Special Issue Article

Structural Health Monitoring
1–16

DOI: 10.1177/1475921720940068
journals.sagepub.com/home/shm
SAGE

A research on an improved Unet-based concrete crack detection algorithm

Lingxin Zhang, Junkai Shen and Baijie Zhu

Abstract

Crack is an important indicator for evaluating the damage level of concrete structures. However, traditional crack detection algorithms have complex implementation and weak generalization. The existing crack detection algorithms based on deep learning are mostly window-level algorithms with low pixel precision. In this article, the CrackUnet model based on deep learning is proposed to solve the above problems. First, crack images collected from the lab, earthquake sites, and the Internet are resized, labeled manually, and augmented to make a dataset (1200 subimages with 256 × 256 × 3 resolutions in total). Then, an improved Unet-based method called CrackUnet is proposed for automated pixel-level crack detection. A new loss function named generalized dice loss is adopted to detect cracks more accurately. How the size of the dataset and the depth of the model affect the training time, detecting accuracy, and speed is researched. The proposed methods are evaluated on the test dataset and a previously published dataset. The highest results can reach 91.45%, 88.67%, and 90.04% on test dataset and 98.72%, 92.84%, and 95.44% on CrackForest Dataset for precision, recall, and F1 score, respectively. By comparing the detecting accuracy, the training time, and the information of datasets, CrackUnet model outperform than other methods. Furthermore, six images with complicated noise are used to investigate the robustness and generalization of CrackUnet models.

Keywords

Crack detection, deep learning, fully convolutional neural networks, computer vision, semantic segmentation, structural health monitoring

Introduction

Damage detection is a highly important part of structural health monitoring (SHM) systems. There are two major types of damage detection algorithms: vibration-based algorithms and vision-based algorithms. Vibration-based algorithms have been widely used in SHM of large bridges. However, these algorithms are more applicable to detect global damages of structures rather than local damages.[1–3] Compared to vibration-based algorithms, vision-based algorithms are more applicable to detect local damages. The most primitive vision-based detection algorithm relied on naked eye observations by engineers. However, manual inspection is time-consuming, and its detection accuracy is highly subjective.

During the service of concrete structures, surface cracks are the most significant indicators to assess and evaluate their damage states.[4] The spatial characteristics of cracks vary according to the stress and damage condition. Except for detecting surface cracks on the engineering structures, detecting cracks on the concrete elements in laboratories is also a complex task. In addition, owing to the numerous and high varieties of such cracks, it is an enormous project to detect them manually. Along with the progress of computer vision technology, automatic vision-based methods have been utilized in detecting cracks on surfaces of various structures, such as buildings,[5,6] pavements,[7–9] bridges,[10] and tunnels.[11]

Key Laboratory of Earthquake Engineering and Engineering Vibration, Institute of Engineering Mechanics, China Earthquake Administration, Harbin, China

Corresponding author:
Junkai Shen, Key Laboratory of Earthquake Engineering and Engineering Vibration, Institute of Engineering Mechanics, China Earthquake Administration, No. 29 Xuefu Road, Nangang District, Harbin 150080, Heilongjiang, China.
Email: sjk070825@163.com

Computer-vision-based crack detection algorithms are mainly classified into two types. The first type manually summarizes the image features and then uses computers to detect cracks. Filter-based crack detection methods are the main solutions in early age of crack detection field. Regarding the robustness of different edge-detection algorithms, Abdel-Qader et al.[12] compared four edge-detection algorithms (fast Fourier transform (FFT), fast Haar transform, Sobel, and Canny) and finally found the FFT-based algorithm to be the most reliable. Fujita et al.[13] combined a median filter and a linear filter based on the Hessian matrix to segment cracks from images. Kim et al.[14] investigated image binarization for crack identification and focused on the optimal parameter determination and comparative performance evaluation of five common binarization methods. The shortage of the filter-based method is that it attaches more importance to local features rather than global features. However, crack is a global property of an image. If too much attention is paid to local features, some cracks in the image will be ignored. In order to enhance detection performance, researchers[15–17] started to combine filter-based methods with classification methods, such as artificial neural networks (ANNs) and support vector machines (SVMs). Jahanshahi et al.[15] developed a method that combined the morphological features of cracks with a classifier (a neural network or an SVM) to extract concrete cracks. Li[16] combined an improved active contour model and Canny filter to detect bridge cracks. Li et al.[17] proposed a crack detection algorithm combined with Canny edge detection and an SVM. The combination with other methods strengthens the ability to perceive the global features of the image, which can effectively improve the detection accuracy. However, the surface noise of crack images and their characteristics are extremely complex because they are affected by the illumination conditions, surface irregularities, and surface stains. The above methods are all artificially targeted to suppress one or several types of noises, and it is difficult to distinguish the cracks from shadows because they have similar level of pixel grayscale. The corresponding crack features used in traditional image processing technologies (IPTs) are designed for a given dataset. However, these simple features are difficult to cope with crack images collected in complex environments.

Recently, compared with traditional IPTs, deep learning (DL) models enabled computers to extract image features automatically. When trained by large datasets, DL-based methods are applicable to detect crack more accurately and robustly than before. DL is a branch of machine learning (ML). The earliest DL model that was applied to image problem was LeNet-5, proposed by Lecun et al.[18] in 1998. Different from other ML models, convolutional neural network (CNN) was used as the main architecture of the model. Except CNN, restricted Boltzmann machine (RBM), recurrent neural networks (RNNs), and so on are also powerful for image processing problems. Cha et al.[19] first combined the CNN model and sliding window method to detect cracks on concrete surface. A training dataset which contains 40,000 subimages with a resolution of 256×256 pixels was fed into the CNN model. The validation results showed that the trained CNN model could detect thin cracks with higher accuracy and more robustness than IPTs. Xu et al.[20] proposed an RBM-based crack identification method for extracting cracks from steel box girders with complicated noises. Xu et al.[21] also introduced a fusion CNN to identify cracks in the steel box girder of bridges and applied binary post-process to find out the exact location of crack pixels. Chen and Jahanshami[22] combined a CNN and a naive Bayesian data fusion scheme to enhance the accuracy and robustness of crack detection. Furthermore, Mohtasham Khani et al.[23] demonstrated that applying smoothing method in pre-processing could significantly increase the performance of crack detection model. It is noted that sliding window method was applied to all the above algorithms. This kind of detection methods could only label the type and the location of each window. The size of the window is much larger than the width of cracks, so some edge details of cracks will be lost. To measure cracks more accurately, it is necessary to detect cracks in pixel level. Furthermore, there are many other methods regarding window-level crack detection.[5,10,24]

In order to improve the detection accuracy, pixel-level crack detection methods gradually become the popular research direction. Zhang et al.[7] designed a network architecture based on CNN named CrackNet, which employed 360 feature maps extracted by filters as input and achieved pixel-level crack detection on pavement surface. Even if the proposed method obtained much higher F1 scores than traditional IPT methods, the consumed computing resource of their model is tremendous. Ni et al.[25] constructed dual-scale CNN to detect crack and automatically extract binary crack map. The disadvantage of the above two methods is the complicated implementation process. To simplify the implementation process, a new end-to-end architecture based on fully convolutional neural network (FCNN) was introduced to detect cracks in pixel level. Yang et al.[26] applied FCN to detect concrete cracks in pixel level. Bang et al.[27] proposed a crack detection method based on encoder–decoder neural network to extract crack from road image collected by black-box camera in pixel level. Meanwhile, the influence of different encoder networks on detection accuracy was also

studied. The encoder–decoder-based methods could detect cracks with higher accuracy and easier implementation process. Except CNN-based and FCN-based methods, Zhang et al.[28] proposed an RNN-based pixel-level crack detection method called CrackNet-R. According to the dynamic change of internal memory of RNN, CrackNet-R performed more robust than CrackNet. There are many other researches regarding pixel-level crack detection.[9,29,30]

Some mature models for other tasks are directly migrated to the crack detection tasks. In order to obtain better training results, a larger dataset is usually used in crack detection task. Due to the difficulty of making crack datasets, training a more generalized model with fewer images is a challenge. In addition, the original model often has a deeper structure and more parameters, which may cause performance redundancy in identifying cracks. Therefore, how to select a model with an appropriate depth based on the difficulty of the detection task is another challenge.

In this article, a Unet-based[31] method called CrackUnet which applies a new loss function is introduced to detect cracks in pixel level. Unet is based on encoder–decoder structure, and the encoder path uses convolution, pooling, and so on to extract crack features. The high-level features will be restored by decoder path. In addition, the skip connection between the high-level features and the shallow features could fusion the multiscale features to improve the segmentation accuracy. Although the original Unet model succeeds in cell detection, it may not adapt to detect cracks in extensively complicated situations. In order to select the most suitable model, we focus on improving the model and studying how the depths of the model and the size of the dataset influence the performance of the crack detection method.

The main contribution of this article includes the following: (1) study how the size of the dataset and the depth of the model influence the performance of the crack detection method; (2) the proposed method outperforms all the state-of-the-art algorithms on previous public dataset; and (3) compared with other methods, the proposed method needs less training time and has higher detection accuracy.

This article is organized as follows. The "Framework of the detection algorithm" section explains the framework of the method, including image pre-processing and the architecture of CrackUnet. The "Experiment implementation and results" section shows the training process and test results. In addition, the effect of the size of the dataset and the depth of the model is discussed. Finally, the comparison between the proposed method and other methods is discussed too. The "Conclusion" section concludes the article.

