BLN-YOLO: A Lightweight Neural Network Framework for Enhanced Forest Fire Detection

Xiong Zijian[1*]

([1] Sichuan Polytechnic Technician College, Wenjiang District, Chengdu, Sichuan, 618500, China)

Abstract: This paper introduces BLN-YOLO, a lightweight model improved from YOLOv8n, addressing two key challenges in forest fire detection: poor real-time performance of traditional methods and the accuracy-efficiency trade-off in lightweight architectures. The model employs a three-stage optimization: 1) Integrating Local Spatial Attention (LSKA) in the backbone to enhance fire feature extraction; 2) Adopting a cross-scale BiFPN feature pyramid in the neck for efficient multi-scale fusion and small-target detection; 3) Replacing detection heads with Low-rank Decomposed (LSCD) structures to reduce parameters. For edge deployment, a two-stage compression is proposed: structured pruning via LAMP criteria removes redundant weights, while knowledge distillation transfers teacher model's feature responses and decision boundaries to mitigate accuracy loss. Experimental results show BLN-YOLO achieves 79.4% parameter reduction, 0.6% mAP improvement, 77.4% memory savings, and 11.4% faster inference compared to YOLOv8n, meeting ultra-real-time detection requirements.

Keywords: Forest Fire Detection; Lightweight Model; Detection Head; Feature Pyramid; Real-time Monitoring

1 Introduction

With the growing impact of global climate change, the frequency and scale of forest fires are on the rise, posing severe threats to global ecological security and economic development. Forest fires, among the most dangerous enemies of forests and the most terrifying disasters in agriculture and forestry, can cause devastating consequences to forest ecosystems[1]. As a country with diverse terrains, China has a relatively high forest coverage rate of approximately 23% globally. The significant casualties and economic losses caused by forest fires highlight the crucial importance of timely and effective fire detection. To minimize casualties and economic losses from wildfires, timely and effective forest fire detection is of great significance.

Forest fire detection methods mainly include multi-sensor technology[2], satellite remote sensing[3], and UAV[4]. Multi-sensor technology is limited by the narrow monitoring range of single sensors; large-scale deployment increases costs and easily causes network congestion. Satellite remote sensing, despite wide coverage, has long operational cycles, hindering rapid detection. UAVs or fixed monitoring stations, relying on image data and deep learning algorithms, excel in flexibility and response speed, becoming research focuses. However, their dependent server-side models with heavy computation and numerous parameters are hard to deploy on resource-constrained devices, restricting real-time detection.

In early practices of applying digital image processing to forest fire detection, Thou-Ho et al.[5] pioneered a video processing-based early fire warning method. It first extracted flame- and smoke-related pixel features via RGB color model-based algorithms, then further quantified these features using disorder measurement methods.In 2015, Yuan et al.[6] proposed a UAV-based method for forest fire detection and tracking. This method comprises two core components: front-end information acquisition and back-end image processing. The UAV, as

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence

Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

the front-end tool, captures real-time images of fire scenes from the air. These image data are then transmitted to the ground control center for subsequent processing and analysis.In 2018, Mahmoud et al.[7] proposed an innovative forest fire detection algorithm. It first adopted background subtraction (MRDB) to effectively eliminate interference from moving regions. In 2023, Giwa et al.[8] developed a new flame-characteristic-based color space method, which significantly highlighted differences between flame and non-flame pixels, thereby optimizing fire recognition accuracy. Both studies focused on forest fire detection, aiming to boost detection precision.In 2023, Wang et al.[9] proposed an improved YOLOv7-based model, YOLO-Fire, for forest fire detection. To enhance detection performance, the study introduced several optimization techniques. Firstly, to accelerate detection speed, the model adopted the bottleneck structure of MobileNetV3 and replaced traditional convolutions with depth-wise separable convolutions.In the same year, Li et al.[10] developed YOLO-SCW, an improved YOLOv7 model for forest fire smoke detection. This model integrated SPD-Conv layers into YOLOv7, effectively reducing the loss of small-target features during feature extraction.In the research on lightweight model-based forest fire recognition technology,In 2020, Liu et al.[11] proposed a lightweight forest fire detection model based on SqueezeNet. By replacing traditional convolutional layers with fire modules, the model significantly reduced the number of parameters and computational complexity. Experimental results showed that it achieved an mAP of 85.2% on the forest fire dataset and could run at over 20 frames per second on embedded devices.In 2021, Zhang et al.[12] developed a lightweight forest fire detection method based on EfficientDet. Through compound scaling that simultaneously optimizes network depth, width, and resolution, the method balanced high performance and low computational cost.

This study investigates the lightweight model YOLOv8n and validates its performance. Given that YOLOv8n exhibits suboptimal trade-off between real-time capability and accuracy as well as high parameter count in forest fire detection, this study proposes a novel forest fire recognition model, BLN-YOLO.

In this study, the Local Spatial Key Attention (LSKA)[13] mechanism is incorporated into the SPPF layer within the backbone network of the YOLOv8 model to further enhance the detection accuracy and robustness of the model.To improve detection accuracy for targets of varying sizes and further reduce the model's computational cost, this study replaces the NECK network in YOLOv8n with the Bidirectional Feature Pyramid Network (BiFPN)[14].

To further lightweight the model, this study replaces the original detection head of YOLOv8 with the Lightweight Shared Convolutional Detection (LSCD)[15]. By introducing strategies such as shared convolution, this study significantly reduces the model's parameter scale, thereby achieving model lightweighting.Then,This paper proposes the use of the Layer-Adaptive Magnitude-based Pruning (LAMP)[16] technique to further compress and optimize the BLN-YOLO model, significantly reducing its complexity, improving inference speed, and enhancing its adaptability for deployment on resource-constrained edge devices.Finally, this paper employs knowledge distillation[17][18][19], utilizing the unpruned BLN-YOLO model as the teacher model to guide the training of the pruned student model, thus enabling effective transfer of knowledge from the teacher model to the student model.

2 Materials and Methods

2.1 Materials

This paper collected forest fire images from multiple sources and channels to construct a comprehensive dataset, with special attention to forest fire situations in various regions, different terrain environments and various weather conditions. These images mainly come from news screenshots, on-site photos provided by firefighters, and pictures collected by social

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

workers during field work. Figure 1 shows the specific information of the dataset. To fully verify the generalization ability of the model, based on different terrain conditions, this study selected three typical forest terrains: mountainous areas, plains and hills, and collected fire image data from these areas respectively. Meanwhile, the dataset includes both daytime and nighttime scenes.

Figure 1 Examples of the forest fire dataset

In current research, due to the limited number of datasets related to forest fires and the great difficulty in artificially constructing real fire scenes, this study has conducted data augmentation on the dataset and divided it into training set, validation set and test set to effectively address this issue.In total, 11,800 images were used for experimentation, consisting of 6,600 originally collected images and 5,200 augmented images.To avoid data leakage, augmentation was applied only to the training set.The final dataset was randomly split into training (7,000 images), validation (2,400 images), and test (2,400 images) sets,maintaining a 6:2:2 ratio.Table 1 presents the class distribution of the forest fire dataset used in this study.

