Use AI to Detect Defect Pin in Electrical Connector

BY

Yong Tian Ze

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr Lee Wai Kong who has given me this bright opportunity to engage in a machine learning project. It is my first time to learn how to do the dataset preparation and learn how to complete a big project. A million thanks to you.

Besides, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project aims to develop an intelligent inspection model capable of detecting defects in small electrical connector pins, which are critical components in many electronic systems. The work is structured into two primary components: data preparation and model development. In the data preparation phase, a custom dataset will be generated, featuring images of electrical connectors with three common types of pin defects: missing, shifted, and rotated pins. High-quality image data is essential for accurate model training and reliable detection outcomes. The model development phase leverages the YOLOv8 object detection algorithm, selected for its balance of speed and accuracy in real-time applications. Image processing techniques are employed to enhance dataset quality, and the dataset is annotated manually to ensure precision in model training. Performance evaluation will be conducted using several key metrics—accuracy, recall, precision, and F1 score—to assess the model's capability in identifying defective pins effectively. This project ultimately seeks to offer a practical and automated solution for improving quality control in electrical connector manufacturing processes, reducing the need for manual inspection and minimizing human error.

Area of Study: Image Processing, Data preparation

Keywords: Data Generator by OpenCV function, Electrical Connector Pin Region Detection Model, Defect Electrical Connector Model, Machine learning, Yolov8 Algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

YOLO	You Only Look Once
CNN	Convolutional neural network
SVM	Support Vector Machine
R-CNN	Regions with Convolutional Neural Networks
SMU	Smooth Maximum Unit
VGG	Visual Geometry Group

Chapter 1
Introduction

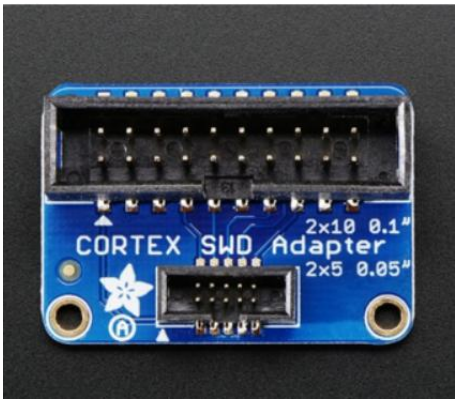
This chapter presents the problem statement, motivation, research objectives, project scope and direction, contributions, and report organization. The aim is to provide a clear and comprehensive understanding of the project, highlighting the underlying challenges, the significance of the research, and the intended outcomes. By outlining these key elements, this chapter aims to ensure that the project's purpose, approach, and significance are clearly conveyed and fully understood.

1.1 Problem Statement and Motivation

With the expansion of high-tech manufacturing facilities, the demand for electrical connectors is steadily increasing in modern society. Electrical connectors are essential components in various industries, playing a crucial role in both power and data transmission. These connectors consist of several key elements, including leads or pins, frames, and housings. Among these, the pins are particularly important, as they establish the necessary connections between different components. However, due to their inherent fragility, pins are susceptible to damage during transportation or production processes. This vulnerability often requires additional inspections before the connectors can be used in assembly or shipped to customers, thus complicating the overall process. Such inspections are critical, as defective connectors can lead to a range of issues, from minor connectivity problems to severe hazards, including potentially fatal electrocutions [1].

The inspection process of electrical connectors typically involves two primary methods: visual examination and electrical testing. While visual inspection is commonly employed, it is subject to limitations such as human error and missed detections. As a result, the integration of artificial intelligence (AI) image processing technology in electrical connector inspection has emerged as a promising trend. Existing literature demonstrates the significant potential of AI in this domain, with research showing its capacity to automate the inspection process effectively, thereby confirming the feasibility of AI-based solutions. Despite this progress, much of the current research has predominantly focused on inspecting larger connectors, as exemplified in Figure 1.1 (a). In contrast, the inspection of smaller electrical connectors with pin spacing of

approximately 0.5mm, as shown in Figure 1.1(b), has received relatively less attention, resulting in limited advancements in this field. Consequently, there is a clear need for the development of a high-accuracy and high-efficiency model specifically designed for the inspection of small electrical connectors. Such a model would not only offer superior accuracy and efficiency compared to human visual inspection but also significantly reduce labor costs for manufacturing companies.

(a) 

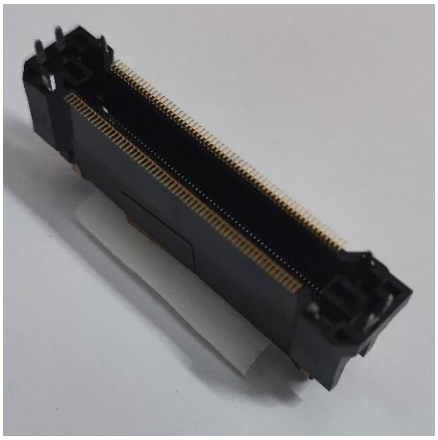
(b) 

Figure 1.1 Electrical Connector Show (a) Normal Size Electrical Connector (b) Small Size Electrical Connector

1.2 Project Objectives

This section will describe the aims and objectives of the project. The main objectives aim to achieve are as follows:

I. Dataset preparation

This objective focuses on collecting and labelling electrical connector images to create a complete and clean dataset for training, testing, and validation. The benefit of a self-prepared dataset is that it ensures high quality, including factors such as image resolution, capturing angle, capturing environment, and more.

II. High accuracy model

The goal of this project is to develop a model that achieves a minimum accuracy of 85%, demonstrating its ability to fully replace human-based inspection for defect detection. This will provide strong evidence of the model's capability to accurately and efficiently perform electrical connector inspections, offering a reliable alternative to manual inspection methods.

III. Reduce the processing time

The goal is for the trained model to achieve a detection speed of one image per second. This is because, in practical scenarios, the detection speed of the trained model must align with the speed of the conveyor belt used in the system. Therefore, a faster detection speed directly translates to improved efficiency, allowing for real-time inspection of electrical connectors without delays, thus enhancing the overall performance of the inspection process.

These three main objectives will help address the challenges in electrical connector inspection and achieve project goals, such as reducing accidents caused by faulty electrical connectors, decreasing reliance on human resources in the inspection process, and minimizing time and costs for the company.

1.3 Project Scope and Direction

The project scope outlines the deliverables and objectives to be achieved by the end of the study. The primary goal of this project is to develop a highly accurate and efficient electrical connector inspection model. In addition to this, the project will involve the creation of a new dataset specifically for small electrical connectors. This dataset will consist of images of defective electrical connector pins, accompanied by labeled position text files, which will serve as a foundation for training and validating the inspection model.

Deliverables

I. Electrical Connector Dataset

The dataset will be created using a pre-installed webcam to capture images of electrical connectors. Following this, OpenCV will be utilized to modify the pins of the connectors in the images, thereby generating defective connector data. The purpose of this approach is to obtain high-quality and comprehensive data that can be used for training the model inspecting real-world cases. This dataset will enable the model to effectively learn and detect defects in electrical connectors during the inspection process.

II. Electrical Connector Inspection Model

The electrical connector inspection model will be trained using the custom-made electrical connector dataset. The deploy model should be of high accuracy and efficiency which can overcome the problem and issue faced in the real world.

1.4 Contributions

This project will provide a new approach to inspect the defect pins. Additionally, this study emphasizes the model's ability to analyze small objects, with each pin measuring less than 1 cm—an uncommon focus in traditional model training. Such applications reflect a significant technological trend in the current development of the tech industry.

Beyond the model analysis, meticulous attention has been given to dataset preparation. The setup of the webcam, including aspects such as angle, lighting, and background, is detailed to ensure optimal image capture. For generating defect images, innovative methods have been employed using OpenCV, such as pixel manipulation to simulate misaligned pins and parameter adjustments to create rotated pins. These techniques not only enhance the realism of the defect dataset but also contribute to a more robust and versatile model for real-world applications.

1.5 Report Organization

This report is organized into seven chapters: Chapter 1 introduces the project, Chapter 2 covers the literature review, Chapter 3 outlines the proposed method and approach, Chapter 4 discusses the system design, Chapter 5 serves as the system implementation, Chapter 6 presents the system evaluation and discussion, and Chapter 7 concludes the report.

Chapter 1 presents the problem statement, motivation, project objectives, scope, direction, contributions, and overall project framework. It provides a foundation for understanding the significance and direction of the project.

Chapter 2 focuses on the literature review, analyzing the strengths and weaknesses of similar projects while introducing the technologies and methods they employ. This section aims to highlight the current state of research and the relevance of the chosen approach.

Chapter 3 outlines the proposed method and approach, covering the system specifications, including the project methodology diagram, activity diagram and use case diagram.

Chapter 4 presents the system design. This part covers the logic used in the project, including the system block diagram, system component specifications, circuit and component design, and the interaction between system components.

Chapter 5 serves as the system implementation, which covers the hardware and software setup activities required to build the system.

Chapter 6 presents the system evaluation and discussion, showing the testing results, model performance, project challenges, and objective evaluation.

Finally, Chapter 7 concludes the report, summarizing the project's challenges, motivation, and proposed solutions, offering a reflection on the work accomplished and recommendations for future improvement.

Chapter 2
Literature Review

This chapter focuses on literature review, do pre-research before starting the project and analyzing the strengths and weaknesses of similar projects while introducing the technologies and methods they employ.

2.1 Review of technology

This section shows the technology research conducted before starting the project; the goal is to assess these papers to ensure the idea can work.

2.1.1 Dataset Preparation

Diaz, Kushibar, Osuala, et al., 2021 [2] provide data preparation guidelines for artificial intelligence in medical imaging. In the paper, it highlights the data acquisition flow, which offers a good view on dataset generation. There are five steps used in dataset preparation: (i) image acquisition at clinical sites, (ii) image de-identification to remove personal information and protect patient privacy, (iii) data curation to control for image and associated information quality, (iv) image storage, and (v) image annotation. Although this project is not focused on electrical connectors, it still provides a valuable reference on how to build a standard dataset and points out which aspects need to be considered when developing a dataset.

2.1.2 A Brief Introduction to OpenCV

Culjak, D., Abram, et al., 2012 [3] provide a concise yet informative overview of OpenCV, emphasizing its powerful capabilities in the field of image processing. The paper outlines key functions and commonly used methods within OpenCV, demonstrating how the library can be applied effectively in various computer vision tasks. An experimental section is also included, showcasing the results of local image processing methods, as illustrated in Figure 2.1.1. This practical demonstration reinforces OpenCV's usefulness in real-world applications. Although the paper does not directly focus on electrical connector inspection, it builds a strong foundation and provides confidence that OpenCV-based techniques are appropriate for defect pin detection tasks. The features and flexibility of OpenCV make it a suitable choice for preprocessing and dataset generation in this project, ultimately supporting the model training process for accurate inspection.

Method	Tskuba	Venus	Teddy	Cones
ADCensus [26]	1.48	0.25	6.22	7.25
AdaptingBP [27]	1.37	0.21	7.06	7.92
CoopRegion [28]	1.16	0.21	8.31	7.18
DoubleBP [29]	1.29	0.45	8.30	8.78
OpenCV	6.79	5.01	14.84	9.57

Figure 2.1.1 Local Method Result

2.1.3 Deep Learning in Image Recognition

Y. Li, 2022 [4] provides an overview of the application of deep learning in image recognition. It first outlines the development of image recognition technology and introduces the main deep learning models used, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs). The paper then summarizes the achievements of deep learning in various application fields such as face recognition, medical image recognition, and remote sensing image classification. It also analyzes the shortcomings in the research process and discusses the future development trends of deep learning in image recognition, including the effective recognition of video images and the theoretical strengthening of models.

About the different types the deep learning, the paper concludes the strength of the different algorithm. The CNN model simplifies image preprocessing by eliminating extensive manual feature extraction, leveraging local connectivity and feature repetition to reduce the number of parameters. The RNN model, with its memory capability, is ideal for sequential data as earlier inputs significantly influence later outputs. GAN introduces a game-theoretic training method between a generator and discriminator, improving efficiency through backpropagation and offering a more rigorous loss function than traditional methods, making it widely applicable in image processing.

This paper demonstrates that deep learning performs effectively in image recognition, despite some challenges that still need to be addressed. Nevertheless, it offers a significantly improved approach compared to traditional methods.

2.1.4 YOLO model review

Terven J, Córdova-Esparza DM, et al., 2023 [5] provides a comprehensive review of the evolution of the YOLO (You Only Look Once) object detection framework, from the original YOLOv1 to the latest YOLOv8 and YOLO-NAS models. The review examines the key innovations and contributions in each YOLO iteration as figure 2.1.2 show, discussing the changes in network architecture, training techniques, and performance tradeoffs between speed and accuracy. The paper also covers the standard object detection metrics, such as average precision (AP) and non-maximum suppression (NMS), and how they are computed for different datasets like PASCAL VOC and Microsoft COCO. Finally, the review highlights the potential future research directions to further enhance real-time object detection systems.

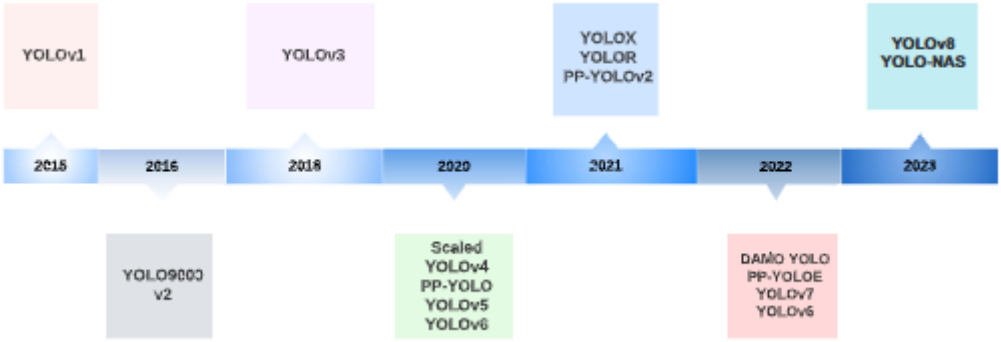


Figure 2.1.2 Timeline of YOLO versions

In the paper, the operation logic of the YOLO model is divided into two main steps: detecting possible regions containing objects (region proposals) and applying a classifier to these proposed regions. Additionally, the paper highlights that the YOLO model adopts a more streamlined output structure, using two separate outputs—one for classification to determine object probabilities and another for regression to predict bounding box coordinates. All versions of the YOLO model's performance are shown in Figure 2.1.3. The YOLO family focuses on balancing speed and accuracy, aiming to deliver real-time performance without sacrificing the quality of detection results.

Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	Darknet	Darknet24	63.4
YOLOv2	2016	Yes	Darknet	Darknet24	78.6
YOLOv3	2018	Yes	Darknet	Darknet53	33.0
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	YOLOv5CSPDarknet	55.8
PP YOLO	2020	Yes	PaddlePaddle	ResNet50 vd	45.2
Scaled YOLOv4	2021	Yes	Pytorch	CSPDarknet	56.0
PP YOLOv2	2021	Yes	PaddlePaddle	ResNet101 vd	50.3
YOLOR	2021	Yes	Pytorch	CSPDarknet	55.4
YOLOX	2021	No	Pytorch	YOLOXCSPDarknet	51.2
PP YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	YOLOv7Backbone	56.8
DAMO YOLO	2022	No	Pytorch	MAE NAS	50.0
YOLOv8	2023	No	Pytorch	YOLOv8CSPDarknet	53.9
YOLO NAS	2023	No	Pytorch	NAS	52.2

Figure 2.1.3 YOLO Family Comparison

In conclusion, this paper presents the function and workflow logic of the YOLO models. It also compares the performance of different YOLO versions.

2.2 Review of the Existing Systems

This section focuses on researching cases similar to this project and studying their methods and approaches to identify possible improvements.

2.2.1 Machine Vision Inspection of Electrical Connectors Based on Improved Yolov3

W. Wu, Q. Li et al., 2020 [6] proposed a research paper inspired by an improved YOLOv3 (You Only Look Once, version 3) algorithm for detecting electrical connector defects, such as solder spots on solder cups. The key innovations include using K-means clustering to obtain better anchor boxes, fusing feature maps from different layers to improve the detection of small objects and modifying the network architecture to enhance feature reuse and acquisition.

In their work, the YOLOv3 network is optimized for detecting small targets, although the initial prediction results did not meet expectations. The anchor box mechanism, a set of predefined candidate boxes with fixed height and width, directly affects the accuracy and speed of the network model. Since the size of the anchor boxes generated by the dataset does not meet the defect size requirements, K-means clustering is applied. The paper uses Average Intersection over Union (Avg IOU) as the criterion for evaluating the clustering results, as shown in formula (1).

$$f = \arg\max \frac{\sum_{i=1}^{k} \sum_{j=1}^{n_k} I_{IOU}(T, P)}{n} \quad (1)$$

In the experiments, the number of clusters k is selected within the range of 1 to 20. The conclusion is that as the number of clusters increases, the Avg IOU also increases.

