COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Computer-Aided Civil and Infrastructure Engineering **33** (2018) 1090–1109

Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network

Xincong Yang

Department of Building and Real Estate, the Hong Kong Polytechnic University, Hung Hom, Hong Kong and School of Civil Engineering, Harbin Institute of Technology, Harbin, Heilongjiang, China

Heng Li*, Yantao Yu, Xiaochun Luo & Ting Huang

Department of Building and Real Estate, the Hong Kong Polytechnic University, Hung Hom, Hong Kong

&

Xu Yang

School of Civil Engineering, Harbin Institute of Technology, Harbin, Heilongjiang, China

Abstract: *The spatial characteristics of cracks are significant indicators to assess and evaluate the health of existing buildings and infrastructures. However, the current manual crack description method is time consuming and labor consuming. To improve the efficiency of crack inspection, advanced computer vision-based techniques have been utilized to detect cracks automatically at image level and grid-cell level. But existing crack detections are of (high specificity) low generality and inefficient, in terms that conventional approaches are unable to identify and measure diverse cracks concurrently at pixel level. Therefore, this research implements a novel deep learning technique named fully convolutional network (FCN) to address this problem. First, FCN is trained by feeding multiple types of cracks to semantically identify and segment pixel-wise cracks at different scales. Then, the predicted crack segmentations are represented by single-pixel width skeletons to quantitatively measure the morphological features of cracks, providing valuable crack indicators for assessment in practice, such as crack topology, crack length, max width, and mean width. To validate the prediction, the predicted segmentations are compared with recent advanced method for crack recognition and ground truth. For crack segmentation, the accuracy, precision, recall, and F1 score are 97.96%, 81.73%, 78.97%, and 79.95%, respectively. For crack length, the relative measurement error varies from −48.03% to 177.79%, meanwhile that ranges from −13.27% to 24.01% for crack width. The results show that FCN is feasible and sufficient for crack identification and measurement. Although the accuracy is not as high as CrackNet because of three types of errors, the prediction has been increased to pixel level and the training time has been dramatically decreased to several per cents of previous methods due to the novel end-to-end structure of FCN, which combines typical convolutional neural networks and deconvolutional layers.*

*To whom correspondence should be addressed. E-mail: *heng.li@polyu.edu.hk*.

1 INTRODUCTION

Periodic crack detection plays a crucial part in the maintenance and operation of existing buildings and infrastructures. According to the morphological and positional features of cracks, the intrinsic damage, deterioration, and the potential causes can be inferred, which provides reasonable guidance on structure assessment (Rafiei and Adeli, 2017; Rafiei et al., 2017). However, due to the huge number and high variety of cracks, it is a tedious task to detect, describe, and record the cracks manually.

© 2018 *Computer-Aided Civil and Infrastructure Engineering*.
DOI: 10.1111/mice.12412

Over the last decade, a body of scholars proposed various algorithms to automatically detect cracks on concrete surfaces, pavements, etc. However, most of the popular methods can only address specific crack detection problems. For example, straightforward approaches apply local and global brightness thresholds to determine cracks where the illumination of images is adjusted to normal standard; feather-based and edge-based approaches separate an image into small blocks and then identify cracks by matching the small blocks with predefined features, where the crack features have to be defined clearly and accurately because the performance of these approaches are seriously depending on the predefined features. What's more, the previous approaches only represent the cracks from image, the postprocessing issue, such as how to extract the crack spatial characteristics in quantitative ways, is still an open challenge.

With the emergence of deep learning in computer vision, deep neural networks (DNNs) have shown the remarkable potential in classification, detection, and recognition (Liao, 2017). Compared with conventional methods, though DNNs have more layers and more parameters, they promote the object detection from image level to pixel level where each pixel of an image can be identified whether it belongs to target objects. Such pixel-level detection provides a potential way to detect and measure cracks precisely. Therefore, in spite of the extraordinary variety of the crack, it is possible to detect and analyze cracks at pixel level by applying DNN. The objective of this research is to propose a deep learning-based approach—fully convolutional network (FCN)—to crack detection and analysis. In addition, the crack image library is an open source for further research.

2 RELATED RESEARCH

Computer vision-based crack detection is one of the most popular topics in both the construction industry and academia. A number of approaches have been proposed to address this challenge. From the aspect of manual inputs, existing approaches can be divided into rule-based and machine learning-based (ML-based) approaches according to the difference of computer vision methods.

Rule-based approaches are approaches that use predefined features to identify or represent the cracks from images, which are based on a remarkable assumption that the crack pixels were always continuous and darker than their surroundings (Cheng et al., 1999). In respect to morphology, it is possible to use the structural features to detect cracks, such as histogram and threshold (Oliveira and Correia, 2009); Kirschke and Velinsky (1992) developed a histogram-based method to identify and locate small cracks on asphalt and cement concrete pavements; Cheng et al. (1999) proposed an automatic crack detector according to fuzzy set theory with superior performance. To increase the robustness of illumination and shading variance, general global transforms and local edge detection were introduced to crack contour detection, such as fast Haar transform (FHT), fast Fourier transform (FFT), Sobel and Canny edge detectors (Sinha and Fieguth, 2006; Yu et al., 2007; Santhi et al., 2012; Adhikari et al., 2014; Nisanth and Mathew, 2014; Yeum and Dyke, 2015). According to the conclusion of Abdel-Qader et al. (2003), although the FHT approach was relatively more reliable, the accuracy of crack detection was still potentially to be increased. And hence, three kinds of improvements were put forward: increasing the adaptation of global transforms, devising crack-specific filters, and combining multiple global and local detectors.

1. Novel global transforms: Ying and Salari (2010) partitioned the gray scale images into small windows and applied beamlet transform to extract linear surface cracks; Ito et al. (2002) and Yamaguchi et al. (2008) used global wavelet transform to measure the fine cracks at pixel level; Li et al. (2014) proposed a long-range detection and a region-based active contour model to extract cracks form concrete surface images (Li et al., 2017).
2. Specific crack filters: Huang and Xu (2006) divided images into grid cells and constructed crack-specific filters to produce the crack "seeds," which are grown and joined to become cracks; Sinha and Fieguth (2006) fused a statistical filter and two crack detectors to detect cracks on pipes; Nishkawa et al. (2012) summarized multiple filters from associated sequential images to detect major and minor cracks, respectively; Zalama et al. (2014) worked out a lite solution using Gabor filter to identify longitudinal and transverse cracks on pavements.
3. Hybrid detection: Ayenu-Prah and Attoh-Okine (2008) combined the common local edge detectors and bidimensional empirical mode decomposition (BEMD) to generate a nonparametric and data-driven approach; Subirats et al. (2006) presented a continuous wavelet transform with coefficient maps to generate binary prediction images; Nejad and Zakeri (2011) decomposed the input images using wavelet-radon transform at multiple resolution levels and applied neutral networks to classify the detected cracks; Wu et al.

(2014) came up with a crack defragmentation technique named MorphLink-C, containing two steps, dilation transform and thinning transform, which provided valuable average crack width for road assessment; Chen et al. (2017b) adopted self-organizing map optimization and high-pass filter to mark bridge cracks.

Although rule-based approaches were fast and easy to be embedded in vision systems, the robustness of performance was not as expected where the brightness, shading, and resolution varied from predefined configurations. Therefore, ML-based approaches also attracted scholars to conduct and deliver feasible solutions to crack detection.

Dating back to early times, typical supervised ML-artificial neural networks (ANN) was mainly used to classify the subimages, to determine whether the small tiles of crack images were parts of actual cracks (Kaseko and Ritchie, 1993; Liu et al., 2002; Lee and Lee, 2004; Moon and Kim, 2011). However, due to the limitation of computational capability, the structures of applied ANN were simple leading to insufficient detection performances. Thereby, researchers tried to conduct other ML-based approaches to achieve a better detection. Gavilán et al. (2011) extracted pavement crack candidates by multiple directional nonminimum suppression and then devised a linear support vector machine (SVM) classifier to distinguish the crack types; O'Byren et al. (2013) developed a texture-based solution with a nonlinear SVM model, extending the crack detection from linear to nonlinear domain; Cha et al. (2016) stated a mixed method for specific loosed bolt, combining linear SVM and Hough transforms; Zou et al. (2012) built a three-step CrackTree to generate a probability map for crack localization and identification. Recently, as convolutional neural networks (CNNs) gained a notable success in computer vision recognition, researchers introduced this state-of-the-art technique to detect structural damages immediately. Zhang et al. (2016b) proposed a six-layer CNN to detect and characterize the road cracks; Cha et al. (2017b) came up with an eight-layer CNN to scan a large-resolution image by small blocks; Cha et al. (2017c) also proposed a region-based deep learning for multiple damages in real time and introduced phase-based optical flow model to take time into consideration. With the improvement of computing capability, researchers found that CNN performed better with deeper layer, and deep CNN as a kind of DNN was proposed and implemented in practice. For example, Chen et al. (2017a) proposed an 11-layer DNN integrated with Naïve Bayes to detect multiple damages for the inspection of nuclear power plants; Zhang et al. (2017) devised a five-layer CNN with predefined line filters to detect cracks on 3D asphalt surfaces; Lin et al. (2017) constructed an automatic damage feature-extraction DNN. The results of these researches showed DNN performed successfully in realistic situations compared with conventional methods. What's more, Koziarski and Cyganek (2017) discussed the noising issues in DNN; Ortega-Zamorano et al. (2017) even worked out implementations of Field-Programmable Gate Array (FGPA) for DNN.

With the development of hardware, the dimension of input images was increased to provide a wealth of relevant information to model, meanwhile the resolution of output images was developed to fulfill the specific task demands, as shown in Figure 1. Therefore, dealing with a large data set raised a multiscale and multilevel challenge that must be addressed. Deep CNN, as one kind of DNN, has presented its great potential for multiscale and multilevel image processing. First, both gray and colorful images, even Red-Green-Blue (RGB)-depth images could be mixed as the input of DNNs (Jahanshahi et al., 2012; Jahanshahi and Masri, 2012; Ouyang and Xu, 2013; Huang et al., 2014; Jiang and Tsai, 2015; Zhang et al., 2016a; Guldur Erkal and Hajjar, 2017; Zhang and Wang, 2017). On the other side, both local and global features could be learned by DNNs, leading to multilevel outputs, including image level, grid level, and pixel level. Therefore, identification and localization could be achieved concurrently without latency time between the two recognition tasks.

