symmetry

MDPI

Article

LW-YOLO: Lightweight Deep Learning Model for Fast and Precise Defect Detection in Printed Circuit Boards

Zhaohui Yuan *, Xiangyang Tang *, Hao Ning and Zhengzhe Yang

School of Software, East China Jiaotong University, Nanchang 330013, China; 2022218083500011@ecjtu.edu.cn (H.N.); 2022218083500019@ecjtu.edu.cn (Z.Y.)
* Correspondence: yuanzh@whu.edu.cn (Z.Y.); xytang@ecjtu.edu.cn (X.T.)

Abstract: Printed circuit board (PCB) manufacturing processes are becoming increasingly complex, where even minor defects can impair product performance and yield rates. Precisely identifying PCB defects is critical but remains challenging. Traditional PCB defect detection methods, such as visual inspection and automated technologies, have limitations. While defects can be readily identified based on symmetry, the operational aspect proves to be quite challenging. Deep learning has shown promise in defect detection; however, current deep learning models for PCB defect detection still face issues like large model size, slow detection speed, and suboptimal accuracy. This paper proposes a lightweight YOLOv8 (You Only Look Once version 8)-based model called LW-YOLO (Lightweight You Only Look Once) to address these limitations. Specifically, LW-YOLO incorporates a bidirectional feature pyramid network for multiscale feature fusion, a Partial Convolution module to reduce redundant calculations, and a Minimum Point Distance Intersection over Union loss function to simplify optimization and improve accuracy. Based on the experimental data, LW-YOLO achieved an mAP0.5 of 96.4%, which is 2.2 percentage points higher than YOLOv8; the precision reached 97.1%, surpassing YOLOv8 by 1.7 percentage points; and at the same time, LW-YOLO achieved an FPS of 141.5. The proposed strategies effectively enhance efficiency and accuracy for deep-learning-based PCB defect detection.

Keywords: printed circuit boards (PCBs); defect detecting; deep learning; lightweight model; multi-scale feature fusion

Citation: Yuan, Z.; Tang, X.; Ning, H.; Yang, Z. LW-YOLO: Lightweight Deep Learning Model for Fast and Precise Defect Detection in Printed Circuit Boards. *Symmetry* **2024**, *16*, 418. https://doi.org/10.3390/sym16040418

Academic Editor: Changxin Gao

Received: 1 March 2024
Revised: 26 March 2024
Accepted: 28 March 2024
Published: 3 April 2024

1. Introduction

As the complexity of PCB (printed circuit board) manufacturing processes escalates, even minor defects can significantly impair product performance, leading to diminished yield rates [1]. It is critical to precisely identify defects such as shorts, open circuits, spurs, spurious copper, mouse bites, and missing holes [2] in PCB production and usage to enhance product yield. Traditional PCB defect detection methods encompass visual inspection and automated defect detection technology [3,4]. The former hinges on manual observation for the identification of small surface defects, demanding sustained attention and being prone to visual fatigue and distraction-induced errors [5]. The latter comprises methods like optical inspection, X-ray inspection, and infrared thermal imaging [6], providing high detection efficiency and accuracy but necessitating costly equipment and time. These methods may even inflict damage on the PCB. Utilizing symmetry features for PCB defect detection can effectively identify and classify defects such as scratches, cracks, stains, and missing solder joints, enabling timely rectification of potential issues. Zhang proposed a method based on convolutional neural networks for accurately classifying and recognizing symmetry in planar engineering structures [7]. However, the classification accuracy achieved was 86.69%, indicating a need for further improvement. Hence, devising high-precision and high-efficiency methods for detecting PCB defects continues to pose a challenge.

Deep learning image recognition algorithms have demonstrated impressive success across various fields [8–11]. Their integration with PCB defect detection displays good

adaptability [12–14] and has shown progress. The deep learning object detection method involves training neural network models to identify PCB surface defects, including steps such as constructing annotated datasets, designing network architectures, and adjusting training parameters and strategies. During the training process, weight parameters are continuously updated to reduce loss values, improve accuracy, and ultimately evaluate model performance through valid or test sets. Research results indicate that this method is effective in detecting PCB defects. Wei et al. utilized Automatic Optical Inspection (AOI) technology and CNN for PCB defect detection [15], demonstrating higher accuracy and stability compared with traditional methods. Due to the complexity of AOI technology involving multiple fields such as computer science, optoelectronics, machine vision, and pattern recognition, further improvements are needed in algorithm real-time performance, preprocessing accuracy, and defect classification precision. Kaya et al. proposed a method utilizing a deep-learning-based hybrid optical sensor for the noncontact detection and classification of small-sized defects on large PCBs, addressing issues of time consumption and fatigue [16]. However, the method involves the use of optical microscopic sensors, which introduces operational complexity. Kim et al. proposed a PCB detection system based on a skip-connection convolutional autoencoder, achieving efficient defect detection [17]. However, this method uses manually generated PCB defect images, which requires further model verification of actual PCB defect data, and the inference speed needs to be improved.

Ding et al. [2] proposed a small defect detection network called TDD-Net, achieving notable results by leveraging the multiscale and pyramid-level characteristics of deep convolutional networks in constructing a feature pyramid with high portability. Hu et al. [18] proposed an improved Faster-R-CNN model, utilizing ResNet50 as the backbone network and fusing GARPN (Region Proposal by Guided Anchoring) and ShuffleNetV2 residual units. Peishu Wu et al. [19] proposed an improved multiscale small target detection method and validated its efficacy and satisfactory results on the PCB dataset. Liao et al. [20] enhanced the YOLOv4 network, proposed the YOLOv4-MN3 model, and achieved accurate surface defects detection on PCB using the MobileNetV3 backbone network and optimization strategy. The experimental results reveal that the model attained performance of up to 98.64% mAP and 56.98 FPS (Frames Per Second) on six different surface defects. Cheng et al. [21] proposed a fast Tiny RetinaNet network to address the class imbalance problem by mitigating the influence of easily classified samples on classification results. Yu et al. [22] improved small defect detection by proposing DFP (diagonal feature pyramid) to enhance performance. Other deep-learning-based PCB defect detection algorithms include [23,24], but most of these methods generally suffer from large model sizes and slow detection speed issues. For instance, the parameters of the aforementioned models usually exceed 60 MB, and the detection speed typically falls below 90 FPS, making it challenging to meet the high-performance demands of PCB industrial production.

To effectively augment the efficiency of deep-learning-based PCB defect detection algorithms and achieve lightweight detection models, this paper posits the following improvement strategies based on previous work: Firstly, we design a multiscale feature fusion structure founded on the characteristics of the BiFPN (Bidirectional Feature Pyramid Network) [25] network structure to tackle the issue of missing and displaced features in FPN (Feature Pyramid Network) [26] and PAN (Path Aggregation Network) [27] structures, thereby enhancing model accuracy while decreasing model size. Secondly, to address the issue of large calculation load and slow inference speed caused by the Bottleneck structure used in C2f (CSPDarknet53 to 2-Stage FPN), we utilize PConv (Partial Convolution) to optimize the conventional convolution structure based on the FasterNet [28] network structure characteristics, effectively reducing calculation load and boosting model inference speed. Thirdly, to solve the problem that the CIoU (Complete Intersection over Union) loss function cannot optimize prediction boxes and ground truth boxes when they share the same aspect ratio, leading to a complex computation process, we optimize the loss function based on MPDIoU (Minimum Point Distance Intersection over Union) [29] to improve accuracy and simplify the computation process.

Drawing on the above research and thinking in addressing the issue of defect detection in printed circuit boards, this study is grounded in the relevant literature [25,26,28,30], with a focus on exploring the optimization of detection accuracy and inference speed. A novel improved model, LW-YOLO, is proposed for this problem, which integrates three optimization strategies, including network structure enhancement and algorithm optimization, leading to improved detection accuracy and inference speed. The primary contributions of this paper are as follows:

1. We propose a straightforward and efficient bidirectional feature pyramid network to effectively fuse features of different scales, thereby improving object detection accuracy.
2. We optimize the Bottleneck convolution structure based on the FasterNet module network structure design and introduce it into the new model, reducing redundant calculations and memory access while enhancing detection speed.
3. We introduce a new loss measurement method based on MPDIoU, which overcomes existing loss function limitations and improves object detection task convergence speed and regression result accuracy.

The rest of this paper is organized as follows: Section 2 introduces the proposed method in detail, and Section 3 introduces image preprocessing and related information about the dataset used in this paper. In Section 4, we evaluate the performance of this model through ablation experiments and comparative experiments. Finally, Section 5 summarizes the entire paper.

2. Methodology

2.1. YOLOv8 Model

The YOLOv8 algorithm is an efficient one-stage object detection method that consists of several key modules: input preprocessing, backbone network, neck module, and output processing. In the input preprocessing stage, the algorithm processes the input image through techniques such as data augmentation, adaptive anchoring calculation, and adaptive grayscale filling. The backbone network and neck module are the core of the YOLOv8 network, extracting features of different scales from the input image through a series of convolution and feature extraction operations. The neck module, an improvement on the C3 module, leverages the advantages of the ELAN structure [31] and introduces bottleneck modules to enhance gradient branches for better capture of gradient flow information. Overall, the YOLOv8 algorithm retains its lightweight characteristics while enhancing the richness of gradient information. Figure 1 illustrates the basic structure of this algorithm.

