

Lightweight Yet Precized: A Redesigned YOLO for High-Accuracy Road Crack Detection on Vehicle-Mounted Devices

Chongyu Luo

Power China Chengdu Engineering Corporation Limited

Ji Li

Hunan Agricultural University

Xuan Song

KiannaSongxuan@outlook.com

Guizhou Polytechnic of Construction

Yabing Liu

Guizhou Polytechnic of Construction

Wentao Yu

Central South University

Article

Keywords: Road surface crack detection, Vehicle-mounted images, Lightweight design, Precise design, LP_YOLOv8

Posted Date: October 17th, 2025

DOI: <https://doi.org/10.21203/rs.3.rs-7735353/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Lightweight Yet Precized: A Redesigned YOLO for High-Accuracy Road Crack Detection on Vehicle-Mounted Devices

Chongyu Luo¹, Ji Li², Xuan Song³, Yabing Liu³, and Wentao Yu⁴

¹Power China Chengdu Engineering Corporation Limited, Chengdu 610072, China;

²College of Bioscience and Biotechnology, Hunan Agricultural University, Changsha 410128, China

³School of Information Management, Guizhou Polytechnic of Construction, Guiyang 551400, China

⁴School of Automation, Central South University, Changsha 410083, China

Corresponding author: Xuan Song (e-mail: KiannaSongxuan@outlook.com).

This work was supported by the Projects of International Cooperation and Exchanges NSFC (with grant number 61860206014), Guizhou Vocational and Technical College of Industry 2023 School-Level Research Project (with grant number 2023ZK06)

ABSTRACT Road crack detection plays a critical role in ensuring traffic safety and enabling timely maintenance. However, real-time detection algorithms deployed on vehicle-mounted platforms often struggle with several challenges. These include poor sensitivity to small cracks, incomplete feature extraction, and difficulty balancing detection accuracy with model efficiency. This study presents LP-YOLOv8, an enhanced object detection algorithm based on an improved YOLOv8n architecture. The proposed method is designed to offer a lightweight yet precise solution for road crack detection. To reduce model complexity while maintaining high accuracy, we introduce the C2f-faster module, which optimizes the backbone network by minimizing parameter redundancy and suppressing background noise. We further propose a Lightweight Shared Detail-Enhanced Convolution Detection Head (LSDECD) to enhance multi-scale feature perception while minimizing parameter count. To improve the detection of small cracks, we redesign the neck using a Focal Diffusion Pyramid Network (FDPN). FDPN strengthens cross-scale feature fusion by regulating deep-to-shallow feature interactions through learnable attention gating. Moreover, we propose an improved Inner-WIoU loss function that focuses on small objects and samples with general quality labels. The LP-YOLOv8 method was evaluated on the RDD2022 vehicle-mounted image dataset through extensive comparative and ablation studies. Experimental results demonstrate a 3.41% improvement in mAP@0.5 over YOLOv8n, while reducing the number of parameters by 55.6% and computational costs by 36.7%. These findings confirm that LP-YOLOv8 provides an efficient and accurate solution for real-time road crack detection in vehicle-mounted imaging systems.

INDEX TERMS Road surface crack detection; Vehicle-mounted images; Lightweight design; Precise design; LP_YOLOv8.

I. INTRODUCTION

Road networks, serving as the lifeline of modern transportation systems, sustain over 6.3 million kilometers of paved surfaces in China alone [1], facilitating billions of daily vehicle movements. However, these infrastructures are subject to continuous degradation caused by harsh service conditions. These include annual thermal cycling beyond $\Delta 50^{\circ}\text{C}$, heavy traffic loads exceeding 100 million ESALs, and up to 50 freeze-thaw cycles each year [2]. Empirical studies have shown that unrepaired 3 mm-wide cracks can reduce pavement structural capacity by 30% within just 18 months [3]. Even micron-level fissures can

increase water infiltration by 400%, potentially leading to severe potholes. Although such cracks may occupy less than 0.05% of the image area, they are responsible for \$12.4 billion in annual maintenance costs and 17% of weather-related traffic fatalities in China [4, 5]. If not detected and addressed promptly, cracks can significantly weaken the overall structural stability of roads and pose severe safety risks for vehicles. Traditional inspection methods like ASTM D6433 visual surveys [6] and semi-automated edge detection [7] often fall short. They suffer from low inter-rater reliability ($\kappa < 0.6$), a 40% drop in accuracy under low illumination, and high labor cost (2.7 hours per kilometer)

[8]. These shortcomings make them unsuitable for intelligent transportation systems that require real-time (<100 ms), high-precision (mAP@0.5 > 65%) crack detection. With the rapid development of modern inspection equipment, automated detection methods, which utilize road inspection vehicles or drones to collect road surface images or videos and analyze the data using object detection algorithms [9, 10], are becoming the mainstream approach for road damage detection.

Deep learning techniques, particularly convolutional neural network (CNN)-based object detection methods, have revolutionized crack detection. Compared to traditional approaches, deep learning models can automatically learn multi-level abstract features from large datasets, thereby achieving higher robustness and accuracy in complex environments. Existing deep learning-based object detection methods can be broadly classified into two-stage and single-stage methods.

Two-stage methods (e.g., Faster R-CNN [11], Mask R-CNN [12]) typically first generate candidate regions and then classify and localize these regions. While these methods exhibit superior accuracy, their complex computational processes—such as region proposal and ROI pooling—result in slower inference speeds, making them less suitable for real-time detection in vehicular environments. For example, Kang et al. [13] and Kim et al. [14] enhanced crack detection accuracy using improved Faster R-CNN and Mask R-CNN models, respectively. However, the complexity and large number of parameters in these models impose significant limitations on their deployment in vehicle-mounted applications. Wang et al. [15] attempted to address this issue by introducing multi-scale convolutional kernels and optimizing anchor design, which improved small crack detection accuracy but still faced challenges related to real-time performance.

In contrast, one-stage object detection methods (such as the YOLO series [16-18]) classify and regress directly on the image, eliminating the need for candidate region generation and significantly improving detection speed. In recent years, many studies have focused on applying YOLO models to crack detection tasks. For example, Zhang et al. [19] improved detection speed by replacing the feature extraction backbone of YOLOv4 with a lightweight network. Yu et al. [20] introduced a bottleneck transformer into YOLOv5 to reduce redundant parameters while maintaining high detection accuracy. Jiang et al. [21] optimized the YOLOv5 model using a self-attention mechanism, achieving excellent performance on crack images captured by drones. Recent advances in YOLOv5 optimization for vehicle-mounted scenarios by Hu et al. [22] have demonstrated effective neck structure improvements through their Slim-Neck design and C2f modules, achieving a balance between computational efficiency and detection accuracy.

Further advancements have been made with YOLOv8 [23] and YOLOv9 [24], which have significantly reduced model weight and enhanced real-time performance on embedded devices.

Although recent studies [25-27] have improved accuracy using lightweight backbones, attention mechanisms, and loss function enhancements, key challenges persist. In vehicle-mounted scenes, cracks often cover less than 0.05% of an image and must be detected under motion blur and changing lighting. Three major limitations persist in such environments:

(1) Feature Learning with Scale Fragmentation: Existing multi-scale fusion strategies (e.g., feature pyramid networks (FPN) [28], path aggregation network (PANet) [29]) prioritize semantic consistency over geometric fidelity, leading to blurred micron-scale crack edges during cross-resolution aggregation.

(2) Redundant Detection Heads: Traditional detection heads apply independent convolutional stacks at each scale, accumulating 42–58% of total FLOPs without leveraging cross-scale correlations.

(3) Context Myopia: Standard IoU variants optimize bounding box overlap but fail to account for the unique morphological patterns of cracks, such as branching topology and directional continuity.

To address the limitations, this study proposes LP-YOLOv8, a holistic architectural redesign for real-time crack detection, incorporating four interlocking innovations: (1) Lightweight C2f-Faster Module: This module replaces the traditional bottleneck structure with depth wise separable convolutions and channel shuffling operations, achieving a 41% reduction in parameters while preserving multi-scale feature representation capability; (2) Lightweight Shared Detail-Enhanced Convolutional Detection Head (LSDECD): By sharing weights across scales and incorporating deformable convolutions to enhance detail perception, LSDECD significantly improves detection performance for cracks smaller than 32 pixels while simultaneously reducing the computational cost of the detection head; (3) Focal Diffusion Pyramid Network (FDPN): FDPN leverages learnable attention gating to regulate the up sampling of deep features using high-resolution shallow features, addressing information loss issues inherent in traditional pyramid networks during cross-scale fusion; and (4) Improved Inner-WIoU Loss Function: By introducing an adaptive focusing coefficient, this modification enhances bounding box regression by prioritizing crack interior regions, resulting in faster convergence and improved localization accuracy compared to WIoU.

Extensive experiments on a re-organized dataset demonstrate the superior performance of the proposed LP-YOLOv8 model in vehicle-mounted crack detection, providing a robust solution for real-time and precise crack detection in vehicle-mounted images.

