at each location, Faster R-CNN aims to enhance region proposal generation. The default configuration has 9 positions anchors that predict whether an image is in the foreground or background. Anchor boxes are also used to control variations in object aspect ratio and scale. Figure2.10 below illustrates the process of the Faster R-CNN Pipeline.

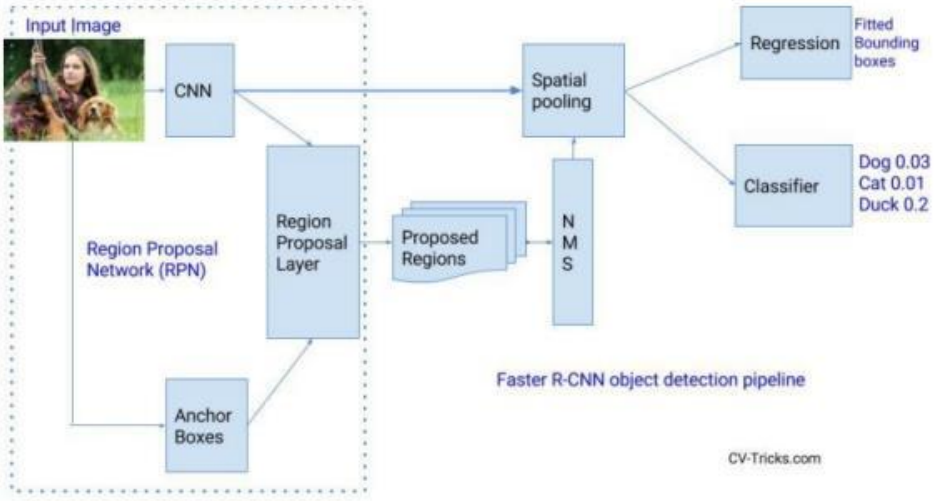


Figure 2.10 Faster R-CNN Pipeline

2.7.6 Comparison of Faster-RCNN, YOLO, and SSD.

J. Kim et al. [52] investigates a deep learning-based technique for recognizing vehicle kinds. This study presents faster-RCNN, YOLO, and SSD, which have high accuracy when dealing with real time detection. Researchers trained each algorithm on an automotive training dataset and examined the results to identify the best model for vehicle type detection. The YOLOv4 model excels other mentioned approaches with 93% accuracy as shown in Table 2.7 and Figure 2.11.

Table 2.7 Deep Learning Models Performance [52]

Models	**F1score**	**Precision**	**Recall**	**mAP**	**FPS**
YOLOv4	0.96	0.93	0.98	98.19	82.1
SSD	0.88	0.90	0.87	90.56	105.14
Faster R-CNN	0.90	0.86	0.94	93.40	36.32

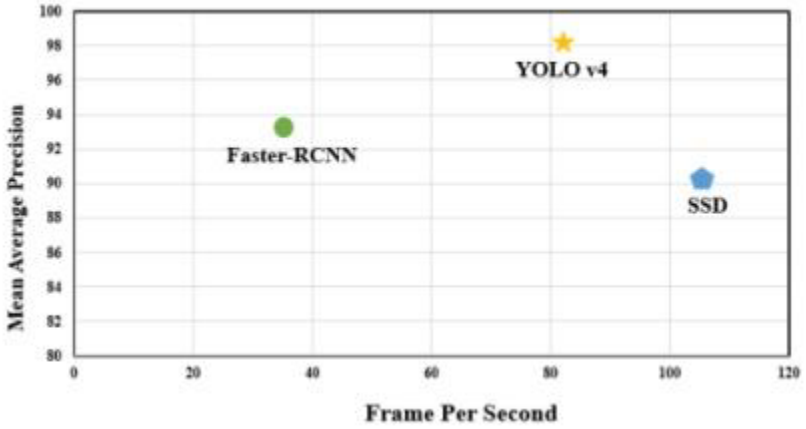


Figure 2.11 Performance Comparison

The Faster-RCNN model is the fastest among R-CNN models, however it does not have a suitable FPS since it uses CNN. On the other hand, SSD is faster, but the model is light and uses mobile-v1, resulting in lower accuracy. YOLOv4 uses FPN (Feature Pyramid Network), where features were predicted for each layer. As high-resolution features are mirrored in detection, it eliminates the drawback of not spotting small objects. Thus, YOLOv4 gave the best outcome.

2.8 Summary of Past Research

To summarize this chapter the finding from various highlighted past research were gather for better overview as shown in Table 2.7

Table 2.8 Past Research Training Algorithm

Author	Title	Algorithm/Method	Datasets	Results and findings
C. Fu, W. Liu, A. Ranga et al. [29]	DSSD: Deconvolutional Single Shot Detector	Deconvolutional Single Shot Detection (DSSD) and Single Shot MultiBox Detector (SSD).	ILSVRC CLS-LOC	When comparing the R-FCN to the SSD 513 model, it is observed that both have similar speed and accuracy. However, the DSSD 513 model offers improved accuracy, with slightly slower speed. On the other hand, the DSSD 321 model maintains a speed advantage over R-FCN with a minor decrease in accuracy. When compared to the SSD, the DSSD model demonstrates enhancements in two specific scenarios. Firstly, in scenes involving small or densely packed objects, where the small input size of SSD proves less effective. Secondly, the DSSD model outperforms SSD for certain classes that possess distinct contextual features.
C. Li, L. Li. H. Jiang et al., [35]	YOLOv6: A Single-Stage Object Detection	YOLO v5 and YOLO v6.	COCO dataset	The YOLOv6-N model achieves a 35.9% AP with a throughput of 1234 FPS on the NVIDIA

	Framework for Industrial Applications			Tesla T4 GPU. In comparison, YOLOv6-S demonstrates impressive performance, achieving a 43.5% AP at a faster speed of 495 FPS, surpassing YOLOv5-S, YOLOX-S, and PPYOLOE-S. Even the quantized version of YOLOv6-S achieving a 43.3% AP at an increased throughput of 869 FPS Additionally, the YOLOv6-M/L models outperform their predecessors, boasting better accuracy performance at 49.5% and 52.3%, respectively.
Wang, C. Y., Bochkovskiy, A., Liao, H. Y. [41]	YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors	YOLOv7, YOLOv5, YOLOv4 and Faster R-CNN	MS COCO dataset	YOLOv7 surpasses other object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS. YOLOv7 showed 69.7% AP@0.5 with 161 FPS. Followed by YOLOv5 with 50.7% AP@0.5 at 83 FPS. While, YOLOv4 has 68.2% AP@0.5 with 70 FPS. Faster R-CNN shows 44% AP@0.5 with 20 FPS.
C. Li, T. Tang, G. Wang et al. [45]	BossNAS: Exploring Hybrid CNN-transformers with Block-wisely Self-	BossNAS	ImageNet, CIFAR-10, and CIFAR-100	The hybrid CNN-transformer model achieves an impressive accuracy of 82.5% on ImageNet, surpassing EfficientNet by 2.4% while maintaining a comparable compute time.

	supervised Neural Architecture Search			
Z. Mahboob, A. Zeb, U. Khan et al. [49]	YOLO v5, v7 and v8: A Performance Comparison for Tobacco Detection in Field	YOLOv5, YOLOv7 and YOLOv8	Tobset Dataset	The YOLOv8m model stands out for its remarkable performance and precision, positioning it as a promising choice for precision application. Its swift inference time of 9.7ms further enhances its suitability for real-world deployment. However, challenges may arise in the detection of small leaves, and the accuracy of bounding boxes may be affected by the presence of weeds. In this context, YOLOv8n emerges as a viable alternative with comparable performance.
Niu, J., Liu, Y., Guizani, M., Ouyang, Z. [51]	Deep CNN-based Real-time Traffic Light Detector for Self-driving Vehicles	YOLOv2, YOLOv2 tiny, YOLOv3, YOLOv3 tiny, SSD, and Faster R-CNN	VIVA dataset	While Yolo3-tiny excels in managing the full resolution of 1280x960, Yolo2-tiny and SSD need to resize the input resolution to meet memory requirements. Although Yolo2-tiny, Yolo3-tiny, and SSD can achieve a commendable speed of up to 10 FPS the resizing (down-sampling) process

				comes with a trade-off, as it contributes to a degradation in detection performance. YOLOv2, IOU = 27.21%, Recall = 15.15% YOLOv2 tiny, IOU = 22.55%, Recall = 13.29% YOLOv3, IOU = 19.9%, Recall = 6.9% YOLOv3 tiny, IOU = 17.71%, Recall = 5.59% SSD, IOU = 11.37%, Recall = 3.3% Faster R-CNN, IOU = 14.77%, Recall = 7.3%
D. Wu, S. Jiang, E. Zhao et al. [53]	Detection of Camellia oleifera Fruit in Complex Scenes by Using YOLOv7 and Data Augmentation	YOLOv3, YOLOv5, YOLOv7 and Faster R-CNN	Custom dataset	In contrast to the YOLOv5s, YOLOv3-spp, and Faster R-CNN target detection networks, the findings indicate that the YOLOv7 model outperforms with a mean Average Precision (mAP) of 95.74%. It also boasts a remarkable F1 score of 93.67%, Precision at 94.21%, Recall at 93.13%, and an average detection time of 0.025s

2.9 Conclusion

An extensive study and benchmarking of training algorithm SSD, DSSD, R-CNN and YOLO object detectors, including YOLOv8, v7, v5 and v6, was performed. YOLOv8m is the most robust of all the mentioned object detection. In terms of real-time implementation and processing performance, YOLOv8n is a reasonable solution, with an inference time of 9.7 ms. In conclusion, the current study demonstrates that YOLOv8 is the best option for creating a detection system that requires high precision, especially when dataset availability is limited [49]. The results indicated that YOLOv8 outperformed its progenitors, with YOLOv7 showing worse performance due to alterations in its convolution layers, obtaining a mAP of 94.5% [49].

YOLOv8n's small size, with just 3.2 million parameters, makes it an ideal candidate for use in real-world applications. Despite a little discrepancy in mAP compared to the most accurate model (YOLOv8m) which contains 25.9 million parameters, YOLOv8n has a huge speed improvement, being more than twice as fast as YOLOv8m. Thus, YOLOv8 will be utilized within this project. Furthermore, the datasets used are a custom dataset of the radiator picture that collected from the assembly lines, so the data acquisition needs to be done manually. The data acquisition tool CVAT will be utilized.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The YOLOv8 model is commonly utilized in object detection. It may encounter numerous issues when detecting diverse objects and situations [53]. Therefore, researchers must enhance the model in accordance with the actual scene and adapt it to various settings. Currently, the model's improvement focus is mostly on increasing detection accuracy and speed. YOLOv8 is a relatively advanced object detector at the moment [41].

3.2 Project Overview

Here the flow of the whole project was illustrated using the flow chart as shown in Figure 3.1

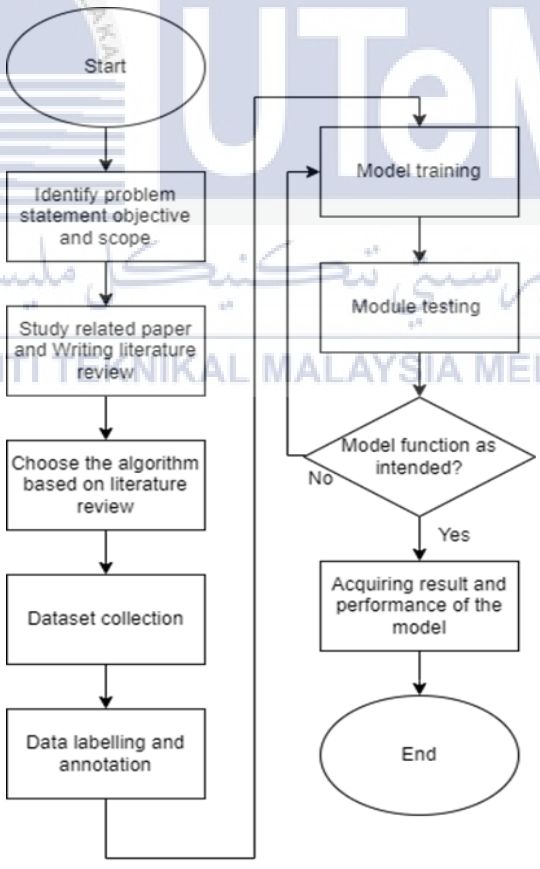


Figure 3.1 Project Flowchart

3.2.1 Training Algorithm for Object Recognition and Classification

The training process flowchart illustrated in Figure 3.2 shown the with respect to training algorithm YOLOv8n.

