Comparison of Deep Convolutional Neural Networks and Edge Detectors for Image-Based Crack Detection in Concrete

Sattar Dorafshan[1a], Robert J. Thomas[b], Marc Maguire[a]

[a]*Department of Civil and Environmental Engineering, Utah State University, Logan, UT, USA*

[b]*Department of Civil and Environmental Engineering, Clarkson University, Potsdam, NY, USA*

Abstract

This paper compares the performance of common edge detectors and deep convolutional neural networks (DCNN) for image-based crack detection in concrete structures. A dataset of 19 high definition images (3420 sub-images, 319 with cracks and 3101 without) of concrete is analyzed using six common edge detection schemes (Roberts, Prewitt, Sobel, Laplacian of Gaussian, Butterworth, and Gaussian) and using the AlexNet DCNN architecture in fully trained, transfer learning, and classifier modes. The relative performance of each crack detection method is compared here for the first time on a single dataset. Edge detection methods accurately detected 53–79% of cracked pixels, but they produced residual noise in the final binary images. The best of these methods was useful in detecting cracks wider than 0.1 mm. DCNN methods were used to label images, and accurately labeled them with 99% accuracy. In transfer learning mode, the network accurately detected about 86% of cracked images. DCNN methods also detected much finer cracks than edge detection methods. In fully trained and classifier modes, the network detected cracks wider than 0.08 mm; in transfer learning mode, the network was able to detect cracks wider than 0.04 mm. Computational times for DCNN are shorter than the most efficient edge detection algorithms, not considering the training process. These results show significant promise for future adoption of DCNN

[1] Corresponding author, Address: Department of Civil & Environmental Engineering
Utah State University, Logan, Utah 84322-4110. Tel: +1-435-757-3740
Email addresses: sattar.dor@aggiemail.usu.edu (S. Dorafshan), rthomas@clarkson.edu (R.J. Thomas), m.maguire@usu.edu (M. Maguire)

methods for image-based damage detection in concrete. To reduce the residual noise, a hybrid method was proposed by combining the DCNN and edge detectors which reduced the noise by a factor of 24.

Keywords: Concrete, crack detection, deep learning, neural network, edge detection, image processing, vision-based, structural health monitoring

1. Introduction

At least a third of the more than 600,000 bridges in the United States include a concrete superstructure or wearing surface [1]. Routine inspections of concrete bridges are conducted periodically to assess overall condition and to identify surface cracking or other degradation [2]. Manned inspections of this type are costly, time consuming, and labor intensive [3] [4] [5]. Unmanned and autonomous inspections are a potentially viable alternative to manned inspections [5] [6] [7] [8] [9] [10]. Inspections performed by robots or unmanned aerial systems (UAS) are typically image-based, meaning that the inspection platform takes images that are then processed and/or reviewed by an inspector. Previous literature demonstrates several successful applications of image-based inspections to detect cracks [11, 12], spalls [13, 14], delaminations [14, 15, 16], and corrosion [17] in concrete bridges.

Image-based inspections of this type can be performed in three general ways: Raw image inspection, image enhancement, or autonomous image processing. Raw image inspection means that the inspector views the images taken during the inspection without any additional processing [5, 18]. The number of images collected depends on a number of factors, but is commonly in the hundreds of thousands [5, 18]. Manual identification of flaws in such large images sets is time consuming and prone to inaccuracy due to inspector fatigue or human error. Enhanced image inspection refers to the use of some image processing algorithm to make it easier to identify flaws in inspection images. This is typically performed using one of several edge detection algorithms, which greatly magnify the visibility of cracks within images. In doing so, the aforementioned problems with inspector fatigue can be mitigated to some degree. Finally, autonomous

image processing refers to the use of an algorithm that detects cracks within images. This is typically accomplished using machine learning algorithms or other artificial intelligence schemes.

This paper discusses the latter two approaches and compares their performance. Image enhancement methods includes the application of a variety of image processing techniques on visual images to detect cracks including but not limited to morphological operations [19], digital image correlation [20, 21], image binarization [22, 23], percolation model [24], wavelet transforms [25], and edge detectors [12] [27] [29] [33] [34] [36] [37] [38] [36]. The autonomous approach for crack detection on the other hand requires a set of training images to learn the features of cracks. Similarly, several researchers have shown the feasibility of autonomous crack detection in visual images using combined image processing techniques and artificial neural networks [30, 40]. Deep convolutional neural networks (DCNNs) have been recently used for concrete crack detection [41, 42, 43].

Despite the abundance of image-based crack detection studies, direct comparisons between these methods is a gap. Save two noteworthy exceptions, most research focuses on developing new methods for crack detection rather than comparing the performance of existing methods. Abdel-Qader et al. [27] compared the performance of the fast Haar transform, Fourier transform, Sobel filter, and Canny filter for crack detection in 25 images of defected concrete and 25 images of sound concrete. The fast Haar transform was the most accurate method, with overall accuracy of 86%, followed by the Canny filter (76%), Sobel filter (68%), and the Fourier transform (64%). he processing time was not considered in the analysis and the criteria for recoding true of false positives in the binary images were not clear. Lack of definition for metrics such as true positive has seen in the past studies. Mohan and Poobal [44] reviewed a number of edge detection techniques for visual, thermal, and ultrasonic images, but the information presented was from several studies that considered vastly different data sets, and so the results are not directly comparable. A comparison between two edge detectors, Canny and Sobel, and a convolutional neural network is done in [42]. However, the comparison was performed on four images. In addition, the edge detectors were used without pre-processing which is not a very common practice. Another shortcoming of the comparison in

[42] is the lack of accuracy definition of the edge detector results. This paper compares image processing and deep learning techniques together as a reference for future study. which includes a direct comparison of the performance of four common edge detection methods in the spatial domain (Roberts, Prewitt, Sobel, Laplacian of Gaussian) and two in the frequency domain (Butterworth and Gaussian) and an AlexNet-based DCNN in three modes of training (fully trained, transfer learning, and no-training) by applying them to an annotated dataset designated for crack detection.

2. Dataset

The dataset used in this study consisted of 100 images of concrete panels that simulated reinforced concrete bridge decks for the purpose of verifying various non-destructive testing. These panels were constructed previously in Systems, Materials, and Structural Health laboratory (SMASH Lab) at Utah State University. Images are collected with a 16 MP digital single lens reflex camera with 35 mm focal length and no zoom. The target was normal to the axis of the lens at a distance of approximately 0.5 m. The background illumination was in the range 400–1000 lx, as measured by a NIST traceable digital light meter purchased new just prior to measurement. The finest crack width was approximately 0.04mm and the widest was 1.42mm. The original image size was 2592 × 4608 px and the field of view was approximately 0.3 × 0.55 m. Images were stored as JPEG with average file size near 5 MB. In order to comply with the architecture of the DCNN, each original image was divided into 180 sub-images with size of 256 × 256 px. The sub-images were labeled in two categories, 1,574 sub-images with cracks and 16,426 sub-images without cracks. Figure 1 illustrates the studied dataset with one example of high-resolution image, a sub-image labeled as C from the original image if it had a crack, and a sub-image labeled as U from the original image if it did not. For DCNN applications, this dataset was divided into training dataset, validation dataset, and testing dataset as shown in Table 1. The testing dataset was selected randomly from 100 original images. The images in this dataset are a portion of the bridge deck images of the structural defect dataset (SDNET2017 [45]). The sub-images in the testing dataset have also been segmented in the pixel-level as Cp and Up for semantic comparison where Cp stands for pixels with cracks and Up stands for sound pixels.

The results of the pixel-level segmentation on the testing dataset are presented in Table 2. In this table, the Cp ratio stands for the number of pixels in each image labeled as crack to total number of pixels in that image.

