

Lightweight Safety Helmet Detection Algorithm Based on Improved YOLO11

Hao Su*

College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, China

* Corresponding author: Hao Su

Abstract: To enable the deployment of safety helmet detection on edge computing devices with limited resources, we propose a lightweight safety helmet detection algorithm, which is a safety helmet detection method based on YOLOv11n. Firstly, this method replaces the backbone network of YOLOv11n with the StarNet model to reduce the number of model parameters while extracting high-dimensional implicit features. Secondly, it adopts a multi-branch and multi-scale fusion module as the neck network (MBS-FPN). Through two bidirectional paths as the feature fusion network layer, and by repeating the same layer multiple times, higher-level feature fusion is achieved. In addition, shared convolution is introduced into the default detection head to reduce the number of model parameters and computational load, enhance the model's learning ability in multi-scale features, and improve the model's generalization ability in different backgrounds. Finally, the proposed lightweight safety helmet detection algorithm is verified on the public safety helmet dataset SHWD. The results show that compared with the original algorithm, the mAP50 and R are increased by 0.4% and 0.1% respectively, the number of parameters is reduced by 49.6%, and GFLOPs are decreased by 36.4%. In addition, by comparing the visual detection results of various algorithms, it is further proved that the lightweight safety helmet detection algorithm can provide a certain technical basis for the deployment of safety helmet detection.

Keywords: Deep learning; Safety helmet detection; YOLO.

1. Introduction

Safety helmet detection algorithms are mainly applied in scenarios with certain risk coefficients[1], such as construction sites, industrial production workshops, power industry operation sites, mining and tunnel projects, etc. In some of these scenarios, the deployment devices are AI edge computing terminals[2]. Edge computing terminals can be flexibly deployed near physical devices according to specific scenario requirements, such as beside machines in factory workshops, in monitoring rooms of construction sites, or in underground AI smart cameras. This distributed deployment mode enables them to more directly perceive changes in the physical environment[3]. Meanwhile, the miniaturized design of edge terminals makes them easy to install and maintain, allowing them to adapt to complex and scattered application scenarios. However, edge computing terminals are limited by hardware costs, power consumption, and installation space, resulting in relatively limited computing power, storage, and energy supply. For example, the hardware configurations of Nvidia's Jetson series and Huawei's Ascend series edge computing devices are far lower than those of cloud servers. Therefore, deploying safety helmet detection algorithms on

edge terminals requires lightweight improvements to the model. Model lightweighting is one of the key technologies in deep learning engineering. On the premise of minimizing the loss of model performance, it reduces the model's computational complexity[4, 5], decreases the number of model parameters, and compresses the model size through a series of technical means to meet the needs of deployment resources. Among them, designing lightweight neural network structures is an important direction in these technical means.

2. Lightweight Safety Helmet Detection Algorithm

This chapter adopts YOLO11 as the baseline model. Firstly, StarNet is used as the backbone network of the model. Secondly, MBS-FPN (Multi-Branch & Scale - FPN) is employed to fuse the features extracted from the backbone network, and LS-Detect (Lightweight Shared Conv Detect) is adopted as the detection head of the model. Finally, a series of related experiments are conducted on the SHWD dataset, which proves that the model in this chapter has achieved lightweight improvement while ensuring accuracy.