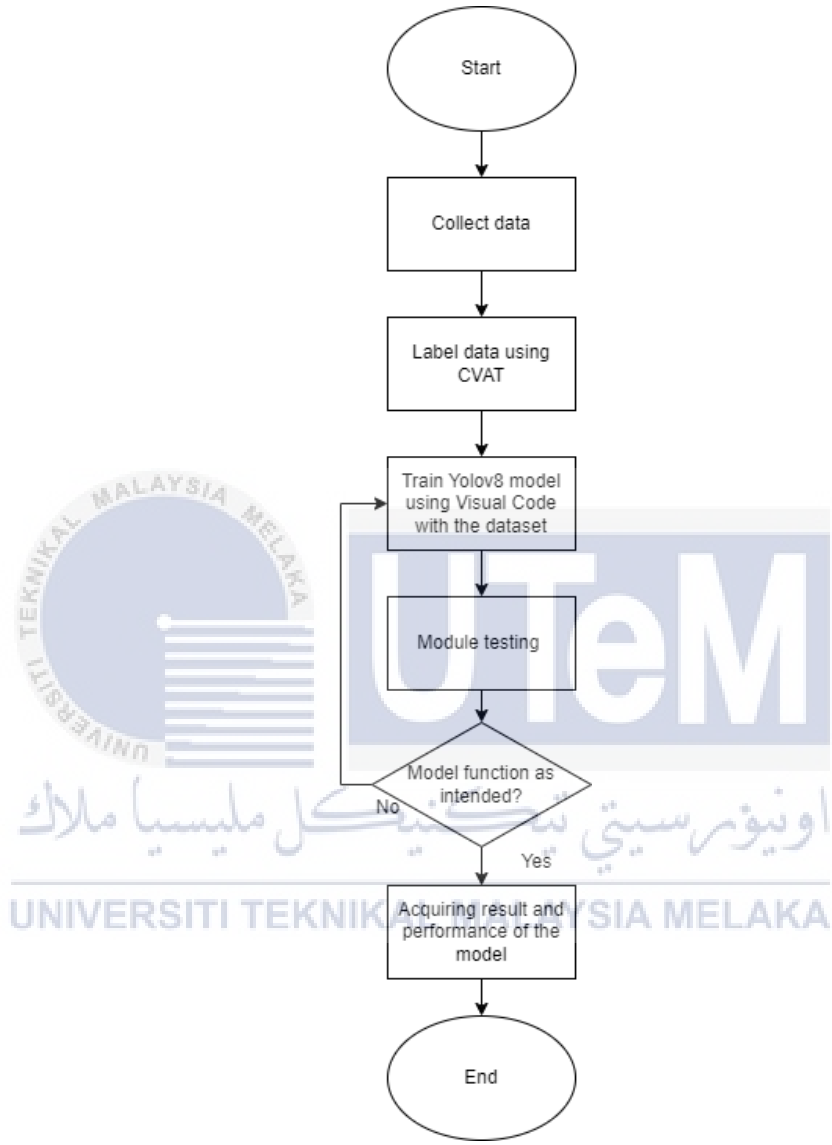


Figure 3.2 Recognition and Classification Using Artificial Intelligence Flowchart

3.2.1.1 Experimental Setup

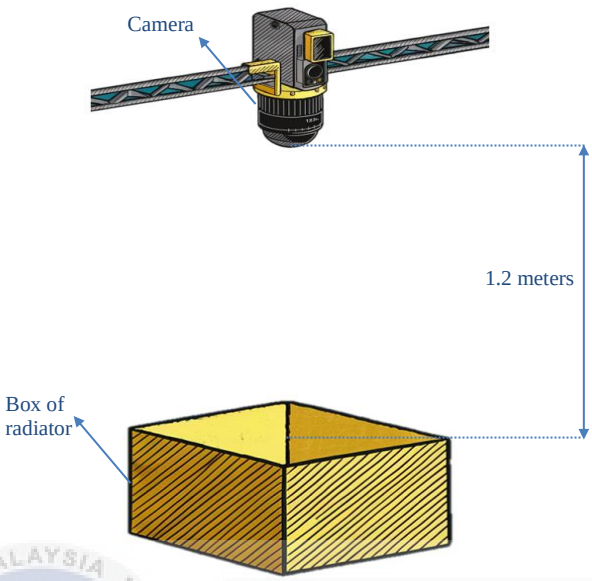


Figure 3.3 Experimental Setup

The camera is positioned at a height of 1.2 meters from the top of the box. The setup involves a controlled environment where the box of radiators is placed within a predefined area to ensure consistency. The camera is securely mounted on metal beam to maintain a stable view and is connected to a computer running the trained YOLOv8n model. Adequate lighting is provided to minimize shadows and enhance image clarity. This setup aims to simulate real-world conditions and assess the model's performance in recognizing and accurately counting the radiators from a top-down perspective.

3.2.1.2 Experimental Datasets

For datasets collection, the dataset is collected at BMW Powertrain Assembly, Sime Darby Auto Engineering the dataset collected inside the assembly line where assembly parts are placed before assembly process. The datasets collected for this study is a set of videos and images of a radiator inside a box with the imitation of the camera position (top-down perspective). The video which is recorded using mobile phone is restricted to 30 fps. Videos and images captured from above to simulate the real-world situation on the factory when operator assemble the parts as shown in Figure 3.3.

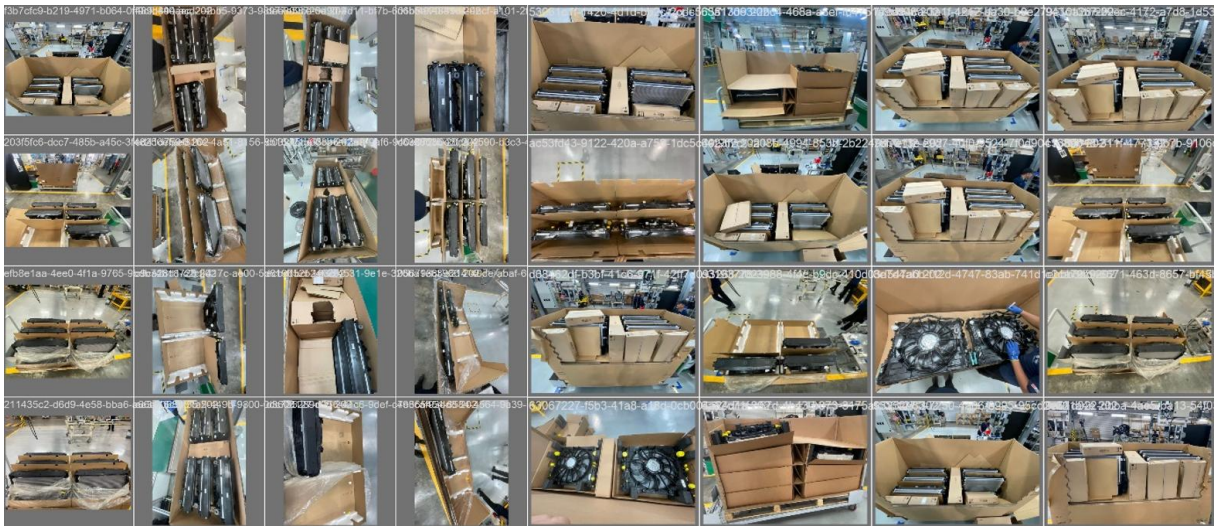


Figure 3.4 Image Datasets Overview

Consequently, the datasets used for results are generated and annotated using Computer Vision Annotation Tool (CVAT) as shown in in Table 3.1.

Table 3.1 Experiment Dataset

Dataset Specification	
Number of images	426 images
Train data	426 images
Validation data	426 images
Number of annotations	1679 annotations
Average image size	1.3 Mp
Median image ratio	1880 x 1280

3.2.1.3 Data Labelling and Annotation.

The process of giving labels or annotations to data used in machine learning and deep learning algorithms is known as data labelling and annotation. Usually, this is done to train and evaluate models, as well as to enhance their performance and accuracy. Data labelling is done manually, and the labels or annotations can take a variety of shapes, such as segmentation masks, class labels, or bounding boxes. The effectiveness of the deep learning model is significantly influenced by the calibre and volume of the labelled data. For this project, the annotation and labelling of data is conducted in Computer Vision Annotation Tool (CVAT) as shown in Figure 3.4.

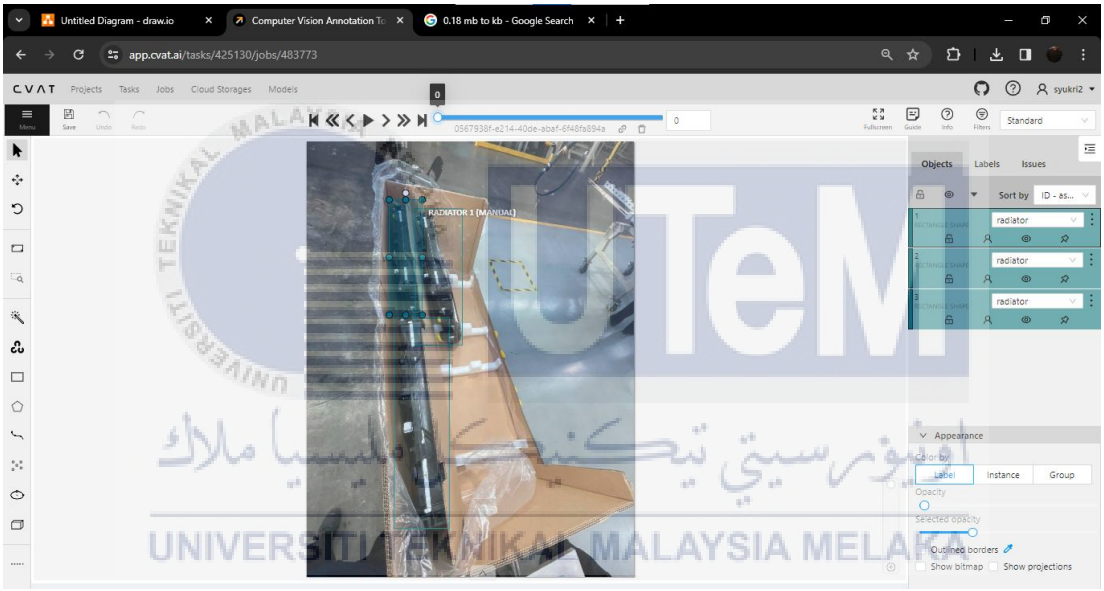


Figure 3.5 Computer Vision Annotation Tool (CVAT) Software

Figure 3.4 shows the interface in Computer Vision Annotation Tool (CVAT) which enables users to annotate custom data according to the user objectives. After the object in the image was annotated, the data then labelled. Since the object detection according to the project requirement is only one object type, so only one label was needed.

3.3 Experimental Environment

The environment setup (laptop) used to undergo the process of the algorithm training was shown in Table 3.2

Table 3.2 Experimental Setup Specification Hardware Specification

Hardware Specification	
Central Processing Unit	Intel(R) Core (TM) i5-5300U CPU @ 2.30GHz
Graphic Processing Unit	Intel(R) HD Graphics 5500
Operating System	Windows 10 Pro
Random Access Memory	8 GB DDR3
Software Specification	
Python	Version 3.11.5
Anaconda Navigator	Version 2.5.1
Visual Code	Version 1.85.1

3.4 Training Model

After data labelling and annotation has finished, the dataset is used to train the deep learning model (YOLOv8). The model will be trained using Anaconda Navigator and Visual Code in python coding language. It supports several well-known machine learning and deep learning libraries, which are also a part of AI, and they can be instantly loaded into library.

The first step in model training is to upload the labeled datasets by using the download code generated from Computer Vision Annotation Tool (CVAT). The installation of required dependencies and the cloning of YOLOv8 is conducted by using the code;

```
# To clone the yolo algorithm from ultralytics
from ultralytics import YOLO

# To Load a YOLOv8 model
model = YOLO("yolov8n.yaml")  # To build a new model from scratch
```

Then create a “yaml” file name config.yaml.;

```
# To specify the data file path
path: C:\Users\User\Documents\data
train: images/train
val: images/train
```

```
#To specify classes
names:
  0: radiator
```

After all the steps above are done, the training process of the object detection model will take place by using the code;

```
# Use the YOLOv8 model for training
# Epoch for number of cycle trains
results = model.train(data="config.yaml", epochs=100)
# Load the path of the data from yaml file
```

For video inference, the code;

```
import os
import time

from ultralytics import YOLO
import cv2

model_path  = os.path.join('.', 'runs', 'detect', 'train3', 'weights',
'best.pt')
model = YOLO(model_path)

VIDEOS_DIR = os.path.join('.', 'videos')
video_path = os.path.join(VIDEOS_DIR, 'radiator2.mov')
cap = cv2.VideoCapture(video_path)

ret = True

while ret:
    ret, frame = cap.read()

    results = model.track(frame, persist=True)
    frame_ = results[0].plot()
    cv2.imshow('frame', frame_)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
```

3.5 Evaluation Indicators of Model

The YOLO algorithm divides the input image into a grid of S x S cells and localizes each object within its associated cell, along with its respective probability score. A single regression module determines the attributes of the bounding boxes and presents them in the form of vector. Intersection over Union (IOU) metric is used to eliminate unrelated boxes. IOU:

$$IoU_{pred}^{truth} = \frac{A \cap B}{A \cup B} \quad (1)$$

Precision, Recall, Mean Average Precision (mAP) and F1 score were used to evaluate the performance of the model accurately and objectively. Precision is the most common evaluation index, and it is the number of right targets divided by the number of detected targets. In general, the higher the Precision is, the better the detection effect will be. Precision is a very intuitive evaluation index, although sometimes high Precision does not represent the truth. Consequently, mAP, Recall and F1 score were established for thorough evaluation. Precision, Recall, mAP, and F1 score were calculated as follows:

Precision:

$$P = \frac{TP}{TP + FP} \times 100\% \quad (2)$$

Recall:

$$R = \frac{TP}{TP + FN} \times 100\% \quad (3)$$

where, TP (True Positive) represents the number of objects that are correctly detected. While FP (False Positive) represents the number of other objects detected and FN (False Negative) represents the number of object that are undetected/missed.

Average Precision:

$$\mathrm{AP} = \int_0^1 \mathrm{P(r)}\ \mathrm{dr} \quad (4)$$

Mean Average Precision:

$$\mathrm{mAP} = \frac{1}{n}\sum_{i-1}^{n} \mathrm{AP}_1 \quad (5)$$

F1 score:

$$\mathrm{F1} = 2 \times \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}} \quad (6)$$

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Training Result

The result of the trained YOLOv8 algorithm was shown in Table 4.1 and Figure 4.1. Where, it will further explain in this chapter.

Table 4.1 Summary of Training Parameters and Results for Yolov8

Epoch	Training Time	Precision	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
10	1.664 hours	0.59862	0.59321	0.62497	0.27365	0.59 at 0.212
25	2.856 hours	0.92441	0.8916	0.95978	0.54239	0.91 at 0.345
50	5.461 hours	0.95921	0.95593	0.98747	0.61195	0.96 at 0.414
75	8.103 hours	0.96098	0.96816	0.98932	0.64048	0.96 at 0.404
100	15.087 hours	0.97493	0.97277	0.99043	0.6642	0.97 at 0.401

Table 4.2 Summary of Training Parameters and Results for Yolov9

Epoch	Training Time	Precision	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
10	6.651 hours	0.67825	0.6316	0.71249	0.36352	0.67 at 0.227

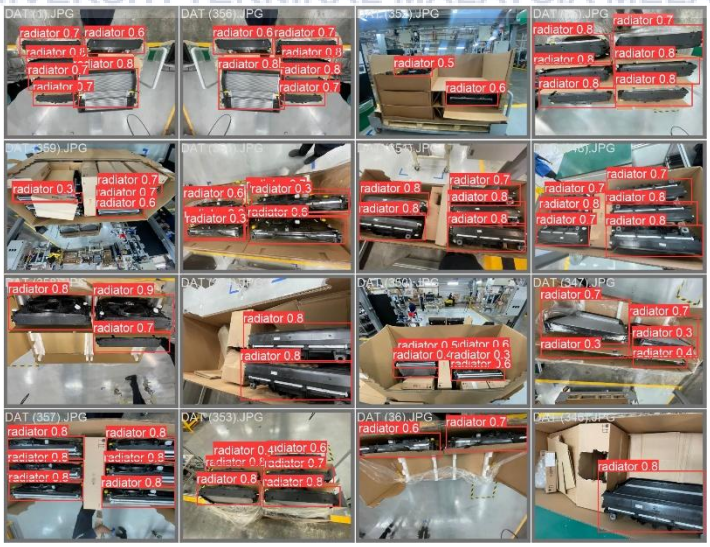


Figure 4.1 Validation image overview

4.2 Confusion Matrix

From the confusion matrix as shown in Figure 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7. is a matrix used for the evaluation of a machine learning model's performance, especially in classification tasks can be obtained. It allows for a detailed examination of how well the model is performing in terms of making predictions for different classes. The value was tabulated in Table 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8. The evaluation parameters like Precision, Recall, mAP and F1 score were calculated using this value for every epoch using equation (1), (2), (3), (4), (5) and (6).

