Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080112 Published Online: 29 April 2026

ARTICLE

# Deep Learning Driven Real-Time PCB Inspection Using an Optimized YOLO v9 Architecture

**Jigar Sarda[1], Rohan Vaghela[1], Akash Kumar Bhoi[2], Chang-Won Yoon[3,*] and Mangal Sain[4,*]**

[1]Department of Computer Science & Engineering, Chandubhai S. Patel Institute of Technology, Charotar University of Science & Technology, Anand, Gujarat, India

[2]Department of Electronics and Communication Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India

[3]Department of Computer Engineering, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan, Republic of Korea

[4]Division of Computer & Information Engineering, Regional Innovation Center, Dongseo University, Busan, Republic of Korea

*Corresponding Authors: Chang-Won Yoon. Email: yooncw@dongseo.ac.kr; Mangal Sain. Email: mangalsain1@gmail.com

Received: 08 February 2026; Accepted: 27 March 2026

**ABSTRACT:** Printed circuit boards (PCBs) are essential components that strongly influence the performance and reliability of modern electronic systems. However, minor and visually subtle manufacturing defects can degrade product quality and pose serious challenges for automated inspection systems. Existing deep learning–based methods often struggle to simultaneously achieve high detection accuracy, real-time processing speed, and compact model size. This study proposes an enhanced approach for real-time PCB defect detection using advanced object detection models. A dedicated dataset of bare PCBs was developed and carefully annotated with six defect categories: open circuits, missing holes, spurs, mouse bites, short circuits, and spurious copper. Multiple YOLO-based models were trained and evaluated on this dataset, among which the YOLOv9-small model demonstrated superior performance, achieving a mAP@50 of 99.2% and a mAP@50–95 of 63.1%. To further enhance computational efficiency, optimization techniques such as pruning and quantization were applied to the YOLOv9-small model. These optimizations reduced model size by 36.4% and increased processing speed by 24.7%, while maintaining high detection accuracy with a mAP@50 of 98.6%. Overall, the experimental results demonstrate that the optimized YOLOv9-based model provides a highly accurate, efficient, and practical solution for automated PCB defect detection in real-world electronics manufacturing environments.

**KEYWORDS:** Printed circuit boards (PCB); defects detection; you look only once (YOLO); deep learning; computer vision

## 1 Introduction

Printed circuit boards (PCBs) form the foundation of nearly all modern electronic devices. Since the introduction of the first PCB-related patent by Albert Hanson in 1903 [1], technological progress has been closely intertwined with advancements in PCB design and manufacturing. Early PCB prototypes featured relatively simple circuit layouts; however, as electronic components such as transistors, diodes, and integrated circuits evolved, they were integrated into more sophisticated designs, leading to the development of printed circuit board assemblies (PCBAs). PCBAs, commonly found in electronic devices, are produced by soldering various components onto laminated copper traces that serve as conductive pathways for electrical circuits. Despite their seemingly simple appearance, PCB designs must comply with rigorous manufacturing standards to ensure both reliability and performance. For instance, Adam's IPC-2221 design standard [2] specifies that the optimal width of a copper trace depends on the allowable temperature rise resulting

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

from current flow. Additionally, the number of PCB layers required is determined by the complexity of the circuit. PCBs are generally categorized as single-sided, double-sided, or multi-layer [3]. Multi-layer PCBs, fabricated by stacking multiple single-sided layers, enable compact, functional, and high-density circuit designs, satisfying the growing demands of modern electronic systems.

With the rapid global expansion of the electronics industry, ensuring the quality and reliability of PCBs has become increasingly critical for maintaining overall product performance. To achieve stringent quality standards, extensive research efforts have focused on developing automated defect detection systems leveraging computer vision and Automated Optical Inspection (AOI) techniques. AOI is a non-contact inspection method that plays a vital role in identifying soldering and assembly defects. It has become an indispensable element of large-scale PCB production lines due to its ability to improve inspection accuracy while minimizing operational costs. However, traditional machine vision–based algorithms remain highly sensitive to variations in manufacturing conditions and product characteristics, limiting their effectiveness in high-throughput industrial environments [4].

In actual PCB production, the inspection process is further complicated by factors such as minor component misalignments, differences in solder mask colors, and changes in lighting conditions. These variations introduce significant challenges not only for conventional AOI systems but also for deep learning–based inspection methods. For example, a model trained on a specific input type—such as grayscale images from the DeepPCB dataset—may fail to generalize effectively to different image modalities, including full-color images. This limitation underscores the difficulty of achieving robust model generalization across diverse real-world scenarios. Moreover, attaining real-time detection performance, typically defined as 30 frames per second or higher, while maintaining high defect-level accuracy remains a formidable challenge. The situation is further complicated by the scarcity of rare defect types, which limits available training data and demands costly manual annotation. These factors collectively motivate the exploration of advanced learning paradigms such as few-shot and semi-supervised learning, which, although promising, remain in the early phases of research and practical adoption [5].

### *1.1 Related Works*

In recent years, deep learning has been increasingly integrated with computer vision, leading to remarkable advancements. Numerous studies have explored the application of deep learning in manufacturing inspection through various methodologies [6]. For instance, TDD-Net employs Faster R-CNN as its backbone, while YOLOv3 utilizes SSD as its backbone. Modifications to the attention mechanism have been shown to improve accuracy, particularly in detecting small defects on PCBs. However, several challenges remain. The TDD-Net, which uses Faster R-CNN as its backbone, suffers from low detection speeds due to the two-stage processing structure of Faster R-CNN, resulting in long feature extraction times [7]. In contrast, although YOLOv3 achieves higher detection accuracy, it is less optimized compared to other models such as TDD-Net, which may exhibit faster processing speeds but lower overall accuracy [8]. Additive manufacturing, commonly known as 3D printing, is employed to produce complex products with diverse customizations. However, the 3D printing process often suffers from defects such as warping, cracking, and stringing, which reduce product quality. Conventional inspection techniques for 3D printing are primarily manual and lack the capability for real-time defect detection. To address this limitation, deep learning–based approaches have been introduced [9]. In this context, the development of real-time defect detection systems utilizing layer-by-layer inspection with convolutional neural networks (CNNs) and low-cost embedded vision technology has been proposed, achieving high accuracy with low computational cost for efficient defect identification during the printing process. Furthermore, in [10], the performance of various deep learning models was evaluated for the detection of warping, stringing, and cracking defects in 3D printing

using a Raspberry Pi–based image acquisition system. Modifying the attention mechanism has also been shown to enhance accuracy, particularly when identifying small defects on PCBs.

Furthermore, in [11], the comprehensive survey of various machine learning techniques was presented to classify the defects in the wafer using the defect classification system, which proves the high accuracy of the ResNet and CNN models using deep learning techniques and the balance between accuracy and efficiency using the XGBoost algorithm; moreover, the taxonomy was also proposed to improve the accuracy of the defect classification system with future research directions to enhance the generalization of the defect classification system in the semiconductor manufacturing domain. The Single Shot Multibox Detector achieves moderate accuracy with a high inference frame rate in real-time defect detection systems. Therefore, the trade-off between accuracy and speed remains in defect detection systems using deep learning techniques in the PCB domain.

In [12], the author introduces an advanced deep learning (DL) network for detecting defects in printed circuit boards (PCBs), employing a multi-scale feature pyramid network model to tackle the challenges associated with the real-time identification of minor defects. The study in [13] presents a thorough deep learning framework for identifying and categorizing defects in PCBs using an object detection model. The authors utilize the HRIPCB dataset to compare their model's performance with that of leading models, including RetinaNet, Faster R-CNN (FRCNN) with ResNet50, and various YOLO versions. In [14], the YOLOv7 model is applied to detect faults and assess the quality of PCBs. Additionally, ref [15] proposes a novel algorithm called CDI-YOLO, designed to improve the identification of PCB defects. This algorithm aims to strike a balance between fast processing speeds, high detection accuracy, and a compact model size, addressing common challenges faced by current deep learning methods, especially in real-time applications within PCB manufacturing, such as the need for quick decision-making and resource efficiency in production lines. The researcher in [16] proposes enhancements to the YOLOv7 model to increase its accuracy and speed in defect detection. This fixes issues with keeping an eye on PCB production lines in real time by enabling faster and more accurate identification of defects, which is crucial for maintaining quality control during manufacturing. YOLO-PDD is a new way to find problems with PCBs, which is suggested in [5]. It uses YOLOv5 and a multi-scale module that comes from the Res2Net architecture. This method resolves problems with finding defects in PCBs, like processing in real time, noise, low accuracy, and being able to work in different manufacturing settings. The author of [17] also talks about a way to detect mistakes in PCBs while they are being made. This system uses both traditional visual computing and DL to make production better by improving defect detection accuracy and reducing processing time during PCB manufacturing. Table 1 also displays some of the work already completed in this field.

**Table 1:** Overview of DL based PCB defect detection methods.

| Reference | Year | Key Findings | Limitations |
|---|---|---|---|
| [18] | 2026 | • Identified "open circuit" and "spur" as hardest defects.<br>• Provides algorithm selection roadmap for industry. | • Still dependent on dataset standardization and quality.<br>• Does not deeply address real-world deployment constraints. |
| [19] | 2026 | • Proposed lightweight YOLO-CD model for edge deployment.<br>• Improved small-defect detection using attention mechanisms. | • Accuracy still lower than heavier models.<br>• Complex architecture modifications increase design complexity. |

(Continued)

**Table 1 (continued)**

| Reference | Year | Key Findings | Limitations |
|---|---|---|---|
| [20] | 2026 | • Lightweight and efficient model<br>• Strong small defect detection and robust under noisy conditions<br>• Good cross-domain generalization. | • Slightly lower FPS and not best on all datasets<br>• Compatibility varies across models.<br>• Requires tuning for real-world use. |
| [21] | 2025 | • YOLOv9 achieved very high accuracy (mAP 98.4%) for PCB defect detection.<br>• Demonstrated improvement over YOLOv5 and YOLOv8. | • Dataset is relatively small (693 images) → may limit generalization.<br>• Uses synthetic dataset, not fully real-world. |
| [22] | 2025 | • YOLOv8x achieved high performance.<br>• Deep learning outperforms traditional inspection methods.<br>• Custom dataset improves model effectiveness. | • Requires large, high-quality datasets.<br>• High computational cost for training and deployment.<br>• Challenges in achieving real-time processing consistently. |
| [23] | 2024 | • WRGMSFA + BiFPN + WIoU v3 significantly boost small-defect localization and overall detection precision.<br>• Low model complexity and fast inference demonstrated on PKU-Market-PCB dataset. | • Spur-type defects still exhibit lower recall due to shape ambiguity.<br>• Validated only on PKU-Market-PCB. |
| [24] | 2024 | • Introduces multi-scale feature paths and separate localization/classification losses,<br>• Enhances fine-grained defect detection in cluttered PCB assembly images. | • Quantitative mAP/FPS not reported in this file.<br>• Evaluation confined to a single in-house dataset. |
| [25] | 2024 | • Employs depth-wise separable convolutions in a Mobile-style backbone to reduce parameters.<br>• Maintains competitive detection accuracy on PCB surface scans. | • No explicit performance metrics (mAP/FPS) presented.<br>• Tested only on limited PCB scan set. |
| [26] | 2024 | • Enhances C3 modules and modifies FPN to better recall tiny defects.<br>• Shows improved mAP for small-object categories compared to baseline YOLO models. | • Dataset size and split details not specified in file.<br>• Lacks end-to-end speed (FPS) comparisons. |

