Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.087068

ARTICLE

FGE-YOLO: A Lightweight YOLOv8-Based Model for Printed Circuit Board Defect Detection

Chun-Hsiu Yeh[1,*], **Xian-Zhong Lin**[1,*], **Yi-Teng Lin**[1], **Yung-Chen Chou**[2] **and Wei-Cheng Shen**[1]

[1]Department of Information Engineering and Computer Science, Feng Chia University, Taichung City, Taiwan
[2]Bachelor's Degree Program in Artificial Intelligence Technology and Application, Feng Chia University, Taichung City, Taiwan
*Corresponding Authors: Chun-Hsiu Yeh. Email: chunhyeh@fcu.edu.tw; Xian-Zhong Lin. Email: p1331934@o365.fcu.edu.tw

Received: 10 June 2026; Accepted: 24 July 2026; Published: 15 September 2026

ABSTRACT: Printed circuit board (PCB) defect detection is critical for industrial quality control, where detection models must identify small and irregular defects while satisfying real-time inspection requirements. However, conventional deep learning-based detectors often require substantial computational resources, making deployment on edge devices difficult. To address this issue, FGE-YOLO is proposed as a deployment-oriented lightweight object detection model based on YOLOv8. The proposed model integrates a FasterNet-based backbone, a GhostConv-Based Neck, and an Efficient Channel Attention (ECA) mechanism. In the backbone, standard convolutions are retained in the shallow P1 and P2 stages to preserve low-level spatial details, while FasterNet-based modules are introduced in deeper stages to reduce redundant computation. In the neck, the original C2f modules are replaced with C3Ghost modules constructed from GhostConv operations to improve multi-scale feature fusion efficiency. ECA modules are further introduced to recalibrate channel-wise feature responses and enhance defect-related feature representation. Experiments were conducted on the augmented HRIPCB dataset containing six common PCB defect categories. Compared with the YOLOv8s baseline, FGE-YOLO reduces the number of parameters from 9.8M to 2.54M and decreases the computational cost from 23.4 GFLOPs to 7.1 GFLOPs. On the workstation GPU platform, the inference speed increases from 325 to 453 FPS, while mAP@0.5 remains nearly unchanged. Although mAP@0.5:0.95 decreases from 0.786 to 0.737, the proposed model provides a practical trade-off between strict localization accuracy and computational efficiency. Deployment experiments on an NVIDIA Jetson Orin Nano Super using TensorRT FP16 further demonstrate the practical feasibility of the proposed model. Under dynamic frequency scaling, FGE-YOLO reduces the TensorRT engine size from 21.8 to 7.2 MB, decreases the end-to-end latency from 18.68 to 18.08 ms, and increases throughput from 53.54 to 55.34 FPS. It also reduces the total board energy consumption per image from 0.1971 to 0.1901 J. These results indicate that FGE-YOLO is suitable for resource-constrained PCB inspection scenarios where model compactness, inference efficiency, and energy consumption are important considerations.

KEYWORDS: PCB defect detection; FGE-YOLO; lightweight detection; GhostConv-based neck; edge deployment

1 Introduction

Printed circuit boards (PCBs) are essential components in modern electronic products because they provide mechanical support and electrical interconnections for electronic devices [1]. With the increasing demand for miniaturized and highly integrated electronics, PCB manufacturing processes have become more complex, and defects may occur during lamination, drilling, electroplating, and etching [2–4]. Common PCB surface defects include Missing Hole, Mouse Bite, Open Circuit, Short Circuit, Spur, and Spurious Copper [5,6]. These defects may affect circuit connectivity, signal transmission, and product reliability,

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.

especially in high-reliability applications such as aerospace, medical electronics, defense systems, and precision instruments [7,8]. Therefore, accurate and timely PCB defect detection is important for improving production yield and ensuring product reliability [9].

Traditional PCB inspection mainly relies on manual visual inspection or conventional Automated Optical Inspection (AOI) systems. Manual inspection is labor-intensive and easily affected by fatigue, lighting conditions, and subjective judgment, making it unsuitable for high-speed and large-scale manufacturing [9]. Conventional AOI systems improve inspection efficiency by using image processing methods such as template matching, image subtraction, threshold segmentation, and rule-based feature extraction [1,10,11]. However, these methods are sensitive to illumination changes, background interference, image misalignment, and variations in defect morphology. As a result, their robustness is limited when detecting small defects or defects located in complex PCB textures [12,13]. In addition, the deployment cost of high-end AOI equipment may be a burden for small and medium-sized electronics manufacturers [2,14].

In recent years, deep learning has provided a more adaptive solution for visual defect detection. Convolutional neural networks (CNNs) can automatically learn discriminative feature representations from image data and have shown stronger adaptability than handcrafted feature extraction methods [1,4]. Among deep learning-based object detectors, the YOLO series has been widely used in industrial inspection because it provides a favorable balance between detection accuracy and inference speed through a single-stage detection framework [2,8,15,16]. This makes YOLO-based models suitable for real-time inspection scenarios.

Despite these advantages, PCB defect detection still faces two major challenges. First, PCB defects are often small, irregular, and weakly defined. They may occupy only a small portion of the image and appear in complex circuit patterns or background textures [1,17]. During deep feature extraction, repeated downsampling and convolution operations may weaken small defect features, leading to missed detections or inaccurate localization [13,15,18,19]. Defects such as Mouse Bite and Spur are particularly difficult to detect because their visual patterns are subtle and easily confused with normal PCB traces.

Second, real-time deployment on resource-constrained edge devices remains challenging. With the development of the Industrial Internet of Things (IIoT) and cloud-edge-end collaborative manufacturing systems, there is an increasing demand for deploying visual inspection models on embedded platforms, edge devices, mobile terminals, and low-cost industrial controllers [20,21]. However, high-accuracy detection models often require a large number of parameters and high computational cost, which increases memory usage, power consumption, and inference latency [2,4,15]. Therefore, PCB inspection models must balance model compactness, inference speed, and small-defect feature preservation.

To address these challenges, this study develops FGE-YOLO as a deployment-oriented lightweight detector for PCB defect inspection. The proposed framework does not simply replace all standard convolutional stages with lightweight operators. Instead, it adopts a stage-specific design that retains standard convolutions in the shallow P1 and P2 stages to preserve edges, contours, and fine spatial details, while introducing FasterNet-based modules only in the deeper P3–P5 stages to reduce redundant spatial computation. In the neck, the original C2f modules are replaced with C3Ghost modules constructed from GhostConv operations to reduce the computational burden of multi-scale feature fusion. Efficient Channel Attention is further introduced after high-level feature extraction and major fusion nodes to recalibrate defect-sensitive channel responses with limited overhead. Through this task-oriented integration, FGE-YOLO aims to balance model compactness, inference efficiency, and small-defect localization rather than maximizing a single accuracy metric. Its practical deployment capability is further evaluated on an NVIDIA Jetson Orin Nano Super using TensorRT FP16.

The main contributions of this study are summarized as follows:

- A stage-specific lightweight backbone is developed for PCB defect inspection. Instead of fully replacing the YOLOv8 backbone, FGE-YOLO retains standard convolutions in the shallow P1 and P2 stages and introduces FasterNet-based modules only in the deeper P3–P5 stages. This design reduces redundant computation while preserving low-level spatial information required for small-defect localization.
- A lightweight multi-scale feature fusion strategy is constructed by replacing the original C2f modules in the YOLOv8 neck with C3Ghost modules based on GhostConv operations. This modification reduces the parameters and computational cost of the neck while maintaining feature aggregation across multiple scales.
- Efficient Channel Attention is integrated after high-level feature extraction and major fusion nodes to compensate for the possible weakening of defect-sensitive features caused by lightweight convolutional operations. This provides channel-wise feature recalibration with limited additional computational overhead.
- Comprehensive ablation and deployment experiments demonstrate the practical value of the proposed design. The results show that retaining the shallow standard-convolution stages improves strict localization accuracy compared with a fully lightweight backbone, while TensorRT FP16 deployment on an NVIDIA Jetson Orin Nano Super verifies the model's compact engine size, real-time inference capability, and energy efficiency.

