Automation in Construction 104 (2019) 129–139

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Computer vision-based concrete crack detection using U-net fully convolutional networks

Zhenqing Liu[a,*], Yiwen Cao[a], Yize Wang[a,*], Wei Wang[b]

[a] School of Civil Engineering and Mechanics, Huazhong University of Science & Technology, Wuhan, Hubei, China
[b] Department of Architecture and Building Engineering, Tokyo Institute of Technology, Yokohama, Kanagawa, Japan

ARTICLE INFO

Keywords:
Crack detection
U-net
FCN
Vision-based
Data-driven

ABSTRACT

For the first time, U-Net is adopted to detect the concrete cracks in the present study. Focal loss function is selected as the evaluation function, and the Adam algorithm is applied for optimization. The trained U-Net is able of identifying the crack locations from the input raw images under various conditions (such as illumination, messy background, width of cracks, etc.) with high effectiveness and robustness. In addition, U-Net based concrete crack detection method proposed in the present study is compared with the DCNN-based method, and U-Net is found to be more elegant than DCNN with more robustness, more effectiveness and more accurate detection. Furthermore, by examining the fundamental parameters representing the performance of the method, the present U-Net is found to reach higher accuracy with smaller training set than the previous FCNs.

1. Introduction

A large number of buildings have gradually approached their design life expectancy; therefore, it is necessary to check the integrity of the structure. At the same time, with the aging of the population and the increasing of labor costs, how to continuously and automatically monitor the structure with the least amount of manpower has become an important research direction [1–3]. In the past, the results relying on manpower inspection were susceptible, not only the results were unreliable, but also time consuming. Moreover, when detecting bridges or tunnels, it is necessary to close them, which not only affects the normal operation of the traffic, but also does not guarantee the safety of the inspectors. In serious cases, dangerous accidents may occur. Therefore, many research groups have proposed structural health monitoring (SHM) technology [4–11]. In order to establish an SHM system, a vibration-based structural system identification method using numerical transformation method has been used. For example, Chatzi et al. proposed a method for detecting the location of defects in a structure using the XFEM-GA algorithm [6], and applied a general elliptical approximation with the smallest measurement error of the sensor to indicate its position. However, due to various uncertainties and uneven environmental impacts, this approach still faces several challenges in monitoring large-scale civil engineering infrastructure [7]. Although there are several SHMs capable of large-scale structural detection [8,9], they require a large number of instruments, such as installing a large number of sensors, integrating data from distributed sources, and compensating for environmental impacts. In addition, from the information of SMH system it is not easy to confirm whether the collected data clearly indicates structural damage, or sensor system failure, noise signal, or a combination of the above.

Therefore, a large number of image processing techniques (IPT) based damage detection methods have been proposed. A significant advantage of IPT is that almost all surface defects can be identifiable. Yeum and Dyke carried out a study using IPT combined with sliding window technology to detect cracks [12]. This research is a good demonstration of the potential of IPT. Although their test examples have many crack-like features, these unnecessary features are effectively removed, and significant cracks are extracted using Frangi filters [13] and Hessian matrix-based edge detectors [14]. However, edge detection is an ill-posed problem [15]. The results are susceptible to noise, which is mainly caused by light and distortion. An effective way to overcome these problems is to implement denoising techniques. Total variation denoising is a well-known technique that reduces the noise of image data and enhances the edge detectability of the image [16]. However, the applications of processing images using prior knowledge are limited because the image data captured in the real world varies greatly. IPT needs to design a reasonable feature extraction algorithm for specific data, but it is very difficult to extract high-level features (i.e. semantic information such as the location and width of cracks) from photos. In addition, it is necessary to design a suitable classifier algorithm for

* Corresponding authors.
E-mail addresses: liuzhenqing@hust.edu.cn (Z. Liu), wangyize0125@hust.edu.cn (Y. Wang).

https://doi.org/10.1016/j.autcon.2019.04.005
Received 25 January 2019; Received in revised form 3 April 2019; Accepted 6 April 2019
Available online 24 April 2019

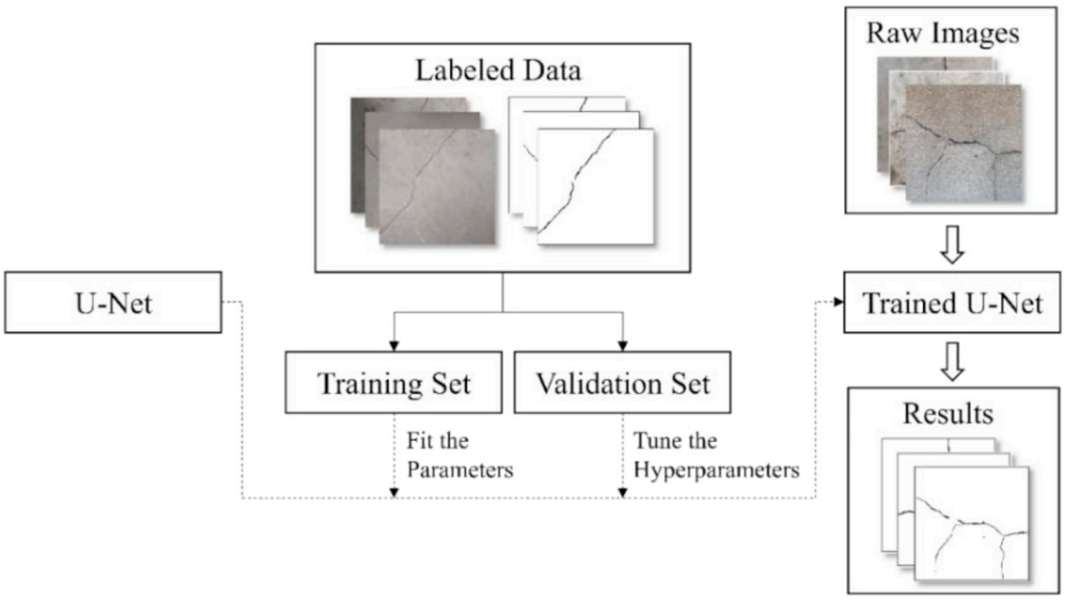

Fig. 1. A schematic of U-Net trained for concrete crack detection.

output which is a very difficult task for the programmers [17].

Deep learning technique is a data-driven method that does not require rules designed manually. The process of building a model only needs to select a suitable network structure (i.e., a series of nested simple mappings), a function to evaluate the model output (i.e., loss function) and a reasonable optimization algorithm. Many researchers have tried to apply deep learning methods in civil engineering, such as, road damage [18] and structure damage [19,20]. These researches of using deep learning to detect damages have performed well on their corresponding tasks. There are two main approaches for the detection of concrete cracks using deep learning technology.

The first approach is based on the method of object detection [21,22]. Object detection is an important task in computer vision and its major task is to find all the objects of interest in the image and determine their position. Typical examples of these methods are NB-CNN proposed by Chen et al. [23] and DDLNet proposed by Li et al. [24]. However, the object detection technology uses a rectangular frame to locate the object, while the crack distribution and shape of the concrete surface are irregular; therefore, the recognition accuracy of these methods is limited.

The second approach is image classification based on region division. That is, selecting a small area of an image each time, and judging whether a crack exists in the selected area. Image classification is also an important task in computer vision. It distinguishes different types of images according to the semantic information. Using image classification technology to classify each small part of the concrete image, and combining the sliding window technology to form the detection of the whole image can achieve good results. One typical method was proposed by Cha et al. (Cha's CNN) [25,26]. Cha's CNN uses a convolutional neural network to classify images and scan the images using different window sliding strategies to complete the detection of the entire image, which can effectively find cracks in the image, but can hardly provide pixel-level concrete crack detections. This is a defect that can hardly be avoided by this type of method, because the accuracy of the detection depends on the fineness of the area division. To achieve higher precision, the area needs to be more finely divided. However, if the region is more finely divided, the information contained therein may not be sufficient to judge whether the crack exists, the classification error will increase, and the accuracy of the detection is reduced.

