Lightweight Road Damage Detection Network Based on YOLOv5

Jingwei Zhao, Ye Tao, Zhixian Zhang, Chao Huang, and Wenhua Cui

Abstract—**The field of computer vision has experienced rapid progress owing to deep learning. The importance of road damage detection in ensuring traffic safety and reducing road maintenance costs is becoming increasingly evident. For detecting road damage, the YOLOv5 algorithm provides a reliable and effective method. However, YOLOv5 still requires a significant amount of computation. This paper proposes a lightweight network for detecting road damage that improves upon the YOLOv5 model in four ways. The algorithm accurately identifies and classifies different types of road damage, while simultaneously reducing the number of parameters and required computations. First, lightweight processing of the model is achieved. The Ghost module and Ghost Bottleneck are employed to construct the novel GBS module and C3Ghost, which replace the existing CBS and C3 modules. Second, the CIoU loss function is transformed into SIoU to improve the precision of target box regression. Furthermore, the original upsampling module is replaced by CARAFE to improve the model's semantic adaptability and receptive field. Finally, the CBAM attention mechanism is employed to concentrate on crucial feature information. The experiment's findings present that, in comparison to the baseline model, the upgraded model has 41.8% fewer parameters. Additionally, there has been a 43.8% reduction in floating-point computation and an improvement of 0.2% in detection accuracy.**

Index Terms—**Road Damage Detection, Lightweight Network, YOLOv5, CARAFE, CBAM**

I. Introduction

BY the end of 2022, the national highway mileage had reached 5,354,800 km. Approximately 99.9% of the total highway distance, or 5,350,300 km, was under maintenance. This illustrates the seriousness of the road maintenance problem. The main cause of road damage is the destruction of the pavement structures. The pavement is the initial component of the highway infrastructure to be subjected to the impact of external forces. Pavements endure vehicle loads and are susceptible to temperature fluctuations, corrosion, and human-induced harm [1]. When cracks or potholes appear on the road, rain exacerbates the expansion of these defects. This situation is dangerous for moving vehicles and can contribute to traffic accidents. Road damage poses a significant threat to travel safety. Therefore, timely detection and mastery of road damage are important tasks to ensure road safety and carry out road maintenance. It is vital to ensure the safety of individuals and the security of their property. Road damage detection techniques represent a vital method for quality control throughout the life cycle construction phase of highway projects [2]. It offers indispensable resources for the inspection, supervision, and control of quality in construction projects related to highways. The application of technology for the detection of damage to roads is of significant importance in the control of progress and the cost of road projects throughout their whole life cycle. Therefore, road damage detection technology has very important practical significance.

Deep learning-based image processing outperforms traditional methods in accuracy, speed, and embeddability, making it a prevalent choice for road defect detection [3]. However, for real-time damage assessment, the current deep-learning models are inadequate. Therefore, a model that improves detection accuracy and reduces complexity is urgently needed. YOLOv5 is a high-accuracy, real-time single-stage object detection model. However, despite its high accuracy, the YOLOv5 model still requires a large amount of computation. Enhancing YOLOv5 addresses this, boosting accuracy while diminishing parameters and computations. Thus, real-time detection can be realized, and the adaptability and portability of the algorithm in edge-computing terminals can be improved.

To address these challenges, this paper introduces modifications to YOLOv5. First, use Ghost modules in place of conventional convolutions in the Backbone and Neck. We add a new GBS module to improve the original CBS. And we use GhostBottleneck to adopt a more effective C3Ghost module in order to minimize parameters without sacrificing accuracy. Second, CIoU has been replaced by SIoU to improve its anti-noise capabilities. Additionally, the CARAFE upsampling module is integrated for bigger receptive fields. Finally, a lightweight CBAM attention mechanism has been employed to further enhance overall performance.

Manuscript received January 22, 2024; revised June 22, 2024.

This work was supported by the National Natural Science Foundation of China (62272093), the Department of Education of Liaoning Province (LJKFZ20220197).

Jingwei Zhao is an undergraduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: wei_eiei@163.com).

Ye Tao is an Associate Professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (Corresponding author, phone: +86-133-0422-4928; e-mail: taibeijack@163.com).

Zhixian Zhang is a Postgraduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: 1849535553@qq.com).

Chao Huang is an undergraduate student of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: 2697759840@qq.com).

Wenhua Cui is a Professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, China. (e-mail: taibeijack@126.com).

II. Related Work

A. Current Status of Research on Traditional Image Processing-Based Road Damage Detection

Conventional image processing approaches, prevalent in road damage detection research, frequently employ threshold segmentation and feature extraction. One popular method for separating the target from the background is threshold segmentation. For instance, Akagic et al. [4] presented an integrated method that utilizes a grayscale histogram and Otsu threshold. This technique divides the input image into smaller images and explores road cracks within each one. Liu et al. [5] achieved better results by processing binary images obtained from a connected domain algorithm (Direction Segmentation Expansion Algorithm) to detect road cracks. However, threshold segmentation techniques are vulnerable to noise and only take into account the image's grayscale information, neglecting spatial information. As a result, to increase segmentation accuracy, this algorithm is frequently used in combination with other techniques [6].

Additionally, some methods extract edge information regarding road degradation using edge detectors. For example, Maode et al. [7] employed morphological filters and a modified median filter to identify cracks. However, these methods have limitations in detecting noise and intricate forms of road damage. Furthermore, a few studies have classified and detected road damage using machine learning techniques like support vector machines (SVMs), for road damage classification and detection. In contrast to single pothole detection, Hoang [8] proposed a supervised learning method based on SVMs to automatically classify potholes in roadways. Gao et al. [9] offered a quick detection method by combining a machine learning model with a library of support vector machines (LIBSVMs). This method is capable of differentiating between various forms of road damage.

However, many techniques for detecting road damage that rely on conventional image processing techniques have certain common flaws and restrictions. First, these methods often depend on manually created feature extraction and threshold selection, which may limit generalizability to diverse road damages. Second, conventional approaches are vulnerable to noise and complicated surroundings, as well as variations in background interference and lighting. This may result in a reduction in the accuracy of detection. Moreover, the performance of traditional methods is limited by the efficiency and computational complexity of image processing algorithms. Therefore, traditional methods are unsuitable for real-time applications and large-scale data processing.

B. Research Status of Road Damage Detection Based on Deep Learning

Deep learning-based methods for detecting road damage have been shown to be useful in real-time applications and various visual tasks. To further the field, researchers have used traditional network topologies such as SSD, Faster-RCNN, YOLO series, and EfficientDet.

Road damage identification and classification using the ResNet-152 feature extraction network and the Faster-RCNN detection framework were presented by Wang et al. [10]. The method described utilizes a feature extraction network and target detection framework, which are powerful but computationally demanding. Gupta et al. [11] presented a method based on SSD and RetinaNet detection framework. They addressed the problem of unfavorable weather conditions in pothole detection by using ResNet-34 and ResNet-50 as feature extraction networks. This method can be used to detect road damage under harsh environmental conditions. A fully convolutional network (FCN) was used by Yang et al. [12] to identify road cracks at the pixel level. These methods can detect cracks with high precision but require greater computational resources and training data. Nguyen et al. [13] used the VGG16 network to automate the detection of pits and cracks. Techniques for data augmentation processing improved the robustness of the model. While the model's network structure is large, posing a challenge to the adaptability requirements of embedded device applications. Grayscale photos of road pits were processed by Baek et al. [14] and fed into the YOLO detection model, resulting in improved detection speed and good performance. However, the model's generalization ability and robustness may be declined because the amount of information is reduced by processing the image.

