A Lightweight YOLOv8–MobileViT Framework for Real-Time PCB Defect Detection and Severity Assessment on Edge Devices

Shree Santh B, Kirupashankar C, S.Manimaran and David Amar Raj

***Abstract*—In current electronics manufacturing, where high throughput and strict quality standards necessitate accurate and efficient inspection methods, automated flaw inspection of printed circuit boards (PCBs) is a must. YOLO-based object detectors have been very good at finding defects in PCBs, but they can't be used on edge devices with limited resources since they are too complicated to run and don't have enough semantic reasoning power. This research suggests a lightweight and smart multi-stage inspection framework for finding PCB defects and figuring out how bad they are in order to deal with these problems. In the first step, a modified YOLOv8 architecture is introduced that combines MobileNet-style depthwise separable convolutions with optimised C2f blocks. This reduces the number of parameters significantly while keeping the ability to represent features at different scales. The suggested detector gets a $\mathrm{mAP}_{0.5}$ of 98.6% on the PKU PCB defect dataset with only 1.15 million parameters. This is around 60% fewer parameters than the baseline YOLOv8 model. In the second stage, we use transfer learning with MobileViT-XS to get global contextual semantics for analysing the severity of faults. Then, we use a fuzzy logic-based grading system to put problems into different levels of severity. Numerous studies on ablation and comparisons show that the proposed design effectively balances detection accuracy and computational efficiency. The real-world tests on the Raspberry Pi 5 and NVIDIA Jetson Orin Nano show that it can make inferences quickly (24.16 ms), uses little power, and can work on diverse edge platforms. The recommended system offers a practical and deployable method for detecting defects in PCBs in real-time within industrial environments.**

***Index Terms*—Printed Circuit Board Inspection, YOLOv8, Lightweight Object Detection, Edge AI, Defect Severity Classification**

I. Introduction

Printed circuit boards are essential components of today's electronics. You can find them in various fields, such as medicine, industry, transportation, and homes. As electrical devices get smaller and more complex, PCB production needs to be more precise and have a higher structural density. Even small defects, like missing holes, short circuits, or excess copper, can make circuits very unstable. This can lead to functional failures and significant financial losses. Automated PCB fault detection that is both accurate and fast has become a crucial part of quality control in today's electronics manufacturing. Recent progress in deep learning, particularly with convolutional neural networks (CNNs), has significantly improved automated defect detection. This improvement enables complete feature learning. Object detection frameworks like Faster R-CNN, SSD, and the You Only Look Once (YOLO) family have become popular for PCB inspection because they can localize and classify defects accurately at the same time. Among these, YOLO-based detectors are particularly appealing for industrial uses due to their real-time inference capability. However, standard YOLO architectures often have a high number of parameters and complex computations, which limits their use on resource-limited edge platforms. Additionally, most current methods focus only on defect detection. They miss the bigger picture, such as assessing the severity of defects. This is important for making smart choices in industrial quality control. To overcome these limitations, this paper introduces a lightweight inspection framework that combines an optimized YOLOv8-based detector with a transfer learning severity classification pipeline. The proposed YOLOv8-MBNet model aims to reduce computational load while ensuring strong multi-scale defect detection performance. This makes it appropriate for real-time edge deployment. Additionally, a transformer-assisted MobileViT-XS module and a fuzzy logic grading system are included to enable the assessment of semantic defect severity. Through extensive experimental testing and real-world use on Raspberry Pi and NVIDIA Jetson Orin Nano platforms, the proposed framework demonstrates a good balance between detection accuracy, computational efficiency, and practical usability for industrial PCB inspection.

The rest of this paper is organized in the following way. Section II reviews related work on PCB defect detection and lightweight YOLO-based inspection frameworks. Section III describes the dataset characteristics and the challenges in industrial PCB inspection. Section IV presents the proposed YOLOv8-MBNet architecture and a transfer learning-based severity estimation process that uses MobileViT-XS. Section V discusses experimental results and performance evaluation, including detection accuracy and efficiency analysis. Section VI provides a detailed ablation study to examine the contribution of individual architectural parts. Section VII presents a qualitative and quantitative comparison with recent leading methods. Section VIII evaluates real-world edge deployment on Raspberry Pi and NVIDIA Jetson Orin Nano platforms. Finally, Section IX wraps up the paper and discusses future research directions.

II. Literature Review

Recent advancements in detecting defects in printed circuit boards have mainly aimed at improving detection accuracy

Amrita School of Artificial Intelligence, Amrita Vishwa Vidyapeetham, Coimbatore, India.

TABLE I: Literature Review on Recent PCB Defect Detection Models

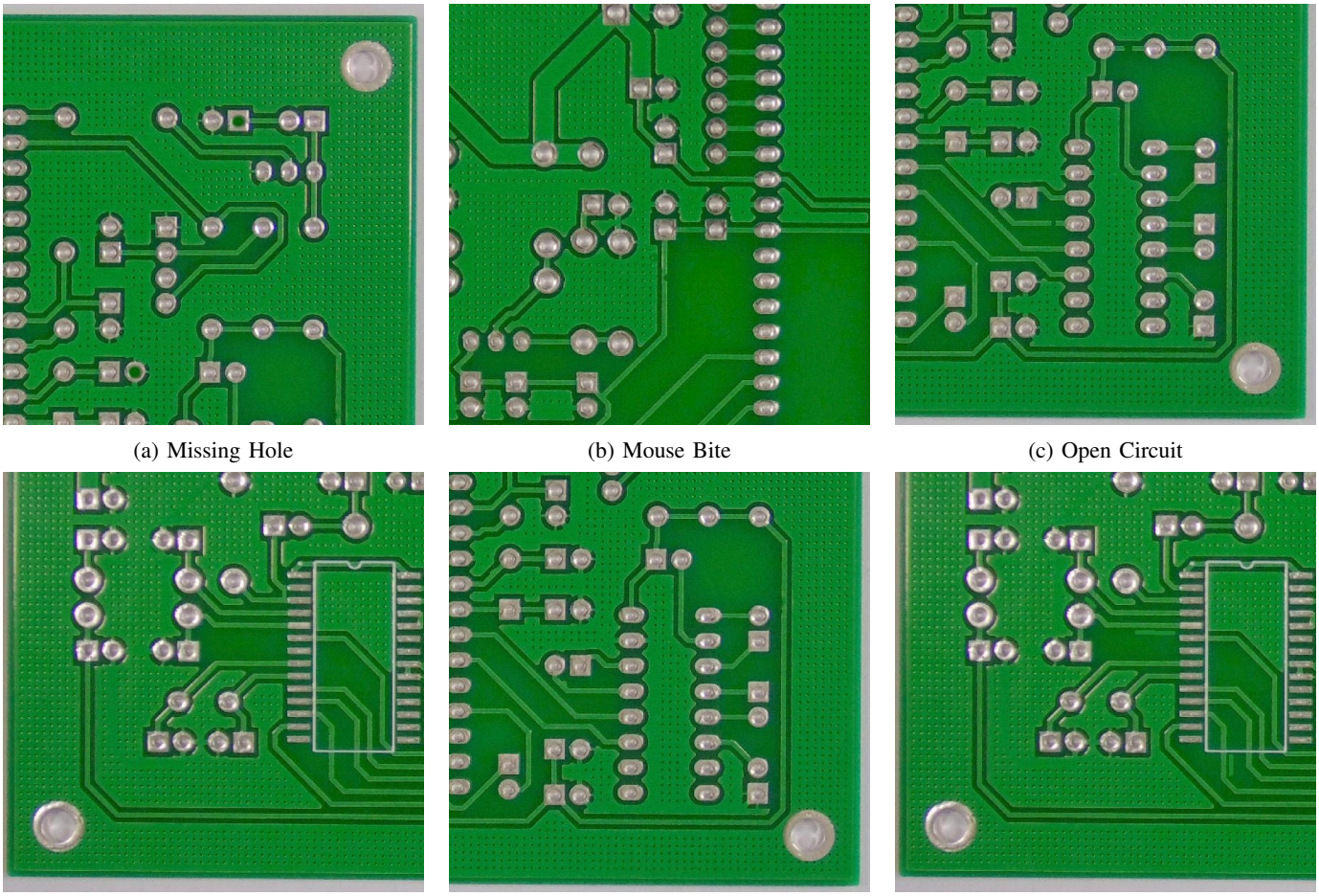
Ref.	Year	Proposed Method / Description	Model Used	Core Strengths	Limitations
[1]	2025	Efficient DNN for industrial surface defect inspection on edge devices	Lightweight CNN	Optimized for edge deployment; robust to noise	No multi-scale modeling; no semantic reasoning
[2]	2024	Autoencoder-assisted YOLO for PCB defect detection	AE-YOLO	Improved background suppression	Additional stages increase complexity
[3]	2026	GSConv-based YOLO for lightweight PCB defect detection	GS-YOLO	Preserves spatial features; lightweight	No severity or semantic analysis
[4]	2024	Compression-based YOLO for tiny PCB defect detection	Compressed YOLO	High recall for small defects	Platform-dependent retraining required
[5]	2025	Channel-attention enhanced YOLO for PCB inspection	SEConv-YOLO	Improved precision via attention	Increased inference latency
[6]	2022	Transformer-integrated YOLO for contextual PCB defect detection	Transformer-YOLO	Strong long-range dependency modeling	Computationally expensive for edge use
[7]	2024	Comprehensive survey of PCB defect detection methods	Review Study	Broad trend and challenge analysis	No deployable framework proposed
[8]	2023	Dense feature aggregation for small PCB defect detection	YOLO-DFA	High sensitivity to small defects	High GFLOPs and memory usage
[9]	2021	Online PCB defect detection system with dataset construction	Online CNN	Demonstrates real-time feasibility	Lacks lightweight optimization
[10]	2022	Detection of densely packed PCB components	Modified YOLO	High precision in dense layouts	Increased architectural complexity
[11]	2023	Multi-scale dilated convolution-based PCB defect detection	MSD-YOLO	Effective multi-scale feature extraction	Dilated convolutions increase cost
[12]	2023	Hierarchical multi-class PCB defect detection	YOLO-HMC	Structured hierarchical classification	No severity grading; limited edge focus
[13]	2020	Review of machine vision-based industrial defect detection	Review Study	Insightful accuracy–efficiency analysis	No lightweight deployable models
[14]	2024	Improved YOLOv8 for dense PCB defect detection	Modified YOLOv8	Enhanced dense defect detection	High parameter count remains