Table 1 Training Set and Validation Set Division

Subset	Original	Augmented	Total	Fire	Non-Fire
Training	4,200	2,800	7,000	3,850	3,150
Validation	1,200	1,200	2,400	1,300	1,100
Test	1,200	1,200	2,400	1,250	1,150
Total	6,600	5,200	11,800	6,400	5,400

2.2 Methods

YOLO, short for You Only Look Once[20], is a target detection algorithm first proposed at CVPR in 2016. It redefines the target detection task as a regression problem and directly predicts the location and category of targets from input images through a single end-to-end network architecture. This innovative "detection-as-regression" approach greatly simplifies the detection process, enabling the entire training process to be efficiently completed in one operation. YOLOv8[21] is one of the most mainstream target detection algorithms currently, with excellent detection efficiency.In contrast to YOLOv5[22], YOLOv8 does away with the convolutional configurations in the upsampling phases of both the Feature Pyramid Network (FPN)[23] and Path Aggregation Network (PANet)[24]. Across the entire model, it swaps out the C3 module for the C2F module and employs a decoupled head design in place of the original head architecture.

In forest fire scenarios, more lightweight models can offer faster detection efficiency and are easier to deploy on edge devices. Thus, BLN-YOLO, improved based on the YOLOv8n model, better meets the requirements for real-time detection.

Based on YOLOv8n, BLN-YOLO introduces the LSKA attention mechanism into the SPPF

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

layer to enhance feature extraction capability, adopts the bidirectional cross-scale BiFPN feature pyramid to replace the original neck network for optimizing feature fusion and reducing redundant connections, and uses the LSCD detection head based on low-rank decomposition to achieve lightweight of the detection head. These modules improve the network structure of the YOLOv8 model, enhancing the model's detection capability while making it more lightweight to meet the real-time requirements of forest fire detection. Figure 2 shows the structure of the original YOLOv8n model, and the network structure of BLN-YOLO is shown in Figure 3

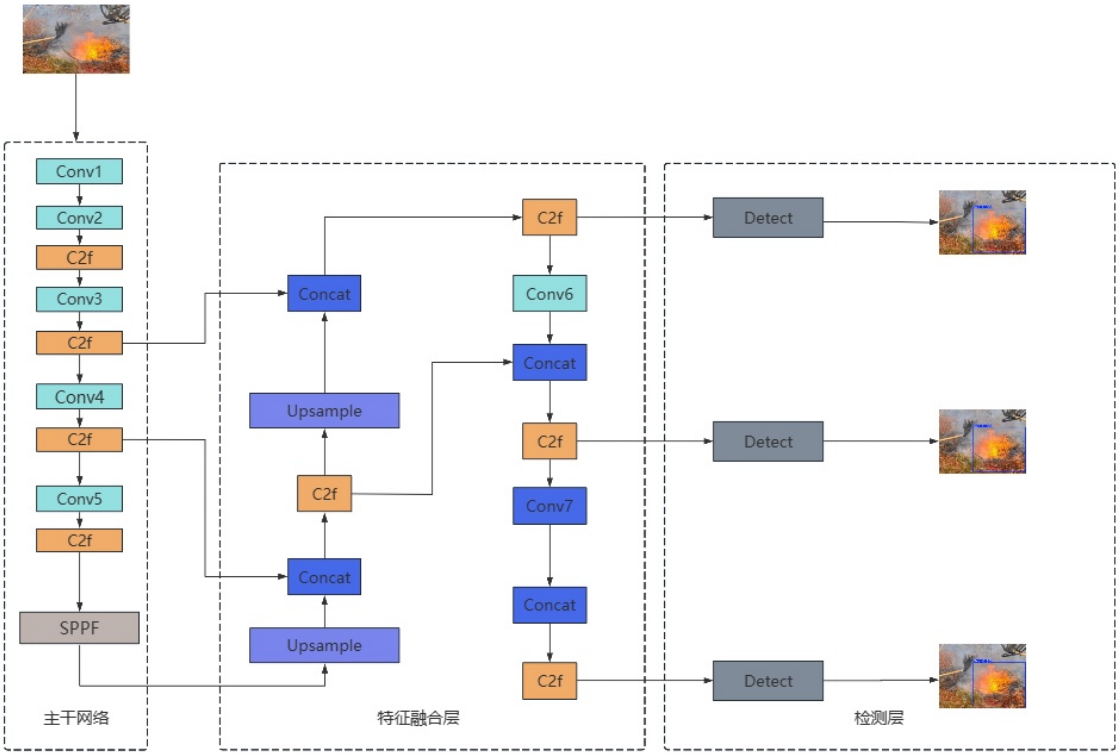


Figure 2 YOLOv8 network structure diagram

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

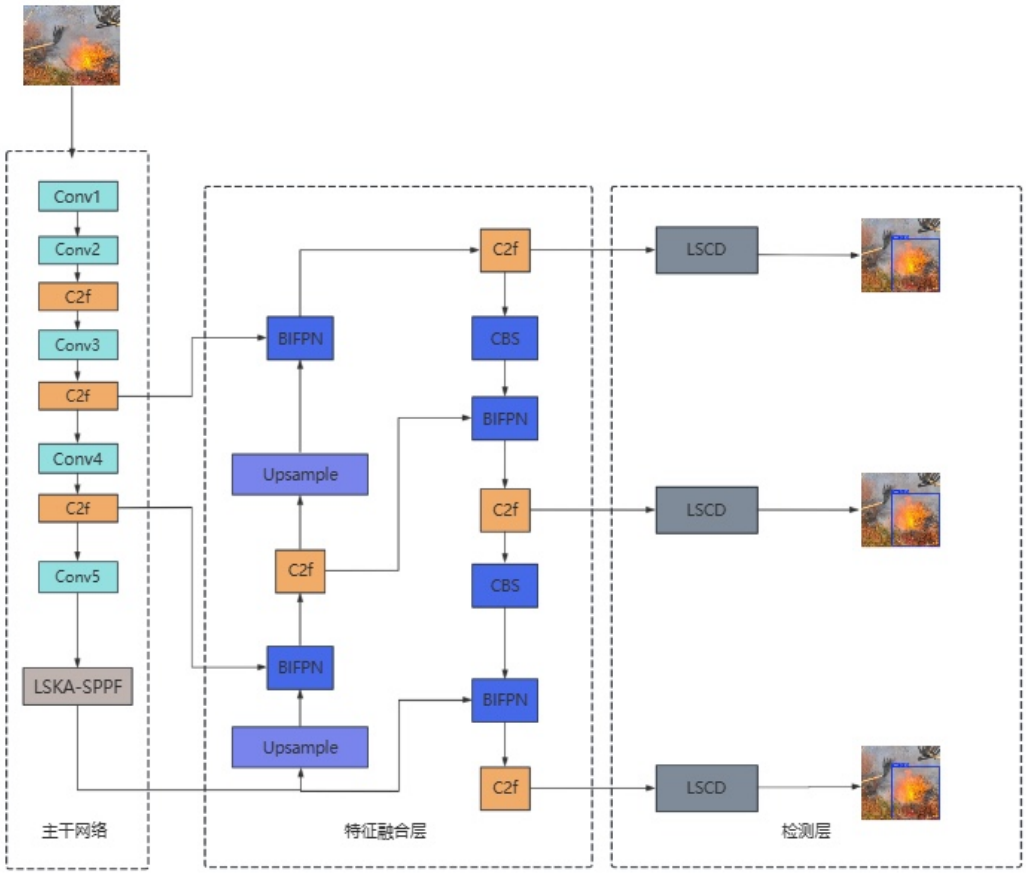


Figure 3 BLN-YOLO network structure diagram

2.2.1 Large Separable Kernel Attention

In the object detection framework of YOLOv8, the Spatial Pyramid Pooling - Fast (SPPF) layer plays a crucial role, whose main function is to achieve effective fusion of multi - scale features. SPPF captures spatial information of different scales by applying pooling windows of different sizes on feature maps at various levels, thereby enhancing the model's ability to adapt to changes in target sizes. First, SPPF performs dimension reduction on the number of channels of the input feature map through 1×1 convolution. This step not only reduces the number of parameters but also improves the expressive ability of the feature map. Then, SPPF uses multiple maximum pooling layers (MaxPool2d) to perform pooling operations of different scales on the feature map. Through carefully designing pooling layers with different kernel sizes, it ensures that while maintaining the spatial dimensions of the feature map, it can effectively integrate feature information of different scales. The pooled feature maps are concatenated in the channel dimension. This process fuses multi - scale feature information into a single feature map, thus providing rich contextual information for subsequent object detection. Finally, the concatenated feature map adjusts the number of channels through another 1×1 convolution layer to match the input requirements of the subsequent layers in the network, and at the same time further fuses features of different scales to generate the final output feature map. The structure diagram of SPPF is shown in Figure 4.