II. RELATED WORK

Accurate and real-time road surface crack detection is essential for the maintenance of transportation infrastructure. This section provides a comprehensive review of existing research in two key areas: (1) crack detection methods, including both traditional and deep learning-based approaches, and (2) improvements to the YOLO object detection framework and its applications to crack detection. Based on this review, we identify current limitations and motivate the design of the proposed LP-YOLOv8 model.

A. Traditional and Deep Learning-Based Crack Detection Methods

Early crack detection approaches predominantly relied on traditional image processing techniques such as Sobel/Canny edge detection [30], Otsu thresholding [31], and morphological operations [32]. While computationally efficient, these methods exhibit strong sensitivity to environmental noise (e.g., shadow, surface texture, oil stains) and lack robustness across pavement types and lighting conditions. Moreover, they fail to generalize across datasets due to rigid hand-crafted feature definitions and heuristic threshold dependencies.

With the emergence of deep learning, data-driven models—especially convolutional neural networks (CNNs)—have largely replaced conventional techniques. Two dominant paradigms have emerged:

1) **Semantic Segmentation-Based Detection:** These approaches predict pixel-level crack masks, often leveraging encoder-decoder architectures. For instance, Zhu et al. [33] introduced CrackNet, a hierarchical encoder-decoder network incorporating residual connections. Sarhadi et al. [34] further embedded attention gates into UNet to improve detection of discontinuous cracks. Despite achieving high granularity, these models often suffer from high latency and low throughput, making real-time deployment infeasible, especially in embedded systems.

2) **Object Detection-Based Crack Localization:** Reformulating cracks as sparse object instances has proven more computationally efficient. Two-stage detectors like Faster R-CNN [11] and Mask R-CNN [12] were adapted for crack detection, but their reliance on region proposal generation and ROI pooling limits inference speed. Recent works have shown that one-stage detectors can better balance precision and speed for micro-defect detection. However, fine-grained detection of linear, small-area cracks still poses challenges due to their morphological variance, sparse distribution, and low contrast.

B. YOLO-Based Crack Detection and Lightweight Enhancements

The YOLO series, from YOLOv1 [35] to the recent YOLOv11 [36], has established itself as a de facto baseline for real-time object detection due to its unified architecture and low-latency inference. The crack detection community has actively embraced and extended YOLO variants in pursuit of on-board deploy ability. Several tailored adaptations exist:

1) **Backbone Optimization:** Su et al. [37] replaced YOLOv5's CSPDarkNet with ShuffleNetv2 to accelerate inference on mobile devices. Zhao et al. [38] introduced GhostNet-based feature extractors, reducing parameters by 60% without degrading mAP.

2) **Contextual Awareness:** Yu et al. [39] integrated a bottleneck transformer into YOLOv5 to capture long-range dependencies in crack morphology. Wu et al. [40] embedded CBAM attention modules into YOLOv5s to enhance feature saliency for narrow cracks.

3) **Drone-Based Crack Inspection:** Jiang et al. [21] utilized YOLOv5 with a dual attention module to improve crack detection from UAV imagery under strong illumination variance and complex road textures.

These enhancements, while effective, generally rely on modifications to individual components (e.g. backbone, attention, loss). Most models still adopt independent detection heads per scale and lack targeted loss design for crack-specific patterns (e.g., bifurcation, continuity).

C. YOLO Architecture Advancements: Backbone, Neck, and Head Optimizations

Recent YOLO variants incorporate various architectural improvements to enhance small object detection and accelerate inference:

Backbone Improvements: Works such as TGC-YOLOv5 [38] merge lightweight GhostConv with Transformer-Guided Context modules, enabling enhanced crack contour extraction under minimal compute budgets. Similarly, MobileNetv3 and EfficientNet-Lite are often employed as drop-in backbones in real-time inspection settings [16].

Neck Fusion Modules: FPN, PANet, and BiFPN are widely adopted to aggregate features across multiple resolutions. However, FPN often amplifies semantic but not geometric information. Some researchers propose adaptive feature fusion (e.g., AFPN [41]) using learned attention gates to better preserve fine-grained crack boundaries during up-sampling and down-sampling processes. The Slim-Neck [22] architecture proposed replaces standard convolutions with GSConv and substitutes the original C3 module with VoVGSCSP, incorporating attention mechanisms to enhance crack feature focusing, which significantly improves small crack detection capability.

Detection Head and Loss Optimization: The decoupled head design of YOLOv8 allows for independent optimization of classification and localization. Recent works have explored parameter-sharing detection heads and lightweight deformable convolutions to reduce redundancy while enhancing detail sensitivity. In terms of loss functions, WIoU and Focal-EIoU improve bounding box regression by introducing uncertainty awareness and focal reweighting. The Inner-WIoU loss further prioritizes regression alignment in the interior region of crack boxes, thus improving bounding performance for elongated, thin targets.

Cross-Domain Transfer and Tracking Integration: Inspired by MobileSAM-Track [42], integrating lightweight tracking-

assisted segmentation enables multi-frame crack analysis in drone-based or vehicle-streamed videos. These methods offer promising pathways for high-fidelity crack localization under motion blur and temporal discontinuity.

Despite recent advances, three critical limitations persist for real-time crack detection in embedded environments: (1) scale-fragmented feature learning leads to poor micron-level crack edge delineation; (2) redundant multi-branch detection heads increase latency and memory overhead; and (3) generic IoU-based losses fail to account for the topological continuity and branching nature of road cracks. To overcome these issues, this paper proposes LP-YOLOv8, which introduces a scale-sensitive lightweight backbone, a detail-enhanced shared deformable detection head, and a topology-aware regression loss to achieve high-precision crack detection with real-time inference capability on resource-constrained platforms.

III. THE PROPOSED ROAD SURFACE CRACK DETECTION METHOD

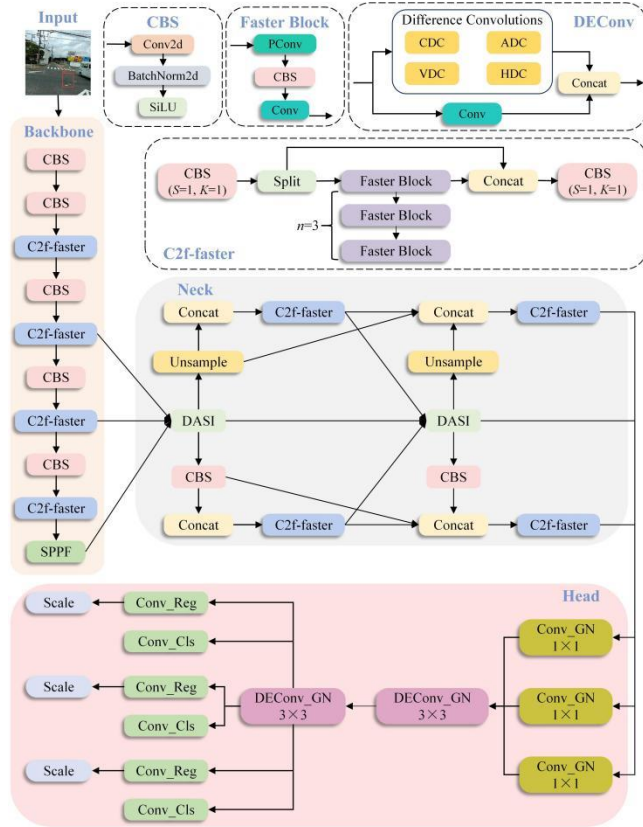


FIGURE 1. The network structure of the proposed LP_YOLOv8n.

To address the limitations of existing YOLO-based models in vehicle-mounted crack detection scenarios, we propose an enhanced model named LP-YOLOv8, as illustrated in Figure. 1. The design focuses on three goals: improving detection accuracy, reducing model parameter count, and accelerating inference speed. Based on the YOLOv8n framework, LP-YOLOv8 introduces lightweight and efficient modifications to

the backbone, neck, and detection head. The following subsections elaborate on each structural improvement.

A. MODEL LIGHTWEIGHTING IMPROVEMENTS

1) C2F-FASTER MODULE

As the depth of deep neural networks increases, the number of channels in feature maps often grows significantly. However, multiple channels may contain redundant or similar information, which not only leads to wasted computational resources but also may reduce the model's efficiency. Therefore, effectively reducing redundant computations while retaining useful information from the feature maps has become a key challenge for improving model performance.

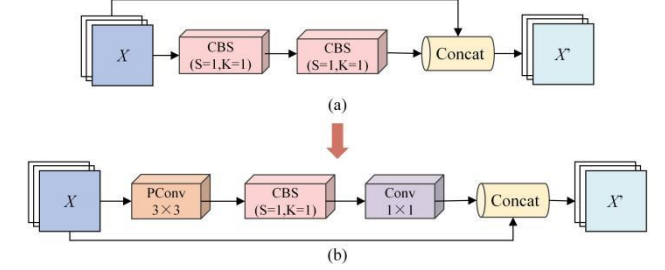


FIGURE 2. Structure comparison between (a) the bottleneck and (b) the faster block.

