HOKKAIDO UNIVERSITY

Title	Cost-effective system for detection and quantification of concrete surface cracks by combination of convolutional neural network and image processing techniques
Author(s)	Miao, Pengyong; Srimahachota, Teeranai
Citation	Construction and building materials, 293, 123549 https://doi.org/10.1016/j.conbuildmat.2021.123549
Issue Date	2021-07-26
Doc URL	https://hdl.handle.net/2115/89199
Rights	© <2021>. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/
Rights(URL)	https://creativecommons.org/licenses/by-nc-nd/4.0/
Type	journal article
File Information	Manuscript(Final+Without track the revisions).pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

Cost-effective system for detection and quantification of concrete surface cracks by combination of convolutional neural network and image processing techniques

Pengyong MIAO (Corresponding author)
Graduate School of Engineering, Hokkaido University, Sapporo, Japan
p.y.miao@outlook.com
Teeranai SRIMAHACHOTA
teeranai.sg@gmail.com

Abstract: This study proposed a semi-automatic system for crack detection and quantification, based on the combination of a trained convolutional neural network (CNN) and a developed application. Specifically, we tested four commonly used CNNs and determined GoogLeNet for this study. Then, the transfer learning and fully training of GoogLeNet were further tested on our testing dataset and a public dataset. The results show that the transfer learning GoogLeNet has relatively balanced performances on these two datasets, with accuracy of 96.69 % and 88.39%, respectively. A new sliding window technique (neighborhood scanning) was proposed and shown almost equivalent performance to the previous dual scanning method. A method for calculating crack width was presented. The average relative error of this method is 14.58% (0.05 mm), i.e., much smaller than the 36.37% (i.e., 0.14 mm) of the previous method. An application was then developed to integrate the proposed methods and other techniques such as edge detectors, boundary tracking, and threshold segmentation to segment, quantify, and analyze cracks. Verifications on 23 untrained raw images (eleven with 10240×2048 pixels, twelve with 2592×4608 pixels) show that: (1) the developed system and a previous pixel-level segmentation system require an average of 9.48s and 10.35s; (2) these two systems show an 80.40% and a 78.64% average intersection over union (IoU). Therefore, the proposed system is a cost-effective solution for detecting and analyzing cracks on concrete surfaces considering its practical

performance and time cost. Practically, the proposed system could be used to analyze the images collected from onsite inspection or from experiment.

1. Introduction

Infrastructures (including numerous concrete structures) must be prudently managed to balance safety, economy, and sustainability requirements, and the maintenance of such infrastructures has become a major social concern [1]. Despite the several limitations of visual inspection, it is a widely accepted methodology used in practice for the asset management of buildings and bridges [2]. The investigation of concrete defects including cracks is a commonly and necessary task in an inspection for assessing the conditions of concrete structures. Currently, owing to advancements in computer technology, researchers are working towards automating the inspection process using digital image analysis [2]. Cracks are of particular importance for the safety and maintenance of concrete structures. Therefore, this study mainly focuses on cracks in concrete structures. Cracks in concrete structures have many causes, such as poor repairs, contractions owing to rapid temperature decreases, fluctuations between contractions and expansions from temperature changes, and extra loads. Regardless of the reason, the occurrence of cracks may affect the appearance of concrete structures, and most importantly, they may indicate significant structural distress or damage [3]. Crack detection and quantification are two major challenges for efficiently assessing the severity of cracks [4]. Literature reviews regarding those two aspects will be described below.

Four ways are available to detect cracks from images: manual detection, image processing techniques, feature-based machine learning, and deep learning-based algorithms. Manual detection is usually time consuming and prone to inaccuracy due to inspector fatigue or human error, and is beyond the scope of this study. Study progress on the image processing techniques,

feature-based machine learning, and deep learning-based algorithms are detailed below.

Many image processing techniques have been proposed and applied. These techniques include: thresholding [5-6]; original or modified edge detection [7-9]; and filter based methods [10-14]. Thresholding is performed to partition an image into multiple parts or regions based on the characteristics of the pixels in the image. Edge detectors and filter based methods detect crack edges by applying various filters to a grayscale image to emphasize discontinuities. However, image processing techniques cannot cope with the random shape and irregular size of cracks [3]. In addition, the results of these techniques are noticeably influenced by the illumination and distortion of images [15]. Although one de-noising technique has been proposed [16] and applied to a study [17], the usage of this technique is limited as images taken from the real-world vary extensively.

Another approach of crack detection is to use machine learning algorithms [18-23]. These algorithms are performed by evaluating whether the signals collected from non-destructive testing indicate defects. In addition, some researchers combined machine learning algorithms with image processing techniques [24-26]. Specifically, image processing techniques are first used to extract features, and then machine learning algorithms to classify these features. Although machine learning algorithms are introduced in their methods, the results of these methods are inevitably affected by the performance of image processing techniques, as image processing is usually the first step in extracting features from images.

In addition, a recent promising development is the introduction and widespread use of deep learning [27]. As a kind of deep learning, convolutional neural network (CNN) has been emphasized in image recognition, as it does not rely on the expert set threshold, can effectively capture the grid topology of images, has high accuracy, can distinguish a large number of

categories, and is robust to image variations [17, 28-29]. Many studies have been conducted to demonstrate the feasibility of CNN and achieved considerable results. Depending on the level of detail required for prediction, the application of CNN in crack detection can be separated into three categories: image classification, object detection, and semantic segmentation [29]. Image classification uses the sub-image database cropped out from raw images to train a classifier for predicting whether a sub-image is cracked or complete. Target detection classifies targets and mark the range and location of each type of target (e.g., crack). Semantic segmentation is to perform pixel level prediction by classifying each pixel as a crack or an intact pixel.

In terms of image classification: Zhang et al. [1] conducted CNN training on pavement images taken by smartphone, and concluded a remarkable improvement in accuracy relative to machine learning classifiers trained on manual features. Cha et al. [15] trained a CNN classifier to classify images as crack or intact regions with help of a slide window. The accuracy of this classifier on cracks exceeds 98%, i.e., significantly better than the edge detection methods. Eisenbach et al. [30] trained a CNN to detect asphalt crack, and its performance is better than the two baseline models. Gopalakrishnan et al. [31] implemented transfer learning to train a classifier on a combination of asphalt and concrete pavement cracking images, and concluded that pre-trained VGG-16 CNN yielded the optimal performance. Zhang et al. [32] used transfer learning to propose a unified detection model for crack and sealed crack, and presented better performance than the three used benchmarks. Li and Zhao [33] established a CNN model with an accuracy of 99.06%, which is applicable for complicated images, such as thin cracks, rough surface, and shadows.

In terms of object detection: Cha et al. [34] used 2366 sub-images cropped from 297 annotated raw images to train a region-based CNN architecture for detecting five types of damage

including concrete cracks. Its average accuracy is 88%. Liu et al. [35] proposed an automatic robot inspection system that using a CNN established by transfer learning as detection for three defects including concrete cracks.

In terms of semantic segmentation: Zhang et al. [36] studied the pixel-level pavement crack detection using the three-dimensional (3D) data (including depth information) from a 3D laser system. The method yields about 90% accuracy, without the capability of detecting hairline cracks. Dorafshan et al. [37] compared the performances of CNN with traditional crack detection methods and concluded that CNN shows significant promise for image-based damage detection in concrete. Fan et al. [38] used a CNN to learn the crack pixels in pavement images. Specifically, the model runs with a fixed-size 27×27 window at every pixel, and then provides a 5×5 pixel as a pixel-level output at the center of that patch. The essential of this approach is same as the image classification CNN. Similarly, Li et al. [39] proposed a pixel-level detection approach by using an 18 × 18 window centered at that pixel. Alipour et al. [29] reported a pixel-level detector by converting the fully connected layer of the image classification CNN architecture into convolutional filters. Then, features extracted from different order convolutional filters are up-sampled to generate a heat map for providing pixel-level prediction. Input of the model is a pixel annotated dataset. The model over 92% of crack pixels and 99.9% of intact pixels in the validation set. Using a similar method, Ni et al. [40] proposed a framework to combine the features extracted from different order convolutional filters to achieve pixel-level classification. Kang et al. [41] proposed a hybrid method to achieve crack segmentation. Specifically, a faster region proposal convolutional neural network (Faster R-CNN) is applied to detect the crack regions. Then, modified tubularity flow field (TuFF) and modified distance transform method (DTM) are used to segment the crack pixels and quantify crack thickness and

length, respectively.

In addition to crack detection, crack quantification is also important for assessing the status of in-service infrastructure and determining corresponding maintenance measures. Further, many researches attempted to correlate detailed crack patterns to the quantitative damage states of concrete beams and panels at different loading stages [42-44].

According to the aforementioned information, the semantic segmentation can not only achieve crack detection but also crack segmentation, making it the optimal approach for detecting cracks. However, semantic segmentation algorithms of Fan et al. [38] and Li et al. [39] are not essentially different from image classification. Although Alipour et al. [29] and Ni et al. [40] achieved real pixel-level detection, crack detection and quantification remains a challenging issue because deep learning training using a database of annotated pixels is time and labor costly.

The main goal of this study is to propose (and put in practice) a semi-automatic system combining a trained convolutional neural network (CNN) and a developed application to detect, quantify and analyze cracks. Specifically, the following works are done:

(1) Images collected from onsite inspections and indoor experiments are used to generate the database for this study.

(2) Four commonly used CNNs established by transfer learning are tested on the training and validation dataset to select a CNN that is suitable for this study.

(3) The transfer learning and fully training of the selected CNN are further tested on the testing dataset and a public dataset SDNET2018 to determine the optimal model.

(4) A new sliding window technique called "neighborhood scanning" is proposed. Then, the performance of this method is compared with that of the previous dual scanning method.

(5) A method for calculating crack width is proposed. The crack widths calculated by this

method and the previous method are compared with that measured by the crack scale.

(6) An application is then developed to integrate the proposed methods and other techniques such as edge detectors, boundary tracking, threshold segmentation to segment, quantify, and analyze cracks.

(7) The performances of the system and a previous pixel-level crack segmentation framework are compared on the 23 untrained raw images in terms of Intersection over Union (IoU) and time cost.

(8) The functions of the developed application in counting the distributions of crack width and crack orientation are illustrated taking an experimental beam as example.

(9) Finally, limitations of this study and its comparisons with other studies are discussed.

2. Methodology

Crack detection and quantification aims to detect crack locations and measure the extent of surface cracks from the collected digital images, as required for quickly diagnosing crack propagation [45]. Fig. 1 shows a flowchart of the system used in this study for detecting and quantifying crack conditions. The entire system includes three stages: pre-processing, crack detection, and post-processing. In the pre-processing phase, images collected from onsite inspections and indoor experiments (Section 3.1) are provided to a computer. Then, the images are subjected to calibration and database generation. In the crack detection phase, the locations of the cracks are identified from the entire image. In the post-processing phase, the characteristics of the cracks, such as the width, length, and orientations are analyzed and visualized. Each of these phases is described more fully below.

2.1 Pre-processing

A digital image is a 2D projection of 3D real-world objects. Images are not necessarily to be

orthographic projections. Although a CNN classifier can identify cracks from raw images without calibration, the perspective error will affect the calculation of the detailed crack properties, such as the width. Therefore, the images need to be corrected and calibrated against such perspective errors to facilitate post-processing. In this study, the calibration is only performed to the indoor experimental images by a composition of rotations, translations, projective transformation, magnifications, and shears, according to [2]. Onsite inspection images are not calibrated because of the inability to confirm the exact perspective angle and the distance from the camera to the structures. An example of the calibration is shown in Fig. 2. In addition, part of these raw images needed to be cropped into unified sub-images to establish a crack detection classifier (Section 3.1). In summary, image preprocessing includes calibrating the image and cropping the raw image into smaller sub-images. The sub-images are manually annotated as crack or intact to generate the database for training and validation.

2.2 Crack detection

As mentioned earlier, a CNN can be used for crack detection in three ways: image classification, object localization, or pixel segmentation. Training a CNN classifier is the primary goal of this study to detect whether a sub-image is crack or intact. Many CNNs were available, such as AlexNet [15], GoogLeNet [40], Resnet18 [46], and VGG-16 [29]. In addition, it is feasible to establish a classifier on the training dataset by two modes: fully training and transfer learning [37]. The former trains the CNN fully from scratch on the training dataset. The latter modifies a few layers of the CNN configuration according to the dataset. In this study, transfer training is first performed to the CNN architectures of AlexNet, GoogLeNet, ResNet18, and VGG-16 to select a suitable CNN for our dataset. Then, the transfer learning and fully training of the selected CNN will be further tested on the testing dataset and a public dataset SDNET2018 to

determine the optimal generic model. Detailed testing procedures and results can be found in Section 3.2.

2.2.1 Overall configuration

In general, a CNN architecture includes an input layer, learning layers, and an output layer. The input layer reads the image and transfers it to the learning layers. The learning layers perform convolution operations by applying filters to extract image features. The output layer classifies the image according to the target categories, using the features extracted in the learning layers. The neural network can be trained by assigning target categories to images in a training dataset and modifying the filter values iteratively through back propagation until the desired accuracy is achieved.

AlexNet is taken as an example to illustrate the modification of CNNs for this study. AlexNet is a remarkable CNN for image classification [33]. It is trained on the ImageNet database, and provides an output with 1000 classes. Since the number of image classes in this study is two (images with and without cracks), the output number of the classes was modified to two. The modified AlexNet is shown in Fig. 3; each dimension in the input image indicates the height, width, and channel (red, green, and blue), respectively. Table 1 presents the detailed specifications of the modified AlexNet CNN. Notably, the Relu activation function is applied after the convolution operation and fully connection operation. In addition, normalization and dropout are also implemented. The softmax layer predicts whether each input image does or does not contain a crack. Similarly, the last two layers of the GoogLeNet, ResNet18, and VGG-16 were modified to classify images as crack or intact.