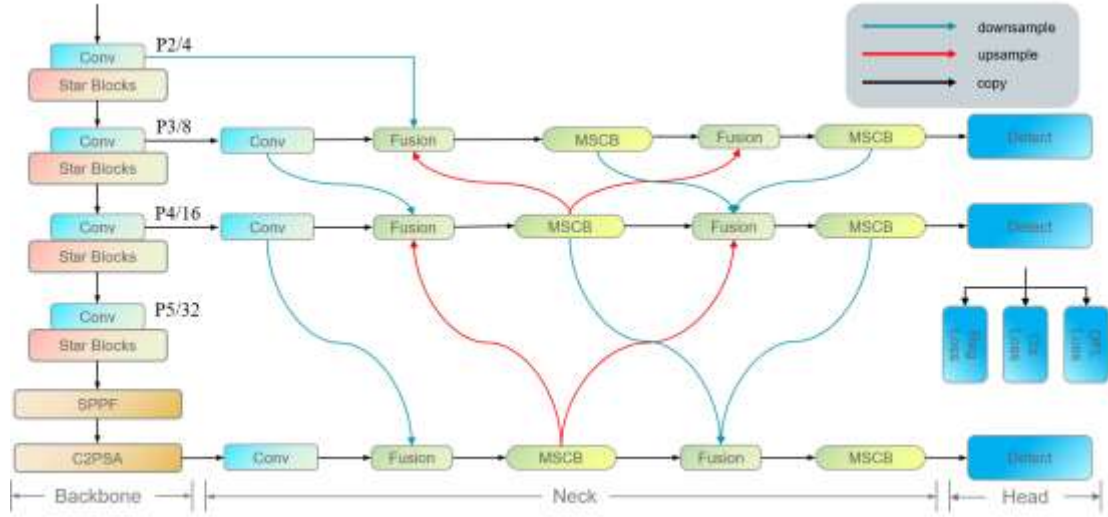


Figure 1: Overall Network Architecture Diagram

2.1. StarNet Backbone Network

In the backbone network of the YOLOv11 model, the C3k2 module can effectively extract image features. However, the stacked Bottleneck layers in the C3k2 module use a large number of convolutional layers, which results in a large number of parameters in the backbone network. Therefore, this paper replaces the backbone network of YOLOv11 with the lightweight StarNet network. The StarNet model conducts in-depth research on the Star operation and finds that this

operation can map inputs to a high-dimensional nonlinear feature space through element-wise multiplication. While maintaining a compact structure, this operation can significantly improve model performance and reduce latency. Compared with traditional summation operations, the Star operation can significantly improve model accuracy in networks with different widths and depths. This indicates that the Star operation has advantages in designing efficient neural networks, making it particularly suitable for resource-constrained application scenarios.

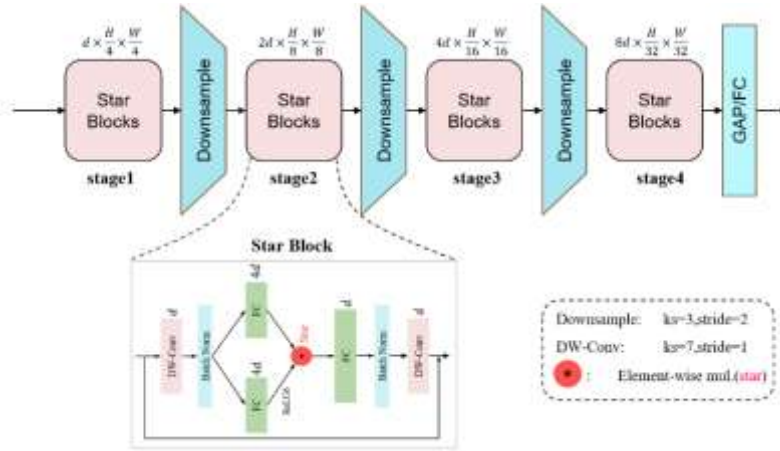


Figure 2: StarNet Network Architecture

The StarNet model structure is a 4-stage hierarchical architecture that utilizes convolutional layers for downsampling and modified demonstration blocks for feature extraction. To meet efficiency requirements, StarNet places a Batch Normalization (BN) layer after the Depthwise Convolution (DWConv) layer, and these two layers can be fused during the inference process. Drawing inspiration from MobileNeXt [66], StarNet incorporates a DWConv at the end of each block. The channel expansion factor is uniformly set to 4, and the network width is doubled at each stage. Additionally, StarNet adopts the ReLU6 activation function

from the MobileNetV2 [46] network. The framework of StarNet is illustrated in Figure 3.