(Continued)

**Table 1 (continued)**

| Reference | Year | Key Findings | Limitations |
|---|---|---|---|
| [27] | 2023 | • Tunes anchors via k-means++ and adds a small-target head fused with Swin blocks.<br>• Reaches mAP@0.5 = 95.97% at 92.5 FPS on DeepPCB. | • Improvements demonstrated only on DeepPCB (B/W images).<br>• Limited classes (6 types) may not generalize to new boards. |
| [28] | 2023 | • Incorporates MBConv blocks, CBAM attention, and BiFPN for bidirectional feature fusion.<br>• Achieves mAP@0.5 = 95.3% with modest parameter increase. | • Parameter and computation overhead increased vs. baseline.<br>• Evaluation restricted to DeepPCB without broader testing. |
| [29] | 2022 | • Leverages local-patch attention with global context tokens to capture both fine and coarse features.<br>• Demonstrates strong classification accuracy, hinting at promise for defect localization. | • Primarily evaluated for classification, with limited detection results.<br>• Transformer overhead may hinder real-time use. |
| [30] | 2022 | • Comprehensive survey of traditional and CNN-based inspection methods.<br>• Identifies key challenges e.g., lighting variation, small object detection that motivate deep-learning approaches. | • Rapidly outdated by latest YOLO-series advances. |
| [31] | 2020 | • Integrates GARPN for anchor refinement and deep FPN fusion to boost two-stage detector precision.<br>• Achieves higher recall of subtle defect instances compared to vanilla Faster-R-CNN. | • Two-stage pipeline is inherently slower; FPS not reported.<br>• Tested on a single dataset without cross-validation. |

### *1.2 Research Gap and Motivation*

It is observed that the existing state-of-the-art methodologies in the field of PCB defect detection have achieved considerable progress in localizing small defects and improving detection accuracy; however, they still exhibit several critical limitations. Among these, the detection of spur-type defects remains challenging due to their ambiguous shapes and the presence of small structural features. Furthermore, most existing methodologies have been evaluated only on single datasets, such as PKU-Market-PCB [20], DeepPCB [27], or proprietary in-house datasets. Testing these methods across multiple datasets is essential to validate the robustness and generalization capabilities of the developed detection models. Moreover, the existing studies often fail to report key quantitative parameters, such as frames per second (FPS), and lack comparative analyses of computational costs. This omission introduces ambiguity regarding their practical applicability in real-time industrial environments.

The need to bridge these research gaps motivates further exploration in the domain of PCB defect detection. There is a clear demand within both industrial and academic communities for effective detection

models capable of accurately identifying complex defect types, such as spur-type defects, and addressing the challenges posed by their shape ambiguity. Additionally, the lack of transparency regarding computational performance, particularly in terms of processing speed and parameter efficiency—drives the development of lightweight, high-efficiency algorithms suitable for real-time operations. Such advancements are especially critical in applications where timely defect detection is essential for maintaining production quality and reducing downtime. The development of these algorithms not only contributes to practical improvements in DL-based object detection but also deepens theoretical understanding within the field.

### *1.3 Contribution*

The main contributions of the study include:

- Identified and incorporated six key PCB defect classes 'mouse bite', 'spur', 'missing hole', 'short circuit', 'open circuit', and 'spurious copper', ensuring comprehensive defect coverage for robust fault detection.
- Validated the performance of single-stage object detectors, particularly YOLO v9, for real-time fault detection capabilities with significant advancements in accuracy and processing speed, tailored for industrial applications.
- Using pruning and quantization techniques reduce inference time with lower memory and energy consumption and enable deployment on edge devices.

Further, Section 2 provides a detailed description of the methodology, encompassing the information about dataset, system configuration, and model architecture. Section 3 outlines the evaluation metrics employed to assess the performance of the models. The results and discussion are presented in Section 4, presenting a detailed investigation of the findings. Finally, Section 5 concludes the study, summarizing important insights and contributions.

## 2 Materials and Methods

The overall detection pipeline begins with a curated PCB defect dataset, fully annotated in the standard YOLO format and uniformly resized to 640 × 640 pixels. The data was splitted into training (80%), validation (10%), and test (10%) sets. We then fine-tune four successive YOLO variants namely v8, v9, v10, and 11 using identical training schedules. Each model is evaluated on the same held-out test set, recording both mean Average Precision at IoU = 0.5 (mAP@0.5) and real-time inference speed (FPS). Finally, we compare these four versions side by side as shown in Fig. 1.

### *2.1 Dataset*

In today's era of advanced automation and diagnostics, PCB fault detection plays an essential part in ensuring the quality and reliability of electronic devices. For the experimental purpose the study, utilized a publicly available PCB defect detection dataset introduced on Kaggle and managed by Norbert Elter [32]. Fig. 2 shows visual illustration of different PCB defect types. The dataset contains 10,668 high-resolution images classified across six defect types such as open circuits, missing holes, spur, mouse bites, short circuits, and spurious copper, along with annotations for bounding boxes. A short description of each of these defects will be given below to provide a better understanding of the proposed detection model:

- Missing Hole: This type of defect will occur when one of the required holes to be drilled into the PCB is missing. These holes are usually used for component placement or electrical connections.
- Mouse Bite: The mouse bite type of defect will usually occur as small, irregular indentations on the edges of the conductive paths or pads. These types of defects usually occur due to over-etching of the PCB.

- Open Circuit: The open circuit type of defect will usually occur when there is a discontinuity in one of the paths of the conductive paths, thus stopping the flow of electrical current.
- Short: The short circuit type of defect will usually occur when two of the paths of the conductive paths are connected unintentionally.
- Spur: The spur type of defect will usually occur as unwanted extensions of the copper traces on the PCB. These types of defects usually occur due to under-etching.
- Spurious Copper: This type of defect will usually occur when unwanted copper patches are deposited on the surface of the PCB, where no copper should exist.

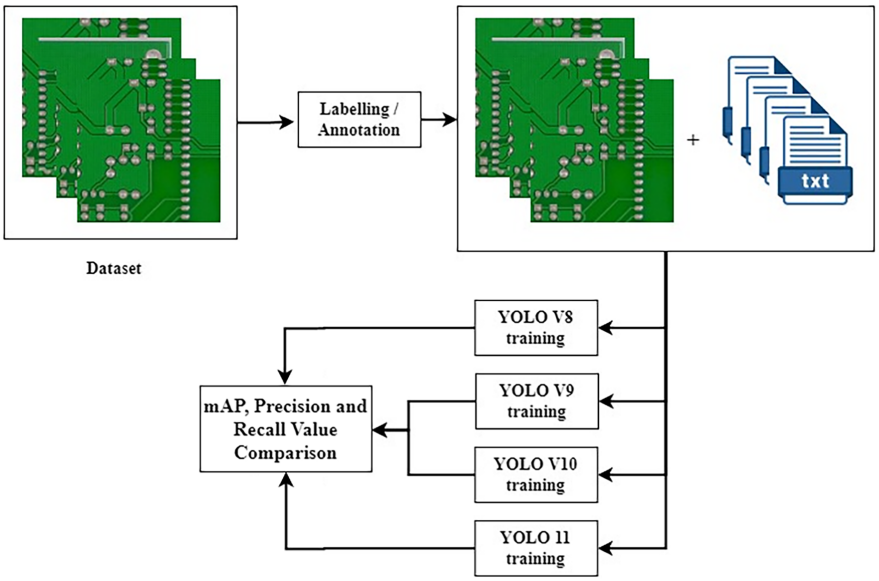



**Figure 1:** Block diagram of the experimental pipeline.

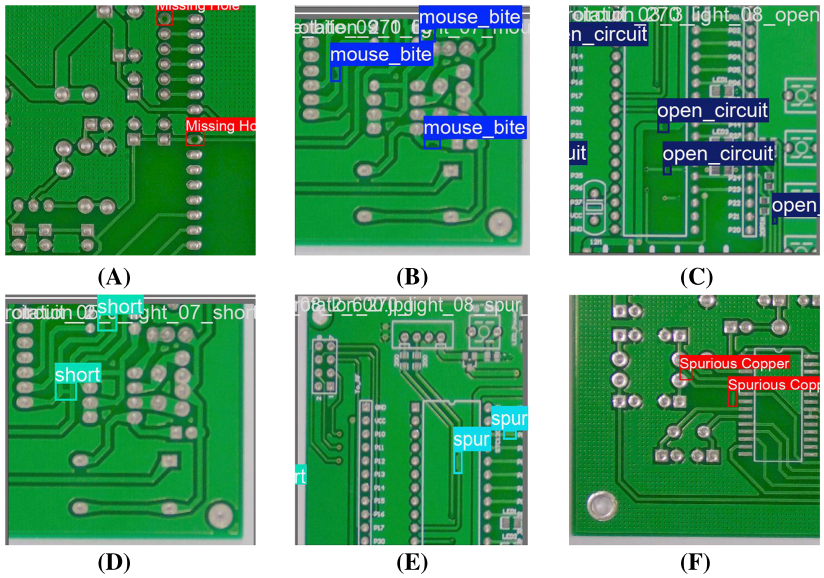



**Figure 2:** Visual illustration of different PCB defect types: (**A**) Missing hole; (**B**) Mouse bite; (**C**) Open circuit; (**D**) Short; (**E**) Spur; (**F**) Spurious copper.

Each image is labelled to highlight specific defects in PCBs, aiding in the development of fault detection algorithms. The dataset is structured to include labelled images, with the split ratio of 80% for the training and 10% for each validation and testing. All images were of resize to 640 × 640 pixels, enabling uniform pre-processing and effective training for models.

### *2.2 YOLO v9*

YOLO v9 is a state-of-the-art object detection framework that achieves an incomparable balance between accuracy and computational efficiency [33]. Its architecture incorporates an integration of the Backbone Network, Neck Architecture, and Detection Head. These modules are precisely designed to address challenges such as gradient instability, feature retention, and multi-scale object detection, making YOLO v9 a multipurpose solution for applications ranging from autonomous navigation to medical imaging. Table 2 presents a comparative analysis of various YOLO v9 model variants, highlighting key computational features such as number of parameters, gradient computations, and layer configurations. Further Fig. 3 shows the detailed block architecture of YOLO v9 detection model.

