FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Use of Artificial Intelligence for Defect Detection in Automotive Components

Mauro Alexandre Gomes dos Anjos

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Miguel Velhote Correia

July 27, 2025

Abstract

The increasing demands for precision, quality, and efficiency in automotive manufacturing highlight the limitations of traditional defect detection methods, which rely heavily on human operators. These methods are often inefficient, prone to variability, and incur high labor costs. This dissertation explores the use of Artificial Intelligence (AI), specifically Computer Vision (CV) and Deep Learning (DL) techniques, to address these challenges and improve the quality control process in automotive production lines.

Focusing on small-object detection, this research develops a deep-learning solution tailored to detecting minute punctures in Printed Circuit Boards (PCBs), a critical component in automotive electronics. Building on existing resources, including prior projects at Controlar, and utilizing a high-resolution dataset, the study employs the YOLOv8 model, a state-of-the-art deep learning architecture, to train and fine-tune a model capable of real-time, accurate defect detection.

The methodology includes dataset preparation, model fine-tuning, and performance evaluation through key metrics such as precision, recall, and mean Average Precision (mAP). The research demonstrates the model's efficacy in detecting small defects, achieving high accuracy and operational efficiency compared to traditional manual inspection methods. A user-friendly application was also developed to enable continuous model improvement by allowing technicians to easily add new images and labels for retraining.

The results show that AI-based defect detection, particularly using YOLOv8, can significantly enhance the precision, scalability, and cost-efficiency of automotive manufacturing quality control systems. This work contributes to bridging the gap between academic research and industrial application, offering a robust AI solution that can be further refined and deployed in real-world production environments.

Resumo

As crescentes exigências de precisão, qualidade e eficiência na indústria automóvel evidenciam as limitações dos métodos tradicionais de deteção de defeitos, que dependem fortemente de operadores humanos. Estes métodos são frequentemente ineficazes, sujeitos a variações e acarretam elevados custos com mão-de-obra. Esta dissertação explora o uso de Inteligência Artificial, especificamente Técnicas de Visão Computacional e Aprendizagem Profunda, para fazer face a estes desafios e melhorar o processo de controlo de qualidade nas linhas de produção automóvel.

Focando-se na deteção de pequenos objetos, esta pesquisa desenvolve uma solução de aprendizagem profunda para a deteção de pequenas perfurações em Placas de Circuito Impresso (PCBs), um componente crítico na eletrónica automóvel. Tendo por base recursos existentes, incluindo projetos anteriores na Controlar, e utilizando um conjunto de dados de alta resolução, o estudo recorre ao modelo YOLOv8, uma arquitetura de aprendizagem profunda de última geração, para treinar e afinar um modelo capaz de detetar defeitos em tempo real com alta precisão.

A metodologia inclui a preparação dos dados, o ajuste fino do modelo e a avaliação de desempenho através de métricas chave, como precisão, recall e média de Precisão Média (mAP). A pesquisa demonstra a eficácia do modelo na deteção de pequenos defeitos, alcançando alta precisão e eficiência operacional quando comparado com métodos tradicionais de inspeção manual. Foi também desenvolvido um aplicativo fácil de usar, que permite o aprimoramento contínuo do modelo, permitindo que os técnicos adicionem novas imagens e rótulos para o re-treinamento.

Os resultados mostram que a deteção de defeitos baseada em IA, especialmente utilizando o YOLOv8, pode melhorar significativamente a precisão, escalabilidade e custo-benefício dos sistemas de controlo de qualidade na indústria automóvel. Este trabalho contribui para a aproximação entre a investigação académica e a aplicação industrial, oferecendo uma solução robusta de IA que pode ser refinada e implementada em ambientes de produção reais.

Acknowledgements

I would like to begin by expressing my deep gratitude to my family, whose unwavering emotional and financial support has been vital throughout this entire journey. Their constant encouragement and belief in me have been a foundation of strength, and I could not have reached this point without them.

I also owe a heartfelt thank you to my friends, whose presence in my life has been invaluable. They have always been there to offer encouragement and camaraderie, especially in times of difficulty, helping me rise and persevere through every setback.

"Our greatest glory is not in never falling, but in rising every time we fall"

Confucius

Contents

List of Figures

List of Acronyms

AI	Artificial Intelligence
AVI	Automated Visual Inspection
BIFPN	Bidirectional Feature Pyramid Network
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
DL	Deep Learning
ECA	Efficient Channel Attention
ELAN	Efficient Layer Aggregation Network
ETFL	Efficient Transformer Few-shot Learning
FP	False Positive
FPGA	Field-Programmable Gate Array
FN	False Negative
GSCConv	Grouped Spatial Convolution
IoT	Internet of Things
IoU	Intersection over Union
mAP	Mean Average Precision
PCB	Printed Circuit Board
PRiiMe	Piston Ring Intelligent Inspection Machine
R&D	Research and Development
SPPCSPC	Spatial Pyramid Pooling with Cross-Stage Partial Convolution
TN	True Negative
TP	True Positive
YOLO	You Only Look Once

Chapter 1

Introduction

In an era where technology drives innovation, the automotive industry is increasingly leveraging Artificial Intelligence (AI) to enhance quality control processes. The production of automotive components demands exceptional precision, as even minor defects can compromise a component's performance, leading to costly recalls, reduced product lifespans, or, in severe cases, compromised safety.

1.1 Motivation

Automated visual inspection (AVI) has become a cornerstone in modern automotive manufacturing, where ensuring product quality and reliability is critical to safety and performance. AVI systems reduce human error, minimise production defects, and increase efficiency on high-throughput production lines. As vehicles become more technologically complex and consumer expectations for flawless quality rise, the role of computer vision and AI in defect detection has grown substantially. Traditional inspection methods, which rely heavily on manual labour, are often unable to meet the speed, precision, and consistency required by the automotive sector.

Deep learning techniques, particularly object detection models such as the You Only Look Once (YOLO) family, have driven significant improvements in both the accuracy and speed of defect detection tasks. These models offer the ability not only to detect visible surface defects such as scratches, corrosion, and deformations but also to assess the severity of the defect, allowing timely corrective actions [14]. Moreover, the flexibility of AI algorithms allows manufacturers to tailor solutions to different component geometries, materials, and production environments — key features in automotive applications where parts vary greatly in size, finish, and tolerances.

Despite these technological advances, manual inspection remains common in certain stages of automotive production. These processes are inherently slow and struggle to keep up with the high-volume demands of modern manufacturing [29]. As production lines become increasingly complex, relying solely on human inspectors to identify and classify defects becomes impractical. Additionally, the cost of manual inspection is significant, particularly when it requires skilled

labour. Rising wages and the inefficiency of using human labour for repetitive, fine-detail tasks further strain production economics in the competitive automotive sector.

This dissertation was developed in collaboration with Controlar, a Portuguese company dedicated to industrial automation and innovation, with a particular focus on the automotive sector. Controlar designs and develops advanced testing and inspection systems for other companies, supplying solutions that ensure quality and reliability throughout the production process. Its expertise spans hardware and software integration for industrial testing, with strong emphasis on customisation, efficiency, and technological advancement.

The research is motivated by a real-world challenge in this field: improving automatic puncture detection in Printed Circuit Boards (PCBs), a common component in automotive electronic systems. These punctures, which must be verified as part of the quality control process, are extremely small and visually variable, depending on factors such as materials, colour, lighting, and welding artefacts. Detecting these defects falls under the category of *small object detection*—a particularly difficult task where many object detection models, including earlier versions of YOLO, may perform poorly due to the limited resolution in feature maps.

While previous projects at Controlar, such as PRiiMe and others, provided foundational experience in applying computer vision to defect detection in automotive components, they had not specifically addressed the challenge of small object detection in PCB inspection[20]. This dissertation builds on those foundations and seeks to fill that gap by fine-tuning a state-of-the-art model using a dataset tailored for this purpose. The objective is to develop a robust and generalizable model capable of detecting punctures across a diverse range of PCB types encountered in automotive applications.

Ultimately, this work contributes not only to the resolution of a specific industrial challenge but also to the broader advancement of AI-based visual inspection systems in the automotive industry, particularly in tasks that require precise and reliable detection of small-scale anomalies in safety-critical components.

1.2 Objectives

The primary objective of this dissertation is to develop and implement an AI-based system capable of accurately detecting puncture defects in PCBs, which are widely used in automotive electronic systems. These defects, often small and visually subtle, pose a significant challenge to traditional inspection systems and require specialised approaches in the field of small object detection.

To address this challenge, the project explores state-of-the-art deep learning methods for object detection, evaluating their effectiveness in identifying small-scale anomalies under various imaging conditions. Among the techniques considered, models from the YOLO family were selected because of their wide adoption, high inference speed, and strong performance in industrial applications.

The specific objectives of this dissertation are:

- **Dataset Preparation:** Convert and organise PCB inspection data into a structured dataset suitable for training deep learning models, including annotation normalisation and format adaptation.
- **Model Training and Fine-Tuning:** Train a selected object detection model on the dataset using iterative hyperparameter tuning, with attention to the unique challenges of small object detection.
- **Performance Evaluation:** Evaluate model performance using relevant metrics, such as precision, recall, mean Average Precision (mAP), and a composite fitness score, with the goal of achieving both accuracy and generalizability.
- **Training Optimisation :** Analyse training and validation losses to refine model parameters and augmentation strategies, identifying the best trade-offs between training time and detection quality.
- **Export and Integration Readiness:** Ensure the trained model can be exported to a deployable format (e.g. ONNX), enabling future integration into real-time industrial inspection systems.

By achieving these goals, the project contributes to the broader application of AI-driven visual inspection systems in the automotive industry, particularly in contexts that demand reliable detection of small and heterogeneous defects.

1.3 Contribution

This research contributes significantly to advancing AI-driven defect detection methodologies within the automotive industry by specifically addressing the unique and challenging scenario of small-object detection, more precisely, the detection of tiny puncture defects in PCBs. Although numerous generalised solutions for automated defect detection are available, effectively adapting these solutions to accurately detect very small and subtle defects in automotive electronic components, such as PCBs, remains a technically challenging and underrepresented problem in applied industrial AI.

The dissertation provides a tailored approach, carefully selecting, fine-tuning, and rigorously validating a deep-learning model designed explicitly for accurate localisation and classification of PCB punctures. Through meticulous documentation of the entire training pipeline— including detailed dataset preparation, annotation conversions, hyperparameter optimisation, and comprehensive performance evaluation —the study offers a clear, practical, and reproducible methodology. This structured and iterative approach leverages expert consultation and metric-driven decision-making, overcoming specific performance bottlenecks encountered during the research.

Furthermore, this work illustrates successful real-world integration, demonstrating how advanced AI models developed in a research environment can effectively be translated into operational improvements within actual automotive manufacturing settings. By employing widely

supported AI frameworks, this research ensures practical compatibility and deployment feasibility in industrial inspection systems.

Compared to traditional manual inspection methods, the proposed AI-driven solution provides significant improvements in terms of consistency, detection accuracy, and scalability, directly enhancing quality assurance practices in automotive production lines. By doing so, this dissertation bridges the gap between general academic advancements and specialised industry requirements, emphasising the practical necessity and value of adapting sophisticated AI methodologies to meet the precise operational needs and constraints of modern automotive manufacturing environments.

1.4 Document Structure

This dissertation is organised into six chapters, each progressively building the narrative from initial motivation to technical implementation and evaluation. Chapter 1 introduces the topic by presenting the motivation behind the project, its main objectives, and the specific contributions made in the context of automotive manufacturing. Chapter 2 provides a comprehensive review of the state of the art in AI-based defect detection, including traditional computer vision techniques, deep learning approaches, and relevant frameworks that informed the methodological choices of this work.

Chapter 3 situates the project within the broader research and development activities of the host company. It highlights the influence of previous initiatives, outlines the rationale behind the selection of AI frameworks, and describes the deployment environment into which the model was ultimately integrated. Chapter 4 presents the methodological approach adopted in this work, beginning with the preparation and structuring of the dataset and continuing through the selection of appropriate augmentation strategies, model architecture, training setup, and fine-tuning process. Particular emphasis is placed on the decisions made to address the challenges of small object detection and on the metrics used to evaluate model performance.

Chapter 5 presents the experimental results and performance evaluation of the trained model. It includes both qualitative and quantitative analysis of training outcomes and inference behaviour on unseen data, validating the effectiveness of the proposed approach. Finally, Chapter 6 concludes the dissertation by summarising the findings, discussing the practical implications of the work, and proposing directions for future development and research.

Chapter 2

State of the Art

This chapter provides an in-depth review of the state-of-the-art in AI-driven defect detection systems, focusing on their application in the automotive industry.

2.1 Introduction

The automotive industry is at the forefront of leveraging emerging technologies to enhance manufacturing processes and ensure product quality. As manufacturing precision and efficiency have become critical to meeting consumer and regulatory demands, traditional defect detection methods have shown limitations in addressing increasingly complex quality control requirements.

2.2 Advancements in Defect Detection Techniques

The adoption of AI in visual inspection (VI) systems has introduced several innovative capabilities. By leveraging supervised learning methods, AI models are trained on labelled datasets to classify defects and predict compliance. Unsupervised learning techniques are employed to identify patterns and anomalies in data, while reinforcement learning models enable systems to adapt to changing conditions on production lines dynamically. Deep learning models, in particular, are valued for their ability to handle large, unstructured datasets, such as raw images and video frames, without extensive preprocessing, making them indispensable for defect detection in complex manufacturing environments [4].

Moreover, the integration of IoT technologies enhances these systems by enabling real-time data collection, processing, and analysis. IoT-enabled VI systems facilitate continuous monitoring of production processes, providing immediate detection of defects and insights into their root causes. This not only reduces the reliance on manual inspection but also enables targeted adjustments to improve manufacturing efficiency and product quality [4].

One notable example of AI application in the automotive industry is the PRiiMe project, which developed an intelligent inspection system for piston rings. This project highlighted the limitations of manual inspection, such as inconsistency and limited scalability, and demonstrated the benefits

of AI-driven systems in reducing human error, increasing throughput, and enabling data-driven continuous process improvements [29].

2.3 Challenges in AI-Based Defect Detection

Despite the advancements, several challenges hinder the widespread adoption of AI-driven defect detection systems in the automotive sector. A significant obstacle lies in the quality and availability of training data. The performance of machine learning algorithms depends heavily on large, high-quality datasets that accurately represent the diversity of defects encountered in production. However, collecting and annotating such datasets is often time-consuming and expensive [25]. Furthermore, defects may vary significantly across different components, requiring domain-specific models and customised training approaches.

Variability in environmental factors, such as lighting, surface texture, and component orientation, can also impact the accuracy of AI models. To mitigate these issues, modern VI systems employ data augmentation techniques to enhance model robustness and domain adaptation methods to improve generalisation across diverse inspection scenarios[4].

Another challenge is the integration of AI systems into existing manufacturing workflows. Transitioning to AI-based inspection often requires technological improvements and a change in organisational culture. Employee training is crucial to ensure effective collaboration between human operators and AI systems. Hybrid approaches, which combine the efficiency of AI with the intuition and critical thinking of human inspectors, have shown promise in achieving superior outcomes.

2.4 Tailored Solutions for the Automotive Industry

The automotive industry faces immense pressure to ensure the quality and reliability of its components, given the stringent standards for safety and performance. Over the years, advances in CV and AI have resulted in the development of innovative solutions tailored to address specific defect detection challenges in automotive manufacturing. Two contributions in this space include the methodologies proposed by Hua et al.[8] and Xie et al.[30], which exemplify cutting-edge applications of CV in automotive defect detection.

Hua et al.[8] introduced a defect detection methodology that focuses on laser brazing welds, which is crucial to ensuring structural integrity in car body manufacturing. Their approach is centred on a model-based segmentation technique that uses 3D laser vision data, effectively identifying defects such as pores, overlaps, and spatters. As shown in Figures 2.1, 2.2, and 2.3, the methodology demonstrates its ability to detect and segment these defects, demonstrating robust performance even under challenging factory conditions. This method involves several critical steps:

- Acquisition of 3D surface profiles of the weld using laser vision sensors.

- Construction of a dynamic ideal contour (DIC) model for the weld surface based on a weighted cubic spline regression (WCSR) model refined with a Fast Expectation-Maximisation (Fast-EM) algorithm.
- Identification and segmentation of defects, such as pores and spatters, through multi-threshold analysis of residuals between the measured data and the DIC model.