The traditional YOLOv3 uses an 8-fold downsampled feature map, which struggles with detecting very small targets. To address this, the method proposes using a 2-fold downsampled feature map that contains more detailed information, making it better suited for detecting small defects. Additionally, the method simplifies the network by focusing only on the 2-fold downsampled feature map for small targets, eliminating the need for larger scales, which reduces computation and improves detection speed and accuracy.

Moreover, the method modifies the residual structure by adding two residual units and reducing the number of DBL units, as shown in Figure 2.1. This helps capture more low-level details and prevents gradient disappearance. This approach enhances the model's ability to accurately detect small defects, making it more suitable for the specific task of electrical connector inspection.

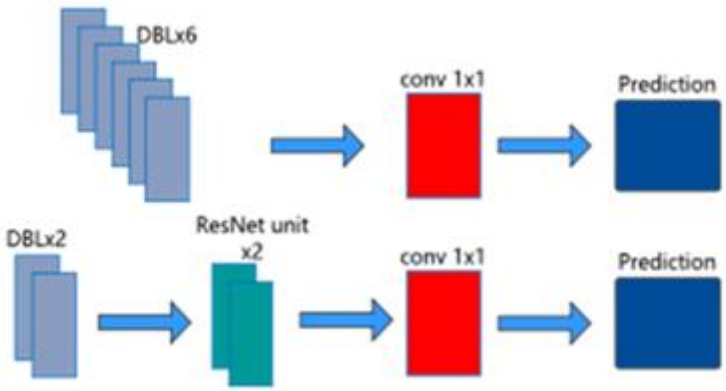


Figure 2.2.1 Residual structure

2.2.2 Simultaneous Detection of Defect in Electrical Connectors Based on Improved Convolution Neural Network

Y. Zhao, J. Li, Q. Zhang et al.,2022 [7] proposed a study on simultaneous detection of defects in electrical connectors using an improved convolutional neural network (CNN) architecture, specifically ResNet-152. ResNet-152 was chosen due to its ability to deliver satisfactory

detection accuracy on public datasets and its smaller parameter size compared to the Visual Geometry Group (VGG) series. The ResNet-152 model consists of five stages, as shows in figure 2.2, with the first stage involving a 7x7 convolution to downsample the input image while preserving as much original information as possible. Stages 2-5 consist of bottleneck layers that utilize residual networks to prevent gradient disappearance and enhance feature extraction.

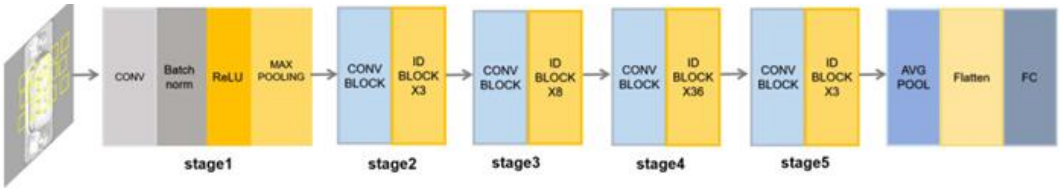


Figure 2.2.2 Five Stages in ResNet-152

To address the challenge of detecting small targets, the original 7x7 convolution kernel was replaced with three 3x3 convolution kernels in the first layer, expanding the receptive field (RF) and increasing sensitivity to small features. The study also introduced the SMU (Smooth Maximum Unit) activation function (Formula 2) to replace the ReLU function (Formula 3), which, although computationally efficient, can lead to feature masking due to sparse processing. The SMU function enhances feature extractability and reduces overfitting.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2}$$

$$\mathrm{SMU}(x) = \frac{(1+\alpha)x + (1-\alpha)x \cdot \mathrm{erf}(\mu(1-\alpha)x)}{2}. \tag{3}$$

In addition, traditional fixed-sized convolution kernels were replaced with Deformable Convolution Networks (DCNs), which adapt to geometric changes in the image, improving the detection of small, irregular features. This approach significantly enhances the model's ability to extract key geometric information, thereby improving defect detection accuracy. Dropout regularization was also employed to mitigate overfitting by randomly discarding neurons during training, ensuring robustness in the deep network.

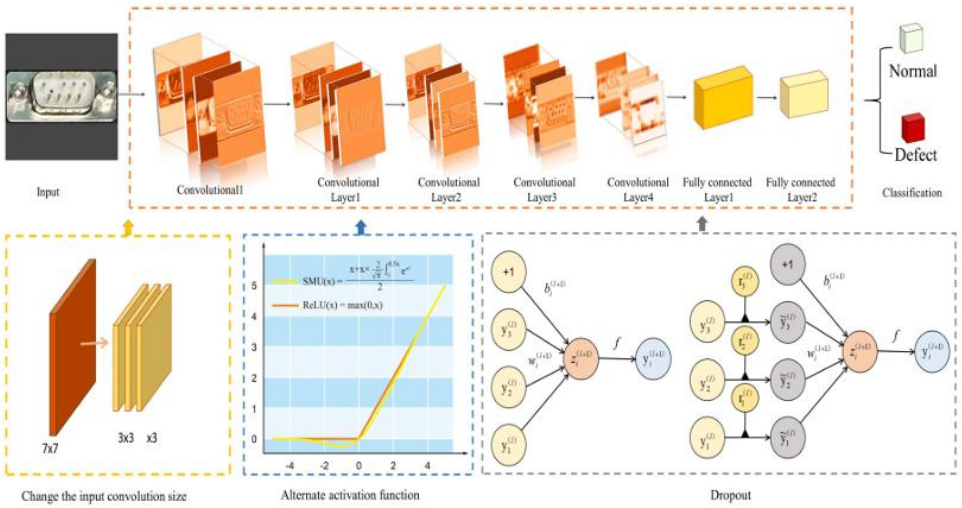


Figure 2.2.3 Improvements to the model

2.2.3 Vision-based Adaptive Stereo Measurement of Pins on Multi-type Electrical Connectors

D. Zhao et al., 2019 [8] presented a study inspired by Faster R-CNN to address the challenges of defect detection in electrical inspections, particularly focusing on the difficulties of detecting small area ratios using deep learning. The proposed approach involves a two-step identification strategy that integrates prior knowledge constraints derived from the manufacturing information of products. The process is divided into two tasks: prior knowledge loading and pin recognition, with each task using a separate Faster R-CNN model. The first model is responsible for classifying electrical connectors and identifying the pin region, while the second model targets the pins within this region, effectively minimizing background interference.

The Faster R-CNN framework utilizes two branch modules and a shared convolutional layer for object detection. The Region Proposal Network (RPN) generates candidate regions, and the Fast R-CNN detector performs target detection within these regions. The shared convolutional layer, based on ZF-Net, allows for the joint training of the RPN and Fast R-CNN detectors during the training phase. During inference, the pre-trained RPN and Fast R-CNN detectors process images sequentially to produce the final detection results.

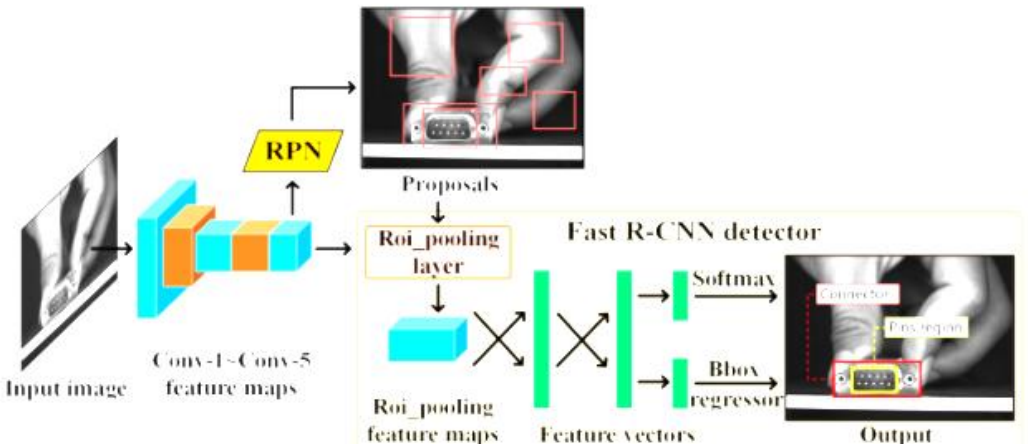


Figure 2.2.4 Principal Structure of Target Recognition based on Faster-RCNN

In the registration process, four control points are selected from the vertices of the pin region. The registration strategy, inspired by iterative correction of point pair correspondence and configuring weight to minimize the objective function, involves calculating the rotation matrix and translation vector. The initial state of the algorithm is crucial, as it aligns the template corners with the obtained corners, providing an excellent initial scaling factor and ensuring stable registration.

After identifying the connector model and pin positions, a hierarchical extraction strategy is implemented to search for expected position characterization points within the rectangular boxes of different pin types. This strategy accommodates potential imaging diversity and ensures robust pin recognition.

To address common issues such as background variation, region adhesions, and noise, an adaptive binarization strategy based on shape and structure constraints is designed. This strategy uses a shape scoring criterion to evaluate the aspect ratio of bounding rectangles and other factors, effectively distinguishing the desired features. Polynomial fitting and feature quality conditions are applied to ensure consistency in 3D point reconstruction. The hierarchical analyzer performs structural analysis to distinguish elements and extract target points that describe the pin position. The analysis is guided by the consistency of pixel extraction and the correctness of point-pair relationships, ensuring reliable pin recognition even in the presence of noise or structural deviations.

2.2.4 Detection of Defect in the Manufacturing of Electrical Motor Stators using Vision System: Electrical Connectors

BCF de Oliveira, ALS Pacheco, RC C Flesch et al., 2016 [9] conducted a study inspired by LabVIEW® and its Vision Development Module, focusing on the automated inspection of electrical connectors. The first step in the methodology involves locating the connector in a captured image using a pattern detection function that compares the image to a database of connector images taken under various lighting conditions. Once the connector is identified, its coordinate system is used to position the inspection tools.

To enhance the contrast in the region of interest, specifically around the holes where clips are located, a thresholding technique is applied with a default value of 127, which has proven effective in preliminary tests. The study introduces several defect detection techniques to ensure reliable inspection.

One technique involves circumference detection, which identifies circles in the thresholded image using Danielsson's distance mapping. Any deviation from the expected diameter of these circles indicates a potential defect. Another technique focuses on edge detection and distance measurement, where the edges of the holes are detected, and the distance between them is measured. If this distance is smaller than the expected diameter, a defect is flagged. Additionally, the methodology includes an inspection of the connector's upper part, particularly in cases where the clip might be out of position due to cable tension. By measuring the width of the clip and comparing it to the expected width, the technique identifies any defects.

After individual testing of these techniques, they are integrated into a software tool that automates the inspection process. The methodology ensures reliable detection of defects by employing redundant techniques that complement each other. Although the electrical connector model used in this study differs from that in the current paper, it offers alternative methods for image analysis, providing valuable insights into detecting electrical leads or holes.

2.2.5 Research on Ai-Based Gold Removal Technology for Aviation Connector Cup Cavity Surface

YZ. Yong, Deru. Song, JC. Li, TY. Qin, ZM. Qie, RZ. Zhang et al., 2023 [10] conducted a study using YOLOv8 to detect aviation connectors, which are similar to electrical connectors composed of leads and housing. The study demonstrates that YOLOv8 outperforms other versions of YOLO in terms of accuracy for this task.

To create an effective dataset for aviation connector inspection, the study emphasizes capturing images from various angles, accurately labelling them, and including both part images and empty environment images to reduce environmental interference. For parts that are difficult to recognize, additional images are necessary to prevent overfitting. Consistent and precise labelling is crucial for the network's performance in tasks like object detection and segmentation, forming the basis for the AIC algorithm's accurate identification and localization.

In data preprocessing, the study ensures that the dataset is standardized for effective training. Data enhancement techniques, such as rotation, flipping, scaling, translation, and noise perturbation, are applied to increase the training set size and improve the model's generalization.

The paper also introduces a weighted algebraic distance least squares ellipse fitting algorithm to ensure precise positioning of the aviation connector solder cup's center and opening direction, which is critical for preventing errors in processes like tinning or gold removal that could damage or scrap the connector.

2.2.6 Summary of the Existing Function

In the above paper, deep learning techniques are used to train a model to solve this problem. The model techniques provided include YOLO, CNN, and Faster-RCNN. We will compare and analyze the advantages and disadvantages of different model techniques to find the most suitable approach for training our model. Since the paper by BCF de Oliveira, ALS Pacheco, and RC C Flesch et al., 2016 [9] offers ideas on image processing, it will not be included in the model analysis. We will conduct our analysis based on accuracy, efficiency, complexity, and the model's requirements for the dataset.

In most cases, Faster-RCNN provides higher accuracy. Known for its high precision, Faster-RCNN can effectively locate and classify multiple objects within an image. D. Zhao et al., 2019 [8] demonstrated the robustness and high accuracy of Faster-RCNN in image analysis. Compared to Faster-RCNN, CNN and YOLO generally exhibit lower accuracy. Different versions of YOLO can also result in varying accuracy outcomes; for instance, YOLOv8 typically offers higher accuracy than YOLOv3. CNN is generally not well-suited for object localization, leading to lower accuracy. However, due to CNN's relatively simple architecture, its accuracy can be improved by incorporating various techniques, as mentioned by Y. Zhao, J. Li, Q. Zhang et al., 2022 [7].

In terms of image processing speed and efficiency, YOLO performs better, and its performance improves with newer versions. Due to its suitability for making quick decisions, YOLO is often used in video processing or autonomous driving applications. To achieve faster processing efficiency, YOLO sacrifices some accuracy. On the other hand, Faster-RCNN is the slowest in image analysis. Although it is faster than its predecessors, RCNN and Fast RCNN, the complexity of the model and its high resource demands mean that it takes longer to analyze images to ensure accuracy. Research by W. Wu, Q. Li et al., 2020 [6] also highlights the differences between YOLO and Faster-RCNN. The efficiency of CNN in handling tasks depends on the added techniques and the specific CNN design for the task. This makes its image processing and analysis speed difficult to predict, but in general, it is faster than Faster-RCNN.

Model complexity indicates the capacity and resource requirements needed to drive a model. The complexity of CNNs is lower compared to YOLO and Faster-RCNN because CNNs have relatively simple architectures. This simplicity provides the best scalability, but it also means that adjustments can be more complex. The simpler architecture also implies lower demands on training hardware. YOLO strikes a balance between complexity and speed, performing well in real-world applications. YOLO's architecture can also be enhanced by adding techniques to achieve better speed and accuracy, as suggested in the research by W. Wu, Q. Li et al., 2020 [6]. Faster-RCNN is complex and resource-intensive, which contributes to its outstanding performance in challenging detection tasks.

All three models require large-scale annotated datasets, but Faster R-CNN has higher requirements and dependencies on the quality of the dataset. Both YOLO and Faster R-CNN need precise bounding box annotations, whereas CNNs, if used solely for classification tasks,

have relatively lower annotation requirements. This also highlights that the quality and diversity of the data are crucial components in training a model.

In conclusion, Faster R-CNN has the highest accuracy, while the YOLO model offers a more balanced approach. CNN, on the other hand, provides the best operability in model design. However, due to the operability of YOLO and CNN, they can also achieve faster speeds and higher accuracy with effective design improvements such as the proposed method in paper [6] and paper [7].

Model	Accuracy	Efficiency	Complexity	Data Requirement
YOLO	Good	High	Moderate	High
CNN	Normal	Low	Less	High
Faster-RCNN	Excellent	Very Low	Most	High

Table 2.1 Conclude Result in Proposed Solutions

Chapter 3
System Methodology/Approach

This section outlines the methodology and approach adopted in the project. Methodology refers to the structured framework or systematic process applied to guide research, project development, and problem-solving. It defines how tasks are organized, executed, and evaluated throughout the project. A suitable methodology ensures that each stage of development is approached consistently, leading to more reliable outcomes and better overall project management.

3.1Agile Development

Agile development is a software development methodology that emphasizes iterative progress, flexibility, and continuous improvement through short, manageable cycles known as sprints [11]. Unlike traditional linear approaches, Agile allows teams to adapt to change quickly, respond to ongoing feedback, and continuously refine their work. Each sprint typically includes planning, design, development, testing, deployment, and review phases, allowing for frequent reassessment and rapid evolution of the system or model being developed.

