Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.058932

ARTICLE

YOLO-LFD: A Lightweight and Fast Model for Forest Fire Detection

Honglin Wang[1], Yangyang Zhang[2,*] and Cheng Zhu[3]

[1]School of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing, 210044, China

[2]School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, 210044, China

[3]Electrical & Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

*Corresponding Author: Yangyang Zhang. Email: 202312490635@nuist.edu.cn

Received: 24 September 2024 Accepted: 26 November 2024 Published: 17 February 2025

ABSTRACT

Forest fires pose a serious threat to ecological balance, air quality, and the safety of both humans and wildlife. This paper presents an improved model based on You Only Look Once version 5 (YOLOv5), named YOLO Lightweight Fire Detector (YOLO-LFD), to address the limitations of traditional sensor-based fire detection methods in terms of real-time performance and accuracy. The proposed model is designed to enhance inference speed while maintaining high detection accuracy on resource-constrained devices such as drones and embedded systems. Firstly, we introduce Depthwise Separable Convolutions (DSConv) to reduce the complexity of the feature extraction network. Secondly, we design and implement the Lightweight Faster Implementation of Cross Stage Partial (CSP) Bottleneck with 2 Convolutions (C2f-Light) and the CSP Structure with 3 Compact Inverted Blocks (C3CIB) modules to replace the traditional C3 modules. This optimization enhances deep feature extraction and semantic information processing, thereby significantly increasing inference speed. To enhance the detection capability for small fires, the model employs a Normalized Wasserstein Distance (NWD) loss function, which effectively reduces the missed detection rate and improves the accuracy of detecting small fire sources. Experimental results demonstrate that compared to the baseline YOLOv5s model, the YOLO-LFD model not only increases inference speed by 19.3% but also significantly improves the detection accuracy for small fire targets, with only a 1.6% reduction in overall mean average precision (mAP)@0.5. Through these innovative improvements to YOLOv5s, the YOLO-LFD model achieves a balance between speed and accuracy, making it particularly suitable for real-time detection tasks on mobile and embedded devices.

KEYWORDS

Forest fire detection; YOLOv5; lightweight; small object detection

1 Introduction

In recent years, with the increasing demand for fire detection, traditional methods employing smoke sensors, temperature sensors, and similar devices have proven effective in small indoor environments but exhibit significant limitations in large spaces and complex outdoor settings. These sensors typically detect only late-stage fire signals (e.g., changes in smoke concentration and temperature), resulting in insufficient early warning capabilities. This limitation is particularly pronounced in rapidly

spreading forest fires, where the sensors' response speed and sensitivity significantly decline [1]. In contrast, deep learning-based visual detection methods offer a more forward-looking solution for fire detection. By utilizing image processing techniques, visual methods can capture early flame features in real-time. This approach demonstrates enhanced detection capabilities, especially in large-scale and complex environments. Among these methods, the YOLO (You Only Look Once) series of single-stage object detection models has garnered considerable attention for its efficiency and real-time performance [2].

YOLOv5 [3], one of the most popular single-stage detection models, generates detection results directly from input images, bypassing the complex region proposal and classification steps found in two-stage detection models [4]. This makes YOLOv5 highly suitable for real-time deployment on resource-constrained devices such as drones and embedded systems. However, despite the YOLO series' impressive performance in detection speed and accuracy, challenges remain, particularly in handling complex backgrounds and detecting small objects.

To address these challenges, this paper proposes an improved lightweight fire detection model—YOLO-LFD (YOLO Lightweight Fire Detector). By optimizing YOLOv5, we have significantly improved inference speed and the detection accuracy of small flames, while only slightly sacrificing overall detection accuracy. This makes the model particularly suitable for deployment on resource-constrained devices.

The main contributions of this work include:

- Introducing Depthwise Separable Convolutions (DSConv) to significantly reduce computational complexity and model parameters.
- Designing and implementing the Cross Stage Partial (CSP) Bottleneck with 2 Convolutions (C2f-Light) and CSP Structure with 3 Compact Inverted Blocks (C3CIB) modules to replace the C3 modules in YOLOv5. This substitution substantially improves inference speed while maintaining efficient feature extraction capabilities.
- Utilizing the Normalized Wasserstein Distance (NWD) loss function to enhance the detection of small fire objects, thereby improving the model's performance in small-object detection tasks.

2 Related Works

Fire detection is a core task in fire prevention and safety management, fundamentally aimed at inferring the system's state (i.e., whether a fire has occurred or there is a potential fire risk) based on a series of cues or signals. These cues may include smoke concentration, temperature variations, gas composition, and visual characteristics. To improve inference accuracy, traditional methods often employ multi-signal inference, sensor fusion, or ensemble techniques. These methods integrate signals from multiple sensors, such as smoke sensors, temperature sensors, and gas sensors, to make comprehensive judgments. For instance, Sahid et al. [5] proposed an early fire detection model by combining flame, carbon monoxide, and smoke sensors. Recently, Nakıp et al. [6] introduced a hybrid architecture, that enhances multi-sensor fire detection and risk assessment by using flattened sample regularization and environmental variable time trends, achieving high accuracy, low computational load, and robust generalization capabilities.

However, applying such multi-signal fusion methods in large spaces or complex environments poses significant challenges. In vast forest environments, deploying and maintaining numerous physical sensors is costly, and the equipment is prone to aging or damage due to environmental influences.

Moreover, the complexity of multi-sensor systems increases the burden on data processing and transmission, making real-time monitoring challenging [7]. Therefore, while multi-signal fusion methods perform well in small indoor environments, their efficiency and feasibility are limited in large-scale forest fire detection.

To address the limitations of traditional sensor-based detection, visual-based fire detection methods have garnered increasing attention from researchers. These methods rely on image processing and machine learning techniques to analyze visual characteristics, such as flames and smoke. Rudz et al. [8] developed a more precise fire detection model by analyzing the color and shape of flames. Similarly, Surit et al. [9] utilized static and dynamic image features to detect smoke in forest fires. Although these machine learning-based visual methods offer new approaches to fire detection, they often rely on predefined feature extraction, resulting in limited robustness in complex scenarios and making detection results vulnerable to environmental changes.

With the rapid advancement of deep learning technology, particularly the introduction of Convolutional Neural Networks (CNNs) [10], the field of object detection has seen the emergence of more intelligent detection models. Compared to traditional fire detection methods, CNN-based fire detection models require less manual intervention and exhibit strong generalization capabilities. These models can automatically learn fire-related features from data, enabling better performance in fire detection tasks. CNN-based object detection methods can be broadly categorized into two types: two-stage detection methods and single-stage detection methods.

Two-stage Detection Methods: These methods first generate candidate regions and then classify them, achieving higher detection accuracy. Typical two-stage models include Regions with CNN features (R-CNN) [11] and Faster R-CNN [12]. For instance, Cai et al. [13] proposed the Cascade R-CNN, a multi-stage object detection framework that addresses issues arising from training with low Intersection over Union (IoU) thresholds, which can lead to noisy detections, and high IoU thresholds, which may degrade performance. Despite their high accuracy in detecting large objects, two-stage methods often have high computational complexity and are challenging to deploy on resource-limited devices.

