EAR-YOLO: An Improved Model for Printed Circuit Board Surface Defect Detection

Rui Geng, Nannan Zhao, Yang Wang, Xinyu Ouyang

Abstract—**Defect detection is an important step in ensuring the quality of PCB manufacturing, so a PCB surface defect detection model based on an improved YOLOv8 algorithm is proposed to address the shortcomings of traditional detection techniques in terms of accuracy and generalization ability to different types of defects. The innovations of this model are mainly reflected in three aspects. First, the model adopts EfficientNetV2 as its backbone network to enhance training speed and accuracy. Second, at the three feature output layers in the neck of the model, a mixed model that benefits from both self-attention and convolution (ACmix) is introduced, thereby improving the model's feature recognition efficiency. Finally, the detection head is redesigned by replacing traditional convolution with the more efficient Receptive-Field Attention convolutional operation (RFAConv) and introducing the Slide Loss classification loss function, resulting in a new detection head named RFASHead. Therefore, this study names the improved model EAR-YOLO. Experimental results show that the improved EAR-YOLO model achieves a mean average precision (mAP) of 82.1% in the public PCB-AOI dataset and a precision of 96.4% in the PKU-Market-PCB dataset. Compared to the baseline model YOLOv8n, the improvements were 7.7% and 5%, respectively, and exhibit superior detection accuracy and generalization compared to other mainstream models.**

Index Terms—**PCB, defect detection, YOLOv8, feature recognition, ACmix, RFASHead.**

I. Introduction

In recent years, the manufacturing industry has been continuously pursuing higher levels of automation and manufacturing efficiency. However, the increase in automation also raises the risk of product defects, which is also true in the PCB industry. Surface mount technology (SMT), as the core technology for PCB production automation, involves the printing stage, which is responsible for accurately applying solder paste to the pads of electronic circuit boards. This stage is the most delicate part of the entire process. The quality of PCB printing directly impacts the quality and efficiency of the SMT assembly. Studies have shown that more than 80% of defects in surface-mounted electronic products are caused by printing quality defects, such as insufficient solder and bridging [1], and these defects result in significant economic losses for PCB packaging manufacturers. Therefore, effective detection of circuit boards to identify anomalies and prevent misjudgment of circuit boards in good condition is crucial.

Traditional detection methods mainly include manual inspection, phase measurement profilometry (PMP), and laser triangulation technology. Manual inspection relies on visual observation and the experience of operators to identify product defects. This method not only consumes a large amount of human resources, but also has low detection efficiency, leading to the problems of missed detection or false positives. Although PMP and laser triangulation technology can replace manual inspection and provide higher detection accuracy, larger measurement ranges, and faster measurement speeds [2], the detection accuracy in practical applications is unsatisfactory due to difficulties in precisely measuring the position and distance of the camera and projector, as well as the complexity of establishing phase mapping relationships. Moreover, factors such as poor real-time performance, large equipment size, and high costs also limit the application of these technologies in PCB surface defect detection.

Before the development of deep learning, traditional machine learning methods were the mainstream approach to defect detection. Traditional machine learning involves extracting features from images and then feeding these features into a classifier for classification based on their differences [3]. This approach treats defect detection as a binary classification problem, with common methods including Support Vector Machines (SVM) [4], decision trees, and shallow neural networks. Lu et al. [5] proposed a PCB defect detection method based on Bayesian feature fusion. Li et al. [6], considering the limitations of using a single feature to describe various defects on bare PCBs, proposed a method to fuse gradient direction information entropy and local binary patterns to construct feature vectors and quantitatively assess the significance of defect features. Hagi et al. [7] proposed a random sampling-based SVM method for classifying electronic circuit board defects. After applying traditional machine learning methods to PCB defect detection, detection efficiency and accuracy were greatly improved compared to manual inspection.

Compared to traditional machine learning, deep learning models can automatically learn feature representations from data, eliminating the need for manual feature design [8]. This capability allows the model to adapt more flexibly to the complexity and variations of the data. Moreover, deep learning models are capable of capturing more subtle patterns and features when handling large-scale data [9], and typically exhibit better generalization ability, ensuring excellent performance on new datasets. In the field of PCB surface defect detection, object detection algorithms such as the YOLO [10] series, SSD [11], and Faster R-CNN

Manuscript received January 7, 2025; revised March 17, 2025. This work is supported by the Fundamental Research Funds for the Liaoning Universities of China (Grant No. LJ212410146005).

Rui Geng is a postgraduate student of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, 114051, China (e-mail: 18110587727@163.com).

Nannan Zhao is a professor of the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan,Liaoning, 114051, China (Corresponding author, e-mail: 723306003@qq.com).

Yang Wang is a professor of the School of IoT Research Institute, Shenzhen Polytechnic University, Shenzhen,Guangdong, 518055, China (Corresponding author, e-mail: Wyang@szpu.edu.cn).

Xinyu Ouyang is a professor of the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, 114051, China (e-mail: 13392862@qq.com).

[12] have been widely applied. These algorithms can be categorized into two-stage and one-stage types [13]. In recent years, many scholars have conducted experimental studies on the two-stage object detection algorithm Faster-RCNN [14]. The TDD-net model proposed by Ding et al. [15], based on Faster R-CNN, focuses on detecting small defects. The model determines appropriate scales through k-means clustering and uses the feature pyramid network (FPN) [16] to enhance semantic feature representation, thereby improving detection accuracy. However, due to the model's complexity, large number of parameters, and localization accuracy issues caused by RoI pooling, TDD-net faces challenges in computational efficiency. Hu et al. [17] improved Faster R-CNN by introducing the ShuffleV2 [18] residual unit, which reduces the computational load. However, due to its two-stage design, the detection time is relatively long, and it faces difficulties in detecting open defects. Adibhatla et al. [19], based on YOLOv1, proposed the micro YOLOv2 network. By pruning the Darknet network and adding batch normalization layers, they improved the inference speed. However, the model lacks additional performance metrics, and high accuracy can only be achieved by detecting local images obtained from cropping the entire PCB image, indicating that the model's generalization ability still needs improvement. Dai et al. [20] combined active learning and semi-supervised learning to propose a solder defect detection method based on YOLOv3 and VGG-16 [21]. Although this method performs well in terms of detection accuracy, the detection speed of the model is slower than that of conventional fully CNN-based models, as the detection task is decomposed into localization and classification. Additionally, the SVM classification efficiency is low, limiting its applicability to other datasets. Adibhatla et al. [22] employed the YOLOv5 algorithm to achieve efficient PCB defect detection. In their experiments, they used three different scales of YOLOv5 models and applied mosaic data augmentation to expand the dataset. The backbone network of the model integrates CSPNet [23], PANet [24], and FPN, enhancing feature fusion and improving the recognition of small targets. However, this study directly applied the YOLOv5 model without innovative modifications to the algorithm and only detected whether defects were present, without classifying defect types in detail. This resulted in relatively low defect confidence scores.

Aiming at the above problems, combined with the challenges faced by the current PCB surface defect detection methods, this study proposes a new PCB surface defect detection method, EAR-YOLO, based on the improvement of YOLOv8, which improves the detection accuracy and also makes the model's generalization ability improved and has a good detection speed. Its main contributions are as follows:

(1) EfficientNetV2 [25] is used as the backbone network, which improves the training speed and maintains high accuracy by gradually increasing the image size and adaptively adjusting the regularization during the training process.

