scientific reports

OPEN

SEPDNet: simple and effective PCB surface defect detection method

Du Lang[1]✉ & Zhenzhen Lv[2]

Replacing time-consuming and costly manual inspections on production lines with efficient and accurate defect detection algorithms for Printed Circuit Boards (PCBs) remains a significant challenge. Current PCB defect detection methods are typically optimized using existing models such as YOLO and Faster R-CNN to enhance detection accuracy. In this study, we analyse a PCB defect dataset characterized by small targets and a concentrated size distribution. SEPDNet (Simple and Effective PCB Defect Detection Network) is designed for the characteristics of the dataset, only one detection head is used, which reduces the number of parameters and improves the detection performance at the same time. SEPDNet uses RepConv (Re-parameterizable Convolution) to improve the backbone representation ability, and FPN (Feature Pyramid Network) is used in the neck part to simplify the model. SEPDNet has fewer than 30% of the parameters of YOLOv9u-s, yet achieves an improvement of 0.025 in the F1 score, 2.7% in mAP50, and 3.8% in mAP50:95 compared to YOLOv9u-s. We propose the method of designing the model according to the characteristics of the dataset. Our experiments show that customizing the model design according to dataset characteristics can achieve strong performance with a simplified structure and fewer parameters.

Keywords Defect detection, Printed Circuit Boards, PCB defects, Small object detection

Printed Circuit Boards (PCBs) occupy a pivotal role in electronic devices, serving the dual purpose of providing mechanical support and facilitating electrical interconnections between various components. PCBs are ubiquitously employed across a wide range of electronic devices, including 3C products, household appliances, automotive electronics, and numerous other sectors. The performance of electronic systems is largely contingent on PCB quality. Consequently, PCB manufacturers must ensure their products meet stringent standards of quality, precision, and reliability. Thus, implementing rigorous quality control measures throughout the PCB manufacturing process, along with effective defect detection, is crucial. Failure to promptly and accurately detect recurring defects can result in the rejection of numerous PCBs, leading to substantial waste and considerable financial losses[1].

Traditional manual inspection methods for PCB defects are inherently vulnerable to external environmental factors, which greatly reduce the efficiency of the defect detection process[2]. Furthermore, the identification of minute defects can cause visual fatigue in inspectors, resulting in increased rates of misclassification[3]. To address these challenges, researchers have effectively integrated machine learning techniques into PCB defect detection, leading to significant advancements. Wang et al.[4] developed an automated detection algorithm tailored specifically for PCB pinholes, utilizing machine learning techniques. The algorithm can efficiently detect pinhole defects as small as 2 mm within 10 s, demonstrating its effectiveness in rapid and precise defect identification. Yuk et al.[5] advanced the field further by applying PCB defect detection methods based on a combination of accelerated robust features and the random forest algorithm. By generating weighted kernel density estimation (WKDE) mappings, which model feature density using weighted probabilities, they achieved precise localization of defect concentration areas, significantly improving overall detection accuracy.

While traditional image processing methods for PCB defect detection have achieved moderate accuracy, they are frequently hindered by time-intensive procedures and increased sensitivity to environmental conditions and input image quality[6]. Recent rapid advancements in deep learning (DL) and computer vision have driven a paradigm shift, establishing DL and convolutional neural networks (CNNs) as dominant techniques for PCB defect detection[7]. These contemporary methods harness deep neural networks to automatically learn and extract discriminative features from complex image data, markedly improving both the efficiency and robustness of the detection process. By integrating DL and CNNs, researchers have successfully addressed the limitations of traditional methods, achieving faster detection rates and greater resilience to environmental fluctuations and image quality degradation.

[1]School of Information and Design, Zhejiang Industry Polytechnic College, Shaoxing 312000, China. [2]Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110167, China. ✉email: 20230037@zjipc.edu.cn

Deep learning-based methods for PCB defect detection are broadly categorized into two types: single-stage detection algorithms and two-stage detection algorithms. Single-stage detection algorithms directly identify defects from images without requiring intermediate steps[8–10]. Two-stage detection algorithms first generate candidate bounding boxes and then detect defects within these regions. Two-stage detection algorithms, exemplified by R-CNN (Region-based Convolutional Neural Networks), Fast R-CNN, Faster R-CNN, and Cascade R-CNN[11–14], generally offer higher detection accuracy but at the cost of slower processing speeds compared to single-stage algorithms. Single-stage detection algorithms are exemplified by the YOLO (You Only Look Once) series and SSD (Single Shot MultiBox Detector)[15]. Although single-stage detection algorithms typically offer lower detection accuracy compared to two-stage algorithms, they are faster and have been widely adopted in industrial defect detection.

Zhang et al.[16] achieved impressive detection results with a cost-sensitive residual convolutional neural network, specifically developed for detecting PCB appearance defects. However, the model is marked by its high complexity and a large number of parameters. Ding et al.[17] introduced TDD-net (Tiny Defect Detection Network), a Faster R-CNN-based model specifically designed to detect small target defects in PCBs. Despite its high accuracy, the model's large size limits its practicality for deployment on embedded devices. Xuan et al.[18] proposed a YOLOX-based detection algorithm incorporating coordinate attention, achieving strong performance in PCB defect detection. However, the model's size of 379 MB presents challenges for specific application scenarios. Wu et al.[19] introduced ghost conv, SE (Squeeze-and-Excitation), and CBAM (Convolutional Block Attention Module) into YOLOv5, named GSC (Ghost SE CBAM) YOLOv5, a deep learning detection approach combining lightweight networks with a dual-attention mechanism to address the challenge of small target detection. However, the complexity of the proposed attention mechanism, along with its speed limitations, still requires further optimization. Bowei et al.[20] proposed a detection algorithm, based on YOLOv5, incorporating depth-wise convolution, an attention mechanism, and a BiFPN (Bidirectional Feature Pyramid Network), to enhance detection accuracy. Wei et al.[21] proposed Transformer-YOLO, utilizing the Swin Transformer to enhance the model's feature extraction capabilities, thereby improving detection accuracy. However, the added computational complexity reduces detection speed. Junlong et al.[22] introduced various optimization methods based on YOLOv5 and developed PCB-YOLO, which slightly decreases detection speed but enhances accuracy.

In industrial applications, PCB defect detection algorithms must carefully balance detection speed and accuracy to meet operational requirements. PCB defect areas are typically small, and to ensure high detection accuracy, they are often upscaled to higher resolutions before detection. However, increasing resolution inevitably increases computational complexity and slows down detection speed.

Existing PCB defect detection methods are typically optimized based on baseline models, with performance enhancements achieved through the incorporation of techniques from other studies. Although these methods can improve detection performance, they lack in-depth analysis of the specific characteristics of PCB defects and fail to effectively tailor model design to these features. The motivation behind our research stems from a central question: Are baseline-optimized methods truly the most effective solution for PCB defect detection? We analysed the PCB defect detection dataset and, based on its characteristics, developed a streamlined defect detection model SEPDNet. SEPDNet avoids the use of specialized optimizer methods and achieves high detection accuracy with only a minimal number of parameters. The main contributions of this paper are as follows:

(1) We abandoned the conventional baseline optimization approach and designed SEPDNet from scratch, achieving exceptional performance in PCB defect detection.
(2) SEPDNet uses RepConv to improve the feature representation of the backbone without affecting the inference speed, uses the FPN structure to simplify the neck module and reduce the model parameters, and uses only a single detector head to match the dataset's small target and concentrated distribution.
(3) We analysed the PCB defect dataset and developed a model tailored to its specific characteristics. Our model uses less than 30% of the parameters in YOLOv9u-s, yet improves the F1 score by 0.025, mAP50 by 2.7%, and mAP50:95 by 3.8%.

Related work
YOLO framework
Real-time object detection has consistently been a central focus in computer vision research, striving to accurately predict object categories and positions within an image with minimal latency. The YOLO models have garnered growing acclaim for their adept equilibrium between performance and efficiency[23]. The initial YOLO model pioneered a revolutionary approach to object detection by formulating it as a unified regression problem[24]. Figure 1 illustrates the general framework of the YOLO series of object detection algorithms, which typically consists of three components: the backbone, the neck, and the head. The backbone is responsible for extracting input feature information, the neck performs feature fusion, and the head generates the final prediction results.