Framework of the detection algorithm

The framework of the detection algorithm includes three modules, as shown in Figure 1. In the first module, the original images are pre-processed, and a crack dataset is created. In the second module, the training dataset is fed into the CrackUnet model for training the parameters, and the validation dataset is fed to fine-tune the hyperparameters. Finally, the test dataset will be sent into the optimized CrackUnet model to evaluate the performance of the trained CrackUnet model.

Image pre-processing

The original crack images are collected from the Internet, the lab, and literatures.[26,32] Owing to the complexity of the collecting source, the size is not standard, and the image quality is also different. The original images need to be pre-processed before they are fed into CrackUnet for training. Pre-processing operations mainly include as follows:

1. *Uniform size*: Large-size images collected from the laboratory and earthquake sites are cropped by a sliding window of 256 × 256 pixels. For the images collected from the Internet and the reference, the size is unified to 256 × 256 pixels by a bilinear interpolation. In total, 120 images with a resolution of 256 × 256 are obtained.
2. *Labeling of cracks*: The crack area is labeled with the manual labeling tool Labelme.[33] The schematic diagram of the manual label annotation is presented in Figure 2. The original image is presented in Figure 2(a), and the labeled result output by Labelme is presented in Figure 2(b). To reduce the impact of the color channel on model training, the color channel in the labeled image is removed. Then they will be converted into an 8-bit grayscale image with a grayscale of 255 for crack pixels and a grayscale of 0 for background pixels, as Figure 2(c) displays.
3. *Data augmentation*: After the first two steps of pre-processing, there are only 120 crack images, which tend to cause overfitting phenomenon. Therefore, data augmentation is necessary for the original dataset. The data augmentation operation in this study refers to the study by Wang and Perez,[34] which included rotating, twisting, and stretching operations; 360, 480, and 240 new crack images are obtained by rotating, twisting, and stretching the original images, respectively. To synchronize a label and an image, label images are inserted into the original images as one of the channels of the original images. Then an augmentation operation

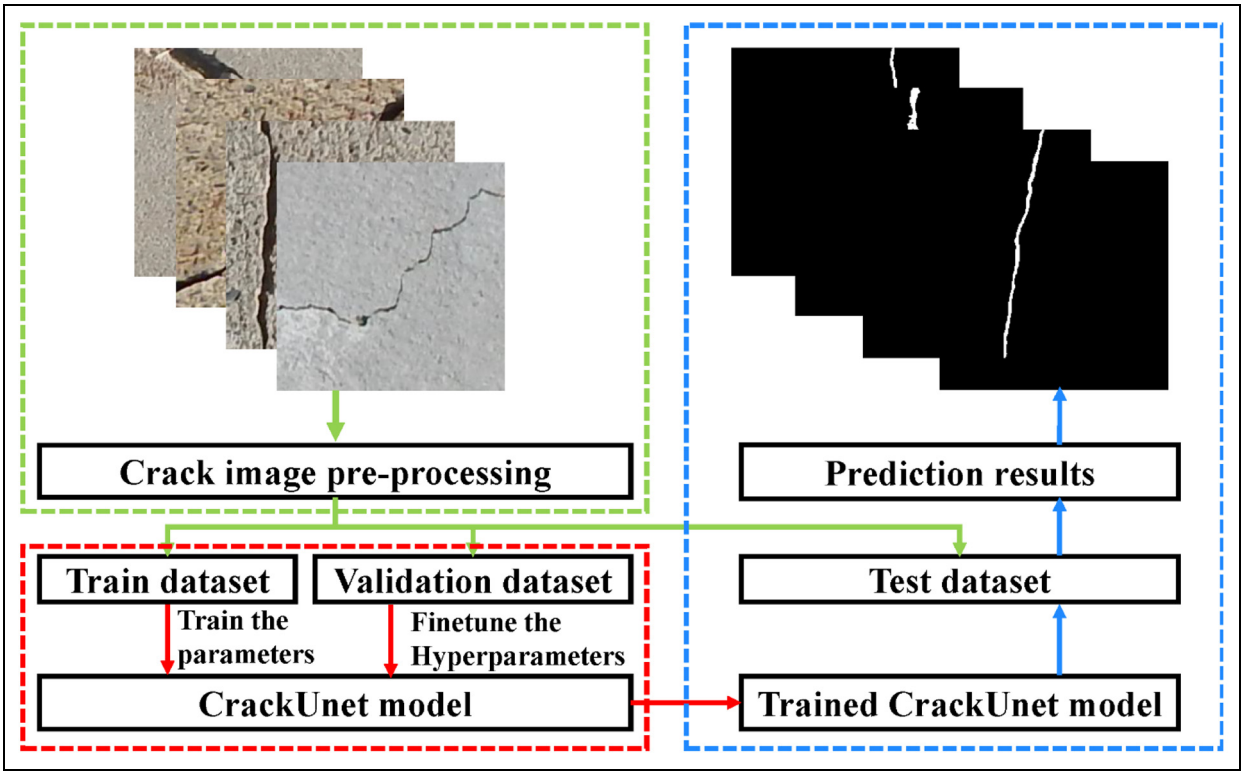

Figure 1. Framework of the CrackUnet detection method.

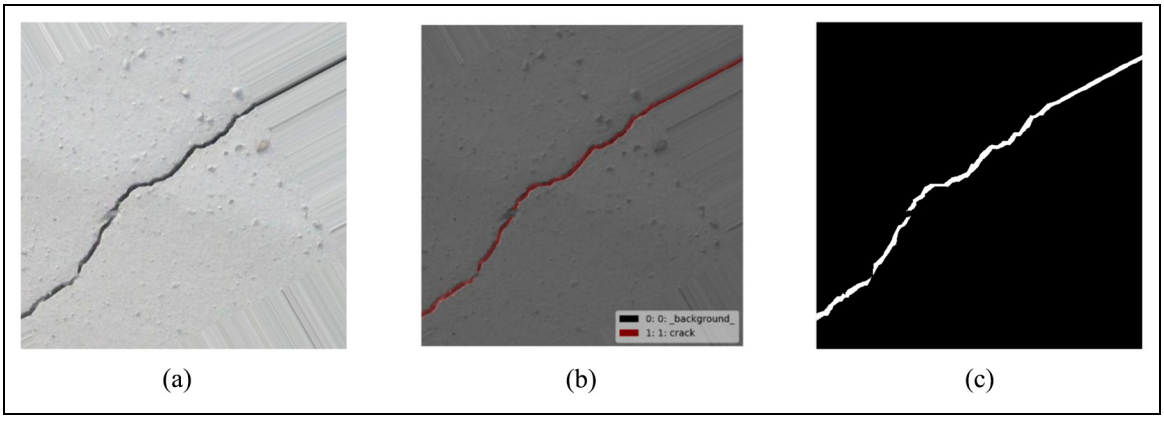

Figure 2. Demonstration of the manual label annotation: (a) original image, (b) labeled image, and (c) removed color channel.

is performed on the original images, including the label images. The augmented images are presented in Figure 3.

After the pre-processing, 1200 images in total were obtained. To assess the generalization ability of the proposed approach, 800 images were randomly selected as the training datasets to train CrackUnet, and 200 images were used as the validation set to fine-tune the hyperparameters. Finally, the remaining 200 images were used as the test dataset to evaluate the generalization ability of the model.

Architecture of CrackUnet

Overall architecture. Inspired by the Unet model proposed in the literature,[31] an improved CrackUnet is proposed to detect cracks with higher accuracy and faster speed in pixel level. Considering the original Unet model was proposed to detect cells, it may not

