A Preprocessing Method to Improve Edge Crack Detection from Railway Tunnel Lining Images

Tiantao Zhang[1], Chengshun Lv[1], Jian Liu[1], Lei Kou[1*], Quanyi Xie[1*], Meidong Duan[2], Xiao Zhang[34], Debao Zhu[5]

[1]School of QiLu Transportation, Shandong University, No. 27 South Shanda Road, 250061, Jinan, China, 202435516@mail.sdu.edu.cn, 202115369@mail.sdu.edu.cn, lj75@sdu.edu.cn, lei.kou@sdu.edu.cn, xiequanyi@sdu.edu.cn

[2]Shandong Hi-speed Company Limited, No. 5006, Olympic Middle Road, 250098, Jinan, China, 202215403@mail.sdu.edu.cn

[3]Weifang Hydrodynamics Science and Technology lndustry lnstitute, Geographic Information Mapping Park, 261200, Weifang, China, sduzhangxiao@sdu.edu.cn

[4]Laoshan Laboratory, No. 2 Qiyunshan 2nd Road, 266200, Qingdao, China, sduzhangxiao@sdu.edu.cn

[5]Shandong High Speed Engineering Inspection Co., No. 12550 East Second Ring Road, 250002, Jinan, China, 202315451@mail.sdu.edu.cn

Abstract: Crack detection is critical for guaranteeing the safety of bridges, railways, and other infrastructures; however, it is a difficult task, particularly for tunnels. Tunnel lining images are primarily acquired using vision sensors, and cracks typically appear throughout an entire image. For crack detection using convolutional neural networks, the recognition accuracy is unsatisfactory when the cracks are at the edge of the image. Hence, an image preprocessing method is proposed to process railway tunnel data. In this method, the relative position of cracks in an image is changed by adding different sizes of borders to the crack images, and four different detection models are used for training to examine the effectiveness of the preprocessing method. Experimental results show that the proposed preprocessing method achieves better detection results for all four models. In the custom dataset, the border size is set to 1/9 of the original image size, which is the most effective size for improving edge crack recognition, where a maximum improvement of 8.4% compared with the control group is achieved. Additionally, black pixels (pixel value 0) are used to fill the border, which is better than using white pixels (pixel value 255).

Keywords: tunnel; crack detection; deep learning; feature distribution

1 Introduction

Cracks are typical defects in road tunnels. Cracks not only affect the structural safety of tunnels, but also cause other problems, such as water seepage and corrosion of metal components [1, 2]. Crack detection is crucial for maintaining structural safety and infrastructure reliability. Classical manual crack-detection methods are time-consuming, laborious, dangerous, and subjective [3-5], which results in high maintenance costs, low maintenance efficiency, and high safety risks. Currently, the method of detecting cracks in tunnels by installing cameras on mobile platforms, such as cars and trucks, is the main detection method [6-10]. However, this detection method causes problems, such as a large data volume and the random distribution of cracks in the images. Tunnel inspection allows more than 10,000 images to be inspected per kilometer, and the manual naked-eye observation method can no longer complete the identification of tunnel crack images.

The quality of tunnel detection image data varies significantly depending on several factors, such as the lighting conditions and the lining surface color inside the tunnel. Numerous studies have shown that conventional image processing algorithms represented by a grayscale threshold [11], edge detection [12], wavelet transform [13], etc., are not suitable for crack detection in complex scenes inside tunnels [14]. In recent years, deep convolutional neural networks have demonstrated strong competitiveness in image classification, object detection, and image segmentation [15-17], as verified from the ImageNet large-scale visual recognition challenge [18]. Consequently, many excellent deep convolutional neural network models have been developed, such as AlexNet [19], GoogLeNet [20], VGG Net [21], and ResNet [22], whose performances are equivalent to or better than that of humans. Owing to such achievements, deep convolutional neural networks have been widely used in infrastructure disease detection. Dorafshan [23] compared the performances of conventional edge detection methods and deep convolutional neural networks on concrete-structured datasets and demonstrated the superior performance of deep convolutional neural networks in managing complex scenes. Ling [24] proposed a novel network architecture known as FPHBN, which integrates contextual information into low-level features in a feature pyramid configuration to perform crack detection and balances the contribution of simple and complex samples to the loss by layering sample weighting during training. Liu [25] proposed a two-stage convolutional neural network-based pavement crack detection and segmentation method, first using the YOLOv3 network for crack detection and then the U-Net for crack segmentation. Shen [26] used a fully convolutional neural network to perform a pixel-level segmentation of dense cracks using a modified fully convolutional neural network CrackSegNet, which comprises a backbone network, dilated convolution, spatial pyramid pooling, and skip connection modules.

These training strategies or deep convolutional neural networks allow one to effectively identify cracks at the center of an image. However, in the actual tunnel detection process, numerous cracks are located at the edge of an image, and the accuracy of the existing algorithms in identifying these cracks is unsatisfactory. Therefore, an image preprocessing method for edge crack recognition is proposed herein. Specifically, this method is used to change the relative position of edge cracks in an image, highlight the target information features, and ease the subsequent analysis and application of the image.

2 Related Studies

2.1 Network Architecture

The initial breakthrough in deep learning applied to object detection was achieved using ImageNet classified pretrained fine-tuned networks [27]. The feature learned in the pretraining task can convey useful information to the object detection task and then fine-tune the model without requiring a significant amount of training data. Pretraining has demonstrated excellent predictive performance for many tasks, including object detection, image segmentation, and action recognition.

