Deep Learning–Based Fully Automated Pavement Crack Detection on 3D Asphalt Surfaces with an Improved CrackNet

Allen Zhang[1]; Kelvin C. P. Wang, M.ASCE[2]; Yue Fei[3]; Yang Liu[4]; Siyu Tao[5]; Cheng Chen[6]; Joshua Q. Li[7]; and Baoxian Li[8]

Abstract: CrackNet is the result of an 18-month collaboration within a 10-person team to develop a deep learning–based pavement crack detection software that demonstrated successes in terms of consistency for both precision and bias. This paper proposes an improved architecture of CrackNet called CrackNet II for enhanced learning capability and faster performance. The proposed CrackNet II represents two major modifications on the original CrackNet. First, the feature generator, which provides handcrafted features through fixed and nonlearnable procedures, is no longer used in CrackNet II. Consequently, all layers in CrackNet II have learnable parameters that are tuned during the learning process. Second, CrackNet II has a deeper architecture with more hidden layers but fewer parameters. Such an architecture yields five times faster performance compared with the original CrackNet. Similar to the original CrackNet, CrackNet II still uses invariant image width and height through all layers to place explicit requirements on pixel-perfect accuracy. In addition, the combination of a convolution layer and a 1×1 convolution layer was repeated in CrackNet II to learn local motifs with different sizes of local receptive fields. CrackNet II was trained with 2,500 diverse example images and then demonstrated to outperform the original CrackNet. The experiment using 200 testing images showed that CrackNet II performs generally better than the original CrackNet in terms of both precision and recall. The overall precision, recall, and F-measure achieved by CrackNet II for the 200 testing images were 90.20, 89.06, and 89.62%, respectively. Compared with the original CrackNet, CrackNet II is capable of detecting more fine or hairline cracks, while eliminating more local noises and maintaining much faster processing speed. **DOI: 10.1061/(ASCE)CP.1943-5487.0000775.**

Author keywords: Deep learning; Convolutional neural networks; Pavement crack.

Introduction

Data-driven modeling and decision making are high priorities in modern transportation management (Wang 2000; Adeli and Samant 2000; Adeli and Jiang 2003; Ciresan et al. 2012; Ma and Wang 2014; Qiu et al. 2016; Kouroussis et al. 2016; Ngo et al. 2017). Pavement cracking is a common type of distress on pavement surfaces and is relied upon in both pavement management and design as input data for various modeling purposes. The automated detection of pavement cracks can be more objective and time-efficient compared with manual means, providing a data-driven approach to pavement condition survey. In the past two decades, numerous algorithms were proposed for fully automated pavement crack detection. According to the type of pavement surface data, these algorithms can be divided into two categories. The first category of algorithms detects cracks on two-dimensional (2D) pavement images. A widely used assumption for pavement crack detection on 2D images is that the crack has a lower intensity than the local background. Such an assumption provides the basis for many algorithms, including the thresholding methods (Cheng et al. 2003; Oliveira and Correia 2009), segmentation-based approaches (Kirschke and Velinsky 1992; Huang and Xu 2006; Ying and Salari 2010), filter-based algorithms (Zhang et al. 2013; Zalama et al. 2014), minimal-path methods (Avila et al. 2014; Amhaz et al. 2014), texture-anisotropy approach (Nguyen et al. 2009), and the CrackTree (Zou et al. 2012). Another assumption for detecting cracks on 2D images is that the crack represents a sharp change in intensity. Following this assumption, several crack detection methods were developed with the use of edge detectors (Attoh-Okine and Ayenu-Prah 2008; Santhi et al. 2012; Nisanth and Mathew 2014) or wavelet transforms (Zhou et al. 2006; Wang et al. 2007). The second category of algorithms detects cracks using three-dimensional (3D) pavement surface data. Compared with 2D pavement images, 3D pavement surface data are less vulnerable to lighting conditions and present more useful information as well as fewer noises in terms of crack detection (Wang 2011; Zhang and Wang 2017). For instance, a specific crack on 2D pavement images

[1]Assistant Professor, Dept. of Civil Engineering, Southwest Jiaotong Univ., Chengdu 610031, China; Postdoctoral Fellow, Dept. of Civil and Environmental Engineering, Oklahoma State Univ., 207 Engineering South, Stillwater, OK 74078.

[2]Professor, Dept. of Civil and Environmental Engineering, Oklahoma State Univ., 207 Engineering South, Stillwater, OK 74078; Professor, Dept. of Civil Engineering, Southwest Jiaotong Univ., Chengdu 610031, China.

[3]Ph.D. Student, Dept. of Civil and Environmental Engineering, Oklahoma State Univ., 207 Engineering South, Stillwater, OK 74078.

[4]Ph.D. Student, Dept. of Civil and Environmental Engineering, Oklahoma State Univ., 207 Engineering South, Stillwater, OK 74078.

[5]Assistant Professor, School of Transportation and Logistics, Southwest Jiaotong Univ., Chengdu 610031, China (corresponding author). Email: taosiyu@home.swjtu.edu.cn

[6]Senior Research Engineer, Dept. of Civil and Environmental Engineering, Oklahoma State Univ., 207 Engineering South, Stillwater, OK 74078.

[7]Assistant Professor, Dept. of Civil and Environmental Engineering, Oklahoma State Univ., 207 Engineering South, Stillwater, OK 74078.

[8]Ph.D. Student, Dept. of Civil Engineering, Southwest Jiaotong Univ., Chengdu 610031, China.

Note. This manuscript was submitted on October 20, 2017; approved on March 1, 2018; published online on July 5, 2018. Discussion period open until December 5, 2018; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, © ASCE, ISSN 0887-3801.

may have different intensity values due to varied lighting conditions. On the contrary, the same crack on a 3D pavement surface should have constant depth values if the data acquisition system is properly calibrated. The consistency and repeatability of 3D pavement surface data provide a better opportunity to successfully detect pavement cracks. The depth-checking methods (Jahanshahi et al. 2013; Ouyang and Xu 2013), interactive crack detection algorithm (Zhang et al. 2016a), hybrid crack detection procedures (Sollazzo et al. 2016), and 3D shadow modeling (Zhang et al. 2017a) were all proposed in recent years for detecting cracks on 3D pavement surfaces. The universal methodology of current crack detection algorithms using 3D pavement surface data is to find pixels with lower elevations and subsequently remove noises by postprocessing procedures. In a general view, most of the automated crack detection methods developed in the last two decades were based on simple assumptions and traditional image processing approaches. Although traditional crack detection algorithms have demonstrated certain successes, their limitations are too profound for them to be used in a full production environment when the complexity and diversity of pavement surfaces are fully realized. Current traditional algorithms for pavement crack detection generally lack the capability of learning from diverse examples and self-adjusting for improvements. On the contrary, advanced machine learning algorithms present explicit approaches to data-driven optimization via learning from examples. If diverse examples are selected properly for realistic representations of actual problems, advanced machine learning algorithms potentially can provide better solutions than traditional methods.

In the field of transportation engineering, machine learning techniques were widely adopted in many successful studies, such as traffic incident detection (Adeli and Samant 2000), work zone capacity estimation (Adeli and Jiang 2003), traffic flow forecasting (Jiang and Adeli 2005), and traffic sign classification (Ciresan et al. 2012). There were also pioneering applications of machine learning techniques, such as artificial neural network (ANN) and support vector machine (SVM), in classifying cracks on pavement surfaces (Kaseko and Ritchie 1993; Kaseko et al. 1994; Lee and Lee 2004; Gavilan et al. 2011; Nejad and Zakeri 2011; Daniel and Preeja 2014). Nevertheless, these studies generally represent only one or two layers of abstraction and cannot fully reflect the complexity of pavement surfaces. For SVM-based methods, handcrafted features are normally used for the separation of different classes. However, handcrafted features may be difficult to formulate distinctive decision boundaries for complex nonlinear regression problems. In addition, the SVM is highly dependent on the provided features, and cannot conduct hierarchical processing on the input features. With respect to traditional ANN models, limited numbers of layers and neurons are normally used for hierarchical feature extraction, resulting in shallow abstraction. It was demonstrated in Zhang et al. (2017b) that a five-layer CrackNet significantly outperforms the three-layer CrackNet, implying the outstanding advantage of deep abstraction.

Due to the complexity and diversity of pavement surfaces, pavement crack detection is a challenging task even for human operators. The automation of pavement crack detection generally demands robust algorithms of high-level intelligence. Deep learning has become a popular approach in recent years to learning from examples and understanding complex problems based on increasing levels of abstraction (LeCun et al. 2015; Goodfellow et al. 2016). Particularly, deep convolutional neural networks (CNNs) have demonstrated striking successes in large-scale object recognition problems (Krizhevsky et al. 2012; Zeiler and Fergus 2013; Sermanet et al. 2014; Szegedy et al. 2014; Simonyan and Zisserman 2015; He et al. 2015a, b). Local connection, space invariance, and distortion invariance are the most significant characteristics of CNNs (LeCun et al. 1998, 2015), leading to increasing CNN-based solutions for computer vision problems. However, due to the existence of pooling layers in CNNs, the original data are progressively downsized through increasing levels of abstraction, resulting in loss of the original information. Therefore, most CNNs were to classify small image patches and could not achieve pixel-perfect accuracies. Some CNNs intended to assign a class label to an individual pixel by regarding the local context around that pixel as a whole patch (Tschopp 2015; Maji et al. 2016), and thus may yield imprecise predictions around the target pixels (i.e., the crack pixels) due to overlaps. Other CNNs were also developed for pixelwise classifications (Pinheiro and Collobert 2015; Shelhamer et al. 2016), which all include pooling layers, and thus inevitably lose original data. In the all convolutional net, the pooling layers were replaced by convolution layers with larger strides (Springenberg et al. 2015), resulting in similar spatial reductions and data losses.

Pixel-perfect accuracy is important for pavement crack detection, meaning the visible geometric features of cracks can be available regarding the shape, orientation, length, and width (Zhang et al. 2017b). Therefore, an automated crack survey with a low level of pixel-perfect accuracy will be less useful. Although current CNNs perform excellently in numerous computer vision tasks, it is challenging to apply them in an automated pavement crack survey where pixel-level accuracy becomes a necessity.

Recently, a deep CNN with four convolution layers, four max-pooling layers, and two fully connected layers was developed for pavement crack detection using 2D images (Zhang et al. 2016b). Nevertheless, the class label assigned to an individual pixel was still based on the local context around the pixel, which resulted in overestimation of crack width. Another deep CNN was proposed for piecewise classification of cracks on concrete surfaces (Cha et al. 2017). It was shown in their study that the deep CNN yielded a high accuracy of nearly 98% in classifying small 2D image patches as cracked or intact patches even under complex illuminations. However, the deep CNN proposed in Cha et al. (2017) was incapable of detecting cracks at the pixel level. The CrackNet was proposed for automated crack detection on 3D asphalt pavement surfaces with explicit requirements on pixel-perfect accuracy (Zhang et al. 2017b). CrackNet consists of five layers, including one convolution layer, three 1×1 convolution layers, and one output layer. A feature generator employs predesigned line filters to produce handcrafted features and prepare the inputs for CrackNet. The data depth, or alternatively the number of channels, used at hidden layers of CrackNet is generally 360, resulting in more than 1 million parameters. Unlike traditional CNNs, the CrackNet does not have any pooling layers that downsize the original data. In other words, the spatial size of the data is invariant through all layers. Therefore, the prediction errors can be learned following a pixel-to-pixel manner. It was demonstrated that the CrackNet achieved a consistently high level of pixel-perfect accuracy in detecting various cracks on diverse 3D asphalt pavement surfaces. However, the feature generator used in the original CrackNet conducts fixed operations and has no learnable parameters, resulting in somewhat flawed learning capability.