In order to improve the accuracy, the task of concrete crack detection is treated as a semantic segmentation task. Semantic segmentation is also an important task in computer vision. Its goal is to classify each pixel in the image. If the images semantic segmentation can be quickly achieved, many problems can be solved. For example, the concrete crack detection divides the crack and non-crack regions into two categories to generate pixel-level detection results, which will greatly improve the accuracy of detection. FCN as a deep learning network structure proposed for image semantic segmentation tasks is an ideal way to do the image semantic segmentation. It is called FCN because it replaces the fully connected layer of CNN with a convolution operation.

Recently, Yang et al. [27] applied FCN to do the detection of the concrete crack, where > 800 (224 × 224) images were used to conduct fivefold cross-validation that 80% were used to feed the model and 20% for validation. The precision, recall and F1 score are 81.73%, 78.97% and 79.95%, respectively at epoch 14th, then, overfitting occurs. Dung & Anh [28] trained a FCN model with 500 images of 227 × 227 pixels and obtained a max F1 of 89.6% in the validation set. However, large training sets in FCN by Yang et al. [27] and Dung & Anh [28] require a lot of manpower work before applying the trained model. Therefore it is meaningful to find a method with much smaller training set but with good accuracy.

A network structure U-Net based on FCN is selected as the main component of the method in the present research, firstly created by Ronneberger et al. and applied to biomedical image segmentation [29]. U-Net is a network of encoder-decoder structures, and the encoder extracts features by convolution, pooling, etc., which gradually reduce the input dimension. According to the information provided by the encoder, the decoder repairs the detailed features by multi-scale feature fusion, up-sampling, etc., and obtains higher precision. The success of U-Net in biomedical images segmentation with less training images and more accurate result than CNN motivates the authors to examine the performance of U-Net in crack detection. Compared to cell detection, the extensively varying real-world situations such as lighting and shadow changes bring much difficulty for crack detection. I should be the first publication applying U-Net to detect concrete cracks to the authors' best knowledge.

A schematic of the U-Net in this study is shown in Fig. 1. In the following sections, the structure of U-Net, model evaluation, and model optimization method will be introduced. The method adjusting the hyperparameters according to the performance on the model validation will be also presented. The comparison of the performance of the method implemented in this paper with that of Cha's CNN on the same

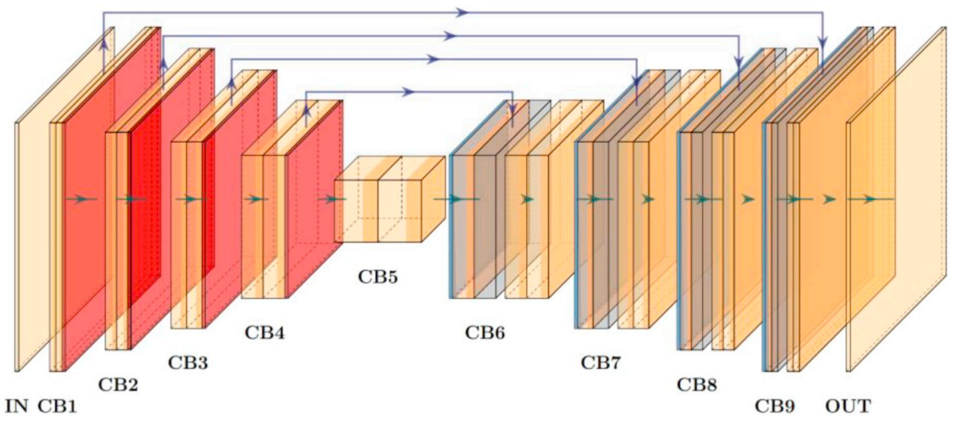

Fig. 2. Overall architecture, IN: input, OUT: output, CB: a set of convolution operation.

test image is carried out as well. In addition, some fundamental performances such as the size of training set and precisions in the FCNs by Yang et al. [27] and Dung & Anh [28] are compared with the present U-Net.

2. U-net network structure

In this section the network structure of U-Net and the various middle layers that appear in it will be introduced. The structure of U-Net is shown in Fig. 2. The quadrangular prisms in the figure represent inputs, outputs, and intermediate layers. The arrows represent operations in the neural network. An image with size 3 × 512 × 512 (i.e., a color image with RGB three channels and a side length of 512 pixels) is input into the constructed neural network. After the calculations, the final output image is 2 × 512 × 512 (i.e., black-and-white image and a side length of 512 pixels). Each pixel position corresponds to a two-dimensional vector indicating the probability that the position is a background or a crack. Table 1 lists the detailed operations for each layer, in which:

1. Conv# indicates convolutional layer.
2. BN# indicates batch normalization layer.
3. ReLU indicates the ReLU activation function.
4. MaxPooling indicates the Max pooling Layer.
5. ConvTrans# indicates the deconvolution layer.
6. Softmax indicates the Softmax activation function.

In addition, Cat means that two images are connected at the channel level as one image. The layers and their corresponding operations are described in detail in the following.

Table 1
Definition and operation of each layer.

Layers		Image size	Operation	Image size	Convolution kernel	Step size	Edge filling
IN	Input	3 × 512 × 512	Conv1 + BN + ReLU	3 × 3 × 3	64	1	1
CB1	L1	64 × 512 × 512	Conv2 + BN + ReLU	64 × 3 × 3	64	1	1
	L2	64 × 512 × 512	MaxPooling	2 × 2	–	2	–
CB2	L3	64 × 256 × 256	Conv3 + BN + ReLU	64 × 3 × 3	128	1	1
	L4	128 × 256 × 256	Conv4 + BN + ReLU	128 × 3 × 3	128	1	1
	L5	128 × 256 × 256	MaxPooling	2 × 2	–	2	–
CB3	L6	128 × 128 × 128	Conv5 + BN + ReLU	128 × 3 × 3	256	1	1
	L7	256 × 128 × 128	Conv6 + BN + ReLU	256 × 3 × 3	256	1	1
	L8	256 × 128 × 128	MaxPooling	2 × 2	–	2	–
CB4	L9	256 × 64 × 64	Conv7 + BN + ReLU	256 × 3 × 3	512	1	1
	L10	512 × 64 × 64	Conv8 + BN + ReLU	512 × 3 × 3	512	1	1
	L11	512 × 64 × 64	MaxPooling	2 × 2	–	2	–
CB5	L12	512 × 32 × 32	Conv9 + BN + ReLU	512 × 3 × 3	1024	1	1
	L13	1024 × 32 × 32	Conv10 + BN + ReLU	1024 × 3 × 3	1024	1	1
	L14	1024 × 32 × 32	ConvTrans1	1024 × 2 × 2	512	2	–
CB6	L15	512 × 64 × 64	Cat L11	–	–	–	–
	L15 + L11	1024 × 64 × 64	Conv11 + BN + ReLU	1024 × 3 × 3	512	1	1
	L16	512 × 64 × 64	Conv12 + BN + ReLU	512 × 3 × 3	512	1	1
	L17	512 × 64 × 64	ConvTrans2	512 × 2 × 2	256	2	–
CB7	L18	256 × 128 × 128	Cat L8	–	–	–	–
	L18 + L8	512 × 128 × 128	Conv13 + BN + ReLU	512 × 3 × 3	256	1	1
	L19	256 × 128 × 128	Conv14 + BN + ReLU	256 × 3 × 3	256	1	1
	L20	256 × 128 × 128	ConvTrans3	256 × 2 × 2	128	2	–
CB8	L21	128 × 256 × 256	Cat L5	–	–	–	–
	L21 + L5	256 × 256 × 256	Conv15 + BN + ReLU	256 × 3 × 3	128	1	1
	L22	128 × 256 × 256	Conv16 + BN + ReLU	128 × 3 × 3	128	1	1
	L23	128 × 256 × 256	ConvTrans4	128 × 2 × 2	64	2	–
CB9	L24	64 × 512 × 512	Cat L2	–	–	–	–
	L24 + L2	128 × 512 × 512	Conv17 + BN + ReLU	128 × 3 × 3	64	1	1
	L25	64 × 512 × 512	Conv18 + BN + ReLU	64 × 3 × 3	128	1	1
	L26	64 × 512 × 512	Conv19 + Softmax	64 × 1 × 1	1	1	0
OUT	output	2 × 512 × 512					

