Journal of Electronic Research and Application, 2026, Volume 10, Issue 5
http://ojs.bbwpublisher.com/index.php/JERA
Online ISSN: 2208-3510
Print ISSN: 2208-3502

PSR-DETR: An Improved RT-DETR for Rail Surface Defect Detection

Guangzhuo Li, Shijie Jia*

School of Rail Transportation Engineering, Dalian Jiaotong University, Dalian 116028, Liaoning, China

***Corresponding author:** Shijie Jia, jsj@djtu.edu.cn

Abstract: This paper addresses low detection accuracy in rail surface defect detection. The problem comes from many defect types, large scale changes, and small dense targets. Hence, an improved model based on RT-DETR is proposed namely PSR-DETR. The PR_BasicBlock module first simplifies the model structure. It reduces parameters and computation cost. Meanwhile, it maintains satisfactory detection performance. Consequently, the network becomes more lightweight. After that, the RetC3 module adds a new attention mechanism. It enhances feature integration. It also strengthens the model's capability to represent and distinguish targets of different scales. Finally, the SSFF module adds extra feature fusion paths. It helps the model emphasize critical regions. As a result, the detection performance is further improved. Experimental results show clear improvements, where the model does not greatly increase parameters or computation. The mAP@0.5 achieves 68.0%. The mAP@0.5:0.95 attains 44.7%, which are improvements of 6.3% and 2.7% over the original model. These findings show that the proposed method is effective and practical for enhancing detection performance.

Keywords: Object detection; RT-DETR; Attention mechanism; Surface defects

Online publication: Jun 29, 2026

1. Introduction

As of the end of 2024, the total railway network length in China reached 159,000 km. High-speed rail made up 45,000 km. The China-Europe Railway Express had operated over 80,000 trains. The expansion of railways supports transportation and economic growth [1]. However, rapid development also brings safety challenges. Train wheels and rails stay in long-term contact. Therefore, friction and impact often occur. At the same time, environmental effects speed up rail damage. These factors lead to surface defects such as wear, dents, and cracks. These defects reduce safety and service quality. Railway accidents can cause serious harm. They affect passengers, the public, and the environment [2]. Thus, it is important to promptly identify and locate rail defects. This helps improve maintenance efficiency, transport efficiency, and operational safety.

At present, numerous methods have been developed for rail surface defect detection. In addition to early manual inspection methods, several intelligent detection approaches have been proposed, including eddy current detection, magnetic flux leakage detection, ultrasonic testing, and vision-based detection methods. Among them, vision-based detection methods have attracted widespread attention in recent years due to their high efficiency and reliability. According to different processing strategies, these methods are generally categorized into conventional image processing approaches and deep learning-based detection techniques.

Traditional image processing methods are often integrated with machine learning techniques to enhance detection performance and efficiency to a certain extent. He *et al.* introduced the background subtraction method from video surveillance and constructed a background representation of rail surface images, proposing a defect detection approach for rail surfaces based on a background subtraction strategy [3]. Cao *et al.* developed a WFSOA algorithm to enhance image segmentation effectiveness and accurately determine threshold values [4]. This method optimizes the threshold searching process within the 2D-Otsu framework, thereby improving segmentation performance and real-time capability in rail defect images. Tastimur *et al.* introduced an approach built upon real-time video processing and morphological feature extraction, which can identify and detect various rail faults such as spalling, fractures, and scratches by applying image processing techniques and the Hough transform [5]. Gan *et al.* utilized a vision inspection system to examine railway surfaces and proposed a novel Background-Oriented Defect Inspector (BODI) [6]. This method obtains background representations through random sampling and combines multiple procedures for defect determination. Experimental results on real railway data demonstrate that this approach performs better than existing techniques in real-time applications. However, these methods usually include many steps. These steps involve image preprocessing, feature extraction, and classifier design, making the overall pipeline becomes complex. As a result, detection efficiency is often low in practice. This problem is more obvious in real-time scenarios.

Deep learning-based methods can learn deep and dataset-specific feature representations from large-scale samples. In comparison with conventional image processing approaches, these approaches show better generalization and robustness. Zhou *et al.* introduced an enhanced rail crack detection method constructed upon YOLOv5, where method reduces missed detections and improves detection accuracy for rail cracks [7]. However, this method is mainly suitable for crack detection. Yuan *et al.* proposed a fast object detection network named MOLO, which significantly improves the network running speed and ensures real-time detection performance, although its detection accuracy still has room for improvement [8]. Feng *et al.* developed two rail surface defect detection models with varying depths [9]. They used MobileNetV2 and MobileNetV3 architectures as backbone networks. These models improve detection speed and overall performance. Zhang *et al.* first used image enhancement to highlight defects [10]. This step also increases contrast between defects and background. Then, they applied an improved YOLOX model. The model uses feature fusion and attention mechanisms. The improved model increases mAP by 2.4%. However, the model size is large. Hence, it is not suitable for mobile deployment. Wu *et al.* studied the effect of surface reflection on detection accuracy [11]. They used image semantic augmentation and improved the YOLOv8 model. These methods improve detection accuracy. However, their work does not consider multi-class defect detection. It also does not include defect category classification.