(2) The structure of the model neck is improved by adding three layers of ACmix [26] channels, which enables the model to have the global perception ability of the self-attention mechanism as well as the ability to capture local features through convolution, thus improving the efficiency of feature recognition.

(3) The detection head is redesigned to use RFAConv [27] instead of the traditional convolution, and the classification loss function is improved to SlideLoss [28], resulting in a new RFASHead.This detection head optimizes the workings of the convolution kernel and enhances the focus on difficult samples, which further improves the detection accuracy and the generalization of the model.

II. Related work

In 2023, Ultralytics introduced the latest iteration of the YOLO series algorithms—YOLOv8. This version integrates and optimizes features from its predecessors. YOLOv8 is capable of performing various tasks, including detection, classification, segmentation, pose estimation and tracking. Its network architecture is primarily divided into three core components: backbone, neck, and head. Figure 1 displays its network structure.

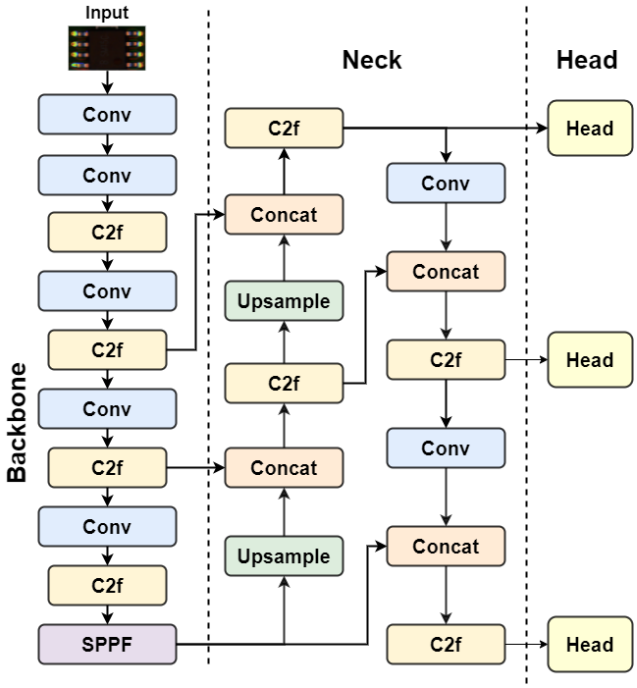

Fig. 1. YOLOv8 network structure diagram

A. Backbone

The backbone of YOLOv8 consists of five convolutional modules, four C2f modules, and one SPPF module, enabling efficient multi-level feature extraction from input images. In the initial stage, the input image undergoes preliminary processing through convolutional layers, gradually transitioning from low-level to high-level feature extraction. Throughout this process, the C2f module incorporates residual connections, significantly enhancing feature propagation and fusion capabilities while effectively mitigating the vanishing gradient problem. Meanwhile, the bottleneck structure improves computational efficiency and reduces the number of parameters through feature compression and reconstruction mechanisms [29]. Additionally, the SPPF module employs multi-scale pooling operations to expand the network's receptive field, allowing for better capture of features across different object scales. Through a series of convolutional operations and feature concatenation, the backbone ultimately

outputs feature maps that provide rich and precise spatial information for subsequent object detection tasks, ensuring that YOLOv8 achieves outstanding detection performance in complex scenarios.

B. Neck

In YOLOv8, the neck plays a pivotal role in feature extraction and fusion. It takes the output feature maps (P3, P4, P5) from the backbone and feeds them into the PAN-FPN network structure [30]. Through upsampling and downsampling operations, it adjusts the size and resolution of the feature maps—upsampling enlarges low-resolution feature maps to preserve detailed information, while downsampling reduces the size of high-resolution feature maps to decrease computational cost and memory usage. Compared to the traditional FPN, PAN-FPN introduces an additional bottom-up path aggregation, further enhancing the transmission of low-level features and enabling shallow features to integrate more effectively with deep semantic features. Additionally, lateral connections are employed in both top-down and bottom-up paths to align the resolutions of multi-level feature maps, ensuring efficient fusion of features across different levels.

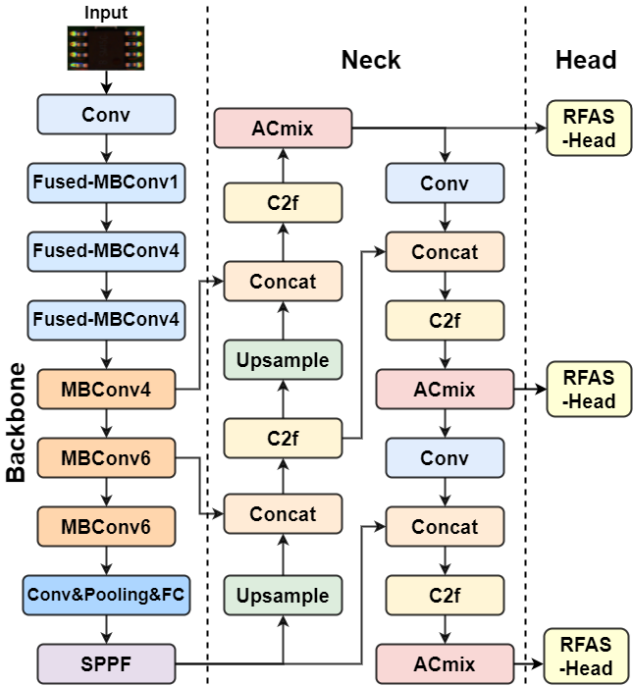

Fig. 2. Improved EAR-YOLO network structure diagram

C. Head

The head in YOLOv8 is responsible for executing the final object detection and classification tasks. It processes three feature maps of different sizes from the neck for further refinement. The head consists of two branches: one for bounding box prediction and the other for class prediction. Each branch undergoes a series of convolutional operations and is processed by prediction layers that include loss functions, enhancing feature representation capabilities. The prediction layers output three key values: bounding box regression, confidence scores, and class probabilities. Finally, non-maximum suppression (NMS) is applied to remove redundant detection boxes [31], retaining only the bounding boxes with the highest confidence, thus producing the final detection results.

III. Method

To improve the accuracy of PCB surface printing defect detection and address the detection needs in various scenarios, we propose a detection model named EAR-YOLO, with its network architecture illustrated in Figure 2. The following sections provide a detailed description of its enhanced structure.

A. Improved Backbone

EfficientNetV2 is an efficient, fast, and flexible neural network that is a next-generation architecture improved upon the EfficientNet model. The network reduces memory usage effectively by lowering the expansion ratio. Additionally, it replaces the larger 5×5 convolution kernel used in EfficientNetV1 with a more compact 3×3 convolution kernel, which not only facilitates deeper stacking of network layers to increase the receptive field, but also enhances computational efficiency. Furthermore, EfficientNetV2 incorporates the SE module concept from MobileNetV3 [32], which adaptively optimizes the weight distribution of feature channels, significantly improving the model's generalization ability. These innovations allow EfficientNetV2 to maintain high efficiency and speed while providing greater flexibility, making it suitable for a variety of computer vision tasks.

As shown in Figure 3a, MBConv is an inverted residual module applied in the shallow layers of EfficientNetV1. However, this structure exhibits speed limitations during the early stages of network training. To overcome this limitation, EfficientNetV2 introduces the Fused-MBConv structure, depicted in Figure 3b, which replaces the 1×1 expansion convolution and 3×3 depthwise separable convolution in the main branch of the original MBConv with a standard 3×3 convolution. By employing Neural Architecture Search (NAS), EfficientNetV2 identifies the optimal combination of MBConv and Fused-MBConv, resulting in the optimized network structure shown in Figure 2.Additionally, EfficientNetV2 incorporates an innovative progressive training strategy. During the early stages of model training, this strategy employs smaller image sizes and less aggressive regularization methods, enabling the network to quickly learn basic feature representations. As training progresses, the image size is gradually increased and regularization is intensified, thereby accelerating the training process while improving the model's final performance.