**Table 2:** Comparative analysis of various YOLO v9 model variants.

| Model | Layers | Parameters | Gradients | GFLOPs |
|---|---|---|---|---|
| **YOLO v9s** | 917 | 7,289,730 | 7,289,714 | 27.4 |
| **YOLO v9m** | 603 | 20,164,827 | 20,164,811 | 77.6 |
| **YOLO v9c** | 618 | 25,536,171 | 25,536,155 | 103.7 |

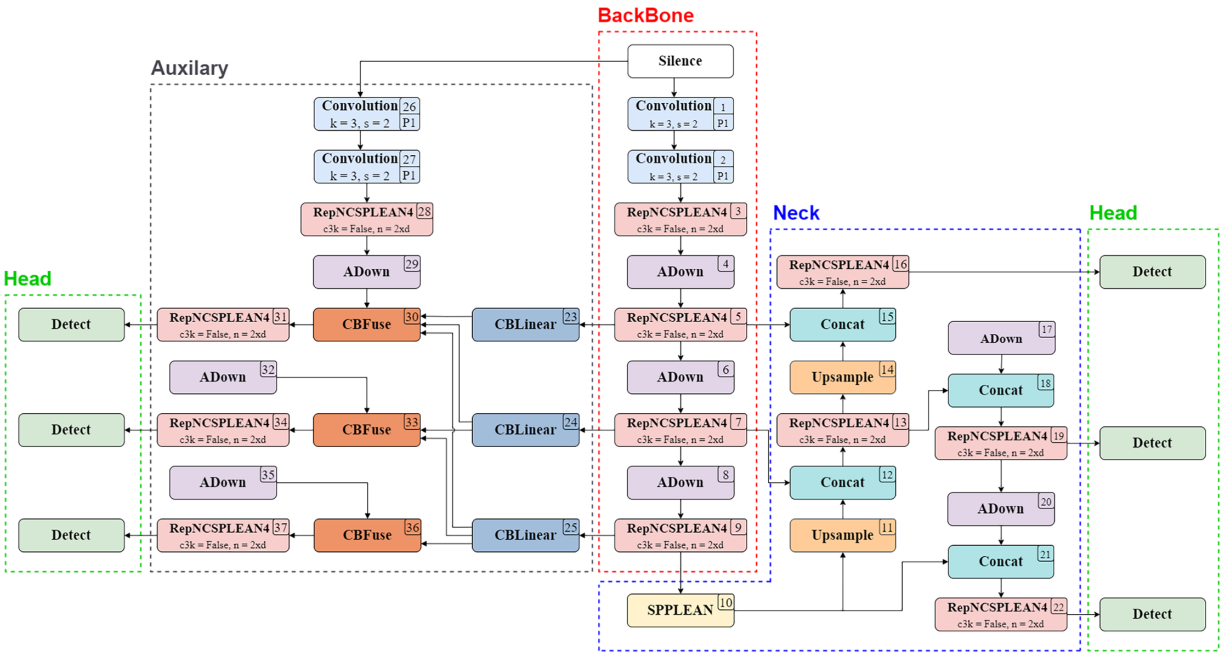



**Figure 3:** Detailed architecture of YOLO v9 detection model.

*2.2.1 Backbone Architecture*

It is the principal feature extraction module of YOLOv9. GELAN is used for superior gradient flow and hierarchical feature extraction. GELAN is an extension of ELAN. GELAN incorporates Cross Stage Partial Network (CSPNet). In CSPNet, feature maps are divided into two branches. One branch processes input using densely connected blocks, and another branch bypasses densely connected blocks. This design helps preserve spatial and contextual information and improve gradient flow during training. GELAN also incorporates gradient path planning for smooth gradient propagation through all layers. This prevents vanishing gradients during deep gradient flow. GELAN is used for hierarchical feature extraction using a hierarchical structure. In this hierarchical structure, features of each layer are added to the next layers. This hierarchical feature extraction is represented using mathematical equations as shown below in Eqs. (1) and (2).

$$F_{output} = \sum_{i=1}^{N} F_{layer_i} \tag{1}$$

$$F_{layer} = F_{conv}\left(F_{ResCSP}\left(y\right)\right) \tag{2}$$

where $F_{layer_i}$ are features extracted at different levels. By maintaining critical information across layers. $y$ is the output feature vector of previous layer.

Another such block in RepNCSPELAN (Reparametrized Nonlinear CSP-ELAN), as depicted in Fig. 4, is one of the major innovations in YOLO v9, designed to optimize efficiency and feature extraction capabilities. This block combines the ideas of CSPNet with ELAN, introducing nonlinearity and reparameterization to obtain efficiency. Reparameterization helps eliminate redundant computations during inference, thus improving efficiency with minimal impact on accuracy. This nonlinearity helps balance the utilization of parameters and feature richness, which is important for detecting small objects with complex features.

Further, to improve the feature representation at various scales, YOLO v9 utilizes a spatial pyramid pooling-enhanced layer aggregation network named SPPELAN, as presented in Fig. 5. The SPPELAN network utilizes both spatial pyramid pooling and layer aggregation capabilities to effectively combine features at various scales. The proposed YOLO v9 utilizes this advanced block named RepNCSPELAN and SPPELAN to ensure that the Backbone Network is capable of extracting robust hierarchical features, thereby detecting objects under difficult conditions.

*2.2.2 Neck Architecture*

Neck Architecture is helpful as an intermediary block between the backbone and the detection head, filtering and aggregating feature representations at different levels of abstraction. In YOLOv9, an Enhanced Path Aggregation Network, known as PANet++, is utilized, which helps in the fusion of low-level features with high-level semantic abstractions. The representation of the two is necessary to achieve precision in space and depth of the background. Hybrid attention is used in the Neck Architecture, which recalibrates the weights of the feature map according to their importance. The attention weights A for the feature map F are given by Eq. (3).

$$A = \sigma\left(CBAM\left(SE\left(F\right)\right)\right) \tag{3}$$

where the *SE* module applies channel-wise recalibration as shown in Eq. (4).

$$SE = \sigma\left(W_2.ReLU\left(W_1.g\left(F\right)\right)\right) \tag{4}$$

where $g(F)$ represents global pooling and $W_1$ and $W_2$ are the weight matrix. These mechanisms prioritize relevant regions, enabling the detection of small, overlapping, or partially occluded objects. By efficiently combining spatial and channel-wise attention, PANet++ achieves a stability between detection and computational efficiency.

