Journal of Real-Time Image Processing (2024) 21:140
https://doi.org/10.1007/s11554-024-01515-8

RESEARCH

Csb-yolo: a rapid and efficient real-time algorithm for classroom student behavior detection

Wenqi Zhu[1] · **Zhijun Yang**[2,3]

Received: 29 April 2024 / Accepted: 8 July 2024 / Published online: 27 July 2024

Abstract
In recent years, the integration of artificial intelligence in education has become key to enhancing the quality of teaching. This study addresses the real-time detection of student behavior in classroom environments by proposing the Classroom Student Behavior YOLO (CSB-YOLO) model. We enhance the model's multi-scale feature fusion capability using the Bidirectional Feature Pyramid Network (BiFPN). Additionally, we have designed a novel Efficient Re-parameterized Detection Head (ERD Head) to accelerate the model's inference speed and introduced Self-Calibrated Convolutions (SCConv) to compensate for any potential accuracy loss resulting from lightweight design. To further optimize performance, model pruning and knowledge distillation are utilized to reduce the model size and computational demands while maintaining accuracy. This makes CSB-YOLO suitable for deployment on low-performance classroom devices while maintaining robust detection capabilities. Tested on the classroom student behavior dataset SCB-DATASET3, the distilled and pruned CSB-YOLO, with only 0.72M parameters and 4.3 Giga Floating-point Operations Per Second (GFLOPs), maintains high accuracy and exhibits excellent real-time performance, making it particularly suitable for educational environments.

Keywords Real-time detection · Dense object detection · Student behavior recognition · Feature fusion · Network pruning · Knowledge distillation

1 Introduction

Student behavior to a certain extent reflects the volume of knowledge acquired during class sessions [1]. Therefore, in traditional education, teachers need to constantly monitor student behavior while teaching, to adjust the pace and methods of instruction. However, in actual classroom environments, a single teacher often faces dozens or even more students, and the large number of students can lead to a situation where the teacher lacks sufficient energy to observe student behavior while teaching. Thus, if an accurate and real-time method of detecting student behavior could be utilized to replace the teacher's observation tasks, it would allow teachers to focus more on the teaching itself.

In the current era of educational informatization and intelligence, classroom student behavior detection, as an emerging instructional aid, is increasingly gaining widespread attention from the academic community and educational practitioners. Traditional classroom behavior analysis primarily relies on manual observation, with a common practice being the analysis of student behavior through classroom video recordings [2]. Due to the large volume of videos, manual processing can lead to fatigue and low efficiency, consuming a significant amount of human resources [3]. Moreover, it cannot provide real-time feedback to teachers, which limits its impact on improving teaching effectiveness.

With the rapid development of deep learning technology, real-time and accurate detection of student behavior in the classroom has become feasible. Compared to traditional methods, target detection methods based on deep learning can automatically learn feature data from a large volume of video data, overcoming the limitations of manual feature extraction [4]. Deep learning-based target detection methods

✉ Zhijun Yang
yangzhijun@ynnu.edu.cn

[1] School of Information Science and Technology, Yunnan Normal University, Kunming 650500, Yunnan, China

[2] Educational Instruments and Facilities Service Center, Educational Department of Yunnan Province, Kunming 650223, Yunnan, China

[3] Key Laboratory of Education Informatization for Nationalities of Ministry of Education, Yunnan Normal University, Kunming 650500, Yunnan, China

offer deeper insights into student learning situations in the classroom, while also reducing the pressure on teachers in classroom supervision, thus better enhancing the quality of courses.

Deep learning-based object detection methods are mainly categorized into two types: The first type includes two-stage detection algorithms, where models first generate proposal boxes and then classify them using deep convolutional networks. Common models include R-CNN [5], Fast R-CNN [6], and Faster R-CNN [7]. Although two-stage algorithms are more accurate, they are slower, less capable of real-time processing, and their large model sizes make deployment challenging. The second type includes one-stage target detection algorithms based on regression, which calculate category probabilities and location information simultaneously. These models offer a balance between accuracy and computational speed, with smaller sizes facilitating deployment in practical applications. Common algorithms include YOLO [8], RetinaNet [9], and SSD [10], with the YOLO series being the most widely applied one-stage detection algorithm [11], where YOLOv8 represents the best balance between network lightweighting and detection accuracy in the YOLO series.

While the YOLO algorithm has found applications in various areas, its deployment for student behavior detection in classrooms still faces significant challenges. Additionally, due to camera resolution limitations, the pixel count representing each student's body in the image is typically very low. Moreover, the significant size discrepancy between students sitting at the front and those at the back in the classroom scenes leads to the presence of multi-scale targets [12]. Addressing these issues necessitates a larger, more precise model. Yet, most schools lack the necessary equipment to run such complex models, necessitating a model that is both lightweight and capable.

This paper designs and proposes a model based on an improved version of YOLOv8n, CSB-YOLO, specifically tailored for real-time detection of student behaviors in classroom settings. CSB-YOLO boasts minimal parameters and computational requirements, making it highly deployable on low-performance devices in schools due to its low device demands and satisfactory accuracy. The main contributions of this paper are as follows:

1. This paper introduces the BiFPN structure [13] into the YOLOv8 model, with the aim of enhancing its capability to detect densely distributed small targets while reducing both computational requirements and parameter count.
2. We have devised a novel Efficient Re-parameterized Detection Head for YOLOv8, replacing the original detection head structure. This modification significantly reduces model complexity, accelerates inference speed, thereby enhancing real-time performance. Additionally, we have enhanced the C2f module by incorporating SCConv [14] to compensate for any potential accuracy loss resulting from the lightweight design of the detection head.
3. The paper employs LAMP pruning [15] to optimize the model's structure, significantly reducing both parameter count and computational load, making it more suitable for deployment on low-performance devices. Moreover, to prevent a reduction in model accuracy due to pruning, the pruned model undergoes BCKD knowledge distillation [16]. This combination of pruning with knowledge distillation achieves nearly lossless lightweighting of the model.

2 Related works

Behavior detection of human targets remains a hot topic in the field of object detection, but behavior detection in classroom settings poses many challenges, such as detecting multi-scale dense targets and the need for lightweight models in practical applications. This section will explore and analyze solutions to these challenges.

Zhang et al. [17] established a dataset for the behavior of students raising hands in classroom environments and discovered that information loss occurred due to the reduction of channel numbers during the construction of the feature pyramid [18]. By applying Spatial Context Augmentation (SCA) and multi-branch feature fusion modules, the precision of hand-raising detection was enhanced. However, the complexity of the network structure may compromise real-time performance.

Wang et al. [19] introduced a method to detect yawning in classroom environments, integrating the feature pyramid within R-FCN [20] to tackle issues such as occlusions and low-resolution facial recognition, and employed channel pruning to diminish both the model's parameter count and computational overhead. While pruning drastically decreases the number of parameters, it inevitably leads to a decline in precision as the pruning ratio escalates, thereby raising a pivotal challenge concerning how to maintain or improve precision amidst the pruning process.

Cheng et al. [21] introduced the concept of a cross-stage local network at the end of the YOLO-v4 [22] network, embedding the Embedding Connection (EC) component to develop an improved YOLO-v4 network for detecting teacher and student behaviors. Bao et al. [23] improved the model based on YOLOv5 by adding a feature fusion layer and incorporating the ghost module [24] to replace standard convolutions, thus enhancing the model's capability to detect behaviors in the classroom. Wang et al. [25] introduced the CBAM [26] attention mechanism to the YOLOv7 [27] foundation, effectively capturing contextual features

and enhancing the network's feature detection capability, allowing accurate detection of multiple students' learning behaviors. Cheng et al. [28] improved the C2f structure with Res2Net [29] on the YOLOv8 base, enhancing the network's ability to extract multi-scale features. They also introduced the EMA [30] attention mechanism in the backbone to address occlusion issues in classroom settings. While the introduction of attention mechanisms significantly enhances the model's feature learning capability and accuracy, it also significantly increases computational overhead, adversely affecting network lightweighting.

Recent studies, such as Liu et al. [31], have enhanced YOLOv8 by adding a small object detection layer equipped with a dedicated detection head specifically for small objects. Although this dedicated detection head does indeed improve the network's ability to detect small objects, it inevitably complicates the network and the additional detection head can limit the network's inference speed. Meanwhile, Xiao et al. [32] have improved network accuracy by incorporating the IMPDIoU loss function into YOLOv8, but this method does not enhance the network's inference speed.

Analysis of the above research reveals that enhancing the network's learning capability for multi-scale targets is crucial for addressing the difficulty of detecting multi-scale dense targets. Feature fusion is an important method to enhance multi-scale feature learning capabilities [33], and adding attention mechanisms can significantly improve detection accuracy. However, the increased computational overhead caused by complex network structures remains a problem to be solved. Network pruning is an effective lightweighting method that significantly reduces the network's parameter count and computational load, but it requires a careful balance between pruning rate and accuracy.

