Deep learning-based crack damage detection using convolutional neural networks

Article *in* Computer-Aided Civil and Infrastructure Engineering · January 2017

DOI: 10.1111/mice.12263

Computer-Aided Civil And Infrastructure Engineering

Deep learning-based crack damage detection using convolutional neural networks

Young-Jin Cha*

Assistant Professor, Department of Civil Engineering, University of Manitoba, Winnipeg, MB Canada

&

Wooram Choi

PhD student, Department of Civil Engineering, University of Manitoba, Winnipeg, MB, Canada

Oral Büyüköztürk

Professor, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA02139, USA.

*Corresponding author

Abstract: *A number of image processing techniques (IPTs) have been implemented for detecting civil infrastructure defects in order to partially replace human-conducted on-site inspections. These IPTs are primarily used to manipulate images in order to extract defect features, such as cracks in concrete and steel surfaces. However, the extensively varying real-world situations (e.g., lighting and shadow changes) can lead to challenges to the wide adoption of IPTs. To overcome these challenges, this paper proposes a vision-based method using a deep architecture of convolutional neural networks (CNNs) for detecting concrete cracks without calculating the defect features. As CNNs are capable of learning image features automatically, the proposed method works without the conjugation of IPTs for extracting features. The designed CNN is trained on 40K images of 256×256 pixel resolutions and, consequently, records with about 98% accuracy. The trained CNN is combined with a sliding window technique in order to scan any image size larger than 256×256 pixel resolutions. The robustness and adaptability of the proposed approach are tested on 55 images of 5888×3584 pixel resolutions taken from a different structure which is not used for training and validation processes under various conditions (e.g., strong light spot, shadows, and very thin cracks). Comparative studies are conducted to examine the performance of the proposed CNN using traditional Canny and Sobel edge detection methods. The results show that the proposed method shows quite better performances and can indeed find concrete cracks in realistic situations.*

1 INTRODUCTION

Civil infrastructures, including bridges, dams, and skyscrapers, are becoming susceptible to losing their designed functions as they deteriorate from use. This inevitable process signifies urgent maintenance issues. For example, a number of bridges built across the United States between the 1950s and 1960s were designed to last 50 years; thus, most of them have already been used for their intended duration (AAoSHaT, 2008). Although this concern has motivated people to inspect these structures on a regular basis (Federal Highway Administration), on-site inspections still require closing bridge systems or building structures in order to diagnose them, due to limited human-resources. Because of this, many research groups have proposed structural health monitoring (SHM) techniques.

In order to establish SHM systems, vibration-based structural system identifications via numerical method conjugations have been used (Teidj et al., 2016; Chatzi et al., 2011; Rabinovich et al., 2007; Cha and Buyukozturk 2015). However, this approach still has several challenges for monitoring large-scale civil infrastructures due to various uncertainties and non-uniformly distributed environmental effects, among other matters. Although many works have had large-scale SHMs performed to cover large-scale structures (Kurata et al., 2012; Jang et al., 2010), dense instrumentation, such as installing numerous sensors, integrating data from distributed sources, and compensating for environmental effects are required (Xia et al., 2012;

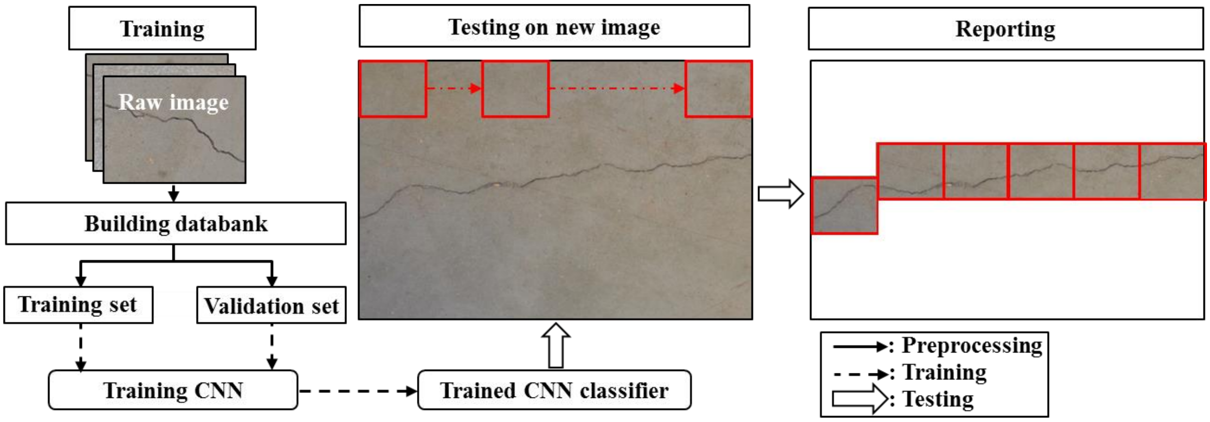


Figure 1 Flow chart for detecting concrete cracks

Cornwell et al., 1999). Lastly, confirming whether the collected data actually indicates structural damage, sensory system malfunction, noisy signals, or a combination of these is not easy before checking the sensing systems and structures in person.

A number of vision-based methods for detecting damages, primarily using image processing techniques (IPTs), have been proposed in order to redeem the complexities (Cha et al., 2017; Chen et al., 2015). One significant advantage of IPTs is that almost all superficial defects (e.g., cracks, corrosion) are likely identifiable. An early comparative study on finding concrete cracks using four edge detection methods—fast Haar transform (FHT), fast Fourier transform, Sobel edge detector, and Canny edge detector—was conducted by Abdel-Qader (2003), who defined FHT as the best solution for the task. This study was followed by an examination of modified edge detection problems (Nishikawa et al., 2012; Alaknanda et al., 2009; Yamaguchi et al., 2008; Sinha and Fieguth, 2006; Song and Civco, 2004). Yeum et al. (2015) proposed a study for detecting steel cracks using IPTs combined with a sliding window technique; this article shows the potential of IPTs very well. Despite their test example having many crack-like features due to the rusty surface of a steel beam, the unnecessary features were effectively removed, and strong crack-like features were extracted using the Frangi filter and the Hessian matrix-based edge detector (Frangi et al., 1999). However, edge detection is an ill-posed problem, as the results are substantially affected by the noises created, mainly from lighting and distortion, and no optimal solutions exist (Ziou and Tabbone, 1998). One effective method for overcoming these issues is implementing denoising techniques. Total variation denoising (Rudin et al., 1992) is a well-known technique that reduces noises from image data and enhances images' edge detectability. This technique was applied to a study (Cha et al., 2016) conducted to detect loosened bolts from images. However, the usage of such contextual (i.e., using prior knowledge) image processing is limited, since image data taken under real-world situations varies extensively.

One possible solution that has more real-world situation adaptability is using machine learning algorithms (MLAs) (LeCun et al., 1998), and several research groups have proposed techniques that can detect structural defects using this method (Butcher et al., 2014; Jiang and Adeli, 2007; Liu et al., 2002). These approaches first collect signals from non-destructive testing and evaluate whether or not the collected signals indicate defects. In recent years, many have implemented a combination of IPT-based image feature extractions and MLA-based classifications (O'Byrne et al., 2014; Wu et al., 2014; Jahanshahi et al., 2013; O'Byrne et al., 2013; Moon and Kim, 2011). Although they imported MLAs in their methods, the results of aforementioned approaches have inevitably inherited the complexities of sensor-implementations in addition to the false-feature extraction of IPTs. Many types of ANNs, including the probabilistic neural network (Ahmadlou and Adeli, 2010), have been developed and adapted to research and industrial fields, but convolutional neural networks (CNNs) have been highlighted in image recognition, which are inspired by the visual cortex of animals (Ciresan et al., 2011). CNNs can effectively capture the grid-like topology of images, unlike the standard neural networks (NNs), and they require fewer computations due to the sparsely connected neurons and the pooling process. Moreover, CNNs are capable of differentiating a large number of classes (Krizhevsky et al., 2012). These aspects make CNNs an efficient image recognition method (Simard et al., 2003; LeCun et al., 2015). The previous issue of CNNs was the need for a vast amount of labeled data, which came with a high computational cost, but this issue was overcome through the use of well-annotated databases (ImageNet; CIFIA-10 and CIFAR-100 dataset; MNIST DATABASE) and parallel computations using graphic processing units (Steinkrau et al., 2005). Owing to this excellent performance, a study for detecting railway defects using a CNN was later proposed (Soukup and

Huber-Mörk, 2014). Rail surfaces are homogenous, and the images are collected under controlled conditions. This cannot be considered the same as detecting concrete surface defects due to non-homogenous surface. Therefore, a carefully configured deep architecture and abundant dataset, taken under extensively varying conditions, is essential for dealing with the true variety of real-world problems.

In this study, we use CNNs to build a classifier for detecting concrete cracks from images. The first objective of this paper is to build a robust classifier that is less influenced by the noise caused by lighting, shadow casting, blur, and so on and to secure a wide range of adaptability. The second objective is to build an initial test bed that will allow other researchers to detect additional types of structural damage, such as delamination, voids, spalling, and corrosion of concrete and steel members. The main advantage of the proposed CNN-based detection of concrete cracks is that it requires no feature extraction and calculation compared to traditional approaches. This research's content is described as follows. Section 2 presents the synopsis of the proposed method. Section 3 introduces the overall architecture of the proposed CNN and explains the detailed CNN methodologies, including both the essential and auxiliary layers. Section 4 exposes specific hyperparameters that are used to train the CNN, as well as the considerations in building the databank (DB). Section 5 demonstrates how the framework evaluates test images and discussions regarding the method's performance and potential. Section 6 concludes this paper.

2 OVERVIEW OF THE PROPOSED METHOD

This section summarizes the entire process of our framework. Figure 1 shows the method's general flow with training steps (solid lines) and testing steps (dashed lines). In order to train a CNN classifier, raw images of concrete surfaces with a broad range of image variations, including lighting, shadow, etc. capable of potentially triggering false alarms, are taken from a complex Engineering building using a DSLR camera The definition of a crack in this paper is that it should be identifiable in images via the naked human eye. Some of the images used contains cracks, while others do not. A total of 332 raw images were used (i.e., 277 images with 4928×3264 pixel resolutions for training and validation and 55 images for testing with 5888×3584 pixel resolutions).

The 277 images are cropped into small images (256×256 pixel resolutions), which are manually annotated as crack or intact images in order to generate a DB. From the DB, the small cropped images are randomly selected in order to generate training and validation sets. The prepared training image set is fed into a CNN in order to build a CNN classifier for separating cracked from intact concrete images in the validation set. When the CNN classifier is validated through the validation set of images in the DB, 55 additional concrete images with 5888×3584 pixel resolutions are taken and scanned by the validated classifier in order to generate a report of crack damages.

3 METHODOLOGY

This section explains the overall architecture, the layers used in this study, and the backgrounds of each layer. The general CNN architecture can be created using multiple layers, such as input, convolution, pooling, activation, and output layers; convolution and pooling operations are conducted in the convolution and pooling layers. A deep CNN is defined when the architecture is composed of many layers. Some other auxiliary layers, such as dropout and batch normalization layers, can be implemented within the aforementioned layers in accordance with the purposes of use. MatConvNet (Vedaldi and Lenc, 2015) is used to perform this study.