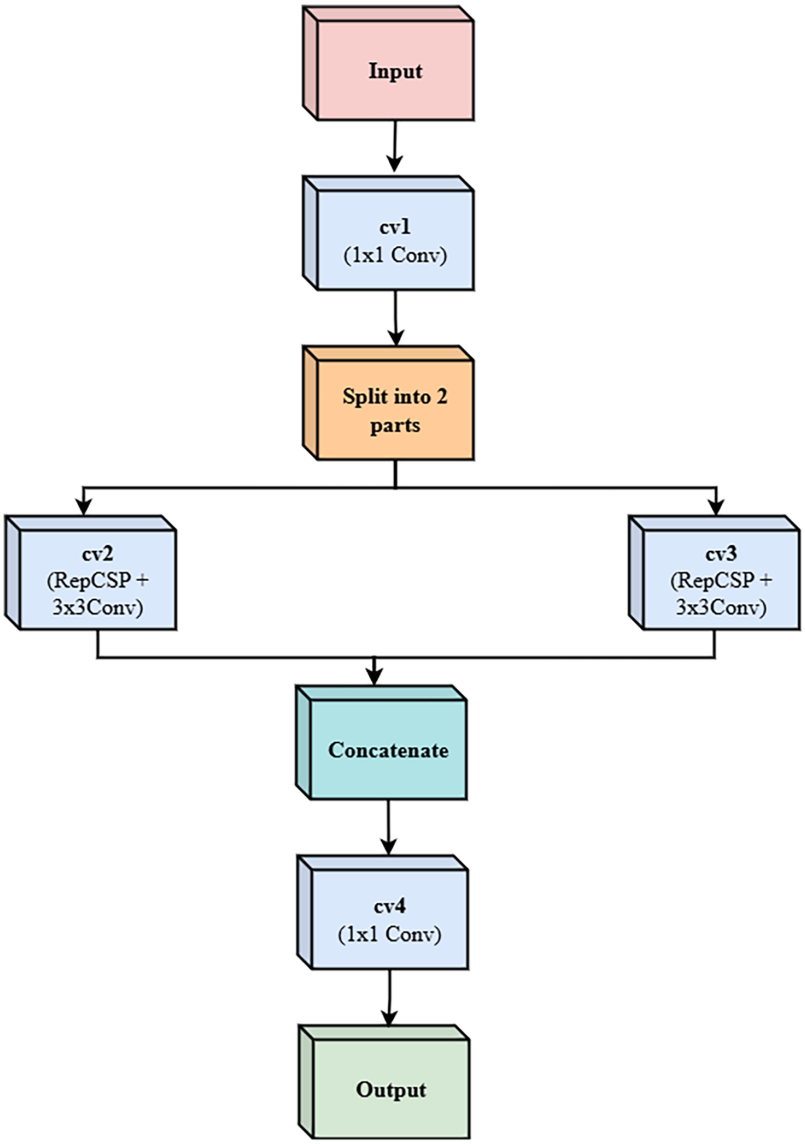



**Figure 4:** Architecture of RepNCSPELAN Block.

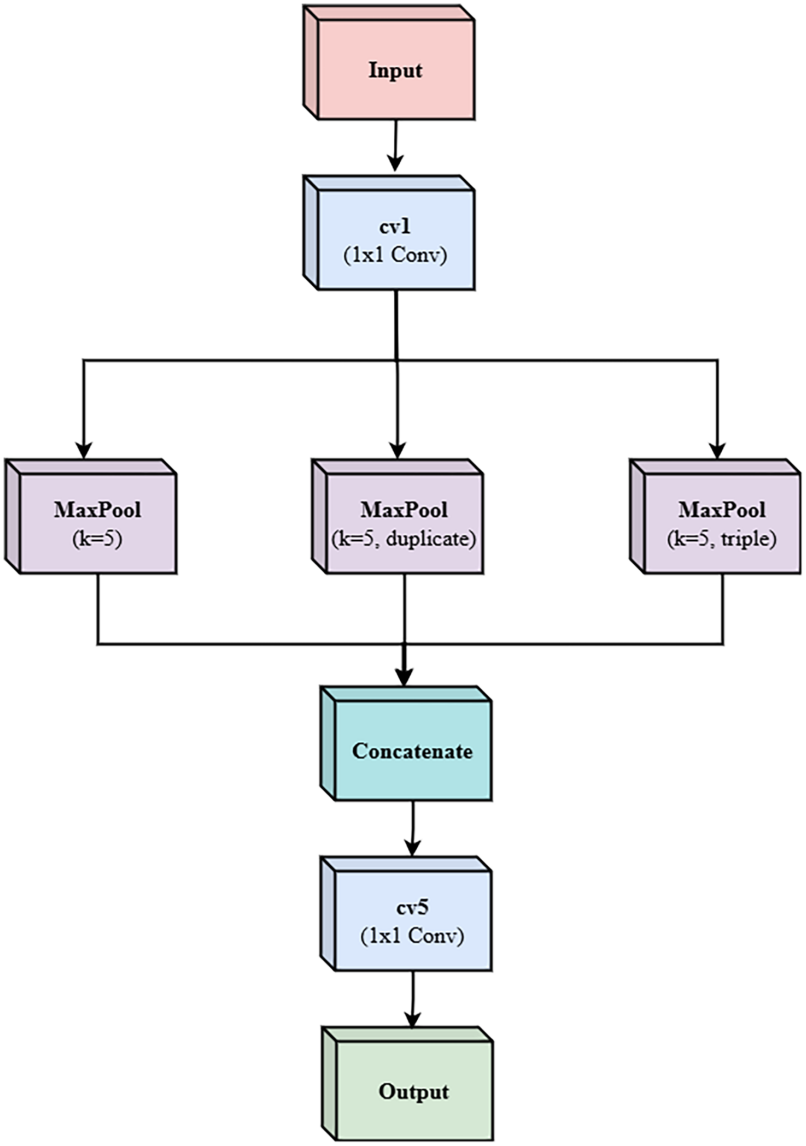



**Figure 5:** Architecture of SPPELAN Block.

### *2.2.3 Detection Head*

The Detection Head is responsible for performing predictive tasks, including bounding box regression, object confidence scoring, and class probability estimation. YOLO v9 employs a decoupled head architecture, which separates the optimization processes for classification and regression. This separation reduces interference between the two tasks, improving training stability and prediction accuracy. The anchor-free detection head removes the need for determined anchor boxes, instead directly predicting object centres. Object confidence is calculated using a sigmoid activation function and Class probabilities are computed using the SoftMax function as shown in Eq. (5).

$$P\left(c|x\right) = \frac{e^{z_c}}{\sum_{c=1}^{C} e^{z_c}} \tag{5}$$

where $z_c$ is the raw score in reference with class $c_i$, and $C$ is number of classes. This anchor-free paradigm enhances the model's capability to detect target of diverse shapes, orientations, and aspect ratios, even within complex scenes.

### *2.3 Experimentation Setup*

A GPU embedded device was used for all-different model training. The device is configured with 2x Intel Xeon 16 core CPU with total 128 GB DDR4 RAM and a 2 TB SATA Enterprise HDD with NVIDIA RTX A5000 Graphics card. Jupyter Notebook version 6.4.3 is used for code implementation and model training. Table 3 provides the complete information about the implementation environment used for the study.

**Table 3:** Details of hardware.

| Component | Details |
|---|---|
| CPU | Intel Xeon 16 Core (x2) |
| RAM | 64 GB DDR4 ECC (x2) |
| HDDs | 2 TB SATA Enterprise |
| SSD | 240 GB SATA |
| Graphics Card | NVIDIA RTX A5000 |
| Operating System (OS) | Unix OS |

In this experimental study, multiple variants of the YOLO framework, specifically YOLO v8, YOLO v9, YOLO v10, and YOLO v11, were employed to address the challenging task of defect detection in PCBs. The YOLO v9 compact model was the first variant released in the YOLOv9 series and served as the baseline for subsequent developments. In this study, YOLO v9 compact, YOLO v9 small, and YOLO v9 medium models were selected because they achieve a better balance between accuracy, speed, and size, making them more appropriate for real-time use in industry. The YOLO v9 nano was excluded because it is intended for ultra-lightweight models, which may not achieve sufficient accuracy, especially when detecting small-scale defects on PCBs. More advanced models, such as small and medium, were preferred to ensure better precision. The main goal is to examine the capabilities of these models in terms of their accuracy in detecting defects. Additionally, study also analyzed their computational efficiency and inference speed to assess their suitability for real-time industrial applications. A diverse set of hyperparameters was employed for the fine-tuning of the model variants, as detailed in Table 4.

**Table 4:** Various hyperparameters setting used for training.

| Hyperparameters | Value |
|---|---|
| Batch Size | 16 |
| Close mosaic | 10 |
| Image Size | 640 |
| Intersection over Union (IoU) | 0.7 |
| Learning rate | 0.001 |
| Momentum | 0.9 |
| Optimizer | Adam |
| Epochs | 50 |

(Continued)

**Table 4 (continued)**

| Hyperparameters | Value |
|---|---|
| Patience | 100 |
| Weight decay | 0.0005 |
| Workers | 5 |

## 3 Evaluation Metrics

Precision is a key metric in evaluating the performance of a predictive model, specifically concentrating on accuracy of correct predictions. It measures the amount of true positive (*TP*) outcomes among all occurrences that model has predicted as positive. Precisely, precision is evaluated by dividing the number of *TP* to the sum of false positives (*FP*) and *TP*, as shown in Eq. (6).

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Further, recall or Sensitivity, is a vital metric for assessing the effectiveness of a predictive model, particularly in its ability to identify all relevant instances. As described in Eq. (7), recall can be estimated by dividing the count of *TP* to the summation of *TP* and false negatives (*FN*).

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Average Precision is represented as the part below the precision-recall curve for each discrete category. Eq. (8) is used to compute the subsequent average precision for each category. To accurately represent the model's performance across the entire dataset, mAP for any model can be computed using Eq. (9), where *N* is the total count of average precisions.

$$AP = \int_0^1 Precision.RecalldR \tag{8}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{9}$$

## 4 Results & Discussion

### *4.1 Results*

Table 5 shows a detailed comparative analysis of mAP values of different YOLO model variants that have been implemented in this study. These results are reported at a 50% confidence threshold. In this regard, the defect classes are as follows: Class 1 "Mouse Bite," Class 2 "Spur," Class 3 "Missing Hole," Class 4 "Short," Class 5 "Open Circuit," and Class 6 "Spurious Copper." Out of all the YOLO model variants implemented in this study, the YOLO v9 small model reported the maximum mAP50 value of 99.2% and a mAP50-95 value of 64.1%. Moreover, the average mAP50 value reported by all the YOLO model variants is greater than 98%. This shows that modern YOLO model variants are highly effective in precisely detecting different types of PCB defects.

**Table 5:** Comparison of mAP values for various YOLO models.

| Model | Variant | mAP50 (in%) | | | | | | | mAP50-95 (in%) |
|---|---|---|---|---|---|---|---|---|---|
| | | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. | |
| YOLO v8 | Nano | 98.8 | 99.0 | 99.5 | 98.2 | 98.8 | 99.1 | 98.9 | 60.1 |
| | Small | 99.0 | 99.2 | 99.5 | 98.6 | 99.2 | 98.9 | 99.1 | 63.0 |
| | Medium | 98.8 | 99.3 | 99.5 | 98.5 | 99.2 | 98.9 | 99.0 | 63.9 |
| YOLO v9 | **Small** | **99.2** | **99.3** | **99.5** | **98.9** | **99.2** | **99.0** | **99.3** | **64.1** |
| | Medium | 98.3 | 98.8 | 99.5 | 98.0 | 99.2 | 98.3 | 98.7 | 60.0 |
| | Compact | 99.0 | 99.4 | 99.5 | 98.5 | 99.0 | 98.8 | 99.0 | 65.0 |
| YOLO v10 | Nano | 98.8 | 98.6 | 99.5 | 97.8 | 99.3 | 98.6 | 98.8 | 60.3 |
| | Small | 99.1 | 98.9 | 99.5 | 98.3 | 99.2 | 98.7 | 98.9 | 62.4 |
| | Medium | 99.0 | 98.7 | 99.5 | 98.3 | 99.0 | 98.4 | 98.8 | 63.5 |
| YOLO 11 | Nano | 99.1 | 99.0 | 99.5 | 98.3 | 99.1 | 98.7 | 99.0 | 59.0 |
| | Small | 98.9 | 99.2 | 99.5 | 98.2 | 99.0 | 98.3 | 98.9 | 61.1 |
| | Medium | 99.1 | 99.0 | 99.5 | 99.0 | 98.9 | 98.6 | 99.0 | 62.7 |

Note: C-1: -Class 1, C-2: -Class 2, C-3: -Class 3, C-4: -Class 4, C-5: -Class 5, and C-6: -Class 6.

It is important to note that the model evaluation was primarily conducted using precision, recall, and mean Average Precision (mAP), which are standard metrics for YOLO-based object detection models. These metrics are directly computed during the training and validation phases. The values of True Positives (TP), False Positives (FP), and False Negatives (FN) are inherently used in the calculation of precision and recall, and thus are implicitly reflected in the reported performance. With the inclusion of confusion matrices shown in Fig. 6, the corresponding TP, FP, FN, and TN values can now be explicitly observed and verified, further strengthening the credibility and reproducibility of the results.

Further Tables 6 and 7 provides a comparative analysis of the precision and recall metrics for various YOLO model variants respectively. As seen from the previous tables, Class 1, Class 2, and the succeeding classes represent the corresponding defect type as defined in the previous table. From the results, it can be seen that the precision and recall values are high across all YOLO models, which confirms the robustness of the models in detecting the defects with minimum false positive values. It is also seen that the average precision of all the models is more than 96%, and the range of the precision values of the individual classes is similar to the range of the values of the recall values.

