Hindawi
Advances in Civil Engineering
Volume 2019, Article ID 6520620, 12 pages
https://doi.org/10.1155/2019/6520620

Research Article

Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique

Shengyuan Li and Xuefeng Zhao

School of Civil Engineering, State Key Laboratory of Coastal and Offshore Engineering, Dalian University of Technology, 116023 Dalian, China

Correspondence should be addressed to Xuefeng Zhao; zhaoxf@dlut.edu.cn

Received 15 January 2019; Revised 24 March 2019; Accepted 16 April 2019; Published 30 April 2019

Academic Editor: Hayri Baytan Ozmen

Crack detection is important for the inspection and evaluation during the maintenance of concrete structures. However, conventional image-based methods need extract crack features using complex image preprocessing techniques, so it can lead to challenges when concrete surface contains various types of noise due to extensively varying real-world situations such as thin cracks, rough surface, shadows, etc. To overcome these challenges, this paper proposes an image-based crack detection method using a deep convolutional neural network (CNN). A CNN is designed through modifying AlexNet and then trained and validated using a built database with 60000 images. Through comparing validation accuracy under different base learning rates, 0.01 was chosen as the best base learning rate with the highest validation accuracy of 99.06%, and its training result is used in the following testing process. The robustness and adaptability of the trained CNN are tested on 205 images with 3120×4160 pixel resolutions which were not used for training and validation. The trained CNN is integrated into a smartphone application to mobile more public to detect cracks in practice. The results confirm that the proposed method can indeed detect cracks in images from real concrete surfaces.

1. Introduction

Crack detection is one of the most important links of concrete structure maintenance, and it directly reflects how safe, durable, and applicable the concrete structure is. Conventional human-based crack detection method relies on trained inspectors to find cracks on the surface of a concrete structure based on their expertise and years of experiences. They assess the concrete structure through analysing position and width of cracks. Although human-based crack detection method is an effective way to detect cracks, the detection results are subjective and vary from one to another because inspectors only make evaluation of current condition according to existing guidelines and their experiences.

To overcome the drawbacks of human-based crack detection method, many image processing techniques (IPTs) are developed to detect concrete cracks [1–3], concrete spalling [4], and potholes and cracks in asphalt pavement [5–7]. The IPTs can not only recognize cracks from images [8] but also measure the width and orientation of the recognized cracks [9, 10]. The simplest way to detect cracks from images is using the structural features, including histogram and threshold [11, 12]. To further improve its performance, general global transforms and edge detection detectors were applied, such as fast Haar transform (FHT), fast Fourier transform (FFT), Sobel, and Canny edge detectors [13, 14]. Although the IPTs are effective to detect some specific images, their robustness is poor because the crack images taken from a concrete structure may be affected by factors such as light, shadows, and rusty and rough surfaces in real-world situations.

To improve the performance of image-based crack inspection methods, researchers turn to machine learning (ML) algorithms [15]. The ML-based methods first extract crack features using the IPTs, then evaluate whether or not

the extracted features indicate cracks [16]. The artificial neural networks (ANNs) and Support Vector Machine (SVM) are typical ML algorithms, and they were adopted to detect concrete cracks, spalling, and other structural damages. However, the performance of this method relies on the extracted crack features, so the results of them have inevitably been affected by false feature extraction using IPTs.

To discard the extracting process of crack features, convolutional neural networks (CNNs) are imported to detect crack in images [17, 18]. CNNs are deep learning algorithms developed from the ANNs, and they are highlighted in image classification and object recognition [19]. Compared to the ANNs, the CNNs learn image features using fewer parameters computations due to the partial connections, sharing weights, and pooling process between neurons. The CNNs need to be trained using large number of manually classified images. The building of a database requires lots of human resources and computations, but the good news is that the existing well-annotated image databases (ImageNet [20], CIFIA-10 and CIFAR-100 [21], MNIST [22]) and parallel computations using graphic processing units (GPU) have solved the problems.

In this paper, a deep CNN is proposed to establish an image classifier for crack detection. The outstanding advantage of the proposed CNN-based crack detection is that it spares multifarious work from features preextraction and calculation compared to traditional methods [23]. Besides, the CNN needs not to convert the format of input images, but automatically learns crack features from images, which reduces workload of crack detection [24]. Moreover, our CNN-based crack detection approach achieves higher accuracy than existing method [25], because our CNN was trained using a large crack database with 60000 images taken from real concrete surfaces. Thanks to the large database, the detection results of our method will not be affected by noises of concrete surfaces such as roughness, light, shadows, or stain and so on.

The content of this research is described as follows. Section 2 introduces the methodology of the proposed method. Section 3 explains the CNN used and its related theories. Section 4 lists the training details of the CNN and results. Section 5 demonstrates testing results of the trained CNN on concrete crack images in realistic situations, and Section 6 is the conclusion of this paper.

2. Methodology

Figure 1 shows a flow chart of using a CNN to detect cracks. It includes three steps: building crack database, training the CNN, and testing the trained CNN classifier. To train a CNN, a large amount of raw images are taken from concrete surface. The collected raw images are cropped into smaller images, and then, cropped small images are manually classified into images with and without cracks. After that, training set and validation set are selected randomly from the database and imported into a CNN for training and validation. The training process generates a CNN classifier that is capable of classifying images into images with and without cracks. Using the trained CNN classifier and an exhaustive search with a sliding window, cracks can be separated from images accordingly.

3. CNN Architecture and Related Theories

This section explains the CNN architecture and related theories. A layer is the basic calculation unit of a CNN, so CNN architecture is formed accordingly once each layer in a CNN is confirmed. In the layers of a CNN, operations of data including convolution, pooling, full connection, and rectified linear unit (ReLU) and softmax can be conducted. Besides, some auxiliary operations, such as normalization and dropout, also can be embedded in layers for the specific purposes.

3.1. CNN Architecture. This paper builds a CNN through modifying the AlexNet [26]. The AlexNet is a remarkable CNN for image classification. It is trained in the ImageNet database, and the output number of its image classes is 1000, while the number of image classes in this paper is 2 (images with and without cracks). Therefore, the output number of classes is changed to 2, and other parameters remain unchanged. The modified CNN architecture is shown in Figure 2 where each dimension in input image indicates height, width, and channel (e.g., red, green, and blue), respectively. Notably, the input size of 227×227 can be enlarged and shrunken through recalculating the specifications according the changed input size. However, if too much image size is chosen, the detection result will inevitably include more background section, which disobeys the aim of crack detection. The softmax layer predicts whether each input data is an image with or without cracks according to output data. Table 1 presents the detailed specifications of the modified CNN. Notably, in the CNN, ReLU is used as activation function after each convolution layer and fc1, fc2, respectively. Besides, operations of local response normalization (LRN) and dropout are also implemented, where local response normalization is followed by pool1 and pool2, and dropout is located after fc1 and fc2.

