Hindawi
Advances in Civil Engineering
Volume 2019, Article ID 8582963, 12 pages
https://doi.org/10.1155/2019/8582963

Research Article

Deep-Learning-Based Bughole Detection for Concrete Surface Image

Gang Yao,[1,2] Fujia Wei,[1,2] Yang Yang,[1,2] and Yujia Sun[1,2]

[1]*Key Laboratory of New Technology for Construction of Cities in Mountain Area, Ministry of Education, Chongqing 400044, China*
[2]*School of Civil Engineering, Chongqing University, Chongqing 400044, China*

Correspondence should be addressed to Fujia Wei; weifujia13@163.com and Yang Yang; yy20052710@163.com

Received 4 March 2019; Accepted 22 May 2019; Published 16 June 2019

Academic Editor: Robert Černý

Bugholes are surface imperfections that appear as small pits and craters on concrete surface after the casting process. The traditional measurement methods are carried out by in situ manual inspection, and the detection process is time-consuming and difficult. This paper proposed a deep-learning-based method to detect bugholes on concrete surface images. A deep convolutional neural network for detecting bugholes on concrete surfaces was developed, by adding the inception modules into the traditional convolution network structure to solve the problem of the relatively small size of input image (28×28 pixels) and the limited number of labeled examples in training set (less than 10 K). The effects of noise such as illumination, shadows, and combinations of several different surface imperfections in real-world environments were considered. From the results of image test, the proposed DCNN had an excellent bughole detection performance and the recognition accuracy reached 96.43%. By the comparative study with the Laplacian of Gaussian (LoG) algorithm and the Otsu method, the proposed DCNN had good robustness which can avoid the interference of cracks, color-differences, and nonuniform illumination on the concrete surface.

1. Introduction

Bugholes are surface imperfections that appear as small pits and craters on concrete surface after the casting process [1]. These imperfections appear as regular or irregular pits with diameters ranging from few millimeters to 15 mm in diameter and are usually scattered randomly around the surface of the concrete. Bugholes are recognized as a major problem in the construction industry very early on [2, 3]. On the one hand, building owners and architects are getting stricter on quality of concrete surfaces. Surfaces are demanded to be flat and free of surface bugholes to leave an aesthetically pleasing impression. On the other hand, even though bugholes are primarily an aesthetic issue for exposed concrete structures and do not affect the structural strength of concrete, it does reduce the adhesion properties of the fiber-reinforced plastic (FRP) material applied to the concrete surface [4]. Additionally, research has indicated that salt accumulated in bugholes causes premature degradation of reinforced concrete (RC) structures [5]. Moreover, these surface bugholes are generally considered a nuisance for subsequent processes since these imperfections need to be filled before paint coating is applied to the concrete surface, and this additional surface preparation is a labor-intensive and costly process.

Owing to its influence on the quality of concrete surfaces, the methods for the detection of bugholes were established very early on. In the 1970s, construction professionals attempted to assess concrete surface quality by manually counting the number and measuring the diameter of bugholes and then calculating percentage of holed areas on the surface, which was considered time-consuming and impractical [2, 3]. Therefore, an improved method classifying concrete surface quality was proposed by Thomson [6], who suggested using bughole photos with different degrees of coverage as reference samples to compare with actual concrete surface. The current method [7, 8] of bughole rating developed by the American

Concrete Institute (ACI) based on the idea introduced by Thomson and the ACI method compares the concrete surface to be assessed with a set of standard surface photographs representing seven scales, and the expert who is performing the comparison determines which scale the concrete surface being inspected belongs to. While simple in principle, some people argue that the comparison with photographs of reference samples can be problematic due to the variability between different printed scales of the reference samples and the subjectivity of the human eye [9, 10]. In addition, one surface may have several types of bugholes combined or a combination of several different surface imperfections such that the use of the reference becomes rather difficult and subjective. Moreover, there is a large amount of concrete engineering in practical states, and the manual inspection method is not only time-consuming and labor-intensive but also costly.

In order to obtain a better evaluation of the concrete surface, more objective and intelligent methods need to be developed. Digital image processing technology is considered a powerful automated tool that can provide objective results quickly [11, 12], and it has been successfully applied in bridge coating quality assessment in recent years. Lee et al. developed an automated processor that can recognize the presence of bridge coating rust defects [13]. In order to solve the problem of nonuniform illumination, Chen et al. proposed the adaptive ellipse approach (AEA), the box-and-ellipse-based adaptive-network-based fuzzy inference system (BE-ANFIS), and the support-vector-machine-based rust assessment approach (SVMRA) [14–16]. In order to adapt to various background colors and overcome the effects of background noise or nonuniform illumination, Shen et al. proposed a rust defect recognition method based on color and texture feature, which combines the Fourier transform and color image processing [17]. Son et al. employed the J48 decision tree algorithm to rapidly and accurately determine rusted surface area [18]. In order to improve the detection accuracy of the rusted areas on steel bridges, Liao et al. proposed a digital image recognition algorithm that consisted of the K-means method and the double-center-double-radius (DCDR) algorithm [19]. Shen et al. proposed an artificial-neural-network-based rust intensity recognition approach (ANNRI) [20]. In addition to detecting rust, researchers also applied a variety of image processing techniques on visual images to detect cracks, including edge detection methods [21–24], morphological operations [25–27], digital image correlation [28, 29], and image binarization [30, 31]. A comparative study of fast Haar transform (FHT), fast Fourier transform, Sobel edge detector, and Canny edge detector showed that FHT has the best performance [22]. Lim et al. used the Laplacian of Gaussian (LoG) edge detector to detect surface cracks in concrete bridge decks and obtain global crack maps through camera calibration and robotic localization [23]. Talab et al. used multiple filters such as Sobel and Area filters to change the small area to the background and used the Otsu method to detect major cracks [24]. Some researchers have verified that the morphological operations are effective for crack detection [25, 26, 32]. Rimkus et al. used digital image correlation (DIC) technique to detect and locate cracks in concrete surfaces [28]. Kim et al. and Li et al. suggested using image binarization methods to extract crack information from digital images [30, 31].

The current research focused on the use of image processing technology to detect surface cracks and rust with relatively few studies on surface bugholes. Zhu and Brilakis proposed an image processing method to detect bugholes (referred to as air pockets in their paper) on the concrete surfaces [33], but this method did not consider the influence of nonuniform illumination. Ozkul and Ismail developed a bughole measuring device for rating the quality of a concrete surface [1]. Silva et al. developed an expert system that uses image analysis methods to classify the surface quality of self-consolidating concrete for precast members by calculating the percentage area, diameter, and distribution of the bugholes [34]. Hirano et al. used a thresholding method to detect bugholes in the images, but using this method, it is difficult to detect small bugholes [35]. Liu et al. established a method to detect surface bugholes via image analysis and published evaluation parameters; the OTSU image threshold segmentation technology is adopted to extract the characteristics of bugholes on the concrete surface [36]. Isamu et al. developed an image analysis method using color images to quantify the bugholes distributed on concrete surfaces [37].