YOLOv5 [28] is the latest version of the YOLO [29] family of algorithms proposed in 2020 to further optimize the speed and accuracy of object detection based on YOLOv4. To accommodate different scenarios, four pretraining models for YOLOv5 were released: yolov5s, yolov5m, yolov5l, and yolov5x. The yolov5s pretrained model, which contains 1001 convolution kernels and 12 residual components, is only 14.4 M, and features a width factor and depth factor of 0.5 and 0.33, respectively.

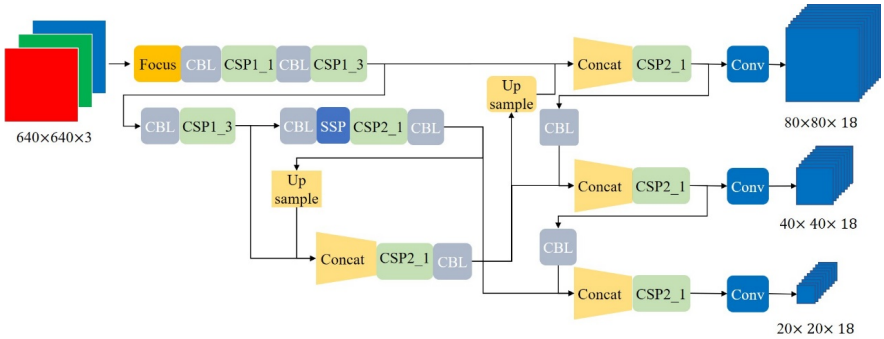


Figure 1
YOLOv5 network architecture

Figure 1 shows the network architecture of YOLOv5. For the backbone of YOLOv5, focus and CSP structures are used to improve the feature extraction speed and accuracy, whereas SPP and FPN+PAN modules are used to enhance the feature fusion capability [30]. The prediction frames of YOLOv5 features three sizes, which is the same as the case of YOLOv4. Because the input size is 640×640, the parameters of the detection head in the YOLOv5 measures 80×80×18, 40×40×18, and 20×20×18.

2.2 Activation Functions

The activation function is an important component of convolutional neural networks and has been shown to be key for achieving high performance in a wide range of tasks. It determines whether to activate a section of the neuron in the neural network and conveys the activation information to the next layer of the neural network. Therefore, the selected activation function significantly affects the training dynamics and performance [31]. To efficiently train the model and enable more accurate predictions on custom datasets, well-established popular activation functions, including SiLU, tanh, ReLU, and LeakyReLU, were configured for the training set. The selected activation function functions are described in detail below.

SiLU [32] is the most widely used class of activation function and is expressed as shown in Equation (1). Similar to other well-established activation functions, it can be easily implemented in the PyTorch [33] framework using wrapped code. Specifically, it features upper and lower bounds in the range (0, 1). Based on training experience, better prediction results can be obtained using the SiLU activation function; however, it is susceptible to overfitting. Although SiLU has been successfully used in the YOLOv5 model, a significant amount of data are required to train the model to reduce overfitting. Thus resulting in higher computational costs and a longer training time, which implies that it may not be the most suitable activation function on the feature dataset.

$$f(x) = x \cdot \sigma(x) \tag{1}$$

where $\sigma(x) = \left(1 + e^{-x}\right)^{-1}$ is the sigmoid function.

The mean value of the output of Tanh is zero, which allows the network to converge faster and reduces the number of iterations, as expressed in Equation (2). However, when the input values deviate significantly from the coordinate origin, the output is almost smooth and has a small gradient. Meanwhile, negative inputs are strongly mapped to negative values, and zero inputs are mapped close to zero, which is not conducive to updating the weights during the backpropagation.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

The ReLU [34] was proposed by Nair and Hinton. It solves the problem of gradient disappearance in tanh back propagation and improves the overall performance of a network. The ReLU is expressed as shown in Equation (3). Compared with SiLU and tanh, which requires exponential computation, the ReLU only determines the relationship between the magnitude of the input values and 0; therefore, it achieves a simpler forward inference and backpropagation in a model. However, the ReLU generates a zero gradient for all negative values of the input, where some important features in the negative values are disregarded; consequently, the permanent death of neurons occurs and the network fails to update appropriately during the backpropagation.

$$f(x) = max(0, x) \tag{3}$$

LeakyReLU [35], which is developed by Maas and Hannun, solves the problem of gradient disappearance and maintains the update of weights during the propagation. The alpha parameter ensures that the gradient does not approach zero throughout the training process, thus improving training performance. The LeakyReLU is expressed as shown in Equation (4).

$$f(x) = \begin{cases} x, x > 0 \\ \alpha x, x \le 0 \end{cases} \tag{4}$$

2.3 Proposed Method

During the actual detection process, cracks at the edge of the image were not easily detected, and the closer the relative position to the center of the image, the higher was the confidence level of the detection results. The basic idea of the method proposed herein is to highlight the features of the target information by changing the relative position of the crack region in the image such that the cracks are more concentrated in the center of the image. Therefore, a preprocessing method to add borders to an image is proposed herein. The steps involved in the preprocessing method are as follows:

1) The images in the dataset are converted to grayscale and the size information of each grayscale image is obtained.

2) The size of the border to be added is set based on the size information of the grayscale image.

3) The target position information is regenerated based on the relative position relationship.

4) The added border is filled with pixel values of 0 (black) and 255 (white).