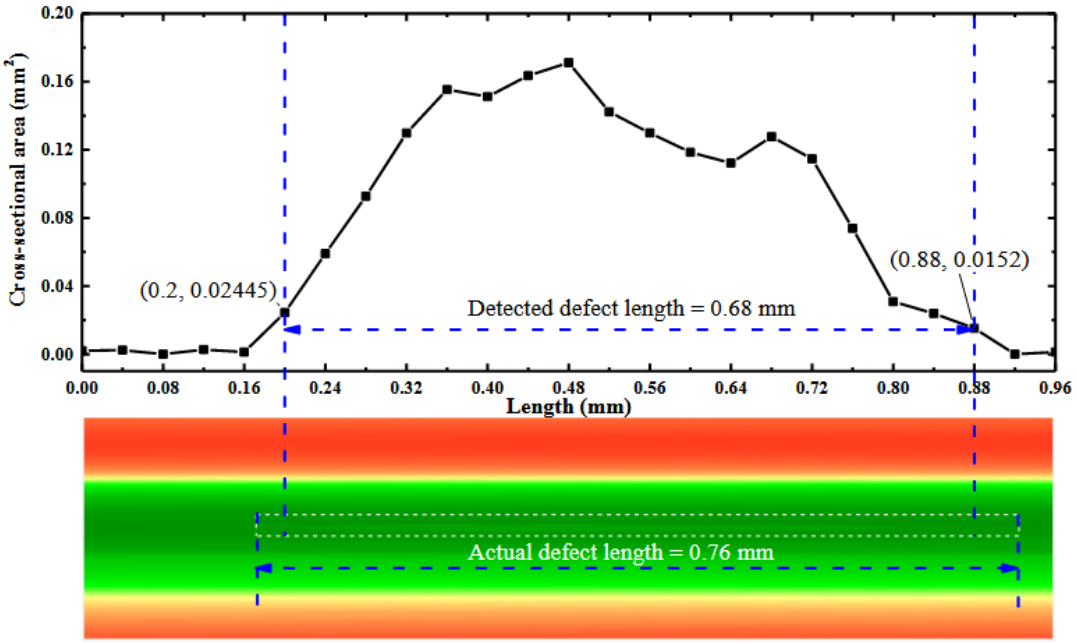


Figure 2.1: Detection of overlaps by the proposed method;reproduced from [8]

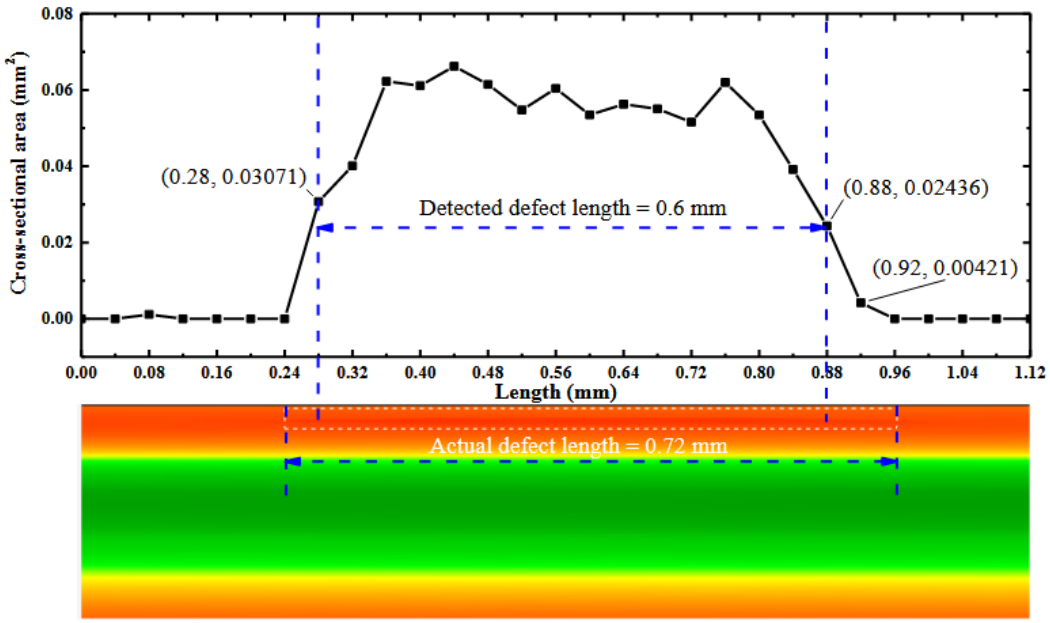


Figure 2.2: Detection of spatters by the proposed method;reproduced from [8]

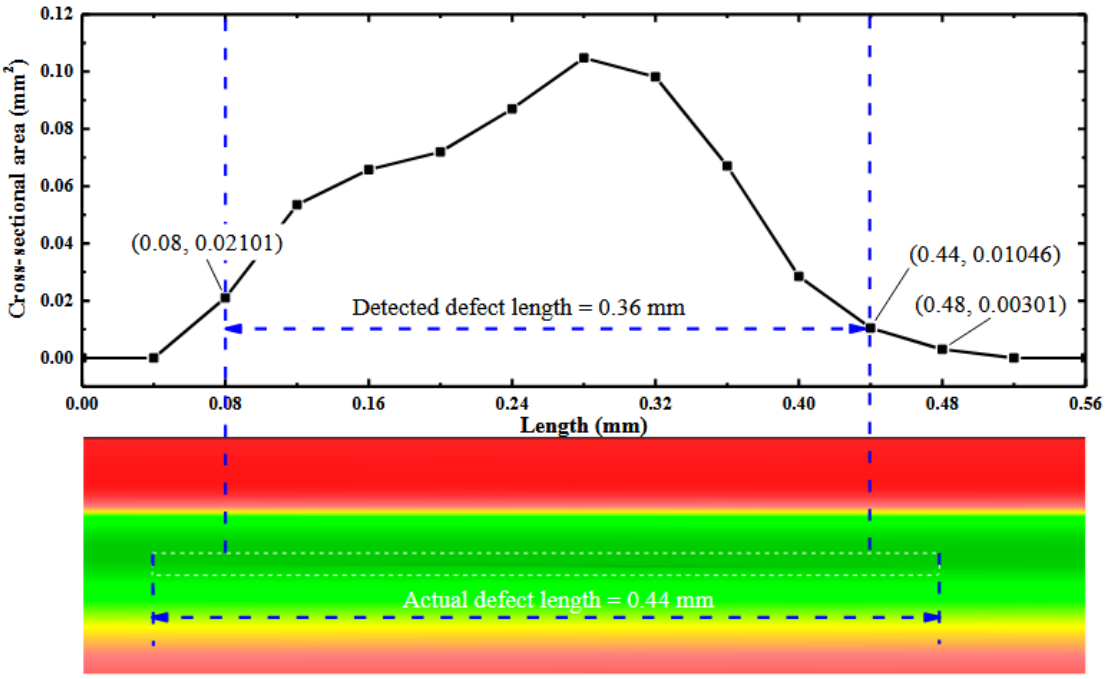


Figure 2.3: Detection of pores by the proposed method;reproduced from [8]

This method demonstrated robust performance in detecting multiple defect types while maintaining precision, even in challenging factory conditions with noise interference and sensor instability. It also highlighted the capability of automated systems to surpass traditional visual inspections by providing real-time, high-accuracy defect detection [8].

Similarly, Xie et al. [30] proposed an adaptive defect detection framework targeting surface anomalies on car body panels. Their method employs a combination of contrast enhancement and edge detection to improve defect segmentation. Key steps in their approach include constructing a contrast-enhanced map using a rotation-invariant variance operator, extracting edges with an enhanced Sobel edge detector, and segmenting the image with Otsu thresholding and post-processing. As shown in Figure 2.4, the combined edge maps and post-processing steps effectively highlight defect regions while minimising background noise.

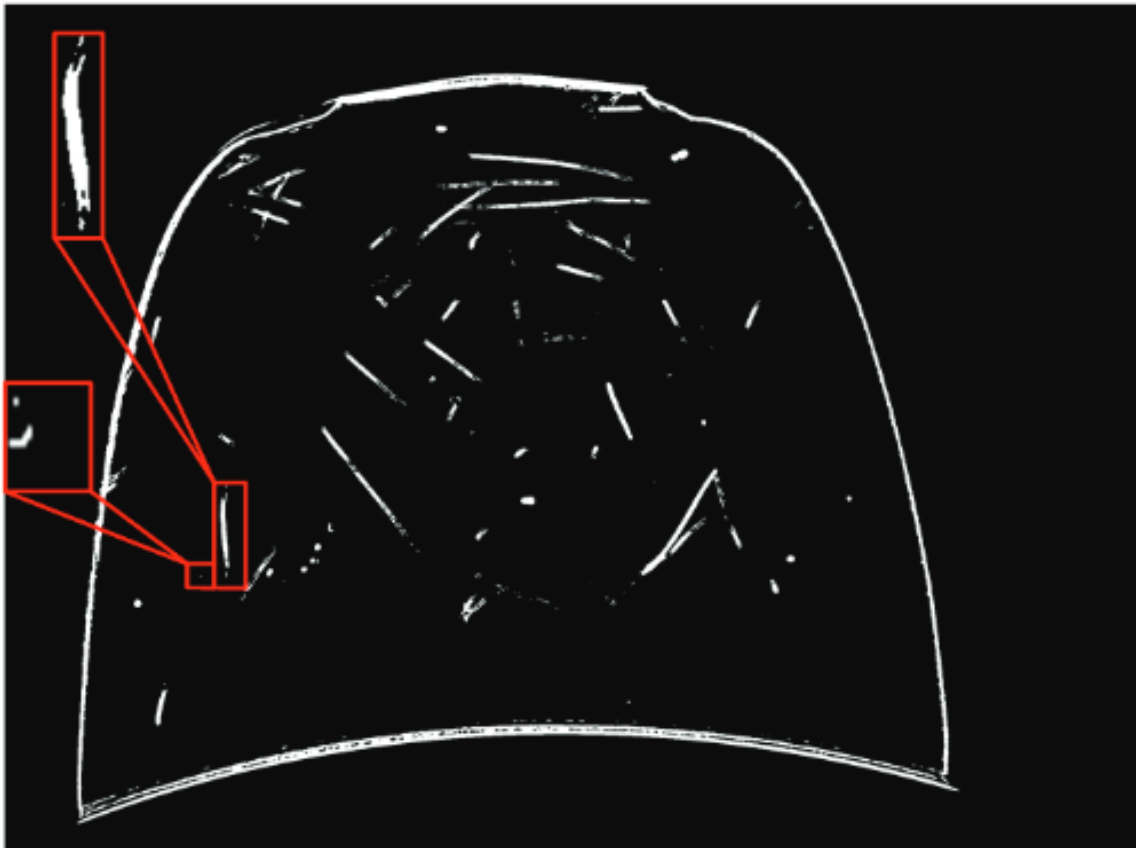


Figure 2.4: The resulting image using post-processing;reproduced from [30]

The results achieved by Xie et al. [30] demonstrated significant improvements in metrics such as Intersection-over-Union (IoU) and precision compared to traditional methods. This showcases the adaptability and accuracy of their approach in addressing the unique challenges posed by automotive body surfaces.

These methods reflect the broader trend of leveraging advanced CV and AI technologies to tackle diverse defect detection tasks in the automotive sector. From weld inspection to panel surface evaluation, these solutions illustrate the growing maturity and applicability of automated systems to replace or augment traditional inspection processes. By adopting personalised algorithms, automotive manufacturers can achieve higher detection accuracy, improved efficiency, and reduced operational costs.

As the field continues to evolve, further integration of IoT for real-time monitoring and data-driven optimisation, as well as advances in adaptive algorithms, promises to expand the scope and impact of AI-driven defect detection systems in automotive production.

2.5 More Recent Advances in Automotive Defect Detection

Recent studies have explored a range of approaches to address defect detection in automotive manufacturing, leveraging advancements in deep learning, machine learning, and cost-effective

imaging systems. These works demonstrate diverse methodologies tailored to specific challenges, such as scalability, accuracy, and the availability of defect samples.

2.5.1 Deep Learning-Based Solutions

Liqun et al.[15] proposed a defect detection system based on an enhanced VGG16 architecture, incorporating the Inceptionv3 module to improve feature extraction. By increasing the width of the network while maintaining depth, this method achieved a detection accuracy of 95.29%, significantly outperforming traditional techniques such as HOG + SVM. To enhance robustness, the authors employed data augmentation strategies, addressing the variability and complexity of vehicle part defects. This approach highlights the potential of tailored deep learning architectures to overcome the limitations of conventional methods.

Xu and Ma[31] introduced a novel few-shot learning model, the Efficient Transformer Few-shot Learning (ETFL), designed to address the challenge of limited defective samples in industrial environments. Their model integrates Efficient Channel Attention (ECA) and Transformer-based self-attention modules to enhance feature extraction. Through experiments on both real-world datasets and the miniImageNet benchmark, the authors demonstrated that ETFL significantly outperforms existing few-shot learning methods, making it particularly suitable for scenarios with low defect occurrence rates.

2.5.2 Predictive Quality Control

Yorulmuş et al.[34] conducted a case study in the Turkish automotive industry, focusing on predictive defect detection in braking systems. Using quality control data from 2018 to 2020, they compared various machine learning algorithms, including Logistic Regression, SVM, Random Forest, and Gradient Boost. Their results showed that Gradient Boosting and CatBoost were the most effective algorithms for identifying rare quality defects. This study underscores the importance of integrating predictive modelling with historical quality data to enhance defect detection within the framework of Industry 4.0.

2.5.3 Low-Cost Imaging Solutions

Korodi et al.[13] developed a cost-effective fault detection solution for end-of-line testing of electronic control units (ECUs) in automotive production. Their system, IP-LC-FD, utilises Raspberry Pis, Python, and OpenCV to detect hardware defects such as misaligned pins, missing components, and surface cracks. With an accuracy rate exceeding 98% and a processing time of less than 7 seconds per unit, the system demonstrated exceptional efficiency while reducing costs by more than 95% compared to traditional industrial systems. This work highlights the feasibility of deploying affordable imaging-based solutions for real-time defect detection.

Together, these advancements illustrate the diversity of solutions available for defect detection, ranging from cutting-edge AI methodologies to practical and cost-effective implementations. They underscore the adaptability of modern technologies in meeting the evolving demands of the automotive industry.

2.6 Inclusion of YOLO in Automotive Applications

The YOLO algorithm has become a leading standard in object detection tasks, widely adopted across various industries, including the automotive sector. Its ability to perform real-time object detection with high accuracy makes it particularly suitable for critical applications such as traffic management, autonomous vehicle development, and defect detection in manufacturing environments.

A bibliometric analysis (2020–2024) highlights the widespread adoption and evolution of YOLO in these domains [6]. This analysis categorises the applications of YOLO into three main areas and examines various architectures, from YOLO v2 to YOLO v8, as well as advanced variants such as YOLO-H, YOLO-X, YOLO-R, and YOLO-C. Among these, YOLOv8, developed by Ultralytics, stands out for its superior performance in both accuracy and computational efficiency, making it the preferred choice for applications requiring high-speed, high-precision detection. In the automotive sector, YOLO has proven instrumental in overcoming challenges such as enhancing safety in autonomous driving systems and optimising production workflows in manufacturing facilities, as illustrated in Figure 2.5, which shows the percentage distribution of applications in the automotive field that use YOLO. Ongoing advancements, such as improved malleability and precision, underscore its critical role in the industry, enabling it to meet the growing demands for accuracy, speed, and adaptability.

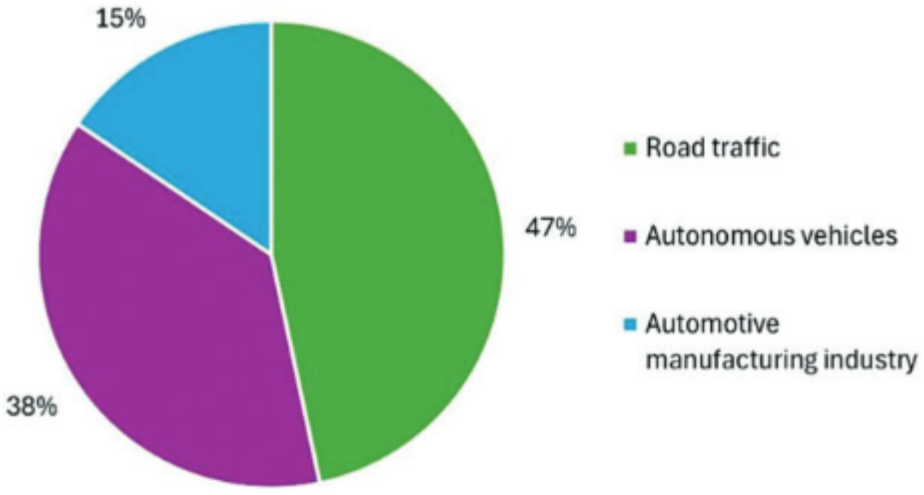


Figure 2.5: Percentage distribution of the applications in the automotive field that use YOLO [6].