Single-stage Detection Methods: In contrast, single-stage detection methods perform classification and localization directly across the entire image, bypassing the region proposal stage. Examples of this type include the YOLO series [14] and SSD [15]. The YOLO series of models have gained widespread application in real-time detection tasks due to their efficient inference speed and good detection accuracy. For instance, Xiao et al. [16] recently proposed the EMG-YOLO algorithm, which leverages multi-scale attention modules, a global feature pyramid network, and pruning techniques to enhance fire detection accuracy while significantly reducing computational complexity.

Additionally, some studies have combined YOLO with other deep-learning methods to improve detection performance. For example, Qian et al. [17] integrated a Conditional Generative Adversarial Network (CGAN) with an improved YOLO model to achieve automated polyp detection, showcasing YOLO's potential for broader applications. However, in forest fire detection tasks, real-time performance and extensive monitoring are critical requirements. Although two-stage detection methods and combining YOLO with other deep learning methods may offer improved accuracy, they typically increase model complexity and computational resource demands, which conflicts with the real-time detection needs in resource-limited environments. On the other hand, single-stage detection methods, such as the YOLO series, exhibit faster inference speeds that can meet real-time detection needs. Moreover, the YOLO model structure is relatively simple, making it easier to deploy on resource-constrained devices.

In our study, we compared various deep learning-based object detection models. Considering the unique requirements of forest fire detection, it is necessary to strike a balance between accuracy, speed, and resource consumption. YOLOv5, as a mature version within the YOLO series, offers improved feature extraction efficiency and reduced computational load compared to YOLOv3 [18] and YOLOv4 [2]. By incorporating the Focus layer [3] and Cross Stage Partial Network (CSPNet) [19],YOLOv5 achieves a more lightweight model structure. Additionally, the application of Mosaic data augmentation enhances adaptability to various target sizes and positional changes, demonstrating excellent performance in small object detection tasks. We also evaluated the latest models, YOLOv8 [20] and YOLOv10 [21]. YOLOv8 optimizes the CSPDarknet backbone and the PANet [22] neck structure, boosting feature fusion capabilities, and introducing an anchor-free design that simplifies the model architecture and improves small-object detection. Meanwhile, YOLOv10 leverages an NMS-free [23] training mode and large-kernel convolutions to further optimize multi-scale feature extraction and real-time performance. While YOLOv8 and YOLOv10 performed well in certain tasks, YOLOv5 demonstrated the best overall performance on our fire detection dataset. It maintained high detection accuracy, had a fast inference speed, and exhibited good stability and generalization capabilities [24]. Consequently, we selected YOLOv5s as the baseline model, as it achieved an ideal balance between accuracy and speed. Although YOLOv5n is the most lightweight version in the series, possessing fewer parameters, it does not meet our accuracy requirements as effectively as YOLOv5s does. YOLOv5s provides higher detection accuracy and exhibits excellent inference speed, better satisfying our dual needs for accuracy and efficiency in fire detection tasks.

In recent related studies addressing the issue of missed detections of small objects, most YOLO-based improvement methods have incorporated attention mechanisms to enhance small object detection performance. For example, Luan et al. [25] proposed a UAV-based fire detection method called YOLO-CSQ, which integrates the CBAM attention mechanism to improve multi-scale fire feature extraction capabilities. Attention mechanisms indeed allow models to focus more on critical features, thereby enhancing detection accuracy for small objects. However, incorporating attention mechanisms often increases the model's computational overhead, resulting in slower inference speed, which conflicts with the real-time detection needs in resource-limited environments. In our study, we addressed the small-object detection problem by utilizing the Normalized Wasserstein Distance (NWD) loss function instead of the traditional IoU metric. This approach improved the model's performance in detecting small flames, and significantly reduced the rate of missed detections, all without compromising inference speed.

3 Methodology

3.1 Network Architecture

To address the challenges posed by objects of varying sizes and the difficulty of detecting small targets in fire detection tasks, this paper presents an improved lightweight model based on YOLOv5 v7.0, named YOLO-LFD. The model significantly optimizes inference speed while maintaining detection accuracy, making it suitable for real-time applications on resource-constrained devices, including drones. Four key modifications were made to the original YOLOv5 architecture:

(1) Introduction of DSConv: Reduces network complexity while improving computational efficiency.
(2) Replacement of the shallow C3 module with the C2f-Light module: Simplifies convolution operations, significantly boosting inference speed while maintaining effective feature extraction.

(3) Replacement of the deep C3 module with the C3CIB module: Improves deeper feature extraction and semantic information processing.
(4) Adoption of the NWD loss function: Replaces the traditional IoU loss, enhancing performance in detecting small fire targets and significantly reducing the missed detection rate.

Among these, modifications (1) and (4) are based on existing methods, whereas (2) and (3) constitute the novel contributions of this paper. By integrating these improvements into the network architecture, YOLO-LFD achieves an optimal balance between speed and accuracy for fire detection tasks. The overall architecture is illustrated in Fig. 1.

