

**DETECTION OF SURFACE DEFECTS OF
ALUMINIUM EXTRUDANTS USING
ARTIFICIAL INTELLIGENCE**

POAN CHEE KENT

UNIVERSITI TUNKU ABDUL RAHMAN

**DETECTION OF SURFACE DEFECTS OF ALUMINIUM
EXTRUDANTS USING ARTIFICIAL INTELLIGENCE**

POAN CHEE KENT

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Science (Honours) Software
Engineering**

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

September 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :



Name : Poan Chee Kent

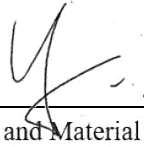
ID No. : 2005739

Date : 13/9/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**Detection Of Surface Defects Of Aluminium Extrudants Using Artificial Intelligence**” was prepared by **POAN CHEE KENT** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : Ts. Dr. Yeo Wei Hong  4/10/2024
Associate Professor/ Head
Department of Mechanical and Material Engineering
Supervisor : Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

Date : _____

Signature :  _____

Co-Supervisor : Khor Kok Chin _____

Date : 4/10/2024 _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2024, POAN CHEE KENT. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Yeo Wei Hong and Dr. Khor Kok Chin for their invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement to complete my final year project.

ABSTRACT

As the demand for aluminium profiles continues to rise, the occurrences of defects on the aluminium surface increase rapidly. While some factories still rely on manual defect detection, the small and unobvious defects often lead to high false detection rates due to the human eye's limitation. Although nowadays some manufacturing industries have implemented algorithms to automate the detection of defects, those algorithms face challenges on dealing with noises and lighting changes. This study aims to replace manual and inefficient automated defect detection with an approach that uses object detection. The objectives of this study include implement YOLOv8 model to identify and categorise the aluminium surface defect with the aid of data augmentation, transfer learning and addition of attention modules. The YOLOv8n model is trained to identify and localise the defect on the aluminium surface with the help of transfer learning. To solve the problem of limited datasets, data augmentation is used to expand the dataset to prevent overfitting. This study also compares the performance between YOLOv8n with and without attention modules. Attention modules included in this study are ECA and ResCBAM. However, the implementation of attention modules does not increase the performance of the model. The final model achieved a mAP@0.5 of 94.3% and 79.7% of mAP@0.5:0.95 compared to the original YOLOv8n model. This study shows the effectiveness and efficiency of YOLOv8n in detecting defects on aluminium surfaces. Besides, this study also proves the effectiveness of transfer learning and data augmentation in improving the overall performance of YOLOv8n in detecting various kinds of defect of aluminium surface.

TABLE OF CONTENTS

DECLARATION	i
APPROVAL FOR SUBMISSION	ii
ACKNOWLEDGEMENTS	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	ixx
LIST OF FIGURES	xi
LIST OF SYMBOLS / ABBREVIATIONS	xivv
LIST OF APPENDICES	xv
CHAPTER 1	1
1 INTRODUCTION	1
1.1 General Introduction	1
1.2 Importance of the Study	3
1.3 Problem Statement	4
1.4 Aim and Objectives	5
1.5 Proposed Solution	5
1.6 Scope and Limitations of the Study	6
CHAPTER 2	8
LITERATURE REVIEW	8
2.1 Previous work on aluminium surface defect detection	8
2.1.1 Unified Method for detecting common and rare aluminium defects	8
2.1.2 Usage of Transfer Learning and Data Augmentation on Defect Detection	13
2.1.3 Defect detection of aluminium profile surface using MS-YOLOv5 based on YOLOv5	19

2.1.4	Lightweight model network based on YOLOv5s	29
2.2	Other usages of YOLO on Metal Surface Defect Detection	34
2.3	YOLOv8	36
2.4	Conclusion	44
CHAPTER 3		45
3	METHODOLOGY AND WORK PLAN	45
3.1	Workflow of model training	45
3.1.1	Data Collection	46
3.1.2	Data Selection	47
3.1.3	Data Annotation	48
3.1.4	Data preprocessing	50
3.1.5	Data augmentation	51
3.1.6	Model training, validation, and testing	53
3.1.7	Performance Evaluation	54
3.1.8	Model Improvement	54
3.1.8.1	Dataset Expansion	54
3.1.8.2	Efficient Channel Attention	55
3.1.8.3	ResBlock + Convolutional Block Attention Module	56
3.1.8.4	Load a pre-trained model	58
3.2	Evaluation Metrics	58
3.2.1	IoU	59
3.2.2	Precision and recall	60
3.2.3	mAP	62
3.3	Gantt Chart	63
CHAPTER 4		65
4	RESULTS AND DISCUSSION	65
4.1	Comparison among YOLOv8 Predecessors and YOLOv8	65
4.2	YOLOv8 Variants Selection	65
4.3	Comparison Between Before and After Data Selection & Data Augmentation	66
4.4	Impact on loading pre-trained model	68

	viii
4.5 Impact of Scratch Class Expansion	71
4.6 Model Improvement with Attention Module	73
4.7 Discussions	74
CHAPTER 5	80
5 CONCLUSION AND FUTURE WORKS	80
5.1 Conclusion	80
5.2 Future Enhancements	81
REFERENCES	83
APPENDICES	90

LIST OF TABLES

Table 2.2: Performance of different detection	13
Table 2.3: Training results when trained with transfer learning and data augmentation	17
Table 2.4: Training results when only trained with data augmentation and when only trained with transfer learning	17
Table 2.5: Comparison of each detection based on precision, recall, mAP, and FPS	26
Table 2.6: AP performance for each algorithm	28
Table 2.7: Result of the four types of defect detection	32
Table 2.8: Outcome of each algorithm	33
Table 2.9: Accuracy of each algorithm	34
Table 2.10: Summary of the reviewed YOLO variants	37
Table 3.1: Reason and expected result of data augmentation methods	51
Table 3.2: Hyperparameter setting of model training	54
Table 3.3: Fundamental concepts in object detection	58
Table 4.1: Comparison between YOLOv8 Predecessors	65
Table 4.2: Comparison between Yolov8 variants	66
Table 4.3: Performance of YOLOv8 in each class with the original dataset	67
Table 4.4: Performance of YOLOv8n in each class after data selection and augmentation	68
Table 4.5: Comparison of performance before and after data augmentation	68
Table 4.6: Performance on each class using transfer learning	69
Table 4.7: Comparison between with and without transfer learning on the augmented dataset	69
Table 4.8: Performance on each class with scratch class expansion	72

Table 4.9: Comparison between before and after scratch class expansion on the augmented dataset	72
Table 4.10: Performance of YOLOv8n with ECA	73
Table 4.11: Performance of YOLOv8 with ResCBAM	74
Table 4.12: Comparison between two attention modules and YOLOv8n (with data augmentation + scratch class expansion)	74
Table 4.13: Effect of each step-by-step improvement on YOLOv8n	75

LIST OF FIGURES

Figure 2.1: Example of aluminium profile surface defect (Zhang et al., 2020)	8
Figure 2.2: Structure of the proposed unified method (Zhang et al., 2020)	9
Figure 2.3: Structure of the rare category transfer network (Zhang et al., 2020)	11
Figure 2.4: Number of samples per category (Zhang et al., 2020)	12
Figure 2.5: Accuracy of UCR method on different defects (Zhang et al., 2020)	13
Figure 2.6: Image with data augmentation (Neuhauser et al., 2019)	14
Figure 2.7: An inception module with dimensionality reduction (Neuhauser et al., 2019)	16
Figure 2.8: mAP@0.5 with TL and without TL over 800000 iterations (Neuhauser et al., 2019)	18
Figure 2.9: Example of defect detection using ResNet50 with DA and TL (Neuhauser et al., 2019)	18
Figure 2.10: Number of datasets for training and testing (Wang et al., 2022)	20
Figure 2.11: RGB comparison histogram. (a) Original (b) Adjusted Gamma (c) Adjusted Contrast (d) Adjusted Brightness (Wang et al., 2022)	21
Figure 2.12: Structure of MS-YOLOv5. (a) Backbone, PE-Neck, and multi-streamnet (b) composition of modules (Wang et al., 2022)	21
Figure 2.13: Architecture of PSConv. (a) input features (b) kernel incorporates various dilation rates (Wang et al., 2022)	22
Figure 2.14: Schematic representation of the ECA module's architecture (Wang et al., 2022)	23
Figure 2.15: Structure of MS-YOLOv5's neck (Wang et al., 2022)	24
Figure 2.16: Structure of multi-streamnet (Wang et al., 2022)	25
Figure 2.17: Comparison of mAP of each detection (Wang et al., 2022)	25

Figure 2.18: F1-cure comparison for each defect (Wang <i>et al.</i> , 2022)	26
Figure 2.19: PR-cure comparison for each defect (Wang <i>et al.</i> , 2022)	27
Figure 2.20: Comparison of YOLOv5 and MS-YOLOv5 in detecting and localizing defects (Wang <i>et al.</i> , 2022)	27
Figure 2.21: More results of MS-YOLOv5 in detecting defects (Wang <i>et al.</i> , 2022)	27
Figure 2.22: A comparison chart of mAP for each algorithm (Wang <i>et al.</i> , 2022)	28
Figure 2.23: The proposed network architecture (Tang <i>et al.</i> , 2023)	29
Figure 2.24: Ghost module structure (Tang <i>et al.</i> , 2023)	30
Figure 2.25: AC3Ghost module Structure (Tang <i>et al.</i> , 2023)	31
Figure 2.26: Object detection results for four defects (Tang <i>et al.</i> , 2023)	33
Figure 2.27: Parallel Network Structure proposed by Ma <i>et al.</i> (2022)	35
Figure 2.28: YOLO-v8 comparison with predecessors (Hussain, 2023)	36
Figure 2.29: YOLOs mAP@0.5 against RF100 (Solawetz, 2024)	38
Figure 2.30: YOLOs average mAP@0.5 against RF100 categories (Solawetz, 2024)	38
Figure 3.1: Workflow for training the YOLOv8 object detection model.	45
Figure 3.2: The labeled dataset from the journal	46
Figure 3.3: Number of images for each class	47
Figure 3.4: Number of images after data selection	48
Figure 3.5: Images uploaded to Roboflow	49
Figure 3.6: Annotating defect on image	49
Figure 3.7: Dataset splitting	50
Figure 3.8: Comparison after data preprocessing	50
Figure 3.9: Augmented data with flip, rotate, brightness and noise	52
Figure 3.10: Download dataset in YOLOv8 format	52

Figure 3.12: Images added to expand the dataset	55
Figure 3.13: Structure of ECA	56
Figure 3.14: Structure of ResCBAM	57
Figure 3.15: Illustration of Intersection over Union (IoU)	59
Figure 3.16: Example of IOU	60
Figure 3.17: Precision-recall curve using the 11-point interpolation approach	61
Figure 3.18: Precision-recall curve using the all-point interpolation approach	62
Figure 3.19: Gantt chart for Preliminary Planning from 1/2/2024 to 27/2/2024	63
Figure 3.20: Gantt chart for Project Methodology Planning from 28/2/2024 to 12/4/2024	63
Figure 3.21: Gantt chart for Model Developing from 17/6/2024 to 11/8/2024	63
Figure 3.22: Gantt chart for Model Developing from 12/8/2024 to 11/9/2024	64
Figure 4.1: Comparison of PR curves for YOLOv8n without transfer learning (left) and YOLOv8n with transfer learning (right)	70
Figure 4.2: Comparison of F1 score curves for YOLOv8n without transfer learning (left) and YOLOv8n with transfer learning (right)	71
Figure 4.3: Evaluation metric of baseline model with original dataset	76
Figure 4.4: Evaluation metrics of the improved model with improved dataset	76
Figure 4.5: Normalized confusion matrix of YOLOv8n with all the improvements	77
Figure 4.6: False negative of scratch	78
Figure 4.7: False negative on shallow scratch	78
Figure 4.8: Two false negatives of scratch	79

LIST OF SYMBOLS / ABBREVIATIONS

AI	artificial intelligence
AOI	automated optical inspection
AP	average precision
CBAM	convolutional block attention module
CMM	common category feature maps
CNN	convolutional neural network
DA	data augmentation
DCNN	deep-convolutional neural network
DL	deep learning
DFS	deep feature selection
DWConv	depthwise separable convolution
ECA	Efficient Channel Attention
Fast R-CNN	Fast Region-based CNN
Faster R-CNN	Faster Region-based CNN
FPS	frame per second
GPU	graphic processing unit
IOU	intersection over union
ML	machine learning
NLP	natural language processing
PSConv	Poly-Scale Convolution
PM	proposal feature mRCM rare category feature maps
ReLU	rectified linear unit
ROI	region of interest
RPN	region proposal network
R-CNN	region-CNN
SPP net	Spatial Pyramid Pooling Network
SVM	support vector machine
SDG	Sustainable Development Goal
TL	transfer learning
WBS	work breakdown structure
YOLO	You Only Look Once

LIST OF APPENDICES

Appendix A: Model Configuration File for YOLOv8n	90
Appendix B: Model Configuration File for YOLOv8n with ECA	91
Appendix C: Model Configuration File for YOLOv8n with ResCBAM	92
Appendix D: Code snippet for ECA attention module in init.py	93
Appendix E: Code snippet for ResCBAM attention module in init.py	93

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Aluminium profiles are known as an essential material for many industrial applications. due to their versatility, strength, and lightweight properties. For decades, aluminium profiles have played an important role in applications ranging from high-speed railways to skyscrapers industries, causing the yield and quality of the aluminium industry to become progressively important (Neuhauser, Bachmann, and Hora, 2019). Unfortunately, the high demand for aluminium profiles causes the need for rapid production and the different complexity of profiles has led to a greater challenge for manufacturers to maintain the balance between quality and quantity supply for the demand. As a result of fulfilling the high demand for aluminium profiles in the industries, the occurrences of defects on the surface of aluminium profiles increase rapidly causing the disposal of the whole aluminium components and further leading to financial losses for the manufacturer. Furthermore, external factors such as ununiform standards of the production process, and different production equipment could also affect the service life of aluminium profiles (Wang et al., 2022). Therefore, it is a must for quality defect inspection to be performed on the surface of aluminium to detect and carry out appropriate actions on the aluminium profiles. Previously, the manual-based detection method which is by human eyes was carried out for defect detection on metal profile surfaces (Campbell, 2013). This was highly reliable due to its high accuracy in identifying defects. However, the manual-based detection method has been slowly eliminated by industries due to its slow detection speed, cost-intensive, and visual acuity limitations (Lin and Wibowo, 2021). Manual-based detection can also be affected by external factors such as labour tiredness and subjectivity, making it inconsistent and unreliable for high-volume production environments.

Fortunately, significant advancements in machine vision, image processing, and artificial intelligence (AI) have massively enhanced the capabilities of vision inspection technology, in turn, has propelled the progress of vision-based automated optical inspection (AOI) systems (Ganovska et al.,

2016) (Jia et al., 2018). Lin and Wibowo (2021) stated that the majority of the AOI systems depend on image processing algorithms prior to the advent of deep learning. In recent years, deep learning has been implemented into the AOI systems with the training of deep learning models with massive amounts of databases. However, effective AOI systems couldn't be carried out if the image quality of the detection target is too poor even with a resilient deep learning detection system. Detecting defects in metal surfaces has been a major challenge due to their small size and diverse types of defects such as scratches, die lines, dents, blisters, and others arising from production processes (Lin and Wibowo, 2021). Furthermore, even the same defects would have different features in terms of size, depth, and direction, causing the detection of metal surfaces to be highly challenging.

There are two approaches for defect detection algorithms using deep learning which are detection in one-stage and two-stage detection algorithms (Zhao and Zhu, 2023). Examples of two-stage detection algorithms are Region-CNN (R-CNN) (Girshick et al., 2013), Fast Region-based CNN (Fast R-CNN) (Girshick, 2015) and Faster Region-based CNN (Faster R-CNN) (Ren et al., 2017). Meanwhile, single-stage detection algorithms include the You Only Look Once (YOLO) (Redmon et al., 2015) (Redmon and Farhadi, 2016) (Redmon and Farhadi, 2018) series and Single-Shot MultiBox Detector (SSD) (Liu et al., 2016). One-stage object detection utilizes a convolutional neural network (CNN) to simultaneously predict object class and location directly from the image in a single pass (Zhao and Zhu, 2023). One-stage detection prioritizes speed by densely sampling the image, but this approach can suffer from class imbalance. The abundance of negative samples relative to positive ones makes training difficult, potentially compromising model accuracy (Shf and Zhao, 2020).

This research studies the application of deep learning-based defect detection algorithms for aluminium surface defect detection and aims to overcome the limitations of manual-based methods, such as high detection costs, cumbersome manual steps, and the inability to simultaneously classify and localize defects. Deep learning offers a potential solution for efficient, automated defect detection with improved accuracy in both classification and localization (Wang et al., 2022).

1.2 Importance of the Study

Aluminium can be found in many sectors, for which the expense of aluminium profile quality control is essential because it might otherwise weaken its structural integrity. This study will focus on the Sustainable Development Goals (SDGs) established by the United Nations and show how AI can find chips, scratches, or dents in aluminium profiles.

The focus of this study is on sustainable industrial practices, which are in line with Sustainable Development Goal (SDG) 9: Sustainable industry, smart productivity, and the construction of sustainable infrastructure. The AI, which can replace the process of manual defect checking with automated inspection, can lower production costs and improve production efficiency. As production scheduling is subject to manual-based detection, the latter might cause delays and create bottlenecks during the course of AI-operated scheduled production as compared to AI-based production scheduling. Through recurrent photos taken of deficiencies and labelled 'defects', this network of AI models may speed up the time of on-the-spot evaluation greatly. It decreases the number of production-related defects and additionally allows for the reallocation of human resources from routine control of product defects to work with more unsuccessful cases and high-level decision-making instead. Implications of this goal encompass innovation, industrialization, and development of business, which are deemed sustainable or productive unlike destructive and polluting. SDG 9's objectives are to foster innovation, promote sustainable and beneficial industrialization, and build resilient infrastructure (Population Matters, 2023). The reflection on how important technological progress is in solving the coating of metals has a role in SDG 9.

Additionally, this study advances Sustainable Development Goal (SDG) 12: Ethical Production and Consumption. As aluminium manufacturing involves huge power usage, any effort aimed at reducing environmental waste can lead to a sharp fall in the overall level of resource consumption by industries. The industrial line with an AI defect detection system helps the engineers spot any metal surface defects. If the instance problem is preliminary they could also cut a tiny part of the aluminium. It might lead to less garbage accumulation and thus less of the environmental damage we currently face through producing

aluminium as well. They can inspect the fault, determine the root of it, and take measures of prevention for future detachment of the product..

1.3 Problem Statement

In recent studies on surface defects in aluminium surface, most of the focus has been on identifying the type of defects in images, such as scratches. However, there has been less attention given to finding the exact location of these defects within the images. While it is helpful to know what kind of defect is present, it is also important to know where the defect is located, especially in industrial production. In a factory setting, workers need to know the precise spot of the defect to fix it or remove the damaged part quickly and efficiently. This lack of emphasis on identifying the location of defects in previous research creates a gap that needs to be addressed. Without the ability to locate defects accurately, it becomes difficult for production teams to take corrective actions, leading to slower production and increased costs.

Generally, traditional image processing and deep learning techniques are two classifications for inspection algorithms in AOI. However, they do not deal with noise, changes in lighting conditions, or complicated patterns (Bhatt et al., 2021). Batool et al. (2021) state that there are restrictions with the common machine learning and image processing techniques, and massive amounts of noisy, poor-quality data are beyond their capabilities. Therefore, these methods' significant use of noise filtering, feature extraction, and selection procedures is part of the data preprocessing. They may result in information loss or distortion which lowers the pattern recognition accuracy.

In many existing studies, the primary focus has been on increasing detection accuracy in algorithms designed for identifying surface defects. While high accuracy is essential, this focus often leads to a significant drawback: a decrease in the detection speed of the algorithm. In industries that rely on real-time inspection, such as manufacturing and quality control, this slower detection process can create serious problems. Industrial inspection systems require both speed and precision to ensure production lines run efficiently without unnecessary delays. If the detection speed is too slow, even if the accuracy is very high, it may fail to meet the real-time demands of industrial environments. This trade-off between achieving higher accuracy and maintaining quick

detection is a critical issue. Slower detection processes can cause bottlenecks in production, delaying the identification and correction of defects, and ultimately impacting productivity.

1.4 Aim and Objectives

This study proposes a deep learning-based defect detection for the surface of aluminium extrudants leveraging image recognition techniques. The method focuses on identifying surface defects with minimal false positive and false negative detections.

Objectives:

- Implement a YOLOv8 detection model for aluminium surface defect detection by utilizing data augmentation and transfer learning
- Detect, localize, and categorize the types of surface defects using the implemented YOLOv8 model, ensuring that the algorithm achieves high accuracy while maintaining a fast detection speed suitable for real-time industrial applications.
- Evaluate the performance of the YOLOv8 object detection model using precision, recall and mean average precision.

1.5 Proposed Solution

This study proposed a solution to detect surface defects on aluminium extrusions using YOLOv8. Wang et al. (2022) proposed a new model, MS-YOLOv5 which is an advancement based on YOLOv5, the neck part of YOLOv5 is replaced with a PE-Neck structure, and the first detection head implements a multi-streamnet. The proposed method excels with the highest mean average precision (mAP) which is 87.4% in contrast to the popular object detection algorithm for detecting aluminium surface defects. The model has an average processing speed of 19.1 frames per second (FPS), which meets the needs of real-time inspections in industrial settings. Besides that, Xu, Zhang, and Wang (2021) also proposed a solution to detect defects with high accuracy

which is an improved YOLOv3 model. By using K-Means++, the modified model achieves 75.1% mAP and a reasoning speed of 83FPS. It achieves real-time inspection while ensuring the high accuracy of defect detection.

According to the two research above, modifying the YOLO models can greatly enhance the model's accuracy in detecting aluminium surface defects and align with the real-time inspection requirement in the industry. Another attention of this study is to find an approach to study new data augmentation techniques and resolve the issue of insufficient aluminium defect images. To solve the problem of algorithm sensitive to noise and light changes, data processing method such as grayscale is used to decrease the effect and impact of lighting and colour on the detecting of defects. Besides, data augmentation techniques such as noises and exposure are used to add noises such as dots on the training data to reduce the vulnerability of model towards noises, while exposure will train the model to be more adaptable towards different kind of lighting conditions. Attention modules will be implemented to YOLOv8 to allow the model to extract the features to be detected and improve its performance.

1.6 Scope and Limitations of the Study

The scope of this study is to implement a comprehensive deep learning defect detection approach using artificial intelligence to detect defects on the surface of aluminium extrudants provided by Aliyun Tianchi. By using a YOLOv8 detection model, the system is able to identify and localize the defect, which makes it possible to identify individual faults on the aluminium surface, such as scratches and other anomalies.

This research investigates the application of a YOLOv8 detection model within a deep learning framework to detect surface defects on aluminium extrusions. The focus is on extrusions that have completed the extrusion process, aiming to identify prevalent issues like scratches, non-conductive, orange peel, leakage and more. Scratch will be the focus of this study besides other defects as it is the most faced defect in PMB Aluminium Sdn Bhd. Besides, the Aliyun Tianchi dataset also has fewer defect images from scratch which requires more attention for further improvement. The main enhancements on the YOLOv8 model would includes data selection, data augmentation, transfer learning,

expansion of class specifically on scratch and implementation of attention modules.

While the current scope prioritizes these common surface imperfections, future iterations can be expanded to encompass more severe defects that impact the entire extrusion, such as tearing and waving patterns. Furthermore, image processing techniques to improve the image quality and resolution such as super-resolution and image reconstruction are not applied in this study. Hence, corrupted images and blurred images collected during dataset gathering would not be considered and contributed to the final findings. It is important to acknowledge that this research prioritizes exploring and reevaluating the core functionalities of the system, rather than focusing on its immediate industrial implementation.

The computing resources accessible to this project are its development limitations. The industry partner, PMB Aluminium Sdn Bhd, has contributed to the dataset, specifically for the scratch class as this is the most common defect found in their manufacturing factory. However, due to the insufficient sample provided, the amount of training data for scratch is not enough to train YOLOv8 to generalise well on scratch. Hence, the scratch class defect images will be also using the dataset from Aliyun Tianchi. Another reason for not using the aluminium from PMB Aluminium Sdn Bhd is because of the different textures on the aluminium surface which causes the model to not identifying well on the aluminium from PMB. To prevent information leakage, all deep learning models were trained on local machines using this dataset. The Intel i5-11400H CPU, RTX3050 laptop graphics card, and 24GB VRAM are installed on the local system. Python is the main development tool for this study whereby the PyTorch framework will be used to develop YOLO.

CHAPTER 2

LITERATURE REVIEW

2.1 Previous work on aluminium surface defect detection

The following section will review the previous work on aluminium surface defect detection.

2.1.1 Unified Method for detecting common and rare aluminium defects

Traditional image processing techniques have been an area of research recently for automatic surface defect detection on aluminium. However, these methods that rely on analysing variations in brightness are only suitable for flat aluminium plates and cannot handle the complex shapes of aluminium profiles. While DCNN methods are suitable for common defects on aluminium profiles, they struggle to identify both usual and unusual defects simultaneously. One of the factors is the variety and irregularity of defects because of physical and chemical. Another factor is the unbalanced data as common defects are easier to find and contribute to a larger training dataset while rare defects occur less frequently, hence the training dataset is lesser. Therefore, the unbalanced between both defects makes it harder for the learning algorithms to perform well. The figure below shows some common aluminium profile surface defects on the first row while the second row shows the rare defects that occur less frequently during production.

