Hindawi
Modelling and Simulation in Engineering
Volume 2019, Article ID 8796743, 8 pages
https://doi.org/10.1155/2019/8796743

Research Article

Improved Crack Detection and Recognition Based on Convolutional Neural Network

Keqin Chen,[1,2] **Amit Yadav**,[2] **Asif Khan**,[3,4] **Yixin Meng**,[2] **and Kun Zhu**[2]

[1]*School of Business Administration, Southwestern University of Finance and Economics, Chengdu 611130, China*
[2]*Department of Information and Software Engineering, Chengdu Neusoft University, Dujianyan, Chengdu, Sichuan 611844, China*
[3]*Crescent Institute of Science and Technology, Vandalur, Chennai 600048, India*
[4]*University of Electronic Science and Technology of China, Chengdu, China*

Correspondence should be addressed to Keqin Chen; 3118268411@qq.com

Received 29 June 2019; Accepted 22 August 2019; Published 14 October 2019

Guest Editor: Qing-Feng Liu

Concrete cracks are very serious and potentially dangerous. There are three obvious limitations existing in the present machine learning methods: low recognition rate, low accuracy, and long time. Improved crack detection based on convolutional neural networks can automatically detect whether an image contains cracks and mark the location of the cracks, which can greatly improve the monitoring efficiency. Experimental results show that the Adam optimization algorithm and batch normalization (BN) algorithm can make the model converge faster and achieve the maximum accuracy of 99.71%.

1. Introduction

Traditional concrete which is used for various construction purposes such as buildings, bridges, and dams often age due to years of use which causes problems that affect the construction facilities. In order to avoid the problems caused by the aging of the facilities, it is necessary to continuously monitor and maintain the facilities. The traditional manual inspection method obviously cannot meet the huge road pavement inspection requirements. At present, many computer vision technologies realize the detection of cracks. Yiyang proposed a crack detection algorithm based on digital image processing technology [1]. Through preprocessing, image segmentation, and feature extraction, Yiyang obtained information about the crack image. The threshold segmentation method is used after smoothing the accepted input image. To determine their image, Yiyang calculated the area and circumference of the circle. Then, by comparison, Yiyang evaluated the presence of cracks in the image. Oliveira and Correia designed an automatic crack detection system [2]. Crack detection here is based on a sample. In the sample paradigm, a subset of the available image databases is automatically selected and used for unsupervised training of system images. They have characterized operations based on the classification of nonoverlapping image blocks. The width of the crack is then estimated based on the detection of the crack block. An improved dynamic programming-based algorithm is proposed to detect cracks [3]. The algorithm performs fast but has low accuracy. The Gabor filter is used to detect cracks [4], which has a good effect on the detection of simple pavement cracks and is severely broken for the detection of complex cracks. Zhang et al. [5] adopted four-layer convolutional neural network (CNN) to realize crack detection, with an accuracy of 87%, which needs to be improved. Zhang et al. [6] proposed a new region growth algorithm to detect road cracks, which is not suitable for detecting small and scattered cracks in the road. The extended finite element formula (XFEM) combined with the genetic algorithm (GA) has been proven to be effective in detecting structural defects [7–9], but this method also has many limitations. In this article, a novel method [10] is used to improve the convolutional neural network [11, 12], so that the convolutional neural network can automatically detect the crack in the image and

mark the corresponding position. The contribution of this article is twofold. The first system based on convolutional neural network is a good technique for detecting cracks in concrete structures, with an average accuracy of 99.71%. Secondly, the automatic detection system for historical buildings is introduced. In order to ensure the integrity of historical building structure, especially the degradation of masonry structure caused by aging and human activities, it has certain reference value for periodic inspection and maintenance of cultural relics. In practical, drones can easily detect ancient buildings, bridges, dams, and temples that are not easily monitored by humans. A convolutional neural network is used to collect images, detect the presence of cracks, and mark the corresponding cracks.