StarNet re-examines the Star operation. When applying the Star operation described in the equation within a single-layer network, StarNet extends it to a combination of $\frac{(d+2)(d+1)}{2}$ distinct terms, as shown in the equation. It is worth noting that except for $\alpha_{(d+1)}x^{d+1}x$, all other terms are nonlinearly related to x , indicating that they represent independent implicit dimensions. Therefore,

$$\begin{aligned}
& w_1^T x * w_2^T x \\
&= \left(\sum_{i=1}^{d+1} w_1^i x^i \right) * \left(\sum_{j=1}^{d+1} w_2^j x^j \right) \\
&= \sum_{i=1}^{d+1} \sum_{j=1}^{d+1} w_1^i w_2^j x^i x^j \\
&= \underbrace{\alpha_{(1,1)} x^1 x^1 + \dots + \alpha_{(4,5)} x^4 x^5 + \dots + \alpha_{(d+1,d+1)} x^{d+1} x^{d+1}}_{(d+2)(d+1)/2 \text{ items}}
\end{aligned} \tag{1}$$

Among them, i, j are used to index the channels, and α represents the coefficient of each term:

$$\alpha_{(i,j)} = \begin{cases} w_1^i w_2^j & \text{if } i = j, \\ w_1^i w_2^j + w_1^j w_2^i & \text{if } i \neq j. \end{cases} \tag{2}$$

Star Blocks utilize computationally efficient star operations to perform computations in the d -dimensional space, achieving representation in an implicit dimensional feature space of $\frac{(d+2)(d+1)}{2} \approx \left(\frac{d}{\sqrt{2}}\right)^2$. This significantly expands the feature dimensionality without incurring any additional computational overhead in a single layer.

StarNet, by stacking multi-layer network structures, demonstrates that implicit dimensions can be exponentially

increased to nearly infinite sizes in a recursive manner. Considering an initial network layer with width d , the application of a star operation generates the expression $\sum_{i=1}^{d+1} \sum_{j=1}^{d+1} w_1^i w_2^j x^i x^j$, as detailed in the equation. This results in a representation of the implicit feature space in $\mathbb{R}^{\left(\frac{d}{\sqrt{2}}\right)^2}$. Denoting the output of the l -th layer Star operation as O_l , we can derive:

$$\begin{aligned}
O_1 &= \sum_{i=1}^{d+1} \sum_{j=1}^{d+1} w_{(1,1)}^i w_{(1,2)}^j x^i x^j & \partial \mathbb{R}^{\left(\frac{d}{\sqrt{2}}\right)^1} \\
O_2 &= W_{2,1}^T O_1 * W_{2,2}^T O_1 & \partial \mathbb{R}^{\left(\frac{d}{\sqrt{2}}\right)^2} \\
O_3 &= W_{3,1}^T O_2 * W_{3,2}^T O_2 & \partial \mathbb{R}^{\left(\frac{d}{\sqrt{2}}\right)^3} \\
&\vdots & \\
O_l &= W_{l,1}^T O_{l-1} * W_{l,2}^T O_{l-1} & \partial \mathbb{R}^{\left(\frac{d}{\sqrt{2}}\right)^{2^l}}
\end{aligned} \tag{3}$$

2.2. MBS-FPN Feature Fusion Module

The PAFPN used in the neck network of YOLOv11 adopts a dual-path approach to enhance feature integration, improving accuracy while controlling computational costs. However, PAFPN still has two significant limitations. Firstly, it tends to merge feature maps of the same scale and lacks comprehensive processing and fusion of multi-scale information from layers of different resolutions. For example, in Block1 of PAFPN, the input consists of the upsampled P5 layer and the peer P4 layer, which ignores the importance of shallow low-level spatial information in the P3 layer.

