U.P.B. Sci. Bull., Series C, Vol. 87, Iss. 4, 2025 ISSN 2286 – 3540

# PCB DEFECT DETECTION ALGORITHM BASED ON SGB-YOLOv5s

Xianli JIN [1,4], Jinqiang LI [2,3], Yangyang ZHAO [1,4*]

*Industrial PCB defect detection requires fast and accurate search for defect types and locations, and target detection methods based on deep learning are mainly used in this field. However, in the actual complex production environment, the existing detection methods have shortcomings in detection accuracy, speed, and model size, and it is difficult to complete real-time monitoring and deployment. To this end, this paper proposes a new PCB defect detection method, SGB-YOLOv5s, which greatly improves the detection accuracy and speed. In SGB-YOLOv5s, the use of 160*160 scale detection head and k-means++ clustering algorithm further enhances the detection performance of the network for small target defects. Secondly, the weighted group convolution involution block (GCI) technologies are used between the backbone network and the neck network to reduce the loss of feature information in the feature fusion stage. Finally, by introducing a bidirectional multi-scale feature pyramid structure (BMFPN) into the neck network, the simultaneous fusion of multi-scale features is realized. The experimental results show that the average accuracy of all defect types in the SGB-YOLOv5s model is map@50 and reaches 99.39%. Compared with the original model, the map@50 of all class defects is increased by 1.15% on the basis of almost no increase in the number of parameters and model size, which proves the effectiveness of the improved method.*

**Keywords**: PCB defect detection, YOLOv5, GCI, Detection anchor, BMFPN

## 1. Introduction

With the rapid development of the electronic information industry, the quality and reliability of printed circuit board (PCB) as a key component in electronic equipment [1-3] directly affect the reliability and stability of the entire electronic product. PCB is widely used in many fields such as mobile phones, computers, medical treatment, communication equipment and automobiles, so it is of great significance to strictly control the production quality of PCB and find and repair defects in time to improve the overall quality and market competitiveness of products. Each PCB is subject to rigorous quality inspection before leaving the

[1] New Engineering Industry College, Putian University, Putian, Fujian, China
[2] College of Mechatronics and Information Engineering, Putian University, Putian, Fujian, China
[3] Key Laboratory of Modern Precision Measurement and Laser Nondestructive Detection colleges and Universities in Fujian Province, Putian University, Putian, Fujian, China
[4] Putian Electronic Information Industry Technology Research Institute, Putian University, Putian, Fujian, China, *Corresponding author, E-mail: zhaoyangyang@ptu.edu.cn

factory, often using optical inspection methods [4] (AOI) or deep learning-based object detection algorithms detects defects on printed circuit boards, which can achieve faster inspection speed and accuracy than manual inspection, while also reducing the cost required in production. The object detection algorithm can be divided into two-stage object detection algorithm and single-stage object detection algorithm according to the different processing methods of input data, among which the two-stage object detection algorithm is mainly represented by the RCNN series network, which has the characteristics of high detection accuracy, but the detection speed is slow, and the single-stage object detection algorithm is mainly SSD and the YOLO model [5], the most obvious feature of this type of algorithm is that it is fast and takes into account the detection accuracy, so it has been favored by many scholars, and on this basis, the model is further studied to meet the needs of real-time industry [6].Therefore, after conducting comparative experiments on various mainstream models in the same environment as the dataset and training parameters, this paper determines that the YOLOv5 model is used as the baseline model to study the dataset and network structure, and the specific experimental results are shown in Table 5 below.

Among them, the single-stage object detection algorithm YOLOv5 network consists of three parts: backbone feature extraction network, neck feature fusion network and head detection network. It should be noted that insufficient feature extraction, poor dataset quality, and poor feature fusion will seriously affect the effect of subsequent detection [7]. In order to prevent the occurrence of network training overfitting due to the small number of datasets, we need to perform data augmentation operations on the datasets used before the network is trained.

In view of the problems existing in the above analysis, this paper proposes a network model with better detection accuracy, speed and model size (SGB-YOLOv5s), as shown in Fig.1, which provides a framework that includes data augmentation technology and object detection technology based on deep learning. Firstly, the original data is enhanced by affine transformation, rotation, cropping and color transformation, so as to increase the number of original datasets, so that the network can extract more defect feature information to prevent the network from losing overfitting. Then, the classification and regression of the marked defects in the PCB are realized through the SGB-YOLOv5 network, which can quickly and accurately locate the location of the defects. The main innovations of the work in this paper are as follows:

(1) For the detection anchor size and scale given by the original YOLOv5 network, the 20*20 scale anchor frame is discarded and the 160*160 scale anchor is designed to improve the detection anchor's ability to feel the wildness of small targets. At the same time, in order to better match the size of defects, k-means++ clustering algorithm is used to regenerate the new anchor size.

(2) In order to reduce the loss of information in the process of transmitting features from the backbone feature extraction network to the feature fusion network, the Group Convolution Pair Block (GCI), that is, the group convolution of the features of the input neck without dimensionality reduction is adopted, and the weights of different features are obtained from the training process for weighted recombination, so as to retain more feature information extracted from the backbone network.

(3) The PANet feature pyramid structure used in the original YOLOv5 is changed to a bi-multi-scale feature pyramid Structure (BMFPN), so that the network is able to carry out the process of transferring the low-level features to the high-level features at the same time, and to fuse the feature information of different scales.

The rest of the article is organized as follows: Section 2 describes the related work. Section 3 describes the design methodology for the SGB-YOLOv5 network. Section 4 analyzes the experimental and simulation results.