3.1 Overall architecture

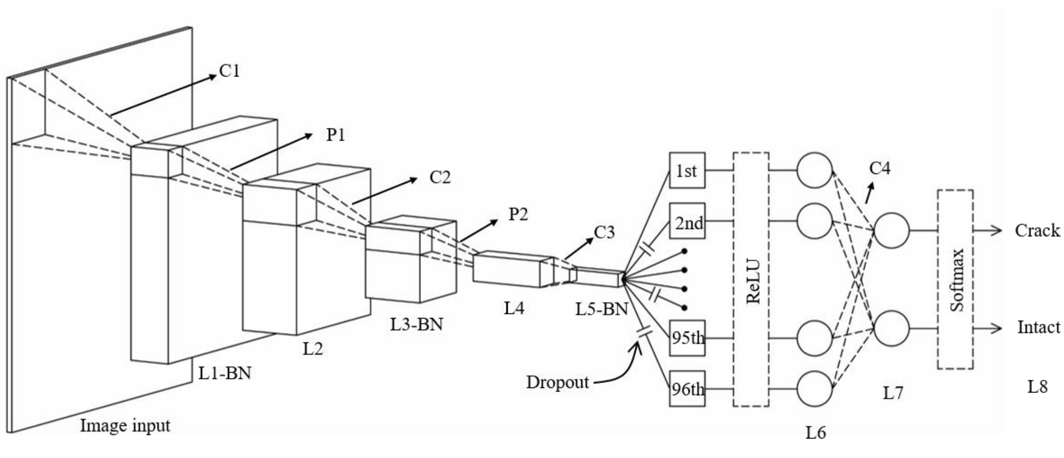


Figure 2 Overall architecture: L#: layers corresponding to operations (L1, L3, L5, L7: convolution layers; L2, L4: pooling layers; L6: ReLU layer; L8: softmax layer); C#: convolution; P#: pooling; BN: Batch normalization

Table 1
Dimensions of layers and operations

Layer	*Height*	*Width*	*Depth*	*Operator*	*Height*	*Width*	*Depth*	*No.*	*Stride*
Input	256	256	3	C1	20	20	3	24	2
L1	119	119	24	P1	7	7	-	-	2
L2	57	57	24	C2	15	15	24	48	2
L3	22	22	48	P2	4	4	-	-	2
L4	10	10	48	C3	10	10	48	96	2
L5	1	1	96	ReLU	-	-	-	-	-
L6	1	1	96	C4	1	1	96	2	1
L7	1	1	2	Softmax	-	-	-	-	-
L8	1	1	2	-	-	-	-	-	-

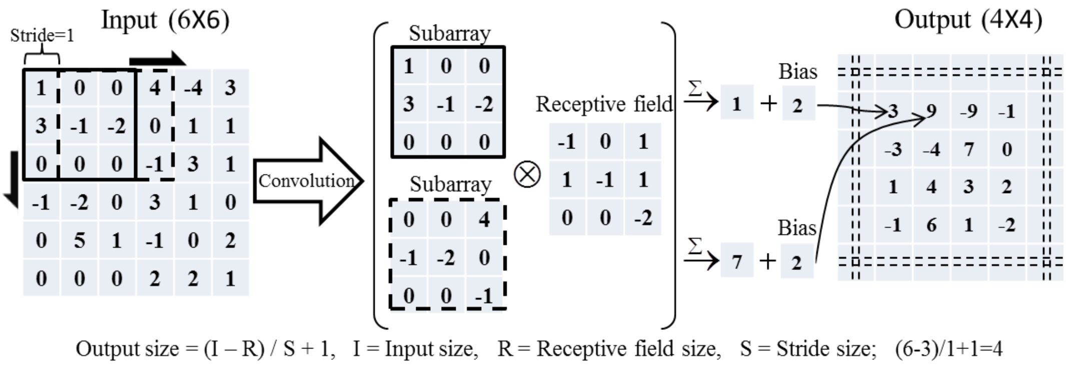


Figure 3 Convolution example

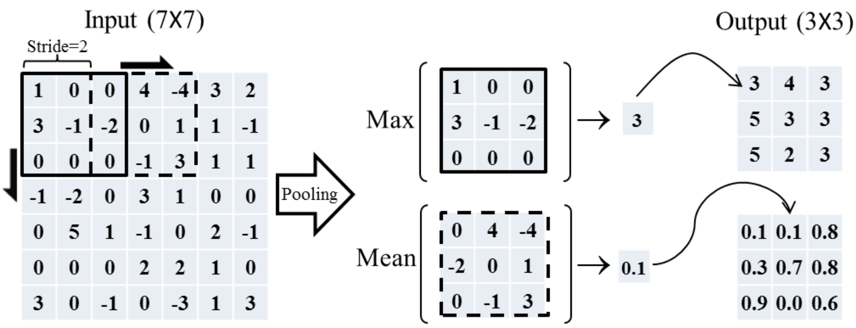
Figure 2 presents the CNN architecture, which is the original configuration for concrete crack detection. The first layer is the input layer of 256×256×3 pixel resolutions, where each dimension indicates height, width, and channel (e.g., red, green, and blue), respectively. Input data passes through the architecture and are generalized with spatial size reduction to 1×1×96 at L5. The vector, including the 96 elements, is fed into the rectified linear unit (ReLU) layer (see Figure 5), which is an activation layer. Finally, the softmax layer predicts whether each input data is a cracked or intact concrete surface after the convolution of C4. Table 1 lists the detailed dimensions of each layer and operation. Batch normalization and dropout layers, which cannot be visualized, are also used. Batch normalization (BN) layers are located after L1, L3, and L5, and a dropout layer is located after the batch normalization layer of L5.

3.2 Convolution layer

A convolution layer performs the following three operations throughout an input array as shown in Figure 3. First, it performs element-by-element multiplications (i.e., dot product) between a subarray of an input array and a receptive field. The receptive field is also often called the filter, or kernel. The initial weight values of a receptive field are typically randomly generated. Those of bias can be set in many ways in accordance with networks' configurations, and one of the most well-known initializations of bias can be found from Krizhevsky (2012). Both values are tuned in training using a stochastic gradient descent algorithm (Section 3.7). The size of a subarray is always equal to a receptive field, but a receptive field is always smaller than the input array. Second, the multiplied values are summed, and bias is added to the summed values. Figure 3 shows the convolutions ($\otimes$) of the subarrays (solid and dashed windows) with an input array and a receptive field. One of the advantages of the convolution is that it reduces input data size, which reduces computational cost. An additional hyperparameter of the layer is the stride. The stride defines how many of the receptive field's columns and rows (pixels) slide at a time across the input array's width and height. A larger stride size leads to fewer receptive field applications and a smaller output size, which also reduces computational cost, though it may also lose features of the input data. The output size of a convolution layer is calculated by the equation shown in Figure 3.

3.3 Pooling layer

Another key aspect of the CNNs is a pooling layer, which reduces the spatial size of an input array. This process is often defined as down-sampling. There are two different pooling options. Max pooling takes the max values from an input array's subarrays, whereas mean pooling takes the



Output size = (I – P) / S + 1, I = Input size, P = Pooling size, S = Stride size; (7-3)/2+1=3

Figure 4 Pooling example

mean values. Figure 4 shows each pooling method with a stride of two, where the pooling layer output size is calculated by the equation in the figure. Owing to the stride size being larger than the convolution example in Figure 3, the output size is further reduced to 3×3. A study by Scherer et al. (2010) showed that max pooling performance in image datasets is better than that of mean pooling. This paper verified that the architecture with max pooling layers outperforms those with mean pooling layers. Thus, all the pooling layers for this study are max pooling layers.

3.4 Activation layer

The most typical way to give non-linearity in the standard ANN is using sigmoidal functions, such as y=tanh(x), but it has been claimed by Nair and Hinton (2010) that saturating nonlinearities slow computations. Recently, the ReLU was introduced (Nair and Hinton, 2010) as a nonlinear activation function. Figure 5 depicts several examples of nonlinear functions. Briefly, while other nonlinear functions are bounded to output values (e.g., positive and negative ones, and zeros), the ReLU has no bounded outputs except for its negative input values. Intuitively, the gradients of the ReLU are always zeros and ones. These features facilitate much faster computations than those using sigmoidal functions and achieve better accuracies.

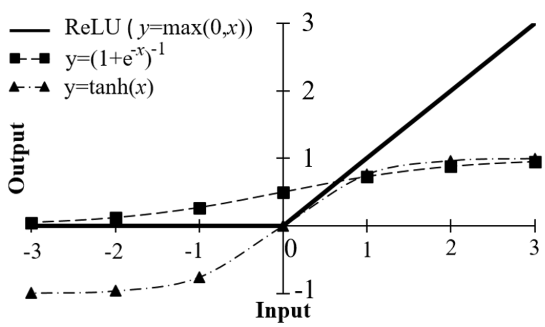


Figure 5 Nonlinear activation functions

3.5 Auxiliary layers

Overfitting has been a long-standing issue in the field of machine learning. This is a phenomenon where a network classifies a training dataset effectively but fails to provide satisfactory validation and testing results. To address this issue, dropout layers (Srivastava et al., 2014) are used. Training a network with a large amount of neurons often results in overfitting due to complex co-adaptations. The main idea of dropout is to randomly disconnect the connections between neurons of connected layers with a certain dropout rate. Accordingly, a network can generalize training examples much more efficiently by reducing these co-adaptations.

A well-known trick, taking the average values of a training dataset (i.e., whitening), has often been used to shorten network training time (LeCun et al., 2012). However, the distribution of layer's input shifts by passing through layers, which is defined as internal covariate shift, and this has been pointed out as being the major culprit of slow training speed. Ioffe and Szegedy (2015) proposed Batch normalization in order to adapt the similar effect of whitening on layers. As a result, this technique facilitates high-learning rate and leads to much faster network convergence.

3.6 Softmax layer

To classify input data, it is necessary to have a layer for predicting classes, which is usually located at the last layer of the CNN architecture. The most prominent method to date is using the softmax function given by Equation (1), which is expressed as the probabilistic expression $p(y^{(i)}=n \mid x^{(i)};W)$ for the i-th training example out of m number of training examples, the j-th class out of n number of 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 i-th input always returns as one, as the function always normalizes the distribution. In other words, Equation (1) returns probabilities of each input's individual classes.

$$P\left(y^{(i)}=n \mid x^{(i)};W\right)=\begin{bmatrix} p(y^{(i)}=1 \mid x^{(i)};W) \\ p(y^{(i)}=2 \mid x^{(i)};W) \\ \vdots \\ p(y^{(i)}=n \mid x^{(i)};W) \end{bmatrix}=\frac{1}{\sum_{j=1}^{n} e^{W_j^{\mathrm{T}} x^{(i)}}}\begin{bmatrix} e^{W_1^{\mathrm{T}} x^{(i)}} \\ e^{W_2^{\mathrm{T}} x^{(i)}} \\ \vdots \\ e^{W_n^{\mathrm{T}} x^{(i)}} \end{bmatrix} \quad \text{for } i=1\cdots m \tag{1}$$

3.7 Softmax loss and stochastic gradient descent

As the initial values of W are randomly assigned during training, the predicted and actual classes do not usually coincide. To calculate the amount of deviations between the predicted and actual classes, the softmax loss function is defined by Equation (2).

$$L=\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{n} 1\left\{y^{(i)}=j\right\}\log\frac{e^{W_j^{\mathrm{T}} x^{(i)}}}{\sum_{l=1}^{n} e^{W_l^{\mathrm{T}} x^{(i)}}}\right]+\frac{\lambda}{2}\sum_{j=1}^{n} W_j^2 \tag{2}$$

The new index L is introduced in order to indicate that $\sum_{l=1}^{n}\exp(W_l^{\mathrm{T}} x^{(i)})$ is independent from $\sum_{j=1}^{n} 1\{\cdot\}$. The term $1\{y^{(i)}=j\}$ is the logical expression that always returns either zeros or ones. In other words, if a predicted class of the i-th input is true for j class, the term returns ones, returning zeros otherwise. The last hyperparameter λ in the equation is a regularization (i.e., weight decay) parameter to penalize large weights, which is also a well-known trick for preventing overfitting (Bengio, 2012; Bottou, 2012).