Unlike traditional methods that applied classifiers to multiple regions of interest, YOLOv1 predicted bounding boxes and class probabilities directly from full images in one pass[25]. YOLOv2, also known as YOLO9000[26], introduced several enhancements over its predecessor. One of the key innovations was the use of anchor boxes, which improved the model's ability to detect objects at different scales. Additionally, YOLOv2 incorporated batch normalization and high-resolution classifiers, which helped improve both accuracy and generalization. YOLOv3 built upon the improvements of YOLOv2 by introducing a more complex and capable architecture called Darknet-53, which included 53 convolutional layers with residual blocks[27]. This allowed the model to better capture spatial hierarchies in images. YOLOv4 focused on balancing speed and accuracy, making it highly suitable for real-time applications[28]. The model introduced the Cross-Stage Partial (CSP) network, which improved learning efficiency by reducing the computational bottleneck and enhancing gradient flow through the network. YOLOv5 introduced new data augmentation techniques, such as mosaic

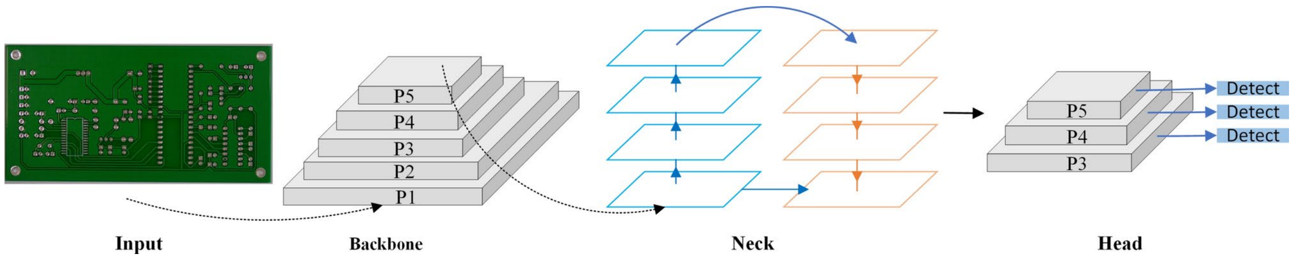

Fig.1. The architecture of the YOLO series.

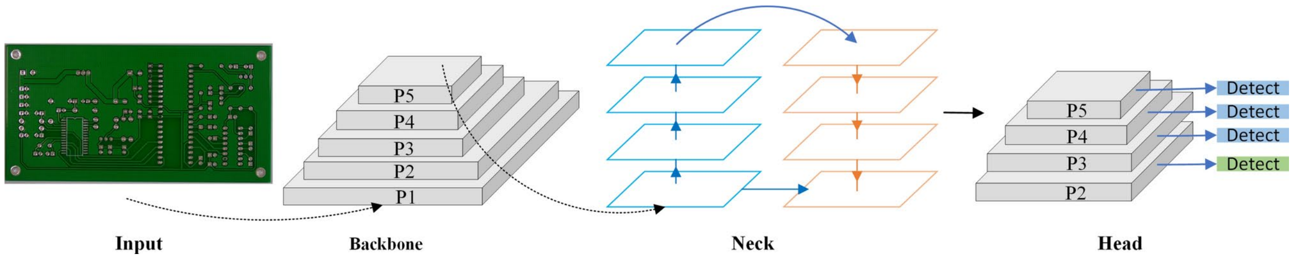

Fig. 2. The YOLO architecture for adding small target detection heads.

augmentation, which helped improve the model's robustness by combining four training images into one, and adaptive anchor computation to better fit the dataset's specific characteristics[29]. YOLOv6 was developed with a focus on deployment optimization. It incorporated techniques such as RepVGG () blocks, which simplify network structure during inference while retaining high performance[30]. YOLOv7 introduced a new direction in model efficiency by integrating ideas from EfficientNet and other modern architectures[31]. It featured compound scaling, which balanced model depth, width, and resolution to achieve better performance on diverse hardware platforms. YOLOv8 included significant advancements with the introduction of C2f. and SPPF modules by recent developments in convolutional architectures[32]. YOLOv8 boasts a balanced approach between accuracy and speed, introducing multiple model sizes (e.g., yolov8n, yolov8s) to cater to diverse requirements. It employs Darknet53 as the backbone and PANet for enhanced feature fusion, achieving robust performance across image classification, object detection, and instance segmentation tasks. YOLOv9 further refines the architecture with PGI (Programmable Gradient Information) and GELAN (Generalized Efficient Layer Aggregation Network) and proposes SPPELAN and RepNCSPELAN4 modules, a lightweight yet powerful design that surpasses existing real-time detectors on MS COCO, particularly excelling in small object detection[33]. YOLOv10 revolutionizes the field by eliminating Non-Maximum Suppression (NMS), adopting a dual-head architecture with a continuous double assignment strategy, and significantly boosting inference speed while maintaining competitive accuracy[34].

Small-object detection

Algorithms developed for general object detection often exhibit suboptimal performance on high-resolution images featuring small objects. Copy-pasting technique oversamples images containing small objects by generating multiple copies of these objects[35]. However, this augmentation necessitates pixel-level labeling, which is not directly compatible with object detection datasets and incurs additional labeling requirements. SSD-MSN extracts richer features of small objects from enlarged regions cropped from the original image[36]. While the additional features enhance detection performance, the selection of regions for enlargement introduces a computational burden. JCS-Net has been proposed for small-scale pedestrian detection, integrating both classification and super-resolution tasks within a unified framework[37]. Although this method improves detection accuracy, the integration of multiple distinct models can significantly decrease detection speed. SAHI (Slicing Aided Hyper Inference) segments input images into overlapping patches, which results in relatively larger pixel areas for small objects compared to the images inputted into the network[38]. Ultimately, the overlapping prediction results are consolidated back into the original image size. SAHI can enhance the accuracy of small target detection; however, its speed is influenced by the number of patches into which the image is divided.

While all the aforementioned methods enhance the detection accuracy of small targets, their practical application in industry is hindered by the requirement for additional labeling or compromised detection speed. To mitigate computational demands and enhance detection speed, object detection models typically operate at resolutions of 1/8, 1/16, and 1/32. During the downsampling process, the model loses a portion of feature information, which particularly impacts the detection of small targets. Commonly employed methods to enhance the accuracy of small target detection in industrial settings include increasing the input image resolution or incorporating an additional detection head at the low-resolution feature map of the model. Both approaches enhance small target detection results by preserving critical feature information. Figure 2 depicts the YOLO architecture with the integration of a dedicated small target detection head. In industrial settings, the detection accuracy for small targets is enhanced by augmenting both the small-target detection head and the input image resolution, provided that detection speed requirements are met.

Methodologies
Dataset analysis
The PCB defect dataset utilized in this study was sourced from Peking University's Open Laboratory for Intelligent Robotics(https://robotics.pkusz.edu.cn/resources /dataset/). The dataset comprises 555 training samples and 138 validation samples, encompassing six types of defects: short, spur, open circuit, mouse bite, spurious copper, and missing hole. The image resolutions in the dataset vary, including categories such as 2464 × 3056, 2156 × 2544, 2316 × 2868, 1921 × 2904, and 2154 × 2759, among others. Figure 3 presents six examples of defect data, with each image illustrating a distinct defect type.

Figure 4 illustrates the distribution of defective instances across each category within the training set, revealing that the number of instances is approximately balanced across categories, with no evident sample imbalance. Figure 5 depicts the distribution of centroid positions for defect targets within PCB images, as well as the distribution of target sizes. In Fig. 5, x denotes the horizontal coordinate of the defect center, while y represents the vertical coordinate of the defect center, with both x and y normalized to the range [0, 1]. Width denotes the width of the defective region, and height signifies the height of the defective region, with both width and height scaled equivalently to x and y. Most defect areas in the PCB defect dataset occupy less than 0.0016 of the total image area. The majority of defects in this dataset are small targets, presenting a significant challenge for defect detection within the PCB dataset. This dataset is characterized by its small and similarly sized detection targets, which inform the design of our network.