3. Edge Detection

In this paper, edge detection refers to the use of filters (edge detectors) in an image processing algorithm for the purpose of detecting or enhancing the cracks in an image such that they can be more easily and efficiently located within a large image dataset. Cracks in a two-dimensional (2D) image are classified as edges, and thus existing edge detection algorithms are likely candidates for crack identification. 2D images are represented mathematically by matrices (one matrix, in the case of greyscale images, or three matrices in the case of red/green/blue color images). An ideal edge is defined as a discontinuity in the greyscale intensity field. Crack detection algorithms can emphasize edges by applying filters in either the spatial or frequency domain. Edge detection algorithms purport to make manual crack detection more reliable. In general, such image processing algorithms follow three steps: (1) edge detection, (2) edge image enhancement, and (3) segmentation (sometimes called binarization or thresholding). Edge detection involves the application of various filters in either the spatial or frequency domain to a grayscale image in order to emphasize discontinuities. Edge image enhancement scales the image and adjusts contrast to improve edge clarity. Segmentation transforms the enhanced edge image into a binary image of cracked and sound pixels.

In the spatial domain, the convoluted image $\boldsymbol{E}$ is the sum of the element-by-element products of the image intensity $\boldsymbol{I}$ and the kernel $\boldsymbol{K}$ in every position in which $\boldsymbol{K}$ fits fully in $\boldsymbol{I}$. For $\boldsymbol{I}_{M\times N}$ (image dimension $M \times N$) and $\boldsymbol{K}_{m\times n}$ (kernel size $m \times n$):

$$\boldsymbol{E}(i,j) = \sum_{k=1}^{m}\sum_{\ell=1}^{n} \boldsymbol{I}(i+k-1, j+\ell-1)\boldsymbol{K}(k,\ell) \quad (1)$$

$\boldsymbol{E}$ is of size $(M-m+1)\times(N-n+1)$. Filters kernels may include x and y components (corresponding to image spatial dimension in horizontal and vertical dimensions), $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$, in which case the edge image $\boldsymbol{E}$ is the hypotenuse of $\boldsymbol{E}_x$ and $\boldsymbol{E}_y$.

Four edge detector filters in the spatial domain were employed in this study: Roberts in x and y directions, denoted as $\boldsymbol{K}_{Rx}$ and $\boldsymbol{K}_{Ry}$ in Eq. 2, Prewitt in x and y directions, denoted as $\boldsymbol{K}_{Px}$ and $\boldsymbol{K}_{Py}$ in Eq. 3, Sobel in x and y directions, denoted as $\boldsymbol{K}_{Sx}$ and $\boldsymbol{K}_{Sy}$ in Eq. 4, and Laplacian-of-Gaussian (LoG) denoted as $\boldsymbol{K}_{LoG}$ in Eq. 5. A 10×10 LoG filter was employed here with standard deviation of $\sigma=2$.

$$\boldsymbol{K}_{Rx}=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \qquad \boldsymbol{K}_{Ry}=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix} \tag{2}$$

$$\boldsymbol{K}_{Px}=\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix} \qquad \boldsymbol{K}_{Py}=\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix} \tag{3}$$

$$\boldsymbol{K}_{Sx}=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \qquad \boldsymbol{K}_{Sy}=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \tag{4}$$

$$\boldsymbol{K}_{LoG}=\nabla^2\big(\boldsymbol{G}(x,y)\big)=\frac{x^2+y^2-2\sigma^2}{4\sigma^4}exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{5}$$

Edge detection in the frequency domain requires transformation of the spatial domain image $\boldsymbol{I}$ into the frequency domain image $\boldsymbol{F}$ by fast Fourier transform (FFT). The edge image $\boldsymbol{E}$ is the element-wise product of the filter kernel $\boldsymbol{K}$ and the frequency domain image $\boldsymbol{F}$:

$$\boldsymbol{E}(u,v)=\boldsymbol{K}(u,v)\odot\boldsymbol{F}(u,v) \tag{6}$$

where u and v are the dimensions of the transformed image in the frequency domain. Two edge detector filters in the frequency domain were employed in this study: Butterworth denoted as $\boldsymbol{K}_B$ in Eq. 7 and Gaussian denoted as $\boldsymbol{K}_G$ in Eq. 8.

$$\boldsymbol{K}_B(u,v)=1-\frac{1}{1+\left[\frac{D(u,v)}{D_0}\right]^{2n}} \tag{7}$$

$$\boldsymbol{K}_G(u,v) = 1 - e^{\frac{-D^2(u,v)}{2\sigma^2}} \tag{8}$$

where $D(u,v)$ is the distance between the pixel (u,v) and the origin of the frequency (the center of the $M \times N$ image) as defined by Eq. 8, D_0 and n are the user-defined parameters to define the order and cut-off frequency in the Butterworth filter; and σ is the user-defined parameter to define the standard deviation of the Gaussian filter.

$$D(u,v) = \sqrt{\left[u - \left(\frac{M}{2}+1\right)\right]^2 + \left[v - \left(\frac{N}{2}+1\right)\right]^2} \tag{9}$$

and $\boldsymbol{K}_B$, and $\boldsymbol{K}_G$, are Butterworth and Gaussian filters.

The scaled edge image $\boldsymbol{E}_{sc}$ is $\boldsymbol{E}$ scaled such that $0 \leq \boldsymbol{E}_{sc} \leq 1$. The enhanced edge image is then:

$$\boldsymbol{E}_e(x,y) = [\boldsymbol{E}_{sc}(x,y) - \min(\boldsymbol{E}_{sc})]\left[\frac{2\sigma_{\boldsymbol{E}_{sc}}}{\max(\boldsymbol{E}_{sc}) - \min(\boldsymbol{E}_{sc})}\right] + \mu_{\boldsymbol{E}_{sc}} \tag{10}$$

where $\min(\boldsymbol{E}_{sc})$, $\max(\boldsymbol{E}_{sc})$, $\sigma_{E_{sc}}$, and $\mu_{\boldsymbol{E}_{sc}}$ are minimum, maximum, standard deviation, and mean of the scaled edge image, respectively. Edge enhancement is a crucial part of the proposed method by improving the segmentation of pixels with cracks from the background pixels. Figure 2 shows an example of the effect of edge enhancement on the final binary image of the proposed algorithm (Sobel edge detector).

The final binary image $\boldsymbol{B}$ is constructed by segmentation, which assigns a value of one to all pixels in which the intensity exceeds some threshold T and a value of zero to all other pixels. In this study, a two level binarization is introduced: the first is based on a pixel intensity threshold T_1 in the enhanced edge image and then based on an area connectivity threshold T_2 on the binary image from the first level. The first threshold operation filters the weak edges from the enhanced edge image (Eq. 11). By applying T_1 the strong edges in the enhanced edge image (80% or stronger than the maximum intensity, $0.8\max(E_e)$) are preserved as cracks. At this point, the strong edges have been identified in the first binary image; however, the surface roughness of the concrete can cause residual noise.

$$T_1 = 0.8\max(E_e) \tag{11}$$

In order to gain more effective segmentations, the morphological operation closing was carried out on the first level binary image. Closing consists of a dilation followed by an erosion using an identical structuring element for both operations (see Figure 3). The purpose of the closing operation is to unify possibly the discrete parts of the crack in the first binary image. Structuring elements define the spatial domain on the binary image in which the morphological operation will be carried out. Circle-shaped structuring elements with generic dimensions were used to perform the closing operation. The radius of the structural element was defined as the minimum Euclidean distance between the centroids of connected components in each binary image. The closing operation on improved the results of each individual edge detector in terms of true positives. Figure 4 shows an example where not applying the closing operation cause the LoG edge detector to miss the more than half the crack after applying the second threshold operation.

The second binarization operation was designed to segment the cracks from the residual noises in the first binary image based on the area of the connected components in the first level binary image (Eq. 12). The connected area $A_c(x, y)$ is the number of contiguous pixels in a connected component, considering eight-neighbor connectivity. $\max(A_c)$ is the area of the largest connected component in the first level binary image. The idea for the area threshold is to control the noise in the final binary image as shown in Figure 5 for the results of the Gaussian high pass filter.

$$T_2 = \max(A_c) \tag{12}$$

4. DCNN

Using direct image-processing techniques for concrete crack detection has several drawbacks. First, the algorithms are tailored for certain images in the studied datasets which affects their performance on new datasets. These algorithms may not be as accurate when tested on new datasets taken in more challenging situations such as low lighting condition, presence of shadows, low quality cameras, etc. Second, the image processing algorithms are often designed to aid the inspector in crack detection and still rely on human judgement for final results [29]. One solution is using machine learning algorithms to analyze the inspection

images [46] [47]. Deep convolutional neural networks (DCNNs) are a type of feedforward artificial neural networks which have revolutionized autonomous image classification and object detection in the past 5 years [48]. A DCNN uses a set of annotated, e.g. labeled, images for training and calculates the learning parameters in the learning layers between the input and output layers thorough thousands to millions iterations.