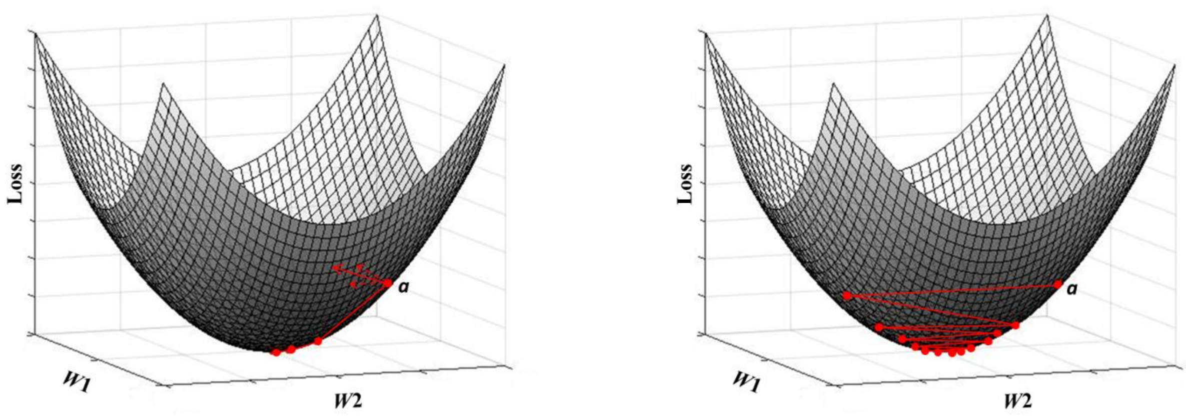
To narrow the deviations, an algorithm that updates receptive field weights is necessary for obtaining the expected results (i.e., predicting true classes). This process is considered for CNN training. There are several known methods, but stochastic gradient descent (SGD) using backpropagation is considered the most efficient and simplest way to minimize the deviations (LeCun et al., 2012). The standard gradient descent algorithm performs updating W on an entire training dataset, but the SGD algorithm performs it on single or several training samples. To accelerate the training speed, the momentum algorithm (Bengio, 2012) is also often used in SGD. The overall updating process is as follows. First, the gradient ∇_W of a loss function is calculated with respect to W, which is given by Equation (3). Second, the hyperparameters of momentum ε and learning rate α are introduced in Equation (4) to update ($\leftarrow$) velocity υ, where momentum is defined as mass times velocity in physics, but with unit mass being what is considered in SGD. Last, the weights are updated using Equation (5). A network can be tuned by repeating the explained process several times until Equation (5) converges. The superscript (i) indicates the i-th training sample, where the range of i is dependent on a minibatch size, which defines how many training samples out of the whole dataset are used. For example, if 100 images are given as the training dataset and 10 images are assigned as the minibatch size, this network updates weights 10 times; each complete update out of the whole data is called an epoch.

$$\nabla_W L\left(W;x^{(i)},y^{(i)}\right)=\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left\{1\left(y^{(i)}=j\right)-p\left(y^{(i)}=j \mid x^{(i)};W\right)\right\}\right]+\lambda W_j \tag{3}$$

$$\upsilon \leftarrow \varepsilon\upsilon-\alpha\nabla_{W_j} L(W;x^{(i)},y^{(i)}) \tag{4}$$

$$W_j \leftarrow W_j+\upsilon \tag{5}$$

The explained algorithm (i.e., gradient descent) is often described as a bead climbing down a convex bowl. For example, if it considers a simple example with two features,



(a) Small learning rate (b) Large learning rate

Figure 6 Example of gradient decent: the dashed arrows at point *a* in (a) shows the partial derivatives with respect $W1$ and $W2$, and the solid arrow is the gradient of the partial derivatives $(\partial L/\partial W1,\ \partial L/\partial W2)$ that always indicates the steepest gradient

(a) Fine images (b) Distorted images (c) Strong light spotted images

Figure 7 Examples of images used in training: the presented images have 256×256 pixel resolutions.

the number of weights is also two. Then, a loss function of the given example can be depicted in a three-dimensional parameter space, as shown in Figure 6. The z-axis indicates the loss, and the x and y axes are weights (i.e., $W1$ and $W2$), respectively. If the partial derivatives of the loss function at point *a* with respect to $W1$ and $W2$ is calculated, a vector (i.e., gradient) is obtained at this point. The projection of the calculated vector on the $W1$–$W2$ plane always tends to head towards the steepest gradient, which is towards the minimum of the loss function. In this process, if a learning rate is given by a small number, a network is trained efficiently, as shown in Figure 6(a). However, if a large learning rate is assigned, the network may converge slowly, as shown in Figure 6(b), or even diverge, which can result in overflow.

4. BUILDING A CLASSIFIER FOR DETECTING CONCRETE CRACKS

This section describes the considerations used in generating the DB and the underlying hyperparameters assigned to train a CNN. Configuring and choosing adequate hyperparameter (e.g., learning rates and regularization parameters) is tedious and no exact guidelines for those parameter optimizations are available. Thus, the optimal network architecture for this concrete crack detection must be explored via trial and error and guided by checking the validation set error (Bengio et al., 2015). However, several useful articles can be found from Bottou (2012), LeCun et al. (2012), and Bengio (2012). All of the described tasks in this paper are performed on a workstation with two GPUs (CPU: Intel Xeon E5-2650 v3 @2.3GHz, RAM: 64GB, and GPU: Nvidia Geforce Titan X × 2ea).

4.1 Databank generation

The total number of raw images is 332 (277 images with 4928×3264 pixel resolutions and 55 images with 5888×3584 pixel resolutions). The images are taken from a complex building at the University of Manitoba with a hand-held DSLR camera (Nikon D5200). Distances to the objects ranged from approximately 1.0–1.5m; however, some images are taken below a 0.1 m distance for tests, and each image's lighting intensity is substantially different. Among the 332 raw images, 277 images are used for training and validation processes, and 55 images are used for the testing process. The 277 raw images are cropped into smaller images of 256×256 pixel resolutions to build the DB for training and validation as a preprocessing step after annotating each image as either an intact or cracked image. Thus, the total number of the prepared training images in the DB is 40K. Images are randomly chosen from the DB for generating training and validation sets. The reason for choosing the relatively small cropping size is that a network trained on small images enables scanning of any images larger than the designed size. However, if smaller images than those selected here are used, the network may catch any elongated features, such as scratches. In addition, smaller images also make it harder to annotate images as defect or intact. The generated DB includes a broad range of image variations for a robust damage classifier, as shown in Figure 7.

As shown in Figure 8, some of the cropped images have cracks on the four edges of image spaces. Those kinds of images are strictly disregarded for the following reasons. First, as explained in Section 3, input images get smaller while the images pass through the CNN, which implies that cracks on edges have fewer chances to be recognized by a network than those with cracks in the middle of images during training. Second, it is not possible to identify whether such crack features are actually cracks or not, which can therefore lead to the training dataset's false annotations. Last, even if a trained network classifies such images, verifying whether the predicted class is false-positive or true-positive is not viable due to the hardly recognizable crack features. To tackle this issue, a sliding window technique is used in the testing step in order to detect cracks located in any positions of the image spaces, as described in Section 5.

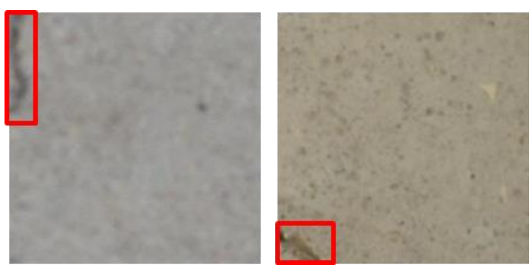


Figure 8 Disregarded images

4.2 Hyperparameters

The explained network is trained using an SGD algorithm with a minibatch size of 100 out of 40K images. Since small and decreasing learning rates are recommended (Wilson and Martinez, 2001), the logarithmically decreasing learning rates, which are depicted in Figure 9, are used. The *x* axis represents epochs so that the learning rates are updated each time. Weight decay and momentum parameters are assigned by 0.0001 and 0.9. The stride sizes of C1 – C3 and P1 – P2 are assigned to 2, and C4 is assigned to 1. The dropout rate at the dropout layer, located before the ReLU, is 0.5.

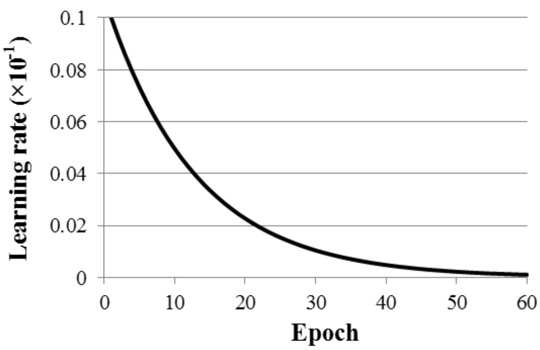


Figure 9 Learning rate

4.3 Training and validation results

Unlike other image-based studies for detecting concrete cracks, feature extraction techniques are not necessary, as CNNs learn features automatically by updating the weights of receptive fields. However, a trick taking the training dataset's mean values is used for the sake of efficient computation (LeCun et al., 2012). Figure 10 summarizes the training and validation results. The ratio of the number of crack and intact images is 1:1, with that of training and validation being 4:1. The training accuracy is thus calculated out of 32K images, and validation is calculated out of 8K images. The achieved accuracy is exceptional. The highest accuracies in training and validation are 98.22% at the 51th epoch and 97.95% at 49th epoch, respectively. The conjugation of two GPUs boosts the consequently recorded training speed by about 90 minutes until the 60th epoch, but the approximately estimated running time on only CPU is about 1 – 2 days. The trained CNN of the 51st epoch is used in testing, which is detailed in Section 5.

