A Rail Surface Defect Detection Method Based on Pyramid Feature and Lightweight Convolutional Neural Network

Yu Liu, *Member, IEEE*, Huaxi Xiao, Jiaming Xu, *Student Member, IEEE*, and Jingyi Zhao

***Abstract*— The application of computer vision technology in defect detection of industrial products is a popular research direction in recent years. This article presents the pyramid feature convolutional neural network (CNN) for defect detection of rail surfaces. First, multi-scale feature maps are extracted based on the characteristics of defects and backgrounds by the pyramid feature extraction module (PFEM). Then the feature maps are input to a lightweight network consisting of a small number of parameters. The network is trained with only 40% data of the dataset using binary cross-entropy loss function and the intersection of union (IOU) loss function. In the experiment, the performance of the proposed method is evaluated using the rail surface defect dataset (RSDD) dataset by comparing it with other methods. The experimental results show that the segmentation performance and real-time performance of the proposed method are better than those of other methods.**

***Index Terms*— Convolutional neural network (CNN), defect detection, feature pyramid, image segmentation, rail steel.**

I. Introduction

THE development of rail transit is facing the challenge of increasing speed and load, which greatly increases the pressure on rail [1]. Under long-term operation, the rails will heat and wear, causing potential security threats which need to be solved urgently. In the past, rail defects were detected by experienced inspectors, which not only requires a lot of human resources, but also has shortcomings like time consumption and low precision. Therefore, automatic nondestructive inspection systems have great market demands [2].

Over the past decades, vision-based defect detection methods have been applied to various industrial products, such as planar industrial products [3], steel [4], fabric [5], solar cells [6]–[8], lithium-ion battery electrodes [9], and thin-film transistor liquid crystals [10].

A. Related Works

With the steady development of machine vision technology, many methods are also applied to rail surface defect classification and positioning tasks. Ma *et al.* [11] present an automatic method to evaluate the severity of rail surface defects from images collected by rail inspection vehicles. It consists of three parts: rail surface segmentation module, structured random forest, and generalized Hough transform. The defect severity classification module combines multiple classifiers by stacking the ensemble model. Li and Ren [12] use the local normalization (LN) method to enhance the contrast of the rail image and then detect defects by applying the defect localization based on the projection profile (DLBP). However, this method is susceptible to noise. Li *et al.* [13] enhance rail images by the local Michelson-like contrast (MLC) measure and put forward a new automatic thresholding method, namely the proportion emphasized maximum entropy (PEME) thresholding algorithm. A new inverse Perona–Malik diffusion model is presented by He *et al*. [14] to enhance images, which takes the reciprocal of gradient as a feature to adjust the diffusion coefficient. This method is susceptible to noise and cannot always position the continuous boundaries of defects. Gan *et al*. [2] propose a coarse-to-fine extractor (CFE) for detecting defects. The coarse extractor uses the mean shift algorithm to locate the outliers and then the fine extractor filters noises. However, the CFE method is challenged with high computational complexity. A coarse-to-fine model (CTFM) [15] is proposed to identify defects at different scales which are sub-image level, region level, and pixel level. At the sub-image level, the background subtraction model is used to filter the defect-free range. At the region level, the region extraction model locates defect regions based on phase-only Fourier transforms. At the pixel level, the pixel subtraction model uses pixel consistency to refine the shape of each defect. Nieniewski [1] introduces a system for rail defect detection and shape extraction using morphological operations. Based on pyramid morphological operation, the authors implement defect detection, defect extraction, and a false-positive reduction scheme. However, the size of structural elements is related to the size of defects.

As a powerful and popular technology, deep learning-based methods have also been applied to rail defect detection. Shang *et al.* [16] propose a two-stage pipeline method for

Manuscript received October 2, 2021; revised March 2, 2022; accepted March 25, 2022. Date of publication April 6, 2022; date of current version April 25, 2022. This work was supported by the Key Research and Development Program of Guangdong Province under Grant 2021B0101200001. The Associate Editor coordinating the review process was Dr. Zhenbing Zhao. *(Corresponding author: Yu Liu.)*

Yu Liu is with the School of Automation Science and Engineering, South China University of Technology, Guangzhou 510640, China, and also with the Research and Development Center of Precision Electronic Manufacturing Technology, Guangzhou Institute of Modern Industrial Technology, Guangzhou 511458, China (e-mail: auylauscut@163.com).

Huaxi Xiao and Jiaming Xu are with the School of Automation Science and Engineering, South China University of Technology, Guangzhou 510640, China (e-mail: auhuaxi_xiao@mail.scut.edu.cn; aujiamingxu@mail.scut.edu.cn).

Jingyi Zhao is with the School of Underwater Acoustic Engineering, Harbin Engineering University, Harbin 150001, China (e-mail: zhaojingyi@hrbeu.edu.cn).

Digital Object Identifier 10.1109/TIM.2022.3165287

1557-9662 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

rail defect detection by localizing and classifying rail images. The rail part is cropped from the original image in the first stage. A fine-tuning convolutional neural network (CNN) is used for rail images classification. Yanan *et al.* [17] apply the CNN model named you only look once (YOLO) v3 to locate rail surface defects with bounding boxes. Using MobileNetV2 [18] as the backbone network to extract image features, Yuan *et al.* [19] propose a railway surface defect detection method based on end-to-end target detection and lightweight convolution network structure. Combining the strengths of fully convolutional networks and conditional random fields (CRFs), Zhang *et al.* [20] propose a deep extractor for defect detection. But the CRF model is complicated and time-consuming, which cannot be applied to real-time inspection. Besides, the results are not convincing enough because only 20% images are used for testing in the experiment, which is too few to represent the entire dataset. Kim *et al.* [21] develop a segmentation model by modifying a fully convolutional network to automatically detect the defects on railroad surfaces, but the performance of the model evaluated by only 10% samples is not convincible enough. Zhang *et al.* [22] present two networks to inspect defects on rail surfaces with the line-level label, which can only determine whether a line of the test image contains defective pixels or not. This method cannot locate defects accurately, calculate the sizes of defects, and classify the defects.

In addition to the networks mentioned above, there are some well-known segmentation networks such as Unet [23], DeepLabs [24]–[26], and Segnet [27]. In the Unet, a skip connection is used to connect the features of the encoder and the decoder. In the Deeplab networks, atrous convolution is proposed to expand the receptive field without introducing additional parameters. Segnet, using the encoder–decoder structure (EDS), saves the indexes of the maximum values in the max pooling operation and only calculates the values at these indexes when upsampling. These three networks are universal methods that can be directly used in image segmentation tasks.

B. Outline of the Work

In this article, we propose the pyramid feature-based lightweight CNN (PFCNN) to detect rail surface defects. The pyramid feature extraction module (PFEM) is designed to extract multi-scale feature maps which are useful for identifying defects by analyzing their characteristics of defects. These feature maps are input to a lightweight CNN containing only a small number of convolution layers. The binary cross entropy (BCE) loss function and the intersection of union (IOU) loss function are used to train the network. The flowchart of the proposed scheme is shown in Fig. 1.