At present, deep learning-based detection approaches are generally categorized into three categories as follows:

(1) Two-stage R-CNN series methods [12–16];
(2) One-stage YOLO series methods [17–21];
(3) DETR-based methods [22–25].

Among them, one-stage methods based on CNNs have strong real-time performance. However, these methods usually depend on Non-Maximum Suppression (NMS) for post-processing. This step can reduce performance when detecting small targets. Compared with CNNs, Transformer can model global relationships [26]. It uses multi-head self-attention to capture dependencies. Consequently, it can better describe complex relationships between objects [27].

The DETR combines CNN and Transformer structures [22]. It removes the NMS post-processing step, thereby being capable of achieving improved accuracy along with more stable performance for object detection tasks. However, DETR models have high computational cost. This limits their use in real applications. To solve this problem, RT-DETR was proposed by Zhao *et al.* [25]. It serves as a real-time, end-to-end detection architecture. It improves both training and inference speed. This is done by using a query-based detection method and an optimized decoupled Transformer decoder. In this framework, ResNet18 is used as the backbone. It extracts feature information from images. The Encoder performs multi-scale feature fusion. This helps improve contextual representation. The Decoder uses IoU-aware query selection. It also uses an end-to-end matching strategy without NMS. These designs make inference more efficient. Therefore, RT-DETR achieves an effective trade-off between computational efficiency and detection performance.

To address the issues associated with rail surface defect detection, including small and densely distributed targets, significant scale variations, and large network parameter size, this study adopts RT-DETR-r18 as the reference model and incorporates three modules: PR_BasicBlock, SSFF, and RetC3, proposing a novel detection algorithm named PSR-DETR. These improvements enable appropriate network lightweighting and better adaptation to rail surface defects with large scale variations, thereby enhancing detection performance.

2. Methodology

2.1. Overview of methodology

The overall framework is illustrated in **Figure 1**, where the parts enclosed by dashed boxes represent the improved components. The presented PSR-DETR network in this paper still consists of three primary modules: the backbone, encoder, and decoder along with an auxiliary detection branch as follows:

(1) In the backbone, the BasicBlock in the S3 stage is replaced by PR_BasicBlock. This change reduces model complexity and computation cost. At the same time, it keeps detection accuracy;
(2) In the Encoder, the first RepC3 is replaced by RetC3. This module includes an attention mechanism. It helps the model use spatial prior information, thereby improving the representation ability of the network;
(3) A Scale Sequence Feature Fusion (SSFF) block is added between the encoder and decoder. This operation enhances the model's capability to identify targets across multiple scales.

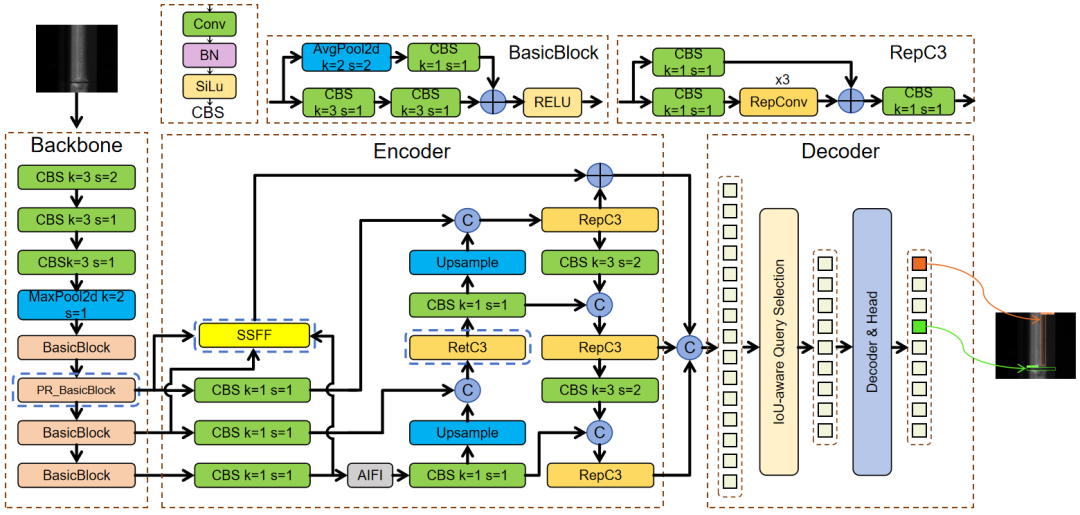


Figure 1. PSR-DETR model structure.

2.2. PR_BasicBlock

The PR_BasicBlock is designed by improving the original BasicBlock. It combines PConv (Partial Convolution) and RepConv (Re-parameterized Convolution) [28,29]. Specifically, a PR_Conv operation is built by merging these two methods. This new convolution replaces the original convolution parts in the BasicBlock. The final structure of PR_BasicBlock is shown in **Figure 2**.