A number of architectures have been employed to create neural networks providing excellent accuracy on open-source labeled datasets, such as ImageNet and MNIST, in the past 4 years [49] [50] [51]. Each architecture includes a number of main layers. The main layers are composed of sub-layers. The total number of layers defined in a software program, like MATLAB, to build an architecture is referred to as "Programmable Layers" in this study. Krizhevsky [49] proposed one of the first architectures of a DCNN, i.e. AlexNet. This architecture has 8 main layers (25 programmable layers) and was the winner of the image classification competition in 2012 (ImageNet [52]). Szegedy et al. proposed another architecture called GoogleNet with 22 main layers (144 programmable layers) and improved the accuracy by introducing inception module in the learning layers which won the 2014 competition [53]. Deep residual learning neural network, ResNet, was introduced in 2016 [54]. ResNet has 50 and 101 main layers (177 and 347 programmable layers) and was the winner of 2016 competition.

DCNNs have been used in vision-based structural health monitoring in recent years for crack detection [42], road pavement cracks [55, 56], corrosion detection [57, 58], multi-damage detection [41, 59] structural health monitoring [62]. Due to popularity of Unmanned Aerial Systems (UASs) for structural health monitoring and bridge inspection [63] applications of DCNNs in UAS-assisted inspections has begun to attract researchers for more robust non-contact damage detection [43, 64, 65].

In general, DCNN architecture includes an input layer, learning layers, and an output layer [66]. The input layer reads the image and transfers it to the learning layers. The learning layers perform convolution operations, applying filters to extract image features. The output layer classifies the image according to 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 filter values iteratively through back propagation until the desired accuracy is achieved.

DCNN can be used for crack detection in three ways: classification [42], localization [41], or segmentation. The goal of classification is to label each image as cracked or sound. The training and validation datasets comprise pre-classified cracked and sound images. The goal of localization is to determine bounding coordinates that identify the location of a crack within an image. As before, the training and validation datasets include both cracked and sound images, but the cracked images have bounding boxes drawn around the location of the crack. The goal of segmentation is to classify each pixel as cracked or sound, and the training and validation datasets comprise a very large number of pre-classified pixels. The computational intensity of DCNN normally necessitates subdivision of images to reduce computational requirements.

The AlexNet DCNN architecture, illustrated in Figure 6 comprises five convolution layers (C1—C5), three max pooling layers (MP1—MP3), seven nonlinearity layers using the rectified linear unit (ReLU) function (ReLU1—ReLU7), two normalization layers (Norm1—Norm2), three fully connected layers (FC1—FC3), two dropout layers (DP1—DP2), one softmax layer (SM), and one classification layer (CL). Each layer is applied to the image using the convolution operation (Eq. 1). Figure 6 shows the architecture of the AlexNet along with its corresponding filter number and size. The kernel values are determined iteratively through training, but the size, number, and stride of the kernels are predetermined. The nonlinearity layers operate on the result of each convolution layer through element-wise comparison. The ReLU function used for nonlinearity is defined as the maximum value of zero and the input:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{13}$$

Following the non-linearity layer, a max pooling layer introduces a representative for a set neighboring pixels by taking their maximum value. The max pooling layers are essential to reduce the computational time and overfitting issues in the DCNN. After the max pooling layer, one or several fully connected layers are used at the end of the architecture. The fully connected layer is a traditional multi-layer perceptron

followed by a softmax layer to classify the image. The mission of the fully connected layers is to connect the information from the past layers together in way that the softmax layer can predict the results correctly during the training process. The optimum combination is achieved from a process called backpropagation algorithm (partial derivatives of the softmax layer output with respect to weights). The purpose of the softmax layer is to ensure the sum of probabilities for all labels is equal to 1. In addition to these basic layers, a DCNN also includes normalization, dropout, and classification layers. Normalization layer normalizes the response around a local neighborhood to compensate with the possible unbounded activations from the ReLu layer. The dropout layer is a probability-based threshold layer that filters responses smaller than a threshold probability (50% is common). The classification layer is similar to the fully connected layers. For detailed explanations of function of each layer and their interaction, readers can refer to Reference [67]

Three modes are used for applying the network on the training dataset. The first mode is to Fully Train (FT) the network from scratch (FT mode) on the training dataset. In this mode all the weights are assigned with random numbers and the computed through iterations based on the training dataset. Obtaining an annotated dataset for concrete cracks as big as ImageNet is not currently feasible. Even if a large concrete crack dataset was available, training process from scratch could take days to complete on hardware with several graphic processor units (GPUs), and would therefore be prohibitively time consuming. However, it is possible to apply a previously trained network (pre-trained network) on a small dataset and obtain reasonable accuracy [68]. Pre-trained networks can be applied on a new dataset in different ways [69]. These methods are usually referred to as “domain adaptation” in the deep learning literature. One can use an already trained DCNN on the ImageNet dataset as a classifier for new images. This type of domain adaptation is referred to as Classifier (CL mode). In CL mode, only the last fully connected layer needs to be altered to match the target labels in concrete dataset. The network then uses the pre-trained weights and forms a classifier based on the training dataset. Note that no actual training happens when CL mode is used. Another studied domain adoption method is to partially retrain a pre-trained network and modify the layers

according to a new dataset. This approach is called fine-tuning or transfer learning (TL mode). In the TL mode, the network has to be re-trained since both classifier and weights have to be updated based on the new dataset. In the TL mode, the weights of the lower-level layers (closer to the input image layer) are preserved. These weights are computed from training on millions of images and consist of generic feature extractors such as edge detectors. Therefore, the determined lower-level weights can be applied on any dataset for feature extraction. On the other hand, the classifier layers (close to end of network) are more sensitive to the training dataset and its labels. To adjust the network to the new dataset, the weights in the high-level layers are updated through training on the new dataset.

5. Experimental Program

5.1. Computational Resources

All computations were performed on a desktop computer with 64-bit operating system, 32 GB memory, and 3.40 GHz processor running a GeForce GTX 750 Ti graphics processing unit (GPU). Image processing was performed in MATLAB.

5.2. Edge Detection

The testing dataset of 319 C and 3101 U sub-images was iteratively processed using each of the six edge detection schemes discussed in Section 3. Unlike the past studies [30, 26, 62], the metrics to evaluate the performance of each edge detector was defined very clearly on a pixel level. The final binary images were compared to the ground truth. True positive (TP) is when the edge detector identified a pixel on the crack pixels (Cp). False negative is when the edge detector did not identify a pixel on the crack pixels (Cp). True negative (TN) is when the edge detector did not identify a pixel on the sound pixels (Up), and false positive is when the edge detector identified a pixel on the sound pixels (Up). Note all comparisons were performed on the final binary images produced by each edge detector. Figure 7 shows examples of how metrics are calculated: (a) the original image is segmented into 1,582 Cp pixels (highlighted) and 63,954 Up pixels, (b) the final binary image super imposed on the original image, Roberts edge detector, identified 2,276 Cp

pixels (highlighted) and 63,260 Up pixels, (c) 1,367 pixels in the final binary image were TP, (d) 215 pixels in the final binary image were FN, (e) 63,046 pixels in the final binary image were TN, and (f) 909 pixels in the final binary image were FP. The metrics in the Figure 7c through Figure 7f are shown in white. Note that for U sub-images, TP and FN are meaningless and only TN and FP are recorded.