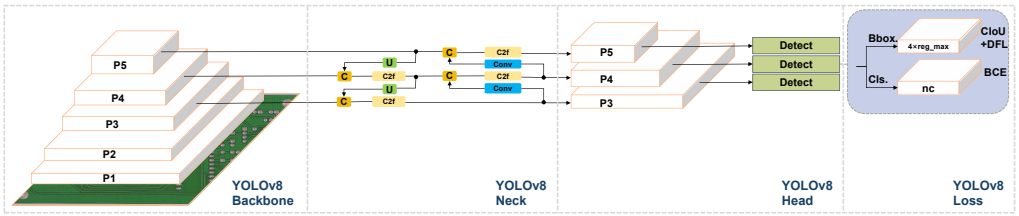

Figure 1. YOLOv8 network structure diagram.

The backbone network and Neck section integrate the design principles of YOLOv7 ELAN, replacing the C3 structure in YOLOv5 with the C2f structure to enhance feature representation capability. Additionally, they employ a decoupled head structure to separate classification and detection tasks. The introduction of TaskAlignedAssigner for positive sample assignment strategy and Distribution Focal Loss for loss calculation comprehensively improves object detection performance.

2.2. LW-YOLO Model

This study draws upon the YOLOv8 model, proposing three enhancements to address its limitations, resulting in a novel network model termed LW-YOLO. The network structure

of LW-YOLO is depicted in Figure 2, while Figure 3 demonstrates the detailed structure of LW-YOLO, which comprises three components, Backbone, Neck, and Head, which are elaborated upon in the following sections.

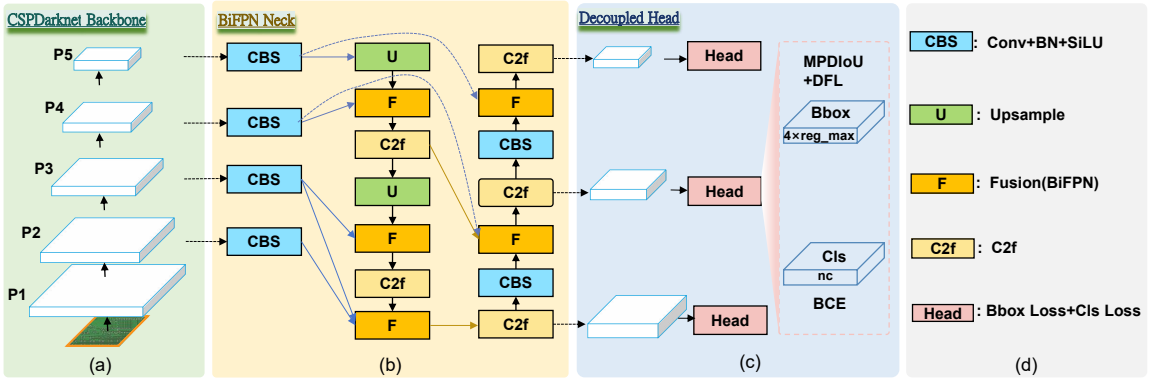

Figure 2. The network architecture of LW-YOLO: (**a**) The C2f in the backbone network undergoes lightweight transformation. (**b**) BiFPN is introduced as the Neck to enable feature propagation across different levels. (**c**) MPDIoU is used as the loss function to simplify the computation. (**d**) Module Explanation.

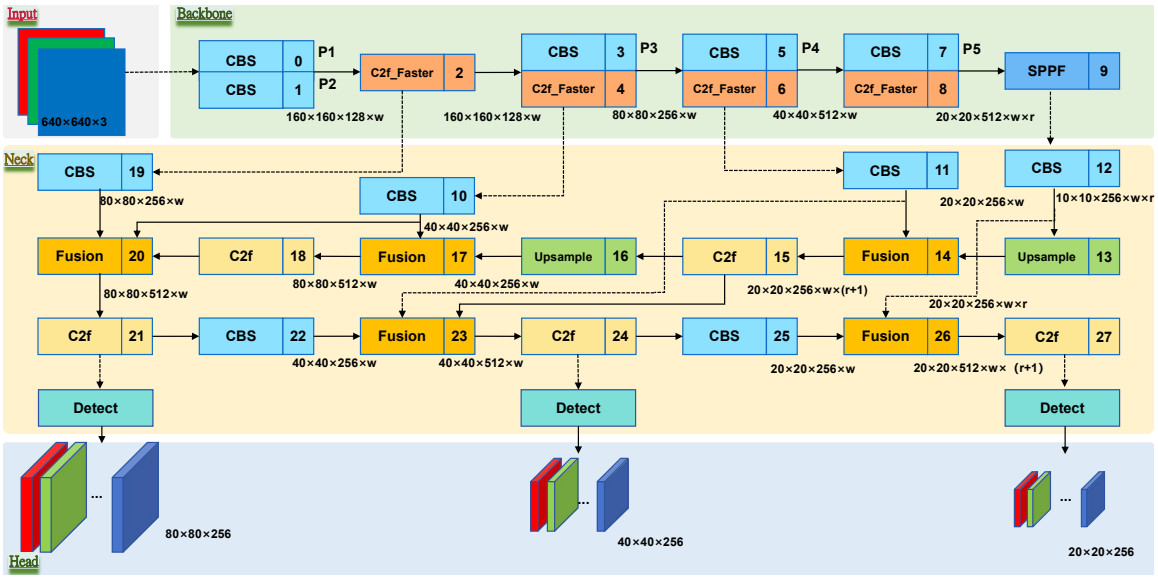

Figure 3. The LW-YOLO adopts a flexible network structure, where the size of feature maps is controlled by parameters w (width) and r (ratio). By adjusting these parameter values, the model size can be customized to meet specific requirements for different application scenarios.

The Backbone extracts the feature representation of the input image and conveys it to subsequent layers for detection. This component incorporates a convolutional neural network, specifically CSPDarknet53 [32], renowned for its high precision, lightweight design, robust feature extraction capability, and transferability. The C2f module employs the compact FasterNet module structure to minimize the model's size without significantly compromising accuracy.

The Neck is utilized to extract multiscale features based on the backbone network. In LW-YOLO, we integrated the BiFPN structure, which effectively disseminates both high-level semantic information and low-level detail information. This integration aids in accurately locating and recognizing objects, delivering more precise detection results across various scales. Furthermore, BiFPN incorporates a lightweight feature fusion module, reducing network parameters and computational complexity while enhancing efficiency and speed.

The Head, the central component of the LW-YOLO model, generates the output results of object detection. It comprises a series of convolutional layers and fully connected layers, amalgamating classifiers and regressors. The classifier identifies the presence of objects in the image and categorizes them accordingly. The regressor predicts the bounding box position and size of the objects. LW-YOLO employs the MPDIoU loss function, facilitating accurate prediction of the position and shape of bounding boxes, thereby boosting the model's performance and precision.

To facilitate readability, we provided annotations for the abbreviations used in this article, as shown in Table 1.

Table 1. Explanation of abbreviations.

Abbreviation	Explanation
YOLO	You Only Look Once
YOLOv8	You Only Look Once version 8
SSD	Single Shot MultiBox Detector
RCNN	Regions with CNN Features
Fast R-CNN	Fast Region-Based CNN
GARPN	Region Proposal by Guided Anchoring
FPS	Frames Per Second
DFP	Diagonal Feature Pyramid
BiFPN	Bidirectional Feature Pyramid Network
FPN	Feature Pyramid Network
PAN	Path Aggregation Network
C2f	CSPDarknet53 to 2-Stage FPN
PConv	Partial Convolution
CIoU	Complete Intersection over Union
MPDIoU	Minimum Point Distance Intersection over Union
Conv	Contrary to Standard Convolution
PWConv	Pointwise Convolution
IoU	Intersection over Union
mAP	mean Average Precision
mAP0.5	mean Average Precision at Intersection over Union (IoU) Thresholds of 0.5
mAP0.5:0.95	mean Average Precision at Intersection over Union (IoU) Thresholds of 0.5:0.95
P	Precision Rate
R	Recall Rate
FLOPs	Floating Point Operations Per Second

2.3. Feature Extraction Network

In deep learning network models, an increase in the number of layers can lead to information distortion or compression at each layer, resulting in feature loss. The integration of feature information from multiple scales can augment the model's detection capability and enhance detection result accuracy. In the YOLOv8 model, the Neck network employs the FPN and PAN network structures. While FPN propagates semantically rich features from deep layers to shallow layers via upsampling, fusing feature information from different scales, PAN propagates position information contained in shallow feature layers to deep feature layers through downsampling, augmenting position and semantic correlations. Despite enhancing the network's feature extraction capability, this combination introduces issues: PAN's input primarily depends on the multiscale features supplied by FPN, potentially leading to the loss or partial replacement of some low-level original features. Consequently, unique details and local information in the backbone network may not be fully transmitted to the PAN network. To rectify these limitations of the original feature extraction network, we introduce an adjusted network structure termed BiFPN. This novel structure combines top-down and bottom-up feature fusion to enhance object

detection task accuracy and performance by addressing shallow feature loss. Figure 4 contrasts the structures of FPN, PAN, and BiFPN, clearly illustrating BiFPN's advantages.