2.2.2 Update of the connection weights

As the initial values of weights are randomly assigned during training, the predicted classes are

usually inconsistent with the actual classes. The softmax loss function was therefore applied to assess the deviations between the predicted and actual classes, as defined by Eq. (1).

$$L = \frac{1}{N}\left[\sum_{i=1}^{N}\sum_{j=1}^{k}(-log\frac{e^{W_j^T x^{(i)}}}{\sum_{m=1}^{k} e^{W_m^T x^{(i)}}})\right] + \frac{\lambda}{2}\sum_{j=1}^{k} W_j^2 \quad (1)$$

where N and k are the number of samples and that of classes, respectively. W are weights. $\sum_{m=1}^{k} e^{W_m^T x^{(i)}}$ is independent of $\sum_{j=1}^{k}(\cdot)$. The parameter λ is a regularization to penalize large weights for preventing overfitting [15].

To minimize the deviation during training, the weights were updated to obtain true classes. The update of the weights is achieved using a stochastic gradient descent (SGD) approach. In addition, a momentum algorithm is combined with the SGD to accelerate the convergence process of the training. As shown in Eq. (2), the gradient $\nabla_W L$ of loss function is calculated with respect to W at time t. Then the velocity V_{t+1} in Eq. (3) is updated ($\leftarrow$) by combination of the previous velocity V_t and the gradient $\nabla_W L$, where Momentum μ and learning rate α are two parameters [47]. Finally, the W are updated using Eq. (4).

$$\nabla_W L\left(W; x^{(i)}, y^{(i)}\right) = \frac{1}{N}\sum_{i=1}^{N}\left[x^{(i)}(-p\left(y^{(i)} = j^{(i)}\middle|x^{(i)}; W\right))\right] + \lambda W_j \quad (2)$$

$$V_{t+1} \leftarrow \mu V_t - \alpha\nabla_W L\left(W; x^{(i)}, y^{(i)}\right) \quad (3)$$

$$W_{t+1} \leftarrow W_t + V_{t+1} \quad (4)$$

The CNN is tuned by repeating the described procedures until desired accuracy is achieved. During the training, the training dataset is usually separated into sub-training sets to speed up the training. These sub-sets are called batch sizes. Each complete update out of a batch size is called an iteration, and each complete update out of the entire training dataset is called an epoch.

2.3 Post-processing

The second objective of this study is to segment and quantity cracks after the established CNN

classifier detect sub-images with cracks. Because cracks information such as width, length, and orientations are important for determine damage situations of structures. The post-processing procedures are shown in Fig. 4. Firstly, the sub-images are converted to grayscale. Then, the contrast of the grayscale image is enhanced. Mask processing is used for reducing noise and smoothing image. Next, edge detectors, boundary tracking and threshold segmentation are performed to segment the image. More details regarding these techniques have been shown in Gonzales and Woods [48]. Finally, the crack properties such as crack thickness, length, and orientation can be obtained, as detailed in Section 2.3.1 and Section 2.3.2. Commercial software is available to achieve such functions, but the use of commercial software makes: (1) the combination of the CNN classifier and post-processing techniques difficult; (2) the use of different software will increase the learning costs for human. In addition, it is time and labor consuming to process all of the sub-images, as every raw image is cropped to thousands of sub-images. To simplify the processing procedures, an application was developed to integrate these techniques. The details of this application are described in Section 4, along with validation with practical examples.

2.3.1 Crack quantification

Once the crack has been delineated, an automatic algorithm can be applied to measure the properties of the crack [3]. The calculable properties include crack width, crack length, crack orientations, and the others. Crack width is the most important parameter for quantifying the cracking of a concrete component. In a previous study, the crack's mean width is measured to represent its width [49]. To measure the crack width more precisely, a different method was proposed, as shown in Fig. 5.

To utilize this method, a neighborhood value δ is pre-defined. S_1 and S_2 are the edges of a

crack. The steps for calculating crack width at point P are summarized in Algorithm 1: (1) forming a dataset using points between the neighborhood lines on edges S_1 and S_2; (2) performing linear regression to the dataset for getting a fitting line *l*; (3) acquiring line *l'* that perpendicular to *l* through point P; and (4) computing the distance from point P to point P'.

In the calculation of the crack width, the points between the neighborhood lines on the edges S_1 and S_2 are applied to obtain a fitting line. If the entire points on the edges S_1 and S_2 are used, the orientation of the crack can be obtained. The ratio of the crack pixels in the raw image can be easily obtained, as all sub-images with cracks are segmented in the post-processing. The crack length can be obtained by calculating the length of the crack skeleton, as indicated in Fig. 5. Details regarding crack skeleton can be found in Gonzales and Woods [48].

2.3.2 Crack statistics and visualization

For each raw image, all cracks can be counted to obtain the statistical characteristics of the cracking. If raw images of the same structural component are collected in chronological order, the crack propagation can be inferred using a wind rose map. In addition, if cracks are detected on each surface of the structural component, crack characteristics can be visualized in 3D [2, 50-51]. These functions are elaborated in Section 4 with practical examples.

3. Building a robust crack classifier

This section introduces the considerations when generating the database and setting the basic hyperparameters, and the procedures of acquiring a robust a CNN. The optimal hyperparameters were confirmed by trial and error, according to Bengio et al. [51]. All of the study is performed on a PC with two GPUs (CPU: Intel© Core© i5-8300H CPU@2.30GHz, RAM: 32GB and GUP NVIDIA GeForce GTX 1050).

3.1 Database generation

In this study, a total of 150 raw images from indoor experiments and onsite bridge inspections were used, as summarized in Table 2. The experimental images were captured with a distance of 1.0 m during the beam bending test. Before the test, these beams have been exposed in the field for one or two years. The pictures collected during the bridge inspection are shot without known the distance from the camera to the object. It was expected that the combination of the images from experiments and inspections would make the classifier general and practically applicable. The experimental raw images were calibrated. Then 58 experimental and 69 onsite inspection images were randomly selected from the corresponding groups. The remaining 11 and 12 images in corresponding groups were used for testing, respectively. The 127 raw images were cropped into sub-images with 256×256 pixel resolution to build the database for training and validation. Totally, the database includes 30,480 sub-images, with the ratio of crack and intact images at 1:1; and includes a broad range of images variances for establishing a robust classifier, as shown in Figs. 6 (a) and 6(b). In addition, sub-images with cracks on the edge of images and with other kinds of damages are disregarded, based on the study of Cha et al. [15], as shown in Fig. 6 (c).

3.2 Optimal model

3.2.1 Performance evaluation

To obtain a CNN model with excellent robustness, four commonly used CNNs from other studies were tested: AlexNet [15], GoogLeNet [40], Resnet18 [46], and VGG-16 [29]; the results are summarized in Table 3. The performances of these CNNs were evaluated using five metrics, as depicted in Fig. 7. In defining these metrics, TP, TN, FP, and FN refer to true positives, true negatives, false positives, and false negatives, respectively. Recall or true positive rate (TPR) is the ratio of correct predictions to total crack sub-images. Similarly, the true negative rate (TNR) indicates the ratio of correct noncrack predictions to the total number of noncrack sub-images.

Precision is the ratio of correct crack predictions to all crack predictions. Accuracy (ACC) is the ratio of correct crack or intact predictions to the total number of sub-images. The F1 score is the harmonic mean of the recall and precision. In addition, time cost is also used as an index to evaluate these four CNNs.

3.2.2 Hyperparameters

All of the CNNs were trained using an SGD algorithm with a mini-batch size of 256 out of 30,480 images. The last layers of all of the CNNs were modified to two outputs, as the output of our dataset was "crack" or "intact". A logarithmically decreasing learning rate was applied in the training, according to Cha and Choi [15]. The dropout rate at the dropout layer was 0.5. The other hyperparameters remained at default values.

3.2.3 Comparisons

The database was divided into 70% for training and 30% for validation. The four commonly used CNNs were trained for 80 epochs on the training set until the loss function reached a plateau, which showed the convergence of the weights. Table 3 presents the detailed performance of the four transfer learning CNNs on the training and validation sets. Fig. 8 shows that the accuracies of the models increase from the AlexNet to the best performing VGG-16. However, VGG-16 spends 346 minutes per epoch in the training of the model. In addition, there is no significant difference in performance between GoogLeNet and RestNet18 on the applied database. The GoogLeNet was therefore chosen to analyze the rest of this study. Furthermore, the performances of the fully training and transfer learning using the GoogLeNet were tested, as summarized in Table 4. Fig. 9 clearly indicates that fully training outperforms transfer learning. Table 5 shows the testing results of the transfer learning model and the fully training model on a public database SDNET2018 [37]. On this public database, the performance of the transfer

learning model is superior to that of the fully training model (Fig. 10), which is contrary to the test results on the testing dataset in this study. In addition, the test results of fully training on SDNET2018 are greatly compromised compared with that on the testing dataset of this study. One possible reason is that the transfer learning can avoid overfitting to some extent. Therefore, the transfer training GoogLeNet was considered the optimal model to implement analysis.

Fig. 11 shows the loss and accuracy during training and validation. Each epoch of training and validation took approximately 39 minutes. The trained GoogLeNet was further tested below.

3.3 Comparisons of scanning approaches

Extensive tests were conducted to validate the optimal CNN from the previous section. Owing to the random distribution of the cracks, and it was difficult to locate cracks depending only on lump-sum scan in a large image; therefore, other algorithms were required to locate the crack positions. In addition, a sub-image with cracks on the edge of it can cause misclassification. To correctly identify the cracks, a 256 × 256 pixel window was designed for scanning the image twice [15], as shown in Fig. 12(a). This method is hereafter referred to as "dual scanning". Except for the dual scanning method, a new scanning method called "neighborhood scanning" was proposed, as depicted in Fig. 12(b). The first scanning of this new scanning method is the same as the previous method. The only difference is that the second scanning of the proposed method is not performed on the entire image, but rather only on the neighborhood of the cracks identified in the first scanning. To avoid repeat scanning in the second operation of the proposed method, the same region is scanned only once.

To compare the proposed and previous scanning methods, 23 of the raw images that were not used to build the training and validation sets were scanned according to the abovementioned two methods. Using the evaluation metrics in Section 3.2.1, the performances of the two scanning

methods are summarized in Table 6. Neighborhood scanning shows a performance equivalent to that of the previous dual scanning method in terms of the five metrics, as shown in Fig. 13. In addition, the achieved accuracy of the two methods are quite remarkable with around 95.5 %, i.e., nearly identical to the accuracy (96.69%) of the validation in the previous section. Encouragingly, the performance of the trained CNN is still impressive, even though field and experimental images are used for testing. The average recorded testing time required for each image using the neighbourhood and dual scanning methods are 6.48 s and 7.11 s, respectively.

Because the raw images used in this study were collected from bridge inspections and beam bending tests, the inspection and experimental images were separately applied to verify the classifier. Figs. 14 and 15 show the testing results of two onsite images. These images can provide a clear understanding of how the classifier functions. For inspection images, all evident cracks can be identified by these two scanning methods. In addition, the unions of the detected crack regions using these two methods are roughly the same. However, the numbers of crack regions detected by the neighbourhood are less than that by the dual scanning method. This occurs because the dual scanning method detected more sub-images with cracks on the edges of them. The false positive and false negative regions (positions) are marked in Figs. 14 (b) and (c) and Figs. 15 (b) and (c). Most false positives are distributed at: (1) the interface between the pier and background, as shown in FP-1~FP-4 in Fig. 14 (b) and FP-1~FP-3 in Fig. 14 (c); (2) the edges of the pier, as shown in FP-5~FP-7 in Fig. 14 (b) and FP-4~FP-6 in Fig. 14 (c); and (3) the corners, as shown in FP-1~FP-4 in Fig. 15 (b) and FP-1~FP-3 in Fig. 15 (c). Most false negatives are caused by insufficient illumination (FN-1 in Figs. 14 (b) and (c), FN-4~FN-5 in Figs. 15 (b) and (c)), and tiny cracks (FN-2 in Figs. 14 (b) and (c), FN-1~FN-3 in Figs. 15 (b) and (c)). These occur because our database included insufficient samples regarding these situations.

Fig. 16 shows testing results for an experimental image using the two scanning methods. Figs. 16 (b) and (c) show that both scanning methods can detect thick cracks, even though the numbers of detected regions in those two methods are different. The dual scanning and neighborhood scanning methods detected 139 and 122 regions, respectively. In Figs. 16 (b) and (c), all false negatives are found to be caused by the tiny cracks, because illumination conditions in the laboratory are more ideal than the onsite conditions. Some false positive regions are located at the interface between the fresh and elder cement, as shown at FP-3 and FP-4 in Figs. 16 (b) and (c). The rest false positive regions are distributed where thin and long voids exist (FP-1 in Figs. 16 (b) and (c)), and where shellfish growth linearly (FP-2 in Figs. 16 (b) and (c)).

The testing results indicate that the built classifier can correctly detect most cracks or intact regions, by combining with the dual or neighborhood scanning methods. However, the classifier will cause misdetections in the previously mentioned situations. In addition, there are no significant differences between the dual and neighborhood scanning methods in terms of the evaluation metrics. The former method can detect more edge cracks. The latter method usually takes less time to scan the same raw image. Considering the unions of the regions detected by these two methods are almost the same, the performance of the neighborhood scanning method is acceptable for the post-processing described below.

4 Development of the post-processing application

The trained CNN can be used to predict the class of a new image combined with the neighborhood scanning method, but the specification of the crack pixel cannot be determined. In addition, the misidentified regions must be addressed in post-processing. Image processing techniques described in Section 2.3 are included in the application. Therefore, different processing techniques can be used depend on images to obtain the optimal segmentation effect.

In addition, the crack property acquisition methods including the algorithm described in Section 2.3.1 are integrated in the developed application. The introductions and verifications of this application are described below.

4.1 Application development

Fig. 17 shows screenshots of the developed application for post-processing. The developed application mainly includes two modules: module 1 for processing the detected crack regions, and module 2 for providing the crack analysis of the raw image. Specifically, the first model is mainly used for the rapid processing of crack regions detected by the classifier in Section 3, so as to obtain crack pixels. Once all of the crack pixels are obtained, they are stitched together and transferred to the second module for calculation of the other properties of the crack, such as the crack width, length, and orientation. These two modules are shown in Figs. 17 (a) and (b), respectively. The application can be run automatically or manually. The former analyzes images according to the default settings and the latter according to the settings of the operator. Verification of the developed application is conducted based on practical examples, as described below.