Precisely because of its lightweight structure, efficient computational performance, and excellent generalization capabilities, EfficientNetV2 was selected as the backbone network for our model. These features enable our model to maintain high accuracy while significantly reducing the number of parameters and computational overhead, greatly improving processing speed. Moreover, it demonstrates higher precision across various detection scenarios, thereby driving a comprehensive enhancement of the overall performance of the YOLOv8 model.

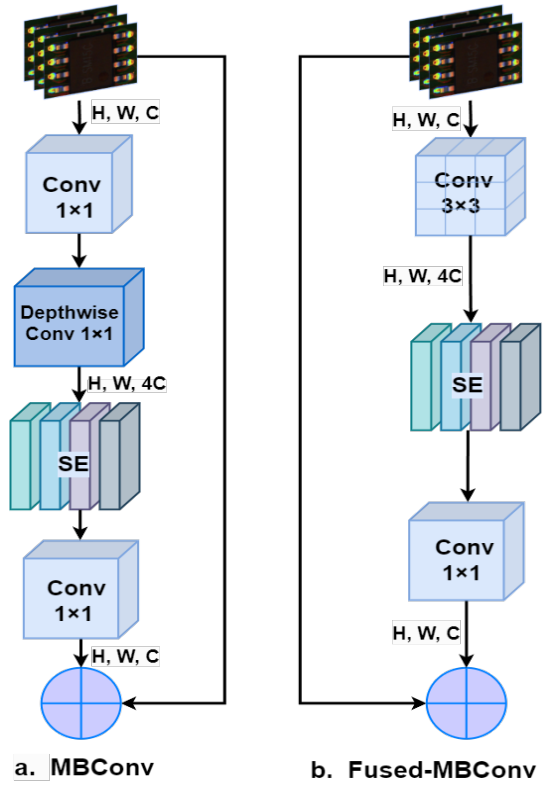

Fig. 3. Structure diagram of MBConv and Fused-MBConv (a) Structure diagram of MBConv; (b) Structure diagram of Fused-MBConv

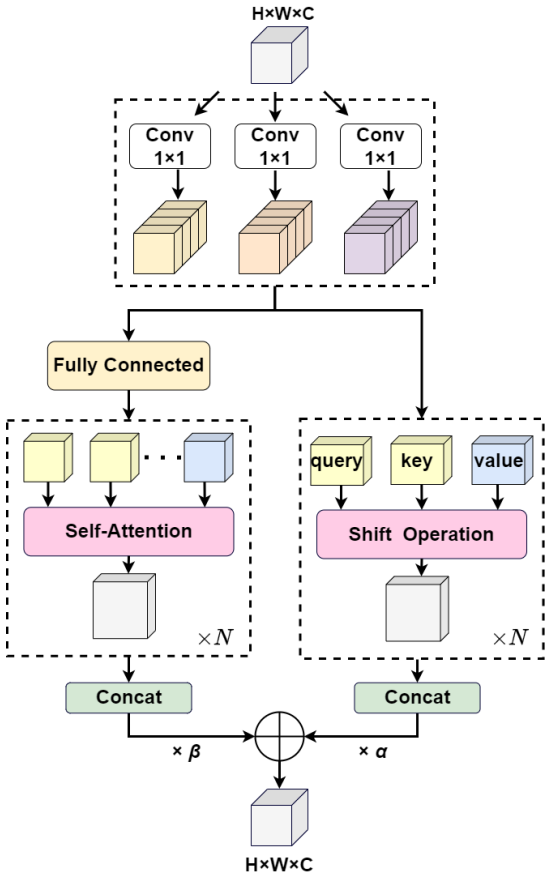

Fig. 4. Structure diagram of ACmix

B. Improved Neck

In the experimental evaluation of the YOLOv8n baseline model, we observed that the model had deficiencies in capturing detailed features of local defects,which led to frequent missed detections in practical applications. This indicates that the model's feature representation capabilities for specific datasets require improvement. As the depth of convolutional operations increases, the receptive field of the feature maps mapped back to the original image expands, which adversely affects the model's sensitivity to local positional information. To address this issue, we optimized the neck structure of the model by embedding an ACmix channel at the output of the P3, P4, and P5 layers. This design effectively integrates local and global feature extraction, enhancing the model's ability to understand and recognize features. Figure 4 illustrates the structure of ACmix, which consists of two primary stages aimed at synergizing global and local feature extraction capabilities.

In the first stage, the input features are transformed through three 1×1 convolutional layers and reorganized into N independent blocks, resulting in a rich intermediate feature set consisting of $3 \times N$ feature maps, where the intermediate feature atlas F_{mid} can be expressed by Equation (1).

$$\begin{aligned} F_{mid} = Concat(&Conv\ 1 \times 1_1(X), Conv\ 1 \times 1_2(X), \\ &Conv\ 1 \times 1_3(X)) \\ F_{mid} &\in \mathbb{R}^{H \times W \times 3N} \end{aligned} \tag{1}$$

The second stage is further divided into two subpaths: the convolutional path and the self-attention path.

1) In the convolutional path, a lightweight fully connected layer is used to project F_{mid}, generating k^2 feature maps, as shown in Equation (2).

$$F_K = FC(F_{mid}), \quad F_K \in \mathbb{R}^{H \times W \times k^2} \tag{2}$$

Simultaneously, a convolution kernel of size k is used for local feature extraction, resulting in F_{conv_out} as shown in Equation (3).

$$F_{conv\_out} = Conv_{k \times k}(F_K) \tag{3}$$

Finally, local information of the input features is gathered through shifting and aggregation operations to obtain F_{conv}, as shown in Equation (4).

$$F_{conv} = \sum_{m=1}^{k^2} Shift(F^m_{conv\_out}), \quad F_{conv} \in \mathbb{R}^{H \times W \times C} \tag{4}$$

2) In the self-attention path, the intermediate feature set is divided into N groups, with each group containing three feature maps. These feature maps originate from the 1×1 convolution layer and serve as query, key, and value, respectively. The attention weights A are then computed through the standard multi-head self-attention mechanism, and A can be represented by Equation (5).

$$A = Softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \tag{5}$$

Here, d_k is the scaling factor used to stabilize training, and Q and K stand for query and key, respectively.

Then, the weights are applied to the value, resulting in the self-attention output F_{att} as shown in Equation (6).

$$F_{att} = A \cdot V, \quad F_{att} \in \mathbb{R}^{H \times W \times C} \tag{6}$$

Finally, the outputs from the two paths are summed to obtain the output feature map F_{out}, as shown in Equation (7).

$$F_{out} = \alpha F_{att} + \beta F_{Conv} \tag{7}$$

Where α and β are the learning factors for the convolution path and the self-attention path, respectively.

After introducing the ACmix module, the network depth is enhanced, enabling the effective mapping of input feature maps by fusing multi-scale features to generate rich intermediate feature representations. These features are reused and aggregated based on the two distinct processing paradigms of self-attention and convolution. By adaptively adjusting the weights of features at different scales, the model achieves a significant improvement in target localization accuracy, thereby enhancing overall representational capability and robustness. The ACmix module decomposes and reconstructs self-attention and convolution operations into 1×1 convolutions, optimizing computational efficiency while meeting our requirements for improvements in the neck section. The multi-scale feature information processed by ACmix is subsequently passed to the head section for further refinement.