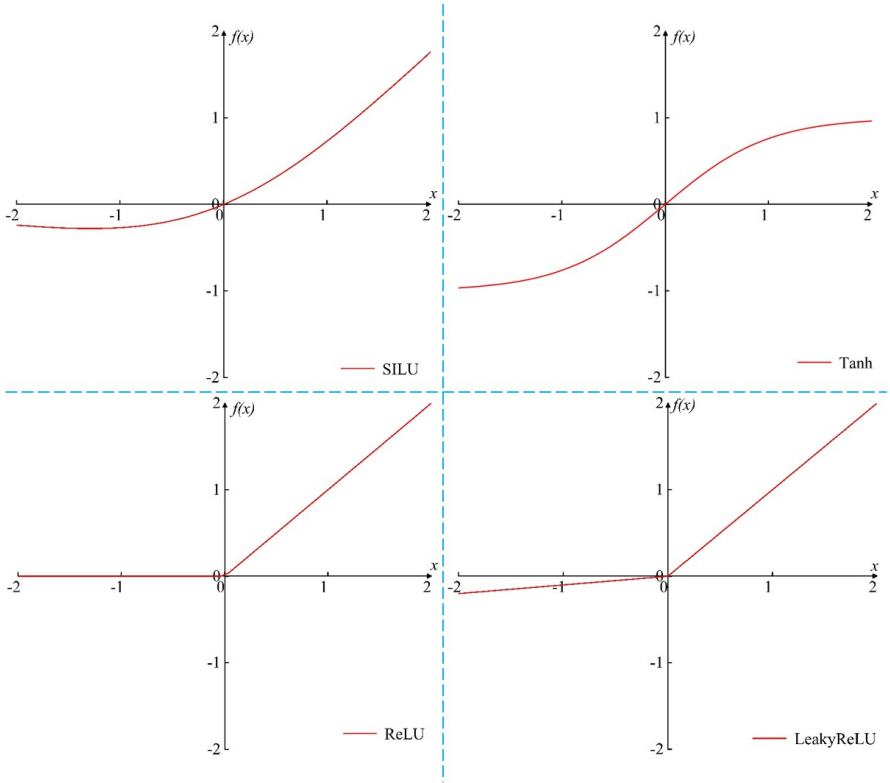


Figure 2
Plots of different activation functions

3 Experimental Design

3.1 Data Preparation

We established a road tunnel lining image dataset to perform training and validation. The dataset comprises a Crack Forest Dataset [36] and lining images obtained from an actual tunnel; the images contain a crack object. To avoid overfitting during training, image enhancement including flipping and rotation was performed on the dataset. Before training, the image data were manually labeled, and some of the annotated images were used as the training set for training. Whereas the remainder were, as the validation set, to evaluate the training accuracy by comparing the prediction results with the results of the training model. We used the well-established labelme tool to label 1744 images and then segmented the dataset randomly into two groups: one with 1494 images

as the training set, and one with 250 images as the validation set. The labelme labeling tool yields outputs in the JSON format, whereas YOLOv5 requires txt format labels for training. Therefore, conversion between JSON and txt formats is required before training, after which txt format labels and images are stored separately.

We added borders of different sizes to the original dataset while transforming the label information based on position relationships. The border sizes were 1/10, 1/9, 1/8, and 1/7 of the length and width of the original image, and the fill pixel values were 0 and 255. The experimental group was defined based on the border size and border fill pixel value. Subsequently, eight categories were defined: (1/10, black), (1/10, white), (1/9, black), (1/9, white), (1/8, black), (1/8, white), (1/7, black), and (1/7, white).

The network was trained based on the following computer hardware environment: Intel i7-11700, Nvidia Tesla T4 16G, and 32 GB of RAM. In terms of the development environment, CUDA 10.2, Python 3.7.4, and PyTorch 1.11.0 were adopted.

3.2 Model Evaluation

In this study, the detection evaluation metrics of the COCO dataset [37] were used to evaluate the crack detection models. The evaluation metrics of COCO are widely used to evaluate computer vision tasks and are among the most mainstream object detection evaluation metrics. Because the dataset in this study contains only one type of object, two metrics of the COCO evaluation were selected to evaluate the crack detection models, i.e., mAP@50 and mAP@50:95.

Precision and recall are widely used to evaluate classification models. Precision represents the number of objects detected by the model as true objects, and recall represents the number of true objects detected by the model.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

In addition, the object detection task introduces the interaction over union (IoU), which is a key concept in detection evaluation metrics. The annotation and prediction of object detection are represented by bounding boxes that create an overlap area between these two regions, and IoU represents the ratio of the intersection and union of the two regions.

$$IoU = \frac{\left(A_p \cap A_{gt}\right)}{\left(A_p \cup A_{gt}\right)} \tag{7}$$

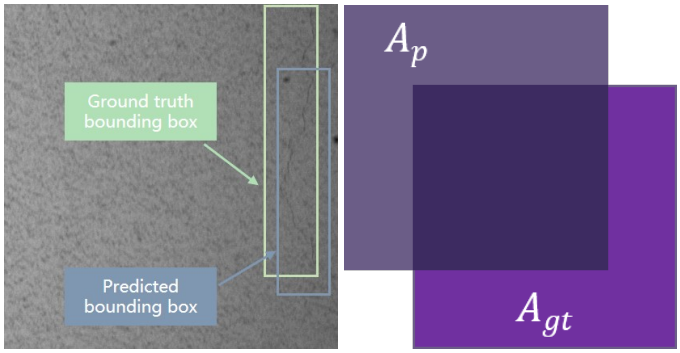


Figure 3
Definition of IoU

3.3 Training Schemes

In this study, four different crack-detection models were obtained by modifying the activation function of the YOLOv5 network: Model-1 (SiLU), Model-2(Tanh), Model-3 (ReLU), and Model-4 (LeakyReLU). To verify the enhancement effect of the proposed image preprocessing method for edge crack identification, the dataset was fed into the network for training. During training, both the control and experimental groups were trained using an official YOLOv5s pretraining model with a batch setting of 4, which did not require much adjustment or interference with the network. The training epochs were set to 300, the size of both the training and validation images was set to 640 × 640, and the remaining parameters were set at default values for training. This training scheme was trained on four different crack identification models, and 36 sets of experiments were conducted.