To summarize, the main contributions of this article are as follows.

1) The PFCNN, combining pyramid features and neural networks, is proposed to detect rail surface defects. Pyramid features containing specific task information are fused to generate a prediction mask with weights automatically calculated by the neural network, without designing complex rules manually as [13], [15] do. We reduce the model complexity of the network with effective features of specific task extracted by the feature pyramid, instead of using many convolutional layers to extract abstract features like [17]–[19].
2) The feature pyramid extraction module is designed to extract multi-scale features based on the characteristics of the defect for the two types of datasets, so these features are more specific and applicable to the object. But abstract features extracted by common networks, such as [19], [23], [27], are limited by the shape and size of convolution kernels and may not be suitable for a certain type of defect. The designed features, containing global features and local features at each layer, avoid the above problem.
3) A lightweight CNN is designed and trained with only 40% data of the dataset for defect segmentation. The proposed network contains a smaller number of convolutional layers than common networks like [23]–[27] and requires fewer training samples to get satisfactory results, which can reduce training time and alleviate the problem of insufficient data.

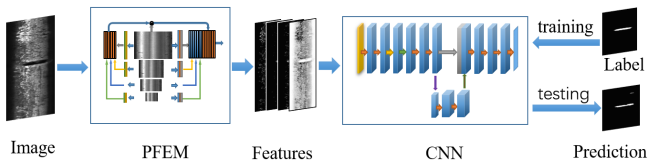


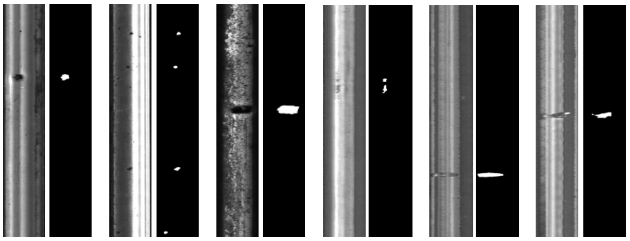
Fig. 1. Flowchart of the proposed scheme.

The rest of the article is organized as follows. In Section I-C, the dataset used in this article will be introduced. In Section II, the proposed method will be described in detail, mainly including two parts, the PFEM and CNN. The experimental parameter settings, the results of the comparative experiments, and the analysis of the experimental results will be shown in Section III. A summary of the entire article is made in Section IV.

C. Dataset Description

The dataset used in this article is a public rail surface defect dataset named (RSDD) from paper [2], which contains two types of rail steel surface images, named Type-I and Type-II, respectively. Type-I data are captured from the express rail, including 67 images while Type-II data are captured from ordinary or heavy orbits, including 128 images. These images are acquired with the industrial charge coupled device (CCD) camera which is a Dalsa Spyder high-speed line scan camera with a resolution of 1024 pixels, a maximum line speed of 65 000 lines per second, equipped with an automatic variable rate filter attenuation system [2]. Each image contains at least one defective region, and the sizes and shapes of the defects are uncertain. Defect masks for images are marked by professional experts in the field of rail detection. The label image is 8-bit gray-scale image, and we use the threshold 230/255 to binarize it, as recommended in [1]. Since the feature maps of PFEM are designed based on the characteristics of the image,

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 2. (a) Type-I images with labels. (b) Type-II images with labels.

the characteristics of the background and defects for the two types of images will be described in detail in the following two paragraphs.

Fig. 2(a) shows three images and their labels in the Type-I dataset. In the Type-I dataset, the image has relatively fixed characteristics in the column direction, and the gray value of pixels in the same column are similar. The gray value of the defect is pretty low, which is often the lowest in a column of the image. The dataset contains three different background patterns, and the third background pattern contains a lot of noise, which makes it difficult to extract defects. The shapes of the defects are square or round, but defects have a great difference in sizes.

Three images with labels of Type-II are shown in Fig. 2(b). In the Type-II dataset, the images also have fixed characteristics in the column direction. The background of the image is relatively stable without too much noise. However, the shapes and sizes of defects vary greatly. Defects are locally dark areas, and the contrast with the background is very small, so the defects are easily missed for detection.

II. Proposed Method

The algorithm proposed in this article consists of two parts, namely the PFEM and the CNN. It is a common practice to use an image pyramid to extract multi-scale features of images. For each layer of the pyramid, several feature maps are extracted and scaled to the original size. Features from different layers are combined into multi-channel data which is input to the CNN. The network consists of a series of convolution kernels of different shapes or sizes and finally outputs the probability of each pixel being positive. The network in this article is a lightweight network with small-scale parameters and a short prediction time, for which it is suitable for real-time detection.

A. Pyramid Feature Extraction Module (PFEM)

For images in the Type-I dataset, the pyramid used in the article has five layers, and the size of each layer is reduced by half. The original image is an 8-bit gray-scale image containing a little information, so we extract a few features for each layer of the pyramid.

Through the observations, we summarize some characteristics of defects. First, the gray value of the defect area is lower than that of the image background. Second, the image has a relatively consistent pattern in the column direction, and the existence of defects destroys this pattern. Third, the boundary of the defect may have strong gradients.

Based on the above three clues, we extract three feature maps for the image, denoted as f_1, f_2, and f_3, respectively. The image is first normalized from 0–255 to 0–1, denoted as I. Since the pixel with the lower gray value is more likely to be a defective pixel, so we calculate the complement of the image, namely

$$f_1(i, j) = 1 - I(i, j) \tag{1}$$

where $j = 1, 2, \ldots, W$ and $i = 1, 2, \ldots, H$. H and W are the height and width of the image, respectively.

The grayscale of each column in the image is relatively uniform, and we calculate the average of each column, denoted as I_{avg}. The average represents the grayscale of the background, and the grayscale of the defect is much smaller than the background, and it is likely to be the minimum gray value of the column. We calculate the minimum of each column and record it as $I_{\min}$

$$I_{\text{avg}}(j) = \frac{1}{H}\sum_{i=1}^{H} I(i, j) \tag{2}$$

$$I_{\min}(j) = \min_{1 \le i \le H} (I(i, j)) \tag{3}$$

where min(∗) represents the minimum function.