To improve the learning capability of the original CrackNet, this paper proposes a modification of the original CrackNet. The modified network called CrackNet II in this paper shares three basic ideas rooted in the original CrackNet. First, the image width and height are invariant through all layers such that supervised learning can be conducted at the pixel level. Second, an individual pixel is compared with its local neighbors through local connections provided at convolution layers. Last, the complex difference between a crack pixel and a background pixel is learned by combining multichannel responses at 1×1 convolution layers. However, compared with the original CrackNet, the proposed CrackNet II presents the following

improvements. First, the feature generator of the original CrackNet is no longer used in CrackNet II. The feature generator of the original CrackNet uses predesigned line filters to enhance the contrast between a crack pixel and the local background. Although such fixed operations can yield handcrafted features, they potentially constrain the learning capability of the network. Thus, for improved learning capability, no layers in CrackNet II are deployed to produce handcrafted features. All hidden layers in CrackNet II have learnable parameters and produce learned features. Consequently, CrackNet II can be fully tuned for optimized performance on example data. Second, the proposed CrackNet II has a deeper architecture with more hidden layers but fewer parameters. It is anticipated that CrackNet II would yield better generalizations on the complexity of pavement crack detection with more hidden layers. In the meantime, the data depth at hidden layers of CrackNet II is reduced, which results in reduced parameters and increased processing speed.

Data Preparation

Data Collection and Image Library

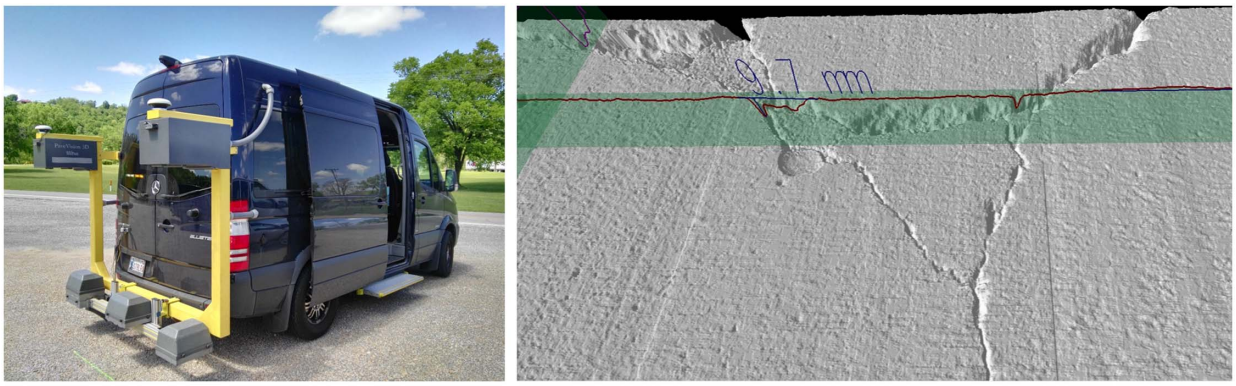
The asphalt pavement surface data used in the paper are 1-mm 3D data from the the PaveVison3D system. The PaveVision3D system is illustrated in Fig. 1 with a Digital Highway Data Vehicle (DHDV) made by WayLink (Stillwater, Oklahoma). The two 3D sensors are in the back of the DHDV on the basis of the triangulation principle. The line laser mounted in the PaveVision3D system projects a line pattern of laser beams onto the pavement surface and provides consistent illumination no matter whether it is daytime or nighttime to minimize sunlight effects. The DHDV can scan the pavement surface at the data collection speed of 100 km/h (approximately 60 mi/h) with full coverage for a 4-m-wide lane at a 30-kHz scanning rate to ensure 1-mm resolution at highway speed.

For machine learning purposes, the research team of this paper built an image library specifically for CrackNet II development that consists of more than 6,000 3D pavement images and corresponding ground-truth images with labeled pavement cracks. Each image covers an area of 4 m (width) by 2 m (length). The 6,000 3D pavement images were collected by the PaveVision3D system in the last 5 years on different pavements and at different collection speeds ranging from 35 to 100 km/h (20 to 60 mi/h). The established image library represents diversified variations of cracks and pavement surface textures. There is no overlap between any two images, and no more than 100 images are from the same pavement section. In addition, the 6,000 3D pavement images include types of cracks with various severity levels defined in the Long-Term Pavement Performance Program (LTPP) protocol (Miller and Bellinger 2014). The ground truths of cracks on all images were manually processed with close supervision on pixel-perfect accuracies by multiple teams. A three-round inspection was conducted to ensure the ground truths were accurate at the pixel level. For the first round, several well-trained operators manually marked the cracks on the provided 3D pavement images with full resolution. For the second round, several other well-trained operators examined and refined the ground truths for correcting errors and reducing subjectivity. Finally, the ground truths were further inspected and verified by experts. The marking error in terms of crack widths and lengths was restricted within ± 2 mm. The continuity of cracks was carefully examined. If the discontinued parts of a crack were all no longer than 5 mm, it was marked as a continuous crack for the entirety. Otherwise, it was considered as separated cracks to ensure that no marked crack had a discontinued part that was longer than 5 mm. The entire process of preparing ground truths of cracks was completed on continuing basis for more than 1 year. Fig. 2 illustrates several representative 3D pavement images and corresponding ground truths of pavement cracks from the established image library.

Three thousand asphalt surface images were selected from the image library for the training and testing of CrackNet II. The 3,000 asphalt surface images stand for varying textures and different mix types, including hot mix asphalt (HMA) and warm mix asphalt (WMA). Particularly, 200 representative images were carefully selected as testing data. In addition, another set of 300 images was selected as validation data. For a comparison study, the 200 testing images and 300 validation images used in the paper were identical to those used for the original CrackNet (Zhang et al. 2017b). The remaining 2,500 images were used as training data.

Data Preprocessing

Two procedures were implemented during data preprocessing: downsampling and surface flattening. The original 3D image from the image library had a fixed size $4{,}096 \times 2{,}048$. If such a big image size is used, there will be up to 70 billion floating-point operations at each hidden layer, which could not be realistically implemented with a high-end desktop computer even with a dedicated high-performance graphics processing unit (GPU). To reduce the computational overhead, the original 3D image was downsized to a $1{,}024 \times 512$ 3D image using the min-pooling technique. A pooling function produces an output at a certain location based on the summary statistic of nearby inputs (Goodfellow et al. 2016). The min-pooling method thereby outputs the minimal value of



WayLink DHDV 1-mm 3D Data

Fig. 1. Representative 1-mm 3D surface data collected by WayLink DHDV.

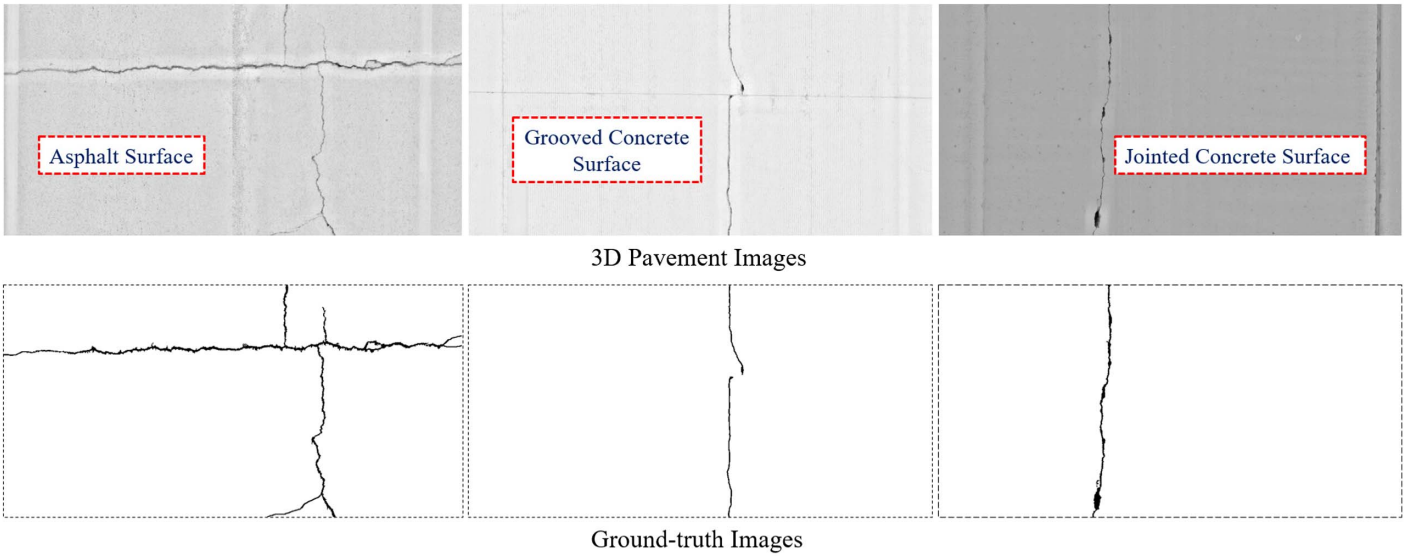


Fig. 2. Representative 3D pavement images and ground-truth images from the image library.

nearby values at a certain location. In this paper, the elevation value of an individual pixel of the downsized 3D image corresponds to the minimal elevation value of a 4×4 block of the original 3D image. Given that the crack pixel has a lower elevation compared with the local surroundings, the min-pooling method is more likely to make fine or hairline cracks visible in the downsized data sets. For instance, if a 1-pixel-wide crack is completely lower than the local surroundings, it should be theoretically maintained on the downsized 3D image through min-pooling. In this paper, all cracks marked on the 3,000 original images were verified to be recognizable on the 3,000 downsized images. It will be demonstrated that CrackNet II can detect a large percentage of cracks (>89%) on downsized images, implying the downsampling procedure does not have a significant impact on the detection accuracy.

A 3D pavement surface may not be flat due to cross slope, pavement rutting, or other surface deformations. Because shared weights are used at convolution layers to compute the weighted sum of local values, similar cracks at different locations may contribute to completely different responses when the 3D surface is uneven. In other words, it will be more difficult for the network to learn local features on uneven surfaces. Therefore, it is necessary to suppress the global unevenness of the pavement surface for enhanced space invariance and local motifs. Mask filtering was proposed in Wang et al. (2014) to remove the global unevenness of longitudinal pavement profiles. However, the mask filtering method was initially developed for one-dimensional data (i.e., pavement profiles). Additionally, it also eliminates some high-frequency signals that may be relevant to hairline cracks. In this paper, the original 3D surfaces were flattened based on elevation differences evaluated at local two-dimensional areas

$$g(x,y) = g(x,y) - \frac{1}{(2k+1)^2}\sum_{i=-k}^{k}\sum_{j=-k}^{k} g(x+i, y+j) \quad (1)$$

where $g(x,y)$ = elevation value at pixel (x,y); k = positive integer; and $(2k+1) \times (2k+1)$ defines the local window size.