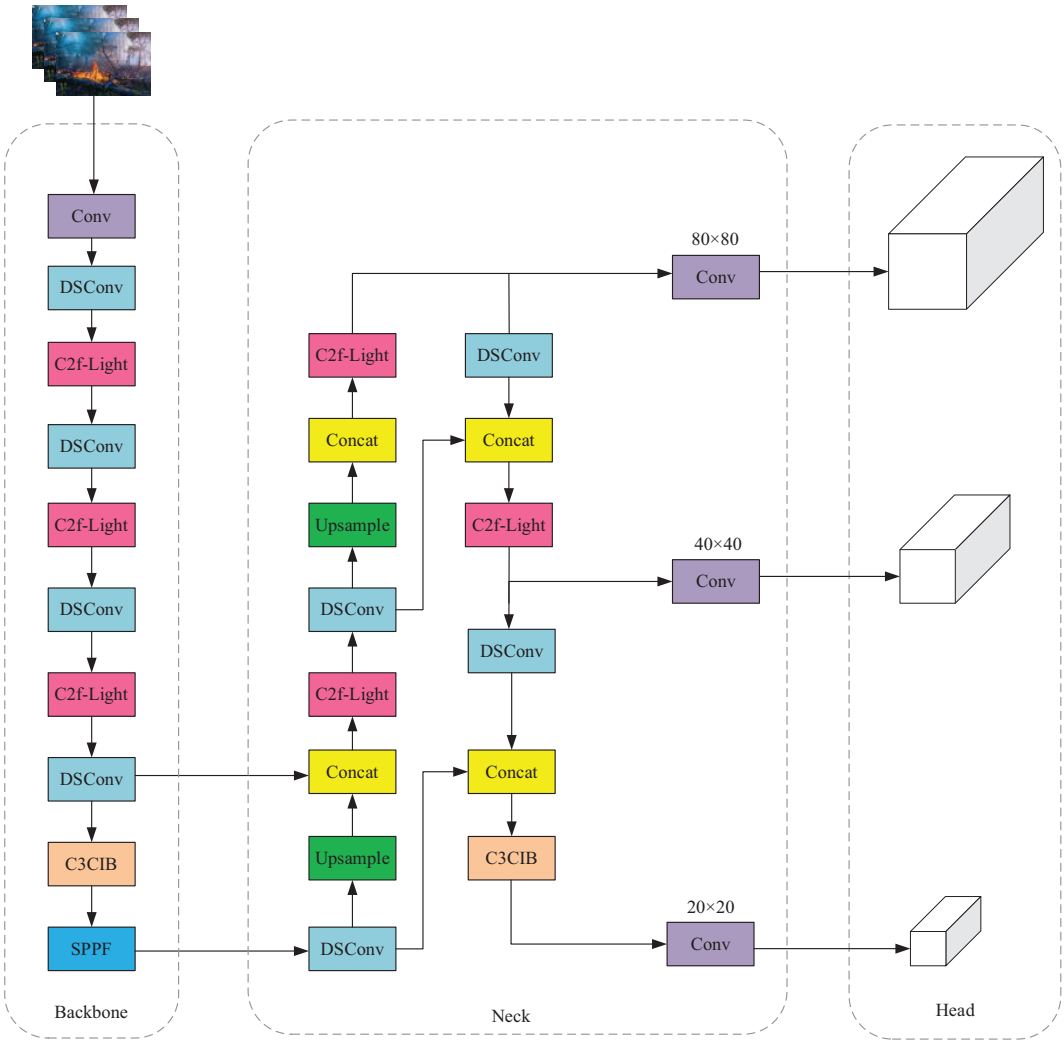


Figure 1: Overall architecture of the YOLO-LFD

3.2 DSConv

The Depthwise Separable Convolution (DSConv) module was introduced [26] into the YOLOv5 network, replacing the original standard convolution (Conv) module to reduce computational overhead and decrease the number of parameters, thereby accelerating the model's inference speed. As

shown in Fig. 2, DSConv consists of a Depthwise Convolution (DWConv) [27] and a Pointwise Convolution (1 × 1 convolution) [1]. The key characteristic of DWConv is that it performs convolution independently on each input channel without fusing information between channels. The computation of traditional convolution can be expressed as follows:

$$Y_{i,j,c} = \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} \sum_{d=0}^{C_{in}-1} X_{i+m,j+n,d} \cdot W_{m,n,d,c} \tag{1}$$

In traditional convolution, the input feature map X and convolution kernel W are weighted and summed across all input channels to generate the output feature map Y. In contrast, DWConv performs independent convolution on each input channel, and the formula simplifies to:

$$Y_{i,j,c} = \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} X_{i+m,j+n,c} \cdot W_{m,n,c} \tag{2}$$

This method reduces the computational cost from the standard convolution's complexity of $O(H \times W \times C_{\text{in}} \times C_{\text{out}} \times K \times K)$ to $O(H \times W \times C_{\text{in}} \times K \times K)$, achieving an approximately C_{out}-fold reduction in computation. However, since DWConv only performs convolution within channels and lacks inter-channel feature fusion, we address this limitation by applying PWConv. The computation formula for pointwise convolution is:

$$Y_{i,j,c'} = \sum_{d=0}^{C_{in}-1} X_{i,j,d} \cdot W_{1\times1,d,c'} \tag{3}$$

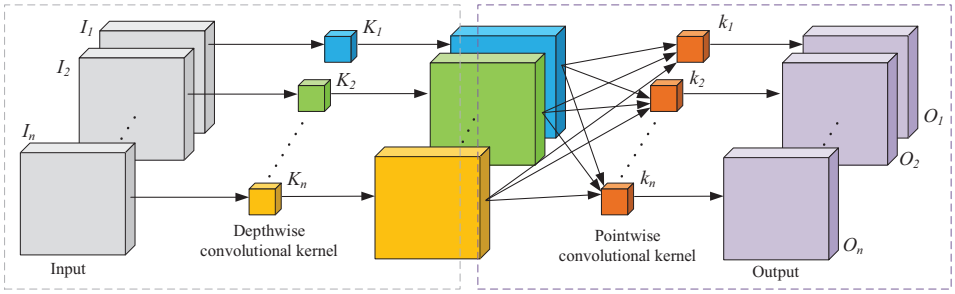


Figure 2: Depthwise-separable-convolutions-DSConv

PWConv is utilized for linear combinations between channels, enabling the integration of features from different channels. By comparing the computational complexity of standard convolution and DSConv, it becomes clear that the former has a computational cost of $H \times W \times C_{\text{in}} \times C_{\text{out}} \times K \times K$, while DSConv's computational cost is reduced to $H \times W \times C_{\text{in}} \times K \times K + H \times W \times C_{\text{in}} \times C_{\text{out}}$. This design significantly reduces the overall computational load and the number of parameters, thus enhancing the inference efficiency of the model.

3.3 C2f-Light

In the C2f module of YOLOv8 [20] (as shown in Fig. 3b), the design aims to enhance feature extraction and information flow by cascading multiple convolutional layers. While the C2f module excels in improving feature extraction, its structural complexity results in increased computational cost and parameter count. To address these issues, we propose the C2f-Light module (as shown in Fig. 3d)

that significantly reduces computational complexity and resource consumption while maintaining efficient feature extraction.

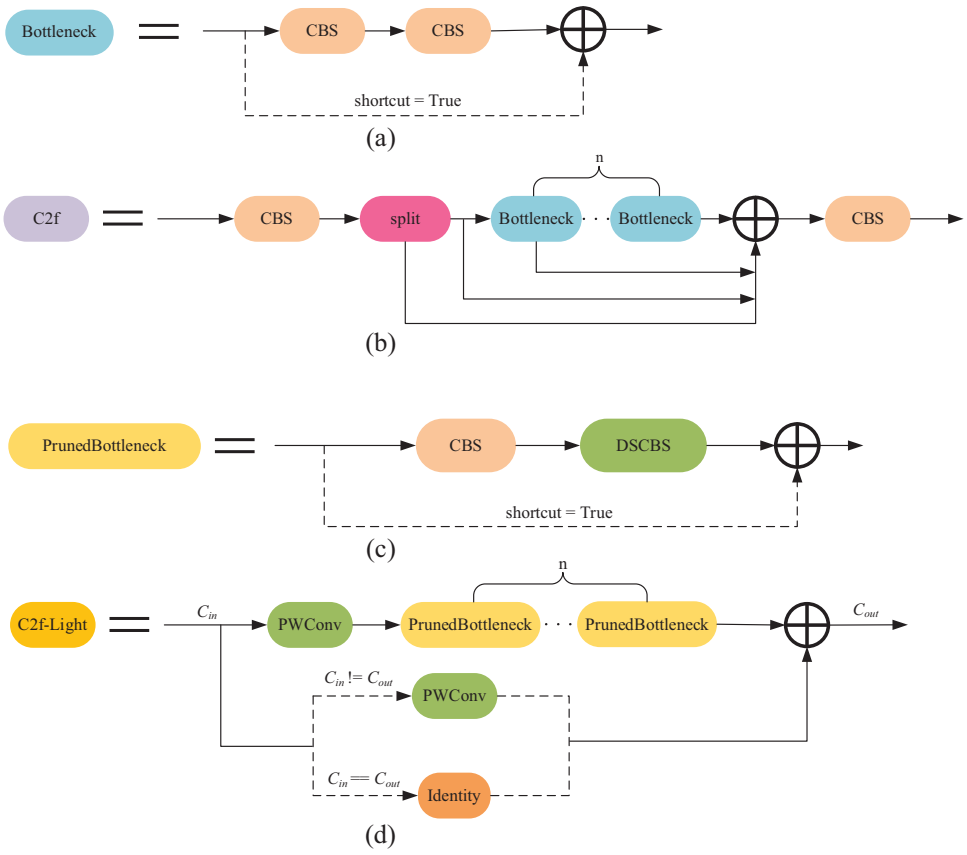


Figure 3: (a) Bottleneck module used in C2f; (b) C2f module used in YOLOv8; (c) The proposed Pruned Bottleneck module; (d) The proposed C2f-Light module

The core innovation of the C2f-Light module is the introduction of DSConv. DSConv decomposes the traditional standard convolution into two parts: Depthwise Convolution (DWConv) and Pointwise Convolution (PWConv). DWConv operates independently on each input channel, focusing on spatial feature extraction without involving inter-channel information exchange. PWConv, through 1×1 convolution, restores the interaction between channels by performing a linear combination of features across channels. This decomposition not only effectively reduces computational burden but also maintains strong feature extraction capabilities.