The grayscale of defects is between I_{avg} and $I_{\min}$. Therefore, we set a one-sided threshold for each column, denoted as I_{th}

$$I_{\text{th}}(j) = \big(I_{\text{avg}}(j) + I_{\min}(j)\big)/2. \tag{4}$$

In the jth column, if the gray value of a pixel is greater than $I_{\text{th}}(j)$, it is judged to be a background point; otherwise, it is a candidate defect point. Since the gray value of the defect is close to the global minimum, an upper limit is set for $I_{\text{th}}(j)$. The mean of $I_{\text{th}}(j)$ named m_{th} is calculated

$$m_{\text{th}} = \sum_{j=1}^{W} I_{\text{th}}(j)/W. \tag{5}$$

Set the element greater than m_{th} in I_{th} to m_{th}

$$I_{\text{th}}(j) = \begin{cases} m_{\text{th}}, & I_{\text{th}}(j) > m_{\text{th}} \\ I_{\text{th}}(j), & \text{otherwise.} \end{cases} \tag{6}$$

We calculate the distance, denoted as D_c, from each element in the image to the corresponding threshold, and the Relu function is applied to it

$$D_c(i, j) = \text{Relu}(I_{\text{th}}(j) - I(i, j)) \tag{7}$$

where Relu(*) is a function and its expression is

$$\text{Relu}(x) = \begin{cases} x, & x > 0 \\ 0, & x <= 0 \end{cases} \tag{8}$$

where x represents the input to the Relu function.

We use the distance from the gray value of the candidate point to the $I_{\text{avg}}(j)$ as the significance, and the second feature map f_2 is obtained by

$$f_2(i, j) = D_c(i, j) \times \big(I_{\text{avg}}(j) - I(i, j)\big). \tag{9}$$

The third feature map is related to image gradient. There may be strong gradient intensity around defects, which helps

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

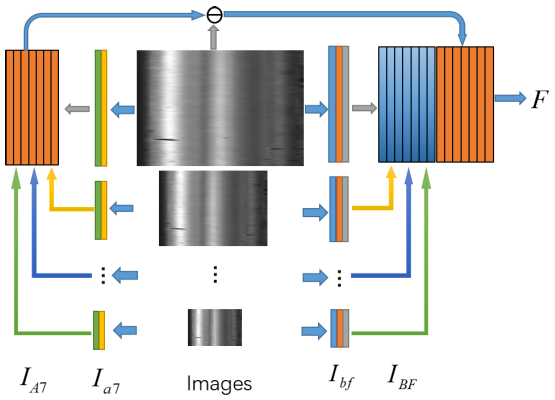


Fig. 3. Pyramid feature extraction module.

us locate defects. We use the Sobel operator to calculate the gradient map of the image, denoted as I_g. Since there may be a large gradient between adjacent columns, it is necessary to decentralize the columns of the gradient map. Calculate the average of the gradient of each column

$$m_g(j) = \sum_{i=1}^{H} I_g(i, j)/H \tag{10}$$

where $m_g(j)$ is the mean of the jth column of I_g. After I_g is decentralized, the third feature map f_3 is obtained

$$f_3(i, j) = I_g(i, j) - m_g(j). \tag{11}$$

We combine f_1, f_2, f_3 into three-channel data, denoted as I_{bf}. I_{bf} is a global feature and we also need local features. The 7×1 and 1×7 window are used to calculate the local mean of the image and the two results are combined into a two-channel data, denoted as I_{a7}.

I_{bf} and I_{a7} from each layer of the pyramid are scaled to the original size and spliced on the channel dimension, denoted as I_{BF} and I_{A7}, respectively. There are five layers in the pyramid, while I_{bf} and I_{a7} have three channels and two channels, so I_{BF} and I_{A7} have 15 channels and ten channels, respectively. We subtract the original image from I_{A7} to enhance the dark pixels and apply the Relu function to the result

$$I_{A7}(i, j, c) = \text{Relu}(I_{A7}(i, j, c) - I(i, j)) \tag{12}$$

where $c = 1, 2, \ldots, 10$, representing the channel dimension of the I_{A7}. After concatenating I_{BF} and I_{A7}, a multi-scale feature map is obtained. Fig. 3 shows the PFEM.

The above steps are designed based on the Type-I dataset, and the steps of feature extraction based on the Type-II dataset are described below. Most of the steps are the same, and only the differences will be explained. The first point is the image pyramid, and the second point is the feature map f_2.

Type-I images have three different sizes, namely 1000×160, 1260×160, and 1282×160, which will be scaled to 1024×160. The sizes of the pyramid images of Type-I are (1, 1), (1/2, 1/2), (1/4, 1/4), (1/8, 1/8), (1/16, 1/16), respectively. $(r1, r2)$ indicates that the size of the image is $(r1 \times H, r2 \times W)$. The sizes of the Type-II images are 1250×55 which are scaled to 1024×64. Since the width of the image is too small to be downsampled by half each time, the sizes of each layer of the pyramid are (1, 1), (1/2, 1), (1/4, 1/2), (1/8, 1/2), (1/16, 1/4), (1/32, 1/4), and (1/64, 1/8), respectively.

Since the characteristics of the defects of Type-II are different from those of Type-I, we redesigned the feature map f_2, and the specific steps are shown as follows. We first calculate the standard deviation of the jth column, denoted as $I_{\text{std}}(j)$

$$I_{\text{std}}(j) = \frac{1}{H}\sqrt{\sum_{i=1}^{H}\left(I(i, j) - I_{\text{avg}}(j)\right)^2}. \tag{13}$$

Then we calculate the mean of all $I_{\text{std}}(j)$, denoted as m_{std}

$$m_{\text{std}} = \sum_{j=1}^{W} I_{\text{std}}(j)/W. \tag{14}$$

Combining $I_{\text{avg}}(j)$ in (2) and m_{std}, we define a threshold for the jth column, denoted as $I_{\text{th}}'(j)$

$$I_{\text{th}}'(j) = I_{\text{avg}}(j) - 2 \times m_{\text{std}}. \tag{15}$$

The gray values of defective pixels are usually lower than $I_{\text{th}}'(j)$ on the jth column. Finally, each value of the feature map, denoted as $f_2'(i, j)$, are computed by

$$f_2'(i, j) = \text{Relu}\left(I_{\text{th}}'(j) - I(i, j)\right). \tag{16}$$

B. Convolutional Neural Network (CNN)

No need to manually design and merge different features is the advantage of CNNs. However, as a black-box model, it has a certain degree of blindness and the features and weights learned automatically cannot be guaranteed to be effective for the task. When the results are different from expectations, it is difficult to find the problem. In addition, training a neural network requires sufficient training data while labeling images takes a lot of labor cost. When the training data is insufficient, the neural network model often performs poorly. What is more, training a network with a large number of parameters is very time-consuming. This article gives a hint to settle the above problems. In the previous section, we extracted the multi-scale features of the image with the PFEM. Feature extraction can be seen as data augmentation increasing the information of the input image, for which the scale of the parameters can be reduced. Besides, fewer data are needed for training the network, which can alleviate the problem of insufficient data. Moreover, fewer parameters of the model and fewer training data can reduce the training time.