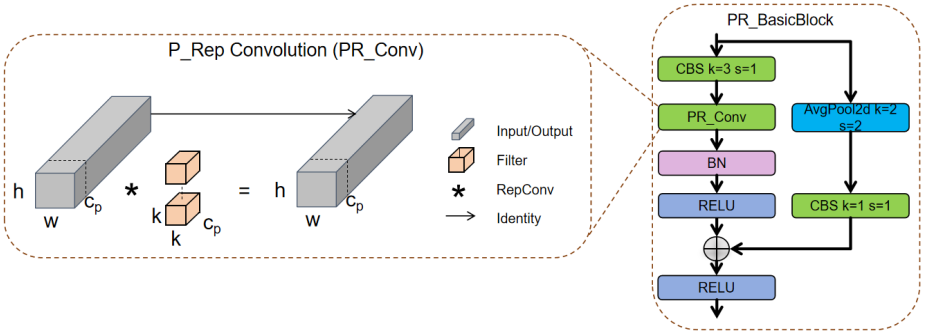


Figure 2. Structure of PR_BasicBlock.

Compared with the original BasicBlock, the CBS layer (Convolution-BatchNorm-SiLU) on the branch without the average pooling (AvgPool2d) path is replaced with PR_Conv. Meanwhile, following the design pattern of CBS ("convolution + normalization + activation"), the structure of PR_BasicBlock is constructed accordingly. In this design, one-quarter of the channels participate in convolution processing, and its output dimension of PR_Conv becomes 40 × 40 × 128.

The structure of RepConv is illustrated in **Figure 3**. The RepConv module contains a 3 × 3 convolution and a 1 × 1 convolution during the training stage, while during inference these branches are re-parameterized into an equivalent 3 × 3 convolution.

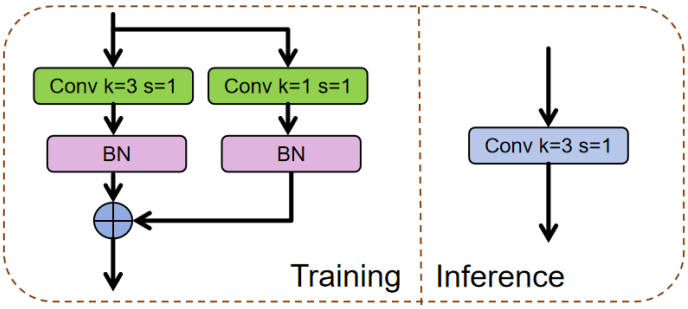


Figure 3. Structure of RepConv.

2.3. SSFF

RT-DETR uses a feature pyramid for multi-scale feature aggregation. In this framework, feature maps are combined by element-wise addition or concatenation. However, these operations cannot fully utilize the dependencies among different feature maps, limiting the effectiveness of the feature pyramid.

To solve this problem, the SSFF (Scale Sequence Feature Fusion) module is introduced [30]. This module helps combine multi-scale features more effectively. Specifically, SSFF merges high-level semantic information from deep layers together with fine-grained details from shallow layers. These feature maps share an identical aspect ratio.

The SSFF module fuses features from the S3, S4, and S5 layers of the backbone. It combines high-level semantic features extracted from deep layers together with detailed information from shallow layers. This block enhances feature representation by combining features from different depths. It also uses 3D convolution to extract cross-layer spatial features.

The architecture of SSFF is presented in **Figure 4**. First, the feature maps from S3, S4, and S5 pass through convolution layers. This step unifies the channel number to 256. After that, the S4 and S5 feature maps are upsampled. Upsampling is performed using the nearest-neighbor method. This makes the spatial resolution of these feature maps identical to that of S3. Next, an unsqueeze operation is applied to each feature map. This adds a new dimension. The 3D feature tensor (height, width, channels) becomes a 4D representation (depth, height, width, channels).

Then, these feature representations are concatenated along the depth axis. This forms a four-dimensional feature representation. Following that, 3D convolution is applied. It is subsequently followed by 3D batch normalization and an activation function. These steps facilitate the extraction of scale-sequence features. Finally, a Squeeze operation is used. It removes the extra dimension and restores a standard feature map.

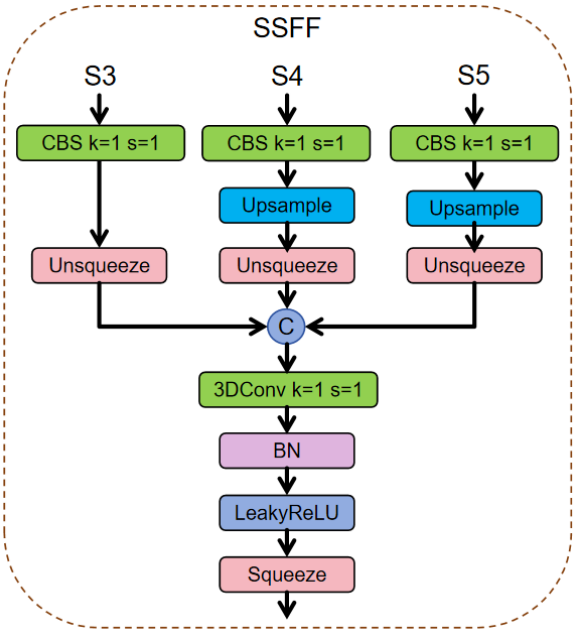


Figure 4. SSFF module structure.