2. Improvement Based on Convolutional Neural Network

2.1. Classical Convolutional Neural Network. Relative to traditional neural network back propagation (BP neural network) [12, 13], the use of weight sharing in convolutional neural networks can greatly reduce network parameters and accelerate the training speed of the network. The performance is stronger than that of BP neural networks. Convolutional neural networks have powerful feature extraction capabilities. Using the advantages of convolutional neural networks, convolutional neural networks are widely used in various fields, including image classification, object detection, autopilot, and image style migration.

A classic convolutional neural network usually includes an input layer, a convolutional layer, a pooled layer (under the layer), a fully connected layer, and an output layer, as shown in Figure 1.

2.1.1. Convolution Layer

$$H_i = f\left(H_{i-1} \otimes W_i + b_i\right), \tag{1}$$

where i indicates the number of layers in the network, H_{i-1} represents the upper layer, $i-1$ indicates the layer output value, and w_i indicates the weight of the network at the i level. The weight is generally carried out by random initialization. The "⊗" symbol indicates a convolution operation, where f indicates that the result of the convolution is input to an activation function that is nonlinearized.

2.1.2. Pooling Layer. The pooling layer includes max pooling, mean pooling, and random pooling. Here, the maximum pooling used in this paper is mainly discussed. Maximum pooling, selects the maximum value in the area within a certain area.

2.2. Improvement Based on Convolutional Nerve. Because the crack to be identified is quite different from the background color of the picture, the task of identification is relatively simple. Therefore, the convolution kernel and pooling size used in the network are generally large, and the network structure is simple. The network is batch normalized (BN) [14] after each layer of pooling. The mathematical expression of batch normalization is normalized, as shown in the formula (2)–(5). The input is normalized to a data distribution with an average of 0 and a variance of 1. Then, the normalization operation destroys the possible data distribution of the data itself and performs matrix shrinking and translation on the normalized result to recover the data distribution characteristics that the data itself may have. Batch normalization can avoid the model overfitting to some extent. At the same time, it can accelerate the convergence of the model and improve the stability of training.

$$\mu_{\mathscr{B}} \longleftarrow \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{2}$$

$$\sigma_{\mathscr{B}}^2 \longleftarrow \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu_{\mathscr{B}}\right)^2, \tag{3}$$

$$\widehat{x}_i \longleftarrow \frac{x_i - \mu_{\mathscr{B}}}{\sqrt{\sigma_{\mathscr{B}}^2 + \in}}, \tag{4}$$

$$y_i \longleftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}\left(x_i\right). \tag{5}$$

2.2.1. Adam Algorithm. In terms of the choice of optimizer for model training, instead of the traditional SGD [15] optimization using the faster convergence Adam [16] algorithm, Adam was submitted by OpenAI's Diederik Kingma and the University of Toronto's Jimmy Ba to the 2015 ICLR paper (Adam), "A Method for Stochastic Optimization." Adam can set a learning rate for each weight parameter, giving a smaller learning rate for those weights that are updated more frequently and giving a larger learning rate for those with fewer updates, while updating the parameters. The momentum technique is introduced to consider the past gradient in the parameter update, so that the updated value of the current time is composed of the gradient and the last updated gradient, which can effectively reduce the oscillation of the loss function, stabilize the training process, and speed up the training. The updated formula is as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{\widehat{v}}_t + \in}}\widehat{m}_t, \tag{6}$$

where θ represents the parameters in the network, the number of iterations of the t, η represents the network learning rate, and ε is a small floating-point number, which is used to avoid meaningless operations when the denominator $[\![\sqrt{(\widehat{v})}]\!]_t$ is zero Also, $\widehat{m}_t$ represents the gradient after increasing the momentum factor, and the function of $\sqrt{\widehat{v}_t}$ is to set a separate learning rate for each parameter.