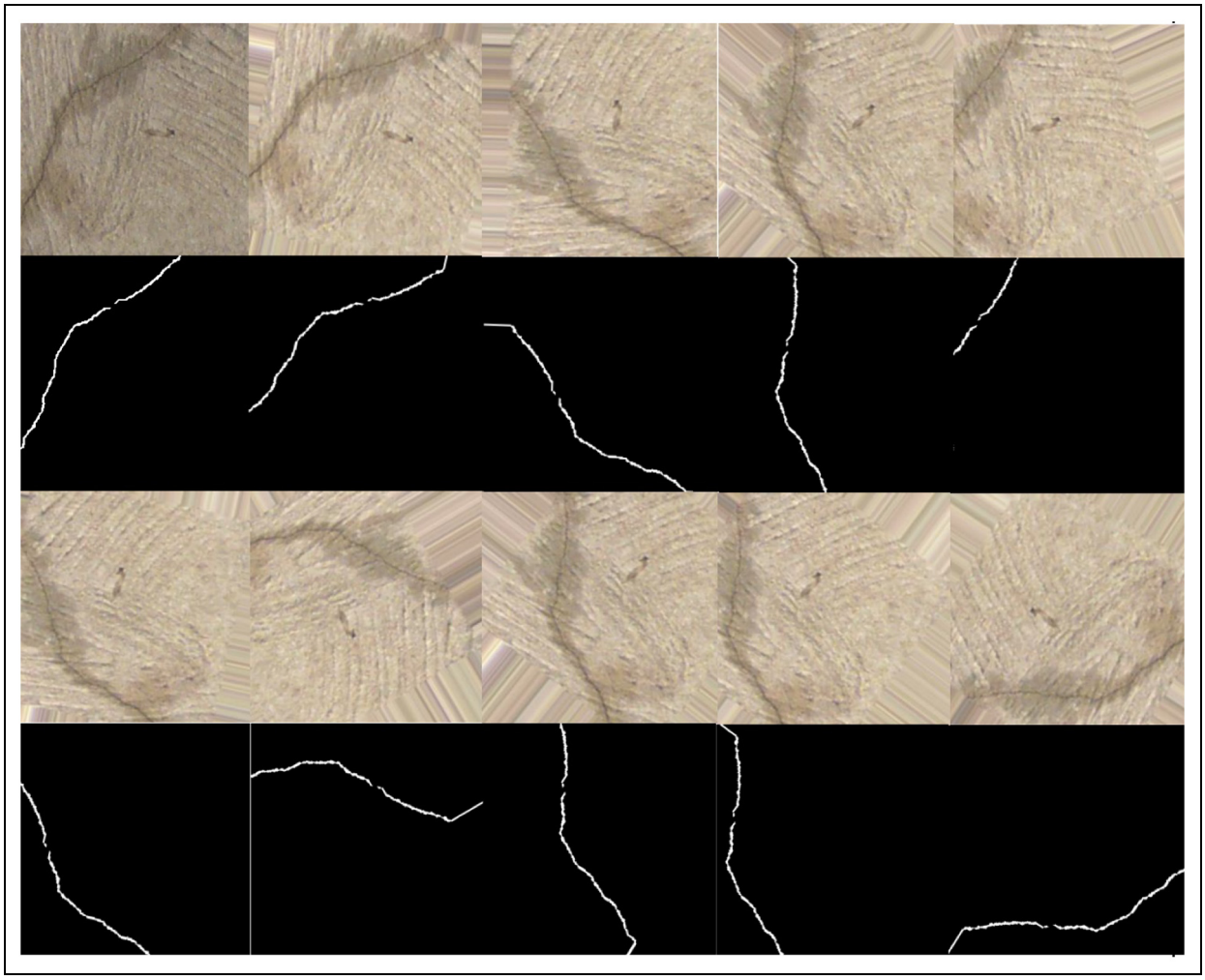

Figure 3. The example results of data augmentation.

adapt to detect cracks. Moreover, it is unknown whether it is necessary to use such a deep model like Unet to detect cracks. To choose the best model with high detection accuracy and speed, CrackUnet7, CrackUnet11, CrackUnet15, and CrackUnet19 are designed, as Figure 4 shows. The advantages of these four models compared with other CNN-based models include the following: (1) the symmetrical structure guarantees the same size of input images and output images, (2) skip connections can fusion the high-level features and shallow features so that the detection accuracy will be higher, and (3) decoder path can restore the spatial information of the cracks to detect cracks in pixel level.

In Figure 4, the overall architecture of four CrackUnet models with different depths is presented. The number behind CrackUnet is the number of convolutional layers. Table 1 lists the layers and parameters of CrackUnet15. As Table 1 summarizes, the first layer (Input) contains images with dimensions of $256 \times 256 \times 3$. Input images are fed into the encoding path containing four encode units. Each of these units includes two convolutional layers (Conv), two activation layers (ReLU, Rectified Linear Unit), two batch normalization (BN) layers, and one dropout layer (Drop). During the down-sampling process, there are maximum pooling layers between each two units. Next, the high-level features are delivered to the decoding path. In the decoding path, there is an up-sampling operation between each two units. Simultaneously, the features extracted by the down-sampling layer are directly passed to the up-sampling layer by concatenation. Following the decoding path is a convolutional layer that uses two 1×1 convolution kernels for the

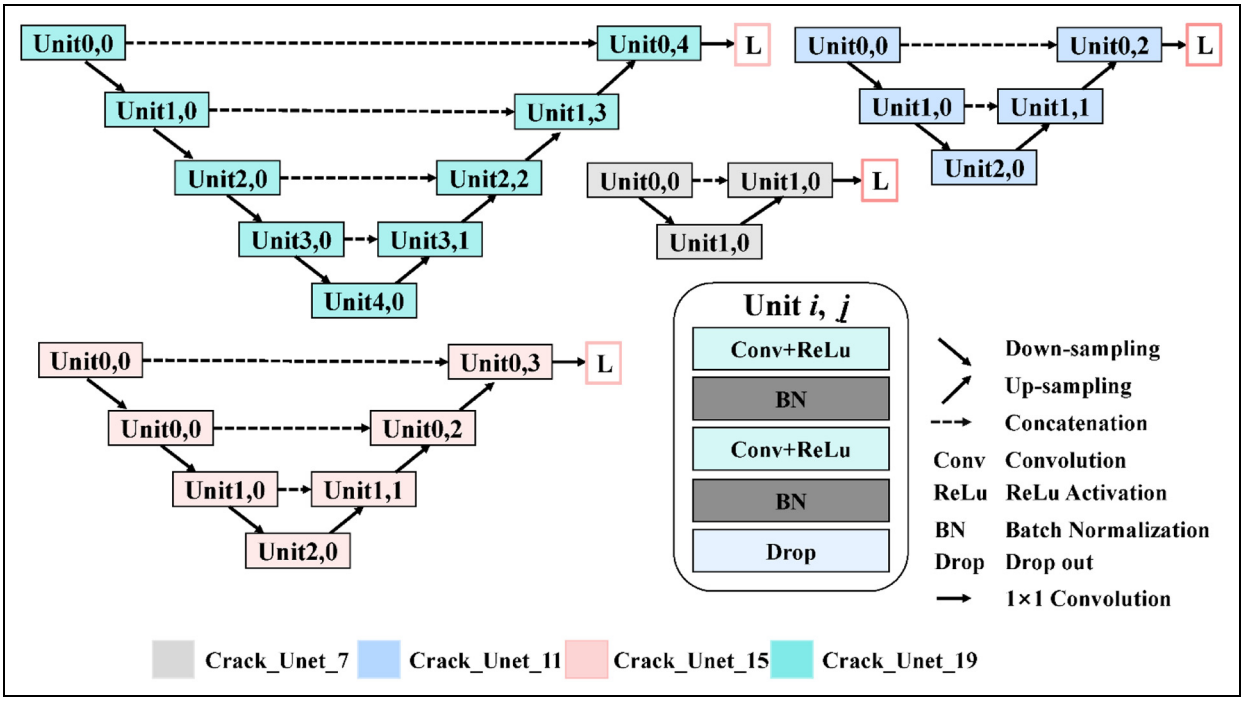

Figure 4. Architecture of CrackUnet.

purpose of dimensionality reduction. Finally, the Softmax function is used to normalize the feature map. The final output image is $256 \times 256 \times 1$.

Convolutional layer. In the convolutional layer, several convolution filters slide across the input tensor with a given stride. When the filter slides to the corresponding position, the convolution kernel and its tensor within the receptive field are convoluted, that is, they are multiplied element-by-element. Then all the multiplied values are summed up, and finally, a bias term is added. To keep the input and output size consistent during the convolutional operation, the same padding (i.e. the fill value is the same as the edge value) is always applied to the input tensors. According to Simonyan and Zisserman,[35] increasing the number of convolution kernels is more effective for extracting the characteristics of the input feature map than increasing the receptive field. Therefore, a 3×3 convolution kernel is uniformly adopted, and the sliding stride is 1.

Down-sampling layer. The down-sampling operation is the key layer in the encoding path, and its main function is to decrease the size of the input tensor. The down-sampling operation includes maximum pooling and average pooling. Maximum pooling only extracts the maximum value of the elements in the receptive field, and average pooling calculates the mean value. Maximum pooling has been proved to better preserve the features in the input feature map during the down-sampling process.[36] In this article, maximum pooling filter which could select the maximum value of each 2×2 receptive field is used as the down-sampling operation.

Up-sampling layer. The up-sampling layer is the key layer in the decoding path, and its main function is to precisely restore and locate features. In this study, the interpolation and convolution methods are adopted to restore the high-level features, as Figure 5 presents. To keep the constant between the input and output sizes, the same padding (i.e. the fill value is the same as the edge value) is always applied to the input tensors. Owing to the up-sampling layers, the model can segment the crack images in pixel level.

Subsidiary layers. Activation function aims to add nonlinearity to the neural network. ReLU activation function is a commonly used activation function to strengthen nonlinear properties.[37] The gradient of ReLU will not vanish even if the input value is abnormally large. Besides, their derivation process is simpler, and it will accelerate the optimization of the neural network. It is defined as follows

$$\mathrm{ReLU}(x) = \max(0, x) \tag{1}$$

Table 1. Architecture of CrackUnet15.