and reducing computational complexity for industrial use. Early efforts highlighted efficient convolutional structures designed for edge sensing environments. Researchers developed lightweight deep neural networks to balance strength and speed of inference while using limited hardware conditions [1]. Although these methods proved effective for surface defect inspection, they did not include specialized tools to address the multi-scale and fine-grained nature of PCB defects. To deal with background noise and improve feature representation, autoencoder-assisted detection frameworks, like AE-YOLO, were introduced. These frameworks achieved a moderate reduction in parameters and better visual discrimination [2]. However, adding auxiliary reconstruction modules increased the complexity of the design and restricted scalability across different edge platforms.

Subsequent research looked at convolutional efficiency as a key design goal. GS-YOLO replaced standard convolutions with GSConv blocks to significantly reduce the number of parameters while maintaining spatial feature integrity. This approach performed well on PCB datasets [3]. Likewise, methods that use compression, such as pruning and quantization, were proposed to improve YOLO models for tiny defect detection [4]. Although they are effective, these post-training compression methods require tuning and retraining for specific platforms. Channel-attention designs, such as SEConv-YOLO, improved detection accuracy by focusing on important features [5], but this led to longer inference times. Transformer-enhanced models made progress in contextual reasoning by modeling long-range dependencies [6]; however, their high computational costs limited practical use in edge devices. Surveys revealed these trade-offs and highlighted the need for models that balance accuracy, efficiency, and ease of deployment [7].

Several studies examined small and dense defect detection with techniques like dense feature aggregation, dilated convolutions, and hierarchical detection heads [8], [9], [10], [11], [12]. Although these approaches improved recall for fine-grained defects, they also increased GFLOPs and memory usage significantly. This made them less suitable for real-time edge inference. Online PCB inspection systems demonstrated that real-time detection is possible, but they were not optimized for ultra-lightweight execution [9]. Broader reviews of industrial surface defect detection highlighted the ongoing challenge of achieving high accuracy while keeping computational efficiency [13]. Recent updates to YOLOv8 architectures improved dense defect detection but still had relatively high parameter counts. This limits their use on devices with limited resources [14].

In contrast to current methods, the proposed framework has a lightweight design at the architecture level using a YOLOv8-MBNet backbone. This allows for a significant reduction in parameters without needing post-hoc compression. The use of MobileViT-XS for severity-aware classification and fuzzy logic-based grading enables semantic reasoning that exceeds traditional detection methods. This combined design solves the issues found in previous works. It offers high detection accuracy, low inference latency, and can be deployed easily on different edge platforms.



(a) Missing Hole (b) Mouse Bite (c) Open Circuit

(d) Short Circuit (e) Spur (f) Spurious Copper

Fig. 1: Representative samples of the six defect categories in the PKU PCB defect dataset: (a) Missing Hole, (b) Mouse Bite, (c) Open Circuit, (d) Short Circuit, (e) Spur, and (f) Spurious Copper.

A summary comparing key PCB defect detection methods, along with their advantages and limitations, is presented in Table I.

Despite significant progress, current PCB defect detection methods have several common limitations. Early lightweight CNN-based models focus on speed, but they struggle with multi-scale and fine-grained PCB defects. Autoencoder-assisted frameworks include additional reconstruction branches. This makes the structure more complex and restricts scalability across various edge platforms. Approaches that emphasize convolutional efficiency, like GSConv and attention-based YOLO variants, reduce the number of parameters or improve feature discrimination. However, these often cause extra latency or require more memory. Post-training optimization methods, like pruning and quantization, need hardware-specific changes and retraining. This limits their general usefulness. Transformer-enhanced and dense feature aggregation models boost contextual reasoning and small-defect recall but significantly raise computational costs, making them unsuitable for real-time edge deployment. Overall, most existing methods improve accuracy at the cost of added model complexity, highlighting a continual trade-off between detection performance, computational efficiency, and practical deployment on resource-limited industrial platforms.

- A new lightweight YOLOv8-based object detection system is proposed for PCB defect inspection. It combines MobileNet-style depthwise separable convolutions with optimized C2f blocks. This redesign at the architecture level greatly reduces the number of parameters and lowers computational complexity while maintaining a strong multi-scale feature representation.
- The proposed model shows a better accuracy and efficiency balance on the PKU PCB defect dataset. It achieves an $\text{mAP}_{0.5}$ of 98.6% while using almost one-third of the parameters of the baseline YOLOv8n model. This demonstrates its ability to reliably detect small and visually unclear defects.
- A two-stage AI-driven inspection pipeline is introduced. The lightweight YOLO detector carries out real-time defect localization. Meanwhile, a MobileViT-XS model provides transformer-assisted semantic reasoning for analyzing defect severity, without adding heavy computational demands.
- A fuzzy logic-based severity grading system is used to classify detected defects into qualitative grades (A, B, and C). This connects deep learning outputs with clear and