C. Improved Head

When optimizing the head section, we considered the limitations of traditional convolution in capturing the specific information differences of defects at various locations, which hinders the effective emphasis on the importance of each feature. Additionally, we observed that the imbalance in sample distribution among different defect categories on PCB surfaces is a persistent issue, leading to insufficient attention to hard samples during the training process. This, in turn, affects the detection accuracy and robustness of the model. To address these issues, enhance the model's focus on hard-to-recognize targets, and improve detection accuracy, we designed the RFASHead, as illustrated in Figure 5. In this design, the traditional convolution module is replaced with RFAConv, and the classification loss function is upgraded to SlideLoss.

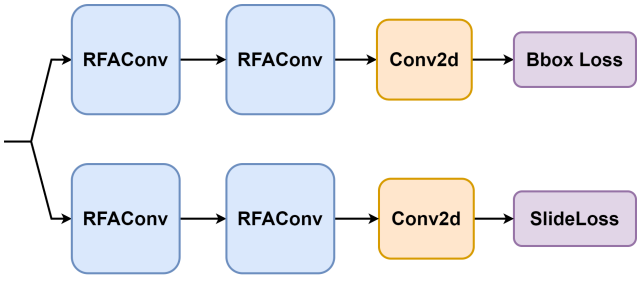

Fig. 5. Structure diagram of RFASHead

1) RFAConv: RFAConv is a convolutional layer design based on the Receptive Field Attention (RFA) mechanism, with its core lying in the integration of spatial attention into the convolution process. This design enables the model to better focus on spatial features within the receptive field, thereby enhancing its ability to deeply understand and effectively process local regions in an image. As shown in Figure 6, the structure of RFAConv with a 3×3 convolution kernel operates as follows: the process is first divided into two parallel parts, and the outputs of these two parts are subsequently reweighted to refine the features.

First, the first step performs global compression on the input features by applying average pooling (AvgPool) to the input tensor $X \in \mathbb{R}^{C \times H \times W}$. This operation reduces the spatial dimensions and captures global information for each channel, yielding the average value for each channel as shown in Equation (8):

$$\begin{gathered} AvgPool(X) = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{c,h,w}, \\ c = 1, 2, \ldots, C \\ AvgPool(X) \in \mathbb{R}^{C \times 1 \times 1} \end{gathered} \tag{8}$$

Subsequently, different 1×1 grouped convolutions $g^{1\times1}$ are applied to $AvgPool(X)$, producing three sets of receptive field feature maps that represent varying degrees of importance. This process is described in Equation (9):

$$g^{1\times1}(AvgPool(X)) = W \cdot AvgPool(X) + b \tag{9}$$

where $W \in \mathbb{R}^{C}$ represents the convolution kernel weights, and b is the bias term.

Then, the grouped features undergo a softmax operation to normalize the weights, resulting in the attention weight map A_{rf}, as shown in Equation (10):

$$A_{rf} = Softmax\left(g^{1\times1}\left(AvgPool(X)\right)\right) \tag{10}$$

Here, $A_{rf} \in \mathbb{R}^{C \times H \times W}$.

In the second part, 3×3 Group Convolution (Group Conv) is applied to the input features, and local sensory field information will be extracted individually for each group of features, thus generating a multi-group sensory field feature F_{rf} with feature tensor dimension $9C \times H \times W$ as shown in Equation (11).

$$F_{rf} = ReLU(Norm(g^{k\times k}(X))) \tag{11}$$

Secondly, feature re-weighting is applied to the outputs of the first and second parts by performing element-wise multiplication between the attention weights and the receptive field features, generating weighted receptive field feature maps that further enhance the model's ability to represent important regional features. Next, through Adjust Shape, the feature map dimensions are adjusted from $9C \times H \times W$ to $C \times 3H \times 3W$, thereby expanding the spatial resolution of the input features, preserving local information, and enhancing the receptive field range.

Finally, a 3×3 convolution operation with a stride of 3 is applied to compress the expanded feature maps back to the original input dimensions of $C \times H \times W$, ensuring that the final output dimensions of the feature maps remain consistent with the input.

This approach improves the accuracy of feature extraction and enables dynamic adjustment of convolution kernel parameters through the introduction of an attention mechanism, allowing it to adapt to the specific requirements of different regions. The computation process of RFAConv is represented by Equation (12) and Equation (13).

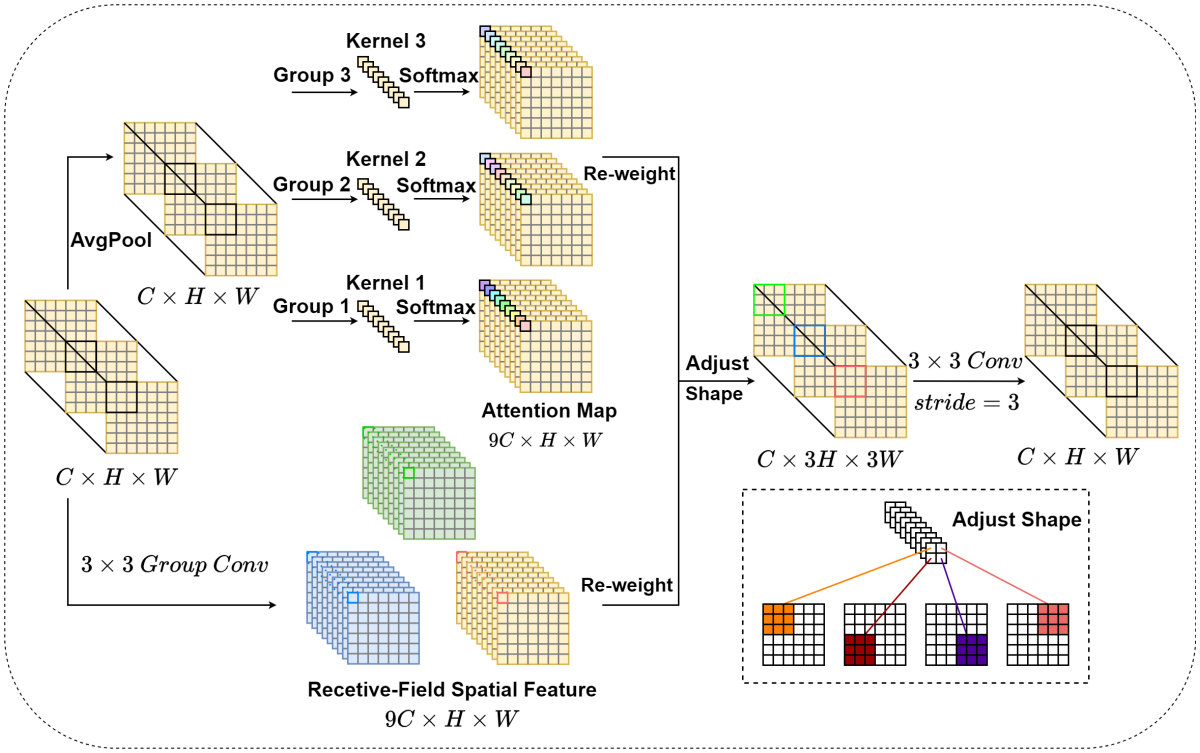

Fig. 6. Structure diagram of RFAConv

$$F = A_{rf} \times F_{rf} \tag{12}$$

$$F = Softmax\left(g^{1\times1}\left(AvgPool(X)\right)\right) \times ReLU\left(Norm\left(g^{k\times k}(X)\right)\right) \tag{13}$$

Here, $g^{(1\times1)}$ represents a 1×1 grouped convolution, k denotes the kernel size, $Norm$ indicates normalization, and X represents the input feature map. F is obtained by multiplying the attention map A_{rf} with the transformed receptive field spatial features F_{rf}.

