

Article

# Lightweight Tennis Ball Detection Algorithm Based on Robomaster EP

Yuan Zhao , Ling Lu, Wu Yang \*, Qizheng Li and Xiujie Zhang

School of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China  
\* Correspondence: yw@cqut.edu.cn

**Abstract:** To address the problems of poor recognition effect, low detection accuracy, many model parameters and computation, complex network structure, and unfavorable portability to embedded devices in traditional tennis ball detection algorithms, this study proposes a lightweight tennis ball detection algorithm, YOLOv5s-Z, based on the YOLOv5s algorithm and Robomaster EP. The main work is as follows: firstly, the lightweight network G-Backbone and G-Neck network layers are constructed to reduce the number of parameters and computation of the network structure. Secondly, convolutional coordinate attention is incorporated into the G-Backbone to embed location information into channel attention, which enables the network to obtain location information of a larger area through multiple convolutions and enhances the expression ability of mobile network learning features. In addition, the Concat module in the original feature fusion is modified into a weighted bi-directional feature pyramid W-BiFPN with settable learning weights to improve the feature fusion capability and achieve efficient weighted feature fusion and bi-directional cross-scale connectivity. Finally, the Loss function EIOU Loss is introduced to split the influence factor of the aspect ratio and calculate the length and width of the target frame and anchor frame, respectively, combined with Focal-EIOU Loss to solve the problem of imbalance between complex and easy samples. Meta-ACON's activation function is introduced to achieve an adaptive selection of whether to activate neurons and improve the detection accuracy. The experimental results show that compared with the YOLOv5s algorithm, the YOLOv5s-Z algorithm reduces the number of parameters and computation by 42% and 44%, respectively, reduces the model size by 39%, and improves the mean accuracy by 2%, verifying the effectiveness of the improved algorithm and the lightweight of the model, adapting to Robomaster EP, and meeting the deployment requirements of embedded devices for the detection and identification of tennis balls.

**Keywords:** tennis ball detection; lightweight; convolutional coordinate attention; feature fusion; loss function; activation function



**Citation:** Zhao, Y.; Lu, L.; Yang, W.; Li, Q.; Zhang, X. Lightweight Tennis Ball Detection Algorithm Based on Robomaster EP. *Appl. Sci.* **2023**, *13*, 3461. <https://doi.org/10.3390/app13063461>

Academic Editor: Hui Yuan

Received: 29 November 2022

Revised: 28 January 2023

Accepted: 2 February 2023

Published: 8 March 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Tennis is gaining popularity as a fashionable aerobic sport, and the number of tennis enthusiasts is increasing yearly. Despite the popularity of tennis, the experience of the sport is hampered by the large number of tennis balls picked up from the court, which significantly affects the enthusiasts' enthusiasm. Most of the traditional tennis ball picking devices are mainly human-driven and require manual intervention, relying on the downward pressure action of the arm to pick up the tennis balls, which requires multiple repetitions of the arm lifting action during the picking process, wasting a lot of time and energy and being less efficient. To solve the problem of the low efficiency of manual ball pickup, the mainstream tennis ball pickup devices on the market are mainly divided into multi-degree-of-freedom-based robotic arm pickup, impeller rotation-based inhalation pickup, and motor-driven paddle rotation-based pickup. The efficiency of using a robotic arm to pick up the ball is low; only one ball can be picked up at a time, and multiple servos are required to cooperate, which is difficult to achieve. Inhalation pickup using high-speed

impeller rotation requires high motor power. It consumes much energy, but the pickup is accurate and efficient. With the motor-driven paddle rotation pickup, the device structure is simple, easy to control, and less challenging to operate. However, there will be a ball jamming problem.

With the rapid development of the robotics industry, service robots are becoming increasingly popular. The International Federation of Robotics (IFR) defines a service robot [1] as a robot that works semi-autonomously or fully autonomously. Service robots are divided into two categories according to their use: home service robots and professional service robots. Service robots for the tennis field are rare in China, and most still need models based on natural environments. However, the world's first tennis AI pick-up machine, Tennibot [2], has emerged abroad, using computer vision and artificial intelligence technology to locate, detect, and catch tennis balls automatically. Intelligent-based pick-up tennis robots are developing towards intelligence and commercialization. Therefore, service robots for the tennis field are in high demand and have good market value and some relevance.

The Robomaster EP is a programming-oriented educational engineering robot from DJI with robust scalability and programmability, as shown in Figure 1. A parallel robotic arm is used instead of the gimbal structure mounted in the center of the chassis, retaining the image transmission system. A monocular camera is fitted on top of the arm for real-time display and transmission of images, and a mechanical claw is fitted at the end of the arm for more complex tasks. The infrared depth sensor is assembled on top, based on the Time of Flight TOF [3] principle, through which the sensor emits modulated near-infrared light. After encountering the reflection of an object, the sensor calculates the distance to the object by calculating the time difference or phase difference between the light emission and reflection to achieve intelligent obstacle avoidance [4] and environment perception [5], and through the interface call algorithm, real-time detection and identification, improving computational performance. Unlike other tennis ball picking devices, Robomaster EP has a more comprehensive and intelligent overall performance, with the following advantages: (1) The size is small, with a total weight of 3.3 kg, which makes it easy to carry. (2) The range of movement of the robot arm and the opening and closing distance of the robot claw is extensive. The robot arm and claw can be used together to move more flexibly and efficiently. (3) The detection range is more expansive. The infrared depth sensor on top of the arm has a detection range of 0.1–10 m. A monocular camera is mounted on top of the arm to display images in real time. (4) The device has strong scalability and programmability, good compatibility, and it easily calls and deploys algorithms.



Figure 1. Robomaster EP.

For tennis ball detection algorithms, image processing methods are generally used for detection and recognition. For example, the Hough transform [6] is used to detect tennis balls segmented from video frames by color segmentation. The basic idea is that a tennis ball can be considered a circle from every direction. Any circle can be mapped from an expression under a two-dimensional coordinate system to a three-dimensional space. However, problems include difficulty in feature extraction, susceptibility to the external environment, low detection accuracy, and poor recognition. In addition, Robomaster EP has high requirements for algorithm model parameters and needs to be adapted to lightweight models for easy portability and implementation of algorithms. Traditional deep learning models, such as SSD [7] and Faster R-CNN [8], have a large number of model parameters and computation, complex model network structure, difficult deployment, lengthy and costly deployment time, and low performance and detection efficiency of the algorithms, which do not meet the deployment needs of embedded devices. This led to a transition to general-purpose deep learning models, such as the YOLOv5 [9] algorithm in the YOLO [10] family of single-stage target detection algorithms, which runs faster and detects quickly for fast detection and identification of tennis balls but has lower detection accuracy compared to two-stage target detection algorithms. Robomaster EP has a more complex model parameter count, computation, and network structure compared to lightweight models. This leads to a more expensive deployment to meet the actual demand. To meet the deployment needs of embedded devices and to improve detection accuracy, Robomaster EP needs to be adapted to the lightweight model to improve efficiency.