This methodology is particularly suitable for this project, which involves developing a machine learning model for electrical connector pin defect detection. The project requires frequent experimentation, dataset adjustments, and model tuning to enhance accuracy and performance. Agile's iterative approach aligns well with this need for continuous refinement. Each development cycle focuses on improving specific weaknesses—such as rotated or missing pin detection—and evaluating the results before planning the next improvement. The flexibility to update model parameters, modify data labelling, and deploy improved versions supports an efficient and structured development process.

The six main steps in Agile development used in this project are:

- Plan:

  Define the sprint goal, such as improving detection accuracy for specific pin defects.
- Design:

  Prepare the structure of the solution, including selecting model algorithms, defining labelling strategies, or adjusting preprocessing techniques.
- Develop:

Implement the design through coding, dataset preparation, and training processes, for example, training a new model version with augmented data.

- Test:

  Evaluate model performance using metrics such as accuracy, precision, recall, or F1 score, and compare results with previous versions.

- Deploy:

  Launch the improved version by saving the model, testing it on new datasets, or integrating it into the full inspection pipeline.

- Review:

  Analyze the results to determine what improvements were successful and identify areas for further enhancement, feeding into the next sprint cycle.

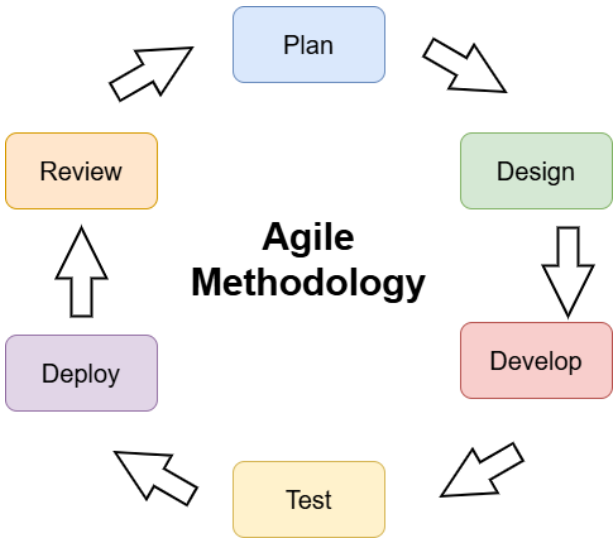


Figure 3.1.1 Agile Methodology

3.2 Dataset Strategy and Model Development

This section presents the core components of the system design, divided into two main parts: data preparation and model development. It explains the rationale behind using a custom-generated dataset and the selection of the YOLOv8 algorithm for object detection, both of which are essential to achieving the goals of this project.

3.2.1 Data Preparation

The collection of a custom dataset is necessary because this project addresses a highly unique case for which no publicly available datasets exist. As real-world data collection is not feasible in this scenario, the dataset is generated using image processing techniques. Although the data does not represent actual real-world cases, the method used introduces an innovative approach for addressing similar problems. This development demonstrates a novel way to simulate data for training purposes when real samples are difficult or impossible to obtain.

3.2.2 Model Development

YOLOv8 algorithm is selected for model training due to its strong learning capabilities and high performance in object detection tasks. YOLO (You Only Look Once) is a real-time object detection algorithm that performs both object classification and localization in a single pass through a neural network. It is widely recognized for its balance of speed and accuracy. YOLOv8, the latest version, offers further improvements in detection performance, flexibility, and deployment efficiency compared to earlier versions. These advantages make it a suitable choice for this project's requirements.

Compared to traditional object detection algorithms such as RCNN and other CNN-based methods, YOLOv8 demonstrates more comprehensive capabilities and faster inference times.

3.3 System Design Diagram

This section will present the logical and functional architecture of the system. It`s often broader and may include multiple views such as architecture, behavior, data flow, and interactions.

3.3.1 Use Case Diagram and Description

This section presents the use case diagram, which illustrates the various functions that can be performed by the developer. Since this project is focused solely on model development and is not intended to be deployed as an application, the only actor involved is the developer. The diagram highlights several key operations available to the developer, including

generating the dataset, annotating images, preprocessing data, configuring and training the model, evaluating its performance, and exporting the final trained model.

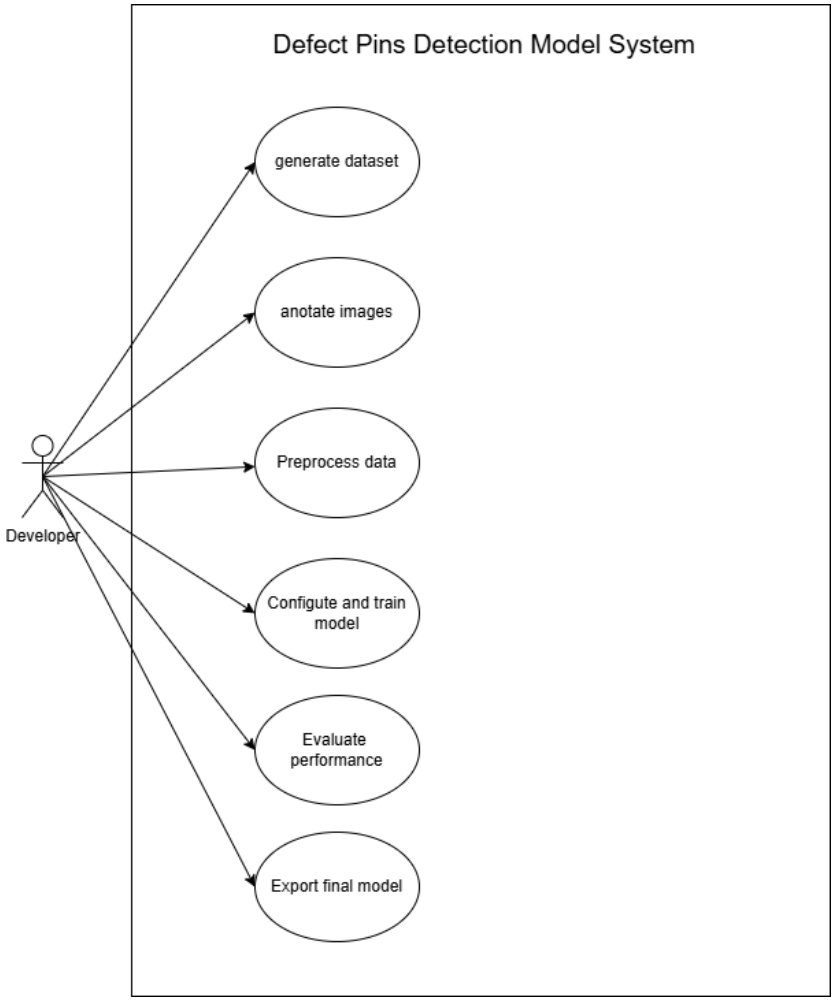


Figure 3.2.1 Use case Diagram

3.3.2 Activity Diagram

This section presents the activity diagram which illustrates the workflow of the project, starting from inserting raw images to model exploration. It shows how different defect types (missing, rotated, and shifted pins) are generated, followed by data augmentation, dataset creation, model training, evaluation, model combination and final exploration. The process ends once the model is fully explored.

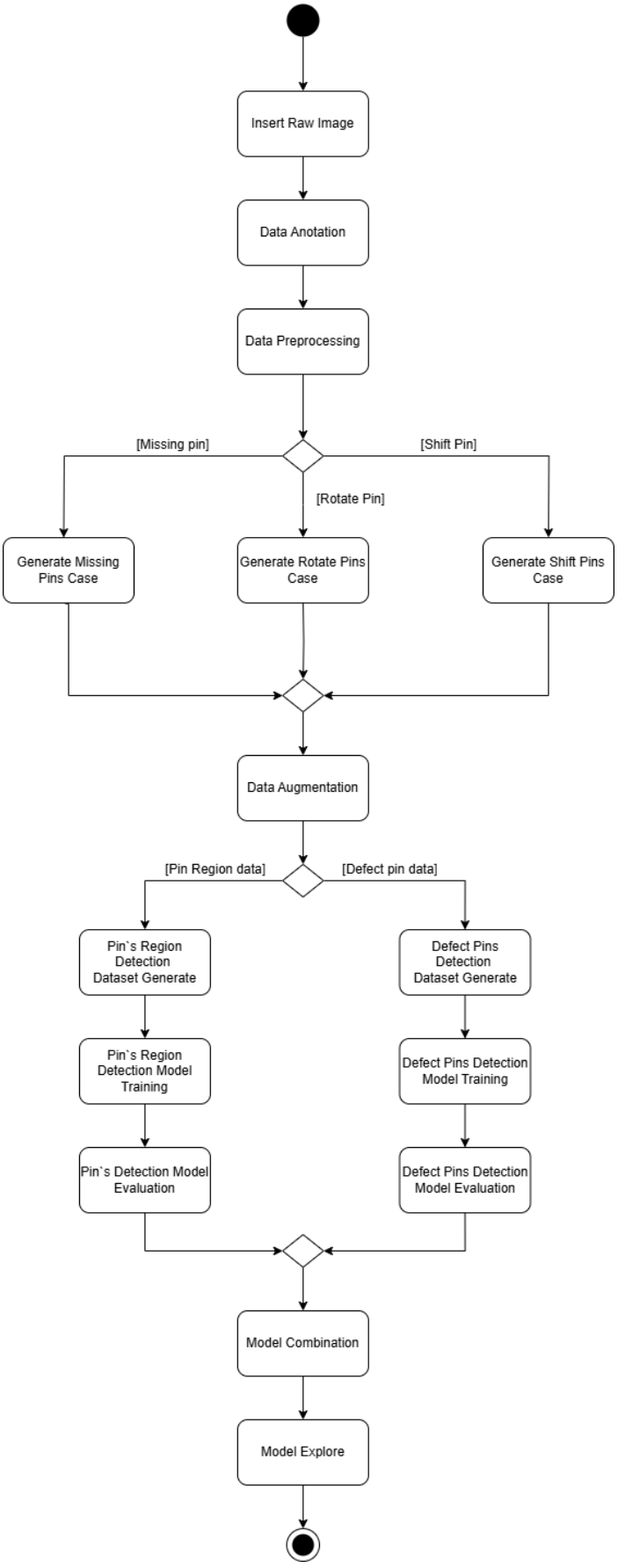


Figure 3.2.2 Activity Diagram

Chapter 4
System Design

To provide a clear and structured understanding of the system`s development, this chapter presents the system design through diagrams and design evolution. The system was not built in a single step, it underwent several updates as testing and evaluation revealed opportunities for refinement. These design versions reflect the effort to enhance system reliability, data flow, and model training efficiency.

4.1 System Iteration

4.1.1 Version 1: Initial Design

The first version of the system block diagram organizes the workflow into three main sections: Data Annotation (red zone), Data Preprocessing (green zone), and Trained Model Operation (blue zone). These sections represent the core operational structure of the project.

In the Data Annotation phase, pin positions are manually labelled to provide accurate reference points for model training. The Data Preprocessing phase generates various defect scenarios—such as missing, rotated, and shifted pins—by manipulating the raw images accordingly. These images, along with their corresponding label files, are stored in the training dataset to simulate real defect conditions.

The Trained Model Operation section outlines how the model behaves in practice. In Version 1, it includes a single-stage model focused on defect pin detection. After receiving a test image, the model detects pin positions and classifies defects, outputting the results to the user.

While this version establishes the essential logic and workflow, a key limitation emerged: the model often misinterpreted empty areas or background space as defect locations. This issue revealed the need for improved data handling and more robust preprocessing strategies, which informed the development of future versions.

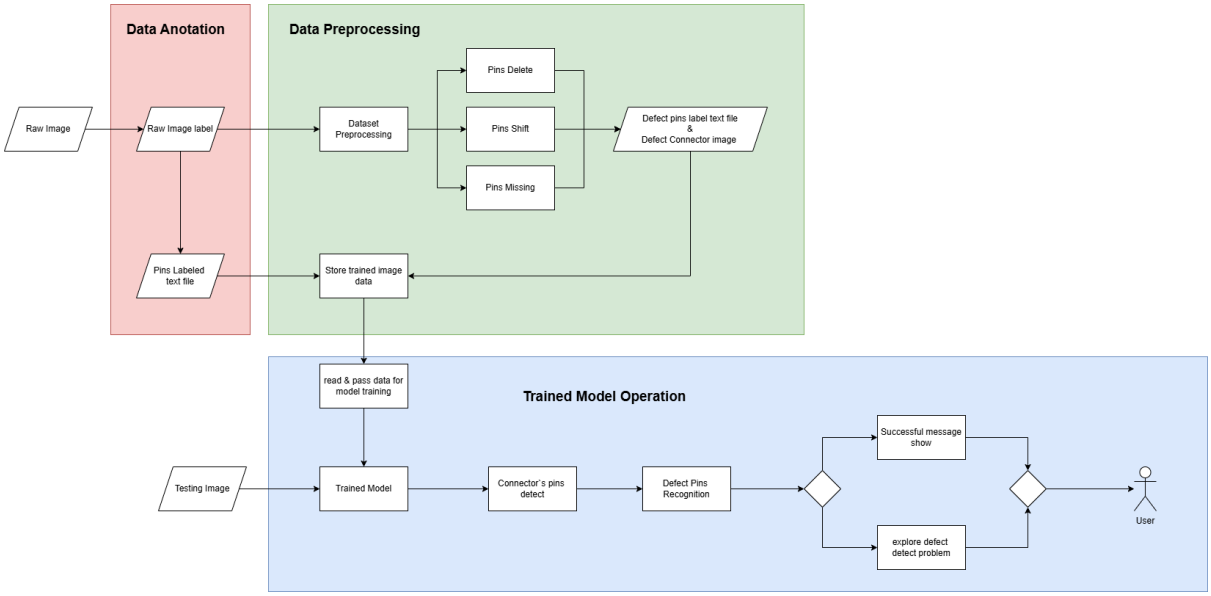


Figure 4.1.1 First Version System Flow Chart

4.1.2 Version 2

In the second version of the system, a new section is introduced: Dataset Generation & Model Training (purple zone). This version prepares two separate datasets—one for pin region detection and another for defect pin detection. The addition of a pin region detection model addresses the misdetection issue observed in the previous version, where the model frequently identified empty or background areas as defective pins.

This issue was likely caused by the YOLOv8 algorithm's limitations in detecting small objects. By incorporating a region detection model, the system first identifies the precise area containing the pins. The coordinates of this region are then passed to the defect pin detection model, allowing it to focus specifically on analyzing the relevant area, thus reducing false detections.

This version significantly improves labelling accuracy and minimizes incorrect detections. However, new challenges emerged during testing. When images were captured under variable lighting conditions—such as brightness shifts, reflections, or glare—the model's ability to detect defects declined. This highlighted a new limit that would be addressed in the next version.

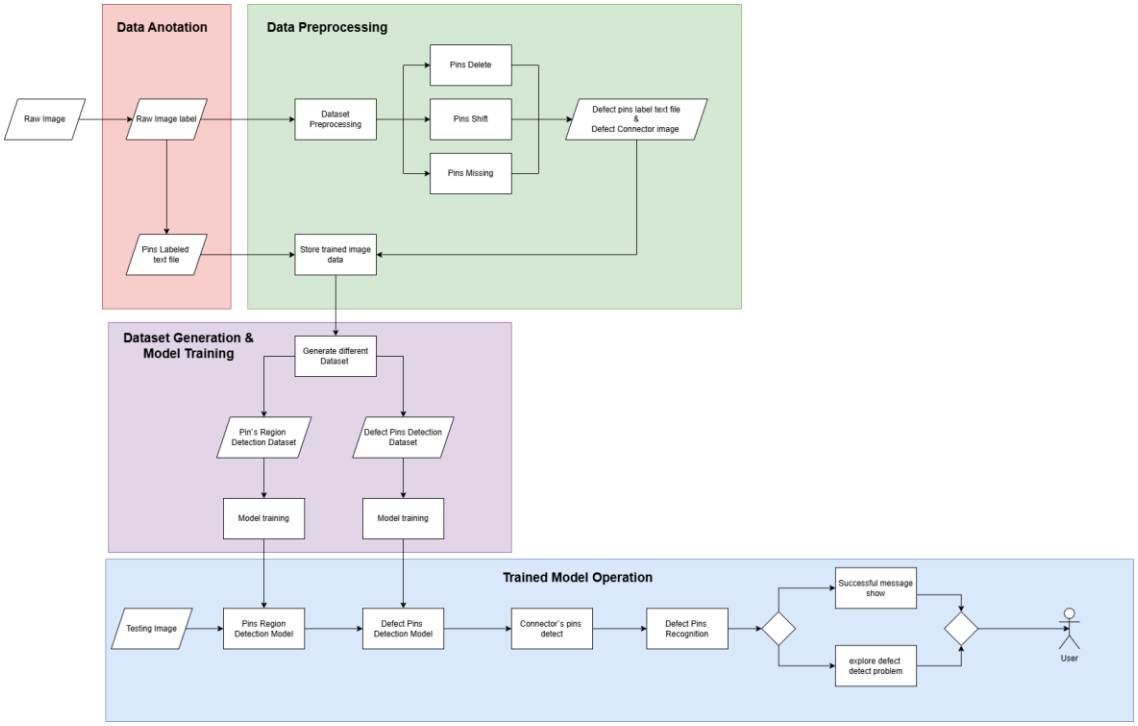


Figure 4.1.2 Second Version System Flow Chart

4.1.3 Version 3

In the third version of the system, an enhancement is introduced within the Data Preprocessing section (green zone). To address the challenge of detection failures under varying lighting conditions, this version incorporates a data augmentation step. This step generates additional training samples by simulating diverse visual conditions, such as brightness variation and reflections, to improve the model's robustness in real-world environments.