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

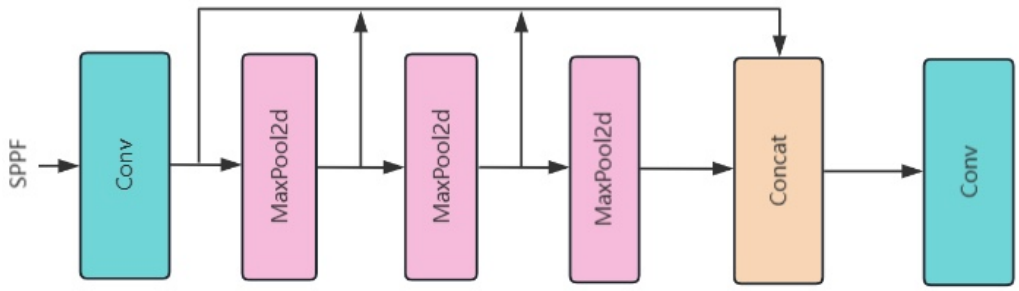


Figure 4 SPPF network structure diagram

Although the SPPF layer in YOLOv8 has the advantages of multi-scale feature extraction and enhancing model expressiveness, it also has some limitations. For example, it may overlook the concentration of attention on specific regions during the feature extraction process, leading to insufficient capture of certain key information. To address this issue, this paper introduces the LSKA attention mechanism into the SPPF layer, which can effectively improve the problem. Through large kernel separable convolution operations and attention mechanisms, LSKA enhances the model's ability to extract spatial and channel information, thereby improving the model's representation capability and the accuracy of object detection. The structure diagram of LSKA is shown in Figure 5

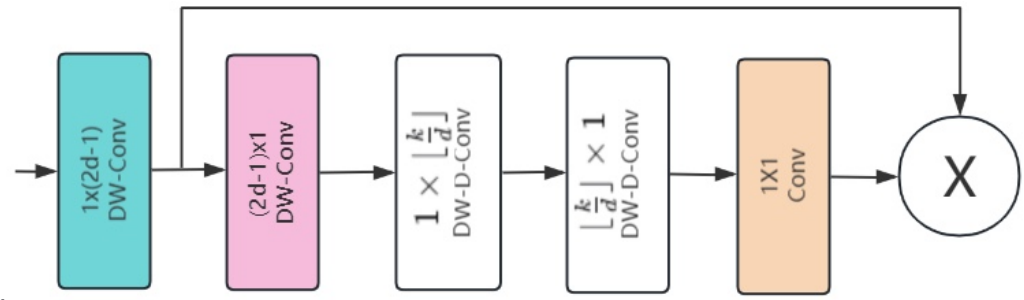


Figure 5 LSKA network structure diagram

In Figure 5, 1×(2d-1) DW-Conv is a depthwise convolution (DW-Conv for short) with a convolution kernel size of 1×(2d-1). This convolution operation convolves the input features in the depth direction, enabling the capture of local spatial information. (2d-1)×1 DW-Conv is another depthwise convolution with a kernel size of (2d-1)×1. This operation performs convolution in the width direction, further capturing local spatial information.The outputs of LSKA, as shown in Equations (1)–(4).

$$M^C = \sum_{H,W} W^C_{(2d-1)\times 1} * \left(\sum_{H,W} W^C_{1\times(2d-1)} * \mathrm{O}^C\right) \#(1)$$

$$Z^C = \sum_{H,W} W^C_{[\frac{k}{d}]\times 1} * \left(\sum_{H,W} w^C_{1\times[\frac{k}{d}]} * X^C\right) \#(2)$$

$$A^C = W_{1\times 1} * Z^C \#(3)$$

$$T^C = A^{C \otimes F^C} \#(4)$$

These formulas outline the LSKA attention mechanism: First, split spatial convolutions (e.g., 1×(2d-1) and (2d-1)×1 depthwise convolutions) in M^C and Z^C extract local spatial information from O^C and X^C along length and width, reducing large-kernel computation. Next, a 1×1 convolution (1×1) fuses cross-channel features in Z^C to form A^C, which integrates

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

multi-channel and spatial information. Finally, A^C is element-wise multiplied (Hadamard product, $\otimes$) with attention weights $F^{C[25]}$ to output T^C, enhancing key information and focusing the model on useful content.

In this study, the LSKA layer is designed to be placed in the SPPF structure after the concatenation of tensors that have undergone three MaxPool2d operations, as shown in Figure 6.

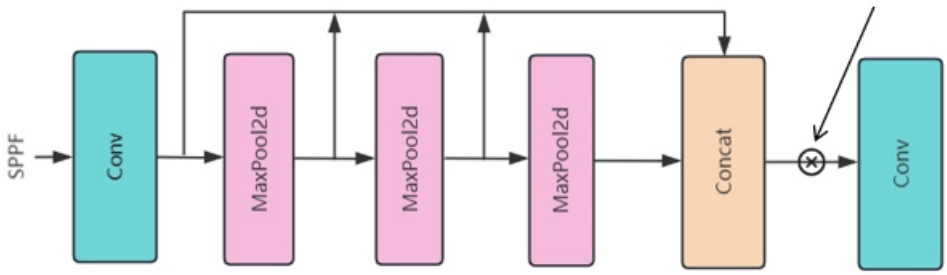


Figure 6 LSKA_SPPF network structure diagram

2.2.2 BIFPN

In the field of object detection, feature fusion is one of the key technologies to improve model performance. The Feature Pyramid Network (FPN) is a popular feature fusion technology, which has also been applied and further optimized in YOLOv8. As shown in Figure 7, the feature pyramid in the neck network of YOLOv8 adopts a PAN-FPN structure similar to that of YOLOv5, namely the Path Aggregation Network (PANet). Through bottom-up and top-down paths, PANet fuses feature maps of different scales, realizes cross-scale information transmission, enhances the model's ability to recognize multi-scale objects, and enables it to locate objects of different sizes more accurately.

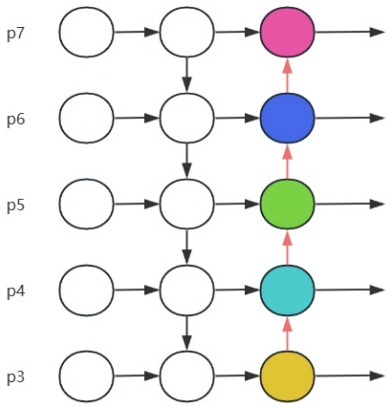


Figure 7 PANet network structure diagram

but also raises the model storage requirements and training difficulty due to the increased number of parameters. To avoid the increase in parameters, inspired by the Bidirectional Feature Pyramid Network (BiFPN), this study adopts its structure to optimize the model, aiming to solve the above problems of PANet. The advantages of BiFPN are as follows: first, it removes nodes connected only to a single input, simplifies the network structure, and realizes effective integration of multiple features while reasonably controlling the amount of computation; second, it directly leads an additional path from the original input to the output node at the same level, enhancing feature fusion; third, it introduces a weighted fusion mechanism, which can adaptively learn the importance of different features, and this is a significant difference from PANet.Figure 8 shows the network structure diagram of BiFPN.