3.2. Related Theories of the CNN Architecture. As listed in Table 1, the CNN consists of five convolution layers, three pooling layers, three full connection layers, and one softmax layer. Besides, other operations, such as ReLU, LRN, and dropout that cannot be visualized, also are included. All of the mentioned components are explained in detail as follows:

(i) *Convolution Layer*. Convolution layer makes CNNs stand out from the ordinary neutral networks. A convolution layer normally includes multiple feature maps. Each feature map consists of neurons arrayed in a rectangle, and neurons from the same feature map share weights called convolution kernels. Convolution kernels are normally initialized in the form of a random decimal array and learn reasonable weights during training [27]. The convolution kernels decrease connection between

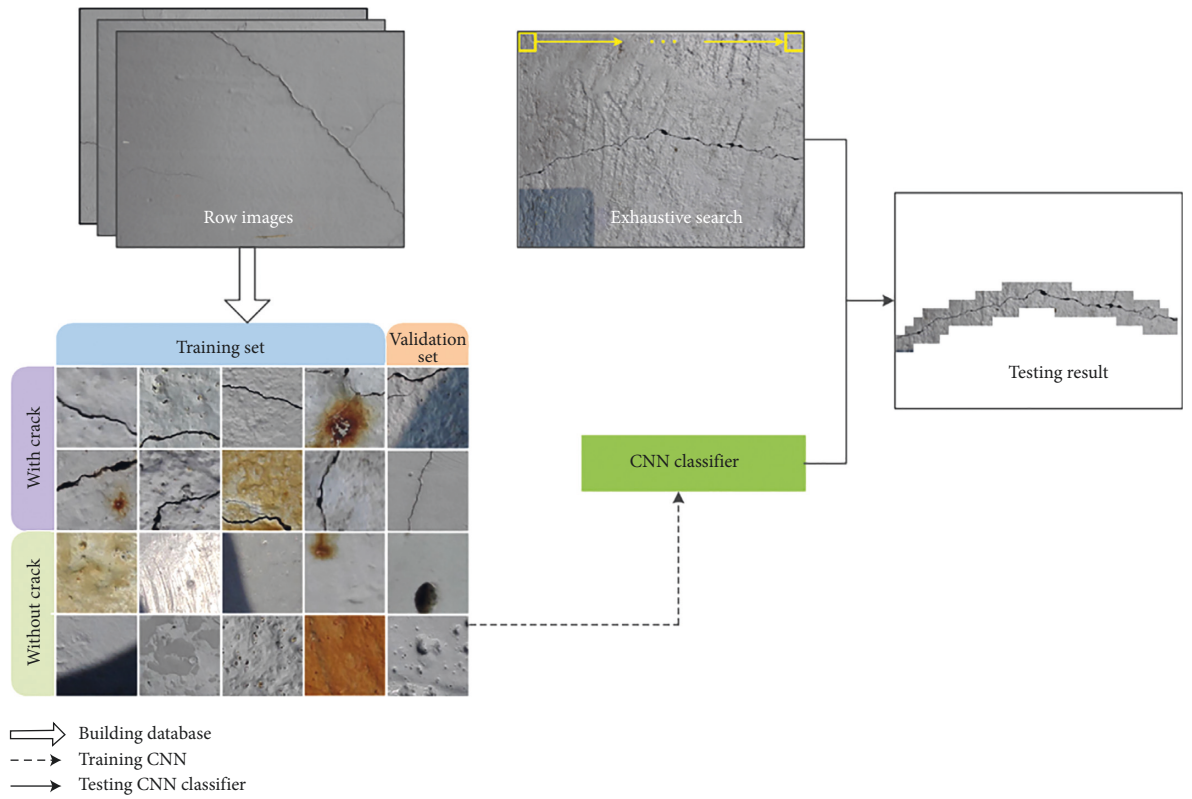


Figure 1: Flow chart for detecting cracks using a CNN.

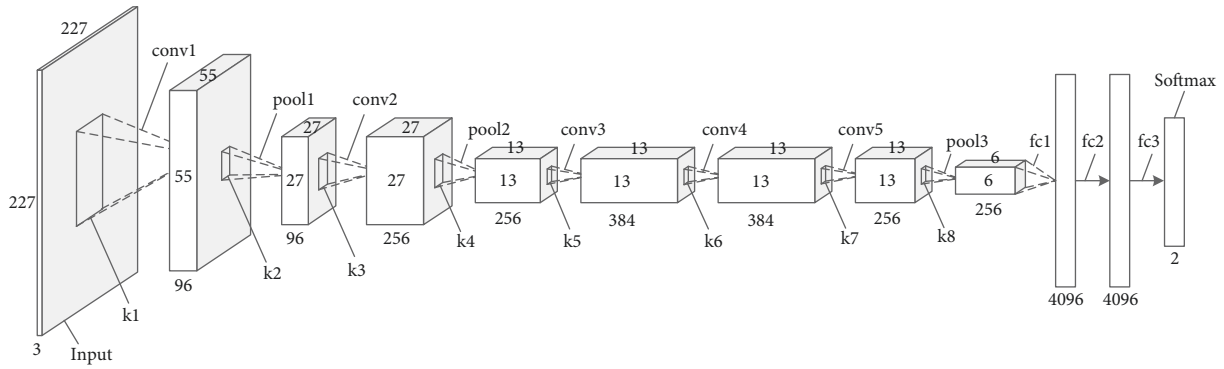


Figure 2: Illustration of a CNN's overall architecture. conv# = convolution; pool# = pooling; fc# = full connection; k# = kernel of each operation.

Table 1: Detailed specifications of a CNN.

Layer	Kernel size	Stride	Pad	Num_output
conv1	11	4	0	96
pool1	3	2	0	96
conv2	5	1	2	256
pool2	3	2	0	256
conv3	3	1	1	384
conv4	3	1	1	384
conv5	3	1	1	256
pool3	3	2	0	256
Fc1	—	—	—	4096
Fc2	—	—	—	4096
Fc3	—	—	—	2
Softmax	—	—	—	2

layers of the network. When conducting convolution operations, convolution kernels convolve with the upper neurons at certain strides; meanwhile, bias is added and outputted. Figure 3 shows a convolution process with bias of 0.