2.4. RetC3

The architecture of the resulting RetC3 block is illustrated in **Figure 5**. The RetC3 module is obtained by replacing the RepConv in the original RepC3 module with the RetBlock proposed by Fan *et al.* [31].

The objective of this change is to incorporate the Manhattan Self-Attention (MaSA) mechanism into RetBlock. MaSA can use explicit spatial prior information. In contrast, the multi-head self-attention in RT-DETR is mainly content-driven. It does not include clear geometric priors. Therefore, combining these two mechanisms can bring complementary benefits. The model can use spatial prior information. It can also keep global interactions. As a result, the network can better handle complex scenes. Its representation and modeling ability are improved. This leads to better overall detection performance.

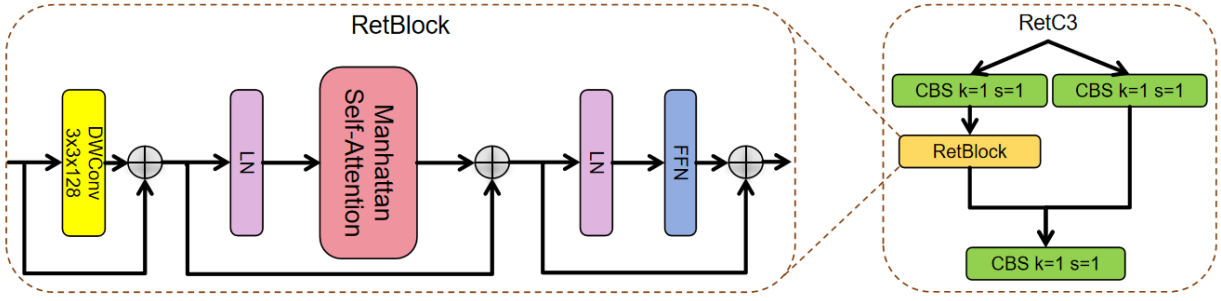


Figure 5. Structure of RetC3.

MaSA is a self-attention mechanism inspired by RetNet, which can be regarded as a two-dimensional extension of the ReSA mechanism in RetNet [32]. ReSA introduces an explicit temporal prior by applying a one-dimensional decay along the time dimension to control the weights between tokens. In contrast, MaSA introduces an explicit two-dimensional spatial decay matrix to control the weights between tokens. This

matrix is related to the Manhattan distance between tokens and is used to provide explicit spatial priors. In this work, the Decomposed Manhattan Self-Attention (DMaSA) approach is adopted. It decomposes the attention computation process together with the decay matrix into both horizontal and vertical directions, respectively computing corresponding attention weights in these two directions. The obtained attention weights are then applied to one-dimensional bidirectional decay matrices($D^H_{nm} = \gamma^{|y_n - y_m|}, D^W_{nm} = \gamma^{|x_n - x_m|}$),where γ denotes the decay factor, and a smaller value indicates faster attenuation with increasing distance. x_n and y_n denote the coordinates corresponding to the n-th token, while x_m together with y_m indicate the coordinate values of the m-th token. D^H_{nm} and D^W_{nm} denote the horizontal decay matrix and vertical decay matrix, respectively.

The decay matrices are incorporated into the attention computation process as shown in **Equation (1)**, **(2)** and **(3)**:

$$Attn_H = \text{Softmax}\left(Q_H K_H^T\right) \odot D^H \tag{1}$$

$$Attn_W = \text{Softmax}\left(Q_W K_W^T\right) \odot D^W \tag{2}$$

$$\text{MaSA}\left(X\right) = Attn_H \left(Attn_W V\right)^T \tag{3}$$

Where $Attn_H$ and $Attn_W$ represent the attention matrices along the horizontal and vertical axes, respectively.

The Softmax function transforms the computed weights into a probability distribution. Q_H and K_H denote the Query matrix and Key matrix in the horizontal direction, while Q_W and K_W denote the corresponding matrices in the vertical direction. The Query represents the element or position that needs to be attended to or processed, and the Key represents the element used for comparison with the Query. K^T_H and K^T_W denote the transpose of the corresponding Key matrices. The symbol $\odot$ represents the Hadamard product, indicating element-wise multiplication. D_H and D_W denote the horizontal decay matrix and vertical decay matrix, respectively, which are used to control the dependency relationships between different positions. V represents the Value matrix, corresponding to the values associated with the input features. MaSA(X) denotes the resulting output of the MaSA module when applied to the input feature X.

3. Experiment

3.1. Experimental methodology

Experimental evaluation was conducted on a Linux server. This platform is equipped with an Intel Core i9-12900K CPU and an NVIDIA GeForce RTX 4090 GPU with 24 GB memory. The network was developed with the PyTorch 2.4.1 framework. It was developed using Python 3.8.