4.2.1 Comparison for YOLOV8 Between 25, 50, 75 and 100 Epochs.

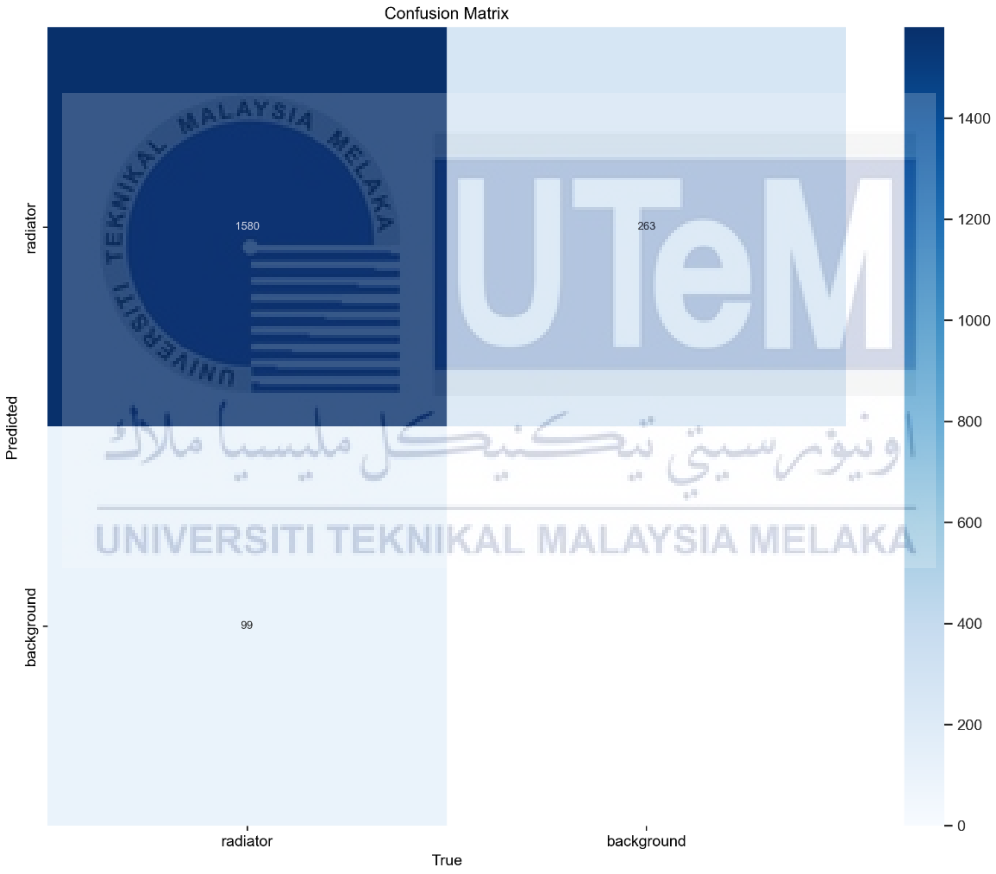


Figure 4.2 Confusion Matrix 25 epoch

Table 4.3 Confusion Matrix Values for 25 epoch

Parameters	**Value**
TP	1580
FP	263
FN	99

The YOLOv8n model, after 25 epochs of training, demonstrates a significant improvement in detection performance. The model achieved 1580 true positives (TP), accurately identifying 1580 radiators. With only 263 false positives (FP), the model shows a relatively low rate of misidentifying non-radiator objects as radiators. Additionally, the model has 99 false negatives (FN), indicating a reduced number of missed detections.

The model achieves a precision of 0.92441, indicating that 92.44% of the detected objects are correctly identified as radiators. The recall rate is 0.8916, meaning the model successfully detects 89.16% of all actual radiators present. The mean Average Precision (mAP) at an IoU threshold of 0.5 is very high at 0.95978, reflecting the model's excellent ability to balance precision and recall across various object sizes and scales. The mAP at a range of IoU thresholds (0.5:0.95) is 0.54239, showing good performance even under stricter localization criteria. The F1 score, a harmonic mean of precision and recall, is 0.91 at a threshold of 0.345, underscoring the model's robust and reliable detection capabilities. These metrics indicate that the model is highly effective in accurately identifying and locating radiators, making it a powerful tool for practical applications in this domain.

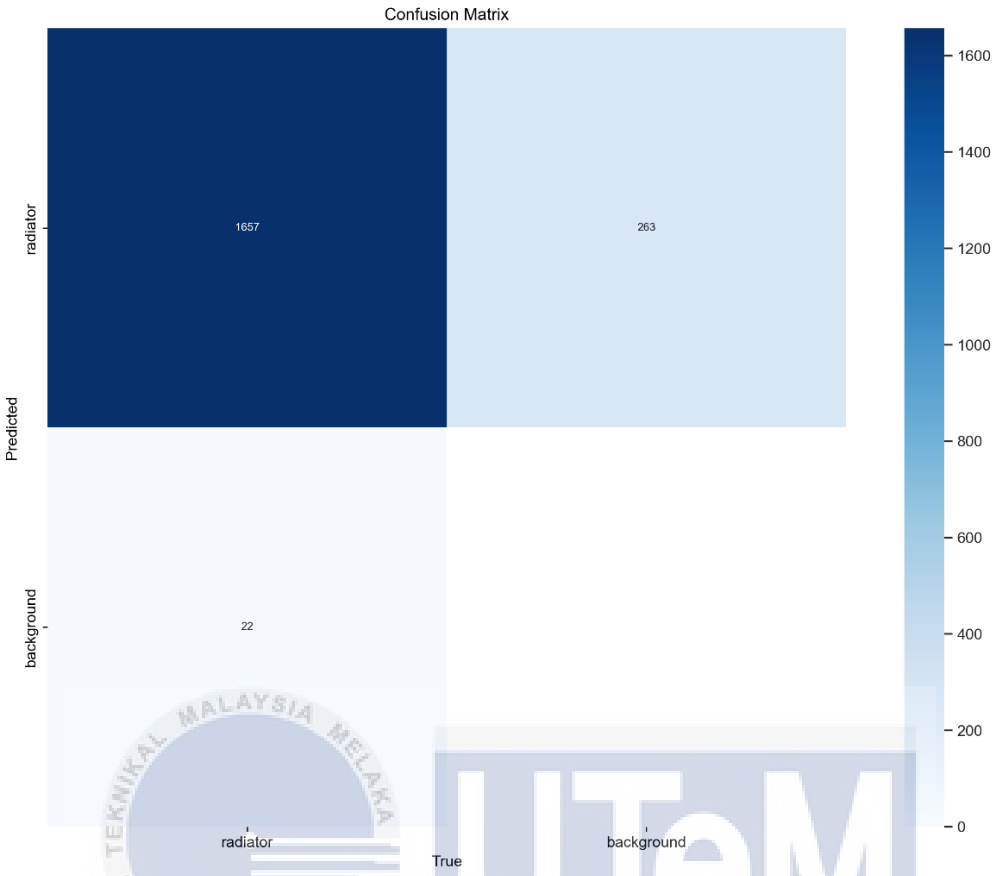


Figure 4.3 Confusion Matrix for 50 epoch

Table 4.4 Confusion Matrix Values for 50 epoch

Parameters	**Value**
TP	1657
FP	263
FN	22

After 50 epochs of training, the YOLOv8 model exhibits outstanding performance in radiator detection, as indicated by the confusion matrix values. The model achieves 1657 true positives (TP), correctly identifying 1657 radiators. With 263 false positives (FP), the model shows a consistent rate of misidentifications, which remains low relative to the number of correct detections. The model has significantly reduced false negatives (FN) to just 22, highlighting its improved ability to detect nearly all radiators present.

The precision of 0.95921 indicates that 95.92% of the detected objects are accurately identified as radiators, while the recall of 0.95593 shows that the model correctly detects 95.59% of all actual radiators present. The mean Average Precision (mAP) at an IoU threshold of 0.5 is exceptionally high at 0.98747, reflecting the model's excellent ability to balance precision and recall across different object sizes and scales. Even with stricter localization criteria, the mAP at a range of IoU thresholds (0.5:0.95) is 0.61195, indicating strong performance. The F1 score, which harmonizes precision and recall, is 0.96 at a threshold of 0.414, underscoring the model's robust and reliable detection capabilities. These highlight the model's high accuracy, minimal error rates, and overall effectiveness.

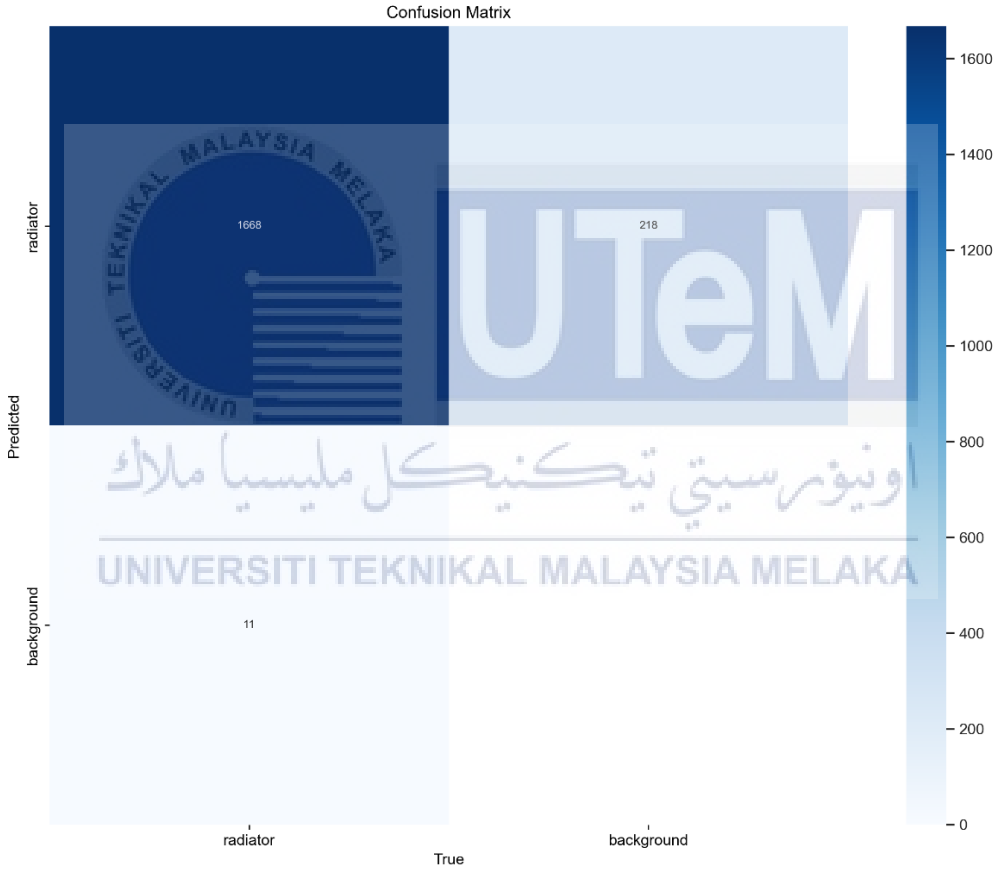


Figure 4.4 Confusion Matrix for 75 epoch

Table 4.5 Confusion Matrix Values for 75 epoch

Parameters	**Value**
TP	1668
FP	218
FN	11

Through 75 epoch, the model gives 1668 true positives (TP), the model correctly identifies 1668 radiators from the dataset. The false positive count (FP) stands at 218, implying that the model incorrectly identifies 218 non-radiator objects as radiators. Furthermore, the model exhibits a very low false negative count (FN) of 11, indicating that it misses only 11 actual radiators in the dataset.

With a precision of 0.96098, the model accurately identifies 96.098% of the detected objects as radiators. The recall rate is 0.96816, indicating that the model successfully detects 96.816% of all actual radiators present in the dataset. The mean Average Precision (mAP) at an IoU threshold of 0.5 is very high at 0.98932, reflecting the model's outstanding ability to balance precision and recall across various object sizes and scales. Even under more stringent criteria, with a mAP at a range of IoU thresholds (0.5:0.95) of 0.64048, the model maintains strong performance. The F1 score, a harmonic mean of precision and recall, is 0.96 at a threshold of 0.404, underscoring the model's robust and reliable detection capabilities. These metrics collectively highlight the model's high accuracy, comprehensive detection coverage, and consistent performance, making it an excellent choice for real-world applications requiring precise and reliable object detection.