As shown in Eq. (1), the modified elevation of a 3D pixel is obtained by subtracting the local mean from the original elevation. The local mean elevation is used to represent the elevation of a local flat plane. Subsequently, subtracting the local mean elevations at all locations is equivalent to placing all local planes at the same elevation, which eliminates the global unevenness automatically. As illustrated in Fig. 3, an uneven 3D surface ($1{,}024 \times 512$) with pavement ruts is flattened by sliding a local window of size 25×25 and repeating the operation defined in Eq. (1). In Fig. 3, after surface flattening, the rutting depths at both sides are reduced from roughly 20 to 2 mm, leading to a desirable level of global evenness of the pavement surface.

Architecture of CrackNet II

Fig. 4 illustrates the architecture of CrackNet II. In general, the proposed CrackNet II consists of 10 layers, including four convolution layers, five 1×1 convolution layers, and one output layer. Identical image width and height are used through all layers to ensure that the error at each pixel is explicitly learned. In addition, the data depth, or the number of channels, is fixed as 90 at most of the hidden layers. In this paper, the data depth at the hidden layers was determined for the trade-off between network speed and model capacity. Although a larger data depth results in higher model capacity, it causes slowness of the network and potentially introduces extra difficulties in training the network. The data depth of 90 was used in the paper to control the processing speed to be nearly one image per second. It will be demonstrated in the paper that such a data depth also yields sufficient model capacity and a high level of accuracy. The input of CrackNet II is a downsampled and flattened 3D image of size $1{,}024 \times 512$. Convolution Layers I, II, III, and IV conduct general-purpose convolutions using 3×3, 5×5, 5×5, and 7×7 filters, respectively. The filter sizes are fixed in such order for three reasons. First, recent state-of-the-art CNNs all use small filters whose sizes are mostly no greater than 7×7 (Krizhevsky et al. 2012; Zeiler and Fergus 2013; Sermanet et al. 2014; Szegedy et al. 2014; Simonyan and Zisserman 2015; He et al. 2015a). Second, increasing filter sizes progressively helps deeper layers to behave more confidently with larger context. Last, it will be shown that such organization of filter sizes yields optimal performance compared with other combinations of filter sizes.

There are 90 filters at each convolution layer to yield 90 output images. Each output image is associated with an individual filter. The 1×1 Convolution Layers I, II, III, and IV provide full connections to previous layers following a pixel-independent manner.

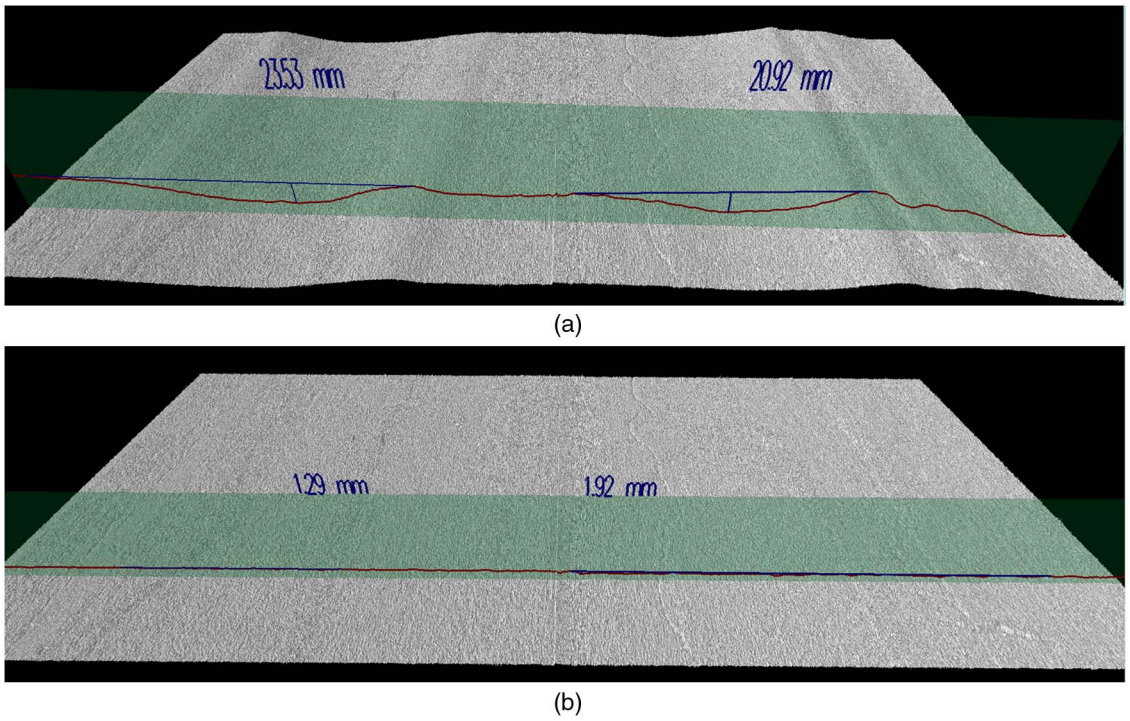


Fig. 3. 3D surface flattening: (a) before 3D surface flattening; and (b) after 3D surface flattening.

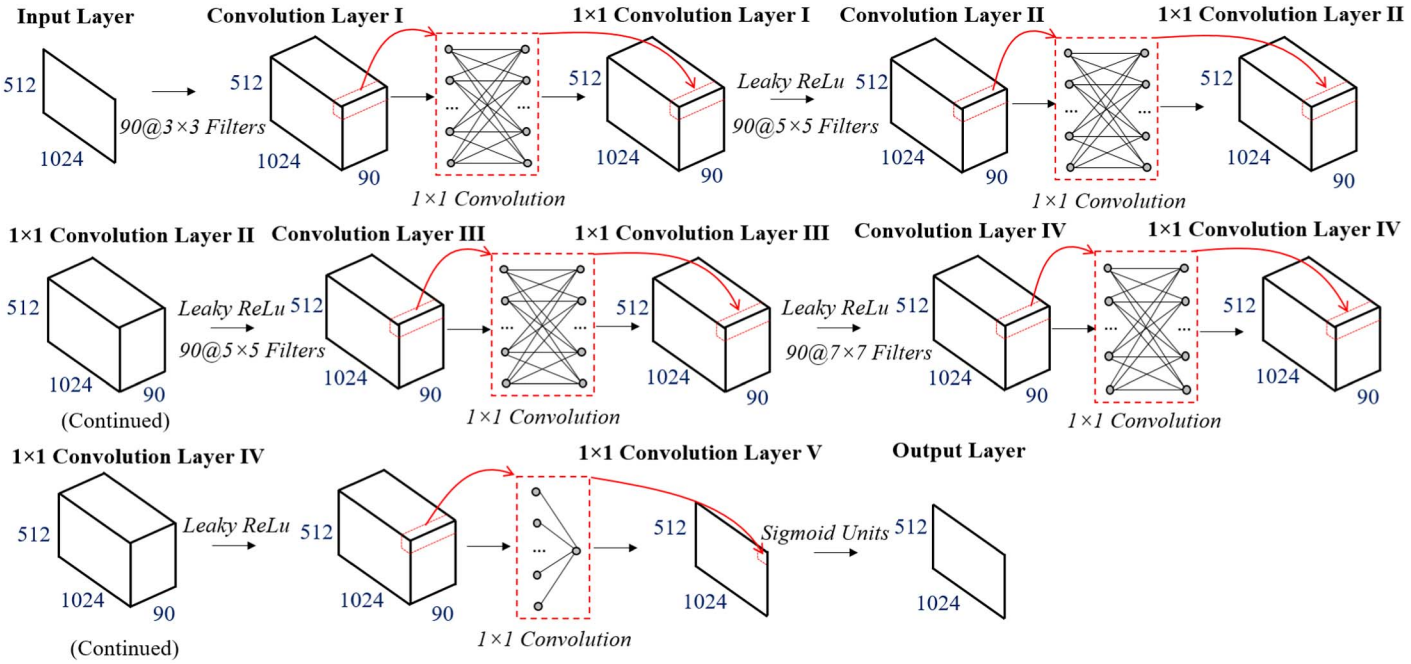


Fig. 4. Architecture of CrackNet II.

Thus, the connections are established across the channels at each pixel, and no two pixels are connected at each channel. Such connections can force the network to integrate or mutate the multichannel responses at each pixel. The 1×1 Convolution Layers I, II, III, and IV do not reduce or increase data dimensionality. Therefore, the data depths at 1×1 Convolution Layers I, II, III, and IV are all 90. The convolution layers and 1×1 convolution layers all conduct linear transforms on the input data. Nonlinear transforms are needed for complex nonlinear regression cases. In recent years, rectified linear unit (ReLu) has become the most widely used activation function to implement nonlinear transforms (Goodfellow et al. 2016). However, ReLu can cause many dead neurons with zero gradients during back-propagation. Consequently, the learning becomes slower as many neurons stop responding to the back-propagated errors (Maas et al. 2013). Leaky ReLu, which is a variation of ReLu, solves the dying neuron problem by activating neurons all the time. Leaky ReLu fulfills nonlinear transform through different activations on negative and positive inputs

(Maas et al. 2013). In this paper, leaky ReLu was applied after each one of the 1×1 Convolution Layers I–IV to perform nonlinear transforms. The 1×1 Convolution Layer V operates cross-channel parametric pooling and integrates multiple channels into a single channel. In other words, the output of 1×1 Convolution Layer V is a single image of size $1{,}024 \times 512$. To address space invariance, shared weights are used at each convolution layer or 1×1 convolution layer. That is, identical weights are used at each pixel of an individual channel. Finally, the output layer uses sigmoid unit to convert the output of 1×1 Convolution Layer V to class scores whose values are between 0 and 1. Thus, the final output image contains predicted class scores for all individual pixels. Because pixel-level crack detection is essentially a binary classification task, class scores are evenly divided to correspond to two different classes. Class score 0.5 thereby becomes the boundary to separate the two different classes. In this paper, class scores no smaller than 0.5 signified that the corresponding pixels were crack pixels, while class scores smaller than 0.5 indicated the corresponding pixels were noncrack pixels.

The architecture of CrackNet II is different from fully convolutional networks (FCNs). CrackNet II does not have any pooling layers, while FCNs normally have several pooling layers that reduce the spatial size successively (Long et al. 2014; Yousefhussien et al. 2017). CrackNet II does not require deconvolution layers introduced in Long et al. (2014) to upsample coarse and downsized outputs to pixelwise predictions. It is believed that pooling layers inevitably lose original information and affect pixel-perfect accuracy. Table 1 shows the number of parameters of CrackNet II. Compared with the original CrackNet, which has about 1 million parameters, the proposed CrackNet II has fewer parameters, leading to potentially fewer computations and faster performance. As shown in Fig. 4, the data width and height are invariant through all layers. Thus, the errors learned at the output layer can be back-propagated to all previous layers following a pixel-to-pixel manner. The objective of convolution layers is to learn local motifs through local connections. In the meantime, the 1×1 convolution layers are to integrate, mutate, and interpret multichannel responses such that cracks and noncrack patterns can be more separable.