According to the existing research, the following three main issues should be considered in a lightweight tennis ball detection algorithm based on Robomaster EP. First, an improved YOLOv5s algorithm is proposed, as the original YOLOv5s algorithm cannot meet the deployment requirements of embedded devices and needs to improve detection accuracy and build lightweight models. Second, the deployment and invocation of the algorithm are implemented through programming for the detection and identification of tennis balls. Thirdly, the detection performance in real scenarios must be verified. An actual tennis court has many tennis balls, as well as the phenomenon of tennis ball occlusion. It is susceptible to weather and light effects, with different scenarios and light intensities leading to different detection results. Using image processing methods, features are challenging to extract, detection accuracy is low, and recognition results could be better. A deep-learning-based tennis ball detection algorithm can improve these problems with better detection performance and robustness in different detection scenarios.

In summary, this study proposes a lightweight tennis ball detection algorithm, YOLOv5s-Z, based on Robomaster EP to achieve accurate detection and recognition of tennis balls. The main work and contributions are as follows: firstly, an improved tennis ball detection algorithm is proposed to construct a lightweight model to improve the detection accuracy while compressing the model and reducing the number of parameters and computation. Secondly, convolutional coordinate attention is incorporated into the Backbone to enhance the ability of the Backbone network to sense the field and capture location information. The original Concat [11] module in feature fusion is modified into a weighted bidirectional feature pyramid W-BiFPN with settable learning weights to enhance the feature fusion capability and achieve efficient weighted feature fusion and bidirectional cross-scale connectivity, introducing EIOU Loss [12] and Meta-ACON [13] to improve the detection accuracy. Finally, the improved algorithm is deployed into Robomaster EP, directly invoked through the interface, and used in real scenarios to detect and identify tennis balls.

## 2. Related Work

### 2.1. Target Detection Algorithms

A traditional target detection algorithm's main steps are: first, select the candidate region on the image, then perform feature extraction, and finally use the classifier for classification. There are disadvantages, such as low detection accuracy, high computational cost, and poor robustness. Deep-learning-based target detection algorithms rely on a large

amount of data to obtain feature information through autonomous learning and model training of convolutional neural networks to achieve high-precision and high-efficiency detection and recognition. Deep-learning-based target detection algorithms are divided into two main categories. The first category is the two-stage target detection algorithm based on region recommendation. The main steps are first generating candidate regions and then classifying the candidate regions. Representative algorithms include R-CNN [14], Fast R-CNN [15], Faster R-CNN, and Mask R-CNN [16]. The main idea of this type of algorithm is to extract the image features by the convolutional neural network, use the region extraction network to give the candidate frame of the image target to be detected, and use the detection head with convolution to classify the target in the candidate frame to complete the detection with high accuracy but slow speed. Another category is the one-stage method target detection algorithm based on the regression idea, that is, end-to-end, single-stage detection of objects for a picture using only one convolutional neural network prediction to obtain the category probability and position coordinates of different targets, which significantly improves the detection speed and the operation speed of the algorithm to meet the demand of real-time detection. The representative algorithms are RetinaNet [17], YOLO, and SSD, the algorithm mainly based on the detection of the target. The main idea of both YOLO and SSE is to extract features directly in the network, transforming target localization into a border regression problem and completing the localization and classification tasks at once, which is faster but less accurate.

## 2.2. Tennis Ball Detection and Identification

There are mainly machine learning, image processing, and deep-learning-based methods for ball detection and recognition. Machine learning methods are too complex, as balls generally have only two features, color and shape, requiring more training time. The training process is relatively complex and not suitable for tennis ball detection. Image processing methods mainly use computer vision or image sensors to process images and apply target-ranging algorithms to identify tennis balls through digital image processing techniques [18] for ranging. Standard methods include the detection of tennis balls based on the Hough transform, developing tennis ball recognition based on OpenCV [19], and using vision sensors to identify tennis balls. For example, in the image processing workflow using toolboxes such as OpenCV, the image is pre-processed, features are extracted using a correlation algorithm, different radius thresholds are set to filter the target, and finally the contour features are used to determine whether it is a tennis ball. Compared to the manual ball-picking method, the detection efficiency of the image-processing-based method is improved. However, it is affected by different detection scenes, periods, and tennis ball occlusion. For example, different detection scenes and periods interfere with tennis ball identification to a certain extent. The change of light causes some target features to be challenging to extract. The phenomenon of tennis ball occlusion leads to missed and false detection.

Compared to machine and image processing methods, deep-learning-based methods can achieve accurate detection and recognition of tennis balls. Gu et al. [20] proposed a model based on AlexNet [21] and SSD for tennis ball recognition, using deep learning to divide the image recognition into two steps and the AlexNet network model to test whether there is a tennis ball in the image. If there is a tennis ball, the method uses an SSD network model to locate the tennis ball; otherwise, it uses AlexNet to continue checking the following image. Gu et al. [22] proposed a deep-learning-based tennis ball collection robot that used the YOLO model for tennis ball recognition and implemented it on an NVIDIA [23] Jetson TX1 board. However, the traditional model has many parameters, extensive computation, low detection accuracy, and complex model structure, which could be more conducive to porting to embedded devices.

### 2.3. YOLOv5

The network structure of YOLOv5 is shown in Figure 2. YOLOv5 is currently a single-stage target detection algorithm with good performance, based on the idea of regression, which reduces the target detection problem to a regression problem. The network model consists of the input side, the Backbone, a Neck network layer, and an output side. The primary role of the input is to pre-process the input image, mainly including Mosaic data enhancement, adaptive anchor frame calculation, and adaptive image scaling. The primary role of the Backbone is to extract image features, mainly including Focus, Conv, C3, and SPPF. The primary role of the Neck network layer is to fuse the information from different network layers in the Backbone to enhance the detection capability of the network. The structure of FPN+PAN is mainly adopted to achieve the transfer of target feature information of different sizes, strengthen the network feature fusion capability, and solve the multi-scale problem. The primary role of the output is to predict targets of different sizes on different feature maps, generate bounding boxes, and predict categories, which mainly includes the calculation of Loss functions and non-maximum suppression operations.

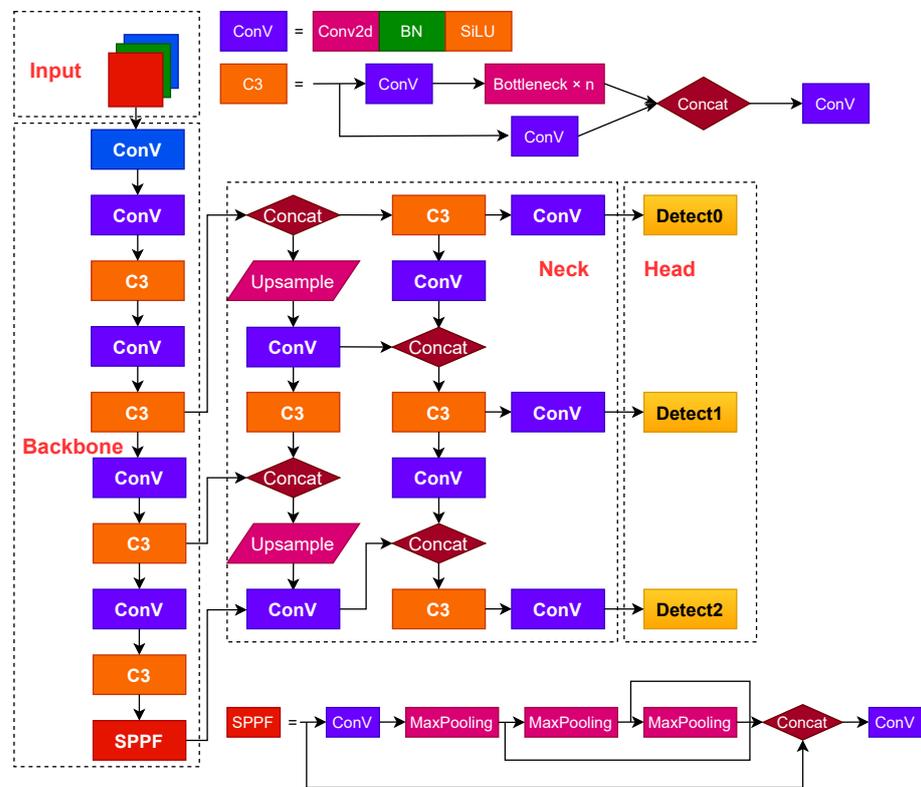


Figure 2. YOLOv5s network structure.

