

## RESEARCH ARTICLE

# Research on Detection of Multiple Types of Speed Bump Defects Based on CRSCCG-YOLOv5s

XINJIAN XIANG<sup>1</sup>, (Member, IEEE), XIZHAO CHEN<sup>1</sup>, XIAOHUI TANG<sup>2</sup>, QIUXIA LUO<sup>2</sup>, AND YI DING<sup>1</sup>

<sup>1</sup>School of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China

<sup>2</sup>Zhejiang Ansheng Science and Technology Stock Company Ltd., Jinhua 321300, China

Corresponding author: Xinjian Xiang (188002@zust.edu.cn)

This work was supported in part by Zhejiang Provincial Key Research and Development Program under Grant 2024C01071 and in part by Zhejiang University of Science and Technology 2023 Postgraduate Research Innovation Fund Projects under Grant 2023yjskc05.

**ABSTRACT** Speed bumps, as a crucial road safety infrastructure, can directly impact the accident rates on specific road sections if they have defects. Therefore, this study proposes a method for detecting speed bump multiclass defects using an improved YOLOv5s model. This model modifies the pooling method in the backbone layer to Spatial Pyramid Pooling Fast (SimSPPF) to enhance the detection speed while improving the feature extraction capability from images. In the neck layer, firstly, the upsampling method was changed to Convolutional Transpose (ConvTranspose) to increase the accuracy of small target detection. Then, the Contextual operation network (CotNet) module was integrated with the original C3 module, enhancing the model's ability to recognize different defects by obtaining better global features. Finally, the convolution module was restructured to a Recursive Gated Convolution (gnconv) module, designed to maximize the model's capacity to capture complex multi-scale, multiclass image features. Additionally, a new data augmentation method (CR) was proposed for data enhancement and balancing of the samples. Experimental results show that the improved YOLOv5s algorithm achieved an accuracy of 97.7%, a recall rate of 91.9%, and a mean precision of 96.4% while maintaining a parameter size of only 8.8M. Compared to other YOLO detection models, the improved model exhibits high accuracy, high confidence, and a low parameter count.

**INDEX TERMS** Speed bump defect detection (SBD), YOLOv5s, spatial pyramid pooling fast, convolutional transpose, recursive gated convolution, data augmentation.

## I. INTRODUCTION

As a crucial component of road infrastructure, detecting defects in speed bumps is crucial for road safety, especially under complex weather conditions or on challenging road segments, such as long downhill roads, roads after rain or snow, and short downhill segments in places like underground garages. If speed bumps have defects, they may pose serious safety hazards. However, current research primarily focuses on the identification of speed bumps. For instance, Al-Shargabi et al. [1] proposed a method using accelerometer sensors to detect road bumps. Yun et al. [2] utilized LIDAR and cameras for speed bump detection in autonomous vehicles. These studies do not specifically address the detection

of defects in speed bumps. Therefore, this paper proposes an improved method for studying speed bump defects based on YOLOv5s, referencing the standards for rubber speed bumps issued by the Ministry of Transportation of China in 2004 and 2008.

Early defect detection methods commonly included simple sensor detection and regular manual inspections. While these methods are convenient and feasible, they have significant disadvantages in terms of cost and time efficiency. With advancements in computer vision, traditional machine learning detection methods and deep learning detection techniques have gradually gained widespread acceptance.

Traditional machine learning methods for defect detection research include the Viola-Jones detector [3] (integral images, Haar-like features, Adaboost classifier, and cascade

The associate editor coordinating the review of this manuscript and approving it for publication was Shaohua Wan.

strategy), HOG features [4] (Histogram of Oriented Gradients that capture texture and shape information), and SURF [5] (Speeded-Up Robust Features, which build scale-space, detect interest points, estimate orientation, and compute descriptors). These methods are highly interpretable, require less data, and are fast to train. However, their performance is limited in handling nonlinear relationships and large-scale data due to their reliance on manually designed features, which struggle to capture high-level semantic information. Compared to deep learning approaches, they are limited in accuracy and generalization.

In the field of target defect detection, deep learning algorithms initially employed Convolutional Neural Networks (CNN) [6] based on classical models. Today's commonly used detection models mainly fall into two categories: Two-stage and One-stage models. Two-Stage models include R-CNN [7], Fast R-CNN [8], Faster R-CNN [9], and models that use ResNet [10] as a backbone. These Two-stage networks have made significant advancements in the field of object detection. R-CNN performs excellently in terms of accuracy but is computationally slow. Fast R-CNN introduced the ROI pooling module to improve speed, and Faster R-CNN further improved detection speeds by incorporating a Region Proposal Network (RPN) to automatically generate candidate regions. ResNet, often used as a feature extractor, is also widely employed in backbone networks, such as in the FC-MSCCD algorithm proposed by Alnaasan and Kim [11]. However, they still share some common drawbacks, such as high computational complexity, the need for extensive labelled data, and long training times.

One-stage network models such as SSD (Single Shot MultiBox Detector) [12], RetinaNet [13], EfficientDet [14], and the YOLO (You Only Look Once) series [15] detect targets of various sizes through convolution operations on different feature maps and simultaneously predict the class and location of each prior box, achieving multi-scale target detection. These one-stage detection networks generally have slightly lower accuracy than two-stage networks, particularly in detecting small targets and complex scenes. However, their advantages include a simplified process, treating object detection as an end-to-end regression problem, which eliminates the need for generating candidate regions, reduces network size, and increases detection speed, making them suitable for applications requiring rapid response.

The YOLO series has two main advantages in defect detection. First, it uses a single neural network to simultaneously predict the categories and locations of multiple objects, requiring only one forward pass, thus significantly reducing computational complexity and improving detection efficiency. Second, it directly outputs class and location information without the need for additional modules like Support Vector Machines (SVM) [16] or Bounding Box Regression (BBR), thus avoiding multi-module coupling and error accumulation and enhancing detection accuracy. These features have made YOLO a widely used algorithm in the field of defect detection. The YOLO series has

been continually improved by researchers, resulting in multiple versions (YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv7, YOLOv8) [17], [18], [19], [20], [21], [22], and has become a mainstream algorithm in the field of computer vision for defect detection.

The YOLOv5 series algorithms are highly applicable. It includes multiple versions (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x) [20], with YOLOv5s being the smallest and fastest, while YOLOv5x is the largest but more accurate. These versions primarily differ in network depth and width. Moreover, as the model parameters increase, detection accuracy improves, but detection speed decreases. YOLOv5s offers higher accuracy, faster speed, and fewer parameters compared to other models, with advantages in parameter count and GFLOPs. Therefore, YOLOv5s is chosen as the base network architecture. According to the standards issued by the Ministry of Transportation of China regarding speed bumps, the defects are categorized based on length, width, and height into askew and deformation and based on appearance, colour, and shape into damage and missing parts. The improvement approach aims to enhance the model's ability to recognize defect types and to better extract input features under complex weather and varying brightness conditions. These improvements include modifications to the SimSPPF, ConvTranspose, CotNet, and gnConv modules. A new data augmentation method, CR, is used to address the limitations of the existing dataset. The improved model can detect defects in speed bumps and can be deployed on embedded mobile devices, thus providing a measure of safety for vehicles on the road while identifying defective speed bumps.

## II. RELATED WORK

### A. DATASET INTRODUCTION

The dataset used in this experiment is a custom dataset created from video clips of problematic speed bumps captured by DJI drones in various complex scenarios, such as parking lots, construction sites, and traffic-heavy roads. During the actual filming, it was observed that problematic speed bumps are less common compared to standard ones, with most exhibiting only alignment defects. Speed bumps with severe damage, missing sections, or deformation defects were even rarer, leading to uneven sample distribution in model training. This could result in poor fitting effects and limited generalization capability of the model. Therefore, data augmentation is necessary to balance the sample distribution for speed bump defects. Additionally, the captured footage was frame-separated at a rate of 5 frames per second, resulting in a total of 2121 original images. According to the standards issued by the Ministry of Transportation of China for rubber speed bumps, the dataset was categorized into five major classes: damage, askew, speed bump, lack, and deformation, as shown in Fig. 1. Finally, CR data augmentation was applied to the 2121 original images, and after filtering and removing unsuitable images, the final dataset comprised 3820 images.



FIGURE 1. Dataset image example.

Data labeling was performed using Labelling, with annotations divided into five categories corresponding to the aforementioned classes: damage, askew, Speedbump, lack, and deformation. The annotation process employed a two-step approach, where the presence of a speed bump is first detected, followed by the identification and classification of its specific defects. Additionally, the dataset was divided into training, testing, and validation sets in an 8:1:1 ratio. After categorization, the annotated dataset of speed bump defects was named SBD (Speed Bump Defects).

**B. CR DATA AUGMENTATION**

Traditional data augmentation methods such as Mosaic data augmentation and Mixup data augmentation have improved model performance to a certain extent, but they have not effectively addressed the challenge of uneven distribution of data samples. Therefore, this paper, based on Mosaic and Mixup, designs a data augmentation method tailored for the distribution of speed bump defect data samples, known as CR (Color space transformation and Random erasing) data augmentation. The CR data augmentation method includes “C” (Color space transformation) and “R” (Random erasing).