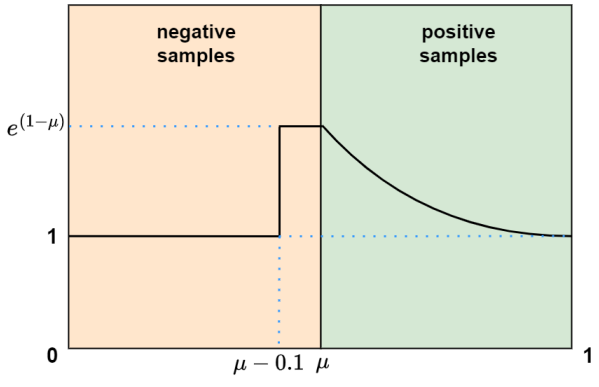

Fig. 7. SlideLoss adjustment mechanism

2) SlideLoss: In object detection tasks, the difficulty of samples is typically determined by the Intersection over Union (IoU) between the predicted and ground truth boxes. The classification boundary ambiguity of samples causes those near the boundary to bear greater pressure in the loss function. To effectively address this issue, the SlideLoss method adopts the average IoU value of all bounding boxes as the threshold μ, which is used to distinguish positive and negative samples: samples with IoU values below μ are categorized as negative samples, while those with IoU values above μ are regarded as positive samples. Furthermore, SlideLoss applies a specific weighting mechanism to samples near the classification boundary. This mechanism dynamically adjusts the loss weight of each sample based on its IoU value, thereby enhancing attention to boundary samples. Figure 7 vividly illustrates this dynamic adjustment process. Specifically, the weighting function of SlideLoss is expressed in Equation (14).

$$f(x) = \begin{cases} 1 & x \leq \mu - 0.1 \\ e^{1-\mu} & \mu - 0.1 < x < \mu \\ e^{1-x} & x \geq \mu \end{cases} \tag{14}$$

3) RFASHead: The design of RFAConv, by integrating spatial attention mechanisms with convolution operations, not only enhances the model's deep understanding and effective feature extraction for local regions of images but also improves its ability to recognize critical features through dynamic adjustment of convolution kernel parameters. Simultaneously, the introduction of SlideLoss effectively addresses the issue of sample imbalance. Specifically, for samples with ambiguous classification boundaries, the dynamic weighting mechanism increases the model's focus on these difficult samples. The improved RFASHead, integrating RFAConv and SlideLoss, achieves comprehensive performance enhancement in PCB surface defect detection tasks. These improvements enable the model to maintain computational efficiency while delivering more reliable and accurate defect detection, demonstrating superior generalization capabilities.

IV. Experiment

A. Dataset

The experiments were conducted using the publicly available datasets PCB-AOI and PKU-Market-PCB, with YOLOv8n as the baseline model to validate the effectiveness of EAR-YOLO. The PCB-AOI dataset is part of the open-source distributed synergy AI benchmarking project KubeEdge-Ianvs, released by the KubeEdge SIG AI members from China Telecom and Raisecom Technology. PCB-AOI contains two defect types: insufficient soldering (Bad_podu) and bridging (Bad_qiaojiao). Since the PCB-AOI dataset had been pre-divided (comprising 1211 augmented training images and 60 test images) at the time of its releaseand had already been trained and validated using the Ianvs algorithm, our experiments followed the original dataset division ratio to benchmark the experimental results against those obtained with the Ianvs algorithm. The PKU-Market-PCB dataset, published by the Peking University Open Lab of Intelligent Robotics, contains a total of 1386 images, of which 693 images have original annotations, while the remaining 693 are augmented through arbitrary-angle rotation without annotations. This dataset includes six types of defects: missing_hole, mouse_bite, open_circuit, short, spur, and spurious_copper. When using the PKU-Market-PCB dataset to evaluate the generalization performance of EAR-YOLO, only the 693 images with original annotations were used. These images were split into training and validation sets with an 8:2 ratio for training purposes.

B. Experimental Environment

The experimental environment was set up on a server equipped with the PyTorch deep learning framework, featuring the following configuration: Intel(R) Xeon(R) Silver 4215R 3.20GHz CPU, RTX 3090 24GB GPU, 100GB memory, and Ubuntu 22.04 operating system.

C. Model Evaluation Method

When evaluating the feasibility of a model, it is important to consider not only speed but also accuracy. Therefore, the experiment uses mAP@0.5 (mean Average Precision with the Intersection over Union threshold set to 0.5, meaning a prediction is considered correct if the IoU between the predicted and ground truth bounding boxes exceeds 0.5, hereafter referred to as mAP) as the metric to assess model accuracy. To calculate mAP, we first need to introduce precision and recall, whose formulas are shown in Equations (15) and (16).

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

In multi-object detection tasks, the Average Precision (AP) for each category is typically computed first, followed by the mAP of the model, as given in Equations (17) and (18). FPS (Frames Per Second) is used as the metric to evaluate the detection speed of the model.

$$AP = \int_0^1 P(R)\, dR \tag{17}$$

$$mAP = \frac{1}{n} \sum_{i=1}^{n} AP_i \tag{18}$$

Here, TP represents True Positive, FP stands for False Positive, FN denotes False Negative, n refers to the number of defect categories, and i indicates the detection instances.

D. Detection Performance Comparison

To further validate the improvement brought by the EAR-YOLO algorithm, we conducted comparative experiments with other mainstream algorithms, including KubeEdge-Ianvs (hereinafter referred to as Ianvs), YOLOv3, YOLOv5, YOLOX, YOLOv7, and YOLOv8. In the experiments, the batch size was set to 16, the momentum to 0.937, and the initial learning rate lr_0 to 0.01. The number of epochs was set to 300 for the PCB-AOI dataset and 400 for the PKU-Market-PCB dataset. Other parameters were kept as default.

1) Experiments on the PCB-AOI Dataset: In the experiments conducted on the PCB-AOI dataset, we included the Ianvs algorithm provided by the dataset publisher. Ianvs is a distributed synergy AI benchmarking testing tool developed based on the Faster R-CNN framework. The other algorithms and their corresponding models involved in the comparative experiments were as follows: YOLOv3 (yolov3_tiny), YOLOv5 (yolov5s), YOLOX (yolox_s), YOLOv7 (yolov7), YOLOv8 (yolov8n), and EAR-YOLO (yolov8n). The experimental results are shown in Table 1, where the defect types are represented as follows:Bp(Bad_podu),Bq(Bad_qiaojiao).