The introduction of data augmentation proved to be essential. As a result, both the pin region detection and defect pin detection datasets were expanded to include these augmented samples. This improvement significantly enhanced the model’s accuracy when applied to real-world images captured under different lighting scenarios.

However, this version still encountered a limitation: it could not reliably detect electrical connectors that appeared rotated in the image. Although adding rotated connector images to the dataset was considered, the wide range of possible rotation angles would introduce excessive variability. This would increase dataset complexity and potentially add noise during training, negatively impacting overall model performance. As such, the rotation issue is planned to be addressed more strategically in the following version.

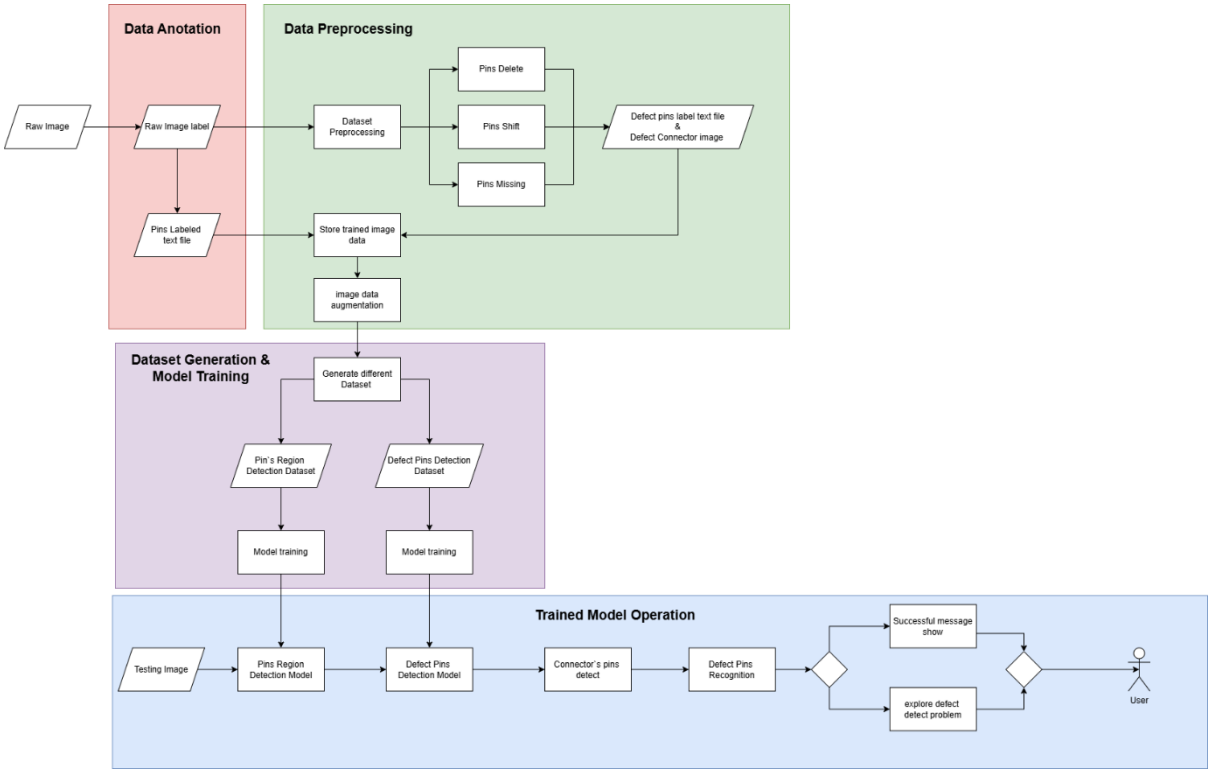


Figure 4.1.3 Third Version System Flow Chart

4.1.4 Final Version

This final version of the project addresses a crucial real-world challenge: in practical industrial environments, not all companies can guarantee that electrical connectors are positioned in a fixed orientation during defect detection. To solve this, a rotation correction function is introduced at the beginning of the Trained Model Operation section (blue zone). This function ensures the connector is aligned properly before being passed to the pin's region detection model, improving detection accuracy under various rotational angles.

The project explores two rotation correction solutions:

- Angle Brute-Force Rotation: The entire testing image is rotated in 10-degree increments. Each rotated image is then passed to the pin's region detection model, and the model's confidence score is recorded. The rotated image that produces the highest confidence is selected and forwarded to the defect pins detection model for further analysis. While this method improves reliability, it is inefficient as it requires up to 36 image rotations per test case.
- Edge-Based Alignment using Canny Detection: This solution uses edge detection to identify the connector's alignment. A horizontal reference line is applied, and the image is rotated until the connector aligns horizontally. The rotation degree is

accurate to two decimal places. This method is significantly more efficient and accurate, making it the preferred solution

Although the brute-force approach is computationally heavier, it serves as a fallback when the edge-based method fails. Together, these two methods ensure robust and reliable detection regardless of the connector's orientation.

The final version demonstrates a comprehensive and practical solution that effectively overcomes key challenges faced in real deployment scenarios. It enhances system reliability and adaptability, making it suitable for use in real company operations.

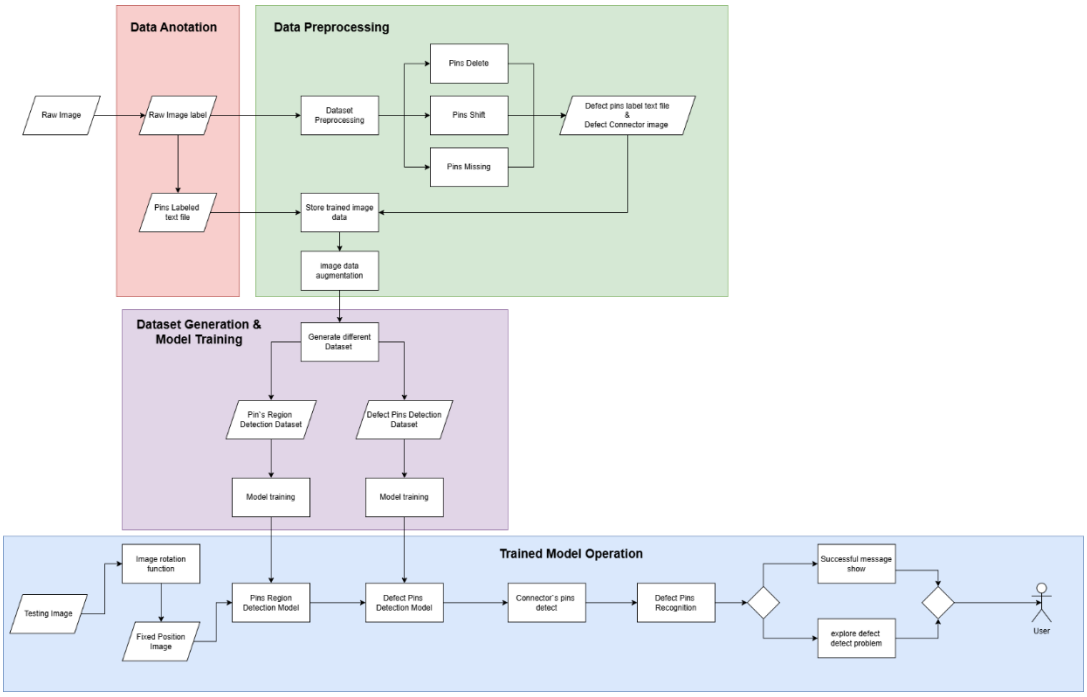


Figure 4.1.4 Final Version System Flow Chart

4.2 System Block Diagram

This section presents the core phases of the project's development process using a workflow diagram. The diagram outlines the fundamental stages involved, including data acquisition, data annotation, data pre-processing, data augmentation, dataset generation, model tuning, model training, model evaluation, and model exploration. Each phase is essential in shaping the model's foundation, providing a clear understanding of the technical workflow and the key operations carried out during the development of the defect detection system.

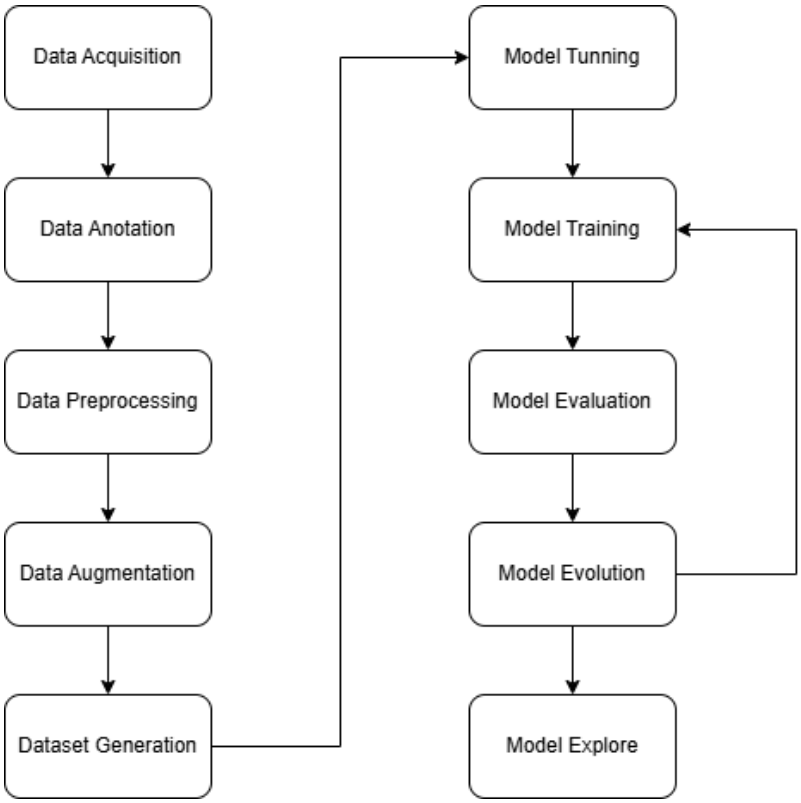


Figure 4.2.1 Project Methodology Diagram

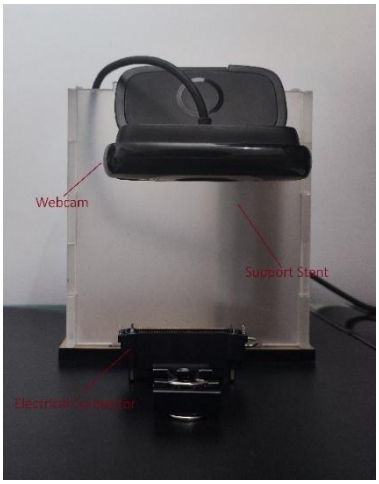
4.2.1 Data Acquisition

This part represents the project's initial step, where the raw electrical connector image will be captured by webcam show in Figure 4.2.2(a). After that, this raw image will be stored in a file. Webcam will be positioned at a consistent angle relative to the vehicle to ensure uniform environmental conditions, such as lighting, capture distance, and other factors. Figure 4.2.2(b) and Figure 4.2.2(c) shows this project`s camera setup. In this setup, the electrical connector is positioned on a black platform to minimize background noise and

enhance image clarity. The camera, measured from its lens, is placed at a height of 7.1 cm above the platform. The pins of the electrical connector are elevated 1.6 cm above the platform, resulting in a 5.5 cm distance between the camera lens and the pins. This precise configuration ensures consistent image quality and accurate pin detection, providing optimal conditions for both dataset creation and model training.

(a)

(b)



(c)

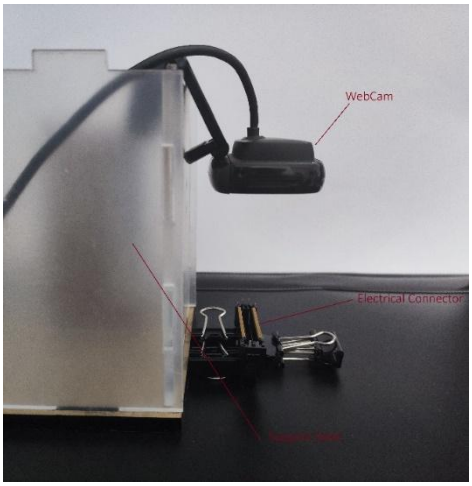


Figure 4.2.2 (a) Logitech Webcam, (b) Front View of Webcam Setup, (c) Side View of Webcam Setup

4.2.2 Data Annotation

This step focuses on labelling images by placing bounding boxes around the pins of the electrical connector. To facilitate this process, the OpenCV library is employed, providing precise tools for defining and annotating pin locations efficiently. The bounding box data, including coordinates and dimensions, is systematically stored in a text file for further processing.

There are two different labelling methods applied to address different problems and model training requirements.

The first labelling method involves labelling the entire pin region for the dataset used to train the pin region detection model. In this method, the entire pin region is labelled with a single bounding box, as illustrated in Figure 4.2.3(a). The position is recorded in the YOLO format, for example: (0 0.535156 0.722917 0.629687 0.229167), where the values represent the class and normalized center coordinates, width, and height of the bounding box.

The second labelling method involves labelling each pin individually, with one bounding box per pin. This approach is used for the defect pin generation and defect pin detection model training. The electrical connector in this project consists of 120 pins, with 60 pins in both the upper and lower rows. Each pin's labelled data is recorded in a format such as: Pin1: (149,304) to (155,323), ensuring precise localization for subsequent model training, as shown in Figure 4.2.3(b).

This meticulous labelling process is crucial for producing a high-quality dataset, significantly enhancing the model's ability to detect and classify defects with greater accuracy and reliability.

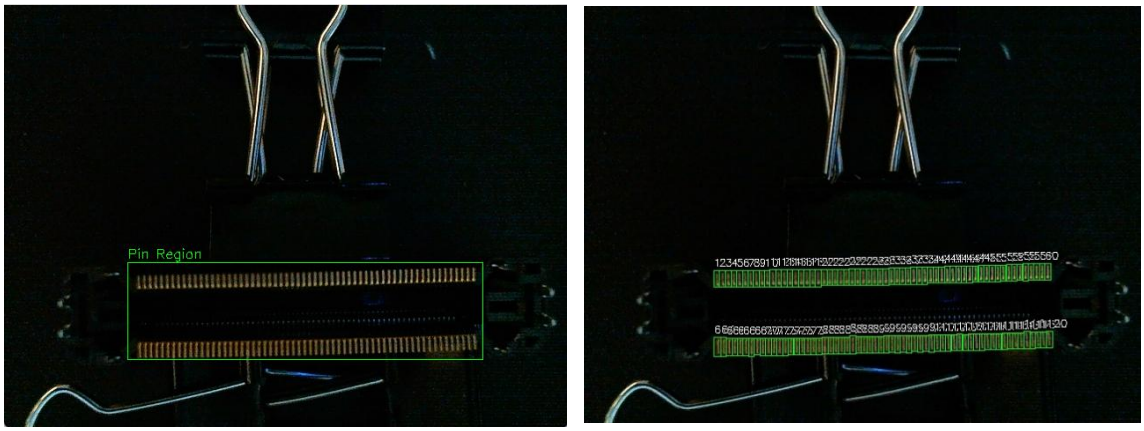


Figure 4.2.3 Labelling Pins Show (a) Pins Region Labelling (b) Pin per 1 box

Figure 4.2.4 illustrates the codding logic workflow. After the raw image is processed through the *PinLabeler()* function, the program presents an interactive interface where users can draw rectangles to label the pins. The labelling is achieved by recording the x and y coordinates of the rectangle's starting and ending points, with the rectangles drawn using

a click-and-drag method. The *PinLabeler()* tool also offers various support features, such as zoom and undo functions, to enhance the labelling accuracy and efficiency. Once all pins are labelled and the process is confirmed and saved, a text file containing the pin position data is generated.