Compared to traditional convolution, DSConv significantly reduces computational complexity by separating spatial and channel-wise computations, leading to improved efficiency, especially when the number of input channels is large. This optimization greatly reduces the number of floating-point operations (FLOPs), thus improving the inference speed and computational efficiency of the model. Additionally, although the Bottleneck structure of the C2f module (as shown in Fig. 3a) enhances feature extraction through a series of Convolution-Batch Normalization-SiLU (CBS) modules, it also inevitably increases the computational load. Therefore, we introduce the Pruned Bottleneck structure (as shown in Fig. 3c), which replaces standard convolution with DSConv, substantially reducing

both computational cost and parameter count. The C2f-Light module, by simplifying convolution operations and integrating DSConv, achieves a significant boost in inference speed while ensuring effective feature extraction.

3.4 C3CIB

As shown in Fig. 4c, the C3 module in YOLOv5 is designed to enhance feature extraction capabilities and information flow through the Bottleneck structure [28]. While the Bottleneck structure improves feature extraction by cascading multiple CBS modules, it also leads to increased computational complexity and a higher number of parameters. This can become a performance bottleneck when handling tasks that require high efficiency and precision. To address this issue, we propose an innovative improvement based on the CIB module from YOLOv10 [21], introducing the C3CIB module, as illustrated in Fig. 4d.

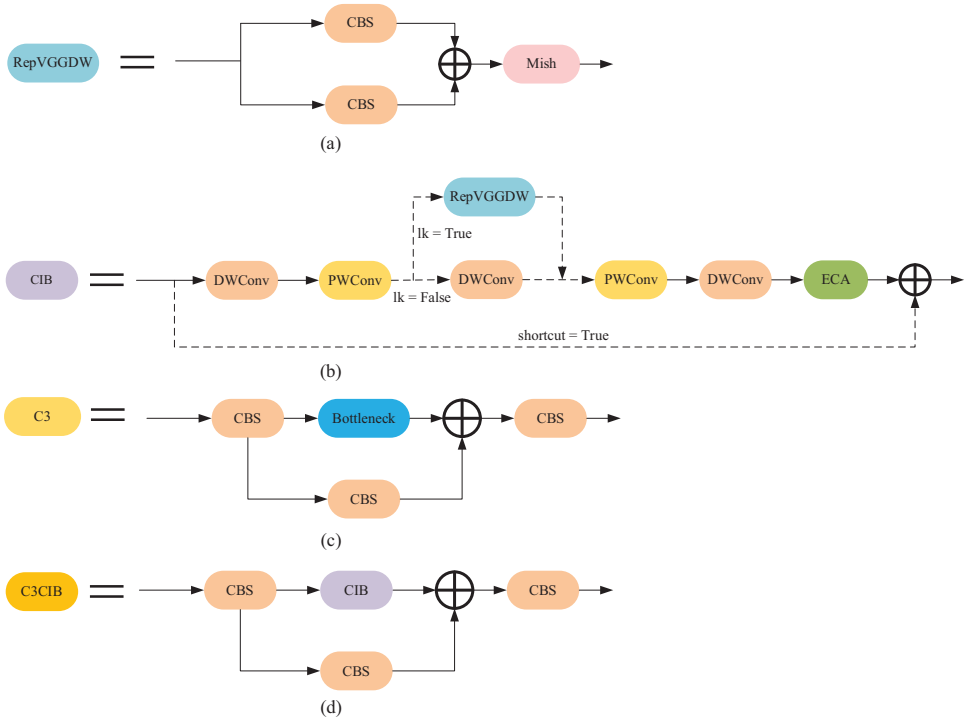


Figure 4: (a) RepVGGDW module used in CIB; (b) CIB module used in C3CIB; (c) C3 module in YOLOv5; (d) The proposed C3CIB module

The C3CIB module first incorporates the Compact Inverted Block (CIB), which aims to significantly reduce computational overhead and improve computational efficiency by combining Depthwise Convolution (DWConv) and Pointwise Convolution (PWConv). Fig. 4b shows the detailed structure of the CIB module: it includes both DWConv and PWConv convolution operations and integrates the Efficient Channel Attention (ECA) mechanism [29]. This mechanism adaptively assigns weights along the channel dimension through local convolution operations, enhancing the feature representation capability. ECA avoids the use of additional fully connected layers, thereby greatly reducing computational costs while ensuring feature capture ability. Moreover, the CIB module incorporates the RepVGGDW module, as shown in Fig. 4a. RepVGGDW enhances nonlinear expression capabilities

through the combination of depthwise convolution and the Mish activation function [30] and improves inference efficiency during the deployment phase through convolution fusion. These designs ensure the high efficiency and nonlinear expression capabilities of the CIB module, enabling it to extract key features more accurately in different scenarios.

In contrast, although the C3 module employs the Bottleneck structure, its computational complexity increases with the depth of the network. To tackle this problem, the C3CIB module replaces the Bottleneck structure in the C3 module by introducing the CIB module, thereby significantly reducing computational costs. Simultaneously, the C3CIB module retains the dual-branch design of the C3 module, allowing input features to be processed along different paths separately, preserving more fine-grained feature information. Additionally, the ECA mechanism within the CIB module further enhances the model's feature capture capability while maintaining efficiency. Compared to the traditional C3 module, the C3CIB module achieves enhanced feature extraction abilities, leading to further improvements in YOLOv5's performance when handling complex scenes.

3.5 NWD Loss Function

In the dataset, several challenges arise, such as small and densely packed flames at the far end of the image, flames at the edges of the image, and defects in flames within small regions. These issues can lead to the loss of critical fire features during pre-feature extraction, ultimately resulting in reduced fire detection accuracy. To address this problem, this paper introduces the Normalized Wasserstein Distance (NWD) loss function [31] to improve the detection accuracy for small fire targets.

As shown in Fig. 5, the sensitivity of Intersection over Union (IoU) varies significantly for different sizes of flames. Box A represents the ground truth bounding box, while boxes B and C represent bounding boxes with diagonal deviations of 1 pixel and 4 pixels, respectively. Specifically, for tiny objects with a size of 7 × 7 pixels, even a small positional deviation can cause a significant drop in IoU (from 0.58 to 0.1), leading to incorrect label assignment. In contrast, for normal objects of 21 × 21 pixels, the IoU changes less under the same positional deviation (from 0.83 to 0.49), indicating a certain degree of degradation in IoU. This degradation affects the label assignment process, ultimately influencing detection accuracy.