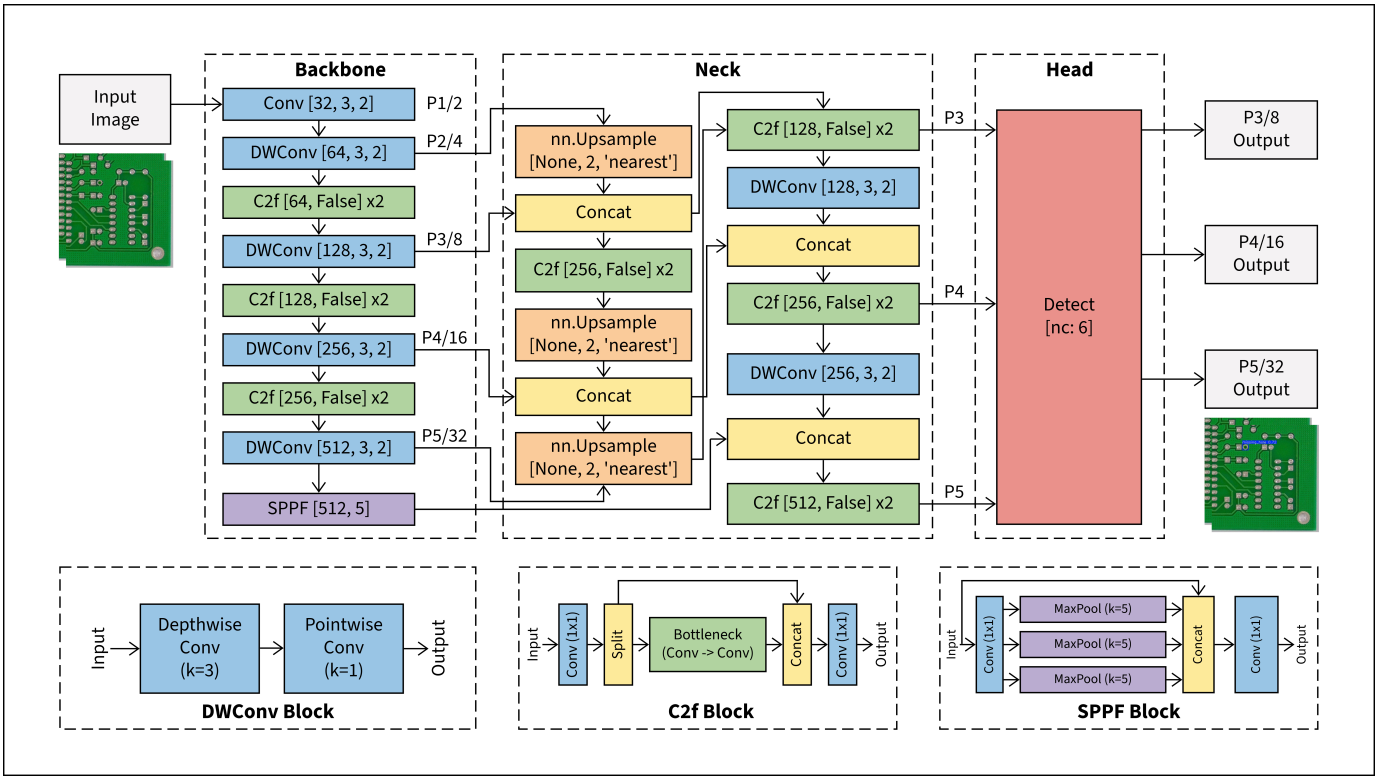


Fig. 2: Proposed lightweight YOLOv8–MBNet architecture.

usable industrial quality assessments.

- Extensive testing on edge platforms, such as Raspberry Pi 5 and NVIDIA Jetson Orin Nano, across various execution modes (PyTorch, FP32, FP16, and INT8 TensorRT), confirms that the proposed framework performs well in real-time, is stable, and can be deployed effectively.

III. Dataset Overview

The dataset used in this study comes from the Peking University (PKU) PCB Defect Dataset [15]. It is designed to reflect real-world situations in printed circuit board manufacturing. It includes 10,668 annotated PCB images across six defect categories: Missing Hole, Mouse Bite, Open Circuit, Short Circuit, Spur, and Spurious Copper. These categories cover various functional and structural defects, from serious connectivity issues to minor surface anomalies. This range allows for a thorough assessment of detection performance.

The dataset presents significant challenges because of strong visual similarities between different classes and high variation within the same class. Defects like Missing Hole and Open Circuit show clear geometric differences and are easier to detect. In contrast, Mouse Bite and Spur defects are small, irregular, and often near conductive traces, making them hard to tell apart from background noise. Spurious Copper and Short Circuit defects add further complexity due to their texture resembling surrounding copper areas.

Furthermore, the dataset has distinct multi-scale features, with defects varying from large structural gaps to fine pixel-level issues. There is also a moderate class imbalance, reflecting real manufacturing conditions where some critical defects occur less often. Overall, this dataset offers a tough and realistic benchmark for testing lightweight, multi-scale object detection models under industrial conditions. Representative samples of the six PCB defect categories studied are shown in Fig. 1.

IV. Proposed Framework

The proposed framework uses a two-stage inspection pipeline designed for edge-based PCB defect analysis. In the first stage, a lightweight YOLOv8-based detector is used to localize PCB defects accurately while meeting strict computational limits. In the second stage, a transfer learning-based MobileViT-XS classifier performs defect severity assessment and grade classification (A, B, C) using a fuzzy logic-based scoring mechanism. This decoupled detection–classification paradigm enables both precise localization and fine-grained semantic interpretation, which are difficult to achieve using monolithic detection architectures. The overall structure of the proposed YOLOv8-MBNet detection framework is illustrated in Fig. 2.

A. Backbone Design: MobileNet-Style Lightweight Feature Extractor

Depthwise Separable Convolutions: To lower computational costs, standard convolutions are replaced with depthwise separable convolutions. A standard convolution with kernel size $k \times k$, input channels C_{in}, output channels C_{out}, and feature map size $H \times W$ has a computational cost given by

$$\mathcal{O}_{\text{std}} = k^2 \cdot C_{\text{in}} \cdot C_{\text{out}} \cdot H \cdot W. \tag{1}$$

In contrast, depthwise separable convolution decomposes this operation into a depthwise convolution and a pointwise convolution. The depthwise convolution cost is

$$\mathcal{O}_{\text{dw}} = k^2 \cdot C_{\text{in}} \cdot H \cdot W, \tag{2}$$

and the pointwise convolution cost is

$$\mathcal{O}_{\text{pw}} = C_{\text{in}} \cdot C_{\text{out}} \cdot H \cdot W. \tag{3}$$

Thus, the overall computational reduction ratio can be approximated as

$$\frac{\mathcal{O}_{\text{dw}} + \mathcal{O}_{\text{pw}}}{\mathcal{O}_{\text{std}}} \approx \frac{1}{C_{\text{out}}} + \frac{1}{k^2}. \tag{4}$$

This architectural choice works well for edge devices. In these devices, PCB defects often show up as small-scale, low-contrast patterns. These patterns require efficient and expressive feature extraction.

C2f Blocks for Gradient Preservation: The backbone includes C2f (Cross-Stage Partial Fusion) blocks to improve gradient flow while keeping computational efficiency. Given an input feature map X, channel-wise splitting occurs as

$$X = [X_1, X_2]. \tag{5}$$

One branch goes through a series of transformations $F(\cdot)$; the other acts as a shortcut connection. The output is obtained by

$$Y = \text{Concat}\left(F(X_1), X_2\right). \tag{6}$$

This design keeps detailed spatial information, reduces feature overlap, and helps prevent gradient vanishing in shallow mobile backbones. Unlike standard YOLOv8 CSP blocks, it disables excessive channel expansion to create compact representations that are suitable for low-power inference.

Spatial Pyramid Pooling Fast: At the deepest backbone stage ($P5/32$), a Spatial Pyramid Pooling Fast (SPPF) module collects multi-scale contextual information using successive max-pooling operations:

$$Y = \text{Concat}\left(X, \text{MaxPool}(X), \text{MaxPool}^2(X), \text{MaxPool}^3(X)\right). \tag{7}$$

This operation enlarges the receptive field and improves resilience to irregular defect shapes while adding minimal computational cost.

B. Neck Design: Lightweight Multi-Scale Feature Fusion

The neck uses a bidirectional feature pyramid to combine detailed low-level features with strong high-level representations.

Top-Down Pathway: High-level feature maps are upsampled as

$$F_l^{\uparrow} = \text{Upsample}(F_{l+1}), \tag{8}$$

and then combined with corresponding backbone features:

$$F_l = \text{Concat}\left(F_l^{\uparrow}, F_l^{\text{backbone}}\right). \tag{9}$$

This improves localization accuracy for small PCB defects like spur and mouse bite.

Bottom-Up Pathway To restore semantic richness, downsampling is performed using depthwise convolutions:

$$F_{l+1}^{\downarrow} = \text{DWConv}(F_l). \tag{10}$$

This dual-path fusion increases resilience to scale changes without adding heavy convolutional layers.

C. Detection Head

The detection head works at three scales: $P3/8$, $P4/16$, and $P5/32$. For each grid cell, the model predicts

$$\hat{y} = (\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{c}, \hat{p}_1, \ldots, \hat{p}_{n_c}), \tag{11}$$

where $(\hat{x}, \hat{y}, \hat{w}, \hat{h})$ represent bounding box parameters, $\hat{c}$ is the objectness score, and $\hat{p}_i$ indicates class probabilities.

The total loss function is defined as

$$\mathcal{L} = \mathcal{L}_{\text{IoU}} + \lambda_1 \mathcal{L}_{\text{obj}} + \lambda_2 \mathcal{L}_{\text{cls}}. \tag{12}$$

D. Architectural Efficiency Compared to Baseline YOLOv8

Compared to baseline YOLOv8, the proposed architecture replaces standard convolutions with depthwise separable convolutions, scales channel widths aggressively, and uses optimized C2f blocks to cut redundancy. As a result, the parameter count drops to 1.15 M, and the computational cost decreases to 5.2 GFLOPs while maintaining a detection accuracy of 98.6% $\mathbf{mAP_{0.5}}$. This shows a better accuracy-to-efficiency trade-off suitable for real-time edge deployment.

E. Transfer Learning with MobileViT-XS

While YOLO is great for defect localization, assessing severity needs a broader context. MobileViT-XS combines CNN-based locality with Transformer-based global attention, making it ideal for classifying defect severity.