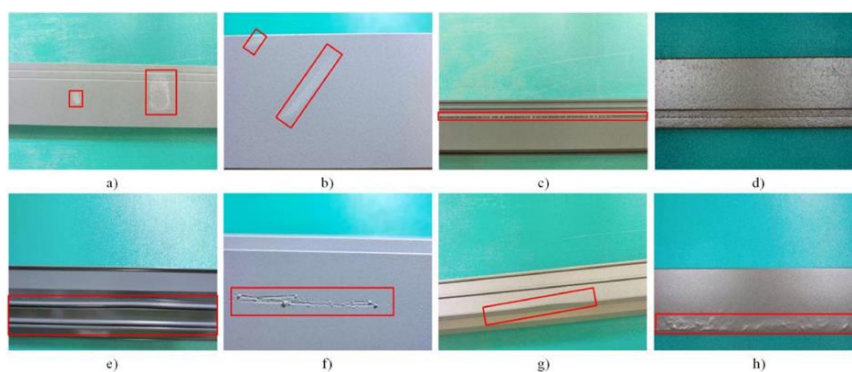


Figure 2.1: Example of aluminium profile surface defect (Zhang et al., 2020)

In this research, Zhang et al. (2020) proposed a unified method that can detect both usual and unusual defects at once. This method extracts usual defect characteristics and generates common category feature maps (CMM) using a well-established network based on ResNet-50. They also use an attention module to transform the CMM into rare category feature maps (RCM). Lastly, they use various data augmentation techniques like resizing, cropping and random rotating the image data to improve the method's ability to learn from a variety of defects.

The unified classification method proposed is called UCR, it contains 3 sub-networks which are the category representation network, the rare category transfer network, and the spatial pooling module. The category representation network acts as the team's expert on common defects, it analyses and extracts features to generate common category feature maps. The rare category transfer network is the team specialist on rare defects, it has a mechanism called self-attention that focuses on the specific area of CCMs. The network will then transform the CMM to RCM that is better suited for identifying rare defects under the guidance of a proposal feature map (PM). The final part acts as the integrator where it takes both the CMMs and RCMs to analyse their similarities across different image regions. This can help the system to understand the overall defect image. The figure below shows the overall structure of the proposed method (Zhang et al., 2020).

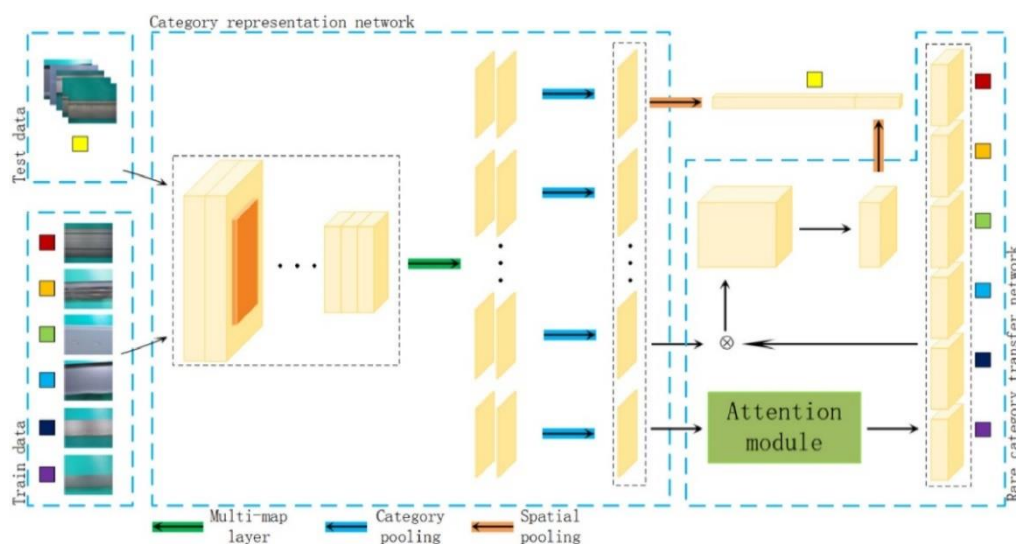


Figure 2.2: Structure of the proposed unified method (Zhang et al., 2020)

The category representation network can be denoted as $s_c = C(\cdot)$ (here x is the input image and s_c is the output containing k_a feature maps while the feature map of a category is represented by each channel. This network is divided into 3 parts where the first part is the feature extractor. This initial step extracts relevant features which is a d -channel feature map from the input image. Besides, it also acts as a general filter to remove background noise from the image and focus on meaningful information. This step is important as the invalid feature cannot be classified correctly, even if the classifier is very powerful. A pre-trained ResNet model but without the final pooling and classification is applied due to its efficiency compared to VGG16 (Visual Geometry Group) and its success in various computer vision tasks. This pre-trained model is fine-tuned on specific defect datasets for optimal performance.

The second part is the multi-map layer that works to take the extracted features and transform them into a format suitable for category-specific analysis. It uses a 1×1 convolution to create $k_m \times k_a$ channel feature maps, where k_m refers to the number of feature maps per category while k_a stands for the total number of categories. If the k_m is set to 1, which means that it becomes a simpler category representation layer. This allows the network to capture various aspects of defects such as its shape and texture.

The last part of this network is the pooling layer that takes the k_m features map generated by each category and merges them into a single-category feature map. The merging process is carried out using the average pooling mentioned above where the average value across channels is calculated into a single value that represents the overall presence or absence of that defect category in the image. The formula used in the process is shown below, where m_j^i represents the j th feature map of category i (Zhang et al., 2020).

$$s_c^i = \frac{1}{k_m} \sum_{j=1}^{k_m} m_j^i$$

As mentioned above, a rare category transfer network relies on PM generated by the attention module to transform CMMs into RCMs. The attention module in this study has 2 branches which are channel attention and spatial attention. Channel attention focuses on specific feature channels within the CMMs and uses 3×3 convolutions to create ‘channel-weight maps’ to highlight the important channels while a 1×1 convolution and sigmoid function to obtain a ‘coarse salient map’ to indicate the overall channel importance. Finally, a dot product operation will combine them to generate ‘channel attention maps’ for each channel. Spatial attention focuses on specific spatial regions within the CMM and uses 1×1 convolutions and a normalization function to generate a spatial proposed mask to emphasize the informative areas. Lastly, the PM is obtained by performing for products between the spatial mask and each channel’s attention map, and the PM is specifically tailored to each rare category (Zhang et al., 2020).

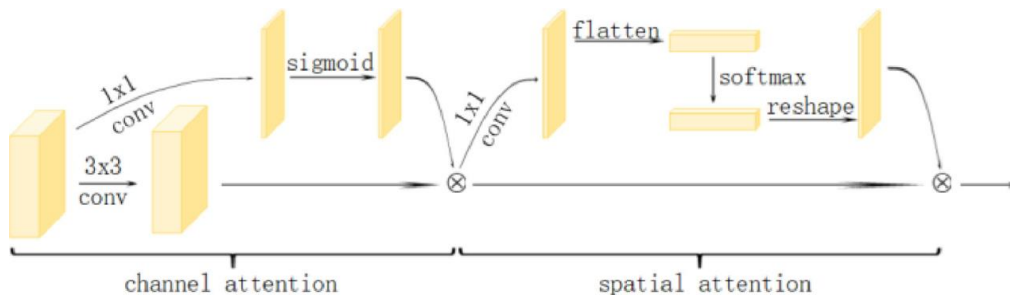


Figure 2.3: Structure of the rare category transfer network (Zhang et al., 2020)

The spatial pooling module unlike the previous parts, does not involve any learnable parameters, instead, it relies on three hyper-parameters to function. The main task of this module is to identify and focus on regions that are likely to contain important features that help in defect prediction within the feature maps. It specifically targets the maximum and minimum values within the feature maps as the extreme values can be informative but each of them carries different types of information. Although both maximum and minimum values are important, maximum values are considered more impactful for classification. This module also ensures that there are an equal number of slots to capture both maximum and minimum values.

The raw data for this study is obtained from a competition hosted by Alibaba and there are 30 defect categories, but the number for each category varied significantly. There are 11 categories with more than 50 samples each, 6 categories with 11 to 50 samples each, and 14 categories with less than 11 samples. However, the 14 categories are not considered due to the overly less samples in the categories while the 11 categories will be common defects and the 6 categories will be rare defects. Figure 2.15 shows the number of samples per category.

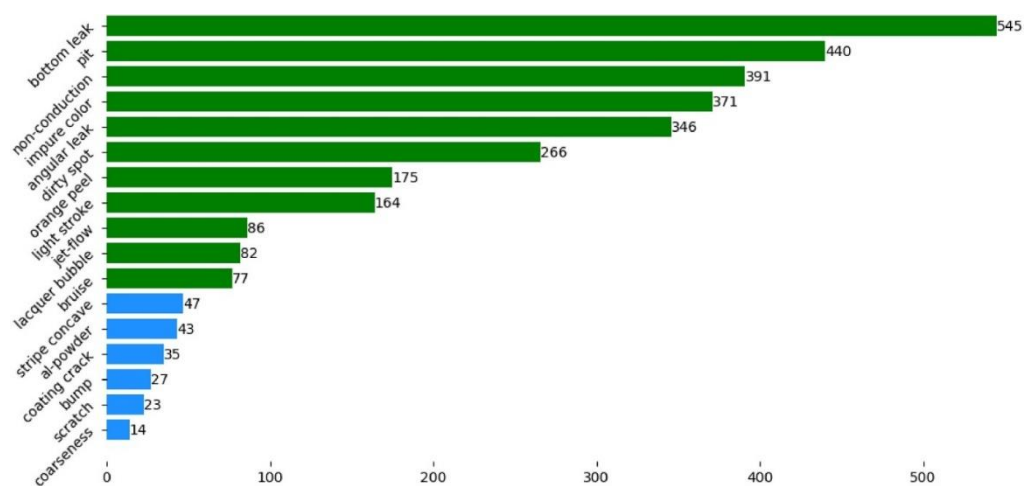


Figure 2.4: Number of samples per category (Zhang et al., 2020)

The accuracy of the UCR method is evaluated by testing the performance of the method on this dataset. The figure below shows the accuracy achieved by the UCR method for both common and rare defects. The result is clear that this method performed well in common defects but some of the rare defects have low accuracy due to the defects may be too similar.

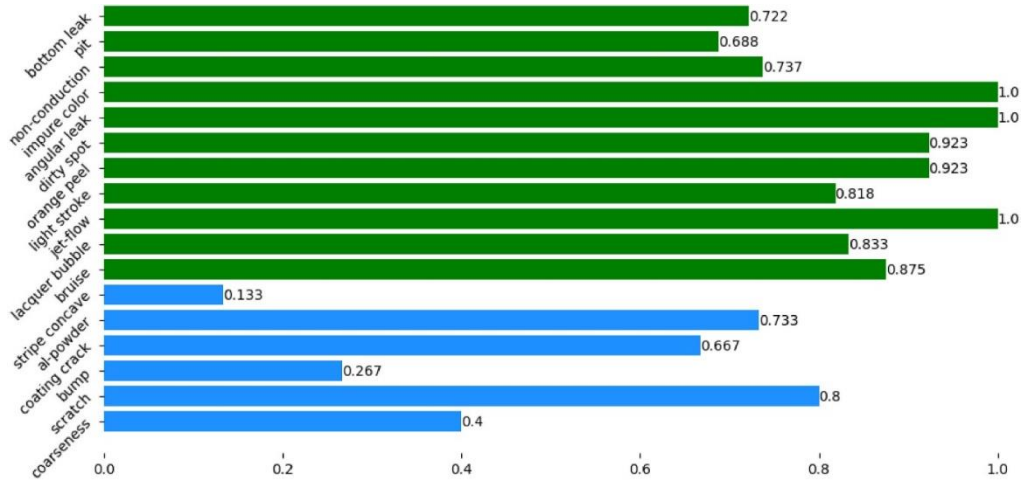


Figure 2.5: Accuracy of UCR method on different defects (Zhang et al., 2020)

Resnet, Deep Feature Selection (DFS), and UCR without attention module are compared with the UCR to evaluate the performance of UCR. As a result, Resnet achieved the highest accuracy for common defects but failed to detect rare defects with only 15.92% accuracy. For rare defects, UCR shines with the highest accuracy of 50% and outperforms all other methods. To show the significance of the attention module in UCR, the UCR without attention module is used in this experiment also, but its performance is bad compared to UCR. In conclusion, UCR demonstrates a good balance between common and rare defect detection with attention module playing a crucial role.

Table 2.1: Performance of different detection (Zhang et al., 2020)

method	ResNet	DFS	URC without att	UCR
common	0.9000	0.8717	0.8061	0.8606
Rare	0.1590	0.4267	0.3667	0.5000

2.1.2 Usage of Transfer Learning and Data Augmentation on Defect Detection

AI is effective for quality control tasks even under common limitations in the real world such as limited training data. It is a common challenge in training neural networks with the lack of extensive training data. However, this could be solved with the help of transfer learning and data augmentation. Transfer

learning leverages the knowledge gained from pre-trained networks for the specific task at hand whereas data augmentations artificially expand the available data. To carry out this study, three cutting-edge deep learning models are used to classify images, which are VGG16, ResNet50, and GoogLeNet, these are the networks that exceed the ImageNet challenge. Demant et al. (1999) state that industrial inspection systems typically rely on cameras that capture images in grayscale. This is because many image processing algorithms designed to detect defects function better on grayscale images compared to colour images. The training dataset from the ImageNet challenge has only 813 images which is quite limited for training powerful deep learning algorithms. Hence, data augmentation techniques are used on the existing images such as shifting, rotating, and flipping horizontally and vertically. By this, it creates new variations of the existing images to help the algorithm to learn from a larger dataset. Empty spaces created by these transformations will be occupied by zero and displayed in black in the training images.

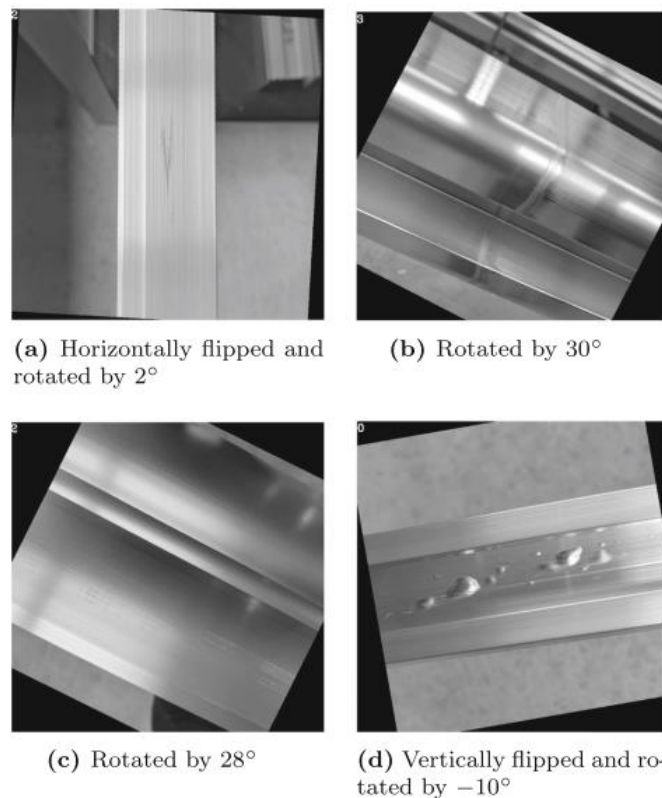


Figure 2.6: Image with data augmentation (Neuhauser et al., 2019)

To accelerate training and compensate for the small dataset size, transfer learning (TL) is applied by leveraging pre-trained models. Since the ImageNet data uses colour images which is size of $224 \times 224 \times 3$, but the defect detection uses grayscale images of size $896 \times 896 \times 1$, the network architecture is modified by doubling the kernel strides taken by the filters in the first two layers for 8 times larger images. Only blue channel weight from the original RGB kernels is used in the first convolution layer because blue light is often used in industry to improve defect visibility on reflective materials. ImageNet challenge classifies 100 categories, so it requires 1000-dimensional output. However, the work only distinguishes between four classes in this study, hence the final layer swapped out for a new layer with just four neurons with randomly initialized weights.

GoogLeNet was a broad and deep network proposed by Szegedy et al. (2015) with an incorporated unique building block called the inception module. Unlike traditional convolutional layers, the inception module applies multiple convolutions with different kernel sizes (1×1 , 3×3 , 5×5) in parallel to the input data to capture features in various scales simultaneously. This network does not need any pre-defined instruction on which convolutions to use, the parallel processing will let the network decide which features are most relevant for the task. While parallel processing increases the amount of information, the inception module also utilizes 1×1 convolutions to reduce the dimensionality before feeding it to the next layer to prevent the network from becoming overly complex. GoogLeNet utilizes a deep architecture with three initial convolutional layers followed by nine unique building blocks called inception modules and a single dense layer. The network incorporates two additional classification points midway through to aid in backpropagation which allows the error signal to flow more effectively and train the initial layers more effectively.

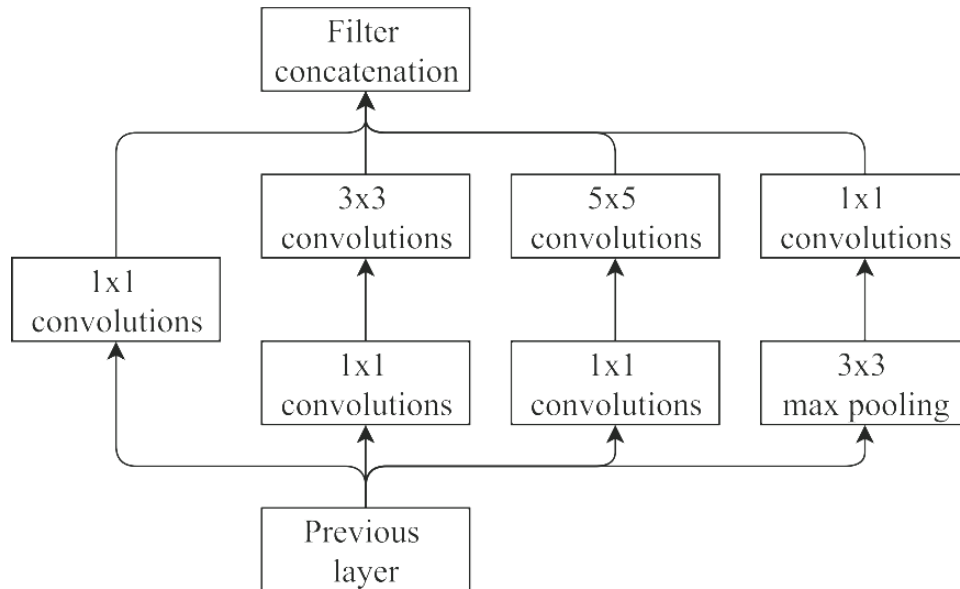


Figure 2.7: An inception module with dimensionality reduction (Neuhauser et al., 2019)

During the training process, the models aim to minimize a function called the ‘multi-class cross-entropy loss function’ which measures the difference between the models’ predictions and the actual labels of the training data. Table 2.3 includes the number of images processed together during training, the total number of training cycles, the learning rate, average training loss, and the score in classifying images on separate validation and test datasets by the network architectures used. VGG16 only has 8 batch sizes because of its complex architecture with a vast number of parameters to learn and burden to the graphic cards. The models are also tested again with only one data augmentation or transfer learning given to monitor their performance. However, this time the number of training cycles is changed according to the early stopping approach to prevent overfitting (Yao, Rosasco, and Caponnetto, 2007). Using only data augmentation, GoogLeNet, and ResNet achieved relatively low test set accuracy with 0.61 and 0.65 respectively while scoring 0.79 and 0.75 with TL. This highlights the benefit of leveraging pre-trained knowledge from a large dataset. VGG16 as the only exception performs well with data augmentation but struggles with only TL. Overall, the results emphasize the importance of combining transfer learning with data augmentation for optimal

performance in this specific task of surface defect detection with a limited dataset (Neuhauser et al., 2019).

Table 2.2: Training results when trained with transfer learning and data augmentation (Neuhauser et al., 2019)

Network (TL + DA)	Batch size	Iterations	Learning rate
VGG16	8	30 000	1e−5
GoogLeNet	16	15 000	1e−5
ResNet50	16	15 000	1e−4

Network (TL + DA)	Training loss	Validation score	Test score
VGG16	0.04	0.62	0.76
GoogLeNet	0.41	0.91	0.80
ResNet50	0.06	0.97	0.98

Table 2.3: Training results when only trained with data augmentation and when only trained with transfer learning (Neuhauser et al., 2019)

	Iterations	Test score
Network (DA)		
VGG16	60 000	0.81
GoogLeNet	30 000	0.61
ResNet50	30 000	0.65
Network (TL)		
VGG16	4 000	0.5
GoogLeNet	5 000	0.79
ResNet50	5 000	0.75

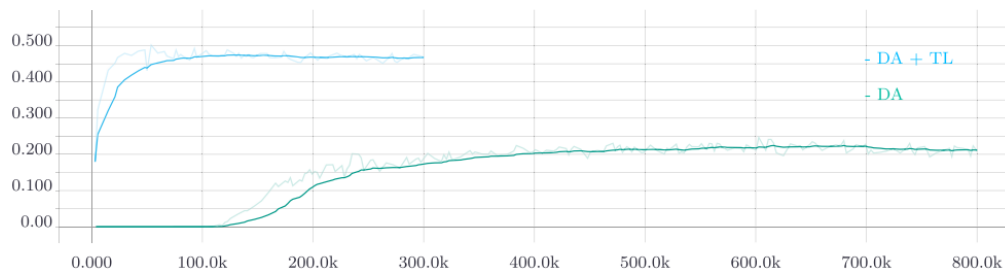


Figure 2.8: mAP@0.5 with TL and without TL over 800000 iterations
(Neuhauser et al., 2019)

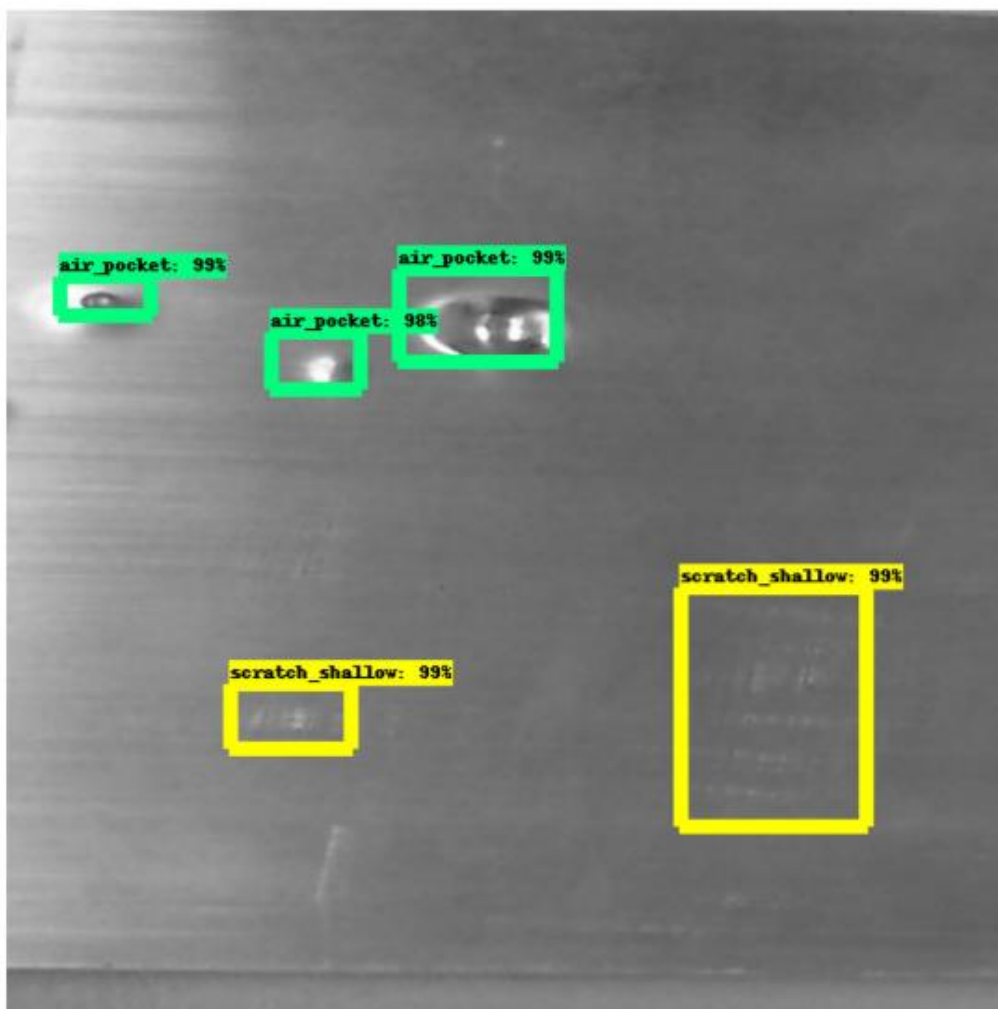


Figure 2.9: Example of defect detection using ResNet50 with DA and TL
(Neuhauser et al., 2019)

2.1.3 Defect detection of aluminium profile surface using MS-YOLOv5 based on YOLOv5

There are a lot of algorithms for defect detection nowadays, but the algorithms for detecting defects on aluminium profiles have a shortage such as slow processing time, limited defect type detection, and challenges in capturing features from the input image. Hence, Wang et al. (2022) proposed a powerful algorithm named MS-YOLOv5 which is based on YOLOv5. A PE-neck using a special convolution technique, Poly-Scale Convolution (PSConv) with a channel attention mechanism is proposed to replace the original YOLOv5 neck. MS-YOLOv5 also includes a multi-treatment that is inspired by pyramid convolutions (PyConv) to improve the detection of randomly distributed defects. It modifies the original YOLOv5 architecture by incorporating residual connections and utilizing the first detection head, increasing the effectiveness of the network in capturing defects scattered across the image surface. In addition, they acknowledge the challenge of limited training data, hence they use data augmentation techniques such as geometric transformations and image processing to the training set. This creates a variation of the existing dataset to decrease the model's dependence on specific features present in the original dataset.