This versatility and widespread applicability of YOLO in various domains, including the automotive industry, is further demonstrated by advances such as the GSBF-YOLO model [35]. As a

prime example of YOLO's adaptability and potential for optimisation, GSBF-YOLO builds upon YOLOv8 to face specific challenges in defect detection for steel strip production. By integrating innovative modules such as the Bidirectional Feature Pyramid Network (BIFPN) for enhanced feature fusion and the Grouped Spatial Convolution (GSCConv) for efficient parameter utilisation and improved feature extraction, GSBF-YOLO exemplifies how YOLO architectures can be fine-tuned for specialised use cases. The enhancements in GSBF-YOLO enable it to achieve superior performance, reducing false positives and negatives even in low-contrast defect scenarios. Validated in the NEU-DET dataset, the model demonstrated notable improvements in metrics such as mean Average Precision (mAP), solidifying its effectiveness for high-precision defect detection in industrial environments.

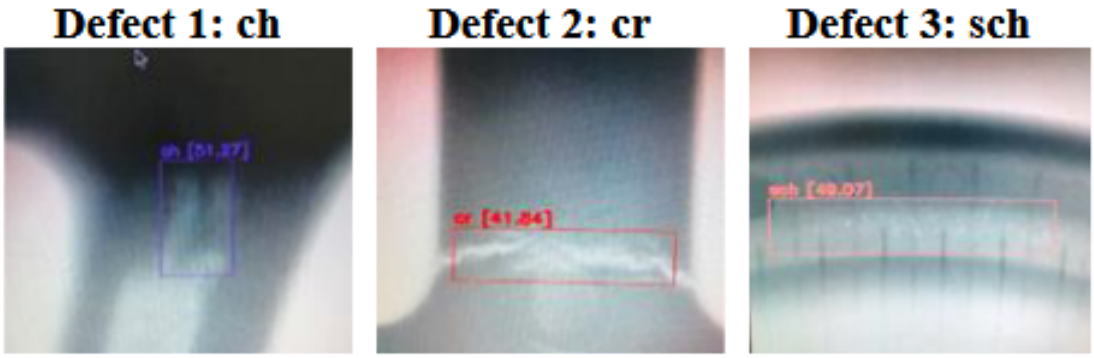


Figure 2.6: The results of defect detection in wheels using YOLO [5].

Recent advances have introduced specialised implementations of YOLO tailored to the unique challenges of automotive defect detection. For example, the TD-YOLOA model was specifically developed to detect tire defects on X-ray images, as shown in Figure 2.6, which illustrates the model's application in identifying defects in tire images [19]. By incorporating advanced components such as the Efficient Layer Aggregation Network (ELAN), Spatial Pyramid Pooling with Cross-Stage Partial Convolution (SPPCSPC), and the Convolutional Block Attention Module (CBAM), TD-YOLOA enhances feature extraction and achieves a mAP of 91.3% while maintaining real-time performance.

Similarly, the YOLOv7 architecture has been applied to detect defects in automotive running lights. This implementation introduces mechanisms like Recursive Gated Convolution (gnConv), Global Attention Mechanism (GAM), and Ghost Convolution to reduce redundancy and improve detection accuracy for small targets. The improved model achieves a recognition accuracy of 89.7% while processing at 61 frames per second (FPS), fulfilling the demand for efficient detection in automotive applications [3].

In the context of automotive component manufacturing, YOLOv3 has demonstrated its efficiency in identifying solder joints on automotive door panels, a critical task in welding lines. This algorithm achieved remarkable precision in detecting small and dense solder joints, with an average detection time of less than 0.2 seconds per image [17].

Further advancements, like the MBEA-YOLOv7 model [9], have incorporated attention mechanisms such as BiFPN and Alpha-IoU loss functions. These improvements face the challenges of detecting small and complex defects on automotive parts, achieving an mAP improvement of 5.7% over the original YOLOv7.

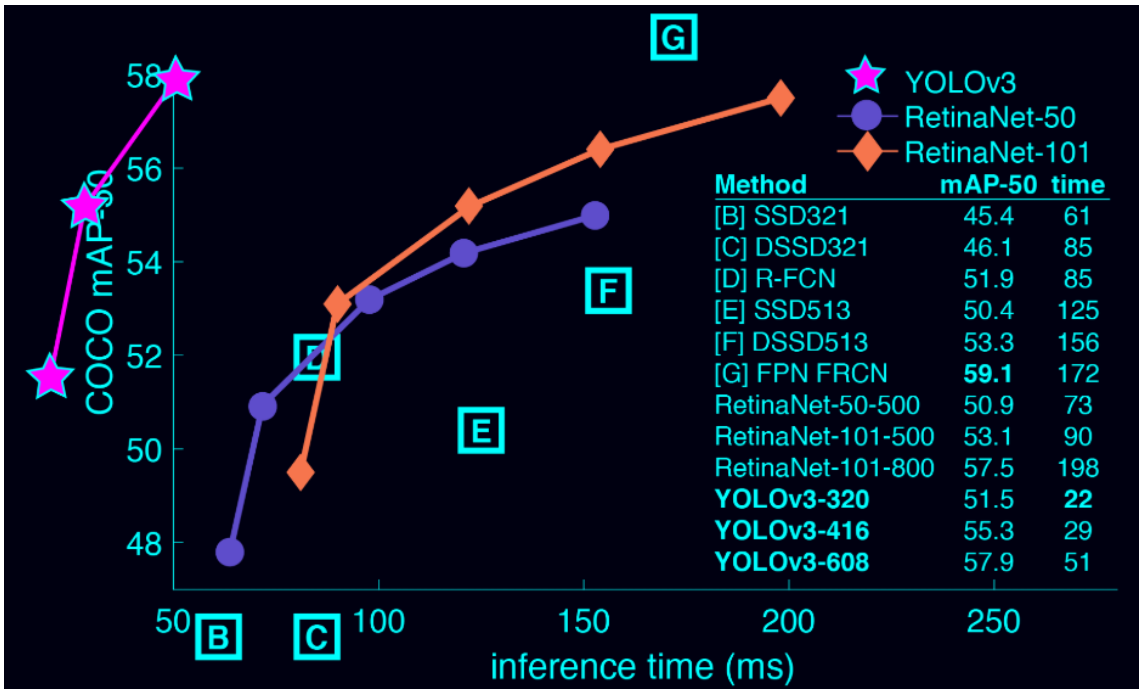


Figure 2.7: YOLO: Comparison to other detectors on COCO dataset [22].

The adaptability and modular nature of YOLO have made it a mainstream solution for defect detection in automotive components. The flexibility of the algorithm, illustrated in Figure 2.7, allows for the integration of innovative modules tailored to specific industrial requirements. For instance, the TD-YOLOA [19] and GSBF-YOLO [35] models demonstrate how advanced attention mechanisms and feature fusion networks can enhance the detection of small defects, low-contrast scenarios, and real-time operational needs [21].

YOLO-based systems continue to transform quality control processes in the automotive industry. Their application not only ensures defect detection with unparalleled precision but also contributes to improved manufacturing efficiency and reduced production costs.

2.7 Small Object Detection

Small object detection is a specialised subfield within computer vision that involves detecting instances of very small objects within images or video frames. It presents unique challenges distinct from general object detection, largely due to the limited spatial information available. Typically, small objects occupy areas of 32×32 pixels or less, as defined by widely accepted benchmarks such as the MS-COCO dataset [28].

Detecting small objects introduces several notable challenges, primarily arising from limited appearance information. With fewer pixels available, small objects are easily confused with background clutter or noise, making reliable detection difficult. Additionally, high localisation precision is essential, as even minor inaccuracies can significantly reduce detection accuracy and result in missed detections or false positives [28]. Scale variation poses further difficulties, as objects may appear small due to their distance from the camera, rather than their physical size, causing traditional detection methods to struggle with inconsistent scale representations. Moreover, deep convolutional neural networks, despite their powerful feature extraction capabilities, often lose critical distinguishing features of small objects through their layers of abstraction, leading to diminished detection performance [16].

To overcome these challenges, researchers have developed specialised techniques designed specifically for small object detection. Multi-scale feature learning methods, such as Feature Pyramid Networks and featurised image pyramids, help preserve essential features across different resolutions, significantly improving detection performance for small objects. Enhanced data augmentation strategies, including targeted scaling, cropping, and controlled translation, have proven beneficial in helping models generalise effectively, even with limited pixel information available. In addition, context-based detection methods leverage surrounding contextual information to boost accuracy, though careful management is necessary to avoid introducing additional noise [28]. The use of Generative Adversarial Networks has also emerged as a promising approach to generate high-resolution synthetic training data that enrich the datasets and help overcome the limitations inherent in real-world training samples. Additionally, sensor fusion techniques combining camera data with other sensing modalities such as LiDAR have shown substantial improvements, providing richer contextual cues and increased robustness in challenging environments [16].

Detecting small objects plays a pivotal role in a variety of practical applications, such as autonomous driving, remote sensing, and industrial inspections, including the detection of minute defects on printed circuit boards. Recent methodologies, integrating multi-scale architectures, advanced data augmentation, and context-aware models, have significantly enhanced detection performance across various applications [32]. As the field continues to evolve, research is increasingly focused on refinement of feature extraction mechanisms, fine-tuning of training strategies, and addressing complex scenarios involving occlusion, rapid motion of objects, and significant variance of the scale. These efforts highlight the importance and ongoing innovation required in the field, aimed at achieving greater reliability and accuracy in detecting small objects, which remain critical for numerous real-world tasks and applications.

Chapter 3

Background and Project Context

3.1 The PRiiMe project

The work conducted during this dissertation builds upon a strong foundation of prior projects and resources within the company. In particular, the PRiiMe project provides valuable background and contextual relevance for this research.

The PRiiMe project developed an automated inspection system for detecting defects in piston rings using computer vision and data analysis techniques [29]. Although the system requires further validation in real-world production settings, its results highlight the feasibility of using automated methods to improve the precision and efficiency of defect detection.

Although the data set was not sourced from PRiiMe, previous Controlar projects, including PRiiMe, the ongoing PCB inspection initiative, and the "Automatic Final Control Inspection System for Automotive Displays" [20], played an important role in shaping the company's expertise in automated visual inspection.

These initiatives collectively provided technical and methodological know-how, especially in areas like dataset labelling, inspection pipeline development, and the use of deep learning models in industrial environments. The lessons learned from these projects helped define realistic performance expectations, identify relevant tools and techniques, and establish good practices for managing image data and training workflows.

While PRiiMe initially appeared to be the source of the dataset, it ultimately contributed more through its indirect influence, serving as an important milestone in Controlar's continued investment in computer vision and AI-driven inspection systems.

3.2 AI Frameworks in the Automotive Industry

The successful implementation of automated visual inspection solutions in the automotive industry relies significantly on selecting appropriate AI frameworks. Several AI frameworks have emerged as industry standards, each offering specific advantages for different stages of AI-driven defect detection—from model development to deployment in production environments. While not all

these frameworks have been directly employed at Controlar, a comprehensive assessment was performed to understand their suitability, leading to an informed framework selection tailored to project requirements.

3.2.1 PyTorch

PyTorch was ultimately chosen as the primary training framework for this dissertation, largely due to its dynamic computational graph capability, which simplifies debugging and experimentation. Its recent advances, particularly in version 2.3 with features like *torch.compile*, significantly reduced inference latency—an essential requirement for real-time defect detection tasks in manufacturing environments [26]. Additionally, PyTorch's support for custom kernel integrations via Triton allowed effective utilisation of GPU acceleration, vital for handling the computational demands of deep learning models.

3.2.2 TensorFlow

Initially, TensorFlow was considered due to its extensive ecosystem, ease of use via the Keras API, and broad hardware compatibility. Version 2.16's introduction of enhanced edge computing features through TensorFlow Lite (TFLite) and optimised quantisation techniques (e.g., INT8 via XNNPack) provided a compelling framework for deploying lightweight inference models suitable for inline automotive inspection tasks [23]. However, due to compatibility and model conversion complexities from PyTorch-based YOLO architectures, PyTorch was ultimately preferred.

3.2.3 JAX

Google's JAX framework, noted for features such as automatic differentiation, just-in-time (JIT) compilation through XLA, and efficient parallelization, was also briefly evaluated. While JAX offers significant computational advantages—especially in rapid prototyping and large-scale experimentation scenarios—it was ultimately considered less aligned with immediate practical needs, given its more experimental nature and less mature deployment ecosystem in industrial defect detection contexts [24].

3.2.4 ONNX for Deployment and Interoperability

The Open Neural Network Exchange (ONNX) format played a crucial role in model deployment within the inspection systems used at Controlar. ONNX facilitated the conversion of trained PyTorch models into a format compatible with real-time inference hardware utilised on the production line. Specifically, the trained model (‘best.pt‘) was converted to an ONNX model (‘best.onnx‘), enabling its integration into a real-time machine tasked with detecting PCB punctures in an industrial environment. The interoperability provided by ONNX allowed flexible deployment and reliable execution across diverse hardware platforms [10].

In summary, the selection of PyTorch, supported by ONNX for model deployment, reflected a careful evaluation process balancing ease of experimentation, performance considerations, and deployment readiness within Controlar's operational context.

3.3 Deployment Environment

The AI models developed in this dissertation were ultimately deployed onto a specialised industrial machine for real-time puncture detection in PCB components. This machine, illustrated in Figures 3.1 and 3.2, integrates hardware optimised for AI inference, highlighting the practical applicability of the selected AI framework and ONNX model format.

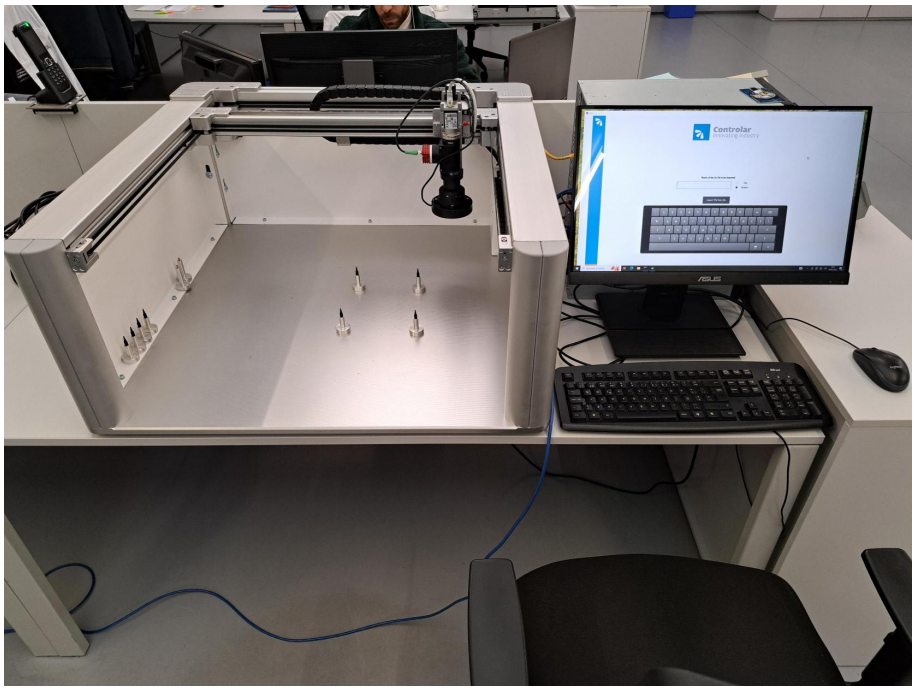


Figure 3.1: Industrial machine used for real-time PCB inspection.

Critical to the effective detection of small punctures in PCBs is the quality of the imaging system. As shown in Figure 3.3, the machine is equipped with specialised illumination and a precision camera system designed explicitly for industrial tasks. The lighting ensures uniform illumination, significantly enhancing image quality and visibility of small defects. Meanwhile, the camera's mobility and high-resolution capture capabilities allow precise scanning of PCBs, ensuring optimal conditions for the AI model to detect punctures reliably.

Together, these visuals demonstrate the dissertation's practical contribution, showcasing the alignment between theoretical objectives, chosen frameworks, and real-world industrial deployment.

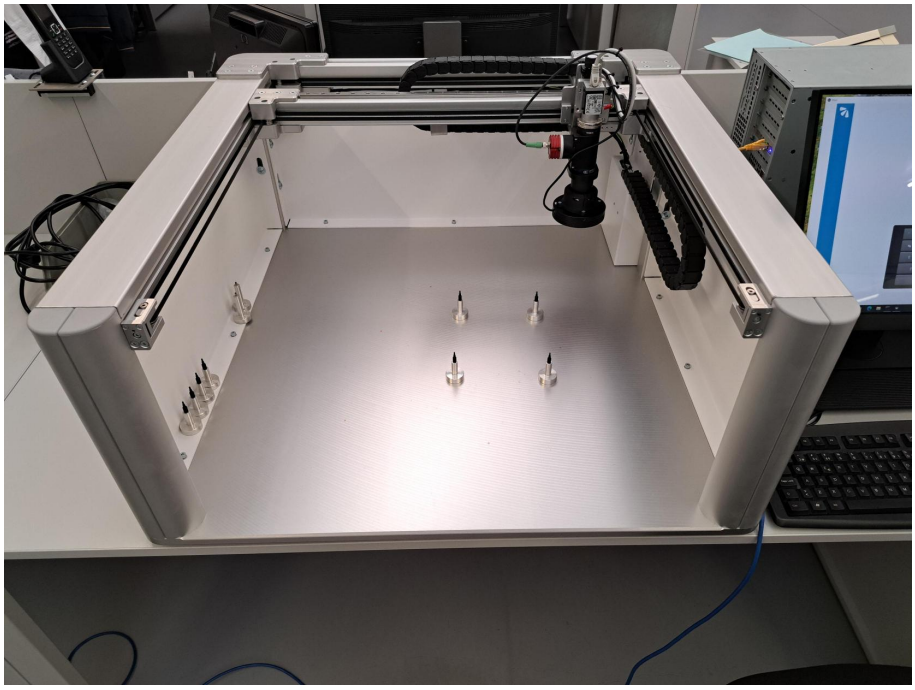


Figure 3.2: Close-up view of the inspection bed used to position PCBs during inspection.