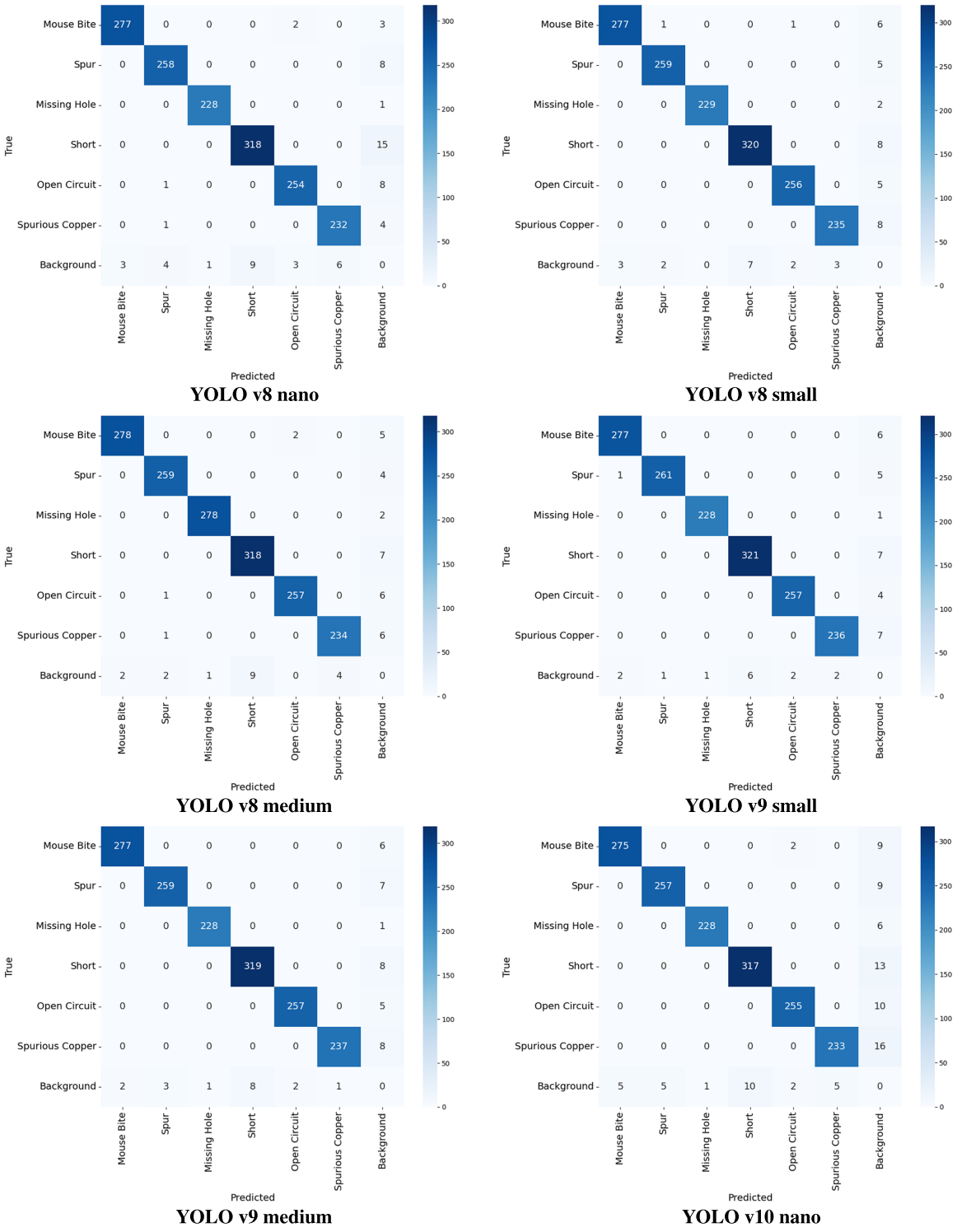



**Figure 6:** (Continued)

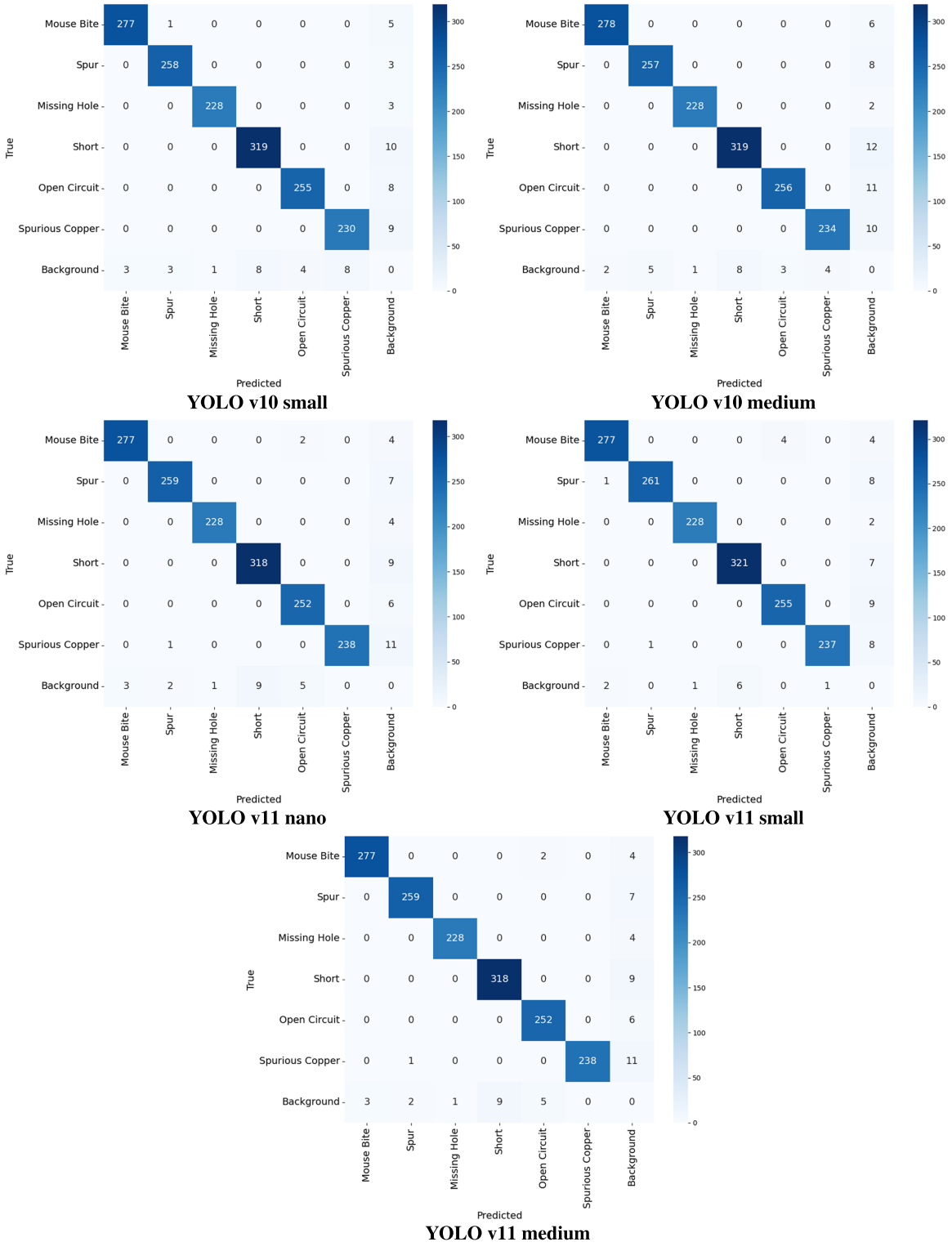



**Figure 6:** Confusion matrix of all YOLO variants.

**Table 6:** Comparison of precision value of various YOLO models.

| Model | Variant | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. |
|---|---|---|---|---|---|---|---|---|
| YOLO v8 | Nano | 0.98 | 0.98 | 0.99 | 0.96 | 0.98 | 0.98 | 0.98 |
| | Small | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.96 | 0.97 |
| | Medium | 0.97 | 0.98 | 0.99 | 0.97 | 0.97 | 0.97 | 0.98 |
| YOLO v9 | Small | 0.98 | 0.98 | 0.99 | 0.97 | 0.98 | 0.97 | 0.98 |
| | Medium | 0.97 | 0.97 | 0.99 | 0.97 | 0.98 | 0.97 | 0.97 |
| | Compact | 0.97 | 0.98 | 0.99 | 0.97 | 0.98 | 0.97 | 0.98 |
| YOLO v10 | Nano | 0.97 | 0.98 | 0.98 | 0.97 | 0.98 | 0.97 | 0.97 |
| | Small | 0.98 | 0.98 | 0.99 | 0.97 | 0.98 | 0.97 | 0.98 |
| | Medium | 0.98 | 0.98 | 0.99 | 0.97 | 0.97 | 0.96 | 0.97 |
| YOLO 11 | Nano | 0.97 | 0.98 | 0.98 | 0.97 | 0.98 | 0.95 | 0.97 |
| | Small | 0.98 | 0.97 | 0.99 | 0.97 | 0.98 | 0.97 | 0.98 |
| | Medium | 0.98 | 0.97 | 0.98 | 0.97 | 0.98 | 0.97 | 0.98 |

**Table 7:** Comparison of recall value of various YOLO models.

| Model | Variant | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. |
|---|---|---|---|---|---|---|---|---|
| YOLO v8 | Nano | 0.97 | 0.98 | 0.99 | 0.95 | 0.97 | 0.97 | 0.97 |
| | Small | 0.99 | 0.99 | 1.00 | 0.97 | 0.99 | 0.99 | 0.99 |
| | Medium | 0.97 | 0.98 | 0.99 | 0.97 | 0.97 | 0.97 | 0.98 |
| YOLO v9 | Small | 0.99 | 0.99 | 0.99 | 0.98 | 0.98 | 0.99 | 0.99 |
| | Medium | 0.99 | 0.98 | 0.99 | 0.97 | 0.99 | 0.99 | 0.98 |
| | Compact | 0.99 | 0.99 | 0.99 | 0.98 | 0.99 | 0.98 | 0.99 |
| YOLO v10 | Nano | 0.97 | 0.96 | 0.99 | 0.96 | 0.96 | 0.95 | 0.97 |
| | Small | 0.98 | 0.98 | 0.99 | 0.97 | 0.98 | 0.95 | 0.98 |
| | Medium | 0.98 | 0.98 | 0.99 | 0.97 | 0.99 | 0.97 | 0.98 |
| YOLO 11 | Nano | 0.98 | 0.97 | 0.99 | 0.96 | 0.97 | 0.98 | 0.98 |
| | Small | 0.98 | 0.99 | 0.99 | 0.97 | 0.97 | 0.98 | 0.98 |
| | Medium | 0.98 | 1.00 | 0.99 | 0.97 | 0.99 | 0.98 | 0.99 |

Fig. 7 presents a clear illustration of the relationship that exists between epochs and various loss functions, as well as evaluation metrics such as precision, recall, and mAP, with respect to YOLO v9 small variant. The validation loss curve and training loss curve depict a declining pattern as epochs increase, reflecting convergence with better optimization. The training loss curve is smooth, while some fluctuations are seen in specific loss values such as bounding box loss values and distribution focal loss values. On the other hand, evaluation metrics depict an increasing pattern with respect to epochs. The mAP50 values depict a smooth upward curve, reflecting better improvement in mAP values. The precision and recall values with respect to validation also depict an increasing pattern with some fluctuations.

