Neural Computing and Applications
https://doi.org/10.1007/s00521-022-08035-1

ORIGINAL ARTICLE

MSSIF-Net: an efficient CNN automatic detection method for freight train images

Longxin Zhang[1] · Yang Hu[1] · Jingsheng Chen[2] · Chuang Li[2] · Keqin Li[3]

Received: 9 February 2022 / Accepted: 4 November 2022

Abstract
Freight trains are one of the most important modes of transportation. The fault detection of freight train parts is crucial to ensure the safety of train operation. Given the low detection efficiency and accuracy of traditional train fault detection methods, a novel one-stage object detection method called the multi-scale spatial information fusion CNN network (MSSIF-Net) based on YOLOv4 is proposed in this study. The adaptive spatial feature fusion method and multi-scale channel attention mechanism are used to construct the multi-scale feature sharing network and consequently realize feature information sharing at different levels and promote detection accuracy. The mean average precision values of MSSIF-Net on the train image test set, PASCAL VOC 2007 test set, and surface defect detection dataset are 94.73%, 87.76%, and 75.54%, respectively, outperforming YOLOv4, Faster R-CNN, CenterNet, RetinaNet, and YOLOX-l. The detection speed of MSSIF-Net is 33.10 FPS, achieving a good balance between detection accuracy and speed. In addition, the MSSIF-Net performance is estimated after adding noise or rotating the train images at a slight angle to simulate a real scene. Experimental results indicate that MSSIF-Net has favorable anti-interference ability.

Keywords Attention mechanism · Feature fusion · Freight train fault · Object detection

1 Introduction

1.1 Background and motivation

With the continuous development of the national economy, the scale of train structures and the complexity of their parts increase. Trains also have heavy-load and high-density characteristics. Traditional train fault detection relies on manual detection, which has low efficiency despite the required large manpower input. The large amount of original images for freight train fault detection is contributed by the trouble of moving freight car detection system (TFDS). TFDS has changed train fault detection from the traditional outdoor manual field detection to the image analysis of indoor train inspectors, largely improving the train detection efficiency. High-speed cameras at a single station can capture tens of millions of train images a day, whereas experienced inspectors can examine only a few thousand a day per person. With the requirements of high reliability and efficiency, automatic train fault detection based on computer vision technology has become a hot topic in the field of object detection.

✉ Longxin Zhang
longxinzhang@hut.edu.cn

✉ Chuang Li
chuangli@hutb.edu.cn

Yang Hu
huyangbjt@163.com

Jingsheng Chen
chenjingsheng49@163.com

Keqin Li
lik@newpaltz.edu

[1] College of Computer Science, Hunan University of Technology, Taishan, Zhuzhou 412007, Hunan, China

[2] School of Computer Science, Hunan University of Technology and Business, Yuelu, Changsha 410205, Hunan, China

[3] Department of Computer Science, State University of New York, 100 Hawk Drive, New Paltz, NY 12561, USA

In recent years, with the development of modern technology, such as information technology, artificial intelligence, and automatic detection, train fault detection has shifted into automation and intelligence. Sun et al. [1] designed the automatic fault recognition system (AFRS) for freight trains based on the convolutional neural network (CNN). Fu et al. [2] proposed a two-stage attention-aware method based on the CNN for train bearing fault detection. Tim et al. [3] used the long short-term memory (LSTM) recurrent neural network to realize the fault diagnosis of railway track circuits. Their comparison with the CNN indicates that the LSTM architecture is more suitable for the fault detection and recognition of railway track circuits. Zhang et al. [4] developed an end-to-end lightweight framework named LR FTI-FDet for the real-time fault detection of freight train images. Most of the abovementioned methods use deep learning methods, in which detection efficiency is greatly improved compared with the current human–machine combination of train detection methods. However, with the increasing demand to ensure the accuracy of automatic train fault detection, improving the accuracy of train fault detection remains to be the focus of numerous research.

1.2 Our contributions

Train images taken by TFDS are mostly in grayscale, and the proportion of train parts in the image is small. Inspired by the multi-scale feature fusion and attention mechanism method, this study proposes an object detection method based on the improved attention mechanism and multi-scale feature sharing network (MFSN). It is called multi-scale spatial information fusion CNN network (MSSIF-Net), which is intended to achieve high-detection accuracy.

The contributions of this study are as follows.

- An MFSN based on a one-stage object detector is proposed.
- As the midpoint of the feature extraction, classification and regression networks of the detector, MFSN can fuse the high-level features with low resolution but rich semantic information and the low-level features with high resolution but rich location details to obtain feature maps with rich semantic and location information at different scales.
- A multi-scale channel attention (MCA) mechanism is introduced by group convolutions, enabling the detector to focus on the object to be detected. MCA, which is embedded into the proposed MFSN, can automatically learn and pay more attention to useful features within the object areas.
- The proposed MSSIF-Net is evaluated on the Train Image Test Set (TITS), and PASCAL VOC 2007 test set and surface defect detection dataset from hot-rolled steel sheets. The experimental results on these test sets indicate that MSSIF-Net can effectively improve detection in terms of the average precision and mean average precision.

The remainder of this paper is organized as follows. Section 2 presents the related work on object detection and attention mechanism. The proposed MSSIF-Net detector is introduced in Section 3. In Section 4, the implementation details, evaluation metrics, and analysis of experimental results of the MSSIF-Net are presented. Section 5 summarizes this study and discusses the next research plan.

2 Related works

2.1 Object detection based on deep learning

Object detection based on deep learning can be divided into anchor-based detection and anchor-free detection, i.e., whether anchor information is utilized. Anchor-based detectors consist of two-stage detectors and one-stage detectors depending on the extraction of region proposals [5, 6]. Initially applied in natural language processing, transformer has been introduced into object detection in recent years.

In the two-stage detection of anchor-based detectors, the region-based convolutional neural network (R-CNN) [7] first applies deep learning to object detection. On this basis, a number of improvement methods, such as Fast R-CNN [8], Faster R-CNN [9], and Mask R-CNN [10], among others, have been proposed. The original image in Fast R-CNN [8] is directly convolved instead of each region proposal, which improves computational efficiency. Similar to R-CNN, Fast R-CNN uses the selective search (SS) [11] algorithm to extract region proposals, but its detection speed is relatively low in scenes requiring high real-time performance. Faster R-CNN [9] is based on the Fast R-CNN, and the region proposal network (RPN) is used to replace the SS algorithm. Thus, the final detection speed and detection accuracy of Faster R-CNN [9] are improved. In Mask R-CNN [10], region of interest (RoI) pooling in Faster R-CNN is replaced by RoI alignment, and the bilinear interpolation method is used to pad the pixels of non-integer positions in the image. Hence, Mask R-CNN has a good application prospect in object detection and instance segmentation.

One-stage object detectors primarily involve You Only Look Once (YOLOv1) [12], YOLOv2 [13], YOLOv3 [14], YOLOv4 [15], single-shot multibox detector (SSD) [16], RetinaNet [17], and so on. YOLOv1 [12] greatly improves the detection speed by using a single network to complete the judgment of object categorization and regression of a location, but the detection accuracy is relatively low. YOLOv2 [13] uses the methods of batch normalization, high-resolution classifier, and k-means algorithm to generate prior bounding boxes for solving the problem of low-detection accuracy in YOLOv1. YOLOv3 [14] exploits Darknet19, a new neural network structure, as the backbone network and is combined with the feature pyramid network (FPN) [18] to complete the multi-scale prediction and ultimately improve the detection accuracy. Bochkovskiy et al. [15] proposed YOLOv4 with high-detection accuracy based on YOLOv3. The backbone network for the feature extraction in YOLOv4 is replaced by CSPDarkNet53, and FPN is replaced by the path aggregation network (PANet) [19]. YOLOv4 also utilizes the spatial pyramid pooling network (SPPNet) [20] to expand the receptive field range of the low-level feature map and the Mosaic and other data enhancement methods to train the detector. SSD [16] adopts the improved VGG16 as the backbone network and adds four convolution operations of different sizes to achieve multi-scale prediction and consequently improve the detection accuracy and speed. Focal loss is used in RetinaNet [17] to address the imbalance problem of the positive and negative samples in the images.

Anchor-free detectors ignore the generation stage of prior bounding boxes and achieve object detection only through the key point estimation of object positions. Examples include CornerNet [21], CenterNet [22], YOLOX [23], and so on. CornerNet [21] utilizes the corner pooling strategy to generate the top-left and bottom-right points for determining the location of objects. The peak points of the heatmap are taken as the center points of the objects by CenterNet [22], and then the center points are utilized to conduct regression and classification of the object. YOLOX [23] converts the YOLO series object detectors, such as YOLOv3 and YOLOv4, into high-performance and anchor-free forms, which benefits the detection accuracy.

For the transformer-based detectors, detection transformer (DETR) [24] is an end-to-end object detection method based on transformer and CNN. In DETR, object detection task is regarded as a set of prediction problem [24]. However, the detection effect on small objects is poor, training the detector consumes much time. Deformable DETR [25] combines transformer and deformable convolution to address the problems of slow training convergence and high computational complexity in DETR. Fang et al. [26] directly migrated the fine-tuned vision transformer [27] model into the object detection task and proposed YOLOS. In YOLOS, the network structure similar to CNN is directly removed, and only the encoder part in transformer can be used to achieve regression and classification of candidate objects. However, to ensure the detection accuracy, the detector training time inevitably increases after transformer is applied into the object detection task.

Here, the one-stage detection of the anchor-based detector is selected as it can balance detection accuracy and speed.

2.2 Attention mechanism

In essence, the purpose of the attention mechanism is to achieve efficient allocation of information resources, and it is widely used in object detection. Its attention domain usually includes the spatial, channel, and hybrid domain. Spatial transformer network (STN) [28] and spatial attention mechanism (SAM) [29] are typical forms of spatial attention. Both use two different spaces for information transformation, retain the key information of the image in the transformation process, and assign the same feature channel attention weight to the relevant spatial position. Squeeze and excitation network (SENet) [30] is a typical channel attention scheme, where weights are used to highlight the importance of different feature channels. Selective kernel network (SKNet) [31] introduces a dynamic selection mechanism into the channel attention mechanism in CNNs. Fast 1D convolution is used in efficient channel attention (ECA) [32] to capture the interaction information between the feature channels, effectively improving the learning efficiency of the model. Frequency channel attention network (FcaNet) [33] applies the idea of frequency domain to the channel attention mechanism and exploits the discrete cosine transform (DCT) to extract more information from feature channels. convolutional block attention module (CBAM) [34] is a hybrid attention mechanism based on channel attention and spatial attention, and it relies on the feature map of an object at the channel and space domains, respectively. Attention mechanism is generally embedded as a module in the object detection model, and it avoids making changes on the original structure of the model. The efficient pyramid split attention (EPSA) proposed in Ref. [35] is innovative in its use of the attention mechanism. The ordinary convolution in the original network is replaced by the attention mechanism EPSA, thus realizing the effective utilization of multi-scale context information.

3 Proposed method

This section introduces the overall architecture, internal structure, and loss function of the proposed detector in this study.