2.1. Architecture

The convolutional layer contains a set of learnable convolution kernels whose height and width need to be manually set. In this study all-zero padding method is adopted in the multiple convolution operations to preserve image boundary information and make it easy to control the size of the output image. From the mathematical point of view, compared with the fully connected layer the role of the convolutional layer is to achieve weight sparseness and weight sharing, reducing the amount of parameters in the neural network.

After obtaining the features through the convolutional layer, the features need to be further processed. Processing of all extracted features will meet computational challenges, and increase the amount of parameters in the network, which will in return increase the risk of overfitting. In this paper the pooling layer [30] selects the maximum value of each 2 × 2 non-overlapping area in each channel of the image and outputs it to the next layer. For an input of size $ch \times h \times w$, after a Max pooling calculation its size becomes $ch \times h/2 \times w/2$.

Deconvolution, also known as transposition convolution, does the opposite of the convolutional layer. In this paper, the image needs to be up-sampled, so the convolution stride is 2. When the deconvolution stride is > 1, it is needed to fill the image in the middle.

Batch normalization layer [31] normalizes the current batch data, which limits the output of each layer to a fixed distribution, speeds up network training, and improves the network adaptability. For a group of training batches with m images $B[x_1, x_2, \ldots x_m]$. The specific description of batch normalization is as follows:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{2}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{4}$$

in which, μ_B and $\sigma_B{}^2$ represents the mean and variance of the current training batch, respectively; $\hat{x}_i$ is the normalized result after subtracting the mean and variance of the current training batach; y_i is the result of scaling and translation, where the scaling factor γ and the translation coefficient β are parameters that need to be learned.

The activation function is a class of functions with nonlinear, monotonic, and differentiable properties. Adding an activation function allows the neural network to fit more nonlinear functions. ReLU [32] is a commonly used activation function in neural networks and is defined as follows:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{5}$$

Because ReLU function has unilaterally suppressed output, large excitatory boundary, and sparse activation of the underlying network, the neural network using ReLU as the activation function has a faster convergence rate when training than the network using Sigmoid function.

The Softmax activation function [33] is often used in classification problems. It compresses the output of each classification into the interval (0; 1), and makes their sum as 1. The transformed output can be regarded as the probability of belonging to each classification. For a c-classification problem, the model belongs to the c-dimensional vector $\vec{z}$, and the probability that each element z_j of $\vec{z}$ corresponds to the j-th classification. For an output prediction classification:

$$p = \arg\max(0, \vec{z}\,) \tag{6}$$

Softmax normalizes the output possibilities and transforms the probabilistic expression that facilitates model parameter optimization. Softmax is defined as follows:

$$\mathrm{Softmax}\,(zj) = \frac{e^{z_j}}{\sum_{k=0}^{c} e^{z_k}} \tag{7}$$

The Sotfmax activation function is different from the hard truncation of the output by the Argmax operation; therefore, it retains a small probability of information and provides more reference for model evaluation.

2.2. Handling class imbalance

A neural network describes the degree of deviation between the predicted and actual values of the model by calculating the loss function during the training process. The larger the difference between the predicted value and the actual value, the larger the loss function, and as a target function that needs to be minimized during training. Choosing a suitable loss function requires consideration of reasonable gradients, choice of machine learning algorithms, and data types. For semantic segmentation tasks, common loss functions include cross entropy loss function, Dice loss function, Lovasz loss function, Focal loss function and the others.

The sample imbalance phenomenon exists in the concrete crack image. Sample imbalance is a problem that many machine learning encounters. If a sample of a certain class in a train set occupies most of the proportion, it is called a simple sample. Although the model can satisfy the simple sample well in the training process (that is, the model can predict the simple sample well and the corresponding loss function value is very low), but due to the large number of simple samples, the contribution to the loss of the entire train set will be very large. This leads to the model not being well trained. It may be that the loss function is stuck in a worse local optimum. There will be a relatively large "uphill" between the best local optimum and the better local optimum. Therefore, the loss function cannot converge to a best result during the training process. In order to solve the sample imbalance problem, the Focal loss function is selected, which is based on the cross entropy loss function. The Focal loss function can be expressed as:

$$\begin{cases} -\alpha \bullet \hat{y}^{\gamma} \bullet \log(1-\hat{y}) \text{ if } y = 0 \\ -\alpha \bullet (1-\hat{y})^{\gamma} \log\,\hat{y}^{\gamma} \text{ if } y = 0 \end{cases} \tag{8}$$

The parameter γ is used to reduce the loss function corresponding to the simple sample, and the parameter α represents the sample imbalance coefficient. For the concrete crack image in this paper, the concrete crack area only accounts for a small proportion (about 1%). As a result, the background becomes a simple sample and may have an absolute dominance over the loss function, which has an impact on training. Applying the Focal loss function can help to overcome this problem.

3. Model training

This section describes the U-Net training process, including train set, verification set, optimization methods, hardware configuration, etc. Due to the complexity of choosing hyperparameters, they must be adjusted after checking the performance of the model on the validation set. All the tasks described in this article are performed on a workstation (CPU: Intel i7 8700K @ 3.7GHz, RAM: 32GB, GPU: Nvidia Geforce 2080ti).

3.1. Training and validation set

Totally 84 images (including 57 images for training the model and 27 for testing the model) were taken at different locations on the campus of Huazhong University of Science and Technology with a resolution of 512 × 512. In order to ensure better generalization performance of the trained models, these images were taken from different