The dataset created for training and testing the detection model comes from the Ali Tianchi database and some actual defect images captured in an Guangxi factory. There is a total of seven types of defects, contributing 3098 defect images including concavity, dirty spot, orange peel, non-conducting, scrape under screen, and embossing. As mentioned above, data augmentation techniques are used due to an insufficient initial dataset which might lead to overfitting and reduced accuracy and generalizability. Data-enhancement strategy is also applied to the training set such as adjusting gamma levels, brightness, and contrast. With these data augmentation methods, the dataset has grown into 7777 images for training, providing more learning opportunities to the model, while 1987 images are used for model testing. Different lighting conditions can alter the visibility of the defect details; hence gamma variation specifically targets enhancing dark details within the images. It applies a non-linear transformation that brightens dark areas to reveal more defect features

that might be obscured. While contrast variation improves the overall image quality by adjusting the contrast between the different image elements and brightness variation simulates the effects of high-intensity light on the image. These techniques are used to expose the model to a wider range of lighting conditions and are more adaptable to real-world scenarios where light conditions would not be perfectly controlled (Wang et al., 2022).

	Concavity	Dirtyspot	Scrape	Embossing	Underscreen	Nonconducting	Orangepeel
Train							
Original	185	143	158	135	178	185	127
H_flip	185	143	158	135	178	185	127
V_flip	185	143	158	135	178	185	127
HV_flip	185	143	158	135	178	185	127
Gamma	185	143	158	135	178	185	127
Contrast	185	143	158	135	178	185	127
Bright	185	143	158	135	178	185	127
Total				7777			
Test							
Original	320	256	216	320	277	278	320
Total				1987			

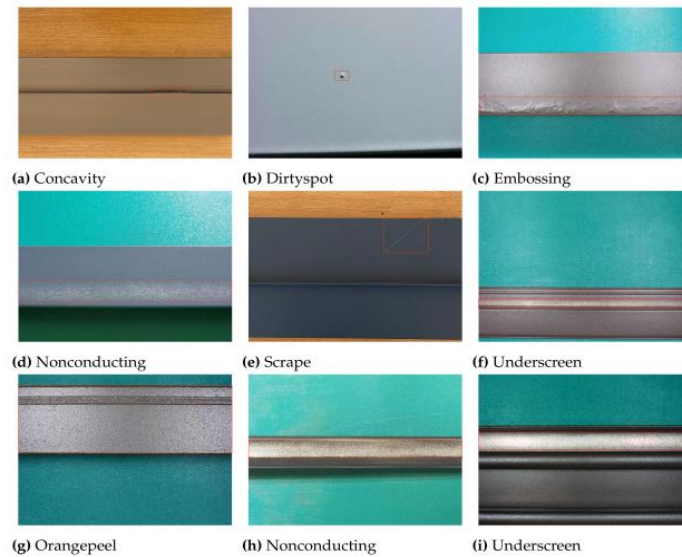


Figure 2.10: Number of datasets for training and testing (Wang et al., 2022)

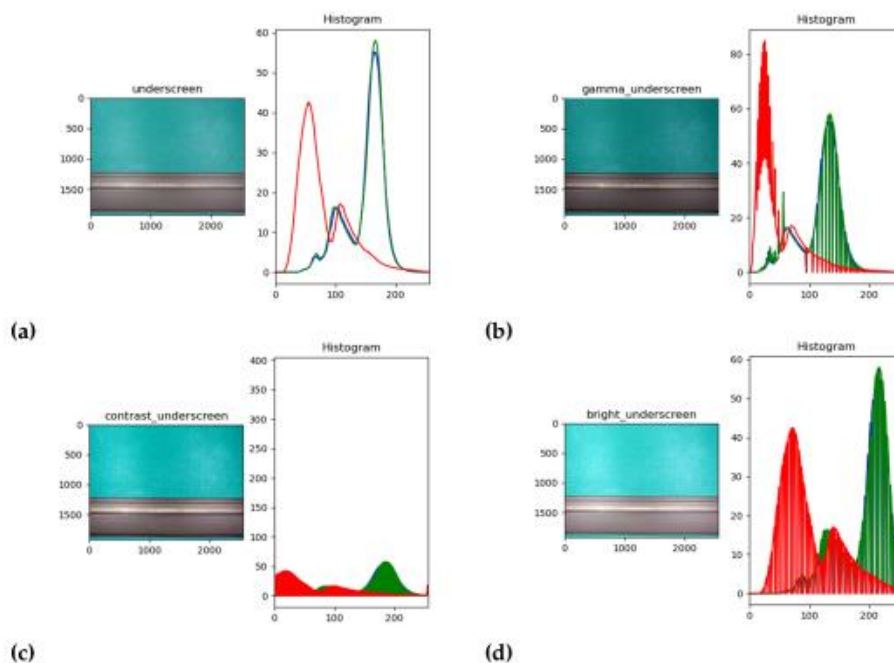


Figure 2.11: RGB comparison histogram. (a) Original (b) Adjusted Gamma (c) Adjusted Contrast (d) Adjusted Brightness (Wang et al., 2022)

The structure of MS-YOLOv5 includes the backbone, neck, and detection head. The backbone is the foundation of the model and is responsible for extracting core features from the input image, MS-YOLOv5 utilizes the well-established CSPDarknet-53 architecture for the backbone. The neck of MS-YOLOv5 uses the PE-Neck while the detection uses the multi-streamnet as the first detection head.

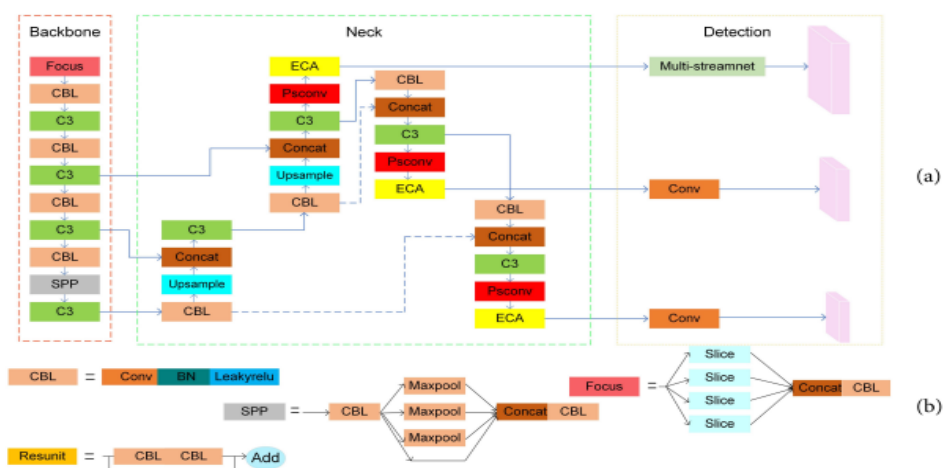


Figure 2.12: Structure of MS-YOLOv5. (a) Backbone, PE-Neck, and multi-streamnet (b) composition of modules (Wang et al., 2022)

PSConv is a special type of convolution that incorporates multiple ‘dilation factors’ within a single kernel, it uses a set of different dilation factors within the same kernel (Woo et al., 2018). PSConv creates a filter that can extract feature information at different scales simultaneously by alternating varying dilation factors in a cyclic manner, allowing them to capture details from both small and large defects within the same image. The calculations by these different kernels within the PSConv kernel alternate across the channel to ensure the information at various scales is extracted for each feature channel, providing a comprehensive representation of the image data.

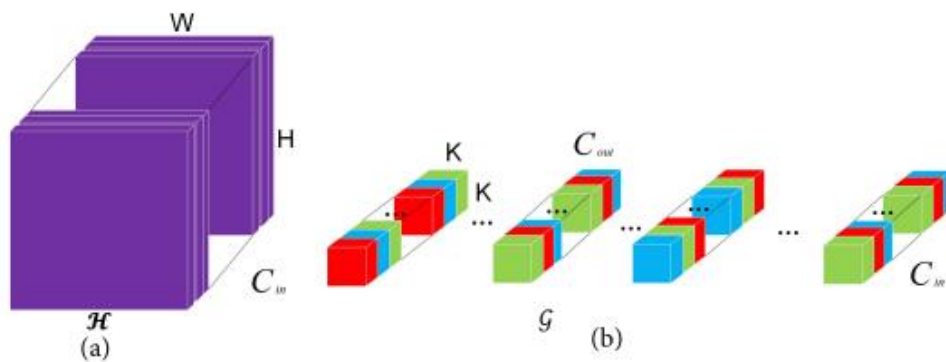


Figure 2.13: Architecture of PSConv. (a) input features (b) kernel incorporates various dilation rates (Wang et al., 2022)

The research mentioned above has shown the importance of attention modules in CNN for improved performance. However, most attention mechanisms have complex structures which leads to large network size, increased training time, and slow inference time which means slower predictions on new data. Efficiency Channel Attention (ECA) is an alternative to overcome the shortage of existing attention mechanisms. Some approaches like SE-Net will oversimplify the process by reducing the feature dimensionality, which will lower the capability of the model to learn and utilize the important channel information effectively. Hence, ECA proposes a dimensionless local cross-information interaction strategy to achieve good performance while maintaining a lightweight structure. This strategy relies on simple 1-dimensional convolutions instead of complex structures. This technique allows

the model to dynamically choose the most suitable 1D convolution kernel for each specific channel, enabling it to capture relevant cross-channel interactions effectively.

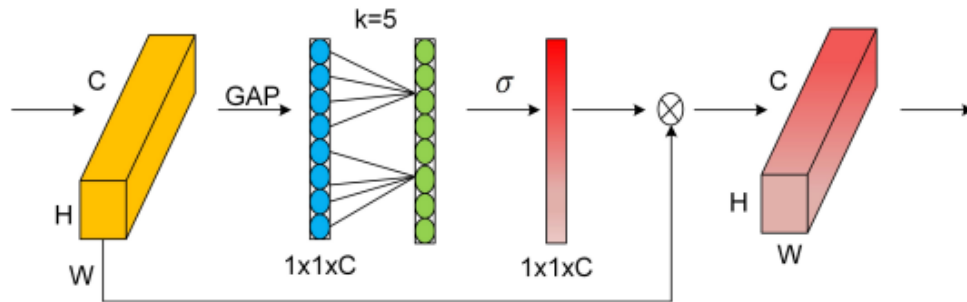


Figure 2.14: Schematic representation of the ECA module's architecture

(Wang et al., 2022)

The standard YOLOv5 neck component aims to extract features at various scales and combine them for object detection, but the existing neck might struggle to capture a wide range of defect sizes effectively. This is because the original neck component performs unnecessary feature extraction in its upper part, causing fragmented features which means breaking down features and making them less informative. Hence, PSConv, as explained earlier, is used in PE-neck to extract information from defects of various sizes within the aluminium profiles. While PSConv focuses on capturing semantic information, it might neglect the precise location. Therefore, the ECA module is used to address this by enhancing the localization information extracted by the network. This PE-Neck also utilizes jump connections to bypass these redundant steps and directly feed suitable features which contain both rich semantic and accurate localization information into the detection layer.

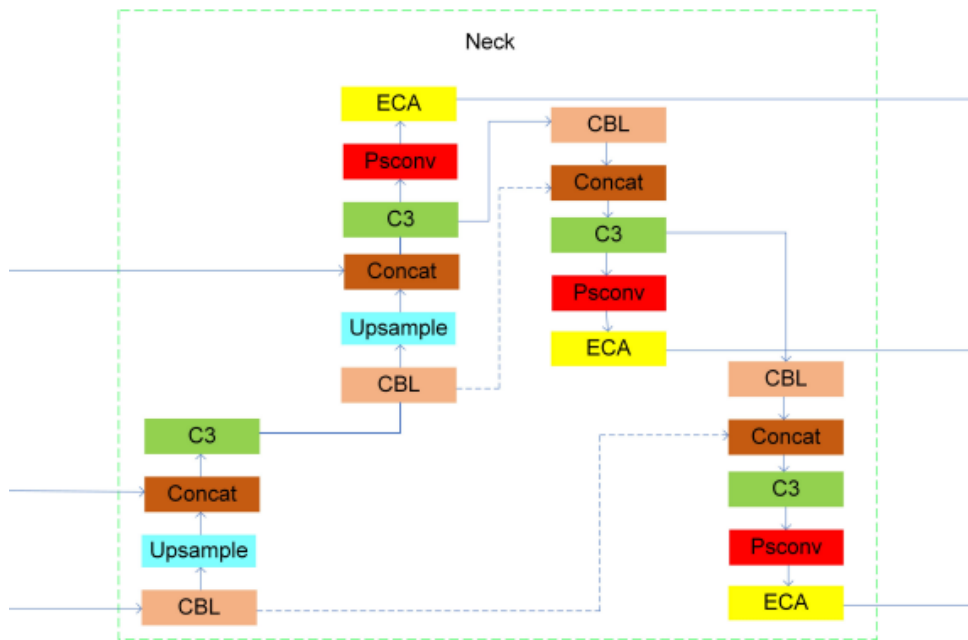


Figure 2.15: Structure of MS-YOLOv5's neck (Wang et al., 2022)

In image processing, convolutions with different kernel sizes are used to extract features at various scales like PyConv and OctaveConv to expand the model's ability to perceive features across different image regions. (Duta et al., 2021). However, this approach may generate a significant amount of redundant information that will cause slow processing time to process unnecessary data, the abundance of redundant data makes it harder to identify defects accurately and increases model complexity which leads to a larger number of parameters that require more resources for training. To overcome these limitations, a multi-stream network is proposed to control redundancy. Instead of relying on varying convolution depths, this network uses the quantity of convolutions at different scales to manage the overall feature map depth to ensure all scales have the same number of feature maps. This approach also optimizes the computational cost by ensuring a fixed amount of feature maps are produced and maintaining a reasonable level of computation. The multi-stream network is inspired by the ResNet architecture to incorporate residual connections to learn and discard redundant information within the network structure without sacrificing performance. Lastly, these improved features are fed into the initial detection stage of the YOLOv5 architecture to enhance the ability of the model to pinpoint defects scattered across the image surface.

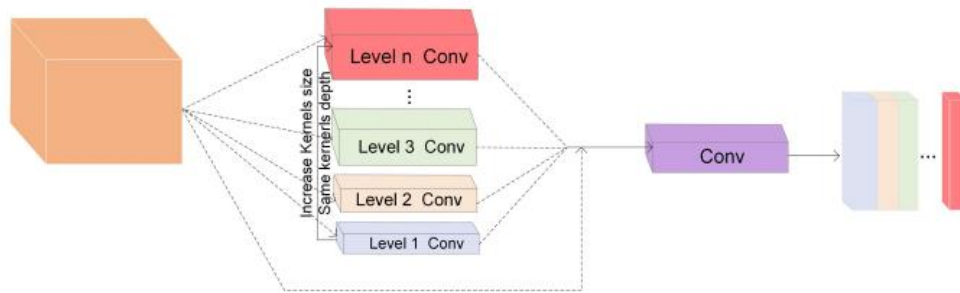


Figure 2.16: Structure of multi-streamnet (Wang et al., 2022)

By adding P-Neck to original YOLOv5, it improved recall by 2.5% and mean Average Precision (mAP) by 1.4%. However, it further boosts its performance by replacing P-Neck with PE-Neck, recall, and mAP increased by 0.6% and 0.2% respectively, where mAP has a large improvement of as much as 1.7%. Integration of a multi-stream network further increases the mAP by 0.2% and precision by 1.2%, but it does not help in increasing recall. Compared to the original YOLOv5, MS-YOLOv5 with PE-Neck and multi-stream network added achieved significant improvement in mAP with 3.3% and 1.1% increase in recall. Notably, the detection speed only increased by 1FPS, making the model more efficient for real-world scenarios.

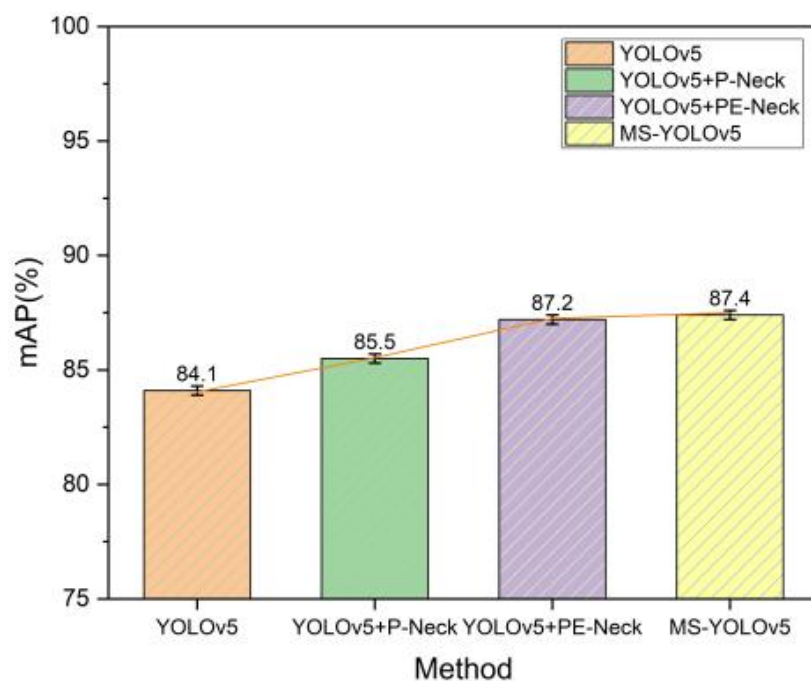


Figure 2.17: Comparison of mAP of each detection (Wang et al., 2022)

Table 2.4: Comparison of each detection based on precision, recall, mAP, and FPS (Wang *et al.*, 2022)

Method	Precision(%)	Recall(%)	mAP(%)	FPS
YOLOv5	91.6	75.4	84.1	20.1
YOLOv5+P-Neck	89.8	77.9	85.5	19.8
YOLOv5+PE-Neck	90	78.5	87.2	19.5
YOLOv5 + PE-Neck + Multi-streamnet(MS-YOLOv5)	91.2	76.5	87.4	19.1

The comparison continues by comparing these methods with F1-cure and PR-cure as metrics for overall performance. F-1 cure considers both precision and recall where MS-YOLOv5 strives in these metrics compared to original YOLOv5. PR-cure plots precision against the recall where the enclosed area represents the mAP. Figures 2.30 and 2.31 show that MS-YOLOv5 has a bigger area under the curve than YOLOv5, signifying the improvement in mAP for MS-YOLOv5. The figure below shows the actual defect detection for both methods. Both methods work well in detecting well-defined defects, but MS-YOLOv5 excels at detecting defects with challenging characteristics. For example, it successfully identifies scrape that has variable sizes and dense distribution while YOLOv5 misses it. Dirtyspot defects are small and randomly distributed, YOLOv5 only detects one, whereas MS-YOLOv5 finds all of them. The MS-YOLOv5 model excels at detecting "under screen" defects, which are particularly challenging due to their similar colour to the background.

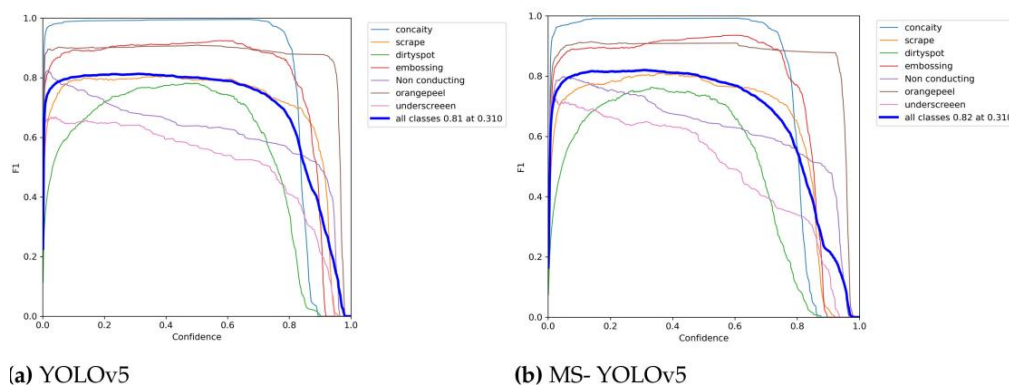


Figure 2.18: F1-cure comparison for each defect (Wang *et al.*, 2022)

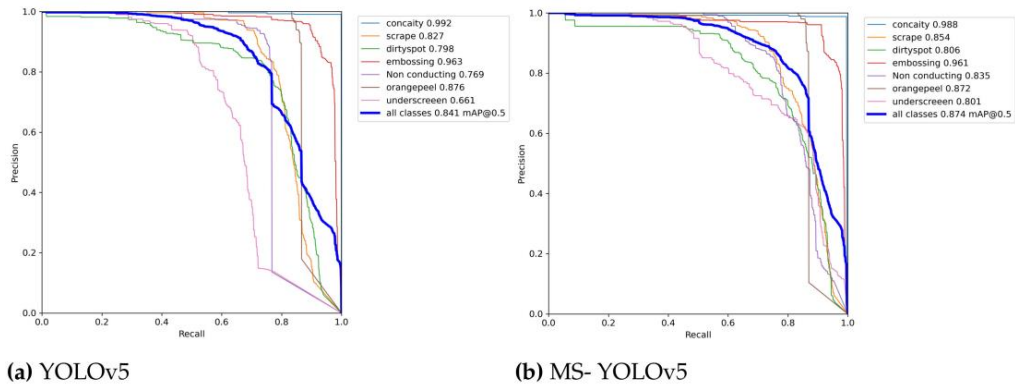


Figure 2.19: PR-cure comparison for each defect (Wang et al., 2022)

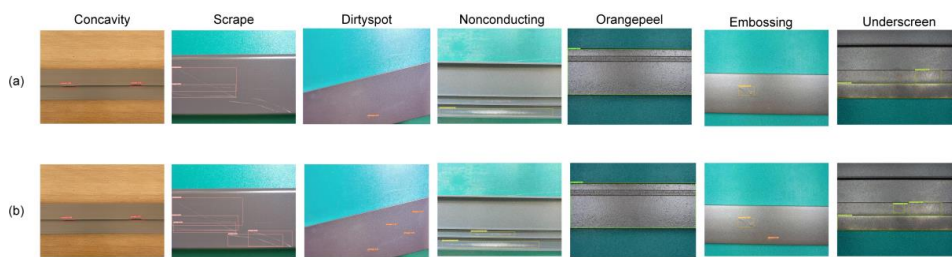


Figure 2.20: Comparison of YOLOv5 and MS-YOLOv5 in detecting and localizing defects (Wang et al., 2022)

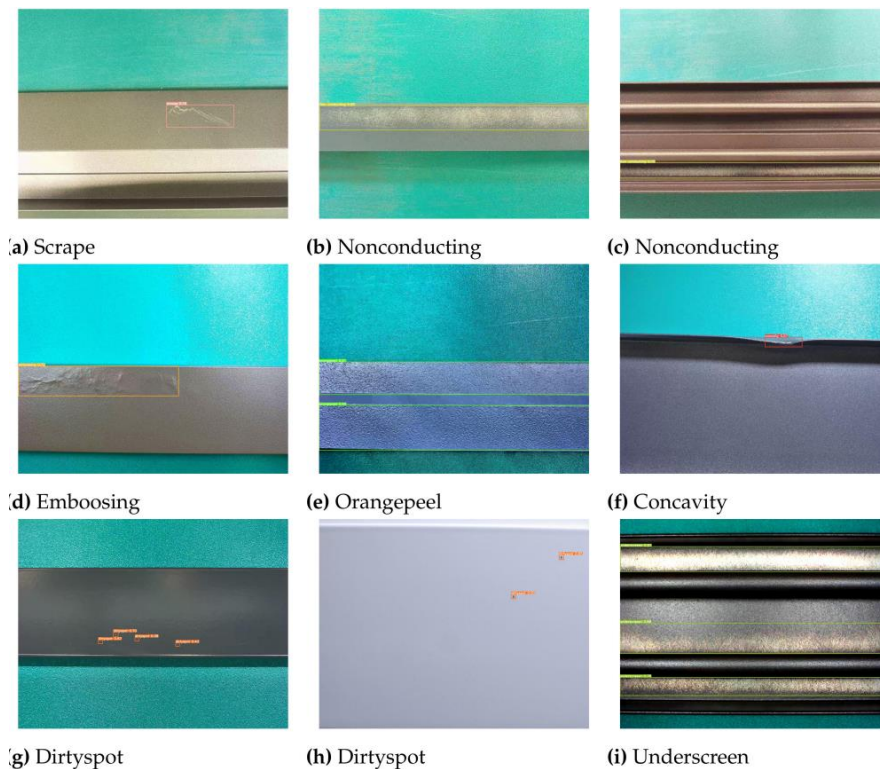


Figure 2.21: More results of MS-YOLOv5 in detecting defects (Wang et al., 2022)

The performance of MS-YOLOv5 is further compared with other established object detection algorithms such as YOLOv4, YOLOv4, SSD, Faster-RCNN, and original YOLOv5. The table below shows the dominance in performance across the board. It achieves an Average Precision (AP) above 80% for each defect type, showing its consistency and reliability detection across all defect categories. Unlike other algorithms, MS-YOLOV5 avoids extreme imbalance in detection performance as it is important for real-world applications that require effective detection regardless of size variations. MS-YOLOv5's overall mAP reaches 87.4% which is higher than any other algorithms compared. It achieved an impressive improvement of 11.54% over YOLOv3, 4.59% over YOLOv4, and 3.3% over the base model YOLOv5 itself. MS-YOLOv5 also achieved a remarkable 20.4% improvement over SSD and 5.71% over Faster-RCNN.