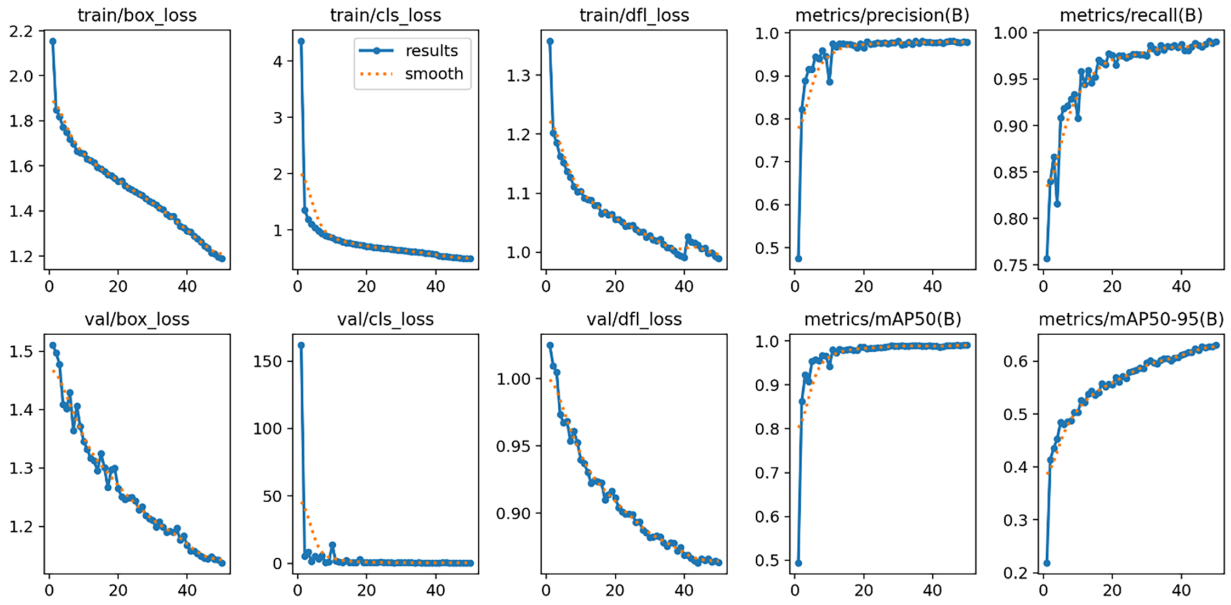
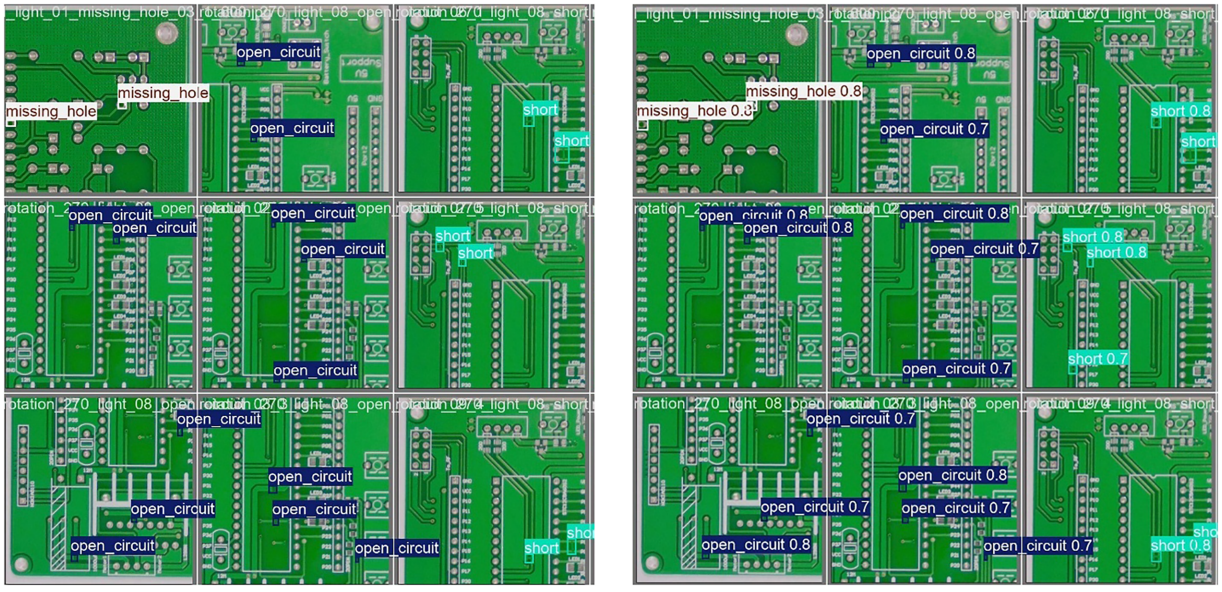



**Figure 7:** Various loss and evaluation graphs of YOLO v9 small variant.

Fig. 8A,B shows some images for demonstrating the efficiency of YOLOv9 small for defect detection. Fig. 8A shows images with actual bounding box annotations, and Fig. 8B shows images with bounding box predictions. The images shown in Fig. 8A,B demonstrates how actual and predicted bounding boxes look. The actual and predicted bounding boxes show how efficient and accurate the model is for defect detection. The actual and predicted bounding boxes look almost similar, and this shows how accurate the model is for defect detection.




**(A)** **(B)**

**Figure 8:** (**A**) Actual bounding box images (**B**) Predicted bounding box images.

Additionally, Table 8 presents the processing times for different model variants across distinct tasks in milliseconds (ms). The Table 8 details the time required for each phase, including pre-processing, inference, and post-processing, with recorded data specific to each model variant. Specifically, pre-processing refers to the preparation and transformation of input data prior to model execution, inference denotes the forward pass of the trained model to generate predictions, and post-processing involves the refinement of model outputs to obtain the final results. From the Table 8 it can be observed that on average 0.4 ms time is taken for the pre-processing while different variants have different inference time according to models' depth and complexity. It can be seen that YOLO v9 small variants have less time while bigger models have more. And similar to the pre-processing the post processing time range between 0.7 to 1.5 ms.

**Table 8:** Various comparison of time taken by model for processing.

| Model | Variant | Pre-Process (ms) | Inference (ms) | Post Process (ms) |
|---|---|---|---|---|
| **YOLO v8** | **Nano** | 0.4 | 1.0 | 1.3 |
| | **Small** | 0.4 | 1.8 | 1.1 |
| | **Medium** | 0.4 | 4.3 | 1.3 |
| **YOLO v9** | **Small** | 0.3 | 3.4 | 0.7 |
| | **Medium** | 0.4 | 6.7 | 1.9 |
| | **Compact** | 0.3 | 6.4 | 1.3 |
| **YOLO v10** | **Nano** | 0.3 | 1.8 | 0.5 |
| | **Small** | 0.4 | 4.1 | 1.3 |
| | **Medium** | 0.4 | 7.8 | 0.7 |
| **YOLO 11** | **Nano** | 0.2 | 2.2 | 1.1 |
| | **Small** | 0.2 | 4.7 | 0.9 |
| | **Medium** | 0.4 | 5.5 | 1.2 |

### *4.2 Model Optimization: Pruning and Quantization*

To further optimize YOLO models for deployment in real-time industrial environments, model pruning and quantization were applied post-training. Pruning helps in removing redundant weights in the network without significant loss in accuracy, thereby reducing the model size and computational complexity. Structured pruning techniques were used to remove less significant convolutional filters based on L1-norm criteria. On the other hand, quantization was employed to convert the 32-bit floating-point weights to 8-bit integers using post-training quantization. This reduces memory usage and accelerates inference, especially on edge devices. These optimizations combined make the model efficient in real-world manufacturing scenarios without affecting detection results. Table 9 compares the detection results of the YOLOv9 models before and after applying the pruning method. Pruning is a method in which redundant weights are removed from the network, reducing the network size and efficiency without affecting the results. The results are given in mAP at 50% IoU, a basic metric for object detection, and mAP50-95, which uses multiple IoU thresholds for a stricter evaluation metric. It is observed that for all six classes, namely, C-1, C-2, C-3, C-4, C-5, and C-6, the average mAP50 remains consistent at 99.0% after applying the pruning method, indicating that detection results are maintained. Interestingly, the average mAP50-95 decreases from 66.0% to 63.1%, indicating better generalization.

**Table 9:** mAP values of YOLOv9 variants with and without pruning.

| Model | mAP50 (%) | | | | | | | Avg. mAP50-95(%) |
|---|---|---|---|---|---|---|---|---|
| | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. | |
| **Without Pruning** | 99.2 | 99.3 | 99.5 | 98.9 | 99.2 | 99.0 | 99.2 | 66.0 |
| **With Pruning** | 99.0 | 98.7 | 98.9 | 98.3 | 98.9 | 99.0 | 99.0 | 63.1 |

Table 10 shows the precision and recall values for the YOLOv9 model before and after pruning. Precision refers to the correctness of the detected objects, while recall refers to the detection of the actual objects. Both precision and recall are measured over six classes and then averaged. From the table, it is evident that the precision and recall values remain the same even after pruning, i.e., precision remains at 0.98, while the recall value drops slightly from 0.99 to 0.98, proving that the performance of the model remains the same even after pruning.

**Table 10:** Precision and recall values of YOLOv9 variants after pruning.

| Model | Precision | | | | | | |
|---|---|---|---|---|---|---|---|
| | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. |
| **Without Pruning** | 0.98 | 0.98 | 0.99 | 0.97 | 0.98 | 0.97 | 0.98 |
| **With Pruning** | 0.98 | 0.97 | 0.98 | 0.97 | 0.98 | 0.97 | 0.98 |
| **Model** | **Recall** | | | | | | |
| | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. |
| **Without Pruning** | 0.99 | 0.99 | 0.99 | 0.98 | 0.98 | 0.99 | 0.99 |
| **With Pruning** | 0.98 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 | 0.98 |

Table 11 shows the effect of post-training quantization on the YOLOv9 model, using the FP16 and INT8 formats. The quantization technique decreases the precision of the weights, leading to a decrease in the size of the model as well as the inference time from the original model. The average mAP50-95 is slightly decreased, which is expected ass the precision of the weights is decreased during the quantization process.

**Table 11:** mAP values of YOLOv9 variants with and without quantization.

| Model | mAP50 (%) | | | | | | | Avg. mAP50-95(%) |
|---|---|---|---|---|---|---|---|---|
| | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. | |
| **Without quantization (FP 32)** | 99.2 | 99.3 | 99.5 | 98.9 | 99.2 | 99.0 | 99.2 | 63.1 |
| **Quantization (FP16)** | 99.2 | 99.2 | 99.5 | 98.0 | 98.8 | 99.0 | 99.1 | 61.2 |
| **Quantization (INT8)** | 99.1 | 98.6 | 98.9 | 98.3 | 98.4 | 98.6 | 98.6 | 60.6 |

Table 12 demonstrates the effect of quantization on the precision and recall values of the models. Precision and re-call values are provided for the FP16 and INT8 quantized models compared to the full pre-cision model. From the values provided in the table, it is evident that the performance of the quantized

models is similar to the original model. Precision values are always close to 0.978, and the recall values are always greater than 0.987.

**Table 12:** Precision and recall values of YOLOv9 with quantization.

| Model | Precision | | | | | | |
|---|---|---|---|---|---|---|---|
| | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. |
| **Without quantization (FP32)** | 0.978 | 0.985 | 0.997 | 0.973 | 0.983 | 0.971 | 0.981 |
| **Quantization (FP16)** | 0.982 | 0.98 | 0.989 | 0.977 | 0.973 | 0.992 | 0.978 |
| **Quantization (INT8)** | 0.982 | 0.977 | 0.985 | 0.979 | 0.973 | 0.969 | 0.978 |
| **Model** | **Recall** | | | | | | |
| | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | Avg. |
| **Without quantization** | 0.996 | 0.992 | 0.996 | 0.982 | 0.996 | 0.987 | 0.992 |
| **Quantization (FP16)** | 0.986 | 0.985 | 0.996 | 0.979 | 0.987 | 0.992 | 0.987 |
| **Quantization (INT8)** | 0.989 | 0.989 | 0.991 | 0.980 | 0.987 | 0.992 | 0.988 |

Finally, in Table 13, we have a consolidated table for all critical metrics before and after the application of pruning and quantization. The metrics include mAP scores, precision, recall, processing time per image, the number of layers, and computational complexity in terms of GFLOPs. Some of the key observations are as follows: The number of layers has been reduced after applying pruning, from 917 to 486. The GFLOPs are reduced after applying quantization, from 27.4 to 26.7. The processing time per image has been reduced, from 5.20 to 4.27 ms. However, it is observed that mAP50 is still as high as 98.6%, precision is as high as 97.8%, and recall is as high as 98.8%, thus ensuring that the model is accurate as well as efficient.