4.2 Practical comparisons

On the raw images in the testing set (Section 3.1), the practical performances of the developed system are compared with that of a previous pixel-level crack segmentation framework [29], using Intersection over Union (IoU) and time cost as evaluation indexes. The equation of IoU is

$$\mathrm{IoU} = \frac{\text{target} \cap \text{prediction}}{\text{target} \cup \text{prediction}} \tag{5}$$

The comparitive results are summarized in Table 7. im1 to im12 are the results for the twelve onsite inspection raw imgaes. im13 to im23 are the results for the eleven expeimental raw images.

The results show that the developed system and the previous framework exhibited an 80.40% and a 78.64% average IoU, respectively. The developed system took an average of 9.48s, and the previous framework took an average of 10.35s. Specifically, the performances of the developed system and the previous framework vary with image conditions. Experimental images take more time than onsite images because of the larger size of the images. For the same raw image, the developed system usually takes less time than the previous framework. IoU shows that the developed system is comparable to the previous framework for the experimental images. The reason is that the images taken in the laboratory have less interference; enabling both methods achieve good results. These two methods also show approximately the same IoU values for the onsite images except for im1, im2, and im5.

On these three images, the IoU of the development system is 0.05 greater than that of the previous framework. Therefore, these three images with complex backgrounds are detailed in Fig. 18. Although the previous framework shows good performance on trained images with monotonous backgrounds and good illumination, its performance is inferior to our developed system when the method is tested on untrained complex backgrounds (Fig. 18).

These 23 examples show that the performance of the developed system is not inferior to the previous framework. In addition, the developed system usually cost less time than the previous framework. Therefore, the developed system is a cost-effective solution that can detect and quantify cracks from images collected from onsite inspections or from experiments.

4.3 Crack analysis

Detailed information such as crack patterns, width and length are also crucial for understanding the damage in structures, because this information can be used to track the damage status of different components in civil structures. Therefore, the crack analysis focuses on the accurate

acquisition of detailed crack information.

4.3.1 Crack width comparison

To verify the accuracy of the crack width obtained from the proposed algorithm, the algorithm (with a δ value of 7) is applied to calculate the crack widths at 57 positions selected from untrained eleven experimental images. Then the calculated results of these 57 positions are compared with the measured results by the crack scale shown in Fig. 19. In addition, Adhikari et al. [2] utilized the mean width to quantify cracks according to Eq. (6).

$$\text{Mean width} = \frac{\text{crack area}}{\text{crack perimeter}/2} \tag{6}$$

Therefore, the mean width method will also be used for comparison to verify the effectiveness of the system for measuring crack widths. All cracks are quantified in units of mm.

Table 8 shows the comparisons of the proposed algorithm, mean width, and the measurements. The relative error and absolute error of the proposed algorithm and that of the mean value method are calculated taking the measured values as truths. Fig. 20 shows the relative and absolute error distributions of these positions. The absolute error of 35 positions obtained using the proposed algorithm is less than 0.05 mm, while that of 25 locations obtained by the mean method is greater than 0.1 mm. Similarly tendency can be found for the relative error. The relative error of the proposed algorithm is less than 20% at more than 40 positions, but that of the mean value method is greater than 40% at more than 20 positions. The average relative error (Table 8) of the proposed algorithm is 14.58% (0.05 mm), i.e., the same as the thinnest crack width measurable by the crack scale (Fig. 19 (a)). In addition, the average relative error of this method is much smaller than the 36.37% (i.e. 0.14 mm) of the mean value method.

Crack widths greater than 0.2 are considered detrimental to concrete structures according to Japan road association [52]. Therefore, the relative errors of these two methods are respectively

divided into two groups broken down by crack width 0.2mm, as shown in Fig. 21. The method of this study has a balanced performance for both cracks smaller or larger than 0.2 mm with relative error of 16.84% and 13.91%, respectively. However, the mean value method produces a relative error of almost 70% for cracks smaller than 0.2 mm, and produces that of nearly 30% for the cracks greater than 0.2 mm. As a result, the proposed method is effective in measuring both thin and thick crack widths, and is more accurate than the previous mean value method at these 57 positions. However, it should be mentioned that more verifications of this algorithm are necessary in future.

4.3.2 Crack width distribution

Fig. 22 shows the crack width distribution at the front, back, and bottom of a beam obtained by the developed application and algorithm. These three images were taken and calibrated after the beam failed in a bending test. Various crack shapes can be found in these images. Different from previous studies [2, 49] that use the mean width of a crack as its width the proposed algorithm can calculate the crack width at each position along the crack, as can be clearly seen from Fig. 22. In addition, these three images can be stitched to provide a 3-D visualization of the beam, as described in Section 4.3.4.

4.3.3 Crack direction statistics.

Field inspections require determining the locations and orientations of cracks. Therefore, after the cracks are identified, the crack orientations can be counted to obtain the statistical characteristics of the crack orientation distribution in polar coordinates. Figs. 23 (a), (b), and (c) show the counted results for Figs. 22 (a), (b), and (c), respectively. In Fig. 23, the crack orientation distributions of the corresponding image can be clearly determined. In addition, if the raw images of the same component are collected in chronological order, a crack propagation

trace can be inferred by combining the findings in Figs. 22 and 23.

4.3.4 3D visualization

If the cracks on each surface of a component are obtained, the developed application can present and analyze the surface damage in 3D [2, 50]. The results of the images in Fig. 22 are stitched to generate a 3D model of a beam. Fig. 24 shows cracks on the front and bottom faces, respectively. Obviously, the magnitudes and orientations of cracks can be intuitively observed based on the cracks directions statistics in 3D visualization. Based on this model, additional information (such as crack density) can be calculated to determine the severity of cracks and cracking patterns. If the crack information are provided and shown in the 3D model of Fig. 24 in chronological order, the model can be used for tracking the crack development at various stages of loading. In the real world, images collected from inspections can be analyzed and projected onto the 3D model of the structure. Infrastructure managers can be notified when the crack exceeds limitations, or can formulate intervention strategies according to crack propagation patterns, thereby facilitating the effectiveness of the management.

5. Discussion and future work

5.1 Practical application results

Once the proposed system is put into use in structures, it is possible to detect and quantify cracks based only on digital images. Thus, the inspection efficiency and reliability are enhanced. However, from the viewpoint of practical applications, it is not feasible for the developed system to extract any object attributes from any image. To further describe the applicable range of the proposed system, Fig. 25 shows some complicated images that will cause the classifier to fail in detection. Failure detections are distributed at: (1) the interface between the backgrounds and infrastructure (Figs. 25 (a) and (b)); (2) the bonding position between the elder and fresh cement

(Fig. 25 (c)); (3) the long-thin void (Fig. 25 (d)); (4) the linear growth traces of shellfish (Figs. 25 (e) and (f)); and (5) sub-images with tiny cracks (Fig. 25 (g)) or without sufficient illumination (Fig. 25 (k)). The failure of the trained classifier in these situations can be traced to the fact that there are insufficient similar sub-images in the training dataset.

In addition, some sub-images with cracks can be correctly detected by the classifier, but these sub-images are difficult to be further processed for the developed application, owing mainly to uneven illumination and thin cracks. In Fig. 26(a), only the crack pixels with good illumination can be determined by the developed application. For the thin cracks, only some scatter debris can be obtained (Fig. 26 (b)), or only a part of the pixel is retrieved (Fig. 26 (c)). These drawbacks can be avoided to some extent by providing images with higher resolution and better illumination, or by improving the developed application.

Furthermore, the neighborhood scanning method is proposed for identifying all crack regions from a raw image. However, the second scanning largely depends on the accuracy of the first scanning. In other words, the second scanning may fail if some positions are misjudged in the first scanning. The neighborhood scanning method takes less time than the dual scanning method, and can detect the union of the crack regions that is almost the same as the dual scanning method. However, the former method is inferior to the latter method in detecting the sub-images with edge cracks. In addition, it is difficult to determine the optimal size of the scanning window, as the testing images may have various sizes and scales. Although the proposed algorithm outperforms the mean width method, the effectiveness of this algorithm is affected by the performance of the crack segmentation. A 3D model is established using the images from a beam after failure, but no images at various stages of loading are used to verify the applicability of the system to observe the evolution of crack patterns. In addition, the original image still requires

calibration to correctly calculate the crack width and other crack characteristics. We hope that these limitations can be resolved by improving our system.

5.2 Comparison with other studies using the CNN

The developed system is a semi-automatic system. Specifically, the trained CNN classifier can automatically identify the crack regions. Then, the developed application can be run automatically or manually to quantify cracks. Compared with the traditional image classification CNN which only realizes crack detection, this system can not only realize crack detection, but also quantify cracks. In addition, the CNN classifier and the developed application can be used independently to meet the needs at different stages. Although the system is not as automated as semantic segmentation, it reduces the cost of building the pixel-level annotation databases for deep learning training. Kang et al. [41] proposed a similar two steps method. In this two steps method, three independent algorithms including Faster R-CNN, modified TuFF, and modified DTM are integrated. The integration and modification of this hybrid method is superior to our system in two points: (1) this hybrid method is more automated than our system; (2) the performance of crack segmentation by this hybrid method is improved compared with that of our system. However, our system shows advantages in three aspects: (1) the two parts of our system can operate independently, and can provide more crack information except for crack width and crack length; (2) the calculated results of crack properties are shown in mm instead of pixels; (3) verification images are more complex than those of Kang et al. (most images contain only one crack).

Furthermore, our system takes less time, and is not inferior to a previous pixel-level crack segmentation framework [29]. Therefore, it can be concluded that the system proposed in this study is a cost-effective solution for crack detection and analysis.

Although this system is tested on 23 complex images from both onsite and experiment, this study is only the first step in building a robust system. Because not all possible crack patterns, background materials, textures and color appearances are included in the database. Therefore, an important part of the required work is to enlarge the database to include a wider variety of crack patterns and background characteristics. Another task is to improve the proposed system by verifying with more examples, by tuning hyperparameter, and by modifying the proposed algorithm. In the future, other classifiers will be developed to detect various types of superficial damage, such as voids, spalling, and corrosion. In addition, the system is expected to collaborate with the bridge management systems (BMSs) to facilitate the processing of inspection images, as manual processing of onsite inspection images is time consuming and costly. Furthermore, the information obtained by our system can be used to understand the preliminary situation of a bridge, and can provide the basis for further detailed investigations if any abnormalities exist.

6. Conclusions

In this study, a semi-automated system integrating a trained CNN model and a developed application was proposed and studied. The trained CNN model is capable of detecting cracks or intact regions from given raw images. The developed application can reveal detailed crack information. A comprehensive analysis of the developed system in this study reveals the following conclusions:

(1) Comparisons of the performances of the AlexNet, GoogLeNet, ResNet18, and VGG-16 configurations using six metrics (including time cost) indicate that the GoogLeNet is a suitable architecture for this study. Then, transfer learning and fully training of GoogLeNet were verified on our testing dataset and a public dataset, respectively. The results show that the transfer learning GoogLeNet has relatively balanced performances on these two datasets, with accuracy

of 96.69 % and 88.39%, respectively. In addition, the transfer learning GoogLeNet can correctly classify 96.03 % of cracks and 97.35% of intact regions in the testing dataset.

(2) The proposed neighborhood scanning method has an accuracy of 95.33 % for cracks and 95.26% for intact regions, similar to that of the previous dual scanning method (with an accuracy of 95.17 % for cracks and that of 95.87% for intact regions). The neighborhood scanning method usually takes less time than the dual scanning method. However, the former is inferior to the latter in detecting the sub-images with edge cracks.

(3) Practical comparisons of the neighborhood scanning method and the dual scanning method show that both methods are susceptible to uneven illumination, complex backgrounds, and tiny cracks.

(4) The verifications of the developed system and a previous pixel-level crack segmentation framework on 23 untrained raw images show that these two methods exhibited an 80.40% and a 78.64% average IoU, respectively. In addition, the developed system usually cost less time than the previous framework for the same raw image.

(5) The proposed algorithm and the previous crack mean value method were used to calculate the crack width at 57 positions in the testing images. The results indicate that the average relative error of the proposed algorithm is 14.58% (0.05 mm), i.e., much smaller than the 36.37% (0.14 mm) of the previous method.

(6) The developed application is capable of counting the statistical distributions of cracks and generating a 3D model of a structure object. Therefore, the developed application has the potential to be used to observe the evolution of crack patterns during beam bending test, or to analyze the images collected from onsite inspections.

(7) Overall, the results show that the developed system is a cost-effective solution for

detecting and analyzing cracks on concrete surfaces, considering its practical performance and time cost. Future work will improve the system, and validate the proposed techniques on more practical and complicated images. Further, the proposed system could be combined with a BMS to enhance efficiency and reliability of decision-making and management.

References

[1] Zhang, L., Yang, F., Daniel Zhang, Y., & Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. Proceedings - International Conference on Image Processing, ICIP, 2016-Augus, 3708–3712. https://doi.org/10.1109/ICIP.2016.7533052

[2] Adhikari, R. S., Moselhi, O., & Bagchi, A. (2014). Image-based retrieval of concrete crack properties for bridge inspection. Automation in Construction, 39(February 2018), 180–194. https://doi.org/10.1016/j.autcon.2013.06.011

[3] Wang, P., & Huang, H. (2010). Comparison analysis on present image-based crack detection methods in concrete structures. Proceedings - 2010 3rd International Congress on Image and Signal Processing, CISP 2010, 5, 2530–2533. https://doi.org/10.1109/CISP.2010.5647496

[4] Adhikari, R. S., Moselhi, O., & Bagchi, A. (2012). Image-based retrieval of concrete crack properties. 39(June), 180–194. https://doi.org/10.1016/j.autcon.2013.06.011

[5] Oliveira, H., and P. L. Correia. 2009. “Automatic road crack segmenta- tion using entropy and image dynamic thresholding.” In Proc., 17th European Signal Processing Conf. New York: IEEE.