The application of digital image processing technology (IPT) has promoted the advancement of quality inspection methods of structural surface. However, the direct use of image processing technology for surface defect inspection has several disadvantages. First, the algorithms are tailored for certain images in the studied datasets, which affect their performance on new datasets [38]. Second, due to the effects of noise such as illumination, shadows, and combination of several different surface imperfections, the detection results of the image processing algorithms may be inaccurate [39–41]. Finally, the image processing algorithms are often designed to aid the inspector in defect detection and still rely on human judgement for final results [25]. One possible solution is using deep-learning algorithms to analyze the inspection images [42]. In recent years, deep-learning algorithm has shown remarkable performance in image object recognition [43–46] and deep convolutional neural networks (DCNN) have attracted wide attention as an effective recognition method [47]. Several researchers have applied this method to detect concrete surface cracks. Zhang et al. conducted a comparative study on the image classification of pavement cracks using three methods: deep convolution network, support vector machine, and integrated learning. Results showed that the detection effect of the deep convolution network was better than the other two methods [48]. Cha et al. used a convolution neural network based on deep learning to detect cracks in concrete images [49]. Wang et al. proposed a convolutional neural network (CNN) for recognizing cracks on asphalt surfaces at subdivided image cells; the accuracy of the proposed CNN can achieve high accuracies 96.32% and 94.29% on training data and testing data, respectively [50].

In this paper, a deep convolutional neural network (DCNN) has been used to detect bugholes on concrete surfaces, and the effects of noise such as illumination, shadows, and combinations of several different surface imperfections in real-world environments were considered. The performance of the proposed method was compared with that of the traditional image processing methods.

2. Main Concrete Surface Defect Classification

There are many different types of quality defects of concrete surfaces. The most common surface defects are cracks, bugholes, and color-differences. Figures 1–3 show the image characteristics of these three types of defects.

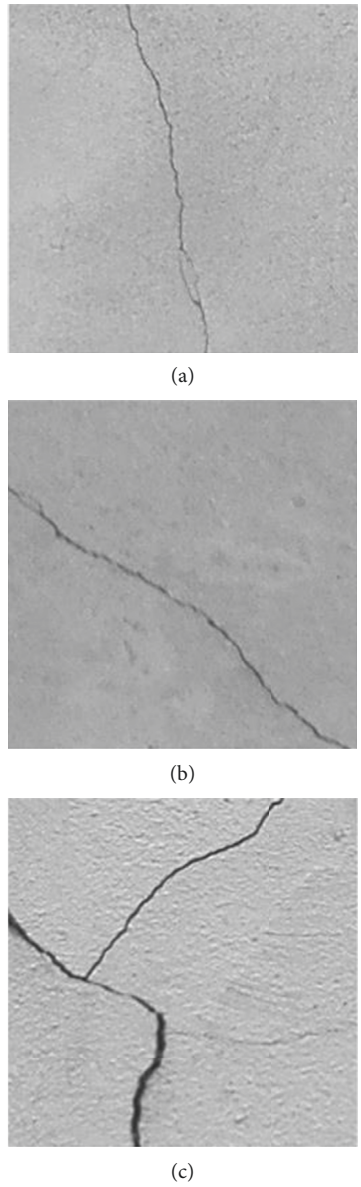
Cracks, bugholes, and color-differences have its own features. Cracks are generally irregularly line-shaped, as shown in Figure 1. Owing to the different reflectivities of light, a bughole is normally darker than the rest of the concrete surface. The greater the depth, the darker the color. As shown in Figure 2(a), typical bugholes are nearly circular. However, irregularly shaped bugholes also exist, as shown in Figure 2(b). Additionally, because of the smaller depth of some bugholes, the contrast with the normal concrete surface is not obvious, as seen in Figure 2(c). Color-difference is the color that deviates from the color of a normal or desired concrete surface. It has neither a particular shape nor a clear border, as shown in Figure 3.

The previous studies ignored the situation that multiple defects existed on a concrete surface at the same time. However, this situation often occurs during the concrete casting stage. The concrete surface bughole images were classified according to a combination of surface defects, as shown in Table 1.

3. Proposed Method for Concrete Surface Bughole Detection

Figure 4 shows the proposed method's general flow with training steps (solid lines) and testing steps (dashed lines). The DCNN was trained for a total 30 training epochs. The model was evaluated on the validation set after every training epoch. When the DCNN classifier was well trained, the testing images were scanned by the validated classifier to generate a report of bugholes.

3.1. Database Establishment. To train a CNN classifier, raw images of concrete surfaces with multiple types of defects (including cracks, bugholes, and color-differences) were taken from several completed construction sites with a mobile phone camera. The shooting distance to the objects ranges from 1.0 to 5.0 m. The bugholes need to be visible in images with the naked human eye. The total number of raw images was 116, with a resolution of $3{,}120 \times 4{,}160$ pixels. Among the 116 raw images, 80 were used for training and validation and 36 were cropped into 800 small images (256×256 pixel resolutions) for testing. Due to the small size of bugholes, the pixels of a single bughole range from 18×18 to 60×60. The 80 raw images were cropped into smaller images (28×28 pixel resolutions), which were manually annotated as bughole (i.e., positive sample) or not bughole (i.e., negative sample) to generate a database, as shown in Figure 5.



(a)

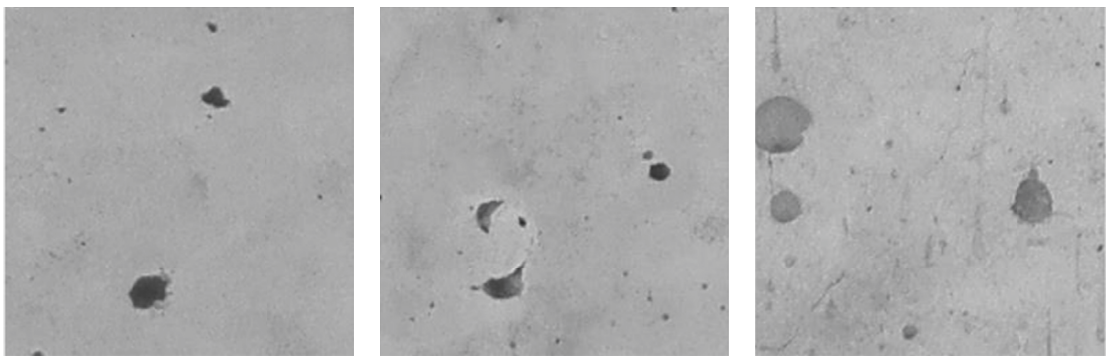
(b)

(c)

Figure 1: Concrete surface crack images.