TABLE I
Comparison of EAR-YOLO with other algorithms on the PCB-AOI dataset

Models	mAP	Bp	Bq	Recall	Precision	FPS
Ianvs	0.738	0.812	0.664	0.811	0.789	37.36
YOLOv3	0.629	0.792	0.466	0.670	0.602	200.00
YOLOv5	0.698	0.835	0.561	0.735	0.663	98.04
YOLOX	0.716	0.815	0.617	0.615	**0.812**	46.87
YOLOv7	0.705	0.826	0.583	0.695	0.717	82.64
YOLOv8	0.744	0.794	0.694	0.723	0.778	**208.33**
EAR-YOLO	**0.821**	**0.843**	**0.799**	**0.846**	0.732	108.70

By comparing the experimental data in Table 1, the proposed EAR-YOLO model achieved the highest mAP of 82.1%, outperforming the Ianvs algorithm by 8.3% and surpassing other YOLO versions (YOLOv3, YOLOv5, YOLOX, YOLOv7, and YOLOv8) by margins ranging from 7.7% to 19.2%. Notably, EAR-YOLO achieved the highest average precision for defect types "Bad_podu" (84.3%) and "Bad_qiaojiao" (79.9%), as well as the highest recall of 84.6%.Although EAR-YOLO did not exhibit the best performance in precision and FPS, its metrics in these aspects remained at high levels. Overall, the EAR-YOLO model proposed in this study not only significantly improved detection accuracy but also maintained considerable detection efficiency, showcasing its superior comprehensive performance.1

2) Experiments on the PKU-Market-PCB Dataset: To evaluate the generalization performance of the proposed improved model across different datasets and its performance on small-sample datasets, experiments were conducted using the PKU-Market-PCB dataset. In these experiments, EAR-YOLO was compared with YOLOv3 (yolov3_tiny), YOLOv5 (yolov5s), YOLOX (yolox_s), YOLOv7 (yolov7), and YOLOv8 (yolov8n). The experimental results are shown in Tables 2 and 3, where the defect types are represented as follows: Mh (Missing_hole), Mb (Mouse_bite), Oc (Open_circuit), Sh (Short), Sp (Spur), and Sc (Spurious_copper).

TABLE II
COMPARISON OF MAP FOR VARIOUS DEFECTS BETWEEN EAR-YOLO AND OTHER ALGORITHMS ON THE PKU-MARKET-PCB DATASET.

Models	Mh	Mb	Oc	Sh	Sp	Sc
YOLOv3	0.995	0.860	0.894	0.980	0.846	0.891
YOLOv5	0.995	0.860	0.899	**0.989**	0.868	0.869
YOLOX	**1.000**	0.899	0.900	0.903	0.903	0.901
YOLOv7	0.996	0.852	0.919	0.967	0.821	0.913
YOLOv8	0.995	0.846	0.907	0.978	0.856	0.900
EAR-YOLO	0.995	**0.932**	**0.973**	0.986	**0.920**	**0.976**

TABLE III
COMPARISON OF EAR-YOLO WITH OTHER ALGORITHMS ON THE PKU-MARKET-PCB DATASET.

Models	mAP	Recall	Precision	FPS
YOLOv3	0.911	0.884	0.941	133.33
YOLOv5	0.913	0.872	0.944	91.74
YOLOX	0.917	0.859	0.939	55.99
YOLOv7	0.911	0.901	0.954	77.52
YOLOv8	0.914	0.874	0.925	**169.49**
EAR-YOLO	**0.964**	**0.932**	**0.957**	117.65

Based on the data in Tables 2 and 3, the EAR-YOLO model demonstrates outstanding performance in terms of mAP, achieving a remarkable 96.4%, which is at the top of the algorithms and 5% higher than the YOLOv8 benchmark model. In the average precision evaluation across the six defect types, EAR-YOLO achieved the highest precision for Mouse bite, Open circuit, Spur, and Spurious copper defects, while also exhibiting commendable precision for the other two defect types. In terms of recall and precision, EAR-YOLO again leads with values of 93.2% and 95.7%, respectively. Overall, the EAR-YOLO model not only excels in precision on the PKU-Market-PCB dataset, but also maintains impressive detection speed.

E. Visualization and Analysis

To provide a clearer visualization of the improvements brought by the EAR-YOLO model compared to the YOLOv8 benchmark model, both the actual detection results and training results were visualized.

1) Visualization Comparison of Actual Detection Results: To compare the detection performance of EAR-YOLO and YOLOv8 in practical applications, several images were randomly selected from the PCB-AOI and PKU-Market-PCB datasets for testing with both models. The test results are shown in Figures 8 and 9.

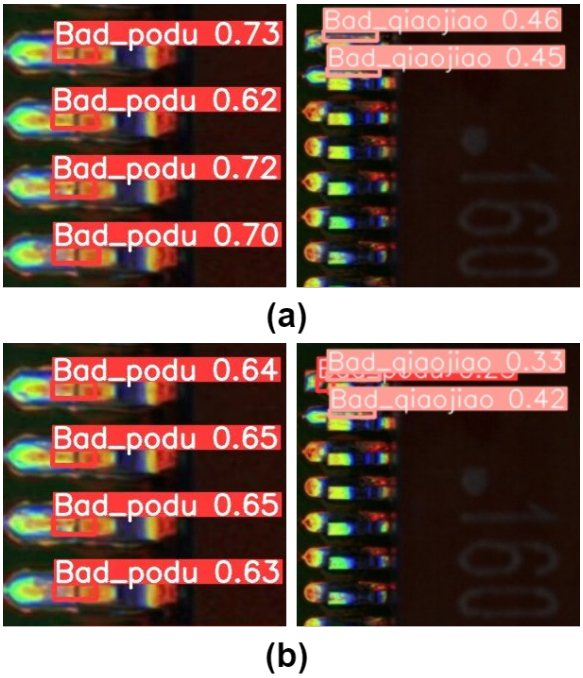

Fig. 8. Detection results of each model based on PCB-AOI dataset. (a) Results of the EAR-YOLO model; (b) Results of the YOLOv8 model.

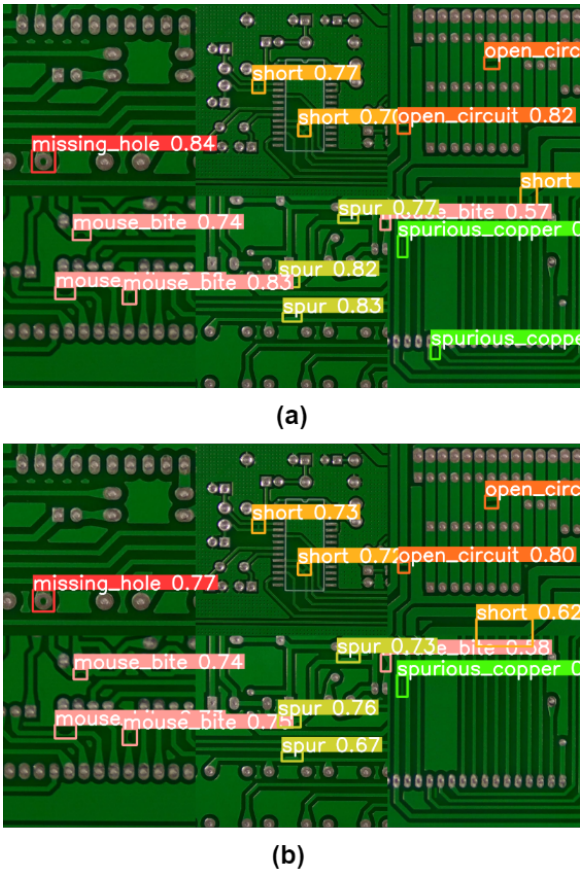

Fig. 9. Detection results of each model based on PKU-Market-PCB dataset. (a) Results of the EAR-YOLO model; (b) Results of the YOLOv8 model.