A lightweight CNN is built in this article to predict defective pixels, and its specific structure is shown in Fig. 4. The network consists of a series of convolution blocks, and the convolution block is composed of three parts: the convolution layer, the batch normalization (BN) layer, and the Relu function activation layer. The input of the CNN is the pyramid features extracted in the previous section. For the Type-I dataset, the number of input channels is 25, and for the Type-II dataset, the number of channels is 35. Conv 3×3 represents a convolution block with a 3×3 convolution kernel. Conv 1×7 and Conv 7×1 are used to fuse features in the row direction and column direction, respectively. Max-pool represents the

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

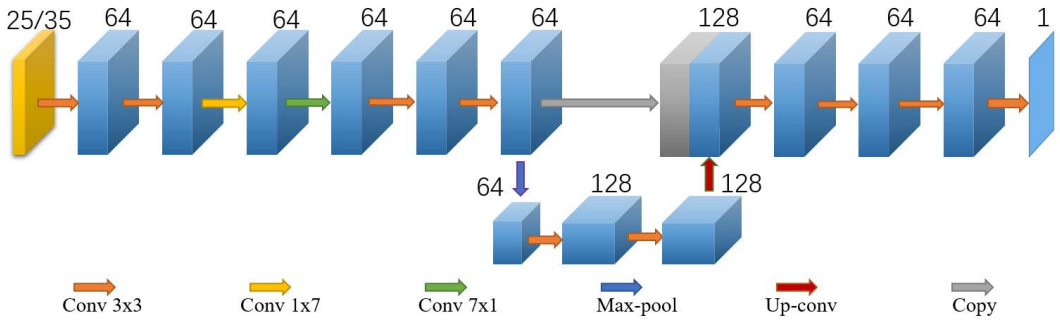


Fig. 4. Proposed CNN structure.

max pooling operation, and the up-conv means the transposed convolution layer.

The EDS is an effective network structure that exists in Unet, Segnet, and other networks. The encoder reduces the size of the feature maps through downsampling and increases the dimension of features. The decoder restores the image size through transposed convolution and merges the features of the encoder. EDS can extract and fuse features of different scales well. Because we have used the pyramid to extract multi-scale features, we only use a two-layer structure here.

The last convolution block of the network outputs the probability of being a defective pixel, using the Sigmoid activation function to limit the output in the range of 0–1. When the probability is greater than 0.5, the probability of the pixel being defective is greater than the probability of being defect-free, so we use 0.5 as the threshold to binarize the probability map.

Defect segmentation is a binary classification task that identifies whether each pixel is a defective pixel or not. The cross-entropy function is used to train the predictive ability of the network, and its specific form is

$$l_{\mathrm{CE}} = -\frac{1}{N}\sum_{k=1}^{N}\left(w_p y_k{}' \log y_k + \left(1 - y_k{}'\right)\log(1 - y_k)\right) \quad (17)$$

where $y_k{}'$ is the label value of the pixel, with a value 0 or 1, y_k is the predicted value, with a value in the range of 0–1, w_p is the weight of positive pixel, and $N = H \times W$. We calculate the average of the cross-entropy of all pixels in the entire image. However, the cross-entropy function only focuses on a single pixel and does not have an overall understanding of the prediction results of the entire image. Besides, since the positive pixels are far fewer than the negative pixels, the network will tend to judge the pixels as negative. As the network iterates, the accuracy rate is improving, while the recall rate is declining. This means that more positive pixels are misjudged as negative, resulting in missing detection of defects. For this reason, we introduce the IOU loss function

$$l_{\mathrm{IOU}} = -\log\left(\frac{Y' \cap Y}{Y' \cup Y}\right) = -\log\left(\frac{\sum_{k=1}^{N} y_k{}' y_k}{\sum_{k=1}^{N} y_k{}' + y_k - y_k{}' y_k}\right) \quad (18)$$

where Y' is the label map and y_k' is an element of Y', Y is the prediction map, and y_k is an element of Y, $N = H \times W$. l_{IOU} makes the defect shape and area predicted by the network closer to the ground truth. Compared with the cross-entropy loss, IOU pays more attention to the difference between the shapes and sizes of the predicted defects and those of ground truths. In the end, the loss of the network is the sum of cross-entropy and IOU loss as follows:

$$\mathrm{Loss} = l_{\mathrm{CE}} + l_{\mathrm{IOU}}. \quad (19)$$

The greater the positive weight w_p, the greater the loss caused by misjudging a positive pixel as a negative one. w_p can increase the recall of the network, but at the same time, it will cause an increase in false positives. The IOU loss function alleviates this problem since the IOU loss function ensures that the shapes and areas of the predicted positive regions are very close to the ground truths.

III. Experiment

This section is the experimental part, which describes the experimental setup and experimental results and analysis. The algorithm in the article is implemented in Python, using Pycharm as the integrated development environment. The code runs on a desktop computer with a 2.8 GHz, 10-generation i9 CPU.

A. Experimental Setup

The Type-I and Type-II datasets contain 67 and 128 images, respectively. We randomly select 40% of the data as the training set, and the remaining data are the test set. Limited by the memory size of the GPU, the batch size is set to 1, and the iteration period is 100. Parameter w_p in l_{CE} is set to 50 for type-I dataset, and 200 for type-II dataset (refer to *D. Parameter Analysis*). The optimization algorithm is Adam, using the default parameters in PyTorch.

B. Evaluation Metrics

As suggested by the previous works [1], [2], we use precision (PR), recall (RC), and F-measure (FM) as the evaluation metrics. And the evaluation metrics are calculated in pixel level and defect level. The pixel-level indexes are calculated by the following formulas:

$$\begin{aligned} \mathrm{PR} &= \mathrm{TP}/(\mathrm{TP} + \mathrm{FP}) \\ \mathrm{RC} &= \mathrm{TP}/(\mathrm{TP} + \mathrm{FN}) \\ \mathrm{FM} &= 2 \times \mathrm{PR} \times \mathrm{RC}/(\mathrm{PR} + \mathrm{RC}). \end{aligned} \quad (20)$$

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

True positive (TP) means the number of defective pixels detected correctly. False positive (FP) indicates the number of negative pixels that are predicted as positive ones. False negative (FN) refers to the number of positive pixels that are predicted as negative pixels. The higher the value FM is, the closer the result is to human judgment. Note that the values of PR, RC, and FM are average values for the test set.

For the defect-level index, PR′, RC′, and FM′ are defined as follows:

$$\begin{aligned} \mathrm{PR}' &= \mathrm{TP}'/P \\ \mathrm{RC}' &= \mathrm{TP}''/M \\ \mathrm{FM}' &= 2 \times \mathrm{PR}' \times \mathrm{RC}' \big/ \big(\mathrm{PR}' + \mathrm{RC}'\big) \end{aligned} \tag{21}$$

where TP′ is the number of correctly detected defects and TP″ represents the number of marked defects recalled by a detection method. When the overlap between the detected defects and the marked defects in the corresponding image is greater than 50% of the detected defects, the detected defect is a correct detection. If the overlap of a marked defect with some detected defects in the corresponding image is greater than 50% of the marked defect, the defect is recalled. P is the total number of defects detected, and M is the total number of defects marked in ground truths. Note that the counts of PR′ and RC′ are average for the entire test set.