3.1 Overall architecture

The detector proposed in this study is MSSIF-Net, which is a one-stage object detection method. The architecture of MSSIF-Net is shown in Fig. 1. MSSIF-Net can be divided into four parts, namely feature extraction network (FEN), spatial pyramid pooling (SPP), MFSN, and regression and classification network (RCN). The working process of MSSIF-Net can be described as follows. First, FEN module is adopted by MSSIF-Net to extract the feature information of parts in freight train images, and then FEN outputs the three feature maps of M_3, M_4, and M_5 for the subsequent modules of SPP and MFSN. Second, the SPP module enlarges the receptive field of feature map M_5 and outputs feature map P_5. Third, the MFSN module obtains the feature maps with semantic information and object location information at different scales, and it improves the detection accuracy by fusing the three feature maps P_3, P_4, and P_5. Finally, the RCN module regresses and classifies the faulty train parts in the feature maps of P_3'', P_4'', and P_5''.

3.2 FEN

The structure of the FEN module is shown in Fig. 2a. The FEN module consists of one initialization block (IB) and five residual bodies (RBs). The IB module contains a convolution layer with a convolution kernel size of 3, a batch normalization (BN) layer, and a Mish activation function, which is used to transform the dimension of the input image channel. The RB module is used for image feature extraction. The input size of the train image is defined as $W \times H \times C$, where W and H are the width and height of the image, respectively, and C is the number of channels in the image. FEN sets the input size of the image to $416 \times 416 \times 3$. When the input size of the image is inconsistent with that set by FEN, the image size is adjusted using a resizer operation. The workflow of FEN can be divided into two steps. First, the input image uses the IB module in FEN to obtain feature map M_0. Second, FEN successively uses the five RB modules with different convolutional kernel sizes to obtain the output feature maps M_3, M_3, and M_5.

As shown in Fig. 2b, the RB module with the input size of $W \times H \times C$ is composed of two branches, each of them stacked by Basic block (Bb) modules with different sizes of convolution kernels. The Bb module includes a convolution layer, a BN layer, and a Mish activation function. The first branch of the RB module adjusts the size of the feature map into $W \times H \times (C/2)$ by using a Bb module with a convolution kernel size of 1×1. Similarly, the second branch transforms the size of the feature map to $W \times H \times (C/2)$, which is composed of two Bb modules with a convolution kernel size of 1×1 and N Residual blocks (Rb) in between them. An Rb module comprises two Bb modules with convolution kernel sizes of 1×1 and 3×3 and a residual edge. An addition operation is used to connect and fuse the input and output feature maps of the two Bb modules. The final output result of the RB module is combined using the feature maps generated by the two branches through a concatenation operation. A Bb module with a convolution kernel size of 1 is then used to adjust the channel number of the final generated feature map to ensure that the RB module can output a feature map with a correct size.

The final outputs of M_3 $(52 \times 52 \times 256)$, M_4 $(26 \times 26 \times 512)$, and M_5 $(13 \times 13 \times 1024)$ in FEN are used in the subsequent SPP and MFSN modules, where M_3 and M_4 correspond to P_3 and P_4, and M_5 applies the SPP module to obtain the feature map P_5 .

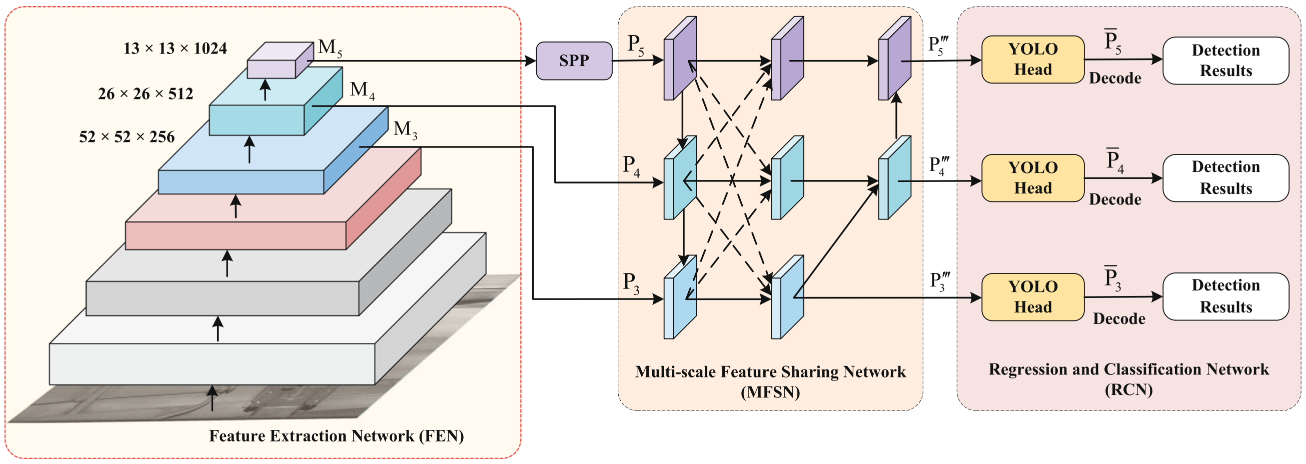
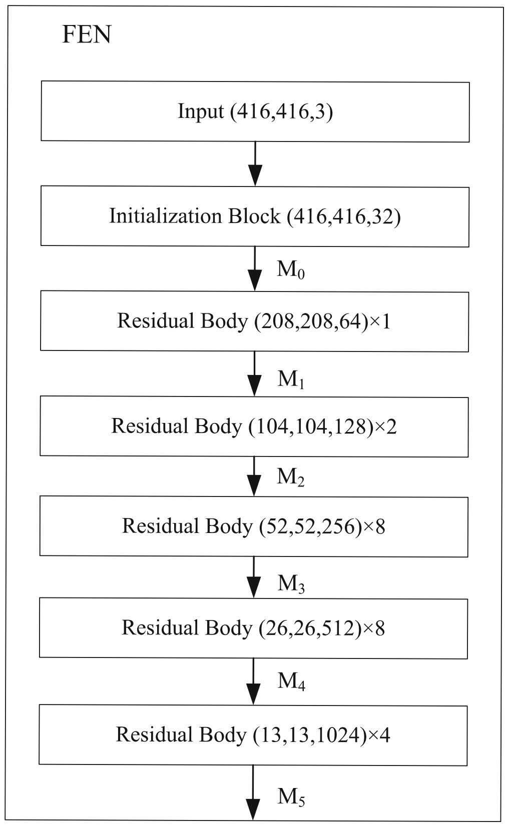
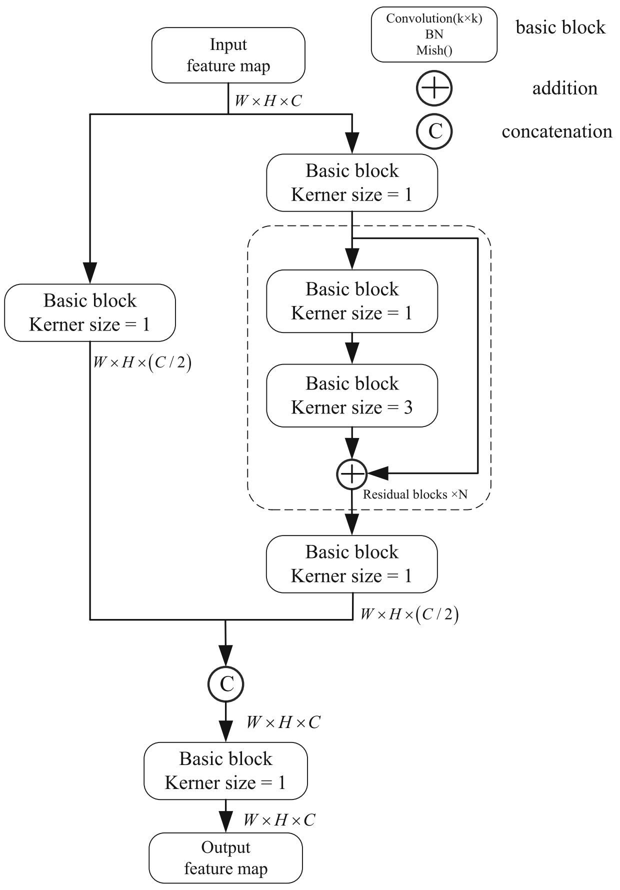

13 × 13 × 1024
M5
26 × 26 × 512
M4
52 × 52 × 256
M3
Feature Extraction Network (FEN)
SPP
P5
P4
P3
Multi-scale Feature Sharing Network (MFSN)
P5'''
P4'''
P3'''
YOLO Head
Decode
Detection Results
Regression and Classification Network (RCN)

Fig. 1 Overall architecture of MSSIF-Net


FEN
Input (416,416,3)
Initialization Block (416,416,32)
M0
Residual Body (208,208,64)×1
M1
Residual Body (104,104,128)×2
M2
Residual Body (52,52,256)×8
M3
Residual Body (26,26,512)×8
M4
Residual Body (13,13,1024)×4
M5

(**a**) FEN structure


Input feature map
Convolution(k×k) BN Mish()
basic block
addition
concatenation
W×H×C
Basic block Kerner size = 1
Basic block Kerner size = 1
W×H×(C/2)
Basic block Kerner size = 1
Basic block Kerner size = 3
Residual blocks ×N
Basic block Kerner size = 1
W×H×(C/2)
C
W×H×C
Basic block Kerner size = 1
W×H×C
Output feature map

(**b**) Residual Body (RB)

Fig. 2 FEN structure and RB

3.3 SPP

In MSSIF-Net, the SPP module is used to replace the fully connected layer in traditional FEN. As shown in Fig. 3, the SPP module is composed of four max pooling layers and a concatenation operation. First, four max pooling layers are utilized by SPP module on feature map M_5 to expand its range of receptive field and improve the detection accuracy of small objects. In the max pooling layers, the convolution kernel sizes are 1, 5, 9, and 13. Second, SPP applies a concatenation operation to fuse the four feature maps generated after the max pooling layer operation to obtain the feature map P_5 ($13 \times 13 \times 1024$), which will be used in the subsequent MFSN model.