2 Related Work

2.1 PCB Defect Detection

PCB defect inspection has gradually shifted from manual and traditional machine-vision methods toward deep learning-based visual inspection because conventional approaches provide limited robustness in high-speed manufacturing environments. Manual inspection is intuitive but labor-intensive and easily affected by fatigue, lighting conditions, and subjective judgment [1,4,9]. Therefore, manual inspection is unsuitable for high-speed and large-scale PCB manufacturing.

Traditional Automated Optical Inspection (AOI) systems use rule-based image processing methods, such as template matching, threshold segmentation, edge detection, morphological operations, and image subtraction [1,9,10]. Template matching methods, such as Normalized Cross-Correlation (NCC), also require precise image registration, and minor rotation or displacement may result in many false alarms [10]. In addition, handcrafted feature rules must often be redesigned for different defect types, which limits the adaptability of traditional AOI systems to changing inspection conditions [1].

Deep learning provides a more adaptive solution for PCB defect detection. CNN-based models can learn hierarchical feature representations directly from image data and have been applied to PCB and industrial defect detection tasks [1,11,22]. With the development of object detection algorithms, PCB inspection has shifted toward instance-level localization, where YOLO-based models can predict both defect categories and bounding boxes [22]. Nevertheless, small, irregular, and texture-similar PCB defects remain difficult to localize accurately. This localization challenge motivates the use of real-time object detectors that can jointly support defect classification, bounding-box prediction, and production-line throughput.

2.2 YOLO-Based Object Detection

Among current object detectors, the YOLO series is particularly suitable for industrial inspection because its single-stage framework provides a favorable balance between inference speed and localization capability [23–26]. Compared with two-stage detectors, YOLO-based models are more suitable for production-line inspection, where images must be processed continuously with low latency.

YOLOv3 introduced multi-scale prediction to improve detection across different object sizes [23]. YOLOv5 further improved engineering flexibility by providing different model scales and combining a Cross Stage Partial (CSP)-based backbone with Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) structures for multi-scale feature fusion [24]. More recent YOLO variants, such as YOLOv10 and YOLOv11, introduce additional training strategies and architectural refinements to improve detection performance and inference efficiency [25]. However, these models often contain more coupled designs, which may introduce additional variables when evaluating the effect of a specific lightweight module.

YOLOv8 provides a clear and modular baseline for controlled architectural modification. Its backbone uses C2f modules to improve feature extraction and gradient flow, its neck adopts an FPN/PAN-based multi-scale feature fusion structure, and its detection head uses a decoupled anchor-free design [26]. The anchor-free mechanism is useful for PCB defects with irregular shapes. However, repeated convolutional and C2f operations in the backbone and neck still contribute considerable parameters, computational cost, and inference latency. These characteristics make YOLOv8 an appropriate baseline for examining how lightweight backbone, neck, and attention designs affect the accuracy–efficiency trade-off.

2.3 Lightweight Neural Network Design

Lightweight neural network design must reduce model complexity while also considering actual hardware execution efficiency and the preservation of defect-related features. Common strategies include replacing standard convolution with more efficient operators and simplifying feature aggregation structures.

MobileNet introduced depthwise separable convolution to reduce theoretical computation, and ShuffleNet introduced channel shuffle and practical design principles related to memory access and parallelism [27–30]. These studies indicate that low floating-point operations (FLOPs) do not always lead to low latency because memory access and hardware execution characteristics also affect actual inference speed.

FasterNet uses Partial Convolution (PConv), which applies spatial convolution to only part of the input channels while preserving the remaining channels through identity mapping [31]. This design reduces redundant spatial computation and memory access overhead. Lin et al. [32] introduced FasterNet into YOLOv8 for real-time PCB defect detection and showed that PConv can improve inference speed while maintaining competitive detection accuracy. These findings support the use of FasterNet in computationally intensive backbone stages, while also suggesting that shallow spatial features should be preserved when detecting small PCB defects.

Lightweighting the backbone alone does not fully address the computational burden of object detection, because the neck also processes multi-scale feature maps at relatively high spatial resolutions. Group Shuffle Convolution (GSConv) combines standard convolution with depthwise separable convolution and feature shuffling to improve feature aggregation efficiency [33]. GhostNet further points out that many feature maps generated by deep networks are similar and can be produced through low-cost transformations instead of full standard convolution [34]. This idea motivates the use of GhostConv-based modules in feature fusion. Accordingly, this study adopts a GhostConv-Based Neck in which C3Ghost modules replace the original C2f modules to reduce redundant computation during multi-scale feature fusion.

2.4 Attention Mechanisms

Attention mechanisms are particularly useful in lightweight PCB detection because computational reduction may weaken subtle defect-related responses. They can emphasize informative defect features while suppressing less relevant activations in complex circuit backgrounds.

CBAM combines channel attention and spatial attention to refine feature maps, but it introduces additional pooling, convolution, and fully connected operations [35]. Coordinate Attention (CA) preserves positional information by decomposing global pooling into horizontal and vertical directions, which is useful for location-sensitive tasks but also increases computational overhead [36]. Efficient Channel Attention (ECA) avoids dimensionality reduction and uses one-dimensional convolution to model local cross-channel interactions [37]. Compared with heavier attention modules, ECA recalibrates channel responses with a small increase in parameters and computation. It is therefore more compatible with the lightweight objective of FGE-YOLO, where defect-sensitive features must be enhanced without substantially reducing inference efficiency. This design rationale leads to the comparative discussion of recent YOLO-based PCB defect detectors in Section 2.5.

2.5 YOLO-Based PCB Defect Detection

Recent YOLO-based PCB defect detection studies have primarily focused on enhancing small-defect representation, reducing model complexity, and improving multi-scale feature fusion. Mo et al. [2] proposed SGT-YOLO based on YOLOv5s. The method employs an SE-ENv2 backbone to preserve detailed and positional information, simplifies the detection structure to focus on small PCB defects, introduces a task-specific decoupled detection head, and constructs a lightweight Global Context-Neck (GC-Neck). These modifications demonstrate that lightweight backbone design and task-specific detection structures can jointly improve the accuracy–complexity balance. Zhou et al. [17] addressed tiny PCB defects using an improved YOLO model and a compression training strategy, showing that model compression can support efficient detection when fine-grained features are adequately preserved.

Recent studies published in 2025 have further explored lightweight PCB defect detection based on YOLO architectures. Kong et al. [38] proposed GESC-YOLO based on YOLOv8n, in which lightweight feature extraction and attention-based enhancement are combined to reduce computational complexity while maintaining defect representation. Li et al. [39] proposed SCF-YOLO, which adopts a lightweight MobileNet-based feature extraction network, a learnable weighted feature fusion mechanism, and an SCF module to improve multi-scale semantic representation. These methods demonstrate the feasibility of replacing computationally expensive backbone and fusion components with lightweight alternatives. However, replacing the backbone extensively may weaken low-level spatial information, while additional enhancement modules can partially offset the computational benefits obtained through lightweighting.

More recently, Li et al. [40] proposed GS-YOLO as a lightweight and high-performance method for PCB surface defect detection. This study further reflects the current research trend toward jointly optimizing model compactness and detection performance rather than considering accuracy or computational cost independently. Together, SGT-YOLO, GESC-YOLO, SCF-YOLO, and GS-YOLO show that lightweight PCB detectors commonly rely on backbone replacement, efficient convolution, enhanced feature fusion, attention mechanisms, or detection-head modification. Nevertheless, the simultaneous reduction of model complexity and preservation of strict bounding-box localization remain challenging, particularly for small and irregular defects.

FGE-YOLO differs from these methods in its stage-specific lightweighting strategy and deployment-oriented evaluation. Instead of replacing the entire backbone with lightweight operators, FGE-YOLO retains standard convolutions in the shallow P1 and P2 stages to preserve edges, contours, and fine spatial details, while FasterNet-based modules are applied only in the deeper P3–P5 stages. The GhostConv-Based Neck reduces redundant computation during multi-scale feature fusion, and ECA provides lightweight channel recalibration after high-level feature extraction and major fusion nodes. Thus, the contribution of FGE-YOLO lies not in introducing an entirely new convolutional operator or attention mechanism, but in coordinating the locations and functions of existing lightweight components for small PCB defects. In addition, its practical efficiency is evaluated on an NVIDIA Jetson Orin Nano Super using TensorRT FP16, providing device-level evidence in terms of engine size, end-to-end latency, throughput, power consumption, energy consumption, and unified memory usage.