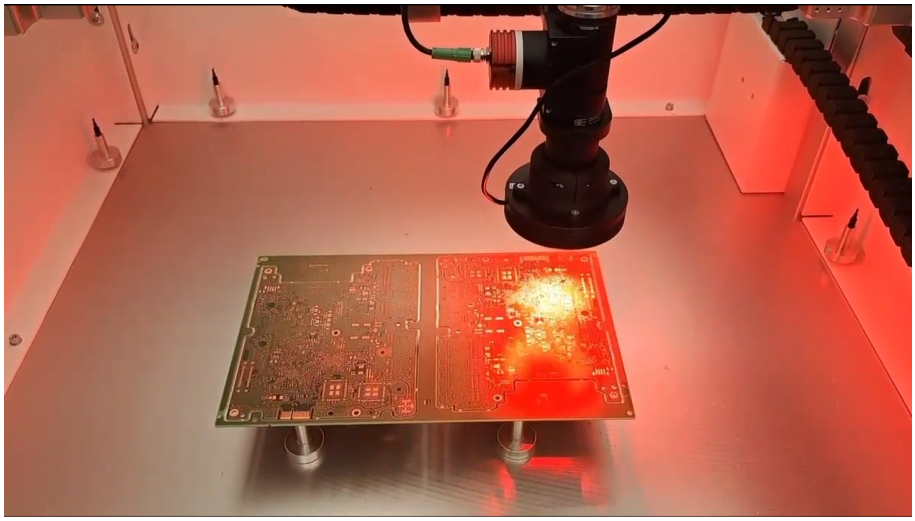


Figure 3.3: Detail of the machine's camera and lighting system scanning a PCB.

Chapter 4

Methodology

This chapter outlines the approach and methodologies used for training an object detection model specifically designed for small defect detection in images. The specific detection task involved identifying tiny punctures within Printed Circuit Boards (PCBs). Due to the extremely small size of these punctures, the task falls under the category of small object detection, posing unique challenges requiring specialised techniques and meticulous attention to detail.

4.1 Dataset Preparation and Format

4.1.1 Initial Dataset and Incremental Expansion

The initial dataset consisted of 5,291 409×489px images of PCBs, each annotated with puncture locations. The labels were originally provided in a semi-colon-separated text format, where each row followed the structure: filename;X;Y;area_of_puncture, e.g. 000001.png;247.52;149.46;230.0. This format, while informative, was not compatible with the YOLO training pipeline.

To prepare the dataset for training, a custom Python script was developed to convert the original dataset labels into the YOLO format. The original labels were stored in a plain text file where each line contained the image filename, puncture centre coordinates (x, y), and the area of the puncture. However, YOLO requires bounding boxes defined by the class index and normalised values for the centre and dimensions of the bounding box.

The script iterates through each annotation, calculates the radius from the provided area (assuming circular shape), applies a configurable enlargement factor, and outputs the normalised YOLO format as: class x_center y_center width height

The enlargement factor was introduced to ensure that the bounding box sufficiently covers the entire puncture, considering possible annotation uncertainty. The script also handles errors gracefully by checking file existence and catching exceptions during image processing.

Figure 4.1 presents the flowchart of the annotation conversion process. This diagram illustrates the key steps in the Python script that converts the dataset's annotation format into the YOLO-friendly format, ensuring compatibility with the model's training pipeline.

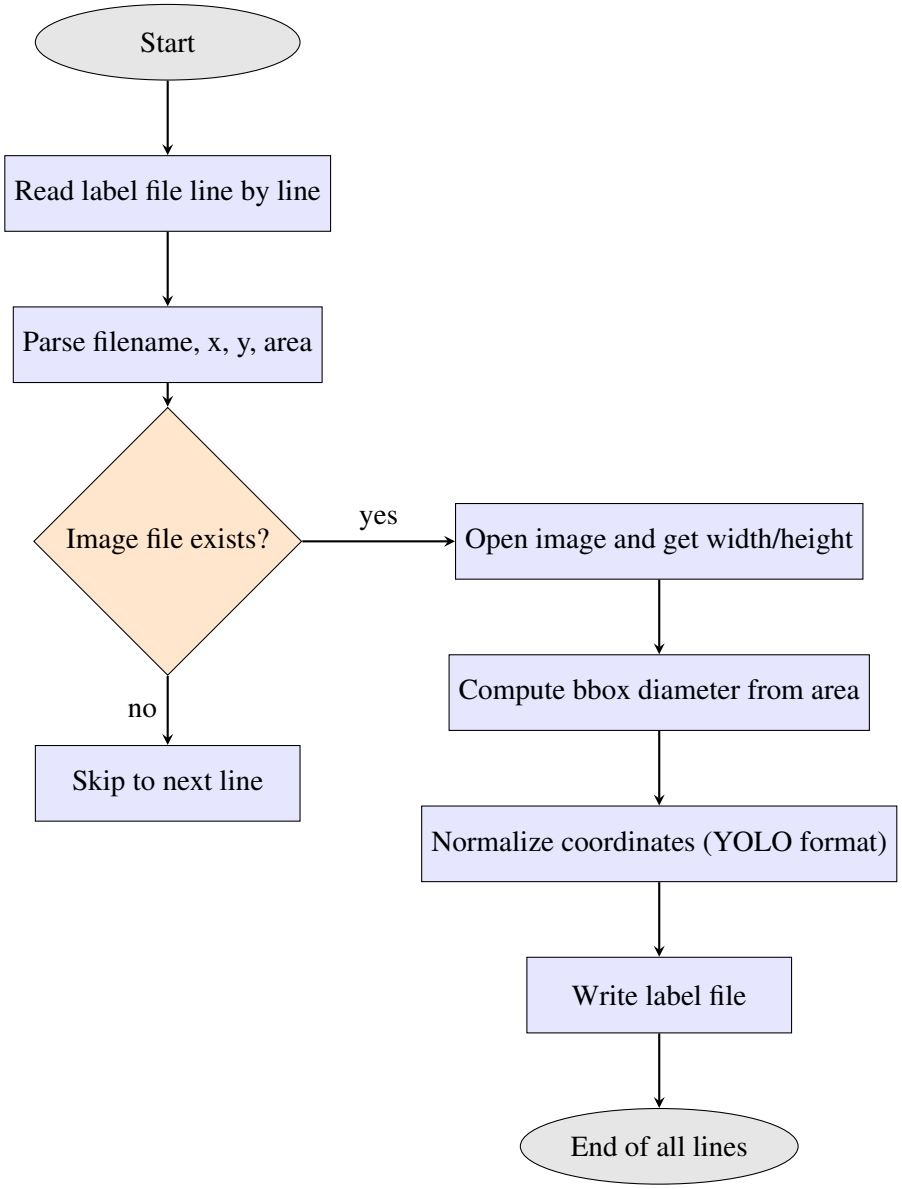


Figure 4.1: Flowchart of the annotation conversion process to YOLO format

For instance, the entry above became 0 0.506176 0.365428 0.034995 0.041840.

Table 4.1: Example of Label Format Conversion

Original Format	**Converted YOLO Format**
000001.png;247.52;149.46;230.0	0 0.506176 0.365428 0.034995 0.041840

Note: The YOLO format specifies the class index followed by the normalized values for bounding box center coordinates and dimensions. All values are relative to the width and height of the image.

This transformation ensured that the annotations were correctly interpreted by the YOLO model and enabled consistent handling of different image resolutions.

Additional images and corresponding labels were incrementally added throughout the project to improve the generalisation capabilities of the model. This iterative data enrichment was essential due to variability in PCB appearance—including differences in color, materials, and the presence of welding artifacts—requiring the model to be robust across diverse visual contexts. Figure 4.2 shows an example of the puncture images that comprise the dataset:

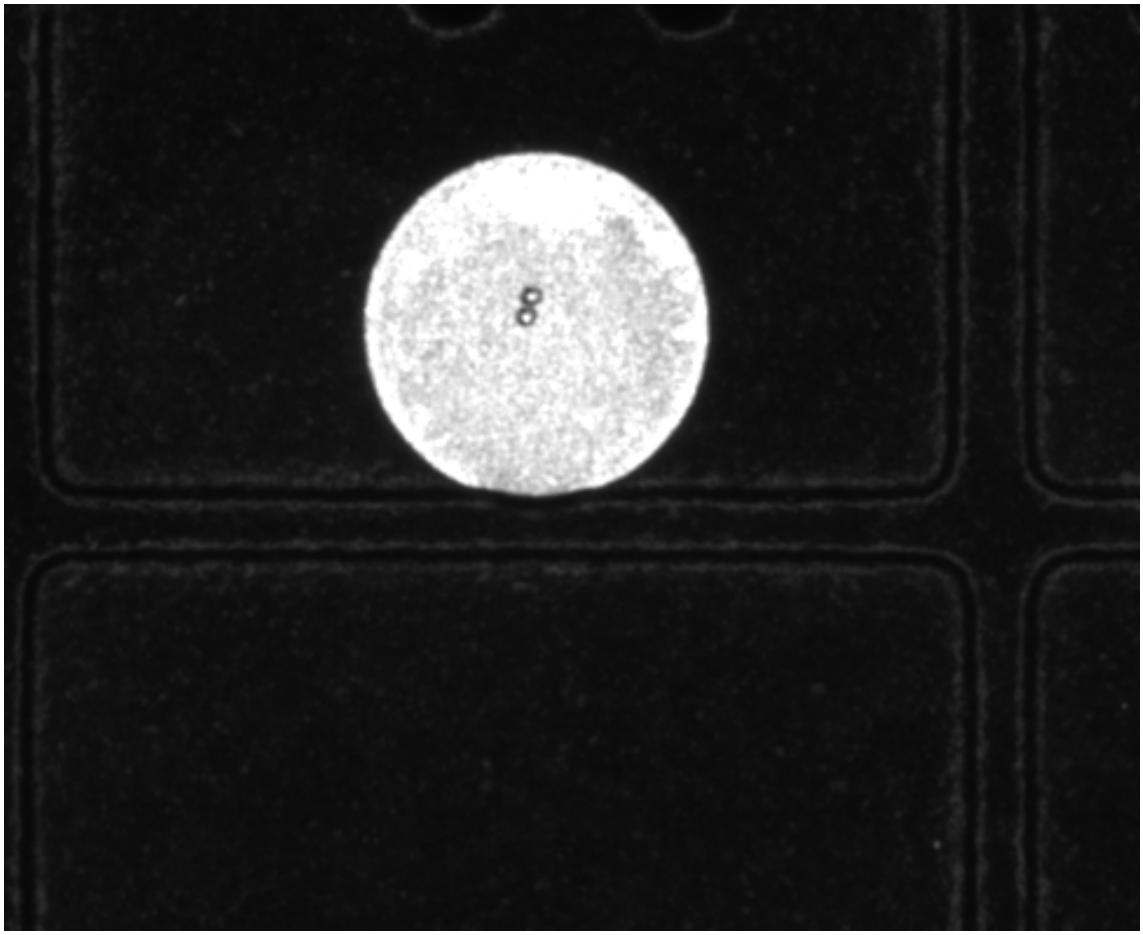


Figure 4.2: Image of a puncture from the dataset

Annotations were carefully verified after conversion to ensure accuracy and consistency, which are critical for training object detection models, particularly for small object scenarios like puncture detection.

To ensure the consistency and completeness of the dataset, another Python script was created to verify that each image file had a corresponding label file and vice versa. This script can be used on any of the dataset partitions (train, validation, or test) and cross-checks filenames between the images/ and labels/ folders. It outputs whether all images are properly labeled or lists any mismatches. This step helped prevent training errors and ensured that no unlabeled data was unintentionally introduced into the pipeline.

Figure 4.3 shows the flow chart of the dataset consistency check process. This diagram outlines the steps of the Python script that is used to verify that all images have corresponding label files, ensuring the integrity of the dataset before training the model.

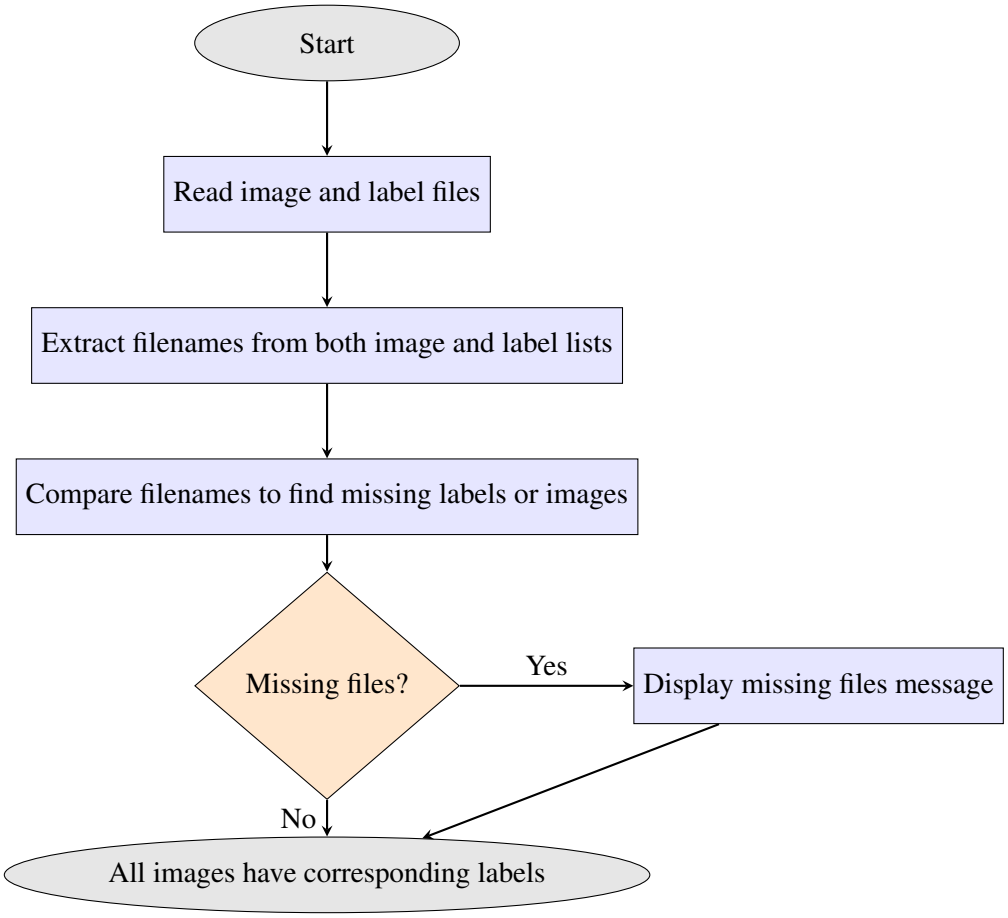


Figure 4.3: Flowchart of the dataset consistency check script to verify matching images and labels

As the training progressed, additional images and labels were incrementally incorporated to enhance the generalisation capability of the model. This approach was critical because PCBs can differ substantially in terms of colour, materials, and manufacturing variations, including the presence of welding residues that could influence image appearance. By continuously expanding the dataset, the model was exposed to a broader range of visual scenarios, significantly improving its robustness and effectiveness in accurately detecting punctures across diverse PCB conditions.

4.1.2 Image Preprocessing

No explicit image preprocessing was applied prior to training beyond adjusting the input resolution through the YOLOv8 configuration. The original images were maintained in their raw form, and resizing was handled internally during model training. YOLOv8 resizes input images to the specified training resolution using a letterbox resize method, which scales the image while preserving its original aspect ratio and adds padding as needed [33]. This approach prevents distortion

and ensures that the spatial relationships within the image—particularly important for small object detection—are preserved intact.