2.3. Network Structure. The network structure used in this paper is shown in Figure 2. The input is a 227-pixel × 227-pixel RGB 3-channel picture. After 24 convolution kernels with a size of 20 × 20 and a stride of 2, 114 × 114 is obtained. The tensor output of ×24, after 7 × 7, the pooling layer with a

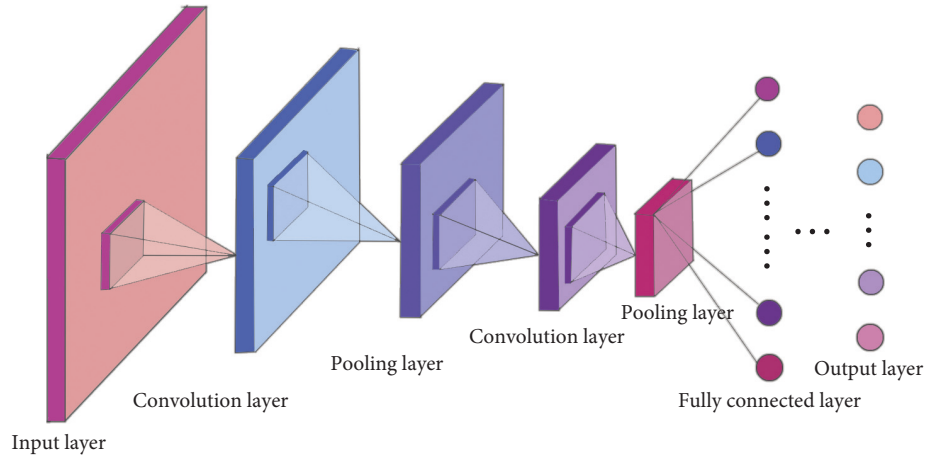


Figure 1: Typical convolutional neural network.

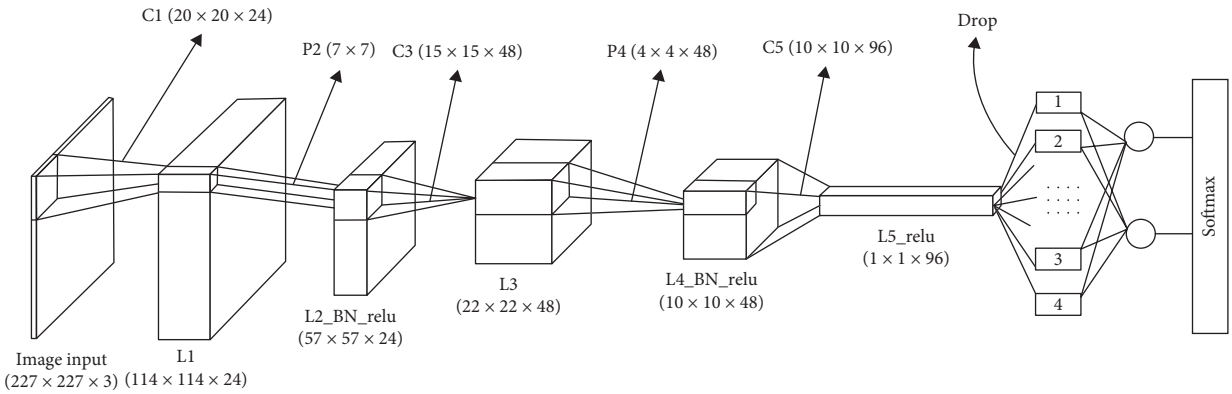


Figure 2: Network structure.

step size of 2 outputs $57 \times 57 \times 24$ tensor, then it is subjected to nonlinearization by batch normalization (BN) and relu activation function processing. Then, after 48 convolutions of 15×15 steps of 2, a tensor of $22 \times 22 \times 48$ is obtained, and after 4×4 steps of 2, a tensor of $10 \times 10 \times 48$ is obtained, also after the tensor. Similarly, after the tensor is obtained, it is processed by batch normalized processing (BN) and relu activation function for nonlinear processing, the activated $10 \times 10 \times 48$ tensor is obtained, and then 96 convolutions of 10×10 steps are obtained to obtain $1 \times 1 \times 96$ tensor. Then, after passing through the fully connected layer of size 2, it is then passed through softmax to convert the output value into a probability output to obtain the result. Note that there are many parameters in the fully connected layer. In order to avoid overfitting, dropout [17] is used to ignore some nodes randomly and proportionally, so that the node does not work in this calculation, and to avoid the overfitting to some extent. The specific network parameter configuration is shown in Table 1.