### 2.4. Lightweight Neural Networks

Since the introduction of AlexNet, neural networks have been widely used in image classification, image segmentation, target detection, and other fields. Due to the limitation of storage space and power consumption, storing and computing neural networks on embedded devices takes much work. With the development of iterative updates of embedded devices and diversification of model application scenarios, traditional neural network models are gradually being replaced by lightweight neural network structures due to many parameters and computation and complex network structures. In recent years, many excellent lightweight neural network structures have emerged. For example, EfficientNet [24] uses a model composite scaling method and AutoML [25] technique to scale up a convolutional neural network in a more structured way using a simple and efficient composite coefficient. SqueezeNet [26] uses a different convolutional approach from the traditional one by propos-

ing a fire module. MobileNet [27] is based on ShuffleNet [28], uses grouped convolution to reduce the number of parameters, and uses channel shuffling to exchange information between different groups. GhostNet [29] proposes a new Ghost module, the basic unit of a neural network, that uses inexpensive operations to generate as many feature maps as possible at a small cost without changing the size of the output map or the channel size, reducing the number of parameters and computation of the model.

### 3. YOLOv5s-Z

In this study, the network structure is improved and optimized for version 6.1 of the YOLOv5s algorithm, as shown in Figure 3. It consists mainly of the input, the lightweight G-Backbone and G-Neck network layers, and the output.

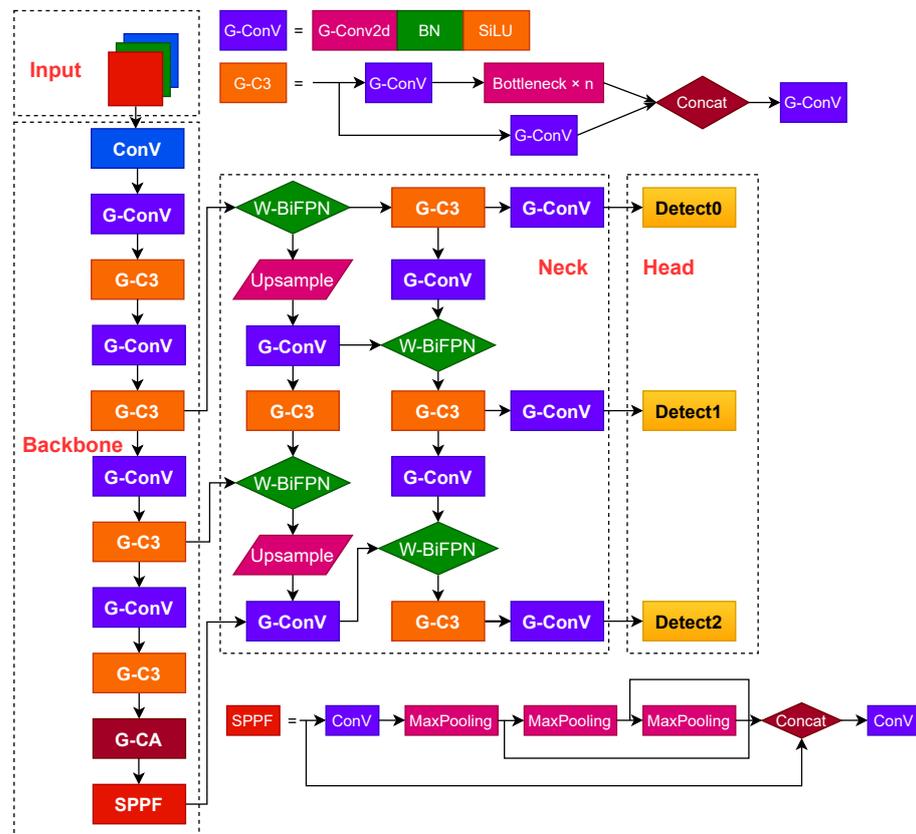


Figure 3. YOLOv5s-Z network structure.

#### 3.1. G-Backbone

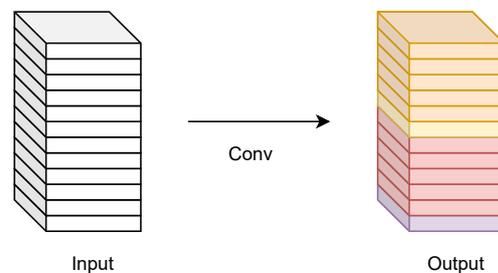
Based on the lightweight neural network GhostNet, the Conv and C3 modules are subjected to Ghost convolution as well as lightweight processing, and the new G-Conv and G-C3 modules are proposed to build a lightweight G-Backbone, which incorporates the convolutional coordinate attention mechanism G-CA in the G-Backbone to enhance the perceptual field and the ability to capture location information of the Backbone network and better extract the features of the input image. GhostNet is a new end-side neural network architecture proposed by Huawei Noah’s Ark Lab, which builds the lightweight neural network GhostNet by stacking Ghost modules to obtain the Ghost BottleNeck. The structure of the G-Backbone network is shown in Table 1.

**Table 1.** G-Backbone network structure.

| Layer | From | Params  | Module | Arguments        |
|-------|------|---------|--------|------------------|
| 0     | −1   | 4656    | Conv   | [3, 32, 6, 2, 2] |
| 1     | −1   | 12,416  | G-Conv | [32, 64, 3, 2]   |
| 2     | −1   | 14,776  | G-C3   | [64, 64, 1]      |
| 3     | −1   | 43,232  | G-Conv | [64, 128, 3, 2]  |
| 4     | −1   | 51,472  | G-C3   | [128, 128, 2]    |
| 5     | −1   | 160,160 | G-Conv | [128, 256, 3, 2] |
| 6     | −1   | 189,808 | G-C3   | [256, 256, 3]    |
| 7     | −1   | 615,200 | G-Conv | [256, 512, 3, 2] |
| 8     | −1   | 621,472 | G3     | [512, 512, 1]    |
| 9     | −1   | 24,608  | G-CA   | [512, 512, 32]   |
| 10    | −1   | 700,208 | SPPF   | [512, 512, 5]    |

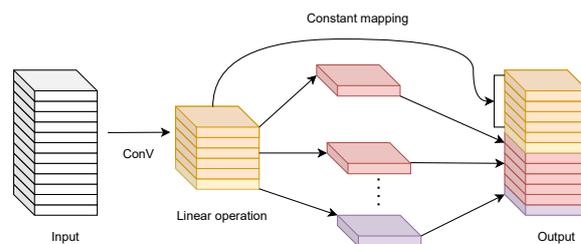
In the above table, **Layer** denotes the number of layers, **From** denotes the layer from which the module comes, where −1 denotes the previous layer, **Params** denotes the number of parameters, **Module** denotes the module’s name, **Arguments** denote information about the module, mainly including the number of input channels, output channels, the size of the convolution kernel, and step size information.