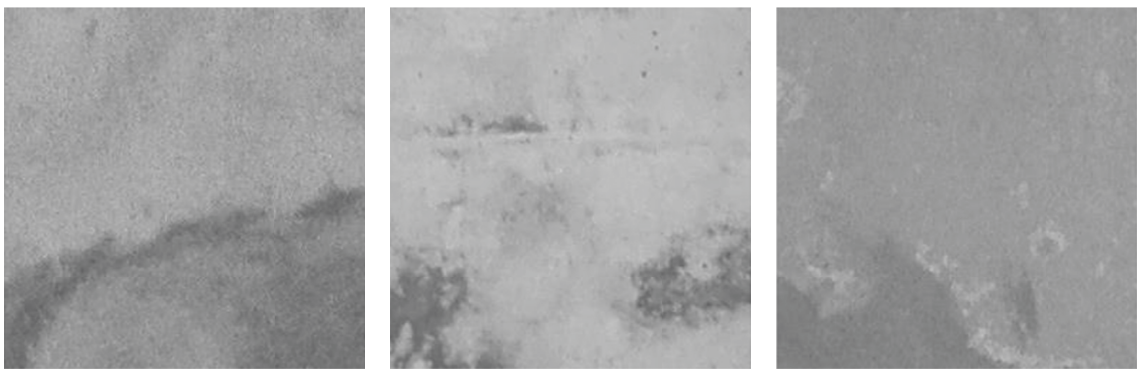
The total number of prepared training images in the database was 4 K. According to the ratio of training set, validation set = 4 : 1 [49], the number of training set images was 3.2 K, and the number of validation set images was 0.8 K.

3.2. Overall Architecture. This section describes the overall architecture of the DCNN used in this study, including parameter selection of each layer. Figure 6 presents the DCNN architecture, which was the original configuration for concrete bughole detection. The first layer was the input layer of $28 \times 28 \times 3$ pixel resolutions, where each dimension indicated height, width, and channel (i.e., red,



(a) (b) (c)

Figure 2: Concrete surface bughole images.



(a) (b) (c)

Figure 3: Concrete surface color-difference images.

Table 1: Image categories of concrete surface bughole.

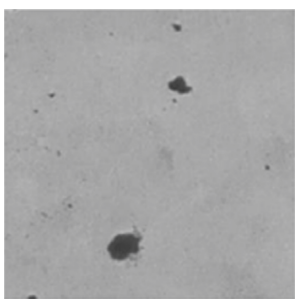
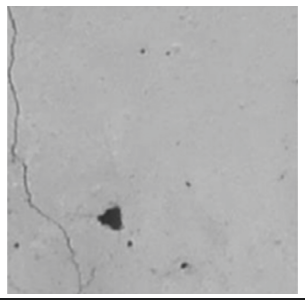
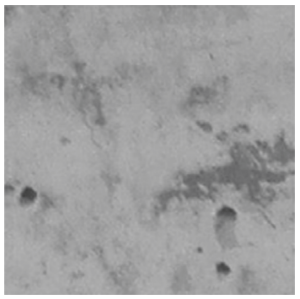
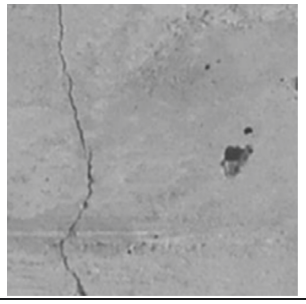
Category	Image	Description
1		Only bughole
2		Crack + bughole

Table 1: Continued.

Category	Image	Description
3		Color-difference + bughole
4		Crack + color-difference + bughole

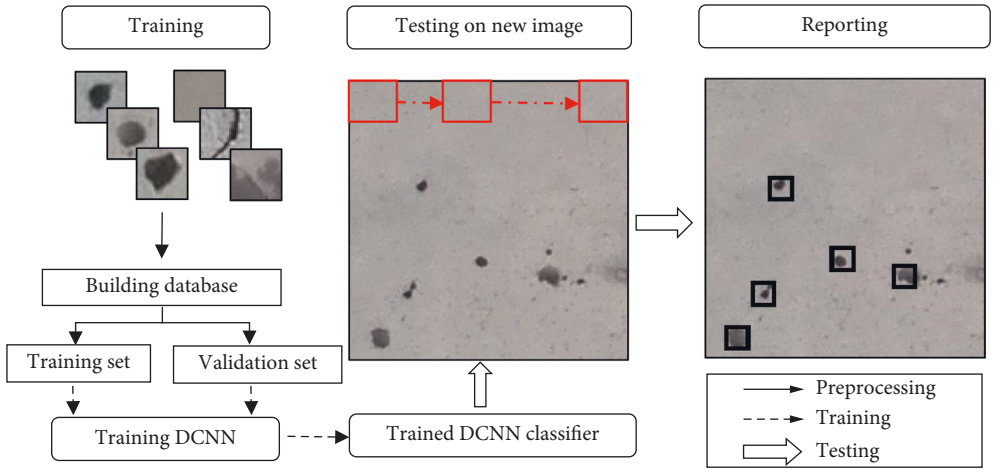


Figure 4: Flowchart for detecting concrete surface bugholes.

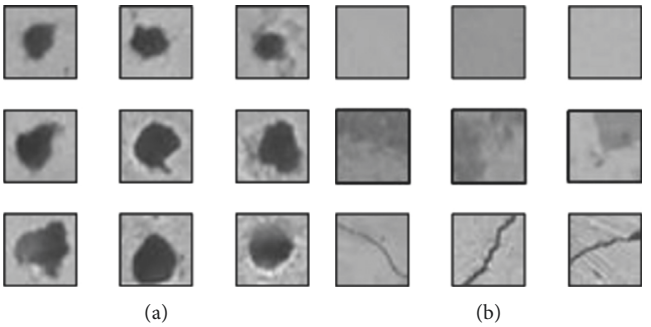


Figure 5: Positive and negative samples.