However, current DNN-based crack detection methods are still at grid-cell level. As a result, engineers have to divide objective images into patches, making it difficult to directly characterize the crack properties. This article utilizes a novel DNN-based method, providing an efficient crack detection approach at pixel level. Compared with previous methods, the structure of DNN in this article is deeper and more comprehensive, which combines the identification and localization of cracks together, making crack detection and description more convenient, robust, and general. The main contributions of this article are:

1. Eliminate the sliding windows by introducing deconvolutional layer, which creates a detector that can recognize and localize cracks at varying scales. Meanwhile, the training time can be dramatically decreased by fine-tuning pretrained models.
2. Combine morphological operations and DNN to extract the geometric characteristics directly from images without manual measurement, such as crack length, max width, and mean width.

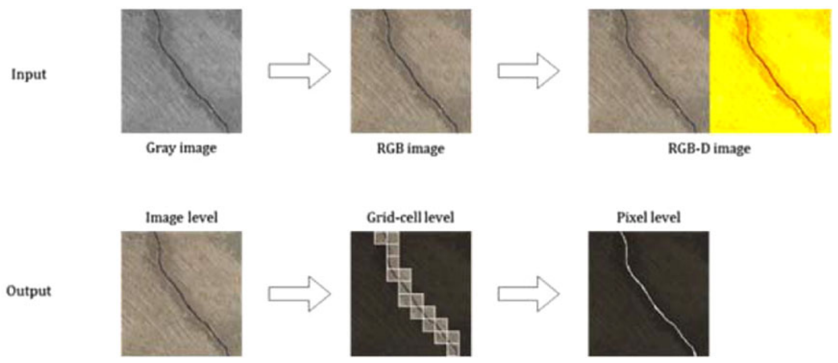


Fig. 1. Developing trend of input and output for crack detection.

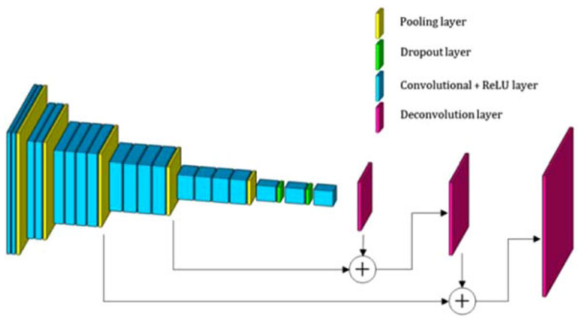


Fig. 2. The framework of FCN.

3 FCN

FCN is an end-to-end, pixel-to-pixel convolutional network on the semantic segmentation (Long et al., 2015). FCN can be treated as an expanded CNN that the prediction has been converted from a class number to a semantic segmentation image, which is also known as dense prediction. The framework of FCN is illustrated in Figure 2. The whole DNN is composed of two parts: down-sampling and up-sampling parts. Down-sampling part contains convolutional layers, pooling layers, and dropout layers; meanwhile, the up-sampling part includes deconvolutional layers.

The down-sampling part is actually a very deep convolutional network with 19 layers, which is also named VGG19 network (Simonyan and Zisserman, 2014). Notably, the down-sampling process can also be replaced with other state-of-the-art DNNs, such as VGG16 (Simonyan and Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), and ResNet (He et al., 2016). VGG19 network is one of remarkable DNNs, and generally used in object classification and recognition for large-scale images. A number of applications in industry have proved the effectiveness of VGG19 and the pretrained parameters can significantly decrease the whole training time. Therefore, this research determines to use VGG19 to perform the crack detection task.

The up-sampling part is the core novelty of FCN, which inverses the process of down-sampling, leading to a dense prediction. Such dense output of FCN is distinct from the classification output of common CNN. First, the up-sampling part combines global information and local information by adding specific layers from convolutional layers and deconvolutional layers. As a result, the local information in the previous convolutional layers mainly answers "what the object is" (i.e., crack detection) and the global information in the deconvolutional layers mainly answers "where the object is" (i.e., crack localization). On the one side, this integrated structure ensures the identification and localization issues to be addressed at the same time; on the other side, the up-sampling part enlarges the classifications of down-sampling to the original image size, contributing to the same image size of input images and output predictions. Therefore, the unique structure ensures the ability of FCN to deal with multiscale and multilevel images in theory. The associated layers and customized layers are introduced in the following sections (for dropout layer, please refer to Srivastava et al., 2014).

3.1 Convolutional layer (Conv)

Each convolutional layer starts with a receptive filter/kernel/field convoluting input image or tensor, adding a bias, and ends with an activation function. These two operations introduce linear and nonlinear transformation components, respectively, converting the input image or tensor into a new tensor to intrinsic feature maps. It is important to note that the convolutional operations capture the local dependencies in the original image, and such filters learn the values on their own during the training process. The more filters the the network contains, the more image features

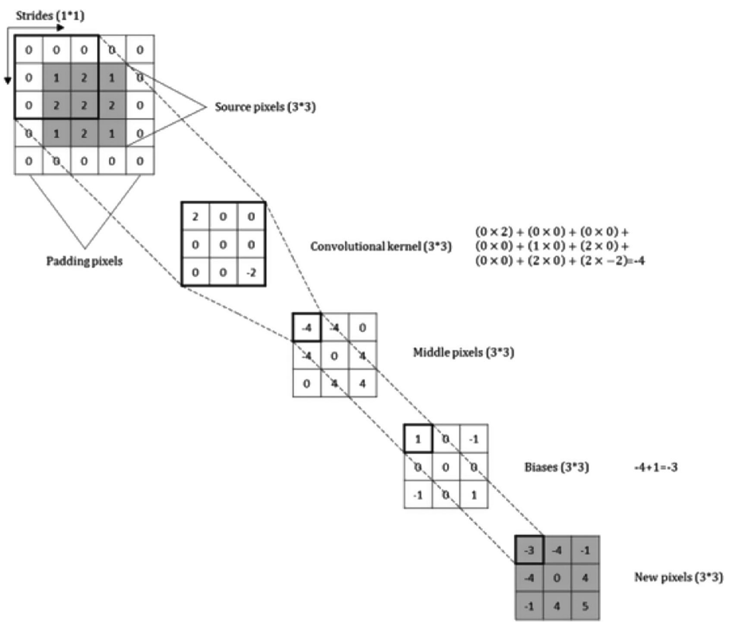


Fig. 3. Convolutional operation example.

get extracted and the better the network becomes at recognizing unseen patterns. That is why deep CNN performs better than light CNN.

Convolution generates the integral of the element-wise multiplication of input data and convolutional kernel. For a single image, the convolution process can be represented by:

$$y_{ij} = \sum_{(m,n)\in k} w_{m,n} x_{is+m,js+n} + b_{ij} \tag{1}$$

where y and x are the element values of output and input. k is the kernel, which is always square. w and b are the element of weights and biases. s represents the amount of pixels by which the convolutional kernel shifts at each step. Figure 3 shows an example convolutional layer for images with depth of one. To keep the constant between input size and output size, zero-padding is always applied to input images as described in Figure 3.

Although some types of cracks can be detected in gray-level images, it is still necessary to detect cracks by colorful patterns as colorful images contain a wealth of information leading to a robust detector under different light conditions. In addition, if the model is applied for other kinds of structural surface damages, such as steel corrosion, concrete voids and pits, leakage, only gray patterns are insufficient to achieve an accurate detection. Therefore, this research determines to collect colorful images as training samples, offering a potential to integrate multiple detectors in future research. As Figure 4 shows, the representation for ith feature map in matrix form is:

$$Y_i = \sum_{j=0}^{d} F_i\left(X_j\right) + B_i \tag{2}$$

where Y is the feature map whereas X is the input tensor. B is the bias vector sharing the same shape with the output feature map. d refers to the depth of the input sensor and $F_i(\cdot)$ represents the convolutional operation with predefined kernels and strides.

Activation introduces nonlinear operations into the processing system by applying a nonlinear function to convolution results afterward. Based on research on computing efficiency, sigmoid, tanh, arctan, and rectified linear unit (ReLU) functions perform successfully in DNNs. To simplify the computation and facilitate the training process, ReLU is selected in this research because of its simple derivative function.

3.2 Pooling layer (Pool)

The objective of a pool layer is to decrease the size of processing data, which is the core approach to down-sampling. There commonly exist two kinds of pooling operations: max pooling and average pooling (see Figure 5). Max pooling only leaves the maximum value given a kernel/window moving across the whole input tensor step by step, whereas the average pooling

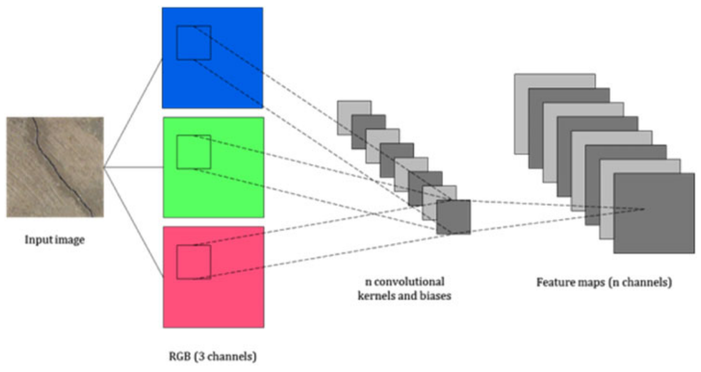


Fig. 4. Convolutional layer example.

computes the mean value. The mathematical form can be illustrated by:

$$y_{ij,max} = max\left(\left\{x_{is+m,js+n}\right\}, (m,n) \in k\right) \tag{3}$$

$$y_{ij}, avg = \frac{1}{k^2} \sum_{(m,n)\in k} x_{is+m,js+n} \tag{4}$$

where k is the kernel and s is the stride length. The output size can be represented by the formula in convolutional layers.

3.3 Deconvolutional layer (Deconv, Tconv)

Deconvolutional layers, are also considered as transposed/backward/up-sampling/fractally convolutional layers. As Figure 6 describes, Deconv is actually a transposed convolutional layer with a specific stride length and padding, which converts a coarse input tensor into a dense output tensor (compare Figure 3 and Figure 6). At first, each element of the input tensor is multiplied by deconvolutional kernel, and then these middle matrixes are combined with strides in both horizontal and vertical axes. When elements are overlapped, their values are added together to extract an extended matrix of input. Finally, the matrix with objective size is cropped, followed by adding biases as shown in Figure 6b. It is obvious that the size of output is larger than that of input, leading to an efficient approach to up-sampling. Notably, implementation of deconvolutional layers is the same as normal convolutional layers, but the filter is the transpose form of normal filter.

Deconvolutional layer is the core part of FCN that enlarges the size of tensor generated by down-sampling process to the original image size. Such structure enables FCN to detect cracks without sliding original images into patches, leading to a satisfactory accuracy with local and global dependencies.