In order to analyze the effectiveness of YOLO, SSD, HOG, SVM, and Faster R-CNN network models for pavement damage detection, Ping et al. [15] carried out experiments. According to the experimental result, the YOLOv3 model of the YOLO network algorithm family performs the best in pavement defect detection. This algorithm produces reliable detection results quickly. And many articles are based on this algorithm for road damage detection research. The YOLOv3 algorithm was employed by Du et al. [16] to build a pavement identification defect model and accomplish automatically extracted features. While this method increases the speed of detection, it lacks the flexibility required by embedded systems. A YOLOv3-based data enhancement technique for crack detection was employed by Tsuchiya et al. [17], which can effectively improve accuracy. It is characterized by the data enrichment techniques to increase the robustness and generalization of the model. However, it requires more computational resources and training time. A crack detection approach based on YOLOv3-Lite was put forward by Li et al.[18], which employs depth-separable convolutions and feature pyramids to design the network architecture. The method to detect cracks combines low-resolution and high-resolution features. This approach improves detection by utilizing features of different resolutions. However, it suffers from a high network design and parameter tuning.

A lightweight end-to-end network for detecting pavement damages was put forward by Liang et al. [19]. The network aims to quickly, automatically, and precisely detect and categorize different kinds of road damage. The method builds a multi-scale feature fusion network by combining a backbone network with several lightweight feature detection modules. This approach is more effective for recognizing and classifying targets at different distances and angles compared with other research. Additionally, they developed an embedded lightweight attention module. By giving multi-scale convolutional kernels weights, this module improves feature information and reduces the number of parameters needed for detection. Wan et al. [20] proposed the YOLO-LRDD algorithm using a new backbone. They increased the data volume in the RDD2020 dataset by using data collected in China. The method they proposed improved the speed of detection by 22%. Additionally, the road damage dataset was found to have issues with inconsistent resolution and low data quality. To address this, Chen et al. [21] introduced the LAG-YOLO method, a powerful deep learning network to detect damage on roads. By upgrading YOLO's network structure, LAG-YOLO preserves high precision while improving its suitability for real-time processing and lightweight deployment. Additionally, a new model module, Attention Ghost, is designed to utilize the attention mechanism of SimAM to decrease model parameters and enhance model performance.

To sum up, the current deep learning-based methods for detecting road damage have unique qualities and drawbacks. Some methods have achieved satisfactory results in terms of accuracy, detection speed, or adaptability to complex environments. However, there are still issues to be addressed, including the high computational resource demand and insufficient generalization ability. But detecting tasks may be restricted by lightweight models' low precision. It can be challenging to achieve the correct balance between effectiveness and efficiency [22]. This research provides a lightweight network-based road damage detection system that aims to decrease computing costs while improving detection accuracy. It can satisfy the requirements for real-time monitoring and maintenance of road damages while controlling the cost.

III. ALGORITHM

A. *Network Architecture Design*

This paper presents four key improvements to the YOLOv5 model. Fig. 1 displays the enhanced lightweight YOLOv5 model.

The model is first lightened by replacing the traditional convolutional operations in the Backbone and Neck with Ghost modules. The Ghost module serves to build a new GBS module that substitutes the original CBS module. In addition, a module called C3Ghost is constructed using the Ghost Bottleneck to replace the original C3 module. These lightweight operations can significantly reduce the amount of computation while maintaining performance.

Next, we have targeted three crucial model positions for further enhancements to make up for the accuracy loss caused by lightweight operation. The model's detection performance was enhanced further by using the SIoU and lightweight CARAFE upsampling operators. Incorporating the CBAM attention mechanism before the three detection heads further improves the detection capability and robustness of the model. The YOLOv5 model's performance and efficacy in the target detection task are significantly enhanced with these improvements.

B. *Lightweight Processing*

In 2020, Han et al. [23] introduced a lightweight module called Ghost. This module generates more feature maps with fewer calculations and parameters [24]. In this study, we substitute the newly constructed GBS module for the original model's CBS module. GBS module enhances model

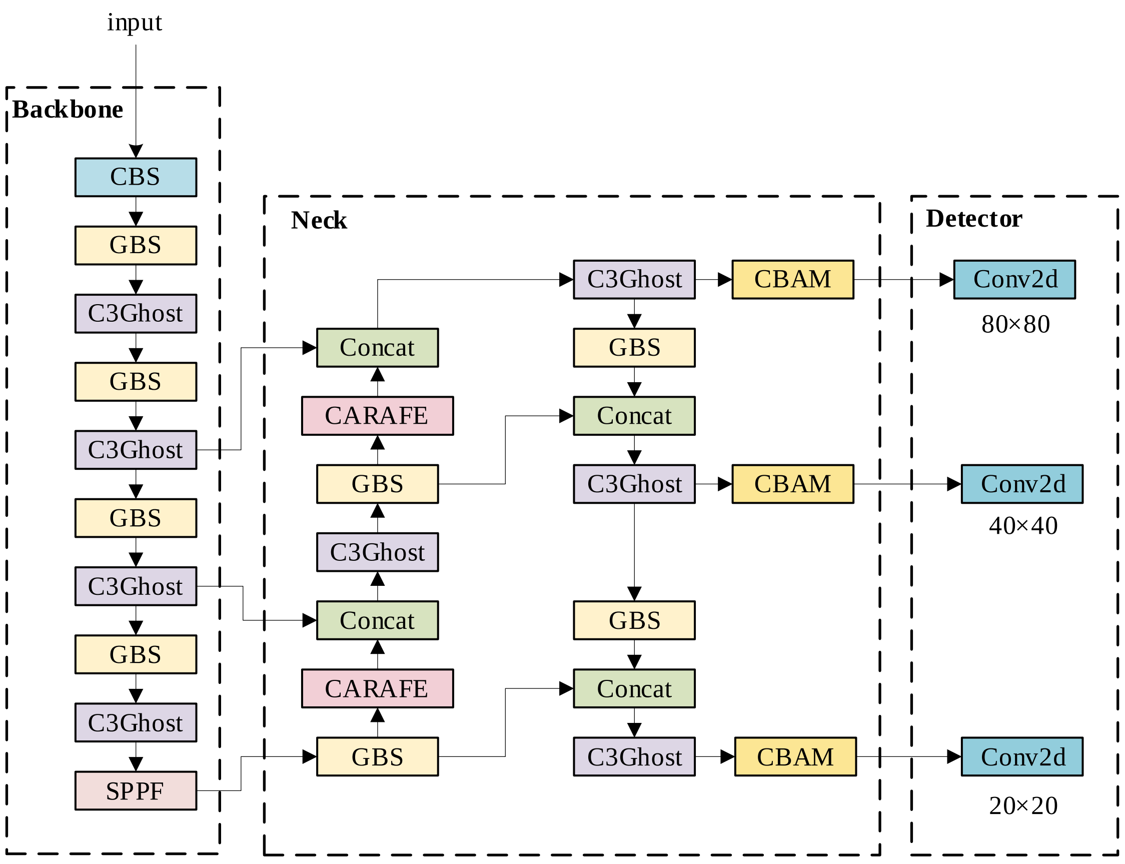


Fig. 1. Improved lightweight YOLOv5 model structure

efficiency by reducing computations and parameters, preserving feature representation and receptive field, and boosting generalization capability. The Ghost module decomposes the input feature map into two sub-maps and introduces a random transformation between them. This processing increases the model's adaptability to different samples and features. This enhances the model's generalization capabilities and lowers the possibility of overfitting. In road damage detection tasks, road conditions may vary due to weather, light, and other factors. The utilization of the Ghost module enhances the model's robustness to changes and improves its generalization performance in various scenarios. The structure of the Ghost module is shown in Fig. 2.

The conventional convolution layer is displayed in Fig. 2(a), while the Ghost module is displayed in Fig. 2(b). The picture illustrates that the Ghost module differs from traditional convolution in that it is divided into two parts. The first part is similar to traditional convolution: an input produces an output feature map after a common convolution operation. However, strict controls exist on the channels in this output feature map. To generate additional feature maps, the second part builds upon the output feature map of the first part by applying a sequence of low-cost linear operations. The output of the complete Ghost module is then generated by splicing the feature maps with the output feature maps of the first part. The results were compared with ordinary convolutional neural networks keeping the output feature map's dimensions constant. It was discovered that the Ghost module required less computation and had fewer arguments overall. The procedure is shown in Fig. 2(b), where Φ is a linear operation.