The team then rated each edge detection scheme in terms of true positive rate (TPR), true negative rate (TNR), accuracy (ACC), positive predictive value (PPV), negative predictive value (NPV), and F1 score, defined as follows

$$TPR = \left(\frac{TP}{TP+FN}\right) \tag{14}$$

$$TNR = \left(\frac{TN}{TN+FP}\right) \tag{15}$$

$$ACC = \left(\frac{TP+TN}{TP+FN+TN+FP}\right) \tag{16}$$

$$PPV = \left(\frac{TP}{TP+FP}\right) \tag{17}$$

$$NPV = \left(\frac{TN}{TN+FN}\right) \tag{18}$$

$$F1 = \left(\frac{2TP}{2TP+FN+FP}\right) \tag{19}$$

In addition, missed crack width (MCW), and computational time (T) are also compared between different edge detectors. MCW is defined as the coarsest crack that went undetected by a particular edge detection scheme, as determined by crack width measurement using a crack width microscope with 0.02 mm resolution. Computational time is defined as the average processing time for ten runs of a particular edge detection scheme, normalized by the number of images (180 sub-images).

5.3. DCNN

Crack detection using DCNN was performed by classification of sub-images in the fully trained, transfer learning, and classifier modes. A total of 12,809 sub-images (1,129 labeled C and 11,680 labeled as U),

were selected at random for inclusion in the training dataset, and 1,771 (125 labeled as C and 1,646 labeled as U) were selected for the validation dataset. The remaining 3,420 sub-images (319 labeled as C and 3101 labeled as U) made up the testing dataset.

Batch size number and validation criterion determine the number of iterations in training process. Larger batch sizes result in faster convergence, but batch size is limited by the available GPU memory. The selected batch size was 10. The training dataset has 12,809 sub-images. Number of iterations to cover all sub-images was simply calculated by dividing the total sub-images to the batch size, i.e. 1281 iterations. This number of iterations is known as an epoch. A maximum of 30 epochs were considered for back propagation on the network, meaning that the network performs as many $30 \times 1281 = 38{,}430$ iterations to finish the training. The network was set to stop iterating once the accuracy in the validation dataset stopped improving in three consecutive epochs. If the validation criterion is not met by the end of 30th epoch, more iterations cycles should be considered for the training.

The network in each mode is used to classify the sub-images in the testing dataset and the results are compared to the ground truth. TP is when the network correctly labeled a sub-image as C, and a FN when the network failed to do so. A TN is when the network correctly labeled a sound sub-image as U and a FN when the network labeled a sub-image as C in a sound sub-image. TPR, TNR, ACC, PPV, NPV, and F1 are calculated according to Eq. 14 through Eq. 19. T and MCW are evaluated in the same manner as the edge detector approach except that the training time is not considered when calculating the T for DCNN.

6. Results and Discussion

6.1. Edge Detection

A summary of results for the six edge detectors applied on the C class and U class sub-images are shown in Table 3 and Table 4, respectively. The metrics for comparison are shown Figure 8a in terms of TPR, PPV, and in Figure 8b in terms of TNR, ACC, and NPV. The latter metrics were significantly affected by the data imbalance between Cp and Up pixels. Nevertheless, the evaluated metrics in this paper are on the

pixel-level which makes the comparison unique compared to previous crack detection studies. LoG produced the highest TPR with 76% followed by Sobel and Prewitt with 76% and 69%. In the spatial domain, Robert edge detector produced lowest TPR, 53%, which was still higher that the TPRs produced by frequency domain edge detector, where Butterworth detected 41% and Gaussian detected only 31% of the crack pixels. LoG edge detector also produced the highest PPV, 60%, followed by Sobel and Prewitt with 56% and 54%. Gaussian high pass filter had only 18% PPV which was the lowest among the studied methods. F1 scores ranged from 23% in sub-images segmented by Gaussian high pass filter to 68% in sub-images segmented by LoG. Roberts and Gaussian high pass filter produced the lowest TNR values, 96% and 97%, respectively and the lowest ACC, both 95%. As for NPV, the lowest values were 95% and 96% when Gaussian and Butterworth edge detectors were used, respectively. Again LoG was the most accurate, 98%, and produced the highest TNR=99% and NVP=99.5%. The difference in metrics in Figure 8b is only 2%-4% but note that these metrics are affected by the gigantic class imbalance between Cp and Up pixels (only 2% of the pixels were Cp). To see this difference better, percentage of reported FP pixels per sub-image, noise ratio (NR), for each edge detector is shown in Figure 8c. To calculate the noise ratio, first the average FN for each method was calculated by dividing total number of FNs to the number of sub-images in each class, 319 in C class, and 3101 in U class. The NR is then calculated as the average FNs divided by total number of pixels in each sub-image, i.e. 256×256.

As seen for sub-images in C class NR values, 2.4% on average, were almost half of the ones in the U class, 5.3% on average. This is due to the fact that the proposed methodology for crack detection is based on the assumption that there is a crack in the investigated image and it is the largest connected component in the first level binary image. Therefore, noise and irrelevant objects are preserved in the final binary image in U class as FN. In addition, the LoG edge detector produced the lowest NR values, 1.1% in the C class and 4.5% in the U class while Roberts and frequency domain detectors were the worst ones in both classes.

Factoring Roberts, overall the spatial domain edge detectors produced better binary images for crack detection compared to frequency domain ones. The same trend can be seen for values of T in Table 3 and

Table 4 where the fastest method was LoG. Finally, LoG detected finer cracks than the rest of studied method with MCW of 0.1 mm. Figure 9 shows an example of crack detection using different edge detectors along with the original image and ground truth. LoG edge detector performed better than all the other studied detectors in all considered metrics.

6.2. DCNN

6.2.1. Training and Validation

Figure 10 shows the achieved accuracy of the DCNN under fully trained and transfer learning during training and validation. In fully trained mode, the validation criterion was met after 14 epochs (17934 iterations), which required 6,200 seconds processing time. The resulting validation accuracy was 97.50%. In transfer learning mode, the validation criteria were met after 7 epochs (8967 iterations), which required 4,100 seconds processing time. In classifier mode, the classifier was constructed in 299 seconds and achieved 98.1% accuracy on the validation dataset.

6.2.2. Testing

Table 5 summarizes the performance of DCNN crack detection in the testing dataset. In general, the DCNN crack detection algorithms performed exceedingly well compared to the traditional detectors. In fully trained mode, the algorithm scored 212 TPs out of 319 cracked sub-images and 3099 TNs out of 3,101 sound sub-images. In transfer learning mode, the algorithm scored more TPs but also scored more FPs. The network in the CL mode performance in terms of TP and TN were in the middle of the FT and TL modes (TP=267 and TF=52).

In all three cases, the accuracy matched or exceeded 97%. However, the TL mode had NPV=99%, F1=89%, and ACC=98% which were the highest among the studied modes. The highest positive predictive value was in the FT mode (PPV=99%) while TL mode produced only PPV=92%. The CL mode produced the highest FPs which lead to the lowest NPV of 98% among the studied modes. The metrics are shown in Figure 11.

As seen the most tangible difference were observed in TPR, PPV, and F1 scores among different metrics since they are more affected by the TPs and C class had considerably less sub-images.

The MCW for fully trained and classifier modes was 0.08 mm. In transfer learning mode, the missed crack width was 0.04 mm. Figure 13 shows fully trained, transfer learning, and classifier DCNN results for a sub-image containing a 0.08 mm crack. As shown in the figure, the 0.08 mm crack was detected only in transfer learning mode, and went undetected in fully trained and classifier modes. The computational time was similar for all three DCNN modes were comparable (2.65-2.81 seconds per 180 sub-images). However, the network in the FT mode required more time for training due to more performed iterations compared to the TL mode, which was expected. In the authors experience, using an AlexNet-based network in TL mode can be up to 50% less time-consuming than the FT mode on concrete image dataset [37, 39]. On the other hand, the network on the CL mode has the advantage of not relying on the training and can be considered the fastest way of testing the network on new datasets. The absence of training in CL mode, however, adversely affected the TNR, ACC, and PPV of the network, which is also an expected outcome [37]. Transfer learning mode was the most accurate and detected the finest cracks, but also took the longest computational time.

