Advances in Computer, Signals and Systems (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2023.070203
ISSN 2371-8838 Vol. 7 Num. 2

Lightweight Steel Bar Detection Network Based on YOLOv5

Ren Junsong, Wang Yi*, Peng Xutao

Department of Computer Science and Engineering, Sichuan University of Science & Engineering, Yibin, Sichuan, China
**Corresponding author*

Keywords: Rebar detection; YOLOv5 algorithm; weighted two-way feature pyramid; attention mechanism

Abstract: The proposed YOLOv5 model for steel bar detection has been improved with the addition of an ECA-Net attention module and Ghost Conv to reduce model volume. The Neck layer's feature pyramid module has been replaced with a weighted two-way feature pyramid network structure for better feature fusion. Additionally, the loss function and image processing have been improved for better detection efficiency. The experimental results on the reinforced data set show that the volume of the improved YOLO -EB model is reduced by 11 % compared with the original version, and the mAP is increased by 1.6 %, which meets the requirements of actual use.

1. Introduction

Whether in the production process of steel bars or the process of use, the statistics of the number of steel bars is a very important link. For example, at the construction site, for a truck with a large number of steel bars that enters the site, the acceptance personnel need to manually root the steel bars on the truck and confirm the specifications and quantities of the steel bars before the steel bar truck enters the site to unload.

To avoid the inevitable errors of manual counting, and save time and manpower, Traditional digital image processing is mainly based on the image recognition and counting of steel bars based on the characteristic information such as the grayscale and shape of the steel bar picture. The detection effect is unstable [1-3].

The success of deep learning technology in object detection tasks on images has led more and more researchers to apply deep learning technology to reinforcement technology tasks. In 2010, Chen Zhikun [4] and others studied the application of the BP neural network to steel bar counting, which proved the feasibility of this method. Compared with traditional algorithms, deep learning methods have higher detection accuracy and speed and are more robust. In 2020, Xie Haizhen [5] proposed a new target monitoring algorithm applied to steel bar counting. In 2021, based on the Faster R-CNN model, Wang Huifang proposed a deep learning-based reinforcement counting algorithm Rebar R-CNN [6].

Based on the YOLOv5 model, combined with the attention mechanism, Bi FPN, Ghost Net, α-IoU, and other technologies, and proposed a reprocessing method for detection pictures, this paper

constructs a lightweight steel target embedded in the attention mechanism The detection model greatly reduces the volume of the original model while improving the detection accuracy.

2. Basic theory

The YOLOv5 is a lightweight target detection algorithm that is implemented using the Python framework. The model consists of four versions, namely YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. These versions are built by deepening and widening the network based on the YOLOv5s, which is the shallowest and smallest feature map width among the four.The network model of YOLOv5 is mainly divided into 4 parts, including input, backbone network (Backbone), Neck module, and output. Its network structure is shown in Figure 1.

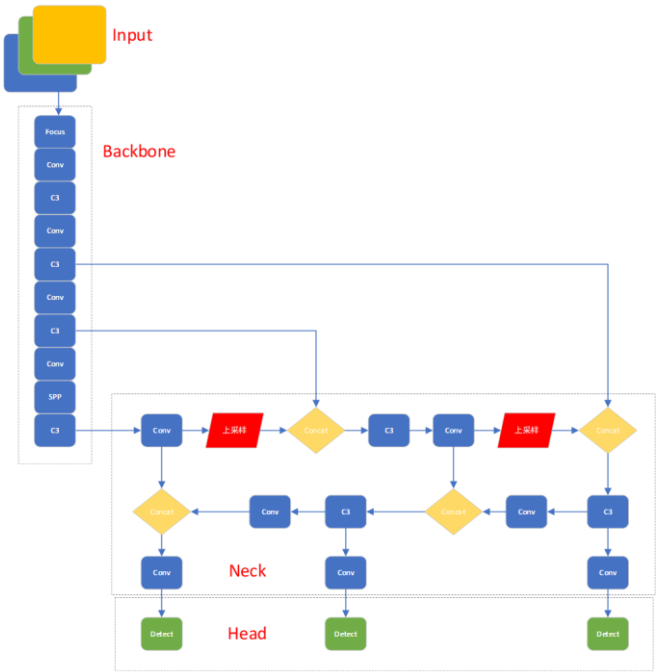

Figure 1 YOLOV5 network structure

2.1 Input

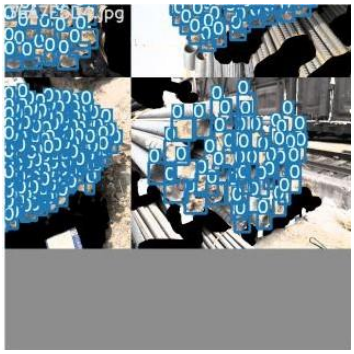

Figure 2 Mosaic effect diagram

Yolov5 takes in three main components for its input: mosaic data augmentation, adaptive anchor box calculation, and adaptive image scaling. During the model training phase, mosaic data augmentation is used to increase the number of small objects in the dataset, thereby enhancing the

model's ability to detect small objects. The result is shown in Figure 2. During model training, the network generates a predicted frame using the initial anchor frame, compares it with the actual frame, and updates the network parameters iteratively in reverse to improve its accuracy. Meanwhile, adaptive image scaling is used to resize the original image to a standardized size, allowing the model to detect objects consistently across different image sizes. Figure 2 is the result of the adaptive scaling of pictures.