To address this issue, FasterNet [43] proposes an innovative method known as Partial Convolution (PConv). This method performs convolution operations only on a subset of the input feature map channels while keeping the remaining channels unchanged. This approach significantly reduces computational load and avoids the loss of useful information. Specifically, FasterNet performs convolution on one-quarter of the input channels, while the remaining three-quarters are left unchanged and later utilized through 1×1 pointwise convolutions. This method reduces computational cost while preserving model accuracy and significantly improves computational speed.

Building on Faster Net's idea of partial convolution, this paper introduces the faster block structure to optimize the Bottleneck module in YOLOv8. Specifically, the faster block reduces computation and improves inference speed by performing convolution only on one-quarter of the input channels. Figure 2(b) shows the structure of faster block. In the design of the faster block, the PConv is applied to only one-quarter of the input feature map channels. This ratio is inspired by the PConv strategy proposed in FasterNet, which demonstrates that performing convolution on a subset of channels can effectively reduce redundant computations while preserving discriminative semantic features. This work does not perform a detailed ablation study on the one-quarter ratio. However, this ratio is widely used in recent lightweight designs. It reflects a common balance between computational efficiency and representational capacity. After the partial convolution, a 1×1 pointwise convolution integrates the unchanged channels. This ensures no important features are lost, even with reduced computation.

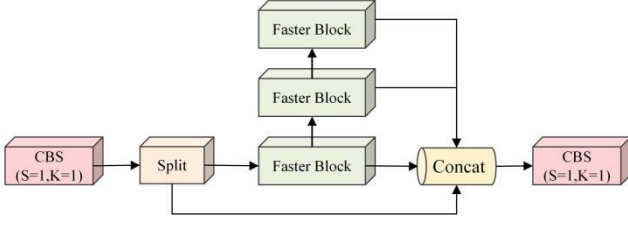


FIGURE 3. The structure of C2f-Faster ($n=3$).

To further reduce information loss caused by partial convolution, the CBS module (composed of convolution, batch normalization, and activation functions) is introduced in the faster block. In the CBS module, the output channel size of the 1×1 pointwise convolution is expanded to twice the original number of channels, fully utilizing the channels that were not convolved. A second 1×1 convolution reduces the channels back to the original size. This keeps the output dimensions of the main and shortcut paths consistent. It ensures a stable flow of information.

In the C2f module of YOLOv8, assume that the output channel size of the first CBS module is $2 \times c$. The following Split layer divides it into two feature maps, each containing c channels, which are stored in the list y . Each subsequent Bottleneck module takes the last element in the list y (i.e., the feature map with c channels) as input. After processing, the output of each bottleneck is added to the list for use in the next bottleneck module. After passing through n Bottleneck modules, the elements in the list y are concatenated along the channel dimension, resulting in a $(2+n) \times c$ -dimensional feature map, which is then passed through a second CBS module to reduce the channel size to the output channel size c_2 . Through this process, the C2f module effectively combines features from different scales, enhancing the model's feature representation ability.

However, although this strategy improves feature fusion, each additional Bottleneck module significantly increases the computational cost. Therefore, this paper proposes replacing the Bottleneck module with the faster block to reduce computational load and improve inference speed. Each faster block performs convolution on only one-quarter of the input channels, and the subsequent 1×1 convolution has a relatively small computational cost. As a result, the computational overhead of each faster block is approximately 1/16 of that of the Bottleneck module. By replacing Bottleneck with the faster block, the overall computational load in YOLOv8 is greatly reduced, leading to significant improvements in inference speed.

Figure 3 illustrates the C2f-Faster structure with $n=3$ (i.e., 3 Faster Blocks). Compared to the traditional C2f module, the C2f-Faster structure not only reduces computational overhead but also maintains high accuracy and strong feature representation capability. This optimization enables the model to infer more efficiently while preserving multi-scale feature fusion.

2) LIGHTWEIGHT SHARED DETAIL ENHANCEMENT CONVOLUTION DETECTION HEAD

The original detection head in YOLOv8 has certain limitations. First, each detection head requires two 3×3 and 1×1 convolutions to extract image information, which significantly increases the number of parameters in the algorithm, accounting for one-fifth of the overall computational cost. This makes it difficult to achieve optimal performance on resource-constrained mobile devices. Secondly, the original algorithm uses a traditional single-scale detection structure, which does not handle multi-scale objects well. The detection head predicts only from one scale of the feature map, ignoring the contribution of features from other scales to the detection result.

To address the abovementioned issue, we propose a lightweight shared de-tail-enhanced convolutional detection head (LSDECD). The main concept involves using a depth-wise convolution (DEConv) [44] that shares weight parameters to fuse feature maps from three scales. This approach reduces both the model's parameter counts and computational complexity. Additionally, we employ group normalization (GN) to enhance the accuracy of the detection head in terms of localization and classification. This design effectively improves the ability to extract multi-scale features, decreases the model's parameter size and memory usage, and boosts training and inference efficiency. The structure of the LSDECD detection head is illustrated in Figure 4.

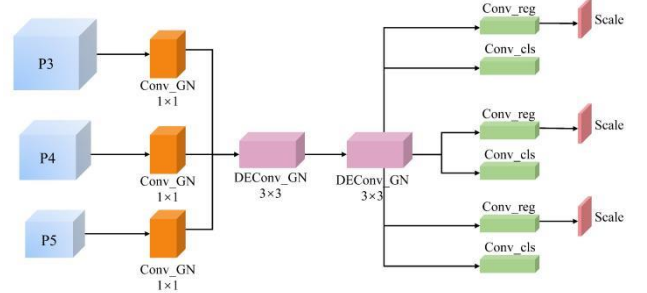


FIGURE 4. The network structure of LSDECD.

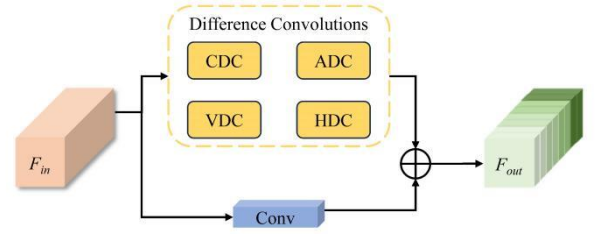


FIGURE 5. The network structure of DEConv.

After receiving the feature maps P3, P4, and P5 from the neck, the LSDECD first passes each branch through a 1×1 convolution with group normalization to adjust the channel dimensions and integrate the feature information from the neck. Then, a 3×3 DEConv with shared weight parameters is used to expand the receptive field, aggregating rich spatial contextual information and multi-scale information while reducing the model's parameter count. Finally, a 1×1 standard

convolution is applied to decouple the calculation of classification loss and regression loss. Additionally, a Scale layer is introduced after each regression branch. The Scale layer contains a learnable dynamic factor λ , which dynamically adjusts the target scale to address the issue of inconsistent target sizes detected by each detection head. The λ is updated in real-time according to the gradient descent algorithm during the training iterations.

DEConv extracts feature information by deploying five different types of convolutional layers in parallel. Specifically, these include standard convolution (Conv), central difference convolution (CDC), angular difference convolution (ADC), horizontal difference convolution (HDC), and vertical difference convolution (VDC). These differential convolution layers focus on extracting edge and texture information from the image by capturing pixel variations in different directions. Meanwhile, the standard convolution layer is responsible for collecting the intensity level information of the image, i.e., the actual brightness or color value of the pixels.

DEConv generates richer and more detailed feature maps by fusing the outputs of these parallel convolution layers, which is beneficial for subsequent classification and regression tasks. The structure of DEConv is shown in Figure 5, and its operational process is as follows:

$$F_{out} = DEConv(F_{in}) = \sum_{i=1}^5 F_{in} * K_i \quad (1)$$

where K represents the convolution kernels for CDC, ADC, HDC, VDC, and the standard convolution, respectively, and $*$ denotes the convolution operation.

B. MODEL PRECISE IMPROVEMENTS

1) FOCUSING DIFFUSION PYRAMID NETWORK

In the feature fusion stage, when using the PAN (Path Aggregation Network) structure to fuse features, multiple up sampling and down sampling operations are re-quired. This results in the loss of semantic features from the original

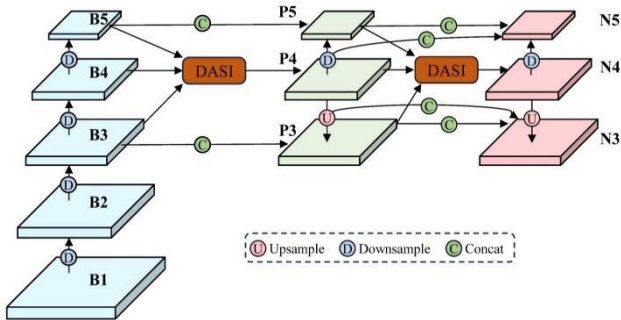


FIGURE 6. The network structure of FDPN.

image, which in turn leads to deep features losing the semantic information of small targets, while shallow features lack sufficient contextual information. For vehicle-mounted crack detection, losing small target features is critical. It may cause small cracks to be missed or wrongly classified. To address this issue, we propose a Focusing Diffusion Pyramid Network (FDPN). This network effectively integrates multi-

scale features from both adjacent upper and lower layers, as well as from the same level, using the Feature Attention and Selection Integration (DASI) module. This approach mitigates the problem of semantic feature loss. Furthermore, the feature diffusion mechanism ensures that each scale receives detailed contextual information. As a result, the model can detect and localize small targets with greater sensitivity, thereby enhancing the accuracy of small target detection. The structure of the FDPN is illustrated in Figure 6.