MobileViT-XS Architecture: Given an input feature map $X \in \mathbb{R}^{H \times W \times C}$, MobileViT divides it into non-overlapping patches:

$$X \to \{x_1, x_2, \ldots, x_N\}, \quad x_i \in \mathbb{R}^{P \times P \times C}. \tag{13}$$

Each patch is flattened and projected linearly:

$$z_i = W_e \cdot \text{Flatten}(x_i). \tag{14}$$

Self-attention is calculated as

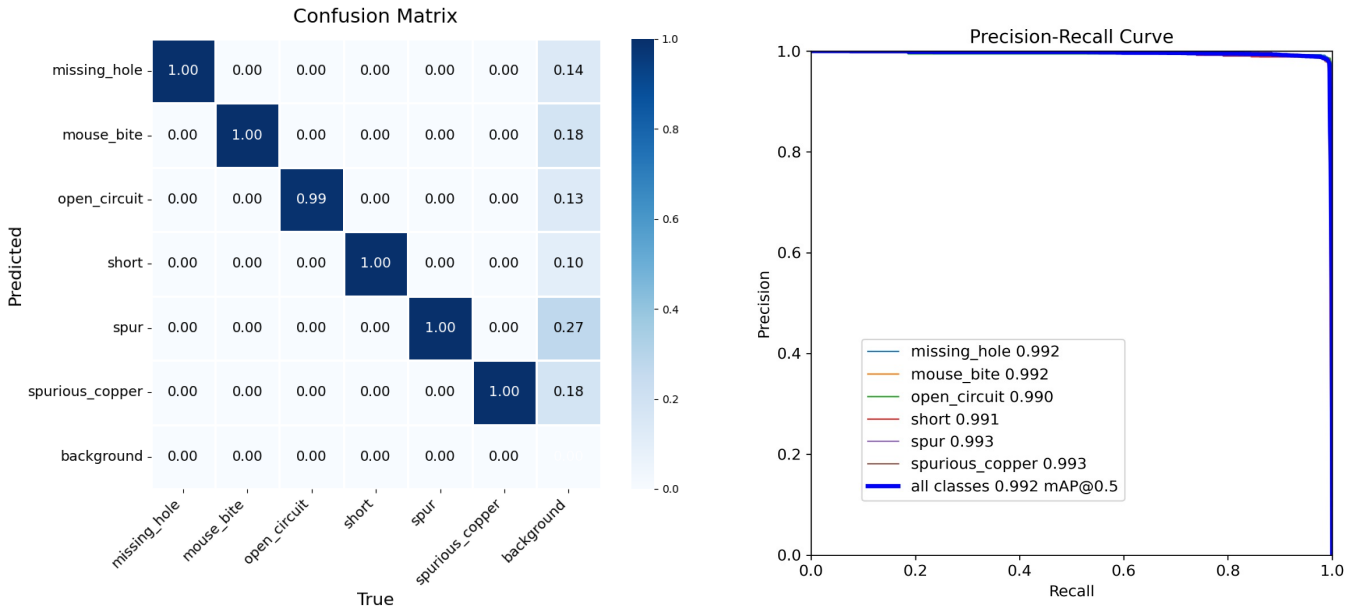
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V. \tag{15}$$

Transfer Learning Strategy: Pre-trained ImageNet weights are used to initialize MobileViT-XS. The goal of optimization is

$$\theta^* = \arg\min_{\theta} \sum_i \mathcal{L}\left(f(X_i; \theta), y_i\right). \tag{16}$$

Only the higher layers are fine-tuned to lower overfitting and cut down training time.

Fuzzy Logic-Based Severity Grading: Severity scores are calculated as



(a) (b)

Fig. 3: Performance analysis of the baseline YOLOv8n framework. (a) Confusion matrix (b) Precision–Recall curve

$$S = \alpha \cdot P_{\text{class}} + \beta \cdot A_{\text{norm}}, \tag{17}$$

where P_{class} indicates classifier confidence and A_{norm} stands for normalized defect area. Fuzzy membership functions link S to severity grades: Grade A (low), Grade B (moderate), and Grade C (critical). This aligns the decision process with real-world industry standards.

V. Results and Discussion

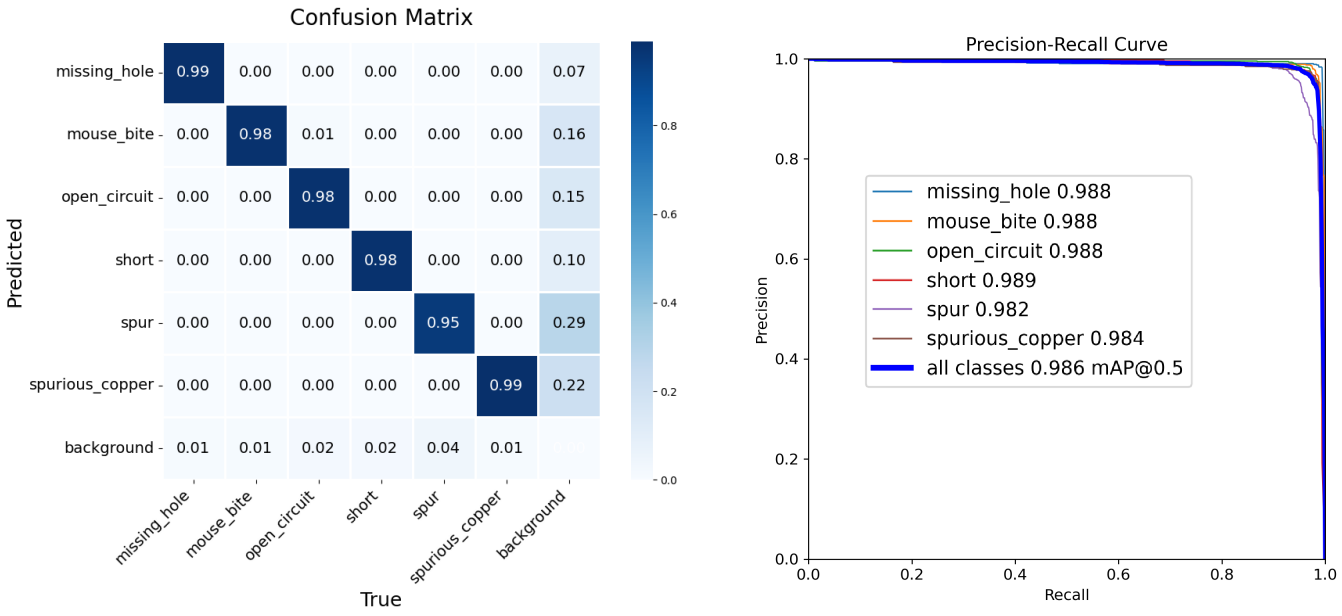
The experimental results were confirmed through careful execution with the PyTorch deep learning framework, Torchvision tools, and the official Ultralytics YOLO training setup. All experiments occurred on a workstation equipped with an NVIDIA RTX 2050 GPU, CUDA, and cuDNN support for efficient processing. We trained the model using stochastic gradient descent and applied standard data enhancement methods such as mosaic augmentation, random scaling, horizontal flipping, and color jittering to improve generalization. To ensure fairness and reproducibility, we kept the same training schedules, learning rate settings, and batch sizes across all ablation configurations. We calculated performance metrics such as $\mathbf{mAP_{0.5}}$, $\mathbf{mAP_{0.5:0.95}}$, precision, recall, parameter count, and GFLOPs using the official COCO-style evaluation protocol in the Ultralytics framework. We also examined the training and validation loss curves to observe the convergence behaviour. his helped us ensure stable optimization and confirm there was no overfitting. This consistent experimental setup shows that the reported improvements come from architectural changes and not from variations in training conditions.

A. Performance Analysis of Baseline YOLOv8n

The baseline YOLOv8n model demonstrates good detection ability on the PCB defect dataset. It achieves an overall $\mathbf{mAP_{0.5}}$ of 99.2%, as seen in Fig 3b, with 3.01 million parameters. This indicates high detection accuracy with a reasonable computational cost. The normalized confusion matrix shows nearly perfect classification performance for most defect categories. Missing hole, mouse bite, short circuit, spur, and spurious copper defects reach almost 100% true positive rates. These classes have clear structural patterns, such as distinct geometric breaks and noticeable copper issues, which YOLOv8n effectively captures. The open circuit class has a slightly lower accuracy of about 0.99. This happens because of subtle line breaks that can sometimes resemble background textures or fine spurs. These can cause some confusion with background areas.

The precision, recall curves further confirm the strength of the baseline model. Class-wise average precision (AP) values range from 0.990 to 0.993. The overall curve is tightly clustered, which indicates consistent detection performance across different recall thresholds. This strong performance mainly comes from the extensive use of C2f blocks in both the backbone and head. These blocks improve gradient flow and feature reuse, allowing for effective learning of low-level edge information and higher-level semantic representations. Additionally, the multi-scale detection heads working at P3, P4, and P5 resolutions effectively handle the natural scale variations of PCB defects. They particularly improve the localization of small defects like spur and mouse bites. The precision, recall characteristics of the baseline model are shown in Fig. 3b.