3 Research Methodology

The proposed FGE-YOLO model improves the deployment efficiency of YOLOv8 for PCB defect detection by integrating lightweight feature extraction, lightweight feature fusion, and channel-wise attention. Specifically, FasterNet-based modules are introduced into the deeper backbone stages to reduce redundant spatial computation, a GhostConv-Based Neck constructed with C3Ghost modules is used to reduce the computational burden of multi-scale feature fusion, and Efficient Channel Attention (ECA) is inserted to recalibrate defect-related channel responses. Through this coordinated integration, each component addresses a different source of computational or representational inefficiency: FasterNet reduces redundant computation in deeper feature extraction, the GhostConv-Based Neck reduces redundancy during multi-scale fusion, and ECA restores channel selectivity with limited overhead.

3.1 FGE-YOLO Model Architecture

The overall architecture of FGE-YOLO is shown in Fig. 1. The proposed model selectively modifies the backbone and neck of YOLOv8 while retaining the original anchor-free detection head. This selective modification preserves the original anchor-free detection mechanism for bounding-box regression and defect classification while concentrating lightweight optimization on the computationally intensive backbone and neck.

In the backbone, the shallow P1 and P2 stages retain the standard YOLOv8 convolutional structure to preserve low-level spatial details, such as edges, contours, and fine textures. FasterNet-based modules are introduced from P3 to P5 to reduce redundant computation in deeper feature extraction stages. After the P5 stage, the Spatial Pyramid Pooling-Fast (SPPF) layer aggregates high-level contextual information, followed by an ECA module for channel-wise feature recalibration. This stage-specific design avoids applying lightweight operators uniformly across the entire backbone, because shallow feature maps contain the edge, contour, and texture information required for precise localization of small PCB defects.

In the neck, the original YOLOv8 feature fusion structure is modified as a GhostConv-Based Neck. In this study, GhostConv-Based Neck refers to a modified neck structure in which the original C2f modules are replaced with C3Ghost modules constructed from GhostConv operations; it does not indicate the use of the complete GhostNet architecture. The neck receives multi-scale features from the P3, P4, and P5 stages and performs feature fusion through upsampling, concatenation, C3Ghost modules, and GhostConv operations. ECA modules are inserted after the major concatenation nodes so that channel recalibration is performed after multi-scale features have been aggregated, rather than before complementary spatial and semantic information is fused.

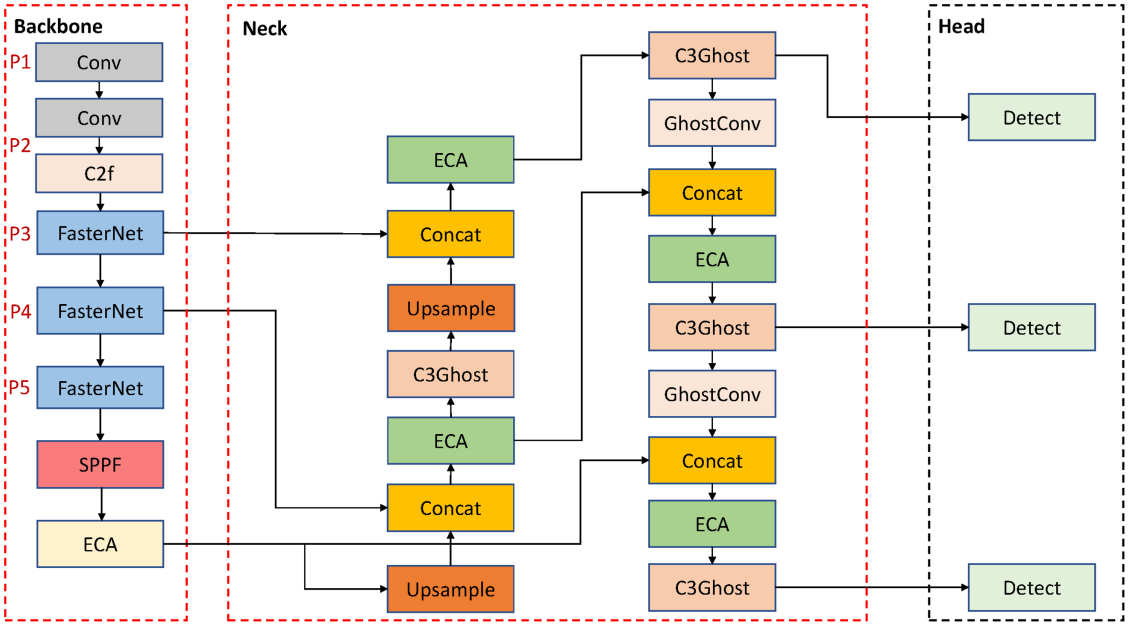

Figure 1: Overall architecture of the proposed FGE-YOLO model.

3.2 FasterNet-Based Backbone Lightweighting

To reduce backbone computation without excessively weakening low-level spatial representation, FGE-YOLO introduces FasterNet-based modules only into the deeper feature extraction stages. FasterNet uses Partial Convolution (PConv), which applies spatial convolution to only a subset of input channels while preserving the remaining channels through identity mapping. Compared with standard convolution, this design reduces redundant spatial computation and memory access cost.

Let the input and output feature maps have the same number of channels c and spatial size $h \times w$, with a convolution kernel size of k. The FLOPs and memory access cost of standard convolution can be expressed as follows:

$$\mathrm{FLOPs}_{\mathrm{Conv}} = h \times w \times k^2 \times c^2, \tag{1}$$

$$\mathrm{MAC}_{\mathrm{Conv}} = h \times w \times 2c + k^2 \times c^2 \approx h \times w \times 2c. \tag{2}$$

For PConv, only c_p channels participate in spatial convolution. Therefore, the corresponding FLOPs and memory access cost are given by:

$$\mathrm{FLOPs}_{\mathrm{PConv}} = h \times w \times k^2 \times c_p^2, \tag{3}$$

$$\mathrm{MAC}_{\mathrm{PConv}} = h \times w \times 2c_p + k^2 \times c_p^2 \approx h \times w \times 2c_p. \tag{4}$$

When c_p is set to $c/4$, the FLOPs of PConv become approximately 1/16 of those of standard convolution. In addition, because PCB defect detection often involves high-resolution feature maps, memory access is strongly affected by feature-map reading and writing. By reducing the number of channels involved in spatial convolution, PConv can lower memory access overhead and improve inference efficiency on GPU-based platforms. This approximation is intended to illustrate the relative reduction in feature-map read/write operations rather than to provide an exact prediction of end-to-end hardware latency, which is also affected by kernel implementation, parallelism, and memory scheduling.

3.3 GhostConv-Based Neck

The neck network is responsible for multi-scale feature fusion and can introduce considerable computational cost because it processes feature maps with relatively high spatial resolution. To reduce this fusion-stage overhead, FGE-YOLO replaces the original C2f modules in the YOLOv8 neck with C3Ghost modules, forming the proposed GhostConv-Based Neck while retaining the original top-down and bottom-up multi-scale fusion paths.

GhostConv generates output features in two stages. First, standard convolution is used to generate intrinsic features. Second, additional ghost features are produced from the intrinsic features through low-cost transformations, such as depthwise convolution. The intrinsic and ghost features are then concatenated to form the final output feature map. Accordingly, C3Ghost is applied to the neck because this stage repeatedly processes multi-scale feature maps and therefore offers substantial potential for reducing redundant feature generation without altering the detection head.

3.4 Efficient Channel Attention (ECA)

Although lightweight convolution reduces computational cost, partial channel processing and low-cost feature generation may weaken subtle defect responses. ECA is therefore introduced as a low-overhead compensation mechanism that strengthens informative channel responses without adding spatial-attention operations or dimensionality-reduction layers. ECA is a lightweight channel attention mechanism that avoids dimensionality reduction and uses one-dimensional convolution to model local cross-channel interactions.

