Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

EADI-YOLO: a lightweight and efficient model for rail surface defect detection

Siwei Ma
Dalian Jiaotong University
Ronghua Li
lironghua705@163.com
Dalian Jiaotong University
Henan Hu
Dalian Jiaotong University

Research Article

Keywords: Rail surface defect detection, EADI-YOLO, Attention mechanism, Lightweight convolution, Dynamic sampling, Bounding box regression

Posted Date: August 1st, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7183947/v1

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

Additional Declarations: No competing interests reported.

EADI-YOLO: a lightweight and efficient model for rail surface defect detection

Siwei Ma [1] ·Ronghua Li [2*] ·Henan Hu [1]

Abstract

As the complexity of railway transportation systems increases, higher demands have been placed on the real-time performance and accuracy of track defect detection, making traditional methods inadequate for intelligent inspection tasks. To address this, this paper proposes a lightweight and efficient improved detection model, EADI-YOLO, based on the YOLOv10n framework, aiming to achieve the optimal balance between detection accuracy, inference speed, and computational cost. The model integrates the C2f-EMAN module into the backbone, combining EMA and NAM attention mechanisms to enhance the representation of key regional features. It adopts the efficient AKConv module to replace standard convolutions, improving multi-scale defect perception while reducing parameter overhead. Furthermore, a DySample dynamic sampling module is introduced in the upsampling stage to better reconstruct feature details and preserve semantic information. An Inner-GIoU regression loss function is designed to optimize bounding box localization accuracy.Experimental results on rail surface defect detection tasks demonstrate the effectiveness of the proposed model. EADI-YOLO achieves a mAP@0.5 of 90.3% and a mAP@0.5:0.95 of 49.1%, with an inference speed of 120 FPS, representing improvements of 2.8%, 6.7%, and 15 FPS over YOLOv10n, respectively. Compared to other mainstream detection models such as RT-DETR and YOLOv13n, EADI-YOLO demonstrates superior detection accuracy while maintaining excellent lightweight and real-time performance. Generalization experiments show that EADI-YOLO maintains an excellent balance between accuracy, speed, and model complexity across different rail defect datasets, making it suitable for intelligent detection of railway track defects.

Keywords Rail surface defect detection; EADI-YOLO; Attention mechanism; Lightweight convolution; Dynamic sampling; Bounding box regression

1 Introduction

Railway track infrastructure plays a crucial role in ensuring the safety, efficiency, and reliability of modern railway transportation systems [1]. However, over long-term operation, the cumulative effects of wheel-rail interactions, environmental factors, and material fatigue can lead to the development of various surface defects on the rail, such as cracks, spalling, corrugation, and wear. If these defects are not detected and repaired in a timely manner, they can seriously compromise the structural integrity of the track, resulting in increased maintenance costs, operational delays, and even major safety incidents [2].

Traditional rail surface inspection primarily relies on manual visual inspection [3] and instrument-based detection [4].While manual inspection has certain effectiveness, it suffers from high labor intensity, strong subjectivity, and poor consistency. Although instrument-based inspection offers higher precision, it commonly faces limitations such as insufficient spatial resolution, high costs, long inspection cycles, and strict operational requirements. Moreover, these traditional methods have limited ability to detect early-stage and minor defects, which can lead to defect accumulation and deterioration, making it difficult to provide timely and effective risk warnings. This hinders the transformation of current rail maintenance toward predictive, refined, and intelligent inspection strategies [5]. Therefore, it is imperative to introduce advanced detection technologies to improve inspection efficiency and defect identification accuracy, enabling full lifecycle management of rail infrastructure.

In recent years, the rapid development of computer vision and deep learning technologies[6] has provided new breakthroughs for the automated detection of rail surface defects . In particular, methods based on the YOLO (You Only Look Once) series [7], U-Net architecture [8], and Transformer-based models [9] have demonstrated outstanding performance in tasks such as defect classification, localization, and segmentation under complex visual environments. However, these approaches still face challenges in practical applications, especially in balancing the trade-off between meeting real-time detection requirements and maintaining high detection accuracy and computational efficiency, leaving considerable room for further optimization.

To address the above challenges, this paper proposes a deep learning model called EADI-YOLO(C2f-EMAN-AKConv-DySample-Inner-GIoU-You Only Look Once) for rail surface defect detection. By integrating advanced attention mechanisms and lightweight neural network architectures, the proposed EADI-YOLO model aims to achieve a dual optimization of high detection accuracy and real-time inference speed.

(1) To extract richer semantic features for defect identification, we integrate the EMAN attention

✉ Ronghua Li

lironghua705@163.com

Siwei Ma

ma_siwei2021@163.com

Henan Hu

huhenan18@mails.ucas.ac.cn

1 School of Mechanical Engineering, Dalian Jiaotong University, Dalian 116028, China

2 School of Electrical Engineering, Dalian Jiaotong University, Dalian 116045, China

mechanism into the backbone's C2f module, yielding a redesigned C2f-EMAN. C2f-EMAN captures spatial dependencies via its EMA submodule (Efficient Multi-Scale Attention) and enhances channel-level saliency through its NAM submodule (Normalization-based Attention Module). By dynamically adjusting the fusion ratio between these two branches with a learnable parameter, C2f-EMAN adaptively balances global and local feature representations—thereby accentuating defect regions while suppressing background noise.

(2) To enhance adaptability to varying defect shapes and scales, the model introduces an AKConv (Alterable Kernel Convolution) module as a replacement for traditional convolution. AKConv uses a learnable offset mechanism to deform the sampling grid based on local feature structures, enabling dynamic receptive field adjustment. This empowers the model to better extract detailed structural cues from irregular and small defect areas, while reducing redundant parameters and preserving computational efficiency.

(3) To improve upsampling quality and semantic preservation, a DySample (Dynamic Sampling) module is employed in place of standard nearest-neighbor interpolation. DySample generates a content-aware point sampling set using a dynamic range factor, allowing for precise alignment with semantic structures. This method ensures better reconstruction of fine details in upsampled feature maps, thereby enhancing the model's detection performance on subtle or densely distributed defects.

(4) To improve localization accuracy and the generalization of regression, the designed Inner-GIoU loss function is introduced. Inner-GIoU constructs auxiliary bounding boxes to calculate the overlap in the inner region, ensuring effective gradient flow even in the absence of overlap. This design enhances localization accuracy and improves the detection capability for small defects.

2 Related works

In recent years, with the rapid development of computer vision and intelligent algorithms, vision-based rail surface defect detection methods have been extensively studied. These existing approaches can generally be categorized into two types: traditional image processing methods and deep learning-based methods. Although significant progress has been made in terms of detection accuracy, challenges remain in inference speed and model complexity.