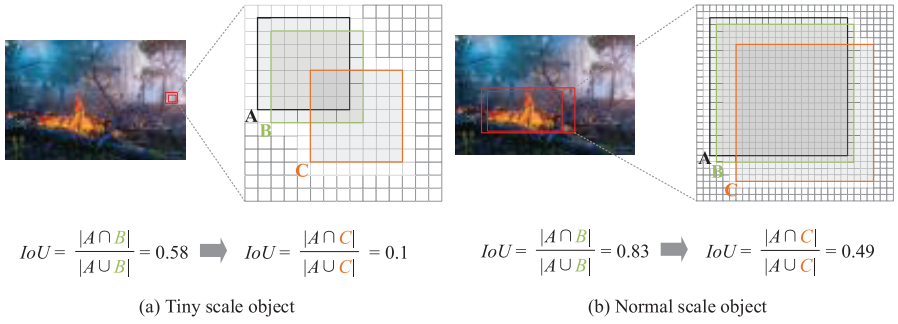


Figure 5: Comparison between tiny and normal-scale objects

IoU-Loss [32] was introduced to eliminate the performance gap between training and testing. However, IoU-Loss fails to provide gradients for network optimization in two specific cases: (1) when there is no overlap between the predicted bounding box P and the ground truth box G (i.e., $|P \cap G| = 0$); or (2) when bounding box P completely contains bounding box G or *vice versa* (i.e., $|P \cap G| = P$ or G). Additionally, these two cases are very common in small object detection. Specifically, a slight deviation

in P can lead to no overlap between P and G, and small objects are easily mispredicted, resulting in |P∩G| = P or G. Therefore, IoU-Loss is not suitable for small object detectors. Although DIoU and CIoU [33] can handle these cases, they are sensitive to positional deviations of small objects as both are based on IoU.

To solve these issues, this paper introduces the NWD loss function. This function models the bounding boxes as two-dimensional Gaussian distributions and uses the Wasserstein distance to measure the difference between the predicted and ground truth boxes. For smaller-scale objects, they are often not strictly rectangular in practice and usually occupy only a few pixels at the center of the bounding box, with irrelevant elements like the background distributed near the edges. To more accurately represent the importance of different pixels within a bounding box, a two-dimensional (2D) Gaussian distribution can be used to model the bounding box. In this model, the central pixels of the bounding box receive the highest weight, and the importance of the pixels gradually decreases from the center to the edges.

Specifically, for a horizontal bounding box $R = (c_x, c_y, w, h)$, where (c_x, c_y), w and h represent the center coordinates, width, and height respectively. Based on the pixel distribution characteristics, it can be expressed using the ellipse equation:

$$\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} = 1 \tag{4}$$

Here, μ_x and μ_y are the coordinates of the ellipse's center, and σ_x and σ_y are the lengths of the semi-axes along the x and y axes, respectively. Therefore, $\mu_x = c_x$, $\mu_y = c_y$, $\sigma_x = w/2$, and $\sigma_y = h/2$. The probability density function of the two-dimensional Gaussian distribution is:

$$f(x|\mu, \Sigma) = \frac{1}{2\pi\sqrt{\Sigma}} \exp\left(-\frac{(x-\mu)^{\mathrm{T}}}{2\sum(x-\mu)}\right) \tag{5}$$

where x, μ, and Σ represent the coordinates (x, y), the mean vector, and the covariance matrix, respectively. Next, the Wasserstein distance is used to measure the difference between two probability distributions. For two 2D Gaussian distributions $\mu_1 \sim N(m_1, \Sigma_1)$ and $\mu_2 \sim N(m_2, \Sigma_2)$, the second-order Wasserstein distance between μ_1 and μ_2 can be simplified as follows:

$$W_2^2(\mu_1, \mu_2) = ||m_1 - m_2||_2^2 + \left\| \sum_1^{\frac{1}{2}} - \sum_2^{\frac{1}{2}} \right\|_F^2 \tag{6}$$

where $\|\cdot\|_F$ is the Frobenius norm. For Gaussian distributions N_a and N_b modeled from bounding boxes $A = (c_{x_a}, c_{y_a}, w_a, h_a)$ and $B = (c_{x_b}, c_{y_b}, w_b, h_b)$, Eq. (6) can be further simplified as:

$$W_2^2(N_a, N_b) = \left\| \left(\left[cx_a, cy_a, \frac{w_a}{2}, \frac{h_a}{2} \right]^{\mathrm{T}}, \left[cx_b, cy_b, \frac{w_b}{2} \cdot \frac{h_b}{2} \right]^{\mathrm{T}} \right) \right\|_2^2 \tag{7}$$

However, $W_2^2(N_a, N_b)$ is a distance measure and cannot be directly used as a similarity measure (i.e., a value between 0 and 1 like IoU). Therefore, we normalize its exponential form to obtain a new measure called the NWD:

$$NWD(N_a, N_b) = \exp\left(-\frac{\sqrt{W_2^2(N_a, N_b)}}{C}\right) \tag{8}$$

where C is a constant closely related to the dataset. To ensure that the box loss reflects both NWD and IoU similarity information, we adjust the relative contributions of NWD and IoU in the total

box loss, setting the weight ratio to 7:3. This gives NWD a greater contribution to the total box loss, thereby enhancing the model's ability to detect small objects. The final loss function is expressed as:

$$Loss = \frac{1}{N}\sum_{i=1}^{N}(0.7 \cdot (1 - NWD_i) + 0.3 \cdot (1 - IoU_i)) \tag{9}$$

where N represents the number of detection boxes, and the average value is taken to aggregate the losses of multiple targets into a single value for further loss computation and optimization.

4 Experiments

4.1 Experimental Environment and Data Preprocessing

The experimental environment was based on a Windows 10 operating system, running on an Intel Core i5-12400F CPU, an NVIDIA GeForce RTX 4060 GPU, and 16 GB of RAM. The software environment included CUDA version 11.8 and the PyTorch framework. The input image size was set to 640 × 640 pixels, with a batch size of 32, and the model was trained for 200 epochs. Starting with pre-trained weights, the model was fine-tuned for 200 epochs to produce the final trained network.

The dataset used in this experiment comprised 2061 images, including scenes such as forest fires and aerial drone footage. These images were sourced from public datasets like VisiFire [34], as well as additional flame and smoke images obtained through online searches. Due to the limited size of the dataset, we applied data augmentation techniques—including Gaussian blur, cropping, brightness adjustment, and flipping—to expand the number of samples to 8244. Examples of the augmented images are shown in Fig. 6. The dataset was randomly divided into training, validation, and test sets with a ratio of 8:1:1.

Figure 6: Partial data enhancement results

To evaluate the model's effectiveness in detecting small fire targets, we selected 1000 images of small fire flames from the dataset for testing. The mean Average Precision (mAP)@0.5_tiny in the following ablation experiments is based on the detection results from this small target test set.

4.2 Evaluation Metrics

To accurately evaluate the performance of the proposed algorithm, precision, recall, average precision (AP), and mean average precision (mAP) are commonly used evaluation metrics in object detection. Their calculations are shown in Eqs. (10)–(13).