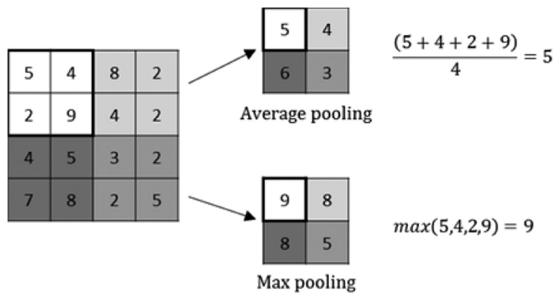


Fig. 5. Pooling layer example.

3.4 Softmax layer

The output of the final deconvolutional layer is a tensor that each pixel within the original image has been scored with each class. To generalize the output, a logistic function named softmax is utilized for multiclass classification. As Figure 7 shows, the softmax function (Bishop, 2006) can transform the input values in a multiclass categorical probability distribution by a normalized exponential as follows:

$$\mathrm{y}_j = \frac{e^{x_j}}{\sum_{i=1}^{N} e^{x_i}} \tag{5}$$

where y are the logits for jth class in model. N is the total number of classes.

3.5 Architecture of FCN

The architecture of FCN is listed in Table 1.

4 TRAINING PROCESS

4.1 Crack data set

To train the FCN model, the authors collected more than 800 images. The width of cracks varies from one pixel to 100 pixels, the shape of which is hard to

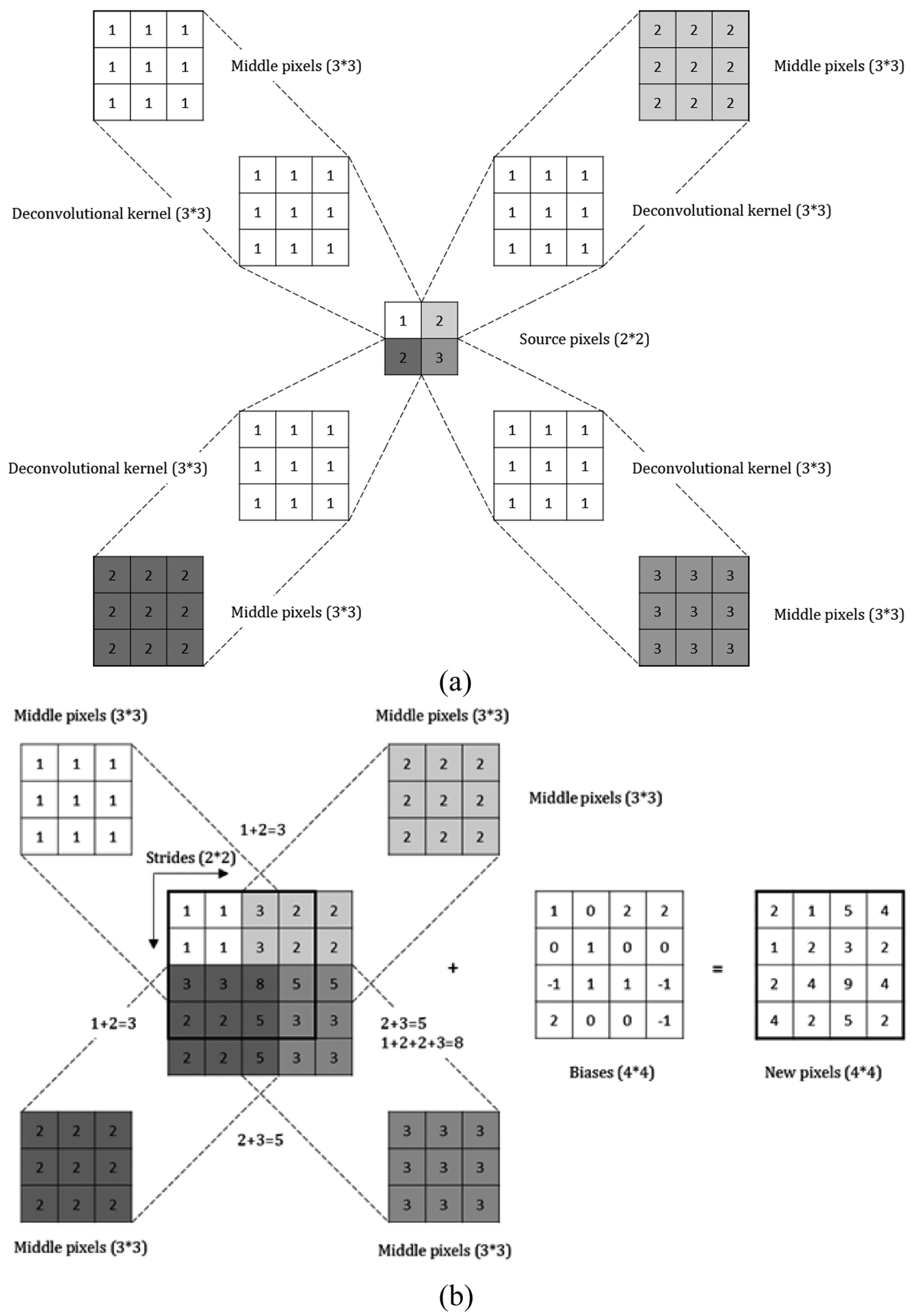


Fig. 6. Deconvolutional layer example.

recognize at image level. To ensure the variability, historical cracks from internet and new cracks from existing buildings in Harbin, China, are involved. Not only pavement cracks, but also cracks on concrete walls are contained and saved in JPG format. Cracks in these images are taken at different distances depending on their sizes, which leads to different levels of resolutions, ranging from 72 dpi to 300 dpi.

The ground truth of cracks is manually marked at the pixel level in PNG format, where each pixel is represented by an eight-bit integer. Here, the objective only focuses on the detection and location of cracks, thus the examiners denoted the background pixels as zero meanwhile the crack pixels as one.

To assess the generalization ability of the proposed approach, the crack data set is divided into five parts

Table 1
The architecture of FCN

Layer	*Filter/Kernel [H, W, IN, OUT]/[H, W]*	*Stride [H, W]*	*Output size [H, W, D]*
Input image			[224, 224, 3]
Conv1_1	[3,3,3,64]	[1,1]	[224,224,64]
Conv1_2	[3,3,3,64]	[1,1]	[224,224,64]
Pool1(avg)	[2,2]	[2,2]	[112,112,64]
Conv2_1	[3,3,64,128]	[1,1]	[112,112,128]
Conv2_2	[3,3,128,128]	[1,1]	[112,112,128]
Pool2(avg)	[2,2]	[2,2]	[56,56,128]
Conv3_1	[3,3,128,256]	[1,1]	[56,56,256]
Conv3_2	[3,3,256,256]	[1,1]	[56,56,256]
Conv3_3	[3,3,256,256]	[1,1]	[56,56,256]
Conv3_4	[3,3,256,256]	[1,1]	[56,56,256]
Pool3(avg)	[2,2]	[2,2]	[28,28,256]
Conv4_1	[3,3,256,512]	[1,1]	[28,28,512]
Conv4_2	[3,3,512,512]	[1,1]	[28,28,512]
Conv4_3	[3,3,512,512]	[1,1]	[28,28,512]
Conv4_4	[3,3,512,512]	[1,1]	[28,28,512]
Pool4(avg)	[2,2]	[2,2]	[14,14,512]
Conv5_1	[3,3,512,512]	[1,1]	[14,14,512]
Conv5_2	[3,3,512,512]	[1,1]	[14,14,512]
Conv5_3	[3,3,512,512]	[1,1]	[14,14,512]
Conv5_4	[3,3,512,512]	[1,1]	[14,14,512]
Pool5(max)	[2,2]	[2,2]	[7,7,512]
Conv6	[7,7,512,4096]	[1,1]	[1,1,4096]
Dropout6			[1,1,4096]
Conv7	[1,1,4096,4096]	[1,1]	[1,1,4096]
Dropout7			[1,1,4096]
Conv8	[1,1,4096,2]	[1,1]	[1,1,2]
Deconv1	[4,4,2,512]	[2,2]	[14,14,512]
Deconv1 + Pool4			[14,14,512]
Deconv2	[4,4,512,256]	[2,2]	[28,28,256]
Deconv2 + Pool3			[28,28,256]
Deconv3	[16,16,256,2]	[8,8]	[224,244,2]
Softmax			[224,244,1]
Prediction			[224,224]

Note: H = height; W = width; IN = in channels; OUT = out channels; D = depth.

according to fivefold cross-validation principle that 80% are used to feed the model and the last 20% for validation.

For the training process, the crack images and ground truth images are both randomly selected and resized to 224 × 224 to feed the model. Notably, the training process of FCN is carried out at the pixel level that each pixel in images can be treated as a learning sample, which means FCN can learn a number of global and local features from both original images and labeled images. Therefore, the batch size to train FCN is small, determined to be 1, 2, and 4 during each feeding time. Meanwhile, the amount of required training images is much smaller than CrackNet CNN proposed in Zhang et al. (2017) and traditional CNNs in Cha et al. (2017a).

4.2 Loss function

To assess the discrepancy between ground truth and predicted logits, cross entropy is selected as the loss function, which performs well in DNNs (Shannon, 2001; Goodfellow et al., 2016). The generated loss is also used to update the model parameters as well as to evaluate the performance of crack detection. For each pixel, the corresponding loss can be formulated as:

$$l = y \ln a + (1 - y) \ln(1 - a) \qquad (6)$$

where y is ground truth and a is the predicted score at the specific pixel. For each image, the total loss is the average of losses at pixels.

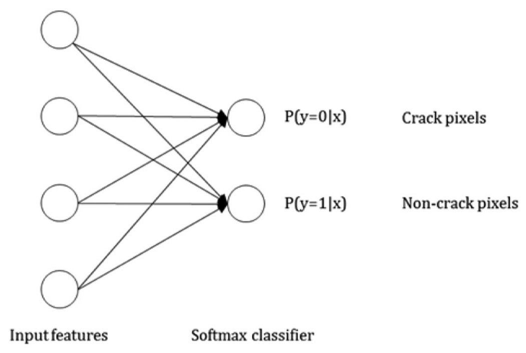


Fig. 7. Softmax layer.