3. Empirical Research

3.1. Lab Environment

Translator: Python3.6

Operating environment: Spyder

Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.60 GHz 2.59 GHz

Running memory: 4G

Operating system: Windows 10 Enterprise

Machine learning framework: TensorFlow 1.8

3.2. Data Set. A training data set of concrete gap images provided by Mendeley [18], including specific images with cracks. The test data comes from Middle East Technical University campus building. The data set is divided into two categories, cracked and crack-free images. Each type of image has 20,000 images, and each image has RGB pixels of 227-pixels × 227-pixels, for a total of 40,000 images. The crack data set image is shown in Figure 3 below. These 40,000 images are cut from 458 high-resolution (4032×3024) images, which are mentioned in Figure 3. High-resolution images differ in terms of the lighting and the like.

3.3. Experimental Result. Before the experiment begins, the data are divided into a part of the verification data set. The size of the model output data batch is 32. Before image input, sample normalization of the image was conducted and label data were encoded by one-hot encoding. Use the network structure of Figure 2 for training. The loss value is printed

Table 1: Network parameter.

Layer	Size	Operator	Filters	Kernel_size	Strides	Padding
Input	$227 \times 227 \times 3$	C1	24	20×20	2	Same
L1	$114 \times 114 \times 24$	P2	—	7×7	2	Same
L2	$57 \times 57 \times 24$	BN	—	—	—	—
		relu	—	—	—	—
		C3	48	15×15	2	Valid
L3	$22 \times 22 \times 48$	P4	—	4×4	2	Valid
L4	$10 \times 10 \times 48$	BN	—	—	—	—
		relu	—	—	—	—
		C4	96	10×10	2	Valid
L5	$1 \times 1 \times 96$	relu	—	—	—	—
		Drop	—	—	—	—
L6	$1 \times 1 \times 2$	C5	2	1×1	1	Valid

The learning rate is set to 0.0001, and the learning rate drops to 0.99 for each iteration of 100; the dropout is set to 0.5.

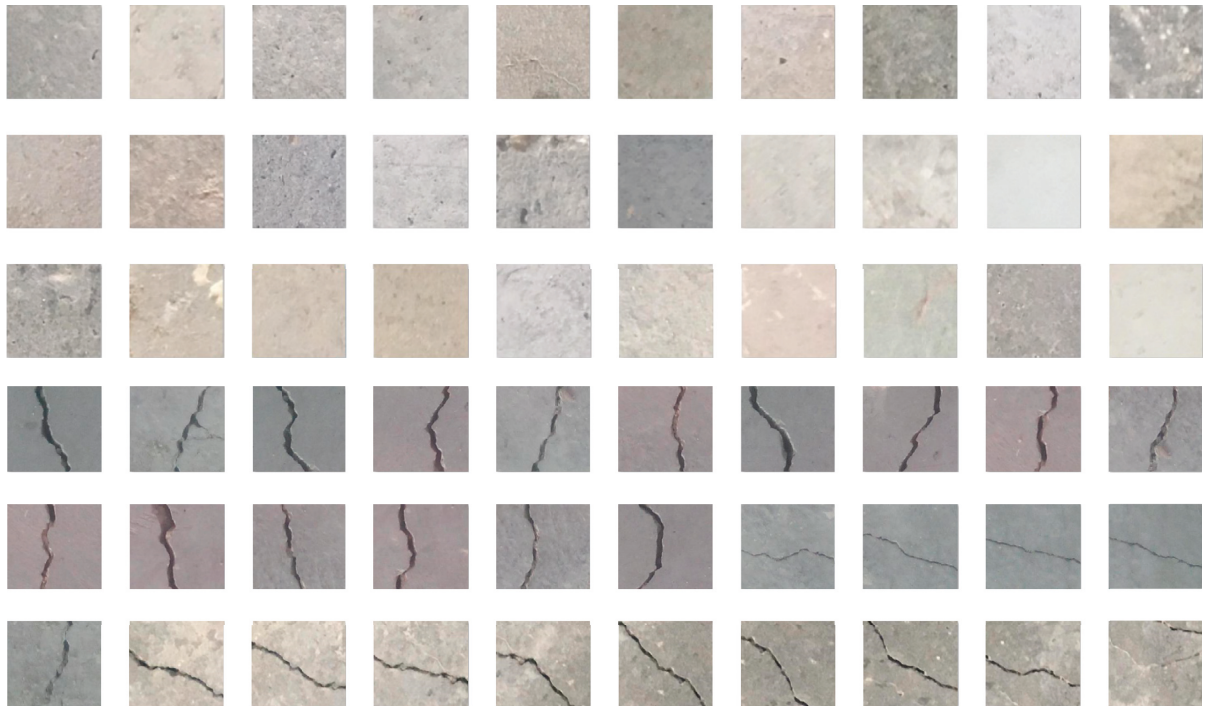


Figure 3: Crack image sample.