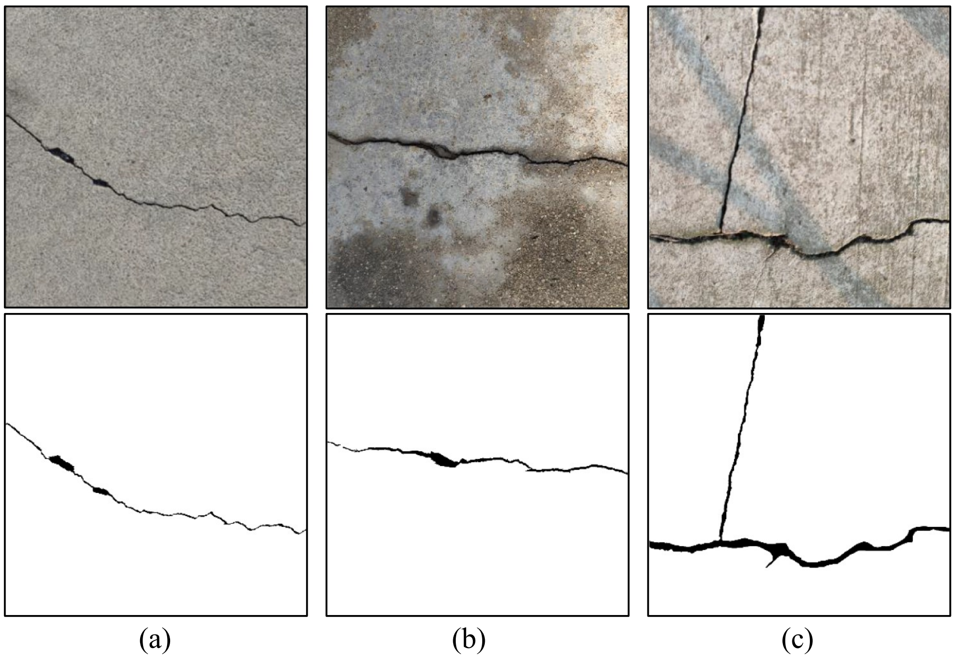

Fig. 3. Examples of the photos for training. (a), (b) and (c): cracks in different conditions. The first row: source images, the second row: labeled images.

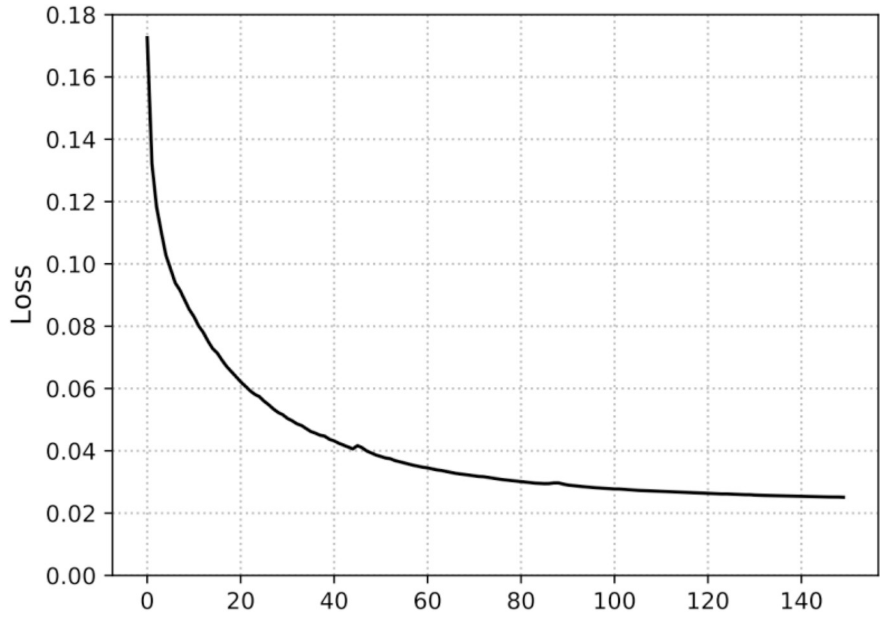

Fig. 4. Change in loss function value during training.

conditions. Each image is manually labeled with a crack location for the training and verification process. Three examples with source images and labeled images are illustrated in Fig. 3(a)–(c).

3.2. Optimization

This paper uses the Adam algorithm to optimize the model. The Adam algorithm is a combination of the Momentum algorithm and the RMSprop algorithm. It is also based on the gradient descent method, but the Adam algorithm has a certain range of parameter changes during each iteration. The parameter will not change sharply due to the large gradient value calculated at a certain time, and the value of the parameter is relatively stable. The initial learning rate of the Adam algorithm in this paper is $\eta = 0.0001$, the exponential decay rate of the first moment estimate is $\beta_1 = 0.9$, and the exponential decay rate of the second moment estimate is $\beta_2 = 0.999$. Each training randomly selects two images as a mini-batch, and an epoch indicates that all images in the train set are trained as a mini-batch. Fig. 4 depicts the change in loss function value during the training process.

3.3. k-Fold cross validation

k-Fold cross validation is usually used to evaluate machine learning models on a limited data sample. $k = 3$ is applied in this paper. The total 57 images are randomly split into 3 groups numbering from 0 to 2. Each time, two groups are selected as training set and the rest one group as validation set. Totally, 38 training images and 19 validation images are adopted.

For the crack detection task in this paper, the commonly used evaluation indicators are: precision (P), recall (R) and F1. The crack

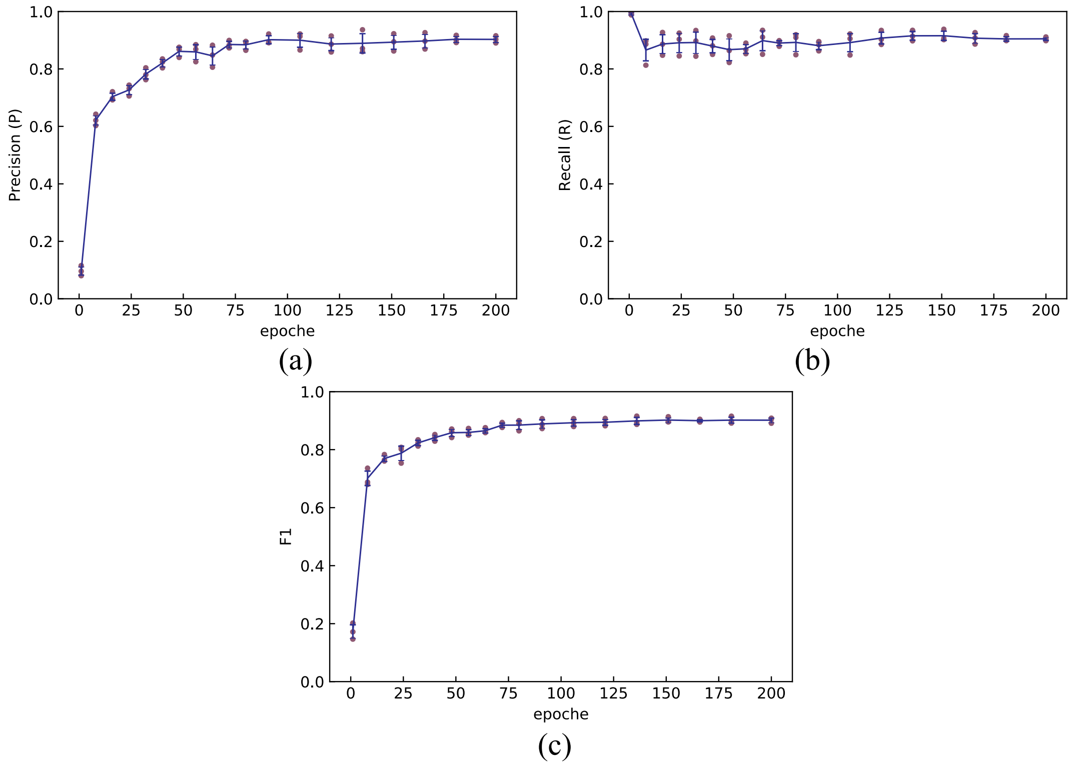

Fig. 5. Changes in indicators during training. (a) P, (b) R, (c) F1. The line is drawn through the average of the indicators for each epoch and the error bars indicate the standard deviation.