[6] Zou, Q., Y. Cao, Q. Li, Q. Mao, and S. Wang. 2012. “CrackTree: Automatic crack detection from pavement images.” Pattern Recog. Lett. 33 (3): 227–238. https://doi.org/10.1016/j.patrec.2011.11.004.

[7] Abdel-Qader, I., O. Abudayyeh, and M. E. Kelly. 2003. “Analysis of edge-detection techniques for crack detection in bridges.” J. Com- put. Civ. Eng. 17 (4): 255–263. https://doi.org/10.1061/(ASCE)0887 - 3801(2003)17:4(255).

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

[9]Nishikawa, T., Yoshida, J., Sugiyama, T. & Fujino, Y. (2012), Concrete crack detection by multiple sequential

image fil- tering, Computer-Aided Civil and Infrastructure Engineering, 27(1), 29–47.

[10]Frangi, A. F., Niessen, W. J., Hoogeveen, R. M., Van Walsum, T. & Viergever, M. A. (1999), Model-based quantitation of 3-D magnetic resonance angiographic images, IEEE Trans- actions on Medical Imaging, 18(10), 946–56.

[11]Wang, K., Q. Li, and W. Gong. 2007. "Wavelet-based pavement dis- tress image edge detection with à trous algorithm." Transp. Res. Rec. 2007 (1): 73–81. https://doi.org/10.3141/2024-09.

[12]Ying, L., and E. Salari. 2010. "Beamlet transform-based technique for pavement crack detection and classification." Comput.-Aided Civ. Infrastruct. Eng. 25 (8): 572–580. https://doi.org/10.1111/j.1467-8667 .2010.00674.x.

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

[14] Yeum, C. M., and S. J. Dyke. 2015. "Vision-based automated crack detection for bridge inspection." Comput.-Aided Civ. Infrastruct. Eng. 30 (10): 759–770. https://doi.org/10.1111/mice.12141.

[15] Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. Computer-Aided Civil and Infrastructure Engineering, 32(5), 361–378. https://doi.org/10.1111/mice.12263

[16] Rudin, L. I., Osher, S. & Fatemi, E. (1992), Nonlinear total variation based noise removal algorithms, PhysicaD:Non- linear Phenomena, 60(1–4), 259–68.

[17] Cha, Y.-J., You, K. &Choi, W. (2016), Vision-based detection of loosened bolts using the Hough transform and support vector machines, Automation in Construction, 71(2), 181– 88. https://doi.org/10.1016/j.autcon.2016.06.008

[18] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), Gradient-based learning applied to document recognition, in Proceedings ofthe IEEE, 2278–324.

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

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

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

[22]Prasanna, P., K. J. Dana, N. Gucunski, B. B. Basily, H. M. La, R. S. Lim, and H. Parvardeh. 2016. "Automated crack detection on concrete bridges." IEEE Trans. Autom. Sci. Eng. 13 (2): 591–599. https://doi.org /10.1109/TASE.2014.2354314.

[23]Shi, Y., L. Cui, Z. Qi, F. Meng, and Z. Chen. 2016. "Automatic road crack detection using random structured forests." IEEE Trans. Intell. Transp. Syst. 17 (12): 3434–3445. https://doi.org/10.1109/TITS.2016.2552248.

[24] Lattanzi, D., and G. R. Miller. 2012. "Robust automated concrete damage detection algorithms for field applications." J. Comput. Civ. Eng. 28 (2): 253–262. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000257.

[25]Oliveira, H., and P. L. Correia. 2013. "Automatic road crack detection and characterization." IEEE Trans. Intell. Transp. Syst. 14 (1): 155–168. https://doi.org/10.1109/TITS.2012.2208630.

[26] Wu, L., Mokhtari, S., Nazef, A., Nam, B. & Yun, H.-B. (2014), Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road as- sessment, Journal ofComputing in Civil Engineering, 30(1), 04014118-1 to 04014118-19.

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

[28] Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural net- works, Advances in Neural Information Processing Systems, 1097–105.

[29] Alipour, M., Harris, D. K., & Miller, G. R. (2019). Robust Pixel-Level Crack Detection Using Deep Fully Convolutional Neural Networks. Journal of Computing in Civil Engineering, 33(6), 1–14. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000854

[30] Eisenbach, M., R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H.-M. Gross. 2017. "How to get pavement distress detection ready for deep learning? A systematic approach." In Proc., Int. Joint Conf. on, Neural Networks (IJCNN). New York: IEEE.

[31] Gopalakrishnan, K., S. K. Khaitan, A. Choudhary, and A. Agrawal. 2017. "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection." Constr. Build. Mater. 157 (Dec): 322–330. https://doi.org/10.1016/j .conbuildmat.2017.09.110.

[32] Zhang, K., H. Cheng, and B. Zhang. 2018. “Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning.” J. Comput. Civ. Eng. 32 (2): 04018001. https://doi.org/10 .1061/(ASCE)CP.1943-5487.0000736.

[33] Li, S., & Zhao, X. (2019). Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique. Advances in Civil Engineering, 2019(Ml). https://doi.org/10.1155/2019/6520620

[34] Cha, Y. J., W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk. 2017b. “Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types.” Comput.-Aided Civ. Infrastruct. Eng. 33 (9): 731–747.

[35] Liu, L., R.-J. Yan, V. Maruvanchery, E. Kayacan, I.-M. Chen, and L. K. Tiong. 2017. “Transfer learning on convolutional activation feature as applied to a building quality assessment robot.” Int. J. Adv. Rob. Syst. 14 (3): 1729881417712620. https://doi.org/10.1177/17298814177 12620.

[36] Zhang, A., Wang, K. C., Li, B., Yang, E., Dai, X., Peng, Y., ... & Chen, C. (2017). Automated pixel‐level pavement crack detection on 3D asphalt surfaces using a deep‐learning network. Computer‐Aided Civil and Infrastructure Engineering, 32(10), 805-819. https://doi.org/10.1111/mice .12297.

[37] Dorafshan, S., Thomas, R. J., & Maguire, M. (2018). Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. Construction and Building Materials, 186, 1031–1045. https://doi.org/10.1016/j.conbuildmat.2018.08.011

[38] Fan, Z., Y. Wu, J. Lu, and W. Li. 2018. “Automatic pavement crack detec- tion based on structured prediction with the convolutional neural network.” Preprint, submitted February 1, 2018. http://arXiv.org/1802 .02208.

[39] Li, Y., H. Li, and H. Wang. 2018. “Pixel-wise crack detection using deep local pattern predictor for robot application.” Sensors 18 (9): 3042. https://doi.org/10.3390/s18093042.

[40] Ni, F. T., Zhang, J., & Chen, Z. Q. (2019). Pixel-level crack delineation in images with convolutional feature fusion. Structural Control and Health Monitoring, 26(1), 1–18. https://doi.org/10.1002/stc.2286

[41] Kang, D., Benipal, S. S., Gopal, D. L., & Cha, Y. J. (2020). Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning. Automation in Construction, 118, 103291. https://doi.org/10.1016/j.autcon.2020.103291

[42] Davoudi, R., G. R. Miller, and J. N. Kutz. 2017. “Computer vision based inspection approach to predict damage state and load level for RC members.” In Proc., 11th Int. Workshop on Structural Health Monitoring. Stanford,

CA: Stanford Univ.

[43] Davoudi, R., G. R. Miller, and J. N. Kutz. 2018a. “Data-driven vision-based inspection for reinforced concrete beams and slabs: Quantitative damage and load estimation.” Autom. Constr. 96 (Dec): 292–309. https://doi.org/10.1016/j.autcon.2018.09.024.

[44] Davoudi, R., G. R. Miller, P. Calvi, and J. N. Kutz. 2019. “Computer vision based inspection approach to predict damage state and load level for RC members.” Comput. Civ. Eng. https://doi.org/10.12783/shm 2017/14225.

[45] Ayaho, M., Masa-Aki, K., & Eugen, B. (2007). Automatic crack recognition system for concrete structures using image processing approach. Asian Journal of Information Technology, 5(5), 553–561. http://docsdrive.com/pdfs/medwelljournals/ajit/2007/553-561.pdf

[46] X. Jia, X. Yang, X. Yu and H. Gao, "A Modified CenterNet for Crack Detection of Sanitary Ceramics," IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Singapore, Singapore, 2020, pp. 5311-5316, doi: 10.1109/IECON43393.2020.9254351

[47] Bengio, Y. (2012), Practical recommendations for gradient- based training of deep architectures, in G. Montavon, G. B. Orr, and K.-R.M¨uller (eds.), Neural Networks: Tricks ofthe Trade, 2nd edn., Springer, Berlin Heidelberg, pp. 437–78.

[48] Gonzales, R. C., & Woods, R. E. (2018). Digital image processing, 4th Edition.

[49] Albareda-Valls, A., Herrera, A. B., Mestre, J. L. Z., & Zaribaf, S. S. (2018). Image post-processing method for quantification of cracking in RC precast beams under bending. Buildings, 8(11). https://doi.org/10.3390/buildings8110158

[50] Torok, M. M., Golparvar-Fard, M., & Kochersberger, K. B. (2014). Image-based automated 3D crack detection for post-disaster building assessment. Journal of Computing in Civil Engineering, 28(5), 1–13. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000334

[51] Bengio, Y., Goodfellow, I. J. & Courville, A. (2016), Deep Learning, An MIT Press book. Online version is available at: http://www.deeplearningbook.org, accessed July 2016.

[52] Japan road association. 2012. Specification for highway bridges, Part III: Concrete bridges. ISBN: 978-4-88950-715-7