Unit	Layer	Filter_Size (H,W)	Kernels	Stride (H,W)	Output_Size (H,W,D)
	Input_image				(256,256,3)
Unit0,0	Conv1_1 + ReLu + BN	(3,3)	64	(1,1)	(256,256,64)
	Conv1_2 + ReLu + BN	(3,3)	64	(1,1)	(256,256,64)
	Dropout_1				
Down-sampling	Pool1(Max)	(2,2)		(2,2)	(128,128,64)
Unit1,0	Conv2_1 + ReLu + BN	(3,3)	128	(1,1)	(128,128,128)
	Conv2_2 + ReLu + BN	(3,3)	128	(1,1)	(128,128,128)
	Dropout_2				
Down-sampling	Pool2(Max)	(2,2)		(2,2)	(64,64,128)
Unit2,0	Conv3_1 + ReLu + BN	(3,3)	256	(1,1)	(64,64,256)
	Conv3_2 + ReLu + BN	(3,3)	256	(1,1)	(64,64,256)
	Dropout_3				
Down-sampling	Pool3(Max)	(2,2)		(2,2)	(32,32,256)
Unit3,0	Conv4_1 + ReLu + BN	(3,3)	512	(1,1)	(32,32,512)
	Conv4_2 + ReLu + BN	(3,3)	512	(1,1)	(32,32,512)
	Dropout_4				(32,32,512)
Up-sampling	Up_1	(2,2)	256	(1,1)	(64,64,256)
Merge	Up_1 + Dropout_3				(64,64,512)
Unit2,1	Conv5_1 + ReLu + BN	(3,3)	256	(1,1)	(64,64,256)
	Conv5_2 + ReLu + BN	(3,3)	256	(1,1)	(64,64,256)
	Dropout_5				
Up-sampling	Up_2	(2,2)	128	(1,1)	(128,128,128)
Merge	Up_2 + Dropout_2				(128,128,256)
Unit1,2	Conv6_1 + ReLu + BN	(3,3)	128	(1,1)	(128,128,128)
	Conv6_2 + ReLu + BN	(3,3)	128	(1,1)	(128,128,128)
	Dropout_6				
Up-sampling	Up_3	(2,2)	64	(1,1)	(256,256,64)
Merge	Up_3 + Dropout_1				(256,256,128)
Unit0,3	Conv7_1 + ReLu + BN	(3,3)	64	(1,1)	(256,256,64)
	Conv7_2 + ReLu + BN	(3,3)	64	(1,1)	(256,256,64)
	Dropout_7				
	Conv8_3 + Sigmoid	(1,1)	2	(1,1)	(256,256,2)
	Softmax				(256,256,1)

BN: batch normalization.

To speed up the training of the model and alleviate the gradient disappearance caused by the deepening of the network model, Google proposed BN in 2015.[38] The update of the training parameters of the upper layer leads to a change in the data distribution of the next layer, which causes data drift and gradient vanishing. The BN layer pulls data back into the central region and makes training deep networks more stable and efficient. A BN performs a rectified normalization by training weights γ and bias β as

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{3}$$

and

$$\bar{x} \leftarrow \frac{x_i - u_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{4}$$

where

$$y_i \leftarrow \gamma\bar{x} + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \tag{5}$$

where x_i and y_i are the input and output values, respectively, in one batchsize; x is a normalized input value; ε is defined as a harmonic parameter and is initially set up in the training process; and m is the number of samples in one batchsize.

Overfitting has always been the most common problem in ML. In the DL field, dropout operation was proposed to specifically solve this problem.[39] The main operation was to randomly disconnect the hidden layers with a certain probability. The DL model will be more generalizable by the disconnecting layers because it reduces the dependence on some local features.

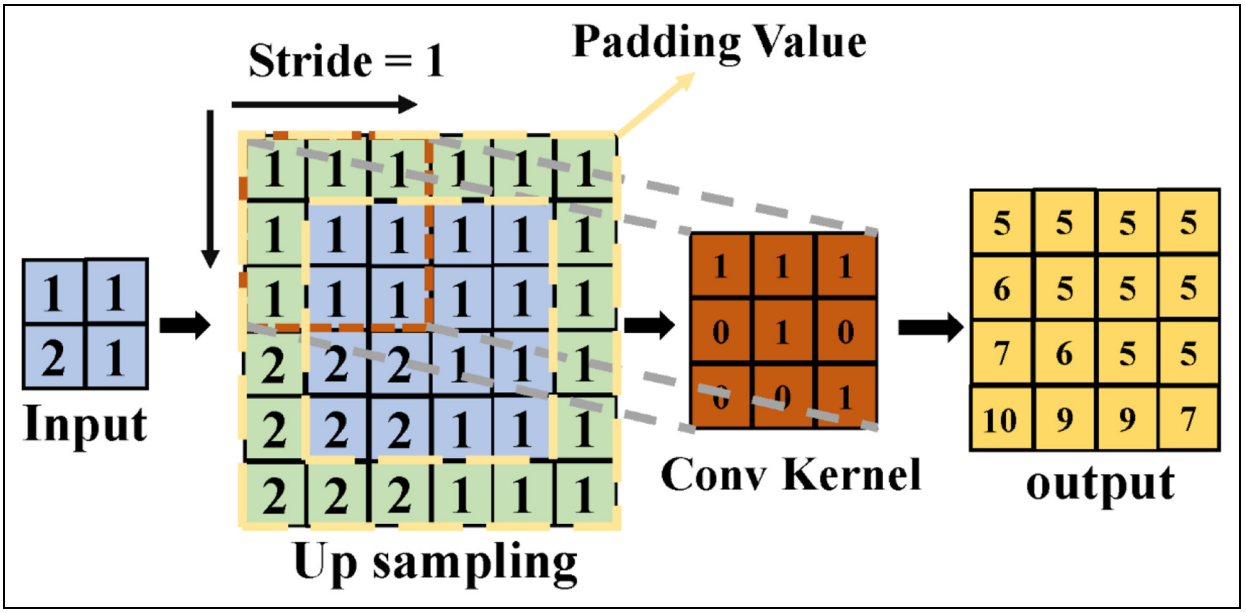

Figure 5. Up-sampling process.

The output of the final up-sampling layer is a tensor, and each pixel within the original image is scored with each class. To uniformly represent the scores obtained, the Softmax function is used for normalization as

$$y_j = \frac{e^{Wx_j + B}}{\sum_{i=1}^{N} e^{Wx_i + B}} \tag{6}$$

where output y_j returns the clustering probability for the individual class j. W and B represent the weights and biases in the previous layer, respectively. N is the total number of classes. In this article, $N = 2$.

Loss function. Loss function is an indicator to evaluate the deviation between the prediction result and the ground truth. The larger the deviation between the prediction results and the ground truth, the larger the loss value. The target of training the model is to minimize the loss value. In semantic segmentation field, cross-entropy loss function, dice loss function, focal loss function, and so on are the common loss functions. Since the crack detection task is a data imbalance task (in Figure 6), the ordinary loss function cannot effectively describe the difference between the predicted value and the ground truth. To solve the training problems brought by the data imbalance, generalized dice loss (GDL) which was proposed by Sudre et al.[40] is selected as the loss function. The GDL can be expressed as follows

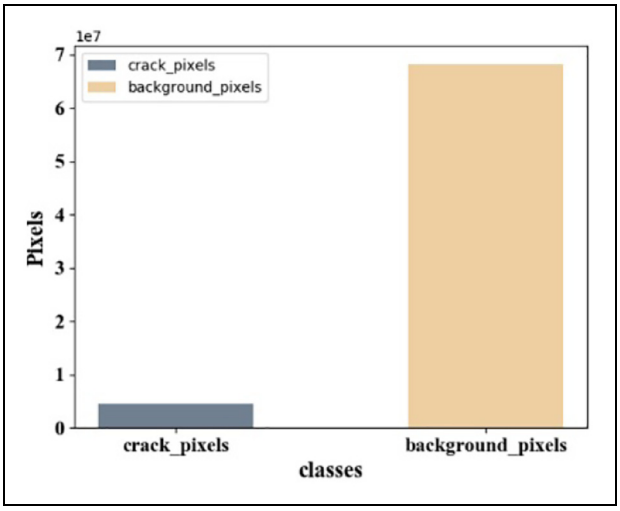

Figure 6. Comparison of cracks and background pixels.

$$\mathrm{GDL} = 1 - 2\frac{\sum_{l=1}^{2} w_l \sum_n r_{ln} p_{ln}}{\sum_{l=1}^{2} w_l \sum_n r_{ln} + p_{ln}} \tag{7}$$

and

$$w_l = \frac{1}{\left(\sum_{n=1}^{N} r_{ln}\right)^2} \tag{8}$$

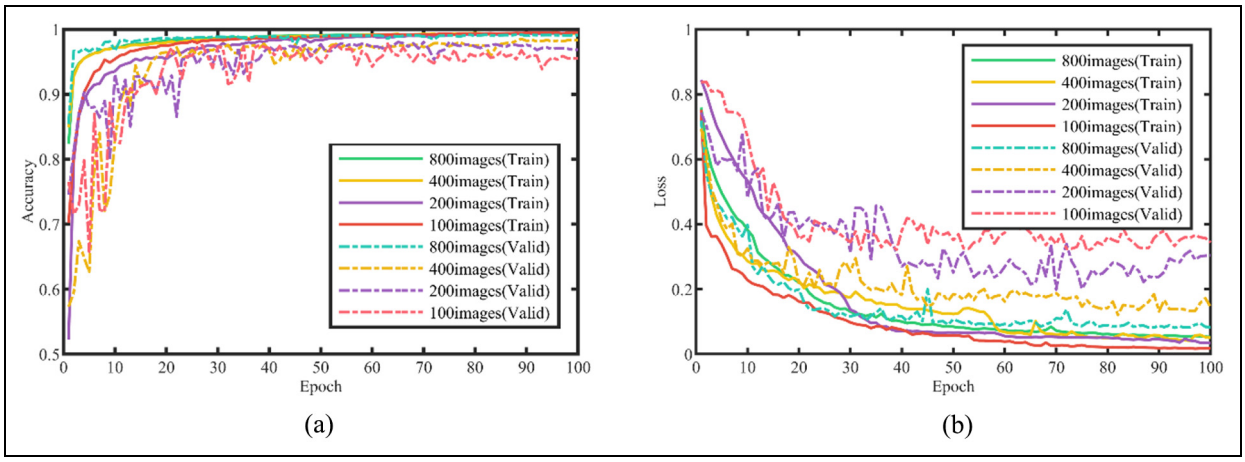

Figure 7. Training and validation results of CrackUnet19 on datasets with different sizes: (a) accuracy and (b) loss.

where l is the total number of classes, r_{ln} is the ground truth of class l in pixel n, p_{ln} is the corresponding prediction value, w_l is the weight of each class, and N is the number of samples in one batchsize. Compared with other loss functions, GDL is more suitable for segmenting extremely imbalanced datasets by adding different weights to different categories.