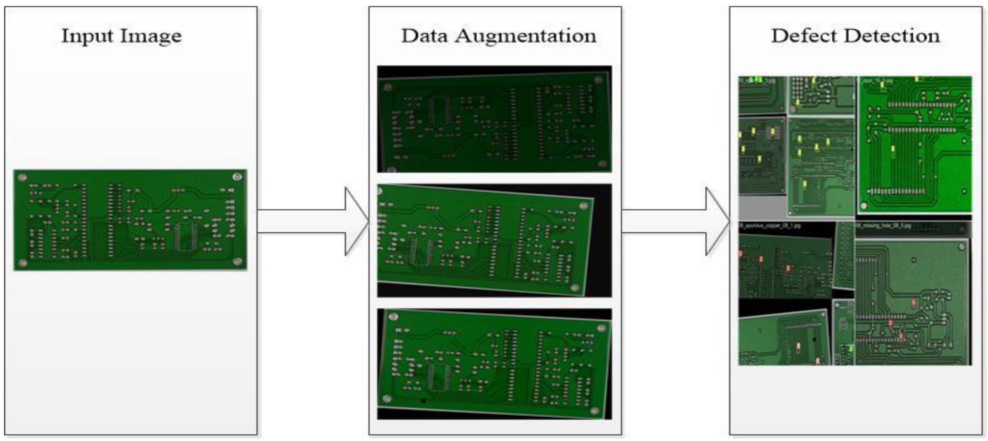

Fig.1. SBG-YOLOv5s framework

## 2. Related Work

### 2.1 Object Detection

The evolution of object detection algorithms can be analyzed through two dimensions: core technological innovations and architectural optimizations. The foundational work began with LeCun et al.'s LeNet-5 network in 1998 [8], which established the CNN architecture through synergistic design of convolutional and pooling layers, laying the theoretical groundwork for modern object detection. In 2005, Dalal et al. [9] developed the HOG Detector using gradient histogram features for efficient detection, though its local feature modeling mechanism lacked global semantic perception capabilities. A significant breakthrough occurred in 2014 when Kaiming He [10] proposed SPPNet, introducing spatial pyramid pooling modules to overcome traditional CNN input size constraints while significantly enhancing model generalization. The technological breakthrough in single-stage detectors commenced with Redmon's YOLOv1 in 2015 [11], achieving millisecond-level detection speed through an end-to-end architecture, albeit with precision limitations from coarse-grained feature fusion. Subsequently, Liu et al.'s SSD algorithm in 2016 [12] enhanced small object detection via multi-scale feature map fusion while maintaining real-time performance, though its memory footprint and hardware dependencies constrained industrial deployment potential. The YOLOv3 architecture introduced by Redmon's team in 2018 [13] employed multi-scale training strategies and Feature Pyramid Networks (FPN) to strengthen multi-scale detection, yet small target localization accuracy remained suboptimal. The subsequent YOLOv4 [14] innovatively integrated the CSPDarkNet53 backbone with cosine annealing learning rate scheduling, improving convergence efficiency while mitigating overfitting. Recent algorithmic optimizations focus on lightweight design and attention mechanism integration. The YOLOv5 framework proposed by the Ultralytics team in 2021 achieved precision-speed balance through adaptive training strategies and Mosaic data augmentation, with Team L [15] extending its application to infrared image detection, validating cross-modal adaptability. Huang et al.'s 2023 improvement [16] utilized C2f convolutions to reduce computational complexity while incorporating EMA attention mechanisms to enhance multi-scale context modeling. Wang et al.'s YOLOv10 in 2024 [17] realized classifier head lightweighting through depthwise separable convolutions and spatial-channel decoupled downsampling, combined with self-attention mechanisms to strengthen feature discriminability. Bakirci et al. [18] applied YOLOv11 to UAV aerial vehicle detection, achieving real-time performance while still facing missed detection challenges with small-scale aerial targets.

### 2.2 PCB Defect Detection

PCB defect detection algorithms can be roughly divided into two categories: traditional methods and deep learning methods. Traditional methods integrate mathematical modeling, signal processing, and computer vision technologies to achieve defect detection through image preprocessing, feature extraction, and classification. Ongshenjit J et al. [19] proposed an algorithm capable of simultaneously detecting and classifying 14 types of PCB defects with high accuracy, but its computational inefficiency makes it unsuitable for real-time applications. Ma J et al. [20] developed an "image subtraction" method that locates defects by comparing pixel differences between standard and test images, demonstrating effectiveness in common areas but exhibiting false positives/misses with complex defects. Melnyk and Tushnytskyy [21] employed K-means clustering [22] to compute defect feature centers, using Euclidean distance thresholds between feature vectors for defect determination, though with limited small-target detection capability. These conventional methods predominantly rely on manual feature engineering. This leads to inherent limitations in generalization capability and sensitivity to image quality.

Deep learning methods primarily consist of two-stage frameworks (exemplified by RCNN series) and single-stage architectures (represented by YOLO series). While two-stage methods achieve higher precision, their computational complexity hinders real-time implementation. Hu B et al. [23] enhanced Faster-RCNN's accuracy through feature pyramid networks, yet inference latency remained problematic. Ding R's team [24] optimized anchor boxes via K-means clustering in Faster-RCNN, improving detection at the cost of increased training computation. Jia Chaoy et al. [25] innovatively replaced VGG with lightweight EfficientNetv2, integrating feature fusion networks and ECA attention mechanisms to strengthen multi-scale feature integration while reducing parameters. Regarding single-stage methods, Adibhatla [26] adopted YOLOv2 for speed improvement but sacrificed precision. Liao X et al. [27] substituted YOLOv4's backbone with MobileNetV3, achieving 40% parameter reduction while meeting real-time requirements. Subsequent work by Adibhatla's team [28] incorporated CSPNet and PANet into YOLOv5, further optimizing the precision-speed balance.

Comparative analysis reveals YOLO series' superior balance between detection accuracy and speed, particularly suitable for edge computing devices. However, existing algorithms still exhibit room for improvement in precision and efficiency for PCB micro-defect detection. The subsequent sections of this paper will focus on targeted optimizations of the YOLOv5 model to enhance its comprehensive performance in PCB defect detection.

## 3. SGB-YOLOv5 Network Design Methodology

### 3.1 Small Object Detection Head (SOD)

The YOLOv5 model initializes nine anchor boxes corresponding to three feature maps of different scales: 80×80, 40×40, and 20×20, with each grid cell in these feature maps utilizing three anchors for prediction. Since the detection performance for micro-targets is closely related to the network's receptive fields, in the original YOLOv5 head architecture, the 80×80 feature map with the smallest receptive field excels in detecting small targets, while the 20×20 feature map with the largest receptive field prioritizes large object recognition. Inspired by the methodology in [29], this study replaces the 20×20 deep-layer feature map with a 160×160 shallow-layer feature map, thereby reducing the receptive field to enhance detection capability for PCB micro-defects.