2.2 Backbone

The original model's Backbone consists mainly of Conv, Focus, C3 modules, and a spatial pyramid pooling module [7]. The Conv module performs convolution, batch normalization, and activation function operations on the input feature map.

2.3 Neck module

This module, like the Neck structure in YOLO v4, mainly uses the feature pyramid structure [8] plus the composition of the Path Aggregation Network, which can strengthen the network for different scaling scales. The ability to fuse object features. In the model, the combination of operations involves the FPN layer, which transmits strong semantic features from top to bottom, and the PAN layer, which transmits strong positional features from bottom to top. The network also aggregates parameters from different backbone layers to different detection layers, enhancing the network's feature fusion capability. The structure of FPN and PANet is shown in Figure 3.

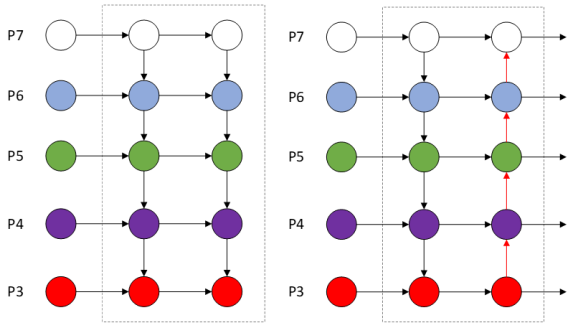

Figure 3 FPN, PANet structure

2.4 Output

The output of the YOLOv5 original model uses the GIOU [9] function as the loss function of the bounding box, adjusts the model parameters, and corrects the position of the predicted box. During target detection post-processing, using NMS [10] to filter multiple target frames, which enhances the detection ability of multiple targets and occluded targets.

3. Model improvement and optimization

3.1 Improvement of backbone network

3.1.1 Add attention mechanism module

The small and dense rebar sections in the picture occupy few pixels and are easily affected by the background, which can result in inaccurate detection. To address this issue, the new model includes an attention mechanism in the last layer of the backbone. This mechanism enables the model to assign

varying weights to different parts of the input, extract more critical information, and make more precise judgments.

Although adding the attention mechanism increases the computational overhead and storage requirements of the model, it can effectively extract feature information for small and dense objects, thus improving detection accuracy.

The ECA [11] attention module (Efficient Channel Attention) mainly starts from the two aspects of dimension reduction and cross-channel information interaction in the SE [12] module. It is inefficient and unnecessary to capture the relationship of all channels, and the latter plays a very important role in improving the performance of the CNN network. The ECA module realizes an optimization improvement of the SE module. The ECA module structure is shown in Figure 4.

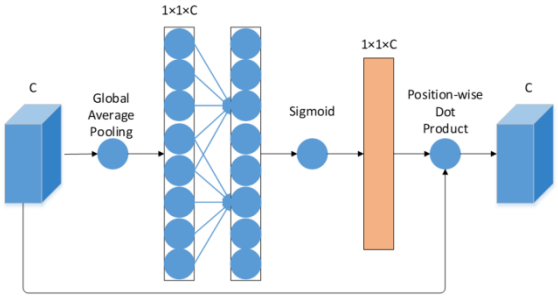

Figure 4 ECA module structure with a k value of 5

The SE attention mechanism compresses the input feature map's channels, which can hinder the learning of dependencies between them. However, the ECA attention mechanism uses 1-dimensional convolution to facilitate local cross-channel interaction and extract dependencies between channels without reducing the dimensionality of the input. This approach avoids channel compression and improves the model's ability to learn inter-channel relationships. Therefore, for the y_i corresponding weights w_1^1, w_1^2, w_1^3... w_1^k, the author only considers the information exchange between the current channel and its k neighboring channels, and the number of parameters is k×C, so the weight matrix is calculated as follows (1) :

$$\omega_i=\sigma\left(\sum_{j=1}^{k} w_i^j y_i^j\right),\ y_i^j \in \Omega_i^k \tag{1}$$

The k domain channels σ represented y_i therein Ω_i^k are Sigmoid functions.

k is determined by the formula (2), the channel dimension C is proportional to the convolution kernel size k, $C=2^{(\gamma*k-b)}$, $|t|_{odd}$ represents the odd number closest to t, b takes 1, γ takes 2.

$$k=\psi(c)=\left|\frac{\log_2(c)}{\gamma}+\frac{b}{\gamma}\right|_{odd} \tag{2}$$

E CA module is as follows:

(1) Perform a global average pooling operation on the input feature map;

(2) The weight of each channel is obtained by performing a 1D convolution operation with a kernel size of k and passing the output through the Sigmoid activation function ω;

(3) The final output feature map is obtained by multiplying the weights with the corresponding elements of the original input feature map.