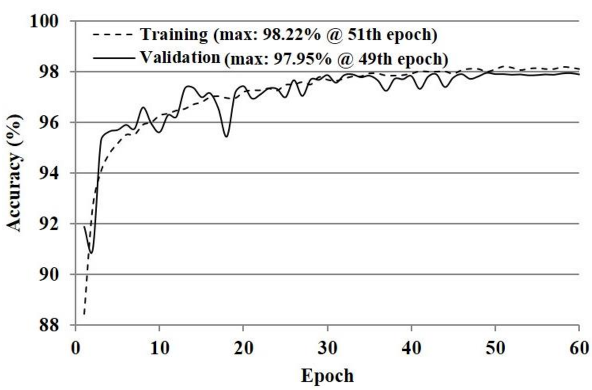
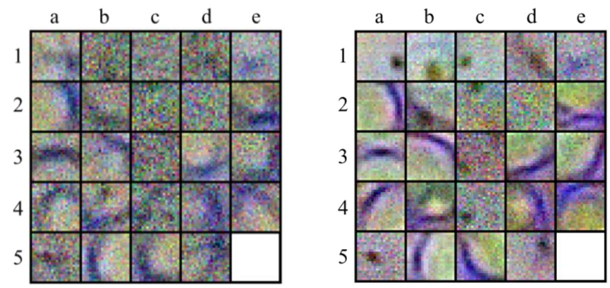


Figure 10 Accuracies for each epoch

Figure 11 represents the receptive field visualizations at the first layer (C1), where the visualizations of each receptive field are acknowledged as learned features of CNNs. Remember that the number of receptive fields of the designed architecture is 24 with 20×20×3 dimensions (Table 1). The visualized features provide intuitions that indicate whether the network needs more training and what kinds of features are recognized by the trained network. For example, Figure 11(b) shows clearer spots and lines than Figure 11(a), indicating that the network is well-trained. In Figure 11(b), the features of a2-4, b3-5, c5, d3-4, and e2-4 can be considered crack features, and those of a1-5, b1, c1-4, and d5 are most likely speculated as concrete surface cavities or aggregates in the training dataset. Receptive fields of a well-trained network generally have smooth patterns, but the



(a) Less-trained network (b) Well-trained network

Figure 11 Learned features : (a): 1st epoch, (b): 51st epoch

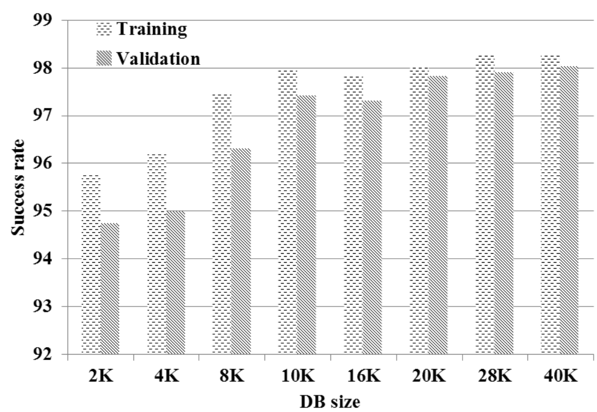


Figure 12 Results of parametric study

noisy features with various colors are still reasonable due to the complex and arbitrary patterns of concrete surfaces.

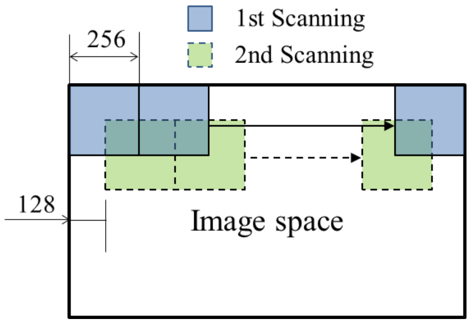


Figure 13 Scanning plan

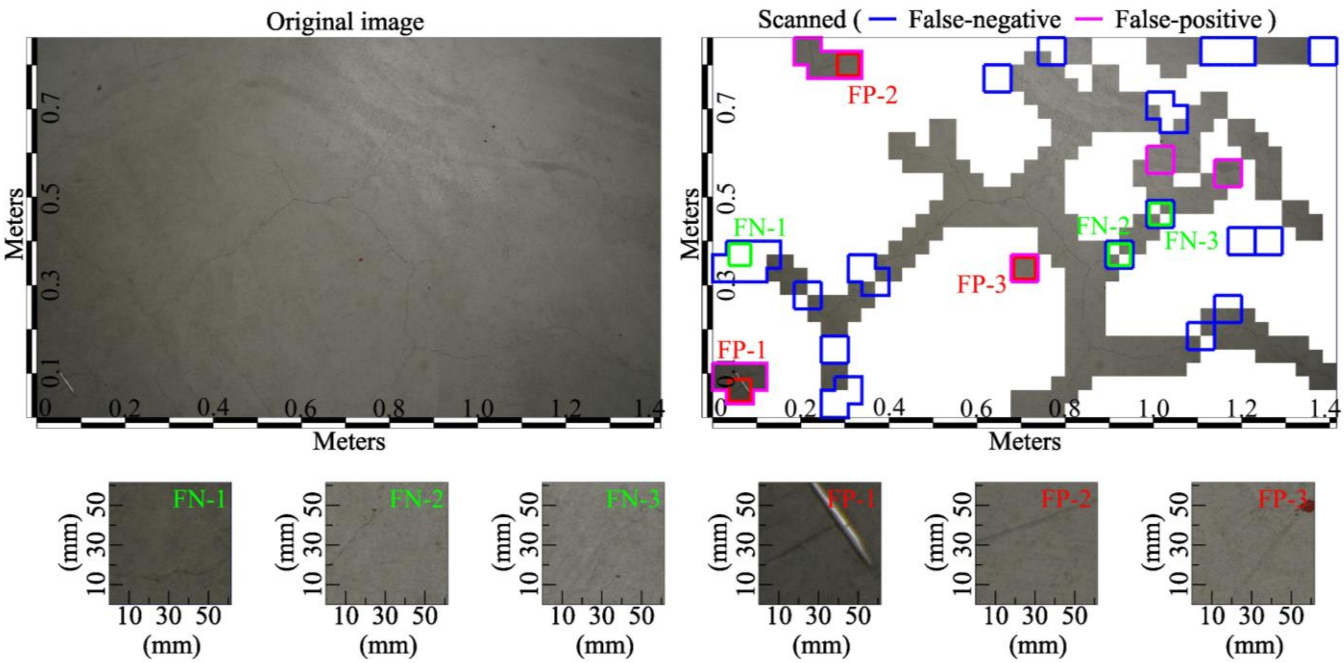
To approximate the desirable number of training images, a parametric study on the datasets comprising 2K, 4K, 8K, 10K, 16K, 20K, 28K, and 40K images with 256×256 pixel resolutions is conducted, as shown in Figure 12. The portions of training, validation, crack, and intact images are the same as the aforementioned 40K image dataset. The architectures for each training datasets are equal to Figure 2. From this parametric study, the required number of properly cropped images is at least 10K in order to obtain a reasonable CNN classifier, which obtains an accuracy of 97.42% in validation of the concrete crack detection problem.

5 TESTING IMAGES AND DISCUSSIONS

To validate the trained and validated CNNs from the previous section, extensive tests are conducted, the method by which the proposed framework scans test images is described, and the results are presented. Note that the image separation in Section 4.1 can cause misclassifications if the cracks are on the edges of image spaces. Therefore, a framework is designed to scan twice with a sliding window technique, as depicted in Figure 13.

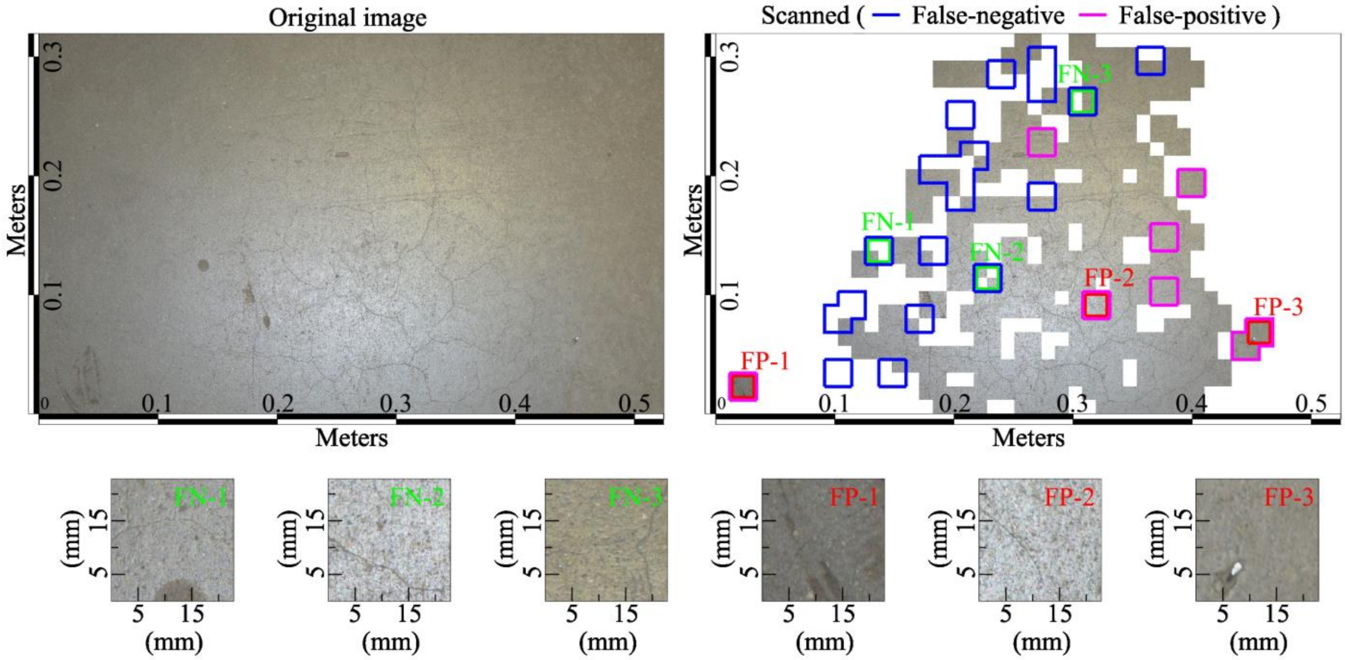
5.1 Testing the trained and validated CNN

To examine the performance of the trained and validated CNN from the previous section, 55 raw images that are not used for training and validation processes are used. These images are taken from a different structure that is a concrete tunnel to connect between Engineering building and Student Centre at the University of Manitoba. The testing results are presented in Table A1 in the Appendix. The achieved results are quite remarkable with a 97% accuracy, which is nearly identical to the accuracy (i.e., 98%) of the validation process in the previous section. Notably, the trained and validated CNN framework shows nearly the same performance without any degradation of the accuracy, even though totally different images are used for testing. Some of the tested images, which are taken under various conditions, are chosen and presented in Figure 14. These presented images can provide a clear understanding of how our framework functions. The image space axes can provide intuitions on each image's dimensions. The water-marked regions of each result image are recognized by the trained network as intact surfaces (false) or otherwise cracked (true). The distributions of false-negative (FN) and false-positive (FP) regions are highlighted as magenta and blue colored boxes. Some FN and FP regions are magnified and highlighted with green and red boxes. The testing duration is recorded as 4.5 seconds for each image. An image containing