Color space transformation involves converting an image from one color space to another, which helps change the color representation of the image. Common color spaces include RGB (Red, Green, Blue) and HSV (Hue, Saturation, Value). In this experiment, the purpose of designing color space transformations is to assist the model in better learning the features of speed bump defects under various lighting and color conditions, thereby enhancing the diversity of training data types. This approach addresses the issue of uneven sample type distribution and improves the model’s generalization

ability under different environmental conditions and color variations. The comparison of images before and after color space transformation is shown in Fig. 2, and the formula for the color space transformation is given in Equation (1), where the input image is RGB and the output image is HSV; 60 represents the angle size for hue segmentation;  $H$  represents the position on the color wheel, i.e., hue, calculated by comparing the maximum and minimum RGB components;  $S$  represents the purity or concentration of the color, i.e., saturation, calculated by the ratio of the maximum and minimum RGB component values;  $V$  represents the brightness or value, directly taken as the maximum value of the RGB components.

$$[HSV] = \begin{cases} [H] = \begin{cases} \frac{60[R - G]}{V - MIN} & (MAX = R) \\ 120 + \frac{60[G - R]}{V - MIN} & (MAX = G) \\ 240 + \frac{60[B - G]}{V - MIN} & (MAX = B) \\ undefined & (V = 0) \end{cases} \\ [S] = \begin{cases} 0 & (V = 0) \\ \frac{MAX - MIN}{MAX} & (V \neq 0) \end{cases} \\ [V] = MAX \\ MAX = MAX(R, G, B) \setminus MIN = MIN(R, G, B) \end{cases} \tag{1}$$

Random erasing is performed by randomly deleting a portion of an image and filling the erased area with randomly generated pixels. Initially, the image is loaded using the PIL library and converted into a NumPy array, which is then transformed into a PyTorch tensor. Subsequently, a function is called to perform the random erasing operation, which





(a) Original Image (b) Color Space Transformation

FIGURE 2. Color space transformation diagram.

includes specifying the erasing probability, size range, aspect ratio range, and the fill color for the erased area. The processed PyTorch tensor is then converted back into a PIL image and saved to the designated output folder. This random erasing data augmentation method can expand the dataset for speed bumps, particularly for categories with missing or damaged samples, by balancing the data samples and enhancing the model’s ability to recognize partial absences or changes in regions of the image. The transformation of images by random erasing can be seen in Fig. 3, and the formula for random erasing is shown in Equation (2).

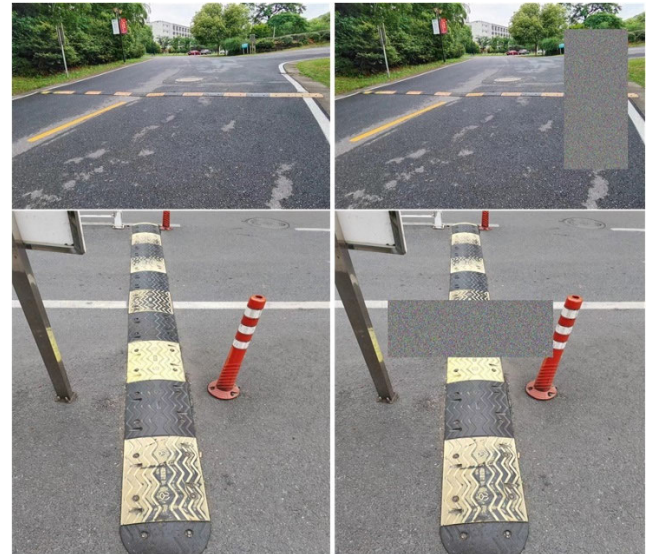
Where, ‘target\_width’ and ‘target\_height’ respectively represent the width and height of the random erasing area; ‘start\_x’ and ‘start\_y’ respectively represent the starting coordinates of the random erasing box; ‘fill\_color’ represents the fill used for the random erasing.

$$I_{out} = erase \begin{cases} I_{in} \\ start\_x, start\_y \\ target\_width, target\_height \\ fill\_color \end{cases} \quad (2)$$

### III. BASIC MODEL

#### A. YOLOv5s

YOLOv5s is a one-stage object detection algorithm that extracts features through a stack of convolutional layers [23], processing images in an end-to-end manner via a convolutional neural network to regress the categories and positions of targets. The framework of the YOLOv5s algorithm mainly consists of three parts: the Backbone (which serves as the



(a) Original Image (b) Random Erasing

FIGURE 3. Random erasing demonstration.

main feature extractor), the Neck (which aggregates and refines features), and the Prediction Head (which outputs the detections). See Fig. 4 for an illustration.

#### B. BACKBONE NETWORK

The backbone network employs the core architecture of the Darknet53 network and consists of key modules, including convolutional layers, CBS modules, CSP modules [24], and SPPF modules. The convolutional layers utilize filters of varying sizes to capture multi-scale features within the image. The CBS module (Cross-Stage Backbone Module) consists of a convolution (Conv), batch normalization (BN), and SiLU activation function [25], as illustrated in Fig. 4(a). This module optimizes the flow of information between feature maps. Its cross-stage connections allow feature maps from different stages to interact, enhancing the network’s ability to understand both global and local features of the image. The CSP module (Cross-Stage Propagation Module), depicted in Fig. 4(b) and 4(d), comprises two forward branches (a main branch and a sub-branch) combined with a Bottleneck stacking module. It facilitates the multi-level integration of features from different stages through the convolutional structure of the main and sub-branches and residual connections, thereby improving the model’s ability to understand multi-scale features and semantic information. The SPPF module (Spatial Pyramid Pooling Feature Module) [26], shown in Fig. 4(c), generates multi-scale feature representations through pyramid pooling operations, enhancing detection performance for targets of various sizes.

#### C. NECK NETWORK

The neck layer includes the Feature Pyramid Network (FPN) [27](see Fig. 5(a)), Path Aggregation Network (PAN)

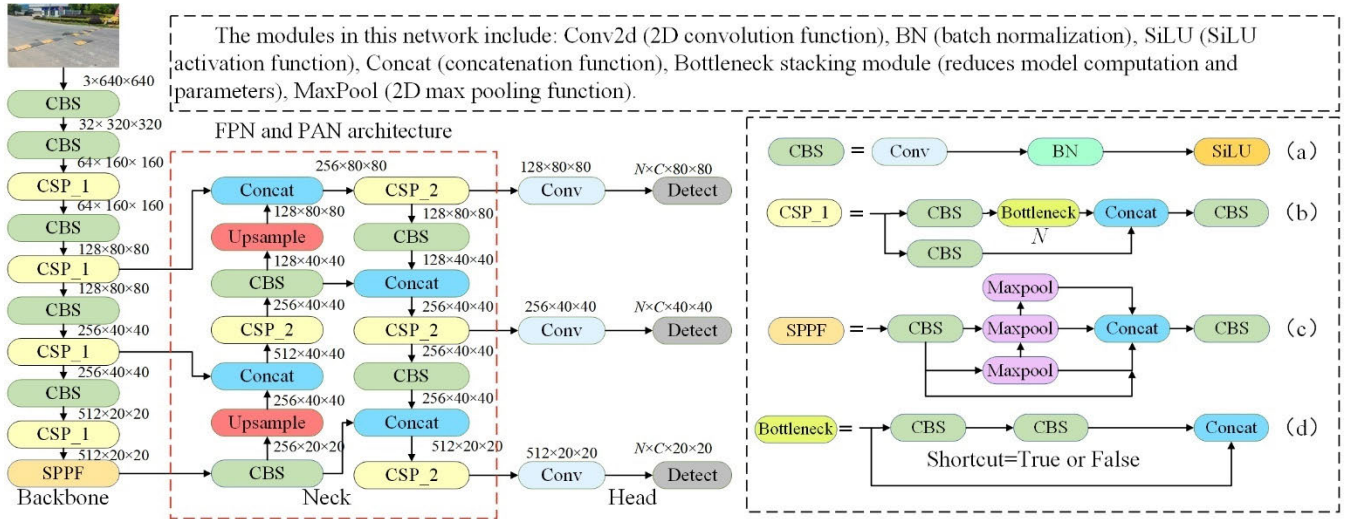


FIGURE 4. YOLOv5s network architecture diagram.

[28](see Fig. 5(b)), and an upsampling module. The FPN facilitates top-down propagation and enhances the semantic information of the image, while the PAN supports bottom-up propagation and reinforces positional information. Together, they integrate feature maps of different scales and generate multi-scale and information-rich feature representations through the upsampling module. The integrated feature maps are then fed into the prediction module, which is used for regression and classification tasks in object detection, thereby improving the model’s performance in multi-scale target detection and complex scenarios.