Similarly, in Block2, there is no direct fusion of the P2 layer that contains key information about small targets. This defect also exists in the last two modules. Secondly, the architecture's strategy for small target detection layers is formulated through a single top-down path and two modules, which severely weakens the model's ability to learn and represent features of tiny targets. In safety helmet detection tasks, however, targets vary significantly in size across different scenarios and may overlap. This makes it crucial for the model to leverage feature maps of different resolutions for feature fusion.

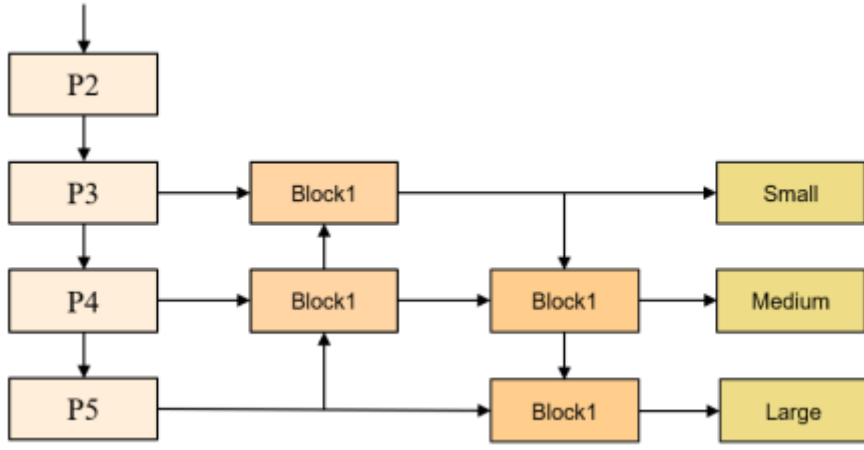


Figure 3: PAFPN Network Architecture



(a) Small



(b) Occlusion



(c) Large

Figure 4: SHWD Datasets

Therefore, this paper proposes a Multi-Branch and Multi-Scale Fusion Module (MBS-FPN) to effectively combine high-dimensional information from large-scale feature maps with fine-grained information from small-scale feature maps. In MBS-FPN, the input image initially passes through the backbone, which consists of four stages: P2, P3, P4, and P5. We design MAFPN as the neck structure, adopting two bidirectional (top-down and bottom-up) paths as feature fusion network layers, and repeating the same layer multiple times to achieve higher-level feature fusion. Among them, the MSCB module is used to extract multi-scale features from feature maps and perform preliminary auxiliary fusion. Meanwhile, Fusion is utilized to fuse features at different resolutions in a weighted manner, ultimately guiding the head to obtain diversified output information across three

resolutions.

When the Fusion module fuses features of different resolutions, since different input features are at different resolutions, their contributions to the output features are usually not equal. To address this issue, we add an additional weight to each input in the Fusion module to learn the importance of each input feature. Its formula is $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$, where to ensure $w_i \geq 0$, a ReLU activation function is applied after w_i , and $\epsilon = 0.0001$ is added to avoid numerical instability. The value of each normalized weight is between 0 and 1, and this fast fusion method has very similar learning behavior and accuracy to softmax-based fusion.