Among the three indicators, PR′ represents the proportion of real defects in the detected defects. Lower PR′ will cause many non-defective samples to be identified as defective samples, thereby increasing the workload. RC′ indicates how many real defects have been detected. This is a very important indicator, because missing any defect may lead to security crisis. FM′ is a criterion used to measure PR′ and RC′ at the same time. PR, RC, and FM have similar meanings in pixel level. In order to comprehensively consider the pixel-level and defect-level performance of algorithms, we define indicator $F2$

$$F2 = 2 \times \mathrm{FM} \times \mathrm{FM}' \big/ \big(\mathrm{FM} + \mathrm{FM}'\big). \tag{22}$$

C. Experimental Results and Analysis

CFE [2], CTFM [15], and MDE [1] use the same RSDD dataset, and they all achieve good performance. We directly use the results in their papers to compare the performance of the method proposed in this article. In addition, we carry out comparative experiments with other CNNs and the well-known Unet++ [28], Deeplab v3+ [26], Segnet [27], Enet [29], and PSPnet [30] are selected to compare with PFCNN. All networks use the same training set, test set, batch size, number of iterations, and optimization method.

By comparing multiple methods, the CFE method is the best in terms of the three indicators. However, CFE is a very complicated and time-consuming method which cannot be used for real-time detection. As a result, in the article of MDE and CTFM, CFE is excluded from comparisons. In the article of MDE, four experimental results obtained from different parameters are given and we select the best result to compare with other methods. The experimental results of different methods applied to the Type-I dataset are shown in Table I, and the experimental results of the Type-II dataset are shown in Table II. The top two highest values for each indicator are shown in bold and bold italic, respectively. Figs. 5 and 6 show four samples from the experimental results of the Type-I and Type-II datasets, respectively. In Fig. 5, the defects are dark areas where material breaks out of the metal surface, which have different sizes. They may result from hard foreign objects on the rail being run over. Small defects are easy to be missed or misjudged when the rail surface is contaminated by oil or other factors. For the defects of different areas in Fig. 5 and the defects of different shapes and contrasts in Fig. 6, PFCNN has obtained satisfactory results.

TABLE I
EXPERIMENTAL RESULTS FOR TYPE-I DATASET

Method	PR	RC	FM	PR'	RC'	FM'	$F2$
	Pixel-level			Defect-level			
CFE	**87.54%**	**85.63%**	**85.12%**	**93.02%**	**85.40%**	**89.05%**	**87.04%**
CTFM	86.54%	77.68%	80.02%	***84.06%***	77.37%	80.58%	80.30%
MDE	81.56%	82.45%	78.68%	69.87%	78.52%	73.94%	76.24%
Unet++	***90.13%***	67.85%	74.10%	79.75%	61.45%	69.41%	71.68%
Deeplab	82.70%	56.68%	62.66%	61.63%	44.58%	51.73%	56.67%
Segnet	84.39%	65.18%	68.91%	37.68%	57.83%	45.63%	54.90%
Enet	80.69%	50.94%	59.31%	29.70%	33.73%	31.59%	41.22%
PSPnet	75.56%	73.12%	72.11%	63.10%	73.61%	67.95%	69.97%
PFCNN	85.51%	**83.06%**	**82.95%**	77.45%	***84.34%***	***80.75%***	***81.90%***

TABLE II
EXPERIMENTAL RESULTS FOR TYPE-II DATASET

Method	PR	RC	FM	PR'	RC'	FM'	$F2$
	Pixel-level			Defect-level			
CFE	**83.88%**	83.58%	**82.11%**	**91.91%**	83.98%	**87.76%**	**84.84%**
CTFM	***84.12%***	73.25%	76.45%	85.83%	83.98%	84.89%	80.45%
MDE	72.57%	***87.07%***	76.61%	82.25%	***89.44%***	85.70%	80.90%
Unet++	60.67%	77.64%	62.12%	39.00%	74.29%	51.15%	56.10%
Deeplab	61.72%	67.46%	54.73%	43.21%	66.67%	52.43%	53.56%
Segnet	70.06%	68.33%	64.38%	54.05%	68.57%	60.45%	62.35%
Enet	75.14%	60.42%	61.81%	47.67%	60.95%	53.50%	57.36%
PSPnet	68.37%	79.18%	71.31%	74.26%	80.00%	77.02%	74.06%
PFCNN	74.55%	**84.49%**	**77.14%**	**86.67%**	***89.52%***	***88.07%***	***82.24%***

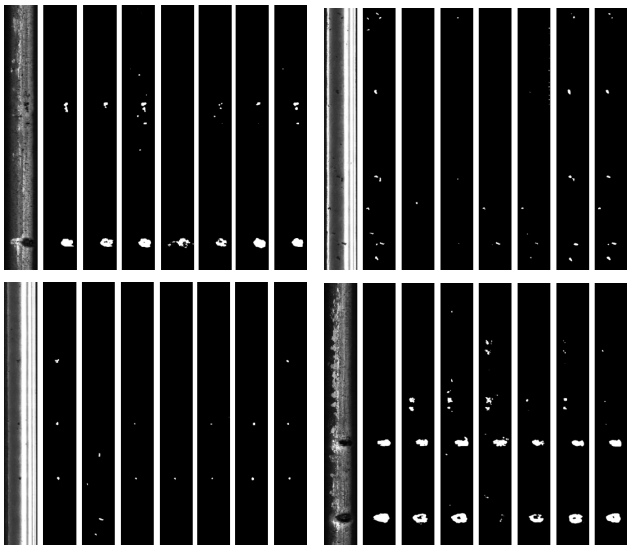


Fig. 5. Experimental results of different methods applied to the Type-I dataset. The sequence of images is the original image, ground-truth image, results from Unet++, Deeplab, Segnet, Enet, PSPnet, and PFCNN.

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

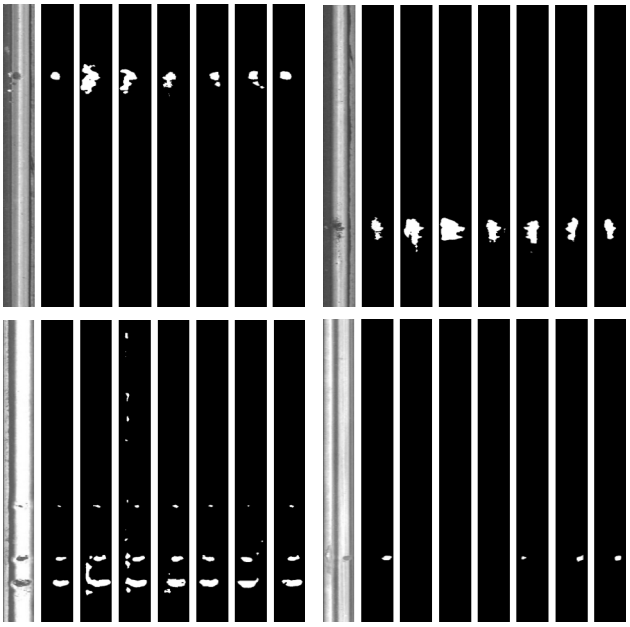


Fig. 6. Experimental results of different methods applied to the Type-II dataset. The sequence of images is the original image, ground-truth image, results from Unet++, Deeplab, Segnet, Enet, PSPnet, and PFCNN.