once in every 100 iterations, and the model is evaluated once every 1000 iterations using the verification data set, as shown in Figure 4. If the correct rate of this evaluation is greater than the last time, the model is retained, and if 100 consecutive times are less than the correct rate of the previous evaluation, then training is stopped, which experimentally proves that the use of Adam optimization algorithm and batch normalization (BN) can make the model converge faster.

After 110 k iteration, the program automatically stops training, and the correct rate of the model is about 99.71%. As can be seen from Figure 5, after 11 k iterations, the maximum correct rate is achieved. It takes about 50 minutes. As can be seen from the figure, due to the continuous reduction of the learning rate, the correct rate is gradually stabilized in the later stages of training.

Table 2 shows Zhang's ConvNet built through the Caffe framework and trained through the use of 5-fold cross-validation [5]. The results obtained from different methods i.e., SVM and Boost are compared. Using ConvNes and Adam, it can be seen that the accuracy result of each method is 0.8112, 0.736, 0.8696, and 0.9971, respectively. The most accurate result is that of the Adam method.

3.4. Recognition Effect Display and Analysis. The large size images used for verification come from Google Images, all containing cracks. Since the image resolution is much larger than the 227×227 input size of the model, the image is first cut into small images of 227×227 and placed into the model. After detection, the results are returned and the crack location is marked in the figure according to the returned results. The specific effect diagram is shown in Figure 6.

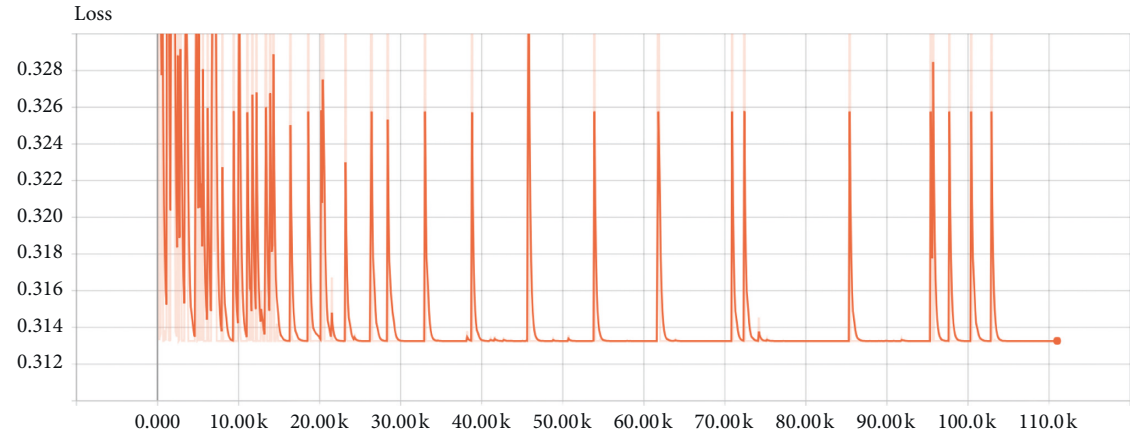


Figure 4: Printed loss value. Each iteration prints a loss of 100 times and evaluates the model once per 1000 iterations using the validation data set.