Optimizer. There are numerous methods to minimize the deviation during backpropagation, such as the stochastic gradient descent (SGD), momentum method, RMSprop, and Adam algorithms;[41–44] however, Adam-based backpropagation is considered the most efficient and fastest approach to decrease the deviations. Compared to other algorithms, the Adam optimizer promotes the gradients to converge in the right direction by calculating the first and second moment estimations of the gradient.

Experiment implementation and results

Model training

Parameter setting. In addition to the convolutional layer parameters, there are some hyperparameters that need to be initialized, including the number of convolution kernels per layer, convolution kernel size, convolution step size, and parameters in Adam optimizer. The encoding path of the CrackUnet19 model in this study uses the VGG-19 model directly. The other three models in turn remove one unit based on VGG-19. The Adam hyperparameters in this study are set up referring to Kingma,[44] and the learning rates of the weights and biases are set as 0.0001.[45] The default exponential decay rates for the first and second moment estimates are set as 0.9 and 0.99, respectively, and ε is set as 10^{-8}. The harmonic parameter in the BN is set as $1e^{-5}$. The batchsize is 20.

Training process. The training process was implemented on a workstation with a single GPU device (Nvidia GeForce GTX-1080Ti) and a CPU (Intel i7-7800X). The code was programmed using Python 3.5, and the virtual environment was established by Keras 1.4 and NumPy 1.14.

To choose the most suitable model and dataset, several sets of experiments were designed. The first experiment is set to study the effect of the size of the dataset on the performance. The first experiment includes two steps. First, four different datasets which contain 100, 200, 400, and 800 images, respectively, are randomly selected from the training dataset. Meanwhile, the corresponding validation datasets are also prepared, which contain 200 images. Second, the four datasets will be fed into CrackUnet19 for training. Figure 7 displays the training results of CrackUnet19 trained by four different datasets.

As shown in Figure 7, CrackUnet19 model performs best in four training datasets. However, CrackUnet19 trained by a small dataset has serious overfitting problems. Smaller dataset contains less types and numbers of images, so CrackUnet19 cannot effectively generalize the learned features to other images. Along with the increase in the number of images in the training dataset, the overfitting phenomenon of the model is alleviated. The larger dataset contains more types of crack images, so the dataset which contains 800 crack images is chosen to conduct the next experiment. The second experiment is set to study the effect of the depth of the model on the performance. Figure 8 displays the training result of four different models trained by 800 crack images.

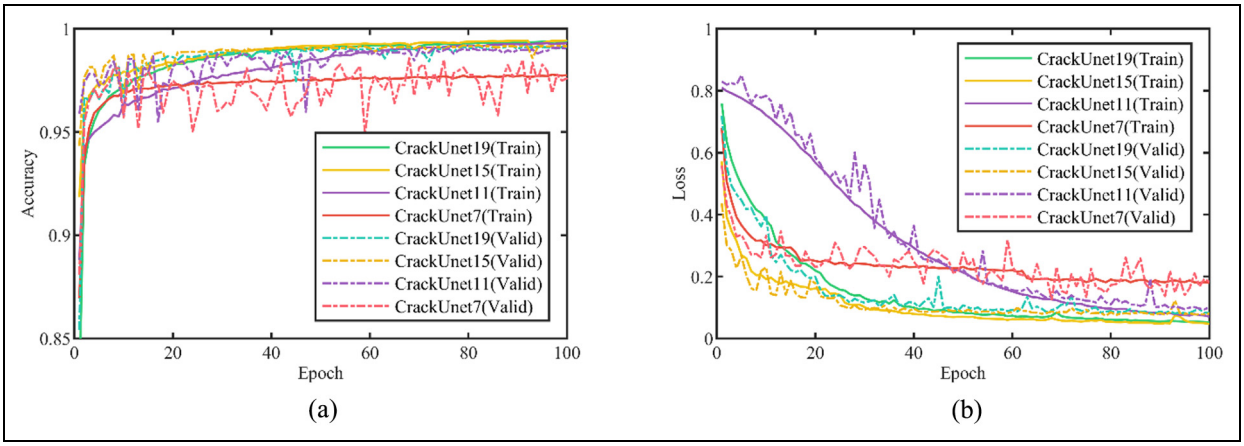

Figure 8. Training and validation results of models with different depths: (a) accuracy and (b) loss.

Table 2. Performance of different models on the test dataset.

Model	Precision	Recall	F1 score	Model parameter storage/MB	Running time per image/s
CrackUnet7	0.7280	0.7622	0.7456	1.61	0.02649
CrackUnet11	0.8993	0.8754	0.8872	7.22	0.02759
CrackUnet15	0.9134	0.8813	0.8971	29.5	0.02937
CrackUnet19	0.9145	0.8867	0.9004	118	0.03528

Figure 8(a) illustrates the increase in the accuracy over epochs. The accuracy rates of CrackUnet7, CrackUnet11, CrackUnet15, and CrackUnet19 are 97.75%, 99.31%, 99.44%, and 99.36%, respectively, which are higher than that (97.96%) of the FCN proposed by Yang et al.[26] The training accuracy of CrackUnet7 is the lowest and the validation accuracy fluctuates significantly. Compared to the other three deeper models, CrackUnet7 has fewer convolutional layers. Because the convolutional layer is mainly used to extract crack features, extremely few convolutional layers may result in inefficient extraction. Figure 8(b) illustrates the decrease in the loss over epochs. It can be clearly seen from Figure 8(b) that CrackUnet7 cannot be effectively optimized in the later period of training. However, except for CrackUnet7, a slight overfitting phenomenon occurs in all the three networks. To further evaluate the performance of these four models, the above four models will be tested on the testing dataset.

Test results and discussion

Evaluation criteria. To evaluate the performance of the model, three metrics are selected to evaluate the performance of four models. The three metrics are precision, recall, and F1 scores, respectively. These three metrics are defined in equations (9)–(11)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{10}$$

and

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{11}$$

where true positive (TP) is the number of true positive pixels, false positive (FP) is the number of false positive pixels, and false negative (FN) is the number of false negative pixels. True positive pixel is a crack pixel in ground truth, and it is also recognized as a crack pixel in prediction result. False positive pixel is a background pixel in ground truth but recognized as a crack pixel in prediction result. False negative pixel is a pixel that is recognized as a background pixel in prediction result, but, in fact, it is a crack pixel in ground truth.

Test results. The above results are based on training sets and validation sets. To further verify the performance of the trained models, test dataset which contains 200 new images outside the training and validation sets is used for testing. The mean value of precision, recall, and F1 score of four different models was summarized in Table 2.