The formula for the FLOPs of floating-point calculations for ordinary convolution is shown in equation (1):

$$F_{conv} = n \cdot h' \cdot w' \cdot c \cdot k \cdot k \tag{1}$$

Equation (2) displays the formula for the number of floating-point calculations (FLOPs) for the Ghost module:

$$F_{ghost} = \frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k \cdot k + (s-1) \cdot h' \cdot w' \cdot \frac{n}{s} \cdot d \cdot d \tag{2}$$

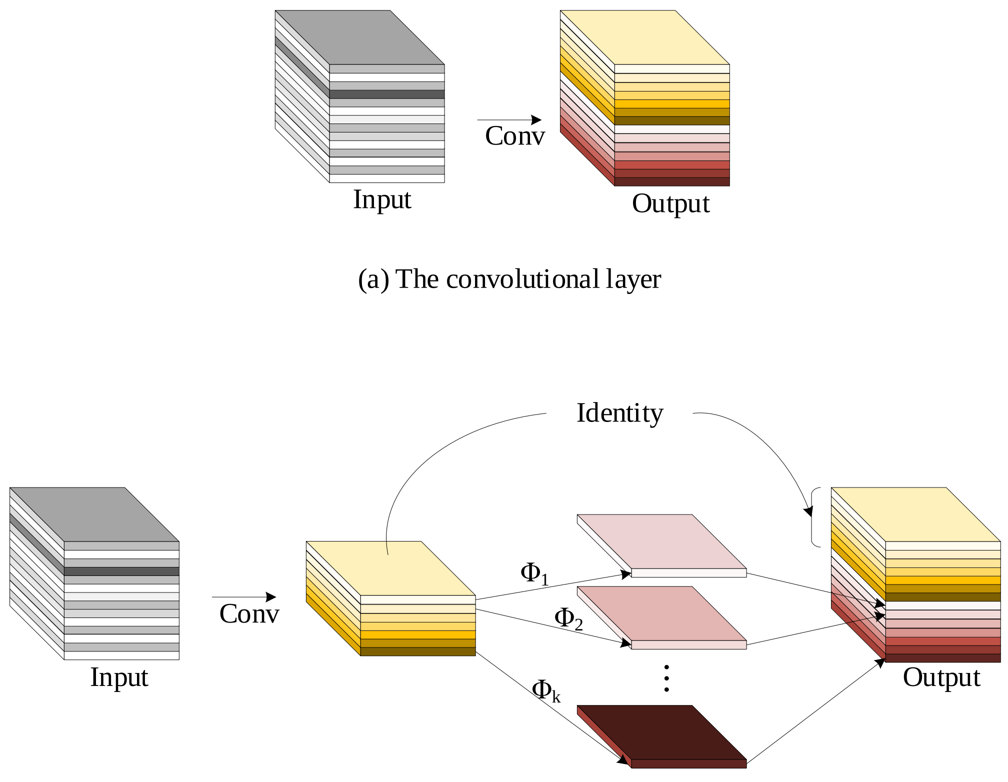
Where n is the number of output channels; h' is the height of the output features; w' is the width of the output features; c is the number of input channels; k is the size of the convolutional kernel; s is the number of feature maps generated in the Ghost module; and d is the convolutional kernel for the linear operation.

The compression of the model was calculated quantitatively, and the compression ratio was used as an index for the calculation. The calculation formula is shown in equation (3):

$$r_s = \frac{F_{conv}}{F_{ghost}} = \frac{n \cdot h' \cdot w' \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k \cdot k + (s-1) \cdot h' \cdot w' \cdot \frac{n}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} = s \tag{3}$$

Where $d \cdot d$ has similar quantities to $k \cdot k$, and $s \ll c$. It is evident from the calculations that the computation of the Ghost module is approximately s times that of standard convolution.

Using the Ghost module in combination with the original CBS module in YOLOv5, a new module is constructed as GBS. It takes the place of the CBS module's conventional convolution. Performance and efficiency can be increased in YOLOv5 by substituting the GBS module for the CBS module's function. In addition, introducing auxiliary paths



(a) The convolutional layer

(b) The Ghost module

Fig. 2. Ghost module

can provide more learning signals, which can help the model training and representation capability. In Fig. 3, the GBS structure is displayed.

Fig. 4 depicts the Ghost module structure. There are two stacked Ghost modules when the step size is set to 1. Increasing the number of channels is accomplished by using the first Ghost module as an expansion layer. The second Ghost module employs a channel reduction to match the shortest pathways. This preserves the number of input and output channels while compressing the network model. Finally, connect the two Ghost modules' inputs and outputs. The downsampling layer realizes the shortcut path when the Ghost Bottleneck's step size is set to 2. A two-step depth convolution is introduced between the two Ghost modules. The feature layer's height and width are compressed using the depth convolution, which also serves to connect the input and output.

In this paper, C3 and Ghost Bottleneck are combined to build the C3ghost module, whose structure is illustrated in Fig. 5. The C3Ghost module reduces the model parameters and computational load by combining three Conv modules with a Ghost Bottleneck structure. This reduces hardware performance requirements and makes the model easier to deploy on the edge.

C. Loss Function Improvement

In neural networks, loss functions are used to measure the difference between model predictions and real data. Choosing an appropriate loss function is important for the model to enhance detection performance and accelerate the convergence of the model. GIoU [25], DIoU [26], and CIoU [26] have been proposed to address the candidate frame regression problem. For the YOLOv5 model, CIoU is adopted for the regression loss function. CIoU is dependent on the summation of bounding box regression metrics, which comprise the aspect ratio, overlap area, and distance between the real and predicted frames. However, the direction of the mismatch between the predicted and true boxes is not considered. This may result in less effective and slower convergence and produce inferior models. Instead of using CIoU as the regression loss function, SIoU [27] is employed in this research. SIoU takes into account the angle between the predicted and real frames compared to CIoU. By introducing a vector angle, SIoU is able to measure the