Traditional image processing methods are lightweight but lack robustness. Liu et al. [10] proposed a method based on visual saliency and the Canny edge detector, which effectively extracts defect contours. However, this method is sensitive to noise and prone to introducing redundant edge information, leading to increased computation and a higher risk of false detections. Ni et al. [11] proposed a method combining region partitioning and fuzzy decision-making, which performs well in complex backgrounds. Nevertheless, it relies on multiple stages—such as region segmentation, contour restoration, and morphological filling—making the pipeline complex and inference efficiency low.

Deep learning-based vision detection methods offer high accuracy but tend to be computationally intensive. Yang et al. [12] proposed a pixel-level defect segmentation method that enhances texture perception through channel feature embedding and weak-correlation dropout. However, the method introduces a large number of high-dimensional features, significantly increasing model size and computational cost. Alvarenga et al. [13] designed an embedded system combining eddy current testing with CNNs to model wavelet features and efficiently classify rail defects. Yet, the feature extraction and CNN architecture are complex, resulting in a large model size and slow inference. Niu et al. [14] developed a GAN-based model with a pyramid graph and residual mechanisms, which improves anomaly discrimination but increases parameter count and inference latency due to multiple loss functions. Zhang et al. [15] enhanced YOLOv3 by integrating DenseNet modules in CP-YOLOv3-dense to improve multi-layer feature reuse, achieving better detection accuracy. However, dense connections lead to significant growth in parameter count and computational overhead, reducing inference speed. Si et al. [16] proposed Rail-STrans based on the Swin Transformer, incorporating several feature enhancement modules to improve small-scale defect detection. Nonetheless, the Transformer-based architecture and complex modules result in a bulky model with high computational demands.

In summary, traditional visual methods are lightweight but lack robustness, while deep learning methods offer high precision but suffer from computational complexity. Existing approaches struggle to balance accuracy and real-time performance, limiting their practical applications in rail defect detection.

To address these limitations, this paper proposes an improved detection algorithm named EADI-YOLO, based on the YOLOv10n framework. The algorithm integrates the C2f-EMAN attention mechanism, AKConv efficient convolution, DySample adaptive sampling module, and Inner-GIoU loss function. This design improves detection accuracy while effectively controlling parameter count and computational cost, achieving a balance between precision and speed, and offering stronger detection capabilities.

3. Proposed method

To enable more accurate and real-time detection of rail surface defects, this paper presents an improved lightweight detection model named EADI-YOLO, built upon the YOLOv10n[17] framework. First, the EMAN attention mechanism is integrated into the backbone's C2f module to form C2f-EMAN. This module first uses the EMA submodule to capture spatial dependencies and then leverages the NAM submodule to enhance channel-wise saliency, enabling suppression of redundant background noise and amplification of critical defect regions, thereby improving detection accuracy.Secondly, the model replaces standard convolution layers with the AKConv module, a deformable convolutional structure capable of dynamically adjusting sampling positions. By learning spatial offsets, AKConv enables the convolution kernel to better adapt to irregular defect shapes and sizes, thereby enhancing the model's capability to extract critical structural features while reducing redundant parameters and computational load.Thirdly, a DySample upsampling module is employed

to replace traditional nearest-neighbor interpolation. DySample leverages content-aware point sampling and offset mechanisms to reconstruct high-resolution feature maps. This strategy facilitates the preservation of fine-grained visual information during upsampling, improving the model's semantic perception and feature fusion efficiency in rail surface defect detection tasks. Finally, to improve bounding box regression accuracy, the Inner-GIoU loss function is proposed. This loss function ensures effective gradient propagation by introducing auxiliary boxes, leading to more stable training and more accurate localization performance. This improvement significantly enhances the model's ability to quickly locate defects. The structure of EADI-YOLO is shown in Fig. 1.

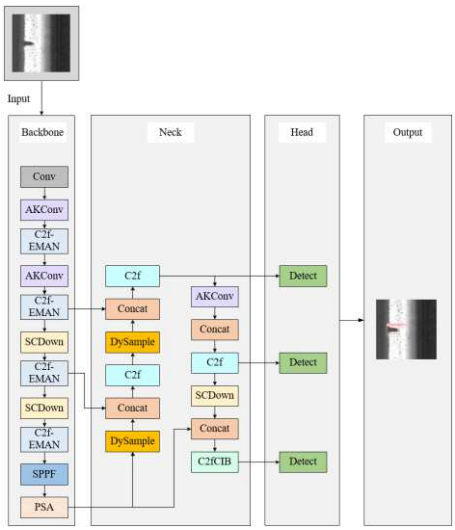


Fig.1 EADI-YOLO structure

3.1 C2f-EMAN module

To extract more comprehensive defect features, we integrate the EMAN attention mechanism (EMA–NAM attention) into the backbone's C2f module, creating C2f-EMAN. This module combines the strengths of EMA, which captures spatial relationships, and NAM, which emphasizes channel relationships. This design enables the model to fully leverage both channel-wise and spatial information, capturing dependencies across different positions and channels in the image. C2f-EMAN not only enhances global and local feature representations but also highlights salient target regions and suppresses redundant background noise, allowing the model to focus on critical details and prominent features. The structure of the C2f-EMAN module is shown in Fig. 2.

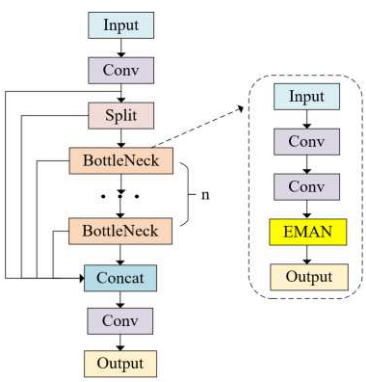


Fig.2 C2f-EMAN module

EMAN adopts a weighted fusion mechanism (with the α parameter controlling the weight ratio between the outputs of EMA attention and NAM attention), enabling the model to adaptively adjust the contributions of the two attention mechanisms according to the requirements of different tasks. This design allows the model to demonstrate excellent feature extraction capabilities across various application scenarios. The structure of EMAN is shown in Fig. 3.

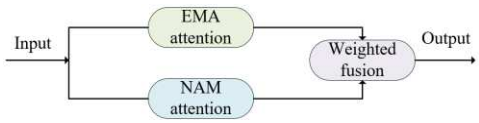


Fig.3 EMAN attention mechanism

The weighted fusion formula of the EMAN module is shown in Equation (1).

$$Weightedfusion = \alpha \cdot EMA + (1-\alpha)NAM \quad (1)$$

Here, α is the weighting coefficient, with a value range of $0<\alpha<1$, used to adjust the contribution proportion of the two in the fusion process.