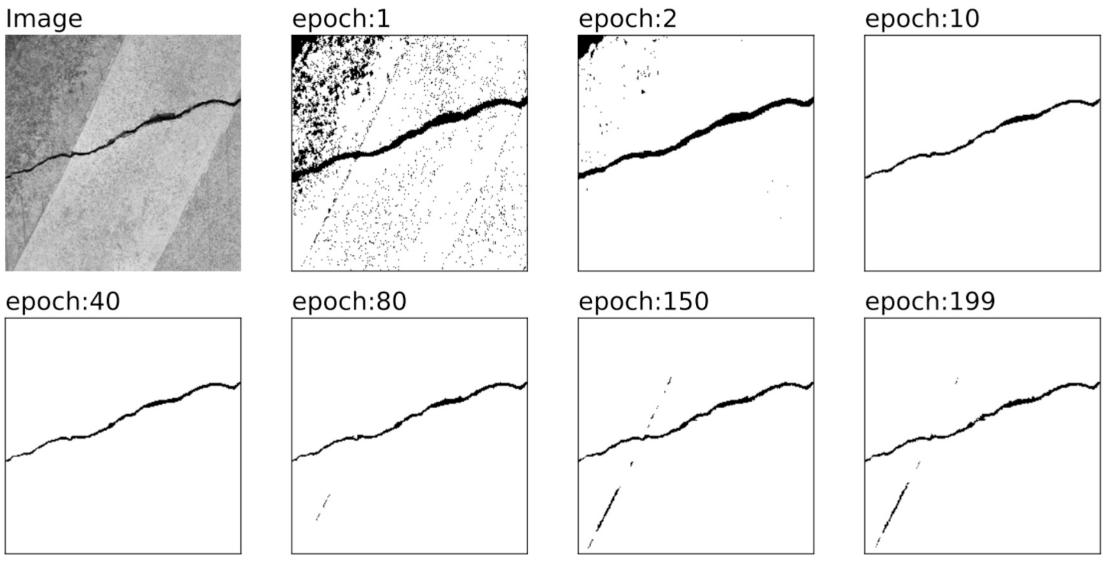

Fig. 6. Output at different epoch.

pixels (i.e. black pixels in images) are defined as positive instances. According to combinations of labeled case and predicted case, pixels are divided into four types: true positive, false positive, true negative and false negative. *TP*, *FP*, *TN*, *FN* represents the summation of pixels in each type, respectively. Then, precision and recall can be defined as follows:

$$\mathrm{P} = \frac{TP}{TP + FP} \tag{9}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{10}$$

F1 is defined basing on the harmonic mean of precision and recall:

$$\mathrm{F1} = \frac{2 \times P \times R}{P + R} \tag{11}$$

Observing the curves in Fig. 5(c), the P, R, F1 on the validation set converges to around 0.90, 0.91, 0.90 after the 80th epoch, respectively. Although the F1 on the train set still has improvement after the 80th epoch, it does not produce better results on the verification set. Overtraining models may increase the risk of overfitting, therefore we select the model trained to the 80th epoch. Fig. 6 is an example of an overfitting that occurs as the training progresses. Table 2 lists the *k*-fold results of validation set at the 80th epoch together with the predict time, TP, FP, TN, FN, precision, recall and F1 values of each image as well as the average of them. On the workstation as mentioned above,

Table 2
k-fold results of validation set at the 80th epoch.

Images	Predict time (s)	TP	FP	TN	FN	Precision	Recall	F1
Fold1								
y_00.png	−0.132	2431	106	259,033	574	0.96	0.81	0.88
y_01.png	−0.101	2071	138	259,265	670	0.94	0.76	0.84
y_02.png	−0.127	2339	69	259,216	520	0.97	0.82	0.89
y_09.png	−0.098	1393	221	260,452	78	0.86	0.95	0.90
y_15.png	−0.147	1993	295	259,566	290	0.87	0.87	0.87
y_17.png	−0.126	2257	323	259,387	177	0.87	0.93	0.90
y_23.png	−0.132	2260	660	258,924	300	0.77	0.88	0.82
y_25.png	−0.142	1315	435	260,361	33	0.75	0.98	0.85
y_27.png	−0.117	2325	252	259,475	92	0.90	0.96	0.93
y_31.png	−0.133	2728	583	258,035	798	0.82	0.77	0.80
y_33.png	−0.144	3598	323	258,041	182	0.92	0.95	0.93
y_38.png	−0.119	3723	275	257,909	237	0.93	0.94	0.94
y_41.png	−0.139	2756	289	258,988	111	0.91	0.96	0.93
y_44.png	−0.132	4581	25	257,217	321	0.99	0.93	0.96
y_46.png	−0.145	3269	249	258,295	331	0.93	0.91	0.92
y_51.png	−0.142	1709	365	260,028	42	0.82	0.98	0.89
y_53.png	−0.107	2256	142	259,666	80	0.94	0.97	0.95
y_53.png	−0.107	2256	142	259,666	80	0.94	0.97	0.95
y_55.png	−0.105	1517	142	260,449	36	0.91	0.98	0.94
Average	−0.126	–	–	–	–	0.90	0.91	0.90
Fold2								
y_03.png	−0.182	1801	108	259,807	428	0.94	0.81	0.87
y_04.png	−0.133	1709	400	259,123	912	0.81	0.65	0.72
y_05.png	−0.100	2039	229	259,511	365	0.90	0.85	0.87
y_06.png	−0.103	2209	147	259,459	329	0.94	0.87	0.90
y_08.png	−0.128	1221	908	259,798	217	0.57	0.85	0.68
y_10.png	−0.096	2015	501	259,198	430	0.80	0.82	0.81
y_12.png	−0.124	2327	160	259,173	484	0.94	0.83	0.88
y_16.png	−0.137	1451	344	259,816	533	0.81	0.73	0.77
y_20.png	−0.151	1947	94	259,756	347	0.95	0.85	0.90
y_22.png	−0.154	5067	92	254,344	2641	0.98	0.66	0.79
y_29.png	−0.137	2130	215	259,600	199	0.91	0.91	0.91
y_34.png	−0.141	2863	272	258,658	351	0.91	0.89	0.90
y_35.png	−0.143	3615	69	258,069	391	0.98	0.90	0.94
y_36.png	−0.128	3411	106	258,281	346	0.97	0.91	0.94
y_37.png	−0.139	2119	206	259,672	147	0.91	0.94	0.92
y_42.png	−0.117	3368	136	258,355	285	0.96	0.92	0.94
y_54.png	−0.123	2874	46	258,538	686	0.98	0.81	0.89
y_55.png	−0.133	1588	472	260,034	50	0.77	0.97	0.86
y_56.png	−0.103	1695	190	260,207	52	0.90	0.97	0.93
Average	−0.130	–	–	–	–	0.89	0.85	0.86
Fold3								
y_07.png	−0.154	1459	141	260,437	107	0.91	0.93	0.92
y_13.png	−0.130	1510	259	260,152	223	0.85	0.87	0.86
y_14.png	−0.125	1848	350	259,882	64	0.84	0.97	0.90
y_19.png	−0.129	1932	487	259,690	35	0.80	0.98	0.88
y_21.png	−0.149	1752	462	259,799	131	0.79	0.93	0.86
y_24.png	−0.119	4666	445	256,453	580	0.91	0.89	0.90
y_26.png	−0.139	1590	498	260,016	40	0.76	0.98	0.86
y_28.png	−0.113	2075	330	259,460	279	0.86	0.88	0.87
y_30.png	−0.138	2250	255	259,059	580	0.90	0.80	0.84
y_32.png	−0.128	2460	815	258,296	573	0.75	0.81	0.78
y_39.png	−0.107	3262	444	258,255	183	0.88	0.95	0.91
y_40.png	−0.104	2777	542	258,730	95	0.84	0.97	0.90
y_43.png	−0.144	3836	306	257,658	344	0.93	0.92	0.92
y_45.png	−0.133	2411	298	259,153	282	0.89	0.90	0.89
y_47.png	−0.133	4351	200	257,038	555	0.96	0.89	0.92
y_48.png	−0.130	3334	276	258,370	164	0.92	0.95	0.94
y_49.png	−0.138	3150	468	258,327	199	0.87	0.94	0.90
y_50.png	−0.107	2372	280	259,409	83	0.89	0.97	0.93
y_52.png	−0.098	1762	258	260,042	82	0.87	0.96	0.91
Average	−0.127	–	–	–	–	0.86	0.92	0.89
Total_average	−0.126	–	–	–	–	0.90	0.91	0.90

the averaged training time of each fold is 16.3 min and the evaluation time is 126 ms per image.