3.4 MFSN

The MFSN proposed in this study is based on the FPN and attention mechanism. The three-layer structure of MFSN is shown in Fig. 4b. The two feature maps of P_3 and P_4

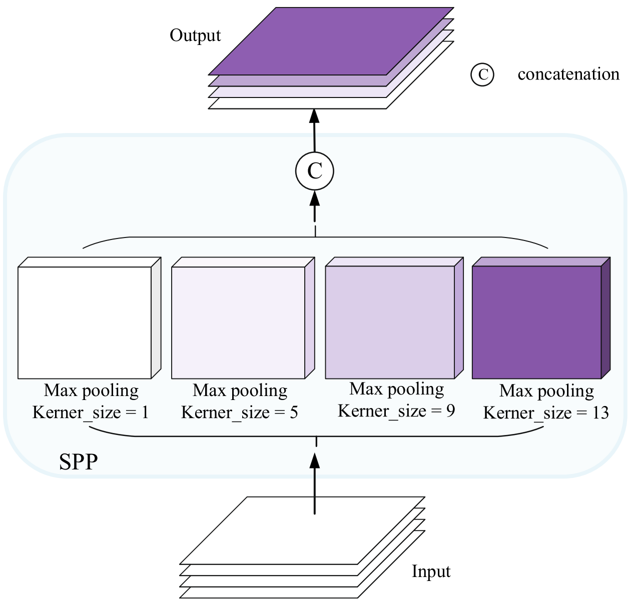

Output
C
concatenation
C
Max pooling Kerner_size = 1
Max pooling Kerner_size = 5
Max pooling Kerner_size = 9
Max pooling Kerner_size = 13
SPP
Input

Fig. 3 Spatial pyramid pooling

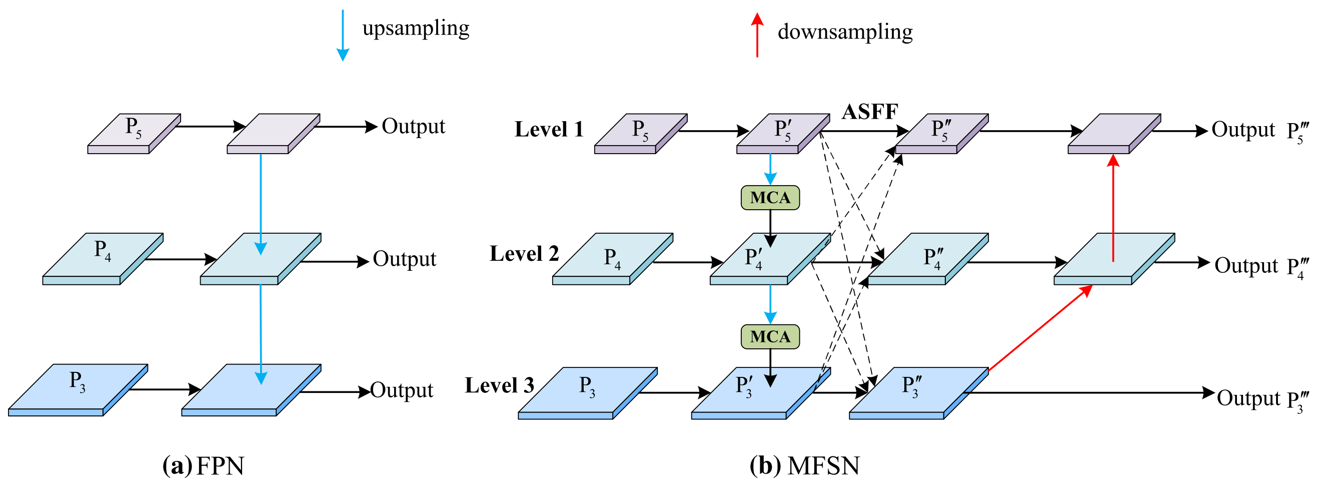


Fig. 4 Structures of FPN and MFSN

outputted by the FEN module and the feature map P_5 outputted by SPP module serve as the input of MFSN. First, upsampling, attention mechanism, concatenation, and other operations are applied by MFSN to obtain the feature maps P_3', P_4', and P_5'. Second, MFSN performs a hierarchical feature fusion of P_3', P_4', and P_5' to obtain P_3'', P_4'', and P_5'' for subsequent spatial information sharing. Finally, MFSN obtains the feature maps P_3''', P_4''', and P_5''' for the RCN module via downsampling, concatenation, and other operations.

The MFSN is superior to the FPN in two main aspects. On the one hand, at the midpoint of the three-layer structure, MFSN adopts ASFF [36] to fuse the feature maps of each layer with the other two layers, thus achieving cross-scale connection and spatial information fusion. On the other hand, MFSN embeds the attention mechanism after the upsampling operation to enhance the channel attention effect for ensuring fine-detection accuracy. As the features of train parts in train fault images are difficult to extract, MCA is proposed on the basis of the fusion of different scale receptive fields and ECA [32]. This scheme enables the MSSIF-Net detector to focus on the fault parts in freight train images.

3.4.1 Basic structure based on FPN

MFSN is based on FPN. The overall structure of FPN is shown in Fig. 4a. In FPN, an addition operation is used for feature fusion after the upsampling. By contrast, in MFSN, a concatenation operation is used to merge the feature channels of the two feature maps after upsampling and downsampling.

3.4.2 MCA

As an important cognitive function of the human brain, attention can selectively process external information, allowing the brain to focus on key information and ignore useless information. In object detection, the importance of different features can be learned by detectors through the attention mechanism. The P_3, P_4, and P_5 is obtained by the FEN and SPP modules, which contain rich semantic information and object location details. However, the feature maps not only involve object feature information but also include other feature information with interference. In FPN, upsampling is employed to magnify the deep feature images. However, upsampling only magnifies the lengths and widths of feature images, but it does not obtain more feature information. As shown in Fig. 4b, the MFSN embeds MCA after the upsampling operation to enable the detector to focus more on the object to be detected. The structure of MCA is shown in Fig. 5.

The MFSN takes the feature map as the input of MCA after the upsampling, as shown in Fig. 5. H, W, and C represent the height, width, and channel number of the feature map, respectively. First, the feature map F, which contains multi-scale feature information, is obtained by the multi-scale module (MS). Second, the channel weight χ is obtained using the feature map F, Global Average Pooling (GAP), fast 1D convolution (Conv1D), and Sigmoid function. Finally, an operation on the element-wise product of the channel weight χ and the original feature map f is performed to obtain the weighted feature map $\tilde{F}$.

The MS module in MCA primarily realizes the extraction of multi-scale feature information. First, the MS module splits the input feature map f into n parts, which are denoted as $[f_0, f_1, \ldots, f_{n-1}]$, according to channel dimension

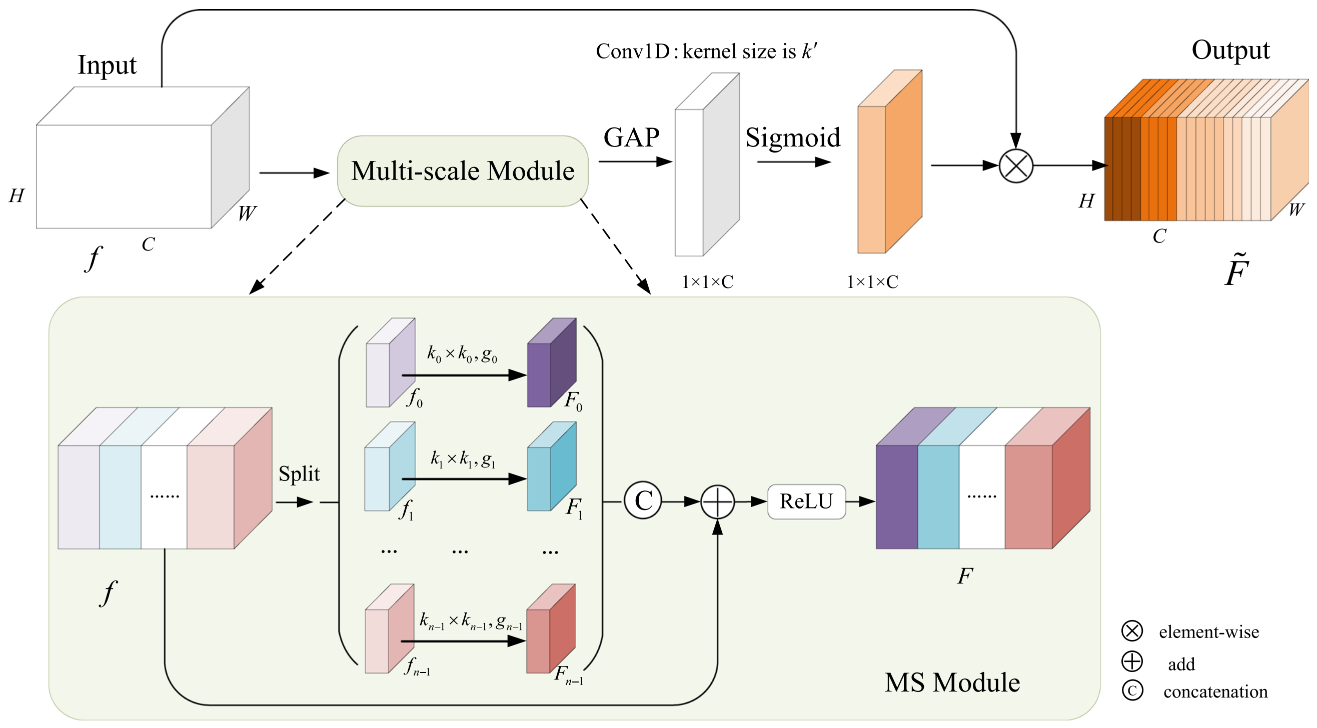


Fig. 5 MCA structure

C. For each feature map $f_i \in R^{H\times W\times C'}$, its channel number is $C' = C/n$, where $i = 0, 1, \ldots, n-1$. Second, the MS module applies Group Convolutions (GCs) with different sizes of convolution kernels in parallel to obtain the feature map F_i containing the multi-scale feature information. Finally, the MS module performs the concatenation operation to merge the results of the GCs, and it exploits a residual edge and the addition operation to achieve feature fusion.

During GC operation, the size of the feature map f_i of each group of convolution input is given by $H \times W \times C'$, in which the size of the corresponding convolution kernel is $k \times k$. The relation between the convolution kernel size k and group size g in a GC is calculated by

$$k = \log_2 g + 1. \tag{1}$$

The feature map F_i with multi-scale information generated by the GC can be expressed as

$$F_i = \text{Conv2D}(f_i), i = 0, 1, 2\ldots, n-1, F_i \in R^{H\times W\times C'}, \tag{2}$$

where k_i is the size of the convolution kernel, and g_i is the group size. The feature map F can be obtained by the concatenation operation, addition operation, and ReLU activation function as follows:

$$F = \text{ReLU}\big(\text{cat}([F_0, F_1, \ldots F_{n-1}]) + f\big), \tag{3}$$

where $[F_0, F_1, \ldots, F_{n-1}]$ is the result of Eq. (2), F_{n-1} is the result of the $(n-1)$-th GC operation, and f is the original feature map.

The feature map F outputted by the MS module serves as the input of GAP, and the output result is F'. F'' is obtained via Conv1D, F' and F'' are calculated by

$$F' = \text{GAP}(F), \tag{4}$$

$$F'' = \text{Conv1D}(F'), \tag{5}$$

where the convolution kernel size of Conv1D is denoted by k'. Here, k' is adaptively selected according to the number of channels in the feature map, which is defined as

$$k' = \left\lfloor \frac{\log_2 C + b}{\gamma} \right\rfloor, \tag{6}$$

where γ and b have the values of 2 and 1, respectively, and C represent the channel numbers. F'' generates the feature channel weight using the Sigmoid function. The feature weight χ can be expressed as

$$\chi = \text{Sigmoid}(F''), \tag{7}$$

where χ has a value in the range of 0 to 1. Finally, feature map f, element-wise product $\otimes$, and feature weight χ are used to yield the weighted feature map $\tilde{F}$, which is calculated by

$$\tilde{F} = \chi \otimes f. \tag{8}$$

3.4.3 Feature fusion in MFSN

MFSN not only performs addition and concatenation operations to fuse features but also exploits the ASFF method to fuse the feature maps of the current layer with those of the other two layers. This method is proposed in Ref. [36] to solve the problem of scale inconsistency among FPNs by learning the weight parameters of different feature layers. By introducing ASFF into MFSN, three feature maps with different scales can be fused with the network. Figure 4b shows the case of Level 2 in which three-layer feature fusion is conducted. First, MFSN adopts the ASFF method to adjust the size of the feature maps P'_5 and P'_3, and the size adjustment of these two feature maps is consistent with the feature map P'_4 in Level 2. Subsequently, feature fusion is implemented according to Eq. (9) as follows:

$$X^l = \alpha^l x^{1\to l} + \beta^l x^{2\to l} + \theta^l x^{3\to l}, \tag{9}$$

where l is the feature layer and α^l, β^l, and θ^l are the adaptive weight parameters learned by this layer. Finally, $x^{n\to l}$ is used to adjust the size of the feature map in feature layer n to that of layer l, where the value of n is $\{1, 2, 3\}$. X^l represents the fusion result of the feature map in the l layer and the feature maps of the other two layers. The fusion result of Level 2 can then be expressed as $P'_4 = X^2 = \alpha^2 x^{1\to 2} + \beta^2 x^{2\to 2} + \theta^2 x^{3\to 2}$.