EMA attention[18](Efficient Multi-Scale Attention) extracts features at different scales by grouping the input features and processes them using adaptive average pooling and convolution, effectively combining global and local information.

First, the EMA mechanism uses a feature grouping module to divide the input feature map $X \in R^{C\times H\times W}$ into multiple subgroups along the channel dimension, where H and W represent the height and width of the feature map, and C denotes the number of channels. By splitting the features into G subgroups, the model is able to learn multi-scale feature information and enhance the representation of each subgroup through reallocated attention weights.

Next, multi-scale feature information is processed through parallel submodules using three parallel structures for encoding. Two 1×1 parallel branches encode spatial information along the vertical and horizontal directions, corresponding to Equations (2) and (3), and their encoded features are concatenated along the height dimension. The third parallel branch uses a 3×3 convolution to capture extended feature information. Together, the three branches retain spatial information within the channels to capture multi-scale feature representations. The Sigmoid function decomposes the features into two directional feature vectors with dimensions $C//G\times 1\times W$ and $C//G\times H\times 1$, further enhancing the representation of multi-scale features.

$$z_c^H(H) = \frac{1}{W}\sum_{0\le i\le W} x_c(H,i) \quad (2)$$

$$z_c^W(W) = \frac{1}{H}\sum_{0\le j\le W} x_c(H,j) \quad (3)$$

Here, i represents the horizontal index, corresponding to the position along the width W dimension of the input tensor x; while j represents the vertical index, corresponding to the position along the height H dimension of the input tensor x.

Finally, to integrate information from different spatial regions, the model uses a cross-domain learning module to aggregate the feature information obtained from the 3×3 and 1×1 branches. It then applies 2D global average pooling to encode the spatial information from these two branches, as shown in Equation (4). After encoding, two spatial attention factors are generated, and their attention weights are applied through weighting, followed by activation using the

Sigmoid function to produce the output feature map. The cross-domain information aggregation not only enables more precise acquisition of target features in the image and reduces interference, but also handles both short-range and long-range dependencies, capturing richer contextual information.Fig. 4 illustrates the EMA attention mechanism.

$$z_c = \frac{1}{H \times W} \sum_{j}^{H} \sum_{i}^{W} x_c(i, j) \tag{4}$$

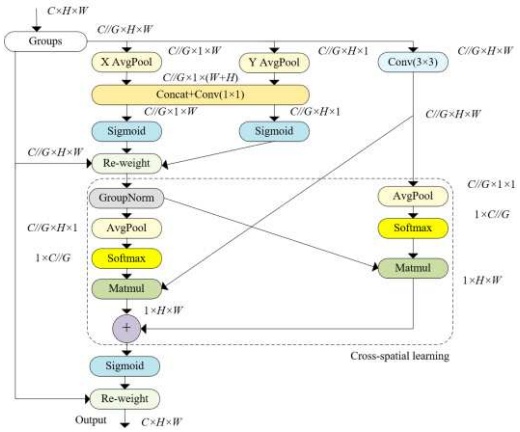


Fig.4 EMA attention mechanism

NAM[19](Normalization-based Attention Module) employs a weight sparsity penalty mechanism to reduce the weights of non-salient features and enhance the representation of salient features, while redesigning the spatial and channel attention modules based on CBAM. In particular, the channel attention module leverages the scale and standard deviation factors from batch normalization (BN) to reflect the importance of weights, thereby avoiding the need to introduce additional fully connected or convolutional layers, as shown in Equation (5).

$$B_{out} = BN(B_{in}) = \gamma \frac{B_{in} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} + \beta \tag{5}$$

Here, B_{in} represents an element in the input batch B, B_{out} is the normalized output value, μ_B is the mean of B, σ_B^2 is its variance, and γ and β are the scaling and shifting parameters in the affine transformation, respectively. ε is a small value used to prevent division by zero.

The structure of the channel attention submodule is shown in Fig. 5a, where *Mc* denotes the output feature, and γ_i represents the scaling factor for channel *i*. The formula for calculating the channel weights based on the BN scaling factor is as follows:

$$W_i = \frac{\gamma_i}{\sum_{j=0} \gamma_j} \tag{6}$$

The corresponding spatial attention submodule is shown in Fig. 5b, where *Ms* represents the output features and λi denotes the scaling factor for channel *i*. The BN scaling factor is used to assess the importance of each pixel, a process referred to as pixel normalization, and the formula for calculating the weights is as follows.

$$W_i = \frac{\lambda_i}{\sum_{j=0} \lambda_j} \tag{7}$$

To enhance the weights of salient features and suppress the influence of non-salient features, a regularization term is introduced into the loss function, as shown in Equations (8)–(10).

$$Mc = sigmoid(W_\gamma(BN(F_1))) \tag{8}$$

$$M_S = sigmoid(W_\lambda(BN(F_2))) \tag{9}$$

$$Loss = \sum_{(x,y)} l(f(x, W), y) + p \sum_{(x,y)} g(\gamma) + p \sum_{(x,y)} g(\lambda) \tag{10}$$

Here, F_1 refers to the feature map input to the channel attention module, and F_2 refers to the feature map input to the spatial attention module. The sigmoid function serves as the activation function; x is the input of the salient target, and y is the feature output of the salient target. $W\gamma$ and W_λ are the weight matrices for the channel and spatial attention, respectively. Loss represents the loss function, g is the regularization function used to constrain the complexity of the attention parameters γ and λ, and p is the penalty coefficient that balances $g(\gamma)$ and $g(\lambda)$.

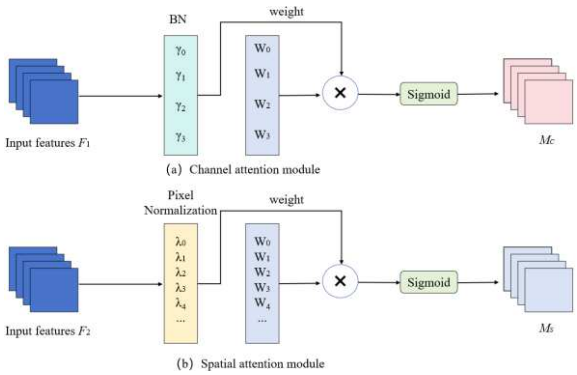


Fig.5 NAM attention mechanism

3.2 AKConv module

AKConv[20] (Alterable Kernel Convolution) is a type of deformable convolution that can flexibly adapt to targets of various sizes and shapes. Replacing the Conv module with AKConv allows the model to extract features more accurately and efficiently when dealing with defects of different scales and forms, not only enhancing the model's performance and structural detection capability but also effectively reducing the number of parameters and computational cost.