Training

Cost Function

The cost function defines the training objective. For the output layer, the sigmoid units produce predicted values between 0 and 1 at all individual pixels. In this paper, the target values for a background pixel and a crack pixel were 0 and 1, respectively. Given the ground truth of cracks, the target values for all pixels can be determined directly. Subsequently, the cross entropy was adopted as the cost function to measure the similarity between the predicted values and target values (Goodfellow et al. 2016; Nielsen 2017)

$$C = -\sum_{i=1}^{n}[y_i \ln a_i + (1 - y_i)\ln(1 - a_i)] \tag{2}$$

where y_i = target value at the ith pixel; a_i = predicted value at the ith pixel; and n = number of pixels of the image.

There are two important reasons to use cross entropy as the cost function. First, it avoids the learning slowness caused by the saturation of sigmoid units. Second, it drives the network to learn at a rate controlled by the similarity between the predicted values and the target values. In other words, the network can learn faster when the errors are larger.

Learning Method

The objective of training is to minimize the cost function using gradient-based optimization methodologies. The gradients are computed through back-propagation. Minibatch gradient descent, which is a variation of stochastic gradient descent, was implemented in this paper to update parameters at each iteration. In other words, the average gradients over a minibatch of examples were used to tune parameters. In addition, the momentum method was applied to reduce the oscillation of gradients. Finally, the regularization of parameters was performed through weight decay that encourages small parameters. In short, the learning method for each iteration in this paper was similar to the method used in Krizhevsky et al. (2012) and can be expressed as

$$\Delta \mathbf{W}_{i+1} = 0.9 \cdot \Delta \mathbf{W}_i - \varepsilon \cdot \left(\frac{\partial \mathbf{L}}{\partial \mathbf{w}}\bigg|\mathbf{w}_i\right)_{\mathbf{B}_i}$$
$$\mathbf{w}_{i+1} = (1 - \lambda)\mathbf{w}_i + \Delta \mathbf{W}_{i+1} \tag{3}$$

where i = iteration index; $\mathbf{w}_i$ = weights learned at the ith iteration; $\Delta \mathbf{W}_i$ = momentum variable at the ith iteration; ε = learning rate; $[(\partial \mathbf{L}/\partial \mathbf{w})|\mathbf{w}_i]_{\mathbf{B}_i}$ = average gradient over the ith batch $\mathbf{B}_i$ evaluated at $\mathbf{w}_i$; and λ = weight decay coefficient, which is fixed as 0.0003 in this paper.

Initialization, Regularization, and Normalization

There are many difficulties in training a deep network. Poor designs on network training can result in various notorious failures, including divergence, overfitting, and vanishing gradient problems. In practice, a deep network is more likely to be trained successfully with careful initialization, effective regularization, and well-designed normalization (Glorot and Bengio 2010; LeCun et al. 2012; Krizhevsky et al. 2012; Ioffe and Szegede 2015; Goodfellow et al. 2016).

Because a deep network normally has a large number of parameters, the initialization of parameters is important. An effective approach to parameter initialization is to keep similar variances of activation values and back-propagated gradients across all layers (Glorot and Bengio 2010; LeCun et al. 2012). In the paper, the normalized initialization proposed in Glorot and Bengio (2010) was adopted to initialize parameters such that the variances of activation values and back-propagated gradients could be maintained efficiently.

The dropout technique is widely used to reduce the risk of overfitting (Hinton et al. 2012). It randomly omits a hidden neuron with a probability of 0.5. The omitted hidden neuron has zero output values, and thus will not be activated in both forward and backward passes. The dropout technique prevents complex coadaptions of neurons and forces neurons to be more independent instead of

Table 1. Number of parameters in CrackNet II

Layer	Number of parameters
Convolution Layer I	$90 \times 3 \times 3 = 810$
Convolution Layer II	$90 \times 5 \times 5 = 2{,}250$
Convolution Layer III	$90 \times 5 \times 5 = 2{,}250$
Convolution Layer IV	$90 \times 7 \times 7 = 4{,}410$
1×1 Convolution Layer I	$90 \times 91 = 8{,}190$ (including biases)
1×1 Convolution Layer II	$90 \times 91 = 8{,}190$ (including biases)
1×1 Convolution Layer III	$90 \times 91 = 8{,}190$ (including biases)
1×1 Convolution Layer IV	$90 \times 91 = 8{,}190$ (including biases)
1×1 Convolution Layer V	91 (including the bias)
Total	42,571

relying on other neurons. The dropout also can potentially improve the network performance through structural regularization. In this paper, the dropout technique was applied at 1×1 Convolution Layers I–IV, where full connections might not be needed.

Recently, batch normalization has become a popular method to normalize the inputs of hidden layers (Ioffe and Szegede 2015). Batch normalization was proposed to solve the internal covariate shift problem occurring when the distribution of each layer's inputs changes during training. It normalizes the inputs of each hidden layer and drives them to have a fixed distribution over a minibatch. Consequently, the covariate shift problem is reduced with nearly fixed distributions, and the network can then learn errors more efficiently. It was demonstrated that batch normalization accelerates the learning process, allows larger learning rates, prevents the network from getting stuck in the saturated regimes of nonlinearities, and regularizes the network by confining the distributions over a minibatch of examples (Ioffe and Szegede 2015). Batch normalization is normally performed before nonlinear activations (e.g., sigmoid or leaky ReLu unit). Therefore, in this paper the batch normalization was implemented at 1×1 Convolution Layers I, II, III, and V, right before the leaky ReLu or sigmoid activation function. The parameters introduced by batch normalization are not counted in Table 1.

Network Speed

Similar to the original CrackNet, CrackNet II is compatible with massively parallel computing. Except the batch normalization procedures, all other tasks involved in both forward and backward passes are accelerated with the aid of a general-purpose GPU. The batch normalization needs to compute the mean and variance of each feature map. Thus, it is difficult to perform batch normalization in a massively parallel manner. However, because the computational task on each feature map for batch normalization is independent, the batch normalization is completed through hyperthreading on multiple CPU cores. In other words, each CPU thread is associated with several feature maps and executes batch normalization on each feature map sequentially. Table 2 lists the speed of CrackNet II in comparison with the speed of the original CrackNet. It is shown that CrackNet II is nearly five times faster than the original CrackNet, indicating a significant improvement in time efficiency.

Hyperparameters

In this paper, the size of the minibatch and the learning rate were manually adjusted through iterations. In particular, smaller batch sizes and larger learning rates were applied in the beginning for faster convergence. Later on, larger batch sizes and smaller learning rates were used for fine tuning. Table 3 shows the minibatch sizes and learning rates used during training.

Table 2. Speeds of CrackNet and CrackNet II

Network	Hardware devices	Forward time (s)	Backward time (s)
CrackNet	GPU: two GeForce GTX TITAN Black (NVIDIA, Santa Clara, California)	5.37	18.51
CrackNet II	CPU: six CPU cores/12 logical processors (Intel i7-3930K, Intel, Santa Clara, California) RAM: 16 GB	1.26	3.53

Table 3. Minibatch sizes and learning rates used during training

	Iteration number				
Parameter	1–200	201–400	401–600	601–800	801–1,000
Minibatch size	10	25	30	35	40
Learning rate	0.05	0.025	0.00625	0.001	0.0005

Training Result

With 2,500 sets of example data, the training was completed after 1,000 iterations using two NVIDIA (Santa Clara, California) GeForce GTX TITAN Black cards in the same computer. During the training, the 300 validation images die not participate in the learning process, and were only used to evaluate the network performance and inspect if the overfitting problem occurred. For time efficiency, the network performances on the 300 validation images were evaluated at every 50 iterations instead of each iteration. In the meantime, the learned parameters were saved at every 50 iterations for final selection of optimal parameters that yield the best performances on the 300 validation images.

Precision and recall are two commonly used indicators for evaluating crack detecting algorithms (Fawcett 2006; Zhang et al. 2016a, 2017a). Precision refers to the percentage of crack pixels classified correctly with respect to all detected pixels, while recall represents the percentage of crack pixels classified correctly with respect to all true crack pixels. From another perspective, the precision can be regarded as the cost, while recall stands for the benefit. Precision and recall frequently conflict with each other. For instance, a high level of sensitivity may yield high recall but low precision. On the other hand, a low level of sensitivity could result in high precision but low recall. Therefore, it is challenging to achieve high precision and high recall simultaneously. The F-measure is the harmonic mean of precision and recall (Fawcett 2006), reflecting the accuracy of an algorithm more appropriately. A high F-measure can be achieved only when the precision and recall are both high.

Fig. 5 compares CrackNet II with the original CrackNet in terms of cost function values and their performances on the 300 validation images. Figs. 5(a and b) show average cost function values evaluated for minibatches of training examples and 300 validation images, respectively. Figs. 5(c and d) show overall F-measures evaluated for minibatches of training examples and 300 validation images, respectively. The original CrackNet was trained with 700 iterations (Zhang et al. 2017b). It can be observed that the original CrackNet converges more quickly than CrackNet II for the first 200 iterations. One possible reason is that the feature generator used in the original CrackNet has already enhanced the contrast between the crack and the background, providing more sensible features in the beginning. However, CrackNet II starts to perform better than the original CrackNet after 200 iterations. The maximum overall F-measure on validation data achieved by CrackNet II is 91.43%, which is higher than the maximum overall F-measure of 89.54% produced by the original CrackNet. Because the maximum overall F-measure is observed at the 850th iteration, the parameters saved at the 850th iteration are considered as the optimal weights of CrackNet II. It can be observed that the overall F-measure on validation data is progressively improved through iterations, indicating that the overfitting problem is avoided successfully. The overall performances of the original CrackNet and CrackNet II on all training images are reported in Table 4. Although the training data sets are slightly different, CrackNet II potentially yields more accurate outputs.