3.5 RCN

RCN can be divided into two steps. First, YOLO Head (YH) is used to obtain prediction results. Second, the final detection result is achieved after decoding the prediction result output by using the YH module.

In MSSIF-Net, the MFSN outputs the feature maps P''_3 $(52 \times 52 \times 128)$, P''_4 $(26 \times 26 \times 256)$, and P''_5 $(13 \times 13 \times 512)$, which serve as the input of the YH module. First, the YH module exploits the classification number N, confidence, and coordinate parameters of the four position offsets to process P''_3, P''_4, and P''_5 respectively. Then, the output feature maps $\bar{P}_3$ $\left(52 \times 52 \times (N+1+4)\right)$, $\bar{P}_4$ $\left(26 \times 26 \times (N+1+4)\right)$, and $\bar{P}_5$ $\left(13 \times 13 \times (N+1+4)\right)$ are obtained, where 1 indicates that the prior bounding box contains objects. Thus, the confidence score is greater than the set score threshold. In this study, a K-means algorithm is applied to adaptively generate nine prior bounding boxes of different sizes based on the freight train image dataset. Prior bounding boxes are generally used to estimate the size and location of objects. Second, $\bar{P}_3$, $\bar{P}_4$, and $\bar{P}_5$ are divided into 52×52, 26×26, and 13×13 grids according to their width-to-height ratio. The feature map of each scale corresponds to three prior bounding boxes of different sizes, and each bounding box corresponds to a specific object category. In the decoding operation, the center coordinate of the prediction bounding box is calculated using each grid and the corresponding offset of the center point coordinate. Following, the length and width of the prediction bounding box are calculated by combining the length and width of the prior bounding box. Finally, the position of the prediction bounding box is determined using the center point coordinate and the length and width coordinate. In addition, the decoding operation judges each category and filters the prediction bounding box based on the bounding box position, category score, and non-maximum suppression [37] operation, subsequently drawing the filtered prediction bounding box in the corresponding position of the original image. The detection result is obtained in the end.

3.6 Loss function

The loss function of the MSSIF-Net detector involves three parts, namely regression loss L_{ciou}, confidence loss L_{conf}, and classification loss L_{cls}.

The regression loss represents the error between the ground-truth bounding box and the prediction bounding box. MSSIF-Net adopts the Complete Intersection over Union Loss (CIoU Loss) denoted by L_{ciou}, which is calculated by

$$\begin{aligned} L_{ciou} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \Bigg[1 - \mathrm{IoU} + \frac{d^2(b, b^{gt})}{l^2} \\ + \frac{16}{\pi^4} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2 \\ \times \left(1 - \mathrm{IoU} + \frac{4}{\pi^2} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right) \right)^{-1} \Bigg], \end{aligned} \tag{10}$$

where S^2 represents the grid to be divided into the image and B represents the prediction bounding box. I_{ij}^{obj} is usually set to 0 or 1, in which the value is used to judge whether an object exists in the i-th grid and whether it can be correctly predicted by the j-th prediction bounding box. IoU refers to the ratio of intersection and union of ground-truth bounding box A and prediction bounding box B, and it represents the accuracy of the prediction bounding box. IoU is defined as

$$\text{IoU} = \frac{A \cap B}{A \cup B}. \quad (11)$$

Furthermore, $d^2(b, b^{gt})$ represents the Euclidean distance between the center point of the prediction bounding box and the ground-truth bounding box, and l is the diagonal distance of the smallest enclosing rectangle formed by the ground-truth bounding box and the prediction bounding box. w, h, w^{gt}, and h^{gt} represent the width and height of the prediction bounding box and the width and height of the ground-truth bounding box, respectively.

The confidence loss adopts the binary cross entropy loss, which is divided into two parts, namely the confidence loss items of the bounding box with objects and without an object. L_{conf} is calculated by

$$\begin{aligned} L_{conf} = &- \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i^j \log(C_i^j) + (1 - \hat{C}_i^j) \log(1 - C_i^j) \right] \\ &- \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{noobj} \\ &\left[\hat{C}_i^j \log(C_i^j) + (1 - \hat{C}_i^j) \log(1 - C_i^j) \right], \end{aligned} \quad (12)$$

where $C_i^j = P(object) \times \text{IoU}_{pred}^{truth}$ is the confidence of the prediction bounding box, which is calculated on the basis of the product of the category probability $P(object)$ of the object and IoU. $\hat{C}_i^j$ is the confidence of the ground-truth bounding box, which has a value of 0 or 1. λ_{noobj} and I_{ij}^{noobj} are the hyperparameters used to calculate the confidence loss when the bounding box does not contain an object. In general, λ_{noobj} has a value of 0.5, I_{ij}^{noobj} has a value of 0 or 1, and the value of I_{ij}^{noobj} is opposite to that of I_{ij}^{obj}.

The classification loss L_{cls} can be expressed as

$$L_{cls} = - \sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in classes}^{c} [\hat{P}_i^j \log(P_i^j) + (1 - \hat{P}_i^j) \log(1 - P_i^j)], \quad (13)$$

where $\hat{P}_i^j$ and P_i^j are the category probability of the ground-truth bounding box and the prediction bounding box, respectively.

In summary, the loss of MSSIF-Net can be calculated by

$$Loss = L_{ciou} + L_{conf} + L_{cls}. \quad (14)$$

4 Experiment and analysis

The implementation details of MSSIF-Net, the performance evaluation metrics used in the experiments, and the ablation experiments and robustness tests of MSSIF-Net on TITS are discussed in this section. The state-of-the-art detectors and MSSIF-Net test results on the TITS and PASCAL VOC 2007 test set are also presented.

MSSIF-Net is implemented on the PyTorch deep learning framework. The operating system used in the experiment is Ubuntu 20.04 with a running memory of 64 GB. An NVIDIA GeForce GTX 1080 Ti graphics card is used to accelerate the calculation. The versions of CUDA and cuDNN are both 11.2.

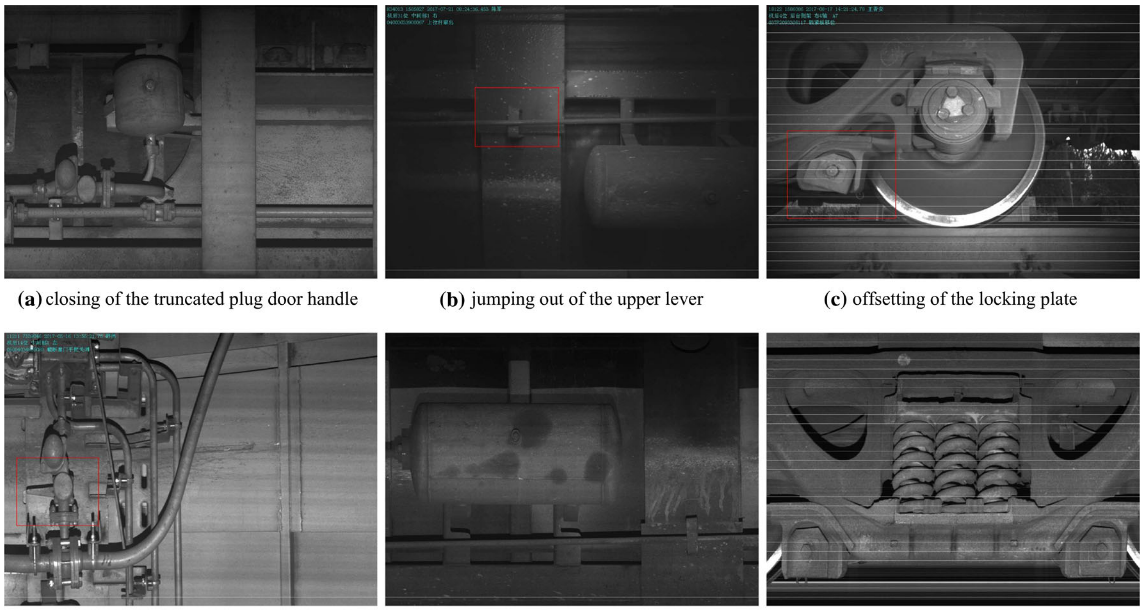
4.1 Implementation details

We describe in this subsection the training and test datasets and the skills utilized in the training.

4.1.1 Dataset description

The train image dataset used in this study is randomly selected from the high-definition train images collected by the TFDS of China Railway Guangzhou Group Company Limited. The LabelImg tool is used to organize the dataset into the PASCAL VOC format. As shown in Fig. 6, the labels of the train image dataset can be divided into four categories according to the train fault type, namely truncated plug door handle (TPDH), upper lever (UL), locking plate (LP), and normal, corresponding to the closing of the truncated plug door handle, the jumping out of the upper lever, the offsetting of the locking plate, and the above-mentioned three train parts without fault, respectively. The train image dataset contains 11,936 freight train images, among which the number of images in TITS is 3580 (i.e., TITS3580). Owing to the influence of factors, such as weather and camera angle, the freight train images in the actual scene captured by the TFDS by using high-speed cameras may have slight deformations and distortions. Outdoor train images are more realistically simulated by TITS3580 when producing a new robustness test set (RTS3580). Thus, RTS3580 is used to test the robustness of MSSIF-Net. The robustness test includes image-rotating and artificially adding noise.