No pretrained model was used during the experiments. The network was trained from scratch. The training parameters are shown in **Table 1**. During training, an early stopping mechanism was applied. When the validation performance failed to improve within a set number of epochs, training was stopped. This helps reduce the risk of overfitting.

Table 1. Training parameters

Parameter settings	Parameter value
Image size	640×640
Learning rate	0.0001
Epochs	200
Batch size	16
Weight decay	0.0001
Momentum	0.9

3.2. Dataset

The experiments in this study use a self-built dataset. It is based on two public datasets. These are the RSDD dataset from Beijing Jiaotong University and the Railway Asset dataset from the Dutch railway authority. The RSDD dataset was designed for semantic segmentation. The Dutch dataset does not include object detection labels. Thus, a new object detection dataset was built from these two datasets.

As illustrated in **Figure 6**, the dataset contains six categories of rail surface defects, including gap (Ga), crack (Cra), crush (Cru), spalling (Sp), dent (De), and scratch (Sc). The dataset contains 2,750 samples, which are randomly partitioned into the training, validation, and testing subsets with a proportion of 8:1:1.

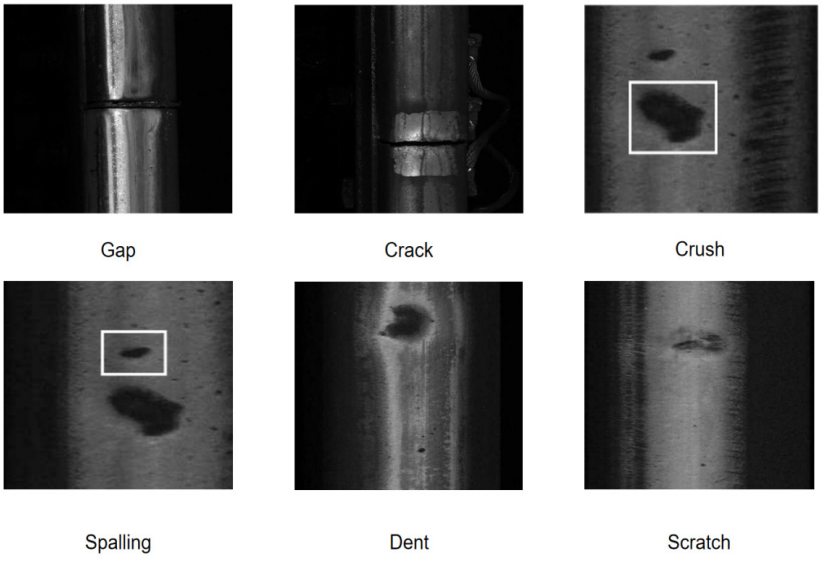


Figure 6. Example of dataset categories.

3.3. Evaluation criteria

To evaluate the improved algorithm, three metrics are used. These are Precision (P), Recall (R), and mean Average Precision (mAP). The corresponding expressions are presented as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

TP refers to samples that are correctly classified as positive. FP refers that are incorrectly classified as positive. FN refers to samples that are incorrectly classified as negative. mAP means mean average precision. It is computed as the mean of AP values across all categories. Each AP corresponds to one category. Then, all AP values are averaged to obtain mAP. The formula is shown below:

$$AP = \int_0^1 Precision\left(Recall\right) dRecall \tag{6}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{7}$$

3.4. Comparative experiments

To evaluate the performance of the proposed PSR-DETR for rail surface defect detection in comparison with the original RT-DETR-r18 model, the AP@0.5 values of six defect categories were compared on the testing dataset, with the corresponding results summarized in **Table 2**.

The results show that PSR-DETR improves AP@0.5 for six defect types. These include gap, crack, crush, spalling, dent, and scratch. The improvements are 2.1%, 5.3%, 8.3%, 7.4%, 8.6%, and 6.1%. These results show that PSR-DETR can detect rail surface defects accurately.

Table 2. Comparison of AP results of six defects between PSR-DETR and RT-DETR

Model	AP@0.5(%)					
	Ga	Cra	Cru	Sp	De	Sc
RT-DETR-r18	95.2	28.4	85	45	68.2	48.1
Ours	97.3	33.7	93.3	52.4	76.8	54.2

To further assess the performance advantages and effectiveness of the proposed PSR-DETR in rail surface defect detection, comparison experiments were conducted with several widely used object detection algorithms. The compared methods include GD-YOLOv8, PP-YOLOE+_s, RT-DETR-r18, YOLOv5m, YOLOv8m, YOLOv10b, YOLO11m, and YOLO12m.

All models use the same training, validation, and test sets. Each model is trained until convergence. This ensures a fair comparison. The corresponding results are summarized in **Table 3**.