The lightweight network accelerates the model's inference speed (measured in FPS) by minimizing its parameters while striving to enhance detection accuracy without compromising the inference performance. For the PCB dataset characterized by small targets and concentrated distribution, we developed a model incorporating a single high-resolution detection head, thereby reducing the number of head parameters. We

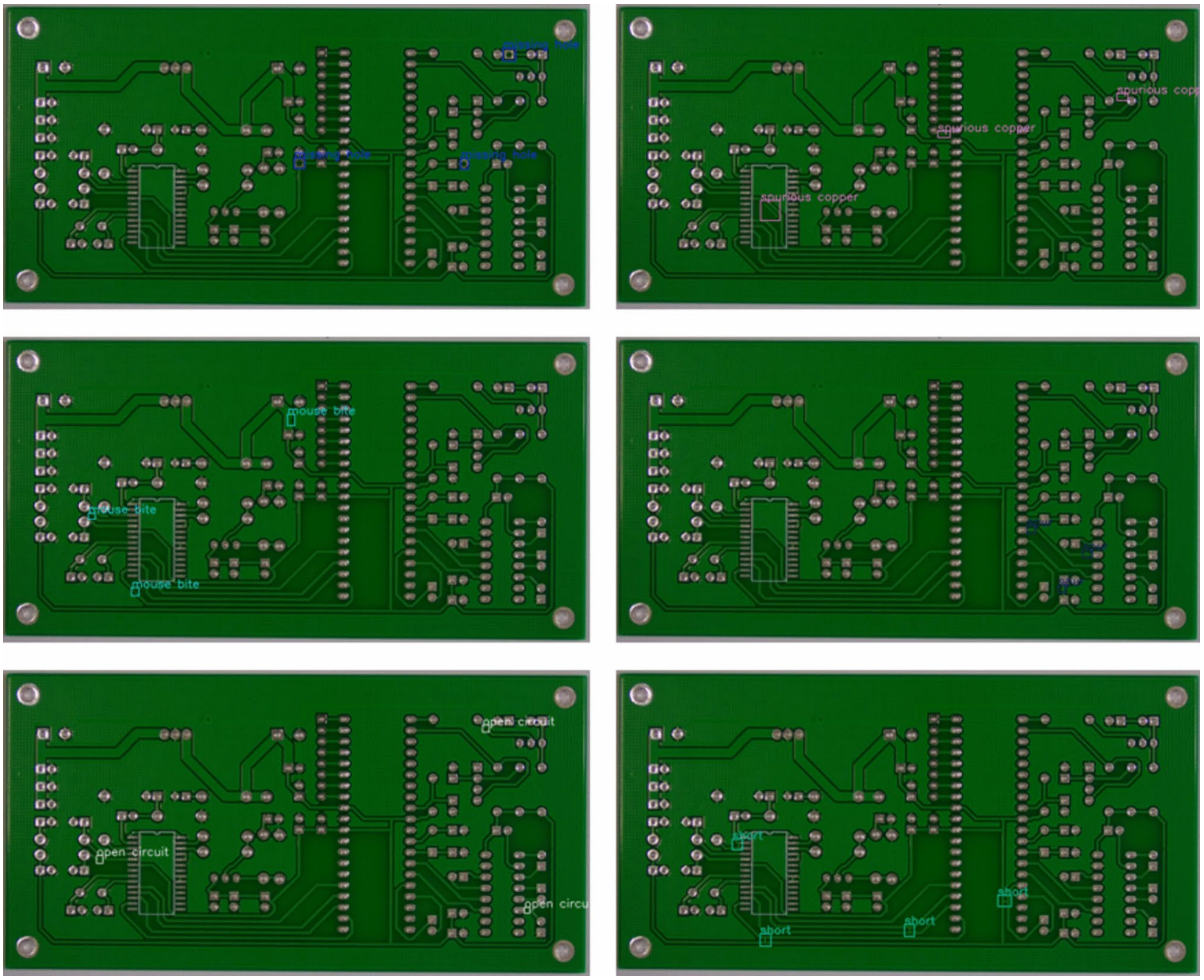

Fig. 3. The examples of the PCB defect datasets.

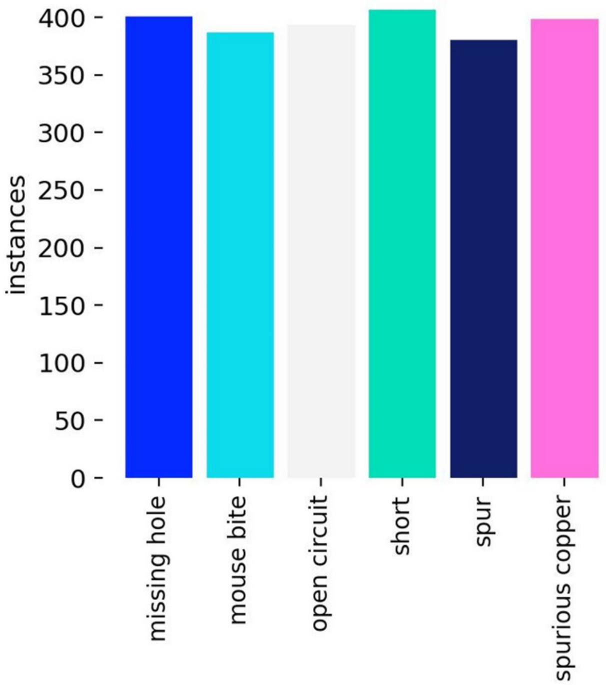

Fig. 4. The instances of each category in the training dataset.

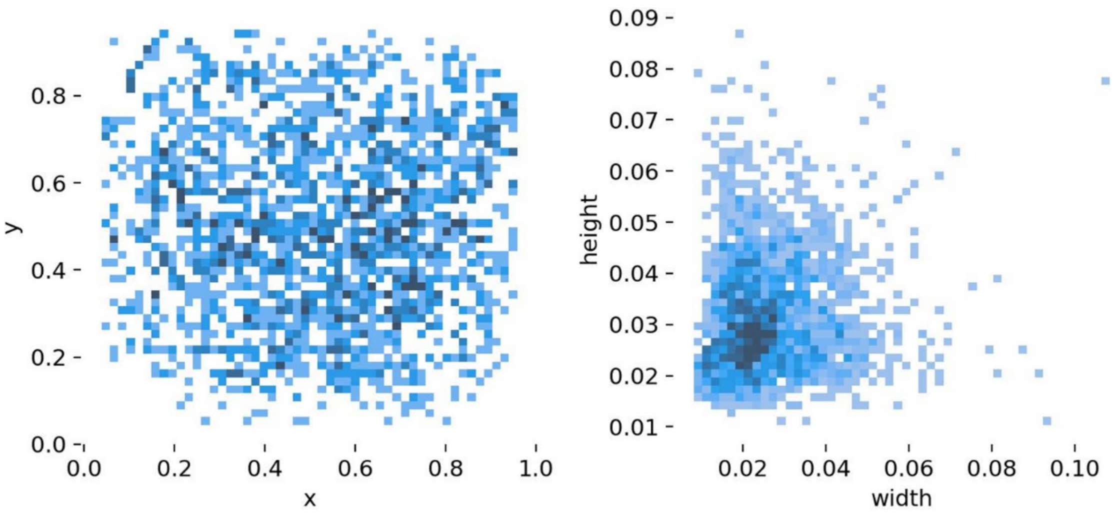

Fig. 5. Target centroid position distribution and target size distribution.

incorporated RepConv (Re-parameterizable Convolution) into the Backbone and Neck modules to enhance model detection performance without incurring additional inference costs.

Proposed network

In contrast to prior approaches to PCB defect detection model design, we refrained from modifying or adding modules to existing models. Instead, we employed existing deep learning techniques to develop a PCB defect detection model from scratch, characterized by a simple structure, minimal parameters, and rapid detection speed, tailored to the characteristics of the PCB dataset. Based on the characteristics of the dataset, we developed SEPDNet.