For the pin region labelling process, the same technique used for individual pin labelling is applied. After manually labelling the pin region in the image, the corresponding label text file is passed into the *convert_to_yolo_format()* function. This function converts the annotated data into YOLO format, resulting in a new text file that aligns with the required input structure for training the YOLOv8 model.

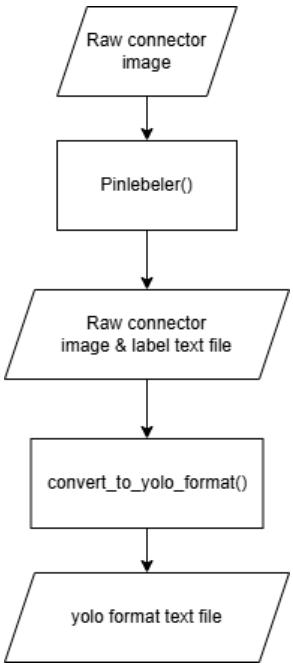


Figure 4.2.4 Label Function Code Workflow

4.2.3 Data Preprocessing

In this step, various OpenCV functions are utilized to generate a defect electrical connector dataset. Building upon the previous step, where the positions of 120 pins were labelled from the original image, this project will manipulate these positions to create new layers. By altering, rotating, or removing these layers, defect pin data is generated and collected.

Background Removing

To avoid image distortion caused by manipulating small layers on the original image, the project employs a technique that removes all areas outside the 120 labelled bounding boxes, replacing them with a black background and retaining only the pins of the electrical connector such like figure 4.2.5 shows.

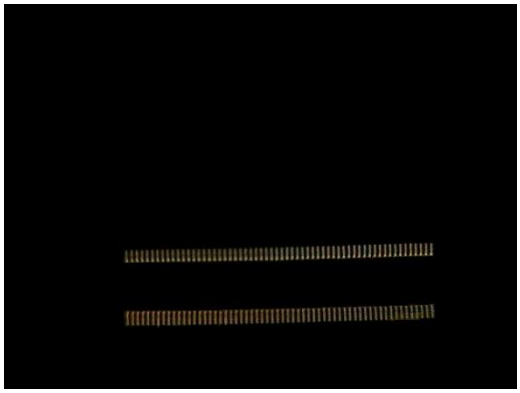


Figure 4.2.5 Background Remove Result Show

Figure 4.2.6 illustrates the workflow of the background removed. Initially, the *read_pins_position()* function identifies the labeled positions of the connector's pins. The main function, *apply_black_background*, then utilizes *np.zero_like()* to convert all non-labeled areas to black. This function is necessary because it can avoid the distortion problem at the next step.

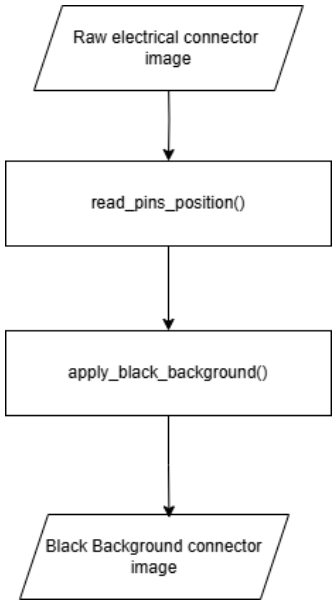


Figure 4.2.6 Background Remove Workflow

4.2.4 Defect Pins Generation

Three types of defective connector pins are created and generated for this project: misaligned pins, rotated pins, and missing pins. These defect types were chosen because they are among the most observed in real-world scenarios. By including these variations in the dataset, the model can be trained to recognize and handle a wider range of potential defects, thereby improving its reliability and applicability in practical inspection tasks.

Figure 4.2.7 illustrates the workflow for generating defective pins. The process begins by inputting key parameters, such as the number of pins to be modified and the total number of images to be generated. It starts with the *loadImage()* function, which loads the background-removed images. This is followed by the *read_pin_position()* function, which reads and records the labelled positions of the pins from the existing annotation files. These initial steps set the foundation for applying controlled modifications to simulate different types of pin defects.

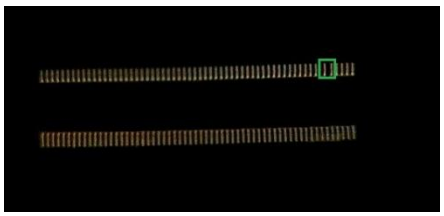
For missing pin generation, the *generate_missing_pin()* function is used, where random pins are selected and their pixel values are set to (0, 0, 0), effectively making them disappear. This simulates missing pins. The adjusted data is then processed by *batch_generate_missing()* to create the defect pin images and log their positions in a text file.

For rotated pin generation, *the generate_rotate_pin()* function is invoked, which randomly selects pins to be marked as defective. Each selected pin is rotated by an angle between 5 and 15 degrees to simulate realistic misalignment. This rotation is performed using OpenCV functions: cv2.getRotationMatrix2D to calculate the transformation matrix and cv2.warpAffine to apply the rotation. After rotation, the adjusted images are processed by the *batch_generate_rotate()* function, which generates the final defective pin images and logs their updated positions in a corresponding text file.

For shifted pin generation, the process triggers the *generate_shift_pin()* function, which uses a random function to select pins for displacement. Given that the distance between electrical connector pins is approximately 0.5mm, any minor deviation is critical. Therefore, in this project, the shift value is set to exceed 1 pixel (approximately 0.12mm). The selected pin's position is adjusted by adding 1 to its coordinates. The *batch_generate_shift()* function then generates defect pin images and records the shifted pin positions in a text file. At Figure 4.2.8 showcases the results for each type of defect pin generation.

Figure 4.2.7 Defect Pins Generation Workflow

(a)



(b)

(c)

Figure 4.2.8 Defect Pins Result Show

4.2.5 Data Augmentation

In this step, the project ensures that the dataset already includes three types of categories within the dataset file. The Albumentations library is integrated to assist with image augmentation. Albumentations is recognized as one of the most powerful libraries for performing data augmentation. The objective of applying data augmentation is to generate a more diverse and robust dataset for model training.

In this project, five augmentation parameters are modified, which are as follows:

1. RandomBrightnessContrast: Adjusts the brightness and contrast of the image.
2. GaussNoise: Adds Gaussian noise to simulate sensor noise or environmental interference.
3. MotionBlur: Applies a slight motion blur to simulate movement during image capture.
4. Perspective: Applies a perspective shift to simulate changes in the camera angle.
5. ColorJitter: Modifies the colors to enhance dataset variability.

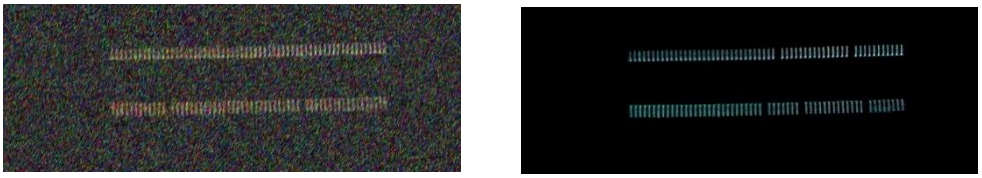


Figure 4.2.9 Augmented Result Show

Figure 4.2.10 illustrates the augmentation workflow. The *Transform()* function uses the Compose method to apply a series of augmentation parameters, as previously described. When a defective pin image undergoes augmentation, its corresponding label file is also updated to reflect any changes in pin positions caused by transformations such as scaling or flipping. This ensures that the dataset remains accurate and consistent for training purposes.

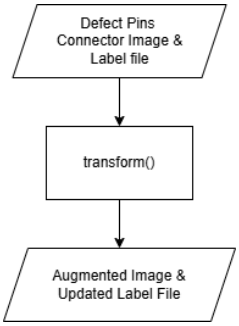


Figure 4.2.10 Augmentation Workflow

4.2.6 Dataset Generate

After the data augmentation process is completed, the project generates and modifies two dataset files. These two datasets are used for training with the YOLOv8 algorithm model. Both datasets are organized in the standard YOLO format, where each contains separate folders for images and labels, and each folder is divided into training and validation sets, as illustrated in Figure 4.2.11.

The first dataset is the pins region detection dataset, which is used to train the pins region detection model. This dataset contains a total of 2000 images, with 1600 images allocated for training and 400 images allocated for validation. The dataset is evenly divided between 1000 real-world images and 1000 augmented images. This distribution is intentional, as real-world images are prioritized to ensure the model performs well under practical conditions, while the augmented images are used to strengthen the model's robustness.

The second dataset is the defect pins detection dataset, which is used to train the defect pin detection model. This dataset consists of a total of 3600 images, with 3000 images used for training and 600 images used for validation. Like the first dataset, it is balanced with half real-world images and half augmented images.

The reason the defect pins detection dataset requires more data than the pins region detection dataset is due to the higher complexity of the defect detection task. Detecting individual defects demands the model to learn finer details and variations, thus necessitating a larger and more diverse dataset to achieve better performance.

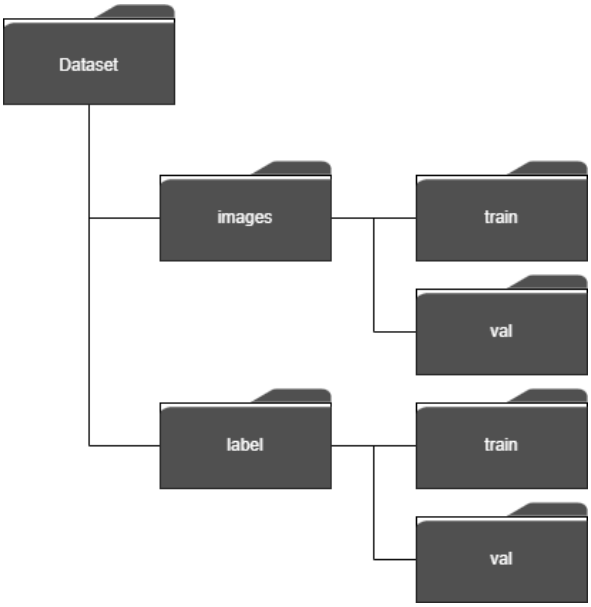


Figure 4.2.11 Dataset File Structure

4.2.7 Model Tunning

In this project, the YOLOv8 algorithm is selected for model training due to its strong learning capabilities and high performance in object detection tasks. However, one known limitation of YOLOv8 is its reduced accuracy when detecting very small objects, such as individual pins on an electrical connector.

To address this challenge, the project adopts a dual-model training approach. The first model is designed to detect the overall pin region, while the second model is trained specifically to identify defective pins. This separation of tasks enhances detection accuracy and ensures better performance when dealing with small-scale features in high-resolution images.

4.2.8 Model Training

This step involves training the model by adjusting various parameters. These training parameters are key factors that directly influence the model's ability to produce accurate detection results. In this project, the modified parameters include epochs, batch size, and image size (imgsz). The epoch defines how many times the entire training dataset is passed through the model during training. A higher number of epochs allows the model to learn more deeply but may increase the risk of overfitting. The batch size determines how many

images are processed at once during each training step, affecting both training speed and memory usage. The image size (imgsz) refers to the dimensions to which all input images are resized before being fed into the model, ensuring consistency and improving computational efficiency.

For the pins region detection model the train parameter is set like:

1. Epoches = 50
2. Batch = 16
3. Imgsz = 640

For the defect pins detection model the train parameter is set like:

1. Epoches = 50
2. Batch = 16
3. Imgsz = 640

These three parameters serve as the basic startup configuration, while other training parameters will be further explored and optimized during the model evolution phase. Figure 4.2.12 illustrates the model training workflow. Both models follow the same training logic. An essential component in YOLO-based training is the YAML file, which defines key configuration details such as the dataset directory and the category classes. For instance, in a YOLO-formatted annotation like (0 0.1111 0.1222 0.123 0.1444), the first number 0 indicates the class label, followed by normalized values for the bounding box's center coordinates, width, and height. After importing the YOLO model, training can be initiated using the *model.train()* function, which handles the training process based on the provided dataset and configuration.

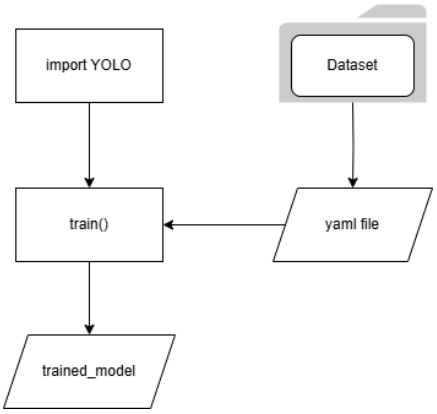


Figure 4.2.12 Model Training Workflow

4.2.9 Model Evaluation

This step involves setting performance metrics to evaluate the effectiveness of the trained model. In this project, the selected evaluation metrics include precision, recall, and mean average precision (mAP), specifically mAP@50 and mAP@50–95. Precision measures the proportion of correctly identified positive predictions out of all positive predictions made by the model, reflecting the model's accuracy in detecting true positives without producing false alarms. Recall indicates the proportion of actual positive cases that were correctly identified by the model, showing how well the model captures relevant instances. Mean Average Precision (mAP) is a comprehensive metric that combines both precision and recall across different thresholds. mAP@50 calculates average precision using an Intersection over Union (IoU) threshold of 0.5, while mAP@50–95 provides a more detailed evaluation by averaging the results over multiple IoU thresholds ranging from 0.5 to 0.95 in steps of 0.05. These metrics collectively offer valuable insight into the model's performance and help determine whether further adjustments to training parameters are necessary to improve results.

4.2.10 Model Evolution

Based on the performance results from the evaluation phase, this step focuses on adjusting the model's hyperparameters until the desired performance is achieved. In addition to the initial parameters, more advanced settings are introduced, such as the learning rate parameters (lr0 and lrf), which control the learning rate schedule throughout training. These learning rate values play a crucial role in determining how quickly or gradually the model learns from the data. This iterative process of tuning and refinement allows for continuous improvement of the model's performance, ensuring it becomes more accurate and reliable with each training cycle.

4.2.11 Model Explore

This final step confirms that the parameter adjustments have been identified, and the model has matured to meet the predefined objectives. At this stage, the model is fully optimized, demonstrating the required accuracy and efficiency, thus validating its readiness for real-world deployment in electrical connector defect detection.

4.3 Models Combination

The previous section presented the logic and workflow behind dataset preparation and the development of each individual model. Building upon that, this section outlines the integration logic and workflow for combining the pins region detection model with the defective pins detection model. This combined approach allows the system to first localize the pin region and then perform detailed analysis to identify specific defects within that region, enabling a more accurate and efficient inspection process. Figure 4.3.1 illustrates the model combination workflow, highlighting the specific functions used to execute the integration between the two models.

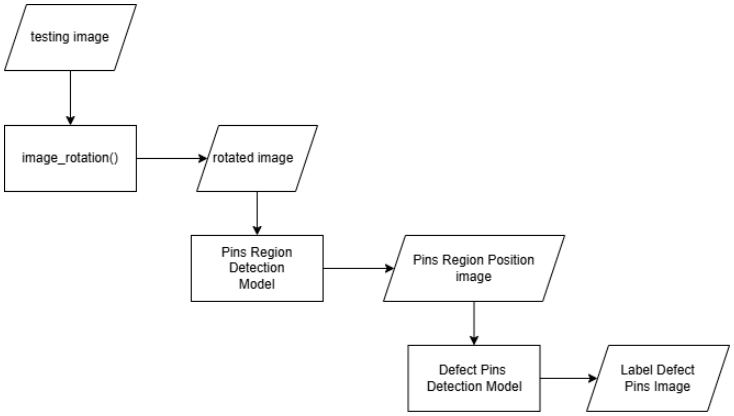


Figure 4.3.1 Model Combination Workflow

Rotation Function

The reason for incorporating a rotation correction function has been previously discussed in the system iteration section. Two rotation correction methods were explored: brute-force angle rotation and edge-based alignment using Canny edge detection. In most cases, the system adopts an edge-based alignment method due to its higher efficiency and accuracy in aligning the connector horizontally. However, the brute-force rotation approach serves as a fallback solution when the edge-based method fails to detect suitable contours or cannot perform the alignment correctly.

The *rotate_to_horizontal()* function takes an image input and optionally applies Canny edge detection. It begins by converting the image to grayscale if it's not already. To reduce noise and improve edge detection accuracy, a Gaussian blur is applied. Then, Canny edge detection is used to extract the edges from the image, helping to outline objects or regions of interest.

Next, contours are identified from the edge-detected image. If no contours are found, the function returns the original image and its processed versions to maintain consistency in output structure.

If contours exist, the function selects the largest contour, assuming it represents the main object in the image. It calculates the minimum area bounding rectangle around this contour and extracts the angle of rotation. If the rectangle is taller than it is wide, 90 degrees is added to the angle to ensure horizontal alignment.