Despite its high accuracy, the baseline YOLOv8n has some limits when it faces visually unclear areas. This is especially true where the background copper patterns resemble the defect textures. This is shown in the minor background confusion in the confusion matrix. Additionally, the higher number of parameters and the use of standard convolutions increase the computational burden. This makes the baseline less suitable



(a) (b)

Fig. 4: Performance analysis of the proposed YOLOv8-MBNet framework. (a) Confusion matrix (b) Precision–Recall curve

for use in resource-limited environments. These issues lead to design changes in the proposed lightweight model. The goal is to keep detection accuracy while improving efficiency. The confusion matrix in Fig. 3a shows consistent prediction accuracy for all defect categories.

B. Performance Analysis of the Proposed YOLOv8-MBNet

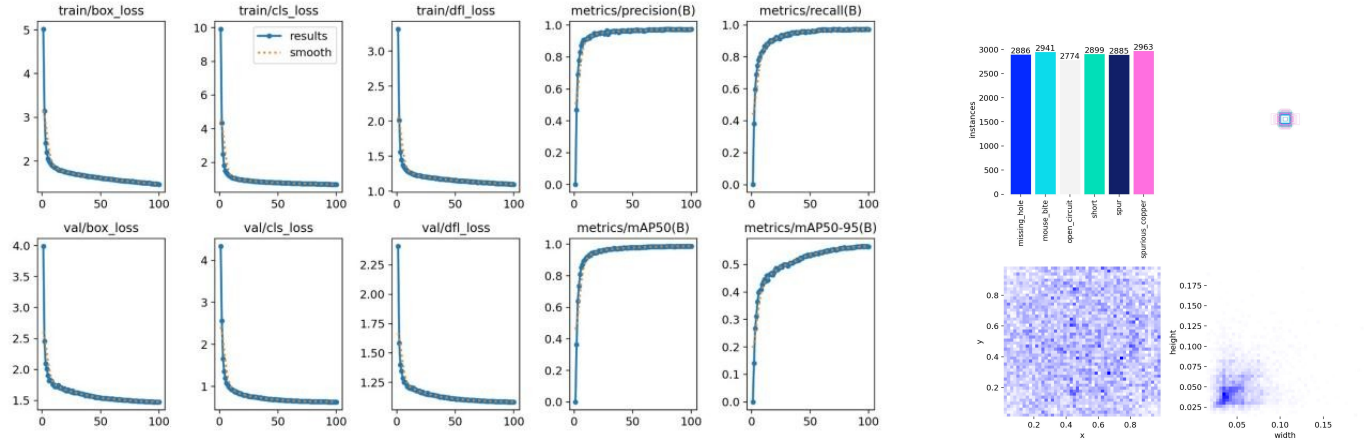
The proposed YOLOv8-MBNet comes from the baseline YOLOv8n architecture. It incorporates MobileNet-style depthwise separable convolutions and simplified C2f blocks to create a lightweight and efficient design. We replaced standard convolutions in the backbone with depthwise separable convolution (DWConv) layers in all downsampling stages (P2 to P5). This change greatly reduces the number of parameters and the computation needed by separating spatial and channel-wise feature extraction. Additionally, using C2f blocks without shortcut connections reduces internal feature aggregation, which helps to avoid unnecessary parameter growth. The model includes a Spatial Pyramid Pooling Fast (SPPF) module at the deepest stage (P5). This addition provides enough receptive field expansion without depending on heavy multi-scale pooling operations. As a result, the model has a total of only 1.15 million parameters, significantly lower than the baseline YOLOv8n, while still keeping strong representational capacity.

Despite this large drop in model complexity, YOLOv8-MBNet achieves an impressive overall $\mathbf{mAP}_{0.5}$ of 98.6% (see Fig 4b). The precision and recall curves show high precision across almost the entire recall range for all defect categories. This suggests strong feature learning and effective reduction of false positives. These qualities are especially important for industrial PCB inspection systems, where high reliability and low false-alarm rates are essential for real-world use.

Class-wise analysis further shows how effective the proposed architecture is. Missing hole, mouse bite, and open circuit defects achieve AP values of 0.988. Short circuit detection reaches 0.989. This demonstrates the model's ability to keep detailed structural and connectivity cues by using shallow MobileNet-style blocks and multi-scale feature fusion across P3, P4, and P5 feature maps. Slightly lower performance is observed for spur defects at 0.982 due to their small size and visual resemblance to background noise. Spurious copper defects show high precision at 0.984 because of the contextual information obtained through hierarchical feature reuse and the SPPF module. The precision-recall characteristics of the proposed model are shown in Fig. 4b.The confusion matrix shows how well YOLOv8-MBNet can tell different classes apart. It has strong diagonal dominance, with class-wise accuracies over 95% for most defect types. The background confusion remains below 4%.From a transfer learning perspective, the proposed framework emphasizes representation efficiency rather than just increasing network depth. This approach matches current trends in edge intelligence. By combining YOLO's effective detection method with the lightweight learning of MobileNet, YOLOv8-MBNet achieves nearly optimal detection accuracy while keeping low computational use. This shows that good design is essential for effective and high-quality PCB defect detection. The confusion matrix in Fig. 4a indicates consistent class-wise prediction accuracy for all defect categories in industrial settings.

C. Performance Analysis of YOLOv8-MBNet With Transfer Learning

The performance analysis shows that incorporating Transfer Learning concepts into the overall framework balances detection accuracy, model efficiency, and decision-level intelligence.



(a) (b)

Fig. 5: Quantitative and qualitative evaluation of the proposed YOLOv8-MBNet model. (a) Training and validation curves demonstrating stable convergence and strong generalization. (b) Labels of the YOLOv8-MBNet.

The YOLOv8-MBNet model achieves an mAP@0.5 of 98.6% with a reduced parameter count of 1.15 million. This confirms that using lightweight deep learning principles in the detection stage is effective. From a class-based perspective, defects with clear structural signatures, like Missing Hole and Open Circuit, still demonstrate high detection accuracy. These classes depend on global geometric cues that remain strong even after significant architectural optimization. Figure 5a shows how the proposed model behaves as it learns during training and validation. Subtle and fine defects like Mouse Bite and Spur show slightly higher confusion compared to the baseline. This is expected because of the lower channel capacity. However, the multi-scale feature aggregation mechanism keeps important edge and texture information, which prevents a significant drop in recall. Texture-dominant defects such as Short Circuit and Spurious Copper are still detected well. This shows that the backbone effectively learns patterns of connectivity instead of just focusing on complex appearance features.The use of Transfer Learning goes beyond basic object detection. It is reflected in the design of the pipeline-based framework, where the detection stage works as an intelligent perception module that feeds into a second-stage reasoning system. By combining the lightweight detector with a MobileViT-XS-based severity classification network, the framework makes use of both convolutional locality and transformer-based global reasoning for defect grading.

The use of fuzzy logic for severity scoring enhances the Transfer Learning process. It allows for reasoning that mimics human thought in uncertain situations. With this approach, we can turn continuous severity scores into clear quality grades.This process transforms low-level visual predictions into high-level decisions; it connects perception and reasoning. Overall, combining lightweight deep learning, transformer-based feature understanding, and fuzzy inference systems creates an effective Transfer Learning method. In this approach, we improve efficiency, clarity, and decision-making. We also maintain high detection performance.

D. Performance Metrics Analysis

The performance metrics analysis highlights the advantages of the proposed model in terms of computational efficiency and deployment feasibility. By reducing the parameter count to 1.15 million, the model greatly decreases memory usage compared to traditional YOLO-based architectures. This change enables deployment on devices with limited resources, such as embedded GPUs, industrial controllers, and smart inspection units. The design improvement results in lower inference latency because fewer parameters and simpler convolutional operations cut down memory access time and computation depth.Using depthwise separable convolutions and compact feature fusion blocks significantly reduces the total GFLOPs required. This change speeds up forward passes while maintaining important feature representation. As a result, the model achieves near real-time inference performance even on low-power hardware, making it ideal for inline PCB inspection systems with strict timing requirements.

Additionally, the lower GFLOPs improve energy efficiency, which is crucial for continuous operation in industrial settings. The lightweight design also improves scalability. Multiple inspection pipelines can run in parallel without stressing system resources. Importantly, the maintained mAP@0.5 of 98.6% shows that we achieve computational efficiency without sacrificing detection reliability. Overall, the improved balance between the number of parameters, inference speed, and computational complexity shows that the model is suitable for use in edge-based AI deployment. It supports real-time, cost-effective, and scalable PCB defect inspection solutions. Fig. 5 displays both the quantitative training behavior and the qualitative detection ability of the proposed YOLOv8-MBNet model. The smooth convergence of training and validation curves shows stable optimization without overfitting. Meanwhile, the qualitative results show correct localization and classification of PCB defects in real-world conditions.