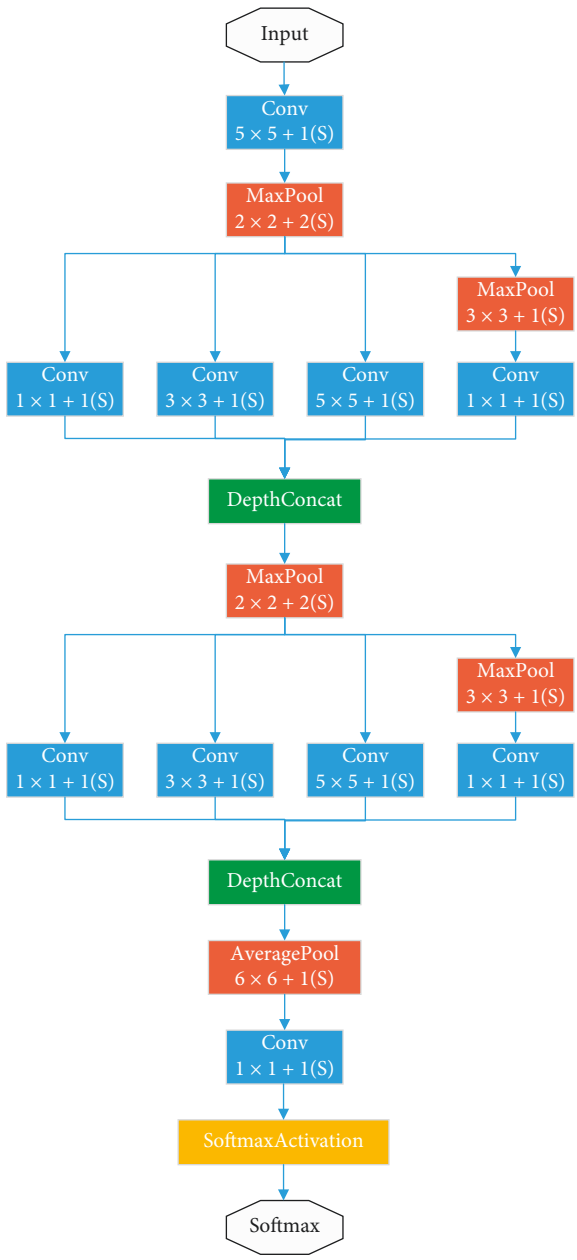


Figure 6: Overall architecture of the proposed DCNN.

green, and blue), respectively. Due to the relatively small size of input image (28×28 pixels) and the limited number of labeled examples in training set (less than 10 K), this network utilized the inception modules architecture proposed by Szegedy et al. [51]. The inception architecture was able to approximate an optimal local sparse structure in convolution vision network, which allowed for utilizing efficient dense computation instead of insufficient numerical calculation on nonuniform sparse data structure directly. By applying the inception modules into the traditional convolution network structure, less network parameters can be obtained, which meant that the model would be more robust to the overfitting, especially when a small dataset was used. Besides, the batch normalization layer was inserted before activation function in all layers, which made the network easier to train and improved the generalization ability of final model [52]. Table 2 shows the convolution configuration and inception architecture.

The convolution layer has proved to be greatly effective in extracting different features from images. Instead of the fully connected layer in traditional neuron network, each neuron of convolution layer is connected to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (equivalently, this is the filter size). The extent of the connectivity along the depth axis is always equal to the depth of the input volume. Meanwhile, the filter has the shared weights for input images. For the first few layers with local receptive fields, convolution layers can extract elementary visual features such as oriented edges, end-points, and corners. These features are then combined by the subsequent layers in order to detect high-order features. For example, suppose that the input volume has size of $28 \times 28 \times 3$ (e.g., an input image with three channels of red, green, and blue). If the receptive field (or the filter size) is 5×5, then each neuron in the convolution layer will have weights to a $5 \times 5 \times 3$ region in the input volume, for a total of $5 \times 5 \times 3 = 75$ weights (and +1 bias parameter), giving output of size $(28 - 5)/1 + 1 = 24$. So, after the convolution layer, the feature map gets a size of 24×24.

In general, the pooling layers will be periodically inserted into the convolution layers of a CNN architecture, reducing the number of parameters and saving computation resources required for data storage, which also avoids overfitting to a certain extent. The pooling units can perform different functions, such as max pooling, average pooling, or even L2-norm pooling, of which the max pooling was the most commonly used [53]. It divides the input image into several rectangular areas and outputs the maximum value for each subarea. Figure 7 shows an example of max pooling, with a stride of 2, where the pooling layer output size is calculated by the equation in the figure.

The ReLU layer applies the nonsaturating activation function $f(x) = \max(0, x)$ to perform a threshold operation on each element of the input. It can increase the nonlinearity of the decision function and the entire network and will not affect the receptive fields of the convolution layer. In addition, the sigmoid function $f(x) = (1 + e^{-x})^{-1}$ and the saturating hyperbolic tangent $f(x) = \tanh(x)$ are often used to increase the nonlinearity. Figure 8 depicts several examples of nonlinear functions. Compared to other functions, ReLU function is more popular, because it can speed up the neural network's training speed without significantly affecting the generalization accuracy of the model [45].

The loss layer is used to determine how the training process penalizes the difference between the predicted and actual results of the network, which is usually the last level of the network. The softmax function is often used in the last layer of the CNN architecture, as the output layer, to classify input data. The softmax function is given by Equation (1), which is expressed as the probabilistic expression,

Table 2: Dimensions of layers and operations.

Type	Patch size/stride	Output size	Depth	#1 × 1	#3 × 3	#5 × 5	Pool proj.	Params
Input		28 × 28 × 3	0					
Convolution	5 × 5/2	24 × 24 × 20	1					1.5 K
Batch norm		24 × 24 × 20	0					
Max. pool	3 × 3/2	12 × 12 × 20	0					
Inception (2)		12 × 12 × 64	2	16	32	8	8	10.3 K
Batch norm		12 × 12 × 64	0					
Max. Po1ol	2 × 2/2	6 × 6 × 64	0					
Inception (3)		6 × 6 × 512	2	192	208	48	64	214 K
Batch norm		6 × 6 × 512	0					
Avg pool	6 × 6/1	1 × 1 × 512	0					
Convolution	1 × 1/1	1 × 1 × 10	1					5.1 K
Softmax		1 × 1 × 1	0					0.01 K

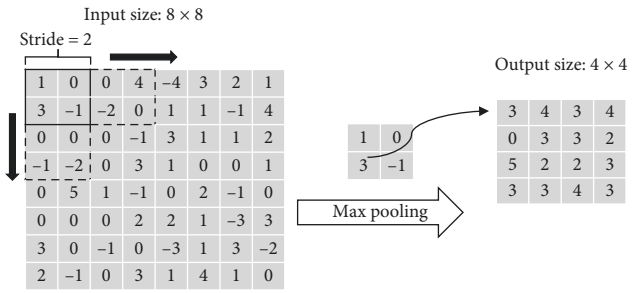


Figure 7: Pooling example: output size = $(I - P)/S + 1$, I = input size, P = pooling size, S = stride; $(8 - 2)/2 + 1 = 4$.