Figure 8 illustrates the detection performance of each model on the PCB-AOI dataset, with each experiment group including two test images. The results indicate that both models accurately detect the Bad_pudo defects in the first test image, but the confidence scores of most detections

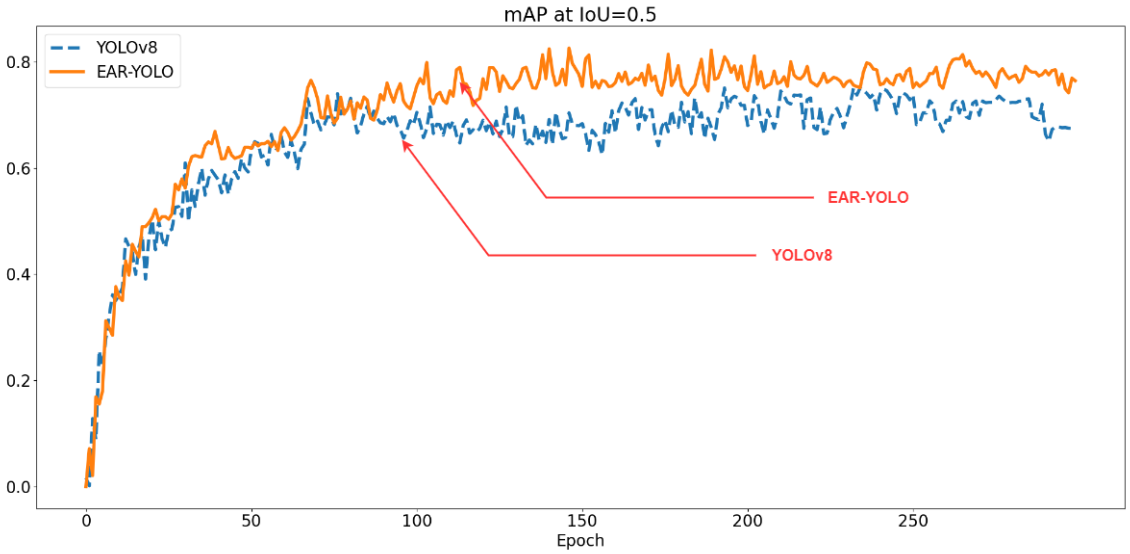

Fig. 10. Training process based in the PCB-AOI dataset.

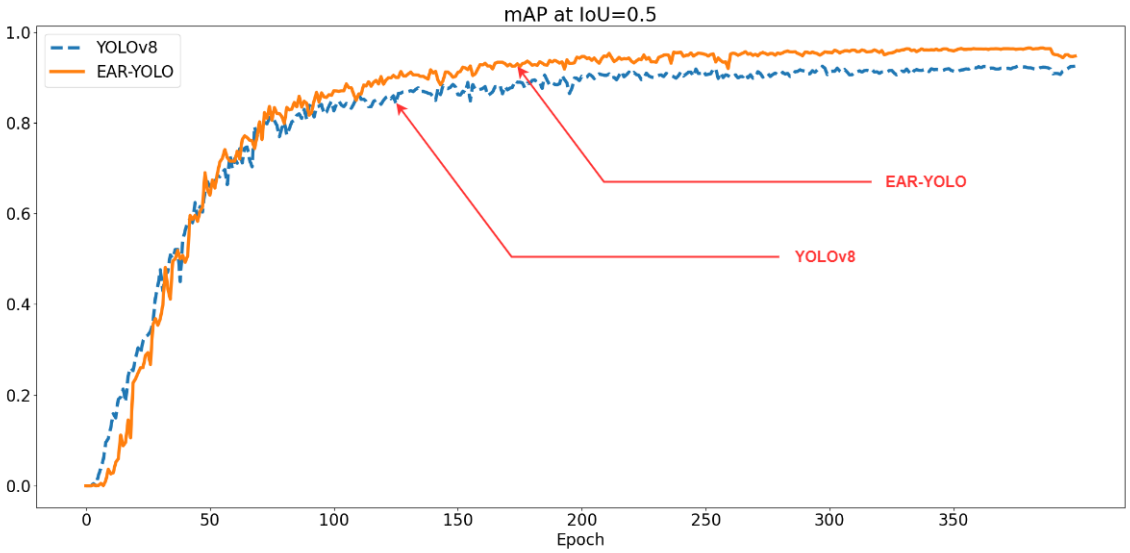

Fig. 11. Training process based in the PKU-Market-PCB dataset.

by the EAR-YOLO model are higher than those of the YOLOv8 model. In the second test image, which contains only two bridging defects, EAR-YOLO accurately identifies all Bad_qiaojiao defects, whereas the YOLOv8 model shows false positives and has lower confidence scores compared to EAR-YOLO.

Figure 9 presents the detection performance of each model on the PKU-Market-PCB dataset, where each test image is formed by stitching six subimages together. In the process of identifying and localizing PCB defects, YOLOv8 exhibits missed detections for spurious_copper defects, whereas EAR-YOLO demonstrates superior performance in this aspect. Regarding the detection accuracy for each defect type, EAR-YOLO consistently outperforms YOLOv8. Overall, EAR-YOLO achieves higher accuracy in both defect localization and detection precision compared to YOLOv8.

In conclusion, EAR-YOLO exhibited outstanding performance across all detection tasks, with almost no missed or false detections, demonstrating its exceptional detection capability and robustness.

2) Visualization Comparison of Training Results: Figures 10 and 11 present the training results in the form of plotted curves, using mAP at IoU=0.5 as the evaluation metric for model comparison. Specifically, Figure 10 illustrates the training process in the PCB-AOI dataset, while Figure 11 corresponds to the training process in the PKU-Market-PCB dataset.

As observed from Figures 10 and 11, the EAR-YOLO model demonstrates a significant advantage over the YOLOv8 model in terms of mAP during the training process in both the PCB-AOI and PKU-Market-PCB datasets. EAR-YOLO not only achieves a rapid increase in mAP at the early stage of training, but also maintains a higher growth rate throughout the entire training phase, ultimately reaching a superior mAP compared to YOLOv8. This indicates that EAR-YOLO possesses stronger learning and generalization capabilities, enabling more effective target recognition on the same datasets. Moreover, the fast convergence characteristic

of EAR-YOLO suggests that it may require less training time to achieve satisfactory detection accuracy in practical applications, making it more efficient than YOLOv8.

F. Ablation Experiment

To investigate the specific impact of EfficientNetV2, ACmix and RFASHead on the performance improvement of YOLOv8, this study adopts a rigorous experimental design. Since our model is built upon YOLOv8n, we selected YOLOv8n as the baseline model for the ablation study. On the PCB-AOI dataset, we utilized mAP, precision, and recall as performance evaluation metrics. The relevant experimental results are detailed in Table 4,where EfficientNetV2, ACmix, and RFASHead are abbreviated in the table as Effi, AC, and RFAS, respectively. Recall and precision are abbreviated as R and P, respectively.

TABLE IV
RESULTS OF ABLATION EXPERIMENTS ON PCB-AOI DATASET.