For the feature fusion of the N3, N4, and N5 layers, unlike the process, an additional step is introduced: after up sampling and down sampling the P4 layer, it relates to a skip connection to the N3 and N5 layers. This helps preserve more feature information, thereby enhancing the model's ability to integrate features. In FDPN, the focus of features is achieved through the DASI module. This module adaptively selects and fuses both high- and low-level features, effectively capturing and preserving multi-scale feature information. Low-level features refer to B3, P3, and N3, while high-level features refer to B5, P5, and N5.

Specifically, DASI first aligns the high-dimensional feature $F_h \in \mathbb{R}^{H_h \times W_h \times C_h}$, the low-dimensional feature $F_l \in \mathbb{R}^{H_l \times W_l \times C_l}$, and the current layer feature $F \in \mathbb{R}^{H \times W \times C}$ through operations such as convolution and bilinear interpolation. Subsequently, these features are divided into four equal parts along the channel dimension, resulting in $(h_i)_{i=1}^4 \in \mathbb{R}^{H \times W \times C/4}$, $(l_i)_{i=1}^4 \in \mathbb{R}^{H \times W \times C/4}$ and $(u_i)_{i=1}^4 \in \mathbb{R}^{H \times W \times C/4}$. The segmented features are then computed using the following formula:

$$\alpha = \text{sigmoid}(u_i), u'_i = \alpha l_i + (1 - \alpha) h_i \quad (2)$$

$$F'_u = [u'_1, u'_2, u'_3, u'_4], \hat{F}_u = \delta(\beta(\text{Conv}(F'_u)))$$

where, the sigmoid function is used to calculate the fusion weight α . When $\alpha > 0.5$, the model prioritizes retaining more detailed information; when $\alpha < 0.5$, it focuses more on contextual information. All u'_i values are merged along the channel dimension to obtain F'_u . Then, through convolution (Conv), batch normalization (β), and ReLU (δ) activation operations, the final feature map $\hat{F}_u$ that integrates multi-scale information is obtained.

2) INNER-WIOU

The original YOLOv8 model uses CIoU as the regression loss function because it considers factors such as overlap area, center distance, and aspect ratio. However, in vehicle-mounted images, road cracks tend to be small and irregular in shape. When manually labeling defects using Labelling software, the bounding boxes are sometimes slightly larger or smaller than the actual defect area, leading to low-quality annotated samples. CIoU heavily penalizes small or poorly annotated samples. As a result, it may misjudge the match between predicted and actual boxes.

Inner-IoU introduces a scaling factor ratio to adjust the size of the predicted bounding box, optimizing

samples at different IoU levels and accelerating the model's learning speed for small target defects, thus achieving faster convergence. Wise-IoU (WIoU) [27] introduces an outlier factor β to evaluate the quality of the predicted bounding box. The outlier factor is negatively correlated with anchor box quality, effectively alleviating issues such as incorrect gradient updates and excessively large magnitudes due to low-quality annotated samples. Thus, this study draws inspiration from Inner-IoU [28] to improve the WIoU loss function and proposes the Inner-WIoU loss function. The method enlarges predicted boxes proportionally. It emphasizes anchor boxes for small and normally labeled defects. This approach effectively addresses the issues mentioned above and enhances the model's generalization ability by minimizing interference during the training phase.

The Inner-WIoU loss function, which is an improvement of WIoU using the concept of Inner-IoU, is calculated as follows:

$$WIoU^{inner} = (\beta)^\gamma \exp\left(\frac{\rho(b, b^{gt})}{(c^2)^*}\right) (1 - IoU^{inner}) \quad (3)$$

$$\beta = \frac{(IoU^{inner})^*}{\bar{L}_{IoU}} \in [0, +\infty]$$

where, ρ represents the Euclidean distance between the center coordinates of the predicted box and the object box; c^2 represents the minimum enclosing area of the predicted box and the object box; the superscript $*$ indicates that this operation is detached from the computation graph to prevent obstructing gradient convergence; γ is a hyperparameter, set to 0.5; and $\bar{L}_{IoU}$ is the moving average with momentum m .

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT

The experiments in this study were carried out on a Linux operating system using an Intel® Xeon® Platinum 8255C CPU and a GeForce RTX 2080 Ti GPU with 12GB of memory. The software environment consisted of Python 3.8, CUDA 11.3, and the PyTorch 1.10 deep learning framework. The initial image size was set to 600x600 pixels, and the initial learning rate was 0.01, with a weight decay factor of 0.0005, and a batch size of 16. Extensive experimentation indicated that the model converged after 200 training epochs. Instead of initializing the model with pre-trained weights (e.g. ImageNet), we adopted random weight initialization. This decision was based on the domain-specific nature of the crack detection task, which significantly differs from the object categories and visual features present in large-scale general datasets. Using random initialization allowed for full task-oriented learning from scratch, avoiding potential feature misalignment and overfitting from unrelated semantic priors. Although this approach may require longer convergence times and more careful tuning of hyperparameters, it provides greater flexibility and control

over the model's learning process. It also enables more precise optimization for challenges specific to vehicle-mounted scenarios, such as motion blur, complex backgrounds, and ultra-thin surface cracks. Throughout the training process, thorough fine-tuning was conducted to enhance the model's capability to adapt to the specific task of road crack detection. This approach provided greater control over the model's architecture and performance, allowing for more effective management of challenges such as blurry small cracks and the extraction of incomplete information from vehicle-mounted images.

B. VEHICLE MOUNTED IMAGE DATASET

This study utilizes the RDD2022 dataset [45] as the source of experimental data. The RDD2022 dataset is provided by the CRDDC challenge and serves as an extension of the original RDD2020 dataset. It consists of road images captured from a vehicle-mounted perspective across five countries: India, the Czech Republic, Japan, China, Norway, and the United States, totaling 46,336 images. The dataset includes four types of road defects: longitudinal cracks (D00), transverse cracks (D10), alligator cracks (D20), and potholes (D40). Because the images in this dataset are collected from street-level views, they feature complex backgrounds and often contain defects at var-ying scales, which poses a challenge to detection accuracy. To ensure the validity of the experiment, images without defects were excluded. After filtering, a total of 9,181 valid images remained. These images were divided into training, validation, and test sets in a 7:2:1 ratio, resulting in 6,427 images in the training set, 1,836 in the validation set, and 918 in the test set.

C. EVALUATION METRICS

The primary metrics used to evaluate object detection accuracy include mean Average Precision (mAP), Recall (R), and the area under the Precision-Recall (P-R) curve. In this study, mAP@0.5 was chosen as the evaluation metric for model accuracy. mAP@0.5 is obtained by averaging the Average Precision (AP@0.5) across all categories. The formulas for calculating Precision (P), Recall (R), AP, and mAP are as follows:

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

$$R = \frac{TP}{TP + FN} \quad (4)$$

$$AP = \int_0^1 P(r) dr$$

$$mAP = \frac{1}{c} \sum_{i=1}^c AP_i$$

where c represents the total number of categories in the images, i is the number of detection instances, TP is the number of correctly predicted positive samples, FN is the number of false negative predictions, and FP is the number of false positive predictions.

Additionally, Frames Per Second (FPS), model complexity in terms of GFLOPs (Gi-ga Floating Point Operations), and

the number of parameters is introduced as light-weight evaluation metrics to assess the degree of model lightweight. FPS reflects the refresh rate of model inference speed, and its calculation formula is as follows:

$$FPS = \frac{frameNum}{elapsedTime} \quad (5)$$

where *frameNum* represents the total number of images detected, and *elapsedTime* represents the total time taken for detection. The larger the FPS value, the faster the detection rate, indicating better real-time performance of the model.

D. COMPARISONS WITH OTHER STATE-OF-THE-ART METHODS

To comprehensively evaluate the superiority of the proposed LP_YOLOv8 algorithm in detecting road cracks from vehicle-mounted images, we conducted comparative experiments against several widely used object detection methods. These include traditional frameworks such as Faster R-CNN and DETR, as well as a variety of lightweight and mainstream YOLO series models: YOLOv5n, YOLOv6n, YOLOv7n-Tiny, Damo-YOLO, YOLOv8n, YOLOv10n, and the recently released YOLOv11n [35]. All models were evaluated on five key performance metrics: detection accuracy (mAP@0.5), F1 score, inference speed (FPS), computational complexity (GFLOPs), and parameter count (Params). The results are summarized in Table 1 and visually presented in Figure 7 via a radar chart. From the Table1, the proposed LP_YOLOv8 model achieved the best performance among all models in terms of both mAP@0.5 and F1 score, reaching 57.3% and 58.1%, respectively. The Figure 7 further emphasizes LP_YOLOv8's well-balanced and leading performance across all evaluation dimensions.