3 CSB-YOLO detection model

3.1 YOLOv8

YOLOv8, an evolution from YOLOv5 by Ultralytics, is a single-stage object detection algorithm. It employs the CSP gradient bifurcation concept and the SPPF module in both backbone and neck parts. Notably, it adopts a more gradient-rich C2f structure over the C3 structure from its predecessor. In the head section, it utilizes a decoupled structure, separating classification and detection heads. YOLOv8 also shifts from the traditional Anchor-Based approach to an Anchor-Free concept, enhancing its lightweight design for deployment on low-performance devices in real-world classroom settings. To further optimize its lightweight nature, this paper builds upon the smallest variant, YOLOv8n.

3.2 Overview of CSB-YOLO

To address the challenge of real-time student behavior detection in classroom scenarios, we devised the CSB-YOLO model, as illustrated in Fig. 1. Initially, to address the challenge of detecting densely distributed student targets in classroom scenarios, we modified the Neck section of YOLOv8 to incorporate the BiFPN structure. Additionally, we engineered an Efficient Re-parameterized Detection Head for the network, replacing the original detection head structure of YOLOv8. This adjustment not only reduces network complexity but also accelerates the inference process, enhancing real-time performance. Furthermore, we devised a C2f_SCConv structure to precisely locate student targets in classrooms, thereby compensating for any potential accuracy loss resulting from lightweight detection heads. To further streamline the network, we applied the LAMP pruning technique, significantly reducing the network's parameter count and computational load, thereby lowering complexity and facilitating easier deployment. Lastly, for the pruned model, we implemented the BCKD distillation strategy to ensure the model maintains high detection accuracy while remaining lightweight.

3.3 BiFPN

In the Neck section, YOLOv8 utilizes the PANet [34] structure for feature fusion. While PANet, compared to the traditional FPN structure, facilitates bidirectional feature fusion while retaining multi-scale feature information, this fusion mechanism depends on numerous nodes, leading to increased computational and parameter requirements of the network. To reduce the computational and parameter demands without compromising accuracy, we have incorporated the BiFPN [13] structure into YOLOv8.

BiFPN represents an advanced version of the FPN architecture, establishing cross-scale connections through bidirectional channels, where each layer receives features from both higher and lower levels. In contrast to PANet, BiFPN eliminates nodes with only one input and employs weighted feature fusion. Additionally, to combine more features, an extra pathway is introduced, linking input and output nodes of the same level. This design allows the network to better balance semantic and spatial information across different layers, preserving shallow semantic details without sacrificing significant deep semantic information. Thanks to the reduction in nodes, BiFPN significantly decreases both computational and parameter requirements of the network, while its efficient multi-scale feature fusion capability maintains accuracy. Figure 2 illustrates three distinct Neck structures.

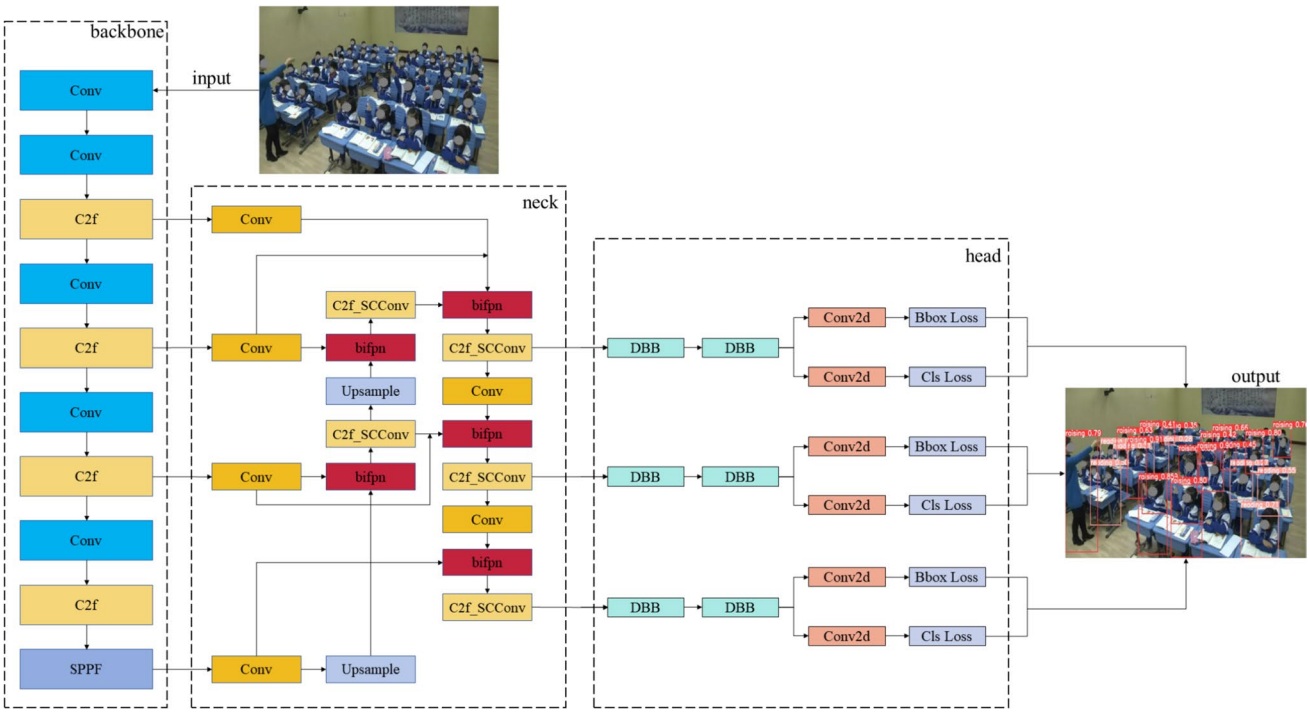

Fig. 1 The architecture of the CSB-YOLO

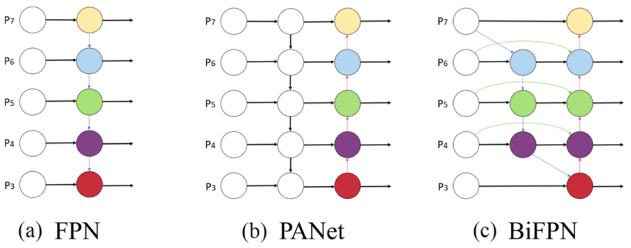

Fig. 2 The structures of three different Necks

3.4 Efficient re-parameterized detection head

YOLOv8 features three detection heads, each with two paths containing two 3x3 convolutions for feature extraction, resulting in a total of twelve 3x3 convolutions within the detection head section of the network. Although this configuration improves accuracy to some extent, the extensive use of convolutional kernels increases the network's parameter count and slows down inference speed.

Drawing inspiration from the parameter-sharing approach employed in the detection heads of RetinaNet [9], we redesigned the detection head of YOLOv8. The redesign consolidates the four 3×3 convolutions used for feature extraction along the two paths within the original detection head into two, allowing both classification and box regression to share these two 3x3 convolutions. This modification reduces the complexity of the head section and increases the network's inference speed, inevitably resulting in a slight reduction in accuracy. To minimize the loss in accuracy while simplifying the network, we introduced the Diverse Branch Block (DBB) [35], replacing the original convolutions.

DBB is a cost-free universal module that utilizes reparameterization techniques, building upon the foundations of ACNet [36] and RepVGG [37] by exploring more equivalent transformations. DBB takes cues from the Inception [38–41] structure, enriching the feature space of the convolutional block with a multi-branch architecture. During the inference phase, the multiple branches are reparameterized and merged into a single main branch, optimizing the network's performance while maintaining precision.

Transform I - Convolution with Batch Normalization (Conv-BN): A convolution layer is often equipped with a BN layer which performs channel-wise normalization and scaling.

$$O_j = \frac{(I * F)_j - \mu_j}{\sigma_j}\gamma_j + \beta_j \tag{1}$$

where $*$ denotes convolution, I is the input, F is the filter μ_j, σ_j are the mean and standard deviation for batch normalization, and γ_j, β_j are the scale and shift parameters. This transformation fuses batch normalization parameters into the convolution filters for inference.

Transform II - Addition of Branch Outputs: The additivity property of convolution allows the merging of outputs

from multiple convolution layers with the same configuration by simply adding their weights and biases:

$$F' = \sum_i F^{(i)}, \quad b' = \sum_i b^{(i)} \tag{2}$$

$F^{(i)}$ and $b^{(i)}$ represent the convolution kernels and biases of the ith branch, respectively. F' and b' are the combined kernel and bias.

Transform III - Sequential Convolutions: A sequence of convolutions, typically involving small kernel sizes like 1x1 followed by larger kernels like KxK, can be merged into one effective convolution. The transformation rearranges and combines the weights from sequential layers to form a single layer that encapsulates the collective effect:

$$F' = F^{(1)} * F^{(2)}, \quad b' = F^{(1)} * b^{(2)} + b^{(1)} \tag{3}$$

This is especially useful for reducing depth and computational complexity in the network.