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

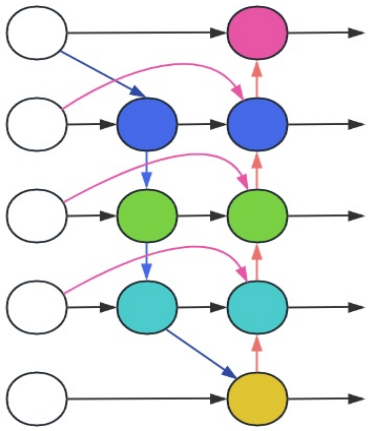


Figure 8. BIFPNnet network structure diagram

Unlike PANet, which only adopts a unidirectional design of top-down and bottom-up paths, BiFPN innovatively constructs reusable bidirectional path modules (including both top-down and bottom-up directions). By iterating these modular feature network layers multiple times, it achieves multi-level and in-depth feature fusion and interaction. The formulas for the two fusion processes (top-down and bottom-up) of BiFPN are shown in Equations (5) and (6).

$$P_6^{\text{td}} = \text{Conv}\left(\frac{\omega_1 \cdot P_6^{in} + \omega_2 \cdot P_7^{in}}{\omega_1 + \omega_2 + \varepsilon}\right)\#(5)$$

$$P_6^{out} = \text{Conv}\left(\frac{\omega_1' \cdot P_6^{in} + \omega_2' \cdot P_6^{td} + \omega_3' \cdot \text{resize}\left(P_5^{out}\right)}{\omega_1' + \omega_2' + \omega_3' + \varepsilon}\right)\#(6)$$

In the BiFPN feature fusion mechanism, for the intermediate feature P_6^{td} of the 6th layer in the top - down path, its input comes from the input P_6^{in} of the current level and the down - sampled feature P_7^{in} of the upper level. For the output feature P_6^{out} of the 6th layer in the bottom - up path, it integrates the input P_6^{in} of the current level, the intermediate feature P_6^{td} of the same level, and the up - sampled feature P_5^{out} of the lower level. ω and ω' enable each path to perform feature transformation using an independent weight matrix, and the other layers are also constructed following a similar cross - scale fusion logic. This multi - level reuse "+" weighted fusion mechanism allows the network to achieve more refined and effective feature fusion while maintaining computational efficiency. It not only improves the fusion efficiency but also enhances the model's ability to learn features of different scales, which is of great significance for visual tasks such as object detection.

2.2.3 LSCD

The detection head of YOLOv8 is a key part of the model, but the BatchNorm (batch normalization) processing method adopted by YOLOv8 is difficult to adapt to forest fire detection: lightweight models need to be deployed on limited resources and often use small batch sizes. However, the performance of BatchNorm depends on the statistical characteristics of batch data. A small batch size will make the mean and variance unable to accurately reflect the dataset distribution, which easily causes feature distribution shifts and weakens the ability to stably recognize forest fire features. The LSCD lightweight detection head uses GroupNorm (group normalization) instead of BatchNorm, taking advantage of GroupNorm and shared convolution. It can minimize computational load and complexity while maintaining effective fusion of feature information, and has advantages such as better performance in fire scenarios

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

and reduced dependence on small batch sizes.

Figure 9 is a diagram showing the operation method of GroupNorm.In Figure 9, N represents the data batch axis, C the channel axis, and H, W the spatial axes. For an image with an input size of [N, C, H, W], it first divides the channels into several groups (like the blue parts in the figure), calculates the variance and mean within each group, and then normalizes the data within the group based on these values. Its calculation range depends on the number of channels C rather than the batch size N, so it is not dependent on the batch size. GroupNorm is very practical in large - scale computer vision applications such as object detection and video classification when computer memory is limited or a small sample size needs to be set.

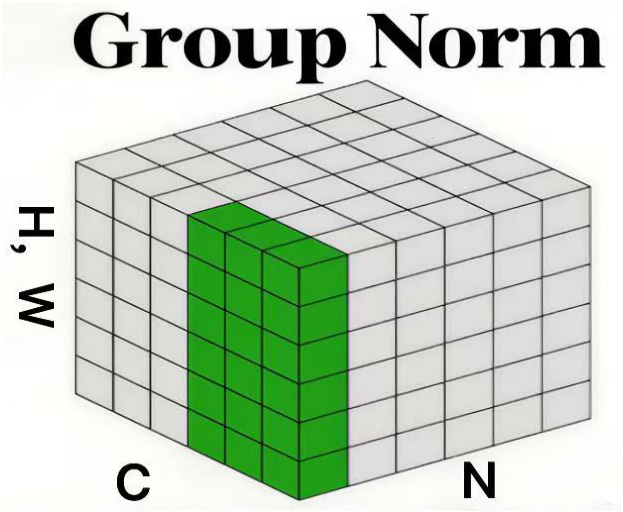


Figure 9 The operation method of GroupNorm

Figure 10 shows the architecture of the LSCD detection head. The GN_Conv1×1 component is a combination of GroupNorm and 1×1 convolution, where "1×1" refers to the size of the convolution kernel. In the architecture, there are two yellow GN_Conv3×3 modules with the same weights; there are also three blue Conv_Box modules and three red Conv_Cls modules, with each module sharing weights internally. Each Conv_Box module is followed by a Scale module, which applies a scaling factor to meet the detection needs of objects of different sizes. In this way, the LSCD detection head can effectively handle multi-scale object detection tasks.

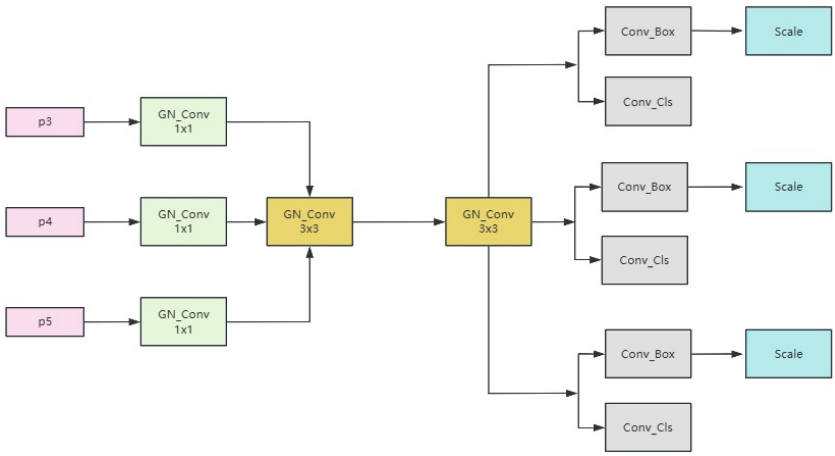


Figure 10 LSCD netstructure diagram

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

2.2.4 LAMP pruning

Although the BLN-YOLO model in this experiment uses a new lightweight design, the improvement in FPS (frames per second) is not as significant as expected. To address this, this study plans to introduce sparsity-based LAMP pruning to further optimize the model and significantly boost FPS. LAMP pruning is a sparsity-based neural network optimization technique. It evaluates the importance of each weight and selectively removes those contributing less to the model output, thus reducing model complexity and computation. The formula for calculating weight importance scores in LAMP pruning is shown in Equation (7).