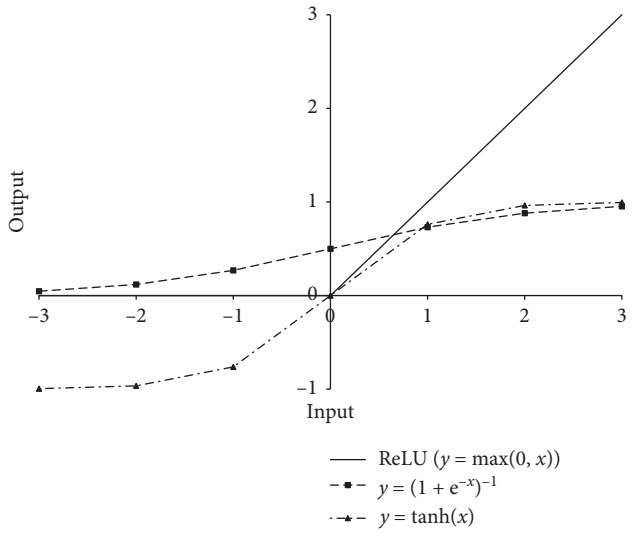


Figure 8: Nonlinear activation functions.

$p(y^{(i)} = n \mid x^{(i)}; W)$, for the ith training example out of m training examples, the jth class out of n classes, and weights, W, where $W_n^T x^{(i)}$ are inputs of the softmax layer. The sum of the right-hand side for the ith input always returns 1, because the function always normalizes the distribution. In other words, the following equation returns probabilities of each input's individual classes:

$$p\left(y^{(i)} = n \mid x^{(i)}; W\right) = \frac{1}{\sum_{j=1}^{n} e^{W_j^T x^{(i)}}} \begin{bmatrix} e^{W_1^T x^{(i)}} \\ e^{W_2^T x^{(i)}} \\ \vdots \\ e^{W_n^T x^{(i)}} \end{bmatrix}, \quad \text{for } i = 1, \ldots, m. \tag{1}$$

The network is trained using a stochastic gradient descent (SGD) algorithm with a minibatch size of 100 out of 4,000 images. SGD, using backpropagation, is considered the most efficient and simplest way to minimize deviations [54, 55]. A disadvantage of the SGD method is that its update direction is completely dependent on the current batch. Thus, its update is very unstable. A number of improvements have been proposed and used, including the proposed use of a momentum method, in which the stochastic gradient of momentum reduction remembers the updated Δw at each iteration and determines the next update as a linear combination of the gradient and the previous update [56]:

$$\begin{aligned} \Delta w &:= \alpha \Delta w - \eta \nabla Q_i(w), \\ w &:= w + \Delta w, \end{aligned} \tag{2}$$

that leads to

$$w := w - \eta \nabla Q_i(w) + \alpha \Delta w, \tag{3}$$

where the parameter, w, which minimizes $Q_i(w)$, is to be estimated and η is a step size (learning rate).

As small and decreasing learning rates were recommended [57], the exponential decay learning rates depicted in Figure 9 were used in this study. The x-axis represents epochs, so that the learning rates are updated each time. As shown in Figure 9, the error tends to converge after 30 iterations. Weight decay and momentum parameters are assigned as 0.0001 and 0.9.

4. Testing and Discussion

To evaluate bugholes detection performance of the trained DCNN, 36 images not used in training were cropped into

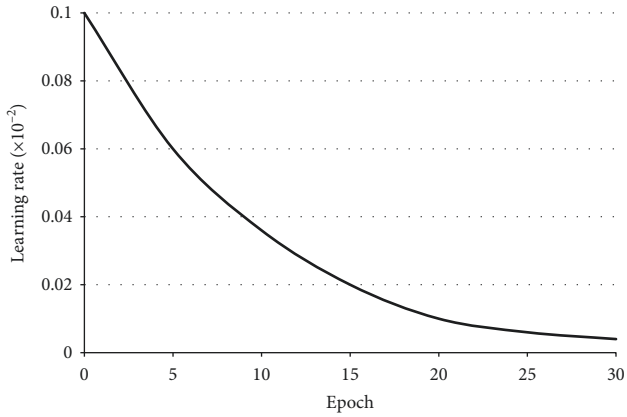


Figure 9: Learning rate.

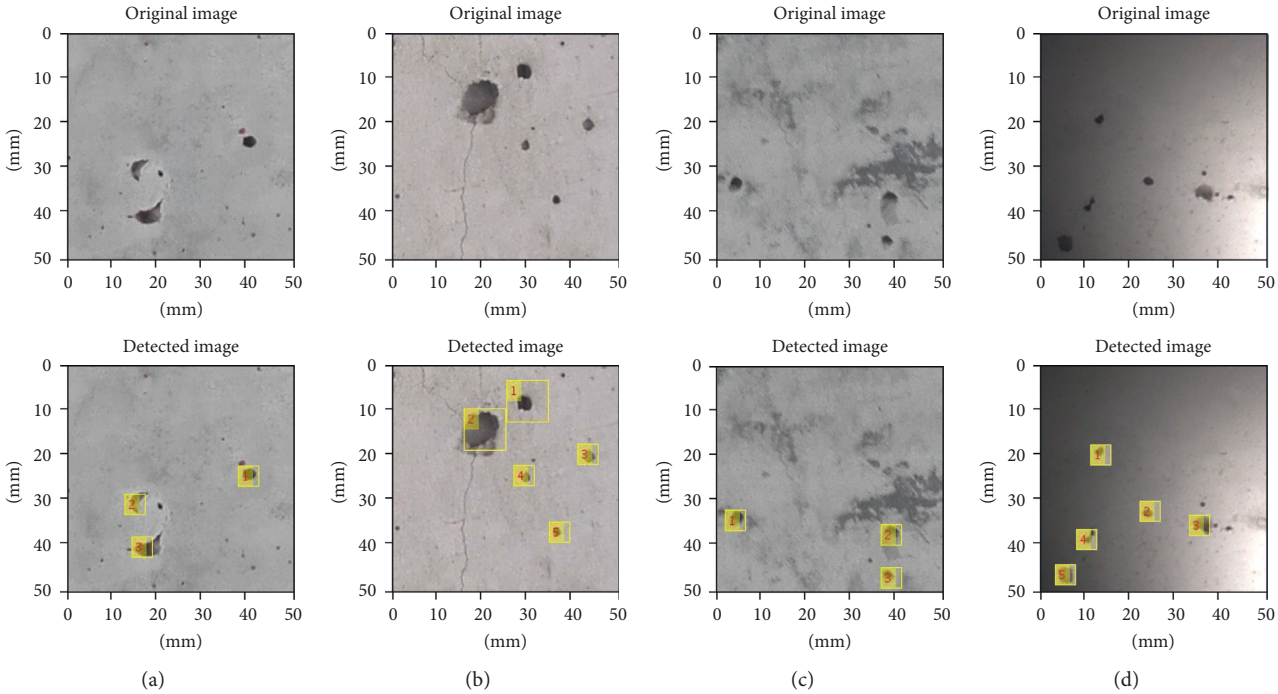


Figure 10: Results of image testing using the trained DCNN. Concrete surface with (a) only bugholes, (b) both bughole and crack, (c) both bughole and color-difference, and (d) nonuniform illumination.