In earlier YOLOv3/YOLOv4 implementations, anchor box initialization relied on k-means clustering applied to the COCO dataset followed by genetic algorithm optimization. However, the standard k-means algorithm's random initialization of cluster centers may lead to local optima, whereas the k-means++ algorithm mitigates this issue through optimized initial centroid selection. Given that the default anchor box dimensions may not align with specific defect datasets, this study employs k-means++ clustering to recalibrate anchor boxes based on defect characteristics, generating more adaptive parameters for enhanced detection performance.

**Algorithm** K-means++ Cluster

**Input:**

$X = \{x_1, x_2, \cdots, x_n\}$, $K = 6$, $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$

$X$: input dataset, $K$: number of clusters, $C$: initially selected cluster center

**Ouput:**

*cluster_centers* Centroids after reclustering

```
1:  while $X \neq$ empty do
2:  count = 0
3:  for $x_i$ in $X$ do
4:      for $c_i$ in C do
5:          D($x_i$) = min(distance($x_i$, $c_i$))
6:          count += 1
7:          end for
8:      end for
9:  while count = 6
10:     return cluster_centers = Kmeans($X$,6)
11:     end while
12: end while
```

### 3.2 Grouped Convolutional Inversion Blocks (GCI)

Depending on the different ways of convolving the input features, there are commonly used standard convolution, grouped convolution, depth convolution, point-by-point convolution and depth separable convolution. The network structure we designed here mainly uses group convolution to extract features from the input feature maps. The difference between group convolution and standard convolution is that it first groups the different feature maps of the input layer, and then uses different convolution kernels to convolve the feature maps of each group, which reduces the number of parameters in the convolution and the amount of computation, thus improving the training speed of the network. The process of standard convolution and grouped convolution is shown in Fig.2 below.

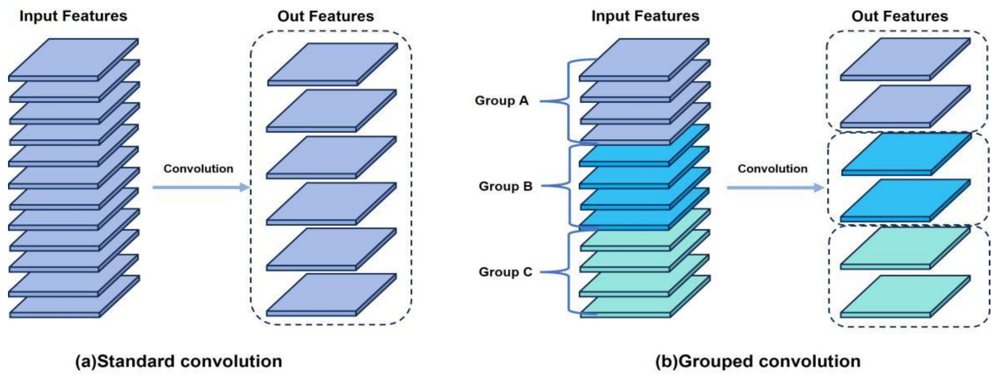

Fig. 2. Standard convolution (a) and grouped convolution (b)

Assuming the size of the input feature map is $C\times H\times W$, the size of the output feature map is $N\times H'\times W'$, the size of the convolution kernel in both standard convolution and grouped convolution is $K\times K$. If the input feature map is divided into $G$ groups, the number of input feature maps in each group is $C/G$, the number of output feature maps in each group is $N/G$, and the size of each convolution kernel is $C/G\times K\times K$. By calculating the number of parameters $P_{\text{Conv}}$ and computation $F_{\text{Conv}}$ for standard convolutional parametric quantities and grouped convolutional parameters $P_{\text{Group}}$ and computation $F_{\text{Group}}$:

$$P_{\text{Conv}} = K\times K\times C\times N \tag{1}$$

$$F_{\text{Conv}} = K\times K\times C\times H'\times W'\times N \tag{2}$$

$$P_{\text{Group}} = K\times K\times \frac{C}{G}\times N \tag{3}$$

$$F_{\text{Group}} = \frac{C}{G}\times K\times K\times H'\times W'\times N \tag{4}$$

From Eq. (1) to Eq. (4), the grouped convolution accounts for the parametric ratio $H_P$ and the computational ratio $H_F$ of the standard convolution, respectively:

$$H_P = \frac{P_{\text{Group}}}{P_{\text{Conv}}} = \frac{K\times K\times \frac{C}{G}\times N}{K\times K\times C\times N} = \frac{1}{G} \tag{5}$$

$$H_F = \frac{F_{\text{Group}}}{F_{\text{Conv}}} = \frac{\frac{C}{G}\times K\times K\times H'\times W'\times N}{K\times K\times C\times H'\times W'\times N} = \frac{1}{G} \tag{6}$$

From Eqs. (5) and (6), it can be seen that the convolution computation of the feature map using grouped convolution is 1/G of the standard convolution both in terms of the number of network parameters and computation, which makes the network model more lightweight and reduces the need for hard computational performance.

In conventional approaches, features extracted by the backbone network are typically processed through 1×1 convolutional kernels for channel dimension reduction before entering the feature fusion network, a method that inevitably leads to feature information loss. To address this limitation, we propose a Grouped Convolutional Inversion (GCI) block. This module employs grouped convolution to process backbone-derived features without dimensionality reduction: 1) The input feature map is partitioned into channel groups, with each group undergoing convolution operations while preserving tensor dimensions. 2) A weight-sharing mechanism enables implicit diffusion of channel-specific pixel information into adjacent spatial regions. This design maintains full channel dimensionality while enhancing feature representation through spatial interactions, effectively mitigating information loss and expanding receptive field coverage to significantly improve micro-target detection. The output feature map retains identical channel dimensions as the input, with its architecture illustrated in Fig.3 and mathematical formulation expressed as:

$$Y_{i,j,k} = \sum_{(m,n)\in\Delta K} \ell_{i,j,m+[K/2],n+[K/2],[kG/C]} X_{i+m,j+n,k} \tag{7}$$

Where $\ell$ is the convolution kernel of the GCI pair fit block, denoted as $\ell \in R^{H\times W\times K\times K\times G}$, where H and W denote the height and width of the feature mapping, respectively, K is the convolution kernel size, and G denotes the number of groups of grouped convolutions. $X$ denotes the input feature tensor of different groups.