SEPDNet is designed with a focus on simplicity, as depicted in Fig. 6. The proposed SEPDNet contains fewer parameters in the backbone, neck, and head sections compared to the YOLO series model shown in Fig. 2. The YOLO model depicted in Fig. 2 undergoes five stages of downsampling, resulting in a minimum feature map resolution of 1/32 of the input. The backbone of SEPDNet is downsampled four times, yielding a minimum

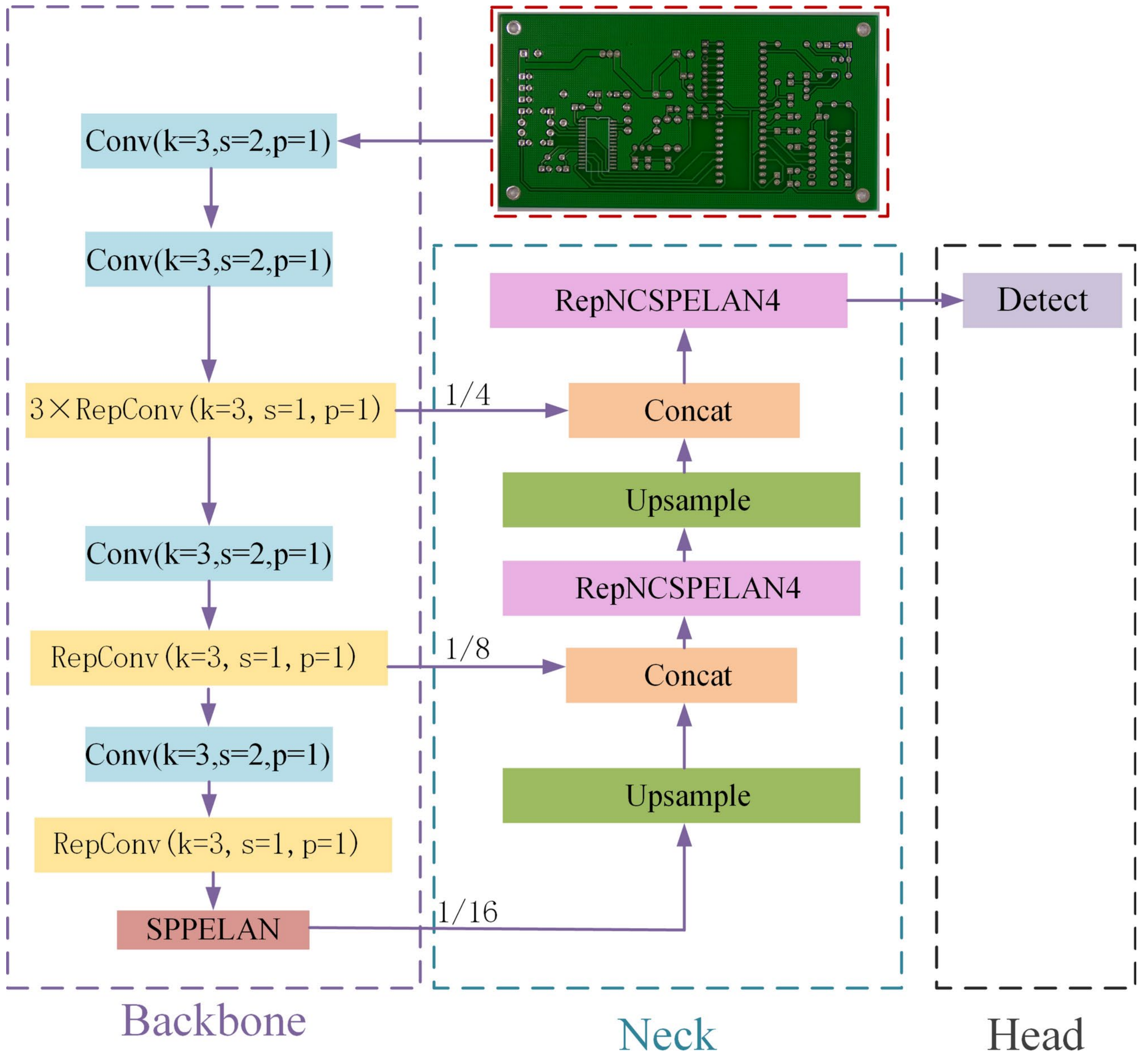

Fig. 6. The architecture of SEPDNet.

feature map resolution of 1/16 of the input. Compared to the YOLO series model, the SEPDNet backbone requires one fewer downsampling operation.

Instead of using the feature fusion approach of PAN (Path Aggregation Network) in the neck, SEPDNet adopts FPN (Feature Pyramid Networks) to create a 1/4-resolution feature map by merging shallow backbone features with upsampled features. Figure 7 illustrates the structures of FPN and PAN, highlighting that FPN is simpler and has fewer layers than PAN. The single detection head is specifically designed to match the centralized target size distribution observed in the PCB defect dataset. The SEPDNet head has only one-third of the parameters compared to the YOLO series' three-head design. The final detection results are derived from direct predictions made on the 1/4 resolution feature map. SEPDNet generates predictions from the 1/4 resolution feature maps, thereby mitigating the issue of downsampling-induced loss of detailed information for small targets. SEPDNet primarily comprises the following components: Concat, Conv, Detect, Upsample, RepCov, SPPELAN, and RepNCSPELAN4. The Concat module is employed to concatenate the feature maps.

Figure 8a illustrates the architecture of the Conv module, which incorporates Conv2d, BatchNorm2d, and the SiLU activation function. Figure 7b depicts the architecture of the RepConv module. During training, RepConv operates with multiple distinct branches, which are subsequently fused into a single convolutional layer during inference through reparameterization. The RepConv module enhances the model's feature extraction capabilities without elevating the inference cost.

Figure 9 depicts the architecture of SPPELAN, RepNCSPELAN4, and RepCSP. The SPPELAN module incorporates multiple pooling layers to expand the receptive field and extract contextual semantic information.

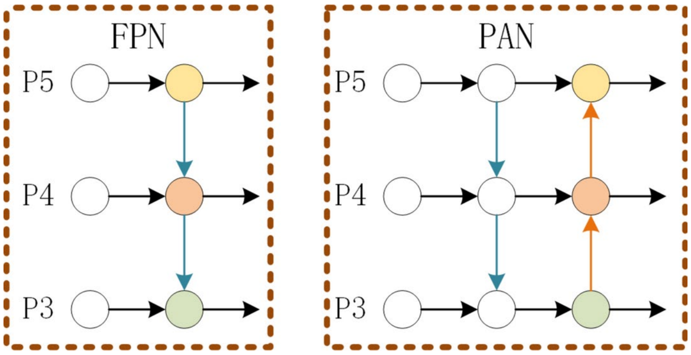

Fig. 7. The architectures of FPN and PAN.

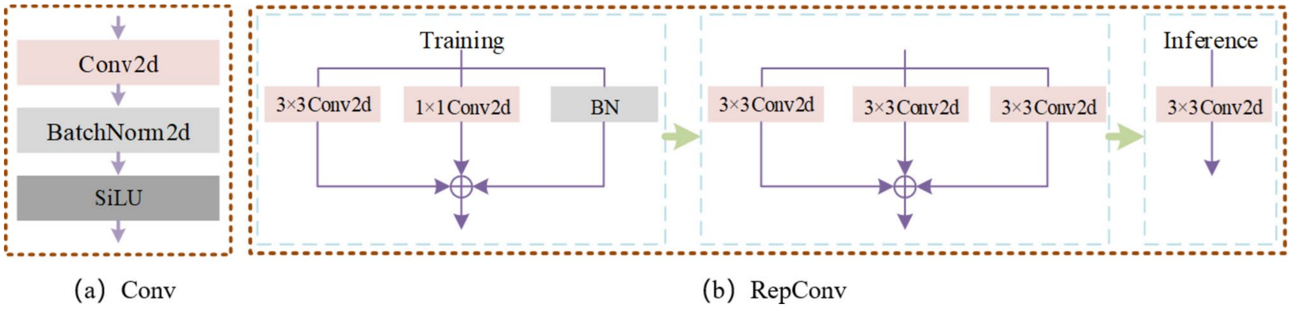

Fig. 8. The architecture of the Conv and RepConv.

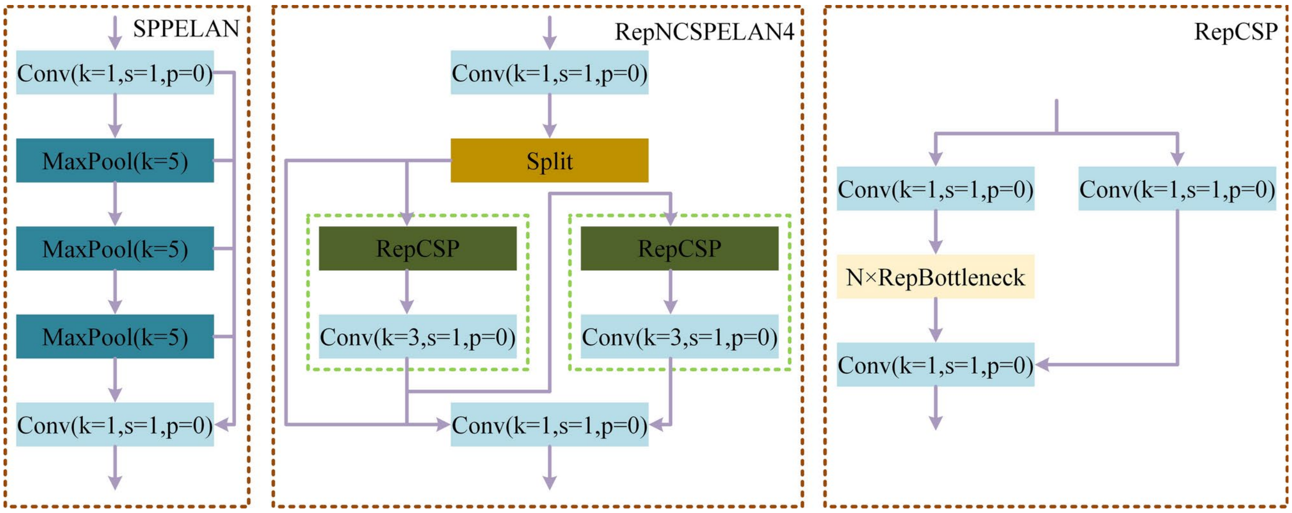

Fig.9. The architecture of the Conv and RepConv.