Figure 13a through c show representative results for DCNN in fully trained, transfer learning, and classifier modes, respectively. Since the objective is to find the cracks, sub-images in the U class are shaded and sub-images in the C class are shown clearly. Incorrectly labeled sub-images (FN and FP) are identified using a box indicating such.

6.3. Comparison

As discussed before, the results presented in Table 3 and Table 4 for edge detectors and in Table 5 for DCNNs are not directly comparable because DCNN results consider sub-images while edge detection results were based on the pixels. However, comparison is possible since the same sub-images and metrics were used to evaluate both approaches. These results are given in Table 6.

All of the methods tested here performed better on sound sub-images than on cracked sub-images (i.e., TN > TF), and so the metric numbers skewed high. For example, only 32% of cracked pixels (Cp) were detected using the Gaussian edge detection scheme. Nevertheless, since more than 97% of sound pixels were correctly detected, the reported accuracy was ACC=95% which is misleading because the PPV for this edge detector was only 18%, which shows its inefficiency. Several noteworthy results become apparent. First, while the previous section claimed that there was no clear winner between DCNN in fully trained and transfer learning modes, the true positive rate for transfer learning was 20% higher than for fully trained. At the same time, the true negative rate for transfer learning was only one percent lower than for fully trained. This, combined with smaller missed crack width and similar computation time requirements, make transfer learning a clear winner among DCNN modes. F1 scores and PPV values were significantly for DCNN in all modes were significantly greater than the edge detector techniques.

This analysis also shows that DCNN methods performed better at image based concrete crack detection than any of the edge detection methods (expect for FT mode). The LoG edge detector exhibited the highest true positive rate of all six edge detectors, accurately identifying nearly 79% of cracked pixels. LoG also detected the finest cracks of any edge detector, with MCW of 0.1 mm. The TPR among DCNN methods was about 86% and 84% in TL and CL modes, respectively, which was a significant improvement over LoG. In addition, the TFR for the DCNN approach had superiority over the edge detectors due to the high NR ratios (refer to Figure 8c). Furthermore, DCNN methods were able to detect finer cracks than edge detectors. In fully trained and classifier modes, the MCW was 0.08 mm, a marginal improvement over LoG. In transfer learning mode, the MCW was an impressive 0.04 mm.

Computational times also show the superiority of DCNN over edge detectors; computational time was almost 50% less for the DCNNs over edge detectors. However, crack detection using DCNN requires time for training (in FT and TL modes) and classifier construction (in CL mode), which are not taken into account when reporting the computational time. The assumption is that, in the future, pre-trained DCNN will be available for this purpose, so it is not necessarily appropriate to include training time in this comparison. In

fact, DCNN can be trained using a very large dataset with images of varying quality (e.g., resolution, lighting condition, focus), making it more robust and applicable to most situations. Edge detectors are typically manually tuned to maximize performance for a particular dataset or subset, diminishing their robustness.

These results highlight the significant promise of DCNN methods for image based crack detection in concrete. The evidence presented here shows that edge detection methods—which represent the current state of practice—perform reasonably well. DCNN methods provide autonomous crack detection and provide significant performance enhancements over edge detection schemes. The results presented here for DCNN are only a preliminary step in the development of DCNN methods for concrete crack detection. Future work will demonstrate the use of more advanced DCNN for the same problem in the hopes that more advanced networks will provide even better crack detection performance.

The reader should note that the results presented here are for high quality images taken in good lighting and free of vibration. The extension of these results to noncontact image-based inspection and damage detection will require application of the same methods to images with imperfections resulting from poor lighting, vibration, or other issues [43]. This work is ongoing, but the results presented here show promise for autonomous crack detection in concrete structures using noncontact image-based methods.

Despite being recently introduced to structural health monitoring and inspection, DCNNs have improved the vision-based structural defect detection. This study shows the superiority of an AlexNet DCNN over traditional edge detectors for concrete crack detection. The performance of the network can be further enhanced if more powerful architectures such as GoogleNet or RestNet are implemented for crack detection. Unlike edge detectors, the DLCCNs can be used for any types of defect in structures, if enough annotated images are available for training. Formation an annotated image dataset for structural defects, such as ImageNet, is vital for further applications of DCNNs in structural engineering. With this dataset available, new architectures can be developed to focus on finding a handful of structural defects instead of 1000 different objects, which will reduce the computational time associated with training process. In

addition, domain adaptation methods such as transfer learning, will be more effective if the network is previously trained on the structural defects dataset. Improving the performance of domain adaptation techniques makes real-time defect detection in robotic vision-based inspections feasible. In other words, a pre-trained DCNN on the structural defect dataset, can be directly used to accurately classify new images taken by an unmanned aerial system to different structural defects as the inspection is taking place.

7. Hybrid Crack Detector

Unless semantic networks are used for crack detection, edge detectors are still providing segmentation in the pixel level. This information puts the edge detector in favor of the DCNN for fine monitoring and measurements of cracks but creating the training dataset with classified pixels can be very time consuming and challenging. On the other hand, the sole use of edge detectors has the disadvantage of residual noise or non-crack objects misidentified as cracks. Even with the most effective edge detector, LoG, there was more than 4% of TN (combined of FNs of the images in both C class and U class) which is 9,457,066 sound pixels identified as cracks in the testing dataset. Figure 14 shows examples of TN (highlighted in red) in the three C class sub-images after the final binary image from the LoG edge detector was super-imposed on the original images.

Since the DCNN in FT mode provided such accurate classification for the U class sub-images, only two cases of FP, the network was first used to label all the sub-images in U and C classes. No edge detector was applied on the sub-images identified as U class by the network. The LoG edge detector was applied on the rest of the images in the testing dataset. Combining the two approaches, number of FNs were reduced to 70% of the ones reported only by the LoG edge detector. This leads to an average reduction of the NR values from 2.45% to 0.11%.

Using this technique also improved the overall performance of the of the edge detectors. As mentioned before, the edge detectors performed better on the sub-images with cracks due the effect of second level threshold which was the reason to evaluate their performance on C class and U class sub-images separately

in Table 3 and Table 4. However, PPV and F1 score metrics would be considerably lower if the both classes were considered in calculating them. For the best edge detector, i.e. LoG, PPV=6% and F1=11% were achieved when both classes were used. However, using the hybrid technique resulted in the almost the same PPV and F1 score provided in Table 3 for the LoG since only C class images were analyzed (with exception of two sub-images in the U class).

8. Conclusions

This paper presents a comparison of edge detection and DCNN algorithms for image based concrete crack detection. The dataset consisted of 3420 sub-images of concrete cracks. Several common edge detection algorithms were employed in the spatial (Roberts, Prewitt, Sobel, and LoG) and frequency (Butterworth and Gaussian) domains. AlexNet DCNN architecture was employed in its fully trained, classifier, and fine-tuned modes. Edge detection schemes performed reasonably well. The best method—LoG—accurately detected about 79% of cracked pixels and was useful in detecting cracks coarser than 0.1 mm. In comparison, the best DCNN method—the network in transfer learning mode—accurately detected 86% of cracked images and could detect cracks coarser than 0.04 mm. This represents a significant performance enhancement over edge detection schemes and shows promise for future applications of DCNN for image based crack detection in concrete. In addition, a methodology was proposed to reduce the FNs reports by 70% by applying the edge detectors only on sub-images not labeled as uncracked. In addition, a hybrid crack detector was introduced which combines the advantages of both approaches. In the hybrid detector, the sub-images were first labeled by the network in the fully trained mode. Since it produced the highest TNR, the edge detector is not applied on the sub-images labeled as U (uncracked) by the network. This technique reduced the noise ratio of the LoG edge detectors from 2.4% to 0.11% and has the similar effect on the other edge detectors as well.