4 Results and Discussion

4.1 Effect of Board Size

The training results were obtained by performing the training schemes shown in Tables 1 and Table 2. For the Model-1 crack detection model, the mAP@50 did not improve significantly compared with that of the control group after adding different sizes of borders to change the relative positions of the edge cracks. Based on a border size of 1/7 and using the white fill value, the mAP@50 metric of Model-1 was 0.2% lower than that of the control group. However, the mAP@50:95 metric of Model-1 was better than that of the control group after adding different sizes of borders, where the maximum improvement was 3.6%. The different performances of mAP@50 and mAP@50:95 indicate that the image

preprocessing method performs better under high IoU conditions and that the detection bounding boxes are closer to the ground truth.

Table 2
mAP50 training metrics

Size	Value	Model-1	Model-2	Model-3	Model-4
-	-	0.933	0.856	0.931	0.922
1/10	black	0.935	0.892	0.940	0.942
	white	0.936	0.897	0.939	0.944
1/9	black	0.939	0.916	0.950	0.949
	white	0.936	0.918	0.941	0.947
1/8	black	0.937	0.905	0.938	0.940
	white	0.933	0.907	0.936	0.937
1/7	black	0.933	0.901	0.941	0.936
	white	0.931	0.898	0.936	0.931

Table 2
mAP50:95 training metrics

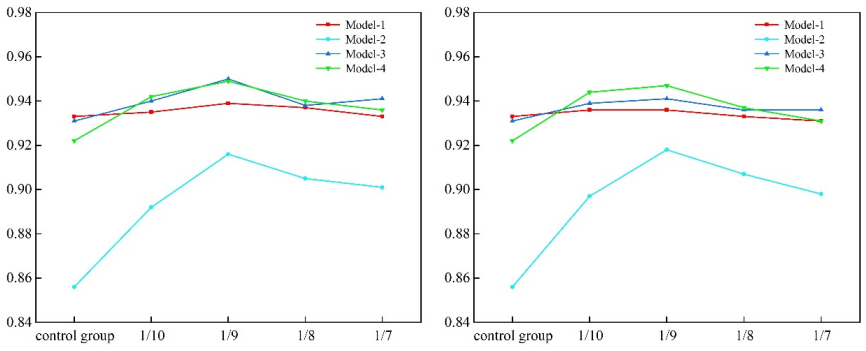
Size	Value	Model-1	Model-2	Model-3	Model-4
-	-	0.693	0.534	0.671	0.677
1/10	black	0.715	0.599	0.702	0.705
	white	0.718	0.592	0.694	0.706
1/9	black	0.729	0.618	0.712	0.711
	white	0.724	0.617	0.692	0.704
1/8	black	0.723	0.606	0.691	0.705
	white	0.720	0.600	0.688	0.691
1/7	black	0.722	0.616	0.697	0.709
	white	0.706	0.577	0.689	0.692

Because the Model-3 and Model-4 crack detection models differed by only their activation response values when negative inputs were used, the values of the final training indicators were similar. When the frame size was set to 1/9, Model-3 achieved the highest value for mAP@50 among the 36 sets of tests, with a maximum value of 95.0%. Meanwhile, the mAP@50 of Model-4 was 94.9%, which differed by only 0.1% from that of Model-3.

For the Model-2 detection model, its mAP@50 and mAP@50:95 values were 85.6% and 53.4% higher than those of the control group, respectively. After adding different sizes of borders to change the relative position of the edge cracks, the training metrics improved to varying degrees. Among them, mAP@50 and mAP@50:95 indicated the highest values of 91.8% and 61.8%, respectively, which were 6.2% and 8.4% higher than those of the control group, respectively, i.e., significantly surpassing the performances of the other three crack recognition models.

Figures 4 and 5 illustrate the effect of different border sizes on the training results. Under different border sizes, the four detection network models showed a consistent trend. As the border size increased gradually, the values of the training results first increased and then decreased. When the border size was set to 1/9 of the original image size, the mAP@50 and mAP@50:95 of the four different detection network models reached the highest values. As the border size increased further, the values of the metrics began to decrease but remained than those of the control group. When the border size was increased to 1/7, only the value of mAP@50 in Model-1 was lower than that in the control group, whereas the values of the final results for the other three models remained higher than those of the control group.

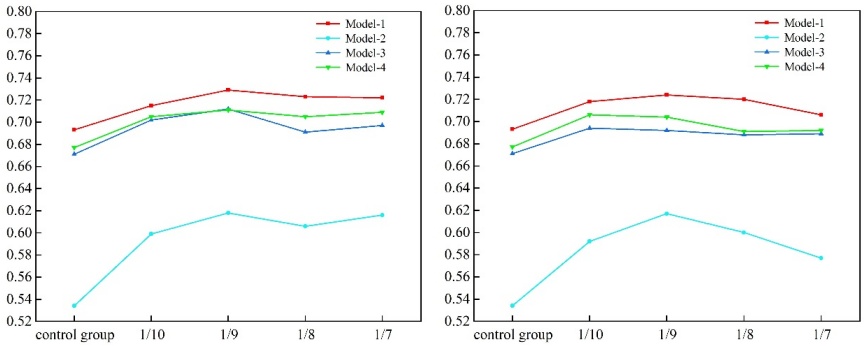
The analysis above indicates that adding a border to the dataset under the same training size reduces the object area of the crack. Additionally, the image preprocessing method, which is used to change the relative position of edge cracks based on the training results, can effectively improve the accuracy of the crack detection network. In particular, under high IoU conditions, the improvement in accuracy was more significant. This method demonstrated good applicability as well as desirable improvements in the four crack detection networks used in this study.