It can be seen that the idea and operation of the ECA attention mechanism are extremely simple

and have little impact on the network processing speed, so this paper chooses ECA as the attention module.

3.1.2 Replace the Conv module in the original network

The Conv module mainly performs convolution on the input feature map, batch normalization, and activation function operation. The structure is shown in Figure 5.

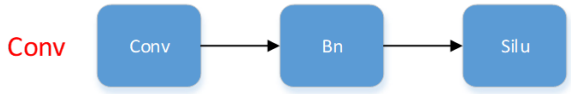

Figure 5 Conv module structure

Ghost Conv [13] module is shown in Figure 6. The original one-step convolution is changed to two-step convolution. The first step is to perform conventional convolution, but the number of output channels is reduced. The second step is in Depth separable convolution is performed on a one-step basis. It is worth noting that the number of output channels of depth separable convolution may be an integer multiple of the number of input channels, and the number of convolution kernels is equal to the number of output channels. In addition, the second step of convolution has a parallel connection branch, which is directly the output of the first step of convolution. Ghost Conv extracts rich feature information through conventional convolution operations, and uses connection branch operations to generate redundant feature information, which can not only effectively reduce the computing resources required by the model, but also make the overall model smaller Simple, and easy to implement.

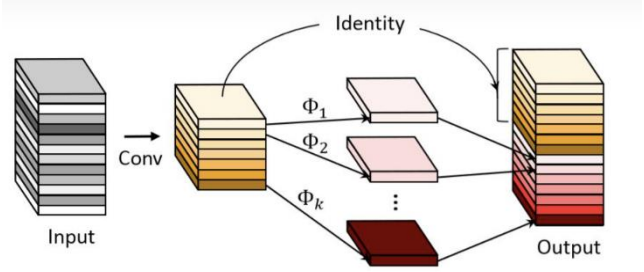

Figure 6 GhostConv module structure

3.2 Improvement of feature pyramid structure

BiFPN [14]is a novel feature fusion method that was proposed by Google in 2020. The method employs efficient bi-directional cross-scale connections and weighted feature fusion to combine features. It first performs top-down feature fusion and then bottom-up feature fusion. The paper proposes several optimization methods for cross-scale connections to enhance the efficiency of the model.One of these optimization methods is based on the idea that if a node has only one input edge without feature fusion, its contribution to the feature network that fuses different features will be smaller. Therefore, BiFPN simplifies the bidirectional network by removing the intermediate nodes of P3 and P7 in PANet.Another optimization method that BiFPN uses is to add spanning connections to connect input nodes of the same scale to output nodes. Since they are in the same layer, this method incorporates more features without adding too much computational cost. Lastly, in contrast to the single top-down and bottom-up paths of PANet, BiFPN considers each bidirectional path as a feature network layer and repeats the same layer multiple times to achieve higher-level feature fusion. The BiFPN network structure is shown in the figure 7.

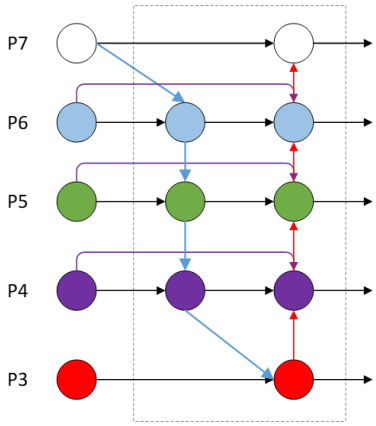

Figure 7 BiFPN module structure

When fusing features with different resolutions, a common approach is to resize them to the same resolution and then fuse them. However, this approach does not account for the unequal contributions of different input features. To address this issue, He Kaiming suggests adding a weight to each input and letting the network learn the importance of each input feature. Since scalar weights are unbounded, it can lead to unstable training. Therefore, BiFPN uses weight normalization to constrain the value range of each weight. After comparing three different methods, BiFPN finally uses Fast normalized fusion. The formula is as follows (3) :

$$O = \sum_i \frac{\omega_i}{\varepsilon + \sum_j \omega_j} \cdot I_i \tag{3}$$

ω_i and ω_j are learnable weights, ε (epsilon) = 0.0001 is a very small number to ensure that the denominator is not 0, I_i representing the characteristics of the input. Among them , ω_i >0 , after passing Relu to ensure the stability of the value, the value of each normalized weight is also between 0 and 1.