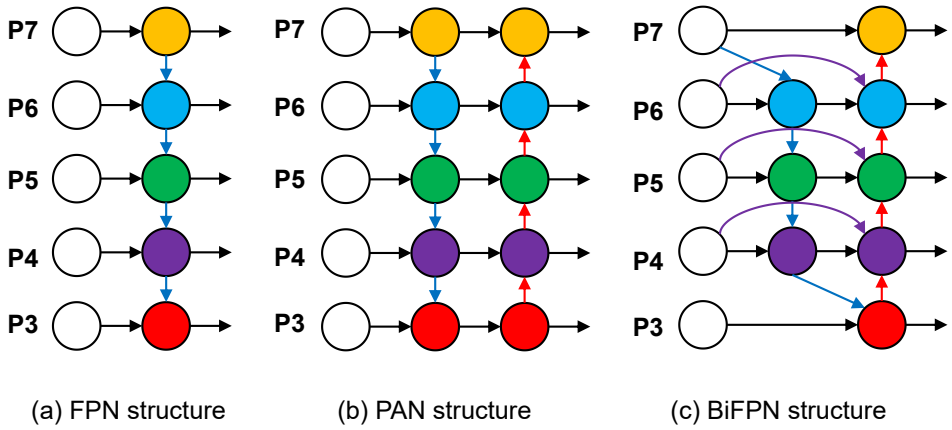

Figure 4. The design of the feature network: (**a**) FPN with a top-down pathway. (**b**) PAN adding a bottom-up pathway on top of FPN. (**c**) BiFPN implementing two optimizations for cross-scale connections.

BiFPN is an improved structure based on the PAN network. With respect to bidirectional cross-scale connections, we implement several key steps to enhance the feature fusion effect. Firstly, we eliminate nodes with only one input edge due to their relatively minor contribution to the results, thereby simplifying the network structure and reducing computational costs. Secondly, we establish a connection between the original input node and the output node, enabling more feature information to be fused at a lower cost, increasing feature diversity, and enhancing overall expressive power. Lastly, we merge top-down and bottom-up paths into a single module, allowing the module to be stacked repeatedly for higher-level feature fusion. These enhancements render BiFPN more potent and efficient in terms of feature fusion capability. To fuse weighted features, BiFPN introduces a rapid normalization fusion method, as demonstrated in Equation (1). In this method, each feature learns a weight to represent its importance and differentiate between various features for discriminative fusion.

In Equations (1)–(3), W_{i} denotes the weight learned through the Rectified Linear Unit (ReLU) activation function, which modulates the contribution levels of various input features. The symbol ε represents a small value (for instance, 0.0001), employed to augment the stability of the denominator in the equation. The weights learned, W_{i} ($W_{\mathrm{i}} \geq 0$, I_{i} representing input features), are confined to non-negative ranges by the ReLU activation function, ensuring weight stability and rationality. This fusion methodology intensifies feature fusion effects and bolsters the model's adaptability to diverse features.

$$O = \sum_{i} \frac{w_{\mathrm{i}}}{\varepsilon + \sum_{j} w_{\mathrm{j}}} \cdot I_{\mathrm{j}} \tag{1}$$

$$P_6^{td} = Conv\left(\frac{W_1 \cdot P_6^{in} + W_2 \cdot Resize(P_7^{in})}{W_1 + W_2 + \varepsilon}\right) \tag{2}$$

$$P_6^{out} = Conv\left(\frac{w_1^{'} \cdot P_6^{in} + w_2^{'} \cdot P_6^{td} + w_3^{'} \cdot Resize(P_5^{out})}{w_1^{'} + w_2^{'} + w_3^{'} + \varepsilon}\right) \tag{3}$$

For performing cross-scale connections and weighted feature fusion, BiFPN extracts P_2, P_3, P_4, and P_5 features from the backbone as BiFPN input. Using node P_6 as an illustration, the fusion process of two features is depicted in Figure 4c. P_6^{td} symbolizes intermediate features from top-down (the blue circle in the middle); P_6^{out} signifies output features from bottom-up (the blue circle on the right); and P_6^{in} and P_7^{in} represent left input features. Resize is typically used for resolution matching through upsampling or downsampling interpola-

tion, while Conv is generally employed for feature processing via convolution operations. By leveraging mutual connections and fusion between different levels, BiFPN successfully achieves bidirectional cross-scale transmission and efficiently integrates information via rapid normalization fusion.

2.4. Bottleneck Lightweighting

As neural networks increase in layer quantity, the number of feature map channels also expands. However, as multiple channels of feature maps may contain similar or identical information, redundancy between feature maps may occur. This redundancy necessitates additional floating-point operations (FLOPs), resulting in increased computational delay.

For an input feature of size $h \times w \times c$, when performing convolutional operations using a $k \times k$ convolutional kernel, the required memory access can be calculated using Equation (4), where c is the number of input data channels.

$$h \times w \times 2c + k^2 \times c \approx h \times w \times 2c \tag{4}$$

DWConv (Depthwise Convolution) [33] computes output channel features using sliding operations on each input channel, and the required memory access can be calculated as Equation (5), where c' is the number of input data channels.

$$h \times w \times 2c' + k^2 \times c' \approx h \times w \times 2c' \tag{5}$$

DWConv is often used with PWConv (Pointwise Convolution) [34] to improve model accuracy, which means $c' > c$, where the memory access of PWConv is typically larger than that of Conv(Contrary to standard Convolution), leading to increased latency. To improve detection speed and reduce memory access, a new convolution module needs to be introduced to replace the conventional convolution and solve the inefficiency problem of DWConv.

Unlike Conv and DWConv, PConv in FasterNet applies regular convolution to a subset of input channels to extract spatial features, rather than convolving the entirety of the input channel. If the feature maps are stored in a continuous or regular pattern in memory, the first or last continuous channel can adequately represent the entire feature map. Experimental findings [28] demonstrate that PConv, when compared with standard Conv, markedly diminishes computational complexity, necessitating only 1/16 of the computational operations. Similarly, PConv also curtails memory access to merely 1/4 of that required by traditional convolution.

Drawing parallels to DWConv, FasterNet introduces PWConv based on PConv to capture correlations between input channels. As depicted in Figure 5, by integrating PWConv with PConv, two distinct structures are formed: a T-shaped Conv structure and two independent convolution structures.

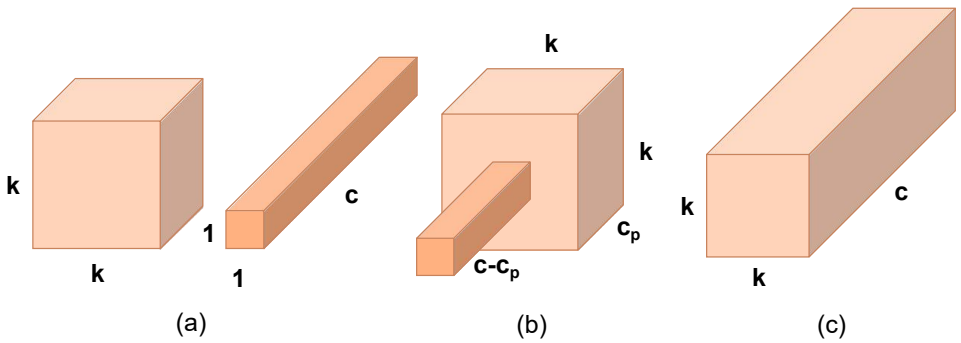

Figure 5. The structures of different convolution modules. (**a**) PConv+PWConv. (**b**) T-shaped Conv. (**c**) Regular Conv.

The T-shaped convolution structure incorporates a Pointwise Convolution (PWConv) layer on the foundation of Partial Convolution (PConv), performing convolutional oper-

ations on the spatial dimension to further extract features. Compared with conventional Conv, the T-shaped Conv places greater emphasis on features at the central position. Although the T-shaped convolution module can directly adopt higher computational efficiency, it demands more FLOPs and is computationally more complex relative to PConv and PWConv. For identical input and output feature maps, the FLOPs of T-shaped Conv and two independent convolutions are demonstrated in Equation (6) and Equation (7), respectively. Here, $c > c_p$ and $c - c_p > c_p$, c_p are the first or last channel numbers of the channels stored contiguously in memory.

$$FLOPs_{\text{T-shaped}} = h \times w \times (k^2 \times c_p \times c + c \times (c - c_p)) \tag{6}$$

$$FLOPs_{(\text{PConv+PWConv})} = h \times w \times (k^2 \times c_p^2 + c \times c_p) \tag{7}$$

The FasterNet module comprises three convolutional layers, as depicted in Figure 6, one of which is the PConv layer and the remaining two are PWConv layers. The Bottleneck module of C2f is supplanted by the FasterNet module, as illustrated in Figure 7. By implementing PConv, we can reduce model computation while preserving accuracy. Relative to the original Bottleneck structure, each FasterNet module incurs a computational cost of approximately 1/16. This is attributable to the fact that in the FasterNet module, we only convolve 1/4 of the original channel number, and the computational cost of subsequent 1×1 convolution is relatively marginal. Consequently, the computational cost of the entire FasterNet module can be roughly 1/16 of the Bottleneck structure. By optimizing the Bottleneck structure in YOLOv8, our enhancement scheme yielded significant results in reducing computational load and improving inference speed, a fact that is thoroughly corroborated in the experimental section in Section 4.