$$\text{score} \quad (u; w) := \frac{(w[u])^2}{\sum_{v \geq u} (w[v])^2} \#(7)$$

score(u; w) represents the score of node u under weight matrix w. w[u] denotes the weight value of node u in weight matrix w. It represents the sum of squared weights from node v to all subsequent nodes (including u itself, which may refer to the range from u to leaf nodes in specific contexts).In pruning training, as Speed-up (the proportion of pruned parameters) increases, the model parameters and computation (GFLOPs) drop significantly, with the model size reducing accordingly, while mAP@0.5 decreases sharply. This indicates that although pruning reduces model complexity, it causes performance loss. Especially when Speed-up reaches 1.35, the model size and computation shrink drastically, and mAP@0.5 also falls significantly, as shown in Table 2. After pruning, the model parameters and computation decrease noticeably, but the mAP value drops sharply, affecting performance. To address this, this study adjusts parameters through callback training to restore or improve performance. On the pruned model, training continues with a smaller learning rate to ensure the compact and efficient model still achieves optimal performance in specific tasks. The results of callback training are shown in Table 3.

Table 2 The corresponding model parameters for different Speed-up values

Speed-up	mAP50(%)	Parameter（M）	GFLOPs	size(MB)
1.0	85.8	1.81	6.1	3.5
1.05	60.6	1.49	5.8	3.1
1.10	67.3	1.19	5.6	2.5
1.15	60.3	0.96	5.3	2.1
1.20	50.8	0.81	5.1	1.8
1.25	43.4	0.71	4.9	1.6
1.30	35.7	0.65	4.8	1.5
1.35	30.9	0.63	4.7	1.4

Table 3 Comparison chart of model callbacks

Model	mAP50(%)	Parameter（M）	GFLOPs	size(MB)
before-finetune	30.9	0.63	4.7	1.4
After-fintune	86.7	0.63	4.7	1.4

As shown in Table 3, "before-finetune" refers to the BLN-YOLO model after pruning but without callback training, with the mAP value dropping to 30.9%; "after-finetune" is the BLN-YOLO model after pruning and callback training, with the mAP value rebounding to 86.7%.

2.2.5 knowledge distillation

In this paper, the BLN-YOLO model before pruning is used as the teacher model, and the pruned model as the student model. Knowledge transfer is achieved through a dual loss

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence

Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

mechanism: on the one hand, the error loss between the student model's predictions and the real labels is calculated; on the other hand, the matching loss between the student's output and the teacher's soft labels is measured. When the prediction distribution of the student model tends to be consistent with the distribution of the teacher's soft labels, it indicates that it has effectively inherited the teacher's feature expression ability.

As shown in Figure 11, the input data is processed by the teacher model (blue dashed box) and the student model (green dashed box respectively). After passing through the Softmax layer, soft labels, soft predictions and hard labels are generated. The loss of soft labels is calculated by the distillation loss function, and the loss of hard labels by the student loss function, which jointly guide the learning of the student model.

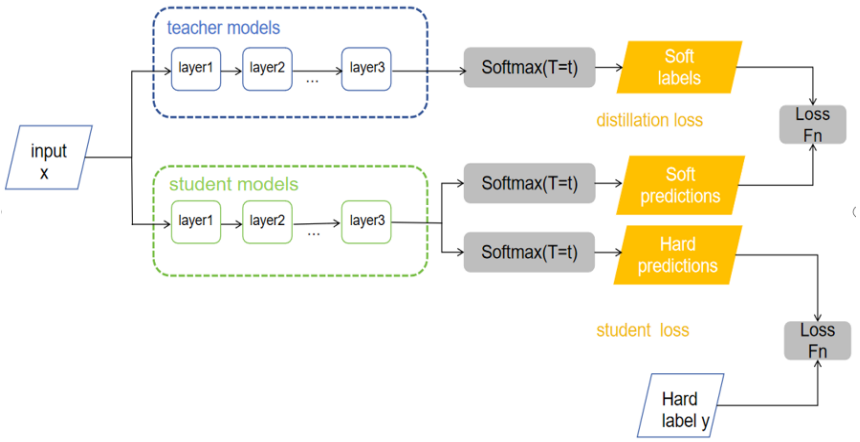


Figure 11 The schematic diagram of knowledge distillation architecture

In knowledge distillation, first, the teacher model is trained to accurately capture patterns and correlations in the data, usually using standard neural network training methods and loss functions (such as cross-entropy loss), as shown in Equation (8). Then, the trained teacher model is used to generate soft labels for input data. Next, the student model is trained. During training, the student model not only minimizes the cross-entropy loss with real labels (Hard Labels), as shown in Equation (9), but also minimizes the distillation loss with the teacher model's soft labels, as shown in Equation (10). Finally, the student model's parameters are updated by optimizing the total loss function, which is the weighted sum of cross-entropy loss and distillation loss, as shown in Equation (11).

$$p_{\text{teacher}}^{T} = \frac{\exp\left(z_{\text{teacher}}/T\right)}{\sum_{j} \exp\left(z_{\text{teacher}}/T\right)} \#(8)$$

$$L_{CE} = -\sum_{i} y_{\text{true},i} \log\left(p_{\text{student},i}^{T=1}\right) \#(9)$$

$$L_{KD} = T^{2} \cdot D_{KL}\left(p_{\text{teacher}}^{T} \parallel p_{\text{student}}^{T}\right) = T^{2} \cdot \sum_{i} p_{\text{teacher},i}^{T} \log\left(\frac{p_{\text{teacher},i}^{T}}{p_{\text{student},i}^{T}}\right) \#(10)$$

$$L = \alpha \cdot L_{CE} + (1-\alpha) \cdot L_{KD} \#(11)$$

The knowledge distillation results are shown in Table 4, where BLN (D) refers to the model obtained after feature distillation on BLN (P). The optimized BLN (D) model has significantly improved object detection accuracy, effectively compensating for the accuracy degradation caused by pruning. LAMP channel pruning reduces model parameters and computational

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

requirements by eliminating redundant channels; while the soft labels generated by the teacher network contain rich feature correlation information, which can help the student model capture deep connections in the data and enhance data understanding ability.

Table 4 Comparison of the model before and after knowledge distillation

Model	mAP50(%)	Parameter（M）	GFLOPs	Size(MB)	FPS
Yolov8n	86.4	3.01	8.1	6.2	130
BLN（P）	86.7	0.63	4.7	1.4	144
BLN（D）	87.0	0.62	4.6	1.4	147

2.3 Experimental environment and rparameter settings

The experiments were run on the Windows 10 system, with an Intel(R) Xeon(R) Platinum 8352 V CPU @ 2.10 GHz (12 vCPUs) and a high-performance NVIDIA GeForce RTX 4090 GPU with 24 GB memory. The deep learning framework used was Python 3.8 with CUDA 11.8. All input images were uniformly resized to 640×640 pixels. Table 5 details the key hyperparameter settings for the model training in this study, and all experiments were conducted under the same parameter settings.