After Global Average Pooling (GAP), ECA directly applies one-dimensional convolution to the channel descriptor. The generated channel weights are then passed through a Sigmoid activation function and multiplied with the input feature map to recalibrate channel responses. The kernel size of the one-dimensional convolution is adaptively determined according to the number of channels. Let C denote the number of channels. The kernel size k is calculated as follows:

$$k = \psi(C) = \left| \frac{\log_2(C)}{\gamma} + \frac{b}{\gamma} \right|_{odd}, \tag{5}$$

where $|t|_{odd}$ denotes the nearest odd integer to t. The constants γ and b are typically set to 2 and 1, respectively. In FGE-YOLO, ECA is placed after the SPPF layer and after the major feature-fusion nodes in the neck. These locations contain high-level contextual features or aggregated multi-scale features, allowing channel recalibration to emphasize defect-sensitive responses where feature redundancy and semantic interaction are greatest.

Overall, FGE-YOLO integrates FasterNet-based backbone lightweighting, a GhostConv-Based Neck, and ECA attention into YOLOv8. The method emphasizes practical deployment efficiency rather than architectural novelty alone, and its effectiveness is evaluated through ablation studies and comparative experiments in Section 4.

4 Experimental Results and Analysis

4.1 Experimental Dataset

4.1.1 Data Sources and Composition

The experiments were conducted using the HRIPCB dataset, also known as the PKU-Market-PCB dataset, released by the Open Laboratory of Intelligent Robotics (HRI Lab) at Peking University. The original dataset contains 693 high-resolution PCB images with 2953 annotated defect instances. To increase sample diversity and reduce the risk of overfitting, this study used a publicly available augmented version of the

HRIPCB dataset. Through operations such as random cropping, rotation, scaling, and flipping, the dataset was expanded to 10,668 defect image samples across six PCB defect categories, including Missing Hole, Mouse Bite, Open Circuit, Short Circuit, Spur, and Spurious Copper. The number of samples in each category ranges from 1732 to 1852, indicating a relatively balanced class distribution. This distribution reduces the risk of model bias toward specific defect categories and supports a fairer evaluation of the proposed model across different defect types.

4.1.2 Defect Category Definition

The six defect categories in the HRIPCB dataset are illustrated in Fig. 2. Missing Hole refers to the absence of a required hole in a pad or via area. Mouse Bite refers to irregular notches along the edge of a conductor. Open Circuit indicates a break in a conductor, whereas Short Circuit indicates an unintended electrical connection between isolated copper traces. Spur refers to small copper protrusions extending from a conductor edge, and Spurious Copper refers to isolated copper residue in non-conductive areas.

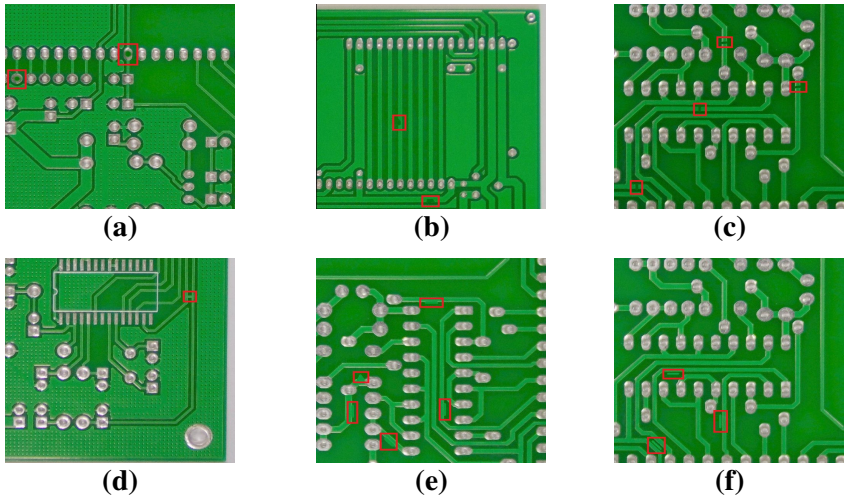

Figure 2: Six common defect categories: (**a**) Missing Hole; (**b**) Mouse Bite; (**c**) Open Circuit; (**d**) Short Circuit; (**e**) Spur; (**f**) Spurious Copper.

4.1.3 Data Preprocessing and Augmentation Strategies

PCB defects usually occupy only a small portion of an image, making feature extraction more difficult during model training. Although the augmented HRIPCB dataset has a relatively balanced class distribution, variations in defect location, size, and appearance may still affect detection performance. Therefore, both offline and online augmentation strategies were adopted.

For offline augmentation, this study used the publicly available augmented HRIPCB dataset, which includes geometric transformations such as random cropping, rotation, scaling, and flipping. For online augmentation, Mosaic augmentation was enabled during YOLOv8 training. Mosaic combines cropped regions from four images into a new training sample, increasing background and object-context diversity. Following the YOLOv8 training procedure, Mosaic augmentation was disabled during the final 10 epochs to allow the model to fine-tune on images closer to the original data distribution.

4.1.4 Dataset Partitioning

For consistent and reproducible evaluation, the 10,668 images were divided into training, validation, and test sets at a ratio of 8:1:1 using stratified sampling. The training set contained 8534 images and was used for model parameter learning. The validation set contained 1067 images and was used for model selection, hyperparameter tuning, and convergence monitoring. The test set contained 1067 images and was used only for final inference evaluation. The main performance metrics reported in this study, including mAP and FPS, were measured on this independent test set.

4.2 Experimental Environment

4.2.1 Experimental Environment Configuration

All experiments were conducted on a workstation equipped with an Intel Core i7-12700KF CPU, 32 GB RAM, and an NVIDIA GeForce RTX 4070 SUPER GPU with 12 GB of video memory. The software environment was based on Windows 11, Python 3.12.9, PyTorch 2.4.1, Torchvision 0.19.1, CUDA 11.8, and cuDNN 9.0.100. The implementation was developed using the Ultralytics YOLOv8 framework.

4.2.2 Baseline Model Selection

YOLOv8 was selected as the baseline framework because of its modular architecture, anchor-free detection head, and suitability for controlled lightweight modification. In the experiments, the small variant, YOLOv8s, was adopted as the baseline implementation to provide a balanced comparison between detection accuracy and computational efficiency. PCB surface defects often have irregular shapes and varying scales, making anchor-free detection useful for reducing dependence on predefined anchor sizes. In addition, YOLOv8 allows the backbone, neck, and attention components to be modified and evaluated independently. Compared with newer YOLO variants, such as YOLOv10 and YOLOv11, YOLOv8 provides a clearer baseline for analyzing the contribution of each proposed lightweight component.

4.2.3 Edge Deployment Environment and Benchmark Protocol

To evaluate practical deployment capability, YOLOv8s and FGE-YOLO were deployed on an NVIDIA Jetson Orin Nano Super with 7.4 GB LPDDR5 unified memory. The platform used JetPack 6.2, L4T R36.4.3, CUDA 12.6, cuDNN 9.3.0, and TensorRT 10.3.0. Both models were exported as TensorRT FP16 engines with an input resolution of 608 × 608 and a batch size of 1.

The benchmark was conducted under MAXN_SUPER power mode with jetson_clocks disabled to retain dynamic frequency scaling. After 10 warm-up inferences, each model was evaluated over five benchmark repetitions, with 100 sequential inference runs per repetition. These repetitions were used to assess deployment variability and are distinct from the five independent training runs used for statistical validation.

End-to-end latency included preprocessing, TensorRT inference, and postprocessing. Throughput was calculated from the mean end-to-end latency. Total-board power was measured from the VDD_IN rail without idle-power subtraction, and peak unified memory was recorded using jtop, including the operating system, CUDA runtime, TensorRT workspace, and intermediate buffers. Latency, FPS, power, and energy per image are reported as mean ± standard deviation over the five benchmark repetitions.

4.3 Evaluation Metrics

4.3.1 Detection Performance Metrics

In object detection, a prediction is considered correct when the predicted class is correct and the Intersection over Union (IoU) between the predicted bounding box and the ground-truth box meets the

specified threshold. For the calculation of Precision and Recall, a prediction was considered a True Positive (TP) when the predicted class was correct and the IoU between the predicted and ground-truth bounding boxes was at least 0.5. Predictions that did not satisfy these conditions were counted as False Positives (FP), while unmatched ground-truth objects were counted as False Negatives (FN). True Negatives (TN) were not included because background regions in object detection are not explicitly counted as negative samples.

Precision measures the reliability of predicted defect regions, while Recall measures the ability to detect ground-truth defects. AP summarizes the precision-recall curve for each class, and mAP averages the AP values across all defect categories. Both mAP@0.5 and mAP@0.5:0.95 are reported. The former reflects detection performance under a relatively lenient overlap criterion, whereas the latter provides a stricter evaluation of bounding-box localization across multiple IoU thresholds.