**D. OUTPUT HEAD LAYER**

The head layer includes three Detect detectors. When the input image size is  $640 \times 640$ , three different scale feature maps output by the Neck layer are processed by convolution

operations within the Detect module. Prediction boxes appropriate for each scale are generated on these feature maps, corresponding to  $20 \times 20$ ,  $40 \times 40$ , and  $80 \times 80$  feature maps. Each prediction layer is responsible for predicting object locations and categories on different scale feature maps produced by the feature fusion module. Finally, by applying Non-Maximum Suppression (NMS) [29], the final object detection results are filtered out.

**IV. IMPROVING THE YOLOv5s MODEL**

Based on the shortcomings observed in the YOLOv5s source code during training and incorporating improvement ideas from related YOLO algorithms [30], [31], a new improved algorithm is proposed. The network structure of the improved algorithm is illustrated in Fig. 5. The specific improvements are described as follows.

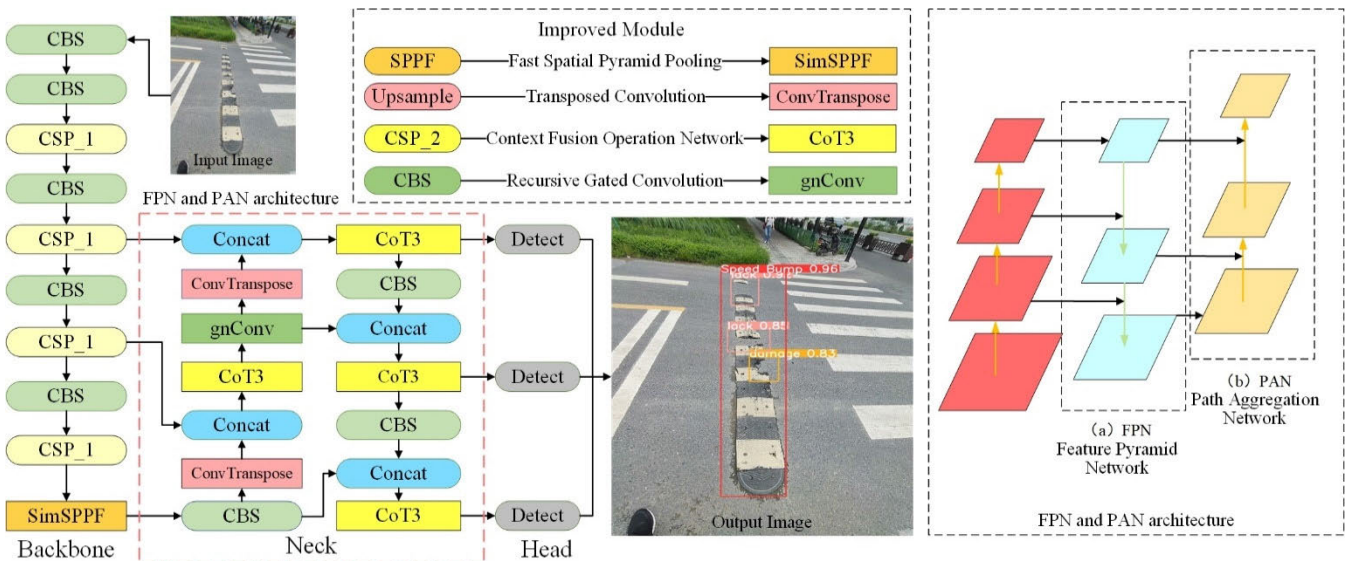


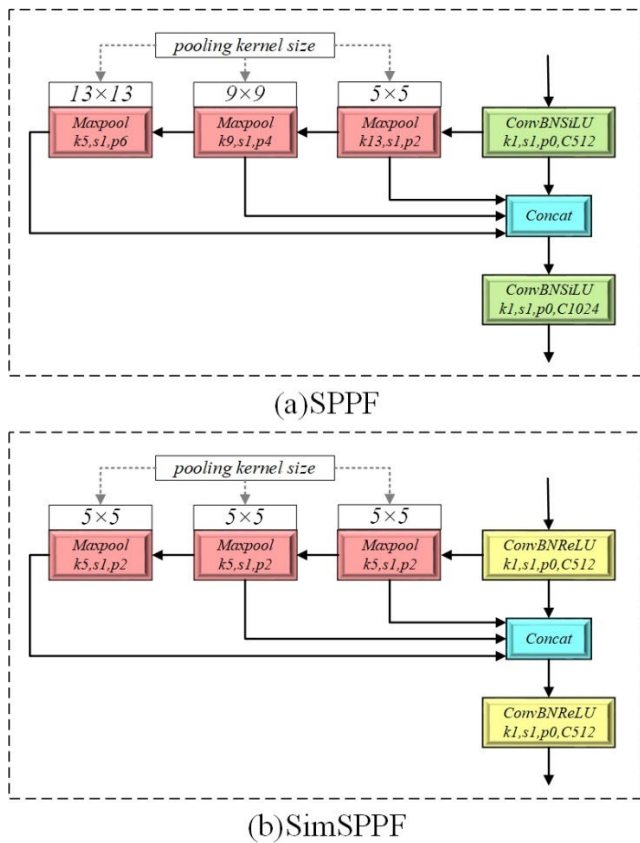
FIGURE 5. Improved YOLOv5s network architecture diagram.



**A. SPATIAL PYRAMID FAST POOLING**

To meet the need for higher response speeds and reduced model computation in practical speed bump defect detection, the SPPF module in the backbone network has been modified to the SimSPPF module. The SPPF utilizes different sizes of pooling kernels ( $5 \times 5$ ,  $9 \times 9$ , and  $13 \times 13$ ) to perform serial maximum pooling operations (MaxPool) [30] on the input feature maps, as shown in Fig. 6(a), to capture contextual information at various scales. In contrast, SimSPPF standardizes the use of a  $5 \times 5$  pooling kernel to replace the original ones and modifies the existing CBS (Conv-BN-SiLU) structure to CBR (Conv-BN-ReLU). The ReLU activation function, being a nonlinear activation function, enables higher computational efficiency while learning complex data distributions. The SimSPPF structure and formula can be seen in Fig. 6(b) and Equation (3).

$$\left. \begin{aligned} F1 &= CBR(F) \\ F2 &= Maxpool(F1) \\ F3 &= Maxpool(F2) \\ F4 &= Maxpool(F3) \\ F5 &= CBR(F1; F2; F3; F4) \end{aligned} \right\} \quad (3)$$



**FIGURE 6.** SimSPPF structure diagram.

**B. CHANGING THE SAMPLING METHOD TO TRANSPOSED CONVOLUTION**

To address the issues of low detection accuracy for small target defects and resolution loss of feature maps caused by

multiple sampling in speed bump defect detection, the sampling method has been changed to a transposed convolution module. Transposed convolution serves a similar function to the original sampling module but differs from sampling with fixed parameter values. It is essentially the inverse process of the forward propagation of a regular convolution in a convolutional neural network, where the positions of the input and output feature maps are swapped. Thus, the parameters of a transposed convolution kernel are similar to those of a regular convolution kernel. However, transposed convolution can achieve consistency in the resolution of feature maps with high and low semantic information and concatenate them into multi-channel feature maps. Then, through multi-channel convolution, features are extracted to achieve multi-channel feature fusion. This allows the model to gain more global information and better detection capabilities for small targets, effectively mitigating the resolution loss caused by multiple sampling.