The core idea of AKConv is to use the offsets learned by the module to dynamically adjust the original sampling positions, allowing the convolution kernel to adapt its shape based on the image features. After undergoing this adaptive sampling, the feature map is processed through resampling, reshaping, convolution, and normalization, and finally produces the output via the SiLU activation function.

The input image has dimensions of $C \times H \times W$, where C is the number of channels, and H and W represent the height and width of the image, respectively. AKConv determines the initial sampling positions of the convolution kernel through a coordinate generation algorithm. These positions are not fixed but can be dynamically adjusted based on the extracted image features and target, which is a key advantage of AKConv. By applying a Conv2d operation to the input image, the corresponding kernel offsets can be obtained with dimensions $(B,2N,H,W)$, where N represents the kernel size.The overall network structure of the AKConv module is shown in Fig. 6.

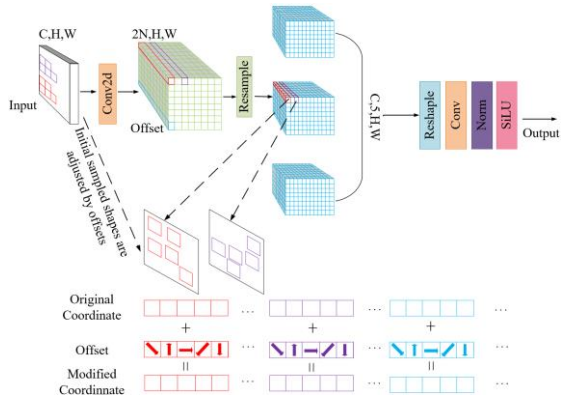


Fig.6 AKConv structure

3.3 DySample module

YOLOv10n uses nearest-neighbor interpolation (Upsample) for upsampling, where the gray value of each transformed pixel is directly assigned from the closest input pixel, thereby achieving image enlargement. However, this method relies solely on the spatial positions of pixels and ignores the semantic information within the feature map. As a result, when processing images with complex textures or rich details, it often leads to blurred or distorted details, making it difficult to accurately reconstruct the fine features of the original image. To address this issue, an efficient dynamic upsampler called DySample is introduced.

DySample[21] (Dynamic Sample) incorporates an offset mechanism that dynamically adjusts the sampling positions based on the content of the input feature map, enabling position-aware upsampling and allowing for more precise preservation and reconstruction of fine details, thereby significantly improving model performance. Compared with traditional upsampling methods, DySample adopts a point-based sampling strategy, which does not require additional high-resolution feature inputs and can generate content-aware upsampling results in a simple and efficient manner. The upsampling process of DySample is illustrated in Fig. 7.

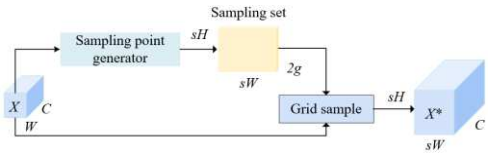


Fig.7 DySample structure

In the module structure of the DySample upsampler, the input includes a feature map X with dimensions $C\times H\times W$ and a point sampling set with dimensions $2g\times sH\times sW$, where the $2g$ channels represent the x and y coordinates in the feature space. By calling the grid sample function and combining it with the spatial information from the point sampling set δ, X is resampled to generate a feature map X^* with the same dimensions as the point sampling set. The formula is shown in Equation (11).

$$X^* = gridsample(x,\delta) \tag{11}$$

The generation process of the point sampling set δ based on the dynamic range factor is illustrated in Fig. 8. After determining the upsampling ratio factor s, for the feature map X with dimensions $C\times H\times W$, a linear layer is used to map the C channels to a $2gs^2$ dimensional space, generating the offset O, as shown in Equation (12).

$$O = linear(X) \tag{12}$$

By transforming the offset O in the pixel space, it is converted into a high-resolution reference sampling grid G with dimensions $2g\times sH\times sW$, and by adding them together, the final point sampling set δ is obtained, as shown in Equation (13).

$$\delta = O + G \tag{13}$$

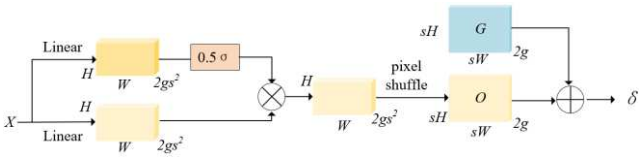


Fig.8 The point sampling set generation process based on the dynamic range factor

3.4 Inner-GIoU loss function

In the field of object detection, the design of bounding box regression loss functions plays a critical role in determining model performance. However, under conditions of complex backgrounds and noise interference, the CIoU loss function used in YOLOv10n exhibits limitations in both generalization capability and convergence speed, which in turn affects the model's ability to distinguish targets. To address this issue, this paper introduces an improved Inner-GIoU loss function to replace the conventional approach, thereby enabling a more accurate measurement of the differences between predicted and ground-truth boxes, and effectively enhancing detection accuracy, generalization, and robustness. Currently, IoU-based loss functions have been widely adopted and are gradually becoming mainstream. The computation is shown in Equation (14), where A and B denote the areas of the predicted and ground-truth boxes, respectively, and L_{IoU} represents the corresponding IoU loss value.

$$L_{IoU} = 1 - \left|\frac{A\cap B}{A\cup B}\right| \tag{14}$$

Currently, most methods are built upon IoU, with further enhancements introduced into the loss functions. When there is no overlap between the predicted box and the ground-truth box, the IoU value becomes zero, rendering it ineffective in reflecting the distance between the two boxes. To overcome this limitation, GIoU[22]introduces the smallest enclosing box, effectively addressing the inability of IoU to provide optimization information in non-overlapping scenarios. As a result, even when the predicted and ground-truth boxes do not intersect, the model can still obtain meaningful gradient directions for bounding box optimization. The formulation of the GIoU loss is presented in Equation (15).

$$L_{GIoU} = L_{IoU} + \frac{|C - A\cup B|}{|C|} \tag{15}$$

In Equation (15), C denotes the minimum enclosing rectangle that contains both the predicted and ground-truth boxes, and L_{GIoU} represents the GIoU loss function. Common loss functions typically introduce additional geometric constraints on top of the IoU loss, which can accelerate model convergence and improve detection performance; however, they do not fully address the inherent limitations of IoU itself. To overcome this, Inner-