RepNCSPELAN4 comprises two RepCSP modules and several Convs, utilizing residual connections. This module represents an aggregated network structure, designed to be lightweight while enhancing both inference speed and accuracy. RepCSP consists of three Convs and N RepBottleneck modules. The bottleneck architecture of RepBottleneck is depicted in Fig. 10, showcasing the network structure with and without residual connections. The RepCSP architecture reduces the number of parameters while preserving the model's feature extraction capabilities.

Figure 1 illustrates the model architecture of the YOLO series, with subsequent models adhering to a similar structural framework. In contrast to the YOLO series algorithms, SEPDNet streamlines the backbone and neck architecture, retaining only a single detection head. Given that the target sizes within this study's dataset are

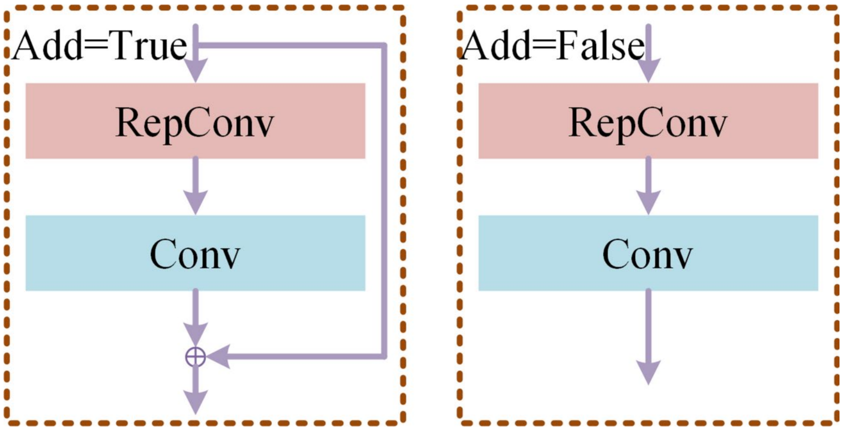

Fig.10. The architecture of the RepBottleneck.

similar, only one detection head is utilized. For different datasets, additional detection heads can be incorporated based on the distribution of bounding boxes.

Experiments
Evaluation metric
To illustrate the effectiveness of our approach, we assess our model using the validation dataset. We present the classical confusion matrix, comprising true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). In the context of object detection, TP refers to the number of ground truth defective objects correctly identified as defective, FP refers to the number of ground truth normal objects falsely predicted as defective, and FN denotes ground truth defective objects misclassified as normal. Once these values are established, precision and recall are computed using Eqs. (1) and (2). Precision quantifies the proportion of objects predicted as defective that are truly defective, while recall measures the proportion of truly defective objects that are correctly identified by the model. Ultimately, the F1 score (Eq. 3) serves as the definitive metric, representing the harmonic mean of precision and recall.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

Average Precision (AP) and mean Average Precision (mAP) are utilized to evaluate defect detection performance. Precision, recall, and Intersection over Union (IoU) are employed to calculate the AP value. AP is assessed across various IoU thresholds. It is computed for 10 IoU thresholds ranging from 50 to 95%, in increments of 5%, and is typically reported as AP50:5:95. AP evaluates the performance for individual classes, while mAP represents the mean AP value across all classes. The Precision/Recall (P/R) curve offers an intuitive visualization, directly reflecting the performance of the detection algorithm. The AP value is derived by computing the area under the P/R curve. The number of parameters and frames per second (FPS) are metrics employed to evaluate time efficiency. FPS was averaged over five experiments.

Experimental environment
We perform experiments utilizing the PCB defect dataset from Peking University's Open Laboratory for Intelligent Robotics, train the model parameters on the training set, and evaluate the model's performance on the validation dataset. To maintain experimental fairness, identical hardware and software configurations are employed across all tests. The hardware specifications for the experiments are as follows: Intel® Core™ i9-12900 K Processor, 64 GB RAM, a single RTX 4090 GPU, and Ubuntu 22.04 operating system.

SEPDNet employs an anchor-free detection head and utilizes the TaskAlignedAssigner sampling strategy during training. SEPDNet was developed using Ultralytics(https://github.com/ultralytics/ultralytics) with PyTorch version 2.2.0, mosaic data augmentation was disabled during training. The loss functions used are cross-entropy loss and ciou loss. Both the loss function and the TaskAlignedAssigner sampling strategy are implemented using ultralytics.

All models in the experiment were trained with identical hyperparameters. The training configuration included a maximum of 300 epochs, a base learning rate of 0.001, a batch size of 16 for a single RTX 4090 GPU, the AdamW optimizer, and an input resolution of 640 × 640 for both training and evaluation.

Head Feature Resolution	Params(M)	F1 score	mAP50(%)	mAP50:95(%)
1/8	2.27	0.925	94.6	50.2
1/16	2.40	**0.938**	94.5	51.0
1/4, 1/8	3.13	0.936	**95.3**	52.1
1/4, 1/16	3.57	0.932	94.9	52.0
1/8, 1/16	3.36	0.922	93.0	50.2
1/4, 1/8, 1/16	3.99	0.934	94.8	52.2
1/4	**2.05**	0.937	95.1	**53.1**

Table 1. The comparison of detection head design. Significant values are in bold.

	F1 score	mAP50(%)	mAP50:95(%)
Conv	0.922	94.5	50.9
RepConv	0.921	94.2	51.5
C3	0.933	94.6	51.8
C2f.	0.920	93.9	51.6
RepNCSPELAN4	**0.937**	**95.1**	**53.1**

Table 2. The comparison of different modules in Neck. Significant values are in bold.

	F1 score	mAP50(%)	mAP50:95(%)
Conv	0.911	94.1	52.1
RepNCSPELAN4	0.920	93.8	51.2
C3	0.924	94.0	51.1
C2f.	0.936	**95.2**	51.7
RepConv	**0.937**	95.1	**53.1**

Table 3. The comparison of different modules in Backbone. Significant values are in bold.

Ablation study

We conducted ablation studies on the Head, Neck, and Backbone components of SEPDNet to evaluate the performance of various modules. Table 1 presents a comparison of performance across varying numbers of detector heads. In Table 1, the values 1/4, 1/8, and 1/16 represent the resolution ratios of the input feature map of the detection head relative to the input image of the model. The results show that the highest mAP50:95 can be obtained at 1/4 resolution using a single detector head with the fewest parameters available, while the mAP50 and F1 scores are only slightly lower than those of the other models. This finding confirms that a single detector head provides superior detection performance when the target size distribution is concentrated.

We performed experiments on the Neck component by replacing RepNCSPELAN4 with Conv, RepConv, C3, and C2f. modules, respectively. As demonstrated in Table 2, RepNCSPELAN4 exhibits a mAP50 improvement of 0.6%, 0.9%, 0.5%, and 1.2% over Conv, RepConv, C3, and C2f., respectively. Additionally, RepNCSPELAN4 achieves a superior F1 score and mAP50:95 metrics compared to the other modules, indicating its enhanced capability in extracting deep features.

We also performed experiments on the Backbone component by substituting RepConv with Conv, RepNCSPELAN4, C3, and C2f. modules, respectively. As illustrated in Table 3, RepConv achieves a superior F1 score and mAP50:95 metrics compared to Conv, RepNCSPELAN4, C3, and C2f. However, RepConv's mAP50 is marginally lower than that of C2f. The multiple branches of RepConv enhance its feature representation capabilities without incurring additional inference costs. Experimental results demonstrate that RepConv outperforms other modules within the backbone.