4.3 Optimizer and learning rate

To minimize the total loss, the model parameters are updated by the Adam optimizer for iterations. Compared with Momentum, RMSprop, etc. algorithms, Adam optimizer applies bias-correction to promote the gradients to converge on the right direction at an admirable speed. The update process can simply be represented as follows:

$$m_{t+1} = \alpha m_t + (1-\alpha)\,\Delta x_t \tag{7}$$

$$v_{t+1} = \beta v_{t+1} + (1-\beta)\,(\Delta x_t)^2 \tag{8}$$

$$\Delta x_{t+1} = -lr\frac{m_{t+1}}{\sqrt{v_{t+1}}+\varepsilon} \tag{9}$$

where $\alpha = 0.9$ and $\beta = 0.999$ are default exponential decay rate for the first and second moment estimates. $\varepsilon = 10^{-8}$ is a small default number to keep the numerical stability during optimization. lr refers to the learning rate during training process.

To train the deep network in this research, it is helpful to anneal the learning rate over training steps, as the decaying learning rate can settle the model down into deeper without wasting computation bouncing chaotically around the global optimization (Wilson and Martinez, 2001). The authors determined to apply an exponential staircase decay function to reduce the initial learning rate as follows:

$$lr_t = lr_0 \times r_d^{\left\lfloor \frac{t}{t_{max}} \right\rfloor} \tag{10}$$

where r_d is the decay rate and t is the iteration step. $\lfloor\cdot\rfloor$ is the floor operation, yielding the greatest integer that is less than or equal to input value.

4.4 Model initialization

To save training time and generalize the training process, for the down-sampling part of FCN, the parameters in convolutional layers are initialized from pretrained VGG19 weights; for the up-sampling part, the filters are initialized by truncated normal distribution with mean of zero and standard deviation of 0.01, and the biases are initialized with constant zero vectors; the keep probability for dropout layers is determined to be 0.4.

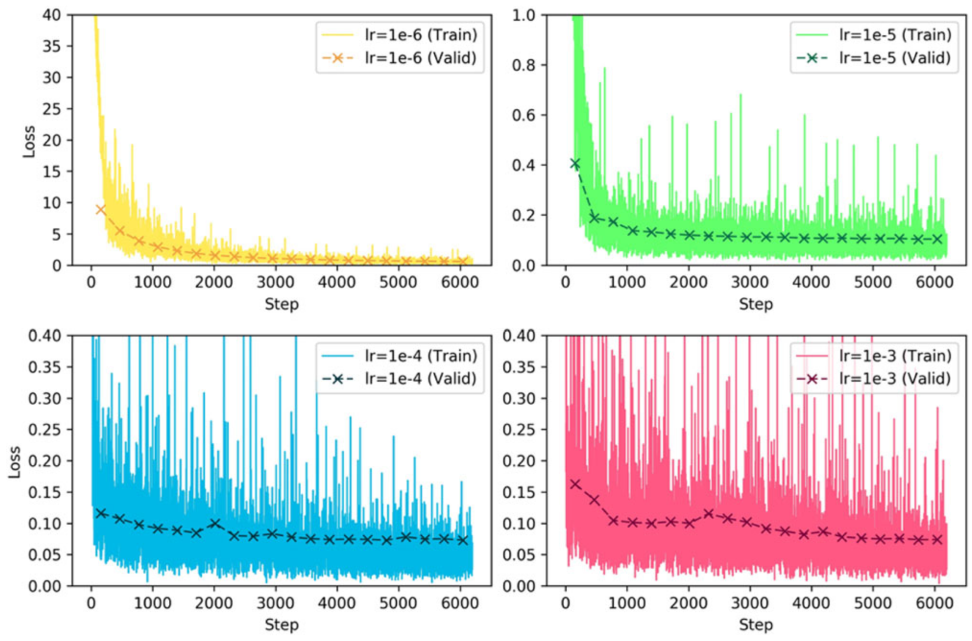


Fig. 8. Training losses and validating losses over training steps.

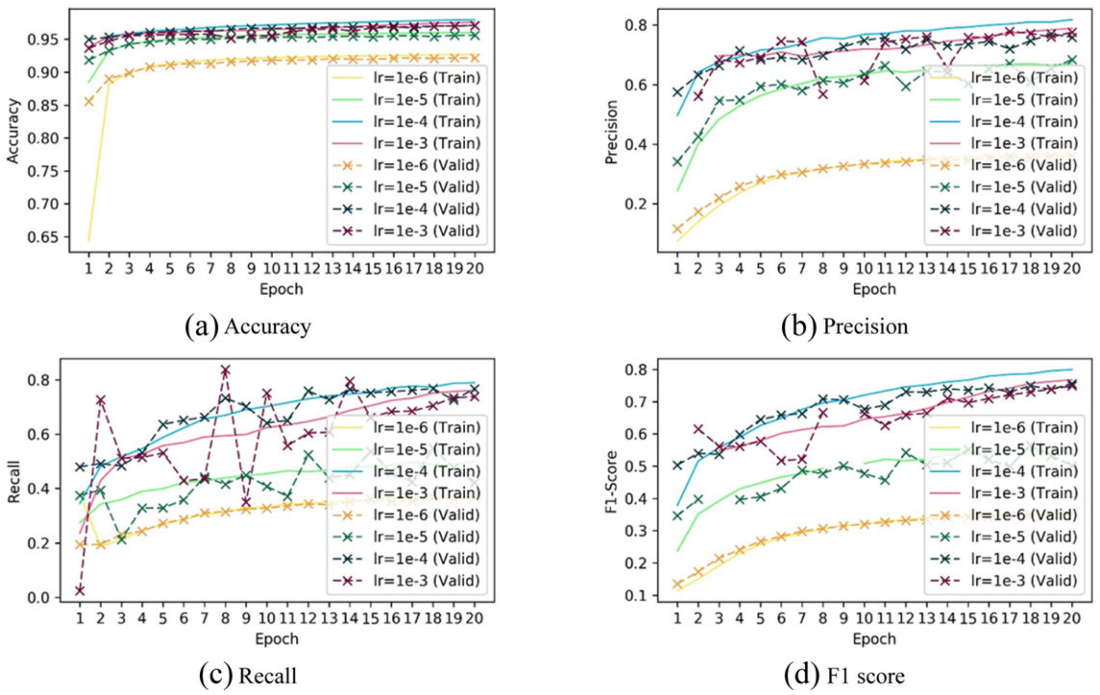


Fig. 9. (a) Accuracy, (b) precision, (c) recall, and (d) F1 score of training and validating processes.

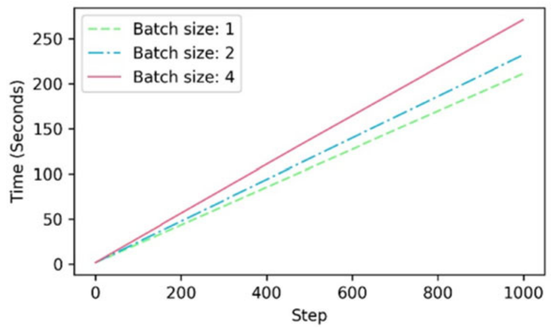


Fig. 10. Training time for 1,000 iterations.

5 TRAINING RESULTS

The training process was carried out on a workstation with a high-performance GPU (GeForce GTX 1080Ti) and a CPU (Intel Core i7, 3.60 GHz $\times$ 8), where GPU was capable for massive parallel computing and CPU is for variable updating.

The code was programmed by Python 3.5 and the virtual environment was established by TensorFlow 1.4, NumPy 1.14. Such configuration ensures the training process can be performed in both Windows and Linux systems.

To identify a suitable initial learning rate, this training process examined the value of 10^{-6}, 10^{-5}, 10^{-4}, and 10^{-3}. The loss curves of training process over steps are displayed as follows.

As shown in Figure 8, solid light curves and dashed dark lines represent the training loss and validating loss, respectively. Commonly, the solid yellow line with an initial learning rate of 10^{-6} seemed to be ideal, because the value of loss decreased quickly at the beginning and slowly in the end. However, after more than 6,000 iterations, the ultimate loss converged to around 0.7, which was inadequate to achieve a global optimization. The training process with initial learning rate of 10^{-5} was plotted by the green line. This line performed well because the loss decreased faster and the optimization was close to zero. Among these subplots, the blue line and red line represented the initial learning rates of 10^{-4} and 10^{-3}. The training processes of these two lines converged fastest and the final optimization was also smallest. Here, all the validating losses performed close to training losses to ensure that the parameters of FCN were not over-fit during training progress.

Figure 9a illustrated the increase of average accuracy over epochs. The max mean of accuracy was 97.96%, which was achieved with an initial learning rate of 10^{-4}. In Figures 9b, c, and d, the max precision, recall, and F1 score were 81.73%, 78.97%, and 79.95%. Therefore, in practice, the authors suggested that using 10^{-4} as the initial learning rate was reasonable for

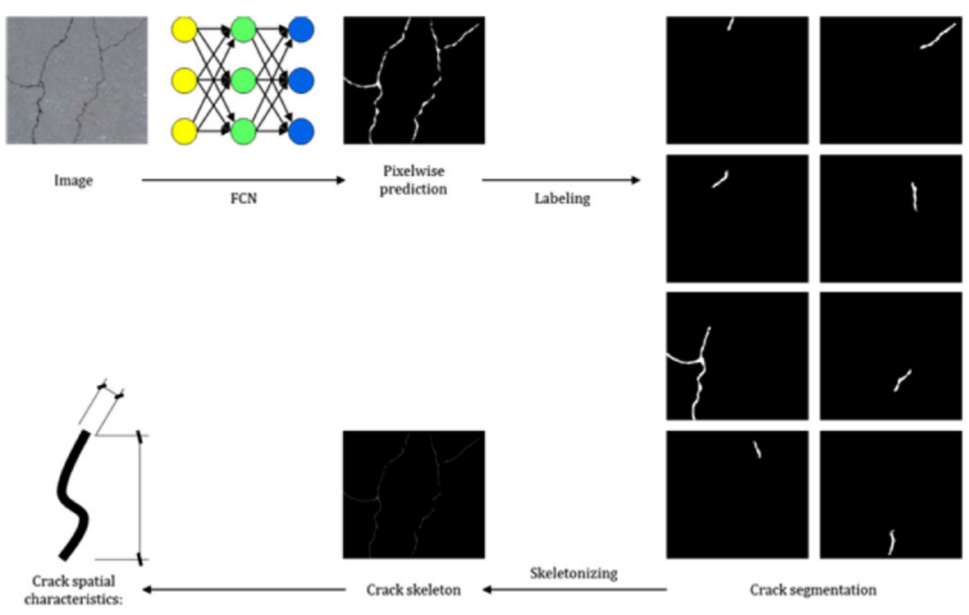


Fig. 11. The framework of crack morphological features extraction.