Three alternative networks were also trained in conjunction with CrackNet II for comparison study. The three alternative networks had the same architecture as CrackNet II, but use filters of different

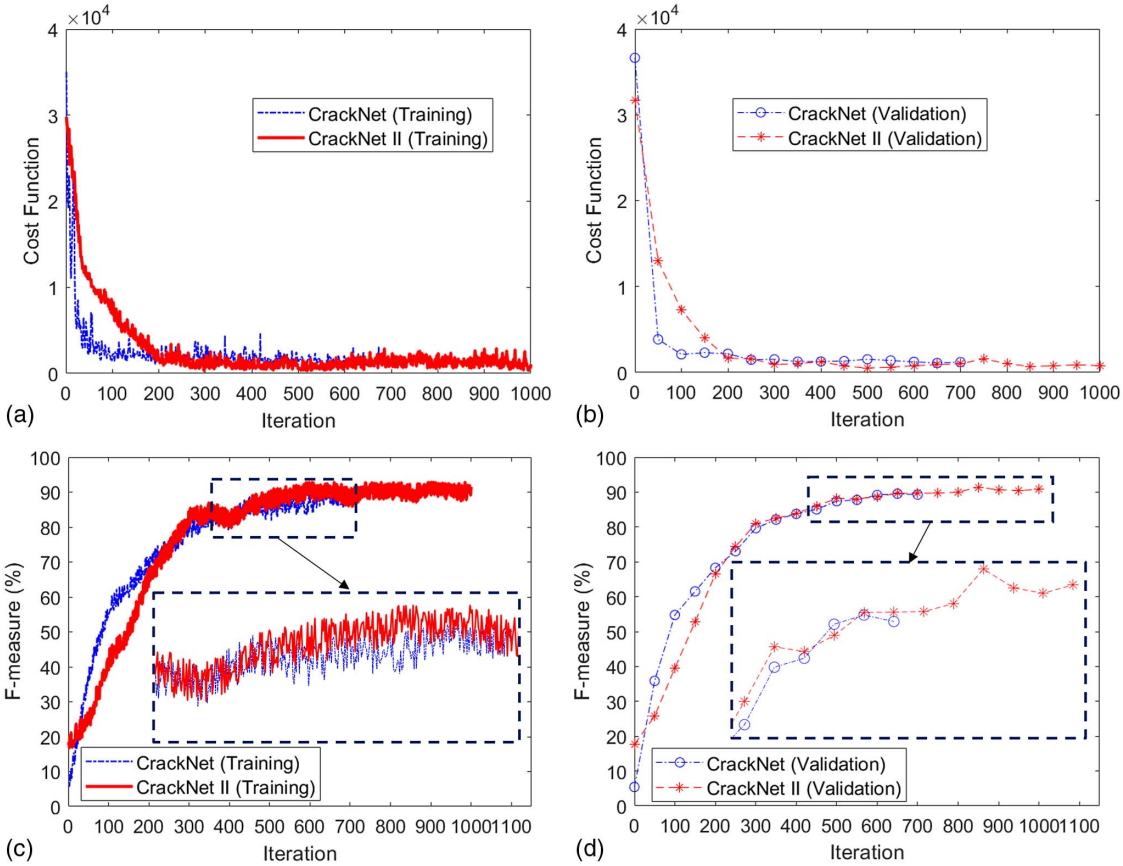


Fig. 5. Training progress.

Table 4. Overall performances on training data

Network	Number of training images	Precision (%)	Recall (%)	F-measure (%)
CrackNet	1,800	89.15	87.16	88.14
CrackNet II	2,500	89.91	91.84	90.86

Table 5. Filters used in the three alternative networks

Network	Convolution Layer I	Convolution Layer II	Convolution Layer III	Convolution Layer IV
CrackNet II-A	90 @ 3×3	90 @ 5×5	90 @ 7×7	90 @ 9×9
CrackNet II-B	90 @ 3×3	90 @ 7×7	90 @ 9×9	90 @ 15×15
CrackNet II-C	90 @ 3×3	90 @ 7×7	90 @ 15×15	90 @ 25×25

sizes at each convolution layer. Table 5 specifies the filter sizes used for the three alternative networks. The three alternative networks were trained the same way with CrackNet II through 1,000 iterations. Fig. 6 shows the evolution of overall F-measures on validation data through 1,000 iterations. It was found that CrackNet II-A is slightly worse than CrackNet II. In addition, CrackNet II-B and CrackNet II-C yield significantly lower accuracies compared with CrackNet II and CrackNet II-A. It seems that increasing local receptive field (filter size) results in worse performances. One of the reasons could be that the local motifs are continuously attenuated with larger receptive fields. Another reason might be that the network using large filters is more difficult to be trained. Such a finding conforms to the tendency to use small filters at convolution layers (Krizhevsky et al. 2012; Zeiler and Fergus 2013; Sermanet et al. 2014; Szegedy et al. 2014; Simonyan and Zisserman 2015; He et al. 2015a). CrackNet II using small but adequate local receptive fields is demonstrated in the paper to produce the highest overall F-measures on validation data.

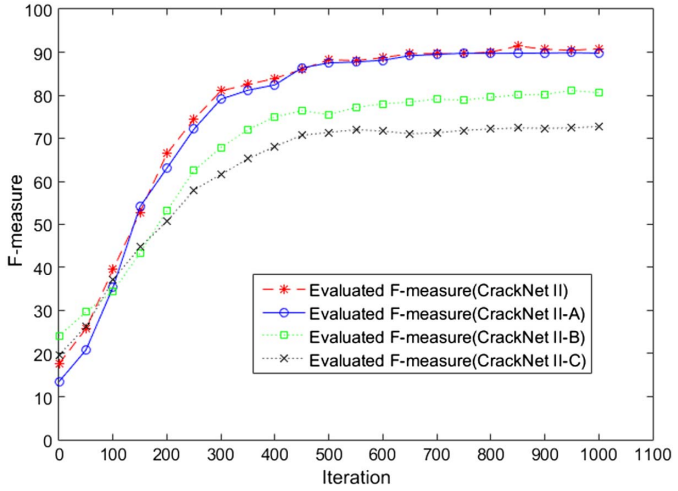


Fig. 6. Performances of the four networks on validation data.

Testing and Evaluation

The trained CrackNet II was applied to the 200 testing images for further validation. The precisions, recalls, and F-measures achieved

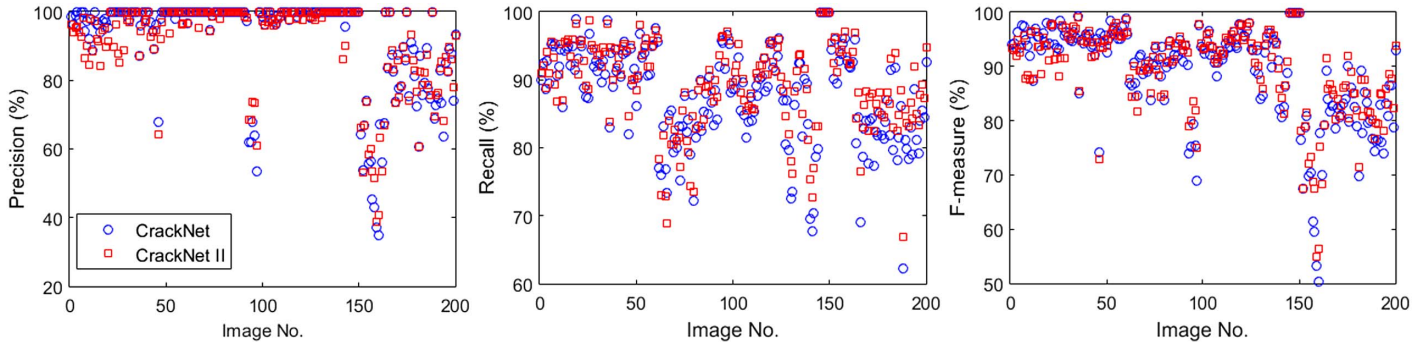


Fig. 7. Precisions, recalls, and F-measures of CrackNet and CrackNet II on testing images.

Table 6. Overall precisions, recalls, and F-measures of CrackNet and CrackNet II on testing images

Network	Precision (%)	Recall (%)	F-measure (%)
CrackNet	90.13	87.63	88.86
CrackNet II	90.20	89.06	89.62

by CrackNet and CrackNet II for all 200 testing images are illustrated in Fig. 7. In addition, Table 6 shows the overall precisions, recalls, and F-measures yielded by CrackNet and CrackNet II.

According to Fig. 7 and Table 6, CrackNet II performs generally better than the original CrackNet in terms of both precision and recall. Particularly, CrackNet II yields tangibly higher recall, while increasing the precision slightly. The improvement in terms of F-measure is nearly 1%. Although the improved values seem small, they contribute to noticeable improvements on the detection outputs. Figs. 8 and 9 show the performances of CrackNet and CrackNet II on several representative testing images. The false-positive errors are highlighted in the small dashed rectangles, while the false-negative errors are highlighted in the small dashed circles. It is demonstrated in Figs. 8 and 9 that CrackNet II is more robust in eliminating local noises as well as detecting fine or hairline cracks.

Discussion

One of the most important concepts of CNNs is to detect local motifs via local receptive fields (Lecun et al. 1998, 2015). The sizes of local receptive fields are directly associated to the filter sizes used at convolution layers. Numerous successes were reported with the use of small filters no greater than 11×11 (Krizhevsky et al. 2012; Zeiler and Fergus 2013; Sermanet et al. 2014; Szegedy et al. 2014; Simonyan and Zisserman 2015; He et al. 2015a). It is also demonstrated in Fig. 6 that CrackNet II using small filters yields better performances. However, as illustrated in Fig. 10, some noise patterns on pavement surfaces are extremely similar to pavement cracks if they are viewed via small local receptive fields. Please ignore the depth differences shown in Fig. 10 because it is highly possible that a noise pattern has a bigger depth compared with pavement cracks. Global context should be needed to completely separate complex noise patterns from pavement cracks. Unfortunately, CrackNet II has limited capability of acquiring global

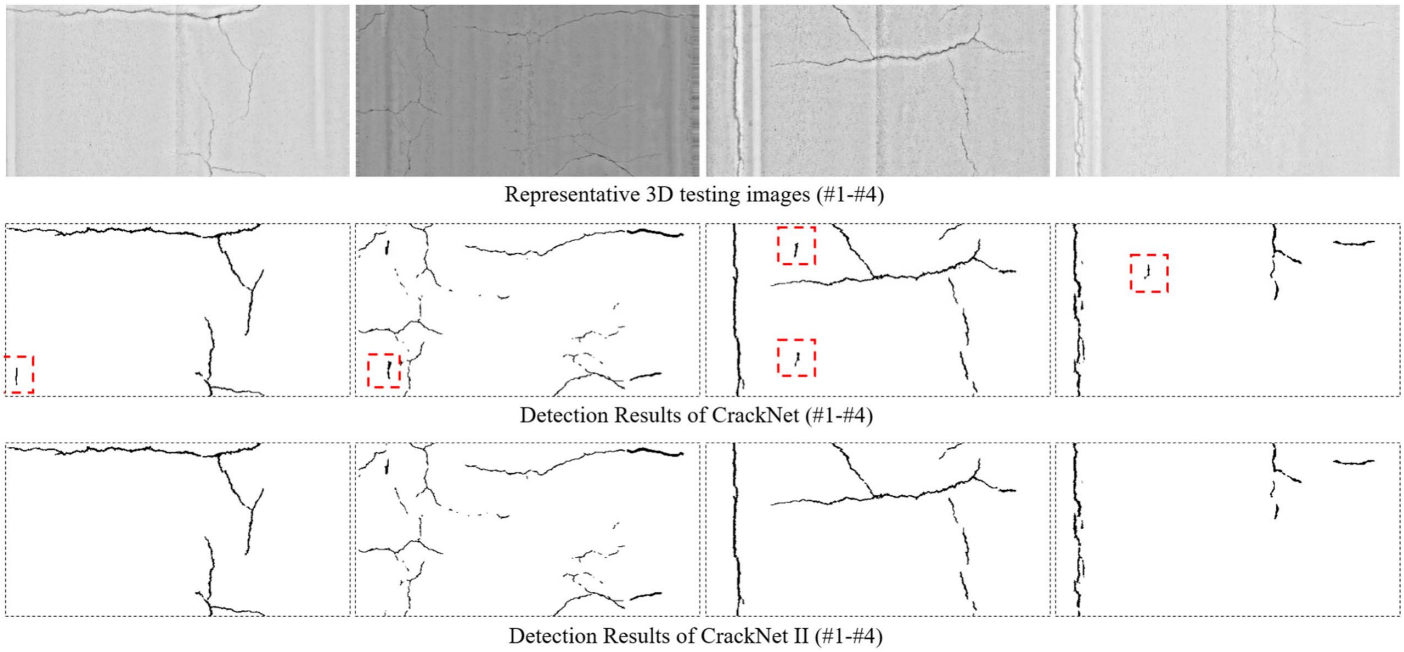


Fig. 8. False-positive errors reduced by CrackNet II.