800 small images (256 × 256 pixel resolutions) for testing. These images were taken from several construction sites. The test results of some images are shown in Figure 10. The yellow box marks the discriminated bugholes, and the number inside indicates the labeled number of bugholes in one image.

From the test results above, the proposed DCNN showed excellent performance in detecting bugholes on the concrete surfaces. When there simultaneously exist multiple types of defects, including cracks and color-differences, the trained DCNN also accurately detects the bughole defect without interference from other defects, as shown in Figures 10(b) and 10(c). Furthermore, under the circumstance of nonuniform illumination, the trained DCNN can also exclude the noise of strong light or shade and recognize the bugholes with a high accuracy, as shown in Figure 10(d).

By the image detection experiment, it can be found that the error mainly occurs at the edge of the image and the color-difference area, as shown in Figure 11. The identification error, as shown in Figure 11(a), is basically caused by improper images cropping, and the reason for identification error in Figure 11(b) seems to be the shape of stronger color-difference region, similar to bugholes.

By performing image testing on the trained DCNN, the recognition accuracy reached 96.43%, as shown in Table 3. Note that if pixels that are actually nonbughole are erroneously detected as bughole, it will be regarded as false positives; conversely, if pixels that are actually bugholes are

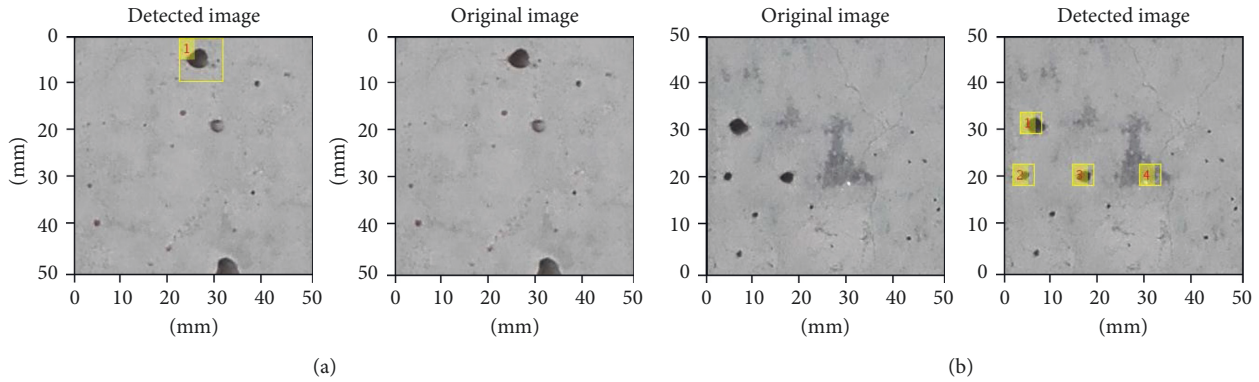


Figure 11: Results of image testing with (a) false-negative error and (b) false-positive error.

Table 3: Accuracy of the proposed DCNN on testing images.

Number of images	800
Number of pixels	52,428,800
Number of pixels with false-positive errors	592,445
Number of pixels with false-negative errors	1,279,263
Percentage of false-positive errors	1.13%
Percentage of false-negative errors	2.44%
Accuracy	96.43%

incorrectly classified as nonbughole, it will be considered as a false negative.

5. Comparative Study

In order to compare the performance of the proposed DCNN-based bughole detection method and the traditional edge detection methods, it is necessary to conduct a comparative study. According to the study of related researchers, the Otsu method has the highest accuracy of identify the bugholes on concrete surface compared with the global threshold method, the nonmaximum suppression edge detection method, and the canny edge detection method. Talab et al. and Liu et al. [24, 36] used the Otsu method to detect cracks in the image and achieved good detection results. Lim et al. [23] used the Laplacian of Gaussian (LoG) algorithm to detect cracks, and the results showed that the LoG algorithm successfully detected the crack in the image. Therefore, the Laplacian of Gaussian (LoG) algorithm and the Otsu method were chosen for comparative study.

The bughole detection results under different methods are shown in Figure 12. For these four types of images, the method presented in this paper all had good recognition results and can accurately identify the region and the number of bugholes in images. Moreover, the trained DCNN can avoid the interference of cracks, color-differences, and uneven illumination on the concrete surface. When there exist only bugholes on the concrete surface, the LoG method and the Otsu method can identify bugholes well, as shown in Figure 12(a). However, when there exist multiple types of defects on the concrete surface as shown in Figures 12(b) and 12(c), the detection performance of both the LoG method and the Otsu method are prone to be poor. Both the Otsu method and the LoG method are hard to obtain good bughole detection results as expected because of the noise of color-difference. As shown in Figure 12(d), when the illumination of the concrete surface was uneven, the Otsu method could not detect the bugholes at the shadows; although the LOG method perform well in detecting bugholes, strong light would drastically lower the detection performance.

6. Conclusion

This paper proposed a deep-learning-based method to detect bugholes on concrete surface images. The concrete images required for the training, validation, and testing were taken from several construction sites by a mobile phone camera. The total number of raw images was 116. The 80 images were cropped into 4,000 smaller images of 28×28 pixel resolutions to build the database for training and validation processes, and the 36 images cropped into 800 small images (256×256 pixel resolutions) were used for the testing process. Due to the relatively small size of input image (28×28 pixels) and the limited number of labeled examples in training set (less than 10 K), this network utilized the inception modules architecture. From the results of the image test, the proposed DCNN had an excellent bughole detection performance and the recognition accuracy reached 96.43%, expanding the application of deep-learning-based methods in the detection of concrete surface defects.

The effects of noise such as illumination, shadows, and combinations of several different surface imperfections in real-world environments were considered. By the comparative study with the Laplacian of Gaussian (LoG) algorithm and the Otsu method, it is clear that the proposed DCNN has