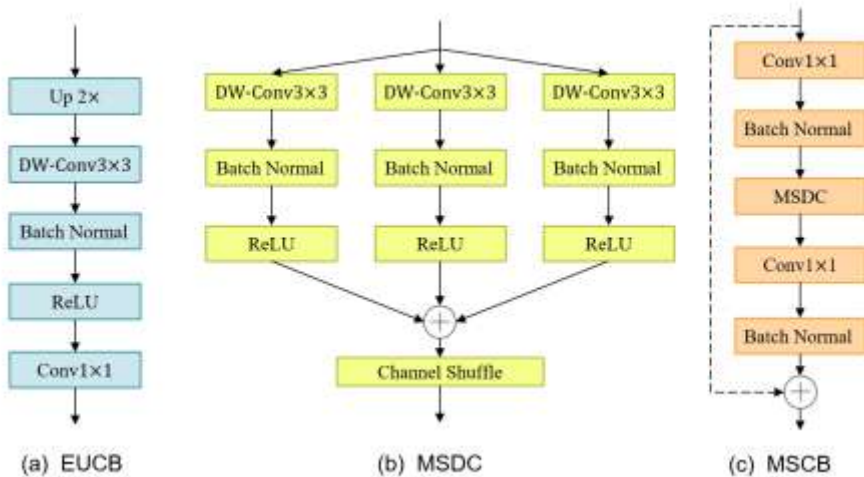


Figure 5: MBS-FPN Module Structure Diagram

MBS-FPN uses an efficient up-convolution block to progressively upsample the feature map of the current stage to match the dimension and resolution of the feature map from the next skip connection. The Efficient Up-Convolution Block (EUCB) first enlarges the feature map proportionally using upsampling with a scale factor of 2. Then, it enhances

the upscaled feature map through a 3×3 depthwise convolution (DW-Conv), batch normalization (BN), and ReLU activation function. Finally, a 1×1 pointwise convolution is used to reduce the number of channels to match the next stage.

$$EUCB(x) = C_{1 \times 1}(ReLU(BN(DWC(Up(x))))) \quad (4)$$

Meanwhile, MBS-FPN introduces an efficient multi-scale convolution block to enhance features generated by the cascaded expansion path. Based on the design of the inverted residual block (IRB) in MobileNetV2 [45], the Multi-Scale Convolution Block (MSCB) performs depthwise convolutions at multiple scales and uses channel shuffle [60] to shuffle channels across groups, enabling more sufficient fusion of channel features from different groups. More specifically, in MSCB, we first expand the number of channels (i.e., expansion factor = 2) using a 1×1 pointwise

convolution layer, a batch normalization (BN) layer, and the ReLU6 function. Then, we use multi-scale depthwise convolution (MSDC) to capture multi-scale and multi-resolution contexts. Since depthwise convolution ignores relationships between channels, we employ a channel shuffle operation to merge inter-channel relationships. Subsequently, we use another pointwise convolution and a batch normalization layer to transform back to the original number of channels. The formula for MSCB is as follows:

$$MSCB(x) = BN(PWC_2(CS(MSDC(ReLU6(BN(PWC_1(x))))))) \quad (5)$$

$$MSDC(x) = \sum_{ks \in KS} DWCB_{ks}(x) \quad (6)$$

2.3. Lightweight Detection Head

The object detection head "Detect" of YOLOv11 uses three sets of convolutions with identical structures to predict targets of different scales, as shown in Figure 10. Although these identical convolution groups can well perform the prediction task, they are not conducive to the lightweight deployment of the model (the parameters of Detect account for

approximately 26.6% of the total model parameters). In addition, the convolution kernels in each convolution group can only learn feature maps on a single-scale convolution group, which is not conducive to improving the generalization ability of the model. Therefore, to achieve the goal of a lightweight model, this paper optimizes the default detection head using shared convolutions, resulting in the LS-Detect (Lightweight Shared Conv Detect) lightweight detection head.

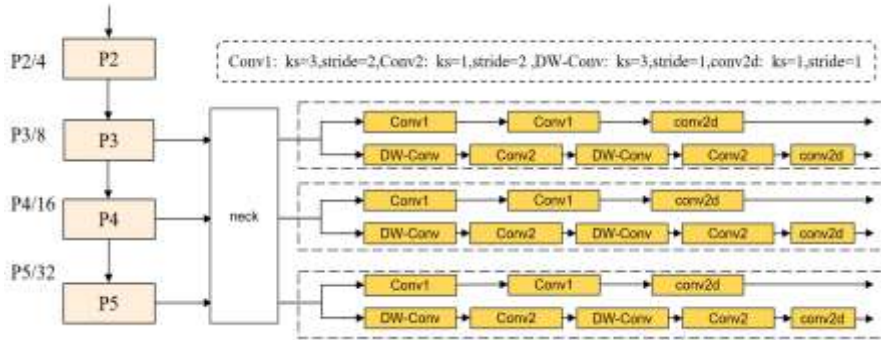


Figure 6: YOLOv11 Detection Head

As shown in the dashed box in Figure 11, a set of shared convolutions is used to replace the convolution branches for object classification and localization in each convolution group of Detect. The output feature maps of P3, P4, and P5 are fed into this set of shared convolutions in sequence. This enables the model to better learn multi-scale features instead of relying on fixed-scale features, thereby improving the