Let B_p and B_g denote the predicted bounding box and the ground-truth bounding box, respectively. The Intersection over Union (IoU) is defined as

$$IoU = \frac{\left|B_p \cap B_g\right|}{\left|B_p \cup B_g\right|} \tag{6}$$

Precision and Recall are calculated as

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

For the (i)-th defect category, Average Precision (AP) is calculated as the area under the precision–recall curve:

$$AP_i = \int_0^1 P_i(R)dR \tag{9}$$

where $P_i(R)$ denotes Precision as a function of Recall for category i. The mean Average Precision (mAP) over all C defect categories is defined as

$$mAP = \left(\frac{1}{C}\right)\sum_{i=1}^{C} AP_i \tag{10}$$

In this study, mAP@0.5 denotes the mean AP calculated at an IoU threshold of 0.5, whereas mAP@0.5:0.95 denotes the mean AP averaged over IoU thresholds from 0.50 to 0.95 with an interval of 0.05.

4.3.2 Model Lightweighting and Efficiency Metrics

The number of parameters represents the total learnable weights in the model and reflects model size and memory requirements. GFLOPs estimate the computational cost of a single forward pass. FPS measures inference throughput under the same hardware and software environment.

Since this study focuses on the acceleration effect introduced by the lightweight architecture, FPS was calculated based on network inference time rather than complete system processing time. Non-structural factors such as image loading, file I/O, visualization, and CPU-based post-processing were not included. Inference was performed with a batch size of 16, and multiple inference cycles were conducted to evaluate throughput under a high-load inspection setting.

For the embedded-device evaluation, energy consumption per image was calculated as average total-board power divided by FPS. Because idle power was not subtracted, this metric represents total system energy per processed image rather than model-only dynamic energy.

4.4 Training Strategy and Loss Function

4.4.1 Parameter Settings and Optimization Strategy

The model was trained for 300 epochs with an input size of 608 × 608 and a batch size of 16. Stochastic Gradient Descent (SGD) was used as the optimizer, with a momentum of 0.937 and a weight decay of 0.0005. The initial learning rate was set to 0.01, and the final learning-rate factor was set to 0.01, reducing the learning rate to 1% of the initial value during training. The close_mosaic parameter was set to 10, meaning that Mosaic augmentation was disabled during the final 10 epochs to stabilize the final stage of training.

4.4.2 Loss Function

YOLOv8 adopts an anchor-free detection head and is optimized using a composite loss function consisting of bounding-box regression loss, classification loss, and Distribution Focal Loss (DFL). The classification loss was implemented using Sigmoid-based binary cross-entropy loss with logits. For bounding-box localization, Complete IoU (CIoU) loss was used to consider bounding-box overlap, center-point distance, and aspect-ratio consistency. DFL was further used to model bounding-box coordinate locations as discrete probability distributions, which is useful for improving localization of small defects with blurred or irregular boundaries.

4.4.3 Training Convergence Process

Fig. 3 shows the training losses and evaluation metrics of FGE-YOLO over 300 epochs.

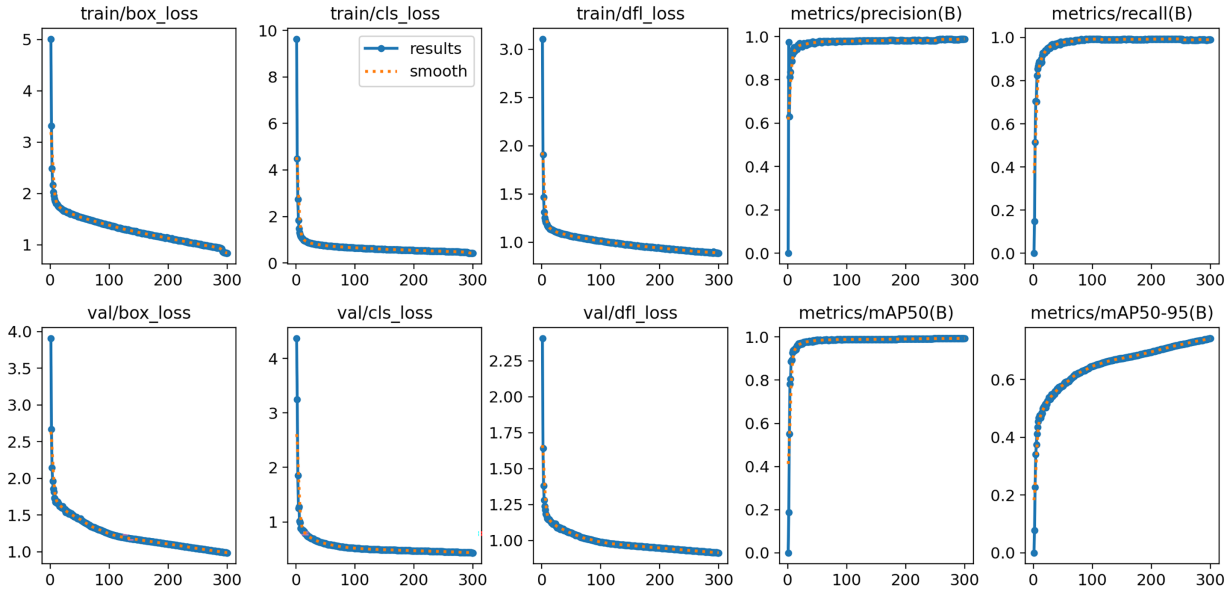

Figure 3: Training loss convergence and accuracy curves of the FGE-YOLO model.

The training and validation loss curves decreased rapidly during the early training epochs and gradually stabilized afterward, with no obvious divergence between them. This behavior indicates stable optimization under the adopted training configuration, and run-to-run variability is further examined through five independent training runs in Section 4.7. Precision, Recall, and mAP@0.5 converged to high levels relatively early, while mAP@0.5:0.95 continued to improve slightly in the later epochs. This suggests that extended training was useful for improving bounding-box localization under stricter IoU thresholds.

4.5 Ablation Studies

4.5.1 Comparison of Attention Mechanisms

Table 1 compares the effects of Coordinate Attention (CA), Efficient Channel Attention (ECA), and Convolutional Block Attention Module (CBAM) based on YOLOv8 with a GhostConv-based neck. The baseline model without attention achieved the highest inference speed of 340 FPS, but its mAP@0.5:0.95 was 0.794. After attention mechanisms were introduced, all variants improved mAP@0.5:0.95, indicating that attention-based feature enhancement is useful for improving localization performance.

Table 1: Comparison of different attention mechanisms based on YOLOv8 with a GhostConv-based neck.

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95	FPS	Params (M)	GFLOPs	Train Time (h)
w/o Attention (Base)	0.988	0.995	0.994	0.794	340	7.4	18.8	4.552
+CA	0.991	0.995	0.994	0.811	282	7.8	19.7	6.102
+ECA	0.991	0.993	0.994	0.801	324	7.6	19.3	5.138
+CBAM	0.992	0.993	0.995	0.825	297	9.7	21.0	5.878

Among the three attention modules, CBAM achieved the highest mAP@0.5:0.95 of 0.825, but its parameters increased to 9.7M, GFLOPs increased to 21.0, and FPS decreased to 297. CA also improved mAP@0.5:0.95 to 0.811, but its FPS decreased to 282. In contrast, ECA improved mAP@0.5:0.95 from 0.794 to 0.801 while maintaining 324 FPS and limiting the increase in GFLOPs. Although CBAM achieved the highest mAP@0.5:0.95, ECA was selected for subsequent experiments because it provided a more favorable balance between localization improvement, computational overhead, and inference speed under the lightweight design objective.

4.5.2 Neck Architecture Comparison Experiment

Table 2 compares different neck architectures under the same FasterNet-based backbone with ECA attention. The compared models include FasterNet-ECA-YOLO (FE-YOLO) with the original Path Aggregation Network (PANet) neck, FasterNet-RepGFPN-ECA-YOLO (FRE-YOLO) with a RepGFPN neck, and FGE-YOLO with the proposed GhostConv-Based Neck.