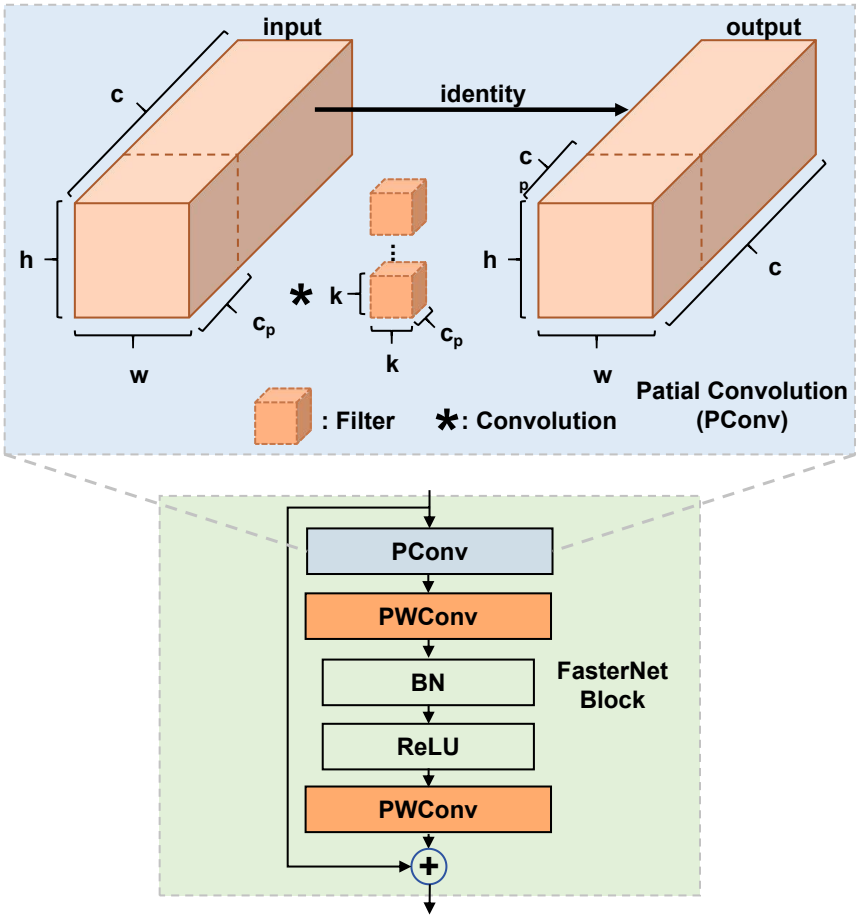

Figure 6. The structure of the FasterNet module.

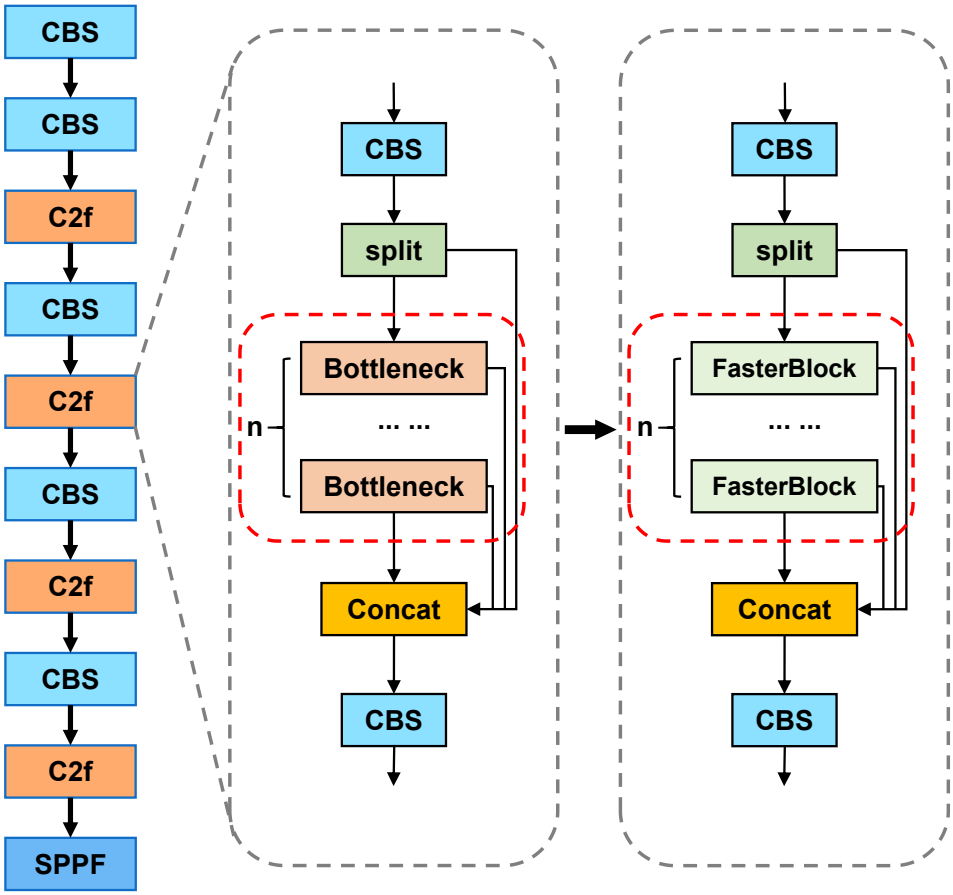

Figure 7. The improvement in the backbone network by introducing the FasterNet module, replacing the Bottleneck in C2f.

2.5. Loss Function

Given that defects on Printed Circuit Boards are typically minuscule, a judiciously designed loss function can significantly enhance the model's target detection performance. In target detection, the bounding box regression loss function is pivotal and directly influences the model's performance. While the Complete Intersection over Union (CIoU) [35] loss function factors in a penalty term for aspect ratio during the computation of the bounding box regression loss, it exhibits a limitation: when the actual bounding box and predicted bounding box share the same aspect ratio but differ in terms of width and height values, the CIoU loss function fails to accurately represent the true disparities between these two bounding boxes. This leads to a reduction in the convergence speed and accuracy of bounding box regression. The formula for CIoU can be articulated as Equation (8):

$$L_{\mathrm{CIoU}} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c_{\mathrm{w}}^2 + c_{\mathrm{h}}^2} + \frac{4}{\pi^2}(\tan^{-1}\frac{w^{gt}}{h^{gt}} - \tan^{-1}\frac{w}{h}) \tag{8}$$

As Equation (8) shows, the Intersection over Union (IoU) denotes the ratio of the intersecting area of the predicted bounding box and the actual bounding box to their combined area. The parameters implicated in the formula are illustrated in Figure 8. The term $\rho(b, b^{gt})$ signifies the Euclidean distance between the center of the actual box and the predicted box; h and w, respectively, correspond to the height and width of the predicted box; h^{gt} and w^{gt}, respectively, represent the height and width of the actual box; and c_{h} and c_{w}, respectively, denote the height and width of the smallest enclosing box formed by the predicted box and actual box.

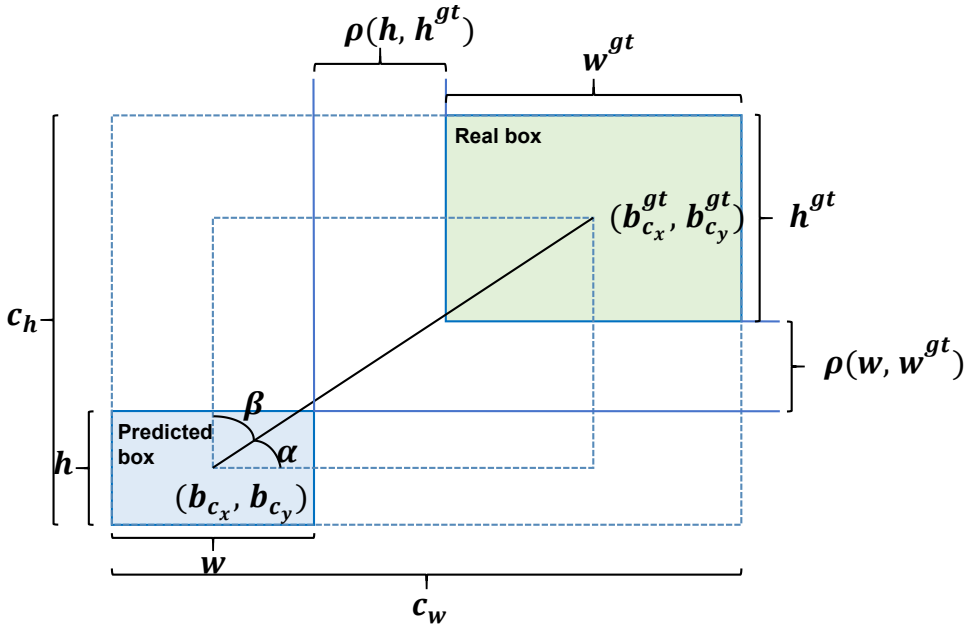

Figure 8. The CIoU loss function.