IoU proposes the use of auxiliary bounding boxes to compute IoU, thereby enhancing regression convergence speed without introducing any additional loss terms. The corresponding formulations are provided in Equations (16) to (20).

$$b_1 = x_c - \frac{w*r}{2}, b_2 = x_c + \frac{w*r}{2} \tag{16}$$

Here, x_c and y_c represent the x- and y-coordinates of the center of the predicted box, where the subscript c denotes the center point. h and w refer to the height and width of the predicted box, respectively. The scale factor r, which ranges from [0.5,1.5], is used to control the size of the auxiliary bounding box. b_l, b_r, b_t and b_b correspond to the x-coordinates of the left and right boundaries and the y-coordinates of the top and bottom boundaries of the predicted box, respectively, where the subscripts l, r, t, and b indicate left, right, top, and bottom.

$$\begin{aligned} i &= (\min(b_r^{(gt)}, b_r) - \max(b_l^{(gt)}, b_l)) \\ &*(\min(b_b^{(gt)}, b_b) - \max(b_t^{(gt)}, b_t)) \end{aligned} \tag{17}$$

In Equation (17), $b_l^{(gt)}$, $b_r^{(gt)}$, $b_t^{(gt)}$ and $b_b^{(gt)}$ represent the x-coordinates of the left and right boundaries and the y-coordinates of the top and bottom boundaries of the ground-truth box, respectively. The variable i denotes the intersection area between the auxiliary predicted box and the auxiliary ground-truth box.

$$u = (w^{(gt)} * h^{(gt)}) * r^2 + (w*h) * r^2 - i \tag{18}$$

In Equation (18), gt denotes the ground-truth box, while $h^{(gt)}$ and $w^{(gt)}$ represent the height and width of the ground-truth box, respectively. The variable u indicates the union area between the auxiliary predicted box and the auxiliary ground-truth box.

$$L_{Inner-IoU} = \frac{i}{u} \tag{19}$$

Therefore, Inner-IoU[23] computes the IoU between auxiliary bounding boxes, denoted as $L_{Inner-IoU}$, which represents the Inner-IoU loss function. This loss function not only inherits some advantages of the traditional IoU loss but also introduces its own distinctive features. Inner-GIoU combines the concepts of Inner-IoU and GIoU, enabling a more precise assessment of the similarity between predicted and ground-truth boxes. This approach effectively addresses the limitations in generalization capability and enhances the model's ability to detect small object defects. The corresponding formulation is provided in Equation (19), where $L_{Inner-IoU}$ denotes the Inner-GIoU loss function.

$$L_{Inner-GIoU} = L_{GIoU} + 1 - L_{IoU} - L_{Inner-IoU} \tag{20}$$

4. Experiment and analysis

4.1 Experimental environment

The experiment was conducted in a Linux Ubuntu 22.04 environment, with a hardware setup consisting of an NVIDIA RTX 4090 GPU (24GB VRAM) and an Intel Xeon Platinum 8352V processor (2.10GHz). The deep learning framework used was PyTorch 2.3.0, running on Python 3.12with CUDA 12.1 support.

During training, the model was configured with an initial learning rate of 0.01, a batch size of 16, and trained for 200 epochs. The input image size was standardized to 640×640, and the IoU threshold was set to 0.5. The SGD optimizer was used, with a momentum parameter of 0.937 and a confidence threshold of 0.25, to improve training convergence speed and ensure model stability.

4.2 Evaluation indicators

This study comprehensively evaluates and compares the overall performance of the model using multiple metrics, including precision, recall, mean average precision (mAP), inference speed (FPS), model size (in MB), computational complexity (GFLOPs), and the total number of parameters, aiming to provide a holistic assessment of the model's accuracy, efficiency, and lightweight characteristics.

Precision measures the proportion of true positive samples among all instances predicted as positive by the model. It reflects the reliability of the model's positive predictions — higher precision indicates fewer false positives. The detailed calculation is shown in Equation (21).

$$P = \frac{TP}{TP+FP} \tag{21}$$

Recall represents the ratio of correctly identified positive samples to the total number of actual positive samples. It reflects the model's ability to capture all relevant positive instances—a higher recall indicates fewer missed detections. The specific calculation is provided in Equation (22).

$$R = \frac{TP}{TP+FN} \tag{22}$$

Mean average precision (mAP) serves as a key metric for assessing the overall effectiveness of object detection models. It is calculated by averaging the Average Precision (AP) across all object categories, providing an integrated measure of the model's classification accuracy and localization precision — a higher mAP indicates better detection performance.

Specifically, mAP@0.5 evaluates performance at a single Intersection over Union (IoU) threshold of 0.5, where a prediction is considered correct if its bounding box overlaps with the ground truth by at least 50%. In contrast, mAP@0.5:0.95 offers a more stringent and detailed evaluation by calculating AP at multiple IoU thresholds (from 0.5 to 0.95 in increments of 0.05) and averaging the results. This multi-threshold strategy allows for a more granular analysis of how well the model performs under varying localization requirements. The formal definition is given in Equation (24).

$$AP = \int_0^1 P(R)dR \tag{23}$$

$$mAP = \frac{1}{n}\sum_{i=0}^{n} AP_i \tag{24}$$

In object detection or defect recognition tasks, TP (True Positive) refers to cases where the model correctly identifies an actual defect; FP (False Positive) refers to instances where the model incorrectly classifies a normal sample as defective; FN (False Negative) refers to situations where the model fails to detect an existing defect and mistakenly classifies it as normal.

FPS (Frames Per Second) measures the number of image frames the model can process within one second. A higher FPS indicates greater computational efficiency and enhanced capability for real-time detection.

The model size represents the storage size of the model. Larger models require more memory and computational resources during deployment and operation. (Unit: MB)

GFLOPs (Giga Floating Point Operations per Second) measures the number of billion floating-point operations a model performs per second. It reflects computational complexity — the higher the value, the greater the computation required.

The number of parameters reflects the structural complexity of a model, representing the total count of trainable weights within the network, typically measured in millions (M). Models with fewer parameters are more lightweight and better suited for deployment in resource-constrained environments.

4.3 Datasets

The dataset used in this experiment was collected by the Turkish National Railway Research and Technology Center (DATEM) with a measurement train, and contains 3 types of track surface defects: track holes (hole), track seams (seam), and track wear (wear). The dataset consists of 1227 images, which are randomly divided into training, validation, and test sets in an 8:1:1 ratio. All images were annotated using the MakeSense software. Fig. 9 displays the count of each defect type and Fig. 10 shows samples of rail defects.