model's generalization ability. Finally, a scale layer is added to scale the features to complete the prediction of targets of different sizes. To address the problem of mismatched input and output channel numbers in shared convolutions, Conv layers are added after P3, P4, and P5 to match the output channel numbers.

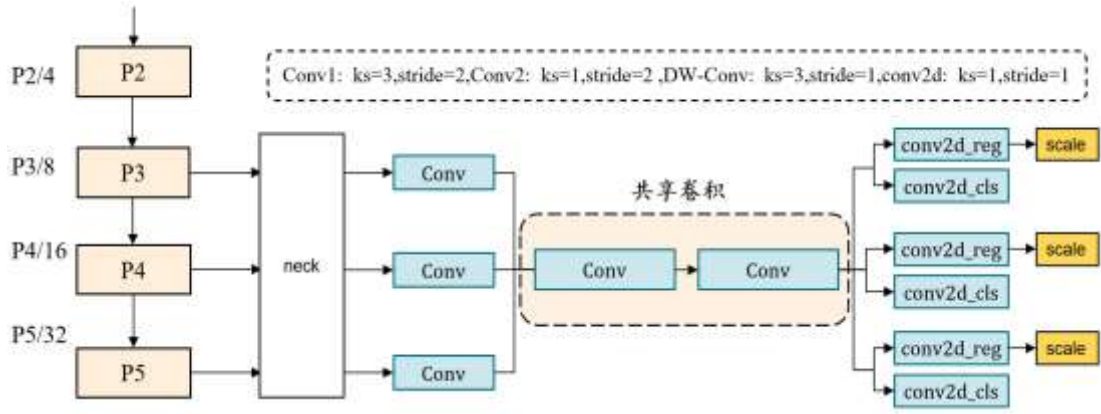


Figure 7: LS-Detect Structure Diagram

Shared convolutions can significantly reduce the number of parameters while saving computational resources, as the same set of parameters can be reused multiple times during the forward and backward propagation of the model, thereby reducing computational and memory consumption. This is particularly suitable for AI devices with limited computing power.

3. Experimental Results and Analysis

To further test the impact of the improvement measures on the overall accuracy of the model, experiments were designed in the training environment to compare changes in the model's Precision (P), Recall (R), mAP50, Parameters (Params), and GFLOPs metrics under different improvement measures. The specific data are shown in Table 1.

Table 1. Comparison Results of Accuracy Under Different Improvement Measures

Model	P(%)	R(%)	mAP50(%)	Params(MB)	GFLOPs
baseline	92.5	87.0	92.1	2.62	6.6
+StarNet	91.9	84.2	91.5	1.94	5.0
+StarNet +MBS-FPN	94.1	85.9	92.2	1.50	4.9
+StarNet +MBS-FPN+LS-Detect	92.4	87.1	92.5	1.32	4.2

Comparing the baseline model with the model using StarNet as the backbone network, the latter reduces 1.6 GFLOPs, indicating that StarNet can achieve lightweight improvements for the model. When comparing the model using only the StarNet backbone network with the model that combines the StarNet backbone network and the MBS-FPN feature fusion module, Precision (P) and mAP50 increase by 1.6% and 1.1% respectively. This is because MBS-FPN can more effectively fuse the high-dimensional implicit features

at different resolutions extracted by StarNet through its bidirectional feature aggregation paths.