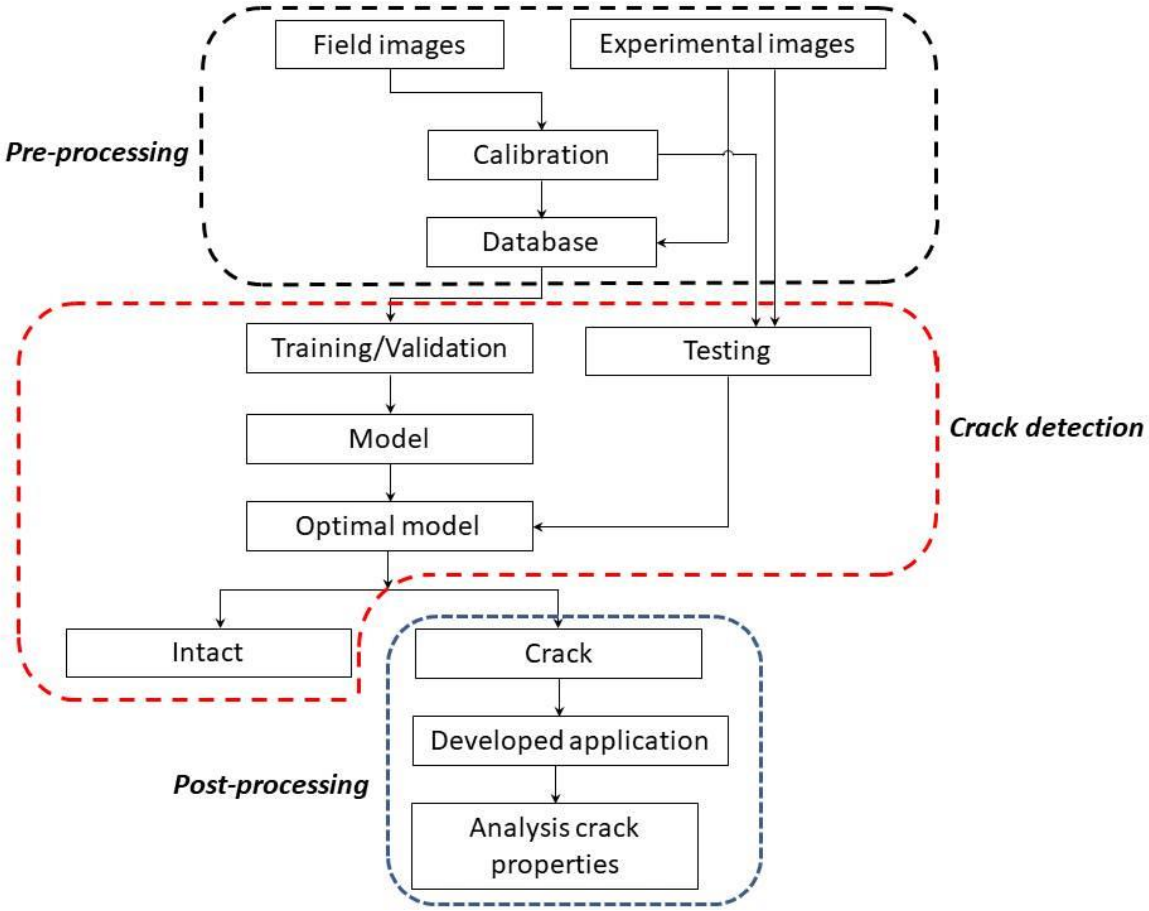


Fig. 1. Flowchart of the system

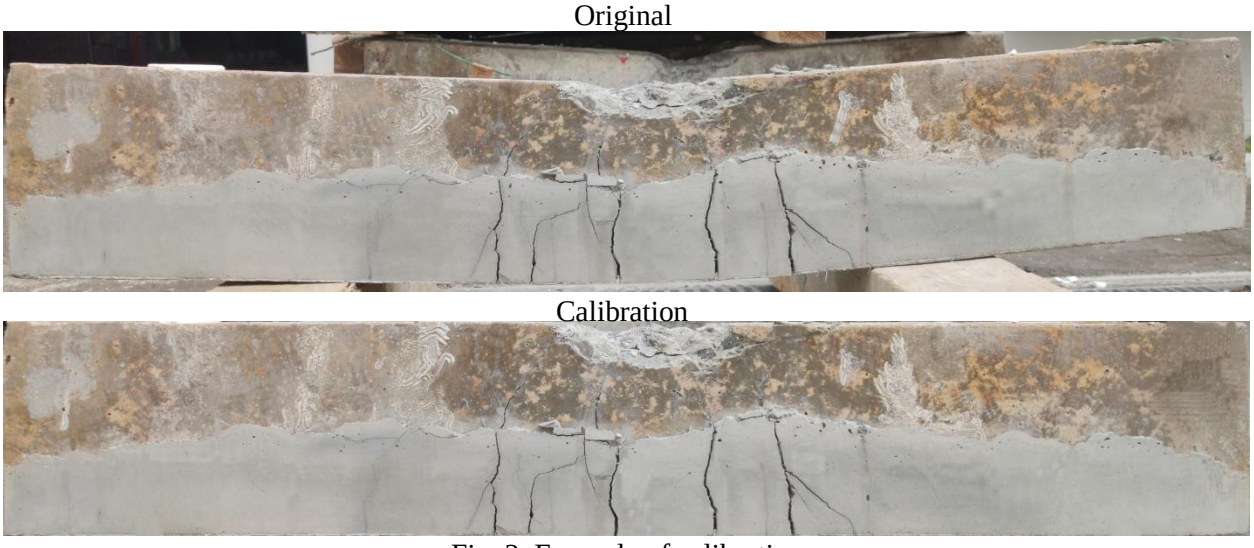


Fig. 2. Example of calibration

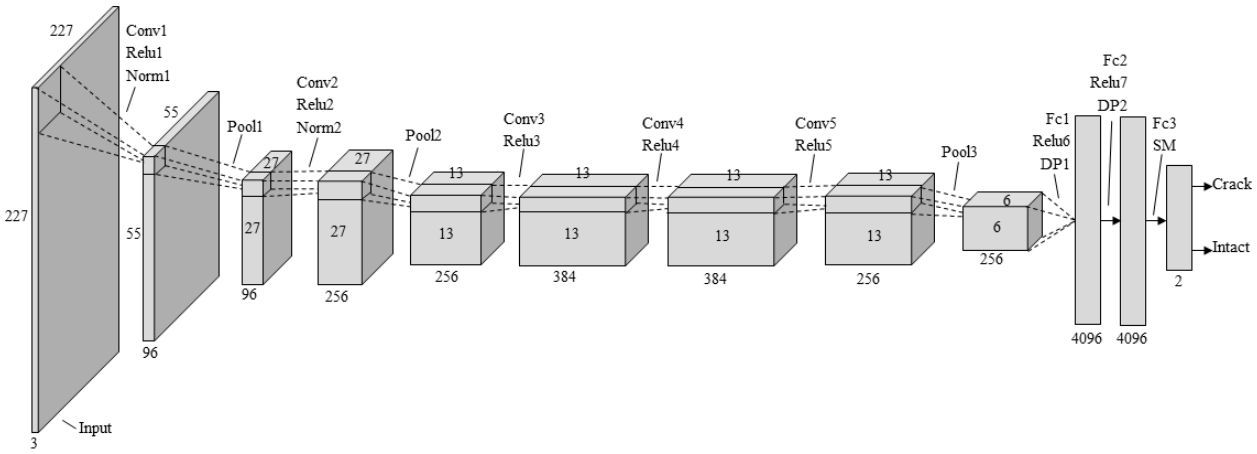


Fig. 3. Illustration of the AlexNet's architecture. conv# = convolution; pool# = pooling; Relu #= activation function; Norm#= normalization; fc# =full connection; k# = kernel of each operation; DP#=Dropout; SM=softmax;

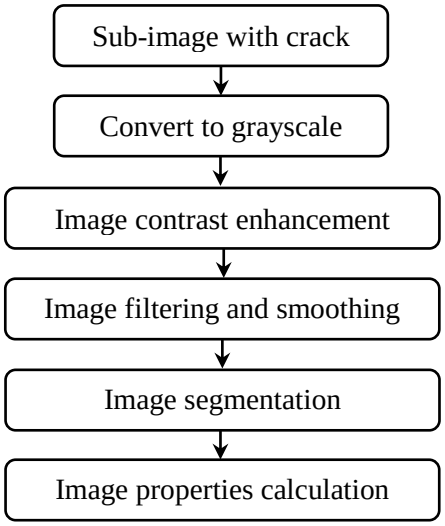


Fig. 4. Flow chart for post-processing

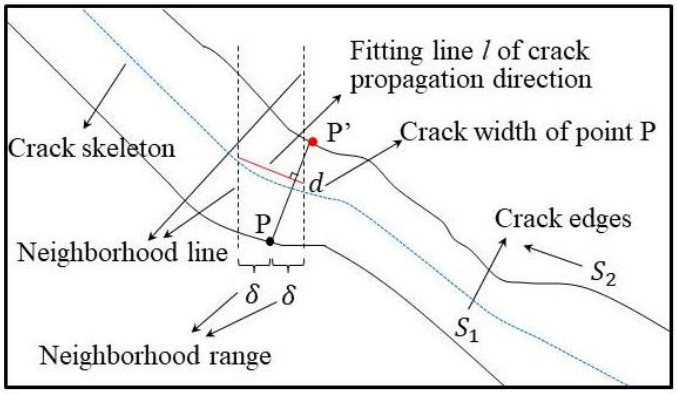


Fig. 5. Depiction for calculating crack properties.



Fig. 6. Typical cropped images: (a) images with crack from experiment; (b) images with crack from onsite inspection; and (c) disregarded images

$$Recall(TPR) = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$

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

$$\text{ACC} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{F1} = \frac{2 * Recall * Precision}{(Recall + Precision)}$$

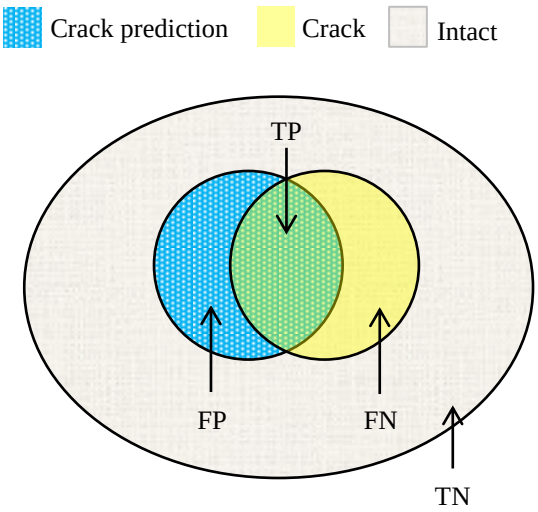


Fig. 7. Performance evaluation metrics used in this study

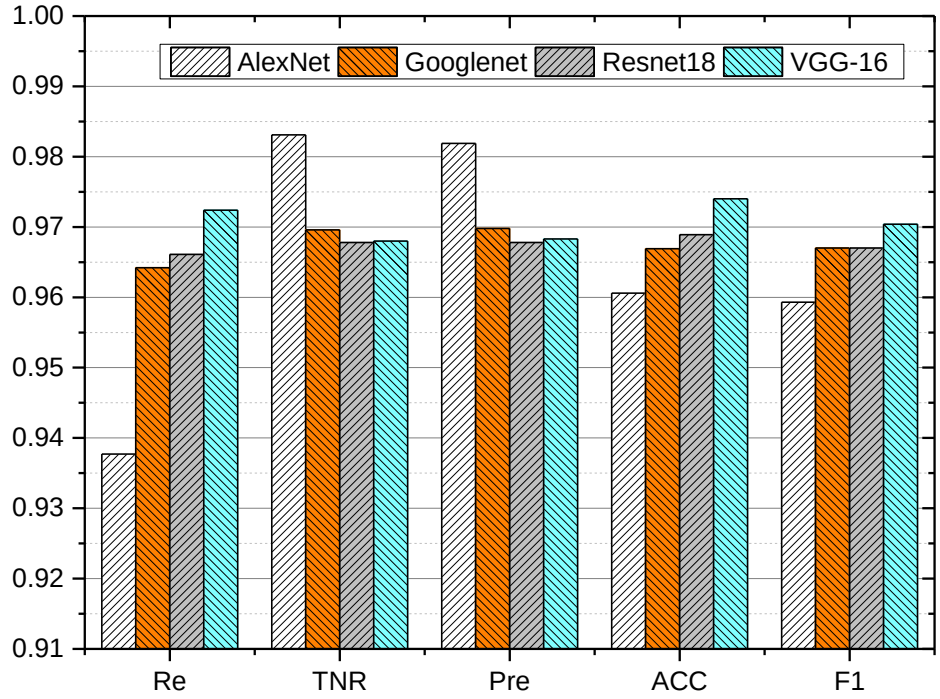


Fig. 8. Performances of different CNNs.

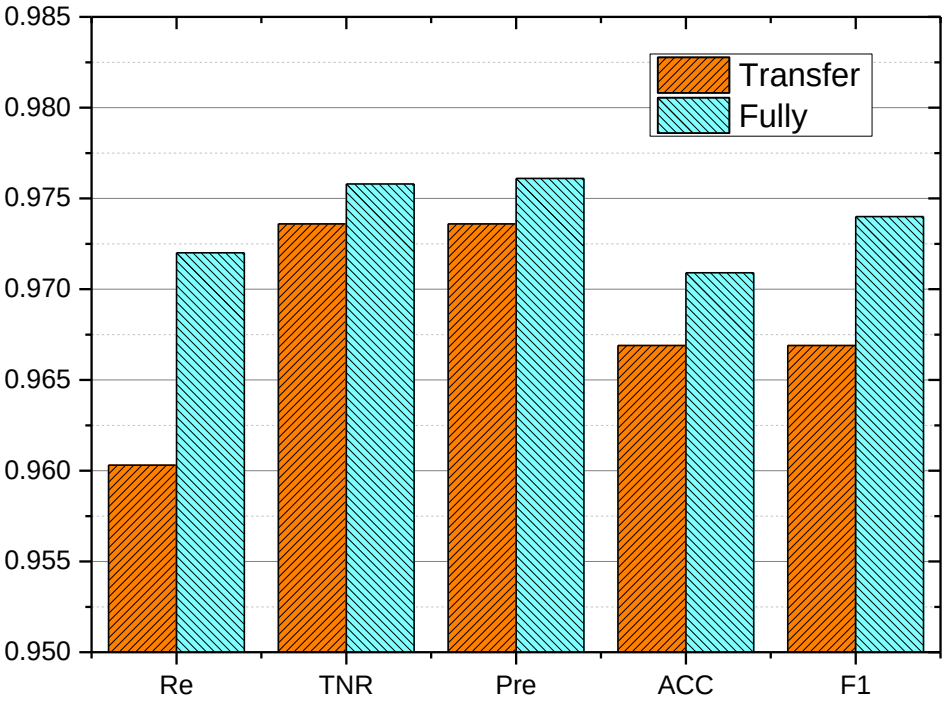


Fig. 9. Metrics of the GoogleNet in transfer and full learning.

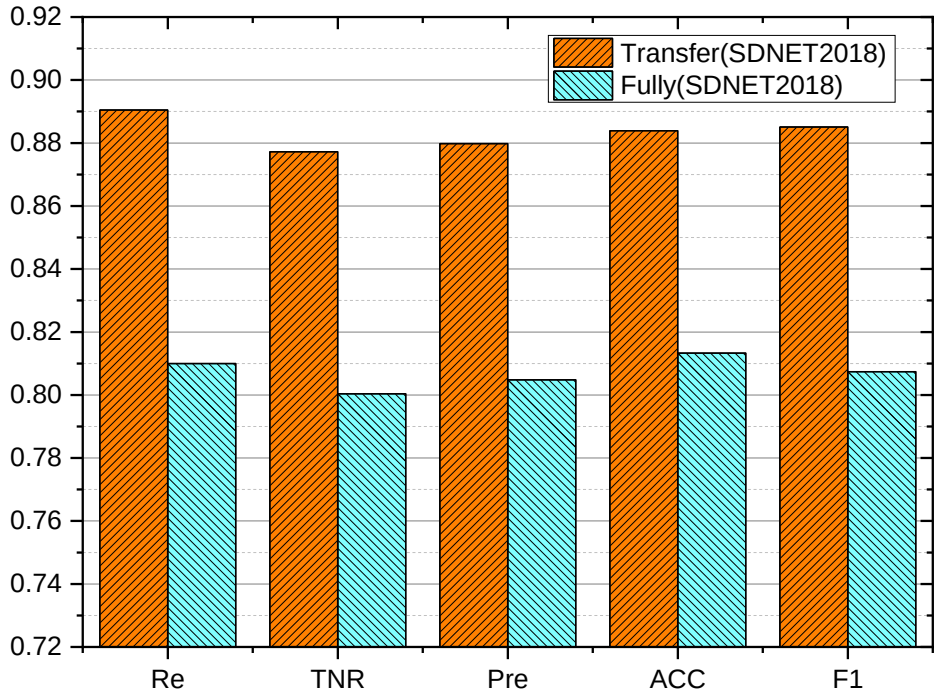


Fig. 10. Performances of the transfer and fully learned models on SDNT2018.