Model(YOLOv8n)	Effi	AC	RFAS	mAP	R	P
A				0.744	0.723	0.778
B	✓			0.766	0.742	0.739
C		✓		0.758	0.717	0.756
D			✓	0.759	0.728	0.772
E		✓	✓	0.765	0.755	**0.798**
F	✓		✓	0.776	0.783	0.708
G	✓	✓		0.775	0.792	0.752
H	✓	✓	✓	**0.821**	**0.846**	0.732

Due to the uneven distribution of defect types and the rich detail of defects in the PCB-AOI dataset, which are difficult to capture, the baseline model A exhibited the poorest performance in detection accuracy. Model B, after introducing EfficientNetV2 as the backbone network and adopting a progressive training strategy, achieved a significant improvement in detection speed. Among them, Models C and D demonstrated performance enhancements upon integrating individual submodules. Model C, integrated with ACmix, enhanced network depth, while Model D optimized convolution operations by incorporating RFAConv and further improved focus on defect samples through SlideLoss. Model E, which combined ACmix and RFASHead, deepened the understanding of features and significantly improved mAP and precision compared to Models A, C, and D through RFASHead's in-depth processing. Models F and G, based on EfficientNetV2, combined with ACmix and RFASHead, respectively, further enhanced mAP and Recall compared to Model B. Finally, Model H, integrating all the improved modules, achieved a comprehensive performance improvement. Although its precision slightly declined, its overall performance far exceeded that of the baseline model.

V. CONCLUSION

To address the issues of poor generalization, low detection accuracy, and slow processing speed in existing PCB surface defect detection models, this paper proposes an improved PCB surface defect detection model based on YOLOv8, named EAR-YOLO. The model adopts EfficientNetV2, enhanced with a progressive training strategy, as its backbone network to accelerate training and improve generalization capabilities. To address the challenge of low detection accuracy, an ACmix channel is added to the feature output layer in the neck of the model, increasing the depth of feature learning, enhancing feature extraction capabilities, and reducing computational overhead. Finally, the newly designed detection head, RFASHead, is introduced to improve the adaptability and feature extraction accuracy of the model for different samples. Experimental results on the public datasets PCB-AOI and PKU-Market-PCB demonstrate that EAR-YOLO achieves mAPs of 82.1% and 96.4%, respectively, which are 7.7% and 5% higher than the YOLOv8n baseline model. Additionally, the FPS reaches 108.70 and 117.65, respectively, meeting the requirements of PCB industrial defect detection.

Due to the addition of a new detection layer after the improvement, the complexity of the model structure increases, resulting in an increase in the model FLOPs, and there is still potential for enhancement of the computational resource consumption. Therefore, the next step of the research focuses on reducing the model resource consumption and the number of parameters to make the model lightweight without affecting the model detection effectiveness.

REFERENCES

[1] J. D. Song, Y. G. Kim, and T. H. Park, "SMT Defect Classification by Feature Extraction Region Optimization and Machine Learning," *The International Journal of Advanced Manufacturing Technology*, vol. 101, pp. 1303–1313, 2019.

[2] J. Schlarp, E. Csencsics, and G. Schitter, "Optical Scanning of a Laser Triangulation Sensor for 3-D Imaging," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3606–3613, 2019.

[3] P. Wang, E. Fan, and P. Wang, "Comparative Analysis of Image Classification Algorithms Based on Traditional Machine Learning and Deep Learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.

[4] T. Evgeniou and M. Pontil, "Support Vector Machines: Theory and Applications," in *Machine Learning and its Applications. ACAI 1999. Lecture Notes in Computer Science*. Springer, 2001, vol. 2049, pp. 249–257.

[5] Z. Lu, Q. He, X. Xiang, and H. Liu, "Defect Detection of PCB Based on Bayes Feature Fusion," *The Journal of Engineering*, vol. 2018, no. 16, pp. 1741–1745, 2018.

[6] Y. Li and S. Li, "Defect Detection of Bare Printed Circuit Boards Based on Gradient Direction Information Entropy and Uniform Local Binary Patterns," *Circuit World*, vol. 43, no. 4, pp. 145–151, 2017.

[7] H. Hagi, Y. Iwahori, S. Fukui, Y. Adachi, and M. K. Bhuyan, "Defect Classification of Electronic Circuit Board Using SVM Based on Random Sampling," *Procedia Computer Science*, vol. 35, pp. 1210–1218, 2014.

[8] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A Survey of Deep Learning and its Applications: a New Paradigm To Machine Learning," *Archives of Computational Methods in Engineering*, vol. 27, pp. 1071–1092, 2020.

[9] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep Learning Applications and Challenges in Big Data Analytics," *Journal of Big Data*, vol. 2, pp. 1–21, 2015.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot Multibox Detector," in *Computer Vision – ECCV 2016*, vol. 9905. Springer International Publishing, 2016, pp. 21–37.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[13] F. Liang, Y. Zhou, X. Chen, F. Liu, C. Zhang, and X. Wu, "Review of Target Detection Technology Based on Deep Learning," in *Proceedings of the 5th International Conference on Control Engineering and Artificial Intelligence*. Association for Computing Machinery, 2021, pp. 132–135.

[14] Y. Liu and Y. Tian, "DCMS-YOLOv5: A Dual-Channel and Multi-Scale Vertical Expansion Helmet Detection Model Based on YOLOv5." *Engineering Letters*, vol. 31, no. 1, pp. 373–379, 2023.

[15] R. Ding, L. Dai, G. Li, and H. Liu, "TDD-Net: A Tiny Defect Detection Network for Printed Circuit Boards," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 2, pp. 110–116, 2019.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[17] B. Hu and J. Wang, "Detection of PCB Surface Defects with Improved Faster-RCNN and Feature Pyramid Network," *IEEE Access*, vol. 8, pp. 108 335–108 345, 2020.

[18] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet V2: Practical Guidelines for Efficient CNN Architecture Design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[19] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, "Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural networks," *Electronics*, vol. 9, no. 9, p. 1547, 2020.

[20] W. Dai, A. Mujeeb, M. Erdt, and A. Sourin, "Soldering Defect Detection in Automatic Optical Inspection," *Advanced Engineering Informatics*, vol. 43, p. 101004, 2020.

[21] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures," *Frontiers in Neuroscience*, vol. 13, p. 95, 2019.

[22] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, "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. 3, pp. 4411–4428, 2021.

[23] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 390–391.

[24] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[25] M. Tan and Q. Le, "Efficientnetv2: Smaller Models and Faster Training," in *International Conference on Machine Learning*, vol. 139. PMLR, 2021, pp. 10 096–10 106.

[26] X. Pan, C. Ge, R. Lu, S. Song, G. Chen, Z. Huang, and G. Huang, "On the Integration of Self-Attention and Convolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 815–825.

[27] X. Zhang, C. Liu, D. Yang, T. Song, Y. Ye, K. Li, and Y. Song, "RFAConv: Innovating Spatial Attention and Standard Convolutional Operation," *ArXiv:2304.03198*, 2023.

[28] Z. Yu, H. Huang, W. Chen, Y. Su, Y. Liu, and X. Wang, "Yolo-facev2: A Scale and Occlusion Aware Face Detector," *Pattern Recognition*, vol. 155, p. 110714, 2024.

[29] D. Zhou, Q. Hou, Y. Chen, J. Feng, and S. Yan, "Rethinking Bottleneck Structure for Efficient Mobile Network Design," in *Computer Vision – ECCV 2020. Lecture Notes in Computer Science*, vol. 12348. Springer, 2020, pp. 680–697.

[30] G. Wang, Y. Chen, P. An, H. Hong, J. Hu, and T. Huang, "UAV-YOLOv8: A Small-Object-Detection Model Based on Improved YOLOv8 for UAV Aerial Photography Scenarios," *Sensors*, vol. 23, no. 16, p. 7190, 2023.

[31] S. Li, X. Zhang, and R. Shan, "Enhanced YOLOv5 for Efficient Marine Debris Detection," *Engineering Letters*, vol. 32, no. 8, pp. 1585–1593, 2024.

[32] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for Mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.