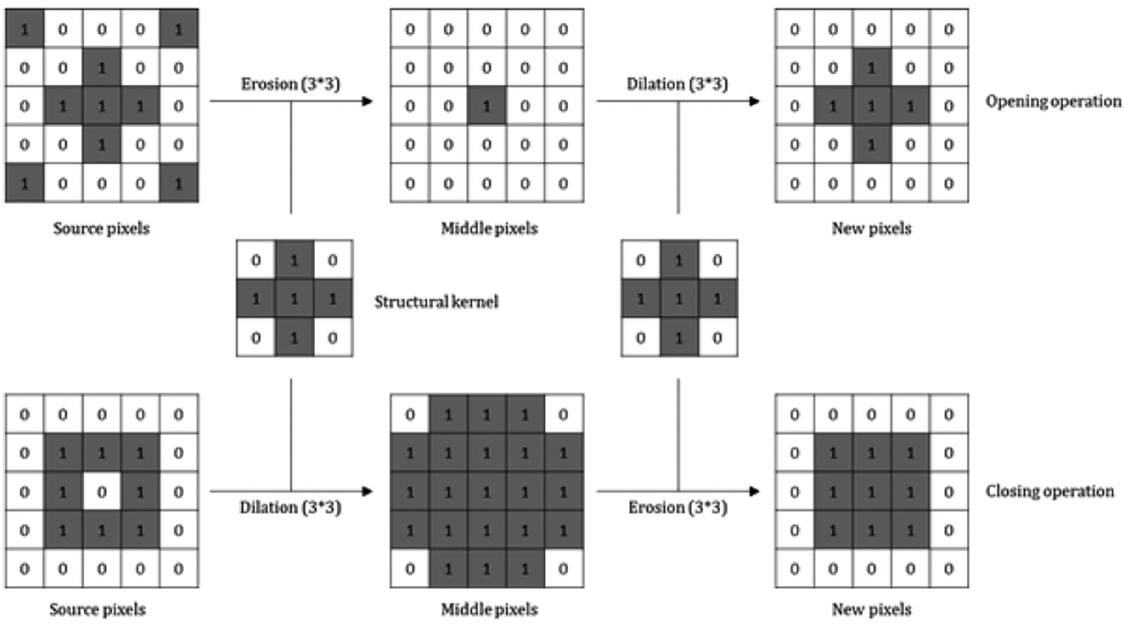


Fig. 12. Dilation and erosion operation example.

crack recognition. Although FCN performs better than Pixel-SVM and 3D shadow modeling, it was not as good as CrackNet (90.13%, 87.63%, and 88.86%) proposed by Zhang et al. (2017), meanwhile the gap between solid training lines and dashed validating lines revealed the potential of over-fit after 14 epochs. The reason might be the invariance of data set as the cracks only contain a few categories.

Although the accuracy of FCN was not highest as expected, the training time was significantly decreased. Compared with 9 days for training CrackNet (Zhang et al. 2017) after 700 iterations, FCN here only took less than an hour for 1,000 iterations. Here, the authors also tested different batch size for feeding. It can be seen from Figure 10 that smaller batch size costs less time for training, but the number of epochs had to be increased

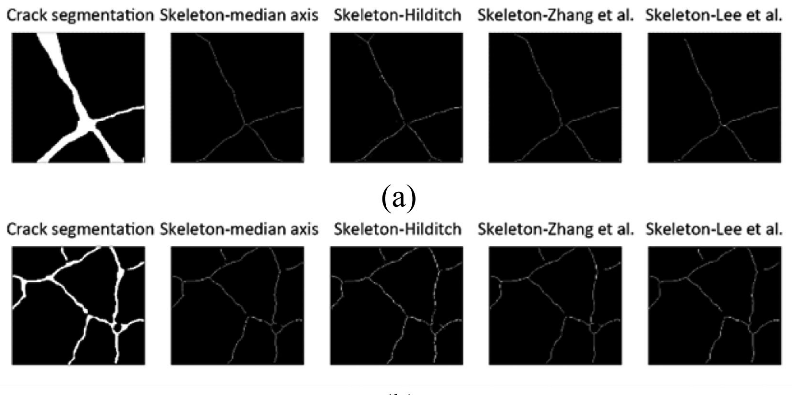


Fig. 13. Comparison of skeletonization algorithms.

to achieve the same iteration. The average training time for one image was round 0.02 seconds and less for prediction.

6 CRACK MORPHOLOGICAL FEATURE EXTRACTION

Once the cracks are detected by FCN, it is possible to extract the morphological aspects from the generated binary images. Such morphological information is valuable for the assessment of existing buildings (Rabinovich et al., 2007). For example, the crack width is one of the remarkable indicators for the durability of concrete structures. Monitoring the width of cracks on the concrete surface, engineers can formulate the cracks, determine the condition of exposure, and determine whether the reinforcement is required to be carried out immediately. To extract such information, the pixel-wise predictions are postprocessed as follows:

As the framework shown in Figure 11, the prediction pixel-wise images are labeled at first, and the detected cracks are divided into several crack segmentations. These segmentations are then thinned to crack skeletons with a width of only one pixel to quantitatively describe the cracks in images, and finally, according to the generated skeletons, the crack morphological features, like width, length, etc.

6.1 Labeling operations

Sometimes, there is more than one crack in an image. It is a necessity to separate individual cracks from background. In this research, the following steps are applied to label each crack in one image: filling small holes, eliminating noisy pixels, labeling individual cracks.

The aim of filling small holes is achieved by closing operation, which can be formulated by

$$closing: \ ((f \oplus \varphi) \ominus \varphi) \tag{11}$$

where $\ominus$ is the morphological operation—erosion, which erodes the boundaries of the object and $\oplus$ is dilation, which increases the coverage of the object. Erosion is actually a replacement of value by local minimum meanwhile dilation is a replacement by local maximum (see Figure 12).

Eliminating isolated noisy pixels is carried out by opening operation, which converses the order of dilation and erosion operations in closing operation. The opening operation can be formulated as:

$$opening: \ ((f \ominus \varphi) \oplus \varphi) \tag{12}$$

For example, as shown in Figure 12, given an image with noisy pixels outside the objective, opening operation first eliminates the noisy pixels by erosion with a specific kernel and then dilates rest pixels to original size; meanwhile given an image with a noisy pixel inside the objective, closing operation applies dilation to warp out the noisy pixel, and then erodes to original size. According to preliminary results of crack recognition, it is found there are not a number of noisy points. Therefore, to avoid potential negative impacts, this research initially applies the closing and opening operation once with a cross structural kernel to transform the pixel-wise predictions into pure crack images.

6.2 Skeletonizing operations

The aim of skeletonizing cracks (also known as thinning digital patterns) is to convert pixel-wise cracks into single-pixel wide representations, providing a clear visualization for the crack's topology. These crack skeletons can be used as a valuable tool for structural

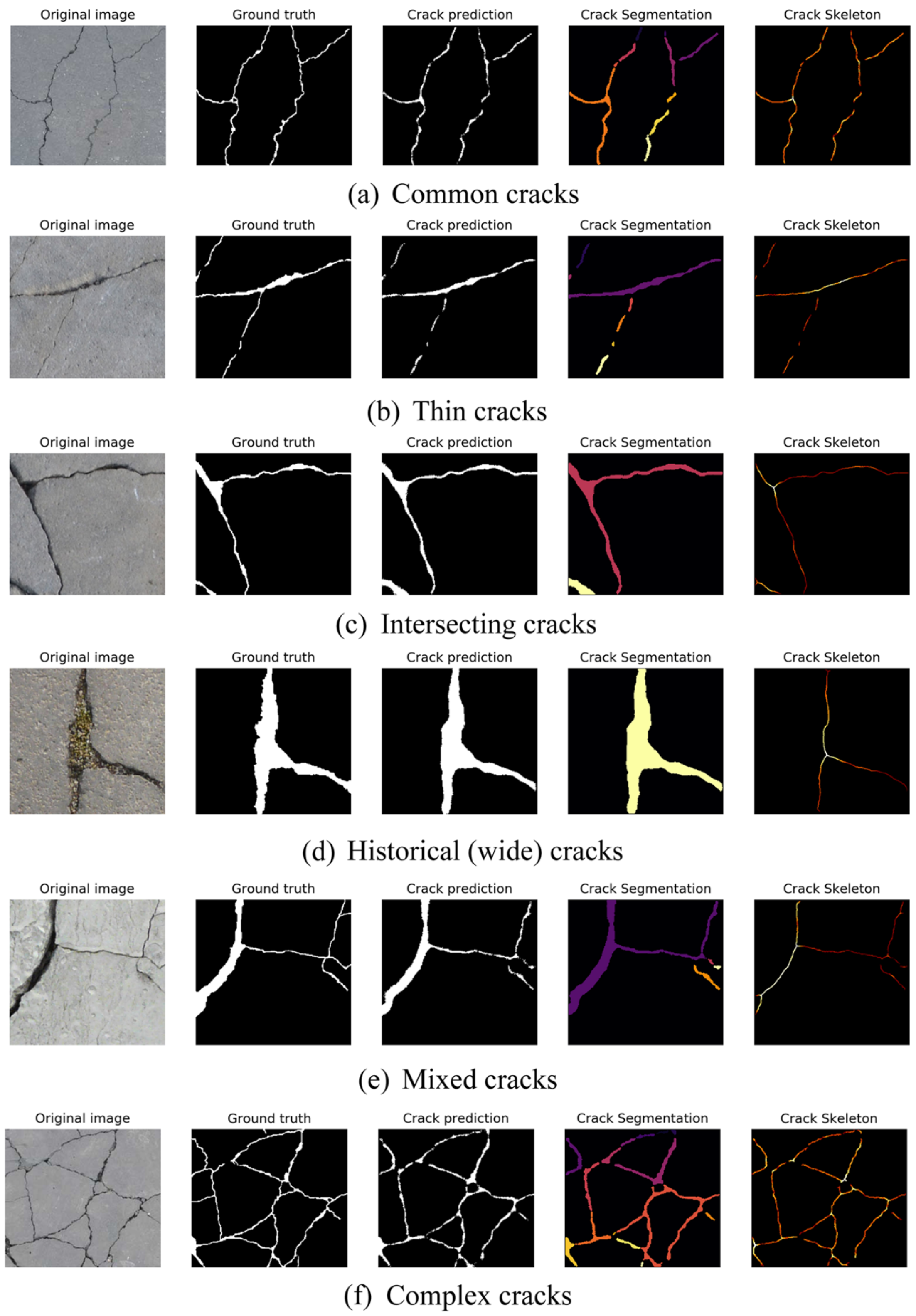


Fig. 14. Typical samples correctly detected by FCN: (a) common cracks, (b) thin cracks, (c) intersecting cracks, (d) historical (wide) cracks, (e) mixed cracks, and (f) complex cracks.