4. Results and discussions

To test the performance of the trained and validated model in the previous section, a set of 27 images that were never used in the train and validation set is used as a test set. The images in the test set are processed using both the methods in this paper (hereinafter referred to as U-Net) and the method proposed by Cha et al. based on CNN (hereinafter referred to as Cha's CNN) to examine the performance of them. In addition, some fundamental performances such as the size of

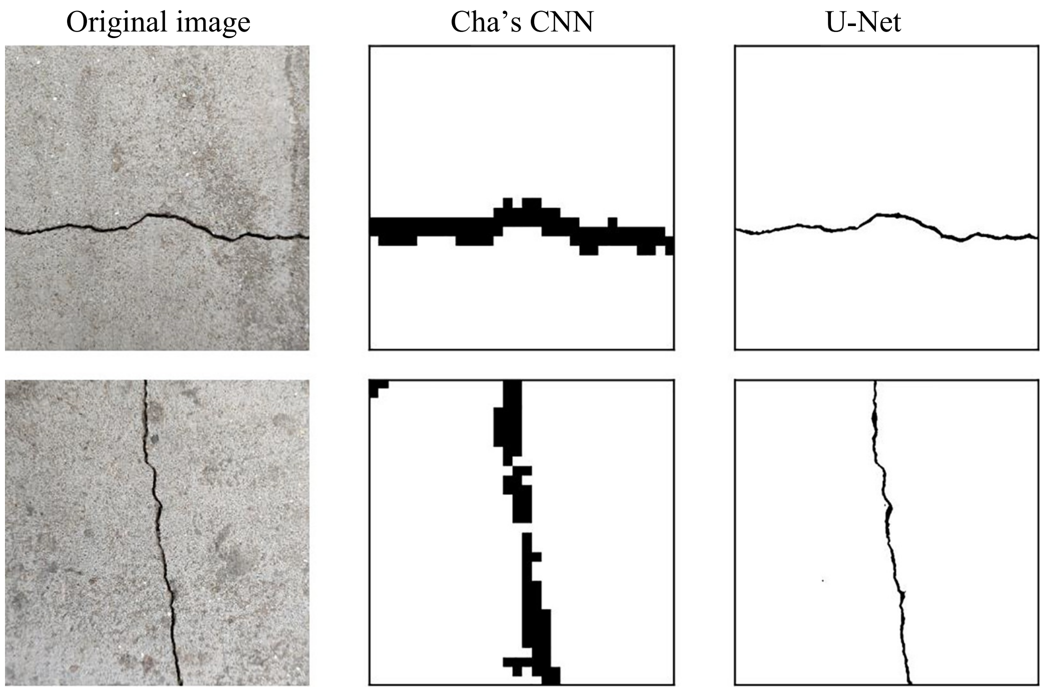

Fig. 7. Results from the two methods for processing two images taken under ideal conditions.

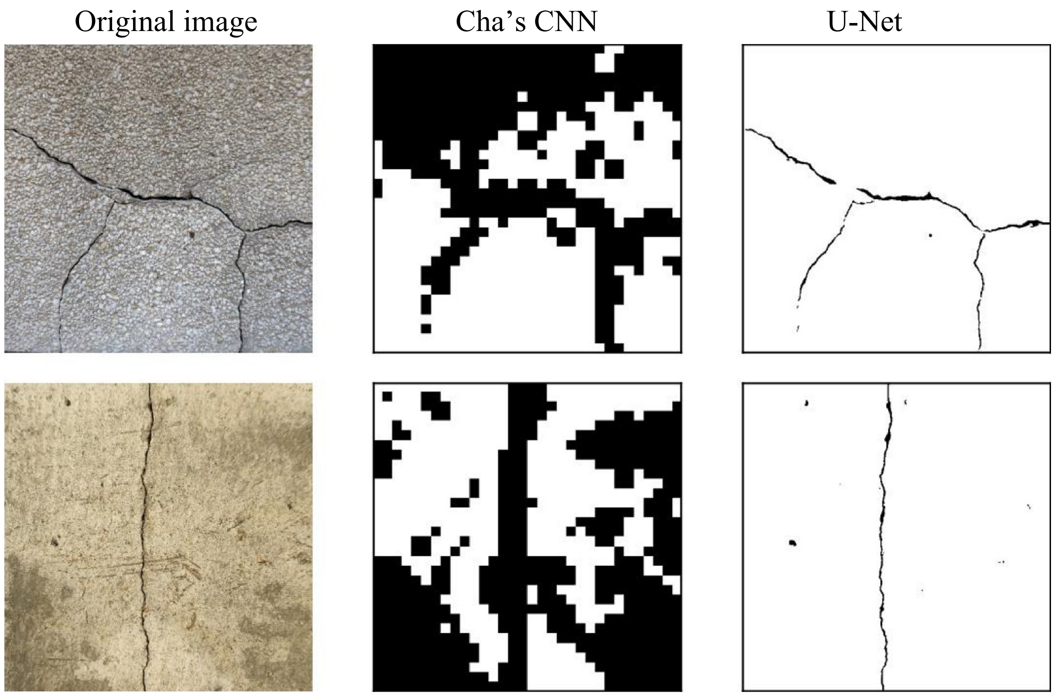

Fig. 8. Results from the two methods for processing two images with rough background interference.

training set and precisions in the FCNs by Yang et al. [27] and Dung & Anh [28] are compared with the present U-Net.

The results from the two methods for processing two images taken under ideal conditions are shown in Fig. 7. Both methods can locate the crack; however, Cha's CNN will induce information loss, so we can only roughly mark a cracked area. On the other hand, U-Net allows for pixel-level detection with greater accuracy.

Fig. 8 shows the performance of the two methods with rough background interference. It can be found that U-Net still show better performance with less background interference, however, Cha's CNN processes the images after dicing, which makes it easy to misjudge scratches and rough background areas as crack areas.

To further test the applicability of the method, the images with thin cracks and under dark conditions which are not used for training is adopted also. It can be found that Cha's CNN model trained without small crack is not able to distinguish the locations of cracks in the image, while U-Net still shows good performance (Fig. 9). And as can be seen from Fig. 10, it is difficult for Cha's CNN to distinguish the crack position under dark conditions, while U-Net is almost unaffected by this condition.