Using this angle, a rotation matrix is created, and the image is rotated around its center. The new width and height of the image are calculated to accommodate the entire rotated image without cropping. Finally, OpenCV's *warpAffine()* function is used to apply the rotation, and the rotated image along with the rotation angle is returned.

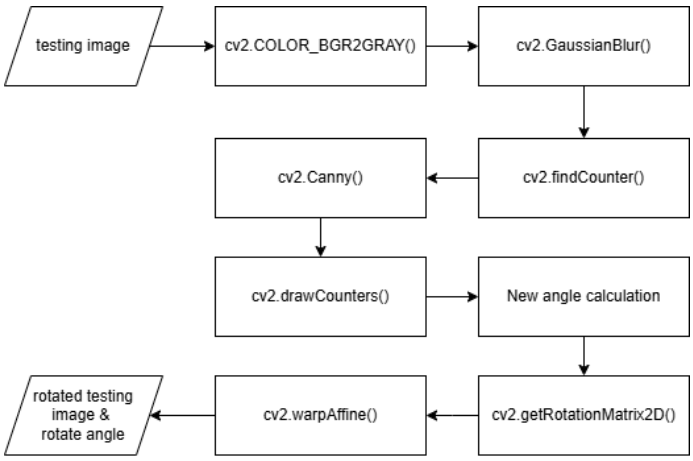


Figure 4.3.2 Rotation information workflow

Combination

The logic behind the integrated detection system is as follows: when a rotated image is passed to the pins region detection model, the system first checks whether the model can successfully detect the pins region. If detection is successful, the position of the pins region is recorded. On the other hand, if the model fails to detect the region, the system notifies the user with the message: "Detection failed. Please make sure your image is of an electrical connector." This workflow is illustrated in Figure 4.3.3.

Once the pins region is successfully detected, the system maps the position data and passes it to the defect pins detection model. If the defect detection model identifies any defective pins, it returns a labeled image showing the defects. Otherwise, it displays the message: "No defective pin detected." Since the pins region detection model has already verified that the input image is an electrical connector, this ensures that all images passed into the defect detection stage are valid connector images, improving the reliability of the final output.

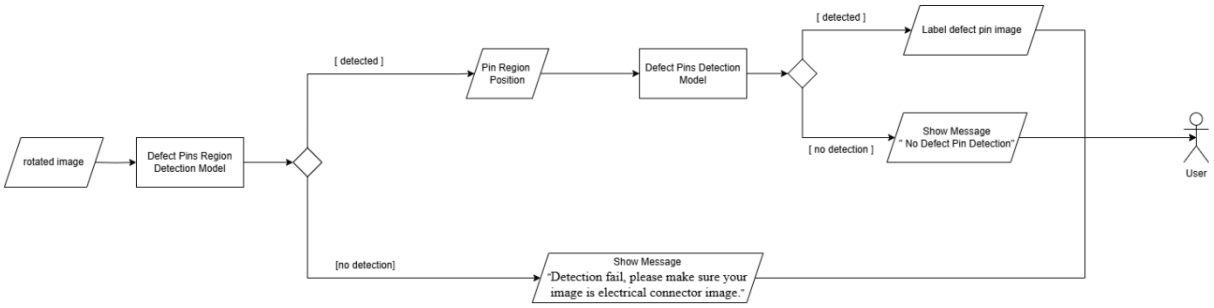


Figure 4.3.3 Model Combination Detail Workflow

Chapter 5
System Implementation

This chapter outlines the system implementation, covering all components involved in the development. It includes details on the hardware setup, software configuration, system operation, and challenges encountered during implementation.

5.1 Hardware Setup

Hardware is the important part in develop and implement our proposed system. There are two hardware which is laptop and webcam. Laptop will be responsible for model development and generating defect images through image processing. It will serve as the primary platform for training and fine-tuning the model, as well as utilizing image processing techniques to simulate various defect scenarios in electrical connector pins, ensuring the creation of a comprehensive and high-quality dataset for model training and validation. Table 3.1 shows the specifications of laptop.

Description	**Specifications**
Model	MSI Katana GF66
Processor	Intel Core i7-12700H
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 3050 Ti
Memory	32.0GB RAM
Storage	512GB

Table 5.1 Specifications of laptop

The webcam will play a crucial role in capturing both the raw images and testing images for this project. Its ease of setup ensures consistent positioning and angle throughout the image acquisition process. This stability enhances the efficiency and accuracy of image extraction, contributing to the creation of a reliable dataset and ensuring uniform conditions for model testing and validation. Table 3.2 shows the specifications of webcam.

Description	Specifications
Model	Logitech Webcam C615
Resolution	HD 1080P

Table 5.2 Specifications of webcam

5.2 Software Setup

This section presents the software and libraries utilized in this project to support the development process.

I. Visual Studio Code (version 1.95)

A popular and free code editor developed by Microsoft. It offers a powerful and versatile coding environment that can be used for a variety of development tasks.

II. Jupyter Notebook

Free software, open standards, and web services for interactive computing across all programming languages. It provides cloud

III. OpenCV 4.0

OpenCV, short for Open-Source Computer Vision Library, is an open-source computer vision and machine learning software library.

5.3 System Operation

This section presents how the project performs model performance evaluation, outlining the steps and procedures used to test the model's functionality and reliability within the system such as Figure 5.3.1 show.

```
# Example usage
if __name__ == "__main__":

    model_1 = YOLO("runs/detect/yolo_pins_detection_2/weights/best.pt")
    model_2 = YOLO("runs/detect/yolo_defect_detection6/weights/best.pt")

    # Replace with your image paths
    image = cv2.imread("images/captured_image_8.jpg")  # Replace with your input image path
    output_path = "images/rotated_image.jpg"  # Replace with your desired output path


    try:
        rotated_image, angle = rotate_to_horizontal(image, use_canny=False)
        print(f"Image rotated by {angle:.2f} degrees")

        result_1 = model_1(rotated_image,conf=0.5, save=True)

        save_dir = "runs/detect/pin_defet_detections"
        os.makedirs(save_dir, exist_ok=True)

        for result in result_1:
            for box in result.boxes:
                x1,y1,x2,y2 = map(int, box.xyxy[0])

                pin_region = rotated_image[y1:y2 , x1:x2].copy()

                results_2 = model_2(pin_region,conf=0.5,save=True, show=True)

                for defect in results_2:
                    for d_box in defect.boxes:
                            x1_def,y1_def,x2_def,y2_def = map(int,d_box.xyxy[0])
                            cv2.rectangle(pin_region,(x1_def,y1_def),(x2_def,y2_def),(0,0,255),1)

                save_path = os.path.join(save_dir,f"pin_{x1}_{y1}.jpg")
                cv2.imwrite(save_path,pin_region)
```

Figure 5.3.1 System Operation

5.4 Implementation of Issues and Challenges

This section describes the implementation challenges and issues encountered during the project. Since the primary objective of this work is model development, the main goal is to deliver highly accurate and efficient inspection models that can be adopted by the company. However, several challenges were faced throughout the development process, including:

I. Captured Camera Ability

The camera plays a critical role in the operation of the entire system, as image capture marks the beginning of the inspection process. In this project, an HD 1080P resolution camera is used as the minimum requirement to ensure the model can accurately detect and analyze the tiny electrical connector pins. Since these pins are very small, low-resolution or blurry images can severely hinder the model's ability to recognize pin

positions or identify defects. As a result, the clarity, sharpness, and overall quality of the captured images have a direct impact on the system's accuracy and reliability.

This dependency on camera quality introduces both hardware and performance-related challenges. If the camera used is of poor quality, suffering from issues like motion blur, poor focus, or inadequate lighting sensitivity—the model's performance will be compromised, reducing overall system stability. Conversely, using a high-end camera that captures images rapidly can create a different problem: the model may not be able to keep up with the high input rate, resulting in data processing bottlenecks. In such cases, images may queue up, causing input jams and slowing down the inspection process. This highlights the need to balance camera capability with the model's processing speed to maintain smooth and efficient system operation.

II. Model Implementation Device

The computing device used for running the inspection system is another critical factor that directly influences the model's performance and overall system efficiency. Since the deep learning models used in this project involve complex computations, especially during inference and training—a device with sufficient processing power is essential. Systems equipped with a GPU are strongly preferred, as they significantly accelerate the model's performance, allowing for faster image processing and real-time defect detection.

Without adequate hardware, particularly in terms of graphics processing capability, the system may experience delays or reduced accuracy due to longer inference times. This becomes a bottleneck when processing high-resolution images or when handling large volumes of image data, especially in environments where rapid inspection is necessary. Therefore, to ensure the reliability and responsiveness of the system in practical deployments, the model must be implemented on a machine with appropriate computational resources, ideally one that includes a modern GPU.

III. System Combination with Inspection Model

Since this project focuses primarily on model development, integrating the trained models into a complete, functional system presents its own challenges. One key issue is ensuring compatibility between the developed models and other system components, such as the user interface or backend services. Differences in software versions, model formats, or runtime environments can cause unexpected errors or failures during integration.

For example, the user interface must be designed to support the input and output formats of the model, including image uploads, prediction visualizations, and real-time feedback. Additionally, the model's dependencies and framework versions must align with the system's runtime environment to avoid compatibility issues. These integration challenges highlight the importance of coordinated development between the model and system components to ensure seamless operation in real-world applications.

5.5 Concluding Remark

In summary, this chapter presented the complete implementation of the electrical connector inspection system, including both the hardware and software components. The integration of camera capture, image processing, and machine learning models was successfully executed to support automated defect detection. Despite some challenges related to hardware limitations and system integration, the implementation provided a solid foundation for real-world application. The outcomes of this chapter demonstrate that the system is functional, and its performance can be further improved through hardware optimization and model fine-tuning in future work.

Chapter 6
System Evaluation and Discussion

This chapter outlines the system evaluation, presenting the performance and results of the trained model. In addition, it highlights the challenges encountered during the model development process.

6.1 Model Testing & Performance Metrics

This section outlines the performance metrics used to evaluate the detection model in this project. The goal is to assess the model's accuracy, effectiveness, and overall capability. Several standard metrics are applied during model training and evaluation, including:

- Precision:

  Precision measures the accuracy of positive predictions. It is the ratio of correctly predicted positive observations to the total predicted positives. A high precision indicates that the model returns more relevant results than irrelevant ones.
- Recall:

  Recall evaluates the model’s ability to find all relevant instances. It is the ratio of correctly predicted positive observations to all actual positives. A high recall means the model detects most of the true cases.
- F1 Score:

  The F1 Score is the harmonic means of precision and recall. It balances the two metrics and is useful when you need a single measure that accounts for both false positives and false negatives.
- mAP50:

  mAP@50 calculates the average precision across all classes with a fixed Intersection over Union (IoU) threshold of 0.50. It provides a general sense of how well the model detects objects at a moderate level of overlap.
- mAP50-95:

  This metric is a more comprehensive version of mAP, averaging the AP over multiple IoU thresholds (from 0.50 to 0.95 in steps of 0.05). It offers a deeper insight into the model’s performance across various levels of detection difficulty.

Pins Region Detection Model

Following this, the performance of the pin region detection model is discussed. In this project, two versions of the model were developed: one trained without data augmentation and another trained with augmented data.

```
Precision: [    0.99987]
Recall: [           1]
F1 Score: [    0.99994]
mAP50: 0.995
mAP50-95: 0.9949999999999999
```

Figure 6.1.1 Region Detection V1 - Model Performance Metrics

Figure 6.1.1 illustrates the performance metrics for the non-augmented model. This version demonstrates high scores in precision, recall, F1 score, mAP50, and mAP50-95, indicating strong detection performance. The reason behind this result is likely due to the simplicity of the dataset, where all validation images closely resemble the training images—consisting of black backgrounds and clearly visible pins. Detecting the entire pin region in such uniform conditions is relatively straightforward, allowing the model to achieve high accuracy.

Figure 6.1.2 Region Detection V1 - Training vs Validation Box Loss Curve

Figure 6.1.2 presents the training versus validation box loss curve, which is used to assess the model's learning behaviour and identify potential overfitting or underfitting. Both the training

and validation losses exhibit a consistent downward trend throughout the epochs, indicating that the model is learning effectively. Although the validation loss shows some fluctuations between epochs 10 to 30, this is expected due to the variability in unseen data. The overall low loss values and the close alignment between the training and validation loss curves suggest that the model generalizes well to the validation set.

These results demonstrate a promising direction for the project. However, the strong performance is currently limited to simple and controlled scenarios. Therefore, in the subsequent model version, efforts were focused on expanding the dataset to better reflect real-world cases and increase model robustness under more diverse conditions.

```
Precision: [    0.99987]
Recall: [           1]
F1 Score: [    0.99993]
mAP50: 0.995
mAP50-95: 0.9949184725626127
```

Figure 6.1.3 Region Detection V2 - Model Performance Metrics

Figure 6.1.3 illustrates the performance metrics of the augmented version of the pin region detection model. These results indicate that the model retains a strong ability to detect pin regions, despite the increased complexity of the validation dataset. In this version, the validation images include challenges such as blur and visual noise. As a result, a slight decrease in performance metrics is observed when compared to the non-augmented version (Version 1). However, the metrics still demonstrate that the model performs reliably under more realistic conditions.

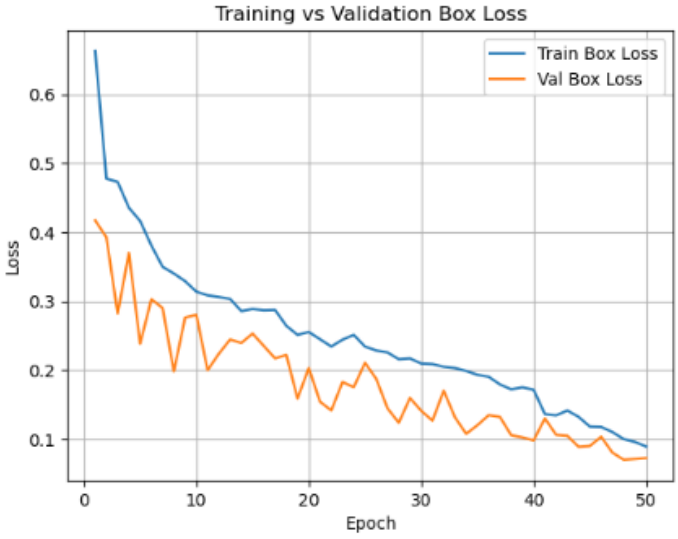
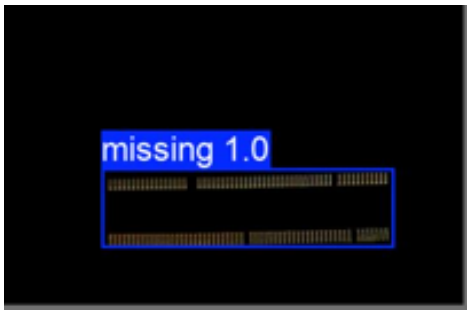


Figure 6.1.4 Region Detection V2 - Training vs Validation Box Loss Curve

Figure 6.1.4 presents the training versus validation box loss curve for the augmented model. The training and validation loss curves both show a consistent downward trend, with the validation loss exhibiting more fluctuation due to the increased complexity of the dataset. These fluctuations are considered normal and indicate the model is learning to generalize across a broader range of real-world variations.
Given its improved robustness in handling real-world cases, this model will be selected for deployment in the following system stages.

Defect Pins Detection Model

For this model, there are three versions of the defect pin detection model that were developed, using two different labelling strategies. The first strategy involved labelling the entire pin region and classifying it as missing, rotated, or shifted, as shown in Figure 6.1.5(a). The second strategy involved labelling only the specific defective area of the pin, illustrated in Figure 6.1.5(b). Two models were trained using this second strategy: one without data augmentation and one with augmented data.

(a) 

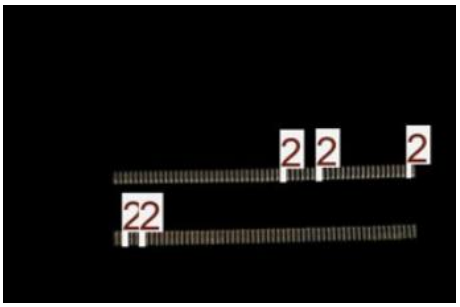
(b) 

Figure 6.1.5 Label Methods (a) Full Region (b) Specific

The first model version, which used full region labelling, demonstrated very poor performance. As shown in Figure 6.1.6, the confusion matrix indicates extremely low accuracy, with the model only showing capability in detecting rotated defect cases. It failed to identify rotated and shifted pins effectively. This limitation revealed that the full region labelling method does not offer sufficient information for the model to learn precise defect patterns. Consequently, changing the labelling strategy was necessary.