**Table 13:** Comparison of different evaluation matrices.

| Variant | Before Optimization | After Pruning | After Quantization |
|---|---|---|---|
| **mAP50** | 99.3 | 99.0 | 98.6 |
| **mAP50-95** | 64.1 | 66.7 | 60.6 |
| **Precision** | 98.0 | 98.2 | 97.8 |
| **Recall** | 99.0 | 98.9 | 98.8 |
| **Processing time (ms)** | 5.20 | 4.27 | 4.27 |
| **Layers** | 917 | 486 | 486 |
| **GFLOPs** | 27.4 | 26.7 | 26.7 |

Further this study evaluates the performance of various optimizers and loss function combinations in training the YOLO object detection model. Specifically, the performance of three optimizers, i.e., Adam, SGD, and AdamW, is evaluated using various loss functions, including Binary Cross-Entropy + Complete Intersection over Union (BCE+CIoU), Focal Loss + CIoU (Focal+CIoU), Binary Cross-Entropy + Generalized Intersection over Union (BCE+GIoU), and Focal Loss + GIoU (Focal+GIoU). The performance of the model is evaluated on six classes with distinct mAP metrics at 50% IoU threshold (mAP50) and averaged over IoU thresholds ranging from 50% to 95%. It is found that the performance of the model using the optimizer-loss combinations varies significantly. It has also been found that the combination of the Adam optimizer with the BCE+CIoU loss function gives the best performance, i.e., an average mAP50 of 99.3% and

an average mAP50-95 of 64.1%, indicating the best possible detection accuracy and robustness. Among the optimizers used, SGD with BCE+CIoU and AdamW with BCE+CIoU were seen to perform competitively with a mAP50 of 99.2%. However, these were seen to obtain slightly lower mAP50-95 values, indicating slightly lower precision with varying values of IoU thresholds than Adam with BCE+CIoU. In the evaluation of the loss functions, models with BCE loss were seen to outperform models with Focal loss in all cases, with the highest mAP50 of 99.1% achieved with the Focal loss function, although with a significantly lower mAP50-95 of 63.0%. This indicates that the BCE loss function provides a more accurate model with varying values of IoU thresholds, as depicted in Table 14. In the evaluation of the IoU loss functions, all models with CIoU were seen to outperform the models with GIoU. In conclusion, Adam with BCE+CIoU loss function and optimizer were seen to provide the best results, thus making it the preferred option for obtaining robust and accurate object detection with the YOLO architecture.

**Table 14:** Performance comparison of YOLO v9 model across optimizers and loss functions.

| Optimizer | Loss Function | C-1 | C-2 | C-3 | C-4 | C-5 | C-6 | mAP50 (%) | mAP50-95 (%) |
|---|---|---|---|---|---|---|---|---|---|
| **Adam** | | **99.5** | **99.2** | **99.3** | **99.3** | **99.3** | **99.4** | **99.3** | **64.1** |
| SGD | **BCE + CIoU** | 99.3 | 99.4 | 99.6 | 99 | 99.3 | 99.2 | 99.2 | 63.8 |
| AdamW | | 99.2 | 99.3 | 99.5 | 98.9 | 99.2 | 99.1 | 99.2 | 63.5 |
| SGD | | 99.1 | 99.2 | 99.4 | 98.8 | 99.1 | 98.9 | 99.1 | 63.0 |
| Adam | Focal + CIoU | 99.0 | 99.1 | 99.3 | 98.7 | 99.0 | 98.8 | 99.0 | 62.7 |
| AdamW | | 98.9 | 99 | 99.2 | 98.6 | 98.9 | 98.7 | 98.9 | 62.5 |
| SGD | | 99.2 | 99.3 | 99.4 | 99.1 | 99.2 | 99.1 | 99.2 | 63.0 |
| Adam | BCE + GIoU | 99.1 | 99.2 | 99.3 | 99.0 | 99.1 | 99.0 | 99.1 | 62.8 |
| AdamW | | 99.0 | 99.1 | 99.2 | 98.9 | 99.0 | 98.9 | 99.0 | 62.5 |
| SGD | | 99.0 | 99.1 | 99.2 | 98.8 | 99.0 | 98.8 | 99.0 | 62.3 |
| Adam | Focal + GIoU | 98.9 | 99.0 | 99.1 | 98.7 | 98.9 | 98.7 | 98.9 | 62.1 |
| AdamW | | 98.8 | 98.9 | 99.0 | 98.6 | 98.8 | 98.6 | 98.8 | 62.0 |

### *4.3 Industrial Deployment and Real-World Applicability*

To establish the applicability of the DL-based defect detection system, this section discusses its implementation in real-world manufacturing environments. The discussion aligns with recent advancements in Industry 4.0, edge AI, and smart manufacturing systems, which emphasize real-time decision-making, automation, and intelligent interaction between physical and digital systems. The DL-based defect detection system can be deployed using industrial-grade hardware components that support real-time processing environments. In this context, edge computing devices equipped with GPU acceleration, such as NVIDIA Jetson modules, represent suitable options capable of efficiently executing DL models. These devices offer an optimal balance between processing power and energy efficiency, making them highly appropriate for real-world manufacturing applications. Industrial-grade cameras are utilized to capture product images on conveyor belts, with proper synchronization ensured during system implementation—an essential characteristic of cyber-physical production systems in the context of Industry 4.0 [34].

The primary advantage of the proposed system is its ability to operate effectively within an edge computing environment. In other words, the system achieves low-latency responses without relying on cloud infrastructure. This capability is made possible by deploying the trained model directly on the edge device, which is highly beneficial for industrial automation. In recent years, Edge AI has proven crucial for enabling

real-time analytics and decision-making in smart factories while simultaneously ensuring data privacy and operational reliability [35].

Another advantage of the proposed system lies in its seamless integration with existing quality control pipelines. The system can interface easily with existing Programmable Logic Controllers (PLCs), which are widely used in industrial automation systems. The PLC can trigger image acquisition based on sensor inputs. Once the deep learning system classifies a product as defective or non-defective, the output signal is transmitted to the PLC, which then activates rejection mechanisms such as air jets, robotic arms, or pushers. This level of automation aligns with the smart manufacturing paradigm, which leverages AI-based systems to enhance operational efficiency with minimal human intervention [36]. Fig. 9 illustrates a schematic representation of the proposed industrial deployment architecture for PCB defect detection.

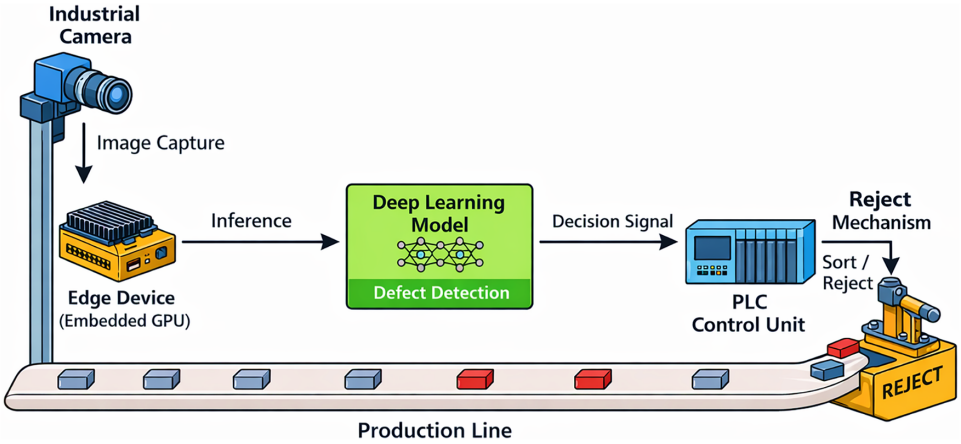



**Figure 9:** Pictorial representations of industrial deployment for PCB defect detection.

### *4.4 Strengths and Limitations*

The proposed work has shown promising results in the field of automated defect detection on a PCB, with the YOLOv9 model achieving impressive accuracy (mAP50 = 99.2%) in detecting six different defect types, using enhanced techniques in feature extraction and detection. The addition of Programmable Gradient Information (PGI) has also been effective in ensuring detection consistency, along with optimization techniques such as pruning and quantization, which reduce the size of the model and improve inference speed.

However, the proposed work has also encountered certain limitations in detecting fine defects such as short connections, owing to their subtle visual characteristics, as well as limitations in bounding box localization. Another drawback of the proposed work is that it has been tested on a specific dataset, which creates concerns regarding the generalization of the proposed work on various datasets related to PCBs. Other problems that have been encountered are class imbalance and sensitivity to image conditions.

### *4.5 Future Scope*

Recent research on the detection of PCB defects has shown that Secure Sockets Layer (SSL)-based frameworks have the capability of utilizing the available labeled as well as a large number of unlabeled data, thereby improving the overall performance of the defect detection model. For example, a semi-supervised model of PCB defect detection, i.e., PCB_SS, has been shown to improve the performance of the defect detection model, especially when the available data is limited or noisy [37]. The semi-supervised approach is highly suitable in industrial applications as it has the following advantages:

- Improve the robustness of the model, especially when the available data is partially incorrect, as the accuracy of the model remains the same even in the presence of noisy data.
- Improve the overall generalization of the model by utilizing a large number of unlabeled PCB data.
- Reduce the overall dependency on the expensive annotation process, especially during the scaling of the model.

In particular, it has been shown that techniques such as consistency regularization, pseudo-labelling, and augmentation-based training strategies are effective for SSL in the field of PCB inspection systems. This is because these techniques are able to allow the model to learn more robust feature representations using labelled and unlabelled data. Although the focus of the current work is on supervised learning, it is recognised that the addition of SSL techniques is a very promising area for future work. This is because, as demonstrated in previous works, the addition of unlabelled data can greatly improve the robustness of the model, particularly in data-limited scenarios, which is a common problem in industrial environments.