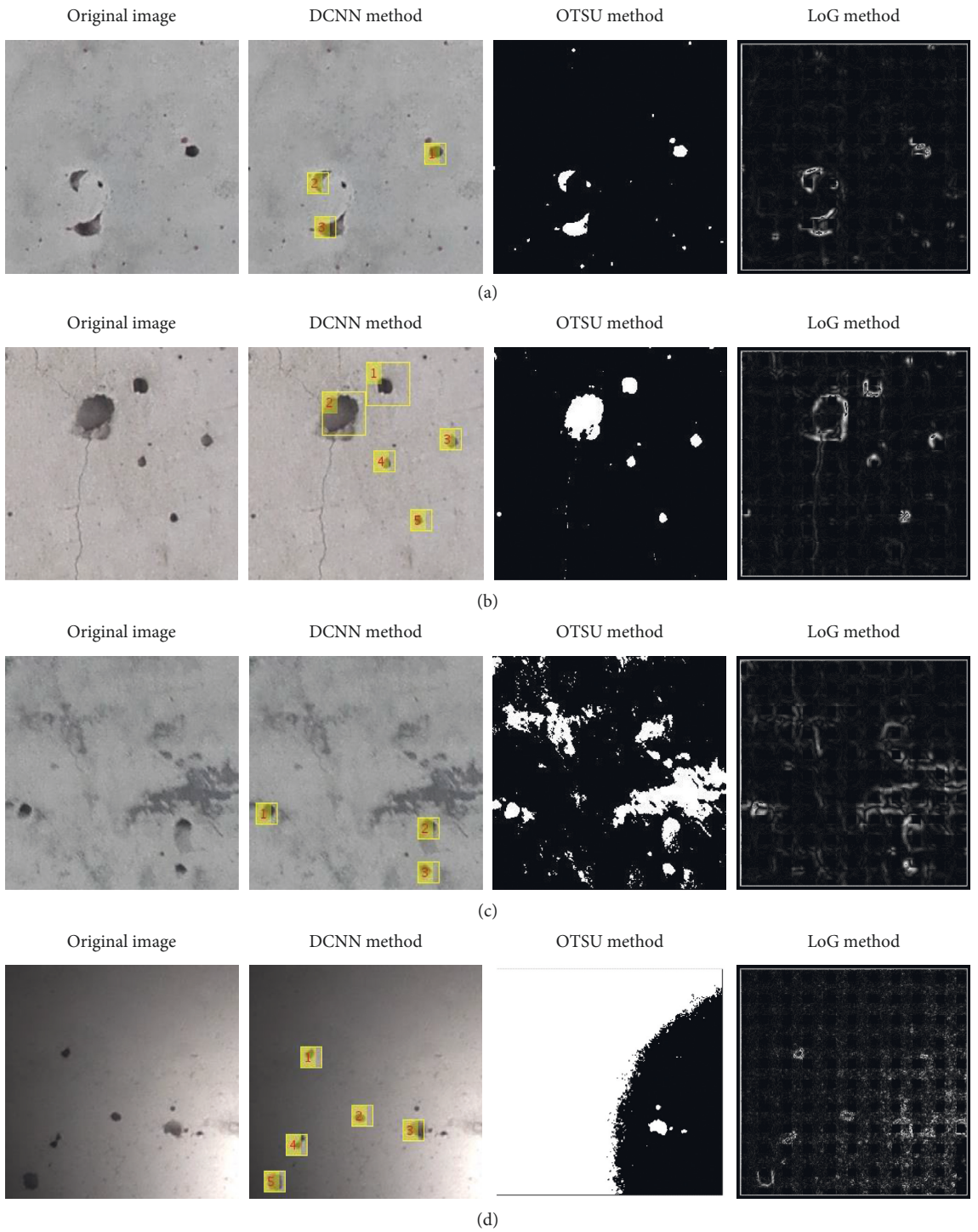


Figure 12: Comparison of image recognition results under different methods. (a) Type 1: only bugholes; (b) type 2: both cracks and bugholes; (c) type 3: both color-differences and bugholes; (d) type 4: uneven illumination.

good robustness and can avoid the interference of cracks, color-differences, and nonuniform illumination on the concrete surface; using both the Otsu method and the LoG method, it was difficult to detect bugholes effectively due to the interference of color-difference and nonuniform illumination.

However, a common limitation of deep-learning-based methods is that the network training requires a large amount of labeled training data, so it is necessary to add more images to extend the database. In the next stage, the major task is to establish a quality evaluation system of concrete surface based on computer vision.

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 work was financially supported by the Fundamental Research Funds for the Central Universities (2019CDXYTM0032, 2019CDQYTM028, and 106112017CDJXY200009) and National Natural Science Foundation of China (51608074).

References

[1] T. Ozkul and I. Kucuk, "Design and optimization of an instrument for measuring bughole rating of concrete surfaces," *Journal of the Franklin Institute*, vol. 348, no. 7, pp. 1377–1392, 2011.

[2] S. Paul, "Voids in concrete surfaces," *Asian College of Journalism*, vol. 67, pp. 868–874, 1970.

[3] T. J. Reading, "The bughole problem," *Asian College of Journalism*, vol. 36, pp. 1119–1134, 1972.

[4] A. S. Kalayci, B. Yalim, and A. Mirmiran, "Effect of untreated surface disbonds on performance of FRP-retrofitted concrete beams," *Journal of Composites for Construction*, vol. 13, no. 6, pp. 476–485, 2009.

[5] K. Ichimiya, T. Idemitsu, and T. Yamasaki, "The influence of air content and fluidity of mortar on the characteristics of surface voids in self-compacting concrete," *Doboku Gakkai Ronbunshu*, vol. 56, pp. 135–146, 2002.

[6] M. S. Thompson, *Blowholes in Concrete Sufface, Concrete*, no. 3, pp. 64–66, 1969.

[7] Concrete International Board-Comission W29, *CIB Report n. 24: Tolerances On Blemishes Of Concrete*, 1973.

[8] American Concrete Institute ACI 309.2R-98, *Identification and Control of Visible Effects of Consolidation on Formed Concrete Surfaces*, 2003.

[9] G. Lemaire, G. Escadeillas, and E. Ringot, "Evaluating concrete surfaces using an image analysis process," *Construction and Building Materials*, vol. 19, no. 8, pp. 604–611, 2005.

[10] C. Laofor and V. Peansupap, "Defect detection and quantification system to support subjective visual quality inspection via a digital image processing : a tiling work case study," *Automation in Construction*, vol. 24, pp. 160–174, 2012.

[11] J. Chermant, "Why automatic image analysis ? An introduction to this issue," *Cement and Concrete Composites*, vol. 23, no. 2-3, pp. 127–131, 2001.

[12] M. Coster and J. Chermant, "Image analysis and mathematical morphology for civil engineering materials," *Cement and Concrete Composites*, vol. 23, no. 2-3, pp. 133–151, 2001.

[13] S. Lee, L. Chang, and M. Skibniewski, "Automated recognition of surface defects using digital color image processing," *Automation in Construction*, vol. 15, no. 4, pp. 540–549, 2006.

[14] P. Chen, Y. Yang, and L. Chang, "Automated bridge coating defect recognition using adaptive ellipse approach," *Automation in Construction*, vol. 18, no. 5, pp. 632–643, 2009.

[15] P. Chen, Y. Yang, and L. Chang, "Box-and-Ellipse-Based ANFIS for bridge coating assessment," *Journal of Computing in Civil Engineering*, vol. 24, no. 5, pp. 389–398, 2010.

[16] P. Chen, H. Shen, C. Lei, and L. Chang, "Support-vector-machine-based method for automated steel bridge rust assessment," *Automation in Construction*, vol. 23, pp. 9–19, 2012.

[17] H. Shen, P. Chen, and L. Chang, "Automated steel bridge coating rust defect recognition method based on color and texture feature," *Automation in Construction*, vol. 31, pp. 338–356, 2013.