In terms of computational complexity, LP_YOLOv8 demonstrates outstanding efficiency, requiring only 3.6 GFLOPs, which is significantly lower than that of Faster R-CNN (370.2 GFLOPs) and YOLOv11n (8.9 GFLOPs). This drastic reduction in floating-point operations during inference translates into faster processing and lower energy consumption. Such lightweight computational demand makes LP_YOLOv8 especially suitable for deployment in resource-constrained environments, such as vehicle-mounted embedded systems and edge computing platforms. Importantly, this efficiency is achieved without compromising detection performance, underscoring the model's real-time applicability.

In terms of parameter count, LP_YOLOv8 contains only 1.1 million parameters, far fewer than YOLOv8n (3M) and DETR (31.2M). This compact model architecture substantially reduces memory usage and hardware requirements, enhancing its feasibility for deployment on devices with limited memory and processing power. The significant reduction in both computational complexity and model size reinforces LP_YOLOv8's practicality for embedded and mobile applications.

When compared with other YOLO-based models, LP_YOLOv8 continues to deliver superior performance. It achieves the highest mAP@0.5 (57.3%) and F1 score (58.1%) among all tested models, while also maintaining notable advantages in both GFLOPs and parameter count.

TABLE I
COMPARISONS WITH OTHER OBJECT DETECTION METHODS ON OUR DATASET.

Method	mAP@0.5/%	F1	GFLOPs/G	Params/M	FPS /frame·s ⁻¹
FasterR-CNN	51.2	49.4	370.2	137.1	54.2
DETR	54.3	55	101	31.2	111.9
YOLOv5n	54.1	54.8	7.1	2.5	138.7
YOLOv6n	55.1	56	11.2	4.2	121.2
YOLOv7n-Tiny	56.5	57.1	13	6.9	212.5
Damo_YOLO	56.9	57.6	6.2	5.5	93.4
YOLOv8n	56.2	56.8	8.1	3	152.9
YOLOv10n	55.5	56.3	8.2	2.7	174.9
YOLOv11n	56.1	56.9	8.1	2.9	173.6
LP_YOLOv8	57.3	58.1	3.6	1.1	322.6

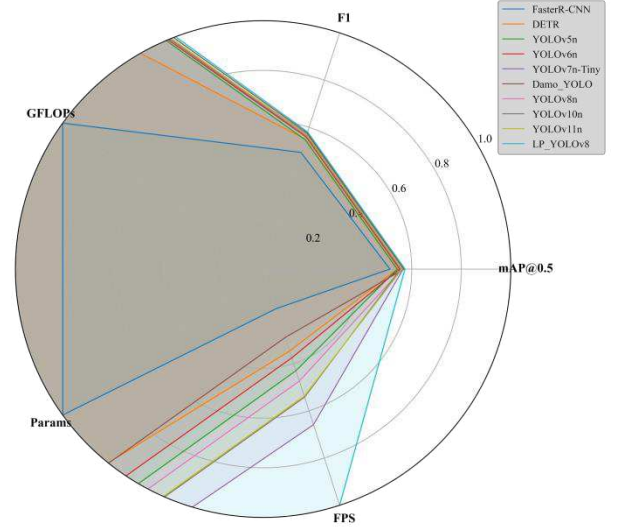


FIGURE 7. Object Detection Performance Comparison.

Specifically, LP_YOLOv8 surpasses YOLOv8n (56.2% mAP, 152.9 FPS) and YOLOv10n (55.5% mAP, 174.9 FPS), achieving an impressive 322.6 FPS, which is more than double the speed of YOLOv8n. Although YOLOv11n offers slight improvements over YOLOv8n and YOLOv10n in accuracy and speed, it still falls short of LP_YOLOv8 across all five evaluated metrics, particularly in terms of model efficiency. Notably, although YOLOv11n demonstrates slightly better performance than YOLOv8n and YOLOv10n in both

accuracy and speed, it still lags behind LP_YOLOv8 in all five metrics, especially in model efficiency.

This remarkable frame rate, combined with high accuracy and low resource consumption, gives LP_YOLOv8 a clear advantage in real-time detection scenarios. Its ability to deliver fast and accurate inference makes it particularly well-suited for applications that demand immediate response, such as autonomous driving, vehicle-mounted inspection systems, and real-time intelligent monitoring.

In summary, LP_YOLOv8 outperforms both classical and state-of-the-art lightweight models in terms of detection accuracy, speed, and computational efficiency. These results strongly support its robustness, scalability, and deployment potential in real-world road crack detection tasks.

E. PERFORMANCE ANALYSIS OF THE LIGHTWEIGHT IMPROVED MODEL

1) IMPACT OF C2F-FASTER ON BACKBONE LIGHTWEIGHTING

To validate the impact of the Faster Block on the lightweight design of the backbone network, we replaced the C2f modules in the 2nd, 4th, 6th, and 8th layers of the back-bone network. The changes in parameter count, computational load, and inference speed after replacement are presented in Table 2. After the replacement, the parameter count was reduced by 0.00408M, 0.02422M, 0.976M, and 0.26195M, with the most significant reduction occurring in the 6th layer, demonstrating a notable lightweight effect. Meanwhile, the computational load decreased by 0.21G, 0.3G, 0.31G, and 0.21G, indicating a significant improvement in computational efficiency through optimization of the processing flow.

Regarding inference speed, the network showed an improvement of 0.42ms, 3.07ms, and 3.87ms in the 2nd, 6th, and 8th layers, respectively. Only the 2nd layer showed a slight decrease of 1.6ms. This suggests that, except for the 2nd layer, the replacement effectively improved inference speed, optimizing real-time processing capability, especially in embedded and edge computing environments. These experimental results demonstrate that the Faster Block significantly reduces model parameters and computational complexity, while improving inference speed, making lightweight models more competitive in practical applications.

2) IMPACT OF LSDECD ON HEAD LIGHTWEIGHTING

To validate the effectiveness of LSDECD, Table 3 presents a performance comparison of Efficient Head (EH), SEAMHead (SEAMH), MultiSEAMHead (MSEAM), and LSDECD applied to the same benchmark model. The EH shows a slight improvement in detection accuracy but with a significant increase in parameter count. The SEAMH maintains accuracy while reducing both parameters count and computational load, demonstrating its good performance. Although the MSEAM reduces computational load by 0.9G, it results in a 0.5% decrease in detection accuracy, and its parameter count increases by 64%, making it less favorable for practical deployment. In contrast, the LSDECD design outperforms the other detection heads, combining the advantages of group

norm and shared convolutions to make the model head more lightweight. It not only improves mAP50 by 0.3% but also reduces parameter count and computational cost by 21.33% and 20.73%, respectively.

TABLE II
COMPARISONS WITH OTHER OBJECT DETECTION METHODS ON OUR DATASET

Layer number	Module	Params/M	FLOPs/G	Time/MS
2	C2f	0.00736	0.39	8.49
2	C2f_Faster	0.00328	0.18	8.07
4	C2f	0.04966	0.64	8.42
4	C2f_Faster	0.02544	0.34	10.02
6	C2f	0.19763	0.64	7.74
6	C2f_Faster	0.10003	0.33	4.67
8	C2f	0.46028	0.37	5.93

F. PERFORMANCE ANALYSIS OF THE PRECISE-IMPROVED MODEL

1) PERFORMANCE COMPARISON OF DIFFERENT FEATURE FUSION NETWORKS

To verify the effectiveness of FDPN, the results of applying AFPN, HSFPN, BiFPN, and FDPN to the same baseline model were compared, as shown in figure 9. Compared to the baseline model, AFPN not only showed a decrease in accuracy but also an increase in computational load, making it unsuitable for vehicle detection tasks in real-world scenarios. HSFPN and BiFPN are somewhat more lightweight, but their accuracy still falls short of FDPN. The FDPN used in this study can flexibly control the representational capabilities of high-level and low-level features, effectively fusing the data flow between different layers. With only a 0.1 G increase in computational load, FDPN achieved the highest detection accuracy of 64.6%, and the parameter count was reduced to 2.55 M.

2) PERFORMANCE COMPARISON OF IMPROVED LOSS FUNCTIONS

First, we conducted comparative experiments to investigate the impact of the ratio coefficient in Inner-WIoU on model performance, with the results shown in Table 3. As the ratio increases from 0.9 to 1.3, there is no significant linear trend in the performance of the loss function, and in most cases, the model's performance remains relatively stable. When the ratio is set to 1.1, the model achieves the best performance with an mAP of 97.1%. This indicates that both too large and too small a ratio is detrimental to model optimization, and the choice of ratio should be reasonably adjusted based on the specific dataset and model characteristics.