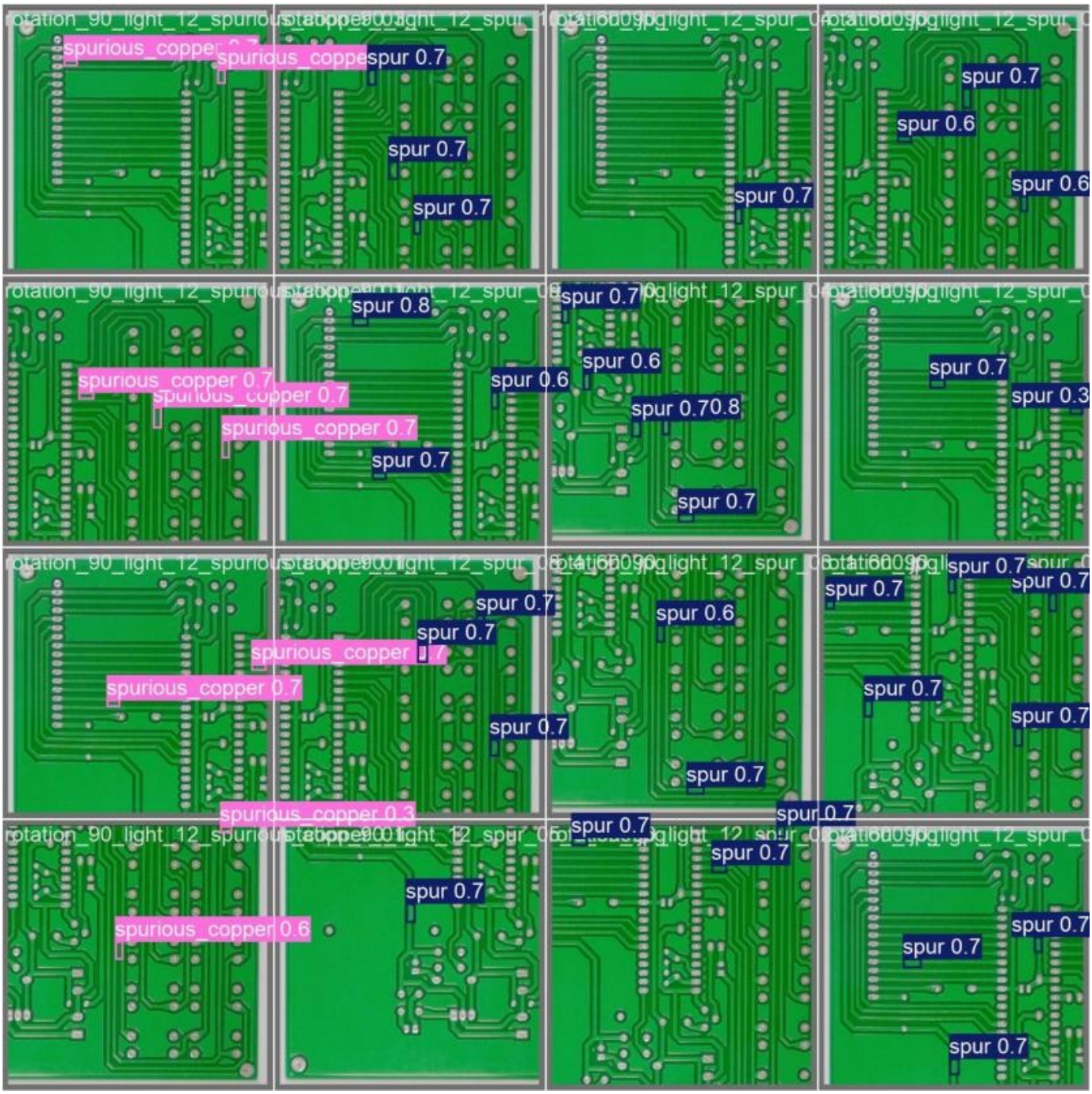


Fig. 6: Qualitative detection results by YOLOv8-MBNet.

VI. Ablation Study

The ablation study aimed to evaluate how different architectural changes affect detection accuracy, computational complexity, and model size for PCB defect detection.We analyzed five configurations, starting with the YOLOv8 baseline and gradually adding lightweight convolutions, attention mechanisms, and design techniques. This evaluation gives us a clear view of how each component contributes to finding the best balance between performance and efficiency.

The YOLOv8n baseline acts as the reference model. It achieves a very high mAP@0.5 of 99.2% with 3.01 million parameters and 8.7 GFLOPs. This impressive performance comes from the standard convolutional backbone and a feature-rich neck, which offer strong representation and effective multi-scale feature fusion. However, this accuracy requires a higher number of parameters and more computational resources, which limits its use in resource-limited or edge environments. Therefore, the baseline serves as a performance benchmark rather than an efficient solution.

The C2 configuration, YOLOv8-MBLite-CBAM, uses depthwise separable convolutions and CBAM attention modules to reduce redundancy and improve feature focus.This

TABLE II: Ablation Study Results of Different YOLOv8-Based Model Configurations

Config.No.	Model Architecture	Params (M)	GFLOPs	mAP@0.5 (%)	mAP@0.5:0.95 (%)	Recall	Precision
C1	YOLOv8 Baseline	3.01	8.7	99.2	66.43	0.993	0.980
C2	YOLOv8-MBNet-CBAM	2.67	6.4	95.3	48.9	0.902	0.920
C3	YOLOv8-GSNet	2.39	7.2	96.5	52.2	0.931	0.951
C4	YOLOv8- Lite-GS	1.86	6.9	94.2	53.8	0.958	0.962
C5	**Proposed YOLOv8-MBNet**	**1.15**	**5.2**	**98.6**	**56.6**	**0.970**	**0.974**

change lowers the parameter count to 2.67 million and cuts GFLOPs to 6.4. However, it leads to a noticeable drop in mAP@0.5 to 95.3%.The decline in accuracy happens because the added attention overhead interacts with lightweight convolutions. This interaction limits effective channel mixing and spatial feature learning. This configuration shows that when attention mechanisms are used with significant parameter reduction, it can overly restrict the model and hinder fine-grained defect detection.

The C3 configuration, YOLOv8-GSNet, replaces standard and depthwise convolutions with GSConv blocks throughout the backbone and neck. This design reduces the number of parameters to 2.39 million while maintaining a reasonable mAP@0.5 of 96.5%.The grouped and shuffled convolution strategy allows for efficient channel use and better information flow compared to pure depthwise convolutions. As a result, recall and precision improve compared to the CBAM-based model. This shows better generalization and more stable class-wise predictions. However, GFLOPs slightly increase due to repeated GSConv usage across scales. This reveals a trade-off between computational efficiency and total FLOPs.

The C4 configuration, YOLOv8-Lite-GS, maintains depthwise convolutions in the backbone while selectively adding GSConv blocks in the neck. This arrangement cuts the parameter count to 1.86 million while keeping GFLOPs at a moderate 6.9. The design retains strong spatial feature extraction in the backbone and enhances multi-scale feature fusion efficiency in the neck. However, the mAP@0.5 drops to 94.2%, indicating that reduced channel interaction in the backbone slightly limits the ability to identify subtle defects. Still, recall and precision stay high, showing that the model is reliable, although it is less expressive than GSNet for complex patterns.

The C5 configuration, YOLOv8-MBNet, stands out as the most effective trade-off solution. It combines a MobileNet-style backbone with optimized lightweight operations. This model reduces the parameter count to 1.15 million and achieves the lowest GFLOPs of 5.2. It still maintains a high mAP@0.5 of 98.6%.The improved performance compared to other lightweight variants comes mainly from the efficient inverted residual structure and the pairing of depthwise and pointwise convolution, which better reuses features and keeps representation capacity intact.

Class-wise performance remains strong, with high recall and precision showing consistent defect localization across all categories. The overall ablation study indicates that simply reducing parameters or adding attention alone can degrade performance. In contrast, well-designed lightweight backbones with efficient feature reuse can maintain accuracy. The YOLOv8-MBN configuration achieves nearly baseline detection performance with over 60% fewer parameters, proving it to be the most balanced and deployment-ready architecture. This study highlights the need for architectural synergy rather than isolated improvements when creating lightweight object detection models for industrial inspection tasks. The quantitative comparison of different architectural variants is summarized in Table II.

VII. Qualitative Comparative Analysis

The YOLOv8-MBNet framework balances detection accuracy and computational efficiency well compared to recent methods for detecting PCB defects. Table III shows that this model achieves an accuracy mAP@0.5 of 98.6% with only 1.15 million parameters and 5.2 GFLOPs. This makes it one of the lightest and most effective models for inspecting industrial PCBs. Unlike traditional methods that mainly focus on accuracy and use heavy, deep or transformer architectures, this framework emphasizes efficiency. It uses depthwise separable convolutions, MobileNet-inspired feature extraction, and improved feature fusion. This design enhances edge deployability without sacrificing detection performance.