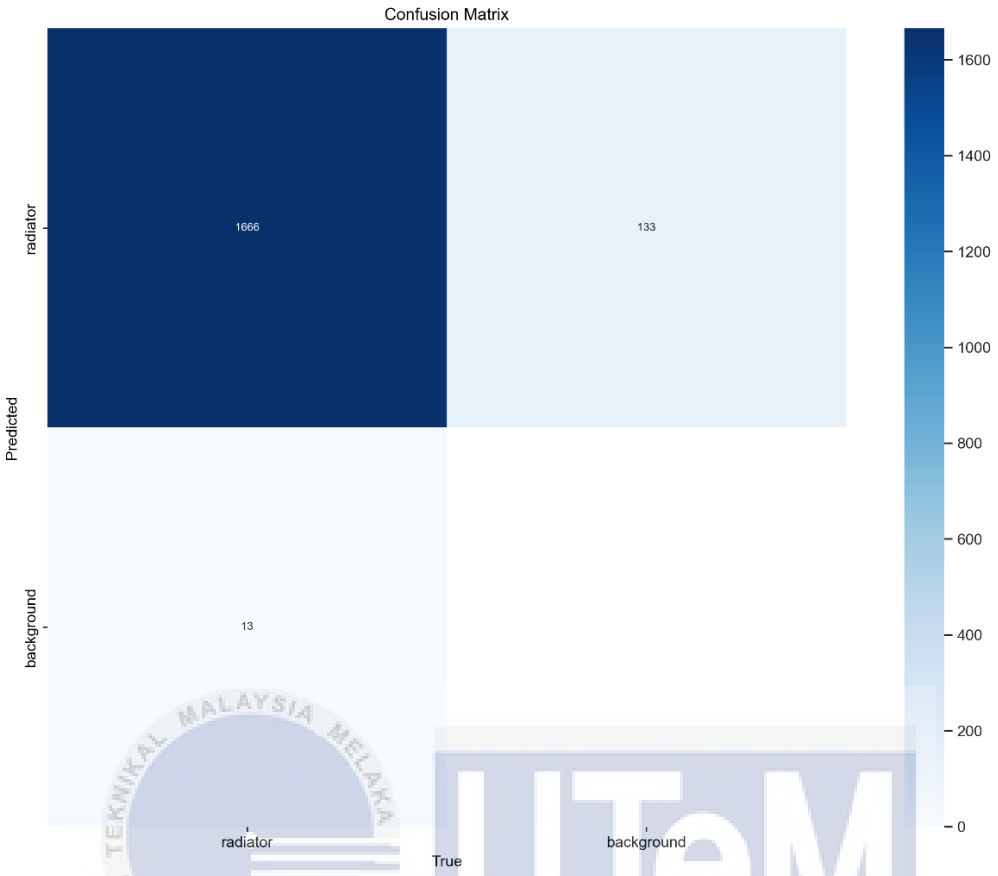


Figure 4.5 Confusion Matrix for 100 epoch

Table 4.6 Confusion Matrix Values for 100 epoch

Parameters	**Value**
TP	1666
FP	133
FN	13

The model achieved 1666 true positives (TP), accurately identifying 1666 radiators from the dataset. It also produced 133 false positives (FP), indicating instances where non-radiator objects were mistakenly identified as radiators. Furthermore, the model has a low false negative count (FN) of 13, meaning it missed only 13 actual radiators in the dataset.

These values translate to a precision of approximately 0.97493, the model accurately identifies 97.493% of the detected objects as radiators. The recall rate is 0.97277, indicating that the model successfully detects 97.277% of all actual radiators present in the dataset. The mean Average Precision (mAP) at an IoU threshold of 0.5 is very high at 0.99043, showcasing the model's outstanding ability to balance precision and recall across various object sizes and scales. Even under stricter criteria, with a mAP at a range of IoU thresholds (0.5:0.95) of 0.6642, the model maintains strong performance. The F1 score, a harmonic mean of precision and recall, is 0.97 at a threshold of 0.401, underscoring the model's robust and reliable detection capabilities.

4.2.2 Comparison Between YOLOV8 and YOLOV9 with 10 Epochs.

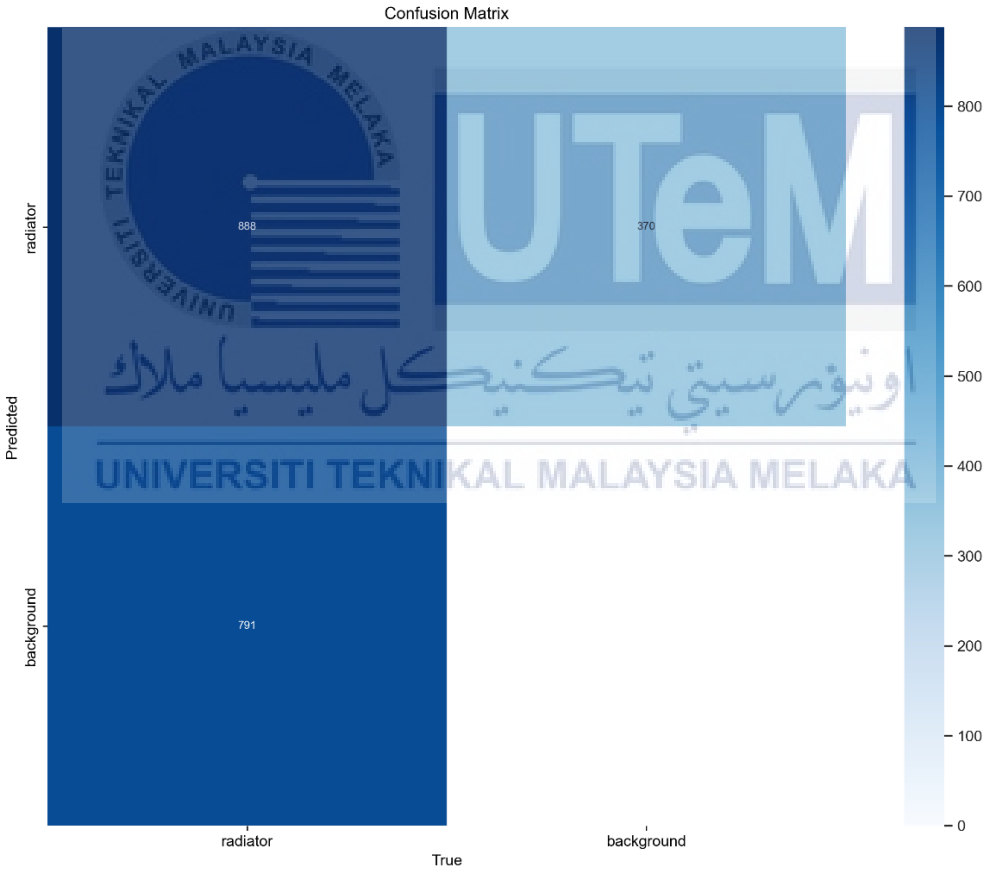


Figure 4.6 Confusion Matrix of Yolov8 for 10 epoch

Table 4.7 Confusion Matrix Values of Yolov8 for 10 epoch

Parameters	**Value**
TP	888
FP	370
FN	791

The model with 10 epochs for Yolov8 gives a mixed performance. With 888 true positives (TP), the model successfully identified 888 radiators correctly. However, the model also produced 370 false positives (FP), indicating instances where it mistakenly identified non-radiator objects as radiators. Additionally, there were 791 false negatives (FN), meaning the model failed to detect 791 actual radiators.

This model trained for 10 epochs gives a precision of 0.59862, the model accurately identifies approximately 59.86% of the detected objects as true radiators. Its recall rate is 0.59321, indicating that it correctly detects 59.32% of all actual radiators present in the dataset. The mean Average Precision (mAP) at an IoU threshold of 0.5 is 0.62497, reflecting the model's ability to balance precision and recall for different object sizes and scales, while the mAP at a range of IoU thresholds (0.5:0.95) is lower at 0.27365, suggesting decreased performance for more stringent localization criteria. The F1 score, a harmonic mean of precision and recall, is 0.59, showing a moderate level of overall performance. This suggests that while the model is fairly competent in detecting radiators, there is room for improvement in enhancing both precision and recall, achieving more robust and reliable detection results.

Figure 4.7 Confusion Matrix of Yolov9 with 10 epoch

Table 4.8 Confusion Matrix Values of Yolov9 with 10 epoch

Parameters	Value
TP	954
FP	481
FN	618

Meanwhile for trained Yolov9 model with 10 epoch, the model achieved 954 true positives (TP), meaning it accurately identified 954 radiators. However, it also produced 481 false positives (FP), indicating instances where the model mistakenly identified non-radiator objects as radiators. Additionally, the model has 618 false negatives (FN), which means it failed to detect 618 actual radiators in the dataset.

The precision of 0.67825 indicates that 67.825% of the objects identified as radiators are correctly classified, showing the model's accuracy in making positive identifications. The recall of 0.6316 suggests that the model detects 63.16% of all actual radiators present in the dataset, indicating some missed detections. The mean Average Precision (mAP) at an IoU threshold of 0.5 is 0.71249, reflecting a good balance of precision and recall for different object sizes and scales within this threshold. However, the mAP at a range of IoU thresholds (0.5:0.95) drops to 0.36352, indicating a decrease in performance under stricter localization criteria, which means the model struggles more with precise object localization. The F1 score of 0.67 at a threshold of 0.227, which balances precision and recall, points to an overall moderate performance level.

However, despite the small differences in performance between the 10-epoch models of YOLOv8 and YOLOv9, YOLOv8 is more favorable due to its significantly shorter training time. YOLOv8 requires only 1.664 hours for 10 epochs, compared to the much longer 6.651 hours for YOLOv9. This substantial difference in training time highlights YOLOv8's efficiency, making it a better choice for rapid prototyping and iterative model refinement, thus enabling faster development and deployment in practical applications.

4.3 F1 Confidence Curve

The F1 curve, a metric used to evaluate the balance between precision and recall in classification tasks, represents the harmonic mean of precision and recall values at different thresholds. Where an F1 score of 1.0 indicates a perfect classifier, and a score of 0.0 indicates a classifier that is making no correct predictions. On the other hand, threshold is the point at which the model decides whether a prediction should be classified as positive or negative. A lower threshold generally leads to more positive predictions.