Table 2.5: AP performance for each algorithm (Wang et al., 2022)

Model	AP(%)							mAP(%)	FPS
	Concavity	Scrape	Dirty Spot	Embossing	Non Conducting	Orange Peel	under Screen		
YOLOv3	97.14	48.07	40.56	78.77	89.59	93.42	83.44	75.86	21.3
YOLOv4	97.71	74.43	51.05	81.15	94.69	96.42	84.25	82.81	20.4
YOLOv5	99.2	82.7	79.8	96.3	76.9	87.6	66.1	84.1	20.1
SSD	72.91	60.96	20.9	64.25	82.14	90.81	76.97	67	21.7
Faster-RCNN	92.14	70.74	38.7	94.03	90.36	97.40	88.49	81.69	12.6
MS-YOLOv5	98.8	85.4	80.6	96.1	83.5	87.2	80.1	87.4	19.1

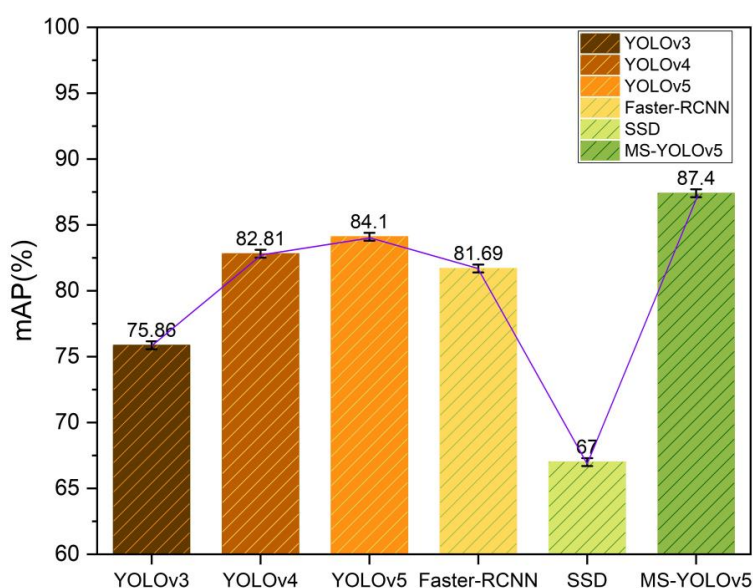


Figure 2.22: A comparison chart of mAP for each algorithm (Wang et al., 2022)

2.1.4 Lightweight model network based on YOLOv5s

There is a study that proposed a new lightweight model network designed for real-time detection of aluminium profiles based on the YOLOv5s framework. Speed and efficiency for real-time inspection are the main priority in this network. This network architecture consists of 4 parts which are the input layer, GCANet Backbone, neck, and prediction. The input layer is responsible for receiving input images and resizing them into a standard format of $640 \times 640 \times 3$ before feeding them into the network. GCANet Backbone is the core of this network which is responsible for feature extraction. It incorporates an attention module with a specific module called C3Ghost to help the model to focus on crucial details in the images, in terms of both colour channels and spatial information. There are 3 scale feature maps which are 80×80 , 40×40 , and 20×20 extracted at different levels. The neck layer utilizes depthwise separable convolution (DwConv) to further compress the model size to make it more lightweight and efficient. The final stage performs the actual defect detection based on the processed features extracted from the previous layer.

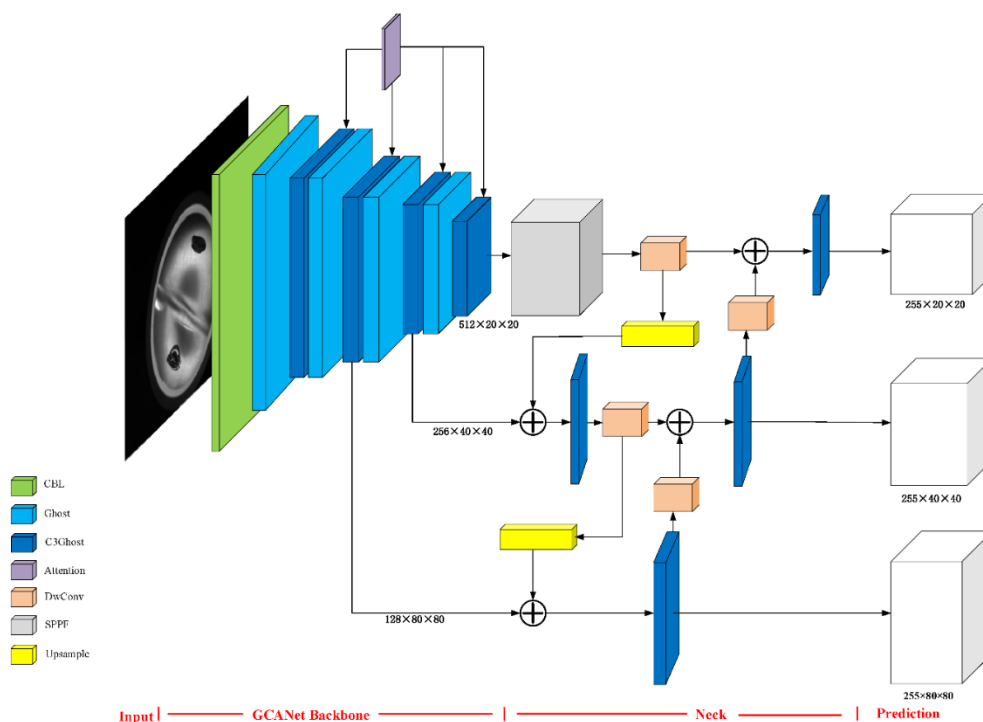


Figure 2.23: The proposed network architecture (Tang et al., 2023)

The GCANet backbone as the core component for extracting features consists of 3 parts which are the CBL module, Ghost Module, and AC3Ghost Module. CBL module is a common building block in CNN which consists of 3 parts also which are Conv, BatchNorm, and Leaky ReLU. Conv applies a standard convolution filter to the input data while BatchNorm normalizes the data across channels and improves training stability whereas Leaky ReLU introduces a non-linear activation function that allows the network to learn complex patterns. The ghost module was originally introduced in Huawei's GhostNet architecture in 2020 and aims to reduce computational cost and the number of parameters. It starts with applying a regular convolution to generate a small number of high-quality feature maps, followed by creating additional feature maps using computationally inexpensive linear operations.

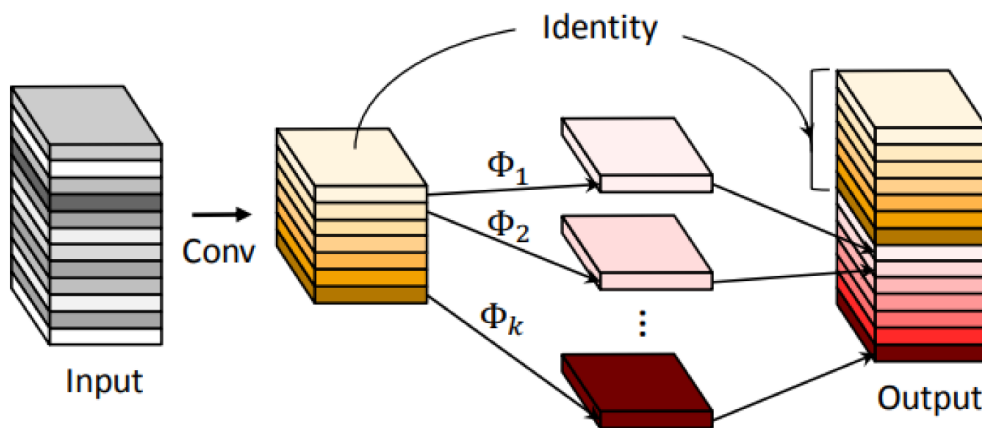


Figure 2.24: Ghost module structure (Tang et al., 2023)

AC3Ghost module utilizes the CBAM attention mechanism proposed by Hou, Zhou, and Feng (2021) and embedded it in C3Ghost. This mechanism allows the network to focus on both channel-wise and spatial information within the feature maps. With this, it helps the model to distinguish between important and unimportant aspects within the image. This module works by receiving data processed by the regular Ghost module and splitting the data into 2 branches for parallel processing. The first branch is to perform a combination of features from different levels using multiple Ghost Bottleneck stacks and three 1x1 convolution modules. This is to combine the features extracted at different levels for a more comprehensive understanding of the image. The second branch

works to reduce the number of channels using a single 1×1 convolution module. The resulting maps from the two branches are combined by concat to create a richer representation of the image. After that, the CBAM attention mechanism is applied to the combined feature maps to focus on the most relevant channel and spatial information for defect detection. The final processed features are passed through a final 1×1 convolution module before being used for the next stage of the network.

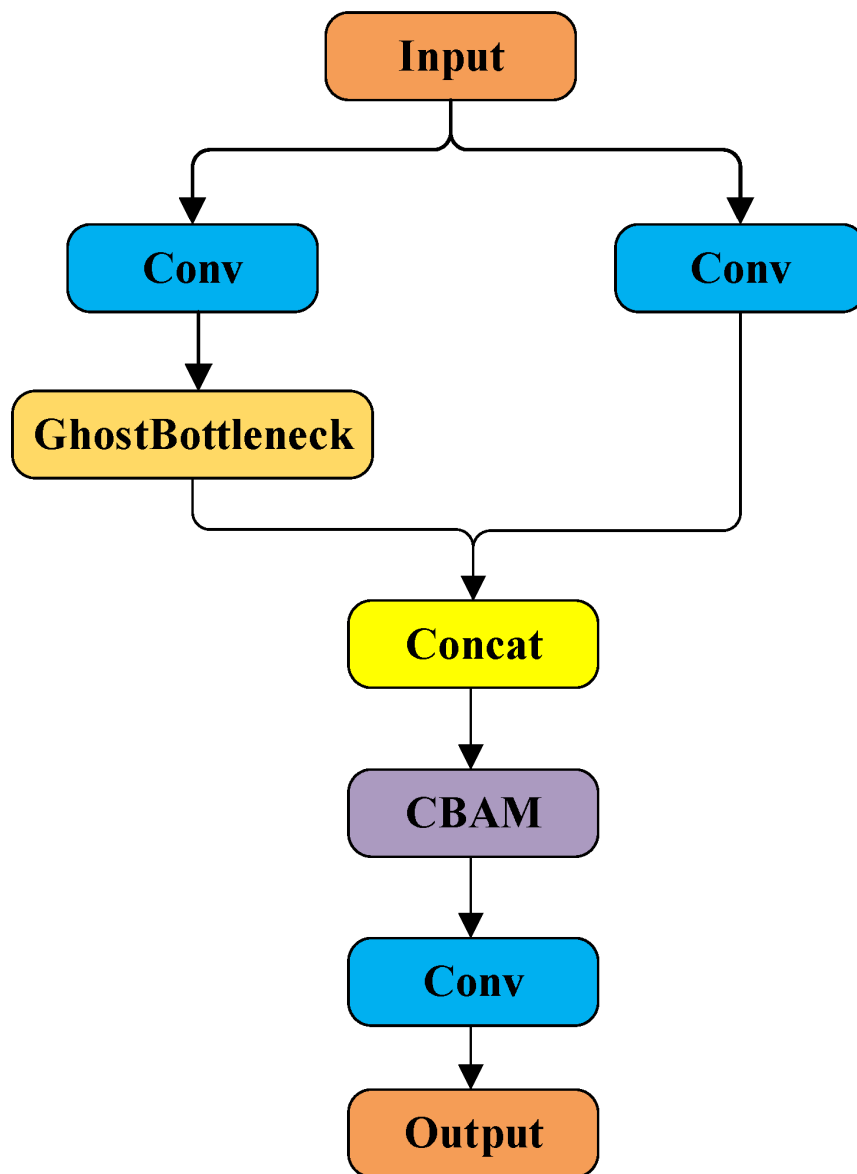


Figure 2.25: AC3Ghost module Structure (Tang et al., 2023)

The evaluation metrics used in this study include average precision (AP), mean average precision (mAP), mAP@0.5 and mAP@0.5:0.95. Average

precision simply means how well the model performed for a single category of defects where it considers both completeness and correctness to avoid false positives. The mAP considers the model’s overall accuracy where it takes the average AP scores across all defect categories the model is trained to detect. The mAP@0.5 is a specific variant of mAP that focuses on detection where the predicted bounding box overlaps with the ground truth bounding box by at least 50%. A high mAP@0.5 indicates the model is good at accurately locating the defects in addition to finding them. The mAP@0.5:0.95 is a metric that extends the concept of mAP@0.5 where the AP is calculated for IoU thresholds ranging from 0.5 to 0.95. This provides a more detailed picture of how well the model performs at different levels of bounding box overlay accuracy. There are only 4 categories of defect included in this study which are pinhole, scratch, dirt, and fold (Tang et al., 2023).

The accuracy of the proposed model is evaluated using a fivefold cross-validation method which means splitting the dataset into 5 groups and four groups will be used to train the model while the remaining one group will be used for testing. As shown in Table 2.6, the model demonstrates strong performance in the folding category with 99.45% precision, 100% recall, and 99.37% AP@0.5%. These high values indicate the model effectively detects these defects with minimal false positives and accurate localization. However, this model faces challenges with pinholes, which have fewer pixels and less texture information. To address this, the paper employs techniques like Gaussian blur and copy-pasting to artificially generate more features during training. This approach improves pinhole detection, achieving an AP of 82.45%. The visual inspection results are shown in Figure 2.36 with all 4 defects detected with a confidence level above 0.8 (Tang et al., 2023).

Table 2.6: Result of the four types of defect detection(Tang et al., 2023)

	Pinhole	Scratch	Dirt	Fold
Precision	83.24	98.86	99.11	99.45
Recall	77.21	99.39	99.22	100.00
AP@0.5:0.95	51.94	79.80	81.10	80.60
AP@0.5	82.45	98.77	98.80	99.37
mAP@0.5:0.95		73.36		
mAP@0.5		94.85		

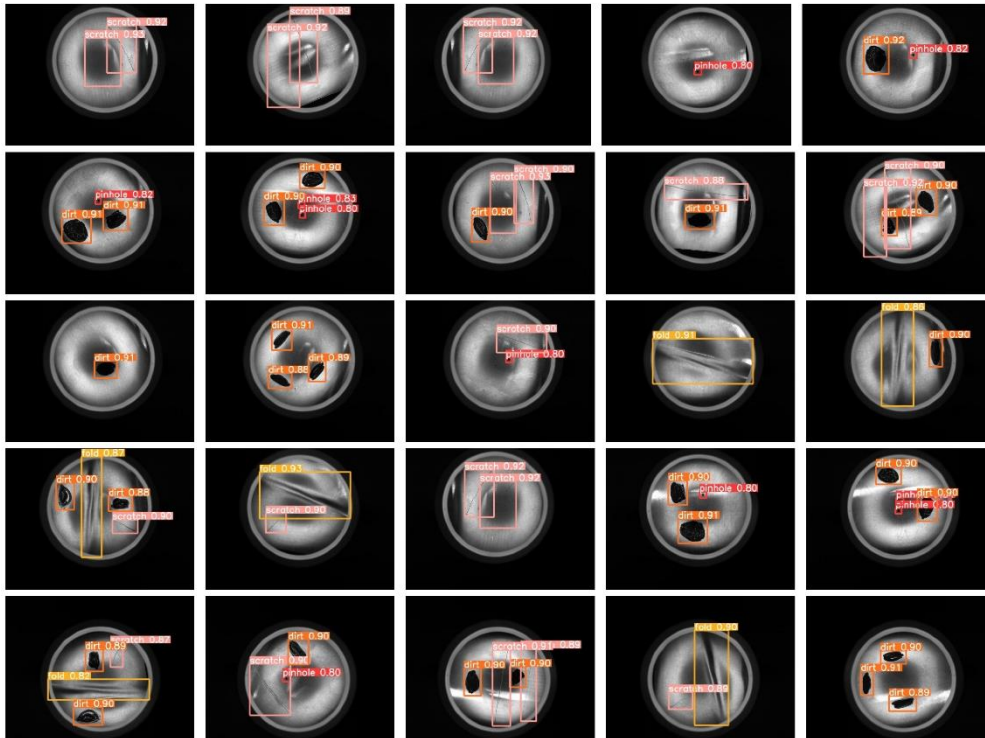


Figure 2.26: Object detection results for four defects (Tang *et al.*, 2023)

The proposed method is further compared against established single-stage object detection such as SSD and other YOLO variations. Among all the detection methods tested, the proposed method achieves the highest in $mAP@0.5$ with 94.85% and $mAP@0.5:0.95$ with 73.36%. Although YOLOv5-Mobilenetv3 and YOLOv5-Shufflenetv2 offer faster detection speeds, their $mAP@0.5$ falls below 90%, indicating their low overall accuracy in detecting defects.

Table 2.7: Outcome of each algorithm (Tang *et al.*, 2023)

Algorithm	$mAP@0.5$	$mAP@0.5:0.95$	Model Size (MB)	Detection Speed (FPS)
SSD	67.86	42.76	91.09	39
YOLOv3-tiny	85.82	65.89	33.19	66
YOLOv4-tiny	89.31	67.86	23.09	86
YOLOv5s	93.09	72.58	14.40	75
YOLOv5-Mobilenetv3	89.71	67.78	5.96	158
YOLOv5-Shufflenetv2	89.65	66.86	3.30	176
Ours	94.85	73.36	7.80	136

2.2 Other usages of YOLO on Metal Surface Defect Detection

Yun et al. (2009) state that X-ray testing is great for spotting defects in steel pipe welds, but it takes humans to figure out what kind of defects it is and where it is located. This is where it shows the importance of AI, machines could be trained to identify weld defects automatically with the help of deep-learning object detection. Hence, Yang et al. (2021) propose the usage of YOLOv5 to find defects in steel pipe welds using X-ray images. As a comparison to the performance of YOLOv5, Faster-RCNN is also used in this study to be trained on the same dataset to compare their precision and total loss during the training. Faster R-CNN's precision bounced around a lot during training, starting high then going down, and slowly coming back up, while YOLOv5 started shaky but steadily improved and then stabilized. For the total loss, Faster R-CNN was stable between 50 and 100 epochs but had a couple of large bumps while YOLOv5 started lower but became more stable between 100 and 150 epochs and efficient after training for a while. YOLOv5 achieves an impressive precision of 97.8%, with a mAP@0.5 of 98.7. YOLOv5 can detect a single image in just 0.12 seconds which is suitable for real-time defect detection during steel pipe production. Overall, YOLOv5 trains faster and ends up being more stable and efficient than Faster R-CNN for this weld defect detection task.

Table 2.8: Accuracy of each algorithm (Yun et al., 2009)

Object detection model	Accuracy or precision/%
GAN+CFM (Wu et al. 2021)	85.9 acc (mIoU)
OSTU+MSVM-rbf (Malarvel and Singh 2021)	95.23 acc
Faster R-CNN+ResNet50 (Ren et al. 2017)	95.5 acc (mAP@0.5 = 78.1)
YOLOv5x	97.8 pre (mAP@0.5 = 98.7)

Shi et al. (2022) point out the difficulties of inspecting steel surfaces during production due to the many kinds of defects that can arise and their small size, strict quality control makes manual inspection even harder. Therefore, they propose an improved version of YOLOv5 which focuses on attention mechanism and k-means clustering for anchors. The attention mechanism is to help transfer the important details, especially from shallow features from the initial processing stage which is the backbone to the neck of the YOLOv5 model.

On the other hand, k-means clustering is to optimize the anchor boxes used in YOLOv5 to help the model in handling the unusual shapes of defects present in their steel surface dataset. Compared to the original YOLOv5, their improved model achieved a higher mAP of 86.35% whereas the original only achieved 81.78%. As a trade-off for the increased accuracy, the detection speed dropped a bit from 52 FPS to 45 FPS.

Ma et al. (2022) use YOLOv4 as a base and modify it with depth-wise separable convolutions and parallel dual attention mechanism to tackle the challenge of slow processing for detecting defects in aluminium strips during production. Depth-wise separable convolutions are used to make the network more efficient by reducing the number of calculations needed while a parallel dual attention mechanism is used to help the module to focus on the importance of the image containing defects. Compared to YOLOv4, their improved network has its network size reduced by 83.38% which makes it run faster and requires less memory. One of the improvements they customized is the anchor boxes used to detect long and thin defects such as scratches in their dataset. It reaches a high accuracy on the real data from an aluminium cold-rolling factory with an mAP of 96.28%.

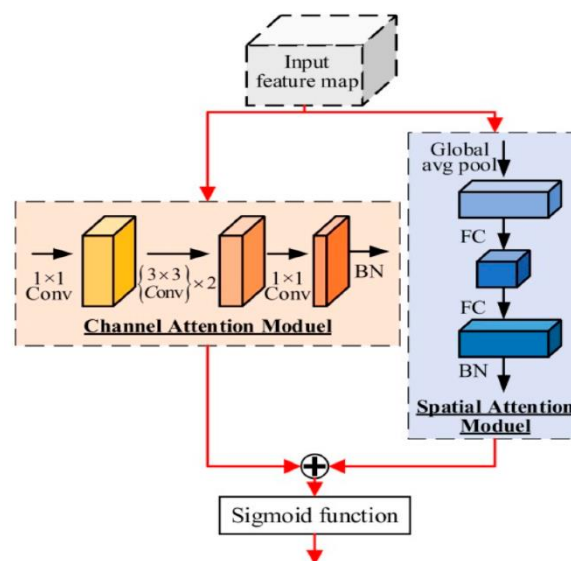


Figure 2.27: Parallel Network Structure proposed by Ma et al. (2022)

2.3 YOLOv8

A new version of YOLO, YOLOv8, was released in January 2023 and is claimed to be better than previous versions. Although there is still no full explanation yet, initial tests on YOLOv8 show that it performs more with lesser computing power compared to its predecessors. It is interesting to note that both YOLOv5 and YOLOv8 are from the same company which is Ultralytics and YOLOv5 is known for its speed and performance. This suggests that YOLOv8 might be designed for devices with lower processing power that need very fast object detection. The figure below shows the comparison between YOLOv8, YOLOv5, YOLOv6, and YOLOv7 trained on 640 image resolution. The result shows that YOLOv8 has better throughput than its predecessors with a similar number of parameters, indicating its efficiency when it comes to hardware usage.

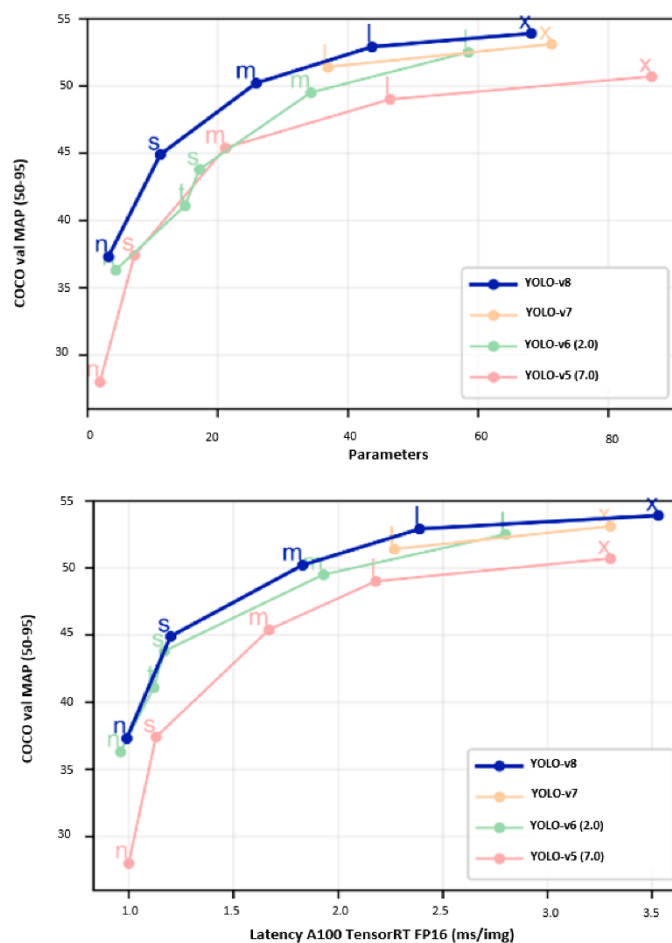


Figure 2.28: YOLO-v8 comparison with predecessors (Hussain, 2023)

Table 2.9 shows the comparison between different YOLO versions based on the framework, backbone, AP, and their main contribution. The AP retrieved in the table is based on a dataset of images from COCO-2017. Common Objects in Context is the industry standard benchmark for evaluating object detection models. It includes images of more than 80 regularly seen objects, with more than 121000 images with annotations for nearly 883000 objects. However, it is important to note that the performance of an object detection model still depends on the model design and training, instead of the dataset itself only.

Table 2.9: Summary of the reviewed YOLO variants (Hussain, 2023)

Variant	Framework	Backbone	AP (%)	Comments
V1	Darknet	Darknet-24	63.4	Only detect a maximum of two objects in the same grid.
V2	Darknet	Darknet-24	63.4	Introduced batch norm, k-means clustering for anchor boxes. Capable of detecting > 9000 categories.
V3	Darknet	Darknet-53	36.2	Utilized multi-scale predictions and spatial pyramid pooling leading to larger receptive field.
V4	Darknet	CSPDarknet-53	43.5	Presented bag-of-freebies including the use of CloU loss.
V5	PyTorch	Modified CSPv7	55.8	First variant based in PyTorch, making it available to a wider audience. Incorporated the anchor selection processes into the YOLO-v5 pipeline.
V6	PyTorch	EfficientRep	52.5	Focused on industrial settings, presented an anchor-free pipeline. Presented new loss determination mechanisms (VFL, DFL, and SloU/GIoU).
V7	PyTorch	RepConvN	56.8	Architectural introductions included E-ELAN for faster convergence along with a bag-of-freebies including RepConvN and reparameterization-planning.
V8	PyTorch	YOLO-v8	53.9	Anchor-free reducing the number of prediction boxes whilst speeding up non-maximum suppression. Pending paper for further architectural insights.