(a) The border is filled with black (b) The border is filled with white

Figure 4

The trend of mAP50 under different border sizes



(a) The border is filled with black (b) The border is filled with white

Figure 5
The trend of mAP50:95 under different border sizes

4.2 Effect of Board Padding Values

To verify the effect of the border fill pixel values on the recognition accuracy, two pixel values, i.e., those of black and white, were set at each border size. Tables 3 and 4 show the difference in the training metrics for two different fill pixel values at the same border size, with the difference calculated as the black-fill value minus the white-fill value. As shown in Table 3, the mAP@50 differed by a maximum of 0.9% and a minimum of -0.5%. Four of the groups with black pixel-value filled borders showed training results with lower values compared with those with white-filled pixel-value filled borders, whereas the remaining showed training results with higher values compared with those with white pixel-value filled borders.

In terms of mAP@50:95, only two groups of black-filled borders indicated lower final values compared with that of white-filled borders, with values of -0.3% and -0.1%, separately. In the remaining eight groups, the values indicated by the black-filled borders exceeded those of the white-filled borders, and the difference exceeded 1% in five groups, with a maximum difference of 3.9%.

A comparison of the differences between mAP@50 and mAP@50:95 shows that the effect of padding pixel values on mAP@50 was less significant than that on mAP@50:95 for different border sizes. For Model-3, the training results with black padding in the border were always better than those with white padding for image preprocessing using any of the border sizes. For the remaining three detection models, at least one set of training results with black padding was worse than those with white padding. In particular, for Model-2, the mAP@50 with black fill was lower than those with white fill for border sizes of 1/10, 1/9, and 1/8; however, a completely opposite performance was indicated for mAP@50:95.

The analysis above shows that the training result obtained using black for border filling based on the same border size is better than using white for filling. Whereas, the border filling value imposes a greater effect on metrics in the high IoU condition, which can reach up to 3.9%.

Table 3

The difference between the mAP@50 of two different fill pixel values

Size	Model-1(%)	Model-2(%)	Model-3(%)	Model-4(%)
1/10	-0.1	-0.5	0.1	0.2
1/9	0.3	-0.2	0.9	0.2
1/8	0.4	-0.2	0.2	0.3
1/7	0.2	0.3	0.5	0.5

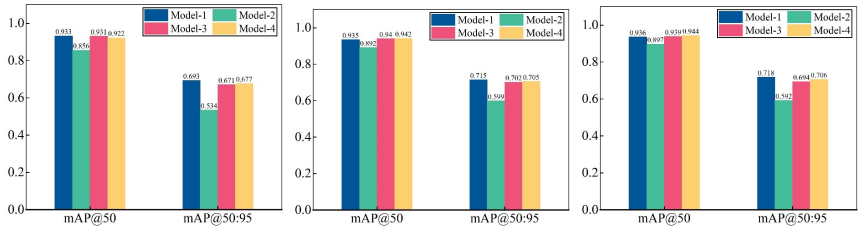
Table 4

The difference between the mAP@50 of two different fill pixel values

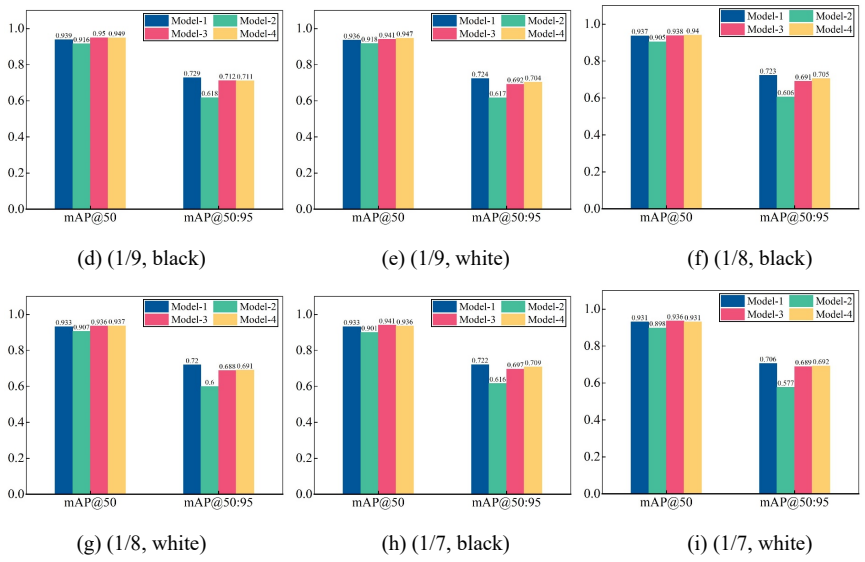
Size	Model-1(%)	Model-2(%)	Model-3(%)	Model-4(%)
1/10	-0.3	0.7	0.8	-0.1
1/9	0.5	0.1	2.0	0.7
1/8	0.3	0.6	0.3	1.4
1/7	1.6	3.9	0.8	1.7

4.3 Effect of Activation Functions

Figure 6 shows the crack detection results for the control and experimental groups. By transforming different activation functions, the crack detection network performs differently on the same dataset. In the control group, Model-1 outperformed the other three crack detection networks in terms of all recognition metrics. After image preprocessing, the mAP@50 of Model-3 began to exceed that of Model-1. As the border size increased gradually, the difference between the mAP@50 training results of the two models increased and reached a maximum of 1.1%. However, the mAP@50:95 remained better than that of the remaining three crack detection networks, which implies that the crack detection network using the SiLU activation function can yield more accurate prediction results for different scenes.