In the original article, the author introduced the situation of two fusion features in the sixth layer, such as formulas (4), (5):

$$P_6^{td} = Conv\left(\frac{\omega_1 \cdot P_6^{in} + \omega_1 \cdot \mathrm{Re}\,size(P_7^{in})}{\omega_1 + \omega_2 + \varepsilon}\right) \tag{4}$$

$$P_6^{out} = Conv\left(\frac{\omega_1' \cdot P_6^{in} + \omega_2' \cdot P_6^{td} + \omega_3' \cdot \mathrm{Re}\,size(P_5^{out})}{\omega_1' + \omega_2' + \omega_3' + \varepsilon}\right) \tag{5}$$

P_6^{td} It is the top-down intermediate feature, P_6^{out} the bottom-up output feature, P_7^{in} the input feature of the 7th layer, and P_5^{out} the output feature of the 5th layer, Resize is upsampling or downsampling, and Conv is a convolution operation. To further improve efficiency, BiFPN uses depthwise separable convolutions for feature fusion and adds batch normalization and activation functions after each convolution.

3.3 Improvement of loss function

YOLOv5's bounding box regression (bounding box, bbox) uses GI o U as the loss function, GI o U The Loss formula is shown in (6) below.

$$L_{(GIoU)} = 1 - IoU + \frac{|C \setminus (A \cup B)|}{|C|} \tag{6}$$

In the formula, C is for any two boxes A and B, and C is a minimum box that can enclose them. However, GIoU also has its disadvantages: If two prediction boxes have the same height and width and are on the same horizontal plane, GIoU degenerates into IoU. However, both GIoU and IoU suffer from two disadvantages: slower convergence and less accurate regression.So this paper chooses α-IoU [16] as the loss function of bbox regression.

α- IoU is a simple transformation based on Generalized IoU, Distance IoU [17], and Complete IoU, by adaptively reweighting the loss and gradient of high and low IoU targets, improving bbox regression accuracy. According to the experiments conducted by the author of α-IoU, α - IoU loss exceeds the other types of losses mentioned above, especially in the case of high bbox regression accuracy mAP75:95, the advantage of α-IoU loss in high precision level is more obvious. Moreover, α can be adjusted to give the detector more flexibility in achieving different levels of bbox regression accuracy.

The original Io U calculation formulas are as follows (7), (8), (9), (10):

$$L_{(IoU)} = 1 - IoU \tag{7}$$

$$L_{(GIoU)} = 1 - IoU + \frac{|C \setminus (B \cup B^{gt})|}{|C|} \tag{8}$$

$$L_{(DIoU)} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{C^2} \tag{9}$$

$$L_{(CIoU)} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{C^2} + \beta v \tag{10}$$

The improved Io U calculation formulas are as follows (11), (12), (13), (14) :

$$L_{(\alpha-IoU)} = 1 - IoU^{\alpha} \tag{11}$$

$$L_{(\alpha-GIoU)} = 1 - IoU^{\alpha} + \left(\frac{|C \setminus (B \cup B^{gt})|}{|C|} \right)^{\alpha} \tag{12}$$

$$L_{(\alpha-DIoU)} = 1 - IoU^{\alpha} + \frac{\rho^{2\alpha}(b, b^{gt})}{C^{2\alpha}} \tag{13}$$

$$L_{(\alpha-CIoU)} = 1 - IoU^{\alpha} + \frac{\rho^{2\alpha}(b, b^{gt})}{C^{2\alpha}} + (\beta v)^{\alpha} \tag{14}$$

3.4 Improved YOLOv5 network structure

The YOLOv5 algorithm itself has good engineering applicability, and users can make improvements according to their own needs based on it. According to the theoretical analysis and research in the previous chapters, a new improved YOLOv5 model YOLO-EB (YOLO-ECA-BiFPN) can be obtained, and its model structure is shown in Table 1 below. In the table, the column "From" indicates the layer where the input comes from. A value of -1 indicates the input is from the previous layer. The "Params" column indicates the size of the parameters, while the "Module" column indicates the type of module used. The "Arguments" column provides information about the module parameters, such as the number of input and output channels, the size of the convolution kernel, stride information,

and so on.