We first analyze the results in Table I. In terms of pixel-level indicators, CTFM has higher PR, and lower RC, which are 86.54% and 77.68%, respectively, while MDE has higher RC and lower PR, which are 82.45% and 81.56%, respectively. PFCNN obtains PR = 85.51%, RC = 83.06%, and FM = 82.95%. The three indicators of PFCNN are higher than MDE. Unet++ gets PR = 90.13% but other indicators are low. The performance of Unet++, PSPnet, Segnet, DeepLab, and Enet is in descending order and none of them perform better than PFCNN. From the defect-level indicators, PFCNN obtains PR′ = 77.45% and RC′ = 84.34% which is second only to CFE. In an image with a complex background, PFCNN may predict some small false-positive areas, which has a little effect on the pixel-level indicators, but will lead to a decrease in PR′.

As can be seen from the results in Table II, the advantage of CTFM lies in high PR, and the advantage of MDE lies in high RC. PFCNN obtains PR = 74.55%, RC = 84.49%, and FM = 77.14%. From the view of FM, PFCNN is comparable to their performance. The reason for the moderate performance of PFCNN in pixel-level metrics is that the shapes of defects change greatly and the boundary is abnormally tortuous, while the defect area predicted by CNN has a smooth boundary. The performance of the other networks is unsatisfactory, and the shapes of defects predicted by them are inaccurate and the defects with low contrast are missed. In terms of defect-level indicators, PFCNN obtains PR′ = 86.67%, RC′ = 89.52%, and FM′ = 88.07%. The PR′ of PFCNN is better than CFTM and MDE, and RC′ and FM′ have achieved the highest values.

Table III shows the memory occupied by the parameters of each network. The memory of parameters for PFCNN is 5 MB and the execution time is 31 and 17 ms for type-I and type-II data, respectively. Among all the networks, PFCNN has the least number of parameters and the shortest prediction time. Note that the time of the proposed method includes the calculation time of the feature pyramid. Table IV shows the execution time of the methods designed for this dataset. The execution time of the PFCNN is only 31 ms which can be applied to real-time detection, while the CFE is 661 ms. Although the CFE achieves pretty high accuracy, its computational complexity greatly limits its application. The fact that PFCNN performs better than other networks proves the importance of the feature pyramid. As long as there are sufficiently effective features, the network can segment defects with a small number of convolution layers. The general neural networks are limited by the receptive field of the convolution kernel and can only obtain local abstract features. First, local features are not suitable for such defects in this article that require column-oriented global features. Second, abstract features are learned by the network, and their effectiveness cannot be guaranteed. Manually designed features can avoid the above two problems, while reducing the scale of networks, shortening the training time and testing time, and reducing the number of samples required for training.

TABLE III
PARAMETER SCALE AND PREDICTION TIME OF NETWORKS

Model	Unet++	Deeplab	Segnet	Enet	PSPnet	PFCNN
Param (MB)	105	680	337	5	535	**5**
Time(ms) I	56	107	55	74	65	**31**
Time(ms) II	32	60	41	45	41	**17**

TABLE IV
EXECUTION TIME OF DIFFERENT METHODS

Model	CFE	CTFM	MDE	PFCNN
Time (ms)	661	271	50	**31**

In general, among all the networks, PFCNN has the best segmentation performance, the fewest parameters, and the shortest prediction time. Among the methods designed for the dataset, the segmentation performance of PFCNN is only worse than CFE, but the execution time of PFCNN is much shorter than CFE. Considering the real-time performance of the algorithm, the overall performance of PFCNN is even better.

D. Parameter Analysis

In this section, we will explore the impact of two parameters on the performance of the algorithm, one is the window size of the I_{A7} in (12), and the other is the w_p in (17). We can see the impact of the window size on the overall performance of the algorithm from Fig. 7. When the window size is 7×1 and 5×1, the algorithm achieves the best performance on type-I and type-II datasets, respectively. Fig. 8 shows the relationship between parameter w_p and algorithm performance. When $w_p = 50$ and $w_p = 200$, the algorithm achieves the optimal performance for type-I and type-II data, respectively.

E. Noise Analysis

In this experiment, we add two kinds of noise to the image, namely salt-pepper noise and Gaussian noise, and then observe

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

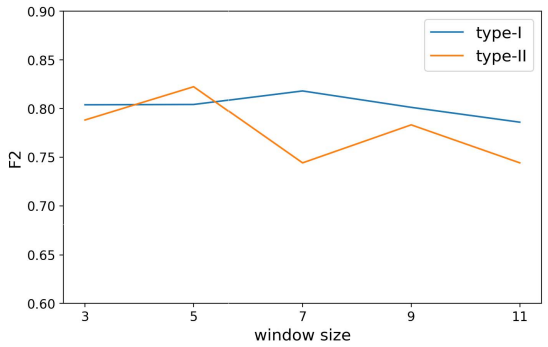


Fig. 7. Influence of local window size on the algorithm performance.

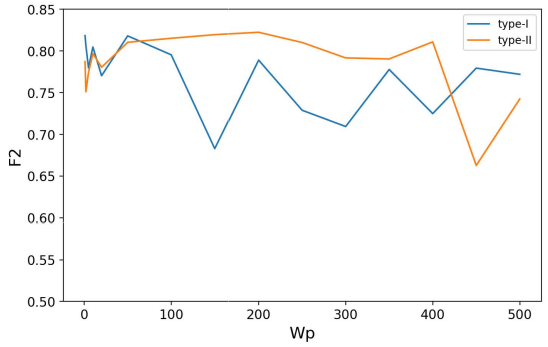


Fig. 8. Influence of parameter w_p on the algorithm performance.

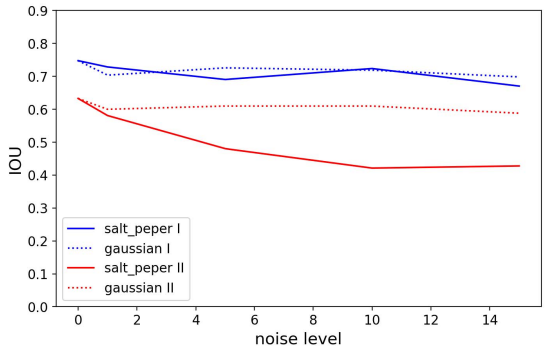


Fig. 9. Influence of noise on the algorithm performance.

the changes in algorithm performance. Salt-pepper noise is a random point with gray scale of 0 or 255. We randomly select $sp\%$ of the image points to become noise, where $sp = 1, 5, 10, 15$, respectively. Gaussian noise is a random value of a normal distribution with zero mean and standard deviation s, where $s = 1, 5, 10, 15$, respectively. Fig. 9 shows the influence of different levels of noise on the performance. The influence of Gaussian noise is relatively small, because the image pyramid has the function of Gaussian smoothing. For type-I data, since the image has pixels similar to salt-pepper noise, as shown in Fig. 5, the salt-pepper noise does not cause too much performance degradation. But for type-II data, the contrast between the defect and the background is small, and the salt-pepper noise reduces the quality of the image, thus causing performance degradation.