4.3.1 Comparison for YOLOV8 Between 25, 50, 75 and 100 Epochs.

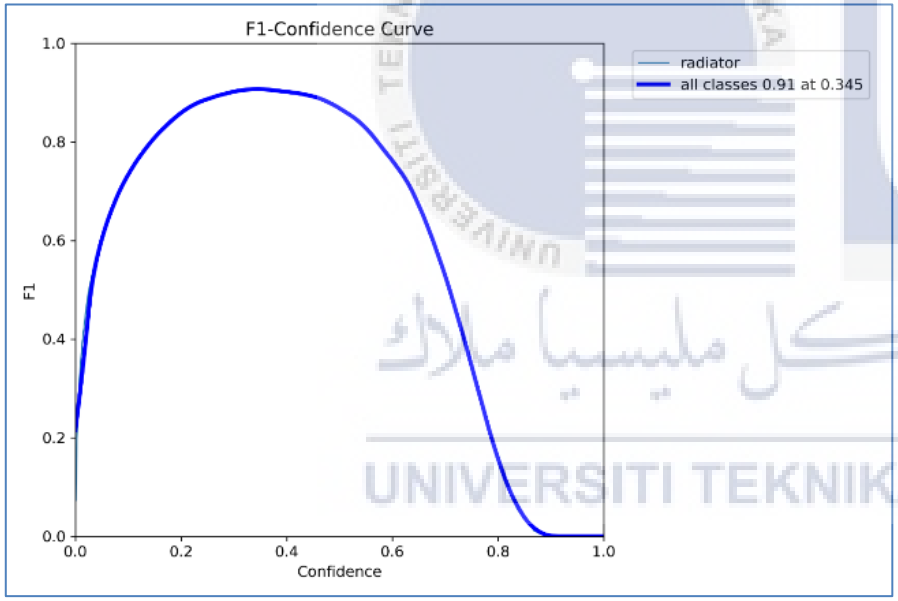


Figure 4.8 F1 Confidence Curve for 25 Epoch

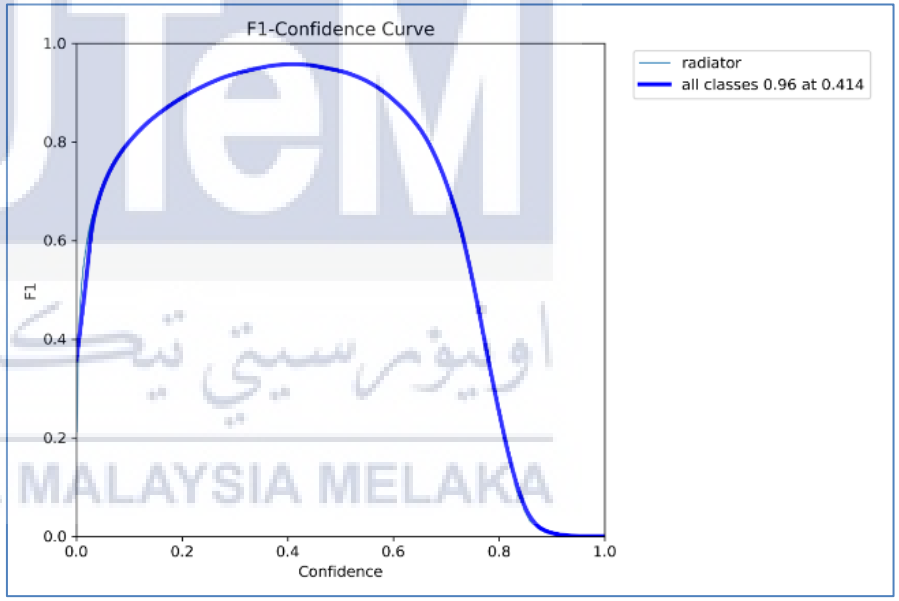


Figure 4.9 F1 Confidence Curve for 50 Epoch

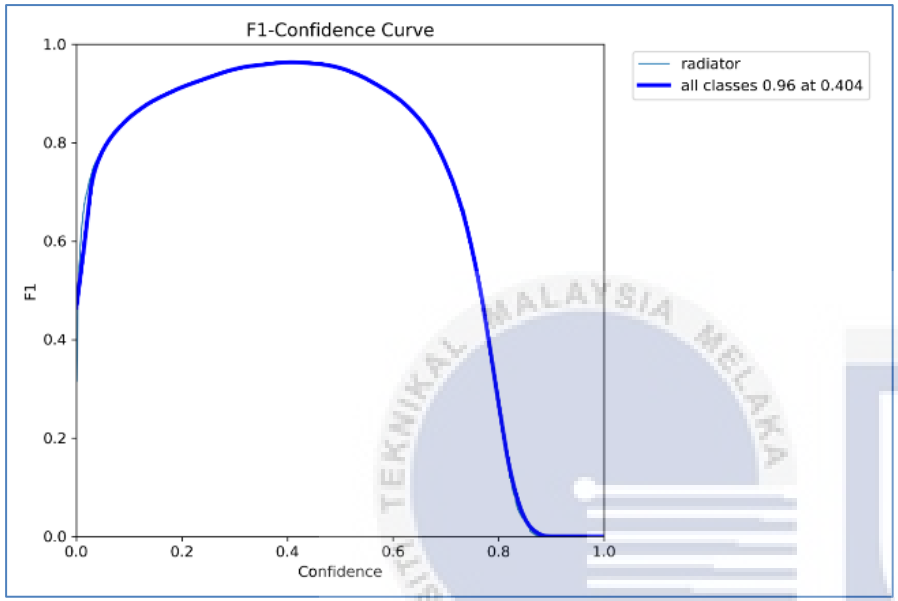


Figure 4.10 F1 Confidence Curve for 75 Epoch

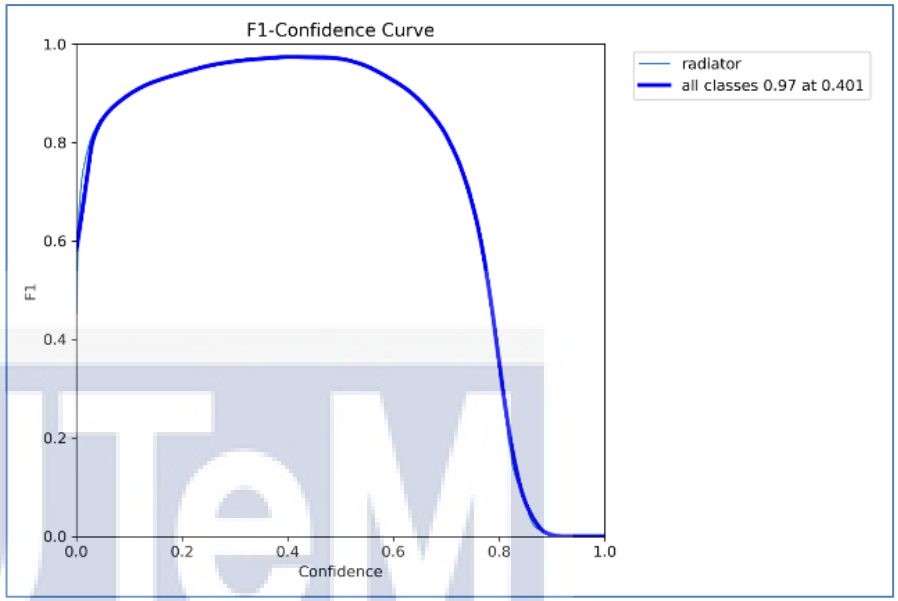


Figure 4.11 F1 Confidence Curve for 100 Epoch

A high F1 score, such as 0.91 in Figure 4.8, indicates that the model's predictions are both accurate (high precision) and comprehensive (high recall) at this particular confidence threshold of 0.345. Adjusting the confidence threshold can impact the trade-off between precision and recall, a higher threshold may yield higher precision but lower recall, while a lower threshold may increase recall but decrease precision.

At 50 epochs, the model maintains high accuracy in both correctly identifying radiators and minimizing missed detections at this threshold. A confidence threshold of 0.414 in Figure 4.9 means that the model assigns a probability of at least 41.4% to a detected object being a radiator before confirming it as such. This relatively moderate threshold helps balance between false positives and false negatives, ensuring that the model is neither too lenient nor too strict in its detections. As a result, the model achieves an excellent F1 score of 0.96, demonstrating its effectiveness in providing reliable and consistent detection results. Following with slightly higher F1 score of 0.97 with higher epoch

Additionally, the mode trained with 75 epochs also attains an F1 score of 0.96 in Figure 4.10, but at a slightly lower confidence threshold of 0.404. This suggests that with additional training, the model has become slightly more confident in its predictions, requiring a marginally lower threshold to maintain the same high level of precision and recall. This indicates improved robustness and a slight enhancement in the model's ability to generalize well to new data.

Furthermore, with 100 epochs the model's performance further improves in Figure 4.11, achieving an F1 score of 0.97 at an even lower confidence threshold of 0.401. This indicates that the model has significantly increased its confidence in making accurate predictions while maintaining a very high level of precision and recall. The lower confidence threshold combined with the higher F1 score reflects a substantial improvement in the model's detection capabilities, making it highly reliable and effective for practical applications. This progressive enhancement in performance underscores the benefits of extended training for fine-tuning and optimizing the model.

4.3.2 Comparison Between YOLOV8 and YOLOV9 with 10 Epochs.

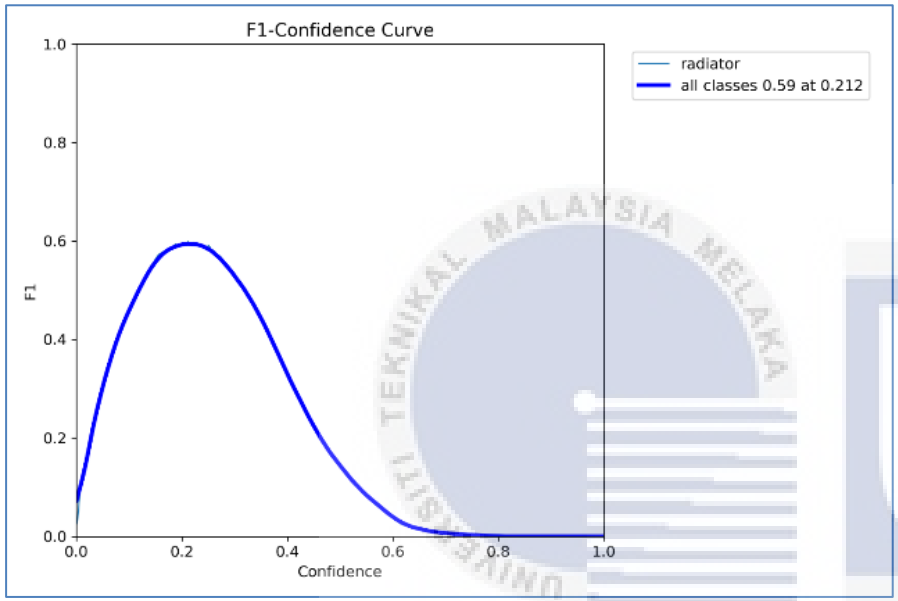


Figure 4.12 F1 Confidence Curve for 10 Epoch with Yolov8

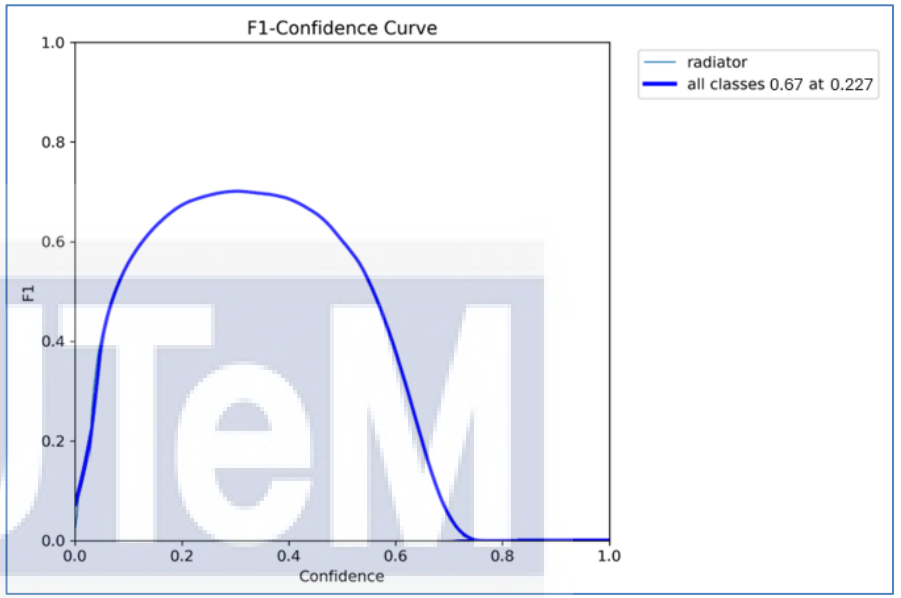


Figure 4.13 F1 Confidence Curve for 10 Epoch with Yolov9

The F1 curve as shown in Figure 4.12 generated after training the model give the F1 score value of 0.59 at a threshold of 0.212. This shows that the classifier is performing reasonably good and balanced performance. It indicates that the classifier has a moderate to high level of precision and recall at 0.212 threshold.

On the other hand, YOLOv9 model trained with 10 epochs gives 0.67 F1 score at threshold of 0.227. The confidence threshold of 0.227 is relatively low, suggesting that the model is set to make predictions even when it is not highly confident about its classifications. A low confidence threshold means the model is more likely to make positive identifications, resulting in higher recall but potentially lower precision. This is reflected in the moderate F1 score of 0.67, which shows that the model is detecting a fair number of true positives but is also generating a significant number of false positives. The curve indicates that while the model can identify a good number of actual radiators, it also has a tendency to incorrectly classify non-radiator objects as radiators.

Despite the slightly improved performance gap between the 10-epoch models of YOLOv8 and YOLOv9, YOLOv8 emerges as the preferable option due to its vastly shorter training time. Training YOLOv8 for 10 epochs takes only 1.664 hours, while YOLOv9 requires a significantly longer 6.651 hours. This stark contrast underscores YOLOv8's efficiency and suitability for rapid development cycles. The reduced training time of YOLOv8 allows for quicker experimentation and model refinement, making it a more practical choice for real-world applications where time is a critical factor.

4.4 Precision-Recall (PR) Curve

A Precision-Recall (PR) curve is used to evaluate the performance of a classification model, particularly in scenarios where the classes are imbalanced. It shows the trade-off between precision and recall at different thresholds. This metric is commonly used in object detection tasks, where it evaluates how well the predicted bounding boxes match the ground truth boxes. Mean Average Precision (mAP) is an average of the precision values calculated at different recall levels. It summarizes the PR curve into a single value, providing an overall assessment of model performance across various thresholds. Intersection over Union (IoU) threshold measures the overlap between predicted and ground truth bounding boxes. An IoU of 0.5 means the predicted box overlaps by at least 50% with the ground truth box.