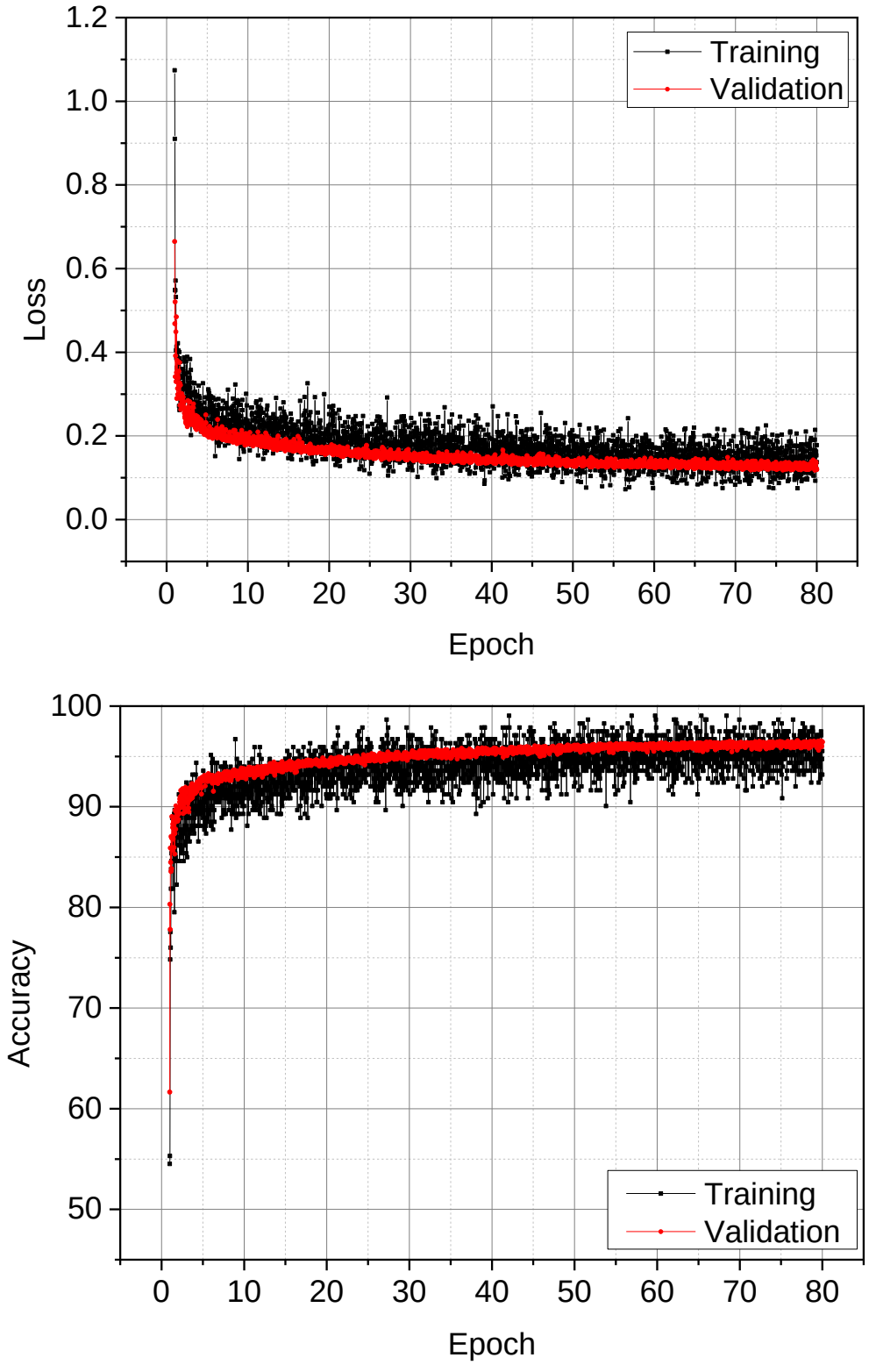


Fig. 11. Loss and accuracy during training and validation.

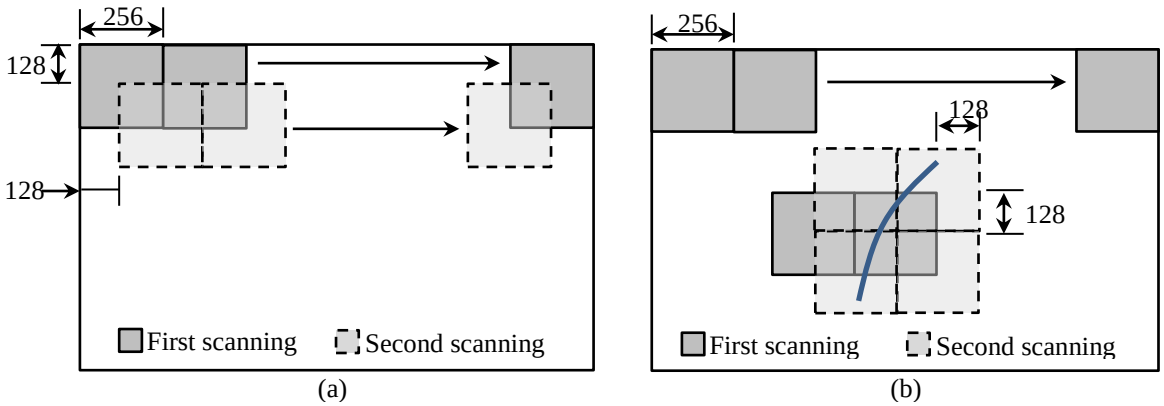


Fig. 12. Crack detection: (a) Dural scanning; and (b) Neighborhood scanning.

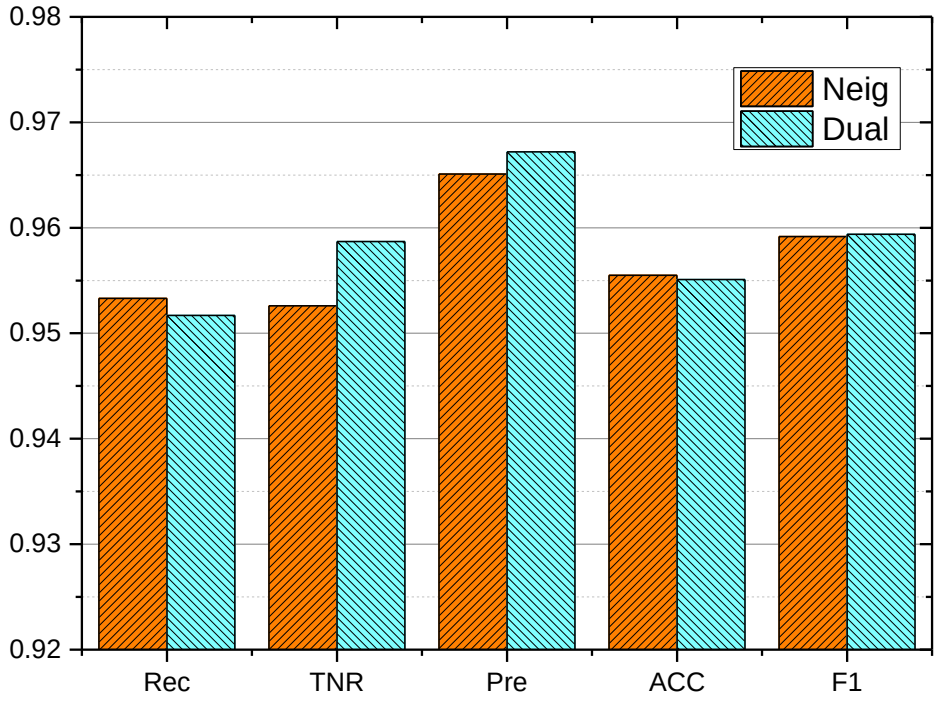
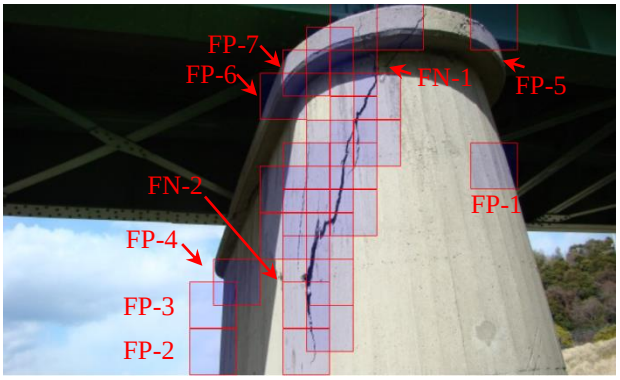


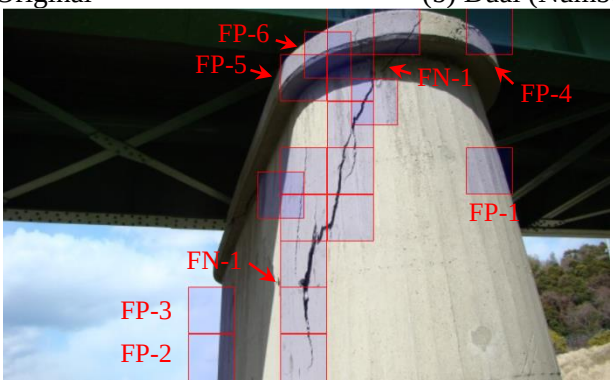
Fig. 13. Test results using dual scanning and neighborhood scanning.



(a) Original



(b) Dual (Number of crack position =28)

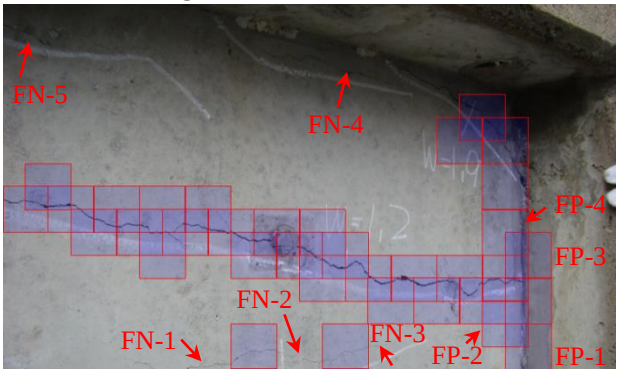


(c) Neighborhood (Number of crack position =19)

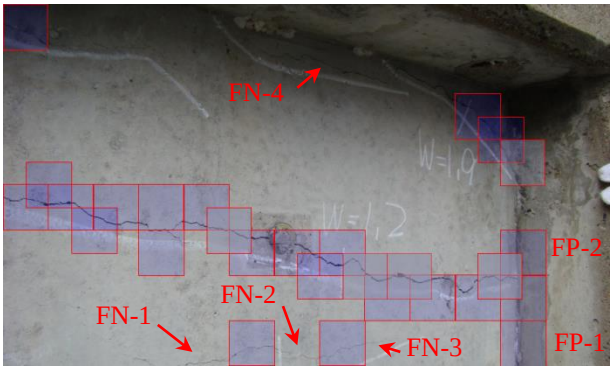
Fig. 14. Crack detection on field image 1.



(a) Original



(b) Dual (Number of crack position =34)

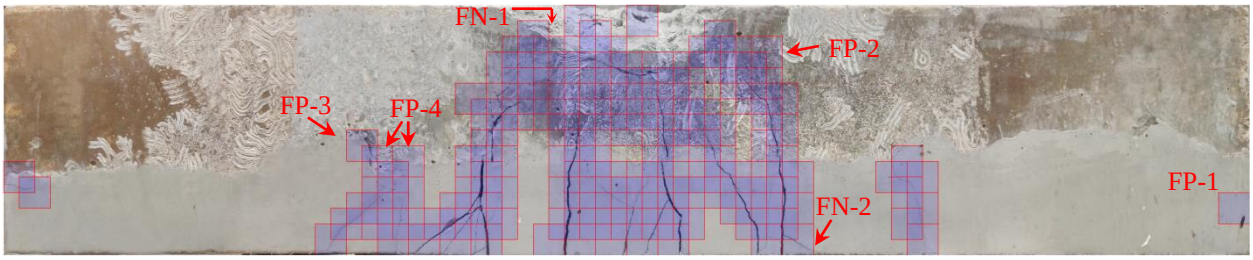


(c) Neighborhood (Number of crack position =28)

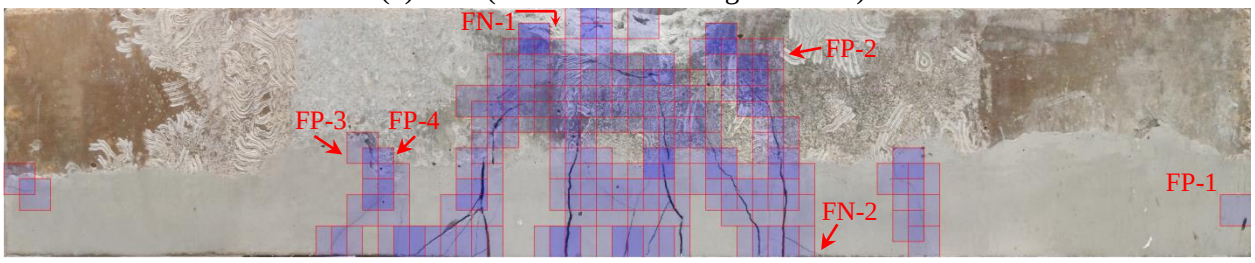
Fig. 15. Crack detection on field image 2



(a) Original

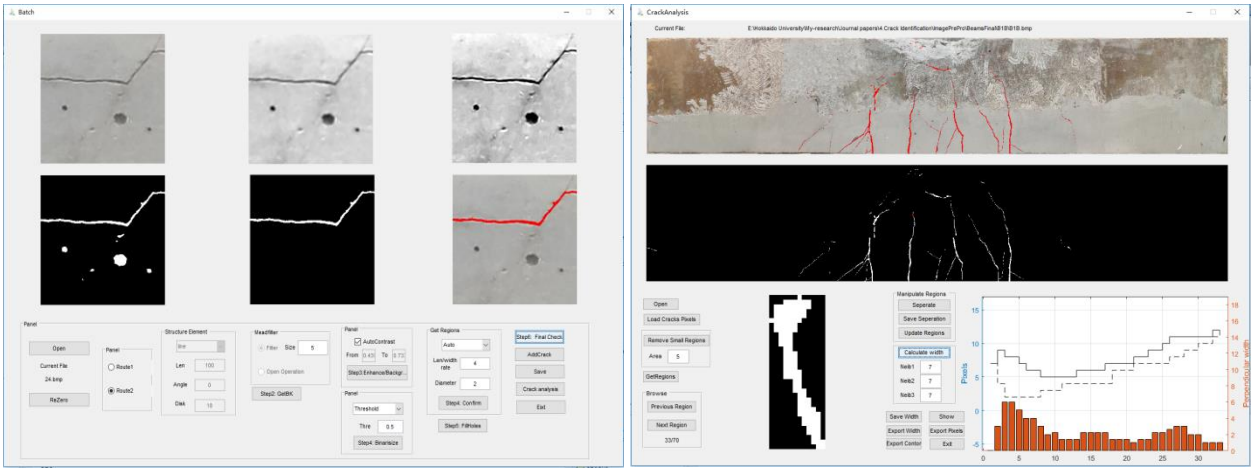


(b) Dual (Number of crack regions =139)