(a) control group (b) (1/10, black) (c) (1/10, white)



(d) (1/9, black) (e) (1/9, white) (f) (1/8, black)

(g) (1/8, white) (h) (1/7, black) (i) (1/7, white)

Figure 6
Crack detection results

For the control group and the preprocessed railway tunnel crack dataset, Model-3 using the ReLU activation function and Model-4 using the LeakyReLU activation function did not indicate significantly different training results, indicating that the ability of the two models in predicting road tunnel cracks as actual cracks was comparable. Among the four models, Model-2 performed the worst in the nine datasets, particularly in the control group; the difference between the mAP@50 and mAP@50:95 of Model-1 and the control group was 7.7% and 15.9%, respectively. We speculate that this situation occurs because the relatively high negative response value of the tanh activation function for processing negative inputs causes the non-convergence of the crack detection network, thus causing the training metrics to be worse than those of the other three crack detection networks. Therefore, we transformed the tanh activation function into a tanhshrink activation function with a higher negative response value when processing the negative inputs. We selected the control group, (1/10, black), and (1/9, black) for verification, whose training parameters are consistent with Model-2. Meanwhile, the crack detection network using the tanhshrink activation function is named Model-5, and the training results are shown in Table 5.

Table 5
Comparison of Model-2 and Model-5 activation function results

Dataset	Control group		(1/10, black)		(1/9, black)	
Model	Model-2	Model-5	Model-2	Model-5	Model-2	Model-5
mAP@50	0.856	0.621	0.892	0.567	0.916	0.746
mAP@50:95	0.534	0.293	0.599	0.223	0.618	0.386

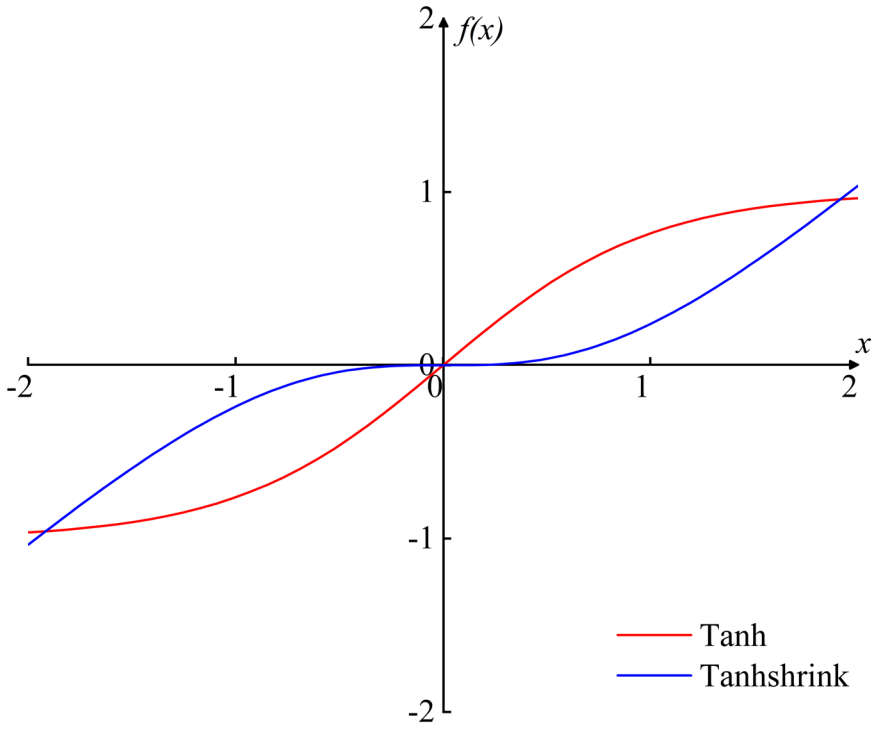
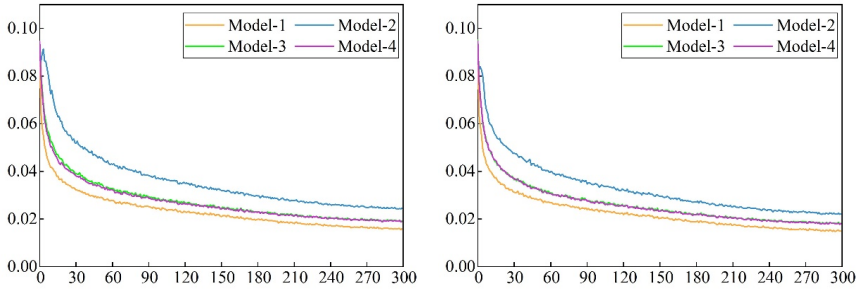


Figure 7
Plots of tanhshrink activation functions

Based on the evaluation results of the supplementary experiments, the values of the training metrics of Model-5 on three different datasets were significantly lower than those of Model-2; its mAP@50 was only 81.4% of that of Model-2 at the maximum, whereas its mAP@50:95 was less than 50% of that of Model-2 on the (1/10, black) dataset, which is sufficient to prove that the negative response value is not conducive to the convergence of the network.