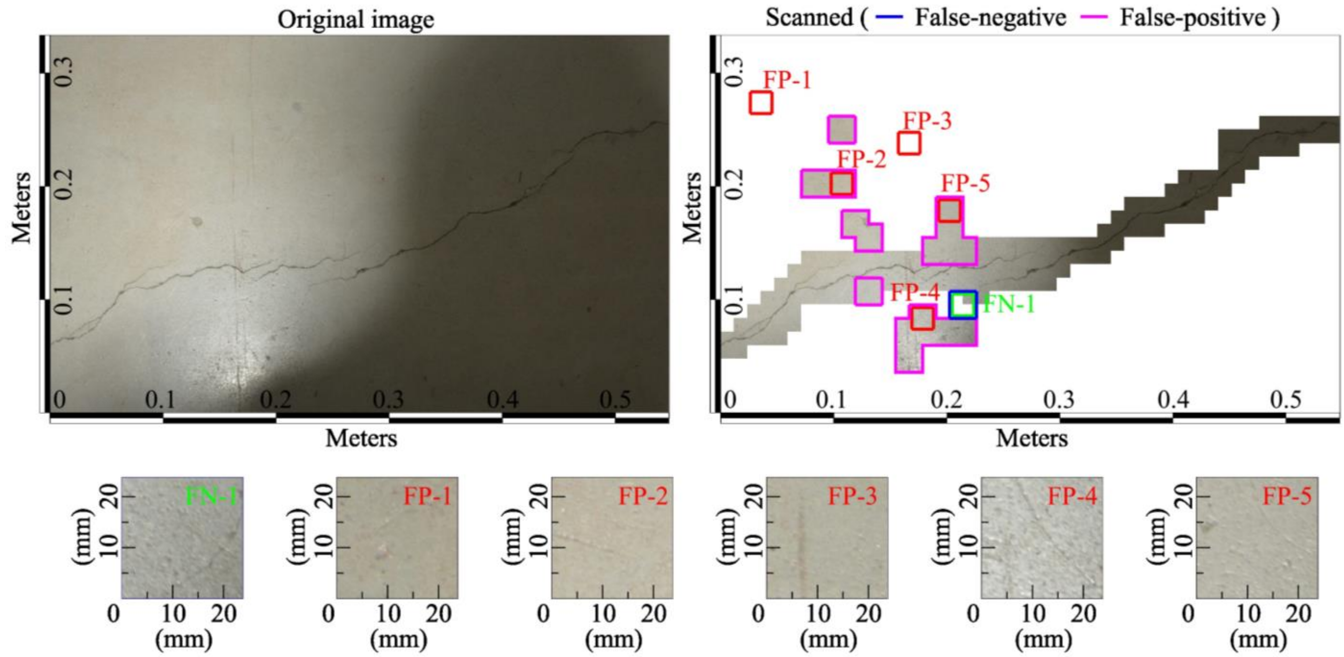


(a) Thin cracks

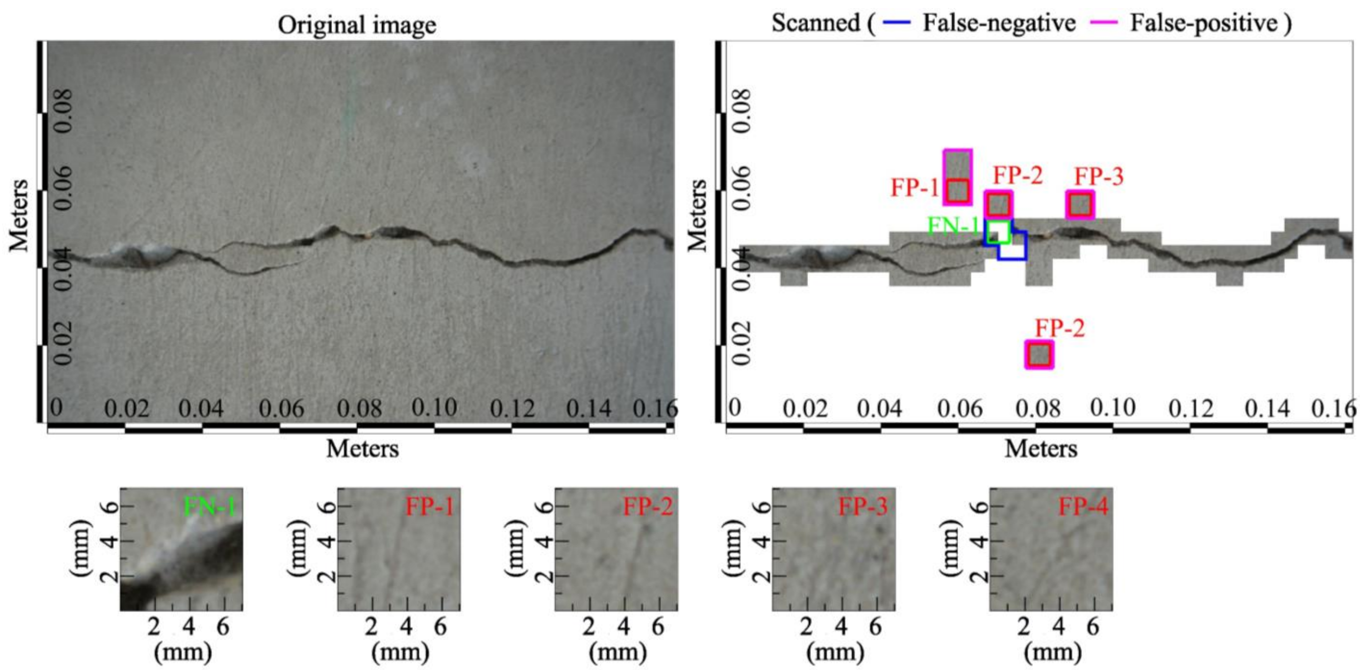
very thin cracks with uniform lighting conditions is tested as shown in Figure 14(a), where the thickest crack width is about



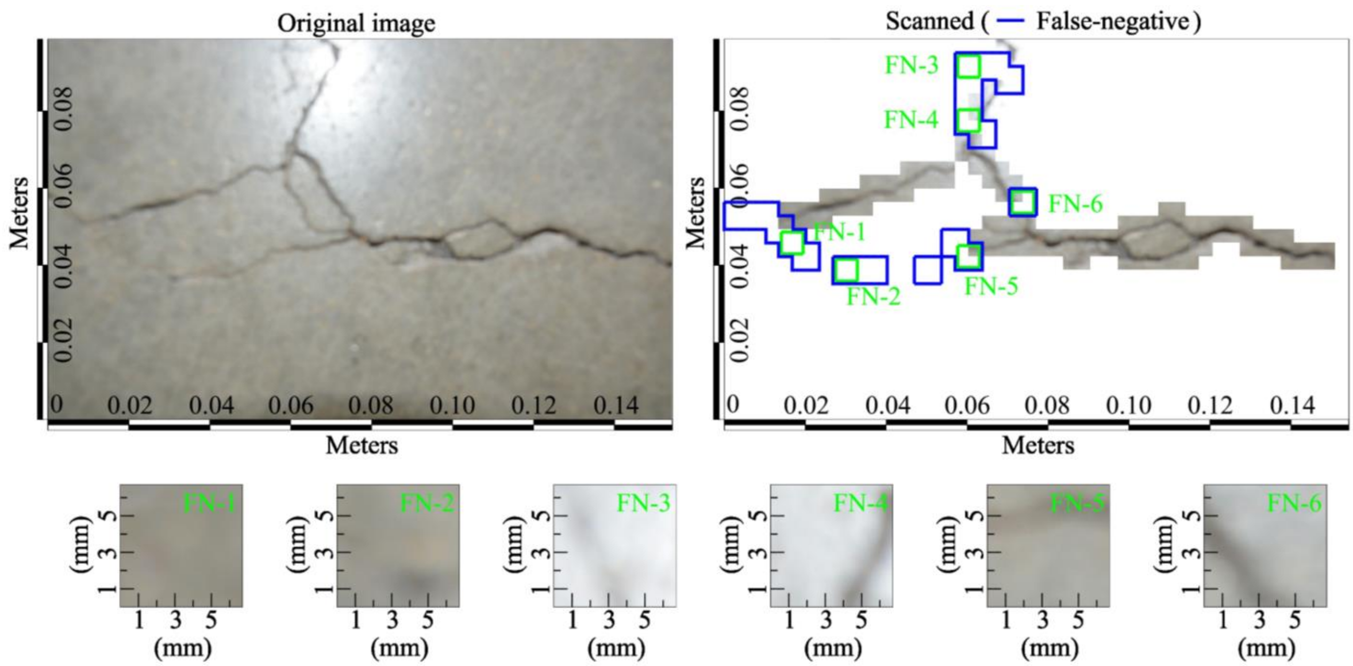
(b) Thin cracks and lighting spot



(c) Shadowed image



(d) Close-up image



(e) Close-up, blurred, and strong lighting spot

Figure 14 Results of image scanning using a trained CNN

1.5mm laying on 4 pixels.

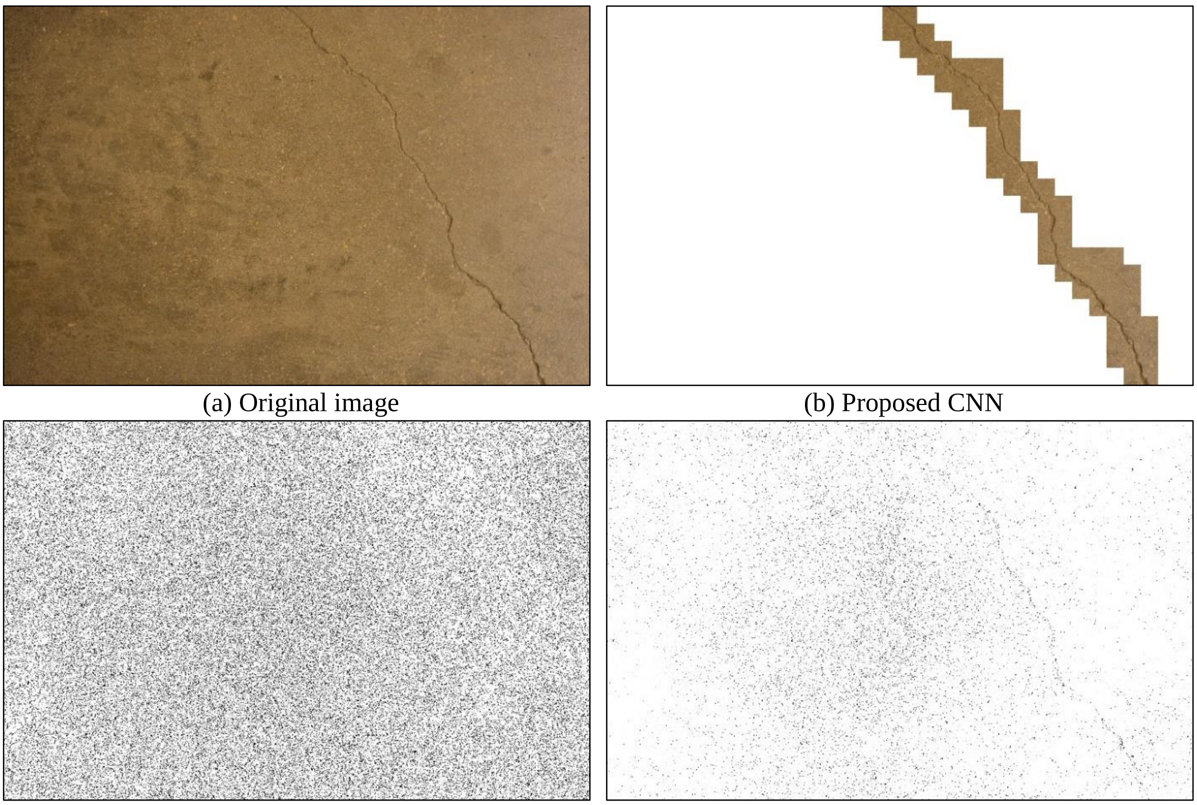
The majority of FN regions are distributed across the image center's periphery due to image distortions on the thin crack regions.