Transform IV - Depth Concatenation: Depth concatenated outputs from different branches are merged into a single convolution layer:

$$F' = \text{Concat}\left(F^{(1)}, F^{(2)}\right), \quad b' = \text{Concat}\left(b^{(1)}, b^{(2)}\right) \tag{4}$$

Concat denotes the concatenation operation along the channel dimension, allowing multiple branches to combine into a single convolution operation.

Transform V - Convolution for Average Pooling: An average pooling operation is modeled as a convolution with a uniform kernel:

$$F'_{d,c,:,:} = \begin{cases} \frac{1}{K^2} & \text{if } d = c \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

d and c are indices for the output and input channels, respectively; K is the size of the pooling (or convolution) window; $F'_{d,c,:,:}$ defines each element of the convolution kernel to implement pooling.

Transform VI - Handling Multi-Scale Convolutions: Convolutions with different kernel sizes are unified into a larger convolution operation through appropriate padding:

$$F'_{\text{padded}} = \text{Zero-Padding}\left(F_{\text{small}}\right) \tag{6}$$

F_{small} represents a smaller kernel, which is padded with zeros to match the size of the largest kernel in the transformation, referred to here as F'_{padded}.

A complete DBB block, as depicted in Fig. 3, consists of four branches. Using the aforementioned six parameter transformation methods, it is possible to convert complex multi-branch structures into standard convolutions and reuse the weights obtained during training. DBB facilitates the separation of training and inference phases: it employs a more complex network structure during training to improve network accuracy and undergoes equivalent transformations during inference to accelerate the inference process.

The detection head designed in this paper, as illustrated in Fig. 4, replaces the original four convolutions of the detection head with two DBB modules. During the inference phase, these two DBB modules are equivalently transformed into two standard convolution modules. Thanks to the reduction in the number of convolutions, both the network's parameter count and computational load significantly decrease. This will be validated in the experimental section.

3.5 C2f_SCConv

In classroom environments, student positions are often densely distributed, leading to frequent occurrences of

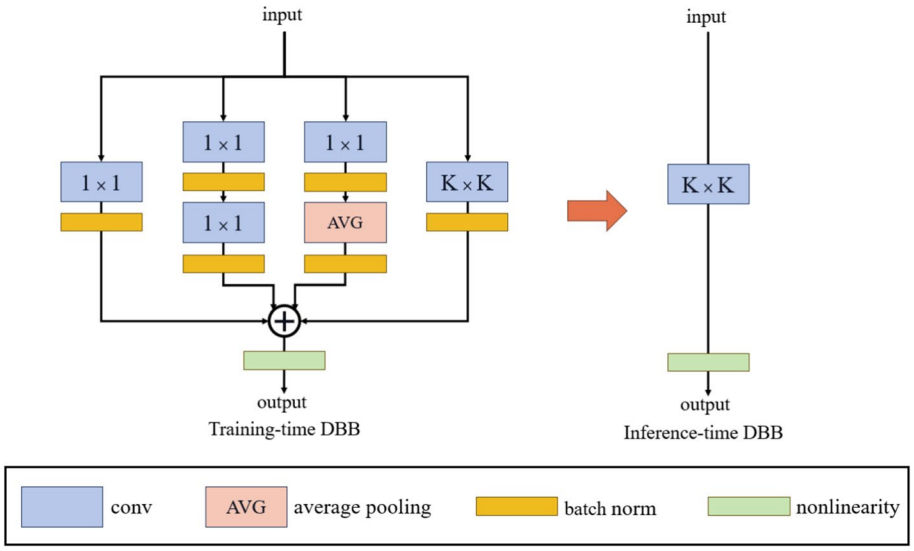

Fig. 3 The basic structure of DBB

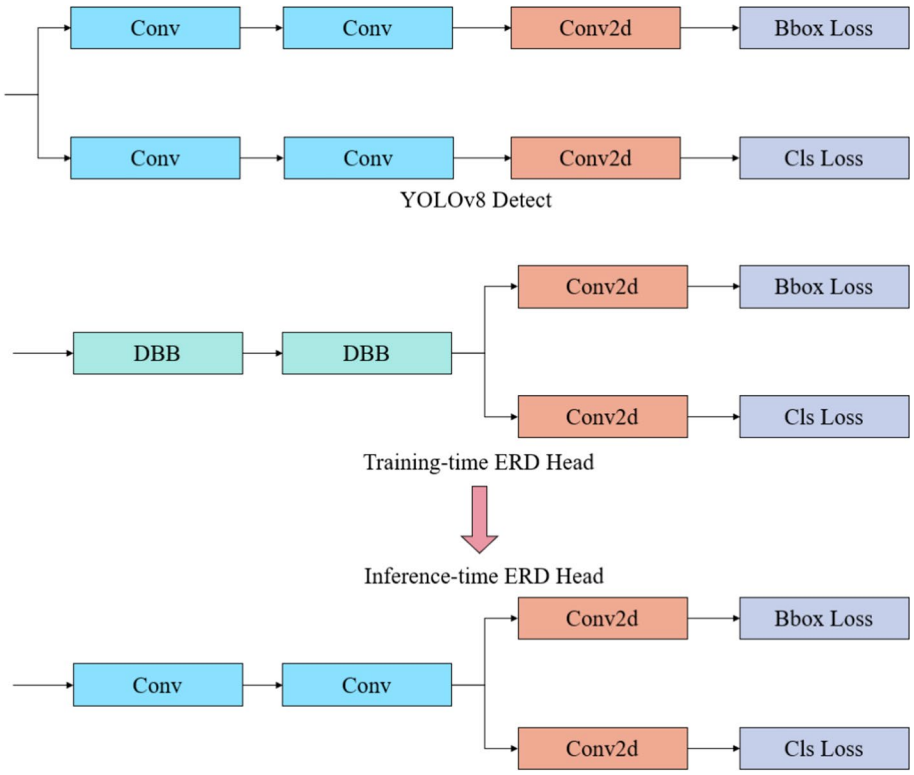

Fig. 4 A structural comparison between ERD Head and the detection head of YOLOv8

overlap and occlusion. The lightweight detection head inevitably reduces the network's ability to detect complex human targets. To address this issue, we redesigned the C2f module in the YOLOv8 structure using SCConv [14], increasing the receptive field of the C2f module to compensate for the accuracy loss caused by the lightweight detection head.

The core idea of SCConv is to enhance the fundamental convolutional feature transformation process of CNNs without modifying the model architecture. It essentially employs grouped convolutions for multi-scale feature extraction, dividing them into two groups along the channel dimension. One pathway conducts regular convolutional feature extraction, while the other pathway utilizes downsampling operations to enlarge the network's receptive field. This enables each spatial location to conduct self-calibrated operations by integrating information from two different spatial scales.

The workflow, as depicted in Fig. 5, involves input and output channels both of size C, with a given set of filters K shaped as (C, C, k_h, k_w), where k_h and k_w represent the spatial height and width, respectively. Initially, the process involves segmentation, resulting in four groups of filters $\{K_i\}_{i=1}^{4}$, each with a shape of $\left(\frac{C}{2}, \frac{C}{2}, k_h, k_w\right)$. Subsequently, the input X is evenly divided into two parts $\{X_1, X_2\}$. The self-calibration operation is performed on X_1 using filters $\{K_1, K_2, K_3\}$, yielding Y_1. In the second pathway, a simple convolution operation is executed: $Y_2 = F_1(X_2) = X_2 * K_1$, aiming to preserve the original spatial context information. Finally, $\{Y_1, Y_2\}$ are concatenated to form the final output Y.