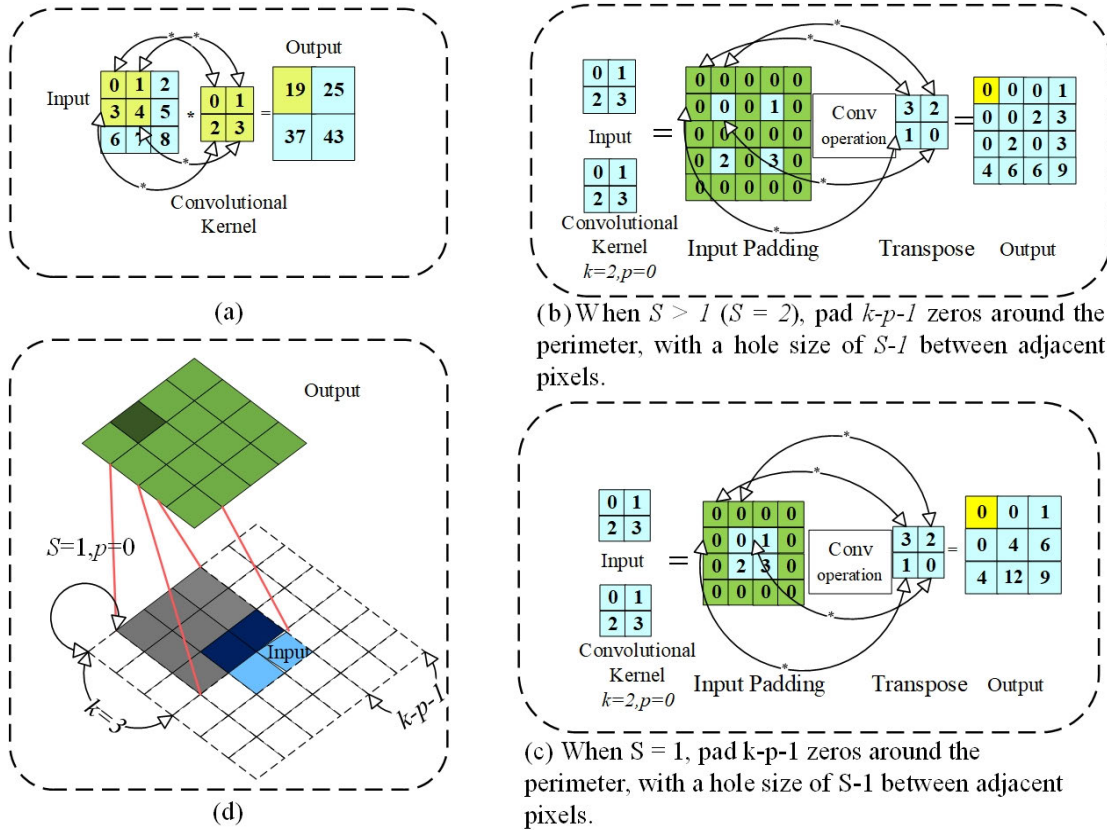
Because the default sampling method in YOLOv5s is nearest neighbor interpolation, which replaces the gray value of a source image pixel with the gray value of the nearest pixel, this method can cause color block stacking during sampling, leading to loss of feature information and reducing detection accuracy for small targets. Transposed convolution effectively solves the problem of color block stacking. Its principle is illustrated in Fig. 7, where (a) shows a diagram of a regular convolution operation, with ‘k’ representing the size of the regular convolution kernel, ‘p’ the padding of the regular convolution kernel, and the output’s first value calculated as  $0*0 + 1*1 + 3*2 + 4*3 = 19$ , with the remaining three values calculated similarly. Transposed convolution operates by transposing the original convolution kernel and performing regular convolution with the padded input. There are two ways to pad the input for transposed convolution:

When the convolutional kernel moves with a stride  $S$  greater than 1 ( $S = 2$ ) and padding  $p = 0$ , as shown in Fig. 7 (b), padding of  $k-p-1 = 2$  zeros is added around the image, and the size of the holes between adjacent pixels is  $S-1 = 1$ . Based on the regular convolution operation, the first output value is 0, and the subsequent values follow similarly. The size of the output feature map is then  $4*4$ . The formula for the output feature map is shown in Equation (4), where  $n$  represents the pixel size of the input image.

$$out = S(i - 1) + k - 2p \quad (4)$$

When the stride  $S = 1$  and padding  $p = 0$ , as shown in Fig. 7 (c), padding of  $k-p-1 = 1$  zero is added around the perimeter, and the hole size between adjacent pixels is  $S-1 = 0$ . Based on the standard convolution operation, the first output value is 0, and the subsequent values are derived similarly. The resulting size of the output feature map is  $3*3$ . The formula for the output feature map is shown in Equation (5), where  $n$  represents the pixel size of the input image.

$$out = (i - 1) + k - 2p \quad (5)$$



**FIGURE 7. Transposed Convolution Diagram. (a) Convolution Operation (b) The first padding method (c) The second padding method (d) Transposed Convolution Flowchart.**

**C. INTEGRATION OF CONTEXT OPERATION NETWORK MODULE AND C3 MODULE**

In order to further distinguish the types of defects in speed bumps under complex scene detection, extract key feature information, and better capture the global features of the problematic speed bumps, it is necessary to enhance the model’s visual representation capabilities. Therefore, this paper adopts the CotNet module, which applies the self-attention mechanism to two-dimensional feature maps, enabling it to fully capture and utilize the contextual information of the input, similar to a Transformer [31].

The CotNet module enables the network to better understand the surrounding environment in feature maps, thereby more effectively capturing target information. Additionally, integrating it with the C3 module allows for the effective fusion of feature maps from different levels to capture target information at various scales, achieving multi-scale feature fusion and reducing the loss of important feature information. The core idea of the CotNet module is to compute keys(K), queries(Q), and values(V) to generate an attention matrix with contextual information. In contrast, traditional self-attention mechanisms compute keys and queries independently for each query, which, although capable of capturing the global information of the image, neglect the extraction of local image details.

The overall workflow of the CotNet module is illustrated in Fig. 8(a), (b). It starts by obtaining the parameters [b, c, h, w] of the input feature map N, representing batch, channel number, width, and height, collectively referred to as the query(Q). The feature map N is then processed through a 1\*1 convolution to produce the value(V), and a k \* k convolution to capture local static context information(K<sup>1</sup>), termed as the key, which can be seen as static modeling of local information. Subsequently, the query(Q) and the local context information(K<sup>1</sup>) are concatenated to produce an intermediate output y, which maintains the dimensions of the input and output but merges the channel counts. The intermediate output y is further processed through two consecutive 1 × 1 convolutions (C<sub>1</sub>C<sub>2</sub>) to generate an attention matrix(A). After mean dimension reduction, the attention matrix is multiplied by the value(V) to obtain global dynamic context information (K<sup>2</sup>). Finally, local context information(K<sup>1</sup>), and the weighted global dynamic context information(K<sup>2</sup>) are combined to produce the final output(Y). The computational formula for the CotNet module is shown in Equation (6).

$$\left. \begin{aligned} A &= [K^1, Q] C_1 C_2 \\ K^2 &= A \times V \\ Y &= K^1 + K^2 \end{aligned} \right\} \quad (6)$$

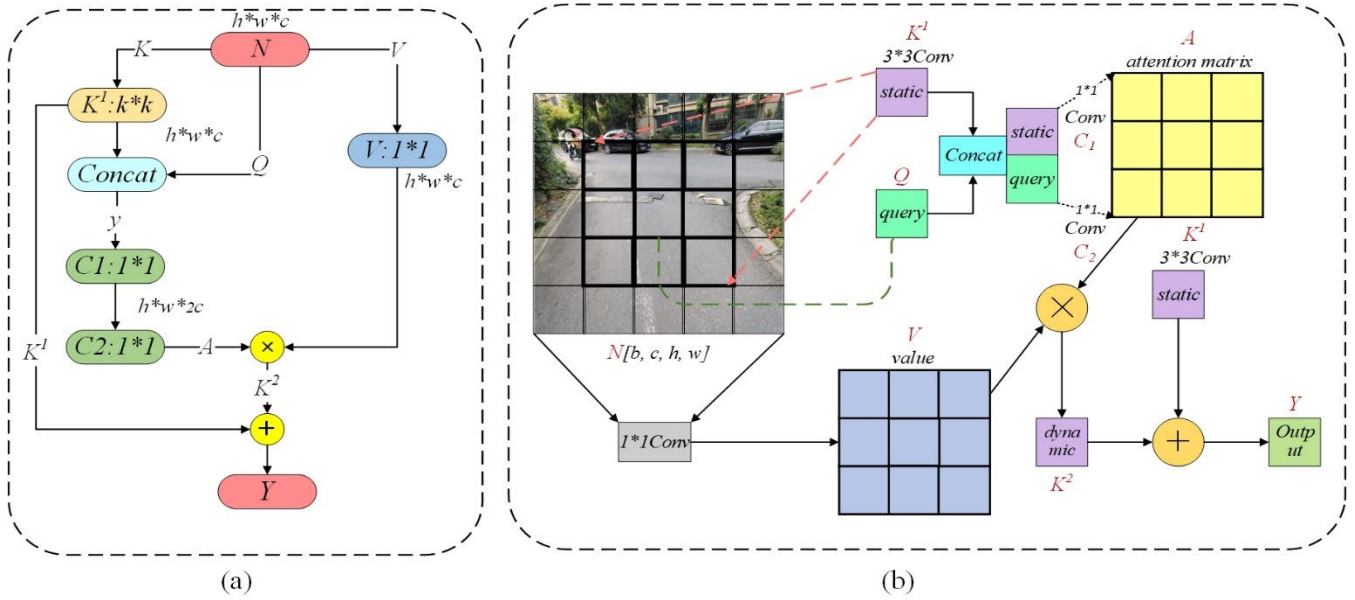


FIGURE 8. CotNet module flowchart.

**D. RECURSIVE GATE CONVOLUTION**

In actual detection scenarios, the identification of defects in speed bumps faces interference from complex factors, such as obstructions caused by fallen leaves, mud, and snow tracks. The detection is also affected by complex weather conditions and lighting environments, including rain, snow, fog, and nighttime settings. These complex situations require the model to capture high-order spatial interactions within images, complex features, and inputs of varying sizes and scales. To address this, we have incorporated recursive gate convolution, gnConv, into our approach.