In comparison between the baseline model and the model integrating all three improvement measures, the parameter quantity decreases by 1.30 MB (approximately 49.6%), and the latter also reduces 2.4 GFLOPs. This proves that the three improvement measures in this paper achieve lightweight optimization of the model while ensuring the original target detection accuracy of the model.

Table 2. Comparison Results of Various Algorithms

Model	P(%)	R(%)	mAP50(%)	Params(MB)	GFLOPs
RT-DETR-R18	93.9	89.6	93.6	19.8	56.9
Mamba-YOLO-T	93.3	87.5	93.1	5.70	12.5
YOLOv11	92.5	87.0	92.1	2.62	6.6
YOLOv12	92.8	87.3	92.8	2.51	5.8
YOLOv13	92.0	86.7	92.6	2.45	6.1
Ours	92.4	87.1	92.5	1.32	4.2

From the data comparison in the Table 2, in terms of accuracy, mAP50 is a key indicator for evaluating the performance of target detection algorithms. RT-DETR-R18 achieves an mAP50 of 93.6%, which is higher than other compared algorithms. The proposed algorithm in this paper is

0.6%, 0.3%, and 0.1% lower than Mamba-YOLO-T, YOLOv12, and YOLOv13, respectively. In terms of precision, the proposed algorithm is 0.4% higher than YOLOv13. The comparison chart of various algorithms is shown in the figure 8.



Figure 8 Visual Comparison Chart of Contrast Model Detection

4. Conclusions

In this chapter, a lightweight safety helmet detection algorithm is designed based on the public SHWD (Safety Helmet Wearing Dataset). Three effective lightweight improvement measures are adopted to achieve lightweight improvement of the model while ensuring the overall accuracy of the algorithm. First, the backbone network of the YOLOv11 algorithm is replaced with the StarNet network. By utilizing the StarBlock, it can effectively extract high-dimensional implicit features from the feature maps and significantly reduce the number of model parameters. Second, the MBS-FPN (Multi-Branch and Shared Feature Pyramid Network) is used as the neck network. It performs feature fusion through two bidirectional paths and employs the effective EUCB (Efficient Upsampling with Convolutional Block) upsampling module and MSCB (Multi-Scale Convolutional Block) feature extraction module to deeply fuse feature maps of different resolutions.

Finally, shared convolution is used to learn more general features, which reduces the number of parameters and the model complexity. This makes the model less likely to overfit

to features at specific positions in the training data and improves the model's robustness in different scenarios. Relevant experiments show that, compared with other algorithms, the superposition of the three lightweight improvement measures adopted in this chapter is effective, achieving a good balance between algorithm accuracy and complexity.

Conflicts of interest

The authors declare no conflicts of interest to report regarding the present study.

Acknowledgements

The authors express their gratitude to Henan Polytechnic University, for administrative and technical support.

References

- [1] XU Zhuang, QIAN Yurong, YAN Feng. GCW-YOLOv8n: Lightweight Safety Helmet Wearing Detection Algorithm[J]. Computer Engineering and Applications, 2025, 61(3): 144-154.

- [2] YANG Guoliang, HONG Xinfang, SHENG Yangyang, XIONG Wenkai. Helmet Detection Algorithm Based on Appearance Feature Enhancement and Fusion[J]. Computer Engineering and Applications, 2025, 61(15): 363-372.
- [3] LI Jiaxin, HU Yang, HUANG Xiezhou, LI Hongjun. Small Target-Oriented Multi-Space Hierarchical Helmet Detection[J]. Computer Engineering and Applications, 2024, 60(6): 230-237.
- [4] XIAO Zhenjiu, YAN Su, QU Haicheng. Safety Helmet Detection Method in Complex Environment Based on Multi-Mechanism Optimization of YOLOv8[J]. Computer Engineering and Applications, 2024, 60(21): 172-182.
- [5] WANG Xinliang, WANG Luying. Helmet Detection Algorithm Based on Multiple Receptive Field Enhancement in Blasting Scene[J]. Computer Engineering and Applications, 2025, 61(7): 315-324.