4.4.1 Comparison for YOLOV8 Between 25, 50, 75 and 100 Epochs

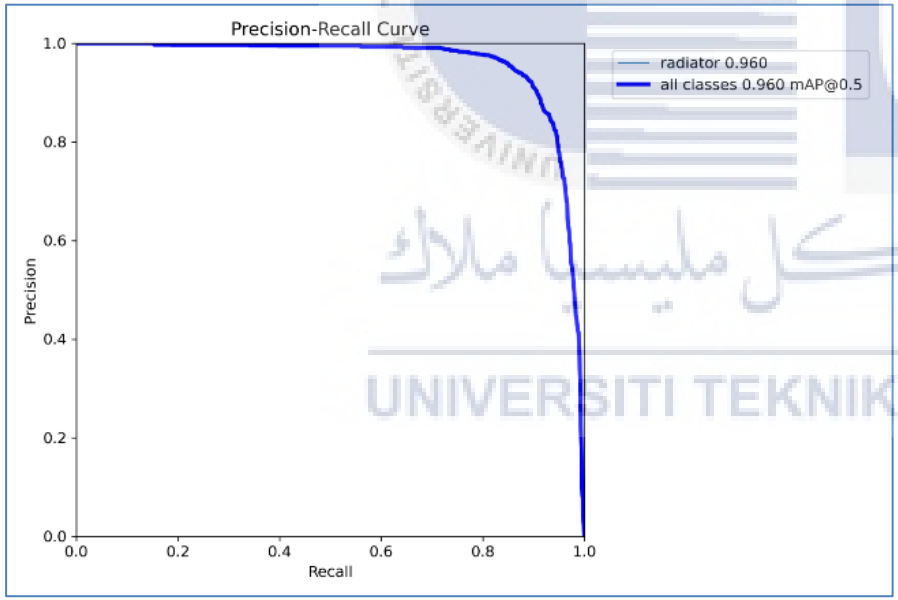


Figure 4.14 Precision-Recall (PR) Curve for 25 Epoch

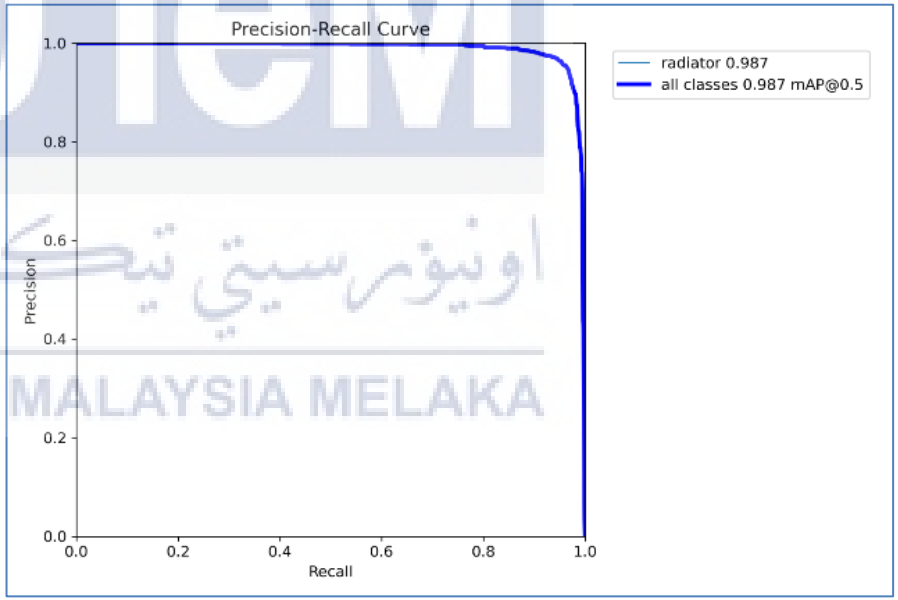


Figure 4.15 Precision-Recall (PR) Curve for 50 Epoch

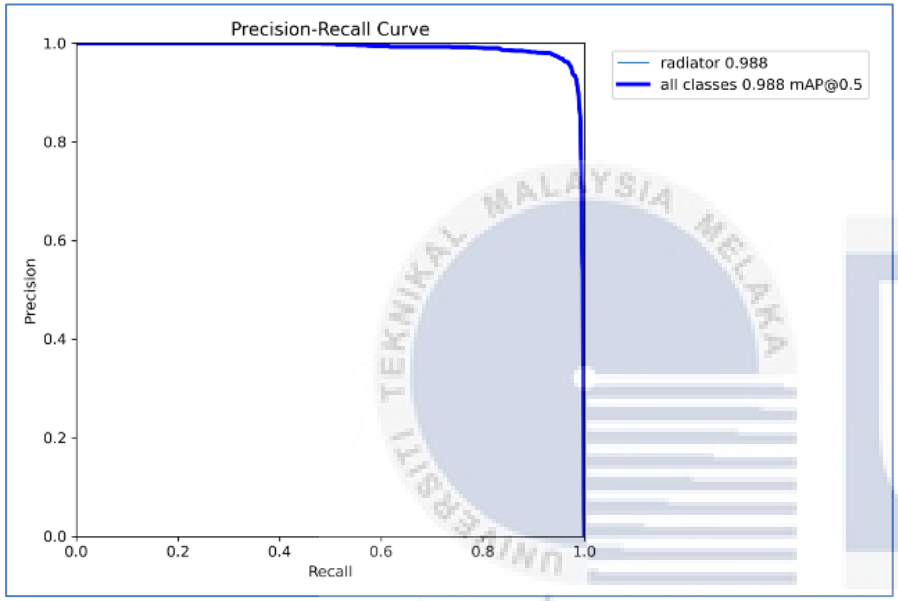


Figure 4.16 Precision-Recall (PR) Curve for 75 Epoch

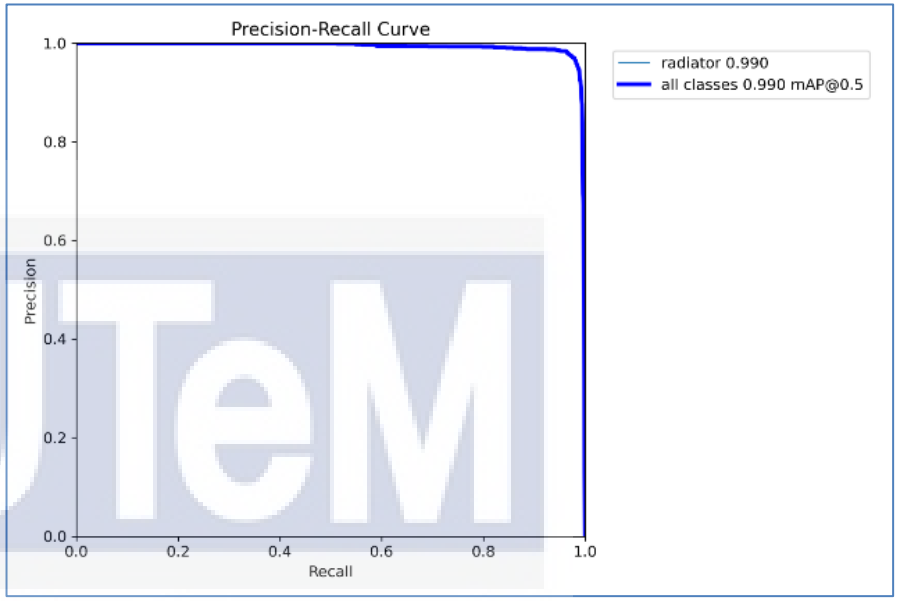


Figure 4.17 Precision-Recall (PR) Curve for 100 Epoch

The precision-recall curve for the YOLOv8 model trained over different epochs demonstrates a clear trend of improvement in both precision and recall as the number of epochs increases. At 25 epochs, the model achieves a precision of 0.92441 and a recall of 0.8916, indicating a strong ability to correctly identify true positives while minimizing false positives. As training continues to 50 epochs, both metrics improve significantly, with precision reaching 0.95921 and recall climbing to 0.95593, showcasing enhanced accuracy and robustness in detection.

Further training to 75 epochs results in a slight increase in precision to 0.96098 and a more noticeable rise in recall to 0.96816, suggesting that the model is becoming increasingly proficient at identifying almost all relevant instances while still maintaining high accuracy. Finally, at 100 epochs, the model achieves an outstanding precision of 0.97493 and recall of 0.97277. This high level of performance reflects the model's exceptional capability to not only detect objects accurately but also to minimize false negatives and false positives effectively.

Overall, the precision-recall curve of the YOLOv8 model illustrates a steady enhancement in detection performance with more training epochs, highlighting its potential for highly accurate object detection tasks. The model's ability to maintain high precision while significantly improving recall suggests a well-balanced and effective training process, making it a reliable tool for practical applications requiring precise and comprehensive object detection.

4.4.2 Comparison Between YOLOV8 and YOLOV9 with 10 Epochs.

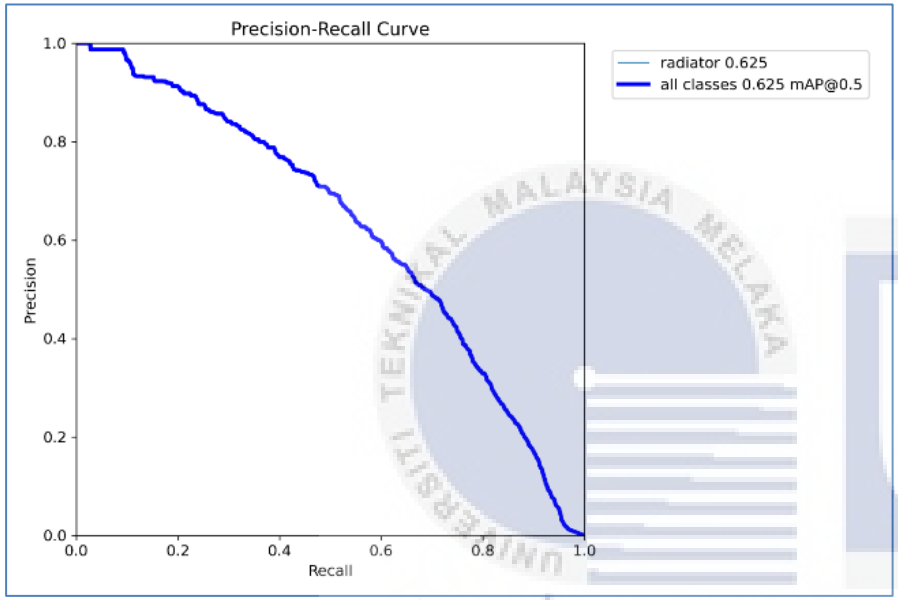


Figure 4.18 Precision-Recall (PR) Curve for 15 Epoch Using YOLOV8

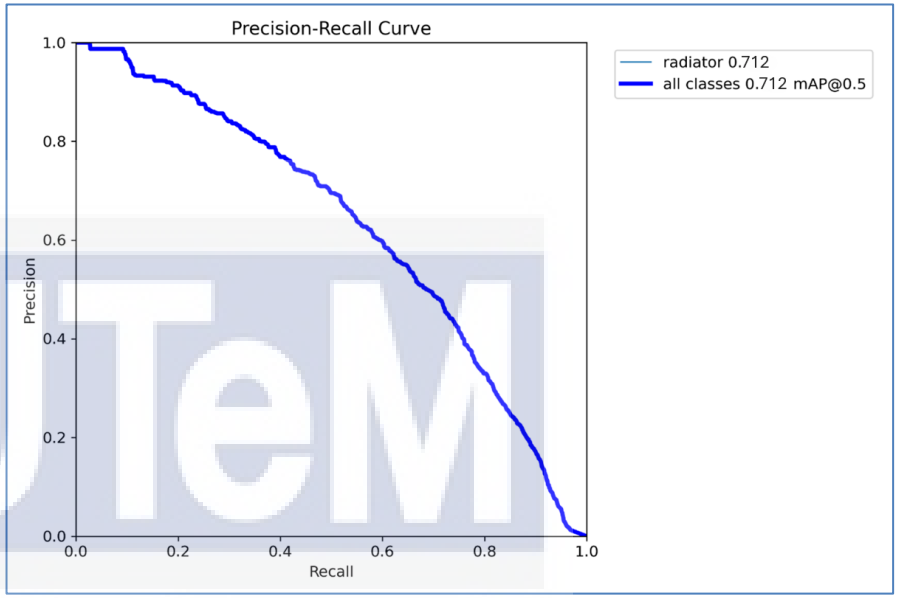


Figure 4.19 Precision-Recall (PR) Curve for 10 Epoch Using YOLOV9

The precision-recall evaluation for the 10-epoch models of YOLOv8 and YOLOv9 reveals notable insights into their initial performance and potential areas for improvement. At 10 epochs, YOLOv8 achieves a precision of 0.59862 and a recall of 0.59321. These values indicate a moderate ability to correctly identify true positives while minimizing false positives, but there is still substantial room for improvement in both precision and recall. The model's performance at this stage suggests that while it can detect relevant objects, it may also produce a considerable number of false negatives and false positives.

In comparison, YOLOv9 at 10 epochs exhibits a higher precision of 0.67825 and a slightly better recall of 0.6316. This indicates that YOLOv9, with the same number of epochs, is somewhat better at correctly identifying true positives and reducing false positives compared to YOLOv8. The higher precision and recall suggest that YOLOv9 may have a more effective initial learning process, potentially due to different architecture or hyperparameter settings. However, it's crucial to consider the training time alongside these metrics. YOLOv9, despite its slightly better initial performance, requires significantly longer training time compared to YOLOv8. This difference in training efficiency can be critical in practical scenarios where time is important.

4.5 Training Loss

Training Loss (“train/box_loss”, “train/cls_loss” and “train/dfl_loss”) refers to the error or loss calculated during the training phase of a machine learning model. It quantifies how well the model is performing on the training data. The loss is calculated using a chosen loss function that measures the difference between the predicted values and the actual values. Then again, Validation (or Test) Loss (val/box_loss”, val/cls_loss”, val/dfl_loss”,) refers to the error or loss calculated on a separate dataset that the model hasn't seen during training. This set is used to evaluate the model's generalization and how well it performs on new, unseen data. Similar to training loss, the validation loss is computed using the same loss function.

Thence, all the graphs show that both of the training and validation loss are decreasing gradually as the training progresses, it suggests that the model is improving its performance as shown in Figure 4.20, 4.21, 4.22, 4.23, 4.24, and 4.25. This decline signifies that the model is learning more about the underlying patterns in the data and is becoming better at making predictions. On account of this, the precision, recall and mean Average Precision (mAP) graphs also increasing gradually over the course of training also indicate it is improving in making prediction.

4.5.1 Comparison for YOLOV8 Between 25, 50, 75 and 100 Epoch

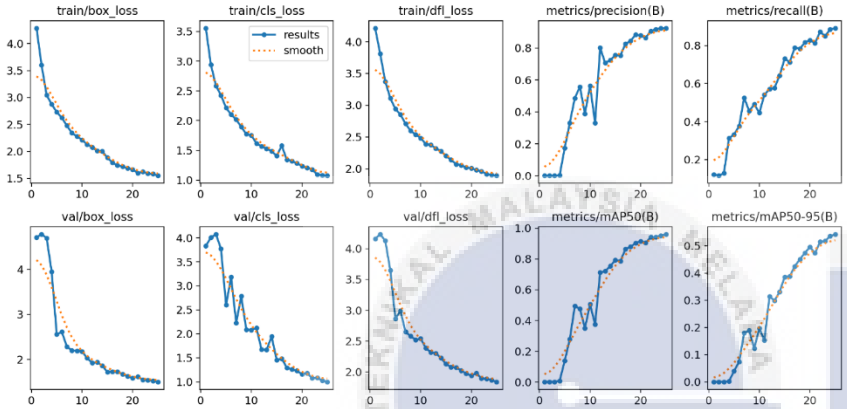


Figure 4.20 Training Loss, Validation Loss for 25 Epoch.

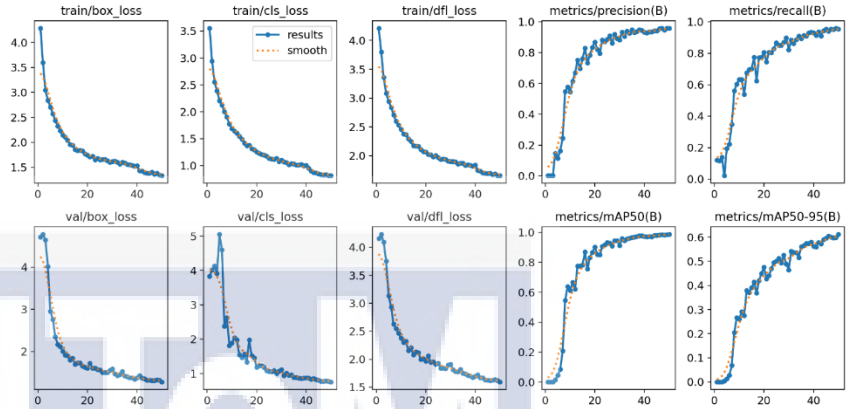


Figure 4.21 Training Loss, Validation Loss for 50 Epoch.

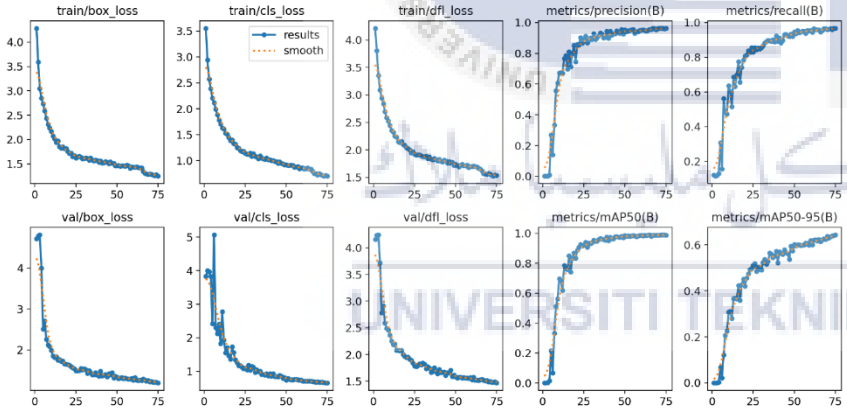


Figure 4.22 Training Loss, Validation Loss for 75 Epoch.