Table 5. Hyperparameters in the Model Training Process

Type	Set
Epoch	300
Batchsize	32
Learning rate1	0.01
Learning rate2	0.0001
Workers	4

2.4 Evaluation of performance metrics

In this paper, the evaluation of performance metrics is conducted through a comprehensive consideration of four key dimensions: the model's mean average precision (mAP@0.5), parameter count, computational complexity, and FPS. The formulas are shown in (11)-(13). In formula (11), AP_i represents the average precision of class i, which is calculated as the number of correctly detected targets (TP) for that class divided by the sum of the number of correct detections, false detections (FP), and the total number of real targets (N) in that class, and it measures the detection accuracy. In formula (12), mAP@0.5, a core performance metric in the field of target detection, measures the average detection accuracy of the model under different Intersection over Union (IoU) thresholds. An IoU threshold of 0.5 means that the overlapping area between the model-predicted bounding box and the real bounding box is at least 50%. The parameter count is directly related to the model's complexity and deployment feasibility, reflecting the total number of trainable parameters in the model. A smaller parameter count makes the model easier to deploy in resource-constrained environments. Computational complexity, typically measured in GFLOPs (billions of floating-point operations), quantifies the number of floating-point operations required for the model to perform one forward propagation, serving as a crucial indicator for evaluating the model's computational efficiency. FPS refers to the number of image frames processed per second; a higher FPS indicates a faster image processing speed, with its calculation formula shown in (13).

Through the comprehensive evaluation of these performance metrics, this study aims to achieve the optimal balance between model accuracy and efficiency, so as to meet the

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

practical needs in different application scenarios.

$$AP_i = \frac{\frac{TP}{TP+FP}}{N} \#(11)$$

$$mAP = \frac{\sum_{i=1}^{Q} AP_i}{Q} \times 100\% \#(12)$$

$$FPS = \frac{1000}{\text{speed}} \#(13)$$

3 Results

3.1 1ablation experiment

First, the integration of the LSKA module into the model increases the mAP@0.5 to 87%, but it also leads to an increase in the number of parameters, computational load, and model size. Second, the incorporation of the Bifpn module effectively reduces the parameter count by 34% and the computational load by 13% while keeping the mAP@0.5 almost unchanged, and it significantly reduces the model size compared with YOLOv8n. Third, the introduction of the LSCD module improves the mAP@0.5 by 0.6% on the basis of YOLOv8, while achieving multi-dimensional optimization—specifically, the computational load decreases by 22% from 3.01 M to 2.36 M, and the parameter count and memory size are reduced by 20% and 25%, respectively. Finally, these modules are used in combination: when LSKA is used together with Bifpn, or LSKA is used together with LSCD, the performance is improved with a relatively moderate increase in model complexity; when Bifpn is used together with LSCD, the parameter count, computational load, and memory size are significantly reduced by 54%, 26%, and 47% respectively while maintaining the mAP@0.5. The BLN-YOLO model, formed by the combination of all three modules, increases the mAP@0.5 from 86.4% to 87.3% and achieves a good balance in terms of parameter count (decreasing by 37% from 3.01 M to 1.89 M), computational load (decreasing by 34%), and memory size (decreasing by 38%) (see Table 6 for comparisons of ablation experiments).

Table 6 Performance Comparison of ablation experiment

Model	LSKA	Bifpn	LSCD	mAP50(%)	Parameter (M)	GFLOPs	Size(MB)
YOLOv8n				86.4	3.01	8.1	6.2
	√			87.0	3.28	8.3	6.5
		√		86.0	1.99	7.1	4.0
			√	87.0	2.36	6.5	4.7
	√	√		87.0	2.27	7.5	4.6
	√		√	86.6	2.63	6.7	5.2
		√	√	86.4	1.63	6.0	3.3
	√	√	√	87.3	1.89	6.2	3.9

Figure 12 plots the curves of loss values changing with training epochs for the YOLOv8n baseline model and the improved BLN-YOLO model during training. It can be observed from the curves that the YOLOv8n model shows stable learning performance during continuous iterations, with its loss curve exhibiting good convergence and regularity.The YOLOv8n model converges after about 250 iterations, with its final loss value stabilizing around 1.2. In

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

comparison, the optimized BLN-YOLO model not only has a significant advantage in the number of iterations required for convergence but also further reduces the final loss value, showing better training results.

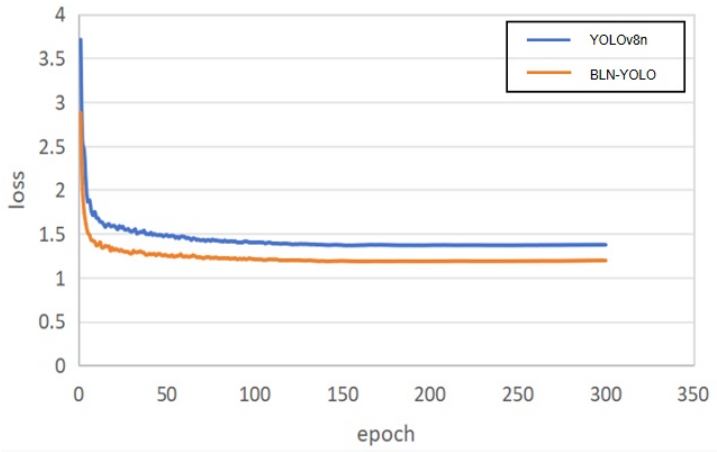


Figure 12 Loss function change curve

Table 7 compares the object detection results of the YOLOv8n and BLN-YOLO models in multi-object and heavy smoke scenarios. It shows that the YOLOv8n algorithm fails to detect some targets (missed detections) and achieves low mAP in multi-object scenarios, while in heavy smoke scenarios, it incorrectly identifies non-targets as targets (false detections).

Table 7 Performance Comparison of ablation experiment

problem	YOLOv8	BLN-YOLO
Missed Detection		
False Detection		
Low map		

3.2 Comparison experiment of different models

In this experiment, six lightweight models—YOLOv8n, YOLOv7-tiny[26], YOLOv9-T[27], YOLOv10n[28], SSD-Lite[29], and Faster-RCNN[30]—were selected for comparison with the improved model. Different models were trained using the same training set, with identical hyperparameter settings during training to ensure experimental fairness.

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

Table 8 presents data obtained using the same dataset. It shows that the lightweight model YOLOv7-tiny is slightly inferior to the baseline model YOLOv8n in this paper, with none of its metrics exceeding those of YOLOv8n. As the lightweight version of YOLOv9, YOLOv9-T has a higher mAP than YOLOv8n, with a slight reduction in parameters; meanwhile, its model size is nearly half that of YOLOv7-tiny in terms of memory usage.YOLOv10n, the lightweight version of YOLOv10, outperforms YOLOv8n in parameters, computation, and memory size but lags behind in mAP@0.5. SSD-Lite and Faster-RCNN adopt lightweight architectures but are overall less effective than the YOLO series. Finally, the improved algorithm shows a significant improvement in forest fire detection performance compared to other algorithms.Figure 13 shows the comparison between the number of parameters of each model and the mAP50 performance.