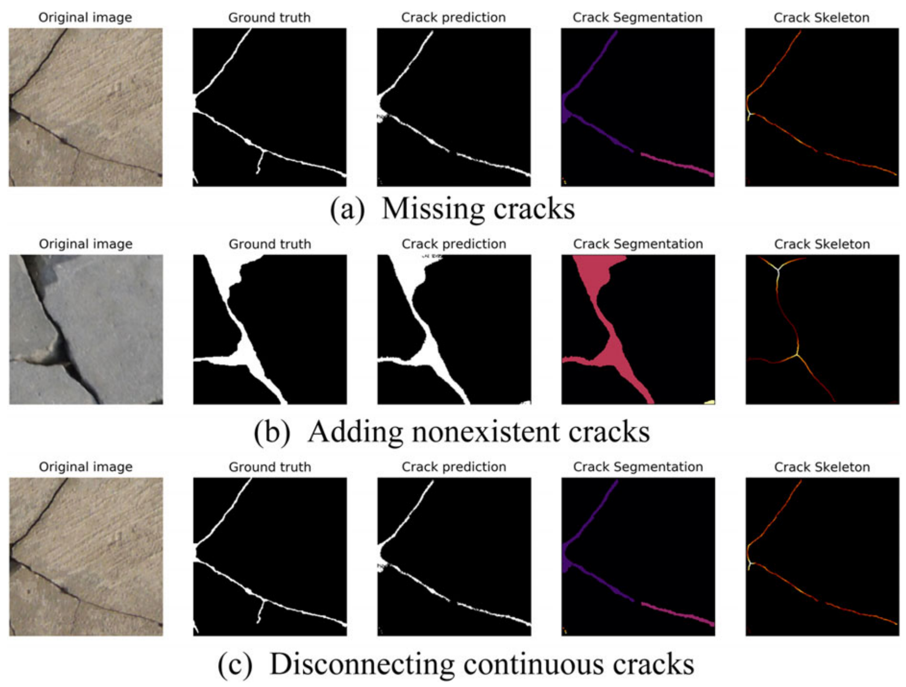


Fig. 15. Samples of detection errors by FCN: (a) missing cracks, (b) adding nonexistent cracks, and (c) disconnecting continuous cracks.

health monitoring. There are several alternatives to skeletonize from the predicted images, such as medial axis approach, Hilditch's algorithm (Hilditch, 1983), and 3D-medial surface/axis thinning algorithm (Zhang and Suen, 1984; Lee et al., 1994). For cracks with a width of higher value, the skeletons yielded by Hilditch's algorithm are blurred with noises, whereas the skeletons generated by Lee et al. are sensitive to image borders that the branches of cracks can not be revealed in Figure 13a. Although the skeletons of complex cracks generated by these algorithms were all acceptable, the Zhang and Suen's and Lee's algorithms were comparatively time consuming because of their iterative operations as shown in Figure 13b. Summing up, this research applies median-axis algorithm to remove the borders of each crack, which was effective for the crack data set in this article. However, managers can preliminarily examine all these algorithms and determine the suitable one for specific tasks (Lee et al., 2013). Similarly, inspectors can apply median-axis algorithm for real-time detection, meanwhile Zhang and Suen's algorithm can be selected to achieve accurate time-delayed recognition.

When cracks are skeletonized by single-pixel wide representations, the length of cracks can be calculated by:

$$L = \int_c f(x, y)\, dl \cong \sum f(x, y)\, dl \tag{13}$$

where $f(x, y)$ is the geometric calibration index and dl represents the finite length of skeleton elements. $f(x, y)$ is defined to calibrate the displacements of pixels in detected images. In this article, the crack data set is assumed to be composed of images without geometric distortion. Therefore, $f(x, y)$ is simplified to be one and the length of cracks can be computed by counting the pixels of skeletons immediately. Meanwhile, the average width of cracks is evaluated by:

$$\bar{D} = \frac{\int_S f^2(x, y)\, dS}{L} \cong \frac{\sum f^2(x, y)\, dS}{\sum f(x, y)\, dl} \tag{14}$$

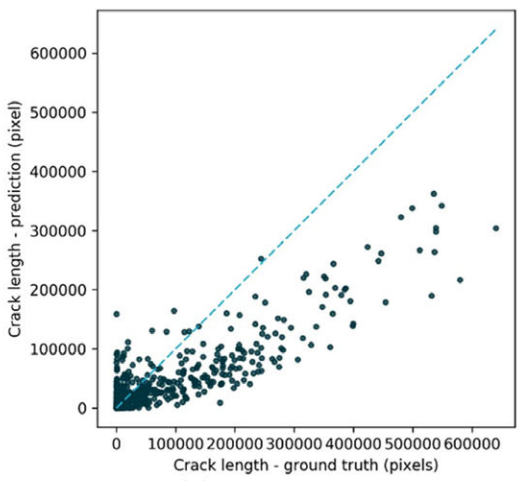
where L is the crack length and dS represents the finite area of crack elements.

Given the image resolution or scale (pixels per metric), the crack length and width in pixels can be transformed into physical length and width in the real world. What's more, the coverage ratio of cracks in images can be quantitatively assessed by computing the proportion of crack pixels to the total number of pixels. Such indicators provide engineers reasonable references to estimate the building structures' working condition.

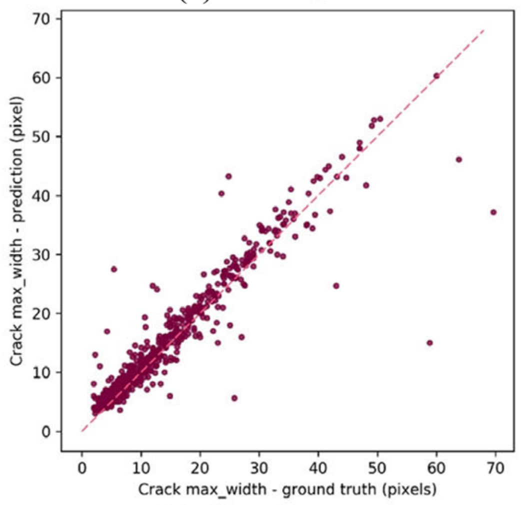
7 RESULTS AND DISCUSSION

7.1 Crack recognition

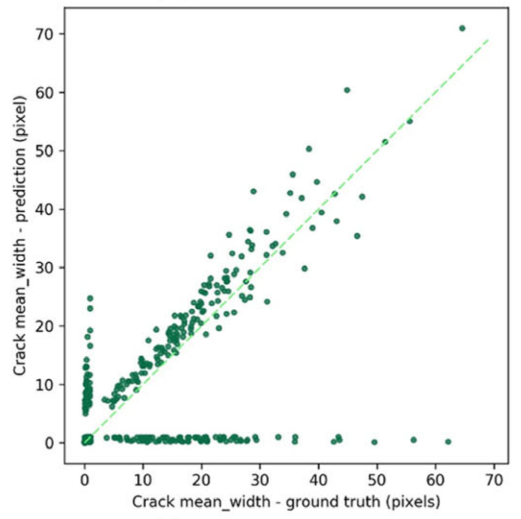
As shown in Figure 14, the first image was the original RGB image; the second one was the ground truth of



(a) Crack length



(b) Crack max width

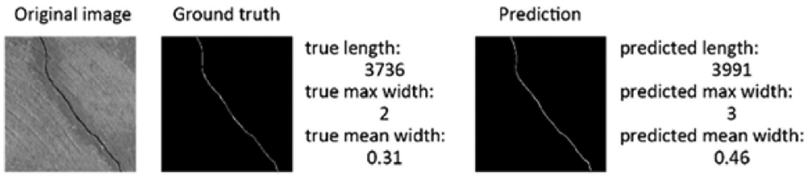


(c) Crack mean width

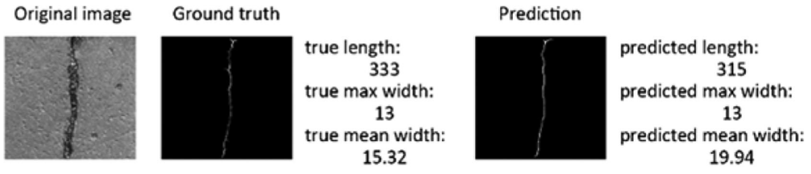
Fig. 16. Comparison of crack measurement indicators: (a) crack length, (b) crack max width, and (c) crack mean width.

cracks. Predicted image generated by FCN was shown in the third column and the crack segmentation in fourth column followed by the ultimate crack skeleton. Here, different colors in crack segmentation described the labels of individual cracks in one image, whereas the colors of crack skeleton suggested the crack width along the single-pixel median axis. The lighter the skeleton appeared, the wider the crack distributed. Five kinds of crack images were sampled, containing (1) common cracks, which could be recognized by direct observation; (2) thin cracks with a small crack width, which may be missed sometimes; (3) intersecting cracks; (4) historical wider cracks that were filled with sand and stones; (5) mixed cracks that had different morphological features; and (6) complex cracks, which contained more than one crack and complex topology. It could be seen from these figures that FCN performed robustly where the width of cracks varies from small to large. No matter how complex the topology was and how many cracks the image contained, FCN was effective, general, and time saving.

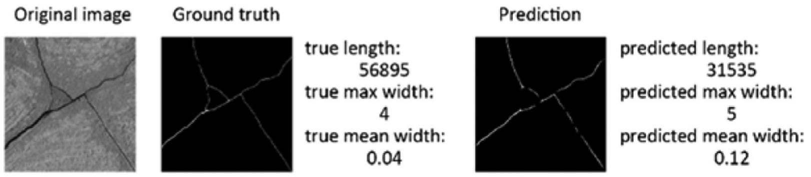
However, it was noticed that there were still failed cases. Errors generated by FCN could be divided into three categories. The first failure happened where the images were blurred or the cracks were very thin. As Figure 15a showed, the thin cracks in the bottom of the image were not detected by FCN and missed in the final skeleton. Such errors were the result of low resolutions that the width of cracks was smaller than one pixel, or the result of complex backgrounds that the color, boundaries of background objects, etc., brought a number of noises into original images. Managers have to provide larger or clearer images, which are taken a bit closer to the crack surfaces. Figure 15b described the second error category as nonexistent cracks at the top of the image that were added to the ultimate crack skeletons. According to the crack predictions, although the predicted accuracy of FCN was sufficient, the generated skeletons were inaccurate. The skeletonizing algorithm contributed a lot to this type of error because it was sensitive to intersections of wider cracks and the image borders where the representation of cracks changes. To illustrate the cracks more accurately, it was suggested to improve the performance of conventional skeletonizing algorithms or develop advanced novel algorithms. The third error category was shown in Figure 15c, which was the consequences of inaccurate crack segmentation. It could be seen that a continuous crack with a bifurcation was divided into three disconnected segmentations. This error was the result of labeling operations. Therefore, the third error category could be alleviated or eliminated by adding reasonable iterations of opening and closing morphological operations.