Compared with FE-YOLO, FRE-YOLO slightly improved mAP@0.5:0.95 from 0.700 to 0.703 and reduced the computational cost from 11.1 GFLOPs to 9.5 GFLOPs. However, its FPS decreased from 447 to 427, and its mAP@0.5 decreased from 0.992 to 0.990. These results indicate that RepGFPN reduced model complexity but did not provide a clear improvement in inference efficiency under the same setting.

The proposed FGE-YOLO achieved the best overall result among the three neck architectures. Its Precision, Recall, mAP@0.5, and mAP@0.5:0.95 reached 0.990, 0.993, 0.993, and 0.737, respectively. Compared with FE-YOLO, its parameters were reduced from 4.7M to 2.54M, GFLOPs decreased from 11.1 to 7.1, and FPS increased from 447 to 453. These results indicate that the GhostConv-Based Neck provided the most

favorable efficiency–accuracy balance among the evaluated neck configurations, reducing parameters and GFLOPs while preserving comparable basic detection performance.

Table 2: Comparison of different neck architectures.

Model	Neck Arch.	Precision	Recall	mAP@0.5	mAP@0.5:0.95	FPS	Params (M)	GFLOPs	Train (h)
FE-YOLO (Base)	PANet	0.984	0.987	0.992	0.700	447	4.70	11.1	2.952
FRE-YOLO	RepGFPN	0.984	0.991	0.990	0.703	427	3.70	9.5	3.942
FGE-YOLO (Ours)	GhostConv-Based Neck	0.990	0.993	0.993	0.737	453	2.54	7.1	4.257

4.5.3 Comprehensive Ablation Study

Table 3 presents the comprehensive ablation study designed to distinguish the individual and combined effects of the FasterNet-based backbone, GhostConv-Based Neck, and ECA mechanism. YOLOv8s was used as the baseline, and FasterNet, the GhostConv-Based Neck, and ECA were introduced in different combinations. F denotes the FasterNet backbone, G denotes the GhostConv-Based Neck, and E denotes the ECA mechanism. FGE-YOLO (Full) replaces the entire backbone with FasterNet, whereas FGE-YOLO (Ours) retains standard convolutions in the shallow P1 and P2 stages. Compared with the YOLOv8s baseline, G-YOLO slightly improves mAP@0.5:0.95 from 0.786 to 0.794, suggesting that GhostConv-based feature fusion can preserve useful spatial information while reducing redundant computation.

Table 3: Comprehensive ablation study of the proposed model.

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95	FPS	Params (M)	GFLOPs	Train Time (h)
YOLOv8s (Baseline)	0.991	0.992	0.994	0.786	325	9.80	23.4	4.482
F-YOLO	0.984	0.984	0.992	0.704	454	4.70	11.1	2.945
FE-YOLO	0.984	0.987	0.992	0.700	447	4.70	11.1	2.952
G-YOLO	0.988	0.995	0.994	0.794	340	7.40	18.8	4.552
GE-YOLO	0.991	0.993	0.994	0.801	324	7.60	19.3	5.138
FGE-YOLO (Full)	0.984	0.993	0.992	0.685	465	2.52	6.4	3.213
FGE-YOLO (Ours)	0.990	0.993	0.993	0.737	453	2.54	7.1	4.257

After replacing the original backbone with FasterNet, F-YOLO reduced the parameters from 9.8M to 4.7M and GFLOPs from 23.4 to 11.1, while increasing FPS from 325 to 454. However, mAP@0.5:0.95 decreased from 0.786 to 0.704, indicating that aggressive backbone lightweighting reduced fine-grained localization ability. In contrast, G-YOLO improved mAP@0.5:0.95 from 0.786 to 0.794 while reducing GFLOPs to 18.8 and increasing FPS to 340. This suggests that replacing the neck with GhostConv-based modules can reduce redundant computation while preserving useful feature information.

The ECA mechanism further improved localization when combined with the GhostConv-Based Neck. GE-YOLO achieved the highest mAP@0.5:0.95 of 0.801 among all configurations, but its FPS was 324 and its GFLOPs were 19.3. When FasterNet and ECA were combined in FE-YOLO, the model retained high inference speed but did not recover the localization loss caused by backbone lightweighting.

The comparison between FGE-YOLO (Full) and FGE-YOLO (Ours) further confirms the importance of retaining shallow spatial features. FGE-YOLO (Full) achieved the highest FPS of 465 and the lowest GFLOPs of 6.4, but its mAP@0.5:0.95 decreased to 0.685. By retaining standard convolutions in P1 and P2, FGE-YOLO (Ours) improved mAP@0.5:0.95 to 0.737 while maintaining 2.54M parameters, 7.1 GFLOPs, and 453 FPS.

Overall, FGE-YOLO (Ours) does not aim to maximize all accuracy metrics. Instead, it provides a practical trade-off between detection accuracy and deployment efficiency. Compared with the YOLOv8s baseline, it reduces the number of parameters from 9.8M to 2.54M and decreases the computational cost from 23.4 GFLOPs to 7.1 GFLOPs. Meanwhile, FPS increases from 325 to 453, while mAP@0.5 remains nearly unchanged.

The decrease in mAP@0.5:0.95 from 0.786 to 0.737 indicates that FGE-YOLO is less accurate under stricter bounding-box overlap criteria, although its mAP@0.5 remains nearly unchanged. This behavior is particularly relevant to PCB defects because many defect regions occupy only a small portion of the image. For small objects, a displacement of only a few pixels may cause a substantial decrease in IoU. In addition, Partial Convolution and GhostConv reduce redundant computation by processing only part of the channel information or generating additional feature maps through low-cost transformations. Although these operations improve computational efficiency, they may weaken fine-grained boundary information required for highly precise bounding-box regression. The comparison between FGE-YOLO (Full) and FGE-YOLO (Ours) supports this interpretation. Retaining standard convolutions in the shallow P1 and P2 stages improves mAP@0.5:0.95 from 0.685 to 0.737, demonstrating that low-level spatial information is important for strict localization. Therefore, FGE-YOLO is positioned as an efficiency-oriented detector for high-throughput and resource-constrained PCB inspection rather than a model designed to maximize strict localization accuracy.

4.6 Comparison with Popular YOLO Models

To assess the relative efficiency of FGE-YOLO within the YOLO family, it was compared with YOLOv8s, YOLOv10s, and YOLOv11s. These variants were selected because they represent comparable model scales intended to balance detection accuracy and computational efficiency. The detailed results are shown in Table 4.

Table 4: Comparison of overall performance between FGE-YOLO and YOLO small models.

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95	FPS	Params (M)	GFLOPs	Train Time (h)
YOLOv8s	0.991	0.992	0.994	0.786	325	9.80	23.4	4.482
YOLOv10s	0.987	0.992	0.993	0.770	320	8.00	24.5	5.930
YOLOv11s	0.989	0.993	0.993	0.771	313	9.40	21.3	4.833
FGE-YOLO (Ours)	0.990	0.993	0.993	0.737	453	2.54	7.1	4.257

The four models achieved nearly identical mAP@0.5 values, ranging from 0.993 to 0.994, indicating that their basic defect-detection capability was comparable at an IoU threshold of 0.5. However, their parameter counts ranged from 8.0M to 9.8M, and their computational costs were all higher than 20 GFLOPs. In contrast, FGE-YOLO reduced the parameters to 2.54M and GFLOPs to 7.1, requiring fewer parameters and lower computational cost than the compared YOLO small models.

In terms of inference speed, FGE-YOLO achieved 453 FPS, outperforming YOLOv8s, YOLOv10s, and YOLOv11s, whose FPS values were 325, 320, and 313, respectively. In terms of detection accuracy, FGE-YOLO maintained an mAP@0.5 of 0.993, comparable to the other YOLO small models. However, its mAP@0.5:0.95 was 0.737, lower than those of YOLOv8s, YOLOv10s, and YOLOv11s. This result indicates that FGE-YOLO

sacrifices part of the strict localization accuracy in exchange for substantial reductions in model size and computational cost.

Overall, FGE-YOLO provides a more compact and higher-throughput alternative among the evaluated YOLO small models. Its principal advantage lies in the reduction of parameters and GFLOPs while maintaining comparable mAP@0.5. However, this efficiency gain is accompanied by lower mAP@0.5:0.95, and the model should therefore be interpreted as an efficiency-oriented option rather than a strict-localization-oriented detector.