Results comparison and analysis

SEPDNet is compared with YOLOv3-tiny, YOLOv5u, YOLOv6u, YOLOv7, YOLOv8, YOLOv9u, and YOLOv10u, as detailed in Table 4. Compared to YOLOv5u-s, YOLOv6u-s, YOLOv7, YOLOv8s, YOLOv9u-s, and YOLOv10u-s, SEPDNet achieves significant improvements: an increase of 0.032, 0.067, 0.047, 0.043, 0.025, and 0.067 in F1 score, respectively; a rise in mAP50 by 3.3%, 7.6%, 3.4%, 3%, 2.7%, and 4.4%; and an enhancement in mAP50:95 by 4.9%, 8.4%, 7.5%, 5%, 3.8%, and 4.5%. Experimental results show that our proposed method achieves 53.1% mAP50:95, whereas YOLOv9u-s achieves 49.3% mAP50:95, outperforming other YOLO detectors in F1 score, mAP50, and mAP50:95. The proposed SEPDNet has a parameter size of only 2.05 M, which is significantly smaller than that of the other models. However, it achieves substantial improvements in

Method	FPS	Params(M)	F1 score	mAP50(%)	mAP50:95(%)
YOLOv3-tiny	**250.0**	12.13	0.806	83.1	39.1
YOLOv5u-s	161.29	9.11	0.905	91.8	48.2
YOLOv6u-s	175.44	16.30	0.870	87.5	44.7
YOLOv7	68.493	36.51	0.890	91.7	45.6
YOLOv8s	158.73	11.13	0.894	92.1	48.1
YOLOv9u-s	105.26	7.17	0.912	92.4	49.3
YOLOv10u-s	169.49	8.04	0.870	90.7	48.6
SEPDNet	204.08	**2.05**	**0.937**	**95.1**	**53.1**

Table 4. The metrics of models. Significant values are in bold.

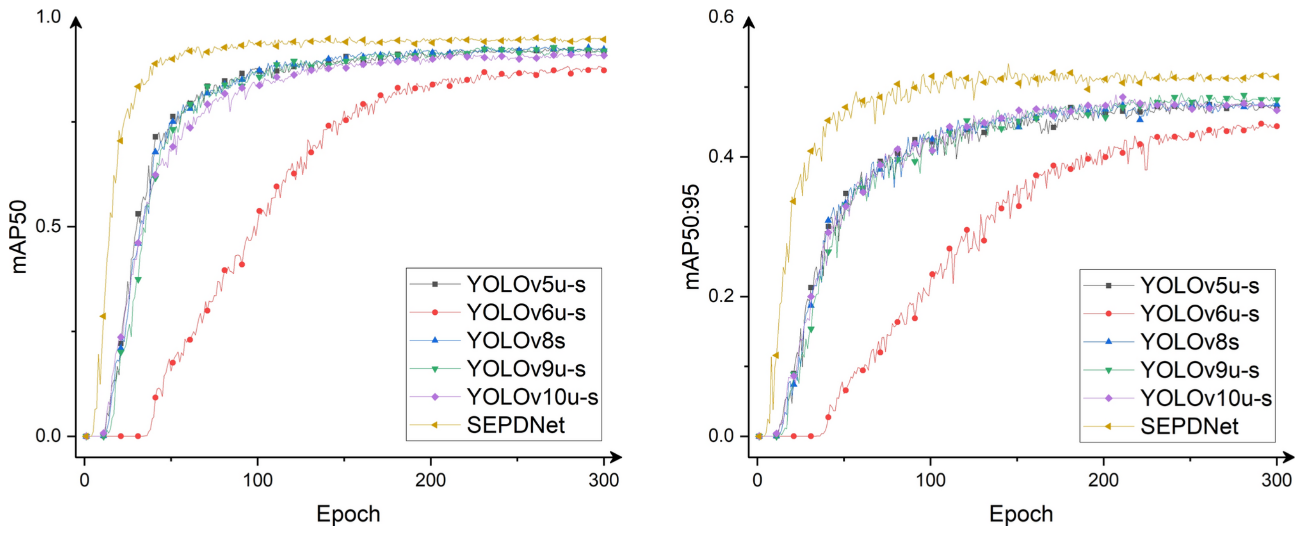

Fig. 11. The validation metric by epochs.

Defeat classes	YOLOv5u-s (%)	YOLOv8s (%)	YOLOv9u-s (%)	YOLOv10u-s (%)	SEPDNet (%)
missing hole	61.1	58.8	61.3	63.4	63.7
mouse bite	47.8	50.9	48.8	48.7	54.9
open circuit	47.6	46.2	48.9	45.2	56.7
short	48.2	47.5	47.3	47.7	50.5
spur	36.7	37.3	39.1	38.5	41.6
spurious copper	47.6	48.2	50.5	47.9	51.4
all	48.2	44.7	48.1	48.6	53.1

Table 5. mAP50:95 for different PCB detection models on the validation dataset.

F1 score, mAP50, and mAP50:95. This validates our assertion that optimal performance is attained through a model specifically designed to align with the characteristics of the dataset.

Figure 11 illustrates the progression of mAP50 and mAP50:95 metrics for YOLOv5u-s, YOLOv6u-s, YOLOv8s, YOLOv9u-s, YOLOv10u-s, and the proposed SEPDNet on the validation set throughout the model's training process. It is evident that, as the metrics stabilize, the proposed SEPDNet consistently outperforms the other models in both mAP50 and mAP50:95 metrics.

Table 5 presents the mAP50:95 metrics for various models across different categories within the validation set. In every category of PCB defects, SEPDNet achieved the highest mAP50:95 metric. This underscores the effectiveness of the SEPDNet we developed for the PCB defect detection task.

Table 6 shows the FPS of different models on CPU i9-13900KF and i9-12900 K, and it can be seen that the inference speed of SEPDNet on the CPU is lower than that of the models in the YOLO series. SEPDNet has some limitations on edge devices that cannot be accelerated with GPUs.

Figure 12 displays the confusion matrix for the proposed SEPDNet detection results on the validation set, providing a detailed view of the model's performance across different categories. The horizontal axis indicates

Method	i9-13900KF	i9-12900 K
SEPDNet	14.53	7.18
YOLOv5u-s	30.40	14.9
YOLOv6u-s	25.51	12.18
YOLOv8s	28.09	13.05
YOLOv9u-s	23.15	12.33
YOLOv10u-s	29.59	14.84

Table 6. FPS for different models in different CPUs.

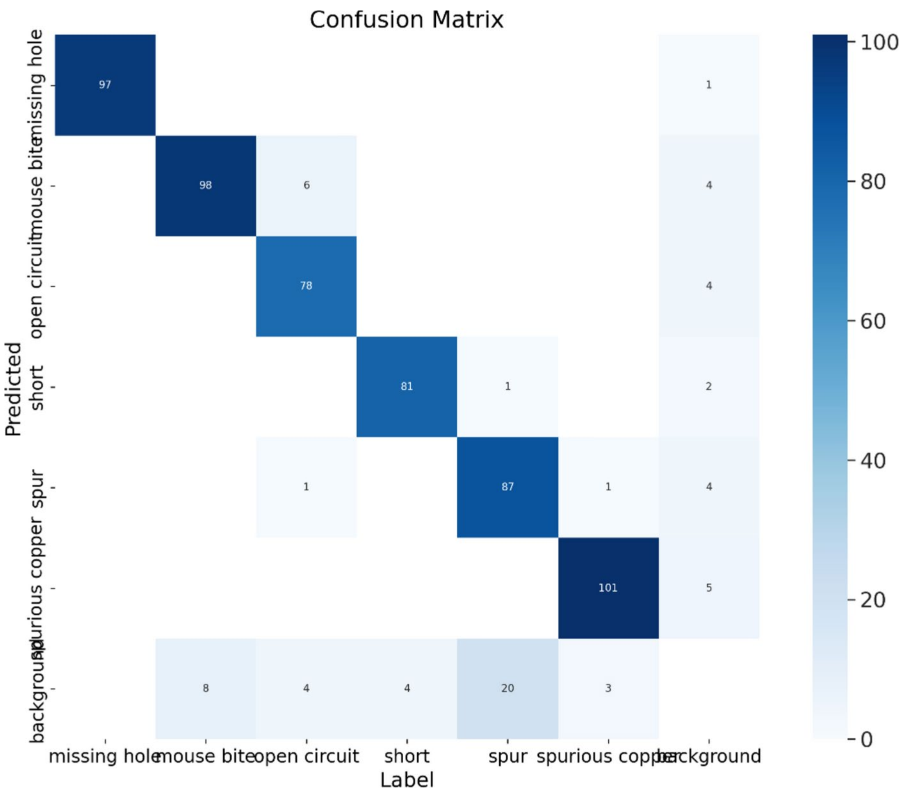

Fig. 12. The confusion matrix of the proposed model.

the number of defects in each category within the validation set, while the vertical axis reflects the number of defects in each category as predicted by the proposed model. Figure 13 illustrates the precision-recall (PR) curves for SEPDNet/YOLOv3s/YOLOv5s/YOLOv6s/YOLOv8s/YOLOv9s/YOLOv10s. A larger area below the PR curves indicates that the model performs better, and the comparison shows that SEPDNet, compared to other models, has better detection performance. Figure 14 presents the prediction results of various models on three distinct PCB defect images. A comparative analysis of the detection results reveals that SEPDNet exhibits a higher accuracy rate than other models.