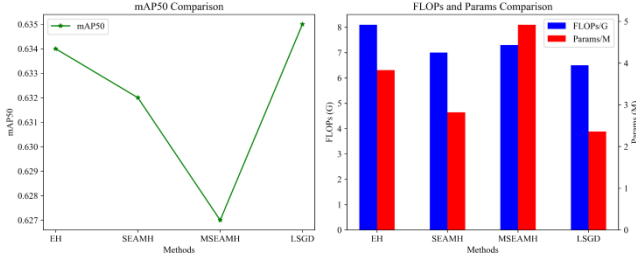


FIGURE 8. Performance comparison of different detection heads.

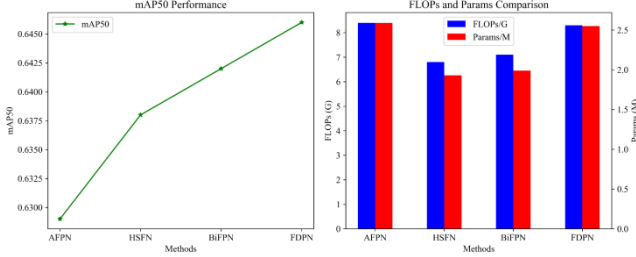


FIGURE 9. Performance comparison of different feature fusion networks.

TABLE III
COMPARATIVE EXPERIMENTAL RESULTS OF RATIO COEFFICIENT AND DIFFERENT LOSS FUNCTIONS

Model	Loss function	P/%	R/%	mAP (%)
LP_YOLOv8	CIoU	96.2	94.1	96.7
	WIoU	96.4	94.6	96.9
	SIoU	95.6	94.1	96.7
	NWD	94.4	92.2	93.5
LP_YOLOv8	Inner-WIoU (0.9)	96.6	93.2	96.2
	Inner-WIoU (1.0)	96.6	93.9	96.7
	Inner-WIoU (1.1)	97.0	94.3	97.1
	Inner-WIoU (1.2)	95.5	94.1	96.9
	Inner-WIoU (1.3)	95.2	92.7	96.0

Furthermore, considering that WIoU is not the latest IoU-based derived loss function, we further compared Inner-WIoU with other recently proposed loss functions, such as SIoU and NWD, with the experimental results presented in Table 3. Compared to CIoU, WIoU, and SIoU, Inner-WIoU improved the mAP@0.5 by 0.4%, 0.2%, and 0.5%, respectively. Notably, NWD performed poorly in the small target road crack detection task on vehicle-mounted images, further confirming the superiority of Inner-WIoU in this specific application scenario.

G. Generalization Evaluation on Additional Datasets

To further validate the generalization capability of the proposed LP-YOLOv8 model, we extended our evaluation to three publicly available road crack datasets: CrackForest [46], DeepCrack [47], and CFD [46]. These datasets represent

diverse conditions in terms of crack morphology, background complexity, illumination variability, and image resolution, thereby providing a robust testbed to assess model transferability and real-world applicability.

We compared LP-YOLOv8 with three representative object detection models: YOLOv11n, YOLOv8n, and Faster R-CNN. The evaluation metrics included mAP@0.5, Params, and FPS, with all models tested under the same training configuration and hardware environment to ensure fairness.

As shown in Table 4, a comprehensive comparison of LP-YOLOv8 against several mainstream object detection models—including Faster R-CNN, YOLOv8n, and YOLOv11n—was conducted on three widely used road crack datasets: CrackForest, DeepCrack, and CFD. The evaluation covers four key metrics relevant to real-world deployment: detection accuracy (mAP@0.5), computational complexity (GFLOPs), model size (Params), and inference speed (FPS).

The proposed LP-YOLOv8 consistently achieves the highest accuracy while maintaining the lowest computational cost and parameter count, along with the fastest inference speed across all datasets.

As further illustrated in Figure 8(a–d), LP-YOLOv8 consistently surpasses other detection models across the four-evaluation metrics. On the CrackForest dataset, LP-YOLOv8 achieves the highest mAP@0.5 of 57.1%, with only 3.6 GFLOPs and 1.1 million parameters, while reaching a remarkable 310.3 FPS, more than twice the speed of YOLOv8n.

TABLE IV
COMPARATIVE PERFORMANCE OF DIFFERENT MODELS ON MULTIPLE ROAD CRACK DATASETS

Dataset	Model	mAP@0.5 (%)	GFLOPs (G)	Params (M)	FPS
CrackForest	Faster R-CNN	52.4	370.2	137.1	58.7
	YOLOv8n	55.3	8.1	3.0	143.6
	YOLOv11n	55.9	8.1	2.9	161.2
	LP-YOLOv8	57.1	3.6	1.1	310.3
DeepCrack	Faster R-CNN	53.7	370.2	137.1	60.1
	YOLOv8n	56.1	8.1	3.0	147.5
	YOLOv11n	56.6	8.1	2.9	164.7
	LP-YOLOv8	57.8	3.6	1.1	318.9
CFD	Faster R-CNN	54.8	370.2	137.1	57.3
	YOLOv8n	56.4	8.1	3.0	149.2
	YOLOv11n	57.0	8.1	2.9	166.1
	LP-YOLOv8	58.4	3.6	1.1	325.6

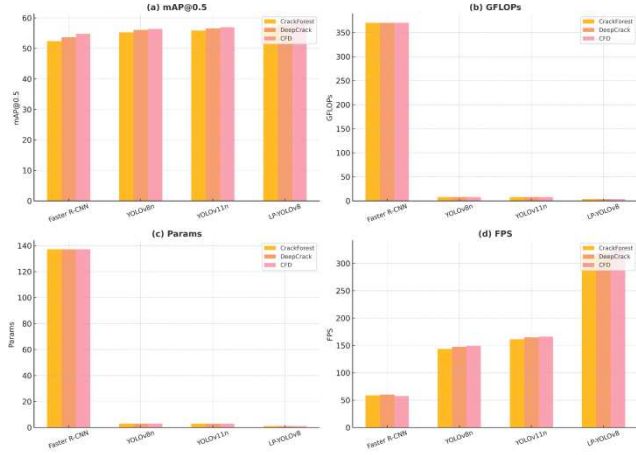


FIGURE 10. Comparative performance of different models—Faster R-CNN, YOLOv8n, YOLOv11n, and the proposed LP-YOLOv8—on three public road-crack datasets (CrackForest, DeepCrack, and CFD) in terms of (a) mAP@0.5, (b) GFLOPs, (c) Params, and (d) FPS.

On the DeepCrack dataset, characterized by noisy textures and micro-scale cracks, LP-YOLOv8 maintains superior performance with a 57.8% mAP@0.5, while preserving its lightweight and high-speed design. It maintains the same minimal resource usage (3.6 GFLOPs and 1.1M Params) and achieves 318.9 FPS, reflecting its robustness under complex conditions.

On the CFD dataset, which involves significant lighting variation and structural diversity, LP-YOLOv8 again outperforms other methods with a 58.4% mAP@0.5, maintaining excellent efficiency with 3.6 GFLOPs, 1.1M parameters, and a top-speed of 325.6 FPS, as evident in Fig. 8(d).

From the perspective of vehicle-mounted deployment, where constraints on power, memory, and latency are critical, three key metrics—GFLOPs, Params, and FPS—directly determine practical viability. Across all datasets, LP-YOLOv8 demonstrates: the lowest GFLOPs, reducing inference latency and energy consumption; the smallest parameter count, enabling compact model storage and fast startup; the highest FPS, with speeds exceeding 310 FPS, ensuring real-time video analysis.

These results verify that LP-YOLOv8 achieves an optimal balance between detection performance and computational efficiency. The architectural innovations introduced—namely, the FDPN for improved cross-scale fusion, the LSDECD detection head for fine-grained detail preservation, and the Inner-WIoU loss for geometry-aware localization—collectively contribute to LP-YOLOv8’s outstanding performance and real-world deployment potential.

H. ABLATION EXPERIMENT

We conducted eight ablation experiments to evaluate the effectiveness of the improved YOLOv8n algorithm. These experiments focused on four modules: the Faster Block, the Focused Diffusion Pyramid Network (FDPN), the

Lightweight Shared Detail Enhancement Convolution Detection Head (LSDECD), and the Inner-WIoU loss function. A checkmark (“√”) indicates the inclusion of a module, while a cross (“×”) signifies its exclusion. The results of the experiments are presented in Table 5.

Integrating the faster block into the backbone helped reduce redundant information. It also enhanced feature extraction ability, resulting in a 0.9% improvement in mAP@0.5. Additionally, it slightly reduced both the parameter count and computational cost. The introduction of the FDPN in the neck further enhanced mAP by 1.3%. This improvement demonstrates that the focusing module can effectively integrate multi-scale feature information and boost the model’s ability to extract small target features through the diffusion mechanism. Furthermore, using the LSDECD detection head led to a modest 0.7% increase in mAP while significantly decreasing the parameter count and computational costs by 21.6% and 19.8%, respectively. This indicates that the design of DEConv and the use of shared parameters can lower model complexity while enhancing detection accuracy.

Replacing the loss function from CIoU to the improved Inner-WIoU reduced the penalty on low-quality labeled samples and small target defects, leading to a 1.1% increase in mAP. After applying these four improvement strategies, the FPS slightly decreased but still met real-time requirements. The mAP increased by 2.6% compared to the baseline, while the parameter count and computational cost were reduced by 35.4% and 23.5%, respectively.