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$AP = \int_0^1 p\,(r)\,dr \tag{12}$$

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{13}$$

In this context, TP refers to the number of true positives correctly predicted by the model, FP represents the number of false positives, and FN denotes the number of false negatives where the model failed to detect a positive sample. N is the total number of classes in the detection task. AP is the area under the precision-recall curve for a single class, while mAP is the mean of APs across all classes. A higher mAP value indicates better performance of the algorithm. In our experiments, we pay special attention to the metrics mAP@0.5 and mAP@0.5_tiny. Here, mAP@0.5 refers to the mean Average Precision calculated at an IoU (Intersection over Union) threshold of 0.5, which measures the overall detection performance of the model. In contrast, mAP@0.5_tiny specifically evaluates the model's performance on the test set extracted for small target detection.

In addition to accuracy metrics, FPS (Frames Per Second) and FLOPs (Floating Point Operations) are also crucial indicators for evaluating model performance. On the current hardware setup, FPS measures the number of images the model can process per second, with higher values indicating faster inference speed. FLOPs reflect the total number of floating-point operations required to process a single image, with lower values indicating lower computational complexity.

4.3 Ablation Experiment

In the ablation experiments, we progressively introduced different improvement modules, as shown in Table 1—including DSConv, C2f-Light, C3CIB, and the NWD loss function—to evaluate their impact on model performance. The baseline model, YOLOv5s, achieved an mAP@0.5 of 94%, an inference speed of 290 FPS, and an mAP@0.5_tiny of 0.9 for small object detection. First, after incorporating DSConv, the inference speed increased to 328 FPS, and the number of parameters and FLOPs were significantly reduced, although the mAP@0.5 slightly decreased to 91.5%. Next, after adding the C2f-Light and C3CIB modules, the mAP@0.5 improved to 92.2%, and the inference speed significantly increased to 343 FPS. Finally, by introducing the NWD loss function, the model achieved an mAP@0.5 of 92.5%, the mAP@0.5_tiny for small object detection improved to 0.912, and the inference speed further increased to 346 FPS, achieving a good balance of performance. Although the

overall mAP@0.5 slightly decreased compared to the previous configuration, there was a significant improvement in small object detection.

Table 1: Ablation study results

Modification				Params	FLOPs	mAP@	mAP@	Inference
DSConv	C2f-Light	C3CIB	NWD	(M)	(G)	0.5	0.5tiny	FPS
				7.01	15.8	0.94	0.9	290
√				4.97↓	11.6↓	0.915↓	0.882↓	328↑
	√			6.46↓	12.5↓	0.925↓	0.892↓	347↑
		√		6.27↓	15.2↓	0.941↑	0.9	292↑
			√	7.01	15.8	0.945↑	0.915↑	295↑
√	√			4.43↓	8.4↓	0.914↓	0.885↓	375↑
√	√	√		3.69↓	7.8↓	0.922↓	0.893↓	343↑
√	√	√	√	**3.69↓**	**7.8↓**	**0.925↓**	**0.912↑**	**346↑**

The final experimental results show that by introducing these improvement modules, the model's number of parameters was reduced by approximately 47.3%, from 7.01 million to 3.69 million; FLOPs decreased by about 50.6%, from 15.8 billion to 7.8 billion; and the inference speed increased by approximately 19.3%, from 290 FPS to 346 FPS. Each module played a positive role in different aspects of performance, and the model performs exceptionally well in small object detection.

4.4 Comparison Experiments

In our comparative experiments, as shown in Table 2, the proposed YOLO-LFD model was systematically compared with the YOLOv5s baseline and other mainstream object detection models. YOLOv5s achieved an inference speed of 290 FPS and an mAP@0.5 of 0.94, while YOLO-LFD, with its optimized structure, attained an inference speed of 346 FPS, representing an improvement of approximately 19.31%. The mAP@0.5 only slightly decreased from 94% to 92.5%, a reduction of 1.6%. These results demonstrate that YOLO-LFD significantly enhances inference speed while maintaining high detection accuracy.

Moreover, YOLO-LFD exhibits substantial advantages in terms of parameter count and FLOPs. Compared to YOLOv5s, YOLO-LFD reduces parameters by 47.3% and FLOPs by 50.6%, making it highly efficient for deployment in resource-constrained environments. YOLOv8n offers faster inference (349 FPS) but achieves an mAP@0.5 of only 0.914, slightly lower than both YOLO-LFD and YOLOv5s. Additionally, YOLO-LFD outperforms YOLOv8n in precision and recall, achieving 91.8% and 87.2% respectively compared to YOLOv8n's 89.8% precision and 85.4% recall. Meanwhile, YOLOv8s reaches an mAP@0.5 of 0.928 but has a significantly lower inference speed of only 211 FPS. YOLOv7-tiny and YOLOv3-tiny achieve inference speeds of 321 FPS and 355 FPS, respectively; however, their mAP@0.5 scores of 0.88 and 0.893 indicate lower detection accuracy, particularly in complex scenarios. YOLOv10n also exhibits a balanced performance, with an inference speed of 332 FPS and an mAP@0.5 of 0.917, but overall, it still falls short compared to YOLO-LFD. YOLOv5n offers a more balanced performance, with an inference speed of 326 FPS and an mAP@0.5 of 0.918, though it still lags behind YOLO-LFD overall. As shown in Fig. 7, YOLO-LFD is positioned closest to

the top-right corner, indicating its superior overall performance and its well-rounded balance between detection accuracy and inference speed.

Table 2: Detection performance comparison

Model	Params (M)	FLOPs (G)	mAP@0.5	Precision	Recall	Inference FPS
YOLOv3-tiny	8.67	12.9	0.893	0.872	0.849	355
YOLOv5n	1.76	4.1	0.918	0.892	0.871	326
YOLOv5s	7.01	15.8	0.94	0.93	0.897	290
YOLOv7-tiny	6.01	13.0	0.88	0.84	0.828	321
YOLOv8n	3.01	8.1	0.914	0.898	0.854	349
YOLOv8s	11.1	28.4	0.928	0.924	0.88	211
YOLOv10n	2.69	8.2	0.917	0.904	0.847	332
YOLO-LFD	3.69	7.8	0.925	0.918	0.872	346