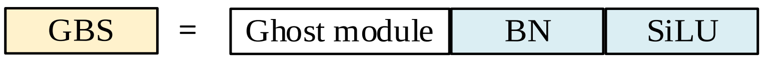


Fig. 3. Structure diagram of GBS

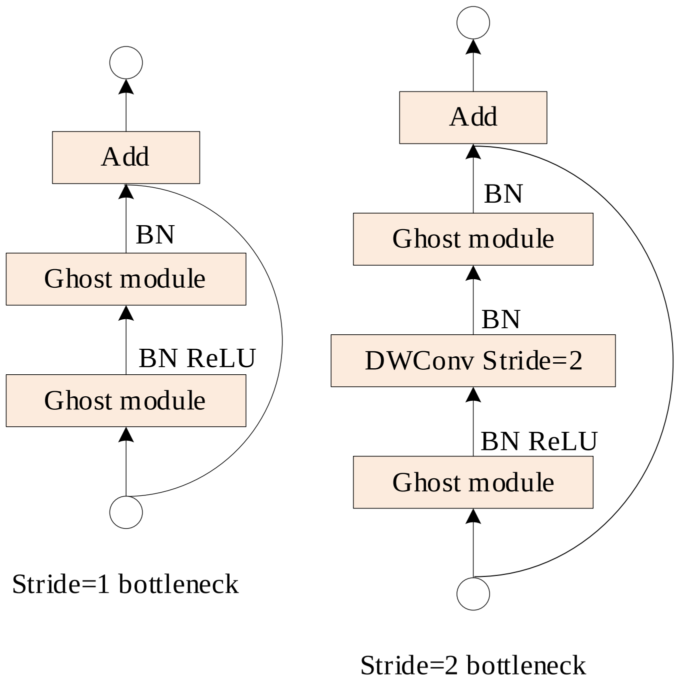


Fig. 4. Ghost Bottleneck

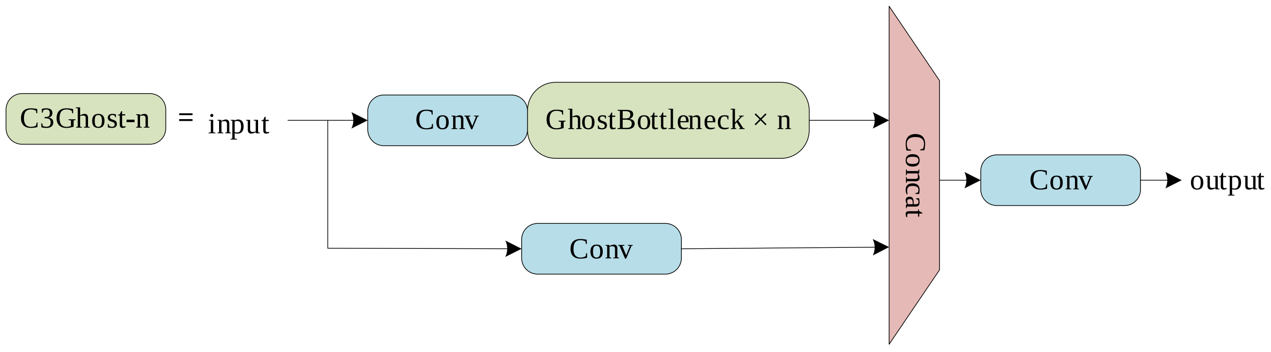


Fig. 5. C3Ghost

similarity between the predicted and real frames more comprehensively. The process of angle loss calculation is shown in Fig. 6, and its related calculations are shown in equations (4)-(8).

There are four components of SIoU, which are Λ (Angle cost), Δ (Distance cost), Ω (Shape cost), and IoU (IoU cost). Angle cost is shown in equation (4).

$$\Lambda = 1 - 2 \cdot \sin^2(\arcsin(x) - \frac{\pi}{4}) \tag{4}$$

In equation (4), $x = \frac{c_h}{\sigma}$, σ is the distance between the center point of the prediction frame and the real frame; c_h is the vertical distance between the center point of the prediction frame and the real frame.

Equation (5) illustrates the distance cost while taking into account the angular cost previously described:

$$\Delta = \sum\nolimits_{t=x,y} (1 - e^{-\gamma\rho t}) \tag{5}$$

In equation (5), $\rho_x = \left(\frac{b_{cx}^{gt} - b_{cx}}{c_w}\right)^2$, $\rho_y = \left(\frac{b_{cy}^{gt} - b_{cy}}{c_h}\right)^2$, $\gamma = 2 - \Lambda$, b_{cx}^{gt} and b_{cy}^{gt} are the horizontal and vertical coordinates of the center point of the real frame; b_{cx} and b_{cy} are the horizontal and vertical coordinates of the center point of the predicted frame; c_w is the horizontal distance between the center point of the predicted frame and the real frame. The shape cost is shown in equation (6).

$$\Omega = \sum_{t=w,h} (1 - e^{-w_t})^{\theta} \tag{6}$$

In equation (6), $\omega_w = \frac{|w - w^{gt}|}{\max(w, w^{gt})}$, $\omega_h = \frac{|h - h^{gt}|}{\max(h, h^{gt})}$; w^{gt} and h^{gt} are the width and height of the real box; ω and h are the width and height of the predicted box. A genetic algorithm is used to calculate the value of θ, which determines the extent to which shape loss should be considered. In different datasets, the value of θ can vary in the range of 2 to 4. The IoU cost is shown in equation (7).

$$IoU = \frac{|B \cap B^{GT}|}{|B \cup B^{GT}|} \tag{7}$$

In equation (7), B^{GT} is the area of the real frame; B is the area of the predicted frame. The final total definition formula for SIoU is presented in equation (8).

$$SIoU\_LOSS = 1 - IoU + \frac{\Delta + \Omega}{2} \tag{8}$$

D. *CARAFE Upsampling Operator*

A convolutional neural network's intermediary layer is frequently the upsampling module. The module extends the feature map size to improve tensor cascading and information transfer. Upsampling allows for richer details and more accurate feature representations. Upsampling can be done using a variety of methods, including inverse convolution, bilinear interpolation, bicubic interpolation, and nearest neighbor interpolation. The upsampling module is frequently applied to tasks like super-resolution, style conversion, and picture segmentation. An interpolation algorithm is used in almost all upsampling techniques. Interpolative upsampling is a process of adding new elements between pixel points to increase the resolution or change the size of an image. This results in a higher-quality image representation. Different interpolation algorithms can provide varying effects and performance during the upsampling process. It focuses only on local features and ignores global information. Furthermore, interpolation upsampling has a limited receptive field and fails to accurately capture the global features of the image. Inverse convolutional upsampling uses the same convolutional kernel to upsample the feature map. This operation cannot be tuned for features, thus ignoring some of the semantic features of the image. Consequently, this approach introduces more parameters and computational complexity, rendering it inappropriate for lightweight network models. Therefore, the nearest neighbor interpolation upsampling in YOLOv5 is replaced in this article by the CARAF [28] upsampling operator. This replacement can solve the above shortcomings and deficiencies.

CARAFE is a lightweight upsampling operator that guides the upsampling process with semantic information from the input feature map. It generates an adaptive upsampling kernel by use of a small convolutional network.

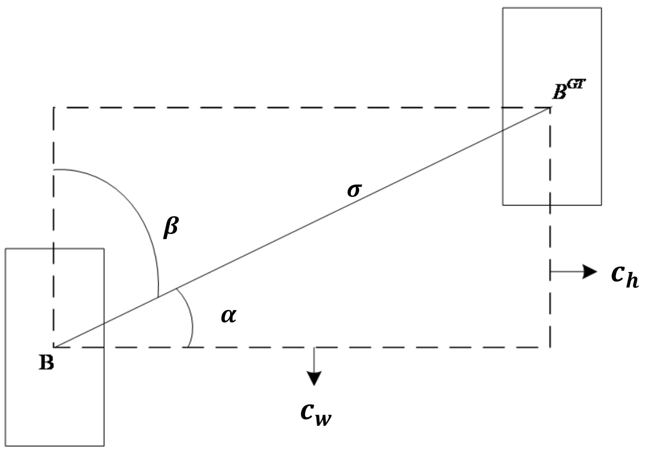


Fig. 6. Channel Exchange Module structure

This kernel performs a dot product operation with neighboring pixels in the input feature map to generate the upsampled feature map. CARAFE has a broader receptive field than conventional upsampling operators (such as the nearest neighbor operator or bilinear interpolation operator). It also has a better semantic adaptation and introduces fewer parameters.

Two components make up CARAFE, as seen in Fig. 7. The upsampling kernel prediction module is one component that produces weights for the kernels utilized in the computation of reassembly. The content-aware reassembly module is the other component. It reassembles the features using the calculated weights. The content encoder, feature map channel compression, and upsampling kernel normalization are the three submodules that make up the upsampling kernel prediction module. Using a 1×1 convolution, the feature map channel compression decreases the number of channels in the input feature map to C_m. By using this method, CARAFE operates more efficiently and requires fewer parameters overall. Reassembly kernels are generated by the content encoder according to the input feature's content. It can generate feature maps with $\sigma^2 k_{up}^2$ channels. Here, σ is the upsampling rate (typically 2) and k_{up} denotes the upsampling kernel's size. In processing, a convolutional layer with a kernel size is added to expand the encoder's receptive field. This allows it possible for the area to make better use of contextual information. Lastly, a softmax operation is applied to ensure that each channel in the upsampling kernel is normalized. Every position from the output feature map is returned to the input feature map via the feature reconstruction module. Assuming region $k_{up} \times k_{up}$ and a predicted upsampling kernel for that region, the final result after upsampling is obtained by performing a dot product operation.

E. Convolutional Attention Module

Any feedforward convolutional neural network can benefit from the simple and effective Convolutional Block Attention Module (CBAM) [29]. Integrating CBAM modules before three detection heads can synthesize global and local information, improving the model's robustness and generalizability. The CBAM module can weigh and integrate features at different scales and layers by adaptively adjusting the channel and spatial attention. The CBAM module improves the model's capacity to recognize various sizes and shapes of road damage by improving the representation and attention mechanisms of features. This results in better adaptation to different road damage variations and complexities.