Besides COCO-2017, YOLOv8 is also tested by researchers at Roboflow using 100 sample datasets from their repository. The evaluation method, co-developed with Intel, is a benchmark that evaluates how well will YOLOv8 perform on the custom dataset. YOLOV8 is evaluated with its predecessors on RF100 and the YOLOv8 has an overall better mAP@0.5.

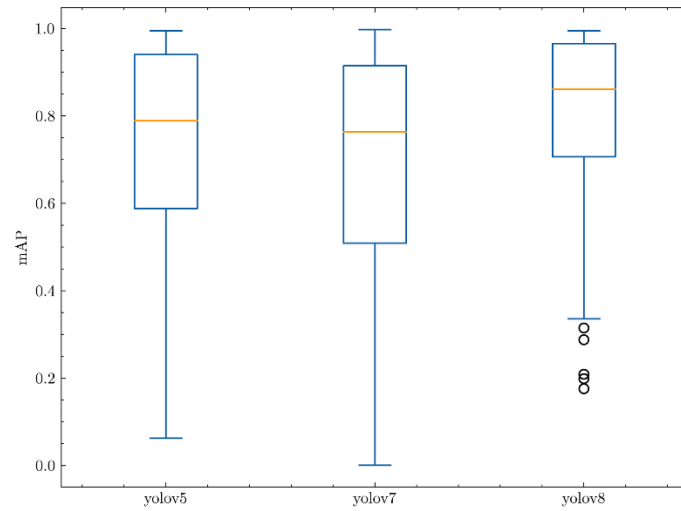


Figure 2.29: YOLOs mAP@0.5 against RF100 (Solawetz, 2024)

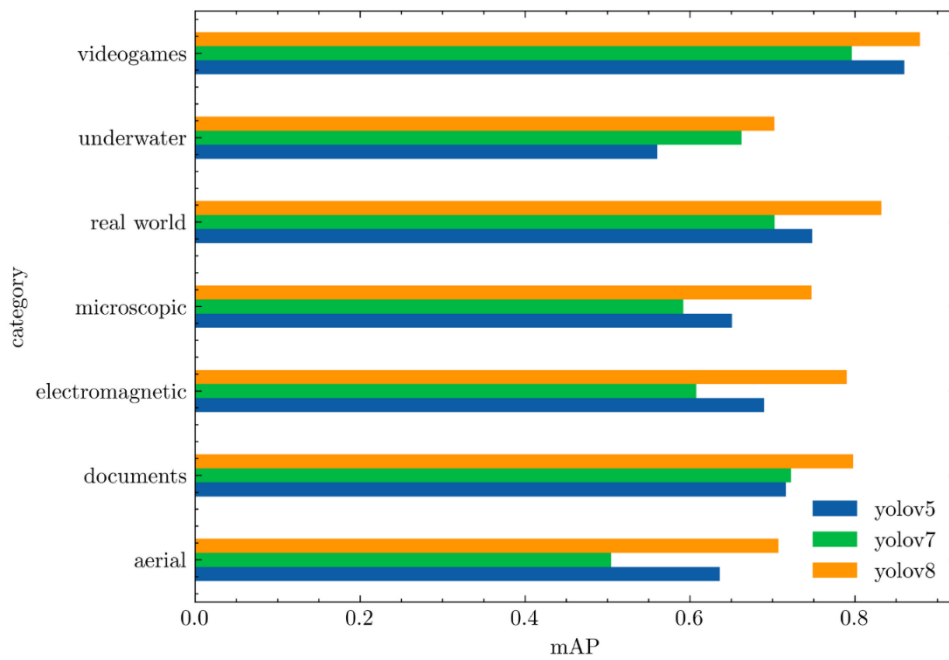


Figure 2.30: YOLOs average mAP@0.5 against RF100 categories (Solawetz, 2024)

Table 2.10: Summary of comparison among findings

No	Author	Title	Technique Used	Strength/ Limitations	Result	Remark
1	Zhang et al. (2020)	Unified detection method of aluminium profile surface defects: Common and rare defect categories	- Unified method with category representation network, attention module, and rare category transfer network	- Able to detect both common and rare defect	- The proposed method has a good balance between common (86%) and rare defect detection (50%) -ResNet performs the best in common defects (90%) but the worst in rare (15.9%) -The method performs badly in rare defects without an attention module (36.67%)	-Importance of attention module
2	Neuhauser et al. (2019)	Surface defect classification and detection on extruded aluminium	-VGG16 -GoogLeNet	- Increased accuracy with TL and DA	- Usage of transfer learning and data augmentation increases the mAP from 0.2 to 0.47 with lower iterations with a small dataset.	- Utilize transfer learning and data augmentation in training CNN.

		profiles using convolutional neural networks	-ResNet50			
3	Wang et al. (2022)	An intelligent method for detecting surface defects in aluminium profiles based on the improved YOLOV5 algorithm	-MS-YOLOv5	- High mAP -Low FPS	- MS-YOLOv5 has the highest mAP of 87.4% and the lowest FPS of 19.1 to fit real-time detection	
4	Tang et al. (2023)	An algorithm for Real-Time Aluminium Profile Surface Defects detection based on lightweight network structure	-YOLOv5s with attention mechanism and depth-separable convolution	-Improve mAP -reduce model size -low FPS	-Improves mAP by 1.76% -reduce model size by 52.08% -reach 17.4 FPS	

5	Yang et al. (2021)	Deep Learning based Steel Pipe Weld Defect Detection	-YOLOv5x	- High accuracy - Fast detection time	- High precision of 97.8% and mAP@0.5 of 98.7 -detection time of 0.12s per picture	
6	Ma et al. (2022)	A lightweight detector based on attention mechanism for aluminium strip surface defect detection	-Yolov4 with the backbone YOLO-DCSAM	-High mAP -Reduced model volume -Increased detection speed	- mAP of 96.28% - Reduce the model volume by 83.38% compared to YOLOv4 - Increase detection speed by 3 times	
7	Shi et al. (2022)	Research on Steel Surface Defect Detection Based on YOLOv5 with Attention Mechanism	- YOLOv5 with attention module and uses K-means algorithm	-Increased mAP -Decreased detection speed	-Higher mAP of 86.35% compared to the original YOLOv5 - Detection speed drops from 52 FPS to 45 FPS	

8	Zhao et al. (2023)	A modified YOLO for the detection of steel surface defects	- YOLOv5 with Res2Net as the backbone and double feature pyramid network in the neck and decoupled head - named as RDD-YOLO	- High mAP	-81.1% of mAP on NEU-DET, 4.3% higher than YOLOv5 -75.2% of mAP on GC10-DET, 5.8% higher than YOLOv5	
9	Li et al. (2018)	Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network	- Improved YOLO network with 27 convolution layers	-High mAP -High recall rate -High detection rate with low FPS	- 97.55% of mAP -95.86% of recall rate 99% detection rate with a speed of 83 FPS	
10	Wang et al. (2022)	Efficient Detection model of steel strip surface defects based on YOLO-V7	-YOLOv7 with de-weighted BiFPN, ECA attention mechanism, and IoU loss function	- High mAP -High detection speed	- mAP of 80.2% on GC10-DET and 81.9% on NEU-DET -55.6 FPS on NEU-DET	

11	Hussain (2023)	YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection	-YOLOv8 and all previous versions of YOLO	-High mAP	- YOLOv8 performs better compared to its predecessors on COCO-2017 and RF100.	

2.4 Conclusion

Based on the resources reviewed above, it can be concluded that traditional, it can be concluded that traditional object detection methods have been eliminated in recent years. Faster R-CNN as a representative of two-stage architecture may offer a high accuracy on aluminium surface defect detection. However, it requires more processing power compared to a single-stage approach which may lead to slower inference speeds and hinder real-time defect detection on a production line. Hence, the single-stage approach will be the focus of choices in this study. SSD as one of the single-stage object detection models offers a good balance between accuracy and speed, but the prioritization of speed may lead to a slight decrease in mAP on complex object detection tasks. Besides, SSD's architecture offers less flexibility for customization which could be a disadvantage for a project that requires fine-tuning for specific defect types.

According to the metal surface defect detection project recently, there were a lot of proposed solutions based on YOLO. Previous versions of YOLO like YOLOv3 and YOLOv5 have excelled in the previous works of object detection. As shown in the table above, YOLOv5 is one of the best object detection models that can be modified to fit different projects while maintaining high accuracy and speed. Its ability to leverage pre-trained weights on generic object detection tasks also allows fine-tuning for specific needs. Among all the versions in the YOLO, YOLOv8 will be selected as the method for this study. YOLOv5 as one of the most famous YOLOs, was introduced by Ultralytics, while YOLOv8 is another cutting-edge technology released by them in January 2023. With the excellent performance of YOLOv5 in the previous works, it is possible to say that YOLOv8 will perform better in object detection. Based on the statistics provided by Ultralytics, YOLOv8 performs better than YOLOv5 in COCO-2017 and RF100. Therefore, YOLOv8 will be chosen as the model for this study. Lastly, attention modules and data augmentation have increased the accuracy of object detection models in the previous works. Hence, this study aims to implement YOLOv8 with a high accuracy of 90% for categorizing and localizing the aluminium surface defects with the help of an attention module and data augmentation.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Workflow of model training

Figure 3.1 shows the workflow for training the YOLOv8 object detection model. The process starts with collecting data to build a sufficient dataset to train the model. After collecting enough data, the data will be pre-processed by resizing it to 640x640. Then, the defects on the image will be annotated to label out the defects using bounding boxes. The annotated data will undergo augmentation to increase the size of the dataset to train the model. It continues with the training, validation, and testing of the model. Depending on the performance of the model, improvements will be made to the model to improve its accuracy in detecting defects. After achieving the optimal performance, its performance will be evaluated from precision, recall, average precision, and mean average precision.

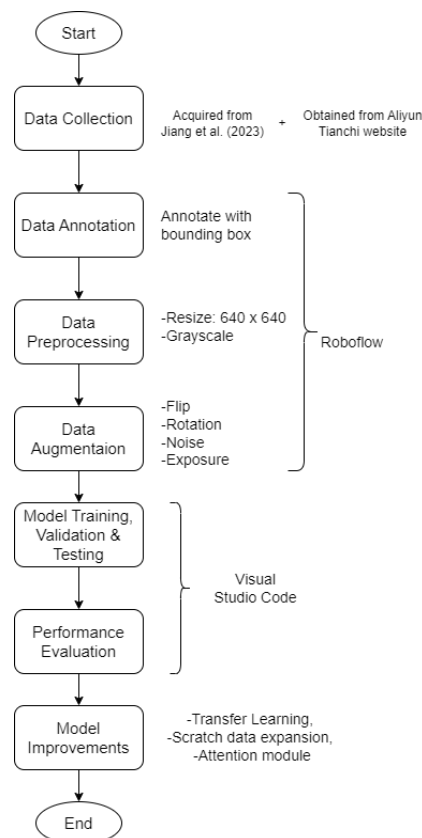


Figure 3.1: Workflow for training the YOLOv8 object detection model.

3.1.1 Data Collection

The dataset used in this study is originally an open-source dataset from the aluminium surface defect detection competition held by Aliyun Tianchi in 2018. However, the dataset from the competition does not provide labels for the defects on the aluminium surface defects which makes it difficult to identify and classify the defects. Mis location and misclassification of the defects during the manual labelling may result in the model learning incorrect patterns with wrong classes, leading to poor generalization. As a result, the model will produce more incorrect predictions, impacting its accuracy and reliability. Hence, to ensure high-quality and accurate labelling, the dataset used in this study was acquired upon request from the paper titled ‘MA-YOLO: A Method for Detecting Surface Defects of Aluminium Profiles With Attention Guidance’, authored by Jiang et al. (2023). This dataset includes 10 aluminium surface defects, including corner leak, crater, dirty point, jet, leakage, non-conductive, orange peel, paint bubble, parti-colour and scratch. There are a total of 2776 labelled defect images with a resolution of 2560 X 1920. Figure 3.2 shows some examples of the defect images.

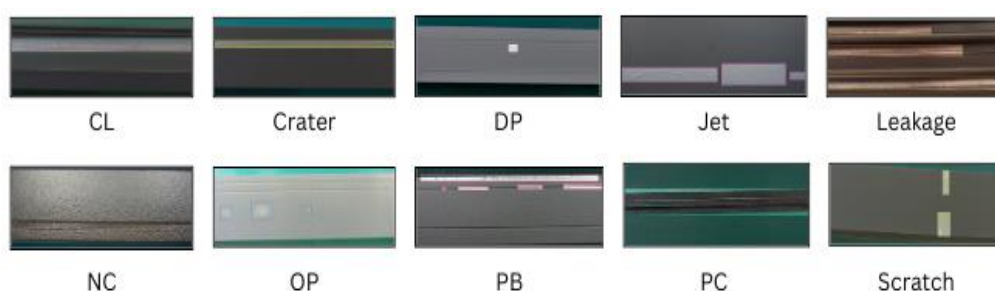


Figure 3.2: The labelled dataset from the journal

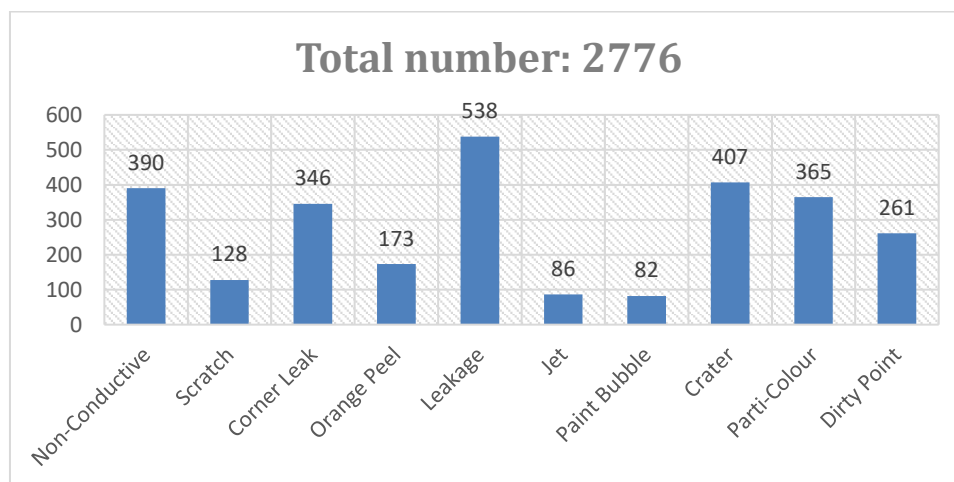


Figure 3.3: Number of images for each classes

3.1.2 Data Selection

Given the significant disparity in data distribution among the various defect classes, 'Jet' and 'Bubble' were severely underrepresented in the dataset. Despite efforts to augment and balance the data, these classes remained insufficient for the model to learn robust representations. To mitigate the negative impact of these underrepresented classes on the overall model performance, they are excluded from this study. This allows the model to focus on learning from classes with sufficient data, leading to improved accuracy and generalization capabilities on the majority of defects.

Despite the relatively low quantity of data for the 'Scratch' class, it was strategically included in the study due to its importance as the primary focus of defect detection in PMB Malaysia Sdn Bhd. To address the data imbalance and enhance model performance, the 'Scratch' class was expanded through the provision of additional aluminium-based samples. Considering that there is insufficient data evidence for 'Dirty Point' and also all the weak variables according to preliminary analysis, we opted not to include this class in the training set. In contrast, 'Orange Peel', even though it has a smaller data amount compared to 'Dirty Point', is projected to enhance the model's performance due to its higher quality and relevance to this study. Consequently, we have included 'Orange Peel' in our training data selection.

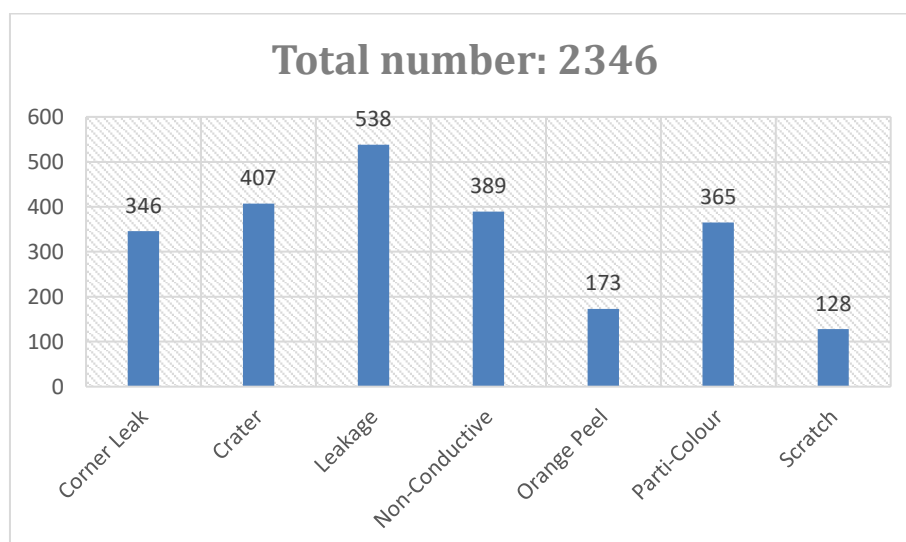


Figure 3.4: Number of images after data selection

3.1.3 Data Annotation

Data annotation is carried out before data augmentation because annotating the original images is a straightforward process while annotating an augmented image needs more consideration on the augmentation like cropping and noise which will be rather complex. Annotating original data can also avoid redundant labelling as the annotations for the original images will be applied to the corresponding augmented version when augmentation creates variations from the original image.

As most of the journal dataset has been labelled, the main focus of data annotation in this study will be labelling the scratch defects on the aluminium obtained from Aliyun Tianchi Aluminium Defect Detection Competition's official website. The defect images taken were uploaded to RoboFlow to carry out manual labelling with bounding boxes. The labelled images are also uploaded to perform preprocessing and augmentation.

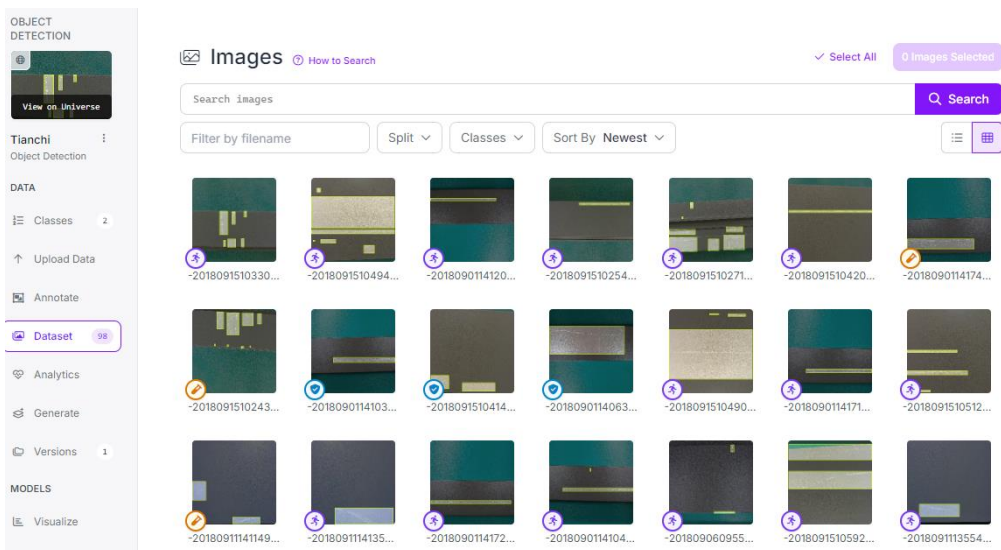


Figure 3.5: Images uploaded to Roboflow

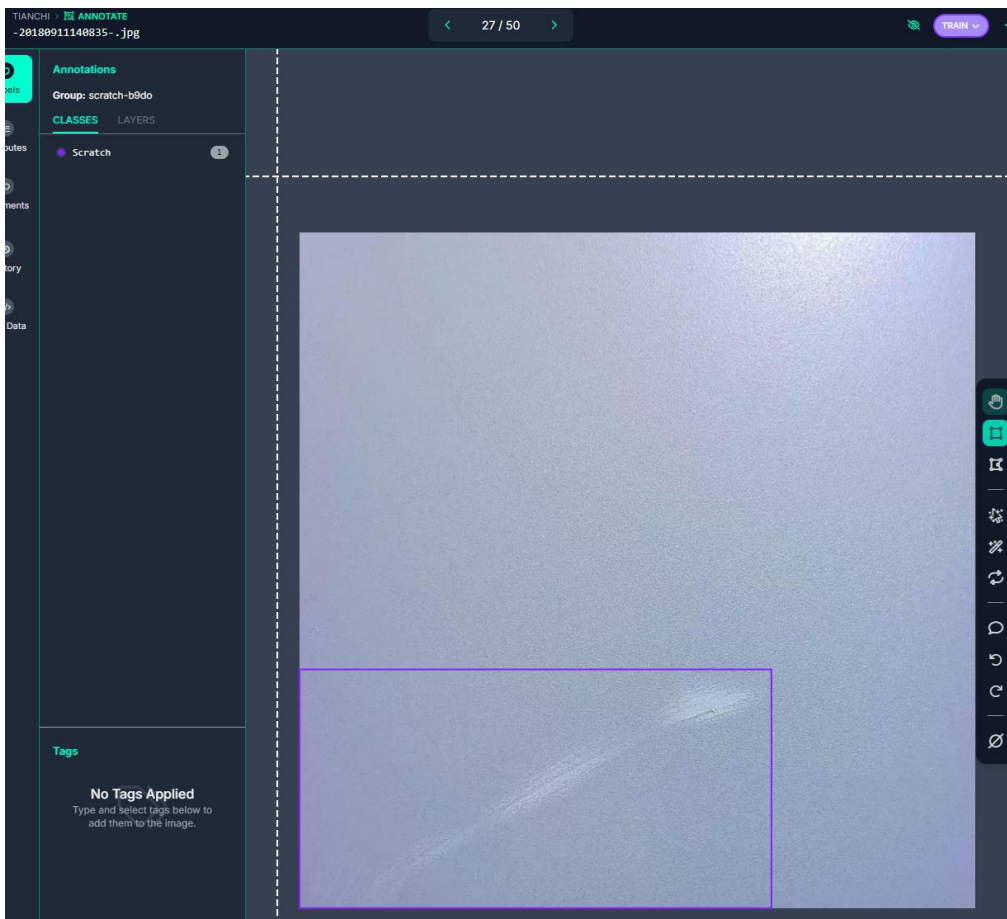


Figure 3.6: Annotating defect on image

3.1.4 Data preprocessing

Before the images performed data preprocessing, they were split into the ratio of 7:2:1 for training, validation and testing set. Therefore, a total of 2346 images in the dataset is split into 1648 for training, 477 for validation, and 221 for testing.



Figure 3.7: Dataset splitting

This is the step to ensure the dataset follows a standard format to maintain the consistency of the dataset for the training of models. Auto-orient is applied to all the data to strip the images of their EXIF data and standardize the pixel ordering. EXIF data is embedded metadata within the image that specifies the intended orientation for proper viewing. All the images are also resized to 640x640 in a standard size so that all the defects can be seen clearly by the models and prevent confusion for images with the same defects but having different defect sizes due to image size. Grayscale is also applied to all of the images. This is to decrease the computational costs and increase the training speed as the dataset is large. It can also make the model less sensitive to variations in lighting and colour which could be beneficial in real-world applications where images may have different lighting conditions. Lastly, grayscale can help the model to focus on the shape and texture of the defects instead of relying on the colour information.

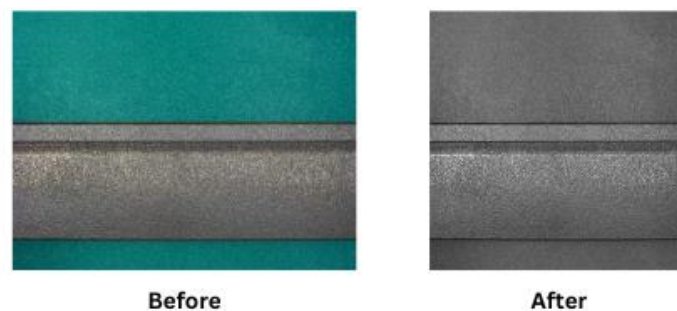


Figure 3.8: Comparison after data preprocessing

3.1.5 Data augmentation

As the current dataset is insufficient for the training of the model, data augmentation is used to expand the dataset to prevent overfitting which will decrease the model's ability of generalization and accuracy. The employed data augmentation techniques include horizontal and vertical flips, rotate, brightness and noise. Brightness augmentation is used to ensure the model can adapt to the different brightness when capturing the aluminium defects images. Noise is added to train our model so that it is more resilient to the camera artifacts. The reasons for the data augmentation methods used and the expected result are shown in the table below.

Table 3.1: Reason and expected result of data augmentation methods

Data Augmentation Methods	Reasons for Use	Expected Results
Flip Horizontal, Vertical	Defects may appear in various orientations in real-world applications. Flipping simulates these different orientations.	By using flipping transformations, the model learns to detect defects regardless of their orientation, improving detection robustness in diverse real-world scenarios.
Rotation Clockwise, Counter-Clockwise, Upside Down	Defects may not always appear in fixed positions. Rotating images introduces variability in defect positioning and orientation.	By using image rotation, the model can better generalize across different angles, improving its detection capabilities for defects appearing in different orientations.
Noise Between -15% and +15%	Noise can be introduced due to varying image capture environments and equipment quality, impacting the clarity of defect images.	By simulating noise, the model becomes more resilient to noisy data, enabling better performance when dealing with imperfect images in real-world environments.
Exposure Up to 0.22% of pixels	Varying lighting conditions during image capture can alter how defects appear on the surface.	By adjusting exposure levels, the model becomes more adaptable to different lighting conditions, improving defect detection under varying brightness and shadow levels.