Due to the use of gated convolution and recursive design structures, recursive gated convolution enables arbitrary-order spatial interaction. However, to better present its efficient high-order spatial interaction capabilities, we have considered the application sequence of different interaction methods in spatial modeling operations, as shown in Fig. 9. By modeling the spatial interactions between features (the red part) and their adjacent areas (the light grey part), where the standard convolution operation (a) usually ignores spatial interactions, which may limit the model’s performance. Dynamic convolution (b) introduces dynamic weights to enhance the spatial modeling capacity of the convolution operation but still has limitations. The self-attention operation (c) achieves second-order spatial interactions by performing matrix multiplication (MatMul) twice, but this comes with computational complexity and parameter overhead issues. The gnConv (d) adopts gated convolution and recursive design, achieving arbitrary-order spatial interactions while also improving the computational efficiency of the model.

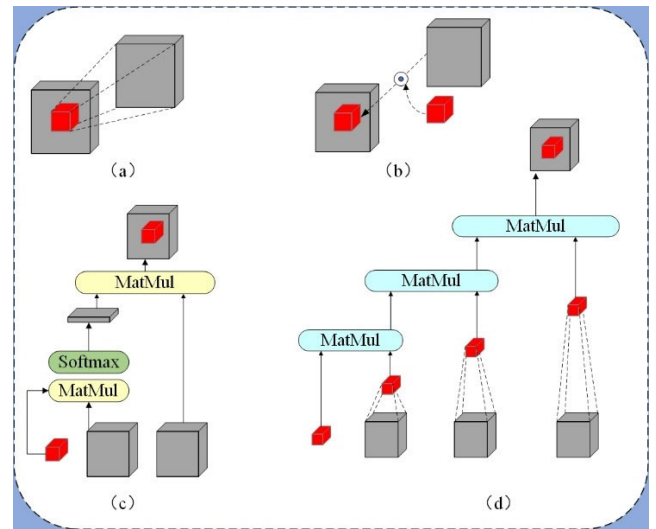


FIGURE 9. Spatial Interaction Representation Diagram. (a)Standard Convolution (b) Dynamic Convolution (c) Self-Attention Convolution (d) Recursive Gated Convolution.

The formula for gated convolution is shown in Equation (7). Let  $x$  represent the input features; then, the output can be expressed as  $y = \text{gConv}(x)$ .

$$\left. \begin{aligned} [p_0^{HW \times C}, q_0^{HW \times C}] &= \varphi_{in}(x) \in R^{HW \times 2C} \\ p_1 &= f(q_0) \odot p_0^{HW \times C} \in R^{HW \times C} \\ y &= \varphi_{out}(p_1) \in R^{HW \times C} \end{aligned} \right\} \quad (7)$$

where  $\varphi_{in}$  and  $\varphi_{out}$  are linear projection operations, namely layers that perform channel mixing;  $f$  represents a depth-wise separable convolutional gate that enables  $p_0^{HW \times C}$  to



interact once with its neighboring feature  $q_0^{HW \times C}$ , achieving first-order spatial interactions.

By incorporating a recursive structure on top of the gated convolution, higher-order spatial interactions are achieved. The formula for this is shown in Equation (8), at the bottom of the page.

After effectively implementing first-order spatial interactions, the model's capacity is further enhanced by introducing higher-order interactions  $n$  times. The specific process is as follows.

Initially, a series of projected features, such as  $p_0^{HW \times C}$  and  $q_0^{HW \times C_0}, \dots, q_{n-1}^{HW \times C_{n-1}}$ , are obtained using  $\varphi_{in}$ . These projected features are derived by splitting the input features according to channel numbers. These features are then fed into a gated convolution, and computations are performed recursively. During the recursive process, division by the scaling factor  $\alpha$  is used to stabilize training.  $f_k$  represents a set of deep convolutional layers, while  $g_k$  is used to match the channel numbers during each recursion. Finally, the output from the last recursive step,  $q_n$ , is fed into the projection layer  $\varphi_{out}$  to obtain the result of gnConv. Throughout this process, through the interactions of  $p_{k+1}$ , the order continuously accumulates, ultimately achieving  $n$  order spatial interactions.

## V. EXPERIMENTAL RESULTS AND ANALYSIS

### A. EXPERIMENTAL SETUP

The hardware platform, configuration parameters, and framework environment used for model training in this experiment are listed in Table 1.

TABLE 1. Experimental setup table.

Hardware platform	Configuration parameters	Framework environment
Operating system	Ubuntu 18.04.6 LTS	Anaconda
CPU	Intel Xeon(R) Silver 4210 CPU @ 2.20GHz × 40	Pytorch 1.13.1
GPU	NVIDIA GeForce RTX 2080 Ti/PCIe/SSE2	CUDA 11.6
Memory	125.6GiB	Python3.8

### B. EVALUATION METRICS

The model utilizes several metrics for evaluating the performance of the CRSCCG-YOLOv5s object detection model, including  $P/\%$  (Precision) [32],  $R/\%$  (Recall) [33] Mean Average Precision  $mAP/\%$  [34], the number of parameters

(Parameters), and the amount of floating-point operations (GFLOPs).

Specifically,  $P/\%$  represents the accuracy of the model, assessing the correctness of its predictions;  $R/\%$  represents the recall, also known as the detection rate, assessing the completeness of the model's target detection. The formulas for these metrics are shown in Equations (9) and (10).

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

$TP$  [35] represents the number of instances correctly identified as positive by the model,  $FP$  [36] represents the number of instances that are actually negative but were incorrectly identified as positive by the model, and  $FN$  represents the number of instances that are actually positive but were incorrectly identified as negative by the model.

The mean average precision,  $mAP/\%$ , is obtained by averaging the  $AP$  [37] across all categories, where  $AP$  is the area under the curve formed by the precision  $P/\%$  and recall  $R/\%$  curve along with the coordinate axes. The formulas for calculating  $AP$  and  $mAP/\%$  are shown in Equations (11) and (12).

$$AP = \int_0^1 P(r)dr \tag{11}$$

$$mAP = \frac{\sum_{i=1}^N AP_i}{N} \tag{12}$$

The number of parameters (Parameters) [38] represents the count of learnable parameters in the model. These learnable parameters are the weights and biases within the neural network. The floating-point operations (GFLOPs) indicate the model's computational capacity, measuring the billions of floating-point operations performed by the model, which reflects its computational efficiency.

### C. ABLATION STUDIES AND COMPARATIVE EXPERIMENTS

To validate the performance of the CRSCCG-YOLOv5s model in detecting defects on speed bumps, a series of ablation experiments were designed. These experiments incrementally combined and compared various improvement modules, utilizing selected evaluation metrics to assess the enhancements of the model. Specifically, Group A added the SimSPPF module to the base YOLOv5s; Group B further integrated the CotNet module with the C3 module on top of the SimSPPF addition; Group C added the gnConv module

$$\left. \begin{aligned} & \left[ p_0^{HW \times C_0}, q_0^{HW \times C_0}, \dots, q_{n-1}^{HW \times C_{n-1}} \right] = \varphi_{in}(x) \in R^{HW \times (C_0 + \sum_{0 \leq k \leq n-1} C_k)} \\ & p_{k+1} = f_k(q_k) \odot \frac{g_k(p_k)}{\alpha}, \quad k = 0, 1, \dots, n-1 \\ & g_k = \begin{cases} Identity & k=0 \\ Linear(C_{k-1}, C_k) & 1 \leq k \leq n-1 \end{cases} \\ & C_k = \frac{C}{2^{n-k-1}}, \quad 0 \leq k \leq n-1 \end{aligned} \right\} \tag{8}$$

TABLE 2. Ablation study table.

Model method	SimSPPF	CotNet	gnConv	ConvTranspose	P/%	R/%	mAP/%
YOLOv5s	×	×	×	×	95.5	91.0	93.5
A	√	×	×	×	95.0	91.7	94.8
B	√	√	×	×	96.8	90.8	95.8
C	√	√	√	×	97.2	92.0	95.3
CRSCCG-YOLOv5s	√	√	√	√	97.7	91.9	96.4

TABLE 3. Comparative experiment table.