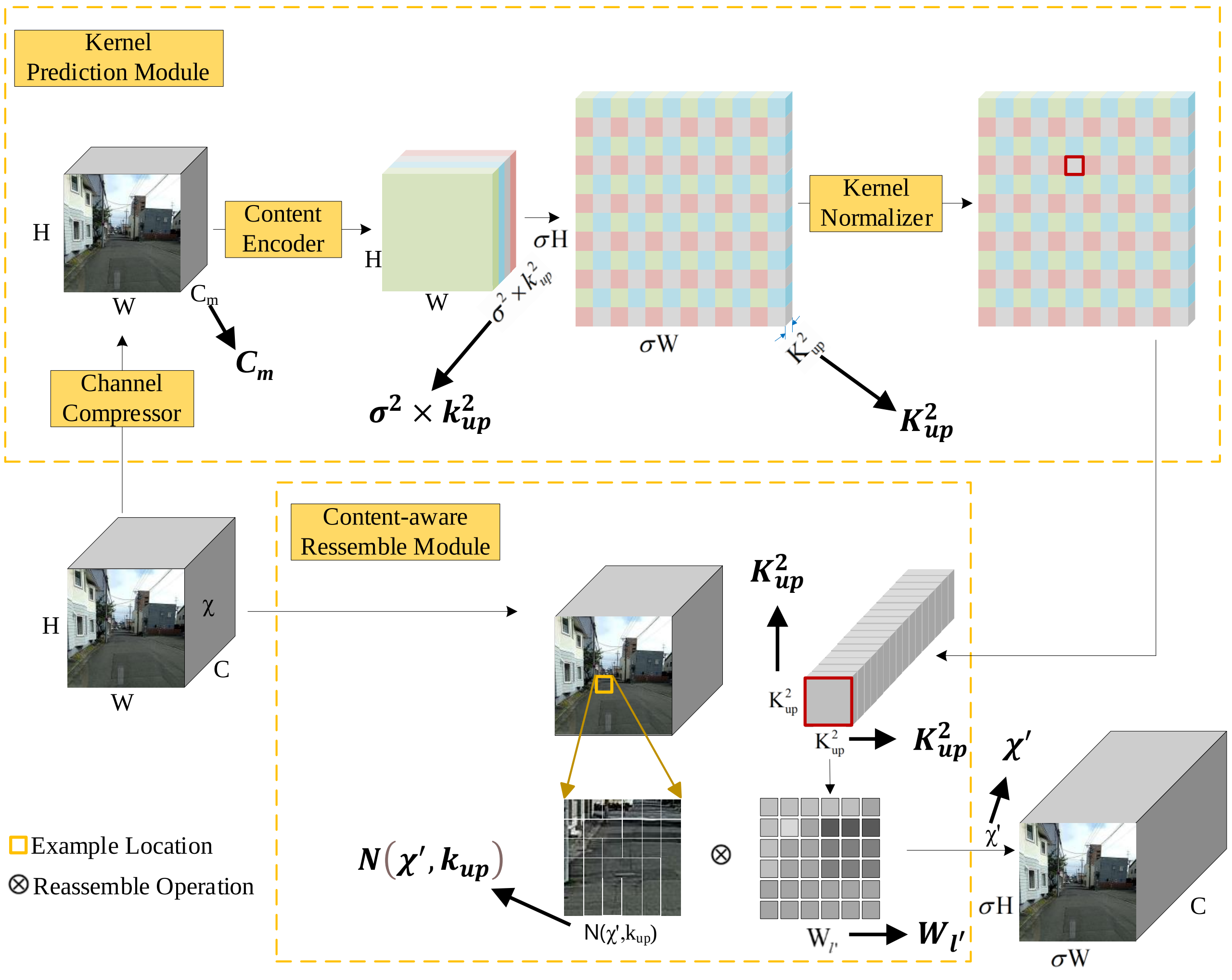


Fig. 7. CARAFE

The CBAM module employs the Channel Attention Module (CAM) and Spatial Attention Module (SAM) to facilitate the adaptive fusion of local and global road damage features. The capacity of the model to identify road damage is improved by this fusion. The channel characteristics can be represented more accurately by utilizing the Channel Attention Module (CAM). The model can adaptively learn channel attention to better capture important features of road damage, improving target discrimination and detection performance. The Spatial Attention Module (SAM) enhances the representation of spatial data by adaptively learning spatial attention. This makes it possible for the model to concentrate better on the spatial structure and location information of road damage, improving target localization precision and accuracy.

Fig. 8 shows that the two primary components of the CBAM's functioning process are channel attention and spatial attention. After processing the input information, the CAM creates the channel attention M_c and adaptively adjusts the weight of each channel. Then the channel attention M_c and the original features F are subjected to Hadamard product operation. After the operation, the CAM adjusts the high-dimensional feature $F^{'}$. Next, $F^{'}$ is processed by the SAM to produce a spatial attention vector M_s. M_s is product-multiplied with $F^{'}$ to obtain the final optimized feature $F^{''}$. The process of CBAM attention generation can be described by equation (9)(10).

$$F^{'} = Mc(F) \otimes F \tag{9}$$

$$F^{"} = M_s(F^{'}) \otimes F^{'} \tag{10}$$

Where $\otimes$ represents the Hadamard product operation. In applying channel attention and spatial attention to high-dimensional features, the attention coefficients need to be broadcast along different dimensions. Broadcasting ensures that they are aligned when performing the Hadamard product operation.

1) Channel Attention Module

Fig. 9 depicts the channel attention module.

The module begins by performing global average pooling and global maximum pooling operations on the input high-dimensional features F. A two-layer neural network (MLP) receives the pooled aggregated features F_{avg} and F_{max} as a result. The first layer has a neuronal count of C/r (where r represents the reduction rate), and employs the ReLU activation function. In the second layer, there are C neurons. It is shared with the first layer of the neural network. The output of the MLP features undergoes an element-wise sum process. The final channel attention characteristic M_c, which adaptively adjusts the weights of each channel, is the result of the sigmoid activation procedure. In the end, M_c is multiplied element-wise by the input high-dimensional feature F. Equation (11) below illustrates how the output features are fed into the spatial attention module:

$$\begin{aligned} M_c(F) &= \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \\ &= \sigma(W_1(W_0(F^c_{avg})) + W_1(W_0(F^c_{\max}))) \end{aligned} \tag{11}$$

In the equation, F is the input feature; $Mc \in R^{C\times1\times1}$ is the one-dimensional channel attention; σ denotes the sigmoid function, while MLP stands for multilayer perceptron; $W_0 \in R^{C/r\times C}$ and $W_1 \in R^{C/r\times C}$ are the parameters of the two hidden layers of the MLP; F^c_{avg} and $F^c_{\max}$ are the feature representations obtained by the two pooling processes that combine the spatial information on each channel.