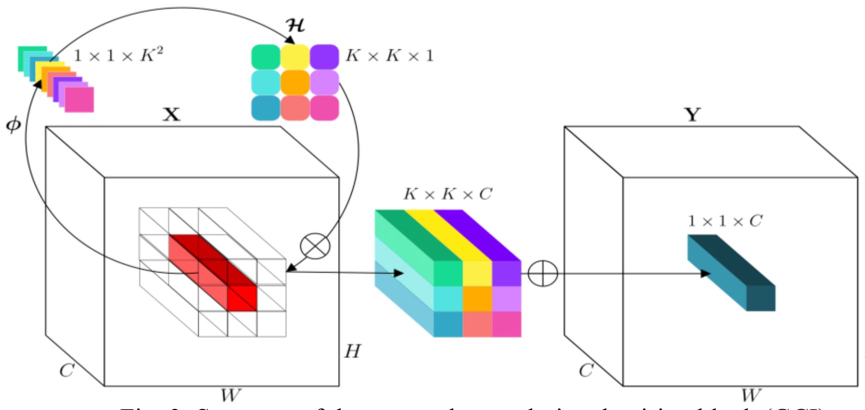

Fig. 3. Structure of the grouped convolutional pairing block (GCI)

### 3.3 Bidirectional Multiscale Feature Pyramid Structure (BMFPN)

YOLOv5 employs a PANet-based multi-scale feature fusion pyramid architecture (illustrated in Fig.4), which combines top-down and bottom-up path aggregation strategies (i.e., FPN+PAN integration) to facilitate feature interactions. While designed to enhance the synergy between spatial details from shallow features and semantic abstractions from deep features, this approach suffers from inefficient feature propagation and computational redundancy in practice.

Specifically:

Shallow features (e.g., 80×80 resolution) retain rich spatial localization details but lack semantic abstraction.

Deep features (e.g., 20×20 resolution) encapsulate high-level semantics but exhibit reduced spatial resolution.

Although PANet theoretically enhances multi-scale detection by integrating FPN (top-down semantic propagation) and PAN (bottom-up positional encoding), it presents critical limitations:

(i) FPN Pathway Deficiency: Upsampling deep features to shallow layers causes significant loss of high-frequency details, degrading small-target feature representation.

(ii) PAN Pathway Inadequacy: When propagating shallow features to deep layers, channel dimension compression in the pyramid structure leads to insufficient cross-level interactions and fails to prioritize feature importance adaptively.

To address these issues, the weighted Bidirectional Feature Pyramid Network (BiFPN) introduces two key innovations:

(i) Bidirectional Cross-Scale Connections: Cyclic top↔bottom feature propagation enhances multi-scale feature capture without computational overhead escalation.

(ii) Learnable Weighted Fusion: A dynamic weighting mechanism optimizes fusion weights across feature levels during training, achieving superior multi-modal feature integration.

As shown in Fig.4, the BiFPN architecture effectively mitigates information decay and redundancy inherent in traditional pyramid structures, significantly improving detection accuracy for PCB micro-defects.

BiFPN structural feature fusion is computed as follows, here the P6 output is used as an example:

$$P_6^{td} = Conv\left(\frac{\omega_1 \cdot P_6^{in} + \omega_2 \cdot \mathrm{Re}\,size(P_7^{in})}{\omega_1 + \omega_2 + \varepsilon}\right) \tag{8}$$

$$P_6^{out} = Conv\left(\frac{\omega_1^{'} \cdot P_6^{in} + \omega_2^{'} \cdot P_6^{td} + \omega_3^{'} \cdot \mathrm{Re}\,size(P_5^{out})}{\omega_1^{'} + \omega_2^{'} + \omega_3^{'} + \varepsilon}\right) \tag{9}$$

Inspired by the idea of BiFPN feature pyramid structure, this paper proposes a bi-directional multi-scale feature pyramid network structure (BMFPN), which firstly fuses the feature maps extracted from different scales in the backbone network to realize the full utilization of the features, and then fuses the feature maps extracted from the same sizes in the backbone network through the weighted fusion of the top-down path propagation network to enhance the ability of the network to learn more features of different scales and achieve better detection results. The structure of the Bidirectional Multiscale Feature Pyramid Network (BMFPN) is shown in Fig.5 below.

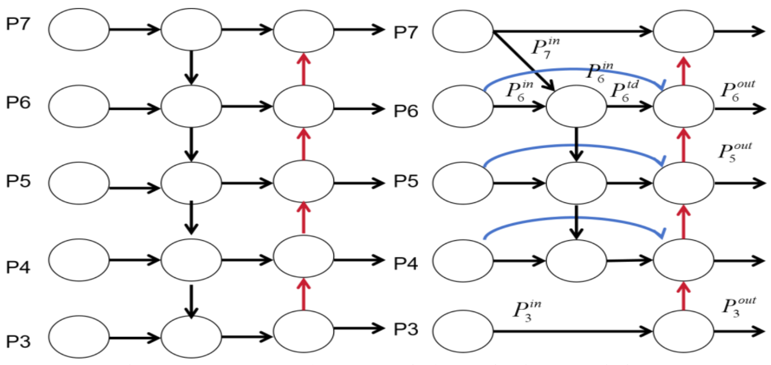

Fig. 4. Structure of PANet (left) and BiFPN (right)