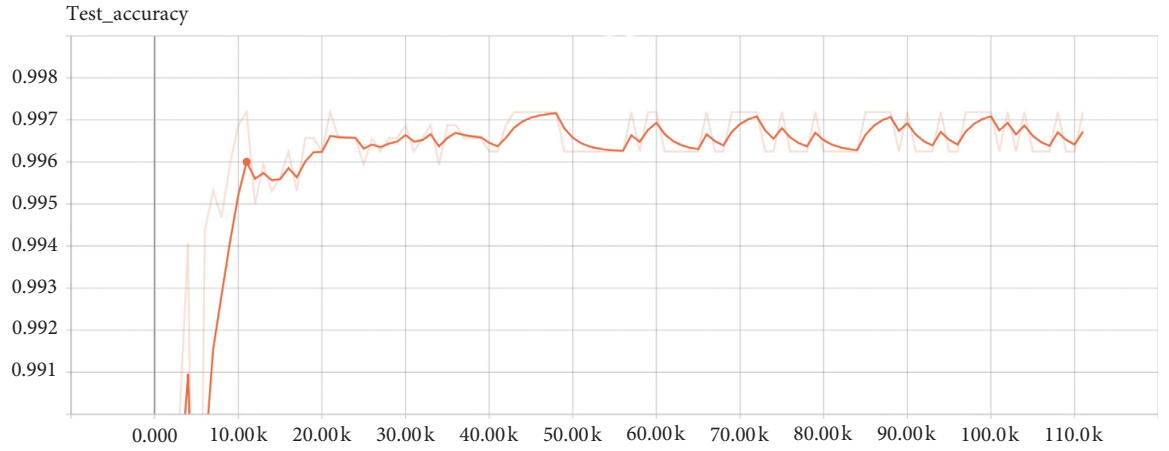
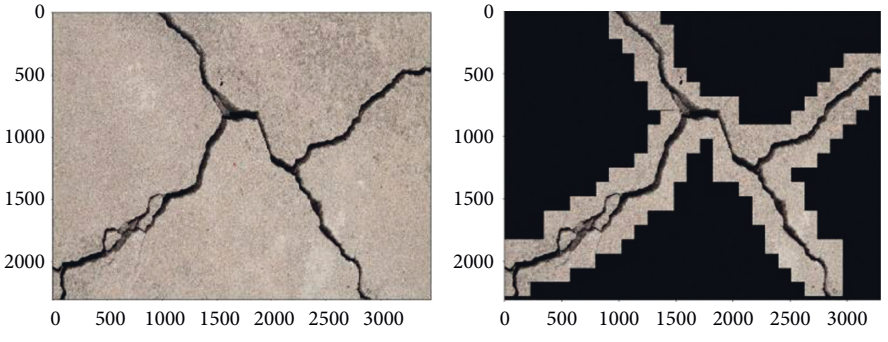


Figure 5: Correct rate change chart. The light line is the set of real data points, and the solid line is the set of points after the smoothing process.

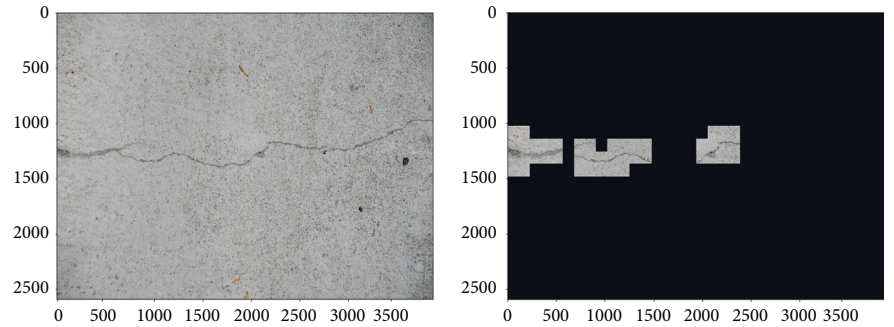
Table 2: Performance comparison of different methods.