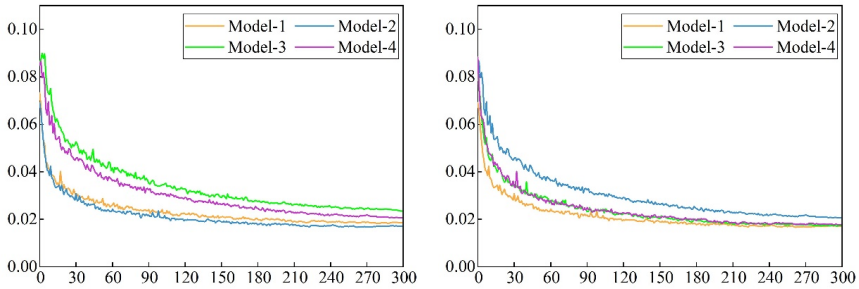
Figures 8 and 9 show the training and validation loss curve trends of the four crack recognition models on the control group and (1/9, black) dataset. As shown in Fig. 8, the loss curves of Model-1, Model-3, and Model-4 declined rapidly at the beginning of training. Whereas, that of Model-2 indicated a transient increase before declining; nonetheless, the convergence rate was low. The loss curves of Model-3 and Model-4 almost coincided, indicating that the two models exhibited significant similarities in the crack detection task. At the end of training, Model-2 and Model-1 indicated the highest and lowest loss values, respectively. Generally, lower loss values indicate better training metrics.



(a) Training loss on control dataset (b) Training loss on (1/9,black) dataset

Figure 8

Four different crack detection models training loss curves



(a) Validation loss on control dataset (b) Validation loss on (1/9,black) dataset

Figure 9

Four different crack detection models validation loss curves

The four models shown in Fig. 9 indicate significant oscillations in their verification loss curves in the early stage of training. This is primarily because when the number of iterations in the "pretraining–fine-tuning" mode is low, the crack anchor predicted by the forward inference of the weighted parameter matrix differs from the actual crack anchor. The training weights at this time cannot be effectively used for crack detection. As the number of iterations increased, the loss curve oscillation of the four models became milder and more stable, whereas the loss curve of Model-1 decreased rapidly at the highest convergence speed; consequently, the model was the first to converge at the 120th epoch. When the training epoch reached 240 rounds, the other three models converged. Because the validation set was defined as the test set, the validation loss was more reliable than the training loss when evaluating the prediction performance of different models. Based on the analysis above, the training results yielded by transforming the activation function of the network in the crack detection task were completely different. Model-1, which uses the SiLU activation function, can yield more accurate prediction results in different scenarios and converge the fastest. Model-2 performed unsatisfactorily in the crack detection task of railway tunnels, and the

high negative response value of the tanh activation function was not conducive to the convergence of the network.

Conclusions

We proposed an image preprocessing method to improve the recognition effect of a tunnel crack model. In the method, different sizes of borders were added to the training set and test set data, thereby changing the relative position of edge crack targets in the image. Furthermore, testing was performed using different recognition models, and the conclusions obtained are as follows:

1) The image preprocessing method proposed herein effectively improved the training metrics of the railway tunnel crack detection model, and as the frame size increased gradually, the training results of different models improved. When the border size was 1/9, the results for the training metrics of the four crack detection networks were optimal. Compared with the control group, Model-2 indicated higher values of mAP@50 and mAP@50:95 by 6.2% and 8.4%, respectively.

2) Based on a comparative analysis of the border fill pixel values, the values of mAP@50 under the same border size were similar, with a maximum difference of only 0.9%. Meanwhile, the mAP@50:95 value of the group with black fill was slightly lower than that of the group with white fill. Whereas, those of the remaining groups were higher than that of the group with white fill, where a maximum difference of 3.9% was indicated. Based on the difference indicated, the border fill pixel value imposed a more significant effect on the training indicators of high IOU conditions, and the effect of black fill was better than that of white fill.

3) In the crack detection experiment, Model-1 performed most stably in different scenarios as well as yielded more accurate prediction results and converged faster. The high negative response value of the tanh activation function was not conducive to the prediction performance of Model-2 in the crack detection task. Therefore, for railway tunnel crack-recognition models, functions with high negative response values should be avoided.

References

[1] M. Gavilan. Adaptive Road Crack Detection System by Pavement Classification. SENSORS, Vol. 11, No. 10, pp. 9628-9657, Oct. 2011, doi: 10.3390/s111009628

[2] A. Németh, S. K. Ibrahim, M. Movahedi Rad, S. Szalai. Laboratory and Numerical Investigation of Pre-Tensioned Reinforced Concrete Railway Sleepers Combined with Plastic Fiber Reinforcement. Polymers 2024, Vol. 16, No. 11, 2024, doi: 10.3390/polym16111498

[3] K. H, S. S. H, C. S. Unmanned Aerial Vehicle (UAV)-powered concrete crack detection based on digital image processing. 6th International

Conference on Advances in Experimental Structural Engineering, Aug. 2015

[4] Yuan Mei, Xinyu Tian, Xuejuan Li, Khaled Gepreel. Fractal space based dimensionless analysis of the surface settlement induced bz the shield tunneling. Facta Universitatis, Series: Mechanical Engineering, Vol. 21, No. 04, pp. 737-749, 2023, DOI:10.22190/FUME230826048M

[5] P. Wang, Y. Hu, Y. Dai, M. Tian. Asphalt Pavement Pothole Detection and Segmentation Based on Wavelet Energy Field. MATHEMATICAL PROBLEMS IN ENGINEERING, Vol. 2017, 2017, doi: 10.1155/2017/1604130

[6] H. Huang, Y. Sun, Y. Xue. Research progress of machine vision based disease detecting techniques for the tunnel lining surface. ModernTunn Technol, Vol. 51, No. s1, pp. 19-31, 2014

[7] S. Fischer, S. K. Szürke, Detection Process of Energy Loss in Electric Railway Vehicles, Facta Universitatis, Series: Mechanical Engineering, Vol. 21, No. 1, pp. 81-99, 2023, DOI:10.22190/FUME221104046F