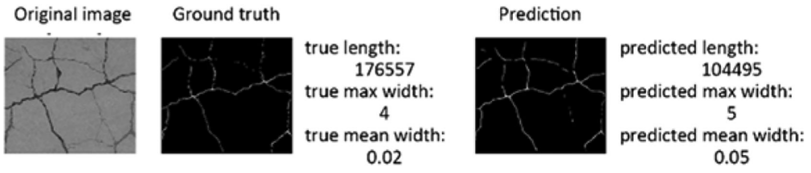
(a) Common cracks



(b) Historical (wide) cracks



(c) Intersecting cracks



(d) Complex cracks

Fig. 17. Samples of crack measurement: (a) common cracks, (b) historical (wide) cracks, (c) intersecting cracks, and (d) complex cracks.

7.2 Crack measurements

From the skeletons generated by the framework proposed in this article, the crack morphological features, such as crack length, max width, and mean width could be calculated. The authors compared the predictions and the ground truth of training data set (776 images); the results were plotted in Figure 16. It can be seen from 16a that the accuracy of crack length was not high as expected; 67.61% of predicted crack length was lower than that of ground truth. The relative error varied from −48.03% where the length of cracks was more than 100,000 pixels, to 177.79% where the length was less. It was observed that the proposed method was prone to overestimate the crack length for small cracks but underestimate for large cracks. One of the possible reasons was the first error that thin cracks were missed by FCN. The other reason was the third error (inaccurate crack segmentations), which treated continuous cracks as discrete cracks leading to the ignorance of connected crack pixels. With respect to crack max width, the proposed framework performed well that the plotted points were close to diagnose line, in terms that the predicted max crack width was close to ground truth (see Figure 16b). The corresponding relative error ranged from −13.27% to 24.01%. In Figure 16c, the case of crack mean width could be obviously divided into four parts: around original point (61.55%), along the horizontal axis (10.97%), along the vertical axis (7.22%), and along the diagnosed line (20.26%). Here, Part 1 and Part 4 were accepted, suggesting that the predicted mean width was accurate. Whereas, Part 2 and Part 3 were the results of bias on predicted crack length, leading to a high relative measurement error ranging from −66.39% to 536.97%. Because the mean crack width was computed by dividing crack size over crack length, smaller predicted crack

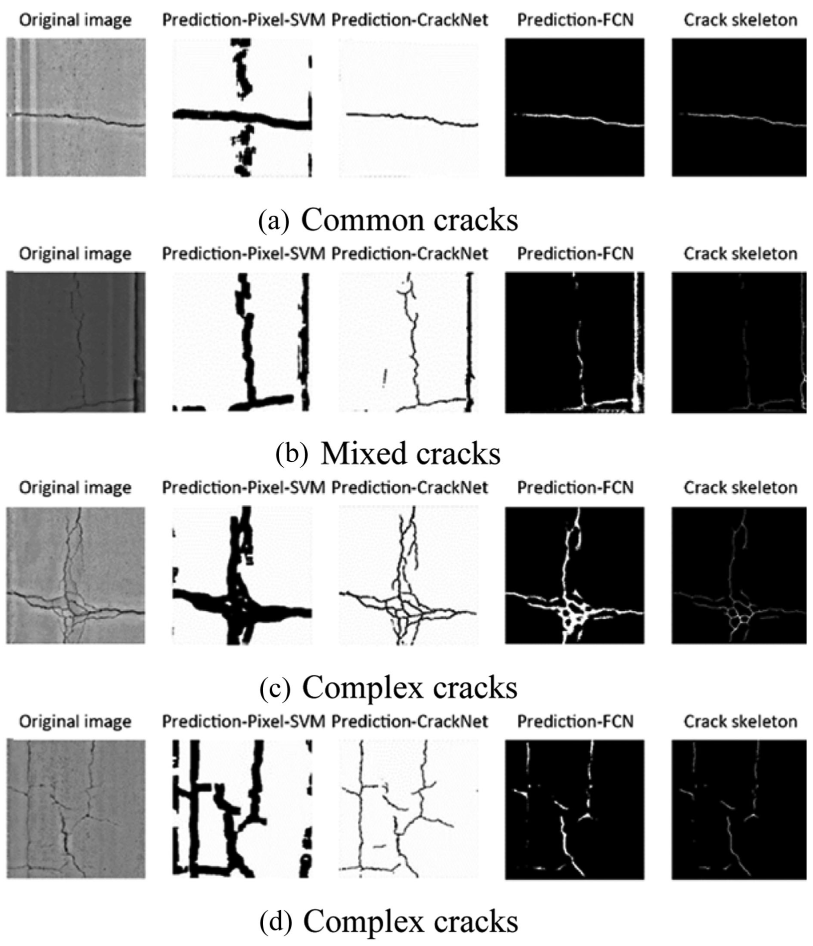


Fig. 18. Comparison between FCN and state-of-the-art crack detections: (a) common cracks, (b) mixed cracks, (c) complex cracks, and (d) complex cracks.

length resulted in Part 2 whereas Part 3 was on the contrary. The proportion of Part 2 was larger than that of Part 3. Such a case also verified the bias on crack length that FCN was prone to shorten the length for complex cracks.

Figure 17 showed some typical samples of crack measurements. The first column contained original images, the second and fifth columns for true skeletons and predicted skeletons separately. These comparisons in Figures 17a and b proved that the proposed method for crack measurement was effective and accurate for common cracks and historical cracks (wide cracks). However, for the intersected cracks in Figure 17c, the predicted length was lower than ground truth because of the missing cracks. What's more, Figure 17d revealed this problem is more serious for complex cracks. The predicted length of crack skeletons was much lower than that of the true cracks, leading to larger mean crack width negatively. This case was contributed to the opening or closing morphological operations, which smoothed the boundaries of cracks when they filled holes or eliminated isolated pixels. Thus, branches of cracks were cut off and the length of cracks decreased. To avoid such a problem, the authors suggested users to adjust the number of opening and closing operations in crack segmentations.

These pixels suggest not only whether the cracks exist but also where the cracks are located. After smoothing and other morphological operations, the predicted images are converted into segmentation images, which are binary images illustrating the pixels associated with cracks. Finally, the crack skeleton can be extracted just by curves with only one-pixel width, which obviously show the location and tendency of surface cracks. At the same time, the crack's length and width can be computed, which are valuable indicators for inspectors to evaluate and monitor the structural health quantitatively.

7.3 Comparison with state-of-the-art crack detections

To compare FCN with existing state-of-the-art DNN for crack recognition, authors collected gray images to examine the performances of Pixel-SVM (Marques and Correia, 2012), CrackNet (Zhang et al. 2017), and the framework proposed in this article. In Figure 18, the images were collected as original gray image, prediction by Pixel_SVM, prediction by CrackNet, prediction by FCN, and the generated crack skeleton. It was obvious that Pixel-SVM overestimated the width of cracks; meanwhile the predictions by CrackNet were always benchmarks. For common cracks in Figure 18a, the cracks were almost recognized correctly by both CrackNet and FCN. However, for thin cracks in Figure 18b, FCN was not able to detect all of the crack branches because of the first type of errors. On the contrary, the CrackNet was so sensitive that fake cracks were also detected. Considering complex cracks showed in Figure 18c, the error of missing cracks was more obvious. Figure 18d revealed a failure case by FCN where the image contained a number of thin cracks. Here, although the accuracy of FCN was not as high as CrackNet, the training time for FCN is less than a percentage of that for CrackNet. In addition, as FCN could deal with gray images and RGB images at the same time, there were potential ways to address the issue of missing cracks: feed FCN with specific training data set for specific tasks, enlarge images to avoid thin cracks within a pixel, etc.

8 CONCLUSION

This research implements a novel deep CNN—fully convolutional neural network (FCN) to detect cracks at pixel level. The network is composed of two parts down-sampling and up-sampling. Down-sampling contains conventional multiple layers, such as convolutional layers, pooling layers, dropout layers; whereas the up-sampling mainly refers to deconvolutional layer. Such integrated structure enables FCN to detect objects at different scales because both local and global features are considered. To examine the performance, a general crack data set is collected as the training and validation samples. During the training process, the best learning rate is identified to be 10^{-4} and the accuracy can be up to 95%. In respect to crack segmentation, the accuracy, precision, recall, and F1 score are 97.96%, 81.73%, 78.97%, and 79.95%. Although the accuracy of FCN is lower than that of existing advanced crack recognition algorithms, such as CrackNet, FCN is an end-to-end model that no postprocessing or preprocessing is needed to detect crack pixels. In addition, the training time has been dramatically decreased to hours rather than days. That is the first major contribution of this research. The second contribution is the crack skeleton proposed in this article for crack measurement. With respect to crack length, the relative measurement error varies from −48.03% to 177.79%; meanwhile, that ranges from −13.27% to 24.01% for crack width. The morphological features of cracks containing but not limited to crack length, max width, and mean width can be extracted from the predicted cracks and crack skeletons. These automatic indices suggest the crack geometric characteristics provide a sufficient way and valuable references for engineers to assess the structural health, which may significantly improve the productivity of inspectors in the future.

However, the results and comparisons expose three types of errors when applying FCN in practice. In the current version, the detection of thin cracks, the intersections, and the cracks close to image borders is inaccurate, the transition from pixels to physical geometric properties, and the real-time processing abilities still have potential to be improved. What's more, if the geometric distortions of crack images could be estimated and corrected, the crack detection would be more accurate and precise. FCN in this paper was only used for crack detection, but it is worthwhile to add multiple training images, such as steel corrosion, concrete voids and pits, leakage, etc. These various samples can generalize the proposed model as well as enhance the stability of FCN. In addition, compared with the highest accuracy and precision in literature, the structure and processing operations of FCN are still required to be updated, such as adding geometric keystone corrections and denoising operations.

To improve the performance and architecture of FCN, the network code, crack image library (https://drive.google.com/open?id=1cplcUBmgHfD82YQTWnn1dssK2Z_xRpjx), and pretrained weights as open sources (https://drive.google.com/open?id=1oX7IO0R_ZkfHwZ_zV4c3_v9_24empwNz) can be downloaded from websites, https://github.com/OnionDoctor/FCN_for_crack_recognition (Yang, 2018), so that engineers are able to apply such technique into their specific tasks. The authors are continuing to improve the model to be more robust and automatic. Adaptive skeletonization algorithms were also developed. These cases do not have a fixed or similar bounding box that the recognition should be at the level of pixel. We welcome all researchers to make contributions to these algorithms.