However, augmented data will only be assigned to the training set, not the validation and testing set. This is because the training set requires a wider variety of data for the model training while augmented data improves its robustness and prevents overfitting. On the other hand., the purpose of testing and validation is to access the model performance on unseen data. If augmented data is included in the testing and validation set, the model would have seen variations of a similar picture during the training because of augmentation. Hence, the purpose of evaluating the performance on unseen data will be

meaningless. Besides, testing and validation also aim to evaluate how well the model detects real-world scenarios, while extensive modification with data augmentation would not provide an accurate picture of the model's ability to handle real-world scenarios.

Once the data augmentation is done, the dataset is ready to be downloaded. The dataset is downloaded in a zip file to a local computer in the YOLOv8 format to be compatible with the training of the YOLOv8 model. Folders of train, valid and test with images and labels included are accessible in the zip file. A yaml file for dataset configuration during the training of YOLOv8 is also included in the zip file. RoboFlow provides this convenience for users to train models without writing the dataset configuration yaml file. The configuration file contains the path for the training, validation and testing set of the dataset. It also contains the seven classes of defects to be detected and the name of each class.

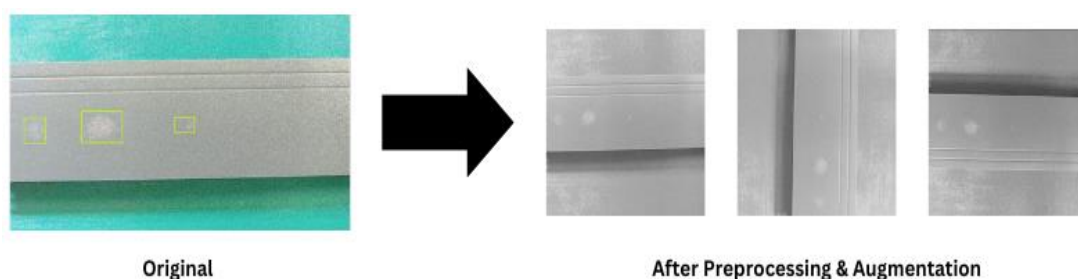


Figure 3.9: Augmented data with flip, rotate, brightness and noise

Download
×

Format

YOLOv8

TXT annotations and YAML config used with YOLOv8.

Download zip to computer
 Show download code

Also train a model for [Label Assist](#) with [Roboflow Train](#).
 2 Available Credits

Cancel

Continue

Figure 3.10: Download dataset in YOLOv8 format

3.1.6 Model training, validation, and testing

Before the training of YOLOv8, Anaconda is required to be installed to create a locally isolated environment to prevent conflicts between multiple projects' dependencies and libraries. Additionally, it simplifies the installation of libraries like PyTorch and CUDA for GPU support. The command `conda create -n yolov8 python=3.8` is used in the command prompt to create a conda environment with the name `yolov8` and Python3.8 installed in it. Then, `conda activate yolov8` is used to switch into the `yolov8` environment. PyTorch is then downloaded as it is the underlying framework for YOLOv8 and it provides core libraries for performing model training. Faster model training can also be achieved through GPU acceleration with CUDA with the help of PyTorch. Once the environment setup is complete, YOLOv8 can be cloned easily using `git clone` and install the requirements using `pip install ultralytics`.

The training of `yolov8` can be started with the command `yolo task=detect mode=train model=./yolov8n.yaml data=data.yaml epochs=100`. Table 3.2 shows the hyperparameter setting used in the model training. There are some key settings of the parameter in this model training such as the input image size is 640X640, batch size is 16, optimized by AdamW with weight decay of 0.0005, initial learning rate of 0.000909, and momentum of 0.9. The model is trained for 100 epochs to ensure its reliability and accuracy. The selection of AdamW as the optimizer is one of the options Ultralytics provides. In YOLOv8, besides using your optimizer, 'auto' can be selected and automatically choose an optimizer that suits your training based on the dataset size and batch size. Ultralytics claims that this function has gone through countless extensive experiments and shows improvement in results by using the chosen optimizer. Hence, the optimizer, initial learning rate and momentum is chosen by Ultralytics based on the dataset size.

During the training, its performance can be tracked on the validation set, measured by validation mAP or validation loss to improve the model's ability to handle unseen data. Once the validation metrics have reached their peak, it is better to stop the training, known as early stopping. This prevents the

model from overfitting to specific training data and performing worse on unseen data.

After all the training has been done, the model will run on a test set. The accuracy of the test set indicates how well the model performs on unseen data, which is crucial when it is deployed in production. This prevents the overfitting of the model to the validation data.

Table 3.2: Hyperparameter setting of model training

Parameters	Setup
Epochs	100
Momentum	0.9
Initial learning rate	0.000909
Final learning rate	0.01
Weight decay	0.0005
Batch size	16
Input image size	640 x 640
Optimizer	AdamW

3.1.7 Performance Evaluation

The key metrics for assessing a model's performance are precision, recall, F1 score and mean average precision. Precision measures the proportion of true positive out of all the predicted positives by the model. Recall, on the other hand, accesses the ratio of model's predicted positives to out all the actual positives available on the image.

Average precision represents the precision for a specific category in the mean average precision generalizes this metric across all categories in the dataset. The most commonly reported version, mAP at an IoU threshold of 0.5, is referred to as mAP@50.

3.1.8 Model Improvement

3.1.8.1 Dataset Expansion

After training the model, a thorough evaluation was performed to assess its accuracy through mean average precision. However, it is noticeable that the model is underperforming in detecting scratch defects due to insufficient data

compared to other defects. Hence, more scratch defect data is collected through an online source which is Aliyun Tianchi Aluminium Defect Detection Competition's official website. As the data sources are unannotated images, data annotation needs to be repeated in this stage. There is a total of 98 scratch images selected and uploaded to RoboFlow for data annotation and data augmentation.

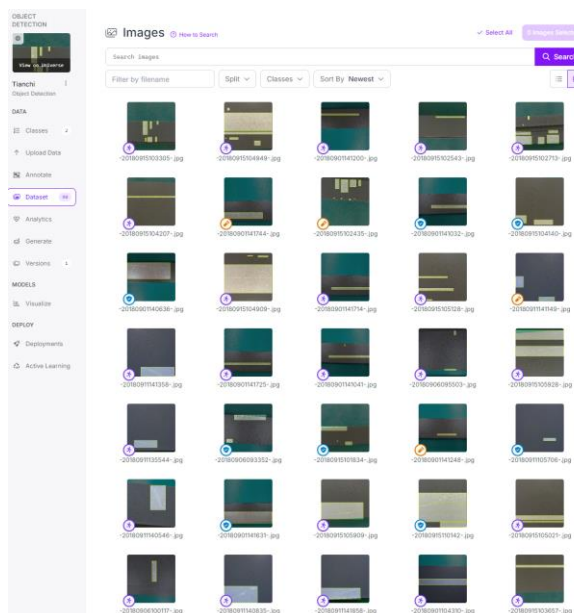


Figure 3.11: Images added to expand the dataset

3.1.8.2 Efficient Channel Attention

Efficient Channel Attention (ECA) is a mechanism designed to improve cross-channel interaction in neural networks. It introduces a novel approach to enhance feature expression by focusing on important channels without using dimensionality reduction. Instead of using complex 1D convolutions, ECA applies an adaptive convolution kernel to capture channel dependencies effectively. ECA aggregates features by performing Global Average Pooling (GAP) followed by channel interaction. Besides, the size of the adaptive convolution kernel is determined based on the channel dimension. It also uses a nonlinear mapping equation that adjusts the kernel size dynamically to suit the features of each channel (Chien et al., 2024).

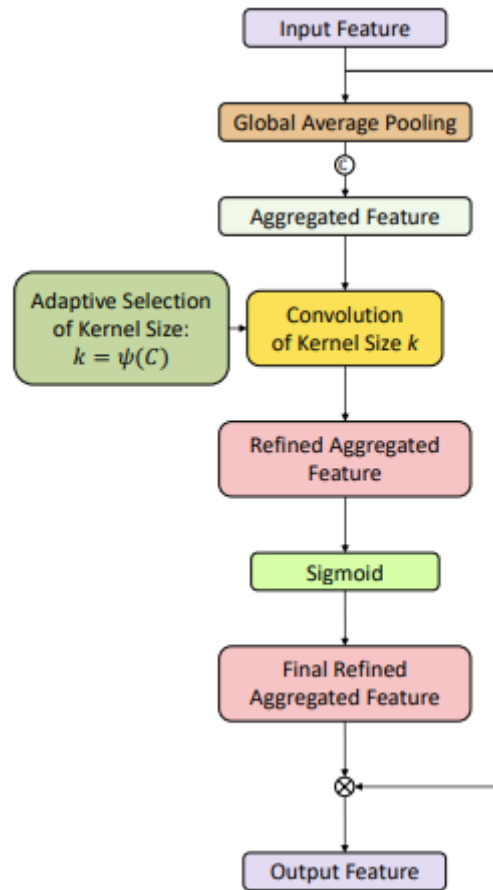


Figure 3.12: Structure of ECA

3.1.8.3 ResBlock + Convolutional Block Attention Module

Residual Convolutional Block Attention Module (ResCBAM) is a combination of the Residual Block and CBAM modules. Residual Block is a fundamental building block that consists of a sequence of convolutional layers with a skip connection. It uses the skip connection mechanism of residual blocks to improve the flow of information and solve the vanishing gradient problem. It includes a shortcut connection that directly maps the input to the output to ease the learning of the model as the network goes deeper and the gradients become smaller.

CBAM is a powerful attention mechanism designed to enhance feature representation in deep neural networks by selectively focusing on important information within feature maps. It effectively combines two key attention mechanisms which are channel attention and spatial attention, they work together to refine the network's focus on critical features.

Channel Attention operates by extracting vital information along the spatial dimensions and also enables one to identify the most important channels using global average pooling and global maximum pooling. The pooled information is then passed through a multi-layer perceptron (MLP) in activation by a sigmoid function for creating attention weights to enable the network to put a lot of weight on more important channels. These weights allow the network to prioritize certain channels, improving its ability to capture useful features.

On the other hand, spatial attention focuses on relevant spatial regions across channels. Similar to channel attention, it highlights important spatial regions within the feature maps with global average pooling and global maximum pooling. This is followed by a 7×7 convolution and a sigmoid activation to put weight on spatial attention to make sure the network looks at the most informative areas in feature maps.

By summing these two mechanisms with skip connections as in Residual blocks, ResCBAM amplifies the model capability to highlight channels and spatial regions that are informative while still constraining the gradients to propagate well, hence being a very strong tool in deep learning frameworks (Chien et al., 2024).bb

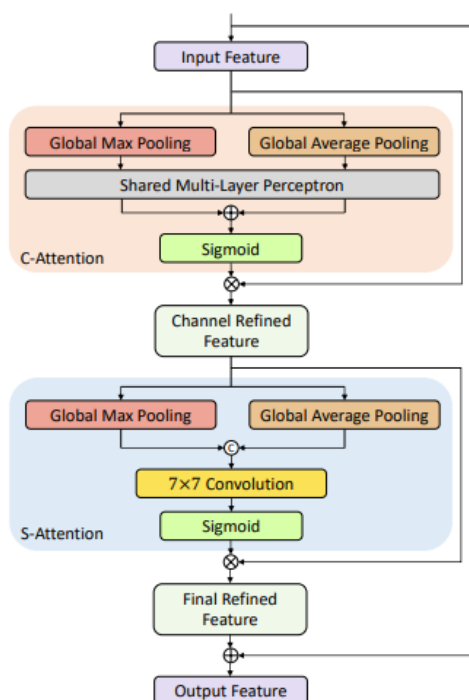


Figure 3.13: Structure of ResCBAM

3.1.8.4 Load a pre-trained model

As training from scratch using yolov8.yaml would require more data and time to achieve the same level of feature understanding, this could lead to a slower convergence and a higher risk of overfitting. Training a model from scratch without a large and diverse dataset would be challenging as it needs to learn the feature representations from the beginning. A training model with a pre-trained model would start the training with a strong generalization model that already understands how to detect key features in images. As a pre-trained model on aluminium defect detection is not found online, the COCO pre-trained model provided by Ultralytics will be used as a foundation for our model training. Despite the COCO dataset not being too related to aluminium defects, but it can help to model to identify common patterns and details on the aluminium surface, rather than a model trained from scratch without any prior knowledge. Fine-tuning this model for aluminium defects gives the advantage of faster training, better generalization, and improved accuracy, even if the target task is different from the original COCO training data.

3.2 Evaluation Metrics

The following section will explain the evaluation metric that will be used in this study to evaluate the model's performance. In the field of object detection, AP is used to assess the accuracy of a model in identifying objects. To understand the various forms of AP, it is crucial to first examine some fundamental concepts that are consistent across all of them:

Table 3.3: Fundamental concepts in object detection

True Positive (TP)	It can be assumed as a correct hit; it is a correct identification of an object, and the bounding box is placed exactly where the object is.
False Positive (FP)	It is a false alarm that the object detection mistakenly identified something else as an object and placed a bounding box around it, but there was no object there.

False Negative (FN)	A false negative is a miss where the object detection completely missed an object that was there.
True Negative (TN)	True negative refers to the bounding boxes that should be detected. However, the concept of TN is normally not considered in object detection because there are endless empty spaces that the system should not identify as ‘not having an object’ within an image.

3.2.1 IoU

Even with the definitions above, a clear establishment is required to identify whether the detection is correct or incorrect. Intersection over Union (IoU) is the most common way of judging the accuracy of object detection. It is a general way to compare the similarity of two sets based on the Jaccard Index. Assume the actual location of the object is marked by a box called the ground-truth bounding box, B_{gt} while the detected area by the detection model is called the predicted bounding box, B_p . IoU measures how well the predicted bounding box overlaps with the ground-truth bounding box and divides it with the union of both boxes.

$$IoU = \frac{B_{gt} \cap B_p}{B_{gt} \cup B_p}$$

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{img}}{\text{img}}$$

Figure 3. 14: Illustration of Intersection over Union (IoU)

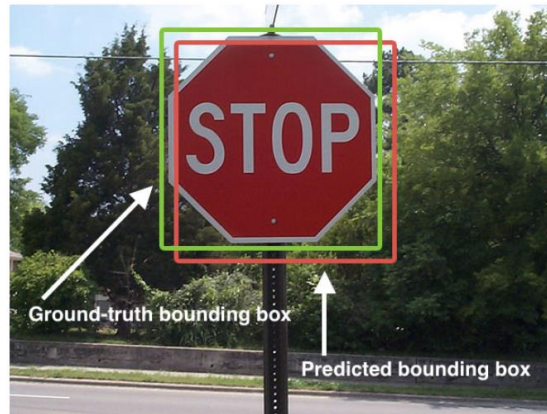


Figure 3.15: Example of IOU

An IoU of 1 means that the predicted box perfectly overlaps the actual object's location while 0 means both boxes do not overlap at all. Generally, the closer the IoU to 1, the higher the accuracy of the model as its predicted boxes match the ground-truth bounding boxes. Only boxes from the same class can be compared with IoU as object detection models also perform the classification of objects. An IoU threshold is set to determine the strictness of the accuracy of the detections. A threshold closer to 1 is tough as it requires nearly perfect overlaps while a lower threshold is more lenient, considering even small overlaps as detection.

3.2.2 Precision and recall

As mentioned above, TN is not considered a metric of performance of object detection due to the countless empty spaces. The evaluation of object detection primarily relies on the concepts of precision and recall. Precision measures the accuracy of the model in identifying the actual objects out of all of the predicted bounding boxes. It is a percentage of predictions that are truly correct detections. Precision can be obtained with the formula below:

$$Precision = \frac{TP}{TP + TN} = \frac{TP}{All\ detections}$$

On the other hand, recall measures the ability of the model to find all the actual objects in the image without missing any. It is the ratio of correctly

detected positive samples to the total number of positive samples present in the ground truth data.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{All ground truths}}$$

A model with a high threshold will have high precision but low recall because it may miss many actual objects. Meanwhile, if the threshold is low, the objects will have high recall as it finds more objects, but the precision will be low as its findings will include a lot of false alarms. To solve this, plotting a precision-recall curve for different thresholds can see the trade-off visually. Ideally, the curve should stay high in both precision and recall as the confidence threshold changes. Therefore, a high area under the curve (AUC) indicates a good balance between precision and recall in different confidence thresholds. However, the precision-recall curve often possesses a zig-zag pattern, making it difficult to calculate the AUC. Techniques like 11-point interpolation and all-point interpolation are used to remove the zigzag pattern to ease the calculation of AUC. The 11-point interpolation is a method that uses 11 evenly spaced standard recall levels while the all-point interpolation utilizes every recall point available.

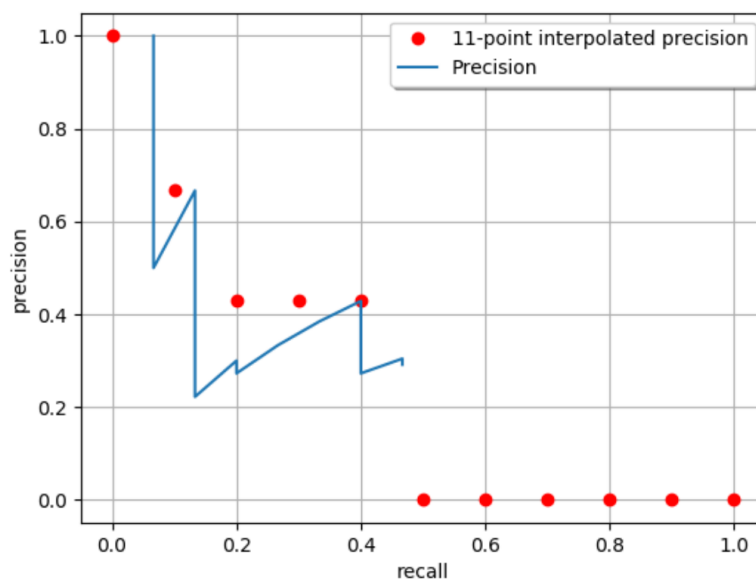


Figure 3.16: Precision-recall curve using the 11-point interpolation approach

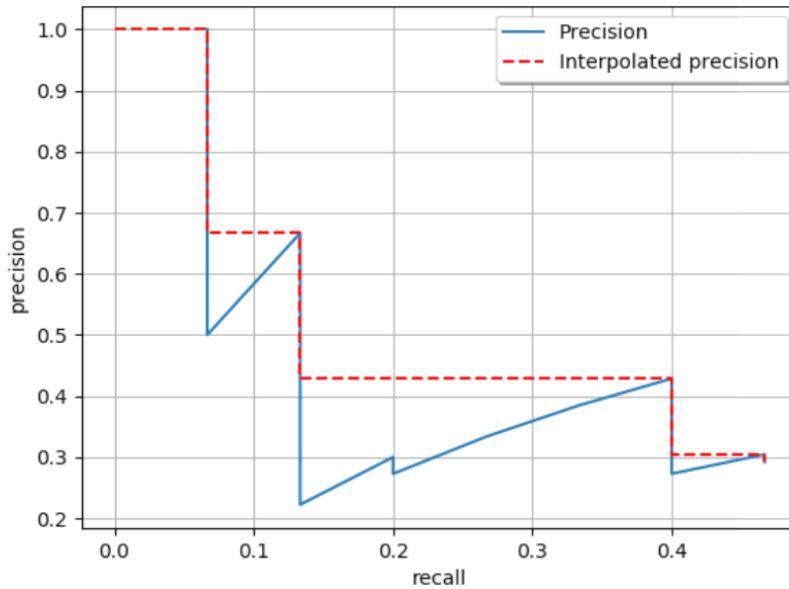


Figure 3.17: Precision-recall curve using the all-point interpolation approach

3.2.3 mAP

As mentioned earlier, an object detector will provide bounding boxes, object classes, and confidence scores. Only the confidence scores that are higher than the confidence threshold can be considered positive detections and contribute to the calculations of precision and recall. Meanwhile, detections with a confidence score lower than the confidence threshold will be considered negative detections and will not be included in the calculation. After that, all the positive detections will be used to calculate the precision and recall, then used in plotting the precision-recall curve. After the preprocessing to eliminate the zigzag pattern of the curve, the AP is obtained based on the area under the curve.

Besides AUC, the F1 score is also one of the approaches to combine precision and recall into a single metric. The higher the F1 score, the higher the precision and the recall. However, AP is calculated for each class as there will be many object classes to be identified in the image. Hence, it is important to obtain the overall precision of the object detection for all classes. This can be obtained by calculating the mAP of the model, which is the average value of the AP for all classes. The F1 score and mAP can be calculated using the formula below where AP_k is the AP of the class k and n is the number of classes.

$$F1\ Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k$$

3.3 Gantt Chart

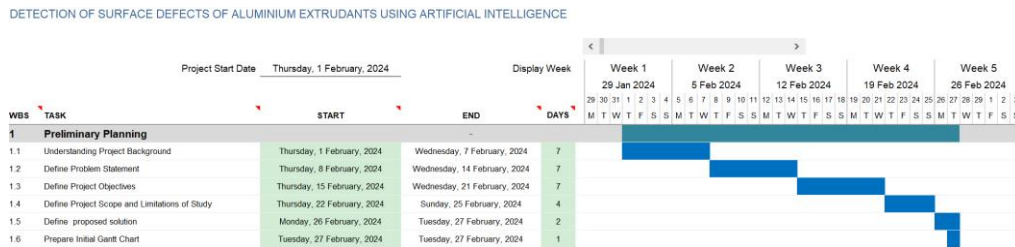


Figure 3. 18: Gantt chart for Preliminary Planning from 1/2/2024 to 27/2/2024

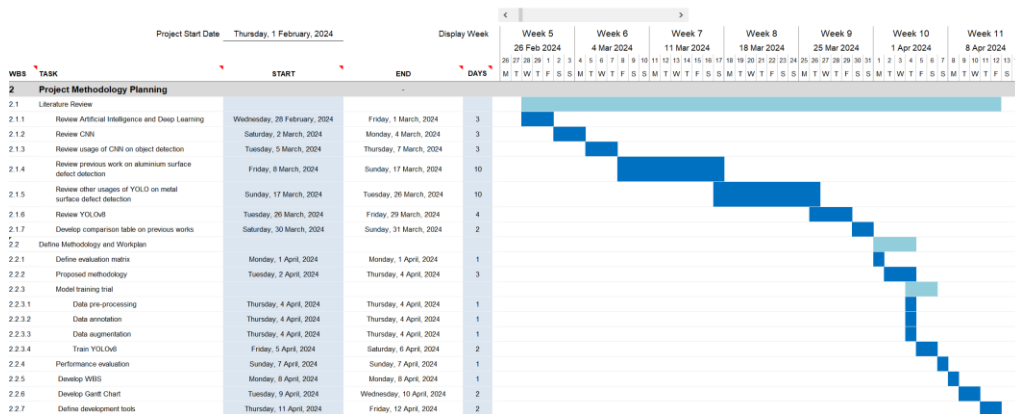


Figure 3. 19: Gantt chart for Project Methodology Planning from 28/2/2024 to 12/4/2024

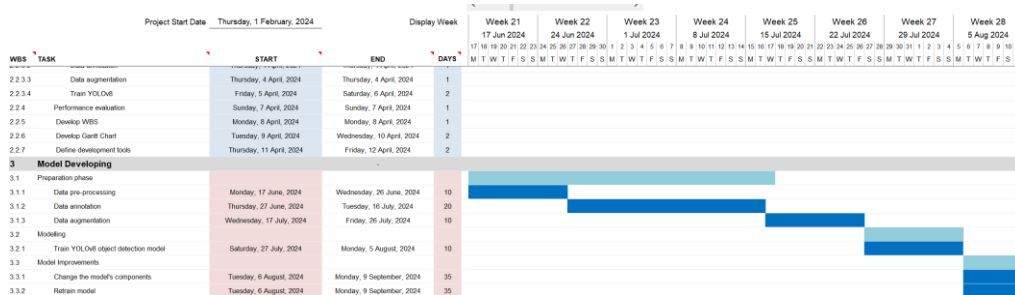


Figure 3. 20: Gantt chart for Model Developing from 17/6/2024 to 11/8/2024

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Comparison among YOLOv8 Predecessors and YOLOv8

Before deep into the focus of this study, the predecessors of YOLOv8 are used to compare with YOLOv8 on the performance of the model on the aluminium surface defects dataset. YOLOv5 and YOLOv7 are used in this study to show the performance of YOLOv8 is better than its predecessors. As shown in the table below, YOLOv8 has a significantly better overall performance compared to YOLOv7. While YOLOv5 is also produced by Ultralytics, it has a competitive performance with YOLOv8n. Despite having a lower precision than YOLOv5, YOLOv8 outperforms YOLOv5 in other aspects which are recall, F1-score and the mean average precision. Hence, this proves that YOLOv8 is capable in this aluminium defect detection study.

Table 4.1: Comparison between YOLOv8 Predecessors

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95
YOLOv5n	79.1	68.4	73.4	73.8	54.9
YOLOv7	65.5	67.7	66.6	65.9	48.3
YOLOv8n	74.4	76.1	75.2	75.9	56.8

4.2 YOLOv8 Variants Selection

As shown in the table above, YOLOv8m has the best performance with mAP@0.5 of 76.8% and mAP@0.5:0.95 of 57.5%. Despite having the highest accuracy among the 3 variants, YOLOv8m is not considered a good model for this study as it does not have a huge leap of improvement compared to the other 2 models. Considering the time needed to train YOLOv8m is significantly more than the other 2 models, this model is not suitable for real-time detection. The same goes for YOLOv8s, it does not achieve a higher accuracy with the trade-offs of efficiency. Instead, YOLOv8s and YOLOv8m both achieve lower accuracy on certain classes of defects compared to YOLOv8n. Therefore, considering the limited computational resources available and the real-time

requirement, YOLOv8n is selected as the model for further improvement in this study as it offers a good balance between accuracy and efficiency.