Table 1 YOLO-EB network structure

Num	from	params	Module	arguments
0	-1	928	Conv	[3, 32, 3]
1	-1	10144	GhostConv	[32, 64, 3, 2]
2	-1	18816	C3	[64, 64, 1]
3	-1	38720	GhostConv	[64, 128, 3, 2]
4	-1	156928	C3	[128, 128, 3]
5	-1	151168	GhostConv	[128, 256, 3, 2]
6	-1	625152	C3	[256, 256, 3]
7	-1	597248	GhostConv	[256, 512, 3, 2]
8	-1	118272	C3	[512, 512, 1]
9	-1	261	Eca	[512, 512]
10	-1	656896	SPPF	[512, 512, 5]
11	-1	69248	GhostConv	[512, 256, 1, 1]
12	-1	0	Upsample	[None, 2, 'nearest']
13	-1	0	Concat	[1]
14	-1	361984	C3	[512, 256, 1, False]
15	-1	18240	GhostConv	[256, 128, 1, 1]
16	-1	0	Upsample	[None, 2, 'nearest']
17	-1	0	Concat	[1]
18	-1	90880	C3	[256, 128, 1, False]
19	-1	75584	GhostConv	[128, 128, 3, 2]
20	-1	0	Concat	[1]
21	-1	361984	C3	[512, 256, 1, False]
22	-1	590336	Conv	[256, 256,3, 2]
23	-1	0	Concat	[1]
24	-1	1182720	C3	[512, 512, 1, False]

4. Experimental image processing

4.1 Input pre-processing

After many experiments, it can be proved that image slicing processing of high-definition steel bar images before detection can improve model training speed and detection accuracy. Figures 8 and 9 are the comparison charts of the box_loss curve and the Map_0.5 curve in a certain test respectively.

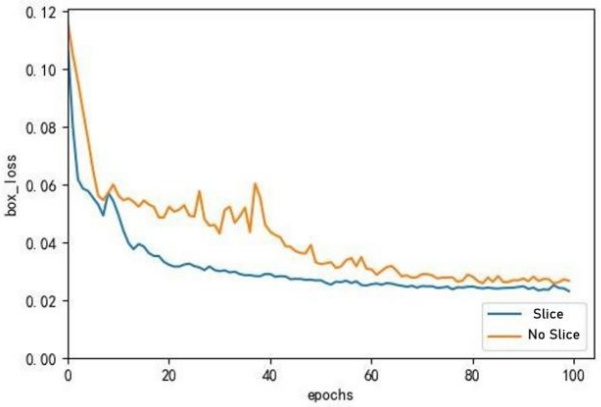

Figure 8 Box_loss comparison chart

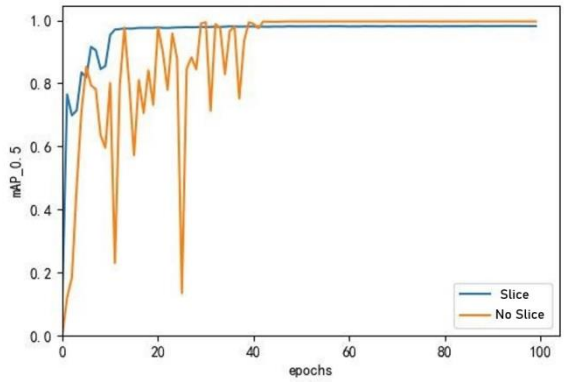

Figure 9 MaP_0.5 comparison chart

4.2 Image processing after detection

After splicing multiple sliced images, there will be a large number of overlapping detection frames on the picture, so we will perform NMS again here to delete the redundant prediction frames, leaving the prediction frames with high confidence.

We prioritize the prediction boxes with high confidence, and then make judgments based on whether the intersecting area between the selected prediction boxes and other prediction boxes is less than a threshold (we set it to 0.3 in the test), and if it is less than, it is considered a correct one. prediction box, otherwise deletes the prediction box. In Figure 10, the center point is used instead of the prediction box for easy viewing.

Figure 10 The renderings after NMS operation

5. Experiments

5.1 Experimental environment

The version of the computer operating system used in this experiment is ubuntu 18.04, and the CPU model is Intel(R) Platinum 8176 CPU @ 2.1 0GHz ×1 12. The GPU model is two NVIDIA 2 080 ti, the memory size of a single graphics card is 11 GB, and the memory size is 187.5 GB. The YOLOv5 model is based on the Pytorch deep learning framework, the programming language is Python, and the torch version 1.7.1 and CUDA1 0.1 are used to accelerate the GPU. The parameter settings are shown in Table 2 below.

Table 2 parameter settings

parameter name	parameter value
Momentum	0.937
Weight _decacy (weight decay)	0.0005 _
Batch size (batch size)	12
Learning_rate (learning rate)	0.01
Epochs (iteration rounds)	100

5.2 Datasets

The data set in this paper comes from the "Intelligent Inventory - Reinforcement Number AI Recognition Contest". The division and use of the data set are shown in Table 3. It can be seen from Table 3 that there are only 250 pictures in the Train dataset. To improve the robustness of the model, this paper uses data enhancement methods such as mosaic, flipping, and contrast enhancement to expand the dataset during the training process. The way of data enhancement is shown in Figure 10. According to the ratio of 8:2, the data set is divided into training set and a test set. The training set has 200 pictures, and the test set has 50 pictures. There is only steel bar detection objects in the data set. As shown in Figure 11, the analysis and visualization results of the data set are shown. (a) is the distribution map of the center point of the object, and the horizontal and vertical coordinates represent the position of the center point. (b) is the distribution map of the object size.