[18] H. Son, N. Hwang, C. Kim, and C. Kim, "Rapid and automated determination of rusted surface areas of a steel bridge for robotic maintenance systems," *Automation in Construction*, vol. 42, pp. 13–24, 2014.

[19] K. Liao and Y. Lee, "Detection of rust defects on steel bridge coatings via digital image recognition," *Automation in Construction*, vol. 71, pp. 294–306, 2016.

[20] H. Shen, P. Chen, and L. Chang, "Human-visual-perception-like intensity recognition for color rust images based on arti fi cial neural network," *Automation in Construction*, vol. 90, pp. 178–187, 2018.

[21] S. N. Yu, J. H. Jang, and C. S. Han, "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel," *Automation in Construction*, vol. 16, pp. 255–261, 2007.

[22] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, pp. 255–263, 2003.

[23] L. Lim, M. Hung, and S. Weihua, "A robotic crack inspection and mapping system for bridge deck maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 367–378, 2014.

[24] A. M. A. Talab, Z. Huang, F. Xi, and L. Haiming, "Detection crack in image using Otsu method and multiple filtering in image processing techniques," *Optik*, vol. 127, no. 3, pp. 1030–1033, 2016.

[25] J. Oh, G. Jang, S. Oh et al., "Bridge inspection robot system with machine vision," *Automation in Construction*, vol. 18, pp. 929–941, 2009.

[26] M. R. Jahanshahi and S. F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Automation in Construction*, vol. 22, pp. 567–576, 2012.

[27] M. Kim, J. C. P. Cheng, H. Sohn, and C. Chang, "A framework for dimensional and surface quality assessment of precast concrete elements using BIM and 3D laser scanning," *Automation in Construction*, vol. 49, pp. 225–238, 2015.

[28] A. Rimkus, A. Podviezko, and V. Gribniak, "Processing digital images for crack localization in reinforced concrete members," *Procedia Engineering*, vol. 122, pp. 239–243, 2015.

[29] M. Hamrat, B. Boulekbache, M. Chemrouk, and S. Amziane, "Flexural cracking behavior of normal strength, high strength and high strength fiber concrete beams, using Digital Image Correlation technique," *Construction and Building Materials*, vol. 106, pp. 678–692, 2016.

[30] H. Kim, E. Ahn, S. Cho, M. Shin, and S. Sim, "Comparative analysis of image binarization methods for crack identi fi cation in concrete structures," *Cement and Concrete Research*, vol. 99, pp. 53–61, 2017.

[31] L. Li, Q. Wang, G. Zhang, L. Shi, J. Dong, and P. Jia, "A method of detecting the cracks of concrete undergo high-temperature," *Construction and Building Materials*, vol. 162, pp. 345–358, 2018.

[32] M. Kim, H. Sohn, M. Asce, C. Chang, and M. Asce, "Localization and quantification of concrete spalling defects using

terrestrial laser scanning," *Journal of Computing in Civil Engineering*, vol. 29, no. 6, pp. 1–12, 2015.

[33] Z. Zhu and I. Brilakis, "Machine vision-based concrete surface quality assessment," *Journal of Construction Engineering and Management*, vol. 136, no. 2, pp. 210–218, 2010.

[34] W. R. L. da Silva and P. Štemberk, "Expert system applied for classifying self-compacting concrete surface finish," *Advances in Engineering Software*, vol. 64, pp. 47–61, 2013.

[35] M. Hirano, I. Yoshitake, A. Hiraoka, and Y. Inagawa, "Evaluation of air bubbles distributed on concrete surface of side wall of tunnel lining," *Cement Science and Concrete Technology*, vol. 67, no. 1, pp. 252–258, 2014.

[36] B. Liu and T. Yang, "Image analysis for detection of bugholes on concrete surface," *Construction and Building Materials*, vol. 137, pp. 432–440, 2017.

[37] I. Yoshitake, T. Maeda, and M. Hieda, "Image analysis for the detection and quanti fi cation of concrete bugholes in a tunnel lining," *Case Studies in Construction Materials*, vol. 8, pp. 116–130, 2018.

[38] S. Dorafshan, R. J. Thomas, and 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, 2018.

[39] B. Santhi, G. Krishamurthy, S. siddharth, and P. K. Ramakrishnan, "Automatic detection of cracks in pavements using edge detection operator," *Journal of Theoretical and Applied Information Technology*, vol. 36, pp. 199–205, 2012.

[40] L. I. Gang and J. U. Yongfeng, "Novel approach to pavement cracking detection based on morphology and mutual information," in *Proceedings of the 2010 Chinese Control and Decision Conference*, pp. 3219–3223, Xuzhou, China, May 2010.

[41] G. Wu, X. Sun, L. Zhou, H. Zhang, and J. Pu, "Research on crack detection algorithm of asphalt pavement," in *Proceedings of the 2015 EEE International Conference on Robotics and Automation*, pp. 1–6, Lijiang, China, 2015.

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

[43] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, pp. 333–338, 2012.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1904–1916, 2015.

[45] A. Krizhevsky, G. E. Hinton, and I. Sutskever, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the Neural Information Processing Systems Conference*, pp. 1–9, Lake Tahoe, NV, USA, December 2012.

[46] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," *Advances in Neural Information Processing Systems*, vol. 26, pp. 2553–2561, 2013.

[47] Y. A. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, no. 7553, pp. 436–444, 2015.

[48] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proceedings of the 2016 IEEE International Conference on Image Processing*, pp. 2381–8549, Phoenix, AZ, USA, September 2016.

[49] 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.

[50] K. C. P. Wang, A. Zhang, J. Q. Li, and Y. Fei, "Deep learning for asphalt pavement cracking recognition using convolutional neural network," in *Proceedings of the International Conference on Highway Pavements & Airfield Technology*, pp. 166–177, Pennsylvania, PA, USA, 2017.

[51] C. Szegedy, S. Reed, P. Sermanet, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Boston, MA, USA, 2015.

[52] S. Ioffe and C. Szegedy, "Batch Normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, Lille, France, July 2015.

[53] 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*, pp. 92–101, Springer-Verlag Berlin, Heidelberg, Thessaloniki, Greece, September 2010.

[54] J. R. Finkel, A. Kleeman, and C. D. Manning, "Efficient, feature-based, conditional random field parsing," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pp. 959–967, June 2008.

[55] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Proceedings of the Neural Networks: Tricks of the Trade*, pp. 9–48, Springer, Berlin, Heidelberg, November 2012.

[56] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initizalization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning Research*, pp. 1139–1147, Atlanta, GA, USA, June 2013.

[57] D. Wilson and T. Martinez, "The need for small learning rates on large problems," in *Proceedings of the International Conference on Neural Networks*, pp. 115–119, Washington, DC, USA, July 2001.