(ii) *Pooling Layer*. Pooling layer is another symbol that sets CNNs apart from the ordinary neutral networks. After the convolution process, the output neurons all include some information of the inputted neurons, which results in information redundancy and increases calculations. To improve algorithm performance and reduce calculations, a pooling operation is added to conduct the convolution results. What's more, pooling process also

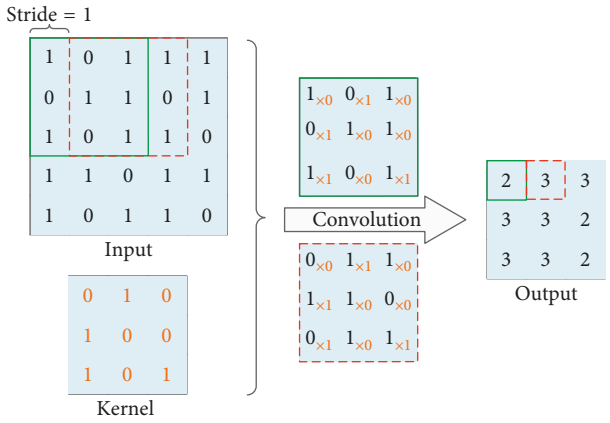


Figure 3: Convolution process.

reduces the spatial size of input array, reserves useful information, and avoids overfitting to some extent. There are two pooling options: max pooling and mean pooling. Studies showed that max pooling outperforms mean pooling when dealing with CNN architecture [28]. Therefore, this paper adopts the max pooling method as shown in Figure 4.

(iii) *ReLU*. The most frequently used nonlinear activation functions in artificial neural networks are sigmoid function ($f(x) = (1 + e^{-x})^{-1}$) and tanh function ($f(x) = \tanh(x)$). However, the calculations of these two saturating nonlinear activation functions are slow. Recently, some researchers introduced another activation function which is ReLU ($f(x) = \max(0, x)$) [29]. Figure 5 shows the comparison of the sigmoid, tanh, and ReLU functions. The ReLU function is a nonsaturated function, and only comparison is implemented in gradient descent training because the gradients of the ReLU are zero or one. If some training samples generate positive input to ReLU, learning will happen in that neuron. As a result, using the ReLU function can achieve faster calculations and convergence speed.

(iv) *LRN*. The characteristics of ReLU spare the need of normalization to avoid saturation. However, the output values generated by ReLU activation function do not have a certain range, which is different form the sigmoid and tanh function, so a LRN is used to generalize the output of ReLU.

(v) *Dropout*. Overfitting is a common problem in neural networks, and it often occurs in a network with a large amount of neurons. To prevent overfitting, dropout randomly deletes some neurons with a given dropout rate when weights are updated [30]. Through employing the dropout, a neural network can present a different architecture for every input, which reduces the coadaptations of neurons.

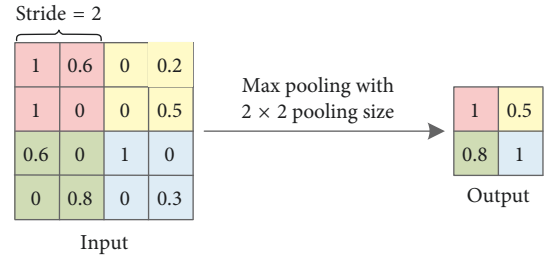


Figure 4: Max pooling process.

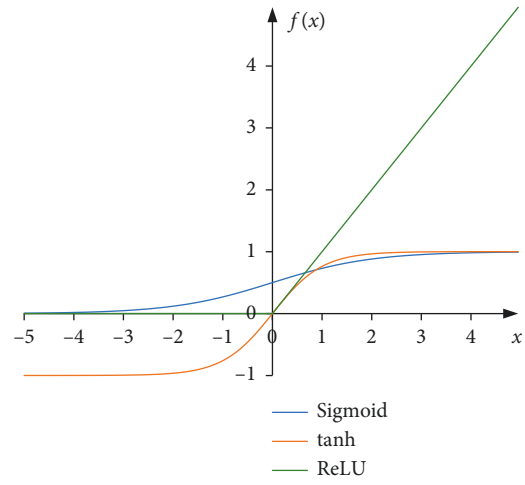
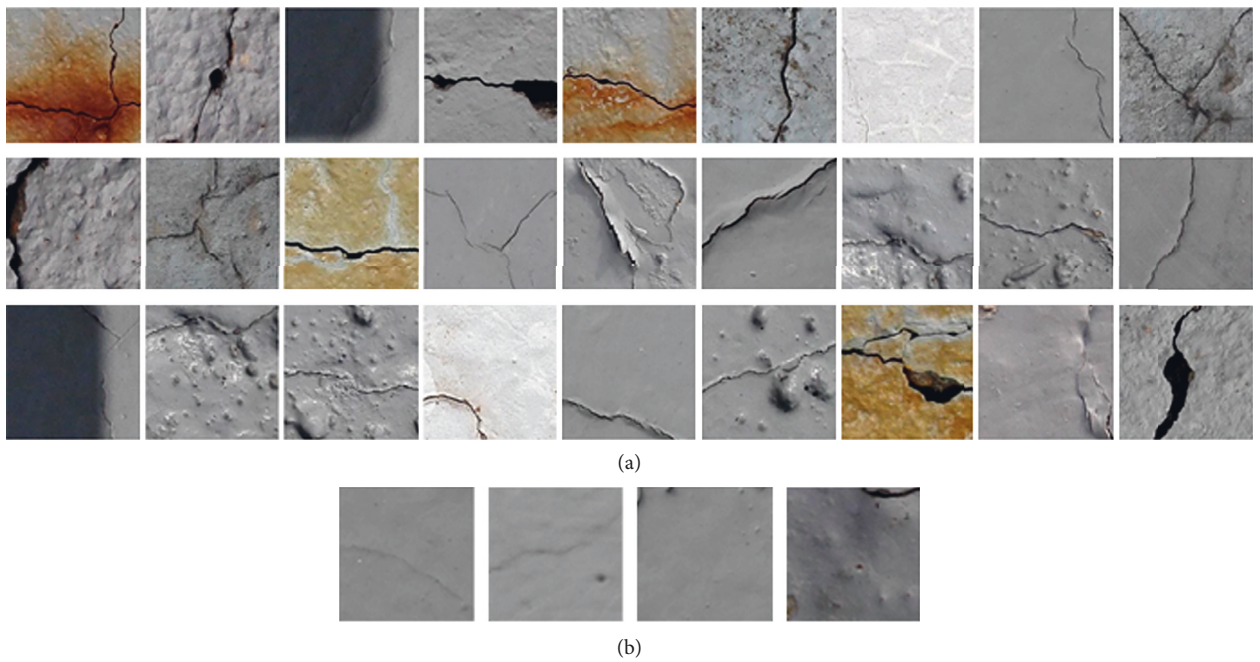


Figure 5: Nonlinear activation functions.

(vi) *Full Connection Layer*. The function of full connection layer is logical inference, which is same with traditional neural networks. Specially, the first full connection layer in our CNN needs to be explained individually. It connects the output of the last convolution layer, where a three-dimensional matrix is turned into a one-dimensional vector by operation of full convolution. That is to say, if input dimension of first full connection layer is $W \times H \times C$, then a convolution kernel with $W \times H \times C$ will be used. Thus, the output of first full connection layer will become a number. As a result, such K convolution kernels with a size of $W \times H \times C$ will correspond with K numbers, where K is the number of neurons of the first full connection layer.