Table 3. Comparison experiment results

Methods	P(%)	R(%)	mAP@0.5(%)	mAP@0.5:0.95(%)	Parameters/M	Flops/G
GD-YOLOv8 [33]	63.6	51.9	57.6	34.9	1.85	7.2
PP-YOLOE+_s [34]	72.9	61.5	64.6	40.4	7.93	17.36
RT-DETR-r18	74.8	56.4	61.7	42	19.88	57
YOLOv5m	69.4	63.1	66.7	41.9	25	64
YOLOv8m	69.7	62.8	66.4	42.2	25.8	78.7
YOLOv10b	71.4	61	65.7	38	20.4	98
YOLO11m	67.7	63.9	65.7	41.6	20	67.7
YOLO12m	67.8	62.7	66.1	41.7	20.1	67.1
Ours	78.8	61.9	68	44.7	19.54	56.8

As indicated in **Table 3**, the proposed PSR-DETR demonstrates performance gains of 10.4%, 3.4%, 6.3%, 1.3%, 1.6%, 2.3%, 2.3%, and 1.9% in mAP@0.5 compared with GD-YOLOv8, PP-YOLOE+_s, RT-DETR-r18, and several widely used YOLO-series models including YOLOv5m, YOLOv8m, YOLOv10b, YOLO11m, and YOLO12m, respectively.

3.5. Ablation experiment

To evaluate the impact of the proposed enhancements, ablation studies were performed using the dataset in **Section 3.1**. RT-DETR-r18 is used as the baseline model. Three modules are added to the network. The results show the gain of each module and the combined effect of all modules. The corresponding results are summarized in **Table 4**.

Table 4. Ablation experiment results

PR_BasicBlock	RetC3	SSFF	mAP@0.5(%)	mAP@0.5:0.95(%)	Parameters/M	Flops /G
			61.7	42	19.88	57
√			64.9	43.5	14	42.9
	√		64.5	43.4	18.52	50.2
		√	66.8	44.5	20.15	61.5
√	√		65.7	43.2	19.26	52.4
	√	√	67.3	44.4	19.82	60.4
√	√	√	68	44.7	19.54	56.8

As shown in **Table 4**, each module improves model performance when used alone. PR_BasicBlock, RetC3, and SSFF all have positive effects. After adding PR_BasicBlock, mAP@0.5 rises from 61.7% to 64.9%. The mAP@0.5:0.95 attains 43.5%. Meanwhile, the parameter count decreases to 14M. The computational cost is reduced to 42.9 GFLOPs. These results show a good balance between lightweight design and accuracy.

The RetC3 module also improves detection performance at a similar scale. The mAP@0.5 increases to 64.5%. The mAP@0.5:0.95 attains 43.4%. The parameter count reaches 18.52M, while the computational cost amounts to 50.2 GFLOPs. Among the three modules, SSFF gives the largest improvement. The mAP@0.5 rises to 66.8%. The mAP@0.5:0.95 attains 44.5%. However, this gain comes with higher cost. The parameters increase to 20.15M, and the computation rises to 61.5 GFLOPs. This shows that SSFF greatly enhances the feature representation capability of the network.

In combined experiments, using multiple modules further improves performance. When PR_BasicBlock and RetC3 are used together, mAP@0.5 increases to 65.7%. The mAP@0.5:0.95 attains 43.2%. The model has 19.26M parameters and 52.4 GFLOPs. This gives a better balance than using a single module. When RetC3 and SSFF are combined, mAP@0.5 reaches 67.3%. The mAP@0.5:0.95 attains 44.4%. The model has 19.82M parameters and 60.4 GFLOPs. These results show stronger combined effects.

Finally, when all three modules are used together, the best results are obtained. The mAP@0.5 attains 68.0%. The mAP@0.5:0.95 reaches 44.7%. These correspond to improvements of 6.3% and 2.7% over the baseline. Meanwhile, the overall model size and computational complexity remain controlled. The parameters are 19.54M, and the cost is 56.8 GFLOPs. These results show that combining multiple modules

provides an effective trade-off between accuracy and computational efficiency.

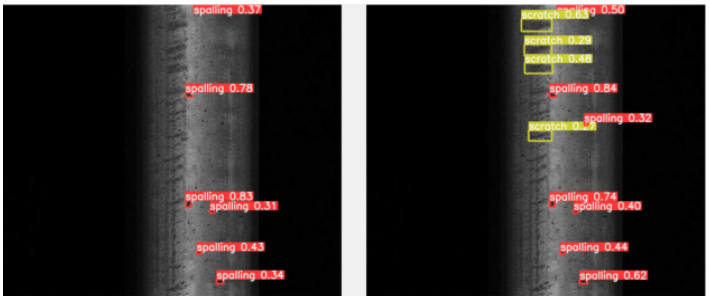
3.6. Visual analysis

To show the performance of the improved network, visual comparison experiments are conducted. The RT-DETR-r18 baseline and the proposed method are tested under identical conditions and parameters. The results are shown in **Figure 7**. In each image, the left side presents the detection outcomes of the RT-DETR-r18 baseline model, and the right side shows those of the enhanced model.