(c) Neighborhood (Number of crack regions =122)

Fig. 16. Crack detection on experimental image



(a) (b)

Fig. 17. The developed application for post-processing: (a) processing for every regions; (b) crack analysis for each raw image

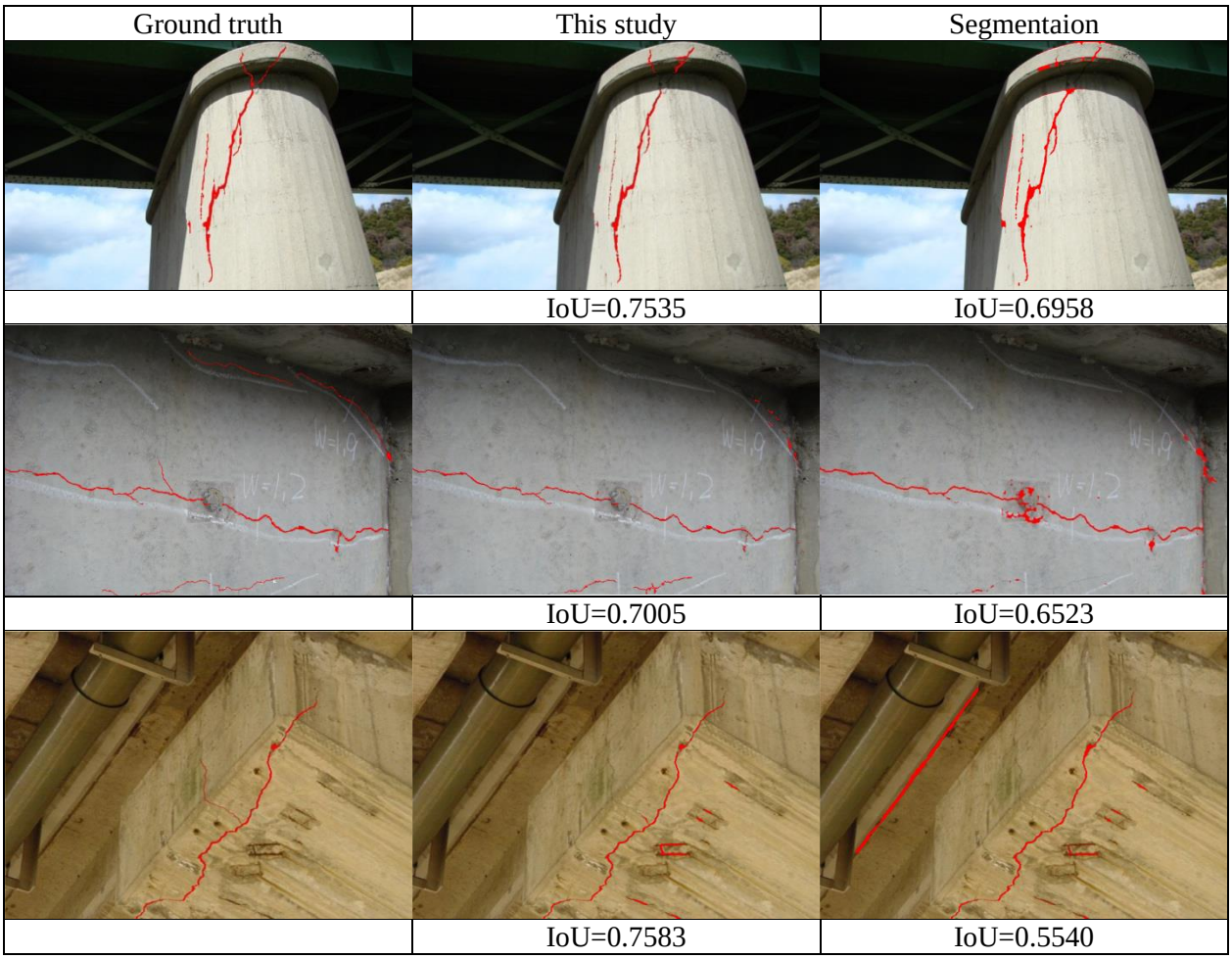


Fig. 18. Comparative studies using our system and a picel-level segmentation framework.



(a) (b)

Fig. 19. (a) Crack scale; (b) Crack measuring.

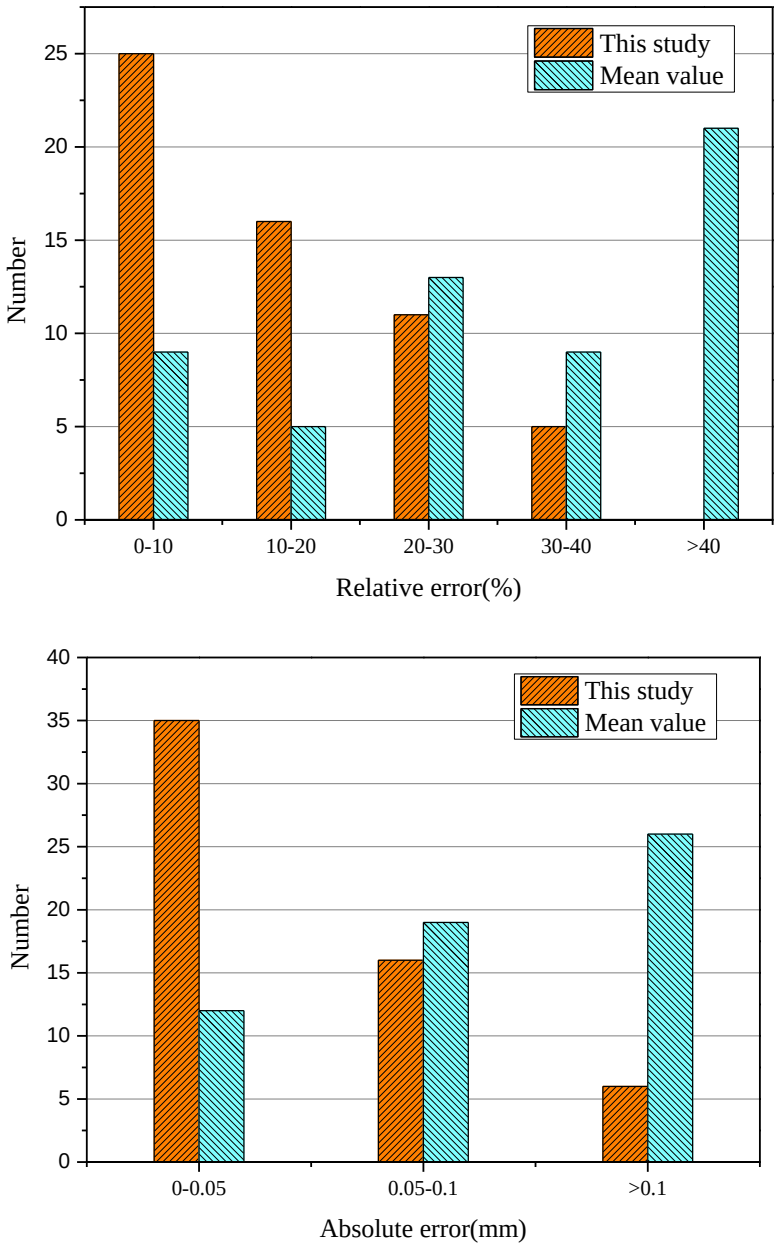


Fig. 20. Relative and absolute error distributions.

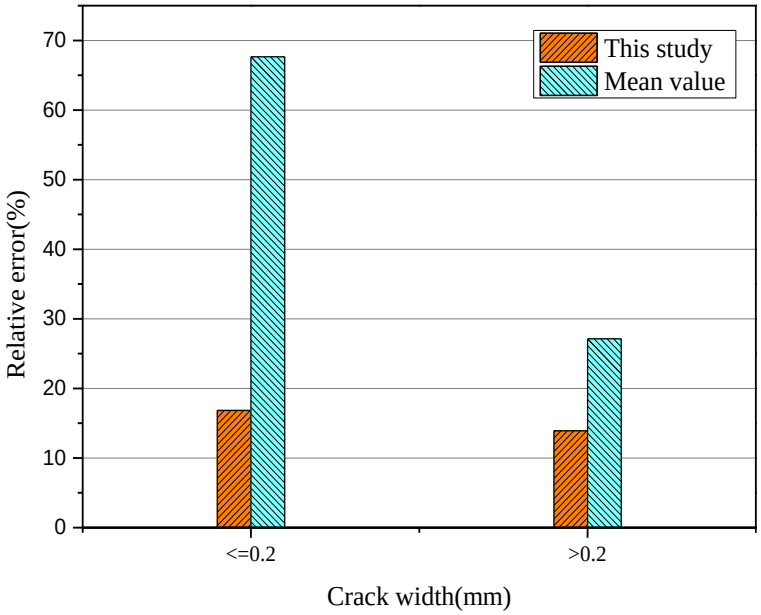
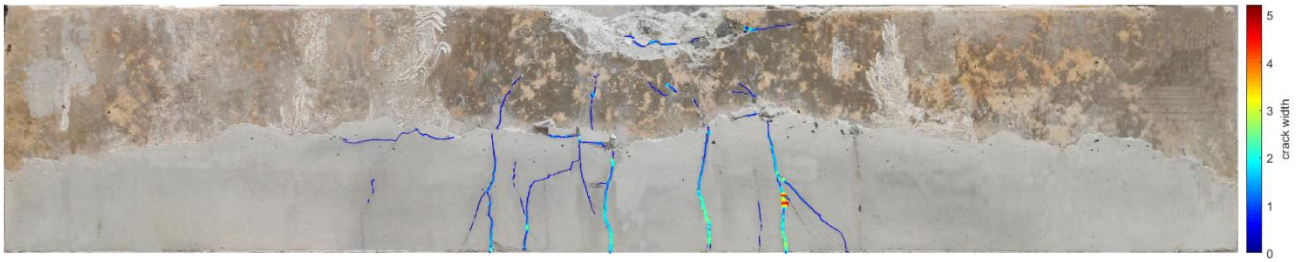
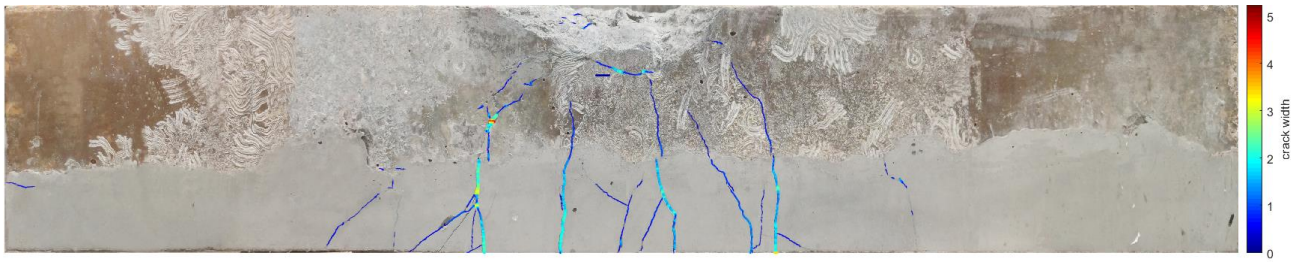


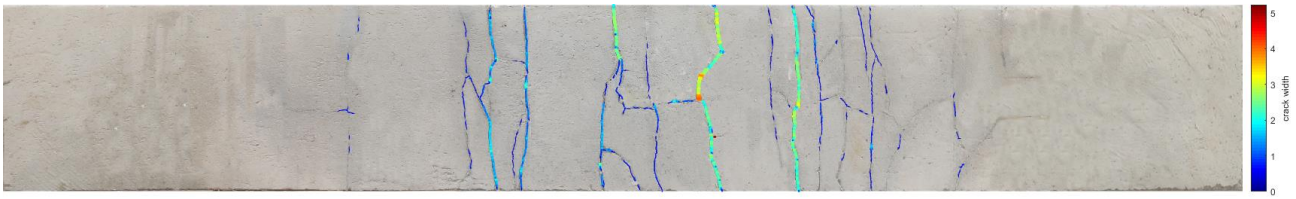
Fig. 21. Relative error of those two methods, broken down by crack width of 0.2 mm



(a)

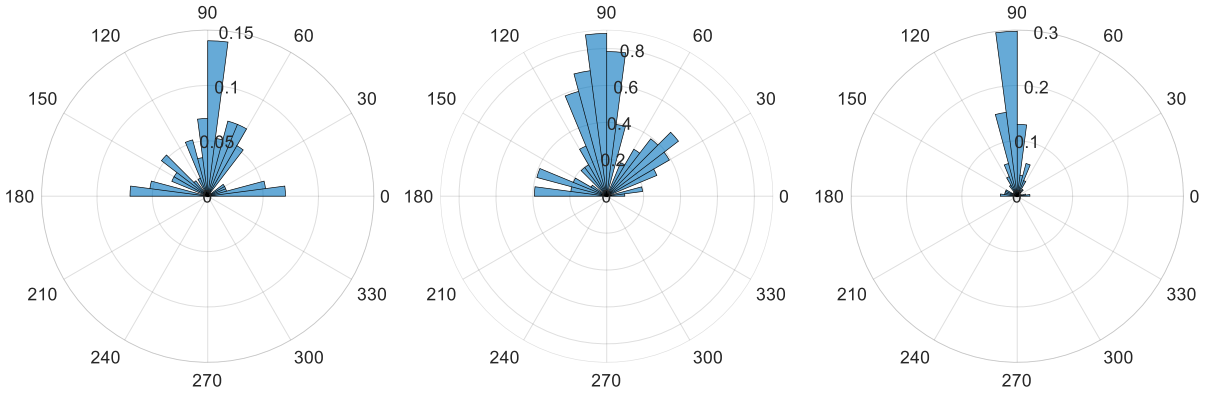


(b)



(c)

Fig. 22. Superimposed images of crack width distribution: (a) front; (b) back; (c) bottom.