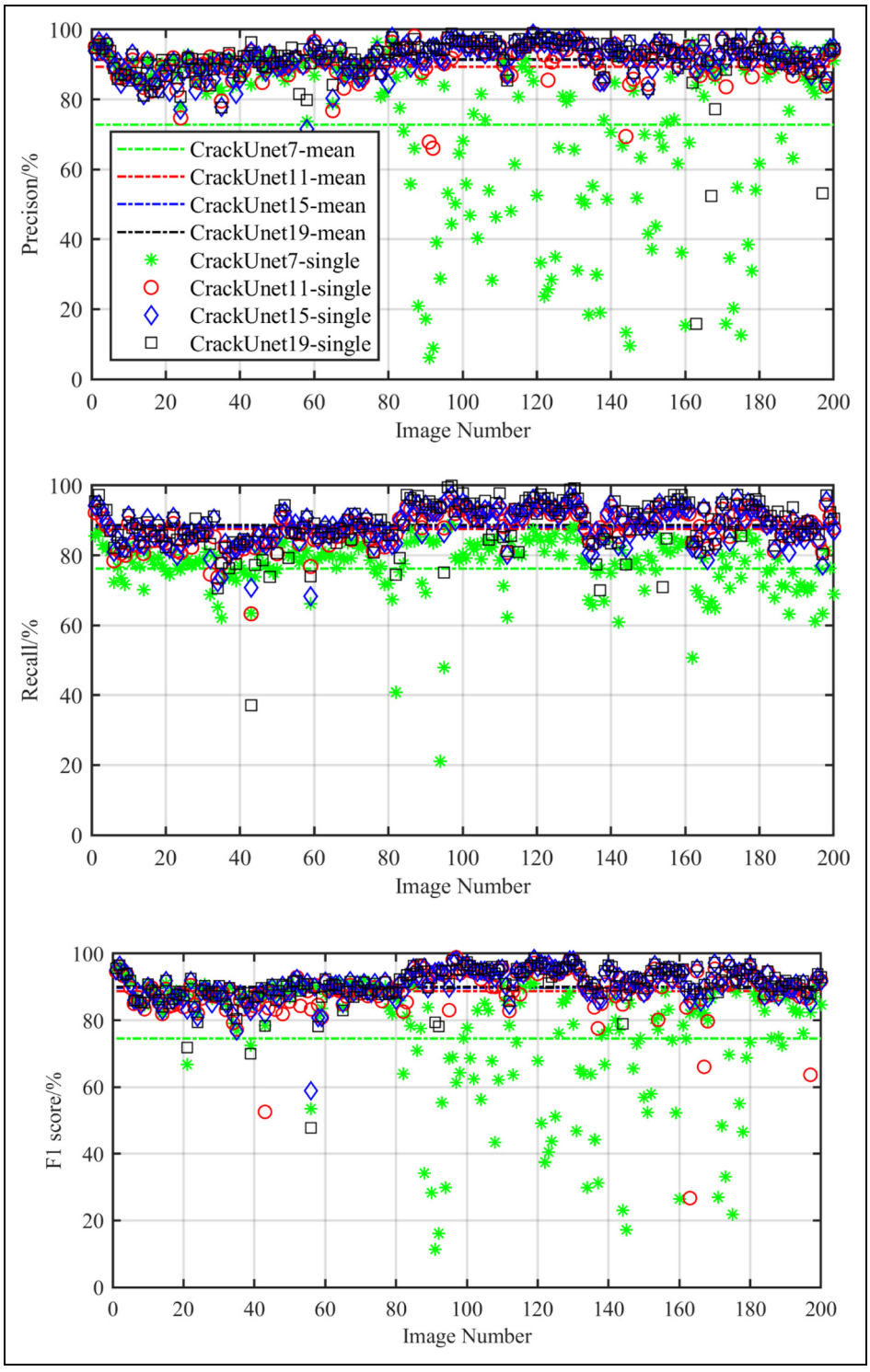

Figure 9. Illustration of precision, recall, and F1 score of four models on test dataset.

In Table 2, we can see that three metrics of CrackUnet11, CrackUnet15, and CrackUnet19 are very close. In addition, from the perspective of segmentation speed, smaller scale model has higher segmentation speed than larger scale model. As displayed in Figure 9, the performance of CrackUnet15 and CrackUnet19 maintains stable in all 200 testing images. The precision and F1 score of CrackUnet7

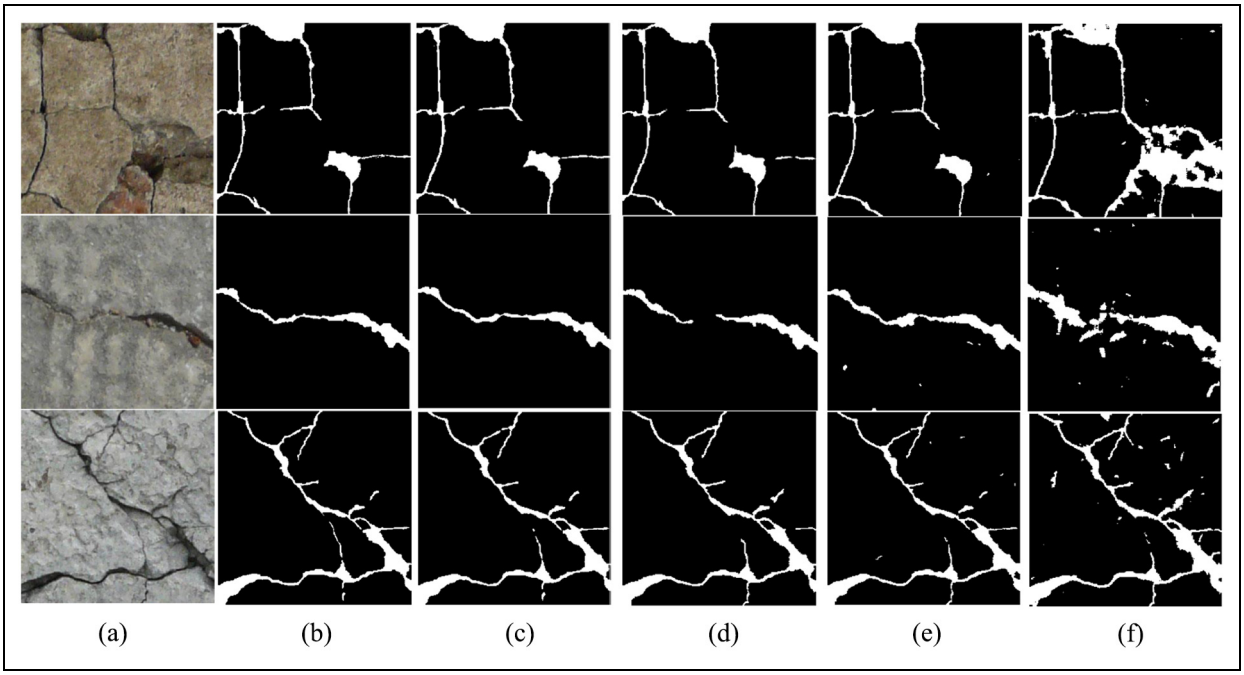

Figure 10. Prediction results of the four models with different scales: (a) original image, (b) ground truth, (c) CrackUnet19, (d) CrackUnet15, (e) CrackUnet11, and (f) CrackUnet7.

fluctuate seriously. This means that if the neural network is too shallow, it will cause poor stability and cannot detect cracks in complex situations. In Figure 10, several types of images with complex backgrounds are presented with results generated by CrackUnet models. As presented in Figure 10, CrackUnet19 and CrackUnet15 have high segmentation accuracy at pixel level and can adapt to various forms of cracks. For the first image, CrackUnet11 missed a part of thin cracks. For the other two images, it could not eliminate the noise completely. CrackUnet7 performed worst on these three images. CrackUnet7 only contains seven convolutional layers, so it cannot effectively extract crack features. Considering the poor performance of CrackUnet7, the next task of comparison with other methods will be undertaken by other three models.

Table 3. Comparison of performance for different methods.

Methods	Precision/%	Recall/%	F1 score/%
Other methods			
FFA[46]	78.56	68.43	73.15
CrackTree[47]	73.22	76.45	70.80
CrackForest[8]	82.28	89.44	85.71
MFCD[48]	89.90	89.47	88.04
CrackNet-V[30]	92.58	86.03	89.18
DenseCrack201P[29]	92.02	91.13	91.58
Proposed methods			
CrackUnet11	93.08	87.01	89.81
CrackUnet15	98.23	**92.84**	**95.44**
CrackUnet19	**98.72**	91.94	95.15

FFA: free-form anisotropy; MFCD: multiscale fusion crack detection.
The bold-faced values mean the highest value in the Table.

Comparison with other methods. In order to evaluate the proposed method, we compare it with other methods on the same dataset (CrackForest Dataset (CFD)). CFD is proposed by Shi et al.[8] which consists of 118 images taken of road surfaces in Beijing, China. The resolution of the raw images is 480 × 320 pixels. The reason why we choose CFD is that the cracks in the images have been annotated annually and it has been widely used by other studies. This dataset is randomly split following 60%/40% rule for training and test. Since the dataset is different from the dataset we used before, CrackUnet19, CrackUnet15, and CrackUnet11, which already trained on original dataset, are further trained on CFD. The performance of the above three models are compared with FFA (free-form anisotropy),[46] CrackTree,[47] CrackForest,[8] MFCD (multiscale fusion crack detection),[48] CrackNet-V,[30] and DenseCrack201P,[29] as presented in Table 3. It is noted that CrackForest, CrackNet-V, and DenseCrack201P are tested under the same condition as our proposed method.

As shown in Table 4, it is worth noting that as the same method based on DL, CrackUnet15 and CrackUnet19 outperform than CrackNet-V and DenseCrack201P. In terms of precision, all three models

Table 4. Fundamental parameters of different methods.