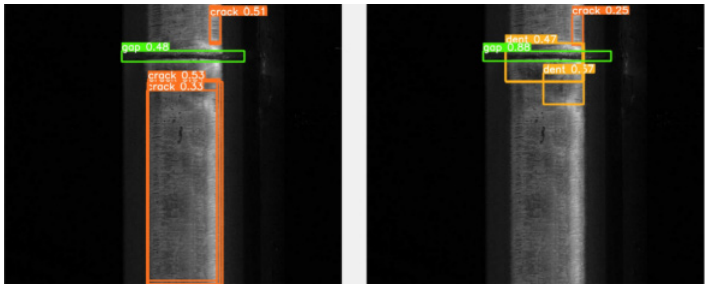
As illustrated in **Figure 7(a)** and **Figure 7(c)**, the baseline model misses some targets. These include spalling, dent, and scratch. Duplicate bounding boxes also appear in **Figure 7(c)**. In contrast, **Figure 7(b)** and **Figure 7(d)** show better results. The improved model reduces missed detections and false detections. It also removes duplicate boxes. At the same time, the confidence scores are higher.

In **Figure 7(e)** and **Figure 7(g)**, the baseline model makes wrong predictions. It classifies gap and scratch as spalling. In **Figure 7(e)**, gap is also classified as crack. In contrast, **Figure 7(f)** and **Figure 7(h)** show better results. The improved model corrects these errors. It also gives higher confidence for the detections.

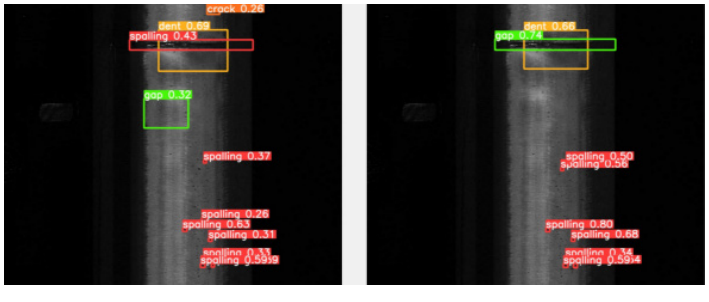
The visualization results further demonstrate the performance advantages of the proposed method in enhancing detection performance and robustness.



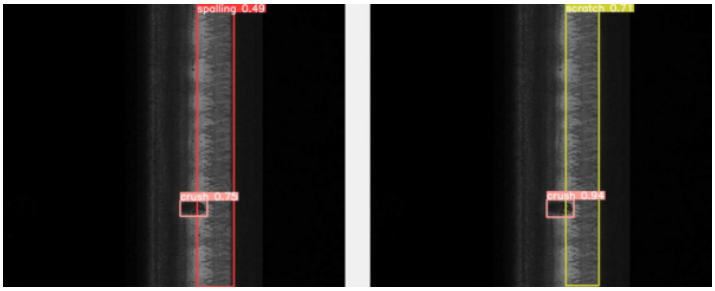
(a) R18 (b) Ours



(c) R18 (d) Ours



(e) R18 (f) Ours



(g) R18 (h) Ours

Figure 7. Visual comparison between RT-DETR and PSR-DETR.

4. Conclusion

To solve low detection accuracy and poor localization in rail images, a method named PSR-DETR is proposed. The problem comes from many small targets and hard background conditions. In this method, Partial Convolution replaces standard convolution. It uses channel redundancy and only processes part of the features. This reduces memory access and computation. Therefore, the model becomes more efficient. In addition, the DMaSA mechanism is added. It enhances the model's capability to capture spatial structures as well as defect regions. Furthermore, the SSFF module is introduced. It fuses features from different scales. This improves the generalization capability and detection performance of the network. Experimental findings indicate that PSR-DETR improves overall performance in defect detection tasks. Meanwhile, the model remains lightweight. Compared with the baseline, the proposed approach provides a more favorable trade-off between accuracy and efficiency. It shows good potential for practical use. In future work, the focus will be on improving robustness under extreme conditions. More efficient deployment methods will also be studied. The goal is to support real-world railway inspection applications.

Disclosure statement

The author declares no conflict of interest.

References

[1] Yang F, Tu W, Wei Z, et al., 2023, Review on the Development Of Railway Civil, Electrical and Power Inspection Equipment. Journal of Traffic and Transportation Engineering, 23(1): 47–69.

[2] Gong W, Akbar M, Jawad G, et al., 2022, Nondestructive Testing Technologies for Rail Inspection: A Review. Coatings, 12(11): 1790.

[3] He Z, Wang Y, Liu J, et al., 2016, High-Speed Rail Surface Defect Image Segmentation Based On Background Subtraction. Chinese Journal of Scientific Instrument, 37(3): 640–649 .

[4] Cao Y, Duan Y, Wu D, 2020, 2D-Otsu Rail Defect Image Segmentation based on WFSOA. Computer Science, 47(5): 154–160 .

[5] Taştimur C, Karaköse M, Akın E, et al., 2016, Rail Defect Detection with Real-Time Image Processing Technique, 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), 411–415.