For a traditional convolutional neural network, the dimension of the input feature map is  $c \times h \times w$ , where  $c$  represents the number of channels,  $h$  represents the height of the feature map, and  $w$  represents the width of the feature map. The size of the convolutional kernel is  $c \times k^2 \times n$ , where  $k$  represents the size of the convolutional kernel, and  $n$  represents the number of channels of the output feature map. Let the size of the output feature map be  $h' \times w' \times n$ . The total computation is  $h' \times w' \times n \times c \times k^2$ , and the number of parameters is  $c \times k^2 \times n$ . This results in the ordinary output data of convolution  $Y = X * f + b$ , where  $X$  represents the input data,  $f$  represents the  $c \times n$  convolution operations with a convolution kernel size of  $k^2$ , and  $b$  represents the bias term. The process of ordinary convolution is shown in Figure 4.



**Figure 4.** The normal convolution.

In contrast to ordinary convolution, the process of Ghost convolution is shown in Figure 5.



**Figure 5.** The Ghost convolution.

The process of Ghost convolution is as follows: first, a small number of feature maps  $Y'$  are generated by ordinary convolution, and the feature maps of each channel in the feature maps  $Y'$  are used to generate Ghost feature maps  $Y_{ij}$  by linear operations.

Then, the two sets of different feature maps are stitched together according to the channels. Finally, the same output result as ordinary convolution is obtained. Then, the output data  $Y' = X * f' + b$  of Ghost convolution is calculated as shown in Equation (1), where  $\phi_{ij}$  denotes the  $j$  linear operation performed on the  $i$  feature map  $Y_i'$  generated in the first step of the convolution.  $Y_i'$  denotes the  $i$  feature map in  $Y'$ .

$$Y_{ij} = \phi_{ij} * Y_i', i \in [1, m], j \in [1, s] \tag{1}$$

The Ghost module mainly contains a tiny number of convolutions, a constant mapping, and  $m \times (s - 1)$  linear operations, each with an average kernel size of  $d \times d$ . The formula for the theoretical speed-up ratio of the Ghost module to upgrade the ordinary convolution is shown in Equation (2).

$$r_s = \frac{n \cdot h' \cdot w' \cdot c \cdot k^2}{\frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k^2 + (s - 1) \cdot \frac{n}{s} \cdot h' \cdot w' \cdot d^2} = \frac{c \cdot k^2}{\frac{1}{s} \cdot c \cdot k^2 + \frac{s-1}{s} \cdot d^2} \approx \frac{s \cdot c}{s + c - 1} \approx s \tag{2}$$

The formula for the compression ratio of the number of parameters between ordinary convolution and Ghost convolution is shown in Equation (3).

$$r_c = \frac{n \cdot c \cdot k^2}{\frac{n}{s} \cdot c \cdot k^2 + \frac{s-1}{s} \cdot d^2} \approx \frac{s \cdot c}{s + c - 1} \approx s \tag{3}$$

In the above equation, the convolution kernel's size represents the convolution kernel's size when linear mapping is performed for each channel. The Backbone provides for better feature extraction of the input images. Table 2 shows the comparison results between the YOLOv5s model and the YOLOv5s-Ghost model.

**Table 2.** Comparison results between YOLOv5s and YOLOv5s-Ghost.

| Model         | Parameters | GFLOPs | Weight/MB |
|---------------|------------|--------|-----------|
| YOLOv5s       | 7,022,326  | 15.8   | 14.5      |
| YOLOv5s-Ghost | 3,684,542  | 8.1    | 7.9       |

As seen from the results in Table 2, replacing the Ghost module reduces the number of parameters by 48%, the amount of computation by 49%, and the model's size by 46%. The experimental results demonstrate that the Ghost module achieves a lighter network structure.

### 3.2. Convolutional Coordinate Attention

This study proposes a convolutional coordinate attention G-CA, as shown in Figure 6. By incorporating the convolutional coordinate attention mechanism in the G-Backbone network, the network can obtain location information of a larger area through multiple convolutions, which further enhances the ability of the Backbone network to sense the field and capture location information and enhances the expression ability of the learning features of the mobile network.

Research in neural networks has shown that channel attention significantly improves the model's performance but ignores some vital location information that facilitates the generation of spatially selective attention maps. Therefore, to alleviate the loss of location information caused by two-dimensional global pooling proposed by attention mechanisms such as SENet [30] and CBAM [31], a novel attention mechanism designed for lightweight networks called coordinate attention [32] was proposed by the National University of Singapore. Compared to channel attention, coordinate attention transforms the feature tensor into individual feature vectors using two-dimensional global pooling, decomposing channel attention into two one-dimensional feature encoding processes that aggregate features along two directions, one of which captures remote dependencies along the spatial direction, and the other retains precise location information along the spatial direction.

The resulting feature maps are eventually encoded separately to produce a pair of direction-aware and position-sensitive feature maps that are complementarily applied to the input feature maps and used to enhance the precise localization of targets.

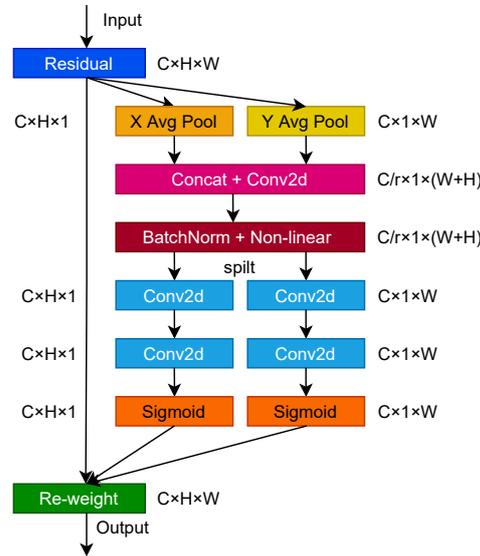


Figure 6. Convolutional coordinate attention.