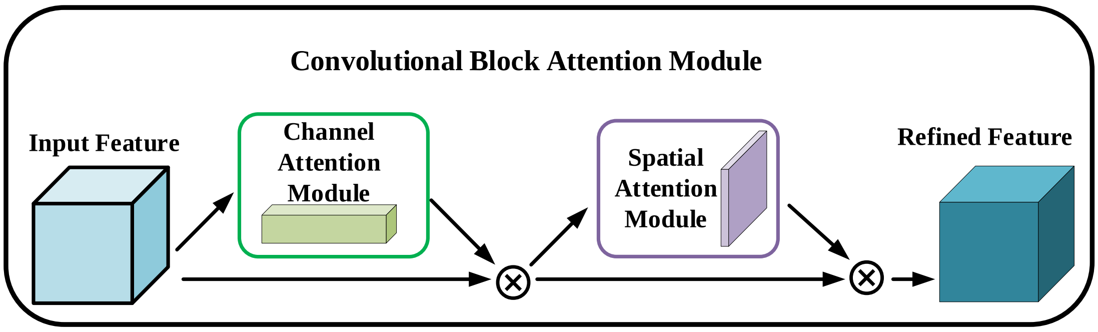


Fig. 8. CBAM

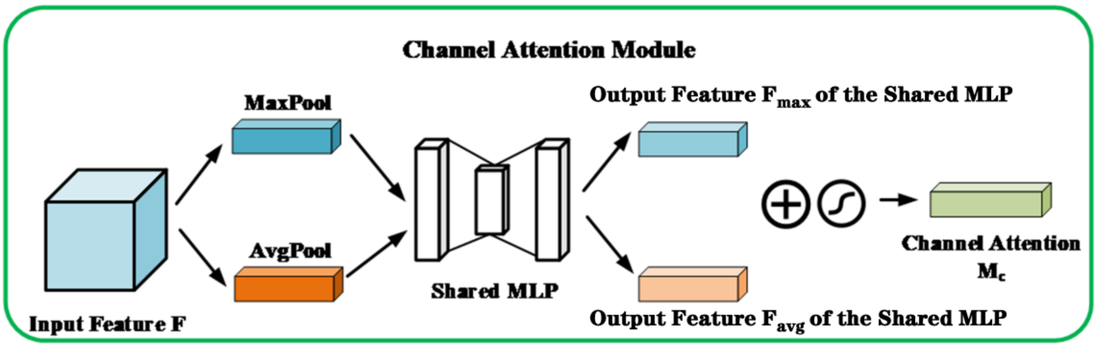


Fig. 9. Channel Attention Module

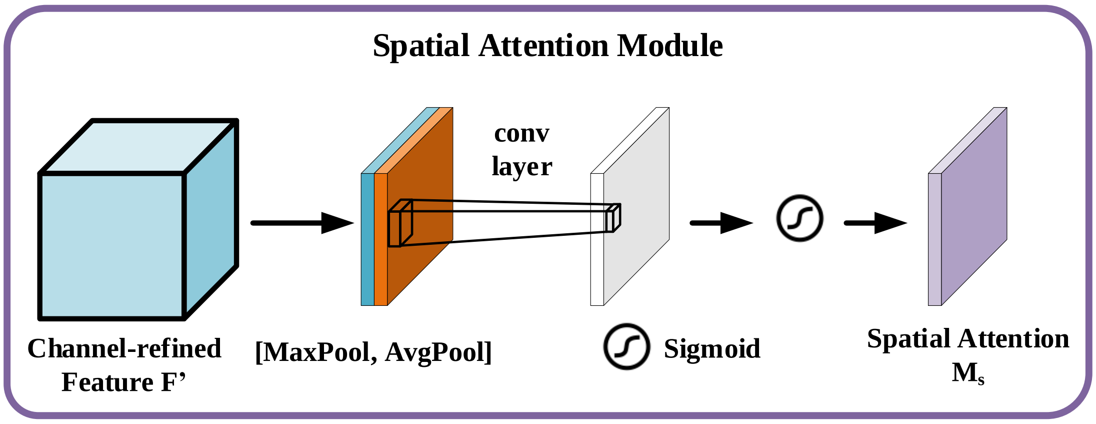


Fig. 10. Spatial Attention Module

2) *Spatial Attention Module*

Fig. 10 depicts the spatial attention module. The high-dimensional features F' output serve as the input features for the spatial attention module. To get two *H*W*1* features, F^s_{avg} and $F^s_{\max}$, first perform a channel-based global average pooling and global maximum pooling operation. Then, divide these two feature maps by channel. The dimensionality diminishes to one channel, i.e., *H*W*1* after a 7×7 convolution procedure. The weight of each spatial location can be adaptively changed when a sigmoid function generates the spatial attention feature M_s. To achieve the final generated feature, the module multiplies this feature by the input feature at the end. The specific equation (12) is as follows.

$$\begin{aligned} M_s(F) &= \sigma(f^{7\times7}([AvgPool(F);+MaxPool(F)]) \\ &= \sigma(f^{7\times7}[F^s_{avg};F^s_{\max}])) \end{aligned} \tag{12}$$

Where F is the input feature; $M_s \in R^{C\times1\times1}$ is the two-dimensional spatial attention; $f^{3\times3}_{dilat}$ is the null convolution operation with a convolution kernel size of 3; F'^s_{avg} and $F'^s_{\max}$ are the feature representations obtained by aggregating the given information at each spatial location for both poolings.

IV. Experiments

The hardware configuration consists of Intel(R) Xeon(R) Silver 4210R CPU@2.40GHz, GPU RTX 3090, and 24 GB video memory. The software configuration includes Windows 10 and Cuda 11.3. The deep learning framework platform is Pytorch 1.10.0, Python 3.8. For all images to comply with the model's input requirements, they were resized to 640 × 640 pixels. The computer hardware's corresponding batch size was fixed at 32. An SGD optimizer was used for perfecting the network. To minimize training time, we utilize the migration learning method and begin model training with pre-training weights provided by official sources.

A. *Experimental Dataset*

The paper evaluates the road damage detection network by analyzing the global Road Damage Detection Challenge (RDD2020) dataset. A rich sample of road damage is provided by the dataset, which consists of 21,041 annotated photos from the Czech Republic, Japan, and India. The information on road damage is composed of labels and bounding box coordinates. The kind of damage connected to the bounding box is described by these coordinates. In this work, the training and validation sets were split into 16,833 and 4,208 photos, respectively, at random in an 8:2 ratio. This dataset covers a variety of common roadway damages such as cracks, potholes, etc. The dataset is shown in Table 1.

TABLE I
Experimental Data

Class Name	Category Details	Number of Training Samples	Number of Validating Samples
D00	Longitudinal Crack	5230	1362
D01	Longitudinal Spike Crack	133	46
D10	Transverse Crack	3562	884
D11	Transverse Spike Crack	32	13
D20	Alligator Crack	6714	1667
D40	Rutting,Bump,Pothole,Separation	4506	1121
D43	Blurred Crosswalk	642	151
D44	Blurred Lane Line	4071	986
D50	Manhole Cover	2842	739

B. *Evaluation Indicators*

1) *Precision (P) and Recall (R)*

To objectively evaluate the outcomes of the experiments, the model's performance is assessed with two commonly used metrics: precision (P) and recall (R). The chance of accurately predicting a positive sample from all anticipated positive samples is known as precision (P). The likelihood of correctly anticipating a positive sample from actual positive samples is known as recall (R). Equations (13) and (14) display the formulas for precision (P) and recall (R).

$$P=\frac{TP}{TP+FP} \tag{13}$$

$$R=\frac{TP}{TP+FN} \tag{14}$$

When a model accurately predicts positive examples, it is referred to as a True Positive (TP). False Positives (FP) are instances where the model predicts positively while the data is actually negative. When positive cases are mistakenly forecasted by the model as negative ones, this is known as a False Negative (FN). When negative examples are accurately predicted by the model, they are referred to as True Negatives (TN).

2) *mAP and F1-Score*

The Average Precision (AP) in this study serves as a metric for detecting accuracy. To assess the model more thoroughly, the F1-Score comprehensive evaluation index is employed. The trade-offs between precision and recall are balanced by the F1-Score, a reconciled average of the two that shows greater scores indicate better model performance. High AP and F1-Score values correlate due to greater network accuracy, whereas mean Average Precision (mAP) represents the average precision across all classes. The equations for AP, F1-Score, and mAP are shown in equations (15), (16) and (17).

$$AP=\int_0^1 P(R)dR \tag{15}$$

$$mAP=\frac{1}{n}\sum_{i=1}^{m} AP^i \tag{16}$$

$$F1-Score=2\times\frac{P\times R}{P+R} \tag{17}$$

3) *Parameters and FLOPs*

Additionally, the model's lightweight effect is evaluated based on the number of parameters (Parameters) and floating-point computations (FLOPs). Parameters refer to the model's parameter count, which impacts the network model's size. FLOPs determine the model's speed and are commonly used to measure its complexity.