This study shows the superiority of an AlexNet DCNN over traditional edge detectors for concrete crack detection. This superiority can be further improved when architectures such as GoogleNet or RestNet are implemented for crack detection. DLCCNs are able to classify multiple defects if enough annotated images

494 are available for training. Formation an annotated image dataset for structural defects, such as ImageNet,
495 is vital for further applications of DCNNs in structural engineering. With this dataset available, new
496 architectures can be proposed to focus on finding structural defects instead of random objects, which will
497 reduce the computational time associated with training process. In addition, domain adaptation methods
498 such as transfer learning, will be more effective if the network is previously trained on the structural defects
499 dataset. Improving the performance of domain adaptation techniques makes real-time defect detection in
500 robotic vision-based inspections feasible. In other words, a pre-trained DCNN on the structural defect
501 dataset, can be directly used to accurately classify new images taken by an unmanned aerial system to
502 different structural defects as the inspection is taking place.

503 **References**
504

[1] Federal Highway Administration, "National Bridge Inventory," FHWA, McLean, VA, 2017.

[2] Federal Highway Administration, "National Bridge Inspection Standards (FHWA–FAPG 23 CFR 650C)," FHWA, McLean, VA, 2017.

[3] B. Chan, H. Guan, J. Jo and M. Blumenstein, "Tuwards UAV-based bridge inspection systems: A reivew and an application perspective," *Structural Monitoring and Maintenance*, vol. 2, no. 3, p. 283–300, 2015.

[4] C. H. Yang, M. C. Wen, Y. C. Chen and S. C. Kang, "An optimized unmanned aeiral system for bridge inspection," in *Proceedings of the Insternational Symposium on Automation and Robotics in Construction*, Vilnius, Lithuania, 2015.

[5] S. Dorafshan, M. Maguire, N. Hoffer and C. Coopmans, "Fatigue Crack Detection Using Unmanned Aerial Systems in Under-Bridge Inspection," Idaho Transportation Department, Boise, ID, 2017.

[6] S. Dorafshan, M. Maguire, N. Hoffer and C. Coopmans, "Challenges in bridge inspection using small unmanned aerial systems: Results and lessons learned," in *Proceedings of the 2017 International Conference on Unmanned Aircraft Systems*, Miami, FL, 2017.

[7] N. Gucunski, S. H. Kee, H. M. La, B. Basily and A. Maher, "Delamination and concrete quality assessment of concrete bridge decks using a fully autonomous RABIT platform," *International Journal of Structural Monitoring and Maintenance,* vol. 2, no. 1, p. 19–34, 2015.

[8] R. S. Lim, H. M. Lag and W. Sheng, "A robotic crack inspectionand mapping system for bridge deck maintenance," *ICCC Transactions on Automation Science and Engineering,* vol. 11, no. 2, p. 367–378, 2014.

[9] N. Gucunski, S. H. Kee, H. La, B. Basily, A. Maher and H. Bhasemi, "Implementation of a fully autonomous platform for assessment of concrete bridge decks RABIT," in *Structures Congress 2015*, Portland, OR, 2015.

[10] S. Dorafshan and M. Maguire, "Bridge Inspection: Human Performance, Unmanned Aerial Vehicles and Automation," *Journal of Civil Strucutral Health monitoring,* pp. 1-34, 2018.

[11] N. Metni and T. Hamel, "A UAV for bridge inspection: Visual servoing control law with orientation limits," *Automation in Construction,* vol. 17, no. 1, p. 3–10, 2007.

[12] S. Dorafshan, M. Maguire and X. Qi, "Automatic Surface Crack Detection in Concrete Structures using OTSU Thresholding and Morphological Operations (UTC 01-2016)," Utah Transportation Center, Logan, UT, 2016.

[13] S. German, I. Brilakis and R. DesRoches, "Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments," *Advanced Engineering Informatics,* vol. 26, no. 4, p. 846–858, 2012.

[14] K. Vaghefi, T. T. M. Ahlborn, D. K. Harris and C. N. Brooks, "Combined imaging technologies for concrete bridge deck condition assessment," *Journal of Performance of Constructed Facilities,* vol. 29, no. 4, 2013.

[15] H. Sohn, D. Dutta, J. Y. Yang, M. DeSimio, S. Olson and E. Swenson, "Automated detection of delamination and disbond from wavefield images obtained using a scanning laser vibrometer," *Smart Materials and Structures,* vol. 20, no. 4, 2011.

[16] T. Omar and M. L. Nehdi, "Remote sensing of concrete bridge decks using unmanned aerial vehicle infrared thermography," *Automation in Construction,* vol. 83, p. 360–371, 2017.

[17] A. Ellenberg, A. Kontsos, F. Moon and I. Bartoli, "Bridge related damage quantification using unmanned aerial vehicle imagery," *Structural Control and Health Monitoring,* vol. 23, no. 9, p. 1168–1179, 2016.

[18] S. Dorafshan, R. Thomas and M. Maguire, "Fatigue Crack Detection Using Unmanned Aerial Systems in Fracture Critical," *Journal of Bridge Engineering,* 2018.

[19] M. R. Jahanshahi and S. F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Automation in Construction,* vol. 22, pp. 567-576, 2012.

[20] M. Hamrat, B. Boulekbache, M. Chemrouk and S. Amziane, "Flexural cracking behavior of normal strength, high strength and high strength fiber concrete beams, using Digital Image Correlation technique," *Construction and Building Materials,* vol. 106, pp. 678-692, 2016.

[21] A. Rimkus, A. Podviezko and V. Gribniak, "Processing digital images for crack localization in reinforced concrete members," *Procedia Engineering,* vol. 122, p. 239–243, 2015.

[22] L. Li, Q. Wang, G. Zhang, L. Shi, J. Dong and P. Jia, "A method of detecting the cracks of concrete undergo high-temperature," *Construction and Building Materials*, vol. 162, pp. 345-358, 2018.

[23] H. Kim, E. Ahn, S. Cho, M. Shin and S. H. Sim, "Comparative analysis of image binarization methods for crack identification in concrete structures," *Cement and Concrete Research*, vol. 99, p. 53–61, 2017.

[24] T. Yamaguchi, S. Nakamuran, R. Saegusa and S. Hashimoto, "Image-based crack detection for real concrete surfaces," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 3, no. 1, p. 128–135, 2008.

[25] M. R. Taha, A. Noureldin, J. L. Lucero and T. J. Baca, "Wavelet transform for structural health monitoring: a compendium of uses and features," *Structural Health Monitoring*, vol. 5, no. 3, pp. 267-295, 2006.

[26] J. Kittler, R. Marik, M. Mirmehdi, M. Petrou and J. Song, "Detection of defects in colour texture surfaces," in *IAPR Workshop on Machine Vision Applications*, Kawasaki, 1994.

[27] I. Abdel-Qader, P. Abudayyeh and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *Journal of Computing in Civil Engineering*, vol. 17, no. 4, p. 255–263, 2003.

[28] I. Abdel-Qader, S. Pashaie-Rad, P. Abudayyeh and S. Yehia, "PCA-based algorithm for unsupervised bridge crack detection," *Advances in Engineering Software*, vol. 37, no. 12, p. 771–778, 2006.

[29] J. K. Oh, G. Jang, S. Oh, J. H. Lee, B. J. Yi, Y. S. Moon, J. S. Lee and Y. Choi, "Bridge inspection robot system with machine vision," *Automation in Construction*, vol. 18, no. 7, p. 929–941, 2009.

[30] H. Moon and J. Kim, "Intelligent crack detecting algorithm on the concrete crack image using neural network," in *Proceedings of the 28th International Symposium on Automation and Robotics in Construction*, Seoul, 2011.

[31] H. Wang, Z. Chen and L. Sun, "Image preprocessing methods to identify micro-cracks of road pavement," *Optics and Photonics Journal*, vol. 3, no. 2, p. 99, 2013.

[32] P. Zheng, "Crack Detection and Measurement Utilizing Image-Based Reconstruction," MS Thesis. Virginia Polytechnic Institute, Blacksburg, VA, 2014.

[33] R. S. Lim, H. M. La and W. Sheng, "A robotic crack inspection and mapping system for bridge deck maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, p. 367–378, 2014.

[34] J. W. Kim, S. B. Kim, J. C. Park and J. W. Nam, "Development of crack detection system with unmanned aerial vehicles and digital image processing," in *Advances in Structural Engineering and Mechanics (ASEM15)*, Incheon, 2015.