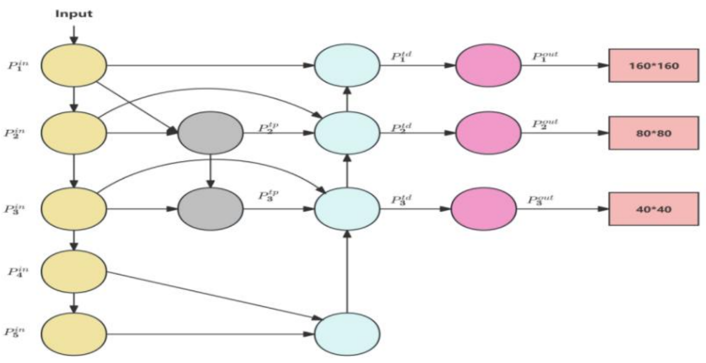

Fig. 5. The structure of the Bidirectional Multiscale Feature Pyramid Network

As can be seen from Fig. 5, the network structure retains the bidirectional feature transfer of the BiFPM network while passing the feature maps $P_4^{in}$ extracted from the backbone network as input nodes into the feature fusion network and fusing feature maps of different sizes by up-sampling. In addition, the bidirectional multi-scale feature pyramid structure (BMFPN) adopts a two- and three-scale feature-weighted fusion approach, and for the 160*160 detection head, the features extracted by the backbone network are directly fused with the feature maps obtained by upsampling, which are finally used as the output features of the 160*160 scale anchor frame. As for the 80*80 and 40*40 scale anchor frames, a three-scale weighted fusion is used, i.e., it contains three components: the feature map extracted from the backbone, and the feature map obtained from the neck via top-down path and bottom-up path.

## 4. Experimental Environment Configuration and Simulation

### 4.1 Experimental Environment

The experimental environment used is shown in Table 1 below.

*Table 1*

**Experimental environment configuration**

| | |
|---|---|
| Operating System | Window11 |
| CPU | AMD Ryzen 7 7840H 3.80GHz |
| GPU | NVIDIA GeForce RTX4060 Laptop |
| Memory | 8GB |
| CUDA Versions and Acceleration Libraries | CUDA11.8+cudnn8.9.7 |
| Deep Learning Framework | Pycharm |

## 4.2 Dataset Preparation

The dataset [30] used in this experiment is the publicly available PCB defect dataset from the Intelligent Robotics Laboratory of Peking University. It includes six types of defects: open_circuit, short, mouse_bite, missing_hole, spurious_copper, and spur, with a total of 1,386 images. For this study, I selected 693 images as the benchmark dataset and used the remaining 693 samples for testing. In order to improve the generalization ability of the model and to prevent premature model fitting during training, the original dataset was augmented by employing data enhancement techniques (panning, rotating, cropping, mirroring, adding noise, and adjusting the brightness of the image, final dataset was augmented to 3500 samples and randomly divided into training and validation sets in the ratio of 9:1. The number of various types of defects in the augmented dataset is shown in Table 2 below.

*Table 2*

**Number of defects in each category before and after dataset enhancement**

| Defect type | Number of defects by category(before) | Number of defects by category(after) |
|---|---|---|
| Open_circuit | 116 | 580 |
| short | 116 | 580 |
| Mouse_bite | 115 | 580 |
| Missing_hole | 115 | 605 |
| Spurious_copper | 116 | 580 |
| spur | 115 | 575 |
| total | 693 | 3500 |

## 4.3 Evaluation Indicators

Precision rate (P): refers to the proportion of the number of samples predicted to be positive to the total number of actual positive samples when the actual samples are positive, and its mathematical expression is:

$$P = \frac{TP}{TP + FP} \tag{10}$$

Recall (R): refers to the proportion of all positive sample species that are predicted to be positive, and its mathematical expression is:

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{11}$$

Where TP denotes the number of positive samples correctly detected, FP denotes the number of negative samples correctly detected as positive samples, and FN denotes the number of negative samples incorrectly detected as positive samples.

Average precision (AP): refers to the average precision, expressed by the area enclosed by the two indicators of accuracy and recall, its mathematical expression is:

$$AP = \int_0^1 P(R)\mathrm{d}R \tag{12}$$

Intersection and integration ratio (IOU): refers to the degree of overlap between the predicted frame and the real frame by calculating, the larger the value represents a better localization of the network to the target, its mathematical expression is:

$$IOU = \frac{PD \cap GT}{PD \cup GT} \tag{13}$$

Where PD denotes the prediction anchor and GT denotes the true anchor.

Mean Average Precision (mAP): refers to the summed average of the precision of each class of defects, its mathematical expression is:

$$\mathrm{mAP} = \frac{\sum_{i=1}^{n} AP}{n} \tag{14}$$

Where i denotes the average accuracy value of a particular class of defects, and in the dataset used in this experiment there are 6 classes of defects, so n is equal to 6.

### 4.4 Ablation Experiments

To validate the effectiveness of the improvement methods proposed in Section 1, this study conducted a series of comparative experiments based on the original YOLOv5s model (Baseline) under consistent training environments and parameters. Nine modified models were sequentially constructed: Model1 added a 160×160 shallow detection head to Baseline. Model2 optimized anchor clustering via k-means++. Model3 incorporated the Grouped Convolutional Inversion (GCI) block. Model4 integrated the weighted Bidirectional Multi-scale Feature Pyramid (BiFPN). Models5-7 combined two improvement strategies each. Model8 fused three optimizations (k-means++, GCI, and BiFPN). And Model9 (SGB-YOLOv5s) synthesized all enhancements. Experimental results (detailed in Table 3) demonstrated progressive improvements in per-class Average Precision (AP), mean Average Precision at 50% IoU (mAP@50), model size, and parameter count.

Notably, the comprehensively optimized Model9 achieved significant mAP@50 gains over Baseline while retaining lightweight characteristics, confirming the synergistic efficacy of multi-strategy integration.