Coordinate attention consists of two main steps: coordinate information embedding and coordinate attention generation. First, given an input feature map  $X$  with dimension  $c \times h \times w$ , and using two pooling kernels with spatial ranges  $(H,1)$  or  $(1,W)$  to encode each channel along the horizontal and vertical coordinates, respectively, the output of the  $c$  channel with height  $h$  and the  $c$  channel with width  $w$  is calculated as shown in Equations (4) and (5):

$$Z_c^h(h) = \frac{1}{W} \sum_{0 \leq i \leq W} X_c(h, i) \tag{4}$$

$$Z_c^w(w) = \frac{1}{H} \sum_{0 \leq j \leq H} X_c(j, w) \tag{5}$$

The above two transformations perform feature aggregation along two directions, respectively, and cascade to generate two feature maps, which generate feature maps of spatial information in horizontal and vertical directions  $f$  by a convolution operation, which is beneficial to the network for accurate target localization, and the calculation formula is shown in Equation (6):

$$f = \delta(F_1([Z^h, Z^w])) \tag{6}$$

After the coordinate information is embedded, the above changes are subjected to the cascade operation, which is a nonlinear activation function that is an intermediate feature map of spatial information encoded along the horizontal and vertical directions, which is decomposed into two tensor sums along the spatial dimension. The transformation operation is then performed using the convolutional transform function, which in turn yields the attention weights of the two spatial directions as  $g^h$  and  $g^w$ , respectively, calculated as shown in Equations (7) and (8):

$$g^h = \sigma((F_h(f^h))) \tag{7}$$

$$g^w = \sigma((F_w(f^w))) \tag{8}$$

The  $\sigma$  in the above equation is the sigmoid [33] activation function, and to reduce the complexity and computational overhead of the model, the number of channels is usually

reduced using a suitable scaling ratio, and the output is expanded as attention weights, respectively. The final output of the coordinate attention mechanism is obtained, and the calculation formula is shown in Equation (9):

$$y_c(i, j) = X_c(i, j) \times g_c^h(i) \times g_c^w \tag{9}$$

### 3.3. G-Neck Network Layer

A lightweight G-Neck network layer is constructed, and a weighted bidirectional feature pyramid W-BiFPN with settable learning weights is proposed to be incorporated into the G-Neck network layer. Based on the weighted bidirectional feature pyramid BiFPN, a settable learning weight coefficient  $W$  is set to strengthen the feature fusion capability further and improve the detection speed to achieve more efficient weighted bidirectional feature fusion, which is more convenient for the network to extract features.

Figure 7 shows the development process of Neck networks in recent years, starting with the top-down unidirectional fusion FPN feature pyramid structure, which establishes a top-down pathway for feature fusion and uses feature maps for prediction, which can improve accuracy to a certain extent but will be limited by the one-way information flow. The network structure of PANet [34] for bidirectional fusion adds to the FPN [35], a bottom-up path aggregation network. The main idea is that the higher-level feature maps have more vital semantic information for object classification. In comparison, the bottom-level feature maps have more vital positional information for object localization. PANet network passes the positional information from the bottom-level layer to the prediction feature layer, making the prediction feature layer have higher semantic information and positional information, which is more conducive to target detection and thus improves detection accuracy. The PANet network structure also has Adaptive feature pooling, and complete connection fusion is proposed in the PANet network structure. Adaptive feature pooling is used to aggregate features between different layers to ensure the integrity and diversity of features, and complete connection fusion is used to obtain an accurate prediction layer. The main idea of the weighted bidirectional feature pyramid network structure BiFPN [36] is compelling bidirectional cross-connections and weighted feature fusion and top-down feature fusion followed by bottom-up feature fusion. Multi-scale feature fusion is the aggregation of features at different resolutions.

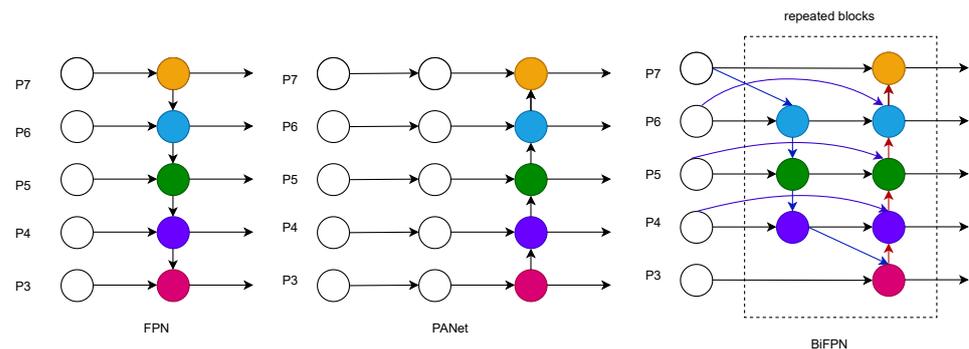


Figure 7. FPN PANet and BiFPN structure.

Different input features have different resolutions, so the contribution to the output features is uneven. To solve the problem, an additional weight is added to each input so that the network learns the importance of each input feature; biFPN uses a fast normalized weighted fusion method, calculated as shown in Equation (10):

$$o = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i \tag{10}$$

where  $w_i \geq 0$  is achieved by adding the ReLU activation function after each  $w_i$ , the weights are divided by the weighted sum of all values to achieve the normalization operation,

and the value of each normalization weight is between zero and one. BiFPN integrates bidirectional cross-scale connectivity and fast normalized fusion, and the computational equations of the fusion feature process are shown in Equations (11) and (12) for BiFPN at Level 6 nodes.

$$p_6^{td} = Conv\left(\frac{w_1 \cdot p_6^{in} + w_2 \cdot Resize(p_7^{in})}{w_1 + w_2 + \epsilon}\right) \tag{11}$$

$$p_6^{out} = Conv\left(\frac{w'_1 \cdot p_6^{in} + w'_2 \cdot p_6^{td} + w'_3 \cdot Resize(p_5^{out})}{w'_1 + w'_2 + w'_3 + \epsilon}\right) \tag{12}$$

In the above equation,  $p_6^{td}$  denotes the intermediate features of the sixth layer in the top-down path, and  $p_6^{out}$  denotes the output features of the sixth layer in the bottom-up path. To improve efficiency, feature fusion is performed using depth-separable convolution, and batch normalization and activation functions are added after each convolution, where *Resize* is the upsampling or downsampling operation, and *w* is the learned parameter to distinguish the importance of different features in the feature fusion process.

### 3.4. EIOU Loss

The default Loss function in YOLOv5s is CIOU Loss [37], and the formula for CIOU is shown in Equation (13). The aspect ratio of the regression frame is considered in the Loss function based on DIOU Loss [38]. The loss of the detection frame scale and the loss of the length and width are added to make the prediction frame more realistic and further improve the regression accuracy. The DIOU calculation formula is shown in Equation (14). The disadvantage is that the aspect ratio needs to be more specific, and the balance of complex and easy samples is not considered. In this study, EIOU Loss is introduced, and the formula of EIOU Loss is shown in Equation (15). Based on the penalty term of CIOU Loss, the aspectual influence factors of the prediction frame and the rear frame are split. The length and width of the prediction and the rear frames are calculated separately to solve the problems in CIOU Loss.

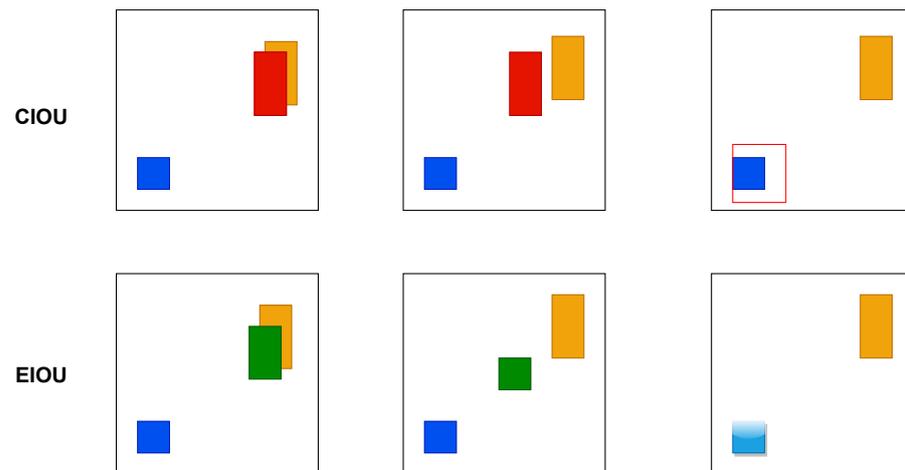
$$L_{CIOU} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{13}$$

$$L_{DIOU} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} \tag{14}$$

$$L_{EIOU} = L_{IoU} + L_{dis} + L_{asp} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \frac{\rho^2(w, w^{gt})}{c_w^2} + \frac{\rho^2(h, h^{gt})}{c_h^2} \tag{15}$$