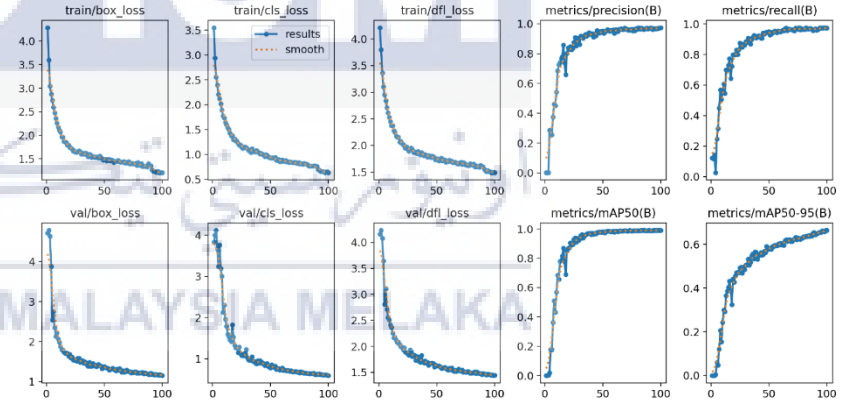


Figure 4.23 Training Loss, Validation Loss for 100 Epoch.

Table 4.9 Last 5 Batch of 25 Epoch

Epoch	Train/box_loss	Train/cls_loss	Train/dfl_loss	Val/box_loss	Val/cls_loss	Val/dfl_loss
20	1.6648	1.2368	2.0109	1.5821	1.1593	1.9409
21	1.6008	1.2042	1.98	1.6165	1.1854	1.9782
22	1.6247	1.1784	1.9607	1.5466	1.0765	1.8956
23	1.589	1.0949	1.918	1.539	1.0842	1.8892
24	1.5822	1.0822	1.9025	1.5232	1.0254	1.8646
25	1.5507	1.0747	1.8925	1.5004	1.0007	1.8386

Table 4.10 Last 5 Batch of 50 Epoch

Epoch	Train/box_loss	Train/cls_loss	Train/dfl_loss	Val/box_loss	Val/cls_loss	Val/dfl_loss
45	1.3845	0.8463	1.691	1.323	0.80141	1.6269
46	1.4033	0.83674	1.6987	1.3124	0.78005	1.6192
47	1.3592	0.82837	1.6752	1.309	0.77542	1.6279
48	1.3802	0.83694	1.6875	1.325	0.78864	1.6377
49	1.3491	0.81703	1.6578	1.3178	0.77353	1.6206
50	1.3329	0.81371	1.6548	1.2811	0.75837	1.5957

Table 4.11 Last 5 Batch of 75 Epoch

Epoch	Train/box_loss	Train/cls_loss	Train/dfl_loss	Val/box_loss	Val/cls_loss	Val/dfl_loss
70	1.2904	0.74733	1.5705	1.2162	0.68001	1.4871
71	1.2834	0.72914	1.5556	1.2244	0.68174	1.4992
72	1.2737	0.7158	1.536	1.2073	0.67585	1.4826
73	1.2578	0.70433	1.5235	1.2071	0.67026	1.4806
74	1.2751	0.71625	1.5446	1.208	0.66451	1.4925
75	1.2477	0.70073	1.5379	1.1939	0.66311	1.4722

Table 4.12 Last 5 Batch of 100 Epoch

Epoch	Train/box_loss	Train/cls_loss	Train/dfl_loss	Val/box_loss	Val/cls_loss	Val/dfl_loss
95	1.2156	0.65209	1.5115	1.1812	0.61422	1.46
96	1.2363	0.65454	1.5132	1.1736	0.61479	1.4578
97	1.2088	0.63215	1.4932	1.159	0.61001	1.4463
98	1.2112	0.64612	1.4918	1.1753	0.61039	1.4534
99	1.2113	0.65144	1.5001	1.1579	0.60069	1.4416
100	1.2072	0.63557	1.4894	1.1562	0.59765	1.4459

Focusing on the last five batches of epochs 25, 50, 75, and 100. At Epoch 25, the training losses for box, classification (cls), and distribution focal loss (dfl) are relatively high, with values of 1.6648, 1.2368, and 2.0109, respectively. The validation losses are slightly lower but still high, indicating that the model is still learning basic patterns.

By Epoch 50, the training losses have significantly decreased to 1.3329 for box, 0.81371 for cls, and 1.6548 for dfl, showing improvement in the model's learning. The validation losses also decreased, indicating that the model is better at generalizing from the training data to new data.At Epoch 75, the training losses further decrease to 1.2477 (box), 0.70073 (cls), and 1.5379 (dfl). The validation losses also continue to drop, though the rate of decrease is slower, suggesting that the model is approaching optimal performance.

Finally, by Epoch 100, the training losses are at their lowest, with box loss at 1.2072, cls loss at 0.63557, and dfl loss at 1.4894. Validation losses follow the same trend, indicating that the model has learned well. Overall, the progressive decrease in losses across these epochs demonstrates the YOLOv8 model's learning curve and highlights the optimal model's performance.

4.5.2 Comparison Between YOLOV8 and YOLOV9 with 10 Epochs.

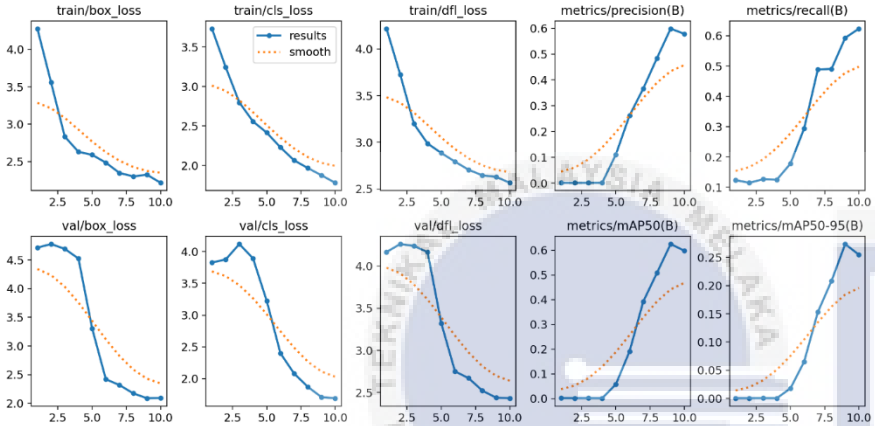


Figure 4.24 Training Loss, Validation Loss Precision and Recall for 10 Epoch with Yolov8.

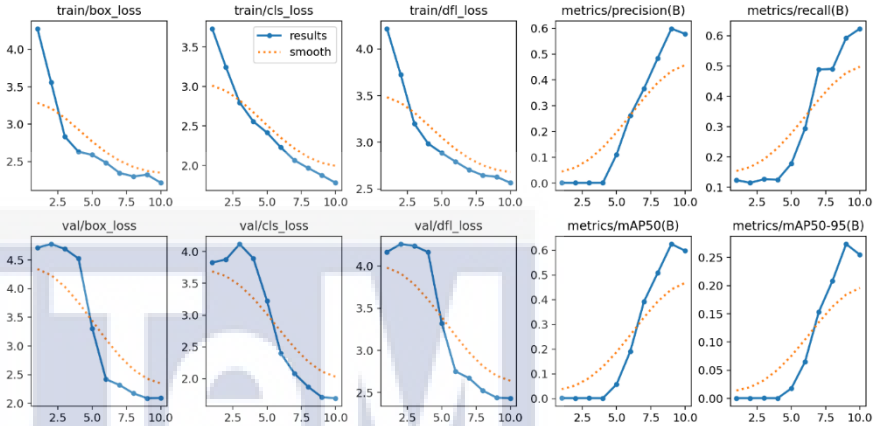


Figure 4.25 Training Loss, Validation Loss Precision and Recall for 10 Epoch with Yolov9.

Table 4.13 Last 5 Batch of 10 Epoch with Yolov8

Epoch	Train/box_loss	Train/cls_loss	Train/dfl_loss	Val/box_loss	Val/cls_loss	Val/dfl_loss
6	2.5976	2.2301	2.7923	2.4519	2.7623	2.7518
7	2.3464	2.066	2.7047	2.3168	2.0813	2.6704
8	2.3017	1.9685	2.6455	2.1732	1.8728	2.5231
9	2.3239	1.8769	2.6298	2.0879	1.7103	2.4401
10	2.2152	1.7819	2.5653	2.091	1.6933	2.4334

Table 4.14 Last 5 Batch of 10 epoch with Yolov9

Epoch	Train/box_loss	Train/cls_loss	Train/dfl_loss	Val/box_loss	Val/cls_loss	Val/dfl_loss
6	2.4849	2.1221	2.7408	2.4571	2.4008	2.7518
7	2.4769	2.0171	2.711	2.3518	2.1933	2.6184
8	2.3579	1.8945	2.6109	2.1301	2.0049	2.5533
9	2.2639	1.7777	2.5249	2.2239	1. 3192	2.4166
10	2.1765	1.7306	2.4483	2.062	1. 2131	2.4109

For Yolov8, there is a noticeable decrease in the training losses across the epochs. The train/box_loss decreases from 2.5976 at epoch 6 to 2.2152 at epoch 10, train/cls_loss drops from 2.2301 to 1.7819, and train/dfl_loss reduces from 2.7923 to 2.5653. Similarly, validation losses also show a declining trend. The val/box_loss decreases from 2.4519 at epoch 6 to 2.091 at epoch 10, val/cls_loss decreases from 2.7623 to 1.6933, and val/dfl_loss reduces from 2.7518 to 2.4334.

For Yolov9, a similar trend of decreasing losses is observed. The train/box_loss decreases from 2.4849 at epoch 6 to 2.1765 at epoch 10, train/cls_loss drops from 2.1221 to 1.7306, and train/dfl_loss reduces from 2.7408 to 2.4483. The validation losses also decline over the epochs, with val/box_loss decreasing from 2.4571 at epoch 6 to 2.062 at epoch 10, val/cls_loss decreasing significantly from 2.4008 to 1.2131, and val/dfl_loss reducing from 2.7518 to 2.4109.

Overall, both YOLO versions show an improvement in performance with a consistent reduction in both training and validation losses over the last five epochs. This indicates that the models are learning effectively and becoming more accurate in their predictions.

4.6 Testing Result

The model is tasked with identifying and localizing objects within 3 videos. When running a video testing, the model successfully recognised and localized all presented radiators within the frames as shown in Figure 4.26, 4.27 and 4.28. This means that while the model accurately detected and classified the distinct objects. This scenario highlights the model's capability to identify objects in real-time video streams.

Figure 4.26 Result of Video Testing 1.

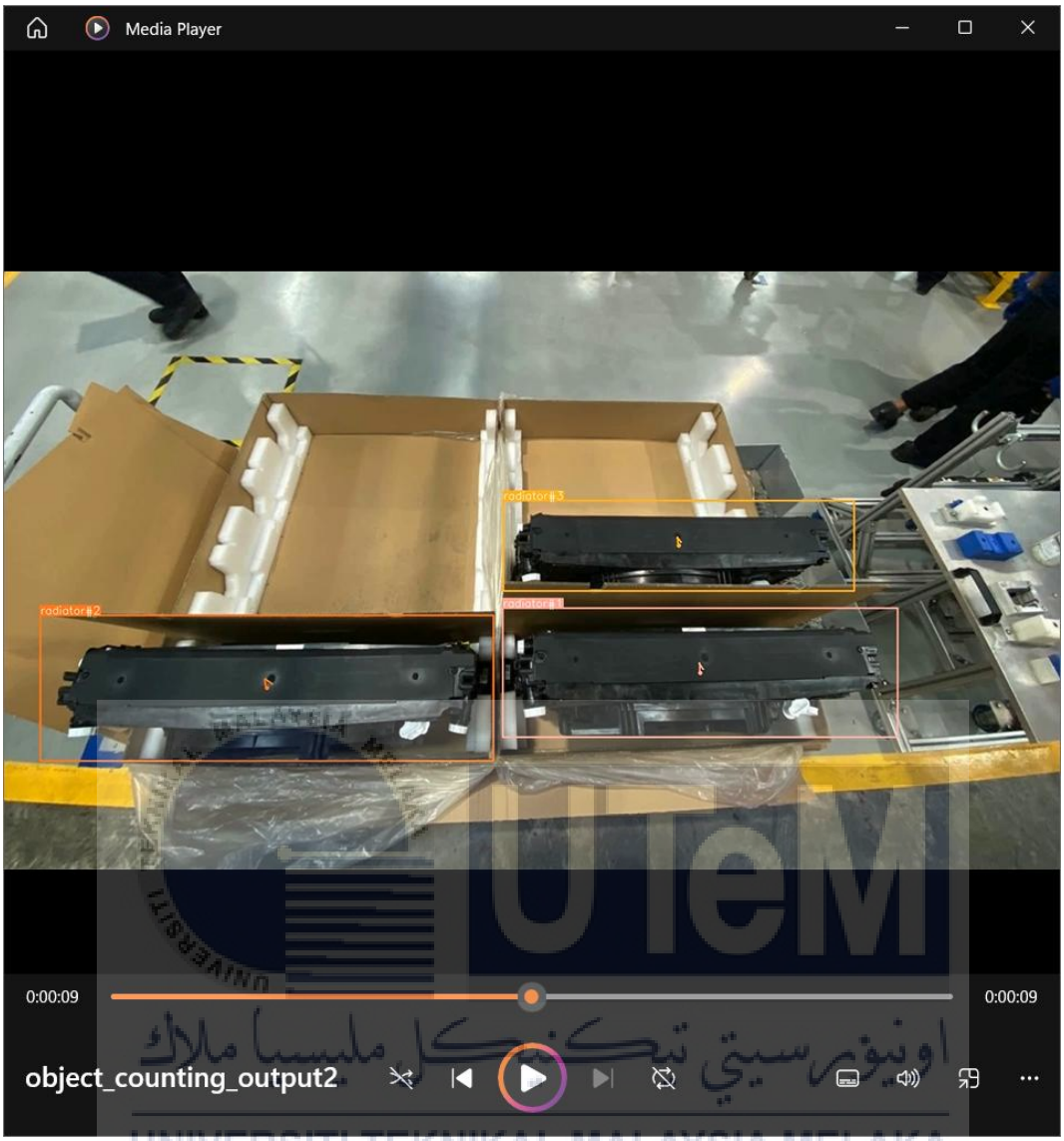


Figure 4.27 Result of Video Testing 2**.**

Figure 4.28 Result of Video Testing 3**.**

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The development of a computer vision system utilizing YOLOv8 for the detection and classification of logistic parts supply, specifically car radiators, is undertaken in this project. The primary objective is to automate the identification and categorization of car radiators within a logistic environment, facilitating efficient supply chain management. Firstly, various images of car radiators in logistic settings were collected. Then YOLOv8 is successfully developed on this dataset, optimizing its ability to accurately detect and classify radiators within varying backgrounds and orientations during daytime or good lighting.