The performance of Cha's CNN in the test of present paper does not reach the performance of the study by Cha et al. [25]. It is because that

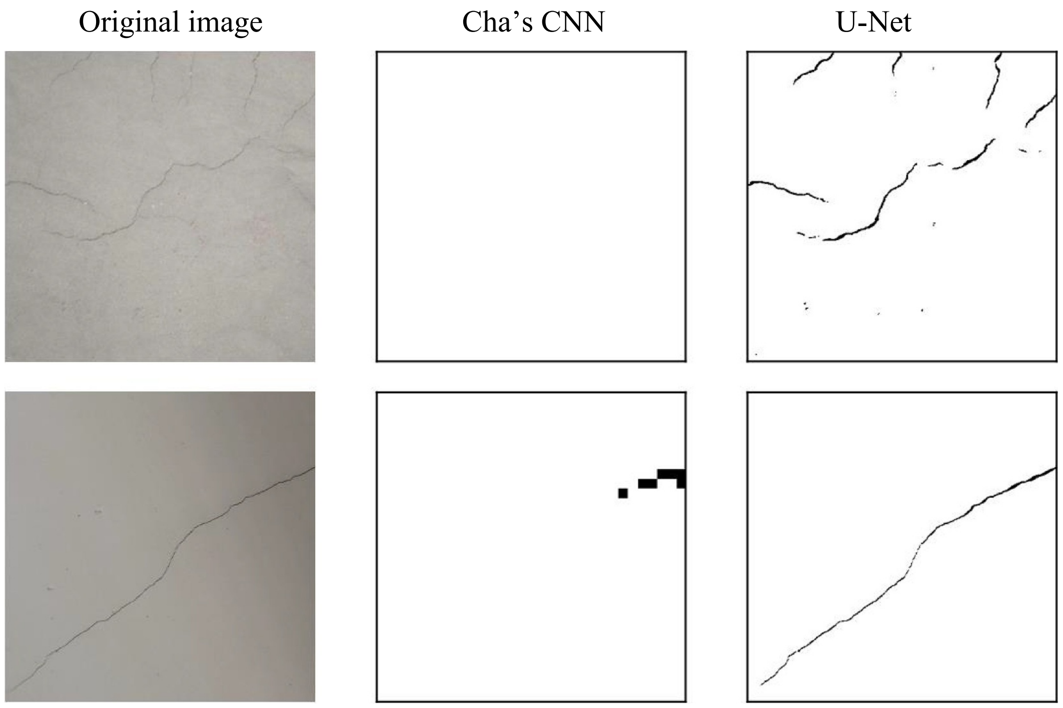

Fig. 9. Results from the two methods for processing two images with thin cracks.

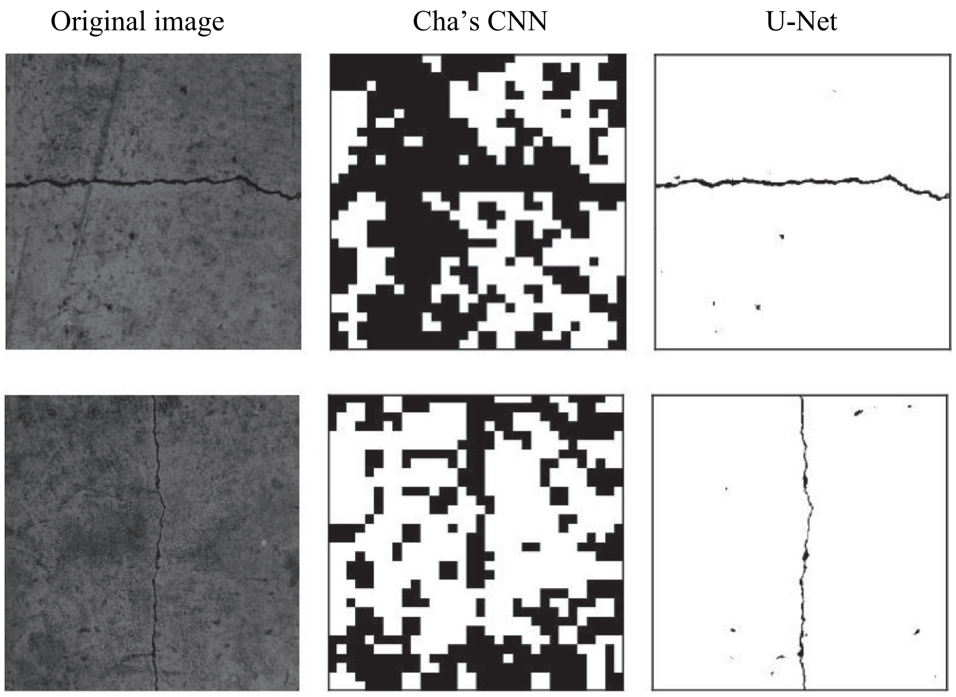

Fig. 10. Results from the two methods for processing two images under dark conditions.

Table 3
Comparisons of FCNs and U-Net.

Methods	Authors	Size of images	Size of training and validation set	Precision	Recall	F1
FCN	Yang et al. [27]	224 × 224	> 800	82%	79%	80%
	Dung & Anh [28]	227 × 227	500	–	–	≤90%
U-Net	Present	512 × 512	57	90%	91%	90%

the data set used in the present study is not the same as the study by Cha et al. [25]. The amount of the data is much smaller than the data set used by Cha et al. [25]. This also indicates that Cha's CNN has a greater requirement for the amount of training data. Cha's CNN can only learn the crack features given in the training set, while the crack features learned by U-Net are not affected by the conditions. U-Net learns the essential features of cracks to a certain extent. As a result, U-Net has a wide range of adaptability in different situations. Therefore, we can say that that the method used in this paper is more accurate and robust than Cha's CNN, and has wider adaptability to the situations that do not appear in the training and verification process.

Table 3 lists the fundamental parameters representing the performance of FCNs by Yang et al. [27] and Dung & Anh [28], and those in the present U-Net. It is clear that the FCN by Yang et al. [27] trained the model using 800 images and the precision, recall and F1 reach 82%, 79% and 80%, respectively, which are all lower than those in the present U-Net. Importantly, the overfitting occurs much earlier (epoch = 14) in the FCN by Yang et al. [27] than that in the present U-Net (epoch = 80). The FCN by Dung & Anh [28] adopted 500 images to train the model which is much larger than that of U-Net in the present work, while the F1 values are almost the same. Therefore, we can safely conclude that the present U-Net can reach high accuracy with smaller training set.

5. Conclusions and discussions

This study focuses on the method applying computer vision technology to identify concrete cracks. According to the characteristics of concrete cracks, it is determined as a semantic segmentation problem in computer vision, and U-Net network structure is used to build a deep learning model for crack detection. The source code and the description of the code can be found in https://data.mendeley.com/submissions/evise/edit/c7cpnw32j6?submission_id=S0926-5805(19)30124-4&token=28a20faa-fbea-4e78-88db-b3881eceb506. The performance of the concrete crack detection method based on U-Net network structure was tested and compared with Cha's CNN. U-Net is found to be more elegant than DCNN with more robustness, more effectiveness and more accurate detections. By examining the fundamental parameters representing the performance of the method, the present U-Net is found to reach high accuracy with smaller training set than the previous FCNs, reducing a lot of manpower work.

Although the method in this article shows good performance, there is still a long way to go for engineering applications. In the experiment, we found several directions that can be tried and improved. Firstly, in order to quickly prototype a model and verify its feasibility, the model in this paper only accepts $3 \times 512 \times 512$ image input. In the case of processing a large number of images, the image needs to be cropped or scaled. But this is not a fundamental solution. How to improve the algorithm to adapt to more input is a direction worth improving. Secondly, there are many artificially adjusted hyperparameters in the implementation of the method. These hyperparameters are derived from the training set and the verification set. Exploring the influence of these hyperparameters on the performance of the model still needs a lot of experimentation. Lastly, maintaining a larger data set and training a more robust model is also a future direction.

References

[1] Y. Yang, C. Yang, C. Huang, Thin crack observation in a reinforced concrete bridge pier test using image processing and analysis, Adv. Eng. Softw. 83 (2015) 99–108, https://doi.org/10.1016/j.advengsoft.2015.02.005.