In detail, the Self-Calibrated operation starts with applying average pooling to the given input X_1, using a kernel size of $r \times r$ and a stride of r, denoted as:

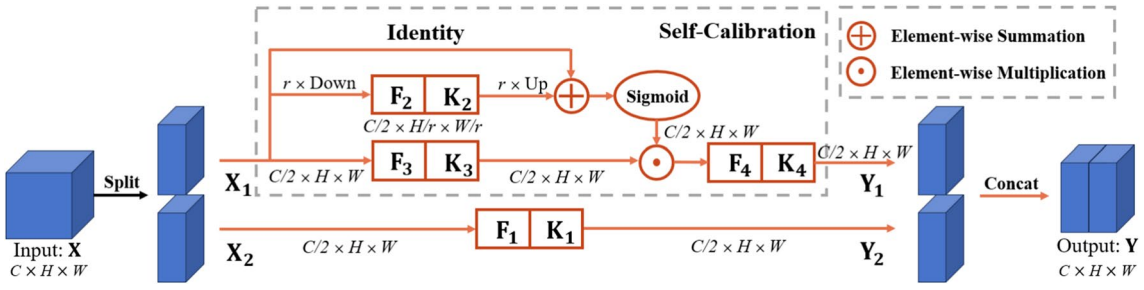

Fig. 5 A schematic illustration of self-calibrated convolutions

$$T_1 = AvgPool_r(X_1) \tag{7}$$

Next, T_1 undergoes a feature transformation using filter K_2:

$$X_1' = Up(F_2(T_1)) = Up(T_1 * K_2) \tag{8}$$

Here, $Up(\cdot)$ represents the operation of linear interpolation, mapping the intermediate quantity from a smaller scale space back to the original feature space. The self-calibration operation can then be represented as:

$$Y_1' = F_3(X_1) \cdot \sigma(X_1 + X_1') \tag{9}$$

where $F_3(X_1) = X_1 * K_3$, σ denotes the sigmoid function, and the symbol "." indicates element-wise multiplication. X_1' serves as a residual term to establish weights for self-calibration. The final output after self-calibration can be written as:

$$Y_1 = F_4(Y_1') = Y_1' * K_4 \tag{10}$$

In this study, the C2f structure, improved by SCConv, is illustrated in Fig. 6. We enhanced the BottleNeck by replacing the convolution of the second CBS with SCConv, resulting in a structure we denote as BottleNeck_SCC. Subsequently, stacking n groups of BottleNeck_SCC forms the C2f_SCConv structure. Owing to SCConv's larger receptive field, C2f_SCConv facilitates more precise localization of student targets within the classroom environment.

3.6 LAMP pruning

Although the enhancements previously implemented have provided the network with a certain level of real-time capability, it is still necessary to further lighten the network, especially considering the device performance within classroom scenarios. Complex neural network models often contain redundancies after training. These redundancies, which are relatively less important, do not substantially improve network accuracy but increase the network's parameter size, thus slowing down inference time.

We adopted a pruning method based on LAMP (Layer-Adaptive Magnitude-based Pruning) scores [15]. LAMP is designed to automatically select the optimal level of sparsity among layers in a neural network to achieve the best balance between model performance and sparsity. Using LAMP scores, an adaptive, global pruning strategy can be implemented, eliminating the need for manual adjustment of hyperparameters.

The LAMP process is as follows: For a feedforward neural network with depth d, where each fully connected/convolutional layer is associated with a weight tensor $w^{(1)}, \ldots, w^{(d)}$. For fully connected layers, the corresponding weight tensor is a two-dimensional matrix; for 2D convolutional layers, the corresponding tensor is four-dimensional.

To unify the definition of LAMP scores for both fully connected and convolutional layers, weight tensors are flattened into one-dimensional vectors. According to a given index mapping, weights are sorted in ascending order, meaning that for $u < v$, it always holds that $|W[u]| \leq |W[v]|$, where $W[n]$ denotes the element in W mapped from index u. The LAMP score for the u-th weight index W is:

$$score(u;W) := \frac{(W[u])^2}{\sum_{v \geq u}(W[v])^2} \tag{11}$$

The LAMP score evaluates the relative importance of a target connection amongst all the surviving connections within the same layer. After calculating the LAMP scores, the connections with the lowest scores are globally pruned until the required global sparsity constraint is achieved. It is important to note that within each layer, there is a single connection with a LAMP score of 1, which is the maximum possible LAMP score. This ensures that at least one connection

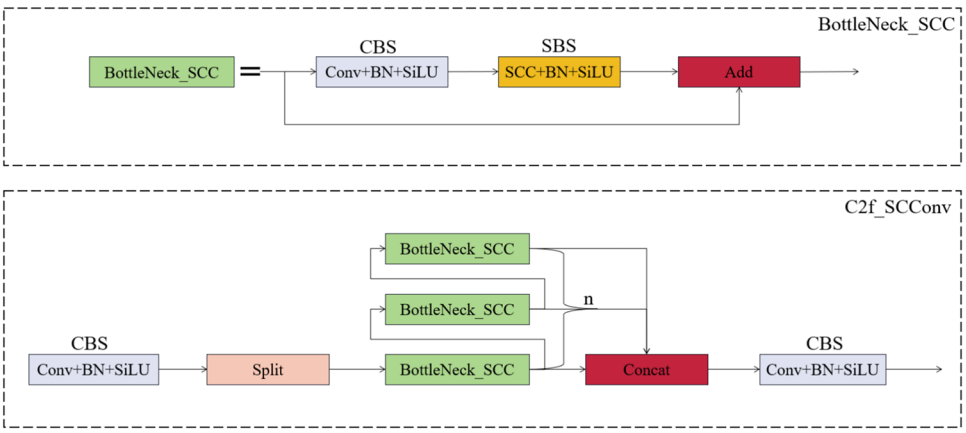

Fig. 6 The structure diagram of the C2f_SCConv module

is preserved in every layer. This process essentially constitutes minimum magnitude pruning with layer-wise sparsity levels chosen automatically, obviating the need for intricate parameter settings.

3.7 BCKD knowledge distillation

BCKD [16] is a new knowledge distillation method tailored for dense object detection tasks, designed to address the inefficiency of traditional classification distillation in such contexts. In dense object detection, the pronounced imbalance among foreground categories leads to traditional softmax-based knowledge distillation methods overlooking the absolute classification scores of each category. This oversight can result in the optimal solution for the distillation loss function not necessarily ensuring the best classification performance in the student model. The fundamental principle of BCKD is to reframe the multi-class classification challenge into multiple binary classification tasks, applying knowledge distillation to each binary classification task individually.

The distillation process of BCKD is illustrated in Fig. 7. It incorporates two novel distillation loss functions specifically designed for object detection tasks: (i) Binary classification distillation loss, L_{cls}^{dis}, which represents the classification logits as multiple binary maps and extracts classification knowledge through a distillation loss resembling binary cross-entropy; (ii) IoU-based localization distillation loss, L_{loc}^{dis}, which transfers localization knowledge from the teacher model to the student model by calculating the IoU values between the bounding boxes predicted by the two models and employing IoU loss.

The Binary Classification Distillation Loss aims to address the severe imbalance between foreground and background categories in dense object detection. This approach tackles the challenge by transforming the multi-class problem into multiple binary classification tasks.

Classification Scores: For each location i and category j the classification logits l_{ij} are transformed into classification scores p_{ij} using the Sigmoid function:

$$p_{ij} = \frac{1}{1 + e^{-l_{ij}}} \tag{12}$$

Binary Cross-Entropy Loss: For each sample x, the classification loss $L_{cls}(x)$ is computed as:

$$L_{cls}(x) = \sum_{i=1}^{n} \sum_{j=1}^{K} L_{CE}\left(p_{ij}, y_{ij}\right) \tag{13}$$

where $L_{CE}\left(p_{ij}, y_{ij}\right)$ is the binary cross-entropy loss, y_{ij} is the true label, n is the number of samples, and K is the number of categories.

Distillation Loss Weighting: Given the classification scores p_{ij}^{t} from the teacher model and p_{ij}^{S} from the student model, the binary classification distillation loss $L_{cls}^{dis}(x)$ is:

$$L_{cls}^{dis}(x) = \sum_{i=1}^{n} \sum_{j=1}^{K} w_{ij} \cdot L_{BCE}\left(p_{ij}^{s}, p_{ij}^{t}\right) \tag{14}$$

where w_{ij} are weights determined based on the score differences, intended to focus the learning on important samples.

The IoU-based Localization Distillation Loss is designed to enhance localization performance by calculating the IoU between bounding boxes predicted by the teacher and student models.

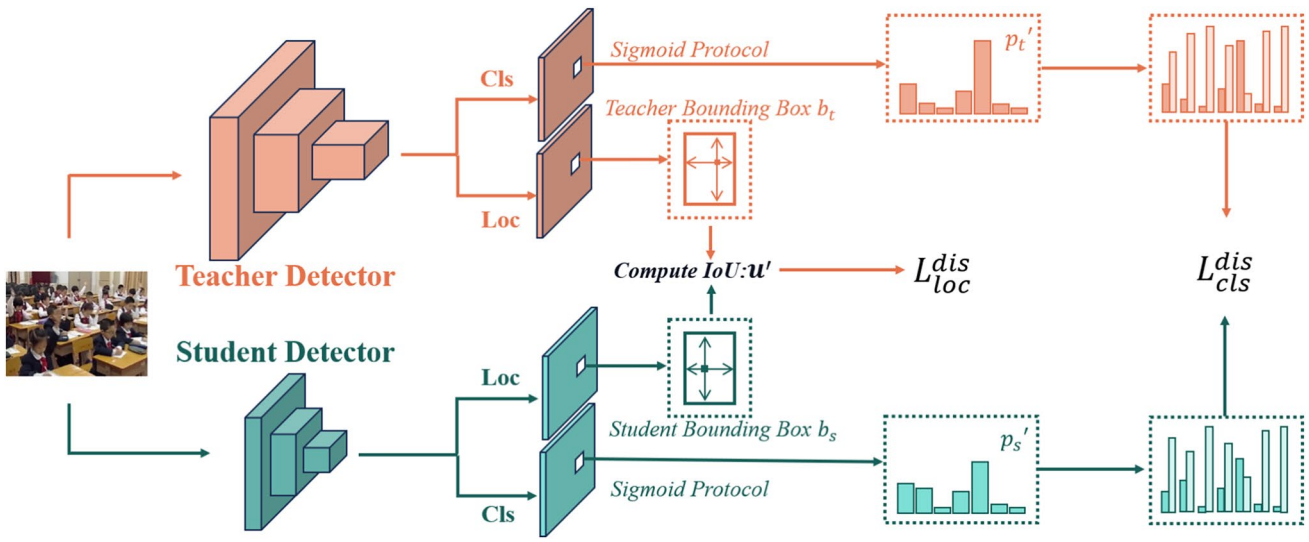

Fig. 7 The workflow of BCKD knowledge distillation

IoU Calculation: Given the bounding boxes b_i^t and b_i^s predicted by the teacher and student models, respectively, their IoU u_i' is computed as:

$$u_i' = IoU\left(b_i^t, b_i^S\right) \tag{15}$$

Localization Distillation Loss: The IoU-based localization distillation loss $L_{loc}^{dis}(x)$ is formulated as:

$$L_{loc}^{dis}(x) = \sum_{i=1}^{n} \max\left(\omega_j\right) \cdot \left(1 - u_i'\right) \tag{16}$$

where $\max\left(\omega_j\right)$ represents the maximum weight among categories, emphasizing crucial localization information.

Total Distillation Loss: The total distillation loss is a linear combination of the binary classification distillation loss and the IoU-based localization distillation loss, aiming to optimize both classification and localization tasks concurrently. The total distillation loss $L_{total}^{dis}(x)$ is defined as:

$$L_{total}^{dis}(x) = \alpha_1 \cdot L_{cls}^{dis}(x) + \alpha_2 \cdot L_{loc}^{dis}(x) \tag{17}$$

where α_1 and α_2 are hyperparameters that adjust the relative importance of classification and localization losses in the total loss.

The design of these loss functions takes into account the foreground-background class imbalance inherent in dense object detection tasks, as well as the significance of localization accuracy. Through these meticulously crafted loss functions, effective training and optimization of the student model are achieved.

4 Experimental results and analysis

4.1 Experimental dataset

Leveraging deep learning for the automatic detection of student behavior is a critical strategy for enhancing teaching effectiveness. Nonetheless, the lack of publicly available datasets on student behavior presents a significant challenge to researchers in this domain. The dataset employed in this study is the SCB-Dataset3 (Student Classroom Behavior dataset) [42], which reflects real classroom scenarios. SCB-Dataset3 includes two subsets: SCB-Dataset3-S, consisting of classroom behavior data from elementary and middle schools, and SCB-Dataset3-U, comprising university classroom behavior data.

Our primary focus is on SCB-Dataset3-S, which is utilized as the training and validation dataset. The SCB-Dataset3-S dataset consists of 5015 images and 25,810 annotations, categorized into three types: hand raising, reading, and writing. Figure 8 displays the number of instances for each category, while Fig. 9 presents example images from

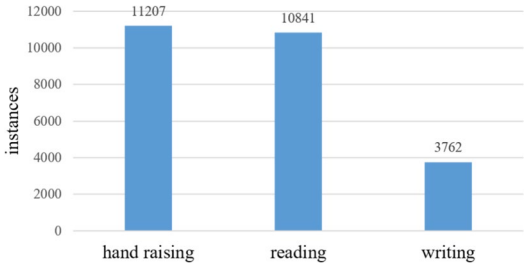

Fig. 8 Number of labels for each category in the SCB-Dataset3-S dataset

the dataset. Furthermore, relying solely on a single dataset may not provide a comprehensive evaluation of a model's performance. Therefore, we also conduct transfer training on the relatively smaller SCB-Dataset3-U dataset to test the model's generalization ability and robustness.

4.2 Evaluation metrics

In the classroom scenarios covered by this study, targets of various scales coexist, necessitating a comprehensive evaluation of an object detection model's effectiveness. Consequently, the precision metrics employed in this paper include F1, mAP0.5, and mAP0.5:0.95. In parallel with testing the model's performance, it's also essential to evaluate the model's number of parameters, model file size, FLOPs, and FPS to thoroughly analyze the model's capabilities.

(1) F1: Represents the harmonic mean of Precision and Recall, allowing for a combined assessment of accuracy and recall rates, with higher results being preferable. The calculation formula is as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{18}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{19}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

(2) mAP: Measures the average precision (AP) across all categories. AP for each category is determined by first plotting the precision-recall curve, then calculating the area under this curve, which represents the AP for that category. mAP is the mean of the AP values across all categories. The formula can be expressed as:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{21}$$

Fig. 9 Sample display of the SCB-Dataset3-S Dataset

(3) Number of Parameters: Used to evaluate the model's size and complexity, it is obtained by summing the number of weight parameters for each layer. For lightweight models, a lower number of parameters is preferred.
(4) FLOPs: Indicates the total number of floating-point operations required to perform a forward pass of the model, serving as an indicator of the model's computational complexity and efficiency.
(5) Onnx file size: Represents the size of the model file in Onnx format, which directly impacts the model's deployability on various devices. The smaller the weight file, the lower the storage space requirement on devices.
(6) FPS (Frames Per Second): Denotes the number of images the model can process per second. A higher FPS indicates better real-time performance of the model.

4.3 Experimental environment and parameter settings

In this study, the Pytorch 1.13.1 deep learning framework was used to train each model for 300 epochs on the SCB-Dataset3-S dataset and conduct transfer training for 100 epochs on the SCB-Dataset3-U dataset. The momentum was set to 0.937, weight decay to 0.0005, initial learning rate (lr0) to 0.01, image input size to 640×640, and batch size to 8. The experimental platform operated on Ubuntu 20.04, with an Intel(R) Xeon(R) Platinum 8255C CPU at 2.5GHz, 32 GB RAM, and an NVIDIA RTX 3080 GPU. Additionally, inference speed tests were performed on a pure CPU device without a GPU, specifically an Intel(R) I5 12400. The model pruning rate is set at 25%. In the knowledge distillation section, we use the unpruned CSB-YOLO as the teacher model and the pruned CSB-YOLO as the student model. The distillation training runs for 300 epochs with a loss rate of 1.2.

4.4 Choice of Pruning Rate

We conducted tests on the SCB-Dataset3-S dataset to assess the impact of pruning rate on accuracy. Table 1 shows the effects of different LAMP pruning rates on the performance of CSB-YOLO, ranging from 10% to 80%. It was found that with the increase in pruning rate, the model's number of parameters, computational load, and model file size all gradually decreased, and the FPS significantly improved. However, the accuracy also gradually declined with the increase in pruning rate. After exceeding a 30% pruning rate, the accuracy began to sharply decrease, and at an 80% pruning rate, the model became unusable. An important observation is that when the pruning rate was below 25%, the fine-tuned model even surpassed the unpruned model in mAP0.5. This is because the original model contained a substantial amount of redundancy, which not only slowed down the model's computational speed but also increased the difficulty of training. Therefore, removing this redundancy could improve the model's training accuracy.

The primary objective of the pruning phase in this study is to maximize model lightweighting without compromising accuracy. At a pruning rate of 25%, the FPS saw a significant increase compared to a 20% pruning rate, while the accuracy was higher than at a 30% pruning rate and very close to that of the original model, making 25% a more cost-effective choice. Subsequent experiments in this paper all employed a 25% pruning rate.

Table 1 The impact of different pruning rates on the performance of CSB-YOLO

Method	mAP0.5	Params(M)	GFLOPs	Onnx file size(MB)	FPS(CPU)
CSB-YOLO(0%)	0.703	1.86	6.2	7.33	31.46
10%-Pruned	0.719	1.08	5.4	4.35	32.35
20%-Pruned	0.705	0.81	4.7	3.33	34.53
25%-Pruned	**0.704**	**0.72**	**4.3**	**2.95**	**37.17**
30%-Pruned	0.696	0.62	3.9	2.59	38.25
40%-Pruned	0.66	0.49	3.1	2.11	39.34
50%-Pruned	0.608	0.37	2.4	1.64	44.32
60%-Pruned	0.53	0.25	1.7	1.18	61.95
70%-Pruned	0.333	0.15	0.9	0.813	92.09
80%-Pruned	0.076	0.13	0.7	0.707	116.16

Since the detection head is responsible for outputting the detection results, its structure should remain unchanged; thus, it was not involved in the pruning process. We conducted a comparison of the number of channels in layers other than the model's detection head, where layers 0 to 9 constitute the backbone of the model, and layers 10 to 27 form the Neck part. Figure 10 illustrates a significant reduction in the number of channels, with the total channels decreasing from 4176 to 2587.