In the above equation, *b* and  $b^{gt}$  denote the centroids of the prediction frame and the actual frame, respectively,  $\rho$  denotes the Euclidean distance calculated between the two centroids, and *c* denotes the diagonal distance of the minimum closed region containing both the prediction frame and the actual frame. We can see that the *EIOU* Loss consists of three main components: the overlap loss between the predicted frame and the real frame  $L_{IoU}$ , the center distance loss between the predicted frame and the real frame  $L_{dis}$ , and the width and height loss between the predicted frame and the real frame  $L_{asp}$ .  $L_{IoU}$  and  $L_{dis}$  continue the method in *CIOU*, and the width and height loss  $L_{asp}$  directly makes the difference between the width and height of the predicted frame, and the real frame minimizes the difference between the width and height of the predicted frame and the rear frame, which makes the convergence faster.

The comparison diagram of the iterative process of CIOU and EIOU Loss prediction frames is shown in Figure 8. The red and green boxes represent the regression process of CIOU and DIOU prediction frames, the blue box is the actual frame, and the yellow box is the pre-defined anchor frame. The comparison chart shows that the width and height of EIOU can be increased or decreased at the same time, but not CIOU. In general, EIOU outperforms CIOU, so this study introduces EIOU Loss as the Loss function.



**Figure 8.** Comparison of CIOU and EIOU iterative process.

### 3.5. Meta-ACON Activation Function

The default activation function in YOLOv5s is ReLU [39], which is the most common activation function, mainly because of its non-saturation and sparsity properties, with the disadvantage that it can have the severe consequence of neuronal necrosis. ReLU is essentially a function and is calculated as shown in Equation (16):

$$ReLU(x) = MAX(0, x) \tag{16}$$

Consider the  $n$  values of the standard maximum function  $MAX$ , whose smoothness and differentiability are approximated by  $S_\beta$ , calculated as shown in Equation (17):

$$S_\beta(x_1, x_1, x_2, \dots, x_n) = \frac{\sum_{i=1}^n X_i \cdot e^{\beta x_i}}{\sum_{i=1}^n e^{\beta x_i}} \tag{17}$$

where  $\beta$  is a connection coefficient, and  $S_\beta$  tends to the maximum when  $\beta$  tends to infinity, and  $S_\beta$  tends to the arithmetic mean when  $\beta$  tends to zero. In neural networks, the common activation function is expressed in the form of  $max(\eta_a(x), \eta_b(x))$ , where  $\eta_a(x)$  and  $\eta_b(x)$  are linear functions.

The Swish [40] activation function, obtained by the NAS [41] search technique, is an approximate smoothing of the ReLU activation function. The general form of Swish’s ACON activation function is obtained by analyzing the general form of the Maxout [42] series of activation functions of ReLU. The ACON generalization yields ACON-A, ACON-B, ACON-C, the Meta-ACON, and other variant forms. This study introduces Meta-ACON, which adaptively selects whether or not to activate neurons and introduces a switching factor to learn the parameter switching between nonlinear activation and linear non-activation to improve the detection accuracy of the algorithm.

## 4. Experimental Results

### 4.1. Experimental Environment

This experiment was based on the Pytorch 1.11.0 framework, CUDA version 11.5, and conducted on the PyCharm platform, and the model training was accelerated by GPU. The specific experimental environment parameters were configured as shown in Table 3.

**Table 3.** Experimental environment.

| Name                    | Configuration Parameters           |
|-------------------------|------------------------------------|
| Operating System        | Windows 11 64-bit                  |
| CPU                     | Intel Core i5-12400F               |
| GPU                     | NVIDIA GeForce RTX3060 12G         |
| Memory                  | 16GB                               |
| Python Version          | 3.8                                |
| Deep Learning Framework | Pytorch 1.11.0 CUDA 11.5           |
| Experimental Platform   | PyCharm Community Edition 2022.2.3 |

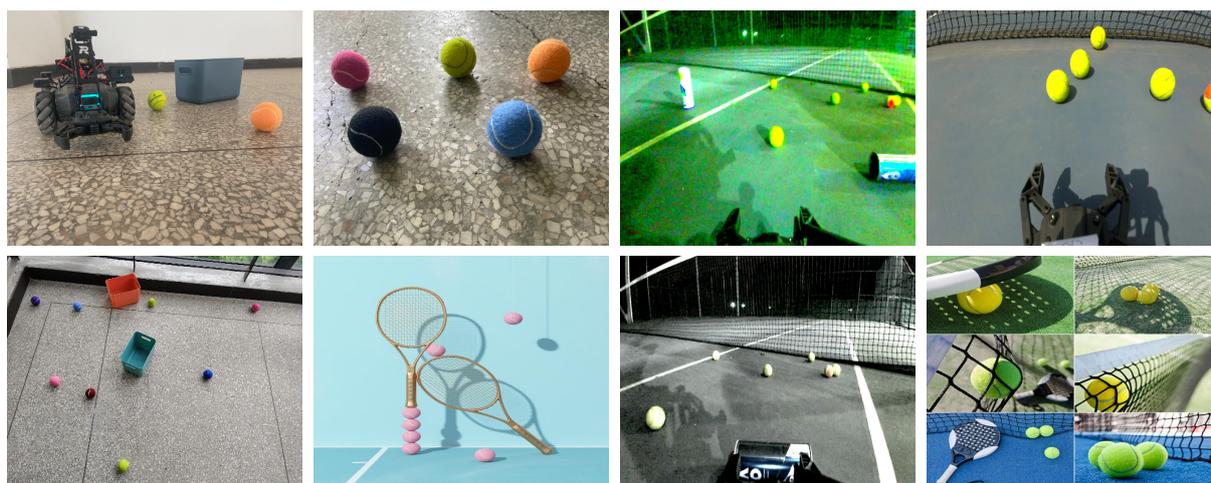
4.2. Datasets

In this study, 1180 homemade tennis ball datasets were used. The sources of the datasets included tennis ball pictures taken by the monocular camera fitted with Robomaster EP, tennis ball pictures taken by mobile phones, and tennis ball pictures obtained by crawlers, containing different colors, different scenes, and different time tennis ball pictures to ensure the diversity of the datasets. The specific information is shown in Table 4.

**Table 4.** Datasets.

| Category | Parameters                               |
|----------|--|
| Color    | green, blue, orange, purple, pink, black |
| Scene    | tennis court, laboratory, open space     |
| Period   | morning, noon, evening                   |

The dataset was normatively annotated with explicit annotation using Make Sense. At the same time, the training set, validation set, and test set were divided according to 8:1:1. Before training the model, some of the datasets were pre-processed, including randomly increasing or decreasing the brightness and contrast of the images, and combined with the Mosaic data enhancement method that comes with YOLOv5 to enrich the datasets and enhance the generalization ability of the model and the robustness of the validation model. Figure 9 shows an example graph representing the dataset.