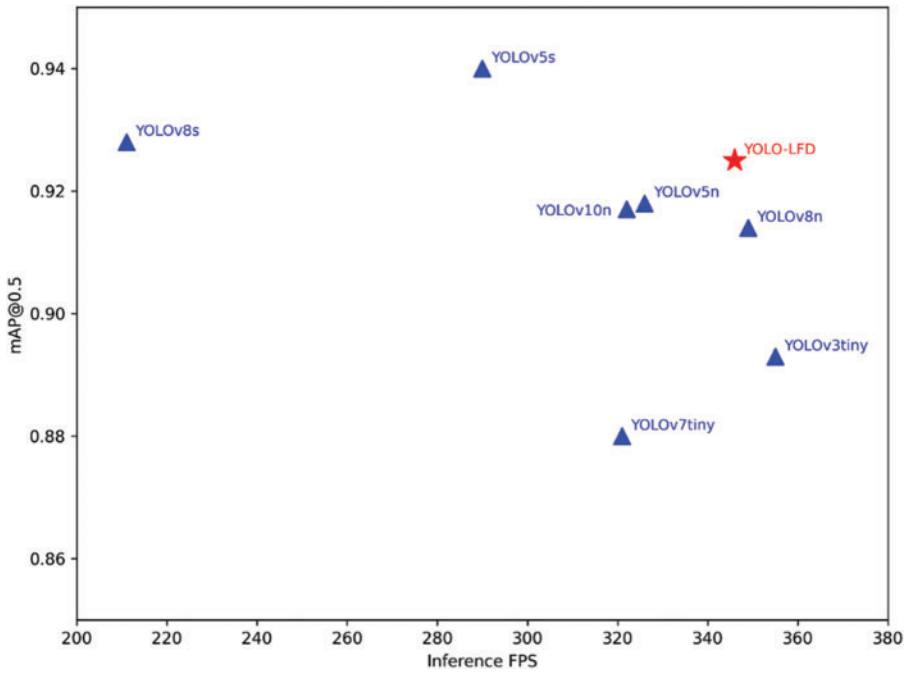


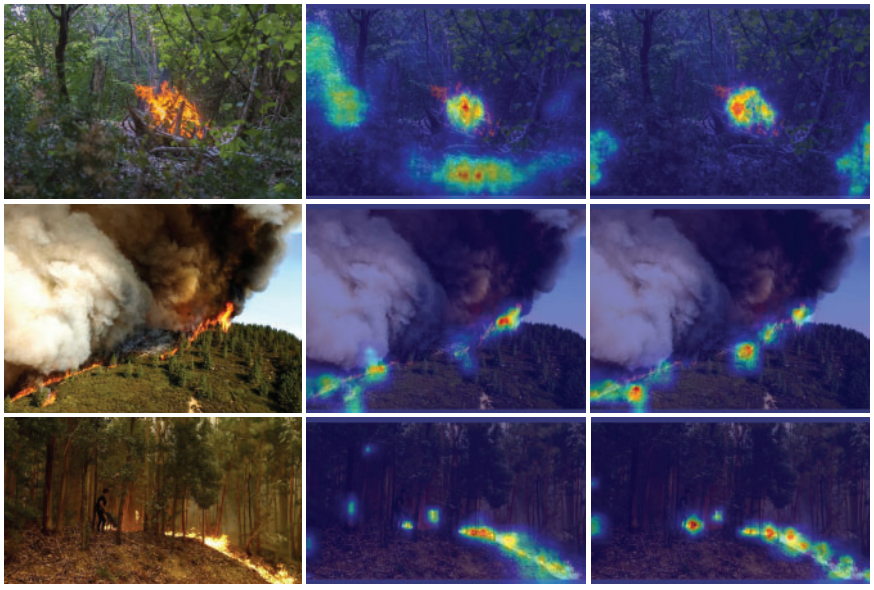
Figure 7: Detection performance comparison

In conclusion, YOLO-LFD's optimizations in inference speed, parameter count, and FLOPs, along with its advantages in precision and recall, make it exceptionally well-suited for resource-constrained environments. It is particularly advantageous for applications like drones and embedded devices that require efficient real-time detection.

4.5 Forest Fire Detection

In the forest fire detection task, we applied the Layer-CAM [35] method to visualize the detection features of the YOLO-LFD and YOLOv5s baseline models, enabling us to observe which regions the

network focuses on after passing through the backbone and detection head during target recognition and localization. We compared the detection performance of the YOLOv5s baseline with that of the improved YOLO-LFD model. As shown in Fig. 8, the YOLOv5s model predominantly focuses on the most prominent fire regions but struggles with detecting flames along the edges or smaller fires, making it difficult to identify subtle fire characteristics. In contrast, the YOLO-LFD model extracts more discriminative features from the image, effectively leveraging information from densely packed small targets. Its heatmaps demonstrate a broader and more detailed attention span, successfully capturing complex scene features like fire expansion at the edges. YOLO-LFD significantly enhances its ability to detect small fire sources, reducing both missed detections and false positives, thereby improving the accuracy and reliability of early forest fire warnings.



(a) original image (b) YOLOv5s (c) YOLO-LFD

Figure 8: Visualization of model-generated feature maps. (a) Images of forest fires. (b) Attention heatmap of YOLOv5s. (c) Attention heatmap of YOLO-LFD

Additionally, Fig. 9 illustrates the specific detection results of both models across various forest fire scenarios. Compared to the baseline model, YOLO-LFD exhibits stronger robustness in detecting fires in low-contrast scenes, reducing missed detections and false positives. Especially in scenarios with severe occlusion and challenging lighting conditions, YOLO-LFD accurately identifies concealed fire sources, markedly improving the model's performance in real-world applications.

(a) YOLOv5s (b) YOLO-LFD

Figure 9: Visualization of the detection results of YOLOv5s and YOLO-LFD

5 Conclusion and Future Work

Fire detection is a critical task in the field of object detection. Based on YOLOv5 v7.0, this paper proposes a lightweight fire detection model named YOLO-LFD. To address issues of computational complexity and resource consumption in existing models, we have implemented several improvements. Firstly, we adopted DSConv, which significantly reduces the computational cost and number of parameters of the model. Secondly, we designed the C2f-Light and C3CIB modules to replace the C3 modules in YOLOv5, optimizing feature aggregation capabilities and effectively enhancing inference speed. To improve the detection performance for small fire flames, we also introduced the NWD loss function, further enhancing the detection accuracy for small targets.

Experimental results show that YOLO-LFD improves inference speed by 19.3% compared to YOLOv5s, with only a 1.6% decrease in mAP@0.5. Additionally, YOLO-LFD reduces the number of parameters and FLOPs by 47.3% and 50.6%, respectively, fully demonstrating its excellent adaptability in environments with limited computational resources. This makes it especially suitable for deployment in scenarios requiring efficient real-time detection, such as drones and embedded devices.

However, we also recognize that deep learning models, especially neural networks like YOLO-LFD, are often considered "black boxes" [36], with internal decision-making processes lacking

intuitive interpretability for humans. In safety-critical fields such as fire detection, the interpretability of models is crucial for enhancing system trustworthiness and reliability. Therefore, our future work will not only focus on further improving the model's accuracy and efficiency but also on enhancing its interpretability. We plan to employ Explainable Artificial Intelligence (XAI) [37] methods, such as Grad-CAM [38] and feature visualization techniques, to deeply analyze the model's feature extraction and decision-making processes, revealing the key regions and features the model focuses on when detecting flames. This will help users understand the basis of the model's judgments, increase trust in the system, and identify potential biases and shortcomings in the model for further improvement.

In addition, we plan to test the model's robustness in more complex scenarios, such as varied flame shapes, extreme weather conditions, and multi-object detection tasks. By incorporating optimizations from the latest versions of YOLO, we aim to further simplify the model architecture and reduce computational costs, ensuring that the model has greater applicability and deployment flexibility while maintaining high performance. These improvements to YOLO-LFD not only provide an effective solution for fire detection tasks but also pave the way for new possibilities in other small-object detection applications.

Acknowledgement: We acknowledge the support of the National Natural Science Foundation of China.

Funding Statement: This work was supported by the National Natural Science Foundation of China (Grant Nos. 62101275 and 62101274).

Author Contributions: Yangyang Zhang conceived and designed the study, improved the modules, and drafted the manuscript. Honglin Wang contributed to the introduction, related work, and assisted with experiments. Cheng Zhu provided guidance and reviewed the manuscript. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data supporting the findings of this study can be obtained from the corresponding author, Yangyang Zhang, upon reasonable request. Please contact 202312490635@nuist.edu.cn for inquiries.

Ethics Approval: Not applicable.

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

References

[1] K. Muhammad, J. Ahmad, and S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management," *Neurocomputing*, vol. 288, no. 5, pp. 30–42, May 2018. doi: 10.1016/j.neucom.2017.04.083.