Table 8 Comparative experiments with different models

Model	mAP50(%)	Parameter（M）	GFLOPs	Size(MB)
YOLOv8n	86.4	3.01	8.1	6.2
YOLOv7-tiny	85.8	6.01	13.2	12.3
YOLOv9-T	87.2	2.01	12.1	6.6
YOLOv10n	86.0	2.27	6.5	5.5
SSD-Lite	79.2	3.40	3.2	13.6
Faster-RCNN	82.2	12.6	28.3	50.4
BLN-YOLO	87.3	1.90	6.2	3.9

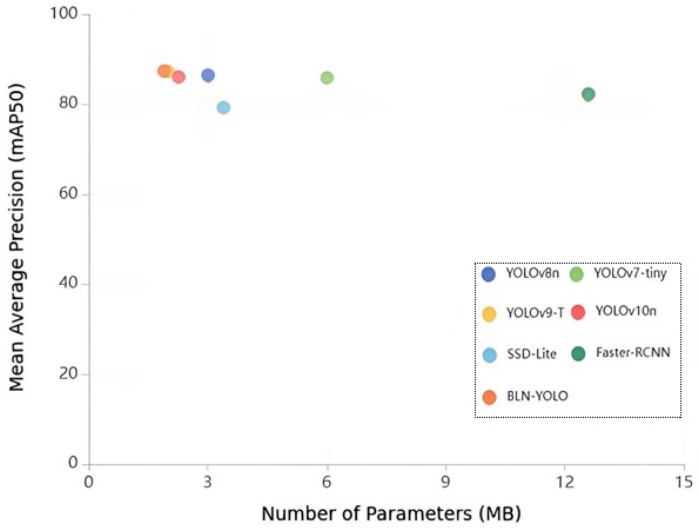


Figure 13 The schematic diagram of knowledge distillation architecture

3.3 Model visualization analysis

Model channel feature visualization technology can clearly show the model's ability to extract features at different levels of images by intuitively displaying feature maps of channel activations. These features include not only basic visual elements (such as edge contours, color contrasts, and texture details) but also deep semantic information. By in-depth analysis of the feature response patterns of each channel, we can accurately identify the weak links in the model's feature extraction, then optimize the network structure in a targeted manner, strengthen the model's ability to capture key features, and ultimately improve the prediction accuracy of object detection tasks. Figure 14 visualizes the channel feature maps of the lightweight model designed in this study, where (a) is the original image from the dataset and (b) is the visualized model channel feature map.

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

(a) (b)

Figure 14 Comparison of the Original Image with the Output Feature Map

To clearly show how the model's output layer focuses attention on different detection targets, this study uses the improved Grad-CAM++[31] algorithm to generate class activation heatmaps. Through gradient-weighted calculations, the algorithm produces high-resolution heatmaps that reflect the spatial positions of target objects and the model's prediction confidence. When overlaid with the original image for visualization, these heatmaps intuitively display the model's attention areas and prediction confidence levels for each detection target. As an upgraded version of the Grad-CAM algorithm, Grad-CAM++ is a gradient-based improved method for model interpretability analysis. It abandons Grad-CAM's approach of assigning equal weights to all elements in gradient feature maps (i.e., obtaining feature map weights by averaging gradients), and instead introduces an additional weighting mechanism to assign different contributions to elements in gradient feature maps, as shown in Equation 14 and 15. Equation 15 is the weight formula for a_{ij}^{kc}.

$$W_k^c = \sum_i \sum_j a_{ij}^{kc} \cdot relu\left(\frac{\partial Y^c}{\partial A_{ij}^k}\right) \#(14)$$

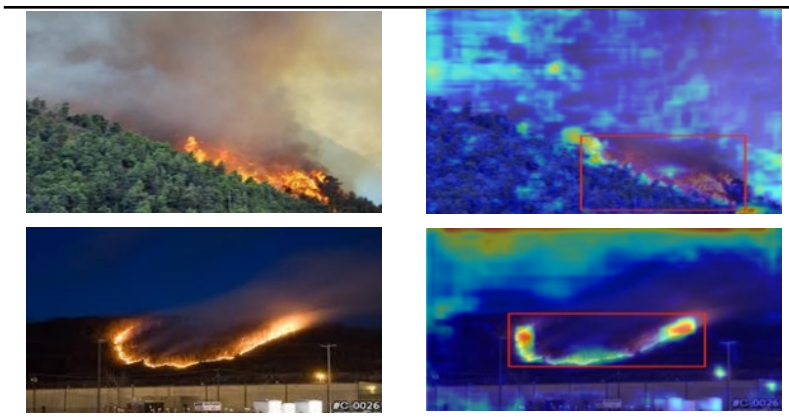
$$a_{ij}^{kc} = \frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \left\{\frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\right\}} \#(15)$$

c represents the current classification target of interest; *k* denotes the k-th feature map channel under analysis; *i* and *j* jointly locate the coordinates of a specific pixel on the feature map of this channel; A^k corresponds to the activation intensity map of the k-th channel; Y^c indicates the model's prediction probability for target *c*, with the requirement that this prediction function must be differentiable. Table 9 shows the visualization results of the Grad-CAM++ algorithm. From the heat distribution map, it can be observed that the lightweight BLN-YOLO model exhibits excellent spatial perception ability in flame target localization, intuitively reflecting the model's mechanism for recognizing and localizing flame features.

Table 9. Visualization effect diagram of Grad-CAM++ algorithm

original image	heatmap

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com



4 Discussion

To address the demand for real-time forest fire detection, this papaer proposes a lightweight neural network model, BLN-YOLO. Through multi-module collaborative optimization and model compression strategies, it solves problems such as high computational complexity of traditional methods and poor adaptability to edge devices. During the experiment on lightweight module design, it was found that although the adoption of the BiFPN feature pyramid reduced the number of model parameters, the FPS (frames per second) decreased, which was inconsistent with our expectations. After investigation, it was revealed that the frequent scale transformations (such as interpolation and convolution adjustment) in the feature fusion process led to higher operational complexity.

Subsequently, we used LAMP structured pruning to directly reduce the actual computational load of the model by directionally removing low-contribution weights and redundant channels, thereby offsetting the computational overhead caused by BiFPN. The schematic diagram of pruning is shown in Figure 15.

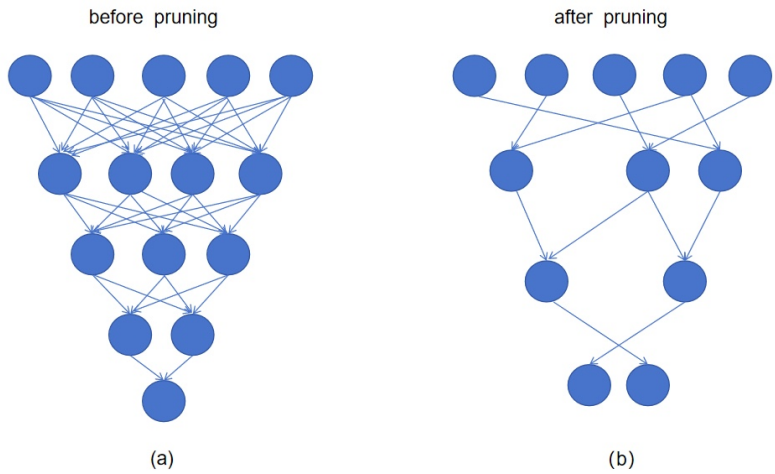


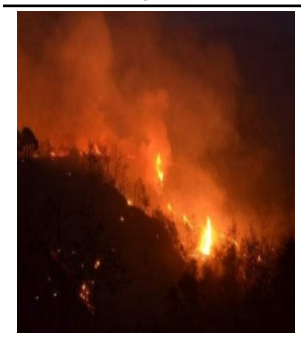
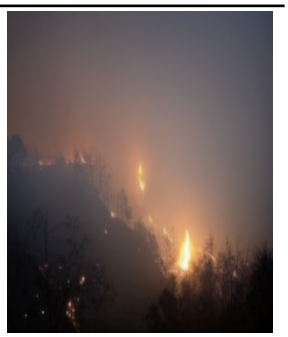
Figure 15 Diagram of Pruning Operation Process

To address the issues of limited forest fire-related datasets and the difficulty in constructing real fire scenarios, this study performed data augmentation on the dataset. In addition to basic ordinary data augmentation, it included complex data augmentation (adopting random erasure and Cutout image processing techniques to generate 4200 image samples) and dark channel[32] prior image dehazing algorithm enhancement (preprocessing and optimizing original samples to tackle problems such as loss of image details, reduced contrast, and difficulty in effectively extracting key texture features caused by haze interference in forest fire images). These data

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

augmentation methods improve the generalization ability and robustness of the model, avoid overfitting, and enable the trained model to better cope with complex situations in practical applications. Table 10 presents the results of the forest fire dataset after being augmented by random erasure, Cutout, and the dark channel prior dehazing algorithm.