C. *Experimental Results*

1) *Ablation Experiments*

We carried out an ablation experiment to show the value and requirement of each improvement module in the model. The benchmark model for this study was YOLOv5. The improvement modules are added gradually for ablation experiments. Every experiment assesses the models for comparison using the mAP, F1-score, Parameters, and FLOPs. Table 2 displays the experimental results.

We first lightweighted the model to minimize the model's Parameters and FLOPs. In our experimental setup, the original feature fusion's CBS module and C3 module were substituted with the GBS module and the C3Ghost module, respectively. Parameters and FLOPs decreased to 3.7 and 8.3 respectively. The significant decline is because the Ghost module uses simple linear operations to replace some more complex convolutional operations. Conversely, the mAP exhibited a 2.8% decline, while the F1-Score demonstrated a 1.6% reduction.

To address the diminished accuracy resulting from lightweight operation, we initially altered the CIoU of the original YOLOv5 model to SIoU. In contrast to CIoU, SIoU incorporates the angle between the predicted and actual bounding boxes into its calculations. After the lightweight operation, the model improves accuracy by 2.2% while keeping the same amount of model parameters and floating-point calculation. Secondly, the CARAFE module was replaced with the original nearest neighbor interpolation in YOLOv5. This replacement exhibits a greater sensory field and superior semantic adaptability. The model obtained an F1-score of 57.6% and a mAP of 55.9%. Lastly, the model incorporates a convolutional block attention module

TABLE II
RESULTS OF ABLATION EXPERIMENT

①	②	③	④	⑤	mAP，%	F1-Score，%	Parameters，M	GFLOPs
√	×	×	×	×	56	58.2	7	16.0
√	√	×	×	×	53.2	56.6	3.7	8.3
√	√	√	×	×	55.4	57.3	3.7	8.3
√	√	√	√	×	55.9	57.6	3.9	8.5
√	√	√	√	√	56.2	57.8	4.1	9.0

(CBAM) in front of the three detection heads. This enables the model to more effectively integrate global and local information. Building upon the preceding enhancements, the model exhibited a 0.3% improvement in mAP. In the end, the enhanced and comprehensive model yielded an F1-score of 57.8% and a mAP of 56.2%. The modified YOLOv5 model exhibits enhanced performance relative to its predecessor. The mAP increased by 0.2%, while Parameters decreased by 41.8% and GFLOPs decreased by 43.8%.

In Table 2, ① shows the YOLOv5 baseline model; ② shows the lightweight processing operation; ③ shows the loss function's enhancement using SIoU; ④ shows the application of the lightweight CARAFE upsampling operator; ⑤ shows the introduction of the CBAM attention mechanism.

2) Performance Comparison of Different Algorithms

We compared the Parameters and FLOPs of various algorithms, as shown in Table 3. According to experiments, the lightweight road damage detecting network described in this article has Parameters and FLOPs of only 4.1 and 9.0, respectively. Compared with the baseline model YOLOv5, it is 41.8% and 43.8% less respectively. Compared with Faster R-CNN, the Parameters and FLOPs are 34.2 and 25.6 less. Compared with EfficientDet, the Parameters and the GFLOPs are reduced. Not only are the Parameters smaller than literature [19], literature [20], and literature [21], but the GFLOPs are also significantly less.

To validate the algorithm's efficacy, it was compared to three other papers, YOLOv5, Faster R-CNN, EfficientDet, and other target detection methods. The RDD2020 dataset, which was processed using the technique outlined in this research, was employed to train and validate the aforementioned algorithms. To evaluate the algorithms, four metrics were chosen: precision (P), recall (R), mAP, and F1-Score. The assessment of these metrics enables a comprehensive comparison and analysis of the algorithms, further validating the feasibility of the methodology in this paper. The RDD2020 dataset's experimental findings for several road damage detection algorithms are displayed in Table 4.

The experiment's findings indicate that the lightweight road damage detection network this study proposes has a 56.2% detection accuracy. The model presented in this research maintains a small gap or even slightly improves detection precision while decreasing the Parameters and FLOPs. Comparing the mAP to the baseline model YOLOv5 shows a 0.2% improvement. Compared with Faster R-CNN, mAP, and F1-Score rose by 4% and 6.4% respectively. Moreover, the proposed network has demonstrated better performance compared to the literature [21] both mAP and F1-Score.

TABLE III
COMPARISON OF PARAMETERS AND GFLOPS OF VARIOUS ALGORITHMS

Model	Parameters，M	GFLOPs
YOLOv5	7	16.0
Faster R-CNN	38.3	34.6
EfficientDet	6.6	11.6
Literature[19]	11	6.6
Literature[20]	19.8	17.4
Literature[21]	4.2	32.7
Ours	4.1	9.0

TABLE IV
PERFORMANCE COMPARISON OF VARIOUS ALGORITHMS,%

Model	P	R	mAP	F1-Score
YOLOv5	58.5	58	56	58.2
Faster R-CNN	57.6	48.6	51.2	51.4
EfficientDet	57.8	55.5	56.9	57.2
Literature[19]	60	58.5	57.4	59.2
Literature [20]	59.2	58.2	57.6	58.7
Literature[21]	54.8	53	52.4	54.3
Ours	60	55.7	56.2	57.8

3) The Loss Value

The loss value curves of the enhanced YOLOv5 model for the training and validation sets are illustrated in Fig. 11. The efficiency of the model training is indicated by the trend of the loss value with the number of iterations. At the completion of training, the model performs better the closer the loss value is near 0. The training and validation sets' loss values gradually converge after 100 iterations. The change in the loss value is relatively smooth after 200 iterations. The training and validation sets' loss values show that the enhanced model training is more effective. The model exhibits a stronger generalization ability overall.

V. Conclusion

First, the Ghost module is used to complete the processing for network lightweight operation. Second, this paper adopts SIoU, CARAFE lightweight upsampling operator, and CBAM attention module. These modules are added to make up for the precision that is lost as a result of the model's lightweight treatment. In comparison to the baseline model, these enhancements led to a reduction of 41.8% and 43.8% in Parameters and FLOPs, respectively. Meanwhile, the detection accuracy is increased by 0.2%.

Road damage detection is crucial for road maintenance and traffic safety. This work explores a lightweight network-based method for road damage detection to increase detection efficiency and decrease computational load. It is an efficient, accurate, and less computational overhead method. An effective improvement method is proposed based on the YOLOv5 model for the characteristics of road damage. The method utilizes the advantage of a lightweight network in feature learning to extract key features in road images, such as cracks and potholes. This study completely takes into account the trade-off between detection performance and computing efficiency while designing algorithms. The computational burden of the model is further reduced while maintaining a high detection accuracy. The technique maintains good accuracy while requiring a large reduction in processing resources as compared to traditional approaches. Therefore, the algorithm has potential value in practical applications for real-time monitoring and maintenance of road damage.

In conclusion, the road damage detection technique based on a lightweight network is explored in this research. An efficient and accurate method with low computational overhead is proposed. The algorithm has potential applications in road damage detection tasks and helps to improve the efficiency of traffic safety and road maintenance.