Table 3 Dataset division and usage

data set	Number of pictures/sheet	use
train	200	model training
Val	50	m AP calculation
test	200	Used to view the actual effect of the model

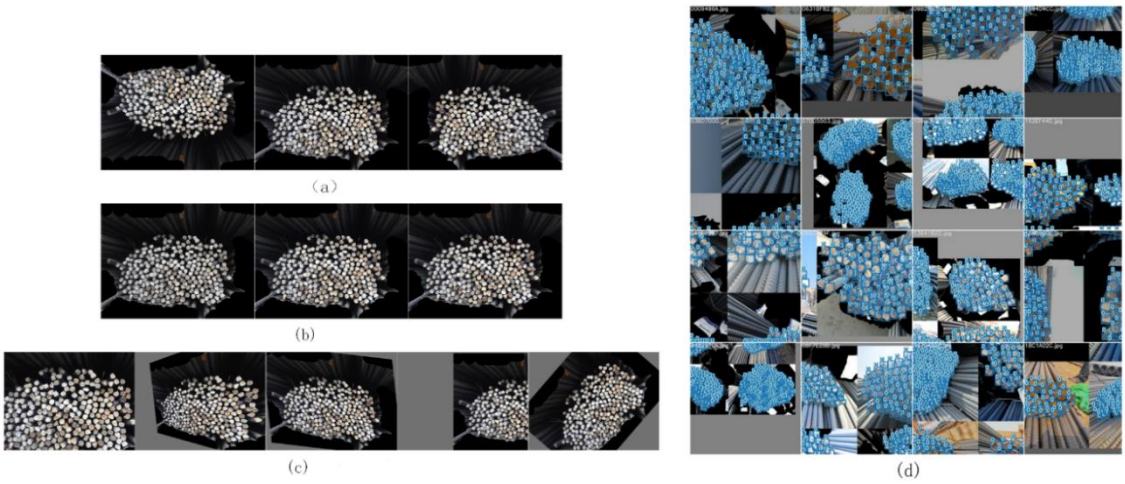

Figure 10 Data_augmentation (a,b,c,d are to flip the picture, change the contrast, rotate, change the display range and combination)

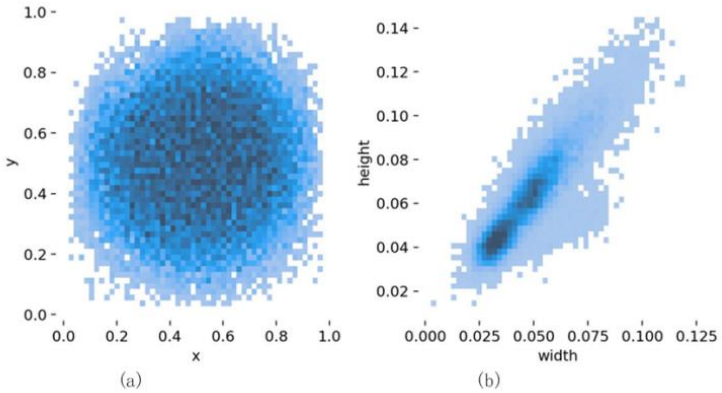

Figure 11 Data set analysis (Figure a shows the target center point distribution, Figure b shows the target size distribution)

5.3 Model evaluation index

MAP is a commonly used evaluation index in the target detection model. The AP (Average Precision) of each category is obtained by calculating the area under the P-R curve. Its full English name is (Mean Average Precision), that is, the average of the average precision. The formula is shown in (15).

Precision measures how accurate the model prediction is, that is, the percentage of correct predictions, and Recall measures the ability of the model to find all positive examples.

$$mAP = \left(\sum_{i=1}^{n} AP_i \right) / n \tag{15}$$

In the formula, n is the number of categories in the data set, because there is only one category for steel bar detection, so n= 1, m AP =AP. Precision is calculated as follows (16) as P, and Recall as follows (17) as R.

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\_\det ection\_boxes} \tag{16}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\_GroundTrue\_boxes} \tag{17}$$

In the formula, TP is a true positive, FP is a false positive, TN is a true negative, and FN is a false negative.