*Table 3*

**Ablation experiments**

| Model | AP/% | | | | | | mAP@ 50/% | Size /Mb | Para meter s/M |
|---|---|---|---|---|---|---|---|---|---|
| | Open_ circuit | short | Mous e_bite | Mis sing _hol e | Spuri ous_c opper | spur | | | |
| Basicline | 98.13 | 99.49 | 99.24 | 99.5 | 99.45 | 95.9 | 98.26 | 26.8 | 7.03 |
| Model1 | 99.49 | 99.49 | 99.23 | 99.1 | 99.5 | 97.79 | 99.1 | 26.78 | 7.02 |
| Model2 | 99.45 | 99.49 | 99.05 | 99.5 | 99.5 | 97.93 | 99.16 | 26.84 | 7.03 |
| Model3 | 99.17 | 99.49 | 99.06 | 99.5 | 99.5 | 96.4 | 98.85 | 28.4 | 7.45 |
| Model4 | 99.21 | 99.5 | 99.43 | 99.5 | 99.48 | 95.9 | 98.84 | 27.09 | 7.09 |
| Model5 | 99.45 | 99.5 | 99.21 | 99.5 | 99.5 | 97.96 | 99.19 | 28.4 | 7.45 |
| Model6 | 99.46 | 99.5 | 99.36 | 99.5 | 99.5 | 97.84 | 99.19 | 27.09 | 7.1 |
| Model7 | 98.92 | 99.5 | 99.26 | 99.5 | 99.5 | 96.54 | 98.87 | 28.78 | 7.54 |
| Model8 | 99.61 | 99.67 | 98.97 | 99.6 | 99.67 | 98.01 | 99.2 | 28.78 | 7.54 |
| Model9 | 99.49 | 99.5 | 99.3 | 99.5 | 100 | 99.03 | 99.39 | 28.6 | 7.5 |

Experimental results demonstrate significant performance improvements across all enhancement strategies. Model1, which replaced the 20×20 detection head with a 160×160 shallow head, increased AP values for open-circuit, excess copper, and burr defects, elevating the overall mAP@50 from 98.26% to 99.1%, validating the efficacy of shallow features for micro-defect detection. Model2, employing k-means++ for anchor optimization, improved AP for all defect types except rodent bites, with a 0.92% mAP@50 gain, confirming that adaptive anchor parameters better align with PCB defect characteristics.

Model3's integration of the Grouped Convolutional Inversion (GCI) block enhanced AP for all defects except short circuits, achieving a 0.49% mAP@50 increase, highlighting its capability to preserve channel-wise information. Model4's adoption of BiFPN boosted open-circuit AP from 98.13% to 99.21% with only 1.08% and 0.85% increases in model size and parameters, respectively, underscoring its efficient feature fusion.

Model5's combined k-means++ and GCI strategy elevated open-circuit and burr AP to 99.45% and 97.96%, surpassing the standalone k-means++ model by 0.16% and 0.06% for rodent bites and burrs, demonstrating synergistic optimization. Model6 maintained short-circuit AP at 99.5% while improving other defects, particularly open-circuit, rodent bites, and burrs.

Though Model7's fusion of GCI and BiFPN slightly reduced open-circuit AP, it enhanced other defect categories. Model8's comprehensive integration of three strategies achieved a 99.2% mAP@50 (up from 98.26%) with 7.39% and 7.25%

increases in model size and parameters, revealing global optimization potential. Ultimately, Model9 (SGB-YOLOv5s) attained 100% AP for excess copper defects and a 1.15% mAP@50 improvement (99.39%) with only 1.8MB and 0.47M increases in model size and parameters, achieving optimal precision-lightweight balance.

### 4.5 Comparative Experiments

To comprehensively evaluate the performance advantages of the proposed SGB-YOLOv5 network, comparative experiments were conducted against mainstream detection models (YOLOv3, YOLOv3-tiny, YOLOv4, YOLOv5s, YOLOv7-tiny, YOLOv8n, and YOLO11) using the publicly available PCB defect dataset from Peking University’s Intelligent Robotics Laboratory. As detailed in Table 4, model performance was assessed through five metrics: Precision (P), Recall (R), mean Average Precision (mAP@50), parameter count (Parameters), and inference speed (Frames Per Second, FPS). Comparative curves of mAP@50 and mAP@50:95 across models (Fig.6-7) visually demonstrate multi-scale detection capability differences. Experimental results confirm that SGB-YOLOv5 achieves superior precision-efficiency balance compared to state-of-the-art models.

*Table 4*

**Comparative experiments**

| Model | P/% | R/% | mAP@50/% | mAP@50:95/% | Parameters/M | FPS |
|---|---|---|---|---|---|---|
| YOLOv3 | 96.7 | 91.6 | 94.7 | 53.7 | 8.68 | 158.5 |
| YOLOv3-tiny | 97.5 | 91.4 | 95.1 | 53.5 | 8.68 | **158.7** |
| YOLOv5s | 98.2 | 97.3 | 98.26 | 70.98 | 7.03 | 73 |
| YOLOv7-tiny | 94.7 | 85.3 | 89.9 | **46.8** | 6.03 | 49.02 |
| YOLOv8n | 96.7 | 88.4 | 93.7 | 59.1 | 3.0 | 101 |
| YOLO11 | 95.3 | 88.6 | 93.1 | 58.7 | **2.58** | 101 |
| SGB-YOLOv5s | **99.4** | **98.73** | **99.39** | **76.85** | 7.49 | 66.7 |

Experimental results (Table 4) reveal that while YOLOv3 and YOLOv3-tiny achieve optimal inference speed (FPS), their core precision metrics—including accuracy (P), recall (R), and mAP@50—remain suboptimal. Although YOLOv7-tiny exhibits low parameter counts, its mAP@50 (89.9%) and mAP@50:95 (46.8%) are notably inferior, coupled with the lowest recall rate among all models. YOLOv8n demonstrates advantages in lightweight design and speed but leaves room for precision improvement. YOLO11, despite having the fewest parameters, underperforms significantly compared to YOLOv5s in both accuracy and recall, justifying our selection of YOLOv5s as the baseline. In contrast, SGB-YOLOv5s achieves optimal values across all four key metrics (P, R, mAP@50, and mAP@50:95). Although its FPS does not peak, it sufficiently meets real-time detection requirements. Crucially, this model comprehensively surpasses the original YOLOv5s with negligible parameter increases, offering an efficient solution for edge device deployment.

As illustrated in Fig.6-7, mAP curves further validate these findings: YOLOv7-tiny exhibits severe overfitting with the weakest detection performance; YOLOv3 and YOLOv3-tiny show overlapping curves indicating comparable performance; YOLOv5s demonstrates superior precision and smoother convergence compared to the former three; YOLOv8n and YOLO11 achieve similar accuracy levels but lag behind YOLOv5s. The enhanced SGB-YOLOv5s attains peak values in both mAP@50 and mAP@50:95 metrics, with accelerated convergence and significant precision advantages, fully validating the efficacy of our algorithmic improvements.