4.5 Selection of Distillation Loss Rate

To achieve enhanced precision while ensuring model lightness, it is imperative to perform BCKD distillation on the model post-pruning. The importance of each layer's weights is discernible through a comparison of channel counts before and after pruning. As observed in Fig. 10, a significant portion of pruning occurs within the model's backbone, indicating the presence of considerable redundancy in this section, which minimally impacts the overall network accuracy. Consequently, we focused on distilling layers 15, 18, 21, 24, and 27, where the change in channel numbers before and after

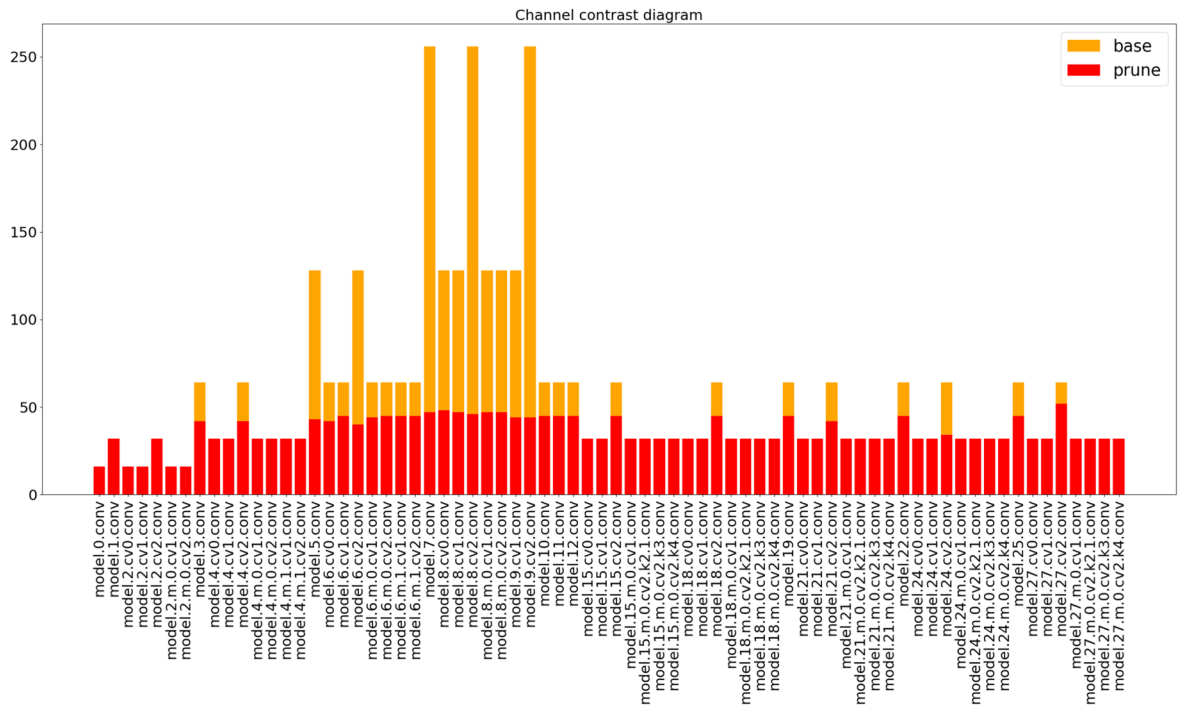

Fig. 10 Comparison of the number of network channels before and after pruning

pruning was minor, signifying their crucial role in maintaining network accuracy.

Additionally, we selected the unpruned CSB-YOLO as the teacher model because the structures of layers 15, 18, 21, 24, and 27 remained almost unchanged before and after pruning, making the teacher and student models structurally similar. This similarity in structure can reduce the complexity of the distillation training process. In addition, experiments were conducted on the SCB-Dataset3-S dataset to evaluate the impact of varying the loss ratio coefficient during the distillation process, with the distillation training spanning 300 epochs. As depicted in Fig. 11, the optimal performance across all accuracy metrics was observed when the loss ratio was set to 1.2. Consequently, a consistent loss ratio of 1.2 was employed in the subsequent experiments of this study.

4.6 Comparison of different distillation methods

To validate the effectiveness of BCKD distillation, we introduced three feature distillation methods: cwd, mgd, and mimic, along with two logical distillation methods: L1 and L2. Tests were conducted on the SCB-Dataset3-S dataset, with the main parameters consistent with those described in section 4.5. The experimental results, as presented in Table 2, reveal that BCKD distillation achieved the best performance among all comparison methods, fully demonstrating the effectiveness of the BCKD distillation approach in classroom environments with densely populated targets.

Table 2 Comparing different knowledge distillation methods

Method	F1	mAP0.5	mAP0.5:0.95
CSB-YOLO(prune)	0.661	0.704	0.518
+cwd [43]	0.656	0.695	0.508
+mgd [44]	0.486	0.456	0.290
+mimic [45]	0.656	0.695	0.508
+L1 [46]	0.661	0.701	0.515
+L2 [46]	0.659	0.705	0.519
+BCKD [16]	**0.666**	**0.711**	**0.523**

4.7 Experiments and comparisons

To verify the effectiveness of the proposed CSB-YOLO, comparative experiments were conducted with commonly used object detection models on the SCB-Dataset3-S dataset. All models underwent 300 training epochs, with YOLOv8n as the baseline model. The results, as shown in Table 3, indicate that after model pruning and knowledge distillation, the CSB-YOLO model exhibited the lowest parameter count among all the models compared, at merely 23.9% of the baseline model's parameters. Its GFLOPs were also reduced to 53% of the baseline. Furthermore, its accuracy surpassed all other smaller models tested; specifically, it achieved an mAP0.5 of 0.711, which is an increase of 0.8% over the baseline, and its mAP0.5:0.95 also improved by 0.3%. Although it is slightly less accurate than larger-scale models such as YOLOv7, YOLOv9-c, and YOLOv3, it significantly outperforms these models in terms of both parameter size and computational efficiency, key metrics for lightweight models.

Table 4 displays the accuracy across various categories, revealing that after pruning and distillation, CSB-YOLO outperforms the baseline in detecting behaviors such as raising hands and writing.

To observe the detection performance of the models more intuitively, we randomly selected image samples and visualized the detection results of YOLOv8n and CSB-YOLO. The results are shown in Fig. 12. It can be observed that from a rear perspective, the performance of both models is relatively similar. However, from a frontal perspective, CSB-YOLO exhibits superior detection effectiveness in areas where student targets are densely packed and also demonstrates commendable detection capabilities for smaller targets in the back rows of the classroom.

To assess the model's generalization capabilities across similar scenes, we employed models trained on the SCB-Dataset3-S dataset as pretrained weights for transfer learning on the SCB-Dataset3-U dataset, with all models undergoing

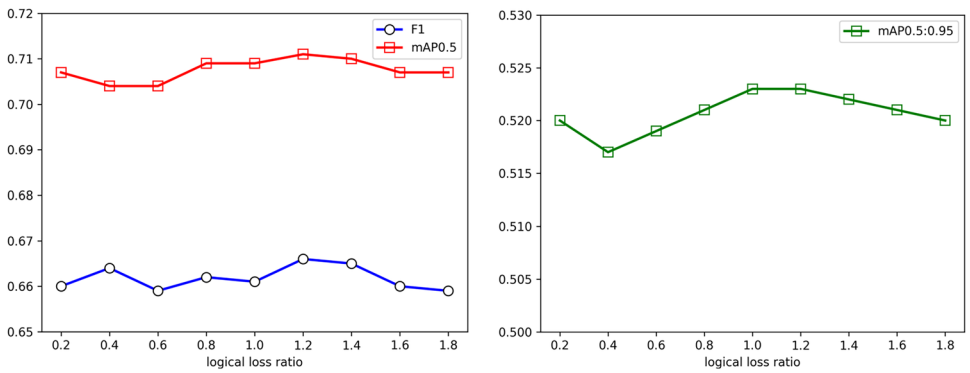

Fig. 11 Accuracy changes under different loss ratios

Table 3 Performance comparisons of different models were conducted on the SCB-Dataset3-S dataset

Method	F1	mAP0.5	mAP0.5:0.95	Params(M)	GFLOPs
YOLOv8n [47]	0.665	0.703	0.52	3.01	8.1
YOLOv8s [47]	0.693	0.733	0.556	11.13	28.4
YOLOv3-tiny [48]	0.605	0.603	0.354	8.67	12.9
YOLOv3 [48]	0.711	0.739	0.552	61.51	154.6
YOLOv5n [49]	0.645	0.673	0.448	1.76	4.1
YOLOv5s [49]	0.683	0.705	0.496	7.02	15.8
YOLOv6n [50]	0.655	0.692	0.511	4.23	11.8
YOLOv7-tiny [27]	0.663	0.701	0.477	6.01	13.0
YOLOv7 [27]	0.726	0.769	0.572	36.49	103.2
YOLOv9-c [51]	0.734	0.787	0.625	50.7	236.6
rtdetr-l [52]	0.692	0.689	0.508	31.99	103.4
CSB-YOLO	0.658	0.703	0.516	1.86	6.2
CSB-YOLO (prune+distill)	0.666	0.711	0.523	0.72	4.3