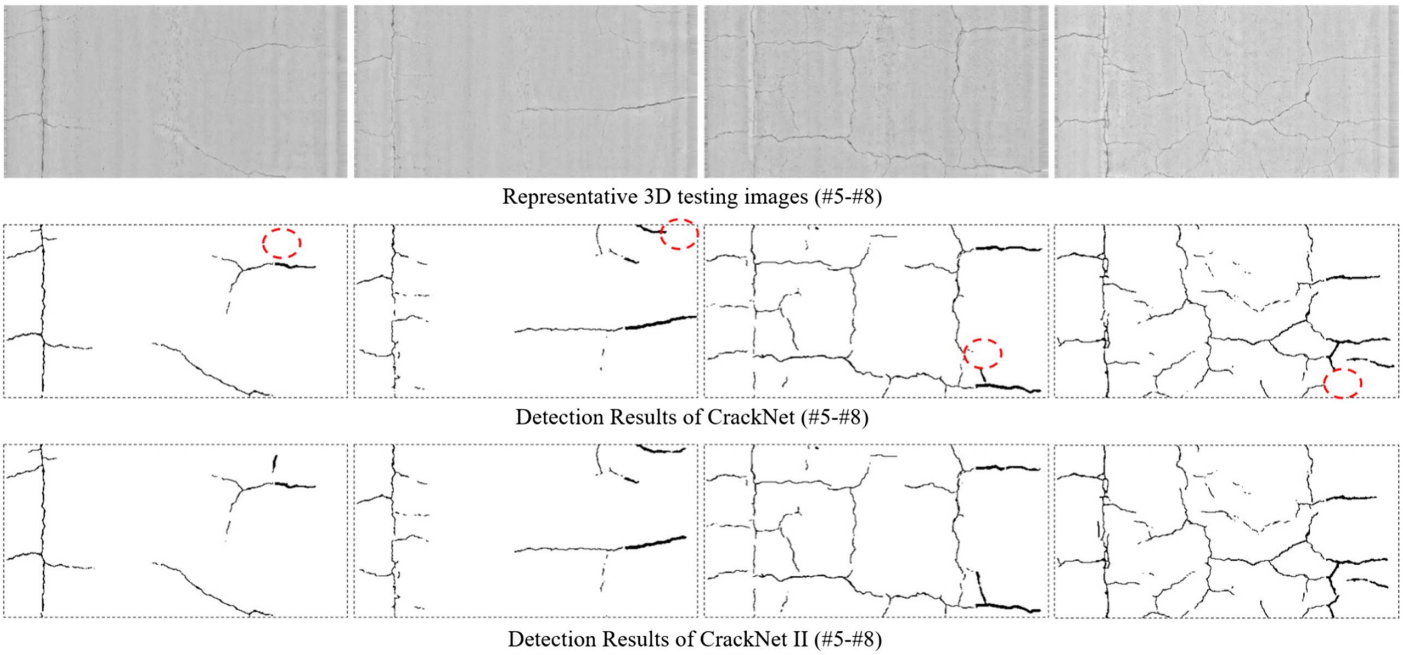


Fig. 9. False-negative errors reduced by CrackNet II.

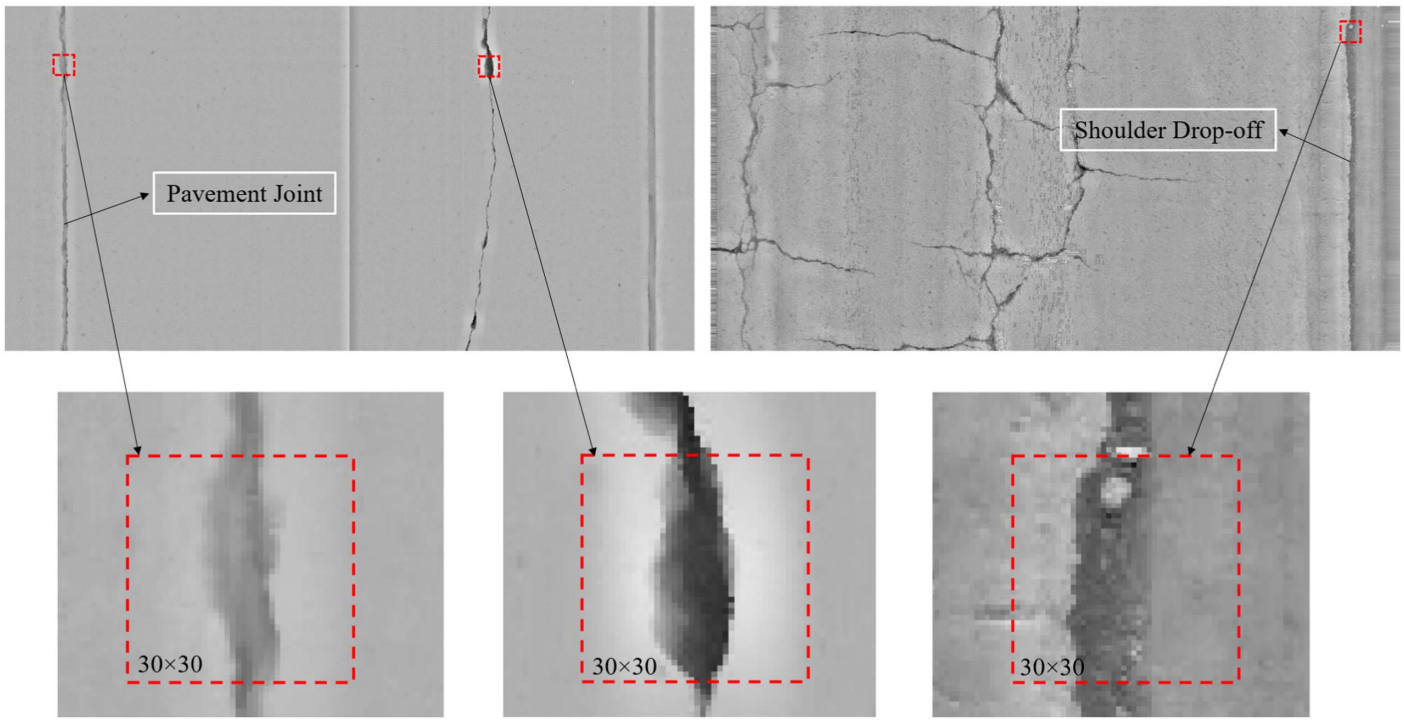


Fig. 10. Comparison between pavement crack and noise patterns via small local receptive field.

Table 7. Typical noise patterns misclassified by CrackNet II

Noise pattern	Possible surface type	
	Asphalt	Concrete
Shoulder drop-off	X	X
Pavement edge	X	X
Pavement joint	—	X
Pavement groove	—	X

context. Therefore, it is very challenging for CrackNet II to correctly classify the typical noise patterns summarized in Table 7. Figs. 11 and 12 show typical false-positive errors resulting from CrackNet II due to the existence of noise patterns.

As illustrated in Fig. 11, CrackNet II misclassifies the shoulder drop-off pattern on Fig. 11(a) as a longitudinal crack. Meanwhile, the pavement edge on Fig. 11(b) is recognized by CrackNet II as a longitudinal crack incorrectly. In addition, as shown in Fig. 12,

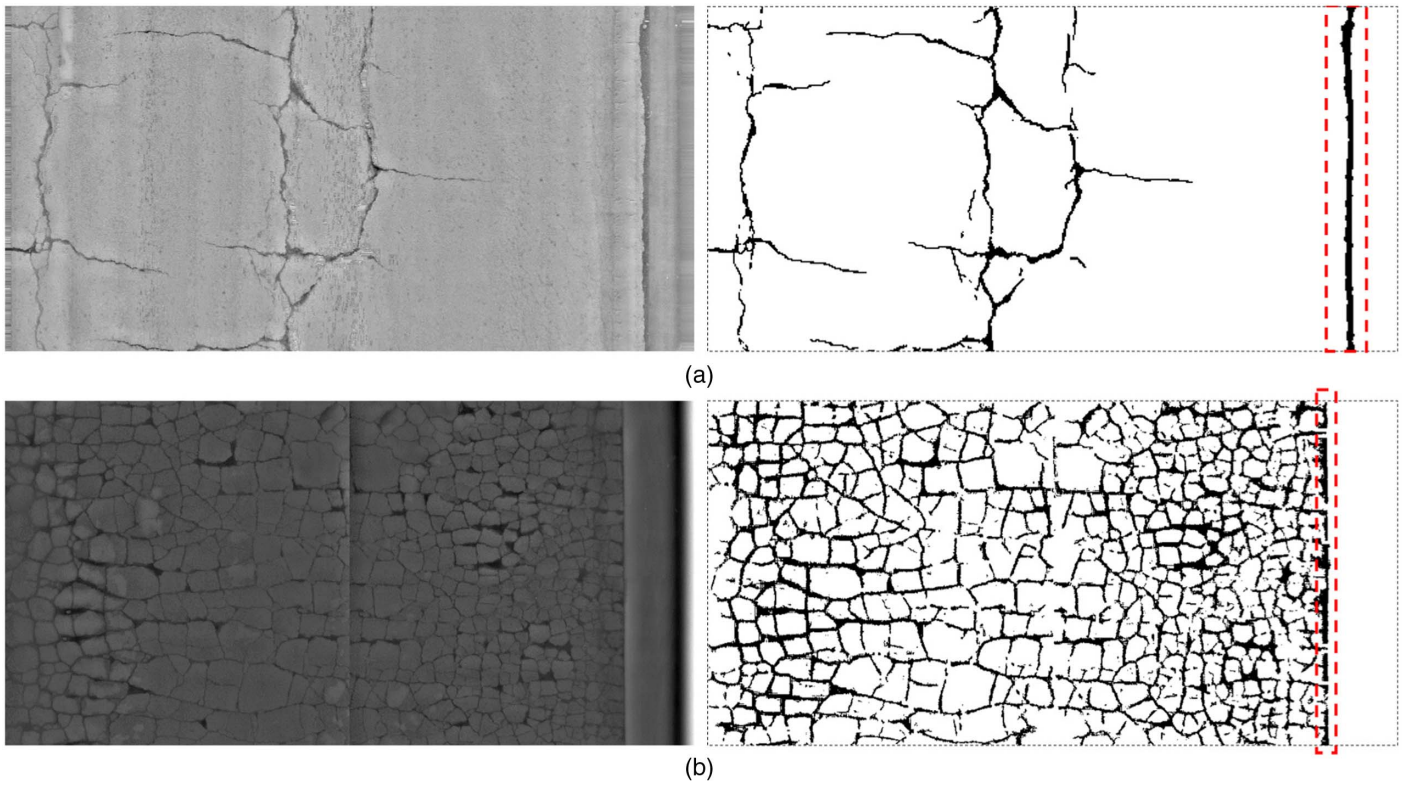


Fig. 11. Typical false-positive errors resulting from CrackNet II on asphalt surfaces.