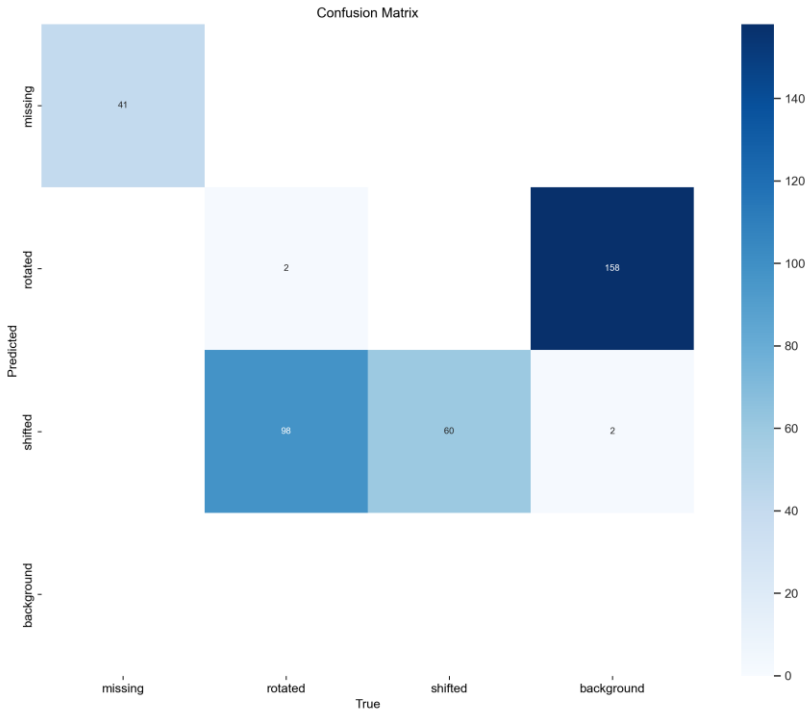


Figure 6.1.6 Defect Detection V1 - Confusion Matrix

The second model version applied specific defect area labelling and demonstrated significantly improved results. Figure 6.1.7 displays its performance metrics, showing separate results for missing, rotated, and shifted defects. These metrics indicate strong performance when tested on clean, controlled images. However, as reflected in the confusion matrix (Figure 6.1.8), some incorrect predictions occurred, likely due to the model misclassifying parts of the background as defect areas. Despite this, the model successfully recognized distinct defect categories with high confidence.

The training versus validation loss curve, shown in Figure 6.1.9, confirms that the model did not experience overfitting. Both curves exhibit a stable downward trend, which suggests that the model is learning effectively and generalizing well on unseen validation data. This version, like the earlier region detection model, demonstrates solid performance in simplified environments but requires further enhancement to handle real-world scenarios with higher variability.

```
Precision: [    0.98552     0.99314     0.95168]
Recall: [    0.98046     0.94878     0.93656]
F1 Score: [    0.98298     0.97045     0.94406]
mAP50: 0.9842467308598458
mAP50-95: 0.8291168400149161
```

Figure 6.1.7 Defect Detection V2 - Performance Metrics

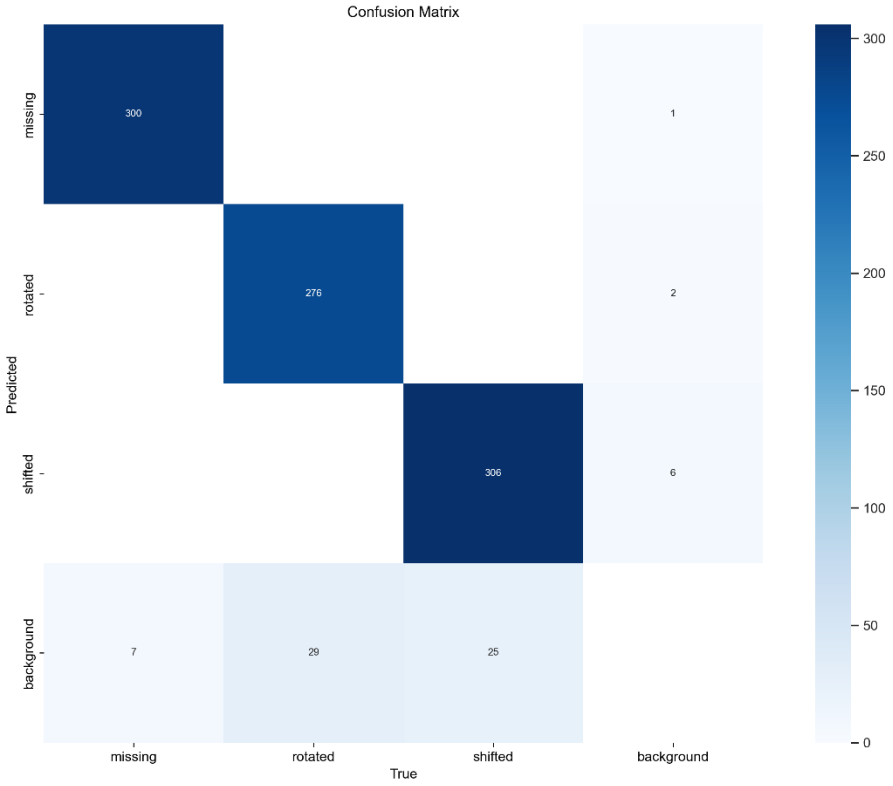


Figure 6.1.8 Defect Detection V2 - Confusion Matrix

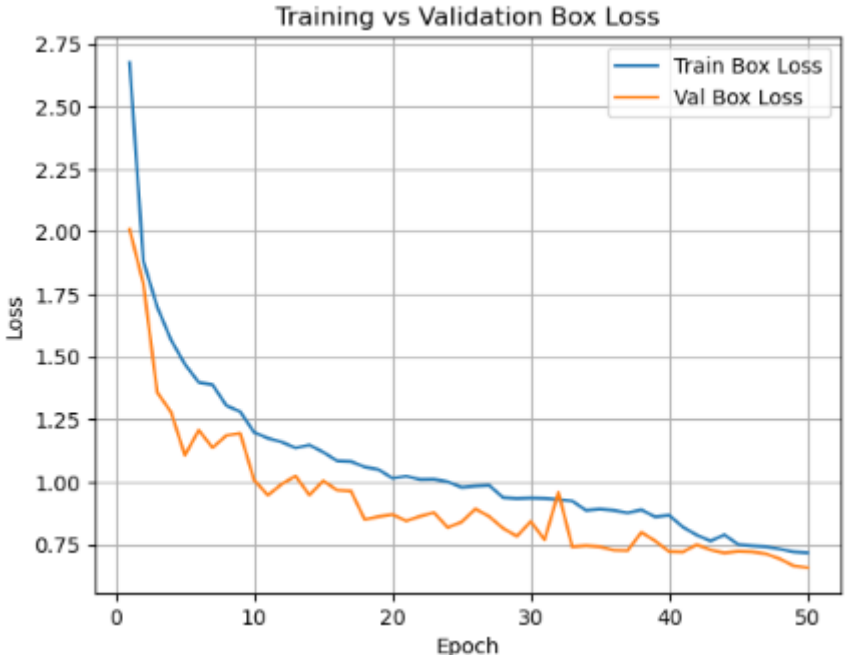


Figure 6.1.9 Defect Detection V2 - Training vs Validation Box Loss Curve

The final version of the defect pin detection model was trained using an augmented dataset. This dataset introduced more complex and realistic conditions, including blurred areas and

noisy backgrounds, which made the defect identification process more challenging. As shown in Figure 6.1.10, the performance metrics of this version declined slightly compared to the previous model. This decrease is expected due to the increased complexity of the dataset, particularly in identifying small or visually cluttered defect areas. The mAP50-95 metric falls below 0.8, which, while lower than the previous version, still meets an acceptable threshold for real-world application. Importantly, the use of data augmentation is essential for improving model robustness in varied environments.

The confusion matrix in Figure 6.1.11 indicates that this model misclassified more instances as background compared to the earlier version. This suggests that the model sometimes struggles to differentiate between actual defect regions and complex background noise in more realistic images.

The training versus validation loss curve, presented in Figure 6.1.12, shows no signs of overfitting. Both loss curves follow a consistent trend, suggesting that the model maintains good generalization despite the increased dataset complexity.

```
Precision: [    0.98158     0.94713     0.93239]
Recall: [    0.95795     0.85246     0.86405]
F1 Score: [    0.96962      0.8973     0.89692]
mAP50: 0.95540317157778896
mAP50-95: 0.7680872110507813
```

Figure 6.1.10 Defect Detection V3 - Performance Metrics

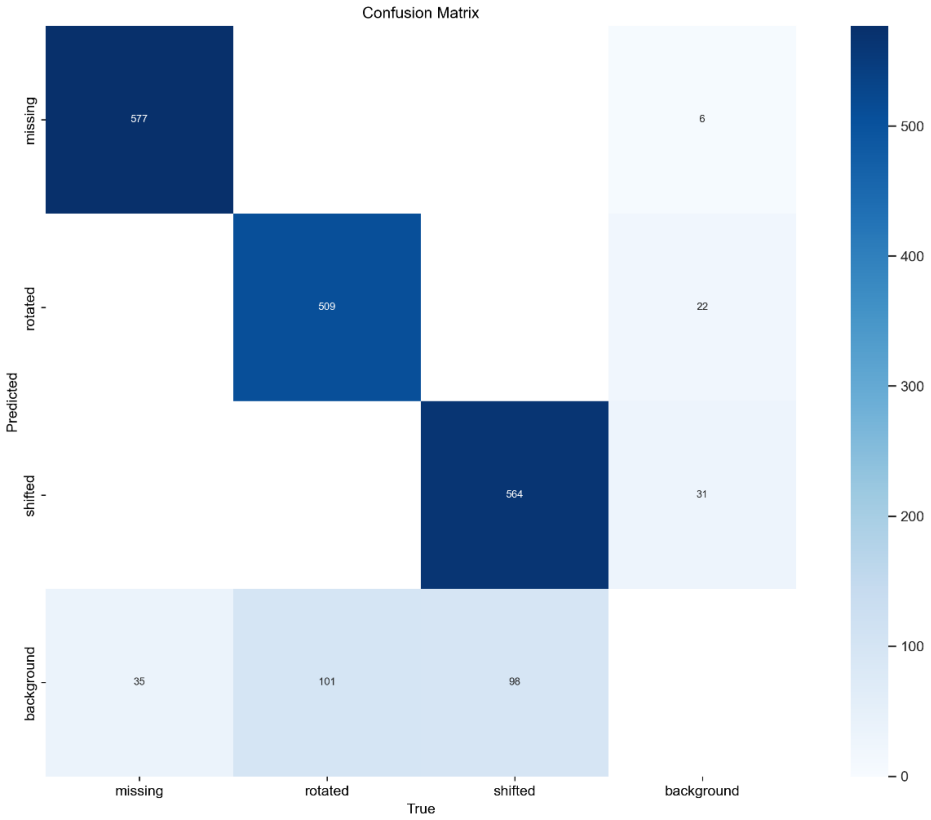


Figure 6.1.11 Defect Detection V3 – Confusion Matrix

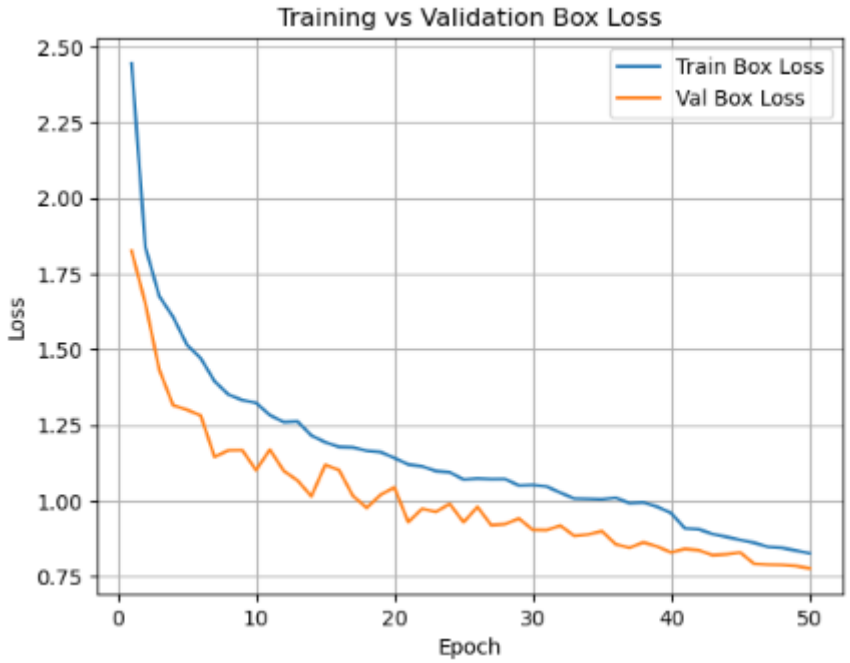


Figure 6.1.12 Defect Detection V3 – Training vs Validation Box Loss Curve

6.2 Testing Setup and Result

This section presents the inspection results for both synthetically generated defect pins and real-world defective pin connectors. Additionally, it outlines the testing setup employed during the evaluation process.

Generated Defect Pin Result

This section presents the inspection results of synthetically generated defect pins. Figure 6.2.1 displays a sample of the generated test image, which was processed using the defect detection function. The model successfully identified the simulated defects, demonstrating its effectiveness in detecting predefined anomalies under controlled conditions.

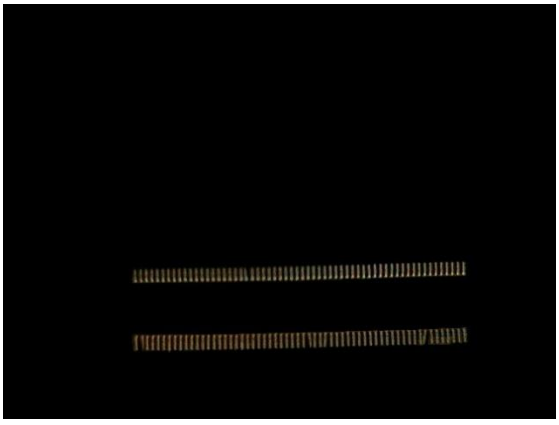


Figure 6.2.1 Testing image sample (Generate Case)

Figures 6.2.2 illustrate the result of testing sample after inspected by the region detection model.

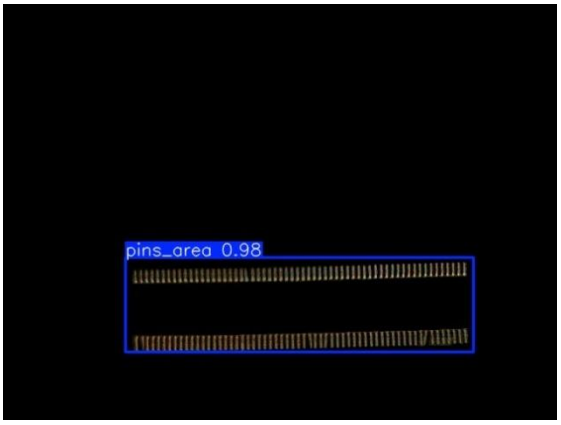


Figure 6.2.2 Region Detection Result (Generate Case)

Figures 6.2.3 illustrate the result of testing sample after inspected by the defect pins detection model.

Figure 6.2.3 Defect Pin Detection Model (Generate Case)

Figure 6.2.4 illustrate the result message of the testing image.

```
Rotation degree: 180.00 (no rotation applied), Pin regions found: 1, Defect pins: 1 rotated, 4 shifted
```

Figure 6.2.4 Function Message Show (Generate Case)

Real-world Defect Pin Result

This section presents the inspection results for real-world defective electrical connectors. Figure 6.2.5 illustrates an electrical connector exhibiting defective pins, captured under varying rotational angles to simulate practical inspection scenarios. The image was processed using the trained defect detection model, which successfully identified the defective pins despite the complexities introduced by rotation and real-world conditions.

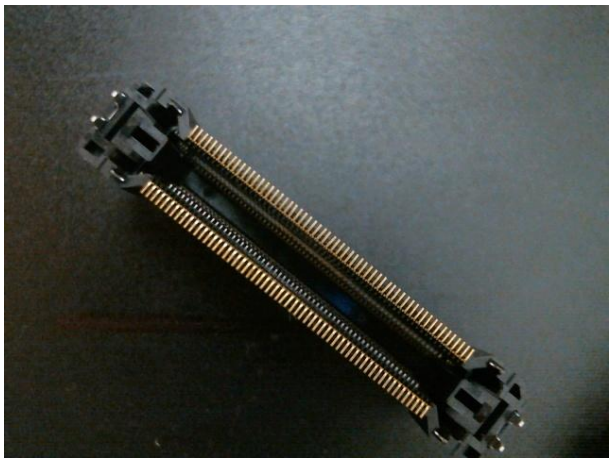


Figure 6.2.5 Testing Image Sample (real-world case)

Figure 6.2.6 illustrates the result of testing samples after being rotated and inspected by the region detection model.