MPDIoU is a bounding box similarity measurement technique predicated on minimum point distance. It holistically considers various related factors in existing loss functions. Compared with other loss functions, MPDIoU simplifies the similarity comparison between bounding boxes, is suitable for bounding box regression considering both overlapping and nonoverlapping scenarios, and attains superior efficiency and accuracy in bounding box regression tasks. MPDIoU employs a novel IoU-based measurement approach, which directly minimizes the distance between the upper-left corner and lower-right corner points of the predicted bounding box and actual bounding box to assess their similarity. This method circumvents traditional area measurement and concentrates on specific positional information of bounding boxes, thereby measuring their similarity with greater precision. The computation method of MPDIoU is as follows:

As Equations (9)–(12) show, A and B are two arbitrary convex polygons, and the parameters involved in the formula are shown in Figure 9. w and h are, respectively, the width and height of the input image. $(x_1^A, y_1^A), (x_2^A, y_2^A), (x_1^B, y_1^B), (x_2^B, y_2^B)$, respectively, represent the coordinates of the upper-left corner and lower-right corner points of A and B.

$$d_1^2 = (x_1^B - x_1^A)^2 + (y_1^B - y_1^A)^2 \tag{9}$$

$$d_2^2 = (x_2^B - x_2^A)^2 + (y_2^B - y_2^A)^2 \tag{10}$$

$$MPDIoU = \frac{I}{A^{gt} + A^{prd} + I} - \frac{d_1^2}{w^2 + h^2} - \frac{d_2^2}{w^2 + h^2} \tag{11}$$

$$L_{\text{MPDIoU}} = 1 - MPDIoU \tag{12}$$

The MPDIoU loss function is used in target detection tasks to help the model accurately predict the position and shape of bounding boxes by measuring their correlation and overlap. Compared with the original CIoU loss function, the MPDIoU loss function provides a more accurate way to measure bounding boxes, which can effectively improve the performance and accuracy of detection models during training and optimization.

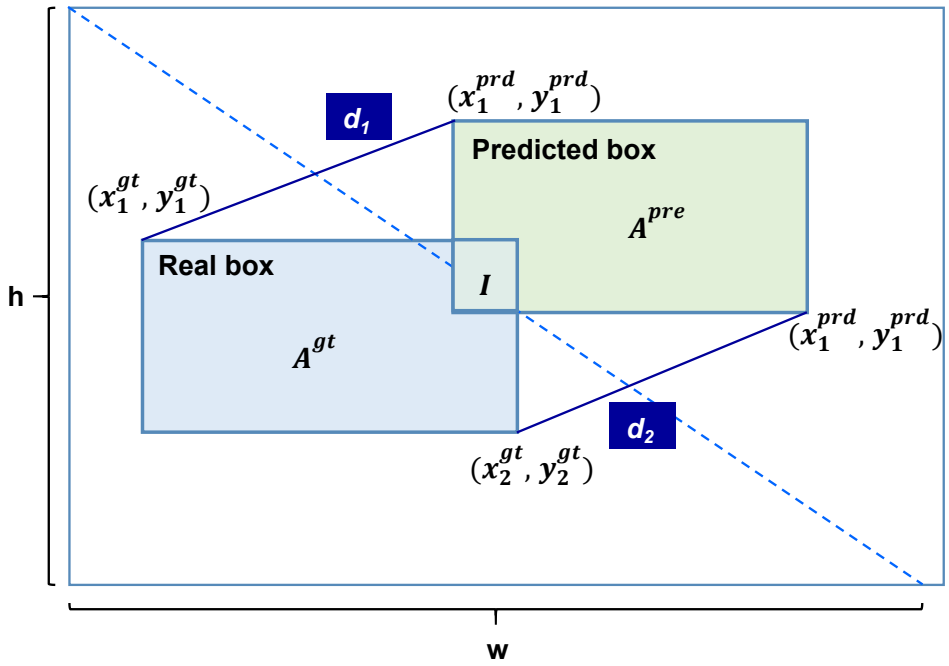

Figure 9. The LW-YOLO loss function.

3. Image Preprocessing and Dataset

The dataset employed for experimentation in this study is derived from the PCB defect dataset released by the Open Laboratory of Peking University. The dataset comprises 1386 PCB images, each averaging a pixel size of 2777 × 2138. It encompasses six distinct types of defects: missing hole, open circuit, short, spur, spurious copper, and mouse bite. Figure 10 presents sample images of these defects. Given the relatively limited number of training samples in the original dataset, the model is susceptible to overfitting on these sparse samples, leading to issues such as poor generalization capability, imprecise parameter estimation, overly intricate network, etc. These challenges can be effectively mitigated by suitably enhancing the original images and augmenting the number of images [36]. Concurrently, various transformations can be applied to the images to bolster the model's generalization capability, robustness, and curb overfitting, thereby enhancing the model's performance and efficacy (Figure 11). In this study, following random flipping, rotation, translation, scaling, and cropping operations, the quantity of images in the dataset was expanded to 2272. To train and evaluate the model, the dataset was partitioned into a training set, validation set, and test set in a ratio of 7:2:1. Table 2 exhibits the distribution of each type of defect image data in the dataset. Post data augmentation, the mean Average Precision (mAP) escalated from 91.81% to 94.20% in Table 3.

Table 2. The types and quantities of PCB datasets.

Defects	Number
missing hole	460
open circuit	464
short	464
spur	460
spurious coper	464
mouse bite	460
total	2272

Table 3. The original dataset with the augmented dataset.

Category	Original Dataset	Augmented Dataset
Number of images	1386	2272
Number of defects	5906	11,812
mAP:0.5(%)	91.81	94.20

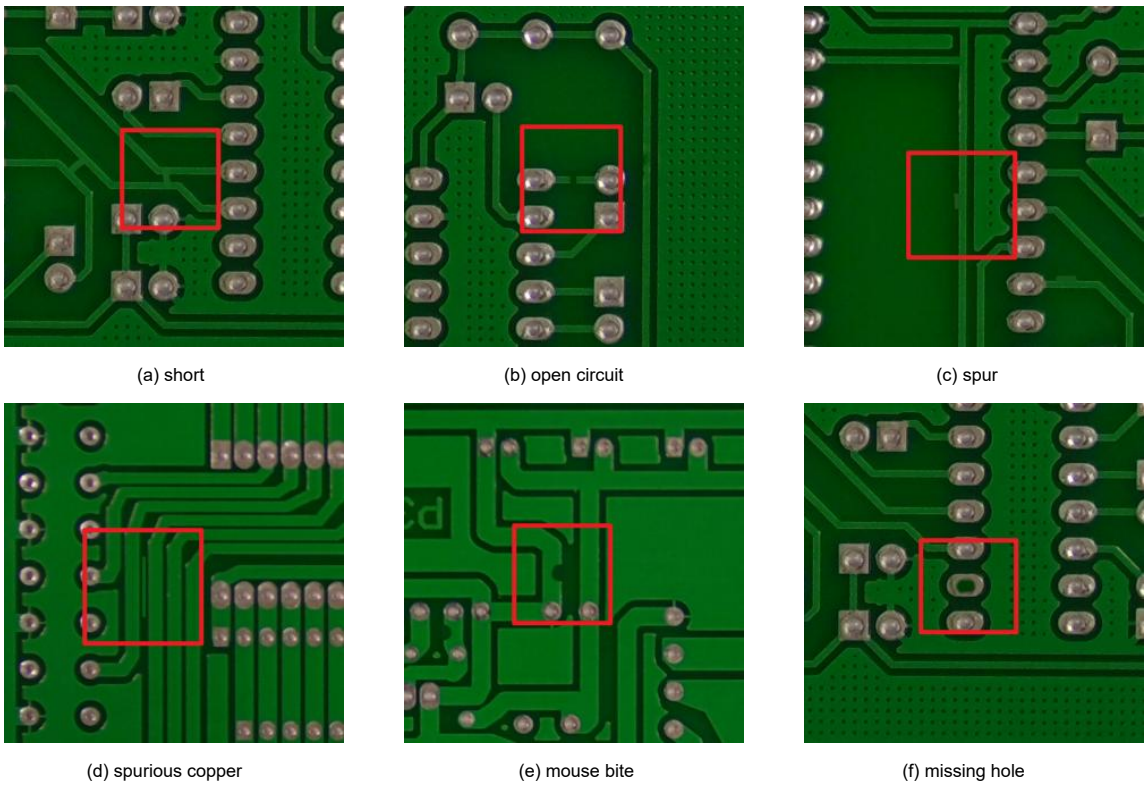

Figure 10. Six types of defects on printed circuit boards: (**a**) short, (**b**) open circuit, (**c**) spur, (**d**) spurious copper, (**e**) mouse bite and (**f**) missing hole.