Model method	P/%	R/%	mAP/%	GFLOPs	Parameters(M)
YOLOv3	97.3	92.1	94.3	154.6	61.5
YOLOv5l	96.2	92.6	93.5	107.7	46.1
YOLOv5m	97.5	93.5	95.5	47.9	20.9
YOLOv5s	95.5	91.0	93.5	15.8	7.1
YOLOv7	96.3	92.2	95.5	105.2	37.2
YOLOv8s	96.0	94.1	95.4	47.2	11.1
CRSCCG-YOLOv5s	97.7	91.9	96.4	26.7	8.8

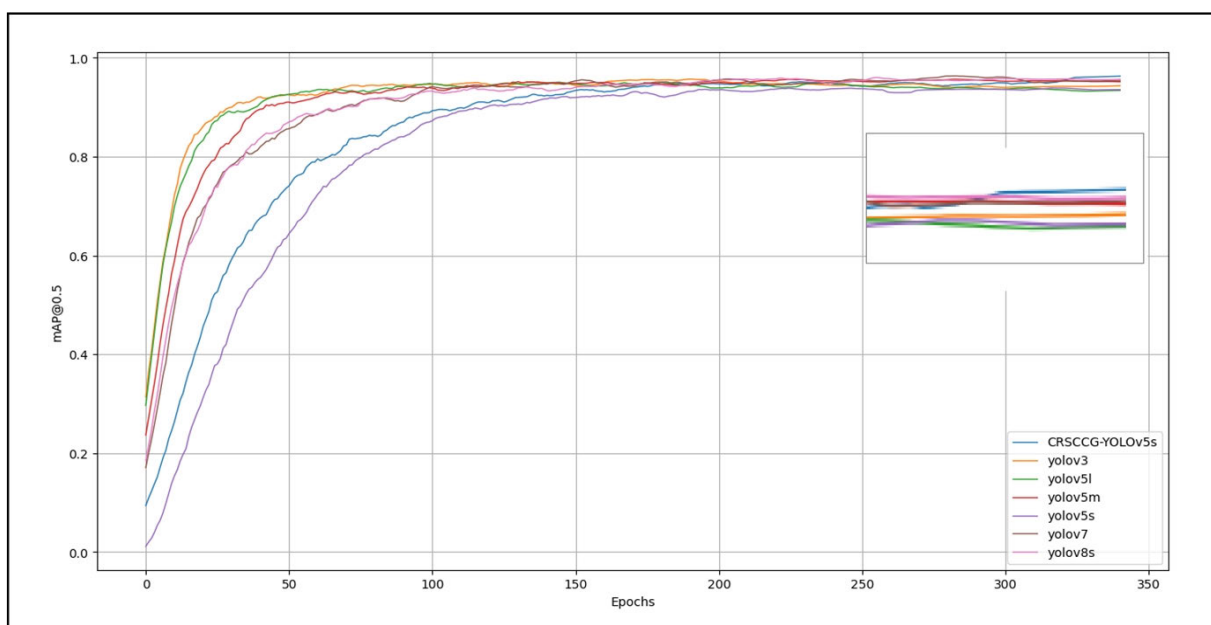


FIGURE 10. mAP% curve comparison chart.

after integrating SimSPPF and CotNet modules; CRSCCG-YOLOv5s represents the integration of all improvement modules, including SimSPPF, CotNet, gnConv, and ConvTranspose.

The results from the ablation experiments shown in Table 2 reveal that initially, Group A experienced a 0.5% decrease in accuracy but a 0.7% increase in recall and a 1.3% increase in mean average precision after adding the SimSPPF module. This indicates that the inclusion of this module enhances the model’s ability to capture multi-scale contextual information. In Group B, which built upon Group A by integrating the

CotNet module, there was a 1.8% increase in accuracy and a 1.0% increase in mean average precision, though recall decreased by 0.9%. This demonstrates that the fusion of this module improves the model’s accuracy in detecting small target defects. Group C, which added the gnConv module on top of Group B, showed a 0.4% increase in accuracy, a 1.2% increase in recall, but a 0.5% decrease in mean average precision, indicating that Group C’s improvements further enhanced the model’s feature extraction capabilities in complex environments, improving both accuracy and recall. Finally, merging all four improvements compared to Group

C resulted in a 0.5% increase in accuracy, a 0.1% decrease in recall, and a 1.1% increase in mean average precision. Overall, compared to the original YOLOv5s code, the final improvements led to a 1.2% increase in accuracy, a 0.9% increase in recall, and a 2.9% increase in mean average precision. In conclusion, the CRSCCG-YOLOv5 algorithm is highly effective in enhancing model performance in scenarios involving small targets, complex scenes, and multi-scale image inputs.

To further validate the effectiveness of the CRSCCG-YOLOv5s algorithm, this study conducted comparisons using the same dataset, on the same hardware, and within the same experimental environment, framework, and training strategy. The evaluations were based on metrics such as accuracy (P%), recall (R%), mean average precision (mAP%), the number of parameters, and floating-point operations, comparing them with YOLOv3, YOLOv5l, YOLOv5m, YOLOv5s, YOLOv7, and YOLOv8s. The results from Table 3 show that the improved YOLOv5s, compared to YOLOv3, experienced a slight decrease in recall by 0.2 percentage points but significantly reduced the number of parameters and floating-point operations, with an increase in mean average precision by 2.1 percentage points. Compared to YOLOv5l, although there was a 0.7 percentage point decrease in recall, there was a substantial reduction in parameters and floating-point operations, with an increase in mean

average precision by 2.9 percentage points. Compared to YOLOv5m, the parameter count was reduced by about half, and the mean average precision increased by 0.9 percentage points. Against YOLOv7, there was an approximate increase of 1 percentage point in both accuracy and mean average precision, with a more than fourfold reduction in parameter count. Compared to YOLOv8s, although there was a 2.2 percentage point decrease in recall, there were improvements of 1.7 percentage points in accuracy and 1.0 percentage points in mean average precision, along with advantages in the number of parameters and floating-point operations. Compared to the original YOLOv5s model, although there was a slight increase in parameters and floating-point operations, accuracy improved by 2.2 percentage points to 97.7%, recall by 0.9 percentage points to 91.9%, and mean average precision by 2.9 percentage points to 96.4%.

In summary, compared to YOLOv3 and YOLOv5l, the CRSCCG-YOLOv5s model boasts smaller size and reduced computational demands while achieving higher mean average precision. When compared to YOLOv5m and YOLOv7, although the accuracy, recall, and mean average precision are similar, CRSCCG-YOLOv5s has a clear advantage in terms of model size and computational requirements. When compared to YOLOv8s, although the improvements in accuracy, recall, and mean average precision are not particularly significant, CRSCCG-YOLOv5s still maintain an advantage

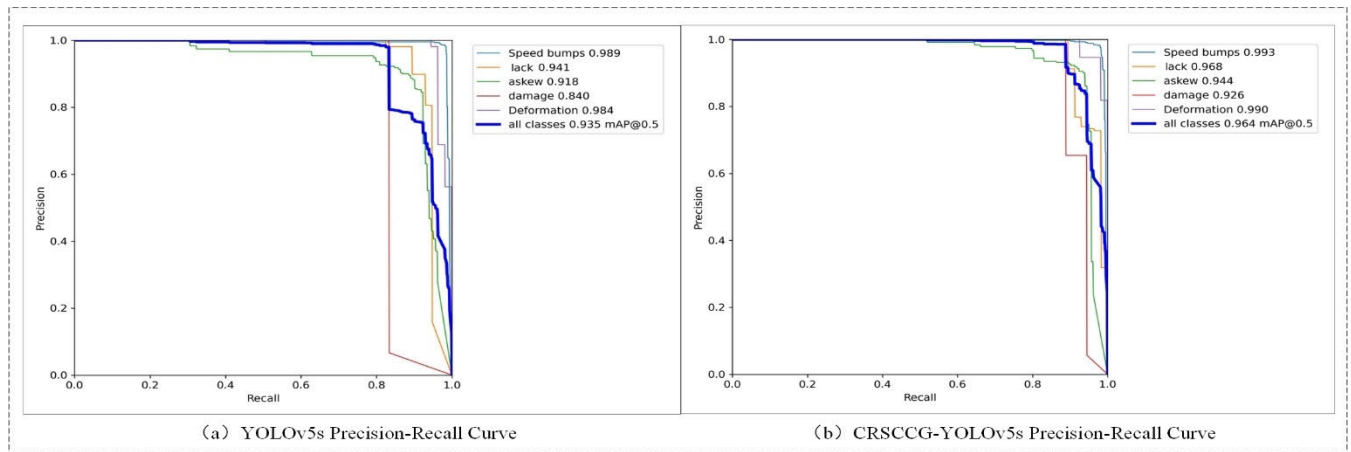


FIGURE 11. Precision-Recall curve comparison chart.