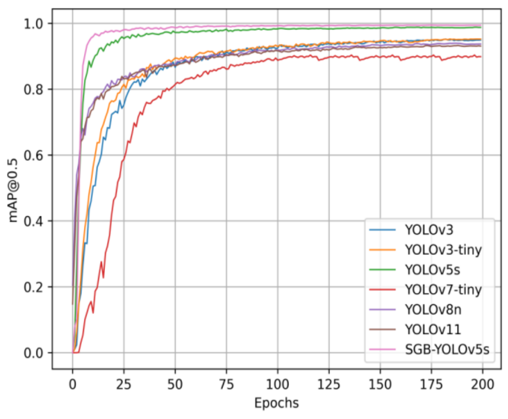

Fig. 6. Curves mAP@50 different models

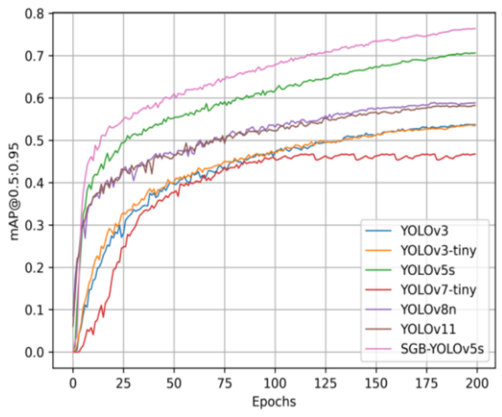

Fig. 7. Curves of different model mAP@50:95

### 4.6 Detection Effect Test

To validate the detection performance of YOLOv5s and SGB-YOLOv5s, Fig.8 demonstrates the comparative testing results of both models on six types of defects in the test dataset: open_circuit, short, mouse_bite, missing_hole, spurious_copper, and spur. The first column displays ground truth annotations, the second column shows detection results from YOLOv5s, and the third column illustrates outputs from SGB-YOLOv5s. Experimental observations reveal that while YOLOv5s achieves defect localization, its confidence scores are notably lower than those of the enhanced model. Specifically, all defect types demonstrated varying degrees of improvement in detection performance when using SGB-YOLOv5s, among which mouse_bite defects showed the most significant enhancement. Furthermore, while the baseline YOLOv5s model exhibited both missed detections and false positives for spur defects, the improved model achieved accurate and error-free detection.

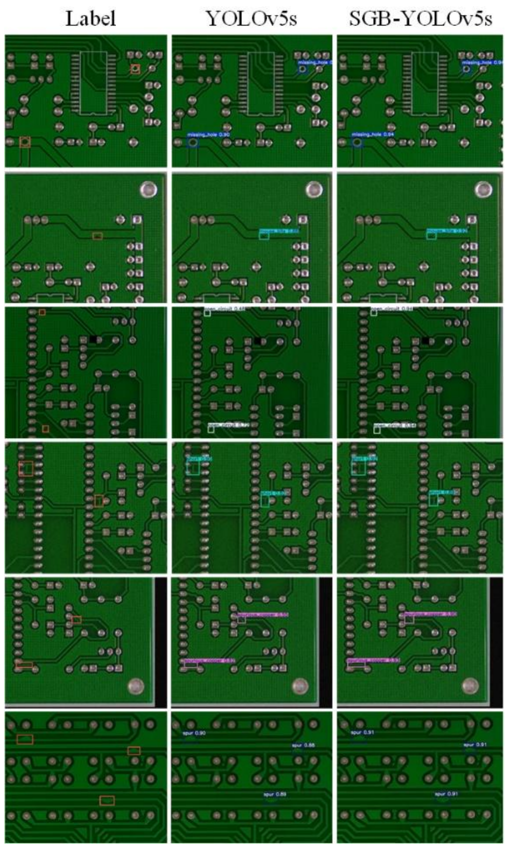

Fig.8. Detection effect of YOLOv5s and SGB-YOLOv5s

## 5. Conclusion

This paper proposes a PCB defect detection algorithm based on the SGB-YOLOv5s model. The algorithm first designs a 160×160-scale detection head and employs the k-means++ clustering algorithm to better adapt to the feature size distribution of PCB defects. Additionally, between the backbone network and the neck feature fusion network, grouped convolution cross-block modules replace conventional 1×1 convolutions. This approach preserves feature channel dimensions while enabling effective feature transmission, with ablation studies demonstrating its ability to prevent feature loss and enhance detection accuracy without increasing network parameters. Furthermore, a weighted bidirectional multi-scale feature pyramid network (BMFPN) is implemented to achieve simultaneous fusion of multi-scale features while improving computational efficiency. Experimental results show that SGB-YOLOv5s achieves a state-of-the-art mAP@50 of 99.39%, surpassing existing mainstream models in detection accuracy. This breakthrough successfully resolves high-precision detection challenges and provides a novel solution for practical PCB defect detection in industrial production.

However, real-world application scenarios present challenges such as complex backgrounds and environmental noise that may compromise detection performance. To address these issues, the study employs advanced data augmentation techniques to simulate authentic PCB defect images captured in practical environments, thereby enhancing model robustness. While the current implementation achieves a detection speed of 66.7 FPS - sufficient for real-time requirements - future research will focus on optimizing network architecture to simultaneously improve inference speed and maintain high accuracy, ultimately delivering a comprehensive solution that balances precision and efficiency for industrial deployment.

**Acknowledgement**

This work was supported in part by the Putian Science and Technology Plan Project under Grant 2023GZ2001PTXY19, in part by the Startup Fund for Advanced Talents of Putian University under Grant 2023015, and in part by the Putian Science and Technology Plan Project under Grant 2023GJGZ003.

## REFERENCES

[1] *F. Liu*, "Research on PCB defect detection technology based on deep learning", Master Thesis, Heilongjiang University, 2023.

[2] *L.Y. Zhao, and Y.Q. Wu*, "Research progress of surface defect detection methods based on machine vision", Chinese Journal of Scientific Instrument, **Vol.**43, no.1, 2022, pp. 198-219.