Starting with YOLOv8, we observe a progressive improvement in performance metrics with increasing epochs. At 10 epochs, the model achieves moderate precision (0.59862) and recall (0.59321), with an F1 score of 0.59. However, as training progresses to 25, 50, 75, and 100 epochs, there is a notable enhancement in precision, recall, and F1 score. Particularly, at 100 epochs, YOLOv8 achieves outstanding precision (0.97493), recall (0.97277), and an impressive F1 score of 0.97, indicating its capability to accurately detect radiators with minimal errors.

On the other hand, YOLOv9 exhibits a different trend in performance. At 10 epochs, the model shows relatively higher precision (0.67825) and recall (0.6316) compared to YOLOv8 at the same epoch. However, the F1 score is lower at 0.67. This suggests that YOLOv9 initially performs reasonably well but may require further training or fine-tuning to achieve comparable results to YOLOv8's performance at higher epochs.

In comparing the training times between YOLOv8 and YOLOv9, a notable difference emerges. YOLOv9, despite achieving relatively higher initial precision and recall at 10 epochs, requires significantly longer training times compared to YOLOv8 across all epochs. YOLOv8 requires only 1.664 hours for training, whereas YOLOv9 takes substantially longer at 6.651 hours. This indicates that YOLOv8 offers much faster training compared to YOLOv9 at this stage of training. The shorter training time of YOLOv8 can be advantageous in scenarios where

rapid prototyping or quick model iterations are required. It allows for faster experimentation with different hyperparameters, data augmentation techniques, and training strategies, leading to quicker model refinement and optimization. On the other hand, although YOLOv9 has a longer training time, it initially demonstrates slightly higher precision and recall compared to YOLOv8 at 10 epochs. This suggests that YOLOv9 may require more time to converge to optimal performance but could potentially offer better performance with extended training.

In summary, YOLOv8 demonstrates superior performance in radiator detection as training progresses, achieving exceptional precision, recall, and F1 score at 100 epochs. YOLOv9 shows potential with decent performance at 10 epochs, but its performance may benefit from additional training and optimization. Overall, YOLOv8 showing robustness and accuracy in complex detection tasks with extended training.

While testing with the video feed, the model successfully identified four out of six objects in video streams. Despite the partial identification, the success in recognizing and localizing the objects signifies the YOLOv8 model's potential to contribute significantly to the automation of logistic part supply chains, offering real-time insights and facilitating streamlined operations in the context of car radiator logistics. However, it's essential to further the training of the YOLOv8 model with larger dataset.

5.2 Future Works

For future work, enhancing the performance of the logistic parts supply detection system can be pursued through several strategies. First, visit the site during the holiday period to collect more datasets for better model training. Besides, data augmentation techniques also can be employed to further diversify the training dataset and train the YOLOv8 model on a more extensive and diverse dataset. Plus, utilize transfer learning by fine-tuning a pre-trained YOLOv8 model on a large dataset (e.g., COCO dataset) before fine-tuning it specifically for radiator detection. Transfer learning can significantly speed up convergence and improve performance, especially with limited training data. Last but not least, monitor the model's performance regularly and fine-tune it as needed with new data or adjustments to the training pipeline. This ensures the model remains effective and adapts to evolving detection requirements.

REFERENCES

[1] Y. H. Wang and H. L. Zhang, "Research on machine vision technology based detection and tracking of objects on video image," *2022 Int. Conf. Image Process. Comput. Vis. Mach. Learn. ICICML 2022*, no. Icicml, pp. 267–270, 2022, doi: 10.1109/ICICML57342.2022.10009811.

[2] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. pp. 246–252, 1999. doi: 10.1109/cvpr.1999.784637.

[3] D. Koller *et al.*, "Towards robust automatic traffic scene analysis in real-time," *Proceedings of the IEEE Conference on Decision and Control*, vol. 4. pp. 3776–3781, 1994. doi: 10.1109/icpr.1994.576243.

[4] S. V. Mahadevkar *et al.*, "A Review on Machine Learning Styles in Computer Vision - Techniques and Future Directions," *IEEE Access*, vol. 10. Institute of Electrical and Electronics Engineers Inc., pp. 107293–107329, 2022. doi: 10.1109/ACCESS.2022.3209825.

[5] J. Yang, Y. Liu, M. Qian, C. Guan, and X. Yuan, "Information extraction from electronic medical records using multitask recurrent neural network with contextual word embedding," *Appl. Sci.*, vol. 9, no. 18, 2019, doi: 10.3390/app9183658.

[6] B. Zhang, C. Quan, and F. Ren, "Study on CNN in the recognition of emotion in audio and images," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, IEEE, Jun. 2016, pp. 1–5. doi: 10.1109/ICIS.2016.7550778.

[7] D. A. Pollen, "Explicit neural representations, recursive neural networks and conscious visual perception," *Cereb. Cortex*, vol. 13, no. 8, pp. 807–814, 2003, doi: 10.1093/cercor/13.8.807.

[8] K. Sakatani, "Using artificial neural networks to understand the human brain," *Res. Featur.*, no. 144, Nov. 2022, doi: 10.26904/RF-144-3511648225.

[9] M. Abdul, H. Ashour, and N. I. Jabbouri, "Improvement of Neural Networks Artificial Output," *Int. J. Sci. Res.*, vol. 6, no. 12, pp. 352–361, 2017, doi: 10.21275/art20178512.

[10] S. Dodia, B. Annappa, and P. A. Mahesh, "Recent advancements in deep learning based lung cancer detection: A systematic review," *Eng. Appl. Artif. Intell.*, vol. 116, no. August, p. 105490, 2022, doi: 10.1016/j.engappai.2022.105490.

[11] M. O. Ojo and A. Zahid, "Deep Learning in Controlled Environment Agriculture: A Review of Recent Advancements, Challenges and Prospects," *Sensors (Basel).*, vol. 22, no. 20, 2022, doi: 10.3390/s22207965.

[12] R. A. Jarvis, "A Perspective on Range Finding Techniques for Computer Vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-5, no. 2, pp. 122–139, 1983, doi: 10.1109/TPAMI.1983.4767365.

[13] R. Yang and Y. Yu, "Artificial Convolutional Neural Network in Object Detection and Semantic Segmentation for Medical Imaging Analysis," *Front. Oncol.*, vol. 11, no. March, pp. 1–9, 2021, doi: 10.3389/fonc.2021.638182.

[14] J. Haupt and R. Nowak, "Compressive Sampling Vs. Conventional Imaging," in *2006 International Conference on Image Processing*, IEEE, Oct. 2006, pp. 1269–1272. doi: 10.1109/ICIP.2006.312576.

[15] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, 2018, doi: 10.1016/j.patcog.2017.10.013.

[16] H. Perez, J. H. M. Tah, and A. Mosavi, "Deep Learning for Detecting Building Defects Using," *Sensors*, vol. 19, no. 16, p. 3556, 2019.

[17] M. Hussain, H. Al-Aqrabi, and R. Hill, "Statistical Analysis and Development of an Ensemble-Based Machine Learning Model for Photovoltaic Fault Detection," *Energies*, vol. 15, no. 15, 2022, doi: 10.3390/en15155492.

[18] A. Kusiak, "Smart manufacturing," *Int. J. Prod. Res.*, vol. 56, no. 1–2, pp. 508–517, Jan. 2018, doi: 10.1080/00207543.2017.1351644.

[19] Z. J. Wang *et al.*, "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 2, pp. 1396–1406, 2021, doi: 10.1109/TVCG.2020.3030418.

[20] 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.

[21] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556

[22] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[24] P. Soviany and R. T. Ionescu, "Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction," *Proc. - 2018 20th Int. Symp. Symb. Numer. Algorithms Sci. Comput. SYNASC 2018*, pp. 209–214, 2018, doi: 10.1109/SYNASC.2018.00041.

[25] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," *J. Phys. Conf. Ser.*, vol. 1544, no. 1, 2020, doi: 10.1088/1742-6596/1544/1/012033.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, 2016, doi: 10.1109/TPAMI.2015.2437384.

[27] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[28] 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, 2017, doi: 10.1109/TPAMI.2016.2577031.

[29] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional Single Shot Detector," Jan. 2017, [Online]. Available: http://arxiv.org/abs/1701.06659

[30] X. Cheng and J. Yu, "RetinaNet with Difference Channel Attention and Adaptively Spatial Feature Fusion for Steel Surface Defect Detection," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, doi: 10.1109/TIM.2020.3040485.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[32] A. Vidyavani, K. Dheeraj, M. Rama Mohan Reddy, and K. N. Kumar, "Object detection method based on YOLOv3 using deep learning networks," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 1, pp. 1414–1417, 2019, doi: 10.35940/ijitee.A4121.119119.

[33] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014, doi: 10.1007/978-3-319-10602-1_48.

[34] S. Shetty, "Application of Convolutional Neural Network for Image Classification on Pascal VOC Challenge 2012 dataset," Jul. 2016, [Online]. Available:

http://arxiv.org/abs/1607.03785

[35] C. Li *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," 2022, [Online]. Available: http://arxiv.org/abs/2209.02976

[36] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "RepVgg: Making VGG-style ConvNets Great Again," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 13728–13737, 2021, doi: 10.1109/CVPR46437.2021.01352.

[37] X. Li *et al.*, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, pp. 1–14, 2020.

[38] Z. Gevorgyan, "SIoU Loss: More Powerful Learning for Bounding Box Regression," pp. 1–12, 2022, [Online]. Available: http://arxiv.org/abs/2205.12740

[39] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf, "VarifocalNet: An IoU-aware Dense Object Detector," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8510–8519, 2021, doi: 10.1109/CVPR46437.2021.00841.

[40] C. Shu, Y. Liu, J. Gao, Z. Yan, and C. Shen, "Channel-wise Knowledge Distillation for Dense Prediction," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 5291–5300, 2021, doi: 10.1109/ICCV48922.2021.00526.

[41] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 7464–7475. doi: 10.1109/CVPR52729.2023.00721.

[42] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," pp. 1–7, Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.08430

[43] C. Y. Wang, I. H. Yeh, and H. Y. M. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," *J. Inf. Sci. Eng.*, vol. 39, no. 3, pp. 691–709, 2023, doi: 10.6688/JISE.202305_39(3).0015.

[44] W. Wu *et al.*, "DSANet: Dynamic Segment Aggregation Network for Video-Level Representation Learning," *MM 2021 - Proc. 29th ACM Int. Conf. Multimed.*, pp. 1903–1911, 2021, doi: 10.1145/3474085.3475344.

[45] C. Li *et al.*, "BossNAS: Exploring Hybrid CNN-transformers with Block-wisely Self-supervised Neural Architecture Search," *Proc. IEEE Int. Conf. Comput. Vis.*, no. Iccv, pp. 12261–12271, 2021, doi: 10.1109/ICCV48922.2021.01206.

[46] P. Dollár, M. Singh, and R. Girshick, "Fast and Accurate Model Scaling," *Proc. IEEE*

Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 924–932, 2021, doi: 10.1109/CVPR46437.2021.00098.

[47] S. Guo, J. M. Alvarez, and M. Salzmann, "ExpandNets: Linear over-parameterization to train compact convolutional networks," *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, no. NeurIPS, 2020.

[48] X. Ding, X. Zhang, J. Han, and G. Ding, "Scaling Up Your Kernels to 31×31: Revisiting Large Kernel Design in CNNs," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2022-June, pp. 11953–11965, 2022, doi: 10.1109/CVPR52688.2022.01166.

[49] Z. Mahboob, A. Zeb, and U. S. Khan, "YOLO v5, v7 and v8: A Performance Comparison for Tobacco Detection in Field," *2023 Int. Conf. Digit. Futur. Transform. Technol. ICoDT2 2023*, pp. 1–6, 2023, doi: 10.1109/ICoDT259378.2023.10325705.

[50] M. Lin, Q. Chen, and S. Yan, "Network In Network," *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, pp. 1–10, Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.4400

[51] Z. Ouyang, J. Niu, Y. Liu, and M. Guizani, "Deep CNN-Based real-time traffic light detector for self-driving vehicles," *IEEE Trans. Mob. Comput.*, vol. 19, no. 2, pp. 300–313, Feb. 2020, doi: 10.1109/TMC.2019.2892451.

[52] J. Kim, J.-Y. Sung, and S. Park, "Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition," in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, IEEE, Nov. 2020, pp. 1–4. doi: 10.1109/ICCE-Asia49877.2020.9277040.

[53] D. Wu *et al.*, "Detection of Camellia oleifera Fruit in Complex Scenes by Using YOLOv7 and Data Augmentation," *Appl. Sci.*, vol. 12, no. 22, 2022, doi: 10.3390/app122211318.

APPENDICES

APPENDIX A

The installation of required dependencies and the cloning of YOLOv8 is conducted by using the code;

```
# To clone the yolo algorithm from ultralytics
from ultralytics import YOLO

# To Load a YOLOv8 model
model = YOLO("yolov8n.yaml")  # To build a new model from scratch
```

Then create a “yaml” file name config.yaml.;

```
# To specify the data file path
path: C:\Users\User\Documents\data
train: images/train
val: images/train

#To specify classes
names:
  0: radiator
```

After all the steps above are done, the training process of the object detection model will take place by using the code;

```
# Use the YOLOv8 model for training
# Epoch for number of cycle trains
results = model.train(data="config.yaml", epochs=100)
# Load the path of the data from yaml file
```

For video inference, the code;

```
import os
import time

from ultralytics import YOLO
import cv2

model_path  = os.path.join('.', 'runs', 'detect', 'train3', 'weights',
'best.pt')
model = YOLO(model_path)

VIDEOS_DIR = os.path.join('.', 'videos')
video_path = os.path.join(VIDEOS_DIR, 'radiator2.mov')
cap = cv2.VideoCapture(video_path)
```

```
ret = True

while ret:
    ret, frame = cap.read()

    results = model.track(frame, persist=True)
    frame_ = results[0].plot()
    cv2.imshow('frame', frame_)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
```