Conclusion

This paper examines the characteristics of the PCB defect dataset, which features small targets with an almost uniform size distribution. In response to the specific characteristics of the defective targets within this dataset and the real-time demands of industrial production, we developed SEPDNet. Compared to YOLO series models, SEPDNet surpasses them in detection accuracy for the PCB defect detection task, while using a significantly smaller number of parameters. In comparison to YOLOv5u-s, YOLOv6u-s, YOLOv7, YOLOv8s, YOLOv9u-s, and YOLOv10u-s, SEPDNet demonstrates substantial improvements: an increase of 0.032, 0.067, 0.047, 0.043, 0.025, and 0.067 in F1 score, respectively; a rise in mAP50 by 3.3%, 7.6%, 3.4%, 3%, 2.7%, and 4.4%; and an enhancement in mAP50:95 by 4.9%, 8.4%, 7.5%, 5%, 3.8%, and 4.5%. In addition to these performance improvements, SEPDNet achieves an exceptionally fast detection speed of 204.08 FPS.

Although SEPDNet achieves good performance on PCB defect detection tasks. The direct application of SEPDNet to other defect detection areas may not yield good results because its single detector head design is only suitable for cases where the target size is concentrated. For detection tasks in different domains, some

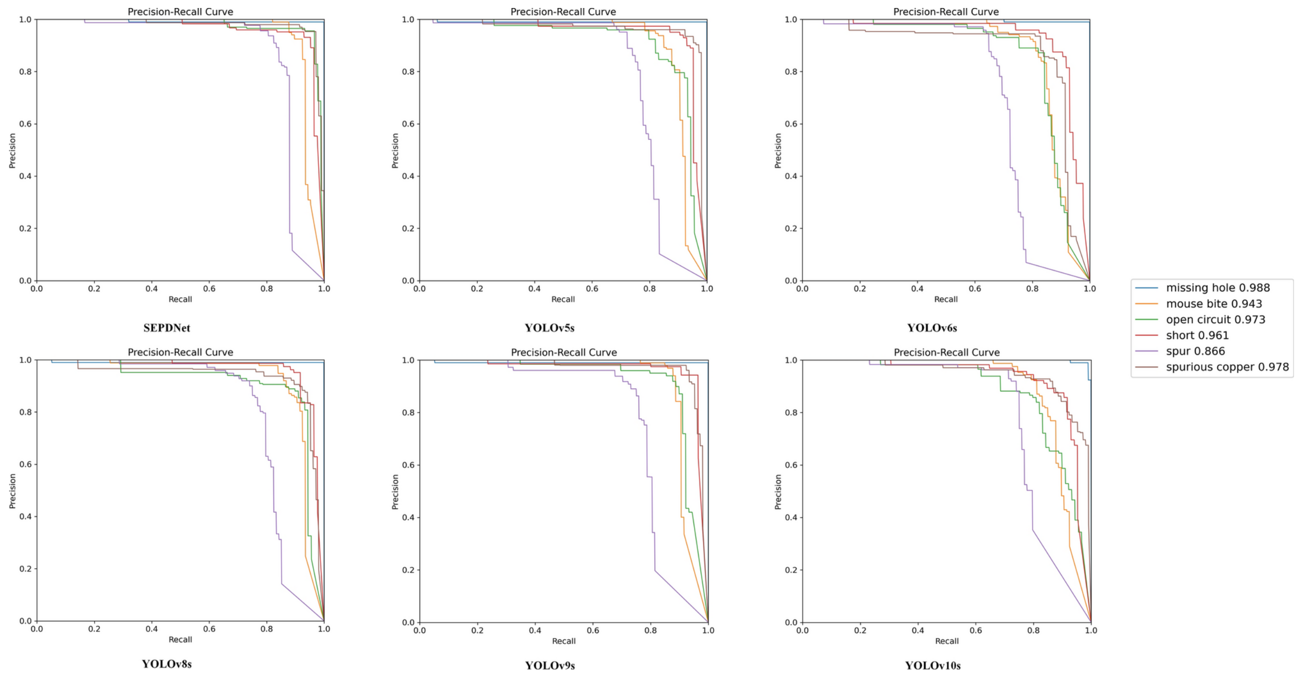

Fig. 13. The precision-recall curves of different models.

improvements need to be made to SEPDNet's backbone, neck, and head based on specific data characteristics. Further improvements are needed in application scenarios without GPU-accelerated computing.

The code for SEPDNet is available at: https://github.com/justld/SEPDNet/tree/main.

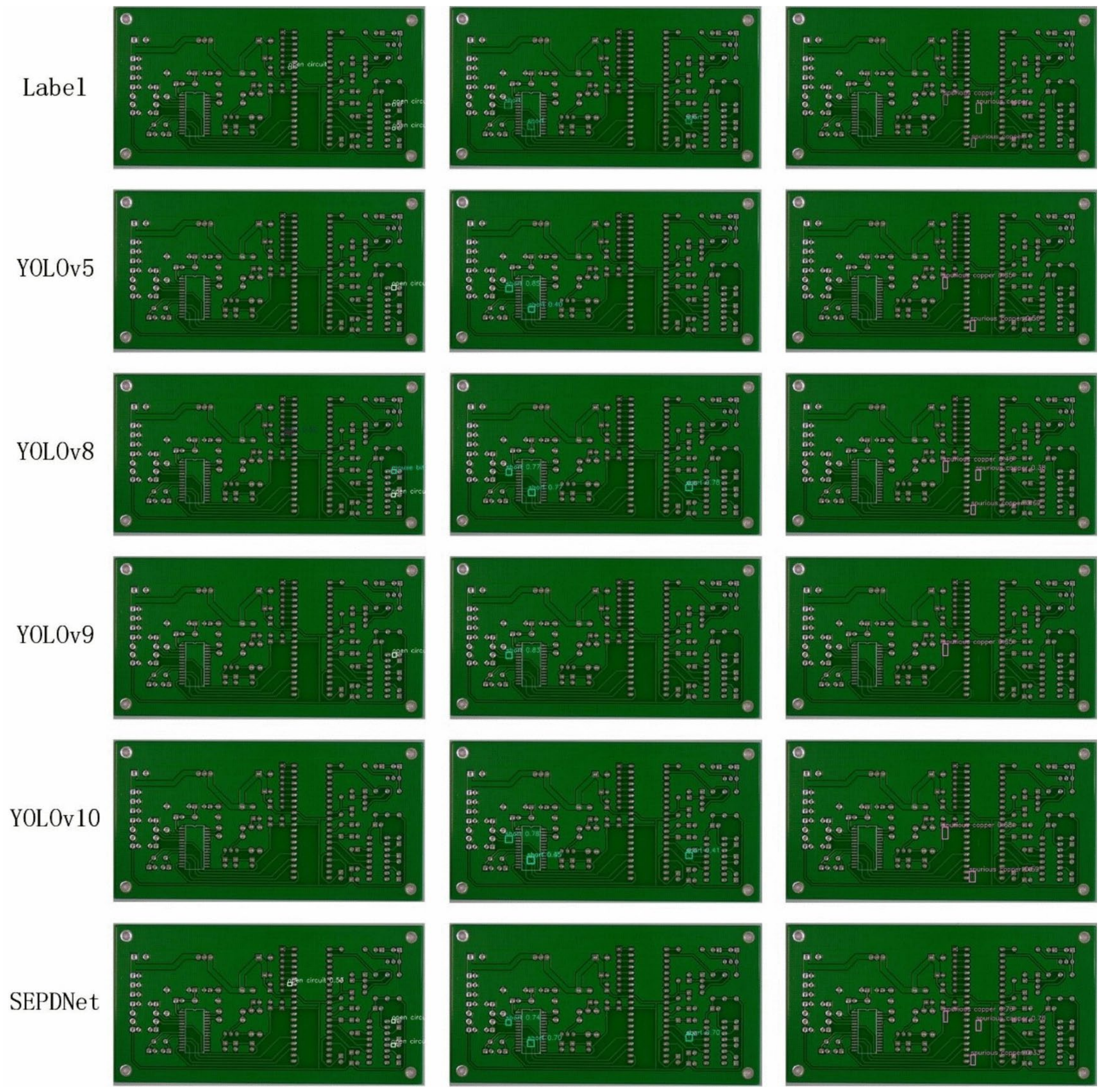

Fig. 14. The predicted results of different models.