**Figure 9.** Representative datasets.

4.3. Training Strategy and Evaluation Index

All models were trained from scratch using the same training strategy and parameters, hyperparameter profiles, and pre-warm training parameters, all without pre-training weights. The parameters were updated iteratively using the SGD optimizer with an initial learning rate of 0.01, a momentum parameter of 0.937, and a batch size of 16. The warm-up

method with an epoch of 3 and a momentum parameter of 0.8 was used to warm up the learning rate, and all models were trained for 300 rounds.

In this study, the model was evaluated using evaluation metrics including mAP@0.5, Recall, Parameters, GFLOPs, and Weight. Here, mAP@0.5 represents the average AP at the IOU threshold of 0.5, which is used to reflect the recognition ability of the model. Recall represents the ratio of correctly detected positive samples to all positive samples, Parameters represent the number of parameters of the model, and GFLOPs represent the number of floating point operations performed by the model. Parameters and GFLOPs are important indicators of the model algorithm, which measure the complexity of the model in the dimensions of time and space, respectively. The calculation equations are shown in Equations (18)–(21).

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i \quad (18)$$

$$AP = \int_0^1 P(R)dR \quad (19)$$

$$Precision = \frac{TP}{TP + FP} \quad (20)$$

$$Recall = \frac{TP}{TP + FN} \quad (21)$$

where  $n$  represents the number of categories,  $p$  represents precision,  $R$  represents recall,  $P(R)$  represents the precision and recall curves,  $TP$  represents the number of detection frames with  $IOU \geq$  set threshold,  $FP$  represents the number of detection frames with  $IOU \leq$  set threshold, and  $FN$  represents the number of missed targets.

#### 4.4. Comparative Experimental Results and Analysis

To verify the effectiveness of the improved algorithm, commonly used target detection algorithms were selected for comparative analysis, and the same training strategy and parameters were used for each group of experiments. The experimental results are shown in Table 5.

**Table 5.** Comparison of different lightweight models for tennis ball detection and localization.

| Model                | mAP@0.5      | Recall      | Parameters       | GFLOPs     | Weight/MB  |
|----------------------|--------------|-------------|------------------|------------|------------|
| YOLOv5s              | 0.957        | 0.911       | 7,022,326        | 15.8       | 14.5       |
| SSD                  | 0.839        | 0.443       | 26,285,486       | 63         | 91         |
| Faster R-CNN         | 0.93         | 0.935       | 28,536,850       | 181        | 109        |
| YOLOv5s+Shufflenetv2 | 0.939        | 0.915       | 3,792,950        | 7.9        | 8.1        |
| YOLOv5s+Mobilenetv3  | 0.947        | 0.895       | <b>3,542,756</b> | <b>6.3</b> | <b>7.5</b> |
| YOLOv3               | 0.964        | 0.944       | 61,523,734       | 154.9      | 123.6      |
| YOLOv3-tiny          | 0.945        | 0.884       | 8,669,876        | 12.9       | 17.4       |
| YOLOv4               | 0.909        | 0.885       | 64,363,101       | 60.5       | 244        |
| YOLOv4-tiny          | 0.868        | 0.854       | 6,056,606        | 7.0        | 22.4       |
| YOLOX-s              | 0.966        | 0.966       | 8,968,255        | 26.9       | 34.3       |
| YOLOX-tiny           | 0.971        | 0.969       | 5,055,855        | 15.4       | 19.4       |
| YOLOv7               | 0.962        | 0.97        | 37,194,710       | 104.9      | 71.3       |
| YOLOv7-tiny          | 0.968        | <b>0.98</b> | 6,014,038        | 13.1       | 11.7       |
| Ours                 | <b>0.978</b> | <b>0.98</b> | 4,100,759        | 8.8        | 8.8        |

From the results of the comparison experiments, it can be seen that the algorithm proposed in this study had the most comprehensive performance, taking into account the needs of lightweight models and detection accuracy, with solid generalization ability, the highest mean accuracy and recall, and slightly more parameters, computation, and model size than the lightweight neural networks Mobilenet and Shufflenet, but Mobilenet and Shufflenet had lower mean precision and recall. Compared to the original YOLOv5s

algorithm, the number of parameters and computation were reduced by 42% and 44%, respectively, the model size was reduced by 39%, and the average precision improved by 2%, verifying the effectiveness of the improved algorithm. Compared with the classical target detection algorithms SSD and Faster R-CNN, the overall performance was much better, with a significant increase in mean accuracy and recall and a significant reduction in the number of parameters, computation, and model size. The performance of the YOLO family of target detection algorithms, YOLOv3, YOLOv4, YOLOX, and the lightweight model, was still much better. Even when compared to the current best-performing YOLOv7 algorithm, the proposed algorithm has better overall performance, reducing the number of parameters and computation by 32% and 33%, respectively, and the model size by 25% compared to YOLOv7-tiny, validating the effectiveness of the improved algorithm and the lightness of the model.

#### 4.5. Ablation Experimental Results and Analysis

To verify the feasibility of the improvement module, six sets of ablation experiments were designed based on YOLOv5s. The same training strategy was used for each set of experiments. The results of the ablation experiments are shown in Table 6. Where improvement point 1 indicates the introduction of the lightweight G-Backbone, improvement point 2 indicates the addition of the G-CA attention mechanism, improvement point 3 indicates the addition of the W-BiFPN module, improvement point 4 indicates the introduction of EIOU Loss, and improvement point 5 indicates the introduction of the Meta-ACON activation function.

**Table 6.** Ablation experiments.

| Model    | G-Backbone | G-CA | W-BiFPN | EIOU Loss | Meta-ACON | mAP@0.5      | Recall      | Parameters       | GFLOPs     | Weight/MB  |
|----------|------------|------|---------|-----------|-----------|--------------|-------------|------------------|------------|------------|
| YOLOv5s  |            |      |         |           |           | 0.957        | 0.911       | 7,022,326        | 15.8       | 14.5       |
| Improve1 | ✓          |      |         |           |           | 0.956        | 0.926       | <b>3,684,542</b> | <b>8.1</b> | <b>7.9</b> |
| Improve2 |            | ✓    |         |           |           | 0.961        | 0.919       | 7,046,934        | 15.9       | 14.5       |
| Improve3 |            |      | ✓       |           |           | 0.959        | 0.918       | 7,170,943        | 16.4       | 14.8       |
| Improve4 |            |      |         | ✓         |           | 0.963        | 0.97        | 7,022,326        | 15.8       | 14.5       |
| Improve5 |            |      |         |           | ✓         | 0.962        | 0.931       | 7,421,478        | 16.2       | 15.4       |
| Ours     | ✓          | ✓    | ✓       | ✓         | ✓         | <b>0.978</b> | <b>0.98</b> | 4,100,759        | 8.8        | 8.8        |

From the results of the ablation experiments, the introduction of G-Backbone significantly reduced the number of parameters, computation, and model size of the network structure. At the same time, the average precision means value remained stable, verifying the effectiveness of the lightweight module. With the introduction of G-CA and W-BiFPN, although the number of parameters and computational effort increased slightly, the mean accuracy and recall rate increased, which verified the effectiveness of the improved module. With the introduction of EIOU Loss, the number of parameters, computation volume, and model size remained unchanged. However, the average precision means value increased slightly, and the recall rate increased by nearly 7%, verifying that the introduction of EIOU Loss outperforms CIOU Loss and improves the detection accuracy. With the introduction of Meta-ACON, although the number of parameters, computation, and model size increased slightly, the mean accuracy and recall rate increased, verifying the effectiveness of the improvement. By incorporating all the improved modules in this paper, the number of parameters, computation, and model size was reduced. The mean accuracy and recall rate increased by 2% and 7%, respectively, which combines the lightweight model and detection accuracy.