[8] Y. Xue, Y. Li. A method of disease recognition for shield tunnel lining based on deep learning. J Hunan Univ (Nat Sci), Vol. 45, No. 03, pp. 100-109, 2018

[9] Szabolcs Fischer. Evaluation of inner shear resistance of layers from mineral granular materials. Facta Universitatis, Series: Mechanical Engineering, DOI:10.22190/FUME230914041F

[10] László Ézsiás, Richárd Tompa, Szabolcs Fischer. Investigation of the Possible Correlations between Specific Characteristics of Crushed Stone Aggregates. Spectrum of Mechanical Engineering and Operational Research, Vol. 1, No. 1, pp. 10-26, 2024

[11] Q. Li, X. Liu. Novel approach to pavement image segmentation based on neighboring difference histogram method. presented at the CISP 2008: FIRST INTERNATIONAL CONGRESS ON IMAGE AND SIGNAL PROCESSING, VOL. 2, PROCEEDINGS, 2008, pp. 792-796. doi: 10.1109/CISP.2008.13

[12] L. Abdel-Qader, O. Abudayyeh, M. Kelly. Analysis of edge-detection techniques for crack identification in bridges. JOURNAL OF COMPUTING IN CIVIL ENGINEERING, Vol. 17, No. 4, pp. 255-263, Oct. 2003, doi: 10.1061/(ASCE)0887-3801(2003)17:4(255)

[13] L. Ying, E. Salari. Beamlet Transform-Based Technique for Pavement Crack Detection and Classification. COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING, Vol. 25, No. 8, pp. 572-580, Nov. 2010, doi: 10.1111/j.1467-8667.2010.00674.x

[14] Q. Gong, L. Zhu, Y. Wang, Z. Yu. Automatic subway tunnel crack detection system based on line scan camera. STRUCTURAL CONTROL & HEALTH MONITORING, Vol. 28, No. 8, Aug. 2021, doi: 10.1002/stc.2776

[15] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. PROCEEDINGS OF THE IEEE, Vol. 86, No. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791

[16] L. Mahmood, Z. Bahroun, M. Ghommem, H. Alshraideh, Assessment and Performance Analysis of Machine Learning Techniques for Gas Sensing E-nose Systems. Facta Universitatis, Series: Mechanical Engineering, Vol. 20, No. 03, pp. 479-501, 2022, DOI:10.22190/FUME220307022M

[17] M. Zeiler, R. Fergus. Visualizing and Understanding Convolutional Networks. presented at the COMPUTER VISION - ECCV 2014, PT I, 2014, Vol. 8689, pp. 818-833

[18] J. Deng. ImageNet: A Large-Scale Hierarchical Image Database. presented at the CVPR: 2009 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, VOLS 1-4, 2009, pp. 248-255

[19] A. Krizhevsky, Ii. Sutskever. ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems, Vol. 25, No. 2012, pp. 1097-1105

[20] C. Szegedy. Going Deeper with Convolutions. presented at the 2015 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2015, pp. 1-9

[21] K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition. The 3rd International Conference on Learning Representations (ICLR), 2015

[22] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. presented at the 2016 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, pp. 770-778

[23] S. Dorafshan, R. Thomas, M. Maguire. Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. CONSTRUCTION AND BUILDING MATERIALS, Vol. 186, pp. 1031-1045, Oct. 2018

[24] F. Yang, L. Zhang, S. Yu. Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, Vol. 21, No. 4, pp. 1525-1535, Apr. 2020

[25] J. Liu. Automated pavement crack detection and segmentation based on two-step convolutional neural network. COMPUTER-AIDED CIVIL AND

INFRASTRUCTURE ENGINEERING, Vol. 35, No. 11, pp. 1291-1305, Nov. 2020

[26] Y. Ren. Image-based concrete crack detection in tunnels using deep fully convolutional networks. CONSTRUCTION AND BUILDING MATERIALS, Vol. 234, Feb. 2020

[27] K. He, R. Girshick, P. Dollar. Rethinking ImageNet Pre-training. presented at the 2019 IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV 2019), 2019, pp. 4917-4926

[28] L. Kou, M. Sysyn, J. X. Liu, Influence of crossing wear on rolling contact fatigue damage of frog rail, Facta Universitatis, Series: Mechanical Engineering, Vol. 22, No. 1, 2024, DOI:10.22190/FUME220106024K

[29] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. presented at the 2016 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2016, pp. 779-788

[30] A. Bochkovskiy. YOLOv4: Optimal Speed and Accuracy of Object Detection. 202

[31] S. Qian, H. Liu, C. Liu. Adaptive activation functions in convolutional neural networks. NEUROCOMPUTING, Vol. 272, pp. 204-212, Dec. 2018

[32] S. Elfwing, E. Uchibe, K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. NEURAL NETWORKS, Vol. 107, pp. 3-11, Nov. 2018

[33] A. Paszke, S. Gross, F. Massa. PyTorch: An Imperative Style, High Performance Deep Learning Library. 2019

[34] V. Nair. Rectified linear units improve restricted Boltzmann machines. In ICML, pp. 807-814, 2010

[35] A. L. Maas. Rectifier nonlinearities improve neural network acoustic models. In ICML, Vol. 30, 2013

[36] Y. Shi, L. Cui, Z. Qi, F. Meng, Z. Chen. Automatic Road Crack Detection Using Random Structured Forests. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, Vol. 17, No. 12, pp. 3434-3445, Dec. 2016

[37] T. Lin. Microsoft COCO: Common Objects in Context. presented at the COMPUTER VISION - ECCV 2014, PT V, 2014, Vol. 8693, pp. 740-755