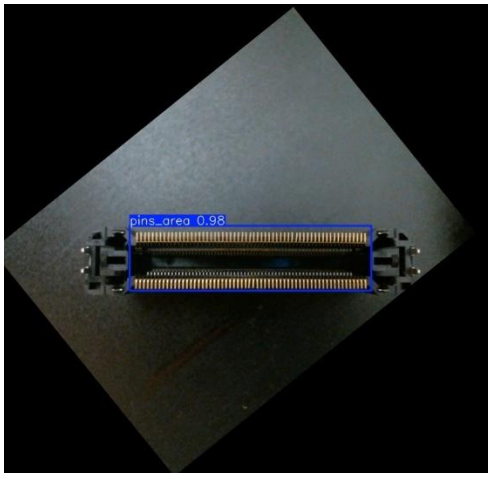


Figure 6.2.6 Region Detection Result (real-world case)

Figure 6.2.7 illustrate the result of testing sample after inspected by the defect pins detection model.

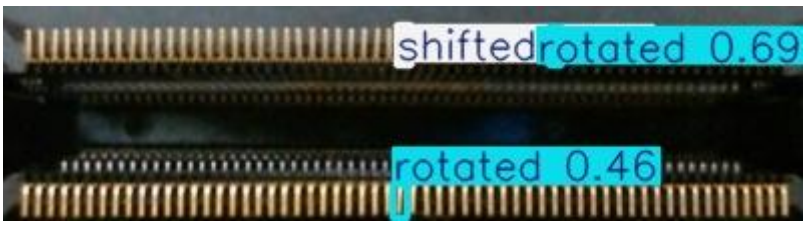


Figure 6.2.7 Region Detection Result (real-world case)

Figure 6.2.8 illustrates the result message of the testing image.

```
Rotation degree: 38.92, Pin regions found: 1, Defect pins: 2 rotated, 1 shifted
```

Figure 6.2.8 Function Message Show (real-world case)

After the testing phase, the project confirms that the dataset is suitable for addressing real-world defect detection challenges. The model demonstrates strong performance in identifying defective pins under practical conditions, validating the dataset's applicability for real-world scenarios.

Real-world Defect Pin Result with No Region Detection Model

This section highlights the necessity of incorporating a region detection model in the defect pin detection process. Figure 6.2.9 illustrates the outcome of testing an image without utilizing the region detection model. The absence of region detection leads to decreased accuracy in identifying defective pins, as the model lacks the ability to focus on specific areas of interest. Implementing a region detection model enhances the precision of defect detection by narrowing down the search area, thereby improving the overall performance of the inspection system.

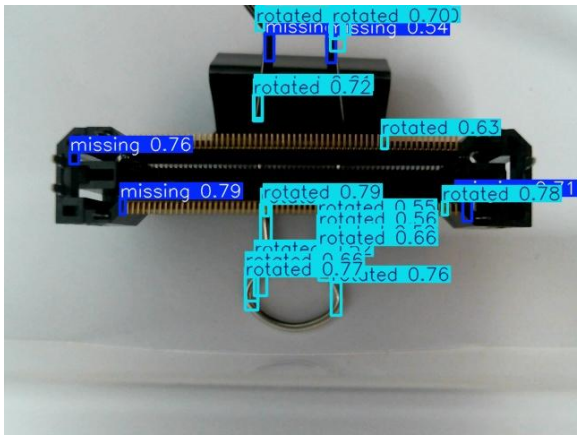


Figure 6.2.9 Region Detection Result (real-world case with no region detection)

6.3 Project Challenges

This section describes the project challenges after evaluating the project model. There are some challenges faced, such as:

- Low Confidence in Detecting Defective Pins

  This problem occurred when the model was applied to real-world defective pin electrical connector inspections. The possible cause is the discrepancy between the generated dataset and real-world cases, leading the trained model to have low confidence in detecting defective pins. However, collecting and labeling real-world cases was not feasible in this project. To address this issue, the generated defective pins can be modified to more closely resemble real-world cases or the model's confidence threshold can be adjusted. This adjustment may cause the model to mislabel defective pins.

Figure 6.3.1 shows the result when the confidence threshold is set higher than 0.5. The red rectangles indicate areas that the model failed to inspect.

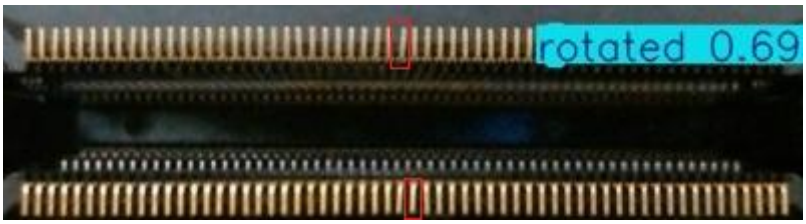


Figure 6.3.1 Detection Fail with High Confidence

- Rotation Function Wrong Rotate the Input Image

This problem has a probability of occurring, but identifying the exact trigger is challenging. It can lead to the region detection model failing to detect pin regions. The solution implemented in the project is the preparation of a backup rotation method, specifically Angle Brute-Force Rotation, which has been applied in the project code.

An alternative solution involves training another model to handle rotation tasks. However, since this project already utilizes two models, adding a new model may increase processing time. Therefore, this solution was not incorporated into the project.

Figure 6.3.2 illustrates a case of incorrect rotation.

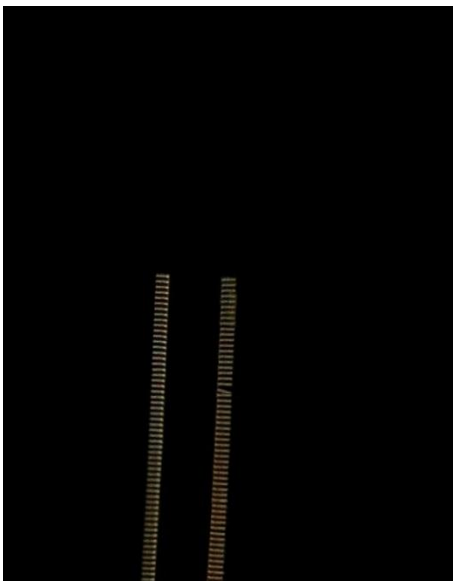


Figure 6.3.2 Wrong Rotation Sample

6.4 Objectives Evaluation

This section outlines the evaluation results and assesses whether the initial objectives of the project were achieved.

I. Developing a High-Quality Dataset

The project successfully achieved this objective. The generated dataset effectively covers real-world cases and encompasses the three categories of defect situations: missing, rotated, and shifted pins.

II. Achieve High Model Accuracy

The initial goal was to attain at least 85% accuracy. Evaluating the model's performance:

- mAP50: Achieved, indicating strong performance in detecting defects at a 50% Intersection over Union (IoU) threshold.
- mAP50-95: Not achieved, as the model's performance declined when evaluated across a range of IoU thresholds from 50% to 95%.

This suggests that while the model performs well under certain conditions, its accuracy diminishes with stricter evaluation criteria.

III. Reduce Processing Time

This objective was met. As illustrated in Figure 6.4.1, the processing times for both models are significantly reduced, demonstrating efficient performance suitable for practical applications.

```
0: 480x640 1 pins_area, 21.6ms
Speed: 2.5ms preprocess, 21.6ms inference, 1.7ms postprocess per image at shape (1, 3, 480, 640)
Results saved to runs\detect\predict145

0: 192x640 2 missings, 3 rotateds, 2 shifteds, 6.6ms
Speed: 1.1ms preprocess, 6.6ms inference, 1.3ms postprocess per image at shape (1, 3, 192, 640)
Results saved to runs\detect\predict146
```

Figure 6.4.1 Model processing show

6.5 Concluding Remark

In summary, this chapter presents the performance metrics of different model versions, testing results, project challenges, and objective evaluations.

The evaluation metrics for the region detection model are detailed in Table 6.1, and those for the defect pin detection model are provided in Table 6.2.

Pins Region Detection Model Performance Metrics					
Version	Precision	Recall	F1-score	mAP50	mAP50-95
Model_v1	0.99987	1	0.99994	0.995	0.994
Model_v2	0.99987	1	0.99993	0.995	0.994

Table 6.1 Pins Regions Detection Model Performance Metrics

Defect Pins Detection Model Performance Metrics											
Version	Precision			Recall			F1-score			mAP50	mAP50-95
Model_v1	0.98	0.62	0.37	1	1	1	0.99	0.76	0.54	0.76	0.76
Model_v2	0.99	0.99	0.95	0.98	0.95	0.94	0.98	0.97	0.94	0.98	0.83
Model_v3	0.98	0.95	0.93	0.95	0.85	0.86	0.97	0.90	0.90	0.96	0.77

Table 6.2 Defect Pins Detection Model Performance Metrics

The testing results encompass generated defect pin detections, real-case detection outcomes, and real-world defect pin detections without utilizing the region detection model.

Project challenges include low confidence in detecting defective pins and incorrect rotation functions affecting input image orientation.

Lastly, the objective evaluation assesses whether the three initial objectives were achieved.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

In conclusion, this project successfully applied computer vision techniques to inspect defect pins in electrical connectors, aiming to develop a high-accuracy and efficient inspection method to reduce the time companies spend on defect pin inspections. The project encompassed two main tasks: dataset preparation and model development.

Due to the uniqueness of the defect pin inspection case and the lack of available open-source datasets, a custom dataset was generated using OpenCV to create synthetic images representing three categories of defect pins: missing, rotated, and shifted. For model development, the YOLOv8 algorithm was employed to train two models: one for detecting the pin regions and another for identifying specific defects within those regions. This two-model approach was adopted because YOLOv8, while powerful, has limitations in detecting small objects. By first narrowing down the area of interest with the region detection model, the defect detection model can focus more precisely, enhancing overall detection accuracy.

The models demonstrated strong performance across various metrics, including precision, recall, F1-score, mAP50, and mAP50-95, indicating their effectiveness in accurately detecting defects in both synthetic and real-world scenarios. However, challenges such as low confidence in detecting defective pins and incorrect rotation functions affecting input image orientation were encountered. These issues were addressed by carefully designing the synthetic dataset to closely resemble real-world conditions and implementing appropriate preprocessing techniques.

Overall, the project achieved its objectives by developing a reliable and efficient method for defect pin inspection in electrical connectors. The combination of custom dataset generation and a two-model YOLOv8 approach provided a robust solution, demonstrating the potential for real-world application in industrial settings.

7.2 Recommendation

To enhance the effectiveness and applicability of the defect pin inspection project, several improvements are recommended. Firstly, refining the synthetic dataset to more accurately reflect real-world conditions—particularly by incorporating realistic rotation scenarios—can improve the model's ability to generalize and increase detection confidence. Secondly, exploring alternative machine learning algorithms beyond YOLOv8, such as ensemble methods or transformer-based models, may offer comparative insights and potentially enhance detection accuracy. Lastly, developing an integrated platform that connects the trained models can provide a user-friendly interface, facilitating easy input of connector images, real-time defect detection, and visualization of results, thereby streamlining the inspection process and making the technology more accessible to end-users.

REFERENCES

[1] admin, "PAVE Technology," *Pave Technology Co*, Feb. 22, 2024. https://www.pavetechnologyco.com/how-are-electrical-connectors-tested-for-safety

[2] O. Diaz *et al.*, "Data preparation for artificial intelligence in medical imaging: A comprehensive guide to open-access platforms and tools," *Physica Medica*, vol. 83, pp. 25–37, Mar. 2021, doi: https://doi.org/10.1016/j.ejmp.2021.02.007.

[3] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," *2012 Proceedings of the 35th International Convention MIPRO*, Opatija, Croatia, 2012, pp. 1725-1730.

[4] Y. Li, "Research and Application of Deep Learning in Image Recognition," *IEEE Xplore*, Jan. 01, 2022. https://ieeexplore.ieee.org/abstract/document/9718847

[5] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, Dec. 2023, doi: https://doi.org/10.3390/make5040083.

[6] W. Wu and Q. Li, "Machine Vision Inspection of Electrical Connectors Based on Improved Yolo v3," *IEEE Access*, vol. 8, pp. 166184–166196, 2020, doi: https://doi.org/10.1109/access.2020.3022405.

[7] Y. Zhao et al., "Simultaneous Detection of Defects in Electrical Connectors Based on Improved Convolutional Neural Network," IEEE Trans. Instrum. Meas., vol. 71, pp. 1–10, 2022, doi: 10.1109/TIM.2022.3169535.

[8] D. Zhao, F. Kong, and F. Du, "Vision-based adaptive stereo measurement of pins on multi-type electrical connectors," Meas. Sci. Technol., vol. 30, no. 10, p. 105002, Oct. 2019, doi: 10.1088/1361-6501/ab198f.

[9] B. C. F. De Oliveira, A. L. Schalata Pacheco, R. C. Costa Flesch, and M. B. Demay, "Detection of defects in the manufacturing of electric motor stators using vision systems: Electrical connectors," in 2016 12th IEEE International Conference on Industry Applications (INDUSCON), Curitiba: IEEE, Nov. 2016, pp. 1–6. doi: 10.1109/INDUSCON.2016.7874551.

[10] Z. Yongzhong, S. Deru, L. Jiachang, Q. Tianyi, Q. Zhimin, and Z. Ruizhe, "Research on AI-Based Gold Removal Technology for Aviation Connector Cup Cavity Surface," in 2023 24th International Conference on Electronic Packaging Technology (ICEPT), Shihezi City, China: IEEE, Aug. 2023, pp. 1–6. doi: 10.1109/ICEPT59018.2023.10491936.

[11] S. Roy, "Agile Development Methodologies: An Essential Guide," *BrowserStack*, Nov. 11, 2022. https://www.browserstack.com/guide/agile-development-methodologies

POSTER

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

FACUALTY OF INFORMATION COMMUNICATION AND TECHNOLOGY

Use AI To Detect Defect Pins in Electrical Connector

INTRODUCTION

This model detects defects pins in electrical connectors, minimizing issues caused by inspection errors.

OBJECTIVE

- Provide an high accuracy model which needs to achive 85% accuracy
- High quality dataset which can use to train the model to overcome the real world case
- Reduce the processing time, the model needs to inspect the defect pins faster than human inspection

PROPOSED METHOD

1. Dataset Preparation
   - OpenCV Function
     - This function generate 3 catefories of the defect pins which is: missing, rotated and shift

2. Use YOLOv8 Algorithm to develop two models:
   - Pins Region Detection Model
     - Detect the pins region, the goal is detect the pins region and past it to the defect pins detection to overcome the YOLOv8 algorithm weakness
   - Defect Pins Detection Model
     - Detect the defect pins in the specific region

MODEL EVALUATION

The pin region detection model anf defect pins detection model performance metrics are showed in the tables.

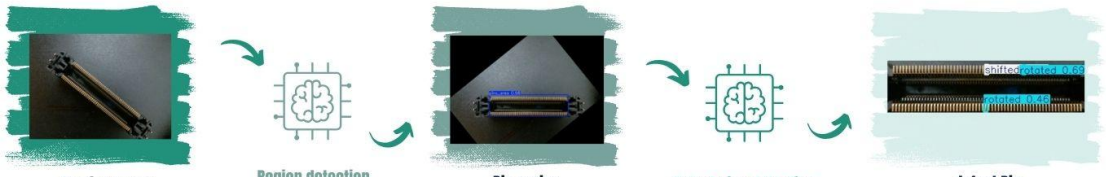
Pins Region Detection Model Performance Metrics

Version	Precision	Recall	F1-score	mAP50	mAP50-95
Model_v1	0.99987	1	0.99994	0.995	0.994
Model_v2	0.99987	1	0.99993	0.995	0.994

Defect Pins Detection Model Performance Metrics

Version	Precision			Recall			F1-score			mAP50	mAP50-95
Model_v1	0.98	0.62	0.37	1	1	1	0.99	0.76	0.54	0.76	0.76
Model_v2	0.99	0.99	0.95	0.98	0.95	0.94	0.98	0.97	0.94	0.98	0.83
Model_v3	0.98	0.95	0.93	0.95	0.85	0.86	0.97	0.90	0.90	0.96	0.77

RESULT



Project Developer : Yong Tian Ze
Project Supervisor : Dr Lee Wai Kong