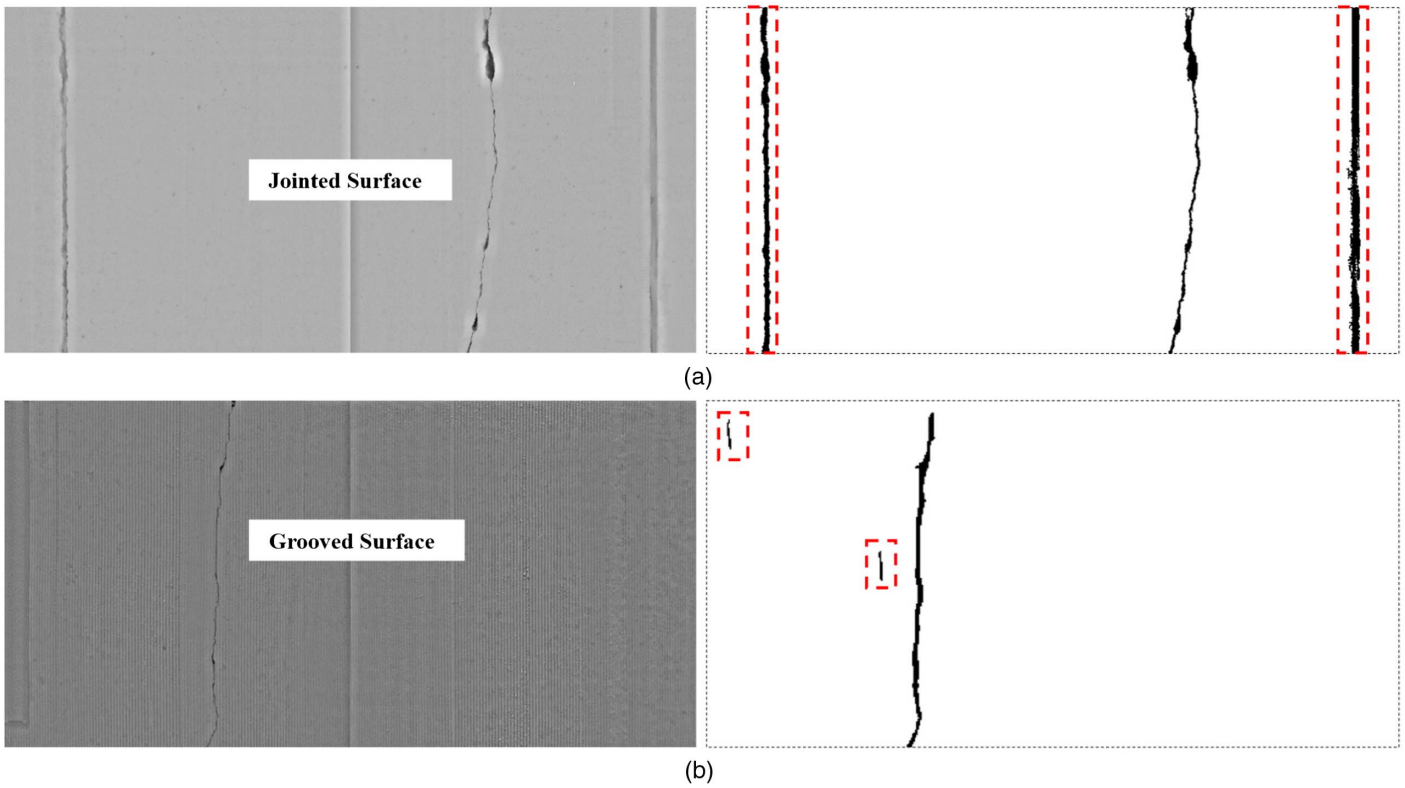


Fig. 12. Typical false-positive errors resulting from CrackNet II on concrete surfaces.

CrackNet II also introduces false-positive errors on concrete surfaces due to the existence of pavement joints and grooves. To distinguish pavement cracks from complex noise patterns efficiently, future developments need to explicitly address the learning of global context as well as the geometric characteristics of pavement cracks.

Because the detection is conducted at pixel level, an individual crack may be detected as unconnected parts, resulting in pixel-level

discontinuity. One possible reason is that the crack is originally discontinued at the pixel level. Another possible reason is that some pixels of the crack have weak responses to the trained filters and thus are misclassified as noncrack pixels. For engineering practice, the discontinuity of a crack will create difficulties in counting the number of cracks correctly and measuring the extent of an individual crack precisely. CrackNet II cannot guarantee the continuity of each detected crack, particularly for cases of originally discontinued cracks. However, based on the detection outputs of CrackNet II, postprocessing methods can be used to connect discontinued parts for entirety. For instance, morphological operations, especially the combination of dilation and erosion, are helpful in connecting discontinued parts (Li and Wang 2016). Alternatively, tensor voting is also a useful technique to enhance the continuity of crack fragments (Zou et al. 2012).

Although the training and testing images used in the paper are all 3D surface data, the architecture of CrackNet II is compatible with other types of data, such as 2D pavement images. All layers in CrackNet II conduct general-purpose operations and do not place any constraints on the type of source data. Thus, CrackNet II is expected to behave similarly for other types of data on the condition that it is trained appropriately.

Conclusions

In this paper, an improved version of CrackNet called CrackNet II was proposed for enhanced learning capability of the network. CrackNet II no longer uses the feature generator adopted in the original CrackNet that provides handcrafted features and executes nonlearnable procedures. Additionally, CrackNet II presents a deeper architecture with more hidden layers but fewer parameters. It was demonstrated that CrackNet II is nearly five times faster than the original CrackNet, which is significant because deep learning–based solutions generally require a very-high-performance computing facility.

Using 2,500 diverse examples of asphalt surfaces selected from the established 3D image library, CrackNet II was trained on two GPU devices recursively with 1,000 iterations. The training of CrackNet II was successfully completed with the aid of various efficient learning techniques, including minibatch gradient descent, momentum, cross entropy, normalized initialization, batch normalization, and dropout. It was shown that CrackNet II performs similarly on training and validation data, implying the overfitting problem is avoided. In the meantime, three alternative networks were trained in conjunction with CrackNet II. Through the comparison study, CrackNet II yields the highest overall F-measure on validation data compared with those alternative networks, indicating that increasing the sizes of local receptive fields does not necessarily promise better performances.

According to the experiment on 200 testing images, the trained CrackNet II behaves generally better than the original CrackNet in terms of both precision and recall. The overall precision and recall of CrackNet II on the 200 testing images were 90.20 and 89.06%, respectively. Compared with the original CrackNet, CrackNet II was found to be more robust in eliminating local noises and detecting fine or hairline cracks. The success of CrackNet II further validates the potential of CNNs in detecting pavement surface distresses at the pixel level. Improvements of CrackNet are continuing at a rapid pace by the authors. Future versions of CrackNet are anticipated to solve more complex problems, including detecting cracks on concrete pavement surfaces and suppressing cracklike patterns (i.e., pavement joints, grooves, shoulder drop-offs, and pavement edges).

References

Adeli, H., and X. Jiang. 2003. "Neuro-fuzzy logic model for freeway work zone capacity estimation." *J. Transp. Eng.* 126 (5): 484–493. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:5(484).

Adeli, H., and A. Samant. 2000. "An adaptive conjugate gradient neural network-wavelet model for traffic incident detection." *Comput.-Aided Civ. Infrastruct. Eng.* 15 (4): 251–260. https://doi.org/10.1111/0885-9507.00189.

Amhaz, R., S. Chambon, J. Idier, and V. Baltazart. 2014. "A new minimal path selection algorithm for automatic crack detection on pavement images." In *Proc., Int. Conf. on Image Processing*, 788–792. Piscataway, NJ: IEEE.

Attoh-Okine, N., and A. Ayenu-Prah. 2008. "Evaluating pavement cracks with bidimensional empirical mode decomposition." *EURASIP J. Adv. Signal Process.* 2008 (1): 861701. https://doi.org/10.1155/2008/251518.

Avila, M., S. Begot, F. Duculty, and T. S. Nguyen. 2014. "2D image based road pavement crack detection by calculating minimal paths and dynamic programming." In *Proc., Int. Conf. on Image Processing*, 783–787. Piscataway, NJ: IEEE.

Cha, Y. J., W. Choi, and O. Buyukozturk. 2017. "Deep learning-based crack damage detection using convolutional neural networks." *Comput.-Aided Civ. Infrastruct. Eng.* 32 (5): 361–378. https://doi.org/10.1111/mice.12263.

Cheng, H. D., J. Shi, and C. Glazier. 2003. "Real-time image thresholding based on sample space reduction and interpolation approach." *J. Comput. Civ. Eng.* 17 (4): 264–272. https://doi.org/10.1061/(ASCE)0887-3801(2003)17:4(264).

Ciresan, D., U. Meier, J. Masci, and J. Schmidhuber. 2012. "Multi-column deep neural network for traffic sign classification." *Neural Networks* 32: 333–338. https://doi.org/10.1016/j.neunet.2012.02.023.

Daniel, A., and V. Preeja. 2014. "A novel technique for automatic road distress detection and analysis." *Int. J. Comput. Appl.* 101 (10): 18–23.

Fawcett, T. 2006. "An introduction to ROC analysis." *Pattern Recognit. Lett.* 27 (8): 861–874. https://doi.org/10.1016/j.patrec.2005.10.010.

Gavilan, M., D. Balcones, O. Marcos, D. F. Llorca, M. A. Sotelo, I. Parra, M. Ocana, P. Aliseda, P. Yarza, and A. Amirola. 2011. "Adaptive road crack detection system by pavement classification." *Sens. J.* 11 (10): 9628–9657. https://doi.org/10.3390/s111009628.

Glorot, X., and Y. Bengio. 2010. "Understanding the difficulty of training deep feedforward neural networks." *J. Mach. Learn. Res.* v9: 249–256.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. "Deep learning." MIT Press. Accessed March 20, 2017. http://www.deeplearningbook.org/.

He, K., X. Zhang, S. Ren, and J. Sun. 2015a. "Deep residual learning for image recognition." Preprint, submitted December 10, 2015. http://arxiv.org/abs/1512.03385.

He, K., X. Zhang, S. Ren, and J. Sun. 2015b. "Spatial pyramid pooling in deep convolutional networks for visual recognition." *IEEE Trans. Pattern Anal. Mach. Intell.* 37 (9): 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. "Improving neural networks by preventing co-adaptation of feature detectors." Preprint, submitted July 3, 2012. http://arxiv.org/abs/1207.0580.

Huang, Y. X., and B. G. Xu. 2006. "Automatic inspection of pavement cracking distress." *J. Electron. Imag.* 15 (1): 013017. https://doi.org/10.1117/1.2177650.

Ioffe, S., and C. Szegede. 2015. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." Preprint, submitted February 11, 2015. http://arxiv.org/abs/1502.03167.

Jahanshahi, M. R., F. Jazizadeh, S. F. Masri, and B. Becerik-Gerber. 2013. "Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor." *J. Comput. Civ. Eng.* 27 (6): 743–754. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000245.