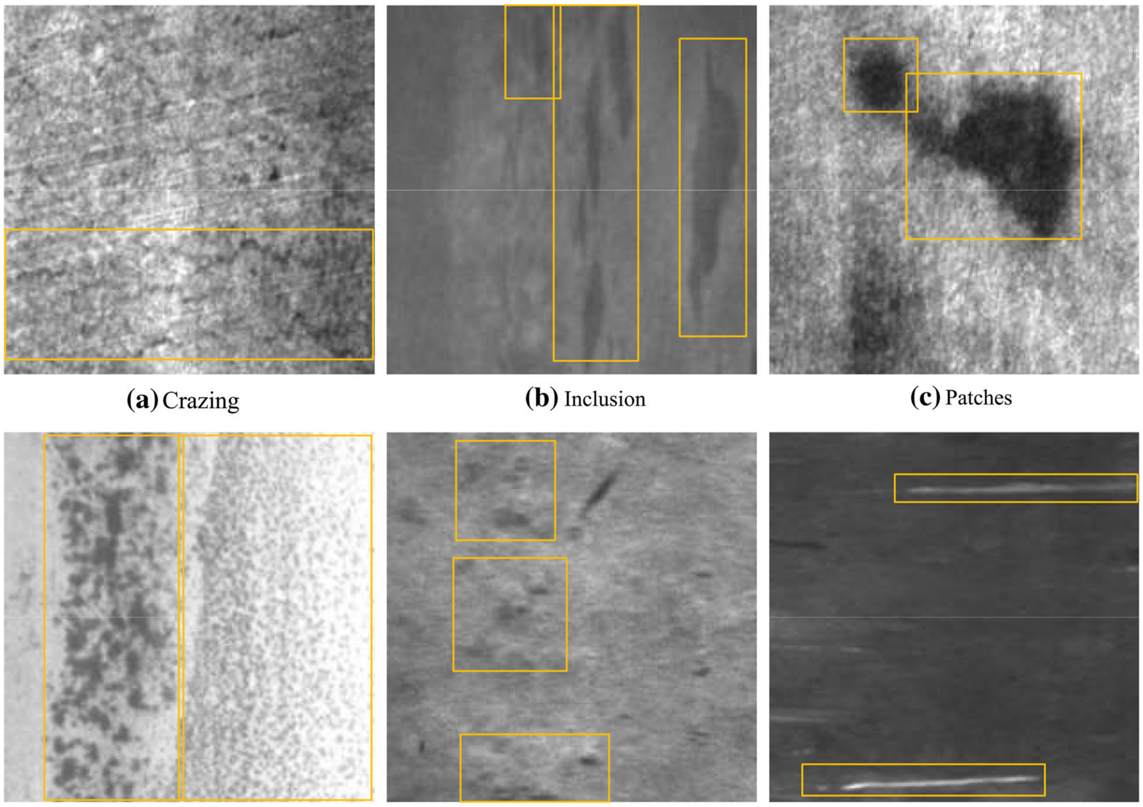
PASCAL VOC 2007 and PASCAL VOC 2012 are used as the training set. In particular, PASCAL VOC 2007 is used as the test set to further evaluate the performance of MSSIF-Net and other well-known detectors. The PASCAL VOC 2007 and PASCAL VOC 2012 datasets divide the objects in an image into the following 20 categories: *aeroplane*, *bicycle*, *bird*, *boat*, *bottle*, *bus*, *car*, *cat*, *chair*,



(a) closing of the truncated plug door handle **(b)** jumping out of the upper lever **(c)** offsetting of the locking plate

(d) Normal **(e)** Normal **(f)** Normal

Fig. 6 Examples of TITS



(a) Crazing **(b)** Inclusion **(c)** Patches

(d) Pitted surface **(e)** Rolled-in scale **(f)** Scratches

Fig. 7 Image of steel surface defect of NEU-DET dataset (Defects to be detected are in the orange box)

cow, *diningtable*, *dog*, *horses*, *motorbike*, *person*, *potted-plant*, *sheep*, *sofa*, *train*, and *tvmonitor*. The number of images in the training set is 16,551, whereas that in the test set is 4952. For simplicity, the test set is named Test4952. As the aforementioned 20 categories have almost no similarity with those in the train faults, the 20 categories in

Test4952 are also tested in the experiment to achieve a comprehensive evaluation of MSSIF-Net performance.

In addition, a dataset for surface defect detection of hot-rolled steel sheets (NEU-DET dataset) [38–40] is used to evaluate the performance of different detector models. As shown in Fig. 7, the defects to be detected include six categories in the NEU-DET dataset, namely crazing, inclusion, patches, pitted surface, rolled-in scale, and scratches. There are 300 images in each category, and each image to be detected has more than one possible number of defects. In addition, more than 5000 ground truth boxes are marked in the dataset, among which 1260 images are used as NEU-DET training dataset for model pre-training and network fine-tuning. Moreover, 540 images are used as NEU-DET testing dataset (NDTest540) for performance testing.

4.1.2 Training skills

MSSIF-Net mainly exploits the idea of transfer learning during training. Deep CNNs perform well in object detection. However, excellent detection results can only be achieved if numerous data are used for CNN training. In practical applications, the number of train fault images is far from being sufficient in relation to the large amount of train data required by detectors; thus, training a new CNN method is inefficient. Under normal circumstances, the idea of transfer learning can be adopted to reduce the training time and training data to a certain extent, thereby improving training efficiency and detector performance. An illustration of this configuration is shown in Fig. 8. The solid blue line is taken as the train path of the detectors, whereas the red dashed line is taken as the test path of the train images. First, the weight file obtained by FEN trained on the ImageNet dataset is used as the pre-training weight, and then the MSSIF-Net detector is trained using the train dataset and the pre-training weight. Second, the weights generated via detector training are used as the input weights of the test. In this manner, the final object detection result can be collected using the testing weights, test set, and the presented MSSIF-Net. The detectors compared in our experiments also utilize the same training skills.

The total number of iterations of the MSSIF-Net training is 200, which is divided into two stages, namely freezing training and thawing training. In the freezing training stage, the backbone of the detector does not change but only fine-tunes the network. The number of freezing training is 100, the learning rate is 0.001, and the batch size is 8. In the thawing training stage, the backbone of the detector is thawed, and the feature extraction model is changed. The learning rate of the thawing training is 0.0001, and the batch size is 16. The confidence score is 0.5.

4.2 Evaluation metrics

The performance evaluation metrics of the object detection adopted in this study include average precision (AP), mean average precision (mAP), frame per second (FPS), and log-average miss rate (LAMR).

4.2.1 AP and mAP

AP refers to the area under the precision-recall (*P*-*R*) curve. The *P*-*R* curve represents the trade-off between precision and recall on a classifier. In general, the higher the AP, the better the performance of the object detector. AP is calculated by

$$\mathrm{AP} = \int_0^1 P(r)dr, \tag{15}$$

where $P(r)$ denotes the *P*-*R* curve. We assume that the number of positive samples correctly identified as positive samples by the detector is a, the number of negative samples incorrectly detected as positive samples is b, and the number of positive samples incorrectly identified as negative samples is c, i.e., $P = a/(a+b)$, $R = a/(a+c)$, and $P(r) = P \times R$. On the TITS dataset, the APs of TPDH,

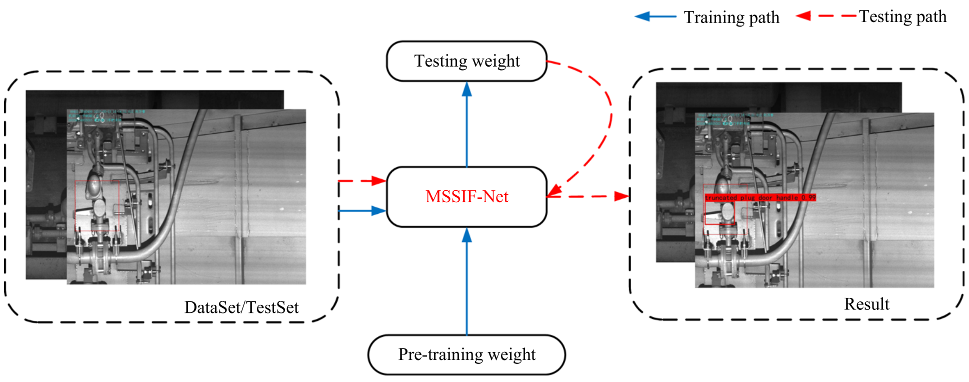


Fig. 8 Detector training and testing process

UL, LP, and normal are detected, as denoted by $\mathrm{AP_T}$, $\mathrm{AP_U}$, $\mathrm{AP_L}$, and $\mathrm{AP_N}$, respectively.

mAP is the average value of AP for each category, which represents the average effect of detection of all object categories. It can be calculated as

$$\mathrm{mAP} = \frac{\sum_{\beta=0}^{n} \mathrm{AP}_\beta}{n}, \tag{16}$$

where n is the total number of categories andth AP_β is the AP value of the β-th category.

4.2.2 FPS

FPS is the number of frames transmitted per second, and it is used to evaluate the detection speed of detectors. A high FPS indicates a high detection speed of the detector. FPS is estimated by

$$\mathrm{FPS} = \frac{frameNum}{elapsedTime}, \tag{17}$$

where $frameNum$ is the number of frames and $elapsedTime$ is the consumed time. In this study, the fixed-time frame method is adopted to calculate FPS. The fixed time is set to 1 second.

4.2.3 LAMR

LAMR refers to the probability of objects being undetected, which can be expressed as

$$\mathrm{LAMR} = \exp\left[\frac{1}{n}\left(\sum_{i=1}^{n} \log_2 a_i\right)\right], \quad a_i > 0, \tag{18}$$

where a_i is the probability of missing points in the n uniformly distributed false-positive per image. The lower the LAMR value, the better the detection effect of the detector.

4.3 Ablation experiments on TITS

Ablation experiments are conducted to verify whether the detector implemented with both MCA and MFSN is more conducive to the improvement of detection accuracy. Experimental detectors include the following types: (1) original YOLOv4, (2) MCA-YOLO embedded with attention mechanism MCA, (3) MFSN-YOLO that use MFSN without embedding MCA, and (4) MSSIF-Net that uses MFSN and embedded with MCA. The experiments are implemented under the same conditions, and the embedded position of MCA is either in PANet or MFSN. Both MFSN and PANet are the unembedded attention mechanism type, as shown in Table 1.

As can be seen from Table 1, the detection accuracy of MCA-YOLO, MFSN-YOLO, and MSSIF-Net is higher than that of original YOLOv4, but the detection speed is slightly decreased. The effectiveness of MCA and MFSN is verified by MCA-YOLO and MFSN-YOLO, respectively. Compared MCA-YOLO with YOLOv4, MCA, as a channel attention mechanism, can effectively increase the detector's attention to the object region, enhance the ability of feature representation, provide more useful feature information for subsequent classification and regression tasks. In this case, mAP is increased from 89.29% to 92.01%. For MCA, group convolution operations with different convolution kernel sizes are used to obtain more feature information of different scales. However, as the computational amount of group convolution is lower than that of ordinary convolution, the detection speed of MCA-YOLO is almost unaffected when MCA is used alone, reaching 35.28 FPS. For MFSN-YOLO, MFSN is used to replace PANet in YOLOv4, and the effect of feature extraction is further enhanced. Moreover, each layer in the middle part of MFSN fuses the feature information of the other two layers. The amount of feature information captured by the image channel is greatly increased, and the mAP of MFSN-YOLO reaches 91.86%. MCA and MFSN are adopted in MSSIF-Net at the same time, and the semantic information of feature maps at different scales is enriched. In addition, owing to the increased computational complexity of MSSIF-Net, the final detection speed is 33.10 FPS, and mAP achieves 94.73%.

The visualization results of MSSIF-Net are shown in Fig. 9, in which Fig. 9a is the original train image. The class activation mapping (CAM) [41] method is used to generate the visualization feature maps of Figs. 9b–d for the confidence score, classification score, and confidence score weighted with the classification score, respectively.