Table 4 Comparisons of the accuracy across various categories for different models on the SCB-Dataset3-S dataset

	mAP0.5			mAP0.5:0.95		
Method	raising	reading	writing	raising	reading	writing
YOLOv8n	0.796	0.737	0.577	0.57	0.56	0.431
YOLOv8s	0.831	0.758	0.611	0.611	0.589	0.468
YOLOv3-tiny	0.671	0.632	0.505	0.362	0.392	0.308
YOLOv3	0.837	0.739	0.641	0.598	0.568	0.49
YOLOv5n	0.78	0.712	0.526	0.496	0.488	0.36
YOLOv5s	0.81	0.728	0.577	0.547	0.534	0.407
YOLOv6n	0.804	0.71	0.562	0.574	0.538	0.421
YOLOv7-tiny	0.795	0.732	0.577	0.509	0.516	0.404
YOLOv7	0.842	0.778	0.688	0.599	0.595	0.521
YOLOv9-c	0.873	0.795	0.692	0.673	0.645	0.558
rtdetr-l	0.814	0.702	0.552	0.566	0.529	0.428
CSB-YOLO	0.803	0.727	0.578	0.568	0.547	0.433
CSB-YOLO(prune+distill)	0.803	0.736	0.593	0.574	0.554	0.441

100 training cycles. The results, as detailed in the Table 5, show that the CSB-YOLO model introduced in this paper is slightly lower in the mAP0.5 accuracy metric compared to the baseline. However, it achieves a significant 5.4% increase in the F1 score, while its parameter count and computational load remain considerably less than those of the baseline. Integrating three accuracy metrics, CSB-YOLO still outperforms all other small models involved in the comparison, thoroughly demonstrating its generalization capability in classroom settings.

In addition to the accuracy testing mentioned above, we also conducted lightweight testing on low-performance CPU devices and Raspberry Pi 5 to evaluate the deployment feasibility of CSB-YOLO on low-performance devices. As shown in Table 6, the experiments demonstrate that CSB-YOLO, following pruning and distillation, has the smallest model file size among all comparison models, at just 2.95MB. Moreover, when running inference on a CPU, it achieves an FPS of 37.17, slightly lower than YOLOv5n but higher than all other comparison models. It's important to note that CSB-YOLO's accuracy significantly surpasses that of YOLOv5n. Furthermore, an FPS of 37.17 is more than sufficient for real-time detection needs. Therefore, CSB-YOLO offers a higher cost-effectiveness, making it highly suitable for deployment on low-performance devices in classroom settings.

Fig. 12 Visualization of detection results for YOLOv8n and CSB-YOLO

Table 5 Performance comparisons of different models were conducted on the SCB-Dataset3-U dataset

Method	F1	mAP0.5	mAP0.5:0.95	Params(M)	GFLOPs
YOLOv8n	0.707	0.75	0.576	3.01	8.1
YOLOv8s	0.801	0.935	0.723	11.13	28.4
YOLOv3-tiny	0.585	0.535	0.335	8.68	12.9
YOLOv3	0.75	0.699	0.553	61.53	154.6
YOLOv5n	0.519	0.434	0.312	1.77	4.2
YOLOv5s	0.639	0.612	0.46	7.03	15.8
YOLOv6n	0.695	0.741	0.57	4.23	11.8
YOLOv7-tiny	0.729	0.688	0.502	6.02	13.1
YOLOv7	0.807	0.765	0.605	36.51	103.2
YOLOv9-c	0.827	0.944	0.771	50.7	236.7
rtdetr-l	0.747	0.695	0.528	31.99	103.5
CSB-YOLO	0.761	0.747	0.576	1.86	6.2

Table 6 Comparison of lightweight metrics among different models

Method	Onnx file size(MB)	FPS(CPU)	FPS(Raspberry Pi5)
YOLOv8n	11.6	24.84	5.09
YOLOv8s	42.6	9.42	1.95
YOLOv3-tiny	33.1	18.24	4.23
YOLOv3	235.0	2.14	0.37
YOLOv5n	7.15	44.2	8.08
YOLOv5s	27.1	15.53	3.22
YOLOv6n	16.3	23.39	5.98
YOLOv7-tiny	22.9	21.8	4.76
YOLOv7	139.0	2.4	0.5
YOLOv9-c	193.0	1.15	0.22
rtdetr-l	122.0	2.38	0.5
CSB-YOLO	7.33	31.46	6.08
CSB-YOLO(prune+distill)	2.95	37.17	7.15

4.8 Experimental comparison of different detection heads

We compared various detection head structures based on YOLOv8n, and the results are shown in Table 7. Our designed detection head structure exhibits the smallest parameter count, computational load, model file size, and the best real-time performance among the compared detection heads. The experiments provide ample evidence that our designed ERD Head achieves excellent lightweight performance.

Table 7 The comparative experiments of different detection heads on the SCB-Dataset3-S dataset

Head	mAP0.5	Params(M)	GFLOPs	Onnx(MB)	FPS(CPU)
Original detection head [47]	0.703	3.01	8.1	11.6	24.84
Detect Aux [27]	0.697	3.01	8.1	11.6	25.61
Dynamic Head [53]	0.722	3.49	9.6	13.5	19.52
RTDETR Decoder [52]	0.646	9.48	16.7	36.5	5.61
P2 Head [47] [31]	0.7	2.92	12.2	11.8	13.7
ERD Head (ours)	0.695	2.64	6.6	10.2	31.6

Table 8 Module ablation experiment

BiFPN	C2f_SCConv	ERD Head	Prune	Distill	mAP0.5	Params(M)	GFLOPs	Onnx(MB)	FPS(CPU)
-	-	-	-	-	0.703	3.01	8.1	11.6	24.84
+	-	-	-	-	0.701	1.99	7.1	7.81	28.92
-	+	-	-	-	0.703	3.47	8.6	13.4	25.11
-	-	+	-	-	0.695	2.64	6.6	10.2	31.6
+	-	-	-	-	0.703	2.08	7.4	8.18	26.18
+	-	+	-	-	0.692	1.77	5.9	6.96	31.43
-	+	+	-	-	0.701	3.1	7.1	12.0	25.59
+	+	+	-	-	0.703	1.86	6.2	7.33	31.46
+	+	+	+	-	0.704	0.72	4.3	2.95	37.05
+	+	+	+	+	0.711	0.72	4.3	2.95	37.17
+	+	+	-	+	0.714	1.86	6.2	7.33	31.19
+	+	-	+	+	0.709	1.1	5.5	4.43	31.42
+	-	+	+	+	0.709	0.78	4.4	3.17	37.21
-	+	+	+	+	0.712	2.18	6.4	8.51	31.56

(a) C2f_SCConv (b) C2f

Fig. 13 Heatmap comparison between C2f_SCConv and C2f

4.9 Ablation experiment

To validate the effectiveness of each enhancement module in terms of both lightweight design and accuracy improvement, we conducted ablation experiments on the baseline model YOLOv8n, incorporating various improvement modules. All experiments were conducted on the SCB-Dataset3-S dataset, with each training session spanning 300 epochs. The results, as presented in Table 8, show that after the integration of BiFPN, there was a slight decrease in mAP0.5 by 0.2%. However, there was a substantial reduction in parameter count, computational load, and model file size, along with a 4.08 increase in FPS. This indicates that the BiFPN structure significantly contributes to the model's lightweight design without compromising accuracy. Building upon BiFPN with the addition of ERD Head led to a further decrease in accuracy, but also resulted in additional reductions in parameter count, computational load, and model file size, while FPS increased to 31.43. This effectively demonstrates the ERD Head's capability to reduce model complexity and enhance computational speed.

To counteract the precision loss caused by the ERD Head, we introduced the C2f_SCConv module, which successfully raised the mAP0.5 to the level of the baseline. As observed in Fig. 13, replacing C2f with C2f_SCConv allows the network to accurately concentrate attention on the students within the scene. This improvement is due to the larger receptive field of C2f_SCConv, which significantly bolsters the network's ability to represent features of irregular human targets that occlude each other, enhancing detection accuracy in complex classroom environments.

After further applying LAMP pruning to the model, the parameter count was reduced to 0.72M, merely 23.9% of the baseline, without compromising accuracy. Concurrently, the FPS increased to 37.17. Ultimately, BCKD knowledge distillation was performed on the pruned model, boosting the mAP0.5 to 71.1%, which is a 0.8% increase from the baseline.