(a) (b) (c)

Fig. 23. Cracks directions statistics from Figs. 22 (a), (b), and (c).

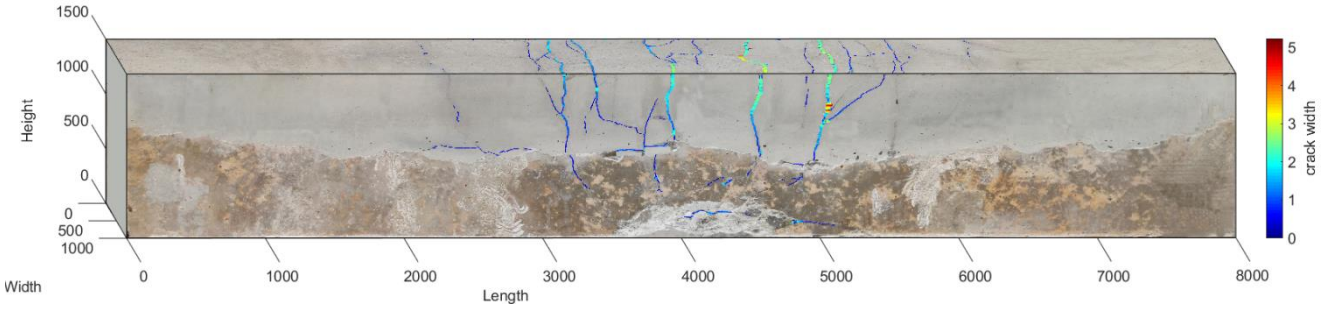
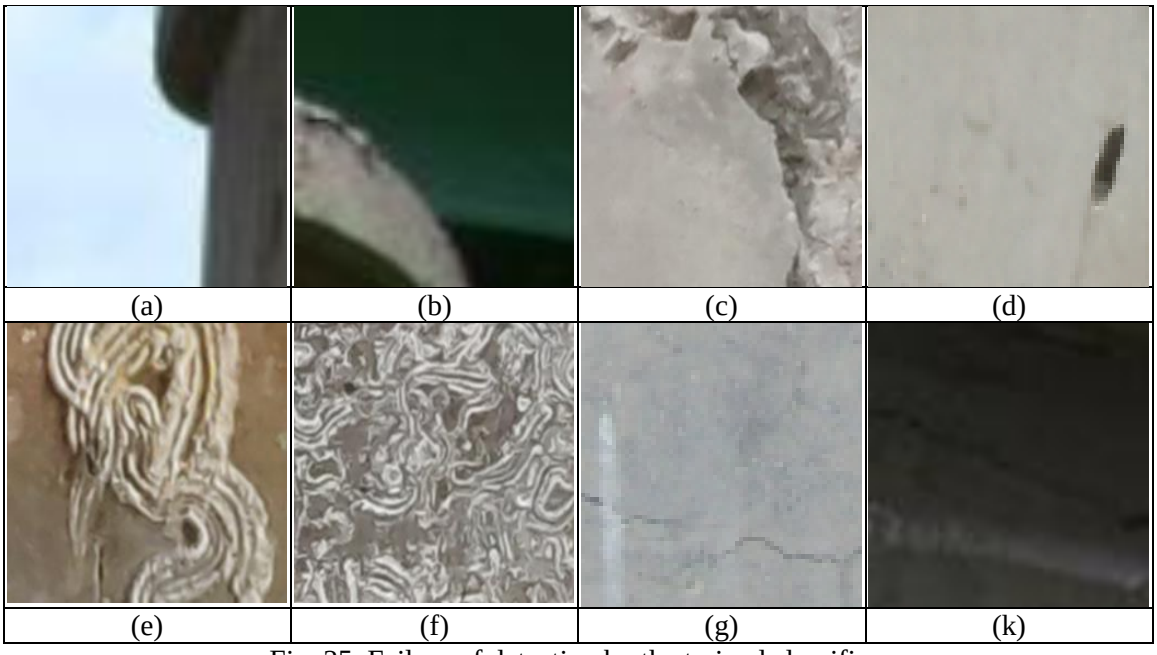


Fig. 24. 3D visualization of a beam



(a)	(b)	(c)	(d)
(e)	(f)	(g)	(k)

Fig. 25. Failure of detection by the trained classifier.

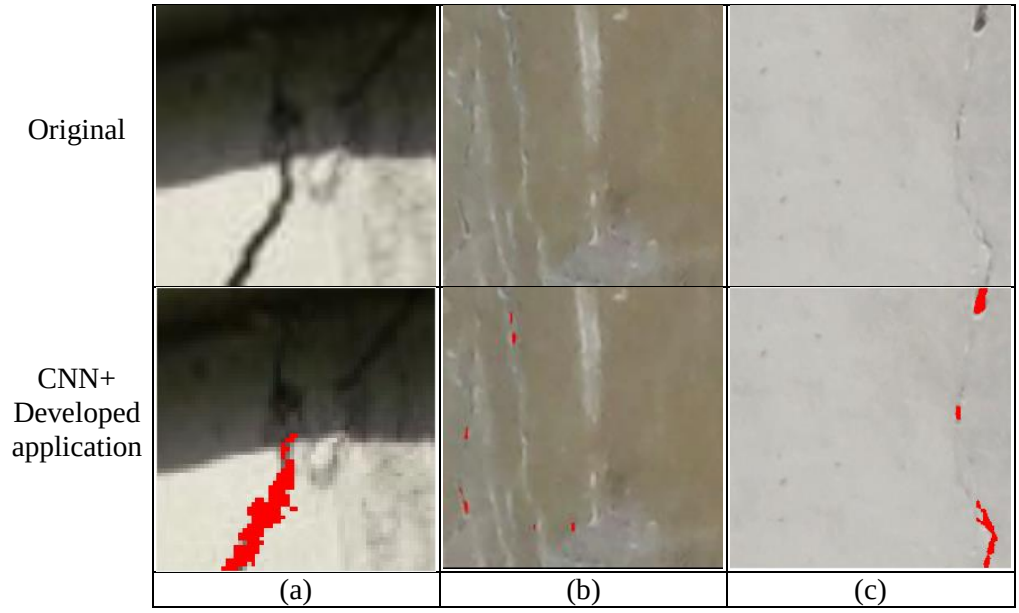


Fig. 26. The superimposed images by the trained CNN and the developed application.

Table 1. Detailed specifications of the AlexNet

Layer	Kernel size (Height×Width×Depth)	Stride	Pad	Output size (Height×Width×Depth)
Input	-	-	-	227×227×3
Conv1	11×11×3	4	0	55×55×96
Pool1	3×3	2	0	27×27×96
Conv2	5×5×3	1	2	27×27×256
Pool2	3×3	2	0	13×13×256
Conv3	3×3×3	1	1	13×13×384
Conv4	3×3×3	1	1	13×13×384
Conv5	3×3×3	1	1	13×13×256
Pool3	3×3	2	0	6×6×256
Fc1	6×6×256	-	-	1×1×4096
Fc2	-	-	-	1×1×4096
Fc3	-	-	-	1×1×4096
SM	-	-	-	1×1×2

Algorithm 1 Crack width calculation.

Input: *neighborhood* δ, $P(x_1, y_1)$, S_1, S_2
Output: *distance d*
for *i between* $x_1 - \delta$ *and* $x_1 + \delta$
    dataset $\leftarrow\{S_1(i); S_2(i)\}$
end for
l ← *linear regression(dataset)*
l' ← *line perpendicular to l through P*
P' ← *intersection of l' and* S_2
d←*distance(P,P')*

Table 2. Specification of raw images and generation of database

Source	Raw Images				Database	
	No	Size	Training/Validation	Testing	Pixel	Training/Validation
Experiment	69	10240×2048	58	11	256×256	30,480
Field	81	2592×4608	69	12		

Table 3. Performances of the four pre-trained CNN configurations.

	Re	TNR	Pre	ACC	F1	Time per epoch (min)
AlexNet	0.9377	0.9831	0.9819	0.9606	0.9593	24.84
Googlenet	0.9604	0.9736	0.9737	0.9669	0.9670	38.82
Resnet18	0.9661	0.9678	0.9678	0.9689	0.9670	45.33
VGG-16	0.9724	0.9680	0.9683	0.9740	0.9704	346.6

Table 4. Performances of the transfer and fully training of GoogleNet.

	Re	TNR	Pre	ACC	F1
Transfer	0.9603	0.9735	0.9737	0.9669	0.9670
Fully	0.9720	0.9758	0.9761	0.9709	0.9740

Table 5. Performances of the transfer and fully training on SDNET2018

	Re	TNR	Pre	ACC	F1
Transfer(SDNET2018)	0.8905	0.8772	0.8798	0.8839	0.8851
Fully(SDNET2018)	0.8100	0.8004	0.8048	0.8133	0.8074

Table 6. Comparisons of the dual scanning and the neighborhood scanning

	Rec	TNR	Pre	ACC	F1	Average Time(s)
Neig	0.9533	0.9526	0.9651	0.9555	0.9592	6.48
Dual	0.9517	0.9587	0.9672	0.9551	0.9594	7.11

Table 7. Comparison of the developed system and the previous framework

Image	IoU		Time cost	
	This	Previous	This	Previous
im1	0.7535	0.6958	7.0209	7.6364
im2	0.7005	0.6523	7.0114	7.6142
im3	0.8184	0.8301	7.0030	7.6435
im4	0.8350	0.8385	7.0373	7.6772
im5	0.7583	0.5540	7.0436	7.6369
im6	0.8427	0.8456	7.0591	7.6207
im7	0.8266	0.8192	7.0398	7.6466
im8	0.8389	0.8270	6.9890	7.6141
im9	0.7544	0.7573	6.9825	7.6854
im10	0.8301	0.8211	7.0557	7.6059
im11	0.8030	0.8015	7.0241	7.5988
im12	0.8265	0.8165	7.0181	7.6024
im13	0.8206	0.8292	12.2009	13.2951
im14	0.7585	0.7603	12.1531	13.3312
im15	0.8441	0.8127	12.1458	13.3013
im16	0.8069	0.8184	12.1485	13.3193
im17	0.8424	0.8159	12.1506	13.3248
im18	0.8228	0.8123	12.1455	13.2847
im19	0.7888	0.7652	12.1369	13.3349
im20	0.7933	0.7993	12.1436	13.3010
im21	0.8359	0.8042	12.2002	13.3236
im22	0.7840	0.7956	12.1751	13.3112
im23	0.8076	0.8165	12.1552	13.3408

Table 8. Comparisons of crack width measurement methods

No	This study (mm)	Mean (mm)	Measured (mm)	Relative Error of predicted values (%)	Relative error of mean values (%)	No	This study (mm)	Mean (mm)	Measured (mm)	Relative Error of calculated values (%)	Relative error of mean values (%)
1	0.30	0.38	0.22	35.08	71.15	30	1.60	1.56	1.15	38.93	35.55
2	0.25	0.34	0.23	8.96	49.32	31	0.43	0.50	0.34	26.87	46.54
3	0.23	0.32	0.25	8.34	29.84	32	0.44	0.49	0.37	19.46	33.40
4	0.40	0.43	0.34	18.10	25.78	33	0.48	0.54	0.41	17.01	31.72
5	0.35	0.43	0.30	17.63	42.97	34	0.15	0.26	0.21	27.50	21.95
6	0.08	0.21	0.06	33.60	245.30	35	0.89	0.76	0.65	35.39	16.79
7	0.11	0.23	0.15	29.15	52.54	36	0.26	0.36	0.27	2.54	32.78
8	0.22	0.34	0.22	1.58	56.65	37	0.73	0.76	0.74	0.77	2.89
9	0.93	0.98	0.72	28.61	36.42	38	0.21	0.29	0.18	18.96	63.14
10	0.40	0.51	0.32	24.65	60.87	39	0.73	0.69	0.67	9.48	2.78
11	0.23	0.34	0.23	0.31	48.33	40	0.30	0.38	0.28	6.97	36.93
12	0.36	0.31	0.30	19.31	1.70	41	0.32	0.41	0.34	6.34	20.26
13	0.24	0.33	0.27	12.18	21.83	42	1.51	1.48	1.19	26.42	24.09
14	0.26	0.34	0.23	14.41	48.67	43	0.16	0.24	0.16	2.63	51.33
15	0.25	0.33	0.26	4.22	25.39	44	0.16	0.28	0.17	6.21	63.18
16	0.32	0.39	0.31	3.03	27.28	45	0.88	0.76	0.92	3.75	16.93
17	0.10	0.25	0.16	34.94	55.32	46	0.22	0.30	0.19	18.13	56.39
18	0.13	0.22	0.17	23.60	26.78	47	0.46	0.52	0.51	8.33	2.70
19	0.17	0.27	0.17	2.73	56.94	48	0.40	0.39	0.35	15.71	11.79
20	0.69	0.70	0.67	3.88	5.14	49	0.18	0.29	0.22	15.69	36.05
21	0.65	0.60	0.63	2.93	3.56	50	0.26	0.34	0.28	7.32	22.48
22	0.63	0.65	0.52	22.57	26.11	51	0.13	0.25	0.15	15.99	68.03
23	0.40	0.50	0.41	2.99	21.28	52	0.12	0.22	0.14	12.20	58.52
24	0.45	0.53	0.37	22.16	42.74	53	0.25	0.32	0.27	8.70	19.78
25	0.65	0.65	0.60	7.70	7.04	54	1.76	1.69	1.52	15.97	11.33
26	0.41	0.51	0.33	23.70	53.51	55	0.64	0.74	0.56	12.94	32.41
27	0.20	0.24	0.18	12.77	35.15	56	0.21	0.27	0.27	21.65	1.19
28	0.18	0.29	0.20	7.98	46.83	57	0.21	0.21	0.21	0.76	1.65
29	0.28	0.36	0.29	1.27	26.26	**Average**	**0.42**	**0.48**	**0.38**	**14.58**	**36.37**