Data availability

The datasets analysed during the current study are available at https://robotics.pkusz.edu.cn/resources/dataset.

Received: 24 September 2024; Accepted: 27 December 2024
Published online: 29 March 2025

References

1. Lv, S. et al. A dataset for deep learning based detection of printed circuit board surface defect. *Sci. Data* **11**, 811 (2024).
2. Lim, J., Lim, J., Baskaran, V. M. & Wang, X. A deep context learning based PCB defect detection model with anomalous trend alarming system. *Results Eng.* **17**, 100968 (2023).
3. Yuan, M. *et al.* YOLO-HMC: An improved method for PCB surface defect detection. *IEEE Trans. Instrum. Meas.* (2024).
4. Wang, W.-C., Chen, S.-L., Chen, L.-B. & Chang, W.-J. A machine vision based automatic optical inspection system for measuring drilling quality of printed circuit boards. *IEEE Access* **5**, 10817–10833 (2016).
5. Yuk, E. H., Park, S. H., Park, C.-S. & Baek, J.-G. Feature-learning-based printed circuit board inspection via speeded-up robust features and random forest. *Appl. Sci.* **8**, 932 (2018).
6. Ling, Q. & Isa, N. A. M. Printed circuit board defect detection methods based on image processing, machine learning and deep learning: A survey. *IEEE Access* **11**, 15921–15944 (2023).
7. Zheng, J., Sun, X., Zhou, H., Tian, C. & Qiang, H. Printed circuit boards defect detection method based on improved fully convolutional networks. *IEEE Access* **10**, 109908–109918 (2022).
8. Chen, X., Wu, Y., He, X. & Ming, W. A comprehensive review of deep learning-based PCB defect detection. *IEEE Access* (2023).
9. Ji, T. et al. A Real-Time PCB defect detection model based on enhanced semantic information fusion. *Signal Image Video Process.* **18**, 4945–4959 (2024).
10. Kaya, G. U. Development of hybrid optical sensor based on deep learning to detect and classify the micro-size defects in printed circuit board. *Measurement* **206**, 112247 (2023).

11. Hu, X., Kong, D., Liu, X., Zhang, J. & Zhang, D. Printed circuit board (PCB) surface micro defect detection model based on residual network with novel attention mechanism. *Comput. Mater. Continua.* **78**, 915–933 (2024).
12. Chiun, O. Y. & Ruhaiyem, N. I. R. Object detection based automated optical inspection of printed circuit board assembly using deep learning. In *International Conference on Soft Computing in Data Science* (ed. Chiun, O. Y.) 246–258 (Springer, 2023).
13. Kiobya, T., Zhou, J., Maiseli, B. & Khan, M. Attentive context and semantic enhancement mechanism for printed circuit board defect detection with two-stage and multi-stage object detectors. *Sci. Rep.* **14**, 18124 (2024).
14. Li, C., Xia, W. & Jiang, Z. Weak Feature Defect Generation with GAN for Faster RCNN Based PCB Defect Detection. In *2023 8th International Conference on Data Science in Cyberspace (DSC)* (ed. Li, C.) 306–312 (IEEE, 2023).
15. Jiang, W., Li, T., Zhang, S., Chen, W. & Yang, J. PCB defects target detection combining multi-scale and attention mechanism. *Eng. Appl. Artif. Intell.* **123**, 106359 (2023).
16. Zhang, H., Jiang, L. & Li, C. CS-ResNet: Cost-sensitive residual convolutional neural network for PCB cosmetic defect detection. *Expert Syst. Appl.* **185**, 115673 (2021).
17. Ding, R., Dai, L., Li, G. & Liu, H. TDD-net: A tiny defect detection network for printed circuit boards. *CAAI Trans. Intell. Technol.* **4**, 110–116 (2019).
18. Xuan, W. et al. A lightweight modified YOLOX network using coordinate attention mechanism for PCB surface defect detection. *IEEE Sensors J.* **22**, 20910–20920 (2022).
19. Wu, L., Zhang, L. & Zhou, Q. Printed circuit board quality detection method integrating lightweight network and dual attention mechanism. *IEEE Access* **10**, 87617–87629 (2022).
20. Du, B. et al. YOLO-MBBi: PCB surface defect detection method based on enhanced YOLOv5. *Electronics* **12**, 2821 (2023).
21. Chen, W., Huang, Z., Mu, Q. & Sun, Y. PCB defect detection method based on transformer-YOLO. *IEEE Access* **10**, 129480–129489 (2022).
22. Tang, J. et al. PCB-YOLO: An improved detection algorithm of PCB surface defects based on YOLOv5. *Sustainability* **15**, 5963 (2023).
23. Wang, C.-Y. & Liao, H.-Y. M. YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems. Preprint at arXiv:.09332. (2024).
24. Hussain, M. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. *Machines* **11**, 677 (2023).
25. Nazir, A. & Wani, M. A. You only look once-object detection models: a review. In *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)* (ed. Nazir, A.) 1088–1095 (IEEE, 2023).
26. Redmon, J. & Farhadi, A. YOLO9000: Better, Faster, Stronger. *IEEE.* 6517–6525 (2017).
27. Redmon, J. & Farhadi, A. YOLOv3: An Incremental Improvement. arXiv e-prints. (2018).
28. Bochkovskiy, A., Wang, C. Y. & Liao, H. Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. Preprint at arXiv:2004. (2020).
29. Jocher, G. *et al.* ultralytics/yolov5: v6. 2-yolov5 classification models, apple m1, reproducibility, clearml and deci. ai integrations. *Zenodo.* (2022).
30. Li, C. *et al.* YOLOv6: A single-stage object detection framework for industrial applications. Preprint at arXiv:.02976. (2022).
31. Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: *Proc. IEEE/CVF conference on computer vision and pattern recognition.* 7464–7475 (2023).
32. Sohan, M., Sai Ram, T., Reddy, R. & Venkata, C. A review on yolov8 and its advancements. In *International Conference on Data Intelligence and Cognitive Informatics* (ed. Sohan, M.) 529–545 (Springer, 2024).
33. Wang, C., Yeh, I. & Liao, H. YOLOv9: Learning what you want to learn using programmable gradient information. Preprint at arXiv:.13616.
34. Wang, A. *et al.* Yolov10: Real-time end-to-end object detection. Preprint at arXiv:.14458. (2024).
35. Kisantal, M. Augmentation for Small Object Detection. Preprint at arXiv:.07296. (2019).
36. Chen, Z., Wu, K., Li, Y., Wang, M. & Li, W. SSD-MSN: An improved multi-scale object detection network based on SSD. *IEEE Access* **7**, 80622–80632 (2019).
37. Pang, Y., Cao, J., Wang, J. & Han, J. JCS-Net: Joint classification and super-resolution network for small-scale pedestrian detection in surveillance images. *IEEE Trans. Inf. For. Secur.* **14**, 3322–3331 (2019).
38. Akyon, F. C., Altinuc, S. O. & Temizel, A. Slicing aided hyper inference and fine-tuning for small object detection. In *2022 IEEE International Conference on Image Processing (ICIP)* (ed. Akyon, F. C.) 966–970 (IEEE, 2022).

Author contributions
Du Lang wrote the main manuscript text. Zhenzhen Lv prepared Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. All authors reviewed the manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to D.L.