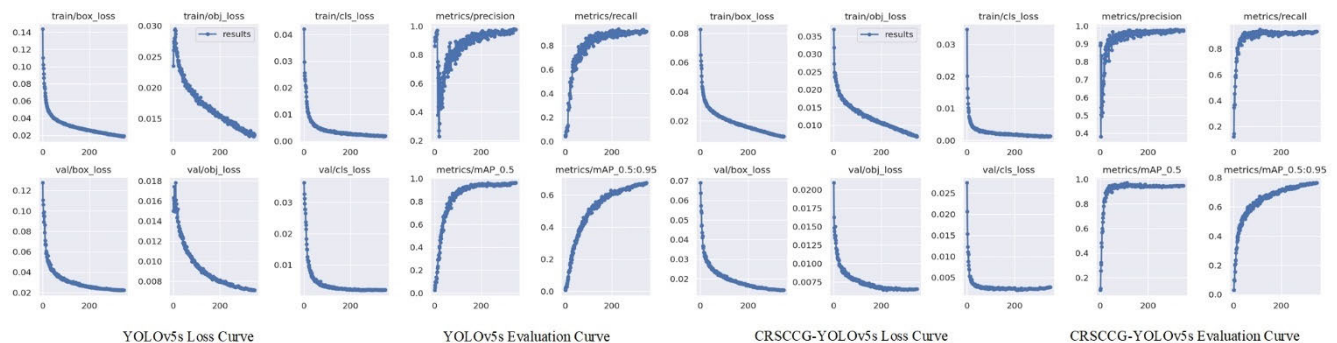
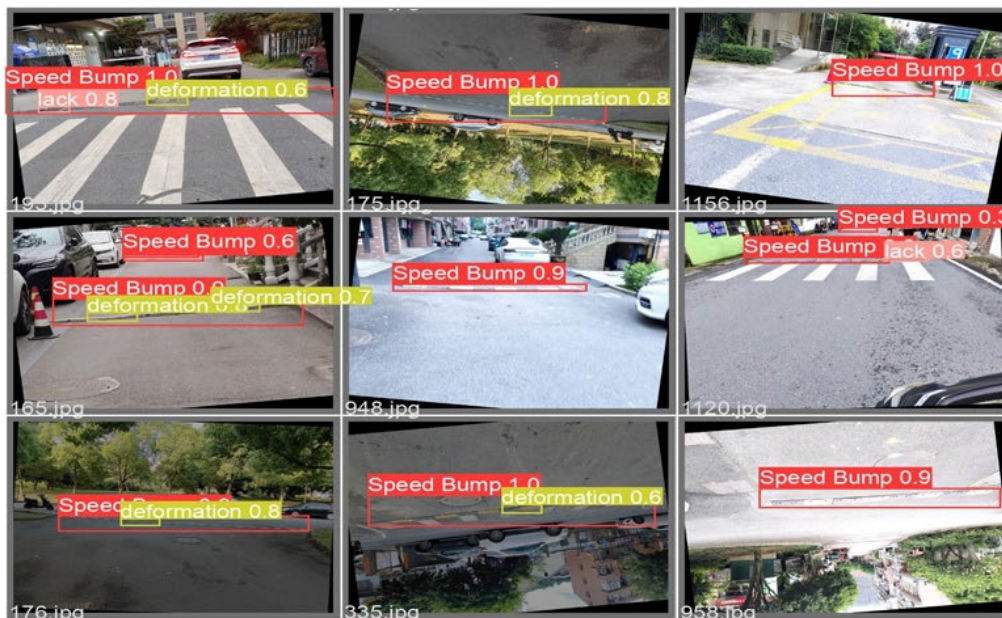


FIGURE 12. Training curve comparison chart.





(a) YOLOv5s Validation Image



(b) CRSCCG-YOLOv5s Validation Image

FIGURE 13. Validation effect comparison chart.

in model parameters and computational load. Additionally, as shown in Figure 10, the mAP% curve comparison indicates that around 350 epochs, the blue curve representing CRSCCG-YOLOv5s is higher than the other curves. Therefore, under the same experimental conditions, using the same equipment and dataset, the CRSCCG-YOLOv5s model stands out among other YOLO series object detection models with its smaller parameter count, compact size, and higher accuracy, recall, and mean average precision.

**D. TRAINING CURVE COMPARATIVE ANALYSIS**

To better demonstrate the effectiveness of the CRSCCG-YOLOv5s algorithm compared to YOLOv5s, an analysis of three types of loss curves and three evaluation curves is presented in Fig. 12. With both models trained for 350 rounds, the CRSCCG-YOLOv5s shows notably faster convergence and less fluctuation in both the training and validation obj loss curves. The precision, recall, and mean average precision curves (mAP0.5, mAP0.5:0.95) all converge around



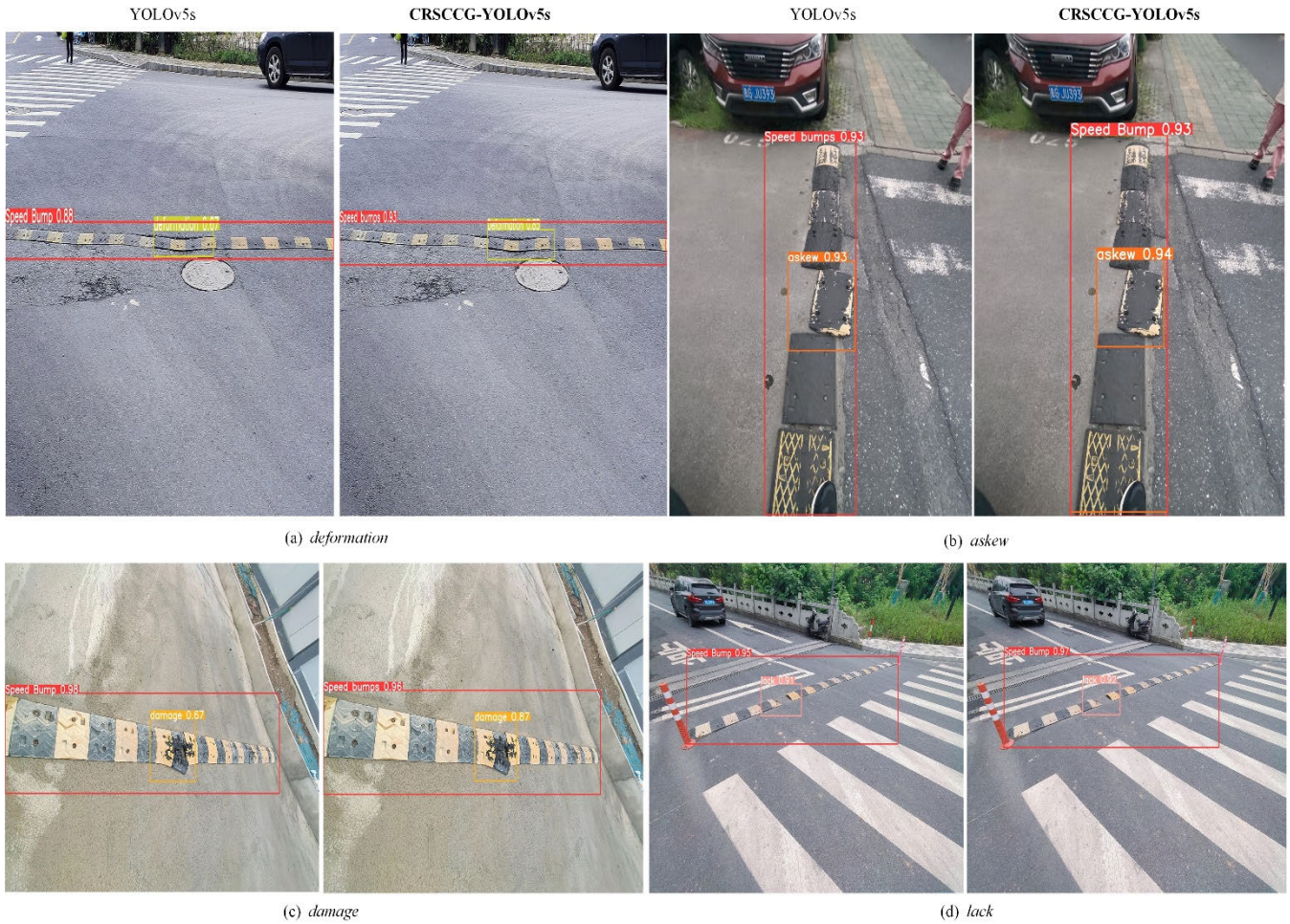


FIGURE 14. Detection effect comparison chart.

200 rounds, which is significantly quicker than YOLOv5s, and with notably less volatility. This indicates that the addition of improved modules can lead to faster convergence of model curves and reduce fluctuation.

Additionally, as shown in the P-R curve comparisons in Fig. 11, the CRSCCG-YOLOv5 model shows an 8.6 percentage point improvement over the YOLOv5s model in detecting the damaged category, and improvements of 2.7 and 2.6 percentage points in detecting the missing and skewed categories, respectively. There are also slight improvements in detecting other defect categories. This demonstrates that the addition of improved modules not only enhances detection for specific defect categories such as damage but also improves the overall detection performance for various defect categories.

**E. VALIDATION IMAGE COMPARATIVE ANALYSIS**

To better compare the effects before and after the model improvement, 16 images were randomly selected from the validation set of the Speed Bump Defect dataset (SBD)

for validation tests using both CRSCCG-YOLOv5s and YOLOv5s. The test results are shown in Figure 11.

From Figure 13, images 193, 165, 176, 175, and 335 demonstrate that CRSCCG-YOLOv5s has made a progress of 0.1 to 0.2 percentage points in the accuracy of deformation defect recognition compared to YOLOv5s. Additionally, images 193 and 1120 show a 0.1 percentage point improvement in the accuracy of missing defect recognition with CRSCCG-YOLOv5s compared to YOLOv5s. This indicates that the improvements in CRSCCG-YOLOv5s, specifically the enhancements for capturing small target defects and multi-scale defect features through the integration of transposed convolutions and the contextual operation network CotNet with C3, have indeed been effective. Furthermore, there has also been a 0.1 percentage point improvement in speed bump recognition, as shown in images 958 and 1136.