Take Recall as the horizontal axis and Precision as the vertical axis, and draw the connection line for each value to get the PR graph. As Recall increases, Precision will gradually decrease and fluctuate around a certain value. AP is the average precision. The integral method is used to calculate the area enclosed by the PR curve and the coordinate axis. The actual operation does not calculate the integral but performs a smoothing operation on it to simplify the calculation. For each point on the PR curve, Precision takes the Click the largest value on the right, and after calculating the AP, adds and averages the APs of all categories to get the mAP on the entire data set.

5.4 Analysis of experimental results

In this paper, the models trained by different networks are used for Val data set and Test data set detection, and the model weight and mAP index in the statistical detection process are counted. The comparison charts of the loss curves of the two models are shown in Figures 12 and 13. It can be seen that the YOLO -EB model has a faster convergence speed and a smaller loss value than the original model.

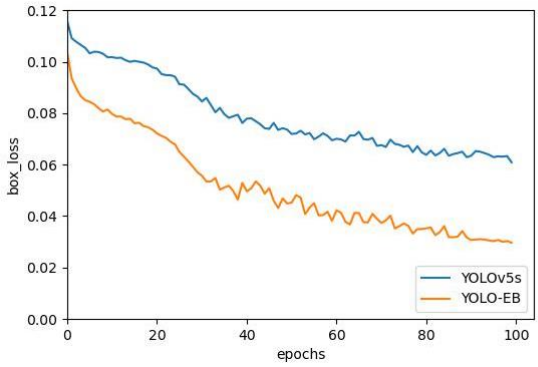

Figure 12 Box_loss graph

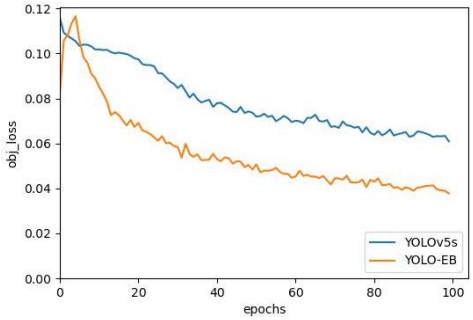

Figure 13 Obj_loss graph

From Table 4 that after the same training times, the YOLOv5s model in the YOLOv5 detection algorithm is a lightweight network model compared with the two-stage Faster-RCNN and the one-stage YOLOv3. The YOLOv5-EB model is in The basis of the YOLOv5s model is reduced by 1.58 MB, and the average precision is also increased by 0.01. YOLOv 3-SPP is comparable to YOLOv5-CB in average accuracy, but its weight is about 3.4 times the weight of YOLOv5-CB model, which is not conducive to the deployment and use in work scenes.

Table 4 Comparison of experimental results

Model	weight /MB	mAP
Faster-RCNN （ResNet50）	330	0.715
YOLOv3-SPP	123.4	0.973
YOLOV5M	42.5	0.970
YOLOV5S	14.08	0.965
YOLO-EB	12.5	0.975

5.5 Ablation experiment

The ablation comparison experiment is to verify the optimization effect of each improved module. The experimental results are shown in Table 5. The improvements represent adding the ECA attention mechanism to the backbone network, modifying the pyramid structure of the original network, modifying the loss function, and the conv structure. Modifications. As can be seen from the data in the table, adding the attention mechanism, modifying the pyramid structure, and modifying the loss function can improve the average accuracy, but the model size is increasing. After modifying the conv structure, although the average accuracy has decreased, the model size has decreased. 15%. After adding all the improvements to the original model, not only the model size was reduced by about 1.1 %, but the average accuracy increased by about 1.6 %.

Table 5 Ablation experiments results

model	Weight / MB	Join the E CA module	Modifying the Feature Pyramid	Modify the loss function	Modify the conv structure	mAP
YOLOV5S	1 4.08	x	x	x	x	0.960 _
model 1	1 4.7	√	x	x	x	0.966 _
model 2	1 4.5	x	√	x	x	0.965 _
model 3	1 4.4	x	x	√	x	0.967 _
model 4	12	x	x	x	√	0.957 _
YOLO-EB	1 2.5	√	√	√	√	0.975 _

5.6 Analysis of model testing results

To determine the actual detection effect of the new model, select some pictures in the test set for testing. Figure 14 is a comparison of the test results of the two models of YOLO-EB and YOLOv5. It can be seen in a, b, and d that the YOLO-EB model can detect the steel bar section that the YOLOv5 model cannot detect. In c and e, the YOLO-EB model can avoid the identification of interferers, while YOLOv5 is misidentified, indicating that the YOLO-EB model shows a stronger performance than YOLO v5.