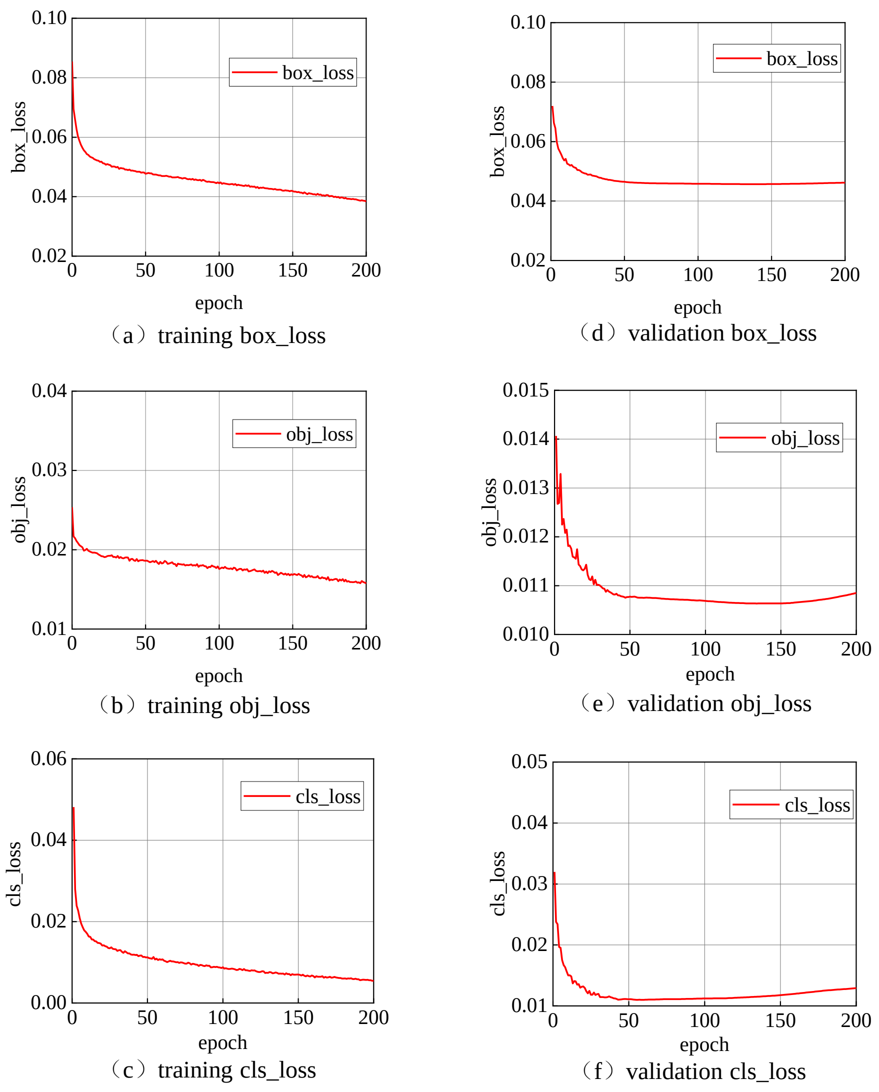


Fig. 11. Loss curve during training and validation process

REFERENCES

[1] A. Hosseini, A. Faheem, H. Titi, et al, "Evaluation of the long-term performance of flexible pavements with respect to production and construction quality control indicator," Construction and Building Materials, vol. 230, pp. 116998, 2020.

[2] A. Pasha, A. Mansourian, M. Ravanshadnia, "Evaluation of work zone road user cost of pavements based on rehabilitation strategy approach," Journal of Transportation Engineering, Part B: Pavements, vol. 147, no. 2, pp. 04021015, 2021.

[3] Y. Wang, K. Song, J. Liu, et al, "RENet: Rectangular convolution pyramid and edge enhancement network for salient object detection of pavement cracks," Measurement, vol. 170, pp. 108698, 2021.

[4] A. Akagic, E. Buza, S. Omanovic, et al, "Pavement crack detection using Otsu thresholding for image segmentation," *Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1092-1097, 2018.

[5] F. Liu, G. Xu, Y. Yang, et al, "Novel approach to pavement cracking automatic detection based on segment extending," *Proceedings of the 2008 International Symposium on Knowledge Acquisition and Modeling*, pp. 610-614, 2008.

[6] G. X. Hu, B. L. Hu, Z. Yang, et al, "Pavement crack detection method based on deep learning model," *Wireless Communications and Mobile Computing*, vol. 32, no. 8, pp. 1-13, 2021.

[7] Y. Maode, B. Shaobo, X. Kun, et al, "Pavement crack detection and analysis for high-grade highway," *Proceedings of the 2007 8th International Conference on Electronic Measurement and Instruments*, pp. 548-552, 2007.

[8] N. D. Hoang, "An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction," *Advances in Civil Engineering*, vol. 2018, no. 1, pp. 7419058, 2018.

[9] M. Gao, X. Wang, S. Zhu, et al, "Detection and segmentation of cement concrete pavement pothole based on image processing technology," Mathematical Problems in Engineering, pp. 1-13, 2020

[10] W. Wang, B. Wu, S. Yang, et al, "Road damage detection and classification with faster R-CNN," *Proceedings of the 2018 IEEE International Conference on Big Data*, pp. 5220-5223, 2018.

[11] S. Gupta, P. Sharma, D. Sharma, et al, "Detection and localization of potholes in thermal images using deep neural networks," *Multimedia Tools and Applications*, vol. 79, no. 8, pp. 26265-26284, 2020.

[12] X. Yang, H. Li, Y. Yu, et al, "Automatic pixel-level crack detection and measurement using fully convolutional network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 12, pp. 1090-1109, 2018.

[13] H. T. Nguyen, L. T. Nguyen, A. D. Afanasiev, et al, "Classification of Road Pavement Defects Based on Convolution Neural Network in Kera," *Automatic Control and Computer Sciences*, vol. 56, no. 1, pp. 17-25, 2022.

[14] J. W. Baek, K. Chung, "Pothole classification model using edge detection in road image," *Applied Sciences*, vol. 10, no. 19, pp. 6662, 2020.

[15] P. Ping, X. Yang, Z. Gao, "A deep learning approach for street pothole detection," Proceedings of the 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService), pp. 198-204, 2020.

[16] Y. Du, N. Pan, Z. Xu, et al, "Pavement distress detection and classification based on YOLO network," *International Journal of Pavement Engineering*, vol. 22, no. 13, pp. 1659-1672, 2021.

[17] H. Tsuchiya, S. Fukui, Y. Iwahori, et al, "A method of data augmentation for classifying road damage considering influence on classification accuracy," *Procedia Computer Science*, vol. 159, pp. 1449-1458, 2019.

[18] Y. Li, Z. Han, H. Xu, et al, "YOLOv3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions," *Applied Sciences*, vol. 9, no. 8, pp. 3781, 2019.

[19] H. Liang, S. C. Lee, S. Seo, "Automatic recognition of road damage based on lightweight attentional convolutional neural network," *Sensors*, vol. 22, no. 24, pp. 9599, 2022.

[20] F. Wan, C. Sun, H. He, et al, "YOLO-LRDD: A lightweight method for road damage detection based on improved YOLOv5s," *EURASIP Journal on Advances in Signal Processing*, vol. 2022, no. 1, pp. 1-18, 2022.

[21] J. Chen, X. Yu, Q. Li, et al, "LAG-YOLO: Efficient road damage detector via lightweight attention ghost module," *Journal of Intelligent Construction*, vol. 2, no. 1, pp. 9180032, 2024.

[22] Jie Cao, Penghui Li, Hong Zhang, and Guang Su, "An Improved YOLOv4 Lightweight Traffic Sign Detection Algorithm," IAENG International Journal of Computer Science, vol. 50, no. 3, pp. 825-831, 2023.

[23] K. Han, Y. Wang, Q. Tian, et al, "Ghostnet: More features from cheap operations," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1580-1589, 2020.

[24] M. L. Li, G. B. Sun, J. X. Yu, "A pedestrian detection network model based on improved YOLOv5," Entropy, vol. 25, no. 2, pp. 381, 2023.

[25] H. Rezatofighi, N. Tsoi, J. Y. Gwak, et al, "Generalized intersection over union: A metric and a loss for bounding box regression," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 658-666, 2019.

[26] Z. Zheng, P. Wang, W. Liu, et al, "Distance-IoU loss: Faster and better learning for bounding box regression," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 12993-13000, 2020.

[27] Z. Gevorgyan, "SIoU loss: More powerful learning for bounding box regression," arXiv, vol. 2205, pp. 12740, 2022.

[28] J. Wang, K. Chen, R. Xu, et al, "Carafe: Content-aware reassembly of features," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3007-3016, 2019.

[29] S. Woo, J. Park, J. Y. Lee, et al, "Cbam: Convolutional block attention module," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3-19, 2018.