In summary, the algorithm designed in this paper effectively balances detection accuracy and structural lightweight design, making it feasible for real-time detection on vehicle-mounted devices.

To more intuitively demonstrate the effect of the proposed model improvements, we compared the mAP@0.5 and Recall curves during training for both the original and improved models. As shown in Figure 11, after 100 training epochs, the improved model outperformed the original model in both mAP50 and Recall, indicating that the improved model enhances road crack detection accuracy and reduces missed detection rates.

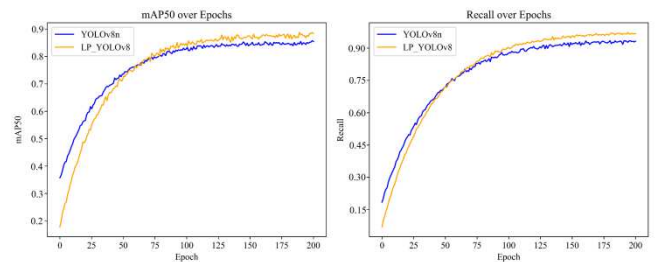


FIGURE 11. Contrast curves of mAP50 and Recall were improved.

TABLE V
COMPARATIVE EXPERIMENTAL RESULTS OF RATIO COEFFICIENT AND DIFFERENT LOSS FUNCTIONS

Model	Faster block	FDPN	LSDECD	Inner-WIoU	mAP/%	Params/M	GFLOPs/G	FPS	Size/MB
a	×	×	×	×	94.5	3.01	8.1	115	6.3
b	√	×	×	×	95.4	2.64	7.1	80	5.6
c	×	√	×	×	95.8	2.74	8.4	84	5.4
d	√	√	√	×	95.2	2.36	6.5	65	5.3
e	×	√	√	√	95.6	3.01	8.1	120	6.3
f	√	×	√	√	96.2	2.37	7.4	93	5
g	√	√	√	×	96.7	1.94	6.2	85	4.5
h	√	√	√	√	97.1	1.94	6.2	85	4.5

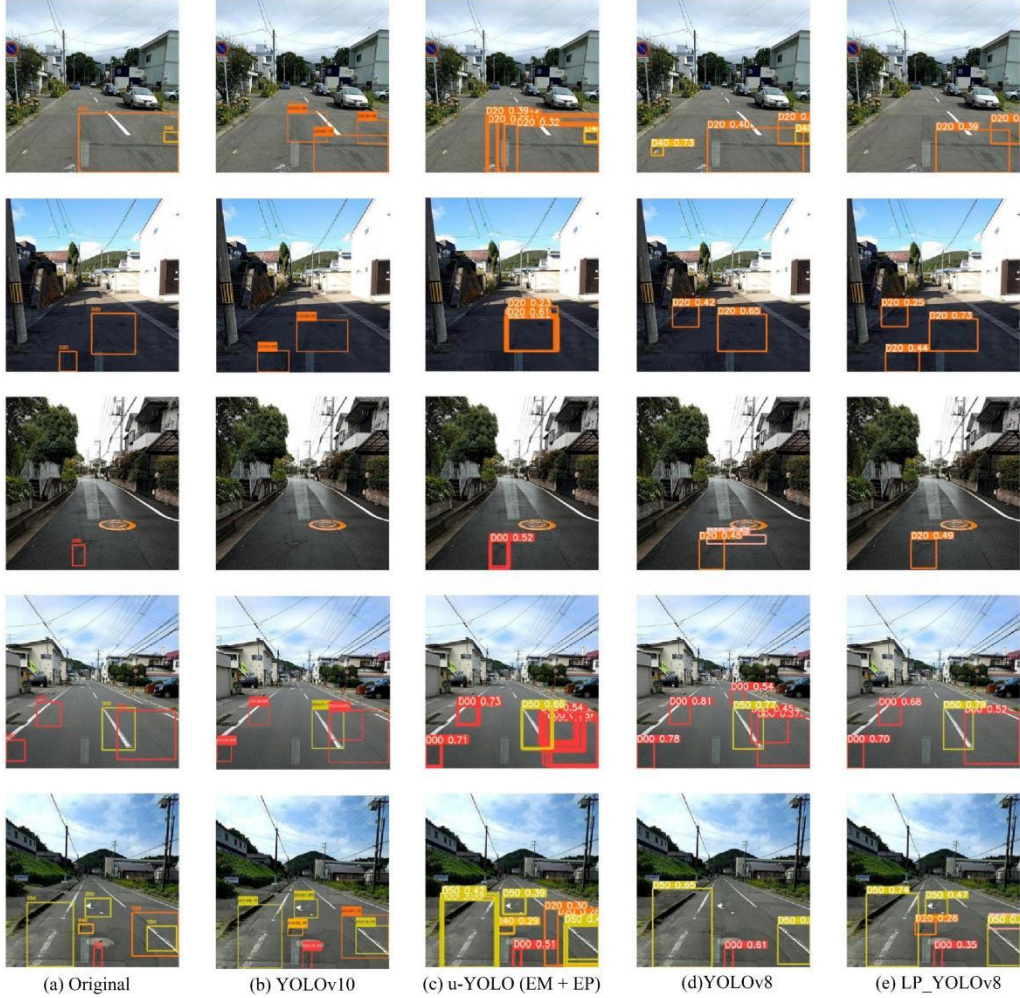


FIGURE 12. Qualitative comparison results on our dataset.

I. VISUAL ANALYSIS

Figure 11 presents a qualitative comparison between the proposed LP_YOLOv8 and other advanced methods (including YOLOv10 [29], YOLOv8, and u-YOLO (EM + EP) [32]) on our dataset. To ensure a fair comparison, we trained and tested these models using the same parameters and

annotated the confidence scores on the prediction boxes for better visualization.

Figure 11(b) shows that YOLOX demonstrates some capability in detecting road cracks, although it may have some limitations in detecting small cracks. However, it accurately locates, resizes, and classifies prediction boxes for other

defects, showing good overall performance, albeit with slightly lower accuracy.

In contrast, u-YOLO (EM + EP) shows a significant improvement in accuracy. However, as an ensemble model, it tends to generate more prediction boxes, as shown in figure 11(c), which may impact its precision. YOLOv8, built upon u-YOLO (EM + EP), enhances performance, significantly increases speed, and maintains high accuracy. Although Figure 11(d) displays good overall performance, YOLOv8 tends to produce more false positives, marking cracks in blank areas, which affects the model's overall accuracy.

Among the four algorithms, LP_YOLOv8 performs the best. figure 11(e) demonstrates that the prediction boxes of LP_YOLOv8 are the closest to the actual locations and size distributions of the defects, with the highest confidence scores. Compared to other common algorithms, the proposed method performs more satisfactorily in road crack detection from vehicle-mounted images.

V. CONCLUSIONS

This study focuses on the detection of road images captured by embedded devices mounted on vehicles. In this scenario, traditional YOLO models face issues such as feature learning with scale fragmentation and redundant detection heads. To address these problems, we propose the LP-YOLOv8 model.

On one hand, by designing the faster block structure and the LSDECD (Lightweight Shared Detail Enhanced Convolution Detection) structure, we achieve a model lightweight while maintaining detection accuracy. On the other hand, the introduction of the FDPN (Focused Diffusion Pyramid Network) feature fusion method and the Inner-WIoU loss function enhances detection accuracy without adding significant complexity to the model. These innovations comprehensively address the challenges mentioned above, enhancing the accuracy and comprehensiveness of feature extraction while achieving both lightweight design and efficient inference.

The experimental results show that LP_YOLOv8 significantly improves the efficiency of road crack detection, meeting the real-time requirements of such tasks. Additionally, it reduces the error rate of road crack detection to a certain degree, enhancing the model's stability and generalization ability. Specifically, the mAP value reaches 59.17%, and the F1 score reaches 58.76%. Compared to the original YOLOv8, the proposed method improves the mAP by 3.41% and the F1 score by 2.25%. Furthermore, the algorithm achieves 85 FPS, enabling fast and accurate detection of road surface cracks.

Although the LP-YOLOv8 model demonstrates excellent accuracy and computational efficiency in road surface crack detection, further research is required in several key directions. First, to mitigate performance degradation under low-light and visually degraded conditions—such as nighttime scenarios and shadow occlusion, which are common in vehicle-mounted environments—future work should explore the integration of

low-light enhancement networks, histogram equalization, and self-supervised denoising techniques to improve robustness against complex illumination variations. Second, the current reliance on static image detection limits adaptability in dynamic inspection scenarios; this can be addressed by incorporating lightweight temporal tracking modules, such as the tracking-segmentation framework in MobileSAM-Track, to extend the model's capability to real-time video stream analysis. Furthermore, at the architectural level, introducing adaptive attention mechanisms or Transformer-based modules to enhance multi-scale feature aggregation in the neck and detection head is expected to improve the detection of ultra-thin or partially occluded cracks, which remain challenging due to their irregular morphological characteristics.