Table 4.2: Comparison between YOLOv8 variants

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95	Params (M)
YOLOv8n	73.3	75.6	74.4	75.4	56.2	3.2
YOLOv8s	74.4	76.1	75.2	75.9	56.8	11.2
YOLOv8m	75	76.4	75.7	76.8	57.5	25.9

4.3 Comparison Between Before and After Data Selection & Data Augmentation

According to Table 4.5, the overall performance of the model has a significant improvement after applying data selection and data augmentation. The precision increases from 73.3% to 86.5%, indicating a reduction in false positives. Recall increases from 75.6% to 88.7%, demonstrating an improved ability to detect all relevant objects, leading to an increase in F1-score to 87.6, showing an improvement in balancing precision and recall. The mean average precision also increases significantly for both mAP@0.5 and mAP@0.5:0.95.

Removing defect classes with insufficient data would definitely increase the overall accuracy of the model as the classes with low precision are removed. However, that does not mean that removing underrepresented classes is just to achieve a higher overall mean average precision. The model is able to allocate more weight and attention to the other majority classes. This allows the model can generalize better to the remaining classes. This can be seen by comparing the average precision of classes such as scratches, orange peel and crater after data selection. Moreover, this also mitigates the risk of overfitting underrepresented classes as their data is insufficient. In real-world use cases, if an underrepresented class is rare or not critical, removing it can simplify the model's deployment and reduce potential false positives or confusion. It helps avoid unnecessary predictions in cases where the underrepresented class is not expected to appear often.

This result is achieved not just because of data selection, but also thanks to data augmentation. Augmenting data helps the model learn from more diverse examples, improving its ability to generalize to unseen data without collecting new data which is time-consuming. The model learns to be invariant

to small changes, such as rotations, flips, or brightness variations. The ‘Scratch’ class benefits the most from data augmentation as it is one of the underrepresented classes. However, it was not removed because it is one of the most common defects found in the manufacturing process. Its accuracy improves from 33.1% to 51.8%.

Table 4.3: Performance of YOLOv8 in each class with the original dataset

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95
all	274	329	0.733	0.756	0.754	0.562
Non-Conductive	39	46	0.787	0.826	0.915	0.783
Scratch	11	17	0.382	0.353	0.331	0.127
Corner Leak	35	35	1	1	0.995	0.795
Orange Peel	19	19	0.923	1	0.982	0.889
Leakage	54	66	0.872	0.924	0.929	0.724
Jet	10	13	0.492	0.447	0.44	0.203
Paint Bubble	8	10	0.324	0.346	0.287	0.089
Crater	40	40	0.755	0.955	0.932	0.518
Parti-Colour	36	36	0.982	1	0.995	0.978
Dirty Point	24	39	0.839	0.718	0.734	0.499

Table 4.4: Performance of YOLOv8n in each class after data selection and augmentation

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95
all	221	240	0.865	0.887	0.904	0.737
Non-Conductive	38	41	0.919	0.829	0.902	0.775
Scratch	9	17	0.7	0.412	0.518	0.227
Corner Leak	36	36	1	1	0.995	0.83
Orange Peel	17	17	0.81	1	0.99	0.909
Leakage	58	58	0.903	0.966	0.972	0.784
Crater	34	34	0.723	1	0.953	0.66
Parti-Colour	37	37	1	1	0.995	0.974

Table 4.5: Comparison of performance before and after data augmentation

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95
YOLOv8n (Before Data Selection & Augmentation)	73.3	75.6	74.4	75.4	56.2
YOLOv8n (After Data Selection & Augmentation)	86.5	88.7	87.6	90.4	73.7

4.4 Impact on loading pre-trained model

As mentioned above in Chapter 3, the training of YOLOv8 models allows the loading of a pre-trained model for faster training and improved model accuracy. In this study, yolov8n.pt is used as a pre-trained model to achieve better model accuracy in detecting aluminium defects. The YOLOv8.pt is a pre-trained model provided by Ultralytics, typically pre-trained on the COCO dataset. Transfer learning is applied by taking the pre-trained model and fine-tuning it accordingly to the aluminium surface defect dataset. It helps the new model to adapt quickly to the new detection task as it already has a strong understanding of general features and does not need to learn everything from scratch.

Freezing of layers could be used with transfer learning to speed up the training process by freezing some of the layers of the model, so there will be only a few layers that will be learning the new features. This action would speed up the training process significantly but will have a lower accuracy compared to non-freezing as a trade-off. However, freezing layers is not considered in this study because YOLOv8n will be used as the detection model, so the training speed will be fast and computational resources will be lower. The overall performance can be optimized by allowing all the weights of the layers to get updated to the new features without freezing any layers.

Table 4.6: Performance on each class using transfer learning

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95
all	221	240	0.945	0.882	0.927	0.785
Non-Conductive	38	41	0.895	0.829	0.901	0.807
Scratch	9	17	0.818	0.529	0.71	0.408
Corner Leak	36	36	1	1	0.995	0.849
Orange Peel	17	17	1	1	0.995	0.919
Leakage	58	58	0.933	0.966	0.97	0.811
Crater	34	34	0.967	0.853	0.924	0.727
Parti-Colour	37	37	1	1	0.995	0.975

Table 4.7: Comparison between with and without transfer learning on the augmented dataset

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95
YOLOv8n (without transfer learning)	86.5	88.7	87.6	90.4	73.7
YOLOv8n (with transfer learning)	94.5	88.2	91.2	92.7	78.5

Table 4.7 shows the improvement of the model in every aspect except recall after applying the pre-trained model. Figure 4.1 shows the PR curves of

the two models. It can be observed that the pre-trained model improves its mAP@50 by 2.3%. The area enclosed by PR curves is mAP whereby YOLOv8n with transfer learning has a larger area compared to YOLOv8n without transfer learning. Figure 4.2 shows the F1 score curves of both models. The F1 curve considers both precision and recall, whereby the higher the curve, the better the performance. The curves of all classes have improved, and the F1 score improved by 3%. It can be seen easily and concluded that the pre-trained model outperforms the baseline model in the aluminium surface defect detection task.

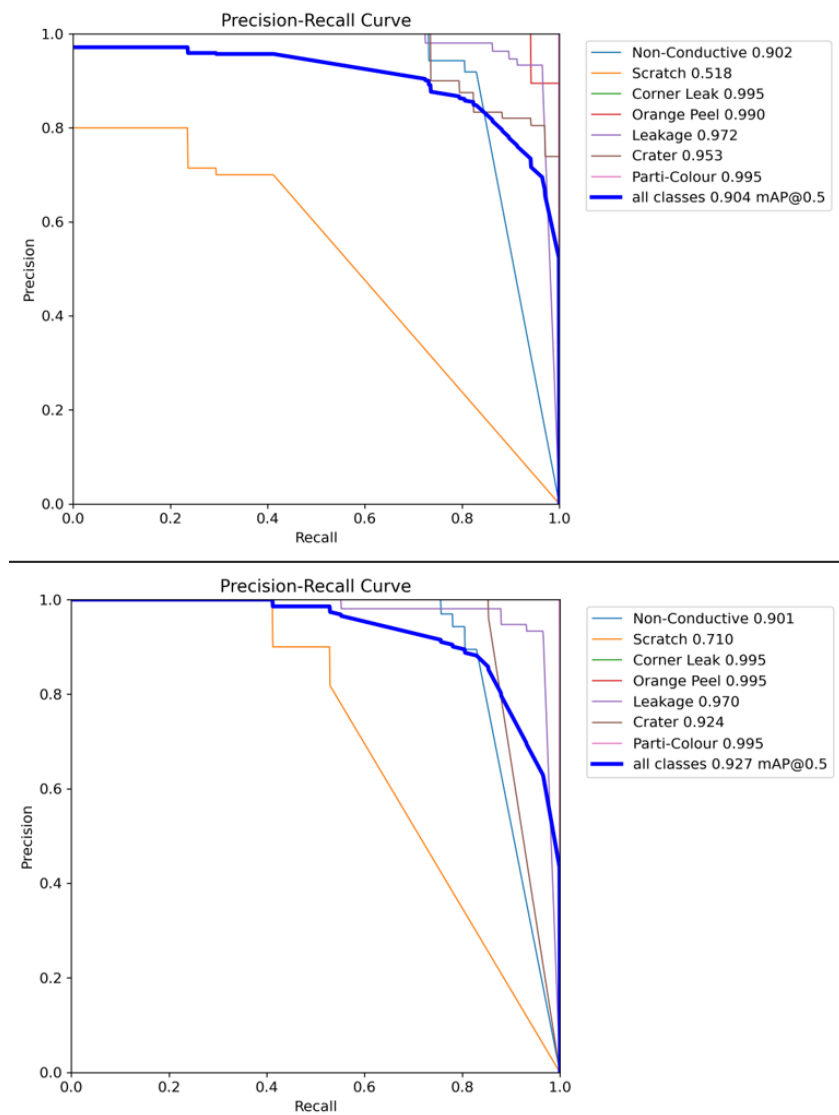


Figure 4.1: Comparison of PR curves for YOLOv8n without transfer learning (left) and YOLOv8n with transfer learning (right)

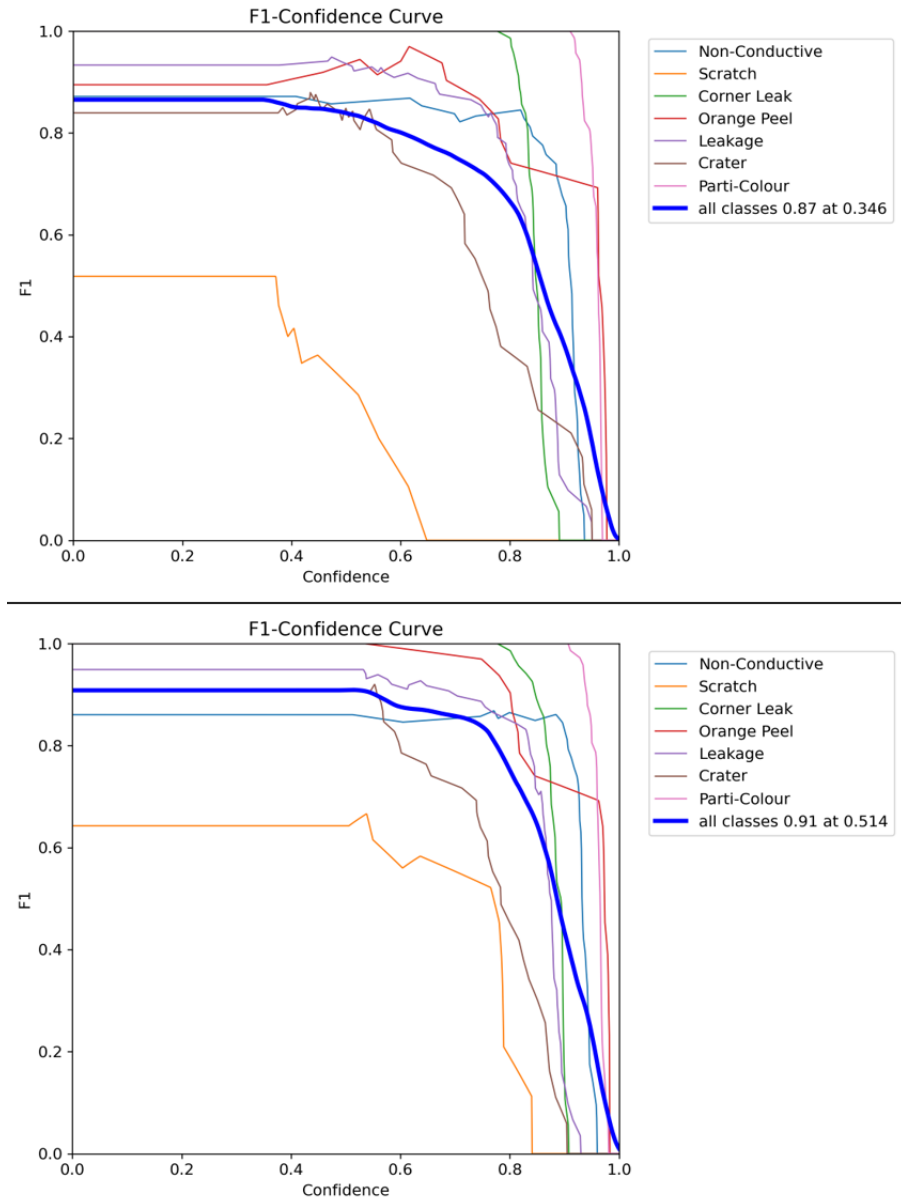


Figure 4.2: Comparison of F1 score curves for YOLOv8n without transfer learning (left) and YOLOv8n with transfer learning (right)

4.5 Impact of Scratch Class Expansion

It is noticeable that scratch remains less accurate than other classes after a few improvements. To solve this problem from its root cause, 98 more images were selected and added to the scratch classes from an online source. After data labelling and augmentation, 210 images were added to the existing training set, 18 images were added to the validation set, and 10 images were added to the testing set. An expanded dataset with diverse examples allows the model to learn

more variations of scratches, preventing overfitting specific examples due to insufficient data.

Table 4.8: Performance on each class with scratch class expansion

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95
all	231	267	0.926	0.907	0.943	0.797
Non-Conductive	38	41	0.862	0.829	0.888	0.792
Scratch	19	44	0.848	0.634	0.786	0.517
Corner Leak	36	36	1	1	0.995	0.845
Orange Peel	17	17	0.916	1	0.995	0.917
Leakage	50	58	0.948	0.944	0.969	0.838
Crater	34	34	0.908	0.941	0.975	0.69
Parti-Colour	37	37	1	1	0.995	0.98

As shown in Table 4.9, the overall performance of the model increases except for precision and mAP@0.5:0.95 with the expansion of the scratch class. This indicates that the model performs better overall after the expansion of the scratch dataset. It is important to notice the accuracy of scratch detection also increases with the help of more training data. Scratch is given attention as it is one of the main focuses in this study and one of the common defects in aluminium surfaces nowadays. It is noticeable that the mAP@0.5 for the scratch only increases from 71% to 77.7% while the mAP@0.5:0.95 for scratch improves as much as 7%. This proves that the expansion of the scratch dataset did help in improving the accuracy of the model in detecting scratches.

Table 4. 9: Comparison between before and after scratch class expansion on the augmented dataset

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95	mAP@0.5 of scratch	mAP@0.5:0.95 of scratch
YOLOv8n (transfer learning)	94.5	88.2	91.2	92.7	78.5	<u>71</u>	<u>40.8</u>

YOLOv8n (transfer learning + scratch class expansion)	92.6	90.7	91.5	94.3	79.7	78.6	51.7
--	------	------	------	------	------	-------------	-------------

4.6 Model Improvement with Attention Module

There are two attention modules used in this study which are ECA and ResCBAM. YOLOv8n consistently outperforms YOLOv8n with ECA across all metrics. This shows that the addition of the ECA attention mechanism did not provide a clear performance boost in this case. In fact, it slightly reduced the model's effectiveness in detecting aluminium defects, which might indicate that ECA's channel-based attention is not well-suited for this specific task, or that the baseline YOLOv8n architecture is already highly optimized for this dataset.

Table 4.10: Performance of YOLOv8n with ECA

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95
all	231	267	0.871	0.888	0.911	0.751
Non-Conductive	38	41	0.819	0.756	0.868	0.787
Scratch	19	44	0.692	0.523	0.605	0.336
Corner Leak	36	36	1	1	0.995	0.822
Orange Peel	17	17	0.894	0.991	0.976	0.886
Leakage	50	58	0.908	0.966	0.969	0.776
Crater	34	34	0.788	0.983	0.968	0.672
Parti-Colour	37	37	1	1	0.995	0.976

YOLOv8n also performs better across all metrics compared to YOLOv8n with ResCBAM. The addition of ResCBAM results in a slight decrease in precision, recall, F1-score, and mAP metrics. This suggests that the attention mechanism introduced by ResCBAM may not provide a clear advantage for this specific aluminium defect detection task. YOLOv8n without ResCBAM appears to maintain stronger detection and localization capabilities, indicating that the base YOLOv8n model is already highly optimized for this

dataset and might not benefit from the additional complexity introduced by ResCBAM.

Table 4.11: Performance of YOLOv8 with ResCBAM

Class	Images	Instances	Box (P)	R	mAP50	mAP50-95
all	231	267	0.915	0.891	0.933	0.752
Non-Conductive	38	41	0.895	0.835	0.934	0.795
Scratch	19	44	0.821	0.477	0.664	0.342
Corner Leak	36	36	1	1	0.995	0.849
Orange Peel	17	17	0.944	1	0.989	0.873
Leakage	50	58	0.899	0.966	0.979	0.768
Crater	34	34	0.845	0.961	0.974	0.66
Parti-Colour	37	37	1	1	0.995	0.974

Table 4. 12: Comparison between two attention modules and YOLOv8n (with data augmentation + scratch class expansion)

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95
YOLOv8n	92.6	90.7	91.6	94.3	79.7
YOLOv8n + ECA	87.1	88.8	87.9	91.1	75.1
YOLOv8n + ResCBAM	91.5	89.1	90.3	93.3	75.2

4.7 Discussions

Table 4.13 shows the improvement in the performance of YOLOv8n after the step-by-step enhancement used. It is clearly shown that the model has improved a lot compared to the original model from scratch. Improvements taken include data preprocessing and data augmentation, transfer learning, dataset expansion, and application of attention module. Except for the attention module, other steps have contributed to the improvement of the models. The model has improved its mAP from 75.4% to 94.3%, an improvement of 18.9% while its mAP@0.5:0.95 has also improved as much as 23.5%. Given the inference time of the model is 9.1ms, it can be calculated that the model has an FPS of 109

which fulfils the needs of the industry requirements. This shows that the model has a balance between high accuracy and a fast detection speed.

Table 4.13: Effect of each step-by-step improvement on YOLOv8n

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95
YOLOv8n	73.3	75.6	74.4	75.4	56.2
YOLOv8n with data augmentation	86.5	88.7	87.6	90.4	73.7
YOLOv8n (data augmentation+ transfer learning)	94.5	88.2	91.2	92.7	78.5
YOLOv8n (data augmentation + transfer learning + scratch class expansion)	92.6	90.7	91.6	94.3	79.7
YOLOv8n + ECA (data augmentation + transfer learning + scratch class expansion)	87.1	88.8	87.9	91.1	75.1
YOLOv8n + ResCBAM (data augmentation + transfer learning + scratch class expansion)	91.5	89.1	90.3	93.3	75.2

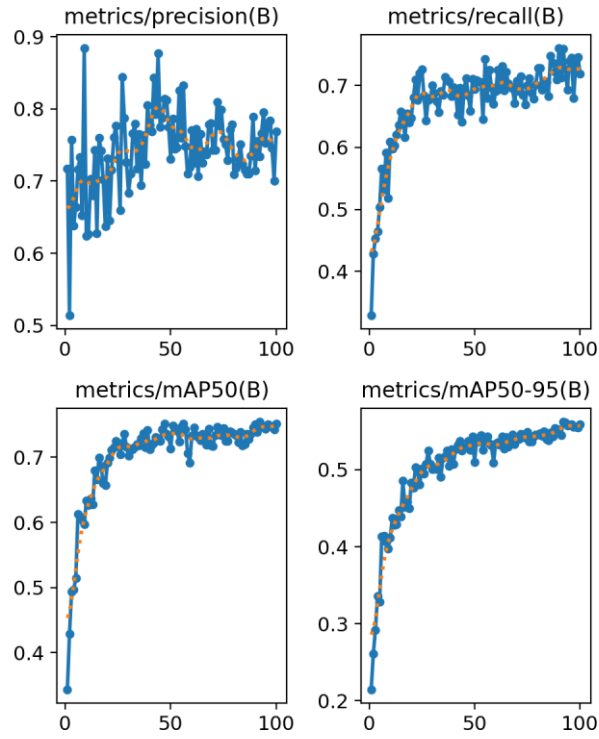


Figure 4.3: Evaluation metric of baseline model with original dataset

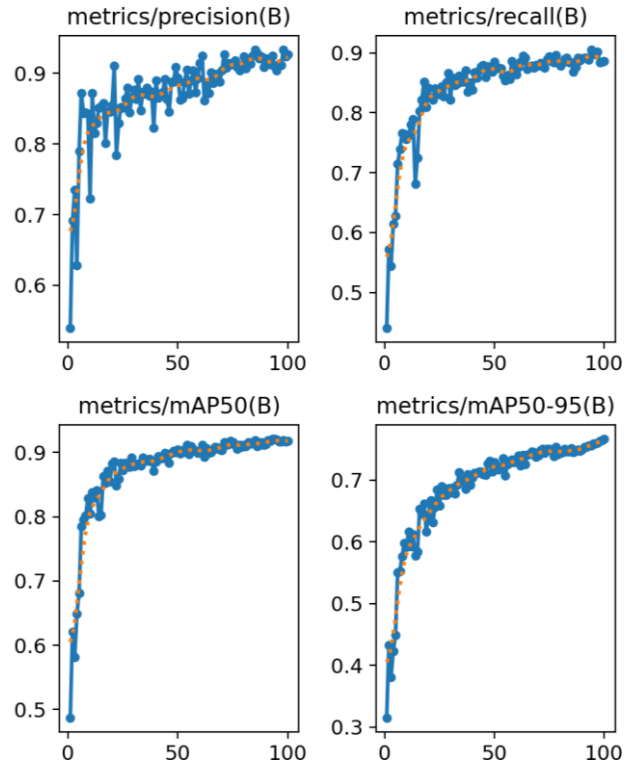


Figure 4.4: Evaluation metrics of the improved model with improved dataset

Based on the two graphs above, it can be observed that the precision and recall have a more stable increment after model improvements which are data augmentation, scratch class expansion and transfer learning. Besides, the second graph has a steadier decrement in both training and validation loss compared to the first graph. Lastly is the obvious increment in value for all aspects after model improvement in the second graph.

Figure 4.5 shows the normalized confusion matrix of the YOLOv8n with improvements including data augmentation, transfer learning and scratch class expansion. It shows most classes such as corner leak, orange peel, crater and parti-colour are perfectly predicted with a value of 1.0. Non-conductive and scratch have relatively lower results as there are 12% and 18% of false negatives for non-conductive and scratch respectively. At the same time, there are 44% of false positives detections on scratch and 25% on crater. Meanwhile, non-conductive and leakage each have 12% false positive detections. According to this result, it shows that this model performs worse in non-conductive and scratch compared to other classes.

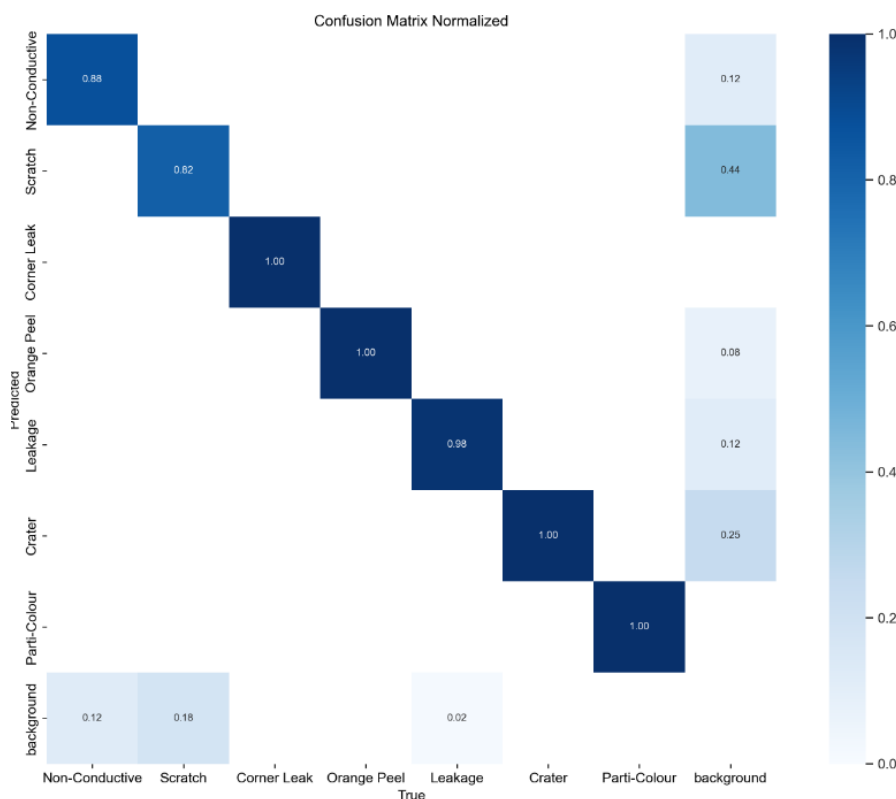


Figure 4.5: Normalized confusion matrix of YOLOv8n with all the improvements

4.8.1 Failure Case Analysis

Despite the improved YOLOv8 has shown great performance in this study, it still experiences several missed and false detections in certain complex scenarios. Some examples of false detection are shown in the figures below. Figure 4.6 shows a missed detection of scratch by the model. The missed detection in this example may be due to the model detecting it as a reflection of light which caused the model to overlook the scratch defect. Figure 4.7 is also an example of the model's missed detection on a shallow and long scratch. The reason may be due to the scratch is not obvious, hence the model overlooked it.