In order to study lighting condition sensitivity, one image with a lighting spot and another with a shadow area are tested, as shown in Figure 14(b) and (c). In Figure 14(b), FN is mainly detected on the edges of the lighting spot. In Figure 14(c), only one FN region exists, but a number of scratches are classified as FP regions. Comparing Figure 14(a) – (c), the proposed method is not susceptible to lighting conditions. In order to study the sensitivity on distance changes, a test image is deliberately taken approximately 70 mm away from a concrete surface and tested, as shown in Figure 14(d), and the result records 99% accuracy. The last test example, as shown in Figure 14(e), is taken about 60mm away. The image is blurred due to the small distance from the concrete surface, and the image contains a lighting spot on crack regions. Taking into account the presented results from Figure 14(a) and Figure 14(d) – (e), the proposed method is not susceptible to distance changes.

5.2 Comparative studies

In order to compare the performance of a new crack detection method with existing traditional methods, four different images from the 55 tested images taken under various conditions are used. The two most well-known traditional methods, Canny and Sobel edge detection are selected. The first case uses normal, uniform lighting, as shown in Figures 15 and 16. The proposed CNN provides clear crack information, as shown in Figures 15(a) and 16 (a). Although the Sobel edge detection provides some crack information, as shown in Figure 15 (d), it does not provide any meaningful information, as shown in the case represented in Figure 16(d). The Canny detection method provides no meaningful information regarding cracks with high levels of noise, as shown in Figures 15 (c) and 16(c). These two cases show that the performances of the Canny and Sobel edge detection methods are quite dependent on the image conditions; conversely, the proposed CNN is not affected by the conditions of the images.

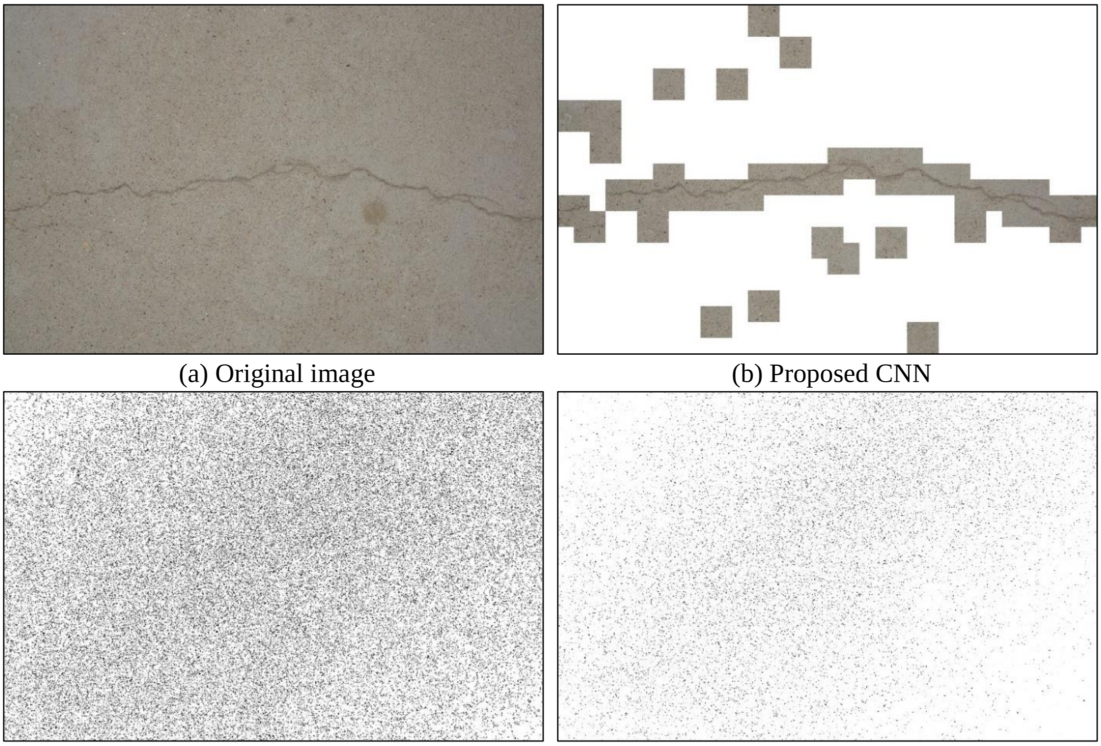
The second type of image includes thin cracks, as shown in Figure 17(a). This case also has results similar to the normal image case, as shown in Figures 15 and 16. The Sobel and Canny methods do not provide crack information properly with high levels of noise. However, the proposed CNN detects nearly all the thin cracks, with few errors, as shown in Figure 17(b). Figure 18 shows a case with a shadowed image. This case also shows that the proposed CNN detects the crack accurately, as shown in Figure 18(b). The Sobel method (Figure 18(d)) includes a dark area that is not damaged and is different from the original image. These cases show that the advantage of the proposed CNN method is that it can provide a raw, unprocessed image of the detected crack which allows engineers differentiate between cracks and noise. The Sobel method provides a processed image with gray scale, which made it difficult to determine if a dark area is either damage or noise. In the case that had thin cracks with lighting case (Figure 19), the Sobel and Canny methods provides no meaningful results, as shown in Figure 19(d). However, the proposed CNN detects cracks accurately. In these comparative studies, the proposed method shows very robust performance in crack detection



(a) Original image (b) Proposed CNN

(c) Canny edge detection (d) Sobel edge detection

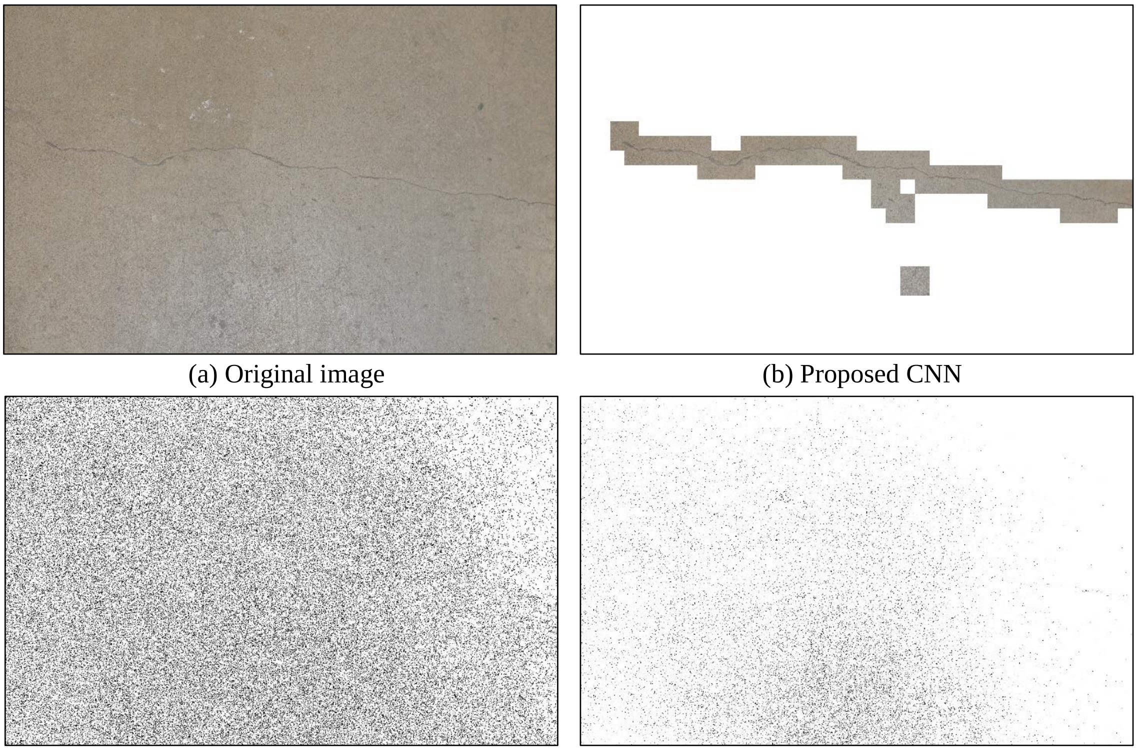
Figure 15 Case 1-normal, uniform lighting



(a) Original image (b) Proposed CNN

(c) Canny edge detection (d) Sobel edge detection

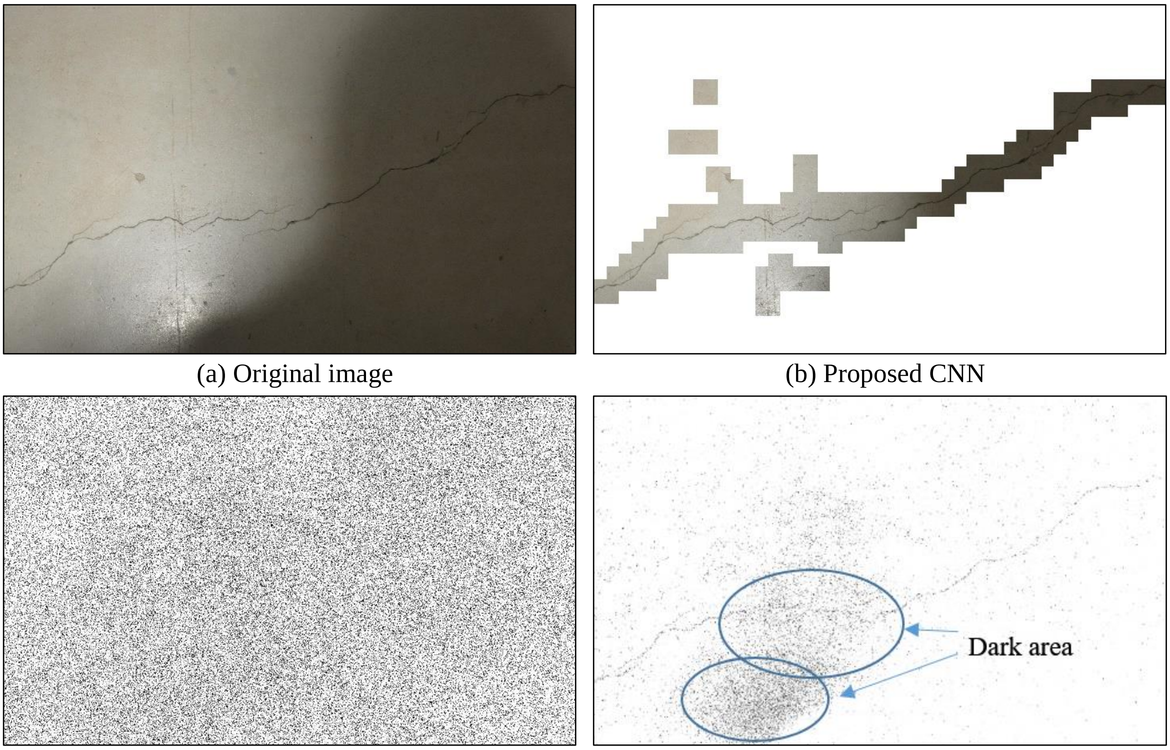
Figure 16 Case 2-normal uniform lighting