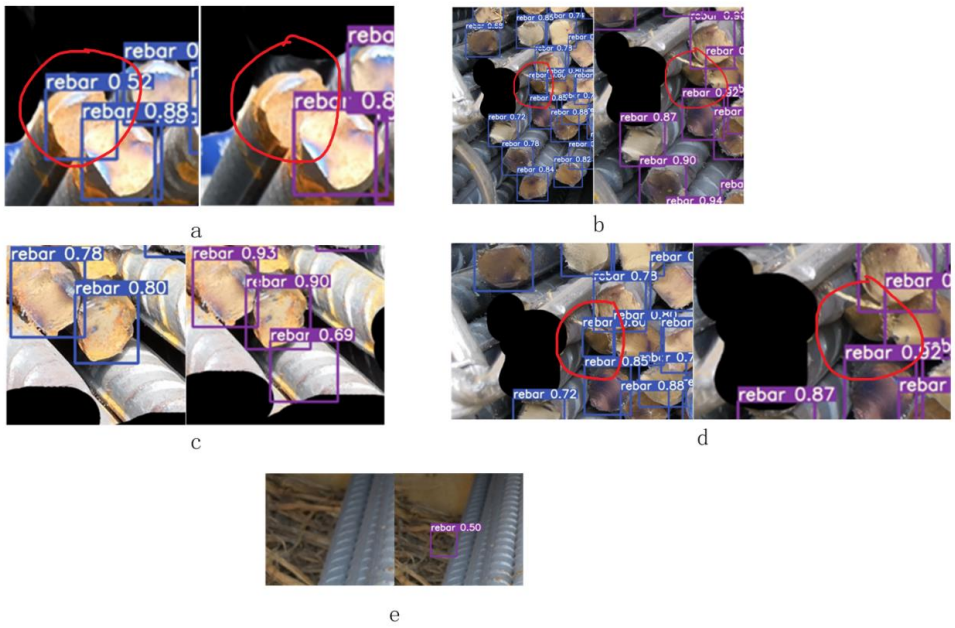

Figure 14 Test result comparison chart (The blue prediction box is the YOLO-EB model, and the purple is the YOLOv5s model)

6. Conclusions

This paper builds a lightweight steel target detection model by combining YOLOv5 with various technologies such as attention mechanism, BiFPN, Ghost Net, and α-IoU. The model is embedded with an attention mechanism that greatly improves the detection accuracy while reducing the model's size. This makes it suitable for deployment in steel factories or construction sites.

However, the model still faces challenges in recognizing pictures with mutual shadowing of steel bars and blurred boundaries between the background and the steel bar section, resulting in misrecognition and missing recognition.

In the next phase of research, the model's structure will be further improved to increase its robustness and recognition accuracy.

Acknowledgments

This work was supported by Tourism Information Mining and Visualization Research Based on Self-Media (ZHZJ19-01).

References

[1] Ghazali MF, Wong LK, See J. Automatic detection and counting of circular and rectangular steel bars[C]//9th

International Conference on Robotic, Vision, Signal Processing and Power Applications. Springer, Singapore, 2017: 199-207. 2017: 199-207.
[2] Niu Caihong. Research and Implementation of Rebar End Image Recognition [D]. Shenyang University of Technology, 2015.
[3] Li Qiang, Chen Zunde. Automatic steel quantity counting system based on fuzzy circular template matching method [J]. Automation Technology and Application, 2004, 23(2): 25-28.
[4] Chen Zhikun, Pan Xiaodi, Wang Fubin, et al. Research on Rebar Counting Method Based on Neural Network [J]. Sensors and Microsystems, 2010(8):4.
[5] Xie Haizhen. Research and application of bar counting algorithm in complex scenes [D]. University of Electronic Science and Technology of China, 2020
[6] Wang Huifang. Research and Application of Object Detection Algorithms for Dense Scenes [D]. University of Electronic Science and Technology of China, 2021.
[7] He K, Zhang X, Ren S, et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, 37(9):1904-16.
[8] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
[9] H Rezatofighi, Tsoi N, JY Gwak, et al. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019.
[10] Neubeck A, Gool L . Efficient Non-Maximum Suppression[C]// International Conference on Pattern Recognition. IEEE Computer Society, 2006.
[11] Wang Q, Wu B, Zhu P, et al. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks[C]// 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
[12] Jie H, Li S, Gang S, et al. Squeeze-and-Excitation Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, PP(99).
[13] Han K, Wang Y, Tian Q, et al. GhostNet: More Features From Cheap Operations[C]// 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
[14] Tan M, Pang R, Le Q V . EfficientDet: Scalable and Efficient Object Detection[C]// 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
[15] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759-8768.
[16] He J, Erfani S, Ma X, et al. $\alpha $- IoU : A Family of Power Intersection over Union Losses for Bounding Box Regression[J]. Advances in Neural Information Processing Systems, 2021, 34: 20230 -20242.
[17] Zheng Z, Wang P, Liu W, et al. Distance- IoU loss: Faster and better learning for bounding box regression [C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(07): 12993 -13000.