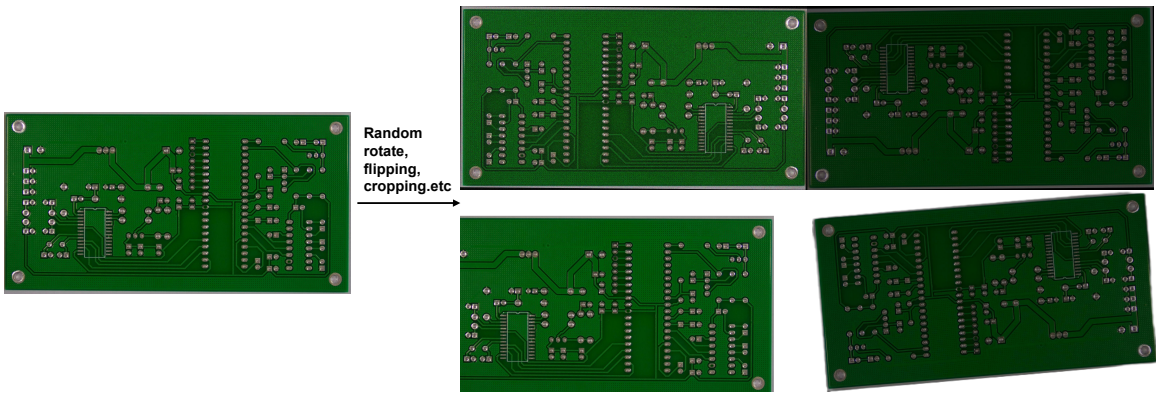

Figure 11. Images obtained using augmentation techniques.

4. Experimental Analysis and Discussion

4.1. Evaluation Metrics

To quantitatively assess the performance of the proposed model, we employed multiple evaluation metrics such as P, R, AP, mAP0.5 and mAP0.5:0.95, Floating Point Operations Per Second (FLOPs), model size, and FPS, as Equation (13)–(17). Precision represents the accuracy of the model's predictions on a specific dataset, while recall evaluates the comprehensiveness of the model's detections. AP denotes the model's precision within a given

category. TP and False FP represent the number of accurately and inaccurately identified samples, respectively, while False Negatives FN signifies the number of samples incorrectly identified as negative or overlooked by the model. N is the number of PCB defect categories. FPS measures the real-time processing speed of the algorithm in terms of the number of images processed per unit time, as depicted in Equation (16), where F_n is the number of images detected by the model, and T is the time taken to detect the images. IoU is a metric employed in deep learning to measure the overlap between predicted results and the ground truth targets.

$$P = \frac{TP}{TP + FP} \tag{13}$$

$$R = \frac{TP}{TP + FN} \tag{14}$$

$$AP = \int_0^1 P_i(r_i)d((r_i)), mAP = \frac{AP}{N} \tag{15}$$

$$FPS = \frac{F_n}{T} \tag{16}$$

$$IoU(box_{gt,box_p}) = \frac{|box_{gt} \cap box_p|}{box_{gt} \cup box_p} = \frac{TP}{FN + FP + TP} \tag{17}$$

4.2. Model Training

The experimental setup comprises an Ubuntu 20.04 operating system and PyTorch deep learning framework. Please refer to Table 4 for more detailed information regarding the experimental environment. Parameters used during training can be found in Table 5.

Table 4. Experimental Environment.

Category	Configuration
CPU	Intel(R) Xeon(R) Platinum 8255C
GPU	NVIDIA GeForce RTX 2080 Ti 11G
System environment	Ubuntu20.04
Framework	PyTorch 1.12.0
Programming environment	Python 3.8

Table 5. Training Parameters.

Parameter	Value
Image Size	640 × 640
Learning Rate	0.01
Weight Decay	0.0005
Momentum	0.937
Optimizer	SGD
Batch Size	32
Dataloader	8
Epoch	550

Figure 12 illustrates the loss values for each iteration during the model's training process. The training loss encompasses box loss, classification loss, and dynamic feature learning loss, denoted by train/box_loss, train/cls_loss, and train/dfl_loss, respectively. It can be observed that the loss values for each class initially fluctuate significantly but gradually stabilize and decrease as training progresses.

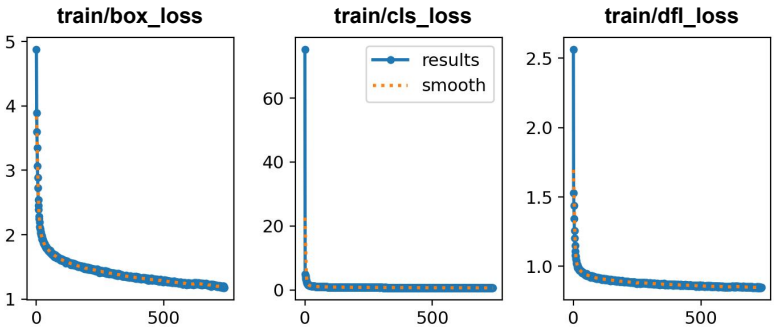

Figure 12. Loss curve during training.

4.3. Defect Detection Test Results

LW-YOLO undergoes numerous iterations of training to update and optimize parameters on the training set, culminating in a set of optimal weights for detection on the test set. As depicted in Table 6, the experimental outcomes suggest that Missing Hole, Open Circuit, Short, and Spurious Copper exhibit higher precision, recall, and Average Precision (AP) values. This can be attributed to these defect categories possessing distinct features, fixed shapes, and being less susceptible to other random shapes or external factors such as background interference. Conversely, the spur and mouse bite categories demonstrate lower recall and AP compared with other categories due to their random shapes and similarity, rendering them more prone to false detections when there are ample defect instances. By employing image processing techniques such as cropping and brightness adjustment to modify the background information, we can accentuate defect features for improved detection. As shown in Figure 13, the results indicate that both spur and mouse bite types of defects possess AP values surpassing 0.9. Figure 14 illustrates the detection results for various defects, where all defects are successfully detected with confidence scores exceeding 0.8.

Table 6. Test Results for Detection of Six Types of Defects.

Defects	Missing Hole	Open Circuit	Short	Spur	Spurious Copper	Mouse Bite
Precision	0.983	0.977	0.988	0.986	0.981	0.971
Recall	0.997	0.92	0.994	0.905	0.987	0.92
AP	0.995	0.974	0.993	0.971	0.985	0.92
mAP			0.964			

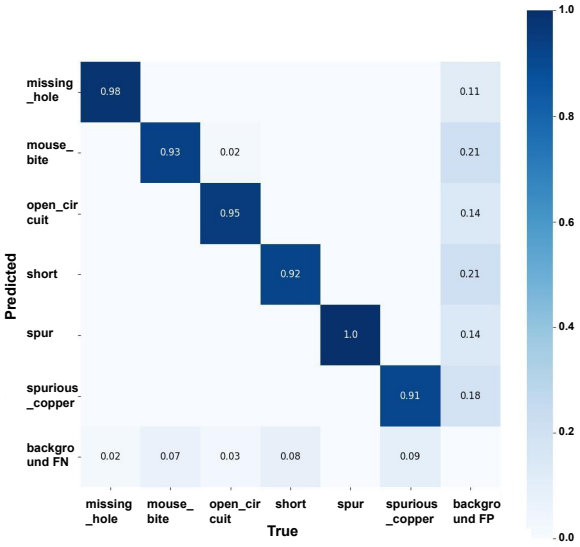

Figure 13. Confusion matrix of LW-YOLO.

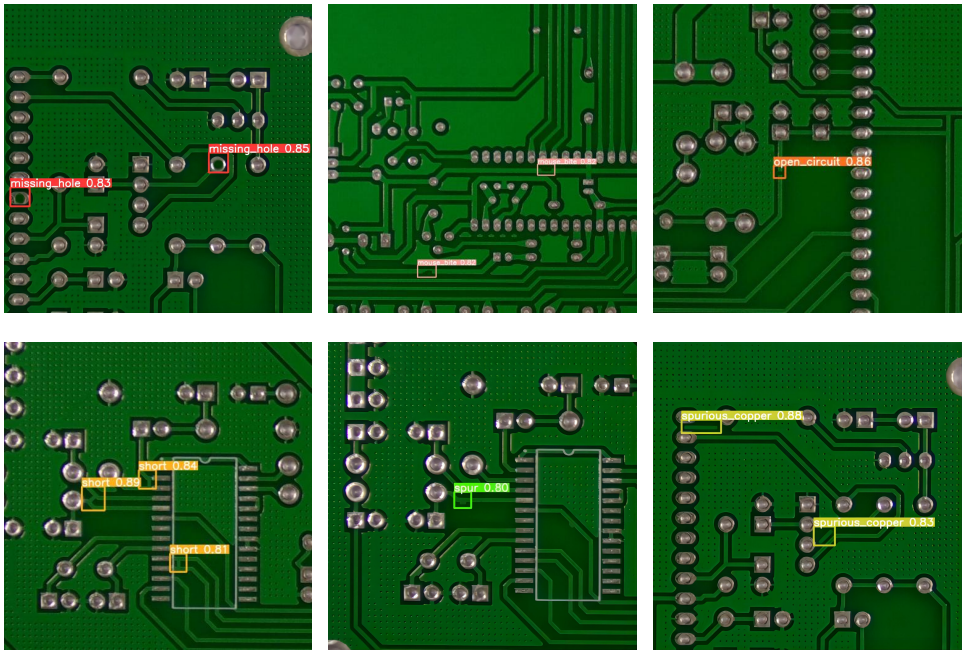

Figure 14. Visualization of the test set.

Figure 15 presents the variation in key evaluation metrics during the training process of our proposed model compared with YOLOv8. These metrics help us understand the model's performance and training progress. The results indicate that our proposed model outperforms YOLOv8 in the first three detection metrics (mAP@50, mAP@50:95, precision) with significant improvements in the first two metrics. Additionally, our model starts to stabilize after approximately 150 epochs of training. Compared with YOLOv8, our model demonstrates better performance in terms of training speed and detection accuracy.