[2] A. Bochkovskiy, C. -Y. Wang, and H. -Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020. doi: 10.48550/ARXIV.2004.10934.

[3] G. Jocher *et al.*, "ultralytics/yolov5: v7.0-YOLOv5 SOTA realtime instance segmentation," *Zenodo*, Nov. 22, 2022. doi: 10.5281/ZENODO.7347926.

[4] G. Sun, S. Wang, and J. Xie, "An image object detection model based on mixed attention mechanism optimized YOLOv5," *Electronics*, vol. 12, no. 7, Mar. 2023, Art. no. 1515. doi: 10.3390/electronics12071515.

[5] D. Sahid and M. Alaydrus, "Multi sensor fire detection in low voltage electrical panel using modular fuzzy logic," in *2020 2nd Int. Conf. Broadband Commun., Wireless Sens. Power. (BCWSP)*, Yogyakarta, Indonesia, IEEE, Sep. 2020, pp. 31–35. doi: 10.1109/BCWSP50066.2020.9249400.

[6] M. Nakıp, N. Keleşoğlu, and C. Güzeliş, "Fire detection and risk assessment via Support Vector Regression with flattening-samples based augmented regularization," *Appl. Soft Comput.*, vol. 164, Oct. 2024, Art. no. 112023. doi: 10.1016/j.asoc.2024.112023.

[7] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Inf. Fusion*, vol. 14, no. 1, pp. 28–44, Jan. 2013. doi: 10.1016/j.inffus.2011.08.001.

[8] S. Rudz, K. Chetehouna, A. Hafiane, O. Sero-Guillaume, and H. Laurent, "On the evaluation of segmentation methods for wildland fire," in *Advanced Concepts for Intelligent Vision Systems*, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5807, pp. 12–23, doi: 10.1007/978-3-642-04697-1_2.

[9] S. Surit and W. Chatwiriya, "Forest fire smoke detection in video based on digital image processing approach with static and dynamic characteristic analysis," in *2011 First ACIS/JNU Int. Conf. Comput., Netw., Syst. Indust. Eng.*, Jeju, Republic of Korea, IEEE, May 2011, pp. 35–39. doi: 10.1109/CNSI.2011.47.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. doi: 10.1145/3065386.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016. doi: 10.1109/TPAMI.2015.2437384.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017. doi: 10.1109/TPAMI.2016.2577031.

[13] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, USA, IEEE, Jun. 2018, pp. 6154–6162. doi: 10.1109/CVPR.2018.00644.

[14] L. Jiao and M. I. Abdullah, "YOLO series algorithms in object detection of unmanned aerial vehicles: A survey," *Service Orient. Computi. Applicati.*, vol. 18, no. 3, pp. 269–298, Mar. 2024. doi: 10.1007/s11761-024-00388-w.

[15] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Computer Vision-ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, vol. 9905, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[16] L. Xiao, W. Li, X. Zhang, H. Jiang, B. Wan and D. Ren, "EMG-YOLO: An efficient fire detection model for embedded devices," *Digit. Signal Process.*, vol. 156, Jan. 2025, Art. no. 104824. doi: 10.1016/j.dsp.2024.104824.

[17] Z. Qian, W. Jing, Y. Lv, and W. Zhang, "Automatic polyp detection by combining conditional generative adversarial network and modified you-only-look-once," *IEEE Sensors J.*, vol. 22, no. 11, pp. 10841–10849, Jun. 2022. doi: 10.1109/JSEN.2022.3170034.

[18] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018. doi: 10.48550/ARXIV.1804.02767.

[19] C. -Y. Wang, H.-Y. Mark Liao, Y. -H. Wu, P. -Y. Chen, J. -W. Hsieh and I. -H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Seattle, WA, USA, IEEE, Jun. 2020, pp. 1571–1580. doi: 10.1109/CVPRW50498.2020.00203.

[20] G. Jocher *et al.*, "YOLOv8: Ultralytics' newest object detection model," *Ultralytics*, 2023. Accessed: Sep. 16, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[21] A. Wang *et al.*, "YOLOv10: Real-time end-to-end object detection," May 23, 2024, *arXiv:2405.14458*.

[22] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8759–8768. doi: 10.1109/CVPR.2018.00913.

[23] N. Carion *et al.*, "End-to-end object detection with transformers," in *European Conf. Comput. Vis. (ECCV)*, 2020, pp. 213–229. doi: 10.1007/978-3-030-58452-8_13.

[24] J. Wang *et al.*, "YOLO-DD: Improved YOLOv5 for defect detection," *Comput. Mater. Contin.*, vol. 78, no. 1, pp. 759–780, 2024. doi: 10.32604/cmc.2023.041600.
[25] T. Luan, S. Zhou, L. Liu, and W. Pan, "Tiny-object detection based on optimized YOLO-CSQ for accurate drone detection in wildfire scenarios," *Drones*, vol. 8, no. 9, Sep. 2024, Art. no. 454. doi: 10.3390/drones8090454.
[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. -C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.
[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1251–1258. doi: 10.1109/CVPR.2017.195.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.
[29] Q. Wang *et al.*, "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 11531–11539. doi: 10.1109/CVPR42600.2020.01155.
[30] D. Misra, "Mish: A self regularized non-monotonic activation function," 2019. doi: 10.48550/ARXIV.1908.08681.
[31] Y. Balaji, R. Chellappa, and S. Feizi, "Normalized wasserstein for mixture distributions with applications in adversarial learning and domain adaptation," in *2019 IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, Republic of Korea, IEEE, Oct. 2019, pp. 6499–6507. doi: 10.1109/ICCV.2019.00660.
[32] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 3431–3440. doi: 10.1109/CVPR.2015.7298965.
[33] Z. Zheng *et al.*, "Distance-IoU loss: Faster and better learning for bounding box regression," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, pp. 12993–13000, Apr. 2020. doi: 10.1609/aaai.v34i07.6999.
[34] A. E. Cetin, "Computer vision based fire detection dataset," 2014. Accessed: Aug. 3, 2023. [Online]. Available: http://signal.ee.bilkent.edu.tr/VisiFire/
[35] P. -T. Jiang, C. -B. Zhang, Q. Hou, M. -M. Cheng, and Y. Wei, "LayerCAM: Exploring hierarchical class activation maps for localization," *IEEE Trans. Image Process.*, vol. 30, pp. 5875–5888, 2021. doi: 10.1109/TIP.2021.3089943.
[36] Z. F. Wu, J. Li, M. Y. Cai, Y. Lin, and W. J. Zhang, "On membership of black-box or white-box of artificial neural network models," in *2016 IEEE 11th Conf. Indust. Electr. Appl. (ICIEA)*, Hefei, China, IEEE, Jun. 2016, pp. 1400–1404. doi: 10.1109/ICIEA.2016.7603804.
[37] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, no. 3, pp. 82–115, Jun. 2020. doi: 10.1016/j.inffus.2019.12.012.
[38] K. Raghavan, B. Sivaselvan, and K. V, "Attention guided grad-CAM: An improved explainable artificial intelligence model for infrared breast cancer detection," *Multimed. Tools Appl.*, vol. 83, no. 19, pp. 57551–57578, Dec. 2023. doi: 10.1007/s11042-023-17776-7.