(vii) *SoftmaxLlayer*. To classify input data, a softmax layer is essential. The softmax layer estimates a possibility $p(y^{(i)} = j^{(i)} | x^{(i)}; W)$ for every class $j^{(i)}$ of k classes using a softmax function presented in equation (1), where W is weight, i is the i-th example out of N input examples, and $W_k^T x^{(i)}$ are inputs of softmax layer. Right-hand side of the equation is a k-dimensional vector to represent the k estimated possibility, and $1/\sum_{j=1}^{k} e^{W_j^T x^{(i)}}$ is used to normalize the distribution of possibilities.



(a)

(b)

Figure 6: Cropped images: (a) images with cracks used for building database; (b) removed images.

$$p\left(y^{(i)} = j^{(i)} \,\middle|\, x^{(i)}; W\right) = \begin{bmatrix} p\left(y^{(i)} = 1 \,\middle|\, x^{(i)}; W\right) \\ p\left(y^{(i)} = 2 \,\middle|\, x^{(i)}; W\right) \\ \vdots \\ p\left(y^{(i)} = k \,\middle|\, x^{(i)}; W\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{W_j^T x^{(i)}}} \begin{bmatrix} e^{W_1^T x^{(i)}} \\ e^{W_2^T x^{(i)}} \\ \vdots \\ e^{W_k^T x^{(i)}} \end{bmatrix}, \quad i = 1, \ldots, N. \tag{1}$$

4. Training and Validating the CNN

To train a CNN classifier used for crack detection, a database that consists of a large amount of concrete images with cracks should be built in advance. The hyperparameters of the CNN need to be finally confirmed during training the CNN through trial and error. All of the study of this paper is conducted on Caffe [31] in Windows system using a workstation that configured with a GPU. (CPU: Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.2 GHz, RAM: 32 GB, GPU: ASUS GeForce GTX 1080 Ti).

4.1. Building Database. To build a database, 1455 crack images with 4160 × 3120 pixel resolutions were taken using a smartphone. These images were captured from surfaces on bridge towers and anchor chambers of a suspension bridge in Dalian, Liaoning, China. Distances from smartphone to the cracks ranged from approximately 0.5–1.0 m. However, some images are taken below a 0.1 m distance for testing, and the lighting conditions of the images are substantially different including, day, night, sun-facing, and shaded surfaces. Among these images, 1250 images were used for CNN training and validation, and the remaining 205 images were used to test the trained CNN. To build training set and validation set, the 1250 images were cropped into smaller images with 256 × 256 pixel resolutions. Then, the cropped images were classified manually into two classes: with cracks and without cracks. Significantly, because the cropped images with cracks only take a small proportion of the total images, some cropped images without cracks were randomly removed to avoid classification imbalance. The number of prepared images in database is 60000, and the quantity proportion of images with and without cracks is 1 : 1. To obtain a CNN classifier with excellent robustness, these cropped images include cracks of different situations and background features, as shown in Figure 6(a). The purpose of cropping images into images with a resolution of 256 × 256 is to fit them with recommended size of CNN input. As shown in Figure 6(b), images with unobvious cracks and corner cracks were removed when building the database. The reason for doing this is that it is very difficult to detect cracks in those images by human eye during manual classification. Besides, wrong classified images will lead to wrong CNN classification. Therefore, it is meaningless to train CNNs using wrong classified images.

4.2. Softmax Loss and Stochastic Gradient Descent. During the process of CNN training, weights and bias need to be initialized. In this paper, weights and bias are, respectively,

initialized using Gaussian and Constant initialization, where the Gaussian initialization adopts Gaussian distribution and Constant initialization set bias as a constant (0 by default). When training the CNN, the predicted and actual classes of the trained CNN do not usually coincide. Hence, a softmax loss function defined by equation (2) is used to calculate the deviations between the predicted and actual classes.

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} f_w\left(x^{(i)} + \lambda r(W)\right), \tag{2}$$

where W is weights, $f_w(x^{(i)})$ is the loss of $x^{(i)}$, and $r(W)$ is regularization with a weight of λ.

To narrow the deviations during the CNN training, the weights have to be constantly modified to predict true classes. Stochastic gradient descent (SGD) is utilized to update weights. To accelerate the convergence speed of training, the momentum algorithm is used in SGD. As shown in equation (3), SGD first updates ($\longleftarrow$) V_{t+1} according to the linear combination of negative gradient $\nabla L(W_t)$ and the previous speed V_t, where learning rate α and momentum μ are the weights of negative gradient and speed, respectively. Then, W_{t+1} is updated in equation (4) using the previous weight and the updated V_{t+1}:

$$V_{t+1} \longleftarrow \mu V_t - \alpha \nabla L(W_t), \tag{3}$$

$$W_{t+1} \longleftarrow W_t + V_{t+1}. \tag{4}$$

The CNN repeats the abovementioned process many times until equation (4) converges. During CNN training, the number of used images in each update is called the batch size. Each complete update out of a batch size is called an iteration, and each complete update out of the entire database is called an epoch.

4.3. Details of Training and Validating the CNN. The CNN repeats the abovementioned process many times until equation (4) converges. During CNN training, the number of used images in each update is called the batch size. Each complete update out of a batch size is called an iteration, and each complete update out of the entire database is called an epoch.

Unlike other image-based crack detection methods, the CNN does not require feature extraction but learns features automatically by updating the weights. The build training set and validation set include 60000 images, and the ratio between training set and validation set is 4 : 1, with that of cracks and without cracks being 1 : 1. Each batch needs 256 images for training and 200 images for validation process. Before being input CNN, the mean of those images is subtracted for efficient computation. Besides, those images with 256×256 pixel resolutions were automatically cropped into images of 227×227 pixel resolutions for data augmentation.

This paper adopts a learning strategy of "step" to adjust learning rate which first set a base learning rate, then multiply the learning rate with a learning rate changing factor (set as 0.1 in this paper) after a certain number of iterations (set as 5000 iterations this paper). Besides, other hyperparameters, momentum, weight decay, and dropout rate, are set as 0.9, 0.0005, and 0.5, respectively.

The accuracy is adopted as the metric of validation performance for the CNN in our study. When training the CNN, the trained models are validated with batch size of 200. In an iteration of validation, the validation accuracy is the ratio between the number of correctly classed images and the total number of validation images at the iteration.

The learning rate affects the validation accuracy and convergence speed during training a CNN [32]. To choose a best base learning rate, the base learning rates used in this paper are set to 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, and 0.0001, respectively. The CNN was trained 15000 iterations under different base learning rates and validated every 50 iterations. Recorded validation accuracies are shown in Figure 7.