[6] Gan J, Wang J, Yu H, et al., 2018, Online Rail Surface Inspection Utilizing Spatial Consistency and Continuity.

IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(7): 2741–2751.

[7] Zhou M, Tang Q, Shi T, et al., 2023, Rail Surface Crack Detection Algorithm based on Improved Yolov5s. Liquid Crystals & Displays, 38(5): 666–679.

[8] Yuan H, Chen H, Liu S, et al., 2019, A Deep Convolutional Neural Network For Detection Of Rail Surface Defect, 2019 IEEE Vehicle Power and Propulsion Conference (VPPC), 1–4.

[9] Feng J, Yuan H, Hu Y, et al., 2020, Research on Deep Learning Method For Rail Surface Defect Detection. IET Electrical Systems in Transportation, 10(4): 436–442.

[10] Zhang C, Xu D, Zhang L, et al., 2023, Rail Surface Defect Detection based on Image Enhancement and Improved YOLOX. Electronics, 12(12): 2672.

[11] Wu Y, Cui C, He Y, 2024, Rail Surface Defect Detection Method based on Semantic Augmentation and Yolov8. Journal of Railway Science and Engineering, 21(1): 1–12.

[12] Girshick R, Donahue J, Darrell T, et al., 2014, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 580–587.

[13] Girshick R, 2015, Fast R-CNN, Proceedings of the IEEE International Conference on Computer Vision, 1440–1448.

[14] He K, Gkioxari G, Dollár P, et al., 2017, Mask R-CNN, Proceedings of the IEEE International Conference on Computer Vision, 2961–2969.

[15] Xu Y, Yu G, Wang Y, et al., 2017, Car Detection from Low-Altitude UAV Imagery with the Faster R-CNN. Journal of Advanced Transportation, 2017: 2823617.

[16] Avola D, Cinque L, Diko A, et al., 2021, MS-Faster R-CNN: Multi-Stream Backbone for Improved Faster R-CNN Object Detection and Aerial Tracking from UAV Images. Remote Sensing, 2021(13): 1670.

[17] Redmon J, Farhadi A, 2018, YOLOv3: An Incremental Improvement. arXiv, arXiv:1804.02767.

[18] Bochkovskiy A, Wang C, Liao H, 2020, Yolov4: Optimal Speed and Accuracy of Object Detection, arXiv, arXiv:2004.10934.

[19] Jocher G, Chaurasia A, Stoken A, et al., 2022, Ultralytics YOLOv5, Zenodo, viewed August 4, 2024, https://github.com/ultralytics/yolov5

[20] Wang C, Bochkovskiy A, Liao H, 2023, Yolov7: Trainable Bag-of-Freebies Sets New State-of-The-Art for Real-Time Object Detectors, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7464–7475.

[21] Jocher G, Chaurasia A, Qiu J, 2023, Ultralytics YOLOv8, viewed July 18, 2025, https://github.com/ultralytics/ultralytics

[22] Carion N, Massa F, Synnaeve G, et al., 2020, End-to-End Object Detection with Transformers, Proceedings of the European Conference on Computer Vision, 213–229.

[23] Zhu X, Su W, Lu L, et al., 2020, Deformable DETR: Deformable Transformers for End-to-End Object Detection, arXiv, arXiv:2010.04159.

[24] Zhang H, Li F, Liu S, et al., 2022, DINO: DETR with Improved Denoising Anchor Boxes for End-to-End Object Detection, arXiv, arXiv:2203.03605.

[25] Zhao Y, Lv W, Xu S, et al., 2023, Detrs Beat Yolos on Real-Time Object Detection, arXiv, arXiv:2304.08069.

[26] Vaswani A, Shazeer N, Parmar N, et al., 2017, Attention is All you Need, Advances in Neural Information Processing Systems.

[27] Liu Z, Lin Y, Cao Y, et al., 2021, Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows, Proceedings of the IEEE/CVF International Conference on Computer Vision, 10012–10022.

[28] Chen J, Kao S, He H, et al., 2023, Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 12021–12031.

[29] Ding X, Zhang X, Ma N, et al., 2021, Repvgg: Making VGG-Style Convnets Great Again, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 13733–13742.

[30] Kang M, Ting M, Ting F, et al., 2024, ASF-YOLO: A Novel YOLO Model with Attentional Scale Sequence Fusion for Cell Instance Segmentation. Image and Vision Computing, 2024(147): 105057.

[31] Fan Q, Huang H, Chen M, et al., 2024, RMT: Retentive Networks Meet Vision Transformers, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5641–5651.

[32] Sun Y, Dong L, Huang S, et al., 2023, Retentive Network: A Successor to Transformer for Large Language Models, arXiv, arXiv:2307.08621.

[33] Yang X, Li Y, Li Y, et al., 2025, Lightweight Rail Surface Defect Detection Algorithm based on an Improved Yolov8. Measurement, 2025(242): 115922.

[34] Baidu PaddlePaddle Team, 2022, PP-YOLOE Object Detection Model, viewed June 8, 2025, https://github.com/PaddlePaddle/PaddleDetection