Method	SVM	Boosting	ConvNets	Adam
Precision	0.8112	0.736	0.8696	0.9971

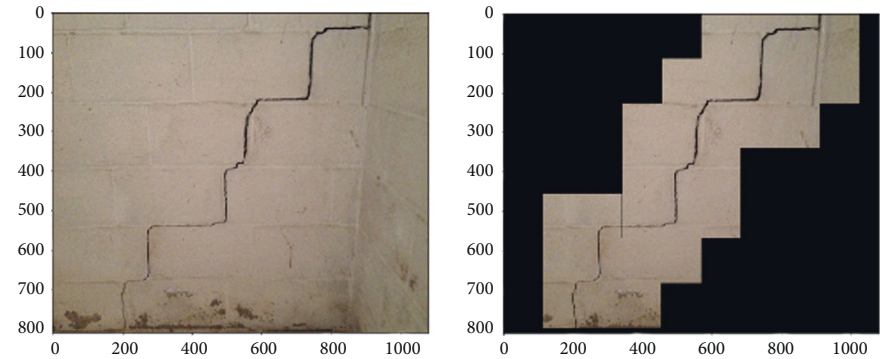


(a)

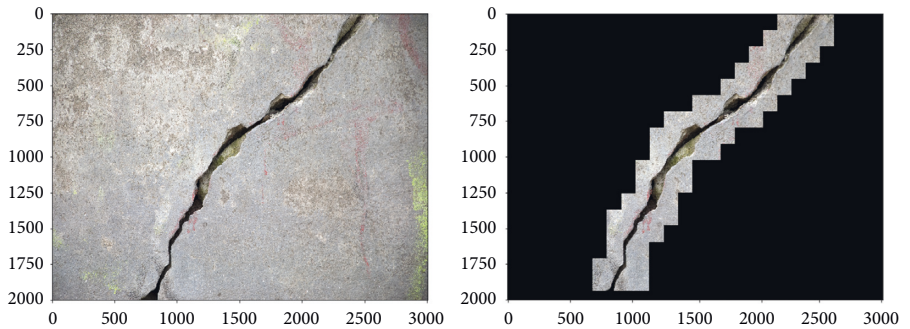
Figure 6: Continued.



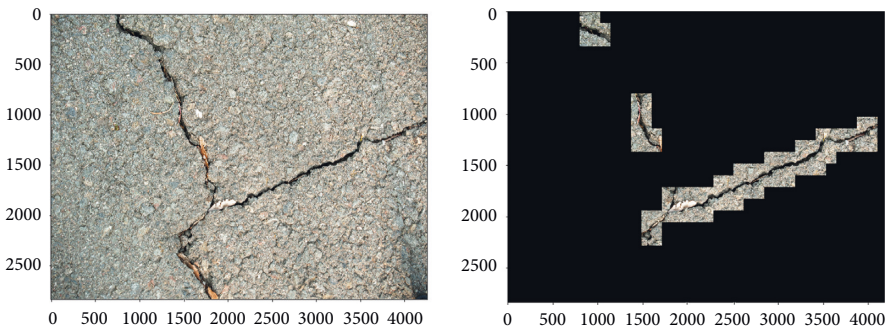
(b)



(c)



(d)



(e)

Figure 6: Continued.