According to Figure 7, the validation accuracies and convergence speeds gradually increase with the base learning rate change in the range of 0.0001 to 0.01, and peak at 0.01. When the base learning rate increases to 0.05, the convergence speed becomes low, but the accuracy remains high. Larger base learning rate of 0.1, however, will lead the accuracy to increase first and then remains 50%, which indicates that the training of the CNN is nonconvergent. The key finding taken from the training results is that choosing a big base learning rate in a certain range can make a CNN converge faster and obtain higher validation accuracy during training.

The training result under a base leaning rate of 0.01 was finally selected as the image classifier due to its highest validation accuracy that is 99.07% achieved at 13450th iteration. With the acceleration of a GPU, the training process took 12 hours to train the CNN 15000 iterations on 60000 images, while it will normally take around one month to complete the same calculation in this workstation only by using CPU. Figure 8 shows the visualized convolution kernels of features in the first convolution layer (conv1) under the base learning rate of 0.01, in which the features of each convolution kernel were obtained automatically from training images. The learned crack-like features of convolution kernels are smooth, and the crack features make CNNs detect cracks from complicated images of concrete surface.

5. Testing the Trained and Validated CNN

After the training process, the trained CNN classifier is tested on new images to verify the adaptation and robustness. Due to the random distributions of the cracks, a strategy of exhaustive search is used to locate the cracks' positions because it is difficult to find cracks depending only lump-sum scan in a large image [18]. Exhaustive search method is often used in programming when there is no rule to solve a problem. Numerous candidate solutions are enumerated and tested one by one in a certain order, and those candidate solutions that meet the requirements are found out as the solution of the problem. As shown in

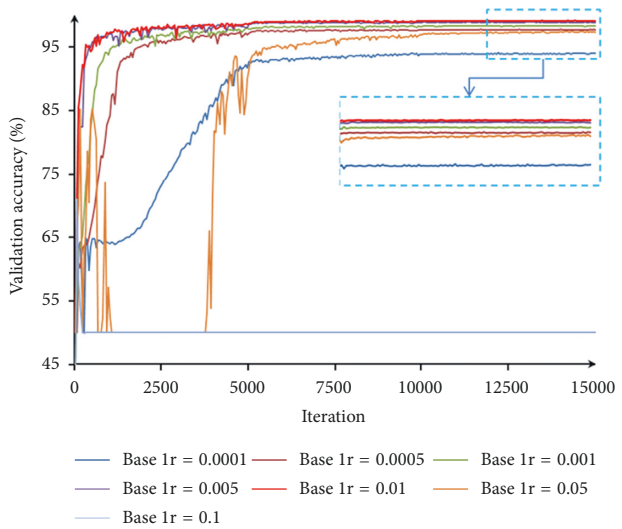


FIGURE 7: Validation accuracies under different base learning rates. base lr = base learning rate.

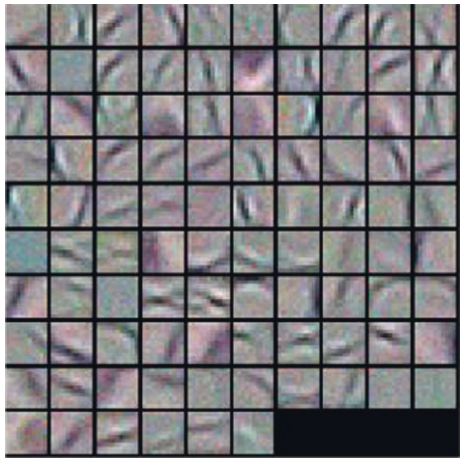


FIGURE 8: Visualized convolution kernels of conv1.

Figure 9(a), a large image with 4160×3120 pixel resolutions is scanned using a sliding window of 256×256 pixel resolutions. When the window slides to a position, the trained CNN image classifier in Section 4.3 will classify the little image in that position, in which a little image with cracks will be retained, otherwise removed. To avoid wrong classifications mentioned in Section 4.1, the whole large image will be scanned twice. Figure 9(b) shows the scanning results of crack detection using the exhaustive search method. It can be found that if only the 1st scanning using exhaustive search was implemented to detect crack, the crack parts located in corners of some scanning windows are disregarded. When the detection result of the 2nd scanning was overlapped to the result of the 1st scanning, the disregarded crack parts also ware successfully detected.

205 raw images with 4160×3120 pixel resolutions that were not used to build training set and validation set were scanned according to the abovementioned exhaustive search. Like the training process of the CNN, the image with 256×256 pixel resolutions where the sliding window was in here was subtracted the mean of training set before being classified by the trained CNN. In testing results, the regions in raw images with actual cracks are defined as positive regions, otherwise negative regions. Meanwhile, the positive and negative regions that can be detected and classified correctly by the trained CNN are defined as true-positive and true-negative regions, otherwise false-positive and false-negative regions, respectively. The calculation of testing accuracy for each image was conducted in equation (5):

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \quad (5)$$

where TP, FP, TN, and FN represent the number of true-positive, false-positive, true-negative, and false-negative regions in the tested images, respectively.

Figure 10 presents testing accuracies of 205 images, and the average accuracy of them is 99.09% which is very close to the highest validation accuracy (99.06%). Encouragingly, the performance of the trained CNN is still impressive even though totally different images are used for testing, and the recorded testing time required for each image is about 60 seconds.

Figures 11–13 show some testing results in different situations, where the false-positive and false-negative regions are highlighted. Figure 11 shows images of normal concrete surfaces with clear microcracks. Their testing results are remarkable where all the cracks are detected successfully. Especially, in Figure 11(b), an image with microcracks is used to test the trained CNN, where the distance from smartphone to the microcracks is about 10 cm. In the detection result, the width of the microcracks is 2 pixels (about 90 μm). The result shows that the proposed method has good performance of detecting microcracks. To examine the robustness of the trained CNN, images taken from damaged surfaces, shadowed, rough, and rusty surface, and even a surface with holes are used for testing. Notably, when taking the image in Figure 11(b), the distance between the thin cracks and smartphone is about 0.1 m and width of the detected thin crack is about 3 pixels, which corresponds to about 0.09 mm. In Figure 12, all cracks in original images are still detected correctly, although there are two false-positive regions in the images from rough and rusty surfaces, respectively. To test the adaptability of the trained CNN further, an image with complex and blurry cracks is chosen in Figure 13, and the testing results provide acceptable testing accuracy of 98.32% with only three false-negative regions and three false-negative regions. It can be seen from the testing results shown in Figures 11–13 that the robustness and adaptability of proposed crack detection method are impressive in real-world situations.