Table 10 Comparison Table of Data Augmentation Effects

Original	random erasure	Cutout	dark channel
			

5 Conclusions

In this study, to address the issues of poor real-time performance of traditional methods and the difficulty in balancing accuracy and efficiency of lightweight models in forest fire detection, we propose a lightweight neural network model BLN-YOLO, improved based on YOLOv8n. Through multi-module collaborative optimization and model compression strategies, the overall performance of forest fire detection is enhanced: The introduction of the LSKA attention mechanism into the SPPF layer of the backbone network strengthens the ability to extract fire features; the adoption of the BiFPN feature pyramid optimizes cross-scale feature fusion, improving detection accuracy for multi-scale targets (especially small targets); and the integration of the LSCD detection head achieves a lightweight design, significantly reducing the parameter scale. Additionally, the two-stage compression strategy combining LAMP structured pruning and knowledge distillation substantially reduces model complexity (79.4% parameter reduction and 77.4% memory savings) while effectively mitigating accuracy loss, ultimately achieving a comprehensive performance optimization with 0.6% mAP improvement and 11.4% faster inference speed.

Ablation experiments and comparative experiments demonstrate that BLN-YOLO outperforms mainstream lightweight object detection models in detection accuracy, model lightweighting, and real-time performance, fully verifying the effectiveness of the proposed improvement strategies. This model can meet the deployment requirements of resource-constrained edge devices and provides an efficient and feasible technical solution for real-time forest fire monitoring and early warning. Future research will further expand the diversity and scale of datasets and optimize the robustness of fire detection under complex meteorological conditions to enhance the model's adaptability in practical scenarios.

References:

[1] Kong S, Deng J, Yang L, et al. An attention-based dual-encoding network for fire flame detection using optical remote sensing[J]. Engineering Applications of Artificial Intelligence, 2024, 127: 107238.

[2] Verstockt S, Vanoosthuyse A, Van Hoecke S, et al. Multi-sensor fire detection by fusing visual and non-visual flame features[M]//Lecture Notes in Computer Science: Image and Signal Processing. Berlin: Springer, 2010: 333–341.

[3] Xu G, Zhong X. Real-time wildfire detection and tracking in Australia using geostationary satellite: Himawari-8[J]. Remote Sensing, 2017, 9(10): 1052–1061.

[4] Cruz H, Eckert M, Meneses J, et al. Efficient forest fire detection index for application in unmanned aerial

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence

Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

systems (UASs)[J]. Sensors, 2016, 16(6): 893.
[5] Chen T H, Wu P H, Chiou Y C. An early fire-detection method based on image processing[C]//Proceedings of 2004 International Conference on Image Processing (ICIP 2004). Singapore, 2004: 1707–1710.
[6] Yuan C, Liu Z, Zhang Y. UAV-based forest fire detection and tracking using image processing techniques[C]//International Conference on Unmanned Aircraft Systems. Denver, CO, USA, 2015: 639–643.
[7] Mahmoud M A I, Ren H. Forest fire detection using a rule-based image processing algorithm and temporal variation[J]. Mathematical Problems in Engineering, 2018, 2018: 1–8.
[8] Giwa O, Benkrid A. A new flame-based colour space for efficient fire detection[J]. IET Image Processing, 2023, 17(9): 1229–1244.
[9] 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[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Vancouver, Canada, 2023: 7464–7475.
[10] Li Y, et al. Deep learning-based forest fire smoke monitoring[J]. Forest Resources Management, 2023(4): 1–8 .
[11] Liu P, Yang Q D, Li S B. An efficient fire detection method based on multiscale feature extraction, implicit deep supervision, and channel attention mechanism[J]. IEEE Transactions on Image Processing, 2020, 29: 8467–8475.
[12] Zhang S Y, Zhao J, Ta N, et al. A real-time deep learning forest fire monitoring algorithm based on an improved pruned + KD model[J]. Journal of Real-Time Image Processing, 2021, 18(6): 2319–2329.
[13] Lau S, et al. Large separable kernel attention: Rethinking the large kernel attention design in CNN[J]. arXiv preprint, 2023: arXiv:2309.01439.
[14] Tan M, Pang R, Le Q V. EfficientDet: Scalable and efficient object detection[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA, 2020: 10781–10790.
[15] Yin B. Lightweight fire detection algorithm based on improved YOLOv8[J]. Computer Science and Applications, 2024, 14(9): 47–55 .
[16] Lee N, Ajanthan T, Torr P H S. Layer-adaptive sparsity for magnitude-based pruning[C]//International Conference on Learning Representations (ICLR). Vienna, Austria, 2023.
[17] Liu T, Yang X, Chen C. Normalized feature distillation for semantic segmentation[J]. arXiv preprint, 2022: arXiv:2210.12964.
[18] Zhu K, He Y Y, Wu J, et al. Quantized feature distillation for network quantization[C]//AAAI Conference on Artificial Intelligence. Washington, DC, USA, 2023.
[19] Heo B, Kim J, Yun S, et al. A comprehensive overhaul of feature distillation[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea, 2019: 1921–1929.
[20] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA, 2016: 779–788.
[21] Jocher G. YOLOv8 by Ultralytics[EB/OL]. (2023)[2023-02-15]. https://github.com/ultralytics/ultralytics.
[22] Zhu X, Lyu S, Wang X, et al. TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. Montreal, Canada, 2021: 2778–2788.
[23] Lin T Y, Dollar P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA, 2017: 2117–2125.
[24] 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 (CVPR). Salt Lake City, UT, USA, 2018: 8759–8768.
[25] Lu Y, Sun M. SSE-YOLO: Efficient UAV target detection with less parameters and high accuracy[J]. Preprints, 2024: 2024011108.
[26] 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[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Vancouver, Canada, 2023: 7464–7475.
[27] Wang C Y, Yeh I H, Liao H Y M. YOLOv9: Learning what you want to learn using programmable gradient information[J]. arXiv preprint, 2024: arXiv:2402.13616.
[28] Wang A, Chen H, Liu L, et al. YOLOv10: Real-time end-to-end object detection[J]. arXiv preprint, 2024: arXiv:2405.14458.
[29] Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector[C]//European Conference on Computer Vision (ECCV). Amsterdam, Netherlands, 2016: 21–37.
[30] Girshick R. Fast R-CNN[C]//Proceedings of the IEEE International Conference on Computer Vision (ICCV). Santiago, Chile, 2015: 1440–1448.

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com

[31] Chattopadhyay A, Sarkar A, Howlader P, et al. Grad-CAM++: Improved visual explanations for deep convolutional networks[C]//IEEE Winter Conference on Applications of Computer Vision (WACV). Lake Tahoe, NV, USA, 2018: 839–847.
[32] He K, Sun J, Tang X. Single image haze removal using dark channel prior[C]//IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Miami, FL, USA, 2009: 1956–1963.

About the author: Xiong Zijian, Chengdu, Sichuan, Research interests: Artificial Intelligence
Corresponding author: Xiong Zijian, Email: 1162531550@qq.com