#### 4.6. Case Study

We further empirically investigated the detection performance in different scenarios. The detection results are shown in Figure 10, all based on real detection scenarios. The top row indicates the detection results of the YOLOv5s algorithm, and the bottom row indicates the detection results of the YOLOv5s-Z algorithm. The detection scenarios are a

tennis racket on the ground, a tennis court in the morning, a tennis court in the evening, and an aisle outside the laboratory. Overall, the detection accuracy of the YOLOv5s-Z algorithm was higher than that of the YOLOv5s algorithm, regardless of the different scenes or periods of the same scene.



Figure 10. Comparison chart of test results.

## 5. Conclusions

In order to adapt Robomaster EP for accurate detection and real-time recognition of tennis balls, this paper proposes the YOLOv5s-Z algorithm, constructs lightweight G-Backbone and G-Neck network layers, proposes a convolutional coordinate attention mechanism, and incorporates it into the Backbone feature extraction network, which enables the network to obtain location information of a larger area through multiple convolutions and further enhances feature extraction. The G-Neck network layer incorporates a weighted bi-directional feature pyramid W-BiFPN with settable learning weights to further enhance the feature fusion capability and achieve more efficient weighted feature fusion and bi-directional cross-scale connectivity. The Loss function, EIOU Loss, is introduced to split the influence factor of the aspect ratio based on the penalty term of CIoU Loss to calculate the length and width of the target and anchor frames, respectively. Meta-ACON's activation function is introduced to adaptively select whether to activate the neurons to improve the detection accuracy. Finally, the YOLOv5s-Z algorithm is deployed into Robomaster EP to achieve accurate detection and real-time recognition of tennis balls, verifying the effectiveness of the YOLOv5s-Z algorithm and the lightweight of the model, which has some practical significance and prospects in the field of tennis ball detection. Further optimization of the network model will be carried out in future work to optimize the network structure more comprehensively, to achieve mobile target detection, and to improve detection efficiency and accuracy.

**Author Contributions:** Writing—original draft preparation, Y.Z.; writing—review and editing, L.L.; Validation, W.Y.; Conceptualization, Q.L.; software X.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Severinson-Eklundh, K.; Green, A.; Hüttenrauch, H. Social and collaborative aspects of interaction with a service robot. *Robot. Auton. Syst.* **2003**, *42*, 223–234. [\[CrossRef\]](#)
2. Perera, D.M.; Menaka, G.; Surasinghe, W.; Madusanka, D.K.; Lalitharathne, T.D. Development of a Vision Aided Automated Ball Retrieving Robot for Tennis Training Sessions. In Proceedings of the 2019 14th Conference on Industrial and Information Systems (ICIIS), Kandy, Sri Lanka, 18–20 December 2019; pp. 378–383.
3. Foix, S.; Alenya, G.; Torras, C. Lock-in time-of-flight (ToF) cameras: A survey. *IEEE Sens. J.* **2011**, *11*, 1917–1926. [\[CrossRef\]](#)
4. Ren, L.; Wang, W.; Du, Z. A new fuzzy intelligent obstacle avoidance control strategy for wheeled mobile robot. In Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; pp. 1732–1737.
5. Schweiker, M.; Ampatzi, E.; Andargie, M.S.; Andersen, R.K.; Azar, E.; Barthelmes, V.M.; Berger, C.; Bourikas, L.; Carlucci, S.; Chinazzo, G.; et al. Review of multi-domain approaches to indoor environmental perception and behaviour. *BUILD. Environ.* **2020**, *176*, 106804. [\[CrossRef\]](#)
6. Illingworth, J.; Kittler, J. A survey of the Hough transform. *Comput. Vision Graph. Image Process.* **1988**, *44*, 87–116. [\[CrossRef\]](#)
7. 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 European Conference on Computer Vision; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
8. 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*. [\[CrossRef\]](#) [\[PubMed\]](#)
9. 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, Virtual, 11–17 October 2021; pp. 2778–2788.
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
11. Zeng, Q.; Li, X.; Lin, H. Concat Convolutional Neural Network for pulsar candidate selection. *Mon. Not. R. Astron. Soc.* **2020**, *494*, 3110–3119. [\[CrossRef\]](#)
12. 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. [\[CrossRef\]](#)
13. Ma, N.; Zhang, X.; Liu, M.; Sun, J. Activate or not: Learning customized activation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 8032–8042.
14. 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.
15. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Washington, DC, USA, 7–13 December 2015; pp. 1440–1448.
16. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
17. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
18. Tsalatsanis, A.; Valavanis, K.; Yalcin, A. Vision based target tracking and collision avoidance for mobile robots. *J. Intell. Robot. Syst.* **2007**, *48*, 285–304. [\[CrossRef\]](#)
19. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*; O’Reilly Media, Inc.: Sebastopol, CA, USA, 2008.
20. Gu, S.; Ding, L.; Yang, Y.; Chen, X. A new deep learning method based on AlexNet model and SSD model for tennis ball recognition. In Proceedings of the 2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA), Hiroshima, Japan, 11–12 November 2017; pp. 159–164.
21. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Van Esesn, B.C.; Awwal, A.A.S.; Asari, V.K. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv* **2018**, arXiv:1803.01164.
22. Gu, S.; Chen, X.; Zeng, W.; Wang, X. A deep learning tennis ball collection robot and the implementation on nvidia jetson tx1 board. In Proceedings of the 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Auckland, New Zealand, 9–12 July 2018; pp. 170–175.
23. Lindholm, E.; Nickolls, J.; Oberman, S.; Montrym, J. NVIDIA Tesla: A unified graphics and computing architecture. *IEEE Micro* **2008**, *28*, 39–55. [\[CrossRef\]](#)
24. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
25. He, X.; Zhao, K.; Chu, X. AutoML: A survey of the state-of-the-art. *Knowl.-Based Syst.* **2021**, *212*, 106622. [\[CrossRef\]](#)
26. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
27. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.

28. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
29. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589.
30. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
31. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
32. Hou, Q.; Zhou, D.; Feng, J. Coordinate attention for efficient mobile network design. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 13713–13722.
33. Han, J.; Moraga, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *Proceedings of the International Workshop on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 195–201.
34. 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.
35. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
36. 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.
37. Zheng, Z.; Wang, P.; Ren, D.; Liu, W.; Ye, R.; Hu, Q.; Zuo, W. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 12993–13000.
38. 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 Midtown, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.
39. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
40. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for activation functions. *arXiv* **2017**, arXiv:1710.05941.
41. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1997–2017.
42. Goodfellow, I.; Warde-Farley, D.; Mirza, M.; Courville, A.; Bengio, Y. Maxout networks. In Proceedings of the International Conference on Machine Learning, Atlanta, Georgia, USA, 17–19 June 2013; pp. 1319–1327.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.