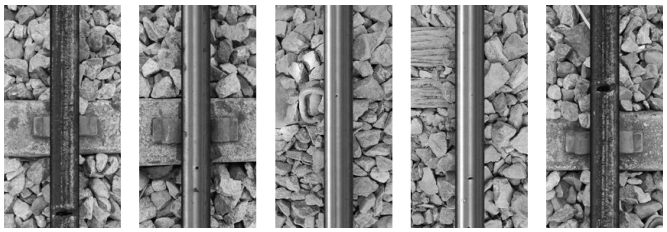


Fig. 10. Images with natural backgrounds.

TABLE V
EXPERIMENTAL RESULTS FOR ARTIFICIAL IMAGES DATASET

Method	PR	RC	FM	PR'	RC'	FM'	$F2$	T(ms)
	Pixel-level			Defect-level				
Unet++	***80.82%***	**79.75%**	**78.74%**	***55.26%***	**81.94%**	***66.01%***	***71.81%***	27
Deeplab	65.71%	52.32%	54.97%	50.98%	51.39%	51.18%	53.01%	39
Segnet	80.77%	61.62%	67.00%	25.82%	63.89%	36.77%	47.49%	***24***
Enet	73.26%	32.64%	42.02%	81.25%	23.61%	36.59%	39.12%	28
PSPnet	55.39%	43.84%	45.49%	58.18%	37.50%	45.61%	45.55%	30
PFCNN	**81.41%**	***76.96%***	***76.60%***	***73.86%***	***75.00%***	**74.43%**	**75.50%**	**19**

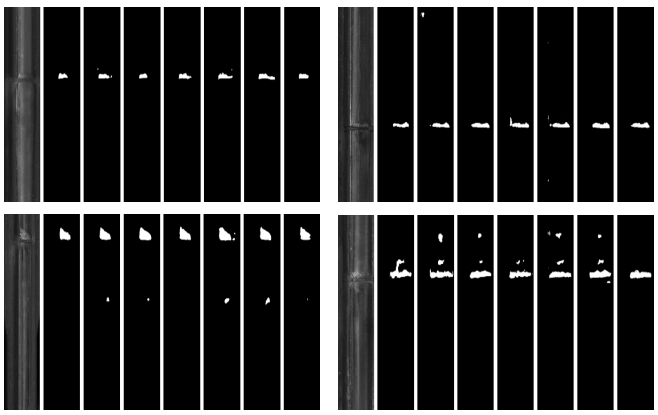


Fig. 11. Four samples from the experimental results of the SEPR dataset. The sequence of images is the original image, ground-truth image, results from Unet++, Deeplab, Segnet, Enet, PSPnet, and PFCNN.

F. Application in Actual Scenarios

In this section, we carry out an experiment to evaluate the performance of the algorithm in actual scenarios. Experimental images are artificially synthesized using Type-I dataset, which are scaled to 512 × 256 and shown in Fig. 10. The rail is surrounded by real background containing stones and other objects, which brings challenges to defect detection. Table V shows the experimental results of comparative methods. The proposed method has the second highest but close to the highest pixel level index, the highest defect level index, and the shortest execution time. When meeting a more complex scene, the performance of the proposed PFCNN has decreased, but it is still better than the comparative methods.

G. Validation on Supplementary Dataset

To further validate the proposed method, we carry out another experiment on a new dataset named spark-erosion-pro-rail (SEPR) which is available on https://www.kaggle.com/oscarvanhees/image-data-of-spark-erosion-prorail. This dataset contains 150 rail images with spark erosion

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
EXPERIMENTAL RESULTS FOR SEPR DATASET

Method	PR	RC	FM	PR'	RC'	FM'	$F2$	T(ms)
	Pixel-level			Defect-level				
Unet++	80.98%	**91.84%**	**85.28%**	64.05%	**93.40%**	75.99%	***80.37%***	35
Deeplab	80.56%	77.49%	75.72%	***82.31%***	78.30%	80.25%	77.92%	34
Segnet	**85.84%**	79.45%	80.35%	69.63%	87.74%	77.64%	78.97%	**18**
Enet	***84.85%***	80.85%	81.45%	64.32%	89.62%	74.89%	78.04%	23
PSPnet	73.03%	***88.11%***	78.55%	72.81%	***92.45%***	***81.46%***	79.98%	26
PFCNN	83.42%	84.66%	***82.78%***	**89.17%**	91.51%	**90.32%**	**86.39%**	***21***

defects. We apply the PFCNN on this dataset based on the characteristics of defects. Four samples are shown in Fig. 11, and experimental results are shown in Table VI. PFCNN obtains FM $=$ 82.78% and FM$'$ $=$ 90.32% at pixel-level and defect-level, respectively. The execution time of PFCNN is 21 ms. Unet++ performs well at pixel-level with FM $=$ 85.28%, but performs poorly at defect-level with FM$'$ $=$ 75.99%. Unet++ introduces too many false-positive regions, resulting in lower PR$'$. Other networks do not give satisfactory results in terms of statistical data. Considering the segmentation performance and real-time performance of the algorithm, the proposed method is successfully applied to this dataset.

IV. CONCLUSION

Based on pyramid features and CNNs, this article proposes the PFCNN method for detecting rail surface defects. The multi-scale feature maps are extracted by the PFEM based on the characteristics of the defects and the image background. The multi-scale feature maps are input into the CNN. This network has a simple structure and is a lightweight network consisting of a small number of parameters which has a short training and predicting time. Using the positive-weighted cross-entropy loss function and IOU loss function to train the network can solve the sample imbalance problem between positive and negative pixels. In the experiments, the presented method has achieved good performance on two types of datasets. PFCNN obtains FM $=$ 82.95% at the pixel level and FM$'$ $=$ 80.75% at the defect level for Type-I RSDD. Similarly, FM $=$ 77.14% and FM$'$ $=$ 88.07% are obtained for Type-II RSDD at the pixel level and defect level, respectively. PFCNN is only second to the CFE on the segmentation performance. But in real-time performance, PFCNN far exceeds CFE. The execution time of PFCNN is 31 ms and CFE is 661 ms for Type-I RSDD. Considering the practical application of the algorithm, PFCNN is superior to other methods. Multi-scale pyramid feature maps containing specific task information can improve the performance of the network on specific objects. Therefore, this algorithm can be applied to other datasets and other types of defects by modifying features in the pyramid based on the characteristics of images.