[35] S. Sankararinivasan, E. Balasubramanian, K. Karthik, U. Chandrasekar and R. Gupta, "Health monitoring of civil structures with integrated UAV and image processing system," *Procedia Computer Science*, vol. 54, p. 508–515, 2015.

[36] A. M. A. Talab, Z. Huang, F. Xi and L. HaiMing, "Detection crack in image using Otsu method and multiple filtering in image processing techniques," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 3, p. 1030–1033, 2016.

[37] S. Dorafshan, M. Maguire and M. Chang, "Comparing automated image-based crack detection techniques in spatial and frequency domains," in *Proceedings of the 26th American Society of Nondestructive Testing Reseach Symposium*, Jacksonville, FL, 2017.

[38] S. Dorafshan and M. Maguire, "Autonomous detection of concrete cracks on bridge decks and fatigue cracks on steel members," in *Digital Imaging 2017*, Mashantucket, CT, 2017.

[39] Y. Noh, D. Koo, Y. M. Kang, D. Park and D. Lee, "Automatic crack detection on concrete images using segmentation via fuzy C-means clustering," in *Proceedings of the 2017 International Conference on Applied System Innovation*, Sapporo, 2017.

[40] G. K. Choudhary and S. Dey, "Crack detection in concrete surfaces using image processing, fuzzy logic, and neural networks," in *In 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, , Nanjing, China, 2012.

[41] Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani and O. Büyüköztürk, "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types," *Computer-Aided Civil and Infrastructure Engineering.*, 2017.

[42] Y. J. Cha, W. Choi and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Computer-Aided Civil and Infrastructure Engineering,* vol. 32, no. 5, pp. 361-378, 2017.

[43] S. Dorafshan, C. Coopmans, R. J. Thomas and M. Maguire, "Deep Learning Neural Networks for sUAS-Assisted Structural Inspections: Feasibility and Application," in *ICUAS 2018*, Dallas, TX, 2018.

[44] A. Mohan and S. Poobal, "Crack detection using image processing: A critical review and analysis," *Alexandria Engineering Journal,* vol. In press., 2017.

[45] M. Maguire, S. Dorafshan and R. Thomas, "SDNET2018: A concrete crack image dataset for machine learning applications," Utah State Universiy, Logan, 2018.

[46] M. O'Byrne, F. Schoefs, B. Ghosh and V. Pakrashi, "Texture analysis based damage detection of ageing infrastructural elements," *Computer-Aided Civil and Infrastructure Engineering,* vol. 28, no. 3, pp. 162-177, 2013.

[47] L. Wu, S. Mokhtari, A. Nazef, B. Nam and H. B. Yun, "Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment," *Journal of Computing in Civil Engineering,* vol. 30, no. 1, 2014.

[48] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85-117, 2015.

[49] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems* , pp. 1097-1105, 2012.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov and A. ... & Rabinovich, "Going deeper with convolutions," *CVPR,* 2015.

[51] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition,* 770-778.

[52] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Miami, FL, 2009.

[53] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition,* Seattle. WA, 2016.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov and A. ... Rabinovich, "Going deeper with convolutions," in *CVPR,* 2015.

[55] L. Zhang, F. Yang, Y. D. Zhang and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Image Processing (ICIP), 2016 IEEE International Conference on*, 2016.

[56] A. Zhang, K. C. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li and C. Chen, "Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network," *Computer-Aided Civil and Infrastructure Engineering,* vol. 32, no. 10, pp. 805-819, 2017.

[57] F. C. Chen and M. R. Jahanshahi, "NB-CNN: Deep Learning-based Crack Detection Using Convolutional Neural Network and Naïve Bayes Data Fusion," *IEEE Transactions on Industrial Electronics*, 2017.

[58] D. J. Atha and M. R. Jahanshahi, " Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection," *Structural Health Monitoring,* p. 1475921717737051, 2017.

[59] S. S. Kumar, D. M. Abraham, M. R. Jahanshahi, T. Iseley and J. Starr, "Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks," *Automation in Construction,* vol. 91, pp. 273-283, 2018.

[60] D. Feng and M. Q. Feng, "Identification of structural stiffness and excitation forces in time domain using noncontact vision-based displacement measurement," *Journal of Sound and Vibration,* vol. 406, pp. 15-28, 2017.

[61] D. Feng and M. Q. Feng, "Computer vision for SHM of civil infrastructure: From dynamic response measurement to damage detection–A review," *Engineering Structures,* vol. 156, pp. 105-117, 2018.

[62] Y. Bao, Z. Tang, H. Li and Y. Zhang, "Computer vision and deep learning–based data anomaly detection method for structural health monitoring," *Structural Health Monitoring,* p. 1475921718757405, 2018.

[63] S. Dorafshan, R. Thomas, M. Maguire and C. Coopmans, "A Practitioner's Guide to Small Unmanned Aerial Systems for Bridge Inspection," in *ICUAS18*, Dallas, TX, 2018.

[64] D. Kang and Y. J. Cha, "Autonomous UAVs for Structural Health Monitoring Using Deep Learning and an Ultrasonic Beacon System with Geo-Tagging," *Computer-Aided Civil and Infrastructure Engineering*, 2018.

[65] D. Kang and Y. J. Cha, "Damage detection with an autonomous UAV using deep learning," in *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018 (Vol. 10598, p. 1059804). International Society for Optics and Photonics*, Denver, CO, 2018.

[66] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*..

[67] Fei-Fei L., J. J and Y. S., Stanford University, 2017. [Online]. Available: http://cs231n.stanford.edu/. [Accessed 21 03 2018].

[68] H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues and R. M. ... Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285-1298, 2016.

[69] Z. Li and D. Hoiem, " Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[70] S. Dorafshan, R. Thomas and M. Maguire, "Image Processing Algorithms for Vision-based Crack Detection in Concrete Structures," *Advanced Concrete Technology*, 2018.

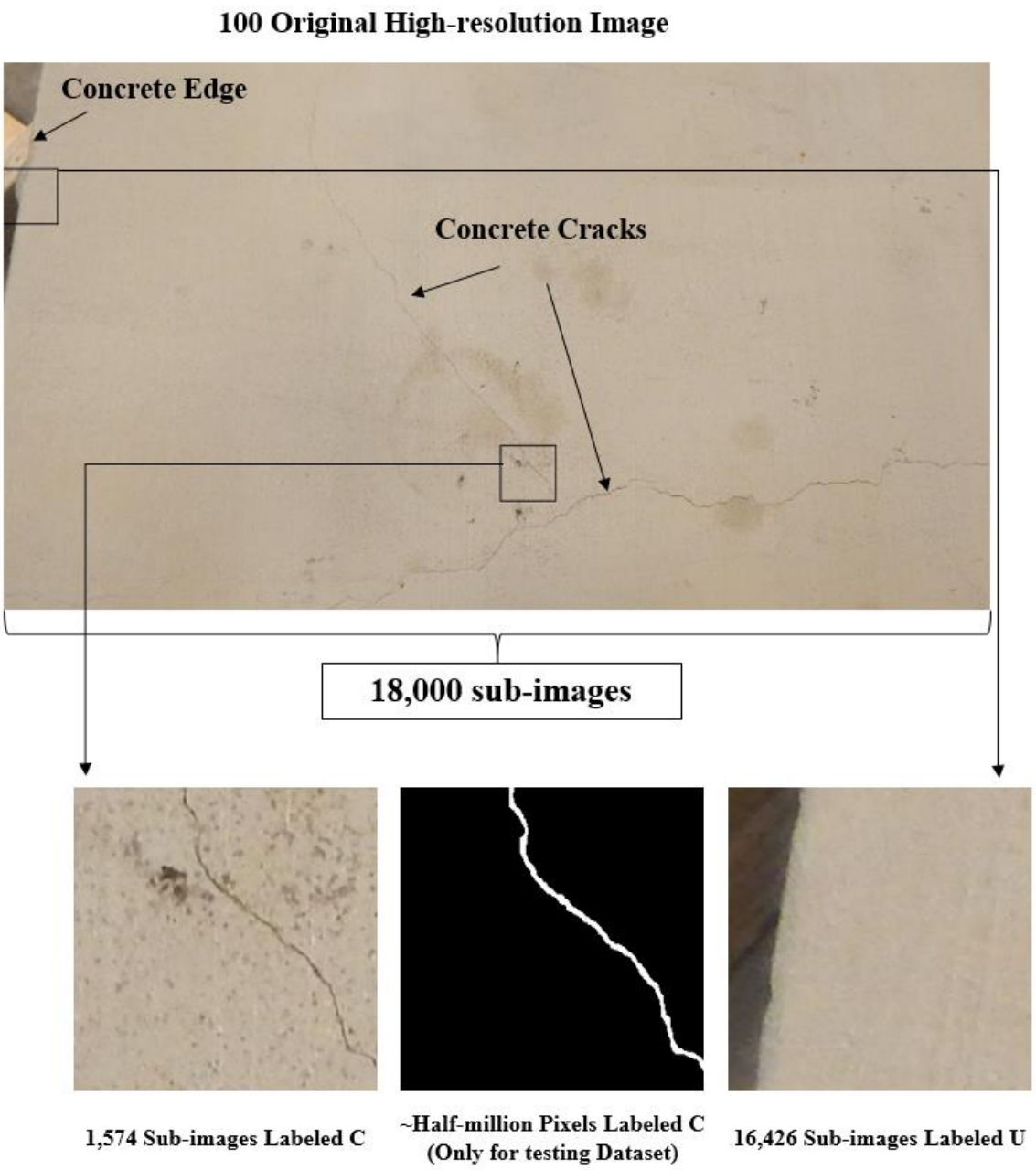
[71] S. Dorafshan, C. Coopmans, R. Thomas and M. Maguire, "Deep Learning Neural Networks for sUAS-Assisted Structural Inspections: Feasibility and Application," in *ICUAS2018*, Dallas, 2018.