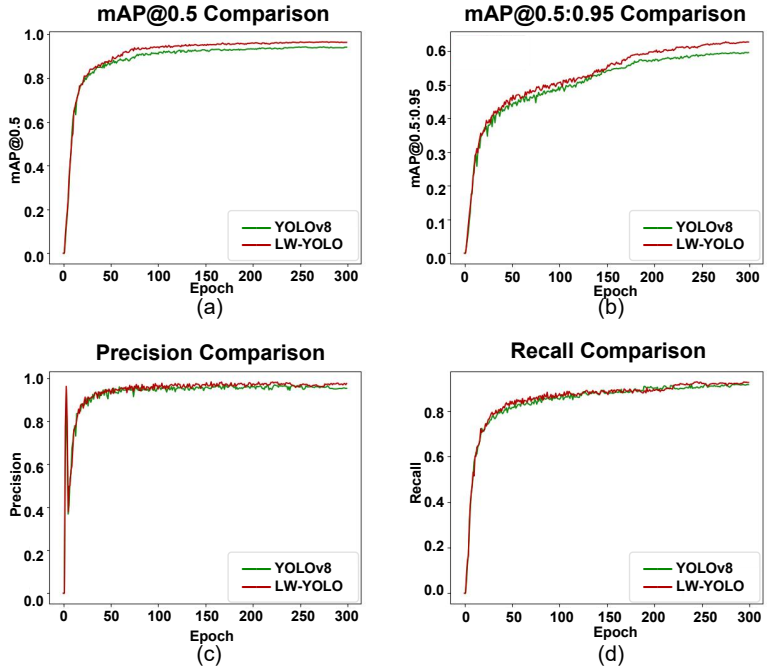

Figure 15. (**a**) Training Curve of mAP0.5 for LW-YOLO and YOLOv8. (**b**) Training Curve of mAP0.5:0.95 for LW-YOLO and YOLOv8. (**c**) Training Curve of Precision for LW-YOLO and YOLOv8. (**d**) Training Curve of Recall for LW-YOLO and YOLOv8.

4.4. Ablation Experiment

Table 7 illustrates that the integration of the Bidirectional Feature Pyramid Network (BiFPN) module enhances precision by 0.5%, while concurrently reducing FLOPs, thereby improving computational efficiency, conserving energy, and facilitating effective deployment of the model. The BiFPN module adaptively adjusts feature weights based on their significance using a dynamic weighting scheme, mitigating feature imbalance and boosting object detection accuracy. Additionally, by sharing weights and reusing features, the BiFPN module effectively curtails computational complexity and augments the computational efficiency of the model. The incorporation of the FasterNet module significantly diminishes FLOPs. The introduction of the Partial Convolution (PConv) module mitigates computational redundancy and memory access while enhancing algorithm speed. The addition of MPDIoU escalates mAP% by 0.5% and precision by 0.9%. MPDIoU is an evaluation metric that holistically considers target overlap and directional relationships, simplifying similarity comparisons between two bounding boxes. It is suitable for both overlapping and nonoverlapping bounding box regression and aids in improving the efficacy and performance of object detection algorithms.

Table 7. Results of Model Ablation Experiments (Bold and Underlined Data in the Table Represent the Best Results).

BiFPN	FasterNet	MPDIoU	P (%)	R (%)	mAP0.5 (%)	mAP0.5:0.95 (%)	FLOPs (G)	FPS
			95.4	93.2	94.2	60.1	8.9	138.2
✓			95.9	92.3	95.6	61.8	7.1	137.1
✓	✓		95.8	92.7	95.9	63.0	**6.2**	**144.8**
✓	✓	✓	**97.1**	**93.5**	**96.4**	**63.4**	6.8	141.5

4.5. Performance Comparison of Different Detection Algorithms

In order to evaluate the performance of LW-YOLO objectively, comparative experiments were conducted against other classical single-stage and two-stage detection algorithms. The single-stage detection algorithms incorporated EfficientDet [25], Single Shot MultiBox Detector (SSD), YOLOv4 with CSPDarknet53 serving as its backbone network, YOLOv5 supplemented with additional data augmentation strategies and Focus structure, and YOLOv7 with Efficient Local Attention Network (ELAN) architecture. The representative two-stage detection algorithm employed was Faster R-CNN, which utilizes region proposal networks and region classification networks for object detection. The results of the comparative experiments are presented in Table 8.

Table 8. Comparison of Experimental Results with Different Algorithms (Bold and Underlined Data in the Table Represent the Best Results).

Models	P (%)	R (%)	mAP0.5 (%)	FPS	Model Size (MB)
Faster-R-CNN [37]	**97.2**	92.9	96.3	82.1	478.49
EfficientDet [25]	81.2	89.4	87.4	140.3	68.5
SSD [30]	82.5	81.3	81.9	129.2	91.6
YOLOv4 [38]	68.5	69.4	68.1	115.2	244.07
YOLOv5 [39]	91.2	90.3	91.1	129.3	**5.3**
YOLOv7 [31]	92.9	91.3	93.8	131.4	12.4
YOLOv8	95.4	93.2	94.2	138.2	6.3
LW-YOLO(Ours)	97.1	**93.5**	**96.4**	**141.5**	6.5

Based on the experimental outcomes, while EfficientDet exhibits superior detection speed, its detection accuracy remains suboptimal, rendering it unsuitable for detecting surface defects on PCBs. Early YOLO series algorithms (such as YOLOv4) possess intricate network structures and a large quantity of parameters, yet their detection accuracy is relatively low. Conversely, YOLOv5n, YOLOv7, and YOLOv8 adopt smaller model sizes and fewer parameters, leading to enhanced detection performance. LW-YOLO surpasses algorithms such as YOLOv4, YOLOv5, and YOLOv7 in terms of mAP and detection speed. When its detection speed

is comparable to that of the YOLOv8 algorithm, LW-YOLO exhibits a higher mAP value, demonstrating superior detection performance. Compared with the Faster R-CNN model, LW-YOLO achieves a similar mAP but a significantly faster detection speed. Furthermore, the improved algorithm model has a compact size of only 6.5 MB, which is substantially smaller than the Faster R-CNN model and marginally larger than the YOLOv8 model, but it outperforms in terms of detection accuracy and frame rate speed. These results suggest that our proposed algorithm not only satisfies the prerequisites for real-time detection but also enhances detection accuracy and provides greater versatility by minimizing model size, indicating significant potential and practical value for PCB defect detection.

4.6. Loss Function Comparison

To substantiate the superiority of our proposed loss function, we executed a comparative experiment on the loss function, assessing the impact of employing MPDIoU and several mainstream loss functions on the LW-YOLO model, while maintaining consistency in other training and classification conditions. The test outcomes, as displayed in Table 9, suggest that utilizing MPDIoU as the bounding box regression loss yields the most optimal detection performance. Moreover, the use of MPDIoU results in a 0.5% mAP enhancement compared with the usage of CIoU, further underscoring the efficacy of the MPDIoU loss function in bolstering detection performance.

Table 9. Comparison of Detection Results with Different Loss Functions Introduced by LW-YOLO (Bold and Underlined Data in the Table Represent the Best Results).

Metrics	P (%)	R (%)	mAP0.5 (%)	mAP0.5:0.95 (%)
CIoU	95.4	92.7	95.9	60.1
DIoU [35]	95.9	92.6	96.0	63.1
GIoU [40]	94.8	92.8	96.2	62.8
EIoU [41]	94.1	92.5	95.1	62.8
SIoU [42]	**97.2**	93.4	96.1	62.8
WIoU v1	94.8	92.9	95.8	62.8
WIoU v2	96.1	92.8	95.9	60.1
WIoU v3 [33]	96.9	93.2	96.2	63.2
MPDIoU	97.1	**93.5**	**96.4**	**63.4**

4.7. Performance Comparison on the DeepPCB Dataset

DeepPCB is a PCB defect dataset on GitHub consisting of 1500 PCB images, each containing six types of defects. The original images are 16 k × 16 k pixels, cropped into 640 × 640 subimages and aligned through template matching. The dataset is divided into training and testing sets, with the trained model tested on the testing set, as shown in Table 10.

Table 10. Comparison of Experimental Results on the DeepPCB Dataset (Bold and Underlined Data in the Table Represent the Best Results).

Models	P (%)	R (%)	mAP0.5 (%)	FPS	Model Size (MB)
Faster-R-CNN	91.3	92.8	92.2	22	478.49
SSD	85.4	84.3	86.2	125.3	91.6
YOLOv4	92.1	89.5	92.5	62.8	244.07
YOLOv5	95.3	**96.1**	93.2	90.8	**5.3**
YOLOv7	89.2	91.2	94.3	112.3	12.4
YOLOv8	96.3	94.8	94.5	129.1	6.3
LW-YOLO (Ours)	**96.7**	95.7	**95.2**	**138.1**	6.5

Table 10 shows that LW-YOLO outperforms other models in terms of accuracy, mAP0.5 value, and FPS, achieving 96.7%, 95.2%, and 138.1, respectively. Although the model size of LW-YOLO is slightly larger than YOLOv8, its accuracy and FPS are superior to YOLOv8. Compared with YOLOv5, LW-YOLO has a slightly lower recall rate, but it is still within satisfactory range. Furthermore, compared with the original YOLOv8, LW-YOLO performs

better in terms of accuracy, mAP0.5 value, and recall. Therefore, it can be considered an excellent model for PCB defect detection.