[2] O. Abudayyeh, A. Bataineh, I. Abdel-Qader, An imaging data model for concrete bridge inspection, Adv. Eng. Softw. 35 (2004) 473–480, https://doi.org/10.1016/j.adven gsoft.2004.06.010.

[3] I. Abdel-Qader, S. Pashaie-Rad, O. Abudayyeh, PCA-based algorithm for unsupervised bridge crack detection, Adv. Eng. Softw. 37 (2006) 771–778, https://doi.org/10.1016/j.advengsoft.2006.06.002.

[4] S. Jeong, R. Hou, J. Lynch, H. Sohn, K. Law, An information modeling framework for bridge monitoring, Adv. Eng. Softw. 114 (2017) 11–31, https://doi.org/10.1016/j.advengsoft.2017.05.009.

[5] D. Feng, M. Feng, Computer vision for SHM of civil infrastructure: from dynamic response measurement to damage detection – a review, Eng. Struct. 156 (2018) 105–117, https://doi.org/10.1016/j.engstruct.2017.11.018.

[6] E. Chatzi, B. Hiriyur, H. Waisman, A. Smyth, Experimental application and enhancement of the XFEM-GA algorithm for the detection of flaws in structures, Comput. Struct. 89 (2011) 556–570, https://doi.org/10.1016/j.compstruc.2010.12.014.

[7] Y. Xia, B. Chen, S. Weng, Y. Ni, Y. Xu, Temperature effect on vibration properties of civil structures: a literature review and case studies, J. Civ. Struct. Heal. Monit. 2 (2012) 29–46, https://doi.org/10.1007/s13349-011-0015-7.

[8] S. Weng, H. Zhu, Y. Xia, L. Mao, Damage detection using the eigenparameter decomposition of substructural flexibility matrix, Mech. Syst. Signal Process. 34 (2013) 19–38, https://doi.org/10.1016/j.ymssp.2012.08.001.

[9] H. Zhu, L. Mao, S. Weng, A sensitivity-based structural damage identification method with unknown input excitation using transmissibility concept, J. Sound Vib. 333 (2014) 7135–7150, https://doi.org/10.1016/j.jsv.2014.08.022.

[10] S. Teidj, A. Khamlichi, A. Driouach, Identification of beam cracks by solution of an inverse problem, Procedia Technol. 22 (2016) 86–93, https://doi.org/10.1016/j.protcy.2016.01.014.

[11] D. Rabinovich, D. Givoli, S. Vigdergauz, XFEM-based crack detection scheme using a genetic algorithm, Int. J. Numer. Methods Eng. 71 (2007) 1051–1080, https://doi.org/10.1002/nme.1975.

[12] C. Yeum, S. Dyke, Vision-based automated crack detection for bridge inspection, Comput. Aided Civ. Inf. Eng. 30 (2015) 759–770, https://doi.org/10.1111/mice.12141.

[13] A. Frangi, W. Niessen, R. Hoogeveen, W. Van, M. Viergever, Model-based quantitation of 3-D magnetic resonance angiographic images, IEEE Trans. Med. Imaging 18 (1999) 946–956, https://doi.org/10.1109/42.811279.

[14] H. Ryu, J. Lee, E. Hwang, L. Jing, H. Lee, W. Choi, A new corner detection method of gray-level image using Hessian matrix, International Forum on Strategic Technology, IEEE, Ulaanbaatar, Mongolia, 2009, https://doi.org/10.1109/IFOST.2007.4798654.

[15] V. Torre, T. Poggio, On edge detection, IEEE Trans. Pattern Anal. Mach. Intell. (2009) 147–163, https://doi.org/10.1109/TPAMI.1986.4767769.

[16] A. Beck, Fast gradient-teased algorithms for constrained total variation image denoising and deblurring problems, IEEE Trans. Image Process. 18 (2009) 2419–2434, https://doi.org/10.1109/TIP.2009.2028250.

[17] K. Agarwal, X. Chen, L. Pan, S. Yeo, Multiple signal classification algorithm for non-destructive imaging of reinforcement bars and empty ducts in circular concrete columns, General Assembly & Scientific Symposium, IEEE, Istanbul, Turkey, 2011, https://doi.org/10.1109/URSIGASS.2011.6050841.

[18] H. Maeda, Y. Sekimoto, T. Seto, Lightweight road manager: Smartphone-based automatic determination of road damage status by deep neural network, ACM Sigspatial International Workshop on Mobile Geographic Information Systems, 2016, https://doi.org/10.1145/3004725.3004729 New York.

[19] X. Fang, H. Luo, J. Tang, Structural damage detection using neural network with learning rate improvement, Comput. Struct. 25 (2005) 2150–2161, https://doi.org/10.1016/j.compstruc.2005.02.029.

[20] X. Wu, J. Ghaboussi, J. Garrett, Use of neural networks in detection of structural damage, Comput. Struct. 42 (1992) 578–581, https://doi.org/10.1016/0045-7949(92)90132-J.

[21] E. Protopapadakis, A. Voulodimos, A. Doulamis, N. Doulamis, T. Stathaki, Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing, Appl. Intell. (2019) 1–14, https://doi.org/10.1007/s10489-018-01396-y.

[22] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, https://doi.org/10.1109/CVPR.2001.990517 Madison, Wisconsin.

[23] F. Chen, M. Jahanshahi, NB-CNN: deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion, IEEE Trans. Ind. Electron. 65 (2018) 4392–4400, https://doi.org/10.1109/TIE.2017.2764844.

[24] R. Li, Y. Yuan, W. Zhang, Y. Yuan, Unified vision-based methodology for simultaneous concrete defect detection and geolocalization, Comput. Aided Civ. Inf. Eng. 33 (2018) 527–544, https://doi.org/10.1111/mice.12351.

[25] Y. Cha, K. You, W. Choi, Vision-based detection of loosened bolts using the Hough transform and support vector machines, Autom. Constr. 71 (2016) 181–188, https://doi.org/10.1016/j.autcon.2016.06.008.

[26] Y. Cha, W. Choi, O. Büyüköztürk, Deep learning-based crack damage detection using convolutional neural networks, Comput. Aided Civ. Inf. Eng. 32 (2017) 361–378, https://doi.org/10.1111/mice.12263.

[27] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, X. Yang, Automatic pixel-level crack detection and measurement using fully convolutional network, Comput. Aided Civ. Inf. Eng. 33 (2018) 1090–1109, https://doi.org/10.1111/mice.12412.

[28] C. Dung, L. Anh, Autonomous concrete crack detection using deep fully convolutional neural network, Autom. Constr. 99 (2019) 52–58, https://doi.org/10.1016/j.

autcon.2018.11.028.

[29] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, International Conference on Medical Image Computing and Computer-assisted Intervention, 2015, https://doi.org/10.1007/978-3-319-24574-4_28 Munich, Germany.

[30] D. Scherer, A. Muller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in: K. Diamantaras, W. Duch, L.S. Iliadis (Eds.), Artificial Neural Networks–ICANN, Springer Verlag, Berlin Heidelberg, 2010, pp. 92–101, https://doi.org/10.1007/978-3-642-15825-4_10.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, A. Rabinovich, Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, (2014), pp. 1–9, https://doi.org/10.1109/CVPR.2015.7298594.

[32] V. Nair, G. Hinton, Rectified linear units improve restricted Boltzmann machines, Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010 978-1-60558-907-7, pp. 807–814 Haifa, Israel, June 21–24.

[33] C. Bishop, Pattern recognition and machine learning. Information Science and Statistics, Springer-Verlag, New York, 2006 (ISBN-13: 978-0387-31073-2).