(a) Original image (b) Proposed CNN

(c) Canny edge detection (d) Sobel edge detection

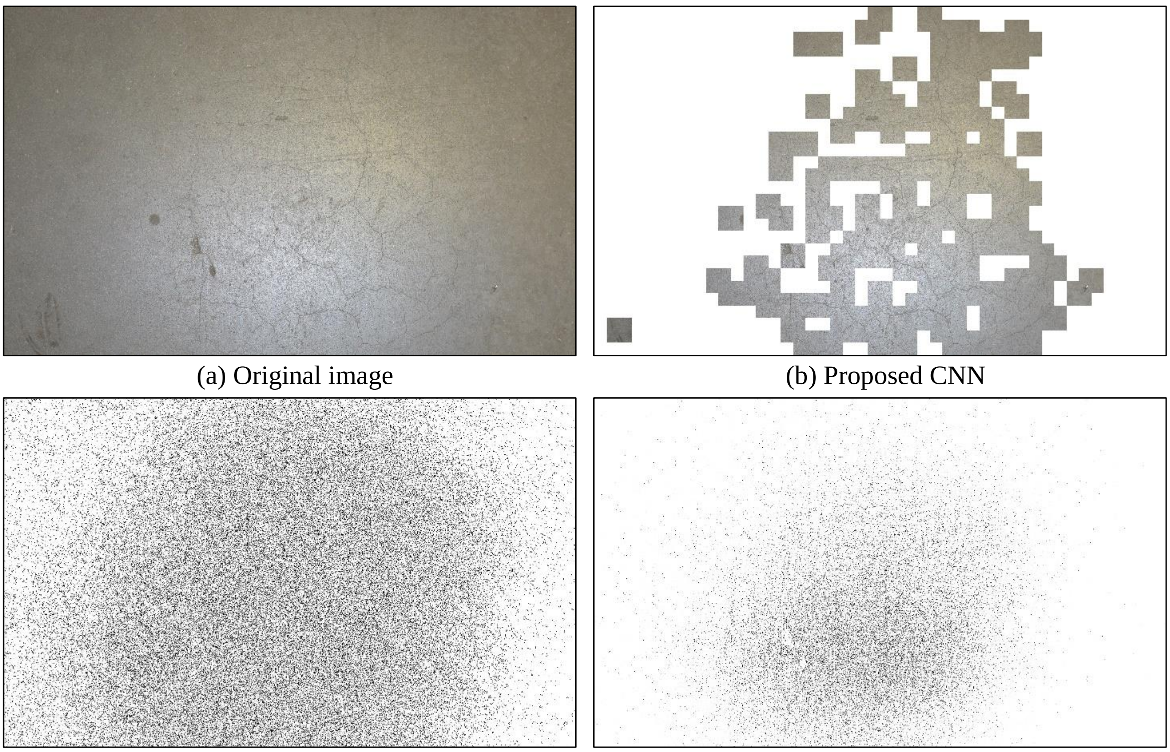
Figure 17 Thin crack case



(a) Original image (b) Proposed CNN

(c) Canny edge detection (d) Sobel edge detection

Figure 18 Shadowed case



(a) Original image (b) Proposed CNN

(c) Canny edge detection (d) Sobel edge detection

Figure 19 Thin crack with lighting

those of the traditional Canny and Sobel methods.

Another primary advantages of using CNNs is that feature extraction techniques are not necessary, as CNN automatically learns features when the network is tuned by SGD. This advantage can save a lot of effort when compared to traditional IPT implementation. For example, suppose that one tries to find cracks on images with lighting spots and shadowed areas. Methods using IPTs may find edges from the verges of lighting spots and shadowed areas rather than crack edges without carefully parameterized methods as shown in the Figure 18 (d). By contrast, CNNs are capable of learning invariant features from a vast amount of images. If certain types of features are not well classified, the only action necessary is to provide the misclassified data and re-train the network. These aspects make CNNs robust in real-world problems. However, the large dataset requirement is also a disadvantage. For example, defects of welded joints and steel cracks are rarely visible and found, so no image repositories of such damage scenarios have been built to date. Therefore, a long-term plan to collect image data is inevitable. A common difficulty of vision-based approaches can also be noted. Extracting internal features, such as the depth of concrete cracks, seems to be impossible, as such features cannot be captured by current technology. In the future, we will build an image repository of damaged civil structures, as well as an advanced classifier that can detect at least five damage types of concrete and steel structures. The classifier will be mounted on autonomously aviating drones that hover around civil structures.

6 CONCLUSIONS

A vision-based approach for detecting cracks on concrete images was proposed using a deep learning method. The concrete images required for the training, validation, and tests were taken with a hand-held camera. A total 332 images were taken and divided into 277 images with 4928×3264 pixel resolutions for training and validation and 55 images with 5888×3584 pixel resolutions for testing. In order to secure a wide range of adaptability, the images were taken under uncontrolled situations. The 277 images were cropped into 40K images with 256×256 pixel resolutions for training and validation processes. The small images were used as the dataset to train the CNN. The trained network recorded accuracies of 98.22% out of 32K images and 97.95% out of 8K images in training and validation, respectively. According to a parametric study, more than 10K images were recommended in order to secure sufficient robustness.

The performance of the trained CNN was evaluated on 55 large images with resolutions of 5888×3584 pixels. The test images were scanned by the trained CNN using a sliding window technique, which facilitated the scanning of any images larger than 256×256 pixel resolutions, and the crack maps were consequently obtained. The test results showed consistent performance although test images taken under various conditions, including strong lighting spot, shadow, blur, and close-up. Moreover, the performances of the proposed method were not susceptible to the quality of images, camera specification, and working distance.

From the comparative studies, which used various conditions with raw images, the proposed CNN method showed very robust performance compared to the traditional, well-known edge detection methods (i.e., Canny and Sobel). The Sobel and Canny edge detection methods provided no meaningful crack information, even though the test images were normal. These methods might not able to treat properly the non-homogeneous concrete surfaces in terms of color and texture. The proposed CNN was especially strong at detecting thin cracks under lighting conditions that make detection difficult when using traditional methods. The proposed method also showed lower levels of noise than the traditional methods and provided raw image results, which allowed for differentiation between noises and errors. As far as the method in general goes, the CNN's ability to learn features from a vast amount of training data is a huge advantage. However, it also means that a CNN implemented method requires a large amount of training data in order to train a robust classifier. One common limitation of almost all vision-based approaches, including the implementations of IPTs and CNNs, is the incapability of sensing internal features due to the nature of photographic images. In the future, the CNN will be developed to detect various types of superficial damage, such as voids, delamination, spalling, and corrosion of concrete and steel structures, to partially replace the biannual visual inspection, is currently the most reliable method for monitoring structure health. This will also be combined with autonomous drones in order to monitor the damage of civil structures.

REFERENCES

Aaoshat (2008), Bridging the gap–restoring and rebuilding the nation's bridges, *Washington (DC): American Association of State Highway and Transportation Officials*.

Abdel-Qader, I., Abudayyeh, O. & Kelly, M. E. (2003), Analysis of edge-detection techniques for crack identification in bridges, *Journal of Computing in Civil Engineering*, **17**(4), 255-63.

Ahmadlou, M. & Adeli, H. (2010), Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integrated Computer-Aided Engineering*, **17**(3), 197-210.

Alaknanda, Anand, R. S. & Kumar, P. (2009), Flaw detection in radiographic weldment images using morphological watershed segmentation technique, *NDT & E International*, **42**(1), 2-8.

Bengio, Y. (2012), Practical Recommendations for Gradient-Based Training of Deep Architectures, In *Neural Networks: Tricks of the Trade:Second Edition*, Springer Berlin Heidelberg, 437-78.

Bengio, Y., Goodfellow, I.J. and Courville, A., Deep learning, An MIT Press book in preparation, online version is available at http://www.deeplearningbook.org.

Bottou, L. (2012), Stochastic gradient descent tricks, In *Neural Networks: Tricks of the Trade:Second Edition*, Springer Berlin Heidelberg, 421-36.

Butcher, J., Day, C., Austin, J., Haycock, P., Verstraeten, D. & Schrauwen, B. (2014), Defect detection in reinforced concrete using random neural architectures, *Computer-Aided Civil and Infrastructure Engineering*, **29**(3), 191-207.

Cha, Y.-J., Chen, J. G. & Buyukozturk, O. (2017), Output-only computer vision based damage detection using phase-based optical flow and unscented Kalman filters. *Engineering Structures*, **132**, 300-313.

Cha, Y.-J. & Buyukozturk, O., (2015). Structural damage detection using modal strain energy and hybrid multiobjective optimization. *Computer-Aided Civil and Infrastructure Engineering*, *30*(5), 347-358.

Cha, Y.-J., You, K. & Choi, W. (2016), Vision-based detection of loosened bolts using the Hough transform and support vector machines, *Automation in Construction*.

Chatzi, E. N., Hiriyur, B., Waisman, H. & Smyth, A. W. (2011), Experimental application and enhancement of the XFEM–GA algorithm for the detection of flaws in structures, *Computers & Structures*, **89**(7), 556-70.

Chen, J. G., Wadhwa, N., Cha, Y.-J., Durand, F., Freeman, W. T. & Buyukozturk, O. (2015), Modal identification of simple structures with high-speed video using motion magnification, *Journal of Sound and Vibration*, **345**, 58-71.

Cifia-10 and Cifar-100 Dataset. available at: https://www.cs.toronto.edu/~kriz/cifar.html, accessed July 20, 2016.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L. & Schmidhuber, J. (2011), Flexible, high performance convolutional neural networks for image classification in *Proceedings of IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 1237.

Cornwell, P., Farrar, C. R., Doebling, S. W. & Sohn, H. (1999), Environmental variability of modal properties, *Experimental Techniques*, **23**(6), 45-8.

Federal Highway Administration. available at: https://www.fhwa.dot.gov/bridge/, accessed March 9, 2016.

Frangi, A. F., Niessen, W. J., Hoogeveen, R. M., Van Walsum, T. & Viergever, M. A. (1999), Model-based quantitation of 3-D magnetic resonance angiographic images, *IEEE Transactions on medical imaging*, **18**(10), 946-56.

Imagenet. available at: http://www.image-net.org/, accessed July 20, 2016.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint arXiv:1502.03167*.

Jahanshahi, M. R., Masri, S. F., Padgett, C. W. & Sukhatme, G. S. (2013), An innovative methodology for detection and quantification of cracks through incorporation of depth perception, *Machine vision and applications*, **24**(2), 227-41.