Network	Parameters	Size of images	Size of training dataset	Training time (h)
CrackNet[7]	1159561	1024 × 512 × 1	1800	96
CrackNet-V[30]	64113	512 × 256 × 1	2568	24
DenseCrack201P 8×[29]	90,008,682	128 × 128 × 3	62,790	None
CrackUnet19	31043461	256 × 256 × 3	800	1.9
CrackUnet15	7702917	256 × 256 × 3	800	1.7
CrackUnet11	1865349	256 × 256 × 3	800	1.4
CrackUnet7	404741	256 × 256 × 3	800	1.1

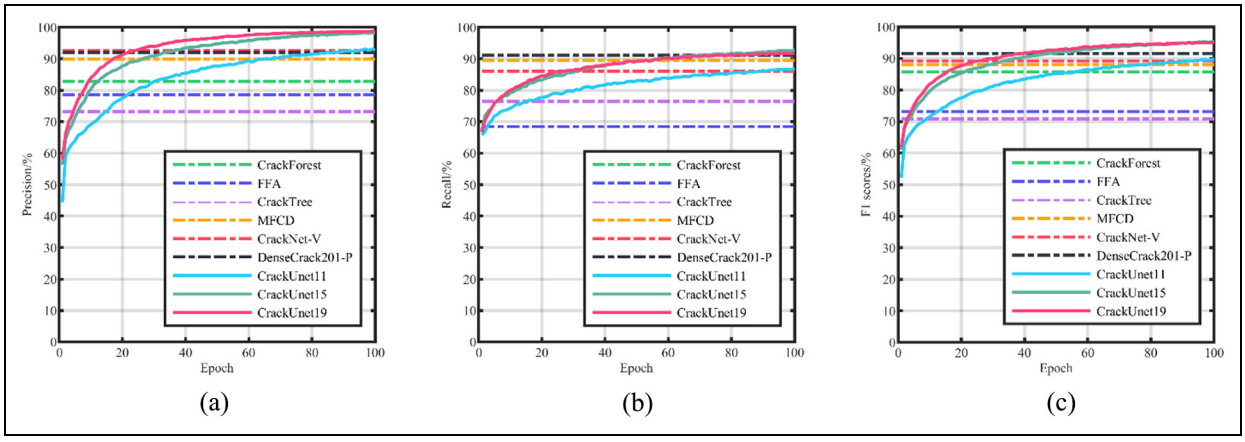

Figure 11. Comparison of performance for different methods on CFD: (a) precision, (b) recall, and (c) F1 scores.

give better performance than all other methods. CrackUnet15 has 5.63% better F1 score than CrackUnet11 and 0.29% better than CrackUnet19, although these three models only differ by four convolutional layers. In Figure 11, it can be seen that after about 40 epochs, the F1 score of CrackUnet19 and CrackUnet15 starts to outperform other methods. This means that after a few training steps, the model can adapt to the new dataset. This shows that CrackUnet models have a good generalization ability on other datasets.

To judge whether the overfitting phenomenon occurred on the further-trained model and to study the severity of overfitting phenomenon, the further-trained CrackUnet models are tested on the individual test dataset again. The F1 score of CrackUnet11, CrackUnet15, and CrackUnet19 are 0.807, 0.856, and 0.864, respectively. The overfitting phenomenon occurred in all three models, but the overfitting phenomenon of CrackUnet15 and CrackUnet19 is not very serious. Therefore, we believe that further training will not cause serious overfitting of the model.

Network speed. According to the small size of the dataset and the structure of CrackUnet, the proposed methods are more time efficient. Table 4 summarizes the fundamental parameters of CrackUnet and other methods. As displayed in Table 4, it is clear that CrackNet and CrackNet-V used more than thousands of images to train the model and DenseCrack201P even used more than 60,000 images for training. The CrackUnet series models only used 800 images for training and achieved good detection accuracy. In terms of training time, CrackUnet models also outperform other methods because of the small training dataset and structural advantages. Compared with other models, the completely symmetric CrackUnet is more efficient to learn crack features. Generally, the improved training efficiency of CrackUnet is more beneficial for real-time processing in the future.

Robustness test. In order to test the robustness of the proposed methods, six images collected under the complicated conditions are processed by using CrackUnet. The detection results of the six new images that contain common disturbances are shown in Figure 12: a crack with rust disturbances (Figure 12(a) and (b)), a crack with speckle noise (Figure 12(c) and (d)), and a crack

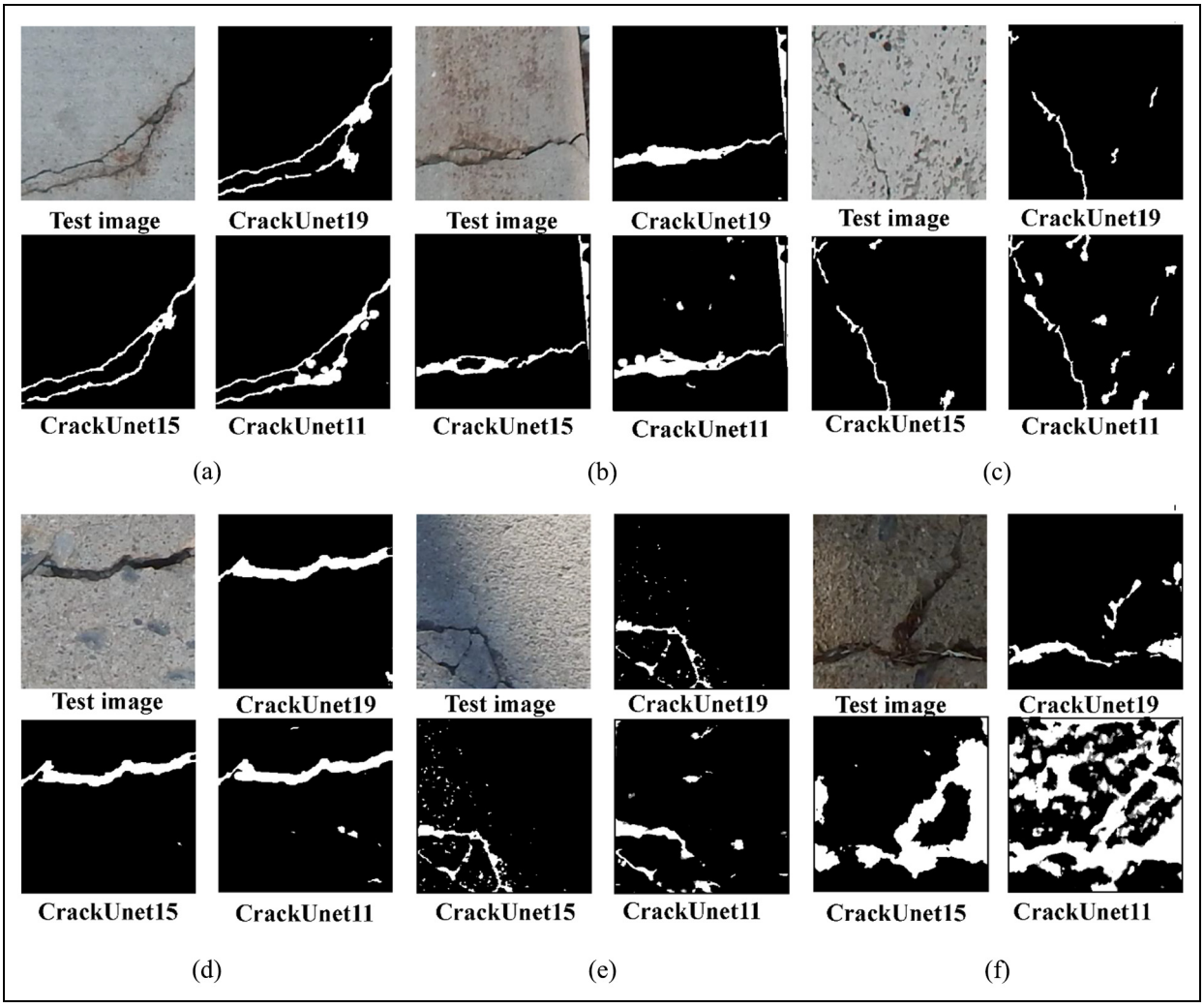

Figure 12. Detection results of four new images: (a, b) cracks with rust disturbances, (c, d) cracks with speckle noise, and (e, f) cracks with strong illumination shadows.

with strong illumination shadows (Figure 12(e) and (f)). It is worth noting that CrackUnet15 and CrackUnet19 give better performance than CrackUnet11 on the six images. Although CrackUnet19 is deeper than CrackUnet15, it still confused some rust disturbances and cracks. Regarding the crack images with speckle noise, CrackUnet19 and CrackUnet15 can both detect the cracks accurately. In terms of cracks with strong illumination shadows, CrackUnet19 performs better than CrackUnet15. CrackUnet11 performs worst on all six images. Generally, CrackUnet19 performs more robust on images with complicated disturbance.

Conclusion

In this article, an improved Unet-based model was proposed for detecting surface cracks in pixel level. A new loss function is applied to the model to solve the training problem caused by the sample imbalance. How the size of the dataset and the depth of the model influence the performance of DL-based model is studied. The generalization of this method on a public dataset (CFD) is also presented. The following conclusions were drawn:

1. A new loss function is applied to the proposed method. The training results display that the new loss function can improve the performance of the model. The highest training accuracy of CrackUnet is 99.31%, which is higher than 97.96% of FCN.
2. Larger dataset can strengthen the detection performance of the model. Too few images in dataset will cause serious overfitting phenomenon, which will reduce the generalization ability. In addition, the performance of the model does not linearly increase with the depth of the model increasing. The

performance of CrackUnet19 and CrackUnet15 on the testing dataset is very close. The performance of the two models on the CFD is also very close. CrackUnet15 has a higher recall of 92.84% and an F1 score of 95.44% on CFD than other state-of-the-art methods.

3. Compared with other methods, the dataset used to train the model is smaller and the training time is less. Noticeably, the detection accuracy of the proposed model is higher.
4. The detection results of six images show that CrackUnet15 and CrackUnet19 can detect cracks under complex conditions. However, CrackUnet cannot accurately detect cracks from images with strong illumination shadows. In the future, the model will be trained by more images with other disturbance including other damage types, other objects, and strong illumination shadows to improve the performance of CrackUnet.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study was financially supported by grants from the National Key R&D Program of China (Grant 2017YFC1500606) and Heilongjiang Touyan Innovation Team Program.