4.7 Statistical Stability over Multiple Training Runs

To evaluate run-to-run stability, YOLOv8s and FGE-YOLO were independently trained five times using different random seeds. The dataset split, training settings, and evaluation procedure were kept identical across all runs, and each model was evaluated on the same independent test set. Table 5 reports the mean and standard deviation of Precision, Recall, mAP@0.5, and mAP@0.5:0.95.

Table 5: Statistical stability over five independent training runs.

Model	Precision*	Recall*	mAP@0.5*	mAP@0.5:0.95*
YOLOv8s	0.9912 ± 0.0015	0.9922 ± 0.0020	0.99402 ± 0.00004	0.7942 ± 0.0060
FGE-YOLO	0.9916 ± 0.0011	0.9900 ± 0.0022	0.99324 ± 0.00041	0.7422 ± 0.0061

Note: *Values are mean ± SD over five independent training runs.

Both models exhibit stable detection performance across random seeds. FGE-YOLO achieves comparable mAP@0.5 to YOLOv8s, with small standard deviations in Precision, Recall, and mAP@0.5. However, its mAP@0.5:0.95 remains consistently lower than that of YOLOv8s, confirming that the lightweight design introduces a stable accuracy–efficiency trade-off in strict bounding-box localization rather than unstable training behavior.

4.8 Model Detection Examples

Fig. 4 presents representative detection examples for the six PCB defect categories. The examples show that FGE-YOLO successfully detects representative instances from all six defect categories, including visually small or irregular patterns such as Missing Hole, Mouse Bite, and Spur. However, these qualitative examples are intended to illustrate typical predictions rather than replace the quantitative evaluation. Together with the quantitative results, these examples indicate that the proposed model provides an efficiency-oriented alternative for PCB defect detection when inference throughput and computational efficiency are prioritized.

4.9 Edge Deployment Evaluation

The edge-device evaluation confirms that FGE-YOLO can operate in real time on the NVIDIA Jetson Orin Nano Super while providing a substantially more compact deployment engine. As shown in Table 6, the TensorRT FP16 engine size decreases from 21.8 MB for YOLOv8s to 7.2 MB for FGE-YOLO, corresponding to a reduction of approximately 67.0%.

FGE-YOLO also provides modest improvements in embedded inference efficiency. Its mean latency decreases from 18.68 to 18.08 ms, throughput increases from 53.54 to 55.34 FPS, and energy consumption per image decreases from 0.1971 to 0.1901 J. The average total-board power remains similar, indicating that the lower energy per image mainly results from the slightly higher throughput.

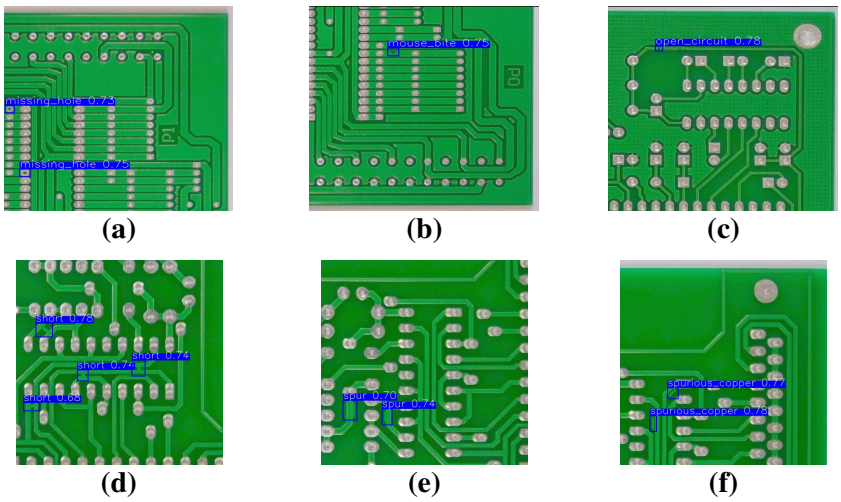

Figure 4: Detection results of the FGE-YOLO model for six types of PCB defects: (**a**) Missing Hole; (**b**) Mouse Bite; (**c**) Open Circuit; (**d**) Short Circuit; (**e**) Spur; (**f**) Spurious Copper.

Table 6: TensorRT FP16 deployment performance on NVIDIA Jetson Orin Nano Super.

Model	Engine Size (MB)	Latency (ms)*	FPS*	Memory (MB)	Power (W)*	Energy (J)*
YOLOv8s	21.8	18.68 ± 0.11	53.54 ± 0.31	3314.0	10.553 ± 0.371	0.1971 ± 0.0069
FGE-YOLO	7.2	18.08 ± 0.49	55.34 ± 1.44	3461.8	10.494 ± 0.866	0.1901 ± 0.0212

Note: *Values are mean ± SD over five repetitions, except for engine size and peak memory.

Peak unified memory usage does not decrease proportionally with model size. YOLOv8s uses 3314.0 MB, whereas FGE-YOLO uses 3461.8 MB. This system-level measurement includes the operating system, CUDA runtime, TensorRT workspace, intermediate activation buffers, and other allocations. Therefore, the reductions in parameters and engine size should not be interpreted as proportional reductions in total system memory usage.

5 Conclusions

This paper presented FGE-YOLO, a compact YOLOv8-based object detector for PCB defect inspection. The model combines a stage-specific FasterNet-based backbone, a GhostConv-Based Neck, and Efficient Channel Attention to reduce redundant computation across feature extraction, multi-scale fusion, and channel recalibration. Rather than fully replacing the backbone with lightweight operators, standard convolutions are retained in the shallow P1 and P2 stages to preserve low-level spatial information, while FasterNet-based modules are introduced only in the deeper P3–P5 stages. Compared with YOLOv8s, FGE-YOLO reduces the parameter count from 9.8M to 2.54M and the computational cost from 23.4 GFLOPs to 7.1 GFLOPs. On the workstation GPU platform, the inference speed increases from 325 to 453 FPS, while mAP@0.5 remains nearly unchanged. However, mAP@0.5:0.95 decreases from 0.786 to 0.737, indicating a measurable trade-off in strict bounding-box localization. The ablation results further show that retaining standard convolutions in the shallow P1 and P2 stages improves mAP@0.5:0.95 compared with a fully lightweight backbone while preserving most of the computational-efficiency gains. Deployment experiments on an NVIDIA Jetson Orin Nano Super using TensorRT FP16 further verified the practical feasibility of FGE-YOLO. The proposed model reduces the TensorRT engine size from 21.8 to 7.2 MB, decreases the mean end-to-end latency from 18.68 to 18.08 ms, increases throughput from 53.54 to 55.34 FPS, and reduces total-board energy

consumption per image from 0.1971 to 0.1901 J under dynamic frequency scaling. Peak unified memory usage does not decrease proportionally with model size because it also includes the operating system, CUDA runtime, TensorRT workspace, and intermediate buffers. Future work will focus on improving strict localization accuracy through hardware-aware architecture optimization, INT8 quantization, structured pruning, knowledge distillation, and validation under continuous industrial-video inspection conditions.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Conceptualization, methodology, supervision, writing—review and editing, Chun-Hsiu Yeh; software, validation, formal analysis, data curation, investigation, writing—original draft preparation, Xian-Zhong Lin; formal analysis, data curation, writing—original draft preparation, Yi-Teng Lin; resources, investigation, software, validation, Yung-Chen Chou; project administration, supervision, resources, writing—review and editing, Wei-Cheng Shen. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in the HRIPCB (PKU-Market-PCB) dataset (https://robotics.pkusz.edu.cn/resources/dataset/) released by the Open Laboratory of Intelligent Robotics, Peking University.

Ethics Approval: Not applicable. This study did not involve human participants or animal subjects.