4.1.3 Dataset Partitioning and YOLO Format

The processed dataset was partitioned into three subsets: 70% for training, 20% for validation, and 10% for testing. Each subset followed the directory structure required by the YOLO framework, consisting of two nested folders: one for the images and another for the corresponding label files. This structure enabled compatibility with Ultralytics' training engine and ensured efficient batch loading and annotation mapping during training. All subsets used the same format and annotation conventions, maintaining consistency across the dataset and evaluation pipeline. Figure 4.4 illustrates the directory structure of the processed dataset:

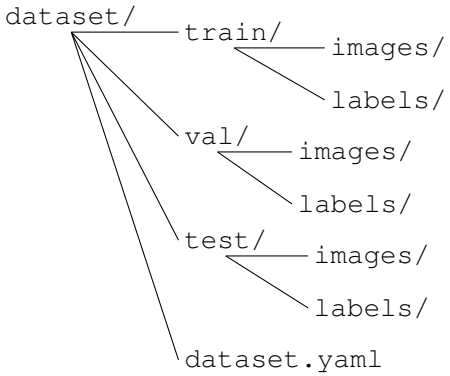


Figure 4.4: Dataset folder structure.

4.1.4 Data Augmentation Techniques

Data augmentation was implemented with a conservative strategy to avoid degrading the performance of the model in small object detection tasks. Certain augmentation techniques, especially those that involve geometric distortions or aggressive transformations, can negatively affect small object detection by making targets harder to distinguish [2].

Initially, a broad set of augmentation techniques was considered to enhance the model's generalization. However, the augmentation strategy was refined to focus on methods that preserved the integrity of small punctures on PCBs. Some techniques were excluded due to their potential to obscure small objects, while others, such as the copy_paste technique, were removed not because of their adverse impact on small object detection, but due to their limited relevance to the specific requirements of this application The final augmentation settings are presented in Table 4.2.

Table 4.2: Final Data Augmentation Configuration

Augmentation	**Description**
`flip_lr = 0.5`	Horizontal flipping to increase orientation diversity.
`hsv_h = 0.015`	Color jittering to introduce hue variation.
`hsv_s = 0.5`	Color jittering to introduce saturation variation.
`hsv_v = 0.4`	Color jittering to introduce brightness variation.
`translate = 0.05`	Minor translation to simulate slight positional shifts.
`erasing = 0.3`	Random erasing to generalize over potential occlusions.

Conversely, the augmentations described in Table 4.3 were disabled due to their potentially harmful effect on this specific small object detection task:

Table 4.3: Disabled Augmentations

Augmentation	**Description**
`mosaic = 0.0`	Combines 4 different images into one by placing them in a grid.
`mixup = 0.0`	Blends two images together by overlaying one on top of another.
`copy_paste = 0.0`	Copies object instances from one image and pastes them into another.

Although these augmentation techniques can be effective in general object detection tasks, their application in small object detection, such as puncture identification in PCBs, requires caution. Mosaic and mixup, in particular, often reduce the effective size and clarity of small objects by blending multiple images or scaling them down, making it harder for the model to learn precise spatial features. Copy-paste, while potentially beneficial in some small object contexts (e.g., object repetition or class balancing), was deemed less relevant for this specific use case, as punctures typically appear in fixed positions with limited variation. For these reasons, the techniques were excluded to maintain object visibility and annotation consistency throughout training.

4.2 Framework and Model Selection

Initially, the TensorFlow framework was explored to develop and train object detection models due to its wide adoption in the industry [27]. However, significant challenges arose during the conversion of the pre-trained PyTorch-based YOLO model provided by Ultralytics into a TensorFlow-compatible format. The absence of direct TensorFlow support from Ultralytics necessitated extensive and complex format conversions, leading to compatibility issues and increased overhead. Consequently, a decision was made to use PyTorch, as it directly supports the YOLOv8 model and offers advantages suited to research-oriented projects, such as dynamic computational graphs, ease of experimentation, and strong community support [7].

Even after transitioning to PyTorch, the initial experiments were conducted using the YOLOv8s (small) model, given its faster training times and lower resource demands. However, it quickly became apparent that the available computational resources could support a more complex architecture, as training was proceeding rapidly with minimal signs of system bottlenecks. Additionally, the initial results left considerable room for improvement in detection accuracy, especially in the context of small object detection. Based on these observations, the YOLOv8m (medium) model was selected for subsequent experiments, providing a better balance between performance and computational cost while still maintaining compatibility with the available hardware setup.

4.3 Training Environment

Model development and experimentation were carried out using Google Colab, utilising a local runtime enabled via Docker containers. This configuration allowed for a versatile training environment, circumventing Colab's typical constraints by providing local computing resources while using the familiar Colab interface. Docker was pivotal in ensuring reproducibility and consistency across experiments by encapsulating dependencies, configurations, and environment setups within containers.

4.3.1 Computational Resources

Training was conducted on a Lenovo LOQ 15 laptop equipped with a 12th-generation Intel Core i5 processor, 16 GB of RAM, and running Windows 11. The GPU used for accelerating model training was an Nvidia GeForce RTX 4050, which featured 6 GB of VRAM. This GPU provided sufficient power to manage complex computations required by the YOLOv8m model, albeit with certain limitations in VRAM, which affected maximum batch sizes and input image resolutions.

4.3.2 Docker and GPU Utilisation

Docker was instrumental in effectively managing GPU resources, ensuring a stable and consistent training environment, as shown in Figure 4.5. Initially, model training was performed using a smaller input resolution of 640 pixels, resulting in relatively short training durations (approximately 0.470 hours for 20 epochs). However, due to the high training loss values identified by the analysis of the YOLO loss metrics, including box loss, classification loss, and distribution focal loss, adjustments were necessary. Consultation with an Ultralytics engineer indicated that optimal image sizes for small object detection should ideally be around 1280 pixels; however, GPU VRAM constraints limited practical configuration to 1024-pixel input resolution with a batch size of 8. Across multiple training attempts, some of which were discarded due to suboptimal results, progressive refinements in configuration led to improvements in performance metrics. Incremental adjustments to input resolution and batch size significantly improved model accuracy, but also increased computational load, elevating training times from 1.74 hours initially to 60.3 hours for

the final successful training run. This progressive fine-tuning allowed careful monitoring and understanding of loss metrics, optimising the balance between performance accuracy and training efficiency.

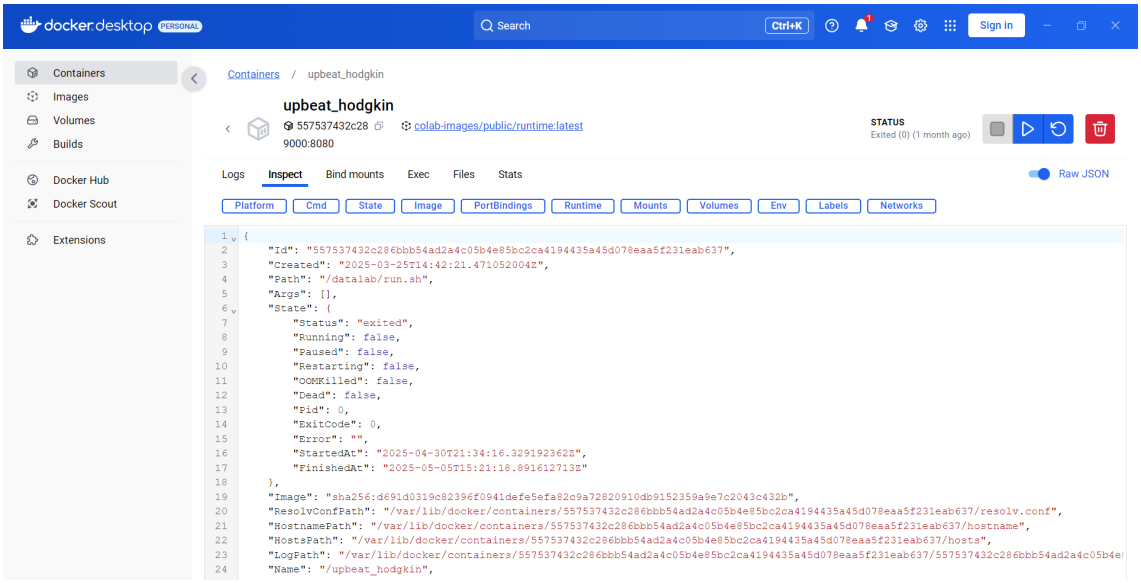


Figure 4.5: Docker Desktop

4.4 Model Architecture

The YOLOv8 architecture, developed by Ultralytics, is a state-of-the-art deep learning model specifically optimized for efficient and accurate real-time object detection. As illustrated in Figure 4.6, the model is structured into three main components: the backbone, neck, and head [11].

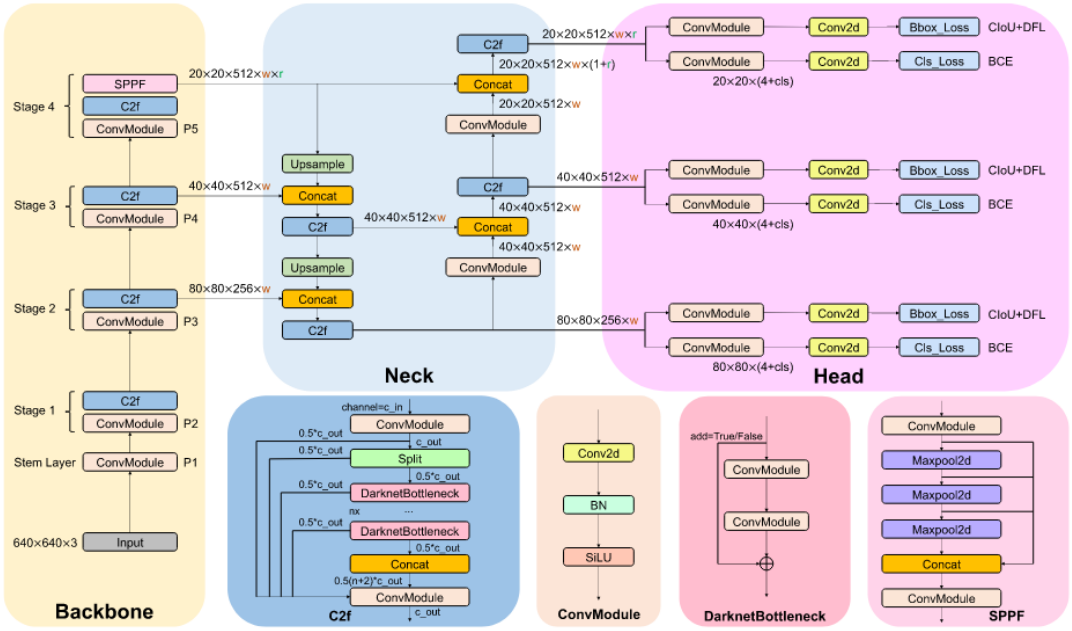


Figure 4.6: Detailed illustration of YOLOv8 model architecture [11].

The **backbone** is responsible for feature extraction from input images. YOLOv8 employs a CSPDarknet53 backbone, integrating Cross Stage Partial connections (CSP) to enhance learning capacity while maintaining computational efficiency. CSPNet partitions feature maps, processes them separately, and merges them, thereby reducing computational complexity and improving inference speed.

The **neck** of YOLOv8 utilises a Path Aggregation Network (PANet) combined with a Spatial Pyramid Pooling (SPP) module. PANet facilitates effective information flow and enhances spatial information reuse across multiple scales, crucial for detecting objects of varied sizes. The SPP module captures spatial features at multiple resolutions, further improving detection capability, particularly for small objects, essential for detecting minute PCB punctures.

The **head** of YOLOv8 consists of multiple detection layers designed to predict bounding box coordinates, class probabilities, and objectness scores directly from aggregated neck features. Notably, YOLOv8 adopts anchor-free detection, simplifying the training process and enhancing generalisation across diverse datasets and scenarios.

Recent research by Bajpai et al. [1] reinforces YOLOv8's effectiveness, particularly highlighting the YOLOv8m variant for challenging detection tasks under difficult conditions, such as underwater environments. Their findings demonstrate YOLOv8m's superior performance over traditional detectors like Faster R-CNN and SSD in efficiency and accuracy, achieving notable F1 scores and improved inference speeds, emphasising its suitability for industrial inspection applications.

The YOLOv8 model family offers multiple variants: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra large), each variant increasing in depth, parameter count, and computational complexity. These variants allow practitioners to

select a model balancing accuracy and inference speed according to hardware and application requirements [33]. Initially, the lightweight YOLOv8s model was tested due to its rapid training and computational efficiency. However, insufficient performance in accurately detecting small defects, indicated by persistently high loss metrics, called for a transition to YOLOv8m. The YOLOv8m variant offers greater architectural complexity, delivering improved accuracy and capturing intricate details essential for reliable small-object detection in PCBs.

Given the complexity of detecting puncture defects and existing hardware constraints, YOLOv8m represented an optimal balance, providing sufficient learning capacity without sacrificing real-time inference performance.

The architectural design is thoroughly documented by Ultralytics [33], providing transparency and facilitating reproducibility.

4.5 Model Evaluation and Metrics

To assess the performance of the trained model, a set of standard object detection metrics was monitored throughout the training process. These metrics included:

Box loss: Measures how accurately the predicted bounding boxes match the ground truth locations. Lower values indicate better localization.

Classification loss: Evaluates how well the model classifies detected objects into the correct class (in this case, there is only one class: puncture).

Distribution Focal Loss (DFL): Enhances bounding box regression by focusing on the distribution of potential box positions, improving spatial precision.

Precision: The proportion of correctly predicted punctures among all predicted punctures.

Recall: The proportion of correctly predicted punctures out of all ground truth punctures.

mAP@0.5 and mAP@0.5:0.95: Mean Average Precision at IoU thresholds. mAP@0.5 uses a fixed IoU of 0.5, while mAP@0.5:0.95 averages the results across multiple thresholds from 0.5 to 0.95.

All of these metrics are widely used in object detection evaluation [33].

Figure 4.7 shows the evolution of these metrics over 50 epochs for both the training and validation sets. The plotted values enabled continuous monitoring of model convergence and performance.

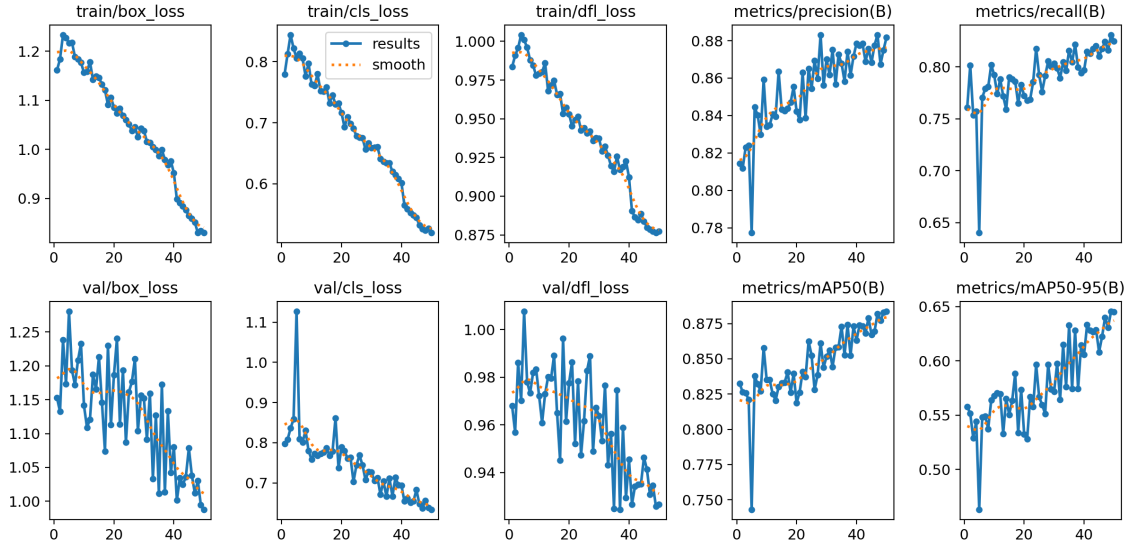


Figure 4.7: Training and validation metrics across 50 epochs. Loss curves are shown for bounding box (box_loss), classification (cls_loss), and distribution focal loss (dfl_loss), alongside evaluation metrics including precision, recall, and mean average precision (mAP@0.5 and mAP@0.5:0.95).

Beyond serving as final evaluation indicators, these metrics were essential during training as diagnostic tools. By tracking the evolution of loss curves and detection performance, it was possible to identify issues early—such as poor generalisation or underfitting—and take corrective action. For instance, persistently high classification or DFL loss across both training and validation datasets suggested suboptimal visibility of small objects in the feature maps.

This insight led to a key architectural adjustment: increasing the input image resolution from 640 to 1024 pixels. This change enabled small puncture defects to be more clearly represented within the model's internal feature maps, resulting in lower loss values and improved precision and recall.

In addition to individual metrics, Ultralytics' fitness score - a composite metric used internally by the training engine - was used to identify the model checkpoint that performed best. This score combines multiple metrics (typically precision, recall, and mAP) in a weighted manner to guide model selection[33]. It proved valuable in balancing trade-offs and ensuring that the best epoch was chosen for model export.