6. Integrating the Trained CNN into a Smartphone Application

The trained CNN can be used to predict the class of a new image, and the popularity of smartphones provides an opportunity to mobile public to detect cracks. For this

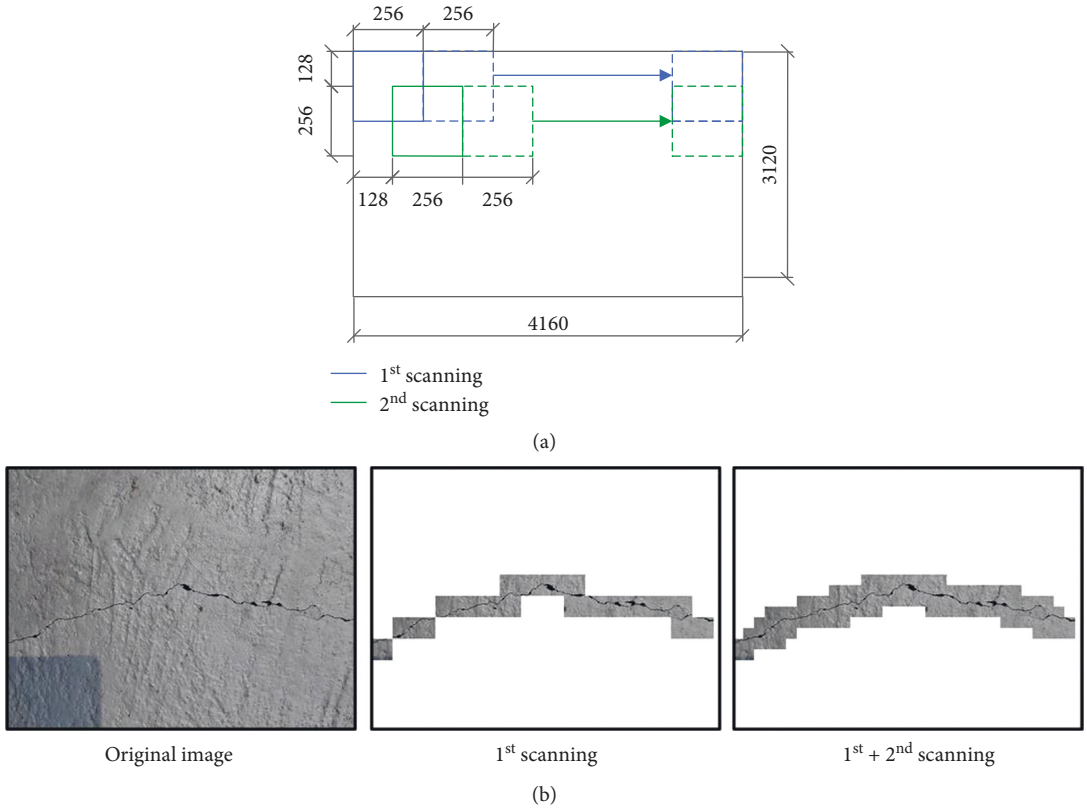


Figure 9: Crack detection using exhaustive search: (a) sketch; (b) detection result.

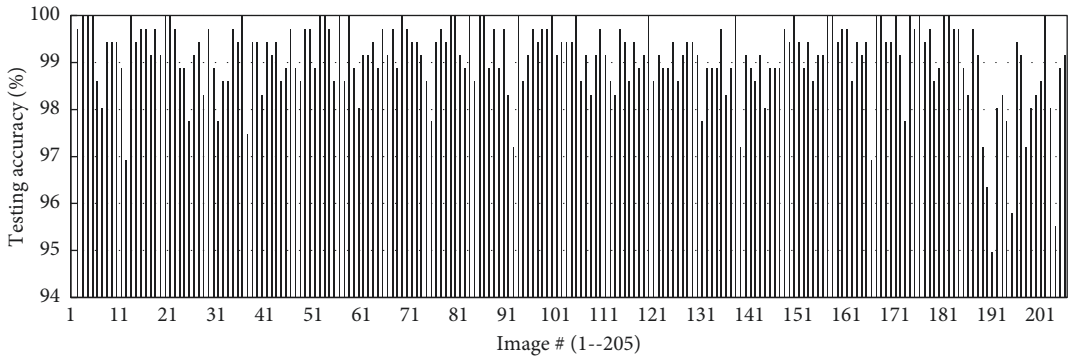
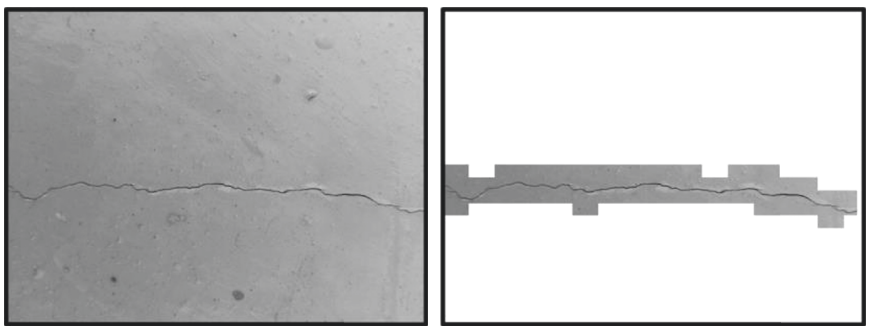


Figure 10: Testing accuracies of 205 images.

purpose, based on the framework of Core ML, the trained CNN model is integrated into a smartphone application to detect cracks in practice conveniently. During the integrating process, the Xcode (version 9.2), an integrated development environment is utilized to create an application with Swift programming language. The generated application named Crack Detector is installed on an iPhone 7 Plus with iOS 11.2. Notably, it can predict not only local photos but also a new photo taken from concrete surface at that time. Figure 14 presents a detection result of the Crack Detector. The result exposes the great performance of the trained CNN. In addition, this smartphone application can draw more attention to crack detection handily.

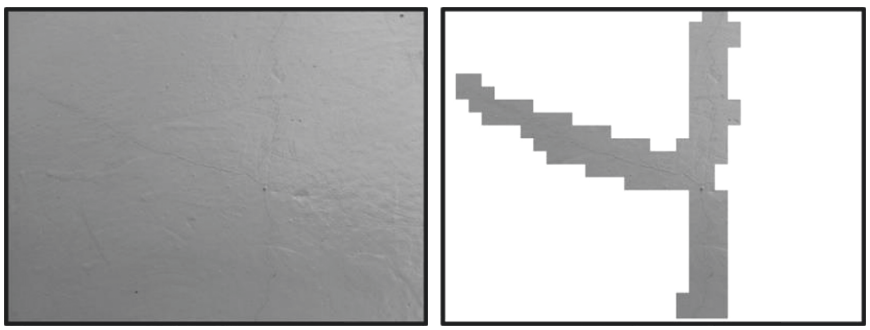
7. Conclusions

This paper proposed a method to detect cracks from images using convolutional neural networks. The CNN architecture of binary-class output for crack detection was designed through modifying the AlexNet. 1455 images with 4160×3120 pixel resolutions captured by a smartphone were used to train, validate, and test the CNN where all of the images were taken from real concrete surfaces. To build the training set and validation set, 1250 images were cropped into 60000 smaller images of 256×256 pixel resolutions. The datasets as open-source can be downloaded from website (https://drive.google.com/open?id=1XGoHqdG-WYhIaTsm-uctdV9J1CeLPhZR). To choose the best base learning rate