Table 1 Ablation experiments on TITS3580

Methods	MCA	PANet	MFSN	mAP(%)	$\mathrm{AP_T}$(%)	$\mathrm{AP_U}$(%)	$\mathrm{AP_L}$(%)	$\mathrm{AP_N}$(%)	FPS
YOLOv4	–	√	–	89.29	100.00	93.18	74.26	89.72	**35.35**
MCA-YOLO	√	√	–	92.01	100.00	94.60	78.53	**94.93**	35.28
MFSN-YOLO	–	–	√	91.86	100.00	93.34	78.60	94.48	31.48
MSSIF-Net	√	–	√	**94.73**	**100.00**	**98.60**	**88.87**	91.45	33.10

Bold values indicate the best results obtained by the detector model in terms of current evaluation metric

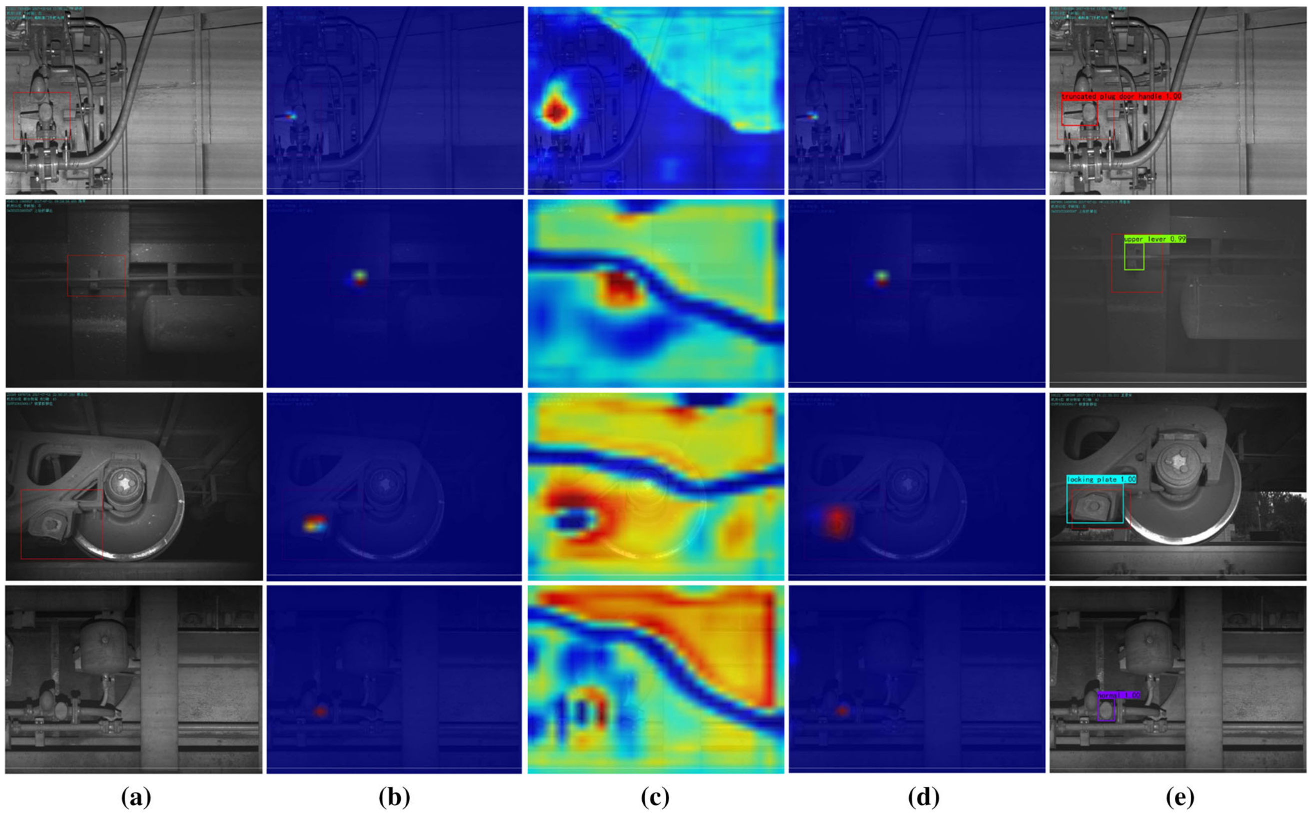


Fig. 9 Visualization examples of MSSIF-Net for train image fault detection

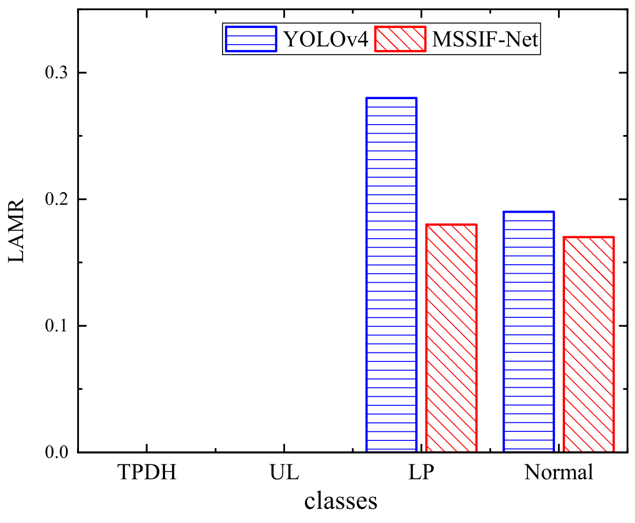


Fig. 10 LAMR results of YOLOv4 and MSSIF-Net

Figure 10 shows the LAMR comparisons of YOLOv4 and MSSIF-Net. The miss rates of LP and normal are low in MSSIF-Net.

4.4 Comparisons with other approaches on TITS

To further verify the performance of MSSIF-Net, one-stage detectors (YOLOv3, YOLOv4 and RetinaNet), two-stage detector (Faster R-CNN), and anchor-free detectors (CenterNet and YOLOX-l) are compared in the TITS (TITS3580). Two-stage detectors generally adopt enumeration methods to generate and filter region proposals in RPN, so their detection speed is slow. The regression and classification of objects in the anchor-free detector rely on key point estimation which usually has fast detection speed. One-stage detector performs regression directly on the bounding box and category probability of the object, which can generally balance detection accuracy and speed well by using prediction bounding box. The experimental results of TITS3580 are reported in Table 2, in which the second column is mAP, AP_T, AP_U, AP_L, and AP_N correspond to AP values of the four train parts fault types. The last column is the detection speed. Given the same experimental conditions, the comprehensive performance of MSSIF-Net is superior to those of the other detectors.

Table 2 shows that anchor-free detectors perform well in terms of FPS. For example, the detection speed of CenterNet can reach 55.47 FPS. One-stage detectors can reach a balance between detection accuracy and speed. For example, YOLOv3 achieves a mAP of 86.01% on TITS3580, and the detection speed can reach 36.35 FPS. Regardless whether one-stage or anchor-free detectors, the detection accuracy is improved at the cost of increasing a

Table 2 The experimental results of different detection methods on TITS3580

Methods	mAP(%)	AP_T(%)	AP_U(%)	AP_L(%)	AP_N(%)	FPS
YOLOv3 [14]	86.01	90.48	83.19	79.33	91.03	36.35
YOLOv4 [15]	89.29	**100.00**	93.18	74.26	89.72	35.35
Faster R-CNN [9]	86.98	88.64	89.97	82.74	86.58	13.08
RetinaNet [17]	81.48	73.11	86.39	76.68	89.75	25.36
CenterNet [22]	81.54	77.51	75.54	84.46	88.63	**55.47**
YOLOX-l [23]	91.73	**100.00**	97.73	75.33	**93.88**	23.23
MSSIF-Net	**94.73**	**100.00**	**98.60**	**88.87**	91.45	33.10

Bold values indicate the best results obtained by the detector model in terms of current evaluation metric

certain amount of computation, such as RetinaNet, YOLOv4, and YOLOX-l. The mAP of YOLOX-l can reach 91.73%. The Faster R-CNN of the two stage detector can obtain a mAP of 86.98%, but its detection speed is lower than the other two types of detectors.

The overall performance improvement values of MSSIF-Net on TITS are 8.72%, 5.44%, 7.75%, 13.25%, 13.19%, and 3.00% better than those of YOLOv3, YOLOv4, Faster R-CNN, RetinaNet, CenterNet, and YOLOX-l in mAP, respectively. As train images are grayscale images, feature extraction is difficult. The MSSIF-Net proposed in this study belongs to the one-stage detector. As more feature information of different scales is obtained, the comprehensive performance of MSSIF-Net better than other detectors. In particularly, MCA in MSSIF-Net is used to increase the attention of the target fault object to be detected, and MFSN is used to generate multi-scale feature maps with rich semantic information. Therefore, it can achieve excellent result in the detection accuracy of any train fault category, and the detection speed can reach 33.10 FPS. The overall trend of the detection accuracy and speed results indicate MSSIF-Net is superior to the other detectors on TITS3580. It can also balance detection accuracy and speed.

4.5 Robustness tests on TITS

Aiming to further verify the anti-interference ability of MSSIF-Net, robustness tests are conducted by adding noise and rotating train images to RTS3580. After adding the interference in the TITS, the AP and mAP of YOLOv4 and MSSIF-Net are compared to further verify the reliability of the MSSIF-Net detector.

4.5.1 Robustness to noise

Noise is also manually added to the images on RTS3580. The three types of noise are random noise, Gaussian noise, and salt and pepper noise.

The experimental results of the noise robustness tests of YOLOv4 and MSSIF-Net are shown in Table 3. For random noise, MSSIF-Net clearly surpasses YOLOv4 by 3.00% in mAP and by −0.79%, 1.48%, 12.51%, and −0.91% in AP_T, AP_U, AP_L, and AP_N, respectively. These results indicate that random noise has almost no effect on the train image, further suggesting that MSSIF-Net maintaines its high accuracy. For Gaussian noise, the mAP of MSSIF-Net is slightly decreased. With respect to AP_T, AP_U, AP_L, and AP_N, the MSSIF-Net has a marginally lower performance (3.70%, 8.06%, −5.01%, and −0.91%, respectively) compared with YOLOv4. However,

Table 3 Noise test results on RTS3580

Methods	mAP(%)		AP_T(%)		AP_U(%)		AP_L(%)		AP_N(%)	
	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net
None	89.29	94.73	100.00	100.00	93.18	98.60	74.26	88.87	89.72	91.45
Random noise	91.96	94.96	96.92	95.83	91.73	93.21	83.59	96.10	95.61	94.70
Gaussian noise	88.46	87.31	98.15	95.45	70.71	62.65	89.33	94.34	95.65	96.81
Salt and pepper noise	88.48	89.21	97.31	92.58	69.98	75.29	90.95	94.24	95.69	94.70