REFERENCES

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.

Adhikari, R. S., Moselhi, O. & Bagchi, A. (2014), Image-based retrieval of concrete crack properties for bridge inspection, *Automation in Construction*, **39**, 180–94.

Ayenu-Prah, A. & Attoh-Okine, N. (2008), Evaluating pavement cracks with bidimensional empirical mode decomposition, *EURASIP Journal on Advances in Signal Processing*, **2008**(1), 861701–08.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, New York.

Cha, Y.-J., Chen, J. & Büyüköztürk, O. (2017a), Output-only computer vision based damage detection using phase-based optical flow and unscented Kalman filters, *Engineering Structures*, **132**, 300–13.

Cha, Y.-J., Choi, W. & Büyüköztürk, O. (2017b), Deep learning-based crack damage detection using convolutional neural networks, *Computer-Aided Civil and Infrastructure Engineering*, **32**(5), 361–78.

Cha, Y. J., Choi, W., Suh, G., Mahmoudkhani, S. & Büyüköztürk, O. (2017c), Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, *Computer-Aided Civil and Infrastructure Engineering*, **32**(5), 361–78.

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*, **71**, 181–88.

Chen, F.-C., Jahanshahi, M. R., Wu, R.-T. & Joffe, C. (2017a), A texture-based video processing methodology using Bayesian data fusion for autonomous crack detection on metallic surfaces, *Computer-Aided Civil and Infrastructure Engineering*, **32**(4), 271–87.

Chen, J.-H., Su, M.-C., Cao, R., Hsu, S.-C. & Lu, J.-C. (2017b), A self organizing map optimization based image recognition and processing model for bridge crack inspection, *Automation in Construction*, **73**(Supplement C), 58–66.

Cheng, H. D., Chen, J.-R., Glazier, C. & Hu, Y. G. (1999), Novel approach to pavement cracking detection based on fuzzy set theory, *Journal of Computing in Civil Engineering*, **13**(4), 270–80.

Gavilán, M., Balcones, D., Marcos, O., Llorca, D. F., Sotelo, M. A., Parra, I., Ocaña, M., Aliseda, P., Yarza, P. & Amírola, A. (2011), Adaptive road crack detection system by pavement classification, *Sensors*, **11**(10), 9628–57.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016), *Deep Learning*, MIT Press, Cambridge.

Guldur Erkal, B. & Hajjar, J. F. (2017), Laser-based surface damage detection and quantification using predicted surface properties, *Automation in Construction*, **83**(Supplement C), 285–302.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–78.

Hilditch, C. (1983), Comparison of thinning algorithms on a parallel processor, *Image and Vision Computing*, **1**(3), 115–32.

Huang, J., Liu, W. & Sun, X. (2014), A pavement crack detection method combining 2D with 3D information based on Dempster-Shafer theory, *Computer-Aided Civil and Infrastructure Engineering*, **29**(4), 299–13.

Huang, Y. & Xu, B. (2006), Automatic inspection of pavement cracking distress, *Journal of Electronic Imaging*, **15**(1), 013017.

Ito, A., Aoki, Y. & Hashimoto, S. (2002), Accurate extraction and measurement of fine cracks from concrete block surface image, IECON 02, in *IEEE 2002 28th Annual Conference of the Industrial Electronics Society*, pp. 2202–07.

Jahanshahi, M. R., Jazizadeh, F., Masri, S. F. & Becerik-Gerber, B. (2012), Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor, *Journal of Computing in Civil Engineering*, **27**(6), 743–54.

Jahanshahi, M. R. & Masri, S. F. (2012), Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures, *Automation in Construction*, **22**(Supplement C), 567–76.

Jiang, C. & Tsai, Y. J. (2015), Enhanced crack segmentation algorithm using 3D pavement data*, Journal of Computing in Civil Engineering*, **30**(3), 04015050.

Kaseko, M. S. & Ritchie, S. G. (1993), A neural network-based methodology for pavement crack detection and classification, *Transportation Research Part C: Emerging Technologies*, **1**(4), 275–91.

Kirschke, K. & Velinsky, S. (1992), Histogram-based approach for automated pavement-crack sensing, *Journal of Transportation Engineering*, **118**(5), 700–10.

Koziarski, M. & Cyganek, B. (2017), Image recognition with deep neural networks in presence of noise: dealing with and taking advantage of distortions, *Integrated Computer-Aided Engineering*, **24**(4), 337–49.

Lee, B. J. & Lee, H. (2004), Position-invariant neural network for digital pavement crack analysis, *Computer-Aided Civil and Infrastructure Engineering*, **19**(2), 105–18.

Lee, B. Y., Kim, Y. Y., Yi, S.-T. & Kim, J.-K. (2013), Automated image processing technique for detecting and analysing concrete surface cracks, *Structure and Infrastructure Engineering*, **9**(6), 567–77.

Lee, T.-C., Kashyap, R. L. & Chu, C.-N. (1994), Building skeleton models via 3-D medial surface axis thinning algorithms, *CVGIP: Graphical Models and Image Processing*, **56**(6), 462–78.

Li, G., He, S., Ju, Y. & Du, K. (2014), Long-distance precision inspection method for bridge cracks with image processing, *Automation in Construction*, **41**(Supplement C), 83–95.

Li, G., Zhao, X., Du, K., Ru, F. & Zhang, Y. (2017), Recognition and evaluation of bridge cracks with modified active contour model and greedy search-based support vector machine, *Automation in Construction*, **78**(Supplement C), 51–61.

Liao, T. Y. (2017), On-line vehicle routing problems for carbon emissions reduction, *Computer-Aided Civil and Infrastructure Engineering*, **32**, 1047–63.

Lin, Y.-z., Nie, Z.-h. & Ma, H.-w. (2017), Structural damage detection with automatic feature-extraction through deep learning, *Computer-Aided Civil and Infrastructure Engineering*, **32**(12), 1025–46.

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

Long, J., Shelhamer, E. & Darrell, T. (2015), Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–40.

Marques, A. & Correia, P. L. (2012), Automatic road pavement crack detection using SVM. Master of Science Degree dissertation, Electrical and Computer Engineering at Instituto Superior Técnico, Lisbon, Portugal.

Moon, H. & Kim, J. (2011), Intelligent crack detecting algorithm on the concrete crack image using neural network, in *Proceedings of the 28th ISARC*, 1461–67.

Nejad, F. M. & Zakeri, H. (2011), An optimum feature extraction method based on wavelet: radon transform and dynamic neural network for pavement distress classification, *Expert Systems with Applications*, **38**(8), 9442–60.

Nisanth, A. & Mathew, A. (2014), Automated visual inspection on pavement crack detection and characterization, *International Journal of Technology and Engineering System*, **6**(1), 14–20.

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., 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.

Oliveira, H. & Correia, P. L. (2009), Automatic road crack segmentation using entropy and image dynamic thresholding, in *17th European IEEE Signal Processing Conference*, pp. 622–26.

Ortega-Zamorano, F., Jerez, J. M., Gómez, I. & Franco, L. (2017), Layer multiplexing FPGA implementation for deep back-propagation learning, *Integrated Computer-Aided Engineering*, **24**(2), 171–85.

Ouyang, W. & Xu, B. (2013), Pavement cracking measurements using 3D laser-scan images, *Measurement Science and Technology*, **24**(10), 105204.

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.

Rafiei, M. H. & Adeli, H. (2017), A novel machine learning-based algorithm to detect damage in high-rise building structures, *The Structural Design of Tall and Special Buildings*, **26**(18), 1–11.

Rafiei, M. H., Khushefati, W. H., Demirboga, R. & Adeli, H. (2017), Supervised deep restricted Boltzmann machine for estimation of concrete, *ACI Materials Journal*, **114**(2), 237–44.

Santhi, B., Krishnamurthy, G., Siddharth, S. & Ramakrishnan, P. (2012), Automatic detection of cracks in pavements using edge detection operator, *Journal of Theoretical and Applied Information Technology*, **36**(2), 199–205.

Shannon, C. E. (2001), A mathematical theory of communication, *ACM SIGMOBILE Mobile Computing and Communications Review*, **5**(1), 3–55.

Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

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

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

Subirats, P., Dumoulin, J., Legeay, V. & Barba, D. (2006), Automation of pavement surface crack detection using the continuous wavelet transform, *2006 International Conference on Image Processing*, pp. 3037–40.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9.

Wilson, D. R. & Martinez, T. R. (2001), The need for small learning rates on large problems, neural networks, in *Proceedings of IJCNN'01, IEEE International Joint Conference*, pp. 115–19.

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.

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.

Yang, X. (2018), FCN for crack recognition. Available at: https://github.com/OnionDoctor/FCN_for_crack_recognition, accessed August 2018.

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.

Ying, L. & Salari, E. (2010), Beamlet transform-based technique for pavement crack detection and classification, *Computer-Aided Civil and Infrastructure Engineering*, **25**(8), 572–80.

Yu, S.-N., Jang, J.-H. & Han, C.-S. (2007), Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel, *Automation in Construction*, **16**(3), 255–61.

Zalama, E., Gómez-García-Bermejo, J., Medina, R. & Llamas, J. (2014), Road crack detection using visual features extracted by Gabor filters, *Computer-Aided Civil and Infrastructure Engineering*, **29**(5), 342–58.

Zhang, A. & Wang, K. C. (2017), The fast prefix coding algorithm (FPCA) for 3D pavement surface data compression, *Computer-Aided Civil and Infrastructure Engineering*, **32**(3), 173–90.

Zhang, A., Wang, K. C., Ji, R. & Li, Q. J. (2016a), Efficient system of cracking-detection algorithms with 1-mm 3D-surface models and performance measures, *Journal of Computing in Civil Engineering*, **30**(6), 04016020.

Zhang, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q. & Chen, C. (2017), Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network, *Computer-Aided Civil and Infrastructure Engineering*, **32**(10), 805–19.

Zhang, L., Yang, F., Zhang, Y. D. & Zhu, Y. J. (2016b), Road crack detection using deep convolutional neural network, in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3708–12.

Zhang, T. & Suen, C. Y. (1984), A fast parallel algorithm for thinning digital patterns, *Communications of the ACM*, **27**(3), 236–39.

Zou, Q., Cao, Y., Li, Q., Mao, Q. & Wang, S. (2012), Cracktree: automatic crack detection from pavement images, *Pattern Recognition Letters*, **33**(3), 227–38.