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

References

1. Ling Q, Isa NAM. Printed circuit board defect detection methods based on image processing, machine learning and deep learning: a survey. IEEE Access. 2023;11(1):15921–44. doi:10.1109/ACCESS.2023.3245093.
2. Mo C, Hu Z, Wang J, Xiao X. SGT-YOLO: a lightweight method for PCB defect detection. IEEE Trans Instrum Meas. 2025;74:3543911. doi:10.1109/TIM.2025.3563011.
3. Liu J, Kang B, Liu C, Peng X, Bai Y. YOLO-BFRV: an efficient model for detecting printed circuit board defects. Sensors. 2024;24(18):6055. doi:10.3390/s24186055.
4. Zhou Y, Yuan M, Zhang J, Ding G, Qin S. Review of vision-based defect detection research and its perspectives for printed circuit board. J Manuf Syst. 2023;70(18):557–78. doi:10.1016/j.jmsy.2023.08.019.
5. Wang Y, Li Y, Kayes DMS, Abdullahi HS, Gao S, Zhang H, et al. Research on a lightweight PCB detection algorithm based on AE-YOLO. IEEE Access. 2024;12:109367–79. doi:10.1109/ACCESS.2024.3439523.
6. Shao M, Min L, Liu M, Li X, liu J, Li X. An enhanced network model for PCB defect detection: CDS-YOLO. J Real Time Image Process. 2024;21(6):196. doi:10.1007/s11554-024-01580-z.
7. Du P, Song X. Lightweight target detection: an improved YOLOv8 for small target defect detection on printed circuit boards. In: Proceedings of the 2024 International Conference on Generative Artificial Intelligence and Information Security; 2024 May 10–12; Kuala Lumpur, Malaysia. New York, NY, USA: ACM; 2024. p. 329–34. doi:10.1145/3665348.3665404.
8. Chen B, Dang Z. Fast PCB defect detection method based on FasterNet backbone network and CBAM attention mechanism integrated with feature fusion module in improved YOLOv7. IEEE Access. 2023;11:95092–103. doi:10.1109/ACCESS.2023.3311260.
9. Chaudhary V, Dave IR, Upla KP. Automatic visual inspection of printed circuit board for defect detection and classification. In: 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET); 2017 Mar 22–24; Chennai, India. p. 732–7. doi:10.1109/WiSPNET.2017.8299858.
10. Annaby MH, Fouda YM, Rushdi MA. Improved normalized cross-correlation for defect detection in printed-circuit boards. IEEE Trans Semicond Manuf. 2019;32(2):199–211. doi:10.1109/TSM.2019.2911062.

11. Kim J, Ko J, Choi H, Kim H. Printed circuit board defect detection using deep learning via a skip-connected convolutional autoencoder. Sensors. 2021;21(15):4968. doi:10.3390/s21154968.
12. Wang G, Chen J, Li C, Lu S. Edge-YOLO: lightweight multi-scale feature extraction for industrial surface inspection. IEEE Access. 2025;13:48188–201. doi:10.1109/ACCESS.2025.3550374.
13. Li Z, Li A, Li W, Kong X, Zhang Y. HSD-YOLO: a lightweight and accurate method for PCB defect detection. In: 2024 International Joint Conference on Neural Networks (IJCNN); 2024 Jun 30–Jul 5; Yokohama, Japan. p. 1–8. doi:10.1109/IJCNN60899.2024.10650691.
14. 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(12):2001611. doi:10.1109/TIM.2024.3351241.
15. Gao Y, Li Z, Wang Y, Zhu S. A Novel YOLOv5_ES based on lightweight small object detection head for PCB surface defect detection. Sci Rep. 2024;14(1):23650. doi:10.1038/s41598-024-74368-7.
16. Li J, Cheng M. FBS-YOLO: an improved lightweight bearing defect detection algorithm based on YOLOv8. Phys Scr. 2025;100(2):025016. doi:10.1088/1402-4896/ad9ef1.
17. Zhou W, Li C, Ye Z, He Q, Ming Z, Chen J, et al. An efficient tiny defect detection method for PCB with improved YOLO through a compression training strategy. IEEE Trans Instrum Meas. 2024;73:2003514. doi:10.1109/TIM.2024.3390198.
18. Liu L, Du D, Sun Y, Li Y. SFMW-YOLO: a lightweight metal casting surface defect detection method based on modified YOLOv8s. Expert Syst Appl. 2025;287:128170. doi:10.1016/j.eswa.2025.128170.
19. Hui M, Yao J, Fu Z, Hai T, Zhang M, Pan T. YOLO-CSS: a lightweight defect detection model for complex substation scenarios. Meas Sci Technol. 2025;36(8):086003. doi:10.1088/1361-6501/adf65d.
20. Lu Y, Li D, Li D, Li X, Gao Q, Yu X. A lightweight insulator defect detection model based on drone images. Drones. 2024;8(9):431. doi:10.3390/drones8090431.
21. Zhou L, Yang S, Wang C, Huang P, Wang S, Wang Y, et al. QCF-YOLO: a lightweight model of surface defect detection for quick-connect fittings. IEEE Sens J. 2025;25(1):1716–31. doi:10.1109/JSEN.2024.3486910.
22. Yu S, Pan F, Zhang X, Zhou L, Zhang L, Wang J. A lightweight detection algorithm of PCB surface defects based on YOLO. PLoS One. 2025;20(4):e0320344. doi:10.1371/journal.pone.0320344.
23. Redmon J, Farhadi A. YOLOv3: an incremental improvement. arXiv:1804.02767. 2018.
24. Jocher G, Chaurasia A, Stoken A, Borovec J, NanoCode012, Kwon Y, et al. ultralytics/YOLOV5: v7.0—YOLOv5 SOTA realtime instance segmentation [Internet]; 2022 [cited 2026 Jan 1]. Available from: https://zenodo.org/record/3908559.
25. Wang A, Chen H, Liu L, Chen K, Lin Z, Han J, et al. YOLOv10: real-time end-to-end object detection. arXiv:2405.14458. 2024.
26. Jocher G, Qiu J, Chaurasia A. Ultralytics YOLO [Python]. 2023 [cited 2026 Jan 1]. Available from: https://github.com/ultralytics/ultralytics.
27. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. 2017.
28. Zhang J, Jing J, Lu P, Song S. Improved MobileNetV2-SSDLite for automatic fabric defect detection system based on cloud-edge computing. Measurement. 2022;201:111665. doi:10.1016/j.measurement.2022.111665.
29. Zhang X, Zhou X, Lin M, Sun J. ShuffleNet: an extremely efficient convolutional neural network for mobile devices. arXiv:1707.01083. 2017.
30. Ma N, Zhang X, Zheng HT, Sun J. ShuffleNet V2: practical guidelines for efficient CNN architecture design. arXiv:1807.11164. 2018.
31. Chen J, Kao SH, He H, Zhuo W, Wen S, Lee CH, et al. Run, don't walk: chasing higher FLOPS for faster neural networks. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2023 Jun 17–24; Vancouver, BC, Canada. p. 12021–31. doi:10.1109/CVPR52729.2023.01157.
32. Lin YT, Lin XZ, Yeh CH, Liao CW, Wang X. Enhancing real-time PCB defect detection with YOLOv8 and a lightweight FasterNet backbone. IET Conf Proc. 2025;2025:261–3. doi:10.1049/icp.2025.2546.
33. Li H, Li J, Wei H, Liu Z, Zhan Z, Ren Q. Slim-neck by GSConv: a lightweight-design for real-time detector architectures. J Real Time Image Process. 2024;21(3):62. doi:10.1007/s11554-024-01436-6.

34. Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C. GhostNet: more features from cheap operations. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020 Jun 13–19; Seattle, WA, USA. p. 1577–86. doi:10.1109/CVPR42600.2020.00165.
35. Woo S, Park J, Lee JY, Kweon IS. CBAM: convolutional block attention module. arXiv:1807.06521. 2018.
36. Hou Q, Zhou D, Feng J. Coordinate attention for efficient mobile network design. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021 Jun 20–25; Nashville, TN, USA. p. 13708–17. doi:10.1109/CVPR46437.2021.01350.
37. Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q. ECA-Net: efficient channel attention for deep convolutional neural networks. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020 Jun 13–19; Seattle, WA, USA. p. 11531–9. doi:10.1109/CVPR42600.2020.01155.
38. Kong X, Liu G, Gao Y. GESC-YOLO: improved lightweight printed circuit board defect detection based algorithm. Sensors. 2025;25(10):3052. doi:10.3390/s25103052.
39. Li Y, Wang Y, Liu J, Wu K, Abdullahi HS, Lv P, et al. Lightweight PCB defect detection method based on SCF-YOLO. PLoS One. 2025;20(4):e0318033. doi:10.1371/journal.pone.0318033.
40. 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.