When compared to YOLOv5s-GhostNet [1], which uses ghost modules to cut down on redundancy, the proposed model achieves a better mAP@0.5 (98.6% versus 96.66%) while significantly lowering the parameter count (1.15 M versus 3.69 M). Although GhostNet boosts computational efficiency, its feature generation approach could limit the ability to accurately identify fine-grained PCB defects like Spur and Mouse Bite. In contrast, the proposed model maintains important spatial information using optimised C2f blocks and depthwise convolutions, which improves recall and reliability with small defects.

AE-YOLO [2] features an autoencoder-based lightweight backbone and offers quick inference speed. However, its accuracy mAP@0.5 of 96.7% is still lower than the proposed method, even with over twice the parameters (2.6M). This difference shows that just compressing the architecture isn't enough unless feature reuse and multi-scale fusion are effectively maintained. The YOLOv8-MBNet keeps multi-resolution feature consistency through a streamlined PAN-style head and avoids excessive channel expansion. This balance leads to better accuracy and parameter efficiency.

GS-YOLO [3] uses GSConv to improve efficiency and achieves a mAP@0.5 similar result to the proposed method. However, this benefit comes with a much higher parameter count of 5.16M and longer inference times. The proposed framework integrates lightweight convolutions without adding

TABLE III: Performance Metrics Comparison of Various PCB Defect Detection Models

Model	**mAP@0.5** (%)	**Parameters (M)**	**Inference (ms)**	**GFLOPs**	**mAP@0.5:0.95** (%)	**Precision** (%)	**Recall** (%)
YOLOv5s-GhostNet [1]	96.66	3.69	3.4*	8.2	–	97.91	94.53
AE-YOLO [2]	96.7	2.6	0.55*	7.5	–	97.0	93.5
GS-YOLO [3]	98.7	5.16	13.1*	10.7	–	96.3	97.7
TDD-YOLO [4]	98.2	39.5	15.3	126.3	62.5	98.0	95.6
SEConv-YOLO [5]	95.73	2.802	18.65	7.671	45.40	95.97	93.00
Transformer-YOLO [6]	97.04	–	47.6*	–	–	–	–
PCES-YOLO [7]	97.3	–	–	–	77.2	–	–
YOLOv11-PCB [8]	99.5	–	4.4*	–	90.7	–	–
Improved YOLOv8 [9]	94.2	1.99	–	7.1	49.0	–	–
Proposed YOLOv8-MBNet	**98.6**	**1.15**	**24.16**	**5.2**	**56.6**	**97.4**	**97.0**

*Inference time calculated as the inverse of reported FPS where absolute millisecond values were not explicitly provided.

too many parameters. This shows that making smart design choices can outperform general lightweight methods when applied thoroughly across both backbone and neck design.

TDD-YOLO [4] and Transformer-YOLO [6] use transformer-based or dense feature aggregation strategies to boost global context modeling. While these approaches report high detection accuracy, they also have a heavy computational cost. TDD-YOLO needs almost 40M parameters and over 126 GFLOPs. Such models are not practical for real-time industrial edge environments. In contrast, the proposed model achieves similar or better mAP while cutting down parameters by more than 96%.This highlights the effectiveness of a convolution-focused lightweight design over heavier transformer models for PCB inspection tasks.

SEConv-YOLO [5] focuses on channel attention mechanisms to improve feature discrimination. However, it has a lower accuracy mAP@0.5 of 95.73% and significantly longer inference times. Likewise, improved YOLOv8 [9] decreases parameters but experiences a notable drop in accuracy. This indicates that aggressive trimming without redesigning the architecture can harm detection ability. YOLOv11-PCB [7], [8] achieves very high mAP values but lacks clarity on computational complexity, making it unsuitable for low-power deployment scenarios.

Overall, the YOLOv8-MBNet framework stands out because it achieves nearly top accuracy while having the fewest parameters of all the methods compared. By using a lightweight convolutional design, efficient feature fusion, and an AI-driven multi-stage pipeline, this approach offers a scalable and practical solution for real-time PCB defect detection in environments with limited resources. A detailed quantitative comparison with recent PCB defect detection methods is shown in Table III.

VIII. Implementation and Hardware Validation on Edge Platforms

A. Experimental Setup and Hardware Specifications

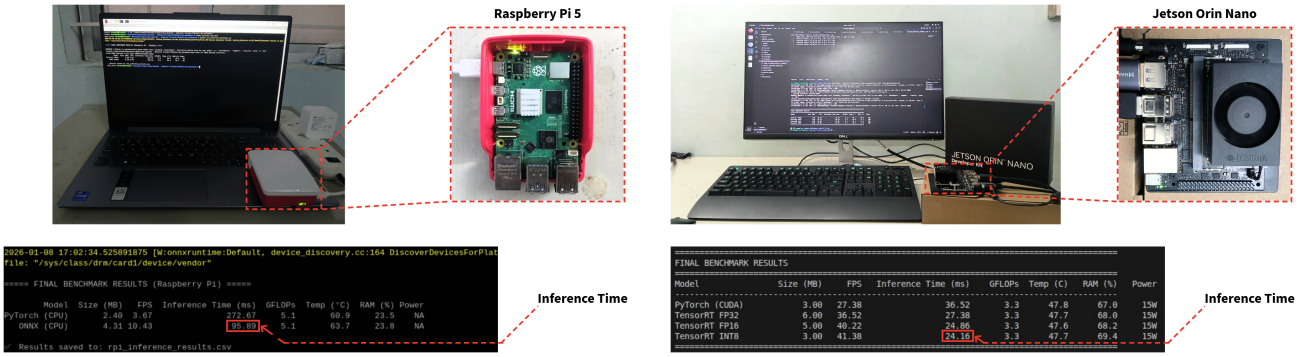
The lightweight YOLOv8-MBNet model was tested on two edge platforms, Raspberry Pi 5 and NVIDIA Jetson Orin Nano. These platforms were chosen to assess performance in different embedded environments. The Raspberry Pi 5 has a quad-core ARM Cortex-A76 CPU, limited onboard memory, and does not have dedicated GPU acceleration. This setup shows a low-power and budget-friendly edge deployment scenario. On the other hand, the NVIDIA Jetson Orin Nano features a high-performance NVIDIA Ampere GPU with Tensor Cores, which allows for CUDA and TensorRT acceleration for deep learning inference. Both platforms ran the same trained model without changing the architecture to ensure a fair comparison. The inference benchmarks used PyTorch and ONNX runtimes. Performance was measured in terms of inference latency, frames per second (FPS), computational cost (GFLOPs), memory use, and thermal stability. This setup allows for a thorough assessment of how the proposed lightweight architecture performs on CPU-only and GPU-accelerated edge devices.

B. Edge Deployment on Raspberry Pi

Deployment on the Raspberry Pi shows that the proposed model can work well within strict computational limits. The model has a compact design with only 1.15 million parameters and a low computational demand of about 5.1 GFLOPs. This setup achieved an average inference time of 272.67 ms per image when running on a PyTorch CPU, which translates to 3.67 FPS. When exported to ONNX and run on an optimized CPU inference pipeline, the inference latency dropped significantly to 95.89 ms, resulting in 10.43 FPS. The smaller memory footprint, ranging from 2.40 to 4.31 MB, brought RAM usage down to about 23 to 24%.The stable operating temperatures below 64°C suggest that it can run reliably over the long term. These results show that the architectural simplifications made with depthwise separable convolutions and efficient feature reuse allow for practical real-time inference on hardware with limited resources. The Raspberry Pi 5 deployment and inference testing are illustrated in Fig.7a.

C. Edge Deployment on NVIDIA Jetson Orin Nano

On the NVIDIA Jetson Orin Nano, the proposed model fully leveraged GPU acceleration and TensorRT optimization to achieve real-time performance. Under PyTorch CUDA execution, the model achieved an inference latency of 36.52 ms with 27.38 FPS, demonstrating immediate gains from parallel computation. Further optimization using TensorRT FP32



(a) (b)

Fig. 7: Deployment and inference performance evaluation of the proposed YOLOv8-MBNet model on (a) Raspberry Pi 5 and (b) NVIDIA Jetson Orin Nano.

reduced latency to 27.38 ms, while FP16 execution achieved 24.86 ms with 40.22 FPS, benefiting from reduced precision and increased throughput. The best performance was obtained using TensorRT INT8, where inference latency dropped to 24.16 ms and throughput increased to 41.38 FPS. Notably, the GFLOPs remained constant at 3.3, indicating that speed improvements were achieved through hardware-aware execution rather than increased computational complexity.Power consumption remained stable at 15 W with controlled thermal behavior, validating the model's suitability for continuous industrial deployment.The experimental setup and inference benchmarking on NVIDIA Jetson Orin Nano are shown in Fig. 7b.