5. Conclusions

Compared with prevalent object detection network models, this paper introduces a lightweight model, LW-YOLO, specifically designed for detecting PCB defects. Initially, this paper addresses the issue of feature information loss in the backbone network by fully leveraging feature information at various levels, thereby enhancing the model's detection capability for objects of different scales. Concurrently, lightweight feature fusion modules are employed in the BiFPN to diminish the network parameter count and computational complexity, thus enhancing network computational efficiency and speed. Subsequently, the lightweight FasterNet module structure is utilized to rectify the redundancy issue of channel information in feature maps and augment the model's inference speed. Finally, a novel MPDIoU loss function is employed to guide the model in accurately predicting the position and shape of bounding boxes, effectively boosting model performance and accuracy.

The experimental results unequivocally demonstrate that the LW-YOLO object detection model developed in this study exhibits significant advantages on the PCB dataset. This model not only outperforms in terms of detection accuracy and speed but also fulfills the requirements for lightweight deployment. This implies that our model showcases superior performance in practical industrial applications. It can effectively detect defects or anomalies on PCB, meet the high-precision object detection requirements in the industrial field, and provide reliable solutions for other domains such as automated production processes, quality control, and fault detection.

Author Contributions: Writing—original draft, X.T. Writing—review and editing, H.N. and Z.Y. (Zhengzhe Yang). Supervision, Z.Y. (Zhaohui Yuan). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Province Science Foundation of Jiangxi, with grant numbers 20224BAB202030 and 20202ACBL202009.

Data Availability Statement: The data were published in the Open Lab on Human–Robot interaction of Peking University at https://robotics.pkusz.edu.cn/resources/dataset/, accessed on 1 March 2024.

Acknowledgments: The authors would like to express their sincere thanks to the referees for their careful reading and suggestions which helped us to improve the paper.

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

References

1. Liu, Z.; Qu, B. Machine vision based online detection of PCB defect. *Microprocess. Microsyst.* **2021**, *82*, 103807. [CrossRef]
2. Ding, R.; Dai, L.; Li, G.; Liu, H. TDD-net: A tiny defect detection network for printed circuit boards. *CAAI Trans. Intell. Technol.* **2019**, *4*, 110–116. [CrossRef]
3. Li, Y.T.; Kuo, P.; Guo, J.I. Automatic industry PCB board DIP process defect detection with deep ensemble method. In Proceedings of the 2020 IEEE 29th International Symposium on Industrial Electronics (ISIE), Delft, The Netherlands, 17–19 June 2020; pp. 453–459.
4. Moganti, M.; Ercal, F.; Dagli, C.H.; Tsunekawa, S. Automatic PCB inspection algorithms: A survey. *Comput. Vis. Image Underst.* **1996**, *63*, 287–313. [CrossRef]
5. Thomas, S.S.; Gupta, S.; Subramanian, V.K. Smart surveillance based on video summarization. In Proceedings of the 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, India, 14–16 July 2017; pp. 1–5.
6. Büchi, G.; Cugno, M.; Castagnoli, R. Smart factory performance and Industry 4.0. *Technol. Forecast. Soc. Chang.* **2020**, *150*, 119790. [CrossRef]
7. Zhang, P.; Fan, W.; Chen, Y.; Feng, J.; Sareh, P. Structural symmetry recognition in planar structures using convolutional neural networks. *Eng. Struct.* **2022**, *260*, 114227. [CrossRef]
8. Adibhatla, V.A.; Chih, H.C.; Hsu, C.C.; Cheng, J.; Abbod, M.F.; Shieh, J.S. Defect detection in printed circuit boards using you-only-look-once convolutional neural networks. *Electronics* **2020**, *9*, 1547. [CrossRef]
9. Lin, G.; Tang, Y.; Zou, X.; Cheng, J.; Xiong, J. Fruit detection in natural environment using partial shape matching and probabilistic Hough transform. *Precis. Agric.* **2020**, *21*, 160–177. [CrossRef]

10. Fu, G.; Sun, P.; Zhu, W.; Yang, J.; Cao, Y.; Yang, M.Y.; Cao, Y. A deep-learning-based approach for fast and robust steel surface defects classification. *Opt. Lasers Eng.* **2019**, *121*, 397–405. [CrossRef]
11. Zhao, Z.Q.; Zheng, P.; Xu, S.t.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
12. Tsai, D.M.; Chou, Y.H. Fast and precise positioning in PCBs using deep neural network regression. *IEEE Trans. Instrum. Meas.* **2019**, *69*, 4692–4701. [CrossRef]
13. Yoon, H.; Lee, J. Pcb component classification algorithm based on yolo network for pcb inspection. *J. Korea Multimed. Soc.* **2021**, *24*, 988–999.
14. Silverstone, A.E.; Rosenbaum, P.F.; Weinstock, R.S.; Bartell, S.M.; Foushee, H.R.; Shelton, C.; Pavuk, M.; Consortium, A.E.H.R. Polychlorinated biphenyl (PCB) exposure and diabetes: Results from the Anniston Community Health Survey. *Environ. Health Perspect.* **2012**, *120*, 727–732. [CrossRef] [PubMed]
15. Wei, P.; Liu, C.; Liu, M.; Gao, Y.; Liu, H. CNN-based reference comparison method for classifying bare PCB defects. *J. Eng.* **2018**, 1528–1533. [CrossRef]
16. Ustabas Kaya, G. Development of hybrid optical sensor based on deep learning to detect and classify the micro-size defects in printed circuit board. *Measurement* **2023**, *206*, 112247. [CrossRef]
17. Kim, J.; Ko, J.; Choi, H.; Kim, H. Printed Circuit Board Defect Detection Using Deep Learning via A Skip-Connected Convolutional Autoencoder. *Sensors* **2021**, *21*, 4968. [CrossRef]
18. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
19. Zeng, N.; Wu, P.; Wang, Z.; Li, H.; Liu, W.; Liu, X. A small-sized object detection oriented multi-scale feature fusion approach with application to defect detection. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–14. [CrossRef]
20. Liao, X.; Lv, S.; Li, D.; Luo, Y.; Zhu, Z.; Jiang, C. YOLOv4-MN3 for PCB surface defect detection. *Appl. Sci.* **2021**, *11*, 11701. [CrossRef]
21. Cheng, M.; Bai, J.; Li, L.; Chen, Q.; Zhou, X.; Zhang, H.; Zhang, P. Tiny-RetinaNet: A one-stage detector for real-time object detection. In Proceedings of the Eleventh International Conference on Graphics and Image Processing (ICGIP 2019), Hangzhou, China, 12–14 October 2019; SPIE: Bellingham, WA, USA, 2020; Volume 11373, pp. 195–202.
22. Yu, Z.; Wu, Y.; Wei, B.; Ding, Z.; Luo, F. A lightweight and efficient model for surface tiny defect detection. *Appl. Intell.* **2023**, *53*, 6344–6353. [CrossRef]
23. Hu, B.; Wang, J. Detection of PCB surface defects with improved faster-RCNN and feature pyramid network. *IEEE Access* **2020**, *8*, 108335–108345. [CrossRef]
24. Mehta, D.; Lu, H.; Paradis, O.P.; MS, M.A.; Rahman, M.T.; Iskander, Y.; Chawla, P.; Woodard, D.L.; Tehranipoor, M.; Asadizanjani, N. The big hack explained: Detection and prevention of PCB supply chain implants. *Acm J. Emerg. Technol. Comput. Syst.* **2020**, *16*, 1–25. [CrossRef]
25. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
26. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2016; pp. 2117–2125.
27. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18-23 June 2018 ; pp. 8759–8768.
28. Chen, J.; Kao, S.h.; He, H.; Zhuo, W.; Wen, S.; Lee, C.H.; Chan, S.H.G. Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 12021–12031.
29. Siliang, M.; Yong, X. MPDIoU: A Loss for Efficient and Accurate Bounding Box Regression. *arXiv* **2023**, arXiv:2307.07662.
30. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475.
32. Wang, C.Y.; Liao, H.Y.M.; Wu, Y.H.; Chen, P.Y.; Hsieh, J.W.; Yeh, I.H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 390–391.
33. Tong, Z.; Chen, Y.; Xu, Z.; Yu, R. Wise-IoU: Bounding Box Regression Loss with Dynamic Focusing Mechanism. *arXiv* **2023**, arXiv:2301.10051.
34. Hua, B.S.; Tran, M.K.; Yeung, S.K. Pointwise convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 984–993.
35. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.

36. Mushtaq, Z.; Su, S.F. Environmental sound classification using a regularized deep convolutional neural network with data augmentation. *Appl. Acoust.* **2020**, *167*, 107389. [CrossRef]
37. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1440-1448. [CrossRef] [PubMed]
38. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
39. Zhu, X.; Lyu, S.; Wang, X.; Zhao, Q. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2778–2788.
40. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 658–666.
41. Zhang, Y.F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and efficient IOU loss for accurate bounding box regression. *Neurocomputing* **2022**, *506*, 146–157. [CrossRef]
42. Gevorgyan, Z. SIoU loss: More powerful learning for bounding box regression. *arXiv* **2022**, arXiv:2205.12740.