## 5 Conclusion

The contribution of this study to the field of automated PCB defect detection is significant, as it experimented with various versions of the YOLO model. Among all versions, YOLOv9 outperformed the others by achieving a high mAP@50 of 99.2% and a mAP@50–95 of 63.1%. The primary reasons behind this achievement are its improved feature extraction and detection strategies, which enhanced several performance metrics, including mAP and F1-score, across all defect types. An important aspect of this study is the integration of PGI with YOLOv9, which enabled the detection of all six defect categories. The capability to identify defects of varying sizes is particularly beneficial for improving detection accuracy across different types of PCBs. This study also demonstrated the practical benefits of model optimization in real-world applications. The YOLOv9-small model, which achieved the best baseline performance, was further optimized using pruning and quantization techniques. As a result, the model size was reduced by 36.4%, and inference speed increased by 24.7%, while maintaining a high detection accuracy with a mAP@50 of 98.6%. Although a minor reduction in accuracy was observed, the model's performance remained robust with the available dataset.

One limitation observed with YOLOv9 was its difficulty in detecting short connection defects. These defects are characterized by very thin lines or bridges that form between two conductors. Detecting such fine defects is challenging because YOLO models often struggle to accurately position bounding boxes or define anchor points for extremely small features. To address this issue, future work will focus on several enhancements, including the addition of more short defect samples, the use of higher-resolution images, the incorporation of attention mechanisms for improved detection, and the application of weighted loss functions to better prioritize rare defects. However, additional testing with other datasets is required to further validate its generalization capability. In future work, we plan to apply transfer learning to enhance model adaptability across different PCB datasets, improve generalization techniques, and expand the model to include more defect categories. These efforts aim to advance the development of robust and efficient deep learning–based inspection systems for smart electronics manufacturing.

**Acknowledgement:** Not applicable.

**Funding Statement:** This work was supported by Dongseo University "Dongseo Frontier Project" Research Fund of 2025.

**Author Contributions:** The authors confirm contribution to the paper as follows: Conceptualization, Jigar Sarda and Rohan Vaghela; methodology, Jigar Sarda and Rohan Vaghela; software, Jigar Sarda, Rohan Vaghela and Mangal Sain;

validation, Jigar Sarda, Rohan Vaghela, Mangal Sain, Chang-Won Yoon, and Akash Kumar Bhoi; formal analysis, Jigar Sarda, Rohan Vaghela, Mangal Sain, Chang-Won Yoon, and Akash Kumar Bhoi; writing—original draft preparation, Jigar Sarda, Rohan Vaghela, Mangal Sain, Chang-Won Yoon, and Akash Kumar Bhoi; writing—review and editing, Jigar Sarda, Rohan Vaghela, Mangal Sain, Chang-Won Yoon, and Akash Kumar Bhoi. All authors reviewed and approved the final version of the manuscript.

**Availability of Data and Materials:** Data available on request from the authors.

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References


1. Coultard F. Past, present & future. Circuit World. 2005;31(2):21731baf.001. doi:10.1108/cw.2005.21731baf.001.
2. DeVogeleer K, Memmi G, Jouvelot P, Coelho F. Modeling the temperature bias of power consumption for nanometer-scale CPUs in application processors. In: Proceedings of the 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV); 2014 Jul 14–17; Agios Konstantinos, Greece. New York, NY, USA: IEEE; 2014. p. 172–80. doi:10.1109/SAMOS.2014.6893209.
3. Millennium Circuits Limited. What is the PCB manufacturing process [Internet]? 2025 [cited 2026 Jan 1]. Available from: https://www.mclpcb.com/pcb-manufacturing-process/.
4. Smith CJ, Adendorff K. Advantages and limitations of an automated visual inspection system. S Afr N J Ind Eng. 2012;5(1):27–63. doi:10.7166/5-1-423.
5. Liu B, Chen D, Qi X. YOLO-PDD: a novel multi-scale PCB defect detection method using deep representations with sequential images. arXiv:2407.15427. 2024.
6. Zhu H, Huang J, Liu H, Zhou Q, Zhu J, Li B. Deep-learning-enabled automatic optical inspection for module-level defects in LCD. IEEE Internet Things J. 2022;9(2):1122–35. doi:10.1109/JIOT.2021.3079440.
7. Ding R, Dai L, Li G, Liu H. TDD-net: a tiny defect detection network for printed circuit boards. CAAI Trans Intell Technol. 2019;4(2):110–6. doi:10.1049/trit.2019.0019.
8. Li J, Gu J, Huang Z, Wen J. Application research of improved YOLO V3 algorithm in PCB electronic component detection. Appl Sci. 2019;9(18):3750. doi:10.3390/app9183750.
9. Bhandarkar VV, Karnati M, Tandon P. Real-time layer-by-layer defects detection in 3D-printed multi-geometry polymer components using deep learning. Measurement. 2026;265:120332. doi:10.1016/j.measurement.2026.120332.
10. Bhandarkar VV, Karnati M, Tandon P. Defect detection in 3D-printed polymer parts using deep learning models: a comparative investigation. Rapid Prototyp J. 2025;31(7):1428–48. doi:10.1108/rpj-09-2024-0395.
11. Taha K. Observational and experimental insights into machine learning-based defect classification in wafers. J Intell Manuf. 2026;37(1):45–95. doi:10.1007/s10845-024-02521-0.
12. Lim J, Lim J, Baskaran VM, Wang X. A deep context learning based PCB defect detection model with anomalous trend alarming system. Results Eng. 2023;17(7):100968. doi:10.1016/j.rineng.2023.100968.
13. Chen X, Wu Y, He X, Ming W. A comprehensive review of deep learning-based PCB defect detection. IEEE Access. 2023;11(6):139017–38. doi:10.1109/ACCESS.2023.3339561.
14. Bhattacharya A, Cloutier SG. End-to-end deep learning framework for printed circuit board manufacturing defect classification. Sci Rep. 2022;12(1):12559. doi:10.1038/s41598-022-16302-3.
15. Xiao G, Hou S, Zhou H. PCB defect detection algorithm based on CDI-YOLO. Sci Rep. 2024;14(1):7351. doi:10.1038/s41598-024-57491-3.
16. Guo H, Zhao H, Zhao Y, Liu W. PCB defect detection algorithm based on deep learning. Optik. 2024;315:172036. doi:10.1016/j.ijleo.2024.172036.
17. de Oliveira GG, Caumo Vaz G, Antonio Andrade M, Iano Y, Ronchini Ximenes L, Arthur R. System for PCB defect detection using visual computing and deep learning for production optimization. IET Circuits Devices Syst. 2023;2023(1):6681526. doi:10.1049/2023/6681526.

18. Yang Z, Li D, Hou L, Nai W. A comprehensive performance evaluation of YOLO series algorithms in automatic inspection of printed circuit boards. Machines. 2026;14(1):94. doi:10.3390/machines14010094.
19. Wang Z, Yuan P, Zhang X. Yolo-cd: a lightweight real-time PCB defect detection model for edge deployments. J Real Time Image Process. 2026;23(1):46. doi:10.1007/s11554-025-01847-z.
20. Li G, Gan Y, Zhang W, Che H. GS-YOLO: a lightweight and high-performance method for PCB surface defect detection. Expert Syst Appl. 2026;303(14):130583. doi:10.1016/j.eswa.2025.130583.
21. Nasri S, Ahmad N, Aini QUA, Qayyum A, Ul Islam N. A YOLOv9: deep learning-based framework defect detection method for PCBs. J Electron Test. 2025;41(4):545–59. doi:10.1007/s10836-025-06194-2.
22. Li W. Detecting defects in PCB manufacturing: an exploration using Yolov8 deep learning. Int J Interact Des Manuf Ijidem. 2025;19(5):3505–15. doi:10.1007/s12008-024-01986-w.
23. Wang X, Zhang H, Liu Q, Gong W, Bai S, You H. You-only-look-once multiple-strategy printed circuit board defect detection model. IEEE Multimed. 2024;31(1):76–87. doi:10.1109/MMUL.2024.3359267.
24. Kong WB, Zhang ZF, Zhang TL, Wang L, Cheng ZY, Zhou M. SMC-YOLO: surface defect detection of PCB based on multi-scale features and dual loss functions. IEEE Access. 2024;12(4):137667–82. doi:10.1109/ACCESS.2024.3434559.
25. Zhou G, Yu L, Su Y, Xu B, Zhou G. Lightweight PCB defect detection algorithm based on MSD-YOLO. Clust Comput. 2024;27(3):3559–73. doi:10.1007/s10586-023-04156-x.
26. Yuan M, Zhou Y, Ren X, Zhi H, Zhang J, Chen H. YOLO-HMC: an improved method for PCB surface defect detection. IEEE Trans Instrum Meas. 2024;73:2001611. doi:10.1109/TIM.2024.3351241.
27. Tang J, Liu S, Zhao D, Tang L, Zou W, Zheng B. PCB-YOLO: an improved detection algorithm of PCB surface defects based on YOLOv5. Sustainability. 2023;15(7):5963. doi:10.3390/su15075963.
28. Du B, Wan F, Lei G, Xu L, Xu C, Xiong Y. YOLO-MBBi: PCB surface defect detection method based on enhanced YOLOv5. Electronics. 2023;12(13):2821. doi:10.3390/electronics12132821.
29. An K, Zhang Y. LPViT: a transformer based model for PCB image classification and defect detection. IEEE Access. 2022;10:42542–53. doi:10.1109/ACCESS.2022.3168861.
30. Ren Z, Fang F, Yan N, Wu Y. State of the art in defect detection based on machine vision. Int J Precis Eng Manuf Green Technol. 2022;9(2):661–91. doi:10.1007/s40684-021-00343-6.
31. Hu B, Wang J. Detection of PCB surface defects with improved faster-RCNN and feature pyramid network. IEEE Access. 2020;8:108335–45. doi:10.1109/ACCESS.2020.3001349.
32. Elter N. PCB defect dataset [Dataset]. [cited 2025 Oct 20]. Available from: https://www.kaggle.com/datasets/norbertelter/pcb-defect-dataset?resource=download.
33. Wang CY, Yeh IH, Mark Liao HY. YOLOv9: learning what you want to learn using programmable gradient information. arXiv:2402.13616. 2024.
34. Fernández-Miguel A, García-Muiña FE, Ortíz-Marcos S, Jiménez-Calzado M, Fernández del Hoyo AP, Settembre-Blundo D. AI-driven transformations in manufacturing: bridging industry 4.0, 5.0, and 6.0 in sustainable value chains. Future Internet. 2025;17(9):430. doi:10.3390/fi17090430.
35. Pan Y, Huang Z, Lev B, Xu L, Olson D. A literature review on the artificial intelligence in manufacturing systems under industry 5.0. Int J Prod Res. 2026:1–42. doi:10.1080/00207543.2026.2623537.
36. Bandhana A, Vokřínek J. AI-driven manufacturing: surveying for industry 4.0 and beyond. Oper Res Forum. 2025;6(4):145. doi:10.1007/s43069-025-00554-6.
37. Pham TTA, Thoi DKT, Choi H, Park S. Defect detection in printed circuit boards using semi-supervised learning. Sensors. 2023;23(6):3246. doi:10.3390/s23063246.