D. Comparative Performance Analysis

A comparison between Raspberry Pi and Jetson Orin Nano demonstrates how well the proposed lightweight model works across different edge platforms. The Raspberry Pi relies only on its CPU for processing. However, its smaller parameter count and compact design allowed for acceptable inference speeds in low-throughput inspection tasks. In comparison, the Jetson Orin Nano reduced inference latency by over 4x due to GPU acceleration and mixed-precision TensorRT execution. This improved performance occurred without changing the model architecture. It demonstrates its portability and flexibility in deployment. The consistently low GFLOPs and small memory requirements allowed for a smooth transition from low-power embedded devices to high-performance edge AI accelerators. This makes the proposed framework practical for real-time PCB inspection in various industrial settings. The lightweight YOLOv8-MBNet model showed efficient inference performance on the Raspberry Pi, even with CPU-only execution. Because of its compact design with 1.15 million parameters and low GFLOPs, the model achieved an inference time of 272.67 ms on PyTorch and 95.89 ms with ONNX optimization. The smaller memory usage and stable thermal performance further prove the model's suitability for low-power, continuous inspection tasks. These results confirm that simplifying the architecture is vital for practical edge deployment on devices with limited resources. On the NVIDIA Jetson Orin Nano, the proposed model provided real-time inference by effectively using GPU acceleration and TensorRT optimization. PyTorch CUDA execution achieved a latency of 36.52 ms. This latency decreased further in TensorRT FP32 and FP16 modes. The most significant improvement came from using INT8 precision, which resulted in a 24.16 ms inference time and over 41 FPS throughput.These gains occurred without increasing the model's complexity, as GFLOPs stayed the same across execution modes. Stable power use and thermal performance show the design's efficiency. The results suggest that the lightweight structure works well with modern AI accelerators, allowing for fast and low-latency inference in real-time industrial PCB inspection. The evaluation highlights the model's adaptability across different edge platforms. While the Raspberry Pi allows for cost-effective deployment with reasonable inference speeds for low-rate inspection, the Jetson Orin Nano supports high-throughput real-time inference through GPU and TensorRT acceleration. The same lightweight model scaled effectively on both platforms, achieving significant latency reduction on the Jetson without any architectural changes. This shows that the proposed design successfully connects low-power embedded devices with high-performance edge AI systems, making it a flexible option for industrial deployment.

E. Importance of Edge Deployment in PCB Defect Detection

Edge deployment is key for effective PCB defect detection systems. Fast decision-making, data privacy, and reliable operations are essential. In fast-paced manufacturing environments, sending high-resolution PCB images to centralized cloud servers causes communication delays and increases bandwidth usage. This can significantly slow down real-time inspection and defect localization. By enabling on-device inference, edge-based solutions allow for immediate detection and response. This reduces production downtime and limits the spread of defective boards along the assembly line.Additionally, edge intelligence improves system reliability and scalability by removing the need for constant network connectivity and central computing resources. Lightweight models used at the edge can be easily duplicated across multiple inspection stations with

low infrastructure costs. This supports distributed and scalable quality control. The proposed YOLOv8-MBNet architecture requires little computing power and provides high detection accuracy. It is especially suitable for these situations. This has been demonstrated on both CPU-only and GPU-accelerated edge platforms. This shows that efficient edge deployment is not just an optimization choice; it is a necessity for achieving real-time, reliable, and effective PCB defect detection systems.

IX. Conclusion

This work presented a lightweight and intelligent multi-stage framework for inspecting defects in printed circuit boards. It aims to address the trade-offs between accuracy and efficiency that limit traditional deep learning inspection systems in industrial environments. A modified YOLOv8 model included MobileNet-style depthwise separable convolutions and optimized C2f blocks. This adjustment significantly reduced the model's complexity while maintaining strong multi-scale feature extraction. The new detector achieved an $mAP_{0.5}$ of 98.6% on the enhanced PKU PCB defect dataset, using only 1.15 million parameters. This result demonstrates that optimizing the architecture can effectively replace methods that compress models after training. In addition to locating defects, the framework improved inspection capabilities with a second-stage semantic analysis using MobileViT-XS. This addition enabled better understanding of the context surrounding detected defects. The use of fuzzy logic for grading severity further enhanced interpretability by turning detection results into actionable quality categories for industry decision-making. Thorough ablation studies and comparisons with recent leading methods confirmed that the proposed design balances detection accuracy and computational efficiency well. Real-world tests on the Raspberry Pi 5 and NVIDIA Jetson Orin Nano in various execution modes showed fast inference, low power usage, and consistent performance. This demonstrates that the framework works well for edge-based inspection. Overall, the proposed method provides a scalable, deployable, and smart solution for real-time detection and assessment of PCB defects. It connects high-performance deep learning models with practical needs in industry.

References

[1] J. Wang, H. Zou, M. Zhou, and R. Su, "An efficient deep neural network for surface defect detection in industrial edge sensing," *IEEE Transactions on Industrial Informatics*, vol. 21, no. 3, pp. 2560–2569, 2025.

[2] Y. Wang, Y. Li, D. M. S. Kayes, H. S. Abdullahi, S. Gao, H. Zhang, Z. Song, and P. Lv, "Research on a lightweight pcb detection algorithm based on ae-yolo," *IEEE Access*, vol. 12, pp. 109 367–109 379, 2024.

[3] G. Li, Y. Gan, W. Zhang, and H. Che, "Gs-yolo: A lightweight and high-performance method for pcb surface defect detection," *Expert Systems with Applications*, vol. 303, p. 130583, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417425041983

[4] W. Zhou, C. Li, Z. Ye, Q. He, Z. Ming, J. Chen, F. Wan, and Z. Xiao, "An efficient tiny defect detection method for pcb with improved yolo through a compression training strategy," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–14, 2024.

[5] S. K. Ong, C. K. Tan, V. M. Baskaran, B. K. Puah, and K. H. Liew, "Enhancing industrial pcb and pcba defect detection: An efficient and accurate seconv-yolo approach," *IEEE Access*, vol. 13, pp. 148 917–148 935, 2025.

[6] W. Chen, Z. Huang, Q. Mu, and Y. Sun, "Pcb defect detection method based on transformer-yolo," *IEEE Access*, vol. 10, pp. 129 480–129 489, 2022.

[7] X. Chen, Y. Wu, X. He, and W. Ming, "A comprehensive review of deep learning-based pcb defect detection," *IEEE Access*, vol. 11, pp. 139 017–139 038, 2023.

[8] K. Li, X. Zhong, and Y. Han, "A high-performance small target defect detection method for pcb boards based on a novel yolo-dfa algorithm," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–12, 2025.

[9] S. Tang, F. He, X. Huang, and J. Yang, "Online pcb defect detector on a new pcb defect dataset," *arXiv preprint arXiv:1902.06197*, 2019.

[10] Q. Ling, C. Kai, and Z. Wenyuan, "Dcd-net: Dense component detection network for printed circuit board assembly inspection," *IEEE Access*, vol. 13, pp. 190 491–190 505, 2025.

[11] G. Zhou, L. Yu, Y. Su, B. Xu, and G. Zhou, "Lightweight pcb defect detection algorithm based on msd-yolo," *Cluster Computing*, vol. 27, no. 3, pp. 3559–3573, 2024.

[12] M. Yuan, Y. Zhou, X. Ren, H. Zhi, J. Zhang, and H. Chen, "Yolo-hmc: An improved method for pcb surface defect detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–11, 2024.

[13] Q. Wang, M. Wang, J. Sun, D. Chen, and P. Shi, "Review of surface-defect detection methods for industrial products based on machine vision," *IEEE Access*, vol. 13, pp. 90 668–90 697, 2025.

[14] Q. Ling, N. A. M. Isa, and M. S. M. Asaari, "Precise detection for dense pcb components based on modified yolov8," *IEEE Access*, vol. 11, pp. 116 545–116 560, 2023.

[15] Peking University, "Pku pcb defect dataset," http://robotics.pkusz.edu.cn/resources/dataset/, accessed: 2024.

[16] L. Shen, B. Lang, and Z. Song, "Ds-yolov8-based object detection method for remote sensing images," *Ieee Access*, vol. 11, pp. 125 122–125 137, 2023.

[17] A. F. Rasheed and M. Zarkoosh, "Optimized yolov8 for multi-scale object detection," *Journal of Real-Time Image Processing*, vol. 22, no. 1, p. 6, 2025.