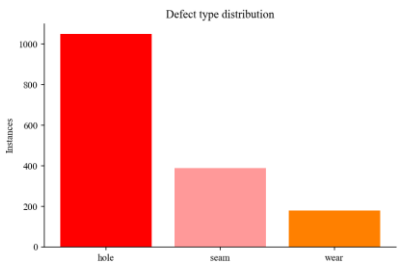


Fig.9 Rail surface defect distribution

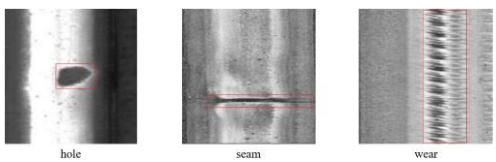


Fig.10 Examples of rail surface defects

4.4 Ablation experiment

To evaluate the contribution of each improved module to the model's performance, a series of ablation experiments were conducted, as shown in Table 1. The baseline model is YOLOv10n, with an mAP@0.5 of 87.5%, an mAP@0.5:0.95 of 42.4%, a frame rate of 105 FPS, 2.70M parameters, and a computational complexity of 8.2G.

In the second configuration, the C2f-EMAN module was introduced to enhance the backbone's focus on critical features, resulting in a 1.8% increase in mAP@0.5 and a 3.1% increase in mAP@0.5:0.95. However, due to the added complexity, the FPS dropped by 7.

In the third configuration, AKConv replaced the standard convolution, boosting mAP@0.5 by 1.0% and increasing the inference speed by 18 FPS, while slightly reducing model size and complexity, thereby improving overall performance.

The fourth configuration incorporated the DySample module to enable adaptive spatial sampling and improve feature alignment, resulting in an mAP@0.5 of 89.0%.

The fifth configuration integrated the Inner-GIoU loss function, which focused on more accurate bounding box regression and steadily improved mAP by 1.6% while maintaining computational efficiency.

In the sixth configuration, combining C2f-EMAN and AKConv led to a 2.3% increase in mAP@0.5 and a 5.9% increase in mAP@0.5:0.95, with a reduction of 0.03M in parameters. The seventh configuration further introduced DySample, pushing mAP@0.5 to 90.1%, achieving a balance between accuracy and efficiency.

Finally, the eighth configuration, named EADI-YOLO, integrates all four modules and achieves further improvements: mAP@0.5 reaches 90.3%, mAP@0.5:0.95 reaches 49.1%, and FPS increases to 120. Compared with the original YOLOv10n, EADI-YOLO improves mAP@0.5 by 2.8%, mAP@0.5:0.95 by 6.7%, and FPS by 15, while reducing the parameter count by 0.02M. The computational complexity and model size remain unchanged.

Fig. 11 clearly demonstrates that EADI-YOLO optimizes model complexity while maintaining accuracy and speed.

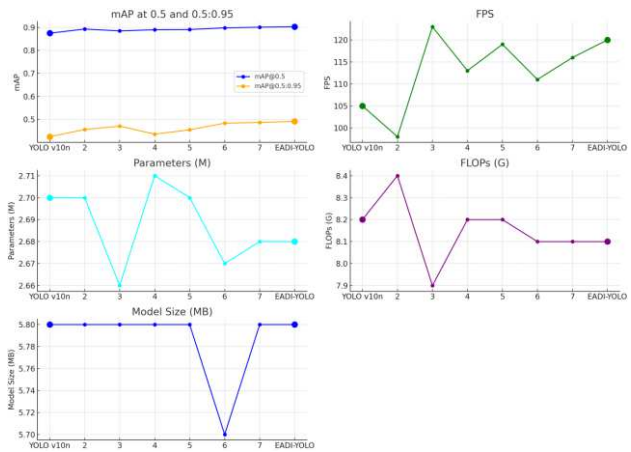


Fig.11 Comparison of improved module performance

Table 1 Ablation experiment

Algorithms	C2f-EMAN	AKConv	DySample	Inner-GIoU	mAP@0.5/%	mAP@0.5:0.95/%	FPS	Parameters/M	FLOPs(G)	Model size/MB
YOLOv10n					0.875	0.424	105	2.70	8.2	5.8
2	√				0.893	0.455	98	2.70	8.4	5.8
3		√			0.885	0.470	123	2.66	7.9	5.7
4			√		0.890	0.435	113	2.71	8.2	5.8
5				√	0.891	0.454	119	2.70	8.2	5.8
6	√	√			0.898	0.483	111	2.67	8.1	5.7
7	√	√	√		0.901	0.486	116	2.68	8.1	5.8

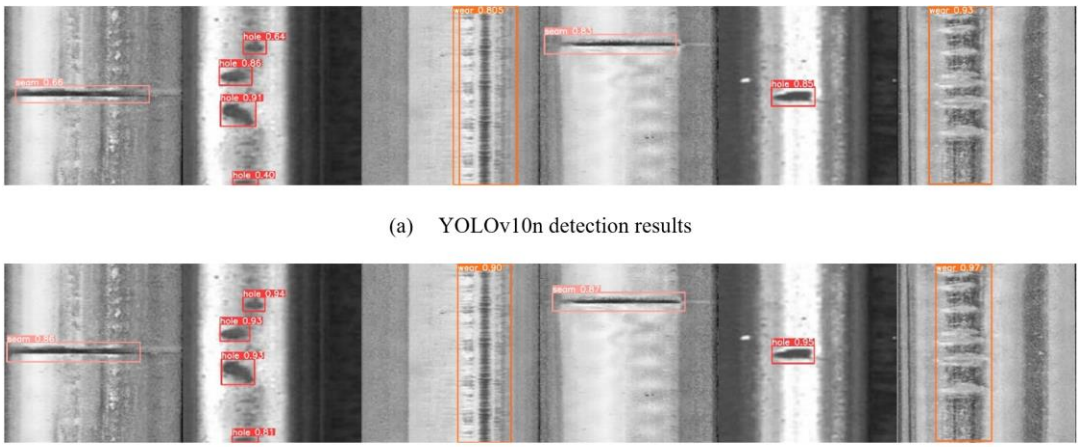
EADI-YOLO	√	√	√	√	0.903	0.491	120	2.68	8.1	5.8

As shown in Table 2, EADI-YOLO outperforms YOLOv10n in detecting all three types of rail surface defects. For hole defects, EADI-YOLO achieves an AP of 0.955, which is a 2.1% improvement over YOLOv10n's 0.934. For seam defects, EADI-YOLO reaches 0.868, improving by 2.7%. For wear defects, it attains an AP of 0.887, which is 3.7% higher than YOLOv10n's 0.850. These results demonstrate that EADI-YOLO exhibits stronger target discrimination capability across various defect types.