**F. DETECTION EFFECT COMPARISON ANALYSIS**

To better highlight the improvements before and after modifications, this study visualizes the detection results of CRSCCG-YOLOv5s compared to YOLOv5s. Defective

speed bump images featuring damage, skewing, deformation, and missing parts were selected to validate the enhancements of the model. These results are compared with the original YOLOv5s, with specific effects shown in Fig. 14.

Compared to the original YOLOv5s, the CRSCCG-YOLOv5s algorithm shows improved detection performance for speed bumps. Specifically, the confidence in deformation detection increased from 0.67 to 0.85, a rise of 0.18 points; in damage detection, confidence increased from 0.67 to 0.87, a rise of 0.20 points; in missing detection, confidence increased from 0.91 to 0.92, a rise of 0.01 points; and in skewed detection, confidence increased from 0.93 to 0.94, a rise of 0.01 points. According to these results, it is evident that the improved module significantly enhances detection performance in complex scenarios, especially for small target defects such as damage and deformation. Even for simpler defects characterized by skewing and missing features, there is a slight improvement in detection accuracy. Overall, the enhancements not only improve the identification of speed bumps but also the detection of various defect types.

## VI. CONCLUSION

This study supplements existing research on speed bump defect detection by proposing the improved CRSCCG-YOLOv5s model, which builds on the original YOLOv5s. The model further detects and classifies defects in speed bumps, such as missing, damaged, deformed, and skewed features. Improvements were made in pooling structures, sampling methods, convolutional architectures, data augmentation, and sample balancing. Specifically, enhancements include the addition of Spatial Pyramid Pooling Fast (SimSPPF), transposed convolutions, Context Operation Networks, Recursive Gated Convolutions, and CR data augmentation for sample balancing. These improvements not only enhance detection speed and accuracy but also strengthen the capability to extract features from small target images, addressing issues like uneven distribution of image sample types.

Experimental results show that CRSCCG-YOLOv5s achieves an accuracy of 97.7%, a recall rate of 91.9%, and a mean average precision of 96.4% on the custom Speed Bump Defects (SBD) dataset, with a model size of 8.8M parameters. Therefore, compared to other YOLO series models (YOLOv3, YOLOv5l, m, x, YOLOv7, and the currently stable YOLOv8), CRSCCG-YOLOv5s excels in precision, confidence, and parameter efficiency. The model performs well in scenarios requiring quick response and provides high-quality detection results, offering an efficient solution for speed bumps defect detection.

Nevertheless, CRSCCG-YOLOv5s still has limitations in accurately classifying the specific defect degrees of speed bumps, such as deformation and damage, and needs improvement in model lightweight optimization. Future work will focus on refining defect degree classification, optimizing model size, enhancing detection speed, and exploring the potential application in other traffic facility detections.

Additionally, to address the challenges where critical features may be obscured due to insufficient lighting in dark or low-light environments, future research will consider integrating infrared sensing technology or image enhancement methods to improve detection performance under these conditions.

## REFERENCES

- [1] B. Al-Shargabi, M. Hassan, and T. Al-Rousan, "A novel approach for the detection of road speed bumps using accelerometer sensor," *TEM J.*, vol. 9, no. 2, pp. 469–476, May 2020.
- [2] H.-S. Yun, T.-H. Kim, and T.-H. Park, "Speed-bump detection for autonomous vehicles by LiDAR and camera," *J. Electr. Eng. Technol.*, vol. 14, no. 5, pp. 2155–2162, Sep. 2019.
- [3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Kauai, HI, USA, Dec. 2001, p. 1.
- [4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, San Diego, CA, USA, Jun. 2005, pp. 886–893.
- [5] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Graz, Austria, May 2006, pp. 404–417.
- [6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580–587.
- [8] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
- [9] 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.
- [10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [11] M. Alnaasan and S. Kim, "Handwritten multi-scale Chinese character detector with blended region attention features and light-weighted learning," *Sensors*, vol. 23, no. 4, p. 2305, Feb. 2023.
- [12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Amsterdam, The Netherlands, Oct. 2016, pp. 21–37.
- [13] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2999–3007.
- [14] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10778–10787.
- [15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
- [16] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 1998.
- [17] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2017, pp. 7263–7271.
- [18] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
- [19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
- [20] G. Jocher. (2020). *YOLOv5: An Improved Version of YOLOv4*. Accessed: Jul. 27, 2024. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [21] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 7464–7475.
- [22] R. Varghese and M. Sambath, "YOLOv8: A novel object detection algorithm with enhanced performance and robustness," in *Proc. Int. Conf. Adv. Data Eng. Intell. Comput. Syst. (ADICS)*, Chennai, India, Apr. 2024, pp. 1–6.



- [23] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed., Cambridge, MA, USA: MIT Press, 1995, p. 3361.
- [24] C.-Y. Wang, H.-Y. M. 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 390–391.
- [25] D. K. Dewangan and S. P. Sahu, "Deep learning-based speed bump detection model for intelligent vehicle system using raspberry pi," *IEEE Sensors J.*, vol. 21, no. 3, pp. 3570–3578, Feb. 2021.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
- [27] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 936–944.
- [28] 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.*, Salt Lake City, UT, USA, Jun. 2018, pp. 8759–8768.
- [29] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Hong Kong, 2006, pp. 850–855.
- [30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
- [31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.
- [32] S. Xu, Y. Ji, G. Wang, L. Jin, and H. Wang, "GFSP-YOLO: A light YOLO model based on group fast spatial pyramid pooling," in *Proc. IEEE 11th Int. Conf. Inf., Commun. Netw. (ICICN)*, Xi'an, China, Aug. 2023, pp. 733–738.
- [33] K. Xiong, Q. Li, Y. Meng, and Q. Li, "A study on weed detection based on improved YOLO v5," in *Proc. 4th Int. Conf. Inf. Sci. Educ. (ICISE-IE)*, Zhanjiang, China, Dec. 2023, pp. 1–4.
- [34] A. Anish, R. Sharan, A. H. Malini, and T. Archana, "Enhancing surveillance systems with YOLO algorithm for real-time object detection and tracking," in *Proc. 2nd Int. Conf. Autom., Comput. Renew. Syst. (ICACRS)*, Pudukkottai, India, Dec. 2023, pp. 1254–1257.
- [35] Q. Wang, Z. Liao, and M. Xu, "Wire insulator fault and foreign body detection algorithm based on YOLO v5 and YOLO v7," in *Proc. IEEE Int. Conf. Electr., Autom. Comput. Eng. (ICEACE)*, Changchun, China, Dec. 2023, pp. 1412–1417.
- [36] J. Da Silva, T. Flores, S. Júnior, and I. Silva, "TinyML-based pothole detection: A comparative analysis of YOLO and FOMO model performance," in *Proc. IEEE Latin Amer. Conf. Comput. Intell. (LA-CCI)*, Recife-Pe, Brazil, Oct. 2023, pp. 1–6.
- [37] F. A. Kurniadi, C. Setianingsih, and R. E. Syaputra, "Innovation in livestock surveillance: Applying the YOLO algorithm to UAV imagery and videography," in *Proc. IEEE 9th Int. Conf. Smart Instrum., Meas. Appl. (ICSIMA)*, Kuala Lumpur, Malaysia, Oct. 2023, pp. 246–251.
- [38] Y. Lu, L. Zhang, and W. Xie, "YOLO-compact: An efficient YOLO network for single category real-time object detection," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Hefei, China, Aug. 2020, pp. 1931–1936.



**XINJIAN XIANG** (Member, IEEE) received the B.E. degree in automation, in 1985, and the M.E. degree in automation, in 1988. He is currently a Professor. He has long been engaged in teaching and research in intelligent manufacturing and intelligent control, having led and participated in one National 863 Project, four National Spark Plan projects, five National Torch Plan projects, four major provincial science and technology projects, four general provincial science and technology projects, four Zhejiang Natural Science Foundation projects, and two Zhejiang Province Educational Reform projects. He has published over 80 research and teaching articles in academic journals, such as *Advanced Materials Research*, *IFIP Advances in Information and Communication Technology*, *Journal of Instruments and Instruments*, *Journal of Power Systems and Automation*, and *Journal of Higher Engineering Education*, 42 of which are included in SCI/EI. He has authored two textbooks and received four provincial or higher awards. His main research interests include intelligent manufacturing, artificial intelligence, robotics, the Internet of Things, and agricultural informatization.



**XIZHAO CHEN** received the B.E. degree in electrical engineering and automation from the College of Electronic and Information Engineering, Taizhou University, Taizhou, China, in 2021. He is currently pursuing the M.E. degree in industrial robotics and intelligent control with the School of Automation and Electrical Engineering, Zhejiang University of Science and Technology, Hangzhou, China. His research interests include deep learning and image processing.

**XIAOHUI TANG**, photograph and biography not available at the time of publication.

**QIUXIA LUO**, photograph and biography not available at the time of publication.

**YI DING**, photograph and biography not available at the time of publication.

• • •