Ultimately, the use of detailed evaluation metrics throughout the training cycle was instrumental not only in validating model performance but also in informing important configuration decisions.

4.5.1 Confusion Matrix

In addition to the previously discussed metrics, confusion matrices were employed as diagnostic tools to assess the model's predictive performance more directly. A confusion matrix provides a straightforward visualisation of the model's predictions compared against the actual annotations, categorising results into True Positives (TP), False Positives (FP), True Negatives (TN), and False

Negatives (FN)[18]. In the specific context of this project, since puncture detection represents a single-class scenario, the confusion matrix mainly focused on identifying true detections (TP), missed detections (FN), and incorrect predictions (FP). This visualisation facilitated the rapid identification of cases where the model might be systematically missing punctures or generating false alarms, guiding subsequent training refinements and hyperparameter adjustments.

Figure 4.8 provides an illustrative confusion matrix example derived from one of the trained models. It highlights the balance between correct detections and misclassifications, underscoring the model's overall strong performance in accurately identifying puncture defects.

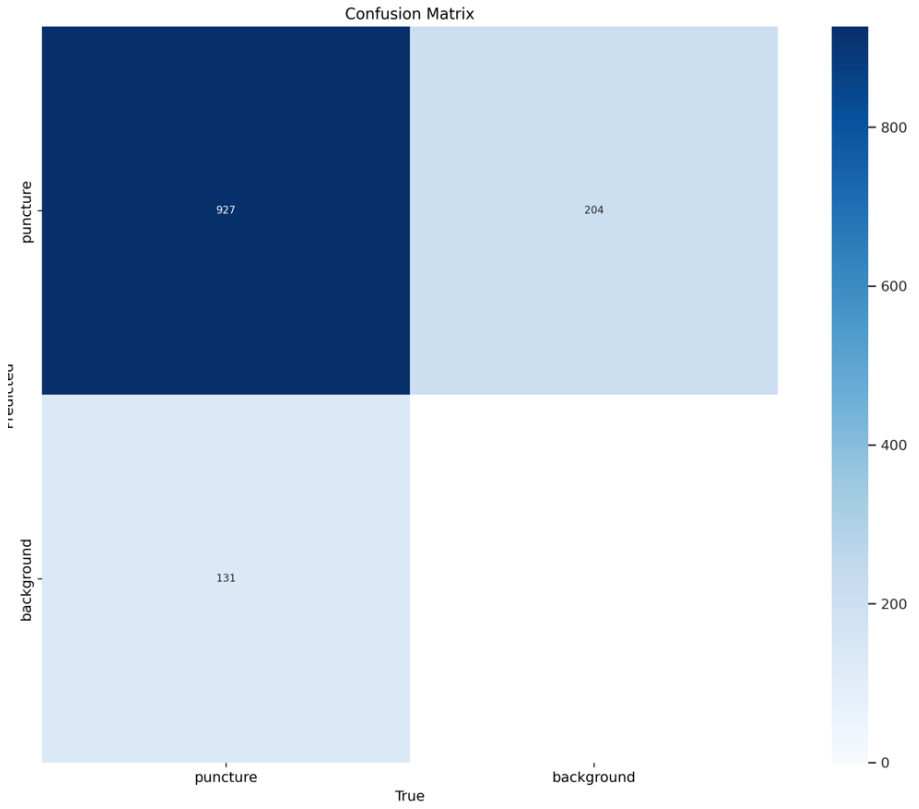


Figure 4.8: Example of a confusion matrix from training validation, demonstrating the distribution of true positives, false positives, and false negatives in puncture detection.

4.6 Fine-tuning Procedure

The fine-tuning process employed in this dissertation followed a structured approach based on transfer learning principles, using pretrained weights as a starting point. Special emphasis was placed on data augmentation technique and careful hyperparameter selection to maximize the model's sensitivity and precision for detecting extremely small puncture defects in PCBs [12].

4.6.1 Initial Fine-Tuning and Challenges

Following the shift from TensorFlow to PyTorch, initial training began with the YOLOv8s model, using a batch size of 8 and the default YOLO input image size of 640 pixels. Despite multiple attempts, the model exhibited persistently high training losses — particularly the bounding box regression loss (box_loss = 1.182, cls_loss = 0.766, dfl_loss = 0.902), as illustrated in Figure 4.9.

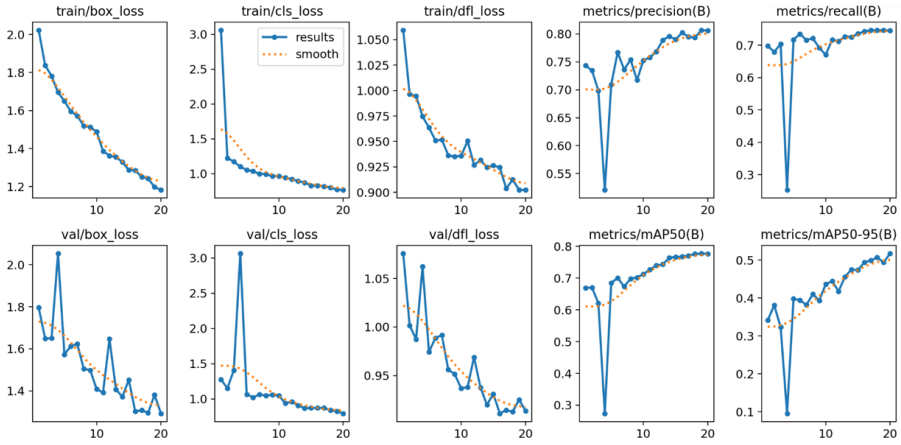


Figure 4.9: Initial YOLOv8s training.

High box_loss indicated poor alignment between the model's predicted bounding boxes and actual object locations, suggesting inadequate learning of the dataset.

To address this, training was repeated using the larger YOLOv8m model, anticipating that greater complexity would improve detection accuracy. Although this reduced losses slightly, box_loss remained problematic, consistently above 1. Additional measures such as increasing batch size to 16, incorporating various augmentations, and adjusting parameters related to lighting and colour variations provided marginal improvement (box_loss: 1.03; cls_loss: 0.64; dfl_loss: 0.84), yet box_loss still did not fall below the desired threshold of 1. Consequently, training was limited to 50 epochs per iteration until the underlying issue could be resolved.

4.6.2 Expert Consultation and Methodological Adjustments

To overcome persistent training challenges, an Ultralytics engineer was consulted. The engineer recommended significantly increasing the input image resolution to 1280 pixels and reducing the batch size to accommodate GPU constraints. Additionally, it was suggested to disable mosaic augmentation, a method known to negatively impact small object detection accuracy.

Given hardware limitations (Nvidia RTX 4050 GPU with 6 GB VRAM), extensive experimentation identified 1024 pixels as the maximum feasible input size, paired with a batch size of 8. A subsequent 50-epoch training run using these adjusted settings lasted approximately 60.3 hours and finally yielded acceptable training losses and metrics (box_loss = 0.8299; cls_loss = 0.5197; dfl_loss = 0.8774). The associated evaluation metrics also significantly improved, demonstrating

clear progress: Precision = 87.7%, Recall = 83.2%, mAP@50 = 88.3%, mAP@50-95 = 64.6%, and a Fitness Score of 0.59.

These improvements clearly highlighted that increased input image resolution improved small object visibility in feature maps, thereby significantly lowering box_loss values.

4.6.3 Cross-Validation Procedure

Following these promising results, a cross-validation procedure was adopted to rigorously validate model generalisation and to prevent overfitting. A five-fold cross-validation was initiated using the best-performing YOLOv8m model obtained previously. To speed up this process, augmentations were initially disabled, expecting shorter training durations.

However, after completing two folds (each requiring roughly 60 hours), the minimal variation between fold performances indicated that additional cross-validation folds might not significantly improve insight relative to the substantial computational effort required. Consequently, cross-validation was terminated early, and the fold demonstrating the lowest loss values and best evaluation metrics was selected for final model refinement.

Subsequently, a final fine-tuning session was conducted using the chosen data fold, this time re-enabling the previously optimal data augmentation configuration (horizontal flipping probability, HSV augmentations, translation, random erasing), for 100 epochs at a 1024-pixel input resolution and batch size of 8.

4.6.4 Final Model Performance

The final training, lasting over 60 hours, achieved the best overall performance observed throughout the project, as shown in Table 4.4:

Metric	**Value**
Precision	87.4%
Recall	84.6%
mAP@50	89.9%
mAP@50-95	72.4%
Fitness Score	0.74

Table 4.4: Final Model Performance Metrics

The final recorded loss values demonstrated substantial improvement, as shown in Table 4.5:

Loss Type	Value
Training - box_loss	0.1898
Training - cls_loss	0.1359
Training - dfl_loss	0.7583
Validation - box_loss	0.8801
Validation - cls_loss	0.5903
Validation - dfl_loss	0.9604

Table 4.5: Final Training and Validation Loss Values

These metrics confirm the successful fine-tuning of the model and its readiness for practical deployment.

4.6.5 Inference Validation and Model Deployment

To validate the model's practical performance on unseen data, a simple graphical interface application was developed using Python's Tkinter library, enabling inference testing on individual images or entire folders of PCB images not encountered during training. The application confirmed that the trained model accurately detected punctures under realistic, operational conditions, providing sufficient confidence for deployment. Figure 4.10 provides a visual representation of the inference interface.

Figure 4.10: Simple graphical application created for inference validation, showcasing model performance on unseen data.

Beyond visual inference, the application was also designed to automatically generate a CSV file containing the inference results, including bounding box coordinates, detection confidence scores, and class labels. These results were saved systematically to an *inference* folder, facilitating convenient post-analysis and review (Figures 4.11 and 4.12). The generated CSV results, as illustrated in Figure 4.11, provide a structured overview of the model's predictions, which can easily be analysed or integrated into downstream processes.

Inference results (CSV)

image	klass	confidence	x_center	y_center	bbox_w	bbox_h
000000.png	puncture	0.387	253.4	244.4	11.3	11.3
000001.png	puncture	0.824	236.9	232.7	13.6	13.7
000003.png	puncture	0.862	227.0	165.5	11.9	11.8
000004.png	puncture	0.770	250.9	217.3	12.8	12.9
000004.png	puncture	0.638	280.2	231.9	10.8	10.8
000005.png	puncture	0.693	208.7	224.7	15.0	14.8
000005.png	puncture	0.309	209.6	226.7	17.4	17.7
000005.png	puncture	0.252	207.8	223.2	12.9	11.7
000006.png	puncture	0.662	223.1	235.4	10.9	11.1
000008.png	puncture	0.819	230.0	202.7	11.9	11.7
000009.png	puncture	0.828	239.5	170.6	11.4	11.2
000010.png	puncture	0.547	245.5	190.8	15.2	15.6
000011.png	puncture	0.872	275.1	250.8	12.6	12.7
000012.png	puncture	0.849	216.0	230.2	12.2	12.2
000014.png	puncture	0.802	212.4	201.1	14.4	14.0
000015.png	puncture	0.815	271.3	216.0	11.7	11.7
000016.png	puncture	0.563	261.4	216.8	6.0	5.8
000017.png	puncture	0.876	190.2	171.2	12.5	12.5
000018.png	puncture	0.861	251.0	188.0	11.7	11.7
000019.png	puncture	0.753	234.4	206.5	11.8	11.7
000020.png	puncture	0.875	259.0	181.2	12.2	12.1
000021.png	puncture	0.798	264.9	190.9	11.0	10.7
000021.png	puncture	0.586	249.5	193.5	12.2	12.4
000022.png	puncture	0.872	255.4	217.0	12.0	12.0
000023.png	puncture	0.834	219.3	243.8	12.0	11.9
000024.png	puncture	0.257	239.7	245.5	6.3	6.1
000025.png	puncture	0.862	217.8	196.0	11.0	11.0
000026.png	puncture	0.858	272.9	175.0	11.7	11.9
000027.png	puncture	0.397	262.5	258.3	12.2	12.4
000028.png	puncture	0.529	216.1	208.1	11.3	11.6
000029.png	puncture	0.857	234.3	174.8	13.6	13.7
000030.png	puncture	0.298	267.5	199.4	11.3	10.8

Figure 4.11: CSV file generated.

All inference outputs, including annotated images and CSV files, were systematically saved in the designated *inference* directory, as demonstrated in Figure 4.12. This allowed for efficient management and review of inference results.

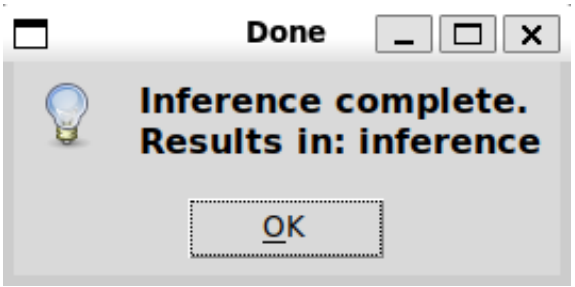


Figure 4.12: Automatically generated folder storing inference results.

The practical results obtained through this validation process were satisfactory to company supervisors, affirming the viability of the trained model for real-world deployment. Subsequently, the finalised model was exported to the ONNX format, ensuring compatibility with the machine.

4.6.6 Illustrative Training Graphs

Throughout the fine-tuning procedure, analysis of loss curves and evaluation metric trends played a critical role in identifying training challenges and informing parameter adjustments. Figures 4.9,

4.13, and 4.14 illustrate representative training sessions, clearly demonstrating improvements in loss metrics and precision/recall following methodological adjustments.

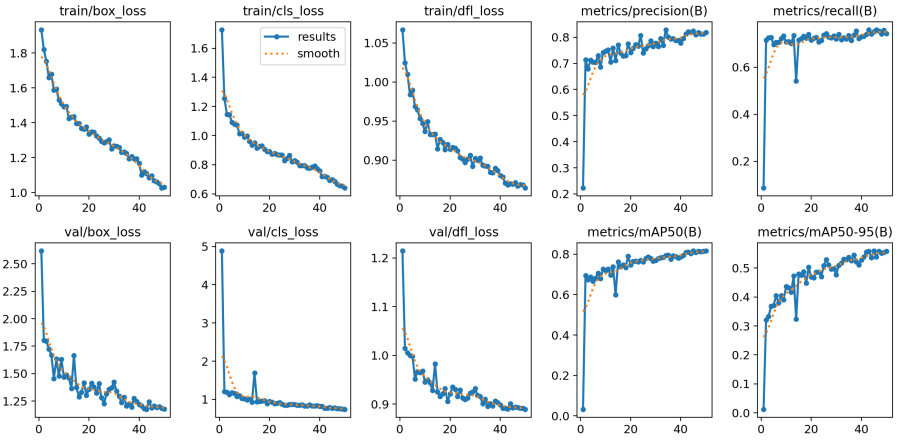


Figure 4.13: Intermediate training with YOLOv8m and adjusted hyperparameters showing moderate improvement.

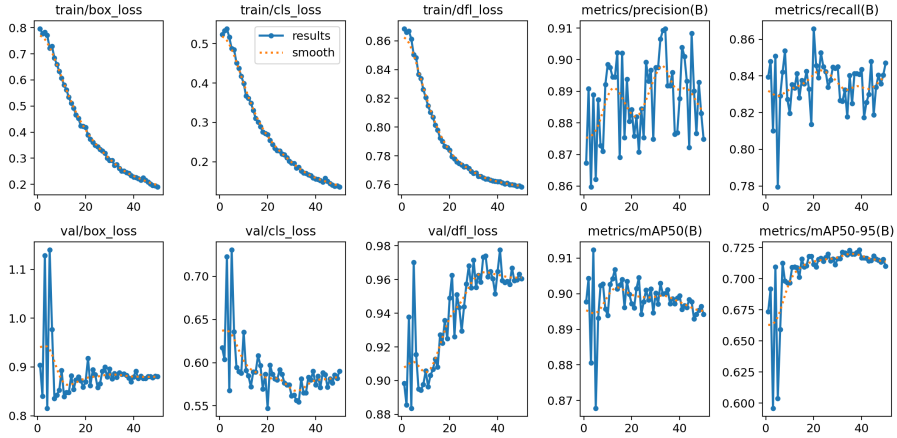


Figure 4.14: Final fine-tuning session demonstrating significantly improved loss metrics and evaluation scores.

Collectively, these figures and the detailed narrative provided here offer clear insights into the iterative and evidence-driven approach taken to successfully fine-tune the object detection model.

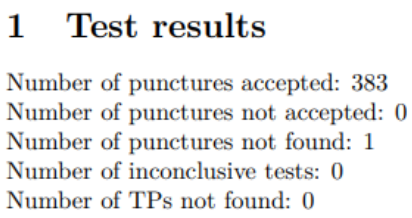
Chapter 5

Result and Discussion

The developed model was successfully integrated into the company's real-time industrial inspection system. To facilitate integration, the previously developed inference script was merged with existing operational software, ensuring that all necessary libraries, dependencies, and the ONNX model file (*best.onnx*) were correctly installed on the inspection machine.