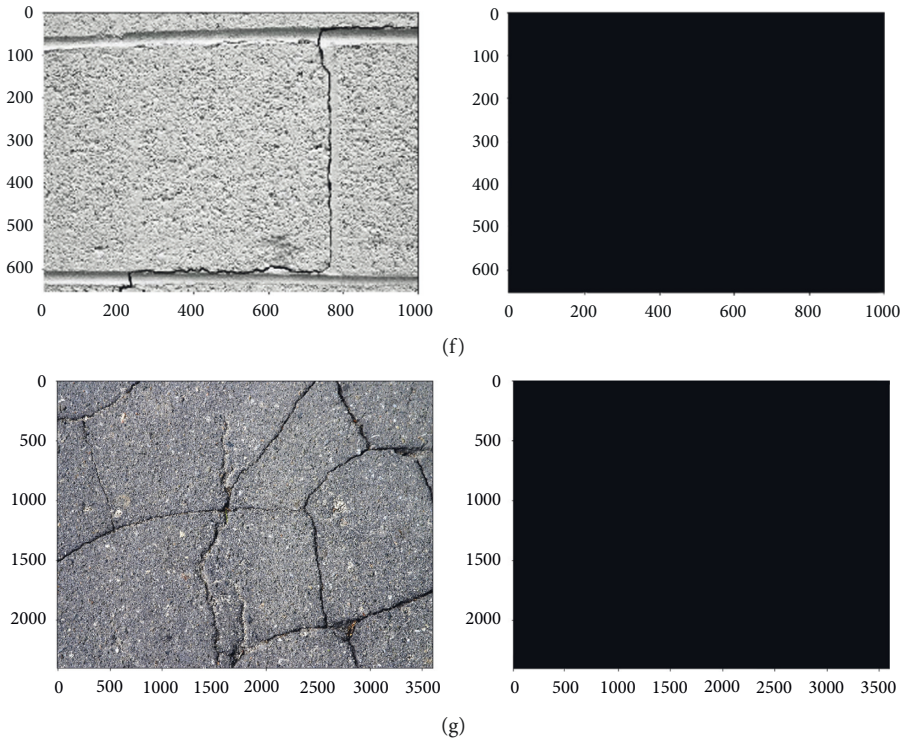


Figure 6: Crack recognition effect display.

By comparing the results of the picture, it can be seen that the model can better distinguish the picture with the contrast of the background color of the crack and the picture, as shown in Figures 6(a), 6(c), and 6(d). However, in Figures 6(b), 6(e), and 6(f), the identification effects are not properly visible for cracks which is unapparent. The reason behind this issue is that cracks are narrow and the colour of the cracks is similar to the background colour of the wall. The result is poor in Figure 6(g) as it does not identify a crack, which may be due to the model itself is based on the black interpretation of the image to determine whether there is a crack. The solution to this problem is to add black to the background in the training dataset or include relatively dark lines of cracks on similar pictures while doing analysis using the neural network.

4. Conclusion

In this paper, the use of relatively simple and improved convolutional neural networks has successfully achieved the identification of cracks and has a very high accuracy rate. For the analysis of relatively simple crack identification, it is suggested that the use of large convolution and pooling methodology with fewer network layers can be helpful to get better results.

Data Availability

The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Z. Yiyang, "The design of glass crack detection system based on image preprocessing technology," in *Proceedings of the 2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 39–42, IEEE, Chongqing, China, December 2014.

[2] H. Oliveira and P. L. Correia, "Automatic road crack detection and characterization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 155–168, 2013.

[3] Y. Huang and Y. Tsai, "Dynamic programming and connected component analysis for an enhanced pavement distress segmentation algorithm," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2225, no. 1, pp. 89–98, 2011.

[4] M. Salman, S. Mathavan, K. Kamal, and M. Rahman, "Pavement crack detection using the Gabor filter," in *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 2039–2044, IEEE, Hague, The Netherlands, October 2013.

[5] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3708–3712, IEEE, Phoenix, Arizona, September 2016.

[6] D. Zhang, Q. Li, Y. Chen, M. Cao, L. He, and B. Zhang, "An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection," *Image and Vision Computing*, vol. 57, pp. 130–146, 2017.

[7] S. Teidj, A. Khamlichi, and A. Driouach, "Identification of beam cracks by solution of an inverse problem," *Procedia Technology*, vol. 22, pp. 86–93, 2016.

[8] E. N. Chatzi, B. Hiriyur, H. Waisman, and A. W. Smyth, "Experimental application and enhancement of the XFEM–

GA algorithm for the detection of flaws in structures," *Computers & Structures*, vol. 89, no. 7-8, pp. 556–570, 2011.

[9] D. Rabinovich, D. Givoli, and S. Vigdergauz, "XFEM-based crack detection scheme using a genetic algorithm," *International Journal for Numerical Methods in Engineering*, vol. 71, no. 9, pp. 1051–1080, 2007.

[10] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.

[11] Y. LeCun, "LeNet-5, convolutional neural networks (2015)," 2016.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[13] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[14] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, http://arxiv.org/abs/1502.03167.

[15] M. A. Vorontsov and V. P. Sivokon, "Stochastic parallel-gradient-descent technique for high-resolution wave-front phase-distortion correction," *Journal of the Optical Society of America A*, vol. 15, no. 10, pp. 2745–2758, 1998.

[16] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] Dataset, http://dx.doi.org/10.17632/5y9wdsg2zt.1#file-c0d86f9f-852e-4d00-bf45-9a0e24e3b932.