[3] *H. Peng, B.W. Zhou, and W.Q. Ouyang,* "Lightweight PCB defect detection research based on dual-channel attention", Optoelectron Laser, **Vol.**35, no.5, 2024, pp. 506-515.

[4] *Y. Du, J. Chen, H. Zhou, et al*, "An automated optical inspection (AOI) platform for three-dimensional (3D) defects detection on glass micro-optical components (GMOC)", Optics Communications, **Vol.** 545, Oct. 2023, pp. 129736.

[5] *V. Sani, M.V.V. Kantipudi, and P. Meduri,* "Enhanced SSD algorithm-based object detection and depth estimation for autonomous vehicle navigation", International Journal of Transport Development and Integration, **Vol.**7, no.4, May 2023, pp. 341-351.

[6] *L. Zhang, T. Chen, J.B. Yu, et al,* "Lightweight PCB Surface Defect Detection Algorithm", Journal of Beijing University of Posts and Telecommunications, **Vol.**47, no.2, 2024, pp. 38-44.

[7] X. Liao, J. Zhang, LÜ S. and, "PCB defect detection method combining shallow features and attention mechanism", Computer Integrated Manufacturing Systems, **Vol.**30, no.3, Mar. 2024, pp. 1092-1104.

[8] *Y. LeCun, L. Bottou, Y. Bengio, et al,* "Gradient-based learning applied to document recognition", Proceedings of the IEEE, **Vol.**86, no.11, Nov. 1998, pp. 2278-2324.

[9] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection", 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR 2005), IEEE, 2005, pp. 886-893.

[10] *K. He, X. Zhang, S. Ren, et al,* "Spatial pyramid pooling in deep convolutional networks for visual recognition", IEEE transactions on pattern analysis and machine intelligence, **Vol.**37, no.9, Jan. 2015, pp. 1904-1916.

[11] *J. Redmon, S. Divvala, R. Girshick, et al,* “You only look once: Unified, real-time object detection”, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779-788.

[12] *W. Liu, D. Anguelov, D. Erhan, et al,* “Ssd: Single shot multibox detector”, 14th European Conference on Computer Vision (ECCV 2016), Springer, 2016, pp. 21-37.

[13] *J. Redmon, and A. Farhadi,* “Yolov3: An incremental improvement”, arXiv, 2018, 1804.02767.

[14] *A. Bochkovskiy, C.Y. Wang, and H.Y.M. Liao,* “Yolov4: Optimal speed and accuracy of object detection”, arXiv, 2020, 2004.10934.

[15] *S. Li, Y. Li, Y. Li, et al,* “Yolo-firi: Improved yolov5 for infrared image object detection”, IEEE access, **Vol.**9, Oct. 2021, pp. 141861-141875.

[16] *Y. Huang, W. Tan, L. Li, et al,* “Wfre-yolov8s: a new type of defect detector for steel surfaces”, Coatings, **Vol.**13, no.12, Nov. 2023, pp. 2011.

[17] *A. Wang, H. Chen, L. Liu, et al,* “Yolov10: Real-time end-to-end object detection”, NeurIPS 2024, MIT Press, 2024, pp. 107984-108011.

[18] *M. Bakirci, P. Dmytrovych, I. Bayraktar, et al*, “Multi-Class Vehicle Detection and Classification with YOLO11 on UAV-Captured Aerial Imagery”, 2024 IEEE 7th International Conference on Actual Problems of Unmanned Aerial Vehicles Development (APUAVD), IEEE, 2024, pp.191-196.

[19] *J. Onshaunjit, and J. Srinonchat,* “Algorithmic Scheme for Concurrent Detection and Classification of Printed Circuit Board Defects”, Computers, Materials & Continua, **Vol.**71, no.1, 2022, pp. 355-367.

[20] *J. Ma*, “Defect detection and recognition of bare PCB based on computer vision”, 2017 36th Chinese Control Conference (CCC), IEEE, 2017, pp. 11023-11028.

[21] *R.A. Melnyk, and R.B. Tushnytskyy*, “Detection of defects in printed circuit boards by clustering the etalon and defected samples”, 2020 IEEE 15th international conference on advanced trends in radioelectronics, telecommunications and computer engineering (TCSET), IEEE, 2020, pp. 961-964.

[22] *M.S. Yang, and I. Hussain,* “Unsupervised multi-view K-means clustering algorithm”, IEEE Access, **Vol.**11, Feb. 2023, pp. 13574-13593.

[23] *B. Hu, and J. Wang*, “Detection of PCB surface defects with improved faster-RCNN and feature pyramid network”, IEEE Access, **Vol.**8, Jun. 2020, pp. 108335-108345.

[24] *R. Ding, L. Dai, G. Li, et al,* “TDD-net: a tiny defect detection network for printed circuit boards”, CAAI Transactions on Intelligence Technology, **Vol.**4, no.2, May 2019, pp. 110-116.

[25] *J. Yin, Y. LÜ, K. Suo, et al*, “PCB defect detection algorithm based on EfficientNetv2”, Journal of Computer-Aided Design & Computer Graphics, **Vol.**36, Jul. 2024, pp. 1-10.

[26] *V.A. Adibhatla, H.C. Chih, C.C. Hsu, et al*, “Defect detection in printed circuit boards using you-only-look-once convolutional neural networks”, Electronics, **Vol.**9, no.9, Sep. 2020, pp. 1547.

[27] *X. Liao, S. Lv, D. Li, et al*, “Yolov4-mn3 for pcb surface defect detection”, Applied Sciences, **Vol.**11, no.24, Dec. 2021, pp. 11701.

[28] *V.A. Adibhatla, H.C. Chih, C.C. Hsu, et al*, “Applying deep learning to defect detection in printed circuit boards via a newest model of you-only-look-once”, Mathematical Biosciences and Engineering, **Vol.**18, no.4, May 2021, pp. 4411-4428.

[29] *Z. Li, and X. Liu,* “PCB defect detection algorithm based on lightweight YOLOv8n network”, Electronic Measurement Technology, **Vol.**47, no.4, Feb. 2024, pp. 120-126.

[30] *L. Dai*, “PKU-Market-PCB dataset”, Peking University, 2019. Available online: https://robotics.pkusz.edu.cn/resources/dataset/