The operational performance of the model is documented through detailed inspection reports generated by the machine after analysing each PCB. Figures 5.1, 5.2, and 5.5 illustrate representative excerpts from these reports.

As shown in Figure 5.1, the model typically exhibited high reliability, accurately detecting the vast majority of punctures. For instance, one detailed report explicitly documents the detection of 383 punctures, with only a single missed detection, indicating excellent overall accuracy and reliability under real-world conditions.


1 Test results

Number of punctures accepted: 383
Number of punctures not accepted: 0
Number of punctures not found: 1
Number of inconclusive tests: 0
Number of TPs not found: 0

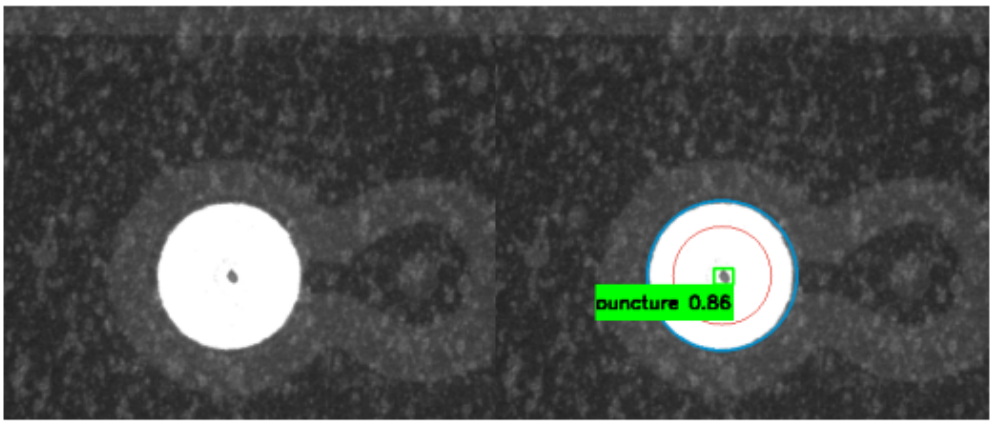
Figure 5.1: Inspection report summary showing overall detection performance, including the number of punctures detected and missed.

5.1 Representative Detection Examples

To better illustrate the model's detection capabilities, several representative inspection images from actual reports are provided. Figures 5.2, 5.3, and 5.4 show clear examples where the model successfully identified punctures with high confidence, exemplifying typical strong detection performance.

2.28 TP: 507

Classification: PUNCTURE ACCEPTED
Distance (mm) between TP center and puncture center: (-0.01, -0.01)
Area (mm²) of puncture(s): 0.015

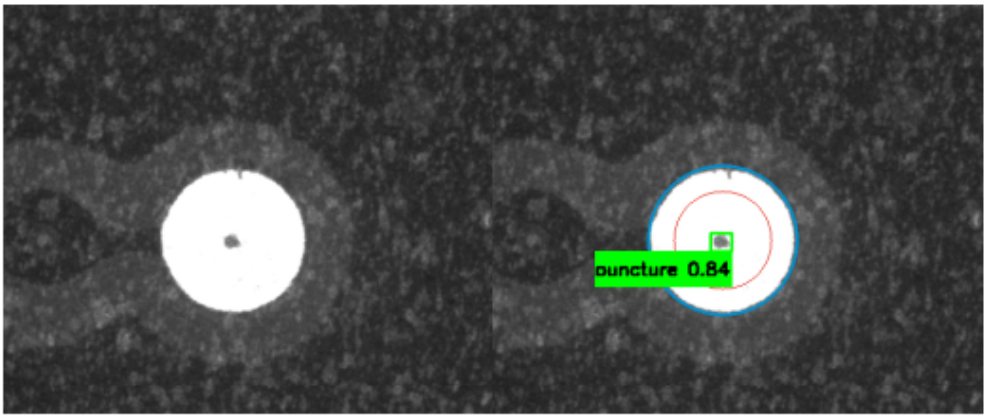


(a) Test point image (b) Inspection image

Figure 5.2: Example of a puncture correctly detected in Report 1, with high confidence in the detection.

2.7 TP: 388

Classification: PUNCTURE ACCEPTED
Distance (mm) between TP center and puncture center: (0.01, -0.01)
Area (mm²) of puncture(s): 0.0187

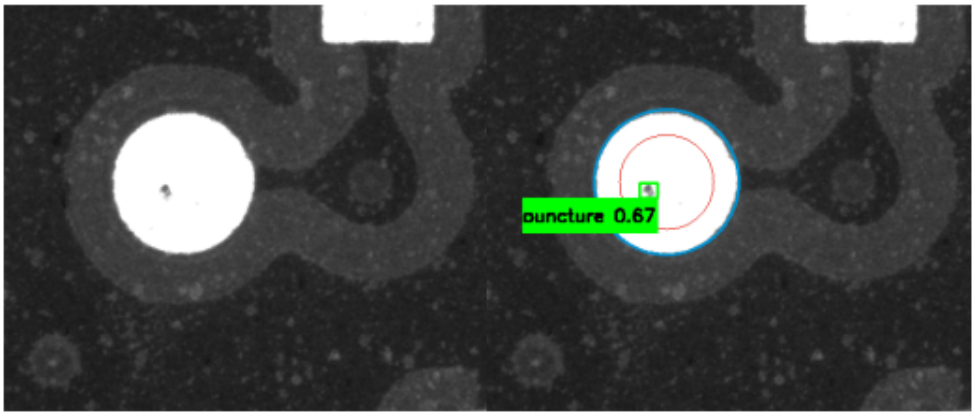


(a) Test point image (b) Inspection image

Figure 5.3: Example of a puncture correctly detected in Report 1, showing the bounding box and model's confidence score.

2.324 TP: 438

Classification: PUNCTURE ACCEPTED
Distance (mm) between TP center and puncture center: (0.13, -0.06)
Area (mm²) of puncture(s): 0.0134



(a) Test point image (b) Inspection image

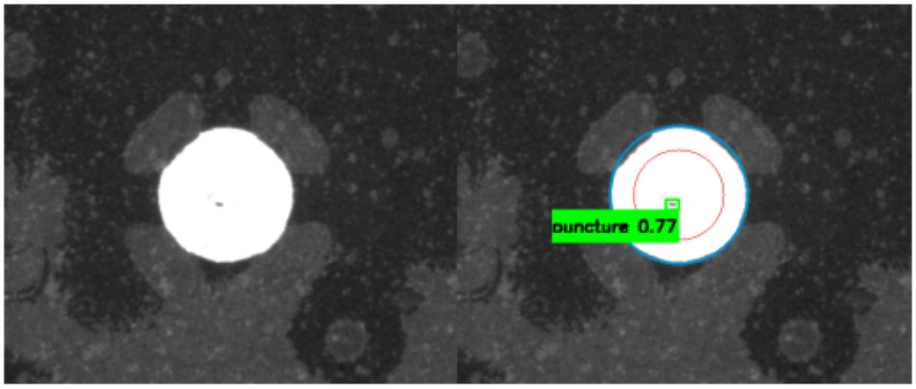
Figure 5.4: Example of a correctly detected puncture (Report 2).

5.2 Subjective Classification Examples

Despite generally strong performance, occasional challenging cases emerged due to variability in PCB types. These instances sometimes required manual intervention through a process termed "subjective classification." This procedure involves minor adjustments to the image acquisition parameters, allowing the model to detect challenging punctures more reliably. Examples of punctures identified through subjective classification are provided in Figures 5.5, 5.6, and 5.7.

2.240 TP: 337

Classification: PUNCTURE ACCEPTED
Distance (mm) between TP center and puncture center: SUBJECTIVE CLASSIFICATION
Area (mm²) of puncture(s): SUBJECTIVE CLASSIFICATION

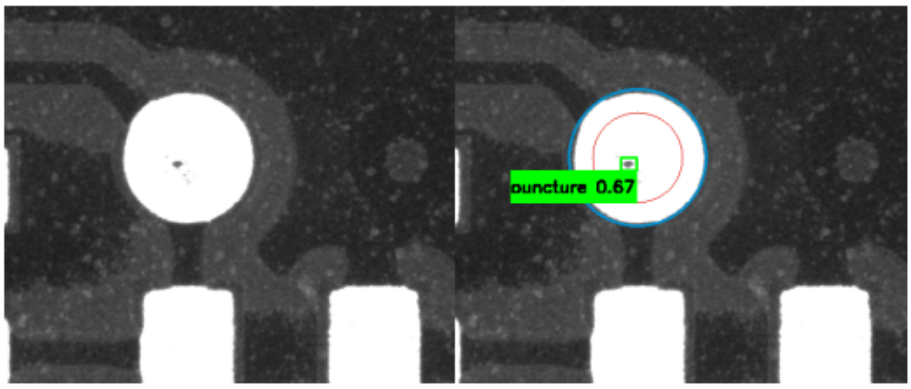


(a) Test point image (b) Inspection image

Figure 5.5: Example of subjective classification adjustments (Report 1).

2.181 TP: 268

Classification: PUNCTURE ACCEPTED
Distance (mm) between TP center and puncture center: SUBJECTIVE CLASSIFICATION
Area (mm²) of puncture(s): SUBJECTIVE CLASSIFICATION



(a) Test point image (b) Inspection image

Figure 5.6: Example of subjective classification adjustments (Report 2).

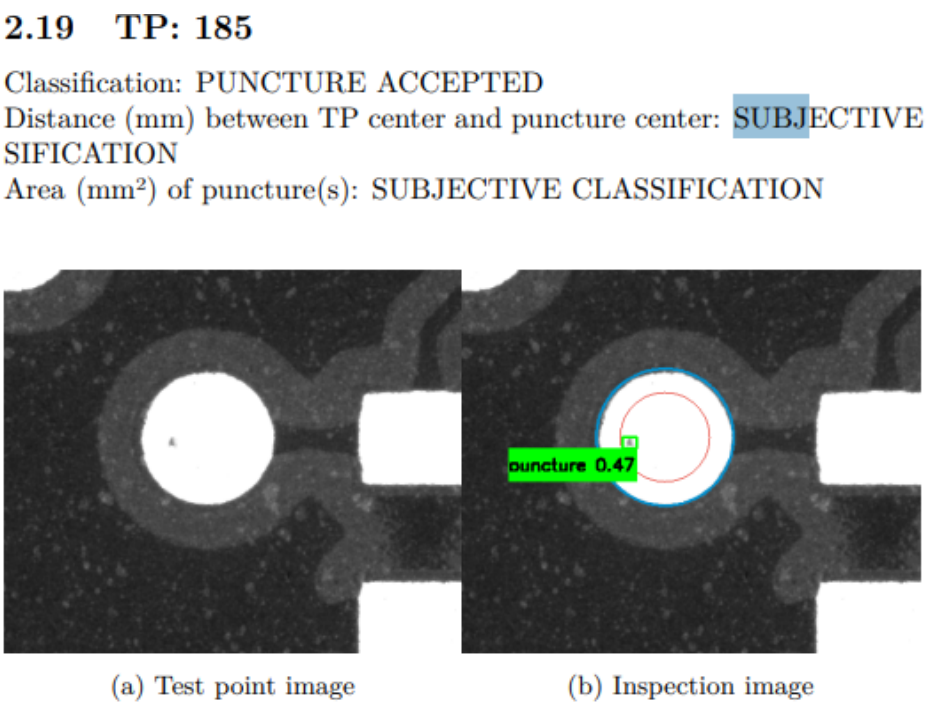


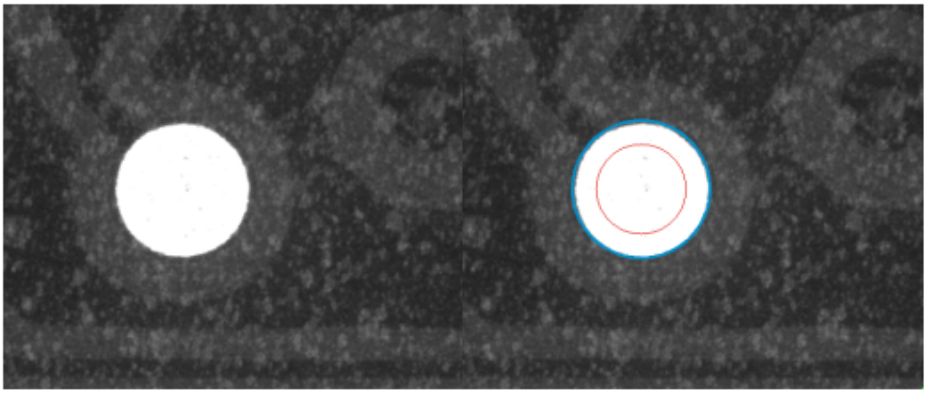
Figure 5.7: Example of subjective classification adjustments (Report 2).

5.3 Missed Detection Examples

In rare cases, the inspection reports indicated missed detections, labelled as punctures not found by the model. However, upon close examination, these cases typically involved scenarios where no clear puncture was visually identifiable within the acceptance circumference defined in the inspection images. As such, these instances represent borderline or ambiguous cases rather than explicit model failures, highlighting potential discrepancies between manual annotations and the model's strict detection criteria. Examples illustrating these ambiguous cases are provided in Figures 5.8 and 5.9.

2.107 TP: 357

Classification: PUNCTURE NOT FOUND
Distance (mm) between TP center and puncture center: NOT PUNCTURED
Area (mm²) of puncture(s): 0.0

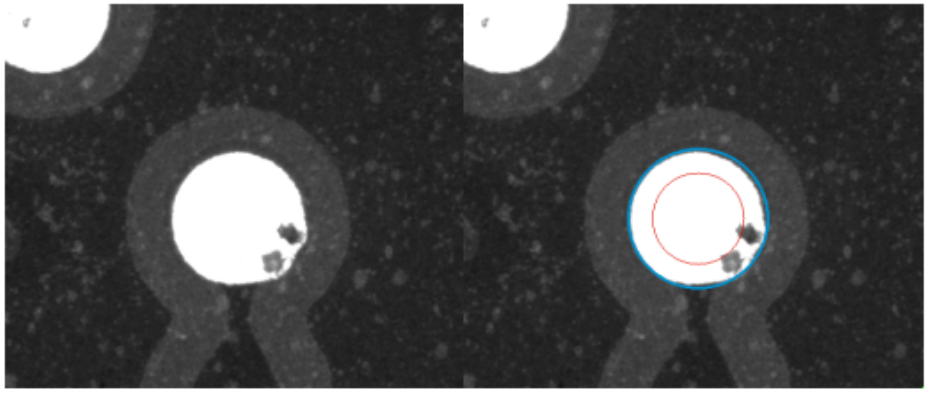


(a) Test point image (b) Inspection image

Figure 5.8: Ambiguous case labelled as a missed puncture detection (Report 1), with no clear puncture visible within the acceptance circumference.

2.318 TP: 327

Classification: PUNCTURE NOT FOUND
Distance (mm) between TP center and puncture center: NOT PUNCTURED
Area (mm²) of puncture(s): 0.0



(a) Test point image (b) Inspection image

Figure 5.9: Ambiguous case labelled as a missed puncture detection (Report 2), with no clear puncture visible within the acceptance circumference.

These examples underline the importance of consistent and precise annotation criteria and suggest that some reported missed detections may reflect annotation ambiguities rather than genuine

shortcomings in the model's detection capabilities.

5.4 Continuous Improvement and User-Friendly Application

To address performance gaps in challenging cases, continuous improvement was proposed by periodically augmenting the training dataset with additional images of problematic PCBs. Responding to this suggestion, the company commissioned a user-friendly graphical application (Figure 5.10), enabling technicians to easily add new images and the corresponding labels to the dataset and subsequently retrain and export improved models. Although still in the beta testing stage, this application has demonstrated practical functionality and offers a clear path towards ongoing model enhancement.

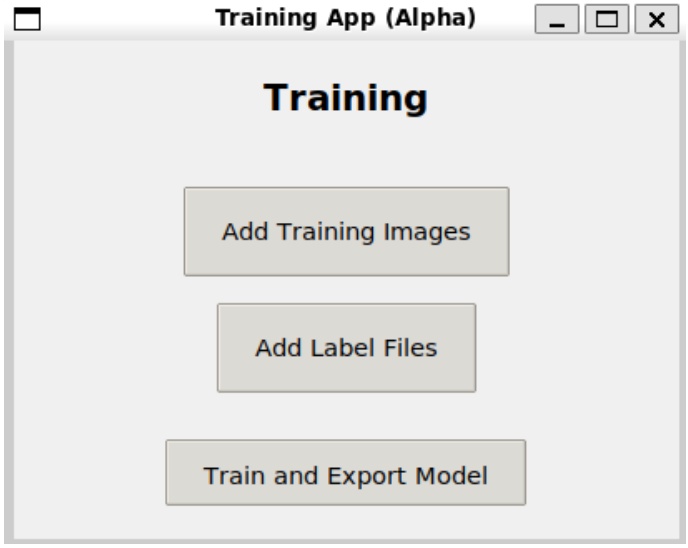


Figure 5.10: Graphical application developed to streamline the addition of new images and labels for continuous model improvement.