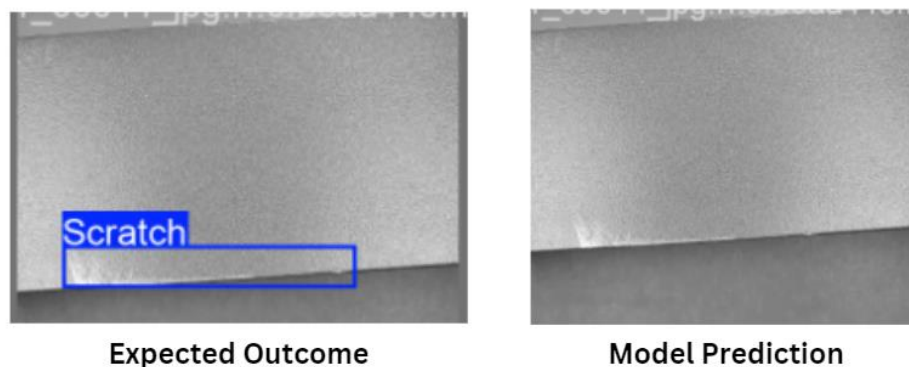


Figure 4.6: False negative of scratch

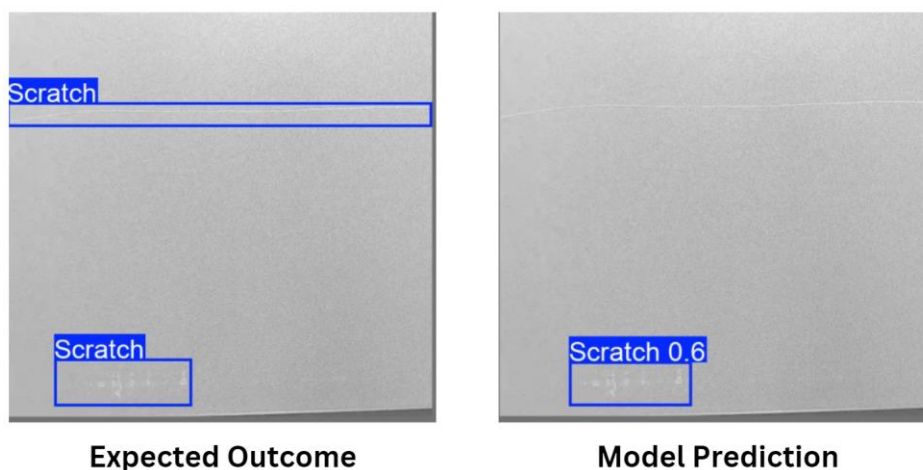


Figure 4.7: False negative on shallow scratch

The next example is also an example of false negative detection of scratch. The model does not detect two scratches on the aluminium surface. The

reason for this error may be due to one of the scratches being too small and not obvious while another scratch is too shallow.

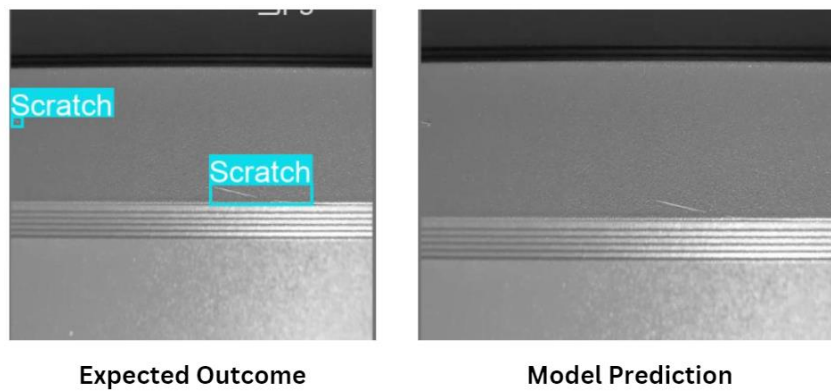


Figure 4.8: Two false negatives of scratch

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In conclusion, this study has implemented a comprehensive deep learning visual-based inspection approach based on object detection techniques. YOLOv8n was selected as the model to carry out defect detection on aluminium surfaces. All the objectives proposed in this study have been successfully achieved including the implementation of YOLOv8 for aluminium surface defect detection by utilizing data augmentation and transfer learning, identifying and categorizing the types of aluminium surface defects and evaluating the performance of YOLOv8. Besides that, the implementation of YOLOv8 with attention modules was also carried out to investigate the impact of attention modules on the performance of YOLOv8.

As a result, data augmentation and transfer learning help in improving the performance of YOLOv8. The YOLOv8n model with transfer learning achieved a mAP@0.5 of 94.3% and 79.7% of mAP@0.5:0.95 on the augmented dataset with the expansion of scratch class. The implementation of ECA and ResCBAM as attention modules for YOLOv8n is carried out as a comparison to the YOLOv8n. Despite attention modules are meant to be used to improve the performance of the model, neither ECA nor ResCBAM improves the mean average precision of YOLOv8n. Both of them achieved 91.1 and 93.3 in mAP@0.5 for ECA and ResCBAM respectively.

The results of this study reveal that while the YOLOv8 model was effective in detecting surface defects, the incorporation of attention modules did not lead to the expected enhancements in performance. This lack of improvement suggests that the attention mechanism may not have been well-suited for the specific characteristics of the dataset used. To further investigate this unexpected finding, it is crucial to analyse the attention module's design and its interaction with the feature extraction process of the YOLOv8 model. Future work could involve testing different types of attention mechanisms or adjusting hyperparameters to determine whether these changes could yield

better results. Understanding why the attention modules underperformed is essential for advancing defect detection technologies and maximizing their efficacy in real-world applications

By automating defect detection in aluminium manufacturing process, this study aims to minimize the need for manual defect detection, allowing human resources to be allocated to more critical tasks like important decision-making. Besides eliminating manual detection, this approach could effectively reduce the time needed for defect detection in the manufacturing process. Directly, the quality of the aluminium product will also increase as aluminium with defect detection will be deposited or remanufactured. YOLOv8 serves as a cutting-edge technology that would replace the old algorithms used in the current manufacturing process as most of them are not resilient to changes in light and noise.

5.2 Future Enhancements

Future improvements and enhancements are recommended:

- Implementing a more advanced object detection model:
As YOLOv9 and YOLOv10 have been released, implementation of them in detecting aluminium surface defect detection can be considered as the latest version often provide better performance.
- Collect more data according to the specific needs:
In order to improve the accuracy according to the needs of the manufacturing process, data collection based on the specific needs of defects to be detected can be done to train the model to perform well on the common defects.
- Cover the defect detection beside the surface:
As this study only focuses on the defects on aluminium surfaces, future enhancements can include detecting defects on entire aluminium extrudants such as tearing and weaving.
- Integrating with mobile or detection machine:

The main purpose of this study is to address the tedious process of defect detection in manufacturing. Hence, integration into mobile app for convenience defect detection or automated detection using detection machine could fully optimized the value of this model.

REFERENCES

- Batool, U., Shapiyai, M.I., Tahir, M., Ismail, Z.H., Zakaria, N.J. and Elfakharany, A., 2021. A Systematic review of deep learning for silicon wafer defect Recognition. *IEEE Access*, [online] 9, pp.116572–116593. <https://doi.org/10.1109/access.2021.3106171>.
- Bhatt, P.M., Malhan, R.K., Rajendran, P., Shah, B.C., Thakar, S., Yoon, Y.J. and Gupta, S.K., 2021. Image-Based surface Defect Detection Using Deep Learning: A review. *Journal of Computing and Information Science in Engineering*, [online] 21(4). <https://doi.org/10.1115/1.4049535>.
- Campbell, F.C., 2013. *Inspection of metals: Understanding the Basics*. ASM International.
- Chien, C.-T., Ju, R.-Y., Chou, K.-Y., Lin, C.-S. and Chiang, J.-S., 2024. YOLOv8-AM: YOLOv8 with Attention Mechanisms for Pediatric Wrist Fracture Detection. *arXiv (Cornell University)*. [online] <https://doi.org/10.48550/arxiv.2402.09329>.
- Demant, C., Waszkewitz, P., Streicher-Abel, B., Strick, M. and Schmidt, G.D., 1999. *Industrial Image Processing: Visual quality control in manufacturing*. [online] Available at: <<http://ci.nii.ac.jp/ncid/BA45690845>>.
- Deng, J., Dong, W., Socher, R., Li, L., Li, K. and Li, F., 2009. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [online] <https://doi.org/10.1109/cvpr.2009.5206848>.
- Duta, I.C., Liu, L., Zhu, F. and Shao, L., 2021. Pyramidal Convolution: Rethinking convolutional neural networks for visual recognition. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/2006.11538.pdf>>.
- Felzenszwalb, P.F., Girshick, R., McAllester, D. and Ramanan, D., 2010. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 32(9), pp.1627–1645. <https://doi.org/10.1109/tpami.2009.167>.
- Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, [online] 36(4), pp.193–202. <https://doi.org/10.1007/bf00344251>.
- Ganovska, B., Molitoris, M., Hošovský, A., Pitel, J., Królczyk, J.B., Ruggiero, A., Królczyk, G. and Hloch, S., 2016. Design of the Model for the On-line

Control of the AWJ Technology based on Neural Networks. *Indian Journal of Engineering and Materials Sciences*, [online] 23, pp.279–287. Available at: <<http://nopr.niscair.res.in/bitstream/123456789/39805/1/IJEMS%2023%284%29%20279-287.pdf>>.

Gidaris, S. and Komodakis, N., 2015. Object detection via a multi-region & semantic segmentation-aware CNN model. *arXiv (Cornell University)*. [online] Available at: <<http://export.arxiv.org/pdf/1505.01749>>.

Girshick, R., 2015. Fast R-CNN. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/1504.08083v2>>.

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] <https://doi.org/10.1109/cvpr.2014.81>.

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 37(9), pp.1904–1916. <https://doi.org/10.1109/tpami.2015.2389824>.

Hinton, G.E., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., W, A., Senior, Vanhoucke, V., Nguyen, P., Sainath, T.N. and Kingsbury, B., 2012a. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, [online] 29(6), pp.82–97. <https://doi.org/10.1109/msp.2012.2205597>.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2012b. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv (Cornell University)*. [online] Available at: <<http://export.arxiv.org/pdf/1207.0580>>.

Hou, Q., Zhou, D. and Feng, J., 2021. Coordinate Attention for Efficient Mobile Network Design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [online] <https://doi.org/10.1109/cvpr46437.2021.01350>.

Hussain, M., 2023. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines*, [online] 11(7), p.677. <https://doi.org/10.3390/machines11070677>.

Jia, F., Lei, Y., Lü, N. and Xing, S., 2018. Deep normalized convolutional neural network for imbalanced fault classification of machinery and its understanding via visualization. *Mechanical Systems and Signal Processing*, [online] 110, pp.349–367. <https://doi.org/10.1016/j.ymssp.2018.03.025>.

- Jiang, L., Yuan, B., Wang, Y., Ma, Y., Du, J., Wang, F. and Guo, J., 2023. MA-YOLO: a method for detecting surface defects of aluminum profiles with attention guidance. *IEEE Access*, [online] 11, pp.71269–71286. <https://doi.org/10.1109/access.2023.3291598>.
- Jizat, J.A.M., Majeed, A.P.P.A., Nasir, A.F.Ab., Taha, Z. and Yuen, E., 2021. Evaluation of the machine learning classifier in wafer defects classification. *ICT Express*, [online] 7(4), pp.535–539. <https://doi.org/10.1016/j.ict.2021.04.007>.
- LeCun, Y., Bengio, Y. and Hinton, G.E., 2015. Deep learning. *Nature (London)*, [online] 521(7553), pp.436–444. <https://doi.org/10.1038/nature14539>.
- Li, J., Su, Z., Geng, J. and Yin, Y., 2018. Real-time detection of steel strip surface defects based on improved YOLO detection network. *IFAC-PapersOnLine*, [online] 51(21), pp.76–81. <https://doi.org/10.1016/j.ifacol.2018.09.412>.
- Li, Z., Liu, F., Yang, W., Peng, S. and Zhou, J., 2022. A Survey of Convolutional Neural Networks: Analysis, applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, [online] 33(12), pp.6999–7019. <https://doi.org/10.1109/tnnls.2021.3084827>.
- Lin, H.-I. and Wibowo, F.S., 2021. Image Data Assessment Approach for Deep Learning-Based Metal Surface Defect-Detection Systems. *IEEE Access*, [online] 9, pp.47621–47638. <https://doi.org/10.1109/access.2021.3068256>.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. In: *Lecture Notes in Computer Science*. [online] pp.21–37. https://doi.org/10.1007/978-3-319-46448-0_2.
- Lopes, A.T., De Aguiar, E., De Souza, A.F. and Oliveira-Santos, T., 2017. Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order. *Pattern Recognition*, [online] 61, pp.610–628. <https://doi.org/10.1016/j.patcog.2016.07.026>.
- Lu, Y., 2019. Artificial intelligence: a survey on evolution, models, applications and future trends. *Journal of Management Analytics*, [online] 6(1), pp.1–29. <https://doi.org/10.1080/23270012.2019.1570365>.
- Lu, Y., Javidi, T. and Lazebnik, S., 2016. Adaptive Object Detection Using Adjacency and Zoom Prediction. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] <https://doi.org/10.1109/cvpr.2016.258>.
- Ma, Z., Li, Y., Huang, M., Huang, Q., Cheng, J. and Tang, S.Y., 2022. A lightweight detector based on attention mechanism for aluminum strip surface

defect detection. *Computers in Industry*, [online] 136, p.103585. <https://doi.org/10.1016/j.compind.2021.103585>.

Neuhauser, F.M., Bachmann, G. and Hora, P., 2019. Surface defect classification and detection on extruded aluminum profiles using convolutional neural networks. *International Journal of Material Forming*, [online] 13(4), pp.591–603. <https://doi.org/10.1007/s12289-019-01496-1>.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H. and Ng, A.Y., 2011. Multimodal Deep Learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, [online] pp.689–696. Available at: <<http://ai.stanford.edu/~ang/papers/icml11-MultimodalDeepLearning.pdf>>.

Population Matters, 2023. *Population and the sustainable development goals - population matters*. [online] Population Matters. Available at: <https://populationmatters.org/un-sdgs/?gad_source=1&gclid=CjwKCAiAuYuvBhApEiwAzq_YiV5YVW8U-Bek8nkFvH6mLXT10CoRBGevS-NESgASDzQ3di6Kb0ejRBoCVm8QAvD_BwE>.

Rai, D.D. and S., 2023. *Surface Defect Detection with Image Recognition Models*. [online] RTInsights. Available at: <<https://www.rtinsights.com/achieving-manufacturing-excellence-with-image-recognition-models-for-surface-defect-detection/>>.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] <https://doi.org/10.1109/cvpr.2016.91>.

Redmon, J. and Farhadi, A., 2016. YOLO9000: Better, Faster, Stronger. *arXiv (Cornell University)*. [online] <https://doi.org/10.48550/arxiv.1612.08242>.

Redmon, J. and Farhadi, A., 2018. YOLOV3: an incremental improvement. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/1804.02767>>.

Ren, S., He, K., Girshick, R. and Sun, J., 2017a. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 39(6), pp.1137–1149. <https://doi.org/10.1109/tpami.2016.2577031>.

Ren, S., He, K., Girshick, R. and Sun, J., 2017b. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [online] 39(6), pp.1137–1149. <https://doi.org/10.1109/tpami.2016.2577031>.

Riesenhuber, M. and Poggio, T., 1999. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, [online] 2(11), pp.1019–1025. <https://doi.org/10.1038/14819>.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C. and Li, F., 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, [online] 115(3), pp.211–252. <https://doi.org/10.1007/s11263-015-0816-y>.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y., 2014. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/1312.6229>>.

Shf, P. and Zhao, C., 2020. Review on Deep based Object Detection. *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, [online] pp.372–377. <https://doi.org/10.1109/ichci51889.2020.00085>.

Shi, J., Yang, J. and Zhang, Y., 2022. Research on Steel Surface Defect Detection Based on YOLOv5 with Attention Mechanism. *Electronics (Basel)*, [online] 11(22), p.3735. <https://doi.org/10.3390/electronics11223735>.

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for Large-Scale image recognition. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/1409.1556>>.

Solawetz, J., 2024. *What is YOLOv8? The Ultimate Guide. [2024]*. [online] Roboflow Blog. Available at: <<https://blog.roboflow.com/whats-new-in-yolov8/>>.

Srivastava, N. and Salakhutdinov, R., 2013. Discriminative Transfer Learning with Tree-based Priors. *Neural Information Processing Systems*, [online] 26, pp.2094–2102. Available at: <<http://papers.nips.cc/paper/5029-discriminative-transfer-learning-with-tree-based-priors.pdf>>.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] <https://doi.org/10.1109/cvpr.2015.7298594>.

Tang, J., Liu, S., Zhao, D., Tang, L., Zou, W. and Zheng, B., 2023. An algorithm for Real-Time Aluminum Profile surface Defects detection based on lightweight network structure. *Metals*, [online] 13(3), p.507. <https://doi.org/10.3390/met13030507>.

Tang, Y., 2013. Deep Learning using Linear Support Vector Machines. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/1306.0239.pdf>>.

Tousch, A.-M., Herbin, S. and Audibert, J.-Y., 2012. Semantic hierarchies for image annotation: A survey. *Pattern Recognition*, [online] 45(1), pp.333–345. <https://doi.org/10.1016/j.patcog.2011.05.017>.

Wang, T., Su, J., Xu, C. and Zhang, Y., 2022. An intelligent method for detecting surface defects in aluminium profiles based on the improved YOLOV5 algorithm. *Electronics*, [online] 11(15), p.2304. <https://doi.org/10.3390/electronics11152304>.

Wang, Y., Wang, H. and Xin, Z., 2022. Efficient Detection model of steel strip surface defects based on YOLO-V7. *IEEE Access*, [online] 10, pp.133936–133944. <https://doi.org/10.1109/access.2022.3230894>.

Wang, Z., Wang, X. and Wang, G., 2018. Learning fine-grained features via a CNN Tree for Large-scale Classification. *Neurocomputing*, [online] 275, pp.1231–1240. <https://doi.org/10.1016/j.neucom.2017.09.061>.

Wei, R. and Bi, Y., 2019. Research on recognition technology of aluminum profile surface defects based on Deep learning. *Materials*, [online] 12(10), p.1681. <https://doi.org/10.3390/ma12101681>.

Woo, S., Park, J., Lee, J.-Y. and Kweon, I.S., 2018. CBAM: Convolutional Block Attention Module. *arXiv (Cornell University)*. [online] <https://doi.org/10.48550/arxiv.1807.06521>.

Wu, X., Sahoo, D. and Hoi, S.C.H., 2020. Recent advances in deep learning for object detection. *Neurocomputing*, [online] 396, pp.39–64. <https://doi.org/10.1016/j.neucom.2020.01.085>.

Xiao, T., Zhang, J., Yang, K., Peng, Y. and Zhang, Z., 2014. Error-Driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification. *Proceedings of the International Conference on Computer Vision (ICCV)*. [online] <https://doi.org/10.1145/2647868.2654926>.

Xu, Y., Zhang, K. and Wang, L., 2021. Metal surface defect detection using modified YOLO. *Algorithms*, [online] 14(9), p.257. <https://doi.org/10.3390/a14090257>.

Yamaguchi, K., Sakamoto, K., Akabane, T. and Fujimoto, Y., 1990. A neural network for speaker-independent isolated word recognition. *First International Conference on Spoken Language Processing*. [online] <https://doi.org/10.21437/icslp.1990-282>.

- Yang, D., Cui, Y., Yu, Z. and Yuan, H., 2021. Deep Learning based Steel Pipe weld defect Detection. *Applied Artificial Intelligence*, [online] 35(15), pp.1237–1249. <https://doi.org/10.1080/08839514.2021.1975391>.
- Yao, Y., Rosasco, L. and Caponnetto, A., 2007. On early stopping in gradient descent learning. *Constructive Approximation*, [online] 26(2), pp.289–315. <https://doi.org/10.1007/s00365-006-0663-2>.
- Yoo, D., Park, S., Lee, J.-Y., Paek, A.S. and Kweon, I.S., 2015. AttentionNet: Aggregating Weak Directions for Accurate Object Detection. *arXiv (Cornell University)*. [online] Available at: <<https://arxiv.org/pdf/1506.07704.pdf>>.
- Yun, J.P., Choi, S., Kim, J.-W. and Kim, S.W., 2009. Automatic detection of cracks in raw steel block using Gabor filter optimized by univariate dynamic encoding algorithm for searches (uDEAS). *NDT & E International*, [online] 42(5), pp.389–397. <https://doi.org/10.1016/j.ndteint.2009.01.007>.
- Zeiler, M.D. and Fergus, R., 2014. Visualizing and understanding convolutional networks. In: *Lecture notes in computer science*. [online] pp.818–833. https://doi.org/10.1007/978-3-319-10590-1_53.
- Zhang, A., Lipton, Z.C., Mu, L. and Smola, A.J., 2021. Dive into Deep Learning. *arXiv (Cornell University)*. [online] Available at: <<http://arxiv.org/pdf/2106.11342.pdf>>.
- Zhang, D., Song, K., Xu, J., He, Y. and Yan, Y., 2020. Unified detection method of aluminium profile surface defects: Common and rare defect categories. *Optics and Lasers in Engineering*, [online] 126, p.105936. <https://doi.org/10.1016/j.optlaseng.2019.105936>.
- Zhao, C., Shu, X., Yan, X., Zuo, X. and Zhu, F., 2023. RDD-YOLO: A modified YOLO for detection of steel surface defects. *Measurement*, [online] 214, p.112776. <https://doi.org/10.1016/j.measurement.2023.112776>.
- Zhao, L. and Zhu, M., 2023. MS-YOLOV7: YOLOV7 based on Multi-Scale for Object Detection on UAV aerial Photography. *Drones*, [online] 7(3), p.188. <https://doi.org/10.3390/drones7030188>.
- Zhao, Z.-Q., Zheng, P., Xu, S.-T. and Wu, X., 2019. Object Detection with Deep Learning: A review. *IEEE Transactions on Neural Networks and Learning Systems (Print)*, [online] 30(11), pp.3212–3232. <https://doi.org/10.1109/tnnls.2018.2876865>.
- Zinkevich, M., Weimer, M., Li, L. and Smola, A., 2010. Parallelized stochastic gradient descent. *Neural Information Processing Systems*, [online] 23, pp.2595–2603. Available at: <<http://martin.zinkevich.org/publications/nips2010.pdf>>.

APPENDICES

Appendix A: Model Configuration File for YOLOv8n

```

# YOLOv8.0n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 12

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f, [256]] # 15 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f, [512]] # 18 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2f, [1024]] # 21 (P5/32-Large)

  - [[15, 18, 21], 1, Detect, [nc]] # Detect(P3, P4, P5)

```

Appendix B: Model Configuration File for YOLOv8n with ECA

```

backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 12
  - [-1, 1, ECAAttention, [512]]

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f, [256]] # 16 (P3/8-small)
  - [-1, 1, ECAAttention, [256]]

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f, [512]] # 20 (P4/16-medium)
  - [-1, 1, ECAAttention, [512]]

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2f, [1024]] # 24 (P5/32-large)
  - [-1, 1, ECAAttention, [1024]]

  - [[17, 21, 25], 1, Detect, [nc]] # Detect(P3, P4, P5)

```

Appendix C: Model Configuration File for YOLOv8n with ResCBAM

```

backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 12
  - [-1, 1, ResBlock_CBAM, [512]]

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f, [256]] # 16 (P3/8-small)
  - [-1, 1, ResBlock_CBAM, [256]]

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 12], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f, [512]] # 20 (P4/16-medium)
  - [-1, 1, ResBlock_CBAM, [512]]

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 9], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2f, [1024]] # 24 (P5/32-large)
  - [-1, 1, ResBlock_CBAM, [1024]]

  - [[17, 21, 25], 1, Detect, [nc]] # Detect(P3, P4, P5)

```

Appendix D: Code snippet for ECA attention module in init.py

```

class ECAAttention(nn.Module):
    """Constructs a ECA module.
    Args:
        channel: Number of channels of the input feature map
        k_size: Adaptive selection of kernel size
    """

    def __init__(self, c1, k_size=3):
        super(ECAAttention, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.conv = nn.Conv1d(1, 1, kernel_size=k_size, padding=(k_size - 1) // 2, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # feature descriptor on the global spatial information
        y = self.avg_pool(x)
        y = self.conv(y.squeeze(-1).transpose(-1, -2)).transpose(-1, -2).unsqueeze(-1)
        # Multi-scale information fusion
        y = self.sigmoid(y)

        return x * y.expand_as(x)

```

Appendix E: Code snippet for ResCBAM attention module in init.py

```

class ResBlock_CBAM(nn.Module):
    def __init__(self, in_places, places, stride=1, downsampling=False, expansion=1):
        super(ResBlock_CBAM, self).__init__()
        self.expansion = expansion
        self.downsampling = downsampling

        self.bottleneck = nn.Sequential(
            nn.Conv2d(in_channels=in_places, out_channels=places, kernel_size=1, stride=1, bias=False),
            nn.BatchNorm2d(places),
            nn.LeakyReLU(0.1, inplace=True),
            nn.Conv2d(in_channels=places, out_channels=places, kernel_size=3, stride=stride, padding=1, bias=False),
            nn.BatchNorm2d(places),
            nn.LeakyReLU(0.1, inplace=True),
            nn.Conv2d(in_channels=places, out_channels=places * self.expansion, kernel_size=1, stride=1,
                    bias=False),
            nn.BatchNorm2d(places * self.expansion),
        )
        # self.cbam = CBAM(c1=places * self.expansion, c2=places * self.expansion, )
        self.cbam = CBAM(c1=places * self.expansion)

        if self.downsampling:
            self.downsample = nn.Sequential(
                nn.Conv2d(in_channels=in_places, out_channels=places * self.expansion, kernel_size=1, stride=stride,
                        bias=False),
                nn.BatchNorm2d(places * self.expansion)
            )
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        residual = x
        out = self.bottleneck(x)
        out = self.cbam(out)
        if self.downsampling:
            residual = self.downsample(x)

        out += residual
        out = self.relu(out)
        return out

```