Original image Testing result

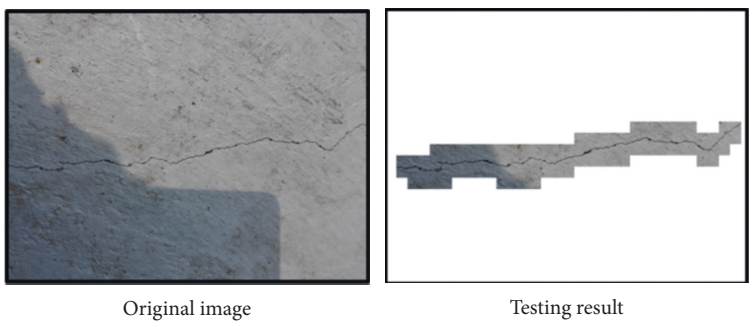
(a)



Original image Testing result

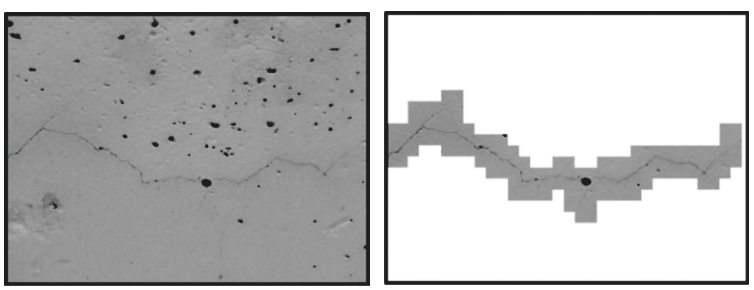
(b)

Figure 11: Normal surfaces: (a) clear crack; (b) microcracks.



Original image Testing result

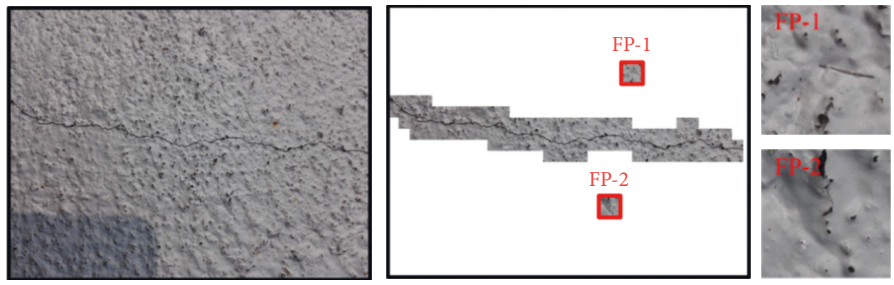
(a)



Original image Testing result

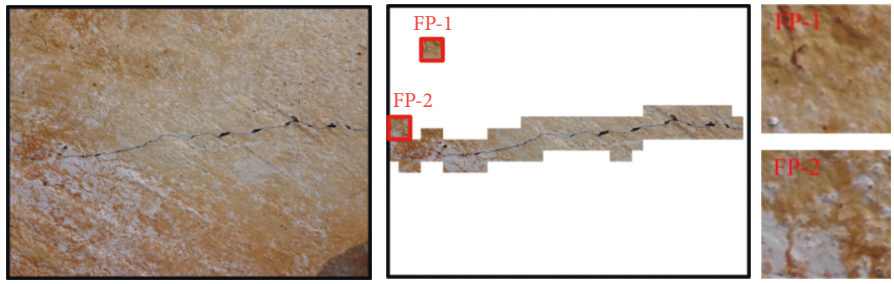
(b)

Figure 12: Continued.



Original image Testing result (—false positive)

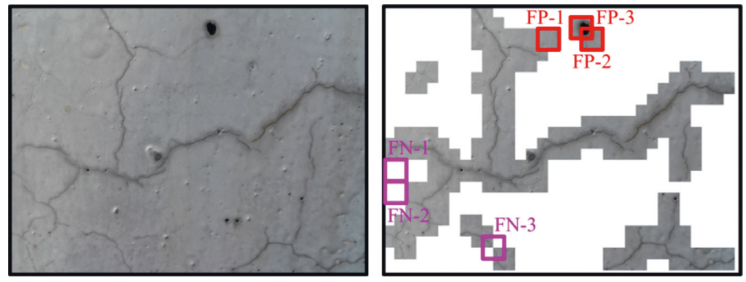
(c)



Original image Testing result (—false positive)

(d)

Figure 12: Damaged surfaces: (a) shadowed surface; (b) surface with holes; (c) rough surface; (d) rusty surface.



Original image Testing result (—false positive; —false negative)

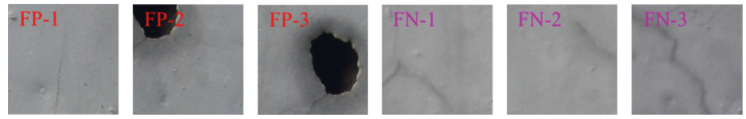


Figure 13: Complex and blurry cracks.

according to validation accuracy, different base learning rates were set during training the CNN. By comparing the training results under different base learning rates, the base learning rate of 0.01 was chosen the best where the CNN achieved the highest validation accuracy of 99.06%. Then the trained CNN under the learning rate of 0.01 was used for the following testing. Combined with an exhaustive search using a sliding window, the trained CNN was tested on remaining 205 images, and average testing accuracy reached to 99.09%. The training and testing codes and their usage can be viewed from the website https://github.com/Shengyuan-Li/CNN-for-Crack-Detection. Images used for the testing were a various range of concrete surfaces with situations such as thin cracks; cracks on shadowed, rough, rusty surface; surface with holes; and even complex and blurry cracks. To mobile more public to detect cracks in practice, the trained CNN model is integrated into a smartphone application named Crack Detector. It is concluded that the proposed method is capable of detecting the cracks on real concrete surfaces without being interfered by noises. The study in this paper proves that a CNN is especially powerful in image classification as it can automatically learn certain features from a large amount of images. The research approach of this paper can also be adopted in other types of damage detections such as scaling of concrete surface, corruption, and peeling paint of steel and concrete and more.