5.5 Summary of Results

Overall, the results achieved through validation and operational use demonstrate the high reliability and practical readiness of the model. Inspection reports from two distinct PCB inspections, thoroughly analysed during this research, confirm consistently strong detection performance, providing confidence in the model's robustness and adaptability to various PCB types.

Even considering a conservative evaluation, where cases requiring subjective classification are counted as missed detections, the model still demonstrates strong performance. Specifically, the first report documents 383 detected punctures, of which only 6 required subjective classification and 1 was completely undetected. The second report similarly reflects high accuracy, with 479 detected punctures, 6 punctures classified subjectively, and only 1 missed detection. These results underscore the substantial capability of the model to reliably identify punctures autonomously, with minimal reliance on manual adjustments.

Continuous refinement, supported by user-friendly tools such as the developed graphical application, will ensure sustained high-level performance, effective handling of challenging scenarios, and continuous improvement of detection quality over time.

Chapter 6

Conclusion

This research explored the application of Artificial Intelligence for defect detection in the automotive industry, particularly addressing the challenge of detecting minute puncture defects in PCBs. By fine-tuning a state-of-the-art YOLOv8 model using a specialised dataset, this dissertation successfully developed an AI-driven solution that significantly improves the precision, scalability, and efficiency of defect detection processes.

Traditional visual inspection methods, which rely heavily on human operators, have inherent limitations such as variability, slow processing times, and high labour costs. These issues are exacerbated when small, subtle defects, such as punctures in PCBs, are detected, a challenge that this research aimed to solve using AI-based techniques. Through this work, the capabilities of deep learning models, particularly those based on the YOLO architecture, have been effectively demonstrated for real-time industrial applications, replacing the need for manual inspection in many cases.

A key finding of this research was the importance of adapting AI models to specific industrial requirements. The fine-tuning process, based on real-world feedback and iterative improvements, allowed the model to meet the unique challenges posed by small-object detection in automotive component manufacturing. By leveraging the flexibility of AI, the system is not only capable of identifying defects but also contributes to overall quality control by improving the speed, accuracy, and consistency of inspections.

Furthermore, the development of a user-friendly graphical application for retraining the model with new images and labels provides a robust framework for continuous model improvement. This tool, still in its beta version, empowers technicians to enhance the model's detection capabilities in real-time, ensuring that the system can adapt to new types of defects and variations in PCB appearance.

The successful integration of the model into the company's existing environment and its validation using inspection reports have shown that the system is both viable and effective in operational settings. Continuous refinement, guided by user feedback and further training, will ensure that the model remains adaptable to future challenges, solidifying AI-driven defect detection as a key tool in the modern automotive manufacturing process.

Looking ahead, this research sets the foundation for further advancements in AI-driven defect detection systems in the automotive industry. The integration of AI with real-time manufacturing workflows offers significant potential for improving quality control processes, reducing operational costs, and increasing production efficiency. Further exploration into augmenting datasets, refining training techniques, and expanding model generalisation will contribute to the continued evolution of automated visual inspection systems, supporting the industry's move toward smarter, more efficient manufacturing solutions.

References

[1] Abhishek Bajpai et al. “Enhancing Underwater Object Detection: Leveraging YOLOv8m for Improved Subaquatic Monitoring”. In: *SN Computer Science* 5.6 (Aug. 14, 2024), p. 793. ISSN: 2661-8907. DOI: 10.1007/s42979-024-03170-z. URL: https://doi.org/10.1007/s42979-024-03170-z (visited on 06/22/2025).

[2] Brais Bosquet et al. “A full data augmentation pipeline for small object detection based on generative adversarial networks”. In: *Pattern Recognition* 133 (Jan. 1, 2023), p. 108998. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2022.108998. URL: https://www.sciencedirect.com/science/article/pii/S0031320322004782 (visited on 06/09/2025).

[3] Jincheng Chen et al. “Research on YOLOv7-based defect detection method for automotive running lights”. In: *Systems Science & Control Engineering* 11.1 (Dec. 31, 2023). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/21642583.2023.2185916, p. 2185916. ISSN: null. DOI: 10.1080/21642583.2023.2185916. URL: https://doi.org/10.1080/21642583.2023.2185916 (visited on 01/09/2025).

[4] Angelo Corallo, Vito Del Vecchio, and Alberto Di Prizio. “Advanced AI-Based Solutions for Visual Inspection: A Systematic Literature Review”. In: *Proceedings of the 26th International Conference on Enterprise Information Systems - Volume 1: ICEIS*. Backup Publisher: INSTICC ISSN: 2184-4992. SciTePress, 2024, pp. 656–664. ISBN: 978-989-758-692-7. DOI: 10.5220/0012618000003690.

[5] *Enhanced Identification of Internal Casting Defects in Vehicle Wheels Using YOLO Object Detection and X-Ray Inspection - ProQuest*. URL: https://www.proquest.com/openview/6ee51c2bbfe0b573f2e5051af3db1115/1?pq-origsite=gscholar&cbl=2069443 (visited on 01/09/2025).

[6] Carmen Gheorghe et al. *Analyzing Real-Time Object Detection with YOLO Algorithm in Automotive Applications: A Review. | EBSCOhost*. ISSN: 1526-1492 Issue: 3 Pages: 1939 Volume: 141. Dec. 1, 2024. DOI: 10.32604/cmes.2024.054735. URL: https://openurl.ebsco.com/contentitem/doi:10.32604%2Fcmes.2024.054735?sid=ebsco:plink:crawler&id=ebsco:doi:10.32604%2Fcmes.2024.054735 (visited on 01/05/2025).

[7] Perry Gibson and José Cano. *Productive Reproducible Workflows for DNNs: A Case Study for Industrial Defect Detection*. June 19, 2022. DOI: 10.48550/arXiv.2206.09359. arXiv: 2206.09359[cs]. URL: http://arxiv.org/abs/2206.09359 (visited on 02/18/2025).

[8] Shuangdong Hua et al. “Defect detection method using laser vision with model-based segmentation for laser brazing welds on car body surface”. In: *Measurement* 178 (June 2021), p. 109370. ISSN: 02632241. DOI: 10.1016/j.measurement.2021.109370. URL:

https://linkinghub.elsevier.com/retrieve/pii/S0263224121003651 (visited on 11/26/2024).

[9] Hao Huang and Kai Zhu. "Automotive Parts Defect Detection Based on YOLOv7". In: *Electronics* 13.10 (Jan. 2024). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 1817. ISSN: 2079-9292. DOI: 10.3390/electronics13101817. URL: https://www.mdpi.com/2079-9292/13/10/1817 (visited on 01/09/2025).

[10] Purvish Jajal et al. *Analysis of Failures and Risks in Deep Learning Model Converters: A Case Study in the ONNX Ecosystem*. Sept. 2, 2024. DOI: 10.48550/arXiv.2303.17708. arXiv: 2303.17708[cs]. URL: http://arxiv.org/abs/2303.17708 (visited on 05/14/2025).

[11] Rui-Yang Ju and Weiming Cai. "Fracture detection in pediatric wrist trauma X-ray images using YOLOv8 algorithm". In: *Scientific Reports* 13.1 (Nov. 16, 2023). Publisher: Nature Publishing Group, p. 20077. ISSN: 2045-2322. DOI: 10.1038/s41598-023-47460-7. URL: https://www.nature.com/articles/s41598-023-47460-7 (visited on 06/22/2025).

[12] Mate Kisantal et al. *Augmentation for small object detection*. Feb. 19, 2019. DOI: 10.48550/arXiv.1902.07296. arXiv: 1902.07296[cs]. URL: http://arxiv.org/abs/1902.07296 (visited on 03/25/2025).

[13] Adrian Korodi et al. "Image-Processing-Based Low-Cost Fault Detection Solution for End-of-Line ECUs in Automotive Manufacturing". In: *Sensors* 20.12 (Jan. 2020). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 3520. ISSN: 1424-8220. DOI: 10.3390/s20123520. URL: https://www.mdpi.com/1424-8220/20/12/3520 (visited on 01/09/2025).

[14] Ashvi Kumaradurai et al. "Fault Detection In Photovoltaic Systems Using Machine Learning Algorithms: A Review". In: *2020 8th International Conference on Orange Technology (ICOT)*. 2020 8th International Conference on Orange Technology (ICOT). Daegu, Korea (South): IEEE, Dec. 18, 2020, pp. 1–5. ISBN: 9781665418522. DOI: 10.1109/ICOT51877.2020.9468768. URL: https://ieeexplore.ieee.org/document/9468768/ (visited on 01/05/2025).

[15] Wang Liqun, Wu Jiansheng, and Wu Dingjin. "Research on Vehicle Parts Defect Detection Based on Deep Learning". In: *Journal of Physics: Conference Series* 1437.1 (Jan. 2020). Publisher: IOP Publishing, p. 012004. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1437/1/012004. URL: https://dx.doi.org/10.1088/1742-6596/1437/1/012004 (visited on 01/09/2025).

[16] Behzad Mirzaei et al. "Small Object Detection and Tracking: A Comprehensive Review". In: *Sensors* 23.15 (Jan. 2023). Number: 15 Publisher: Multidisciplinary Digital Publishing Institute, p. 6887. ISSN: 1424-8220. DOI: 10.3390/s23156887. URL: https://www.mdpi.com/1424-8220/23/15/6887.

[17] Zhimin Mo, Liding Chen, and Wenjing You. "Identification and Detection of Automotive Door Panel Solder Joints based on YOLO". In: *2019 Chinese Control And Decision Conference (CCDC)*. 2019 Chinese Control And Decision Conference (CCDC). ISSN: 1948-9447. June 2019, pp. 5956–5960. DOI: 10.1109/CCDC.2019.8833257. URL: https://ieeexplore.ieee.org/abstract/document/8833257?casa_token=rSyoQGXiv3kAAAAA:6deOXSMEd7qVjEbDY4_LXswQpoBxGVpkCmQ0ysAstGJFmnSYaTusK1lBdCle60tegHYcloHrAa4 (visited on 01/09/2025).

[18] arvindpdmn Pawan_Dubey. *Confusion Matrix*. Devopedia. Aug. 20, 2019. URL: https://devopedia.org/confusion-matrix.

[19] Chen Peng, Xiaoyu Li, and Yulong Wang. "TD-YOLOA: An Efficient YOLO Network With Attention Mechanism for Tire Defect Detection". In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023). Conference Name: IEEE Transactions on Instrumentation and Measurement, pp. 1–11. ISSN: 1557-9662. DOI: 10.1109/TIM.2023.3312753. URL: https://ieeexplore.ieee.org/abstract/document/10243113?casa_token=HroA-3PsMEYAAAAA:5oRL3LHZa1Tv2Q9rHAqZ7Ezypwpl83y3wfEz-bEcD8Ep8vi6UsOB4lh4bhxK5tQJJR0fNHQom0c (visited on 01/09/2025).

[20] João Queirós. "Automatic Final Control Inspection System for Automotive Displays". In: *Proceedings of the FISITA - Technology and Mobility Conference Europe 2023*. FISITA - Technology and Mobility Conference Europe 2023. Barcelona, Spain: FISITA, 2023. DOI: 10.46720/fwc2023-mtm-009. URL: https://www.fisita.com/library/fwc2023-mtm-009 (visited on 11/13/2024).

[21] G Rahull et al. "Automated Defect Detection System for Automobile Accessory Manufacturing Using YOLOv5". In: *2024 IEEE Students Conference on Engineering and Systems (SCES)*. 2024 IEEE Students Conference on Engineering and Systems (SCES). June 2024, pp. 1–6. DOI: 10.1109/SCES61914.2024.10652413. URL: https://ieeexplore.ieee.org/abstract/document/10652413?casa_token=STKDh0xaS8sAAAAA:WzYdsOY2W9QnE4fnKhs7Fkii3CC-gSpfSB43q-3YFmTEuC21PDvVY7r4SpOOWb05N2115bFqaVs (visited on 01/09/2025).

[22] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[23] S A Sanchez, H J Romero, and A D Morales. "A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework". In: *IOP Conference Series: Materials Science and Engineering* 844.1 (May 2020). Publisher: IOP Publishing, p. 012024. ISSN: 1757-899X. DOI: 10.1088/1757-899X/844/1/012024. URL: https://dx.doi.org/10.1088/1757-899X/844/1/012024 (visited on 02/21/2025).

[24] Grigory Sapunov. *Deep Learning with JAX*. Google-Books-ID: wSUmEQAAQBAJ. Simon and Schuster, Oct. 29, 2024. 406 pp. ISBN: 978-1-63343-888-0.

[25] Imran Shafi et al. "Deep Learning-Based Real Time Defect Detection for Optimization of Aircraft Manufacturing and Control Performance". In: *Drones* 7.1 (Jan. 1, 2023), p. 31. ISSN: 2504-446X. DOI: 10.3390/drones7010031. URL: https://www.mdpi.com/2504-446X/7/1/31 (visited on 01/05/2025).

[26] Shuohui Chen et al. "Automatic Recognition of Welding Seam Defects in TOFD Images Based on TensorFlow". In: *Automatic Control and Computer Sciences* 56.1 (Feb. 1, 2022), pp. 58–66. ISSN: 1558-108X. DOI: 10.3103/S0146411622010035. URL: https://doi.org/10.3103/S0146411622010035 (visited on 02/17/2025).

[27] Shuohui Chen et al. "Automatic Recognition of Welding Seam Defects in TOFD Images Based on TensorFlow". In: *Automatic Control and Computer Sciences* 56.1 (Feb. 1, 2022), pp. 58–66. ISSN: 1558-108X. DOI: 10.3103/S0146411622010035. URL: https://doi.org/10.3103/S0146411622010035 (visited on 02/17/2025).

[28] Kang Tong, Yiquan Wu, and Fei Zhou. "Recent advances in small object detection based on deep learning: A review". In: *Image and Vision Computing* 97 (May 1, 2020), p. 103910. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2020.103910. URL: https://www.sciencedirect.com/science/article/pii/S0262885620300421.

[29] Pedro Vieira. "Automatic Inspection Vision System for Defect Detection on Piston Rings". In: *Proceedings of the FISITA - Technology and Mobility Conference Europe 2023*. FISITA - Technology and Mobility Conference Europe 2023. Barcelona, Spain: FISITA, 2023. DOI: 10.46720/fwc2023-mtm-004. URL: https://www.fisita.com/library/fwc2023-mtm-004 (visited on 11/13/2024).

[30] Fang Xie et al. "An Adaptive Defect Detection Technology for Car-Bodies Surfaces". In: *2019 Chinese Automation Congress (CAC)*. 2019 Chinese Automation Congress (CAC). ISSN: 2688-0938. Nov. 2019, pp. 1023–1028. DOI: 10.1109/CAC48633.2019.8996176. URL: https://ieeexplore.ieee.org/document/8996176 (visited on 12/03/2024).

[31] Jiancheng Xu and Jialei Ma. "Auto Parts Defect Detection Based on Few-shot Learning". In: *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*. 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA). Changchun, China: IEEE, May 20, 2022, pp. 943–946. ISBN: 978-1-6654-5911-2. DOI: 10.1109/CVIDLICCEA56201.2022.9823993. URL: https://ieeexplore.ieee.org/document/9823993/ (visited on 01/09/2025).

[32] Hao Yi et al. "Small Object Detection Algorithm Based on Improved YOLOv8 for Remote Sensing". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 17 (2024), pp. 1734–1747. ISSN: 2151-1535. DOI: 10.1109/JSTARS.2023.3339235. URL: https://ieeexplore.ieee.org/abstract/document/10342786.

[33] *YOLOv8 Architecture; Deep Dive into its Architecture -Yolov8*. Running Time: 667 Section: Blog. Jan. 15, 2024. URL: https://yolov8.org/yolov8-architecture/ (visited on 05/25/2025).

[34] Muhammed Hakan Yorulmuş, Hür Bersam Bolat, and Çağatay Bahadır. "Predictive Quality Defect Detection Using Machine Learning Algorithms: A Case Study from Automobile Industry". In: *Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation*. Ed. by Cengiz Kahraman et al. Cham: Springer International Publishing, 2022, pp. 263–270. ISBN: 978-3-030-85577-2. DOI: 10.1007/978-3-030-85577-2_31.

[35] Qianguang Zhang et al. "GSBF-YOLO: A steel strip surface defect detection technique based on improved YOLOv8". In: *2024 6th International Conference on Internet of Things, Automation and Artificial Intelligence (IoTAAI)*. 2024 6th International Conference on Internet of Things, Automation and Artificial Intelligence (IoTAAI). July 2024, pp. 19–23. DOI: 10.1109/IoTAAI62601.2024.10692413. URL: https://ieeexplore.ieee.org/document/10692413/?arnumber=10692413 (visited on 01/05/2025).