Jiang, X., and H. Adeli. 2005. "Dynamic wavelet neural network model for traffic flow forecasting." *J. Transp. Eng.* 131 (10): 771–779. https://doi.org/10.1061/(ASCE)0733-947X(2005)131:10(771).

Kaseko, M. S., Z. P. Lo, and S. G. Ritchie. 1994. "Comparison of traditional and neural classifiers for pavement-crack detection." *J. Transp. Eng.* 120 (4): 552–569. https://doi.org/10.1061/(ASCE)0733-947X(1994)120:4(552).

Kaseko, M. S., and S. G. Ritchie. 1993. "A neural network-based methodology for pavement crack detection and classification." *Transp. Res. Part C* 1 (4): 275–291. https://doi.org/10.1016/0968-090X(93)90002-W.

Kirschke, K. R., and S. A. Velinsky. 1992. "Histogram-based approach for automated pavement-crack sensing." *J. Transp. Eng.* 118 (5): 700–710. https://doi.org/10.1061/(ASCE)0733-947X(1992)118:5(700).

Kouroussis, G., D. Kinet, V. Moeyaert, J. Dupuy, and C. Caucheteur. 2016. "Railway structure monitoring solutions using fibre Bragg grating sensors." *Int. J. Rail Transp.* 4 (3): 135–150. https://doi.org/10.1080/23248378.2016.1184598.

Krizhevsky, A., I. Sutskever, and G. Hinton. 2012. "ImageNet classification with deep convolutional neural networks." In *Proc., 26th Annual Conf. on Neural Information Processing Systems*, 1097–1105. Vancouver, Canada: Neural Information Processing System Foundation.

LeCun, Y., Y. Bengio, and G. Hinton. 2015. "Deep learning." *Nature* 521 (7553): 436–444. https://doi.org/10.1038/nature14539.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-based learning applied to document recognition." *Proc. IEEE* 86 (11): 2278–2324. https://doi.org/10.1109/5.726791.

LeCun, Y., L. Bottou, G. B. Orr, and K. R. Muller. 2012. "Efficient BackProp." In Vol. 7700 of *Neural networks: Tricks of the trade, lecture notes in computer science*, 9–48. Berlin: Springer.

Lee, B. J., and H. D. Lee. 2004. "Position-invariant neural network for digital pavement crack analysis." *Comput.-Aided Civ. Infrastruct. Eng.* 19 (2): 105–118. https://doi.org/10.1111/j.1467-8667.2004.00341.x.

Li, L., and K. C. P. Wang. 2016. "Bounding box-based technique for pavement crack classification and measurement using 1 mm 3D laser data." *J. Comput. Civ. Eng.* 30 (5): 04016011. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000568.

Long, J., E. Shelhamer, and T. Darrell. 2014. "Fully convolutional networks for semantic segmentation." Preprint, submitted November 14, 2014. http://arxiv.org/abs/1411.4038.

Ma, X., and Y. Wang. 2014. "Development of a data-driven platform for transit performance measures using smart card and GPS data." *J. Transp. Eng.* 140 (12): 04014063. https://doi.org/10.1061/(ASCE)TE.1943-5436.0000714.

Maas, A. L., A. Y. Hannun, and A. Y. Ng. 2013. "Rectifier nonlinearities improve neural network acoustic models." In *Proc., 30th Int. Conf. on Machine Learning*. Atlanta: International Machine Learning Society.

Maji, D., A. Santara, P. Mitra, and D. Sheet. 2016. "Ensemble of deep convolutional neural networks for learning to detect retinal vessels in fundus images." Preprint, submitted March 15, 2016. http://arxiv.org/abs/1603.04833.

Miller, J. S., and W. Y. Bellinger. 2014. *Distress identification manual for the Long-Term Pavement Performance Program*. Washington, DC: Federal Highway Administration.

Nejad, F. M., and H. Zakeri. 2011. "An optimum feature extraction method based on wavelet-radon transform and dynamic neural network for pavement distress classification." *Expert Syst. Appl.* 38 (8): 9442–9460. https://doi.org/10.1016/j.eswa.2011.01.089.

Ngo, N. T., B. Indraratna, and C. Rujikiatkamjorn. 2017. "Stabilization of track substructure with geo-inclusions—Experimental evidence and DEM simulation." *Int. J. Rail Transp.* 5 (2): 63–86. https://doi.org/10.1080/23248378.2017.1279085.

Nguyen, T. S., M. Avila, and S. Begot. 2009. "Automatic detection and classification of defect on road pavement using anisotropy measure." In *Proc., 17th European Signal Processing Conf.*, 617–621. Piscataway, NJ: IEEE.

Nielsen, M. 2017. "Improving the way neural networks learn." Accessed February 15, 2017. http://neuralnetworksanddeeplearning.com/chap3.html.

Nisanth, A., and A. Mathew. 2014. "Automated visual inspection of pavement crack detection and characterization." *Int. J. Technol. Eng. Syst.* 6 (1): 14–20.

Oliveira, H., and P. L. Correia. 2009. "Automatic road crack segmentation using entropy and image dynamic thresholding." In *Proc., 17th European Signal Processing Conf.*, 622–626. Piscataway, NJ: IEEE.

Ouyang, W., and B. Xu. 2013. "Pavement cracking measurements using 3D laser-scan images." *Meas. Sci. Technol.* 24 (10): 105204. https://doi.org/10.1088/0957-0233/24/10/105204.

Pinheiro, P. O., and R. Collobert. 2015. "From image-level to pixel-level labeling with convolutional networks." Preprint, submitted November 23, 2014. http://arxiv.org/abs/1411.6228.

Qiu, S., D. X. Xiao, S. Huang, and L. Li. 2016. "A data-driven method for comprehensive pavement-condition ranking." *J. Infrastruct. Syst.* 22 (2): 04015024. https://doi.org/10.1061/(ASCE)IS.1943-555X.0000279.

Santhi, B., G. Krishnamurthy, S. Siddharth, and P. K. Ramakrishnan. 2012. "Automatic detection of cracks in pavements using edge detection operator." *J. Theor. Appl. Inf. Technol.* 36 (2): 199–205.

Sermanet, P., D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. 2014. "OverFeat: Integrated recognition, localization and detection using convolutional networks." Preprint, submitted December 21, 2013. https://arxiv.org/abs/1312.6229.

Shelhamer, E., J. Long, and T. Darrell. 2016. "Fully convolutional networks for semantic segmentation." Preprint, submitted May 20, 2016. http://arxiv.org/abs/1605.06211.

Simonyan, K., and A. Zisserman. 2015. "Very deep convolutional networks for large-scale image recognition." Preprint, submitted September 4, 2014. http://arxiv.org/abs/1409.1556.

Sollazzo, G., K. C. P. Wang, G. Bosurgi, and Q. Li. 2016. "Hybrid procedure for automated detection of cracking with 3D pavement data." *J. Comput. Civ. Eng.* 30 (6): 04016032. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000597.

Springenberg, J. T., A. Dosovitskiy, T. Brox, and M. Riedmiller. 2015. "Striving for simplicity: The all convolutional net." Preprint, submitted December 21, 2014. http://arxiv.org/abs/1412.6806.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2014. "Going deeper with convolutions." Preprint, submitted September 17, 2014. http://arxiv.org/abs/1409.4842.

Tschopp, F. 2015. "Efficient convolutional neural networks for pixelwise classification on heterogeneous hardware systems." Preprint, submitted September 11, 2015. http://arxiv.org/abs/1509.03371.

Wang, K. C. P. 2000. "Designs and implementations of automated systems for pavement surface distress survey." *J. Infrastruct. Syst.* 6 (1): 24–32. https://doi.org/10.1061/(ASCE)1076-0342(2000)6:1(24).

Wang, K. C. P. 2011. "Elements of automated survey of pavements and a 3D methodology." *J. Mod. Transp.* 19 (1): 51–57. https://doi.org/10.1007/BF03325740.

Wang, K. C. P., L. Li, and J. Q. Li. 2014. "Automated joint faulting measurement using 3D pavement texture data at 1 mm resolution." In *Proc., 2nd Transportation and Development Institute Congress*, 498–510. Reston, VA: ASCE.

Wang, K. C. P., Q. Li, and W. Gong. 2007. "Wavelet-based pavement distress image edge detection with à trous algorithm." *Transp. Res. Rec.*, 2024: 73–81. https://doi.org/10.3141/2024-09.

Ying, L., and L. Salari. 2010. "Beamlet transform-based technique for pavement crack detection and classification." *Comput.-Aided Civ. Infrastruct. Eng.* 25 (8): 572–580. https://doi.org/10.1111/j.1467-8667.2010.00674.

Yousefhussien, M., D. J. Kelbe, E. J. Ientilucci, and C. Salvaggio. 2017. "A fully convolutional network for semantic labeling of 3D point clouds." Preprint, submitted October 3, 2017. http://arxiv.org/abs/1710.01408.

Zalama, E., J. Gomez-Garcia-Bermejo, R. Medina, and J. Llamas. 2014. "Road crack detection using visual features extracted by Gabor filters." *Comput.-Aided Civ. Infrastruct. Eng.* 29 (5): 342–358. https://doi.org/10.1111/mice.12042.

Zeiler, M. D., and R. Fergus. 2013. "Visualizing and understanding convolutional networks." Preprint, submitted November 12, 2013. http://arxiv.org/abs/1311.2901.

Zhang, A., Q. Li, K. C. P. Wang, and S. Qiu. 2013. "Matched filtering algorithm for pavement cracking detection." *Transp. Res. Rec.* 2367: 30–42. https://doi.org/10.3141/2367-04.

Zhang, A., and K. C. P. Wang. 2017. "The fast prefix coding algorithm (FPCA) for 3D pavement surface data compression." *Comput.-Aided Civ. Infrastruct. Eng.* 32 (3): 173–190. https://doi.org/10.1111/mice.12243.

Zhang, A., K. C. P. Wang, and C. Ai. 2017a. "3D shadow modeling for detection of descended patterns on 3D pavement surface." *J. Comput. Civ. Eng.* 31 (4): 04017019. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000661.

Zhang, A., K. C. P. Wang, R. Ji, and Q. Li. 2016a. "Efficient system of cracking-detection algorithms with 1-mm 3D-surface models and performance measures." *J. Comput. Civ. Eng.* 30 (6): 04016020. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000581.

Zhang, A., K. C. P. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen. 2017b. "Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network." *Comput.-Aided Civ. Infrastruct. Eng.* 32 (10): 805–819. https://doi.org/10.1111/mice.12297.

Zhang, L., F. Yang, Y. D. Zhang, and Y. J. Zhu. 2016b. "Road crack detection using deep convolutional neural network." In *Proc., Int. Conf. on Image Processing*, 3708–3712. Piscataway, NJ: IEEE.

Zhou, J., P. S. Huang, and F. P. Chiang. 2006. "Wavelet-based pavement distress detection and evaluation." *Opt. Eng.* 45 (2): 027007. https://doi.org/10.1117/1.2172917.

Zou, Q., Y. Cao, Q. Q. Li, Q. Z. Mao, and S. Wang. 2012. "CrackTree: Automatic crack detection from pavement images." *Pattern Recognit. Lett.* 33 (3): 227–238. https://doi.org/10.1016/j.patrec.2011.11.004.