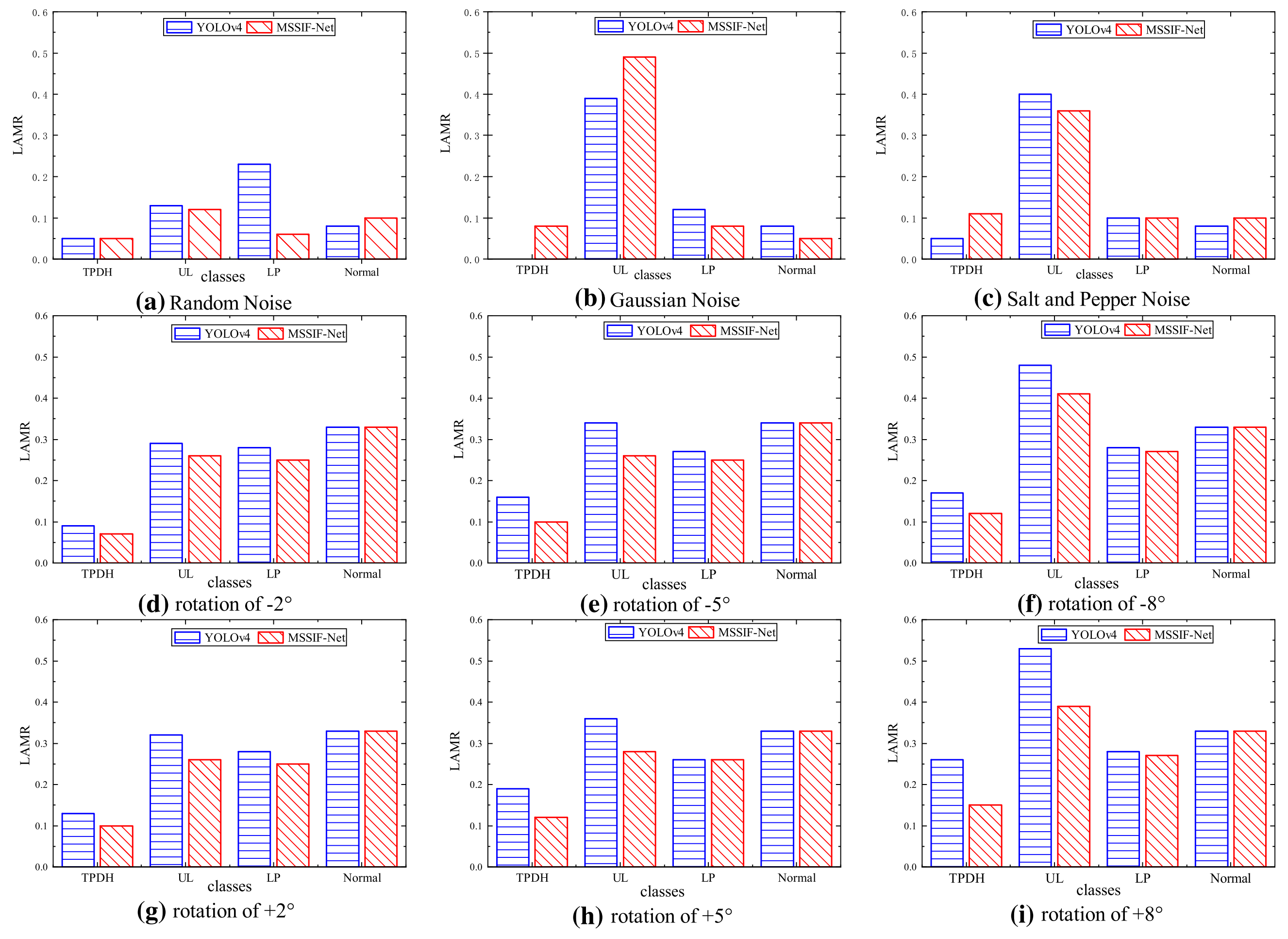


Fig. 11 LAMR results for robustness tests on RTS3580

MSSIF-Net has shown good anti-interference ability for LP and normal detection in the Gaussian noise experiment. For the salt and pepper noise, the performance improvement of MSSIF-Net is 0.73% better than YOLOv4 in mAP. For AP_T, AP_U, AP_L, and AP_N, the performance improvement of MSSIF-Net is −4.73%, 5.31%, 3.29%, and −0.96%, respectively, compared with that of YOLOv4. The trend can be attributed to Gaussian noise and salt and pepper noise greatly influencing the train image quality and interfere with the object features details.

Figures 11a–c shows the increasing LAMRs of MSSIF-Net and YOLOv4 for the three types of noise, especially the UL. Under the interference of noise, the detection accuracy of MSSIF-Net is affected to a certain extent. Nonetheless, the MSSIF-Net continues to have high-

Table 4 Rotation test results of RTS3580

Methods	mAP(%)		AP_T(%)		AP_U(%)		AP_L(%)		AP_N(%)	
	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net	YOLOv4	MSSIF-Net
−2°	83.24	84.41	92.34	95.45	81.56	82.45	81.70	82.07	77.36	77.66
−5°	80.85	82.95	86.37	91.46	78.18	80.56	81.72	82.72	77.14	77.05
−8°	77.13	79.66	84.80	89.76	64.85	70.82	81.34	80.68	77.55	77.41
+2°	81.53	83.92	88.20	93.48	79.04	82.45	81.13	82.07	77.75	77.66
+5°	79.73	82.70	83.57	91.08	75.80	79.96	82.21	82.40	77.35	77.36
+8°	74.54	80.25	80.11	89.98	59.04	71.71	81.73	81.94	77.16	77.37

Table 5 The experimental results of different detection methods on Test4952

Methods	Backbone	Input Size	mAP(%)	FPS
YOLOv4 [15]	CSPDarkNet53	416×416	82.06	33.76
Faster R-CNN [9]	ResNet50	600×600	84.29	13.30
RetinaNet [17]	ResNet50	600×600	82.46	17.03
CenterNet [22]	ResNet50	512×512	83.13	**55.30**
YOLOX-l [23]	CSPDarkNet	640×640	86.46	25.13
MSSIF-Net	CSPDarkNet53	416×416	**87.76**	29.89

Bold values indicate the best results obtained by the detector model in terms of current evaluation metric

detection accuracy on the average, further suggesting its good anti-interference ability.

4.5.2 Robustness to rotation

For this phase of the robustness test, the images in RTS3580 are rotated counterclockwise (−2°, −5°, and −8°) and clockwise (+2°, +5°, and +8°).

The experimental results of the image rotation test are shown in Table 4. For the image rotation of −2°, −5°, −8°, +2°, +5°, and +8° on RTS3580, the MSSIF-Net exceeds YOLOv4 as follows: by 1.17%, 2.1%, 2.53%, 2.39%, 2.97%, and 5.71% in mAP, respectively; by 3.11%, 50.9%, 4.96%, 5.28%, 7.51%, and 9.87% in AP_T, respectively; by 0.98%, 2.38%, 6.01%, 3.41%, 4.16%, and 13.13% in AP_U, respectively; by 0.37%, 1.00%, −0.66%, 0.94%, 0.19%, and 0.21% in AP_L, respectively; and by 0.30%, −0.09%, −0.14%, −0.09%, 0.01%, and 0.21% in AP_N, respectively. As the angle of rotation increases, the accuracy of the detector decreases. We suspect that the rotated train images manifest as different features, affecting the judgment of the detectors. However, the MFSN uses the ASFF method to adaptively fuse the three-layer spatial features. The MSSIF-Net is expected to be more reliable and still offer excellent detection accuracy.

As shown in Figs. 11d–i, MSSIF-Net outperforms YOLOv4 in terms of LAMR. Image rotation has a great impact on all categories of LAMR. However, for MSSIF-Net, the LAMR of the four fault types is generally lower than those of YOLOv4. Although detection accuracy is affected greatly by the image rotation, MSSIF-Net continues to show great detection performance and robustness.

4.6 Experiments on PASCAL VOC 2007

MSSIF-Net is also compared with the other five well-known detectors by using the PASCAL VOC 2007 test set of Test4952. The experimental results are shown in Tables 5 and 6. MSSIF-Net has the best comprehensive performance in terms of accuracy and speed. In Table 5, each column from left to right is detector, feature extraction backbone, input image size, mAP, and FPS. Table 6

Table 6 Detection precision results of single category on Test4952

AP(%)	YOLOv4	Faster R-CNN	RetinaNet	CenterNet	YOLOX-l	MSSIF-Net
Aaeroplane	91.69	81.56	63.62	82.08	**94.23**	92.33
Bicycle	90.16	91.17	86.86	90.78	**91.88**	91.23
Bird	82.23	83.10	71.94	80.91	85.32	**88.61**
Boat	72.91	81.37	70.01	78.73	**81.66**	79.68
Bottle	71.80	66.72	69.00	61.64	**81.80**	**81.80**
Bus	92.49	92.71	90.06	86.56	**93.80**	93.23
Car	91.51	91.35	90.17	84.46	95.00	**95.02**
Cat	88.46	92.92	**96.90**	94.77	90.64	91.06
Chair	63.58	69.97	58.95	66.22	74.21	**76.69**
Cow	86.16	91.78	88.26	87.94	88.76	**95.11**
Diningtable	66.71	77.36	**86.74**	69.72	81.34	80.25
Dog	89.05	91.84	**95.70**	91.64	87.92	92.54
Horses	92.00	92.66	95.30	88.75	91.23	**95.36**
Motorbike	91.50	89.40	**95.57**	88.93	89.99	93.32
Person	89.76	87.80	84.48	83.13	**92.13**	92.00
Pottedplant	58.80	61.30	58.21	59.12	62.55	**64.61**
Sheep	79.88	**89.35**	88.41	87.45	86.82	88.54
Sofa	75.48	80.58	**84.80**	81.74	83.94	82.11
Train	86.38	91.04	92.44	90.95	90.10	**93.25**
Tvmonitor	80.73	81.86	85.07	87.02	85.80	**88.45**

Bold values indicate the best results obtained by the detector model in terms of current evaluation metric

Table 7 Experimental results of different detection methods on NDTest540

Mothods	mAP(%)	AP(%)						
		Crazing	Inclusion	Patches	Pitted surface	Rolled-in scale	Scratches	FPS
YOLOv4 [14]	72.44	39.38	79.09	88.30	75.06	60.37	92.19	34.56
Faster R-CNN [9]	73.22	44.82	71.63	95.30	65.85	71.67	90.05	13.14
RetinaNet [17]	64.32	39.92	75.40	90.49	72.25	72.65	35.18	28.03
CenterNet [22]	71.58	33.52	82.45	91.31	65.55	65.04	91.58	59.36
YOLOX-l [23]	73.75	34.85	80.77	92.17	70.50	73.66	90.52	21.16
MSSIF-Net	75.54	45.26	81.16	91.73	76.89	65.85	92.38	30.14