Table 2. Detection performance comparison on different rail Surface defect types

Algorithms	hole	seam	wear
YOLOv10n	0.934	0.841	0.850
EADI-YOLO	0.955	0.868	0.887

The defect detection results in Fig. 12 demonstrate that EADI-YOLO outperforms YOLOv10n in both localization accuracy and confidence. For example, in the wear region, YOLOv10n produces two highly overlapping prediction boxes, indicating uncertainty in boundary recognition and insufficient clarity in target identification. In contrast, EADI-YOLO outputs a single prediction box with a confidence score of 0.90, showing accurate boundaries and better alignment with the actual defect shape, resulting in more stable predictions. This comparison further confirms the superior performance of the proposed EADI-YOLO algorithm in rail surface defect detection tasks.



(a) YOLOv10n detection results

(b) EADI-YOLO detection results

Fig.12 Comparison of defect detection results

4.5 Comparative experiment

As shown in Table 3 and Fig. 13, EADI-YOLO demonstrates a clear advantage in the mAP metric, with mAP@0.5 and mAP@0.5:0.95 values of 0.903 and 0.491, respectively, higher than all other models, highlighting its superior accuracy. Although EADI-YOLO is not the highest in FPS, Parameters/M, GFLOPs, and Model size, its performance remains within a reasonable range and is the most balanced among all models.

EADI-YOLO strikes an excellent balance between accuracy, speed, and model complexity, maintaining high precision while ensuring good processing speed and low computational cost, making it well-suited for real-time railway track defect detection.

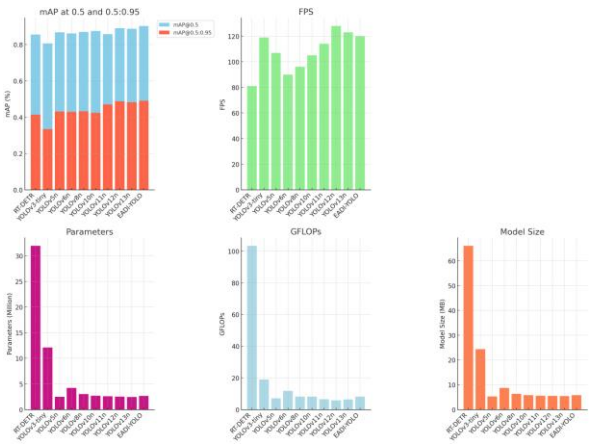


Fig.13 Algorithm performance comparison

Table 3 Comparison of different algorithms

Algorithms	mAP@0.5/%	mAP@0.5:0.95/%	FPS	Parameters/M	GFLOPs	Model size/MB
RT-DETR[24]	0.856	0.414	81	31.98	103.4	66.1
YOLOv3-tiny[25]	0.807	0.334	119	12.13	18.9	24.4
YOLOv5n[26]	0.868	0.431	107	2.50	7.1	**5.3**
YOLOv6n[27]	0.862	0.429	90	4.23	11.8	8.7

YOLOv8n[28]	0.870	0.432	96	3.01	8.1	6.3
YOLOv10n[17]	0.875	0.424	105	2.70	8.2	5.8
YOLOv11n[29]	0.858	0.470	114	2.60	6.5	5.6
YOLOv12n[30]	0.891	0.487	**128**	2.51	**5.8**	5.5
YOLOv13n[31]	0.887	0.482	123	**2.46**	6.4	5.4
EADI-YOLO	**0.903**	**0.491**	120	2.68	8.1	5.8

4.5 Generalization experiment

To evaluate the generalization performance of the EADI-YOLO algorithm, a publicly available rail defect dataset from the Roboflow platform was selected. This dataset contains 1,100 images, covering three types of defects: chipping, scratch, and seams. Of these, 1,000 images were split into training and validation sets in a 9:1 ratio, with the remaining 100 images used as the test set.

Comparative experiments were conducted on this dataset using the RT-DETR, YOLOv10n, YOLOv13n, and EADI-YOLO models, each trained for 200 epochs. The results, as shown in Table 4, indicate that EADI-YOLO outperforms the other models in detection accuracy. Specifically, EADI-YOLO achieved a mAP@0.5 of 85.4%, which is 3.1%, 2.0%, and 1.3% higher than RT-DETR, YOLOv10n, and YOLOv13n, respectively. Additionally, its mAP@0.5:0.95 reached 48.5%, the highest among all compared models. In terms of inference speed, EADI-YOLO achieved 154 frames per second (FPS), demonstrating excellent real-time processing capability and a favorable balance between detection accuracy and inference efficiency. Fig. 14 shows the performance of different models in defect detection, with EADI-YOLO demonstrating more prominent performance in defect localization accuracy, reducing missed detections, and increasing prediction confidence.

Table 4 Comparison of different algorithms performance

Algorithms	mAP@0.5/%	mAP@0.5:0.95/%	FPS
RT-DETR	0.823	0.367	95
YOLOv10n	0.834	0.483	147
YOLOv13n	0.841	0.476	163
EADI-YOLO	0.854	0.485	154



Fig.14 Comparison of defect detection performance

5. Conclusion

To enhance the accuracy and speed of rail surface defect detection, this paper proposes an improved detection model named EADI-YOLO, based on the YOLOv10n framework. The model integrates the C2f-EMAN model, AKConv deformable convolution, DySample dynamic upsampling, and Inner-GIoU bounding box regression loss, offering comprehensive optimizations from feature extraction and structural deformation perception to detail reconstruction and localization accuracy. With only 2.68M parameters and a computational complexity of 8.1 GFLOPs, EADI-YOLO achieves a mAP@0.5 of 90.3% and a mAP@0.5:0.95 of 49.1%, representing improvements of 2.8% and 6.7%, respectively, over the baseline YOLOv10n. Inference speed is also increased to 120 FPS.Compared to other mainstream detection models such as RT-DETR and YOLOv13n, EADI-YOLO demonstrates a better balance between accuracy, speed, and model complexity, showcasing its excellent overall performance. Generalization experiments show that EADI-YOLO maintains this balance on other rail defect datasets as well. Future research will focus on enhancing the model's robustness and cross-domain adaptability, exploring multi-scale feature guidance mechanisms, lightweight architecture search, and dynamic augmentation strategies to further advance real-time intelligent detection in complex railway environments.