Jang, S., Jo, H., Cho, S., Mechitov, K., Rice, J. A., Sim, S.-H., Jung, H.-J., Yun, C.-B., Spencer Jr, B. F. & Agha, G. (2010), Structural health monitoring of a cable-stayed bridge using smart sensor technology: deployment and evaluation, *Smart Structures and Systems*, **6**(5-6), 439-59.

Jiang, X. & Adeli, H. (2007), Pseudospectra, MUSIC, and dynamic wavelet neural network for damage detection of highrise buildings, *International Journal for Numerical Methods in Engineering*, **71**(5), 606-29.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, 1097-105.

Kurata, M., Kim, J., Lynch, J., Van Der Linden, G., Sedarat, H., Thometz, E., Hipley, P. & Sheng, L.-H. (2012), Internet-enabled wireless structural monitoring systems: development and permanent deployment at the New Carquinez Suspension Bridge, *Journal of structural engineering*, **139**(10), 1688-702.

Lecun, Y., Bengio, Y. & Hinton, G. (2015), Deep learning, *Nature*, **521**(7553), 436-44.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), Gradient-based learning applied to document recognition in *Proceedings of Proceedings of the IEEE*, 2278-324.

Lecun, Y. A., Bottou, L., Orr, G. B. & Müller, K.-R. (2012), Efficient backprop, In *Neural networks: Tricks of the trade:*Springer, 9-48.

Liu, S. W., Huang, J. H., Sung, J. C. & Lee, C. C. (2002), Detection of cracks using neural networks and computational mechanics, *Computer Methods in Applied Mechanics and Engineering*, **191**(25–26), 2831-45.

Mnist Database. available at: http://yann.lecun.com/exdb/mnist/, accessed July 20, 2016.

Moon, H. & Kim, J. (2011), Intelligent crack detecting algorithm on the concrete crack image using neural network, *Proceedings of the 28th ISARC*, 1461-7.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines in *Proceedings of The 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel, June 21-24, 807-14.

Nishikawa, T., Yoshida, J., Sugiyama, T. & Fujino, Y. (2012), Concrete crack detection by multiple sequential image filtering, *Computer-Aided Civil and Infrastructure Engineering*, **27**(1), 29-47.

O'byrne, M., Ghosh, B., Schoefs, F. & Pakrashi, V. (2014), Regionally enhanced multiphase segmentation technique for damaged surfaces, *Computer-Aided Civil and Infrastructure Engineering*, **29**(9), 644-58.

O'byrne, M., Schoefs, F., Ghosh, B. & Pakrashi, V. (2013), Texture analysis based damage detection of ageing infrastructural elements, *Computer‐ Aided Civil and Infrastructure Engineering*, **28**(3), 162-77.

Rabinovich, D., Givoli, D. & Vigdergauz, S. (2007), XFEM‐based crack detection scheme using a genetic algorithm, *International Journal for Numerical Methods in Engineering*, **71**(9), 1051-80.

Rudin, L. I., Osher, S. & Fatemi, E. (1992), Nonlinear total variation based noise removal algorithms, *Physica D: Nonlinear Phenomena*, **60**(1–4), 259-68.

Scherer, D., Müller, A. & Behnke, S. (2010), Evaluation of pooling operations in convolutional architectures for object recognition, In *Artificial Neural Networks–ICANN 2010:*Springer, 92-101.

Simard, P. Y., Steinkraus, D. & Platt, J. C. (2003), Best practices for convolutional neural networks applied to visual document analysis in *Proceedings of of the Seventh International Conference on Document Analysis and Recognition*, August 3-6, 958-62.

Sinha, S. K. & Fieguth, P. W. (2006), Automated detection of cracks in buried concrete pipe images, *Automation in Construction*, **15**(1), 58-72.

Song, M. & Civco, D. (2004), Road extraction using SVM and image segmentation, *Photogrammetric Engineering & Remote Sensing*, **70**(12), 1365-71.

Soukup, D. & Huber-Mörk, R. (2014), Convolutional Neural Networks for Steel Surface Defect Detection from Photometric Stereo Images in *Proceedings of International Symposium on Visual Computing*, 668-77.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), Dropout: a simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, **15**(1), 1929-58.

Steinkrau, D., Simard, P. Y. & Buck, I. (2005), Using GPUs for machine learning algorithms in *Proceedings of 8th International Conference on Document Analysis and Recognition*, Seoul, Korea, August 29 - September 1, 1115-9.

Teidj, S., Khamlichi, A. & Driouach, A. (2016), Identification of Beam Cracks by Solution of an Inverse Problem, *Procedia Technology*, **22**, 86-93.

Vedaldi, A. & Lenc, K. (2015), Matconvnet: Convolutional neural networks for matlab in *Proceedings of Proceedings of the 23rd ACM international conference on Multimedia*, Brisbane, Australia, October 26-30, 689-92.

Wilson, D. R. & Martinez, T. R. (2001), The need for small learning rates on large problems in *Proceedings of Proceedings of International Joint Conference on Neural Networks*, Washington, DC, USA, July 15-19, 115-9.

Wu, L., Mokhtari, S., Nazef, A., Nam, B. & Yun, H.-B. (2014), Improvement of Crack-Detection Accuracy Using a Novel Crack Defragmentation Technique in Image-Based Road Assessment, *Journal of Computing in Civil Engineering*, **30**(1), 04014118.

Xia, Y., Chen, B., Weng, S., Ni, Y.-Q. & Xu, Y.-L. (2012), Temperature effect on vibration properties of civil structures: a literature review and case studies, *Journal of Civil Structural Health Monitoring*, **2**(1), 29-46.

Yamaguchi, T., Nakamura, S., Saegusa, R. & Hashimoto, S. (2008), Image‐ Based Crack Detection for Real Concrete Surfaces, *IEEJ Transactions on Electrical and Electronic Engineering*, **3**(1), 128-35.

Yeum, C. M. & Dyke, S. J. (2015), Vision‐ Based Automated Crack Detection for Bridge Inspection, *Computer ‐ Aided Civil and Infrastructure Engineering*, **30**(10), 759-70.

Ziou, D. & Tabbone, S. (1998), Edge detection techniques-an overview, *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, **8**, 537-59.

Appendix

Table A1 Summarized results of scanned images

No.	*# of Pos.*[i]	*# of Neg.*[ii]	*# of TP*[iii]	*# of TN*[iv]	*# of FP*[v]	*# of FN*[vi]	*Accuracy*	*Precision*	*Recall*	*F1*	*Remark*
1	126	482	103	473	9	23	0.95	0.92	0.82	0.87	Figure 14(a)
2	162	446	143	438	8	19	0.96	0.95	0.88	0.91	Figure 14(b)
3	55	553	54	538	15	1	0.97	0.78	0.98	0.87	Figure 14(c)
4	37	571	35	566	5	2	0.99	0.88	0.95	0.91	Figure 14(d)
5	58	550	41	550	0	17	0.97	1.00	0.71	0.83	Figure 14(e)
6	45	269	42	266	3	3	0.98	0.93	0.93	0.93	-
7	23	291	23	289	2	0	0.99	0.92	1.00	0.96	-
8	35	279	35	275	4	0	0.99	0.90	1.00	0.95	-
9	31	283	25	283	0	6	0.98	1.00	0.81	0.89	-
10	31	283	29	281	2	2	0.99	0.94	0.94	0.94	-
11	32	282	32	279	3	0	0.99	0.91	1.00	0.96	-
12	30	284	30	277	7	0	0.98	0.81	1.00	0.90	-
13	30	284	30	283	1	0	1.00	0.97	1.00	0.98	-
14	31	283	31	281	2	0	0.99	0.94	1.00	0.97	-
15	31	283	30	253	30	1	0.90	0.50	0.97	0.66	-
16	38	276	32	271	5	6	0.96	0.86	0.84	0.85	-
17	28	286	28	285	1	0	1.00	0.97	1.00	0.98	-
18	34	392	34	389	3	0	0.99	0.92	1.00	0.96	-
19	30	396	30	391	5	0	0.99	0.86	1.00	0.92	-
20	23	403	23	400	3	0	0.99	0.88	1.00	0.94	-
21	36	390	34	376	14	2	0.96	0.71	0.94	0.81	-
22	39	387	38	366	21	1	0.95	0.64	0.97	0.78	-
23	27	399	26	396	3	1	0.99	0.90	0.96	0.93	-
24	27	399	25	391	8	2	0.98	0.76	0.93	0.83	-
25	22	404	22	386	18	0	0.96	0.55	1.00	0.71	-
26	34	392	34	373	19	0	0.96	0.64	1.00	0.78	-
27	33	393	30	377	16	3	0.96	0.65	0.91	0.76	-
28	31	395	31	381	14	0	0.97	0.69	1.00	0.82	-
29	33	393	33	379	14	0	0.97	0.70	1.00	0.83	-
30	30	396	30	395	1	0	1.00	0.97	1.00	0.98	-
31	46	380	45	379	1	2	1.00	0.98	0.96	0.97	-
32	31	316	31	295	21	0	0.94	0.60	1.00	0.75	-
33	49	298	43	298	0	6	0.98	1.00	0.88	0.93	-
34	53	294	49	292	2	4	0.98	0.96	0.92	0.94	-
35	30	317	27	314	3	3	0.98	0.90	0.90	0.90	-
36	26	321	24	310	11	2	0.96	0.69	0.92	0.79	-
37	43	304	36	301	3	7	0.97	0.92	0.84	0.88	-
38	56	291	55	277	14	1	0.96	0.80	0.98	0.88	-
39	48	299	44	290	9	4	0.96	0.83	0.92	0.87	-
40	43	304	42	280	24	1	0.93	0.64	0.98	0.77	-
41	52	295	52	281	14	0	0.96	0.79	1.00	0.88	-
42	57	290	57	266	24	0	0.93	0.70	1.00	0.83	-
43	50	297	50	253	44	0	0.87	0.53	1.00	0.69	-
44	41	306	41	288	18	0	0.95	0.69	1.00	0.82	-
45	69	278	68	262	16	1	0.95	0.81	0.99	0.89	-
46	57	290	57	262	28	0	0.92	0.67	1.00	0.80	-
47	73	274	63	269	5	10	0.96	0.93	0.86	0.89	-
48	24	323	24	322	1	0	1.00	0.96	1.00	0.98	-
49	21	326	19	324	2	2	0.99	0.90	0.90	0.90	-
50	28	319	26	319	0	2	0.99	1.00	0.93	0.96	-

51	55	292	52	284	8	3	0.97	0.87	0.95	0.90	-
52	27	320	23	307	13	4	0.95	0.64	0.85	0.73	-
53	33	314	33	310	4	0	0.99	0.89	1.00	0.94	-
54	31	316	31	295	21	0	0.94	0.60	1.00	0.75	-
55	61	286	61	244	42	0	0.88	0.59	1.00	0.74	-
∑	2326	18774	2186	18210	564	141	0.97	0.79	0.94	0.86	-

Pos.: crack; Neg.: intact; TP: True-positive; TN: True-negative; FN: False-negative; Accuracy: {(iii)+(iv)}/{(i)+(ii)}; Precision: (iii)/{(iii)+(v)}; Recall: (iii)/{(iii)+(vi)}; F1: 2×(precision × recall)/(precision + recall)