5 Conclusion

This paper proposes a detection model named CSB-YOLO, which is specifically tailored for the detection of student behaviors in classroom settings. Designed to operate efficiently even in crowded classroom scenarios, this model has been optimized for lightweight deployment, allowing it to be easily and cost-effectively implemented on devices with limited computational capabilities typically found in classrooms. CSB-YOLO employs a BiFPN structure, replacing YOLOv8's Neck structure, to reduce parameter size while enhancing feature fusion capabilities. This optimization enables the model to achieve accuracy levels comparable to YOLOv8n with fewer parameters. Additionally, we have designed a novel ERD Head, which significantly reduces the model's parameter count and computational requirements while accelerating the model's inference speed. To further address accuracy concerns stemming from lightweight design, we integrate SCConv into the C2f module, creating the C2f_SCConv structure, thus enhancing the model's ability to represent human features. Employing LAMP pruning drastically reduces parameter size and computational requirements, resulting in an Onnx file size of only 2.95MB and an improved inference speed with an FPS of 37.17. Knowledge distillation further enhances the pruned model's performance. Comparative testing on the SCB-Dataset3-S dataset demonstrates that CSB-YOLO achieves an mAP0.5 of 71.1%, marking a 0.8% increase over YOLOv8n, with parameter count and computational load only 23.9% and 53% of the latter, respectively. The lightweight design of CSB-YOLO ensures ease of deployment in real-world settings while maintaining high accuracy, meeting the demands for real-time student behavior detection in educational environments.

Acknowledgements This paper was supported by the Yunnan Province Wu Zhonghai Expert Workstation Project (No. 202305AF150045) and Yang Zhijun's Industry Innovation Talents Project of Yunnan Xingdian Talents Support Plan (Certificate No.: YNWR-CYJS-2020-017).

Data availability The data used in this study are publicly available.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Yang, B., Yao, Z., Lu, H., Zhou, Y., Xu, J.: In-classroom learning analytics based on student behavior, topic and teaching characteristic mining. Pattern Recogn. Lett. **129**, 224–231 (2020)
2. D'Mello, S.K., Lehman, B., Person, N.: Monitoring affect states during effortful problem solving activities. Int. J. Artif. Intell. Educ. **20**(4), 361–389 (2010)
3. Su, X., Wang, W.: Recognition and identification of college students\classroom behaviors through deep learning. IEIE Transactions on Smart Processing & Computing **12**(5), 398–403 (2023)
4. Liu, S., Zhang, J., Su, W.: An improved method of identifying learner's behaviors based on deep learning. J. Supercomput. **78**(10), 12861–12872 (2022)
5. 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, pp. 580–587 (2014)
6. Girshick, R.: Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, pp. 1440–1448 (2015)
7. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems 28 (2015)
8. 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, pp. 779–788 (2016)
9. 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, pp. 2980–2988 (2017)
10. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: Ssd: Single shot multibox detector, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, Proceedings, Part I 14, Springer, 2016, pp. 21–37 (2016)
11. Sultana, F., Sufian, A., Dutta, P.: A review of object detection models based on convolutional neural network, Intelligent computing: image processing based applications 1–16 (2020)
12. Zhao, J., Zhu, H.: Cbph-net: A small object detector for behavior recognition in classroom scenarios, IEEE Transactions on Instrumentation and Measurement (2023)
13. 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, pp. 10781–10790 (2020)
14. Liu, J.-J., Hou, Q., Cheng, M.-M., Wang, C., Feng, J.: Improving convolutional networks with self-calibrated convolutions, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10096–10105 (2020)
15. Lee, J., Park, S., Mo, S., Ahn, S., Shin, J.: Layer-adaptive sparsity for the magnitude-based pruning, arXiv preprint (2020) arXiv: 2010.07611
16. Yang, L., Zhou, X., Li, X., Qiao, L., Li, Z., Yang, Z., Wang, G., Li, X.: Bridging cross-task protocol inconsistency for distillation in dense object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 17175–17184 (2023)
17. Zhang, G., Wang, L., Wang, L., Chen, Z.: Hand-raising gesture detection in classroom with spatial context augmentation and dilated convolution. Computers & Graphics **110**, 151–161 (2023)
18. 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, pp. 2117–2125 (2017)

19. Wang, Z., Jiang, F., Shen, R.: An effective yawn behavior detection method in classroom, in: International conference on neural information processing, Springer, pp. 430–441 (2019)
20. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks, Advances in neural information processing systems 29 (2016)
21. Chen, H., Guan, J.: Teacher-student behavior recognition in classroom teaching based on improved yolo-v4 and internet of things technology. Electronics **11**(23), 3998 (2022)
22. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: Yolov4: Optimal speed and accuracy of object detection, arXiv preprint (2020) arXiv:2004.10934
23. Bao, D., Su, W.: Research on the detection and analysis of students' classroom behavioral features based on deep cnns, ACM Transactions on Asian and Low-Resource Language Information Processing (2023)
24. 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, pp. 1580–1589 (2020)
25. Wang, Z., Li, L., Zeng, C., Yao, J.: Student learning behavior recognition incorporating data augmentation with learning feature representation in smart classrooms. Sensors **23**(19), 8190 (2023)
26. 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), pp. 3–19 (2018)
27. Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M.: Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7464–7475 (2023)
28. Chen, H., Zhou, G., Jiang, H.: Student behavior detection in the classroom based on improved yolov8. Sensors **23**(20), 8385 (2023)
29. Gao, S.-H., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., Torr, P.: Res2net: A new multi-scale backbone architecture. IEEE Trans. Pattern Anal. Mach. Intell. **43**(2), 652–662 (2019)
30. Ouyang, D., He, S., Zhang, G., Luo, M., Guo, H., Zhan, J., Huang, Z.: Efficient multi-scale attention module with cross-spatial learning, in: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 1–5 (2023)
31. Liu, Q., Jiang, R., Xu, Q., Wang, D., Sang, Z., Jiang, X., Wu, L.: Yolov8n_bt: Research on classroom learning behavior recognition algorithm based on improved yolov8n, IEEE Access (2024)
32. Xiao, G., Xu, Q., Wei, Y., Yao, H., Liu, Q.: Occlusion robust cognitive engagement detection in real-world classroom. Sensors **24**(11), 3609 (2024)
33. Jiang, Y., Zhu, X., Wang, X., Yang, S., Li, W., Wang, H., Fu, P., Luo, Z.: R2cnn: Rotational region cnn for orientation robust scene text detection, arXiv preprint (2017) arXiv:1706.09579
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, pp. 8759–8768 (2018)
35. Ding, X., Zhang, X., Han, J., Ding, G.: Diverse branch block: Building a convolution as an inception-like unit, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10886–10895 (2021)
36. Ding, X., Guo, Y., Ding, G., Han, J.: Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 1911–1920 (2019)
37. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 13733–13742 (2021)
38. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, pmlr, pp. 448–456 (2015)
39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9 (2015)
40. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826 (2016)
41. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 31, (2017)
42. Yang, F., Wang, T.: Scb-dataset3: A benchmark for detecting student classroom behavior, arXiv preprint (2023) arXiv:2310.02522
43. Zhou, Z., Zhuge, C., Guan, X., Liu, W.: Channel distillation: Channel-wise attention for knowledge distillation, arXiv preprint (2020) arXiv:2006.01683
44. Yang, Z., Li, Z., Shao, M., Shi, D., Yuan, Z., Yuan, C.: Masked generative distillation, in: European Conference on Computer Vision, Springer, pp. 53–69 (2022)
45. Li, Q., Jin, S., Yan, J.: Mimicking very efficient network for object detection, in: Proceedings of the ieee conference on computer vision and pattern recognition, pp. 6356–6364 (2017)
46. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network, arXiv preprint (2015) arXiv:1503.02531
47. Jocher, G., Chaurasia, A., Qiu, J.: YOLOv8, (2023) https://github.com/ultralytics/ultralytics
48. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement, arXiv preprint (2018) arXiv:1804.02767
49. Jocher, G.: YOLOv5, (2020) https://github.com/ultralytics/yolov5
50. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., et al.: Yolov6: A single-stage object detection framework for industrial applications, arXiv preprint (2022) arXiv:2209.02976
51. Wang, C.-Y., Yeh, I.-H., Liao, H.-Y.M.: Yolov9: Learning what you want to learn using programmable gradient information, arXiv preprint (2024) arXiv:2402.13616
52. Zhao, Y., Lv, W., Xu, S., Wei, J., Wang, G., Dang, Q., Liu, Y., Chen, J.: Detrs beat yolos on real-time object detection, arXiv preprint (2023) arXiv:2304.08069
53. Dai, X., Chen, Y., Xiao, B., Chen, D., Liu, M., Yuan, L., Zhang, L.: Dynamic head: Unifying object detection heads with attentions, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7373–7382 (2021)