According to the statistical data in Table I, there is room to further improve PR, which can be done by reducing false-positive areas. A possible approach is to add a classifier to further classify the regions detected by PFCNN to reduce false-positive regions. However, this will increase the execution time of the detection algorithm. How to balance the segmentation performance and real-time performance of the algorithm is what we need to take into consideration.

REFERENCES

[1] M. Nieniewski, "Morphological detection and extraction of rail surface defects," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 9, pp. 6870–6879, Sep. 2020.

[2] J. Gan, Q. Li, J. Wang, and H. Yu, "A hierarchical extractor-based visual rail surface inspection system," *IEEE Sensors J.*, vol. 17, no. 23, pp. 7935–7944, Dec. 2017.

[3] H. Kong, J. Yang, and Z. Chen, "Accurate and efficient inspection of speckle and scratch defects on surfaces of planar products," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1855–1865, Aug. 2017.

[4] K. Liu, H. Wang, H. Chen, E. Qu, Y. Tian, and H. Sun, "Steel surface defect detection using a new Haar–Weibull-variance model in unsupervised manner," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 10, pp. 2585–2596, Oct. 2017.

[5] A. Rebhi, I. Benmhammed, S. Abid, and F. Fnaiech, "Fabric defect detection using local homogeneity analysis and neural network," *J. Photon.*, vol. 2015, pp. 1–9, Mar. 2015.

[6] D.-M. Tsai, S.-C. Wu, and W.-Y. Chiu, "Defect detection in solar modules using ICA basis images," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 122–131, Feb. 2013.

[7] J. Xu, Y. Liu, and Y. Wu, "Automatic defect inspection for monocrystalline solar cell interior by electroluminescence image self-comparison method," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–11, 2021.

[8] Y. Liu, J. Xu, and Y. Wu, "A CISG method for internal defect detection of solar cells in different production processes," *IEEE Trans. Ind. Electron.*, vol. 69, no. 8, pp. 8452–8462, Aug. 2022.

[9] J. Xu, Y. Liu, H. Xie, and F. Luo, "Surface quality assurance method for lithium-ion battery electrode using concentration compensation and partiality decision rules," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 3157–3169, Jun. 2020.

[10] C.-J. Lu and D.-M. Tsai, "Automatic defect inspection for LCDs using singular value decomposition," *Int. J. Adv. Manuf. Technol.*, vol. 25, nos. 1–2, pp. 53–61, Jan. 2005.

[11] K. Ma, T. F. Yago Vicente, D. Samaras, M. Petrucci, and D. L. Magnus, "Texture classification for rail surface condition evaluation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–9.

[12] Q. Li and S. Ren, "A real-time visual inspection system for discrete surface defects of rail heads," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 8, pp. 2189–2199, Aug. 2012.

[13] Q. Li and S. Ren, "A visual detection system for rail surface defects," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev*, vol. 42, no. 6, pp. 1531–1542, Nov. 2012.

[14] Z. He, Y. Wang, F. Yin, and J. Liu, "Surface defect detection for high-speed rails using an inverse P-M diffusion model," *Sensor Rev.*, vol. 36, no. 1, pp. 86–97, Jan. 2016.

[15] H. Yu *et al.*, "A Coarse-to-Fine model for rail surface defect detection," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 3, pp. 656–666, Mar. 2019.

[16] L. Shang, Q. Yang, J. Wang, S. Li, and W. Lei, "Detection of rail surface defects based on CNN image recognition and classification," in *Proc. Int. Conf. Adv. Commun. Technol.*, 2018, pp. 45–51.

[17] S. Yanan, Z. Hui, L. Li, and Z. Hang, "Rail surface defect detection method based on YOLOv3 deep learning networks," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2018, pp. 1563–1568.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[19] H. Yuan, H. Chen, S. Liu, J. Lin, and X. Luo, "A deep convolutional neural network for detection of rail surface defect," in *Proc. IEEE Veh. Power Propuls. Conf.*, Oct. 2019, pp. 2019–2022.

[20] Z. Zhang, M. Liang, and Z. Wang, "A deep extractor for visual rail surface inspection," *IEEE Access*, vol. 9, pp. 21798–21809, 2021.

[21] H. Kim, S. Lee, and S. Han, "Railroad surface defect segmentation using a modified fully convolutional network," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 12, pp. 4763–4775, 2020.

[22] D. Zhang, K. Song, Q. Wang, Y. He, X. Wen, and Y. Yan, "Two deep learning networks for rail surface defect inspection of limited samples with line-level label," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 6731–6741, Oct. 2021.

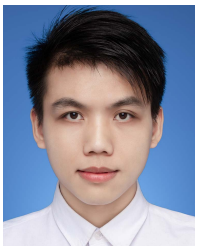
Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Interv.*, 2015, pp. 234–241.
[24] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Proc. 3rd Int. Conf. Learn. Represent.*, vol. 40, no. 4, Jun. 2015, pp. 834–848.
[25] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
[26] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.
[27] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[28] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, Sep. 2018, pp. 3–11.
[29] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.
[30] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239.

Yu Liu (Member, IEEE) received the Ph.D. degree in automatic control from the South China University of Technology, Guangzhou, China, in 2009.

He was a Visiting Student with the Department of Mechanical Engineering, Concordia University, Montreal, QC, Canada, from 2008 to 2009, and a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Nebraska–Lincoln, Lincoln, NE, USA, from 2017 to 2018. He is currently a Professor with the School of Automation Science and Engineering, South China University of Technology, Guangzhou, and with the Research and Development Center of Precision Electronic Manufacturing Technology, Guangzhou Institute of Modern Industrial Technology, Guangzhou. His current research interests include distributed parameter systems, intelligent control, robot control, learning control, intelligent perception and decision making, and machine vision.

Dr. Liu is currently an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, and *International Journal of Space-Based and Situated Computing*.



Huaxi Xiao received the B.E. degree in automation from the South China University of Technology, Guangzhou, China, in 2019, where he is currently pursuing the M.E. degree in control science and engineering with the School of Automation Science and Engineering.

His current research interests include image processing, machine learning, and computer vision.

Jiaming Xu (Student Member, IEEE) received the M.E. degree in mechanical and electronic engineering from Guangzhou University, Guangzhou, China, in 2018. He is currently pursuing the Ph.D. degree in control science and engineering with the School of Automation Science and Engineering, South China University of Technology, Guangzhou.

His current research interests include pattern recognition, computer vision, and intelligent control.

Jingyi Zhao received the B.E. degree in electronic engineering from Harbin Engineering University, Harbin, China, in 1992.

He is currently a Senior Engineer with the College of Underwater Acoustic Engineering, Harbin Engineering University. His research interests include robotics, detection and diagnosis technique, underwater communication and networks, control and filtering theory, and machine vision.

Authorized licensed use limited to: BMS College of Engineering. Downloaded on September 01,2023 at 09:56:45 UTC from IEEE Xplore. Restrictions apply.