ORCID iD

Junkai Shen https://orcid.org/0000-0002-6293-5085

References

1. Xia Q, Cheng YY, Zhang J, et al. In-service condition assessment of a long-span suspension bridge using temperature-induced strain data. *J Bridge Eng* 2017; 22(3): 04016124.
2. Aied H, González A and Cantero D. Identification of sudden stiffness changes in the acceleration response of a bridge to moving loads using ensemble empirical mode decomposition. *Mech Syst Signal Pr* 2016; 66–67: 314–338.
3. Kunwar A, Jha R, Whelan M, et al. Damage detection in an experimental bridge model using Hilbert–Huang transform of transient vibrations. *Struct Control Health Monit* 2013; 20(1): 1–15.
4. Chenag X, Ji X, Henry RS, et al. Coupled axial tension-flexure behavior of slender reinforced concrete walls. *Eng Struct* 2019; 188: 261–276.
5. Kim H, Ahn E, Shin M, et al. Crack and noncrack classification from concrete surface images using machine learning. *Struct Health Monit* 2019; 18(3): 725–738.
6. Lei B, Ren Y, Wang N, et al. Design of a new low-cost unmanned aerial vehicle and vision-based concrete crack inspection method. Struct Health Monit. *Epub ahead of print*3 February 2020. DOI: 10.1177/1475921719898862.
7. Zhang A, Wang KCP, Li B, et al. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Comput Aid Civ Infrastruct Eng* 2017; 32(10): 805–819.
8. Shi Y, Cui L, Qi Z, et al. Automatic road crack detection using random structured forests. *IEEE Trans Intell Transp Syst* 2016; 17: 3434–3445.
9. Zhang L, Yang F, Zhang YD, et al. Road crack detection using deep convolutional neural network. In: *Proceedings of the IEEE international conference on image processing (ICIP 2016)*, Phoenix, AZ, 25–28 September 2016, pp. 3708–3712. New York: IEEE.
10. Cha YJ, Choi W, Suh G, et al. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Comput Aid Civ Infrastruct Eng* 2018; 33(9): 731–747.
11. Xue Y and Li Y. A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. *Comput Aid Civ Infrastruct Eng* 2018; 33(8): 638–654.
12. Abdel-Qader I, Abudayyeh O and Kelly ME. Analysis of edge-detection techniques for crack identification in bridges. *J Comput Civ Eng* 2003; 17(4): 255–263.
13. Fujita Y, Mitani Y and Hamamoto Y. A method for crack detection on a concrete structure. In: *Proceedings of the IEEE 18th international conference on pattern recognition (ICPR 2006), Hong Kong, China*, 20–24 August 2006, vol. 3, pp. 901–904. New York: IEEE.
14. Kim H, Ahn E, Cho S, et al. Comparative analysis of image binarization methods for crack identification in concrete structures. *Cem Concrete Res* 2017; 99: 53–61.
15. Jahanshahi MR, Masri SF, Padgett CW, et al. An innovative methodology for detection and quantification of cracks through incorporation of depth perception. *Mach Vis Appl* 2013; 24(2): 227–241.
16. Li G. Image-based method for concrete bridge crack detection. *J Inform Comput Sci* 2013; 10(8): 2229–2236.
17. Li G, Zhao X, Du K, et al. Recognition and evaluation of bridge cracks with modified active contour model and greedy search-based support vector machine. *Automat Constr* 2017; 78: 51–61.
18. Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. *Proc IEEE* 1998; 86(11): 2278–2324.
19. Cha YJ, Choi W and Buyukozturk O. Deep learning-based crack damage detection using convolutional neural networks. *Comput Aid Civ Infrastruct Eng* 2017; 32(5): 361–378.
20. Xu Y, Li S, Zhang D, et al. Identification framework for cracks on a steel structure surface by a restricted Boltzmann machines algorithm based on consumer-grade

camera images. *Struct Control Health Monit* 2018; 25(2): e2075.
21. Xu Y, Bao Y, Chen J, et al. Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images. *Struct Health Monit* 2019; 18(3): 653–674.
22. Chen FC and Jahanshami MR. NB-CNN: deep learning-based crack detection using convolutional neural network and naïve Bayes data fusion. *IEEE Trans Ind Electron* 2018; 65(5): 4392–4400.
23. Mohtasham Khani M, Vahidnia S, Ghasemzadeh L, et al. Deep-learning-based crack detection with applications for the structural health monitoring of gas turbines. Struct Health Monit. *Epub ahead of print*11 November 2019. DOI: 10.1177/1475921719883202.
24. Maeda H, Sekimoto Y, Seto T, et al. Road damage detection and classification using deep neural networks with smartphone images. *Comput Aid Civ Infrastruct Eng* 2018; 33(12): 1127–1141.
25. Ni FT, Zhang J and Chen ZQ. Zernike-moment measurement of thin-crack width in images enabled by dual-scale deep learning. *Comput Aid Civ Infrastruct Eng* 2019; 34(5): 367–384.
26. Yang X, Li H, Yu Y, et al. Automatic pixel-level crack detection and measurement using fully convolutional network. *Comput Aid Civ Infrastruct Eng* 2018; 33(12): 1090–1109.
27. Bang S, Park S, Kim H, et al. Encoder–decoder network for pixel-level road crack detection in black-box images. *Comput Aid Civ Infrastruct Eng* 2019; 34(8): 713–727.
28. Zhang A, Wang KCP, Fei Y, et al. Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network. *Comput Aid Civ Infrastruct Eng* 2019; 34(3): 213–229.
29. Mei Q and Gül M. Multi-level feature fusion in densely connected deep-learning architecture and depth-first search for crack segmentation on images collected with smartphones. Struct Health Monit. *Epub ahead of print*3 January 2020. DOI: 10.1177/1475921719896813.
30. Fei Y, Wang KCP, Zhang A, et al. Pixel-level cracking detection on 3D asphalt pavement images through deep-learning-based CrackNet-V. *IEEE Trans Intell Transp Syst* 2020; 21: 273–284.
31. Ronneberger O, Fischer P and Brox T. U-Net: convolutional networks for biomedical image segmentation. In: *Proceedings of the international conference on medical image computing and computer-assisted intervention, Munich*, 5–9 October 2015, pp. 234–241. Cham: Springer.
32. Maguire M, Dorafshan S and Thomas RJ. SDNET2018: a concrete crack image dataset for machine learning applications, 2018, https://digitalcommons.usu.edu/all_datasets/48/
33. Wada K. Labelme: image polygonal annotation with python, 2016, https://github.com/wkentaro/labelme
34. Wang J and Perez L. The effectiveness of data augmentation in image classification using deep learning, 2017, https://arxiv.org/pdf/1712.04621.pdf
35. Simonyan K and Zisserman A. Very deep convolutional networks for large-scale image recognition, 2014, http://arxiv.org/abs/1409.1556
36. Scherer D, Muller A and Behnke S. Evaluation of pooling operations in convolutional architectures for object recognition. In: *Proceedings of the 20th international conference on artificial neural networks (ICANN 2010), Thessaloniki*, 15–18 September 2010, pp. 92–101. Berlin: Springer.
37. Nair V and Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML 2010)*, Haifa, Israel, 21–24 June 2010, pp. 807–814. New York: ACM.
38. Ioffe S and Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the international conference on machine learning (ICML 2015), Lille*, 6–11 July 2015, pp. 448–456. New York: ACM.
39. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 2014; 15(1): 1929–1958.
40. Sudre CH, Li W and Vercauteren T, , et alet al. Generalized dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Cardoso MJ, Arbel T and Carneiro G, et al. (eds) *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Cham: Springer, 2017, pp. 240–248.
41. Bengio Y. Practical recommendations for gradient-based training of deep architectures. In: Montavon G, Orr GB and Müller KR (eds) *Neural networks: tricks of the trade*. 2nd ed. Berlin: Springer, 2012, pp. 437–478.
42. Sutskever I, Martens J, Dahl G, et al. On the importance of initialization and momentum in deep learning. In: *Proceedings of the 30th international conference on machine learning (ICML 2013), Atlanta, GA*, 16–21 June 2013, pp. 1139–1147. New York: ACM.
43. Ruder S. An overview of gradient descent optimization algorithms, 2016, https://arxiv.org/pdf/1609.04747
44. Kingma DP and Ba J. Adam: a method for stochastic optimization, 2014, https://arxiv.org/abs/1412.6980
45. Wilson DR and Martinez TR. The need for small learning rates on large problems. In: *Proceedings of the international joint conference on neural networks (IJCNN 2001), Cat. No. 01CH37222*, Washington, DC, 15–19 July 2001, vol. 1, pp. 115–119. New York: IEEE.
46. Nguyen TS, Begot S, Duculty F, et al. Free-form anisotropy: a new method for crack detection on pavement surface images. In: *Proceedings of the 18th IEEE international conference on image processing (ICIP 2011), Brussels*, 11–14 September 2011, pp. 1069–1072. New York: IEEE.
47. Zou Q, Cao Y, Li Q, et al. CrackTree: automatic crack detection from pavement images. *Pattern Recogn Lett* 2012; 33: 227–238.
48. Li H, Song D, Liu Y, et al. Automatic pavement crack detection by multi-scale image fusion. *IEEE Trans Intell Transp Syst* 2019; 20: 2025–2036.