Author Contributions Data collection and preprocessing were carried out by L and H. Model design, improvement, experiments, and result analysis were led by M. The initial draft of the manuscript was written by

M and H. All authors contributed to the revision and review of the manuscript.

Funding This study was supported by the National Natural Science Foundation of China (U24B20159), the Key Project of Scientific Research of Liaoning Provincial Department of Education (LJKZ0475), the Innovation Support Program for High-level Talents in Dalian (2022RJ03), and the Transportation Technology Foundation of Liaoning Province, China (202416).

Data Availability The data involved in this study can be provided by the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors state that there are no conflicts of interest in relation to this article.

Ethical approval and informed consent This study did not involve human participants or animal experiments, nor did it include any patient-related data; therefore, no informed consent statement is required.

References

1. Koohmishi, M., Kaewunruen, S., Chang, L., et al.: Advancing railway track health monitoring: Integrating GPR, InSAR and machine learning for enhanced asset management. Automation in construction, 162, 105378 (2024).
2. Zhang, H., Cui, X., Yin, Y., et al.: Comparison and optimization of rail defect detection methods based on object detection model. Tribology Transactions, 68(1), 171-179 (2025).
3. Cao, Y., Liu, Y., Sun, Y., et al.: Enhancing rail safety through real-time defect detection: A novel lightweight network approach. Accident Analysis & Prevention, 203, 107617 (2024).
4. Zhao, Y., Liu, Z., Yi, D., et al.: A review on rail defect detection systems based on wireless sensors. Sensors, 22(17), 6409 (2022).
5. Wang, M., Li, K., Zhu, X., et al.: Detection of surface defects on railway tracks based on deep learning. IEEE Access, 10, 126451-126465 (2022).
6. Mettes, P., Ghadimi Atigh, M., Keller-Ressel, M., et al.: Hyperbolic deep learning in computer vision: A survey. International Journal of Computer Vision, 132(9), 3484-3508 (2024).
7. Jiang, P., Ergu, D., Liu, F., et al.: A Review of Yolo algorithm developments. Procedia computer science, 199, 1066-1073 (2022).
8. Williams, C., Falck, F., Deligiannidis, G., et al.: A unified framework for U-Net design and analysis. Advances in Neural Information Processing Systems, 36, 27745-27782 (2023).
9. Wang, J., Li, M.: Small target detection model based on swin transformer and multiscale fusion. Signal, Image and Video Processing, 19(11), 1-11 (2025).
10. Liu, Y., Wei, X.: Track Surface Defect Detection Based on Image Processing. Lecture Notes in Electrical Engineering 483, 225 (2018).
11. Ni, X., Liu, H., Ma, Z., et al.: Detection for rail surface defects via partitioned edge feature. IEEE Transactions on Intelligent Transportation Systems, 23(6), 5806-5822 (2021).
12. Yang, H., Wang, Y., Hu, J., et al.: Segmentation of track surface defects based on machine vision and neural networks. IEEE Sensors Journal, 22(2), 1571-1582 (2021).
13. Alvarenga, T. A., Carvalho, A. L., Honorio, L. M., et al.: Detection and classification system for rail surface defects based on Eddy current. Sensors, 21(23), 7937 (2021).
14. Niu, M., Wang, Y., Song, K., et al.: An adaptive pyramid graph and variation residual-based anomaly detection network for rail surface defects. IEEE Transactions on Instrumentation and Measurement, 70, 1-13 (2021).
15. Zhang, J., Kang, X., Ni, H., et al.: Surface defect detection of steel strips based on classification priority YOLOv3-dense network. Ironmaking & Steelmaking, 48(5), 547-558. (2021).
16. Si, C., Luo, H., Han, Y., et al.: Rail-STrans: A rail surface defect segmentation method based on improved Swin Transformer. Applied Sciences, 14(9), 3629 (2024).
17. Wang, A., Chen, H., Liu, L., et al.: Yolov10: Real-time end-to-end object detection. Advances in Neural Information Processing Systems, 37, 107984-108011 (2024).
18. Ouyang, D., He, S., Zhang, G., et al.: Efficient Multi-Scale Attention Module with Cross-Spatial Learning. arxiv preprint arxiv:2305.13563 (2023).
19. Wu, H., Zhu, Y., Wang, L.: A Dense Small Object Detection Algorithm Based on a Global Normalization Attention Mechanism. Applied Sciences, 13(21), 11760 (2023).
20. Zhang, X., Song, Y., Song, T., et al.: AKConv: Convolutional kernel with arbitrary sampled shapes and arbitrary number of parameters. arxiv preprint arxiv:2311.11587, 2-10 (2023).
21. Liu, W., Lu, H., Fu, H., et al.: Learning to Upsample by Learning to Sample. arxiv preprint arxiv:2308.15085 (2023).
22. Tong, C., Yang, X., Huang, Q., et al.: NGIoU loss: Generalized intersection over union loss based on a new bounding box regression. Applied Sciences, 12(24), 12785 (2022).
23. Zhang, H., Xu, C., Zhang, S.: Inner-IoU: more effective intersection over union loss with auxiliary bounding box. arxiv preprint arxiv:2311.02877 (2023).
24. Zhao, Y., Lv, W., Xu, S., et al.: DETRs Beat YOLOs on Real-time Object Detection. arxiv preprint arxiv:2304.08069 (2023).
25. Sha, G., Wu, J., Yu, B.: The Shrank YoloV3-tiny for spinal fracture lesions detection. Journal of Intelligent & Fuzzy Systems, 42(3), 2809-2828 (2022).
26. Tian, J. H., Feng, X. F., Li, F., et al.: An improved YOLOv5n algorithm for detecting surface defects in industrial components. Scientific Reports, 15(1), 9756 (2025).
27. Li, C., Li, L., Geng, Y., et al.: Yolov6 v3. 0: A full-scale reloading. arxiv preprint arxiv:2301.05586 (2023).
28. Sohan, M., Ram, T. S., Reddy, C. V. R.: A Review on YOLOv8 and Its Advancements. Algorithms for Intelligent Systems, 529-545 (2024).
29. Rasheed, A. F., Zarkoosh, M.: YOLOv11 optimization for efficient resource utilization. The Journal of Supercomputing, 81(9), 1-21 (2025).
30. Tian, Y., Ye, Q., Doermann, D.: Yolov12: Attention-centric real-time object detectors. arXiv preprint arXiv:2502.12524 (2025).
31. Lei, M., Li, S., Wu, Y., et al.: YOLOv13: Real-Time Object Detection with Hypergraph-Enhanced Adaptive Visual Perception. arxiv preprint arxiv:2506.17733 (2025).