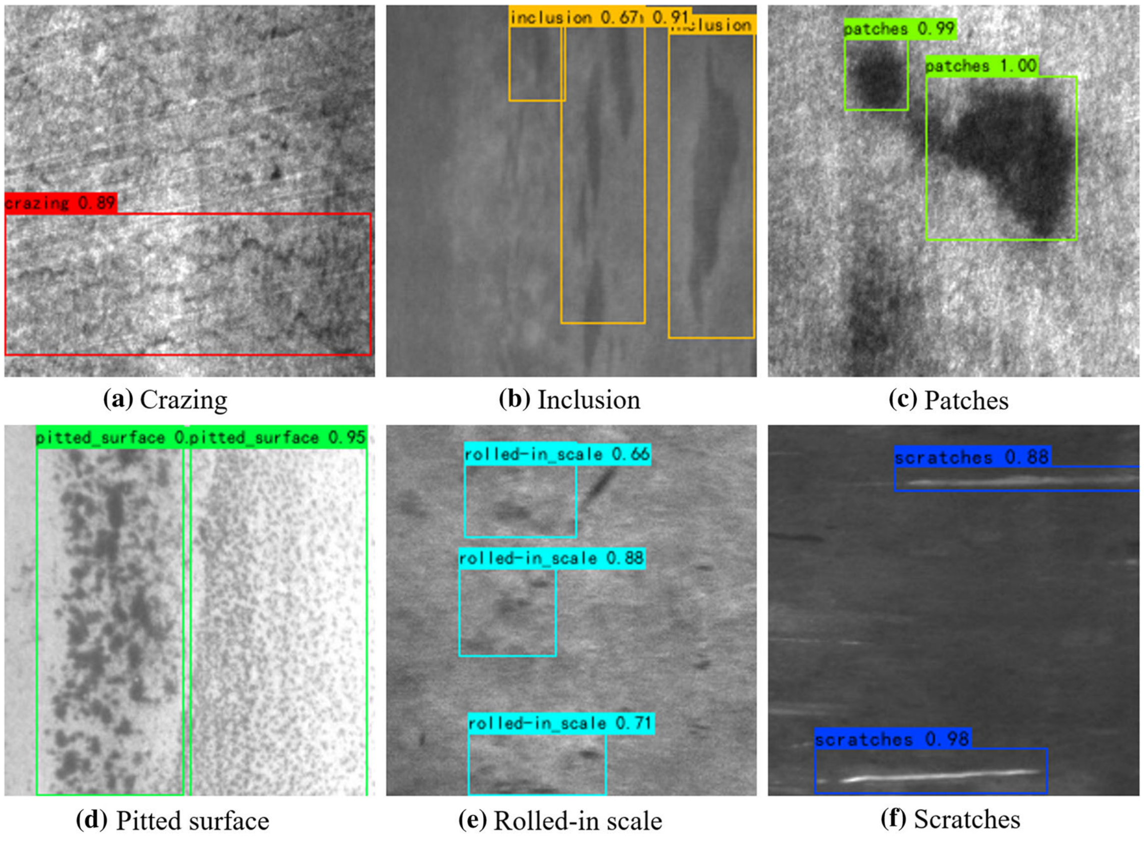


Fig. 12 Results of surface defect detection

compares the AP values of 20 categories in Test4952 between MSSIF-Net and other five detectors.

In mAP, the overall performance improvement of MSSIF-Net on Test4952 is 5.70%, 3.47%, 5.00%, 4.63%, and 1.30% better than those of YOLOv4, Faster R-CNN, RetinaNet, CenterNet, and YOLOX-l, respectively. The detection speed of MSSIF-Net is 29.89 FPS on Test4952, which is higher than Faster R-CNN, RetinaNet, and YOLOX-l. Two tricks (MCA and MFSN) of increasing feature extraction are used in MSSIF-Net, which effectively improve the ability of the detector model to obtain effective feature information. Thus, the detection performance evaluation of MSSIF-Net on the Test4952 dataset is better than other detectors. In particularly, with the multi-scale prediction feature maps generated by MFSN and the improvement of MCA's focus on small objects, the detection accuracy of MSSIF-Net for small objects, such as bottle and chair, are greatly improved, reaching APs of 81.80% and 76.69%, respectively. These trends depict the good detection of MSSIF-Net on small objects.

4.7 Experiments on NEU-DET dataset

The NEU-DET training dataset and NDTest540 are used to train and evaluate MSSIF-Net, respectively. As all the images to be detected from the NEU-DET dataset are grayscale images, and it is not easy to detect defects in the images to be detected. The feature extraction of defects is hard. As shown in Table 7, performance comparison is made between MSSIF-Net and two-stage object detector (Faster R-CNN), one-stage object detectors (YOLOv4 and RetinaNet), and anchor-free detectors (CenterNet and YOLOX-l). MSSIF-Net can still maintain excellent detection performance on grayscale images that are difficult to be detected. The mAP of MSSIF-Net can reach 75.54% and detection speed can reach 30.14 FPS. Compared with other detectors, MSSIF-Net has excellent detection performance and can balance detection accuracy and speed. The detection results of MSSIF-Net are shown in Fig. 12, which can accurately detect the defect part.

5 Conclusion

We propose in this study a train fault image detection method called the MSSIF-Net based on MCA and MFSN to improve the detection accuracy of freight train images. MSSIF-Net encompasses the FEN, SPP, MFSN, and RCN modules. The FEN module extracts image feature information, the SPP module expands the receptive field of the high-level feature map, the MFSN module fuses multi-scale feature information, and the RCN module classifies and regresses the prediction results. MSSIF-Net performs well on the TITS and PASCAL VOC 2007 test set. The overall performance improvement of MSSIF-Net on TITS is 5.44%, 7.75%, 13.25%, 13.19%, and 3.00% better than those of YOLOv4, Faster R-CNN, RetinaNet, CenterNet, and YOLOX-l, respectively, in mAP. On the VOC 2007 test set, the corresponding mAP values of MSSIF-Net are higher by 5.7%, 3.47%, 4.63%, 5.3%, and 1.3% than those of the other five object detection methods. To test the generalization ability of MSSIF-Net, NEU-DET dataset is used to train the detector and evaluate its performance. MSSIF-Net achieves a mAP of 75.54%. In addition, the train image is simulated under real conditions, and the robustness of MSSIF-Net on noisy or rotated train image datasets is tested. Extensive experimental results demonstrate that the proposed MSSIF-Net method can significantly surpass the other detection methods in terms of accuracy and stability. In the next plan of work, we intend to promote the object detection speed while maintaining the original detection accuracy.

Acknowledgements The authors would like to thank the anonymous reviewers for their comments and suggestions to improve the manuscript. This work was partially funded by the National Key R &D Program of China (Grant No. 2018YFB1003401), the National Natural Science Foundation of China (Grant Nos. 61702178, 62072172, 62002115), the Natural Science Foundation of Hunan Province (Grant Nos. 2019JJ50123, 2019JJ60054), the Research Foundation of Education Bureau of Hunan Province (Grant Nos. 20C0625, 18C0528, 19B321), the Key R &D Program of Hunan Province (Grant No. 2021NK2020), and in part by China Scholarship Council (Grant No. 201808430297).

Data availability The PASCAL VOC 2007 and 2012 datasets that support the findings of this study are available in/from hyperlink to data source "https://host.robots.ox.ac.uk/pascal/VOC". The NEU-DET dataset used in this study is available with the identifier "https://doi.org/10.1109/TIM.2019.2915404" [39].

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Sun J, Xiao Z, Xie Y (2017) Automatic multi-fault recognition in tfds based on convolutional neural network. Neurocomputing 222:127–136
2. Fu X, Li K, Liu J, Li K, Zeng Z, Chen C (2020) A two-stage attention aware method for train bearing shed oil inspection based on convolutional neural networks. Neurocomputing 380:212–224
3. De Bruin T, Verbert K, Babuška R (2016) Railway track circuit fault diagnosis using recurrent neural networks. IEEE Trans Neural Netw Learn Syst 28(3):523–533
4. Zhang Y, Liu M, Yang Y, Guo Y, Zhang H (2021) A unified light framework for real-time fault detection of freight train images. IEEE Trans Industr Inf 17(11):7423–7432
5. Leng J, Liu Y (2021) Single-shot augmentation detector for object detection. Neural Comput Appl 33(8):3583–3596
6. Gao F, Ji S, Guo J, Li Q, Ji Y, Liu Y, Feng S, Wei H, Wang N, Yang B (2021) Id-net: an improved mask r-cnn model for intrusion detection under power grid surveillance. Neural Comput Appl 1–17 (2021)
7. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587
8. Girshick R (2015) Fast r-cnn. In: Proceedings of The IEEE International Conference on Computer Vision, pp. 1440–1448
9. Ren S, He K, Girshick R, Sun J (2016) Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
10. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of The IEEE International Conference on Computer Vision, pp. 2961–2969
11. Uijlings JR, Van De Sande KE, Gevers T, Smeulders AW (2013) Selective search for object recognition. Int J Comput Vision 104(2):154–171
12. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788
13. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271
14. Farhadi A, Redmon J (2018) Yolov3: An incremental improvement. In: Computer Vision and Pattern Recognition, vol. 1804 (2018). Springer Berlin/Heidelberg, Germany

15. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934
16. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37 (2016). Springer
17. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2020) Focal loss for dense object detection. IEEE Trans Pattern Anal Mach Intell 42(2):318–327
18. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125
19. Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 8759–8768
20. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
21. Law H, Deng J (2018) Cornernet: Detecting objects as paired keypoints. In: Proceedings of The European Conference on Computer Vision (ECCV), pp. 734–750
22. Zhou X, Wang D, Krähenbühl P (2019) Objects as points. arXiv preprint arXiv:1904.07850
23. Ge Z, Liu S, Wang F, Li Z, Sun J (2021) Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430
24. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: European Conference on Computer Vision, pp. 213–229. Springer
25. Zhu X, Su W, Lu L, Li B, Wang X, Dai J (2020) Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159
26. Fang Y, Liao B, Wang X, Fang J, Qi J, Wu R, Niu J, Liu W (2021) You only look at one sequence: Rethinking transformer in vision through object detection. Adv Neural Inf Proc Syst **34**
27. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929
28. Jaderberg M, Simonyan K, Zisserman A (2015) Spatial transformer networks. Adv Neural Inf Process Syst 28:2017–2025
29. Zhu X, Cheng D, Zhang Z, Lin S, Dai J (2019) An empirical study of spatial attention mechanisms in deep networks. In: Proceedings of The IEEE/CVF International Conference on Computer Vision, pp. 6688–6697
30. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141
31. Li X, Wang W, Hu X, Yang J (2019) Selective kernel networks. In: Proceedings of The IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 510–519
32. Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q (2020) Eca-net: efficient channel attention for deep convolutional neural networks, 2020 ieee. In: CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 11531–11539
33. Qin Z, Zhang P, Wu F, Li X (2021) Fcanet: Frequency channel attention networks. In: Proceedings of The IEEE/CVF International Conference on Computer Vision, pp. 783–792
34. Woo S, Park J, Lee J-Y, Kweon IS (2018) Cbam: Convolutional block attention module. In: Proceedings of The European Conference on Computer Vision (ECCV), pp. 3–19 (2018)
35. Zhang H, Zu K, Lu J, Zou Y, Meng D (2021) Epsanet: An efficient pyramid split attention block on convolutional neural network. arXiv preprint arXiv:2105.14447
36. Liu S, Huang D, Wang Y (2019) Learning spatial fusion for single-shot object detection. arXiv preprint arXiv:1911.09516
37. Neubeck A, Van Gool L (2006) Efficient non-maximum suppression. In: 18th International Conference on Pattern Recognition (ICPR'06), vol. 3, pp. 850–855
38. Song K, Yan Y (2013) A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. Appl Surf Sci 285:858–864
39. He Y, Song K, Meng Q, Yan Y (2019) An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. IEEE Trans Instrum Meas 69(4):1493–1504
40. Bao Y, Song K, Liu J, Wang Y, Yan Y, Yu H, Li X (2021) Triplet-graph reasoning network for few-shot metal generic surface defect segmentation. IEEE Trans Instrum Meas 70:1–11
41. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A (2016) Learning deep features for discriminative localization. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929