REFERENCES

1. Zhang, S., Fu, Z., Li, G. & Liu, A. Lane crack detection based on saliency. *Remote Sens.* **15**, 4146 (2023).
2. Guest, G., Zhang, J., Maadani, O. & Shirkhani, H. Incorporating the impacts of climate change into infrastructure life cycle assessments: a case study of pavement service life performance. *J. Ind. Ecol.* **24**, 356–368 (2020).
3. Smith, J.D., Lee, S. & Kazmierowski, T.J. Use of pavement management system data to enhance pavement performance specifications in Canada. *Transp. Res. Rec.* **2573**, 60–68 (2016).
4. Yao, H. et al. Perspectives on the application of waste materials in asphalt based on molecular dynamics simulations: a review. *Energy Fuels* **38**, 14839–14865 (2024).
5. Zou, X. et al. The effects of weather factors on road traffic casualties: analysis on provincial panel data of China from 2006 to 2021. *Heliyon* **10**, e29189 (2024).
6. Andrei, D. & Arabestani, M. ASTM D6433 pavement condition index variability study. California State Polytechnic Univ., Pomona, CA, USA (2018).
7. Li, Z. et al. Semi-automatic crack width measurement using an OrthoBoundary algorithm. *Autom. Constr.* **158**, 105251 (2024).
8. Huang, S.H. & Pan, Y.C. Automated visual inspection in the semiconductor industry: a survey. *Comput. Ind.* **66**, 1–10 (2015).
9. Quintana, M., Torres, J. & Menéndez, J.M. A simplified computer vision system for road surface inspection and maintenance. *IEEE Trans. Intell. Transp. Syst.* **17**, 608–619 (2015).
10. Sattar, S., Li, S. & Chapman, M. Road surface monitoring using smartphone sensors: a review. *Sensors* **18**, 3845 (2018).
11. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 1137–1149 (2016).
12. He, K., Gkioxari, G., Dollár, P. & Girshick, R. Mask R-CNN. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2961–2969 (2017).
13. Kang, D., Benipal, S.S., Gopal, D.L. & Cha, Y.J. Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning. *Autom. Constr.* **118**, 103291 (2020).
14. Kim, B. & Cho, S. Image-based concrete crack assessment using mask and region-based convolutional neural network. *Struct. Control Health Monit.* **26**, e2381 (2019).
15. Wang, L., Tao, D., Wang, R., Wang, R. & Li, H. Big map R-CNN for object detection in large-scale remote sensing images. *Math. Found. Comput.* **2**, 293–310 (2019).
16. Deng, J., Lu, Y. & Lee, V.C.S. Imaging-based crack detection on concrete surfaces using You Only Look Once network. *Struct. Health Monit.* **20**, 484–499 (2021).
17. Han, D., Qi, H., Wang, S., Hou, D. & Wang, C. Adaptive stepsize forward-backward pursuit and acoustic emission-based health state assessment of high-speed train bearings. *Struct. Health Monit.* **24**, 14759217241271036 (2024).
18. Gallagher, J.E. & Oughton, E.J. Surveying You Only Look Once (YOLO) multispectral object detection advancements, applications, and challenges. *IEEE Access* **13**, 7366–7395 (2025).

19. Zhang, J., Qian, S. & Tan, C. Automated bridge crack detection method based on lightweight vision models. *Complex Intell. Syst.* **9**, 1639–1652 (2023).
20. Yu, G. & Zhou, X. An improved YOLOv5 crack detection method combined with a bottleneck transformer. *Mathematics* **11**, 2377 (2023).
21. Jiang, Y. et al. RDD-YOLOv5: road defect detection algorithm with self-attention based on unmanned aerial vehicle inspection. *Sensors* **23**, 8241 (2023).
22. Hu, H. et al. Road surface crack detection method based on improved YOLOv5 and vehicle-mounted images. *Measurement* **229**, 114443 (2023).
23. Zhu, J., Wu, Y. & Ma, T. Multi-object detection for daily road maintenance inspection with UAV based on improved YOLOv8. *IEEE Trans. Intell. Transp. Syst.* (2024).
24. Redmon, J. & Farhadi, A. YOLO9000: better, faster, stronger. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 7263–7271 (2017).
25. Zhao, M. et al. MED-YOLOv8s: a new real-time road crack, pothole, and patch detection model. *J. Real-Time Image Process.* **21**, 26 (2024).
26. Wang, X., Jiang, H., Mu, M. & Dong, Y. A dynamic collaborative adversarial domain adaptation network for unsupervised rotating machinery fault diagnosis. *Reliab. Eng. Syst. Saf.* **242**, 110662 (2024).
27. Wang, X., Gao, H., Jia, Z. & Li, Z. BL-YOLOv8: an improved road defect detection model based on YOLOv8. *Sensors* **23**, 8361 (2023).
28. Lin, T.Y. et al. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2117–2125 (2017).
29. Liu, S., Qi, L., Qin, H., Shi, J. & Jia, J. Path aggregation network for instance segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 8759–8768 (2018).
30. Hoang, N.D. & Nguyen, Q.L. Metaheuristic optimized edge detection for recognition of concrete wall cracks: a comparative study on the performances of Roberts, Prewitt, Canny, and Sobel algorithms. *Adv. Civ. Eng.* **2018**, 1–15 (2018).
31. Javed, M.A. et al. Lane line detection and object scene segmentation using Otsu thresholding and the fast Hough transform for intelligent vehicles in complex road conditions. *Electronics* **12**, 1079 (2023).
32. Rani, B.M.S. et al. Road identification through efficient edge segmentation based on morphological operations. *Trait. Signal* **38**, 1257–1264 (2021).
33. Zhu, G. et al. RHA-Net: an encoder-decoder network with residual blocks and hybrid attention mechanisms for pavement crack segmentation. Preprint at <https://arxiv.org/abs/2207.14166> (2022).
34. Sarhadi, A., Ravanshadian, M., Monirabbasi, A. & Ghanbari, M. Optimizing concrete crack detection: an attention-based SWIN U-Net approach. *IEEE Access* **12**, 77575–77585 (2024).
35. Hussain, M. YOLOv1 to v8: unveiling each variant—a comprehensive review of YOLO. *IEEE Access* **12**, 42816–42833 (2024).
36. Dong, X., Yuan, J. & Dai, J. Study on lightweight bridge crack detection algorithm based on YOLO11. *Sensors* **25**, 3276 (2025).
37. Su, J., Yang, M. & Tang, X. Integration of ShuffleNet V2 and YOLOv5s networks for a lightweight object detection model of electric bikes within elevators. *Electronics* **13**, 394 (2024).
38. Zhao, Y. et al. TGC-YOLOv5: an enhanced YOLOv5 drone detection model based on transformer, GAM & CA attention mechanism. *Drones* **7**, 446 (2023).
39. Yu, G. & Zhou, X. An improved YOLOv5 crack detection method combined with a bottleneck transformer. *Mathematics* **11**, 2377 (2023).
40. Wu, X., Ma, T., Zhao, Q., Zhu, L. & He, C. Research on YOLOv5s improved algorithm for pavement crack detection in complex environments. *IEEE Access* **12**, 122452–122461 (2024).
41. Yang, G. et al. AFPN: asymptotic feature pyramid network for object detection. In *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2184–2189 (2023).
42. Liu, Y. et al. MobileSAM-Track: lightweight one-shot tracking and segmentation of small objects on edge devices. *Remote Sens.* **15**, 5665 (2023).
43. Chen, B. & Dang, Z. Fast PCB defect detection method based on FasterNet backbone network and CBAM attention mechanism integrated with feature fusion module in improved YOLOv7. *IEEE Access* **11**, 95092–95103 (2023).
44. Chen, Z., He, Z. & Lu, Z.M. DEA-fasNet: single image dehazing based on detail-enhanced convolution and content-guided attention. *IEEE Trans. Image Process.* (2024).
45. Arya, D. et al. RDD2022: a multi-national image dataset for automatic road damage detection. *Geosci. Data J.* **11**, 846–862 (2024).
46. Shi, Y., Cui, L., Qi, Z., Meng, F. & Chen, Z. Automatic road crack detection using random structured forests. *IEEE Trans. Intell. Transp. Syst.* **17**, 3434–3445 (2016).
47. Zou, Q. et al. DeepCrack: learning hierarchical convolutional features for crack detection. *IEEE Trans. Image Process.* **28**, 1498–1512 (2019).

Acknowledgements

The authors would like to thank the editors and anonymous reviewers for their helpful suggestions.

Author contributions

Conceptualization, C.L. and X.S.; methodology, C.L. and J.L.; software, W.Y.; validation, X.S.; formal analysis, C.L.; investigation, Y.L.; data curation, W.Y.; writing—original draft preparation, C.L. and X.S.; writing—review and editing, C.L. and J.L.; funding acquisition, X.S. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding

This work was supported by the Guizhou Provincial Science and Technology Plan Project (with grant number [2024] General 165), Guizhou Vocational and Technical College of Industry 2023 School-Level Research Project (with grant number 2023ZK06).