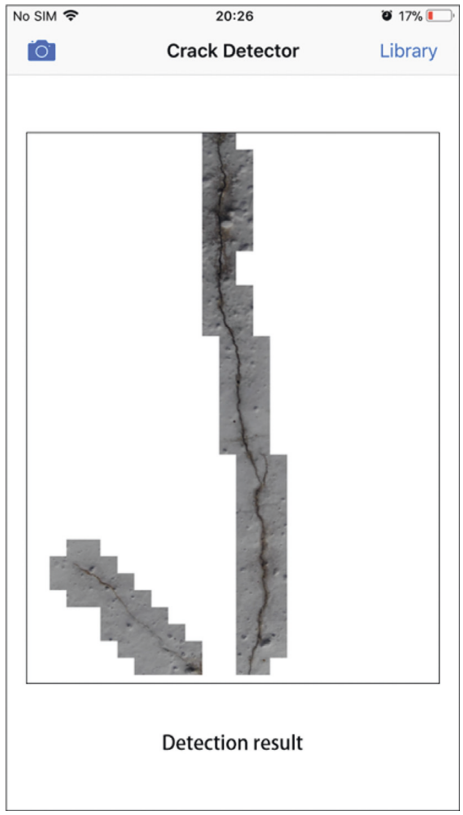


Figure 14: Detection result of the Crack Detector.

In future studies, more images with more types of concrete damages under various conditions will be provided and added to the existing database to increase the adaptation and robustness of the proposed method, and comparative studies will also be performed.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was financially supported by National Key R&D Programs during the Thirteenth Five-Year Plan Period (grants 2016YFC0802002-03 and 2016YFE0202400) and Natural Science Foundation of China (grant 51479031).

References

[1] Y. Fujita, Y. Mitani, and Y. Hamamoto, "A method for crack detection on a concrete structure," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, pp. 901–904, Hong Kong, August 2006.

[2] T. Yamaguchi, S. Nakamura, R. Saegusa, and S. Hashimoto, "Image-based crack detection for real concrete surfaces," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 3, no. 1, pp. 128–135, 2008.

[3] H. D. Cheng, J.-R. Chen, C. Glazier, and Y. G. Hu, "Novel approach to pavement cracking detection based on fuzzy set theory," *Journal of Computing in Civil Engineering*, vol. 13, no. 4, pp. 270–280, 1999.

[4] S. German, I. Brilakis, and R. Desroches, "Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments," *Advanced Engineering Informatics*, vol. 26, no. 4, pp. 846–858, 2012.

[5] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507–515, 2011.

[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, 2016.

[7] W. Wang, A. Zhang, K. Wang, A. Braham, and S. Qiu, "Pavement crack width measurement based on laplace's equation for continuity and unambiguity," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 2, pp. 110–123, 2018.

[8] Y. Fujita and Y. Hamamoto, "A robust automatic crack detection method from noisy concrete surfaces," *Machine Vision and Applications*, vol. 22, no. 2, pp. 245–254, 2011.

[9] T. Nishikawa, J. Yoshida, T. Sugiyama, and Y. Fujino, "Concrete crack detection by multiple sequential image filtering," *Computer-Aided Civil and Infrastructure Engineering*, vol. 27, no. 1, pp. 29–47, 2012.

[10] B. Y. Lee, Y. Y. Kim, S.-T. Yi, and J.-K. Kim, "Automated image processing technique for detecting and analysing concrete surface cracks," *Structure and Infrastructure Engineering*, vol. 9, no. 6, pp. 567–577, 2013.

[11] H. Oliveira and P. Correia, "Automatic road crack segmentation using entropy and image dynamic thresholding," in *Proceedings of the IEEE Signal Processing Conference*, pp. 622–626, Taipei, Taiwan, April 2009.

[12] K. R. Kirschke and S. A. Velinsky, "Histogram-based approach for automated pavement-crack sensing," *Journal of Transportation Engineering*, vol. 118, no. 5, pp. 700–710, 1992.

[13] B. Santhi, G. Krishnamurthy, S. Siddharth, and P. Ramakrishnan, "Automatic detection of cracks in pavements using edge detection operator," *Journal of Theoretical and Applied Information Technology*, vol. 36, no. 2, pp. 199–205, 2012.

[14] I. Abdel-Qader, O. Abudayyeh, and M. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 4, no. 255, pp. 255–263, 2003.

[15] S. W. Liu, J. H. Huang, J. C. Sung, and C. C. Lee, "Detection of cracks using neural networks and computational mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 25-26, pp. 2831–2845, 2002.

[16] M. S. Kaseko, Z. P. Lo, and S. G. Ritchie, "Comparison of Traditional and neural classifiers for pavement-crack detection," *Journal of Transportation Engineering*, vol. 120, no. 4, pp. 552–569, 1994.

[17] E. Protopapadakis, K. Makantasis, G. Kopsiaftis et al., "Crack identification via user feedback, convolutional neural networks and laser scanners for tunnel infrastructures," in *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016)*, pp. 725–734, Rome, Italy, February 2016.

[18] 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.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[20] J. Deng, W. Dong, R. Socher et al., "ImageNet: a large-scale hierarchical image database," in *Proceedings of the Computer Vision and Pattern Recognition 2009 (CVPR 2009)*, pp. 248–255, IEEE, Anchorage, AL, USA, June 2009.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, Toronto, Canada, 2009.

[22] Y. LeCun, L. Bottou, Y. Bengio et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] Z. Qu, F. R. Ju, Y. Guo et al., "Concrete surface crack detection with the improved pre-extraction and the second percolation processing methods," *PloS One*, vol. 13, no. 7, Article ID e0201109, 2018.

[24] B. Santos, J. Valença, and E. Júlio, "Classification of cracks on concrete surface using false colour HSV images, including near-infrared information," in *Proceedings of the Optical Sensing and Detection V*, p. 1068003, Strasbourg, France, July 2018.

[25] S. Yokoyama and T. Matsumoto, "Development of an automatic detector of cracks in concrete using machine learning," *Procedia Engineering*, vol. 171, pp. 1250–1255, 2017.

[26] A. Krizhevsk, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems, 25 (NIPS 2012)*, pp. 1097–1105, Harrahs and Harveys, Lake Tahoe, NV, USA, December 2012.

[27] P. Krähenbühl, C. Doersch, J. Donahue et al., "Data-dependent initializations of convolutional neural networks," in *Proceedings of the International Conference on Learning Representations 2016 (ICLR 2016)*, San Juan, Puerto Rico, May 2016.

[28] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proceedings of the 20th International Conference on Artificial Neural Networks (ICANN)*, pp. 92–101, Thessaloniki, Greece, September 2010.

[29] V. Nair and G. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pp. 807–814, Haifa, Israel, June 2010.

[30] N. Srivastava, G. Hinton, A. Krizhevsky et al., "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] Y. Jia, E. Shelhamer, J. Donahue et al., "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678, Orlando, FL, USA, November 2014.

[32] D. Wilson and T. Martinez, "The need for small learning rates on large problems," in *Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN'01)*, pp. 115–119, IEEE, Washington, DC, USA, July 2001.