505

506
507

List of Figures

List of Tables

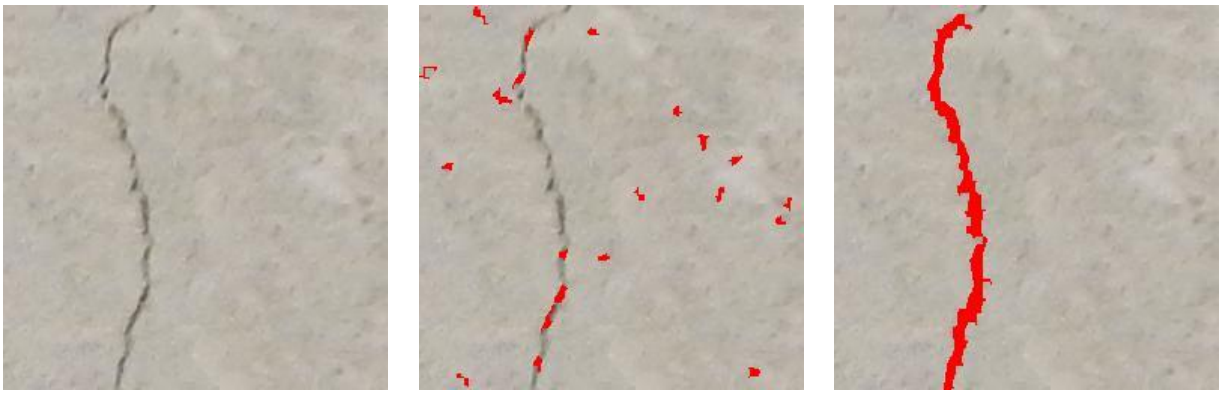


549

550 **Figure 1** Illustration of the dataset

551

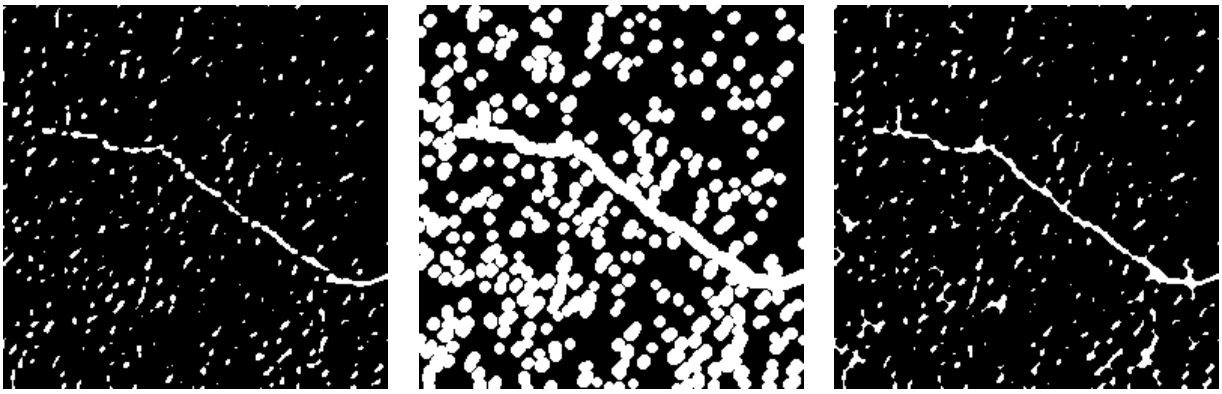
552



(a) (b) (c)

553 **Figure 2** The effect of edge enhancement on the final image of the edge detectors, Sobel, (a) original
554 image, (b) final binary image superimposed on the original image (b) without the edge enhancement, (c)
555 with the edge enhancement

556



(a) (b) (c)

Figure 3 Closing operation illustration (a) first level binary image, (b) dilation, and (c) erosion using a disk structuring element with diameter of 4 px. (LoG edge detector).

(a) (b) (c)

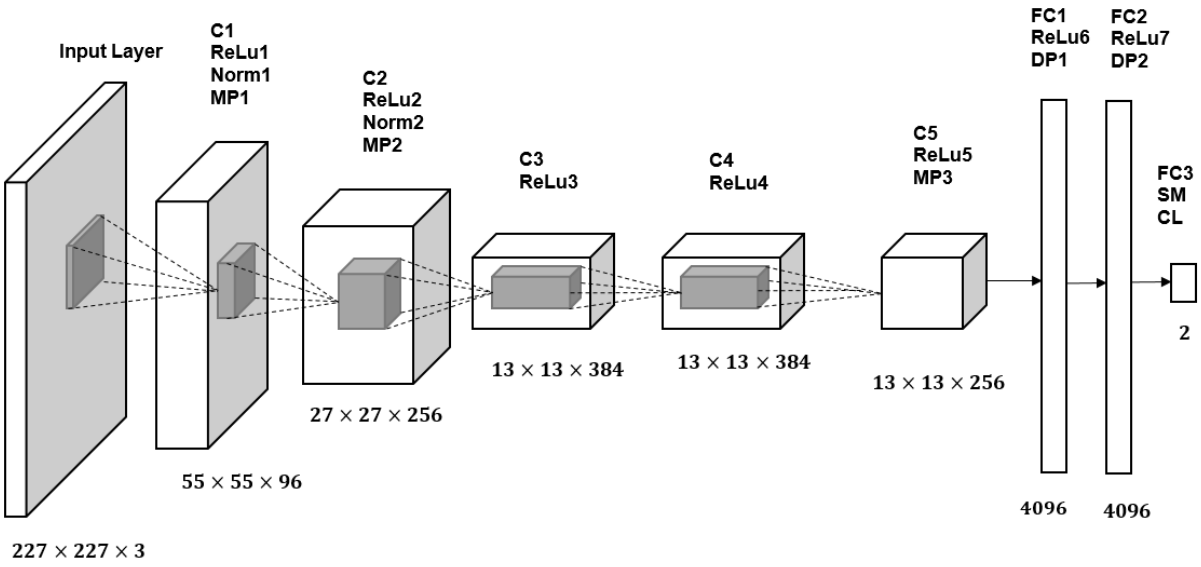
560 **Figure 4** Crack in the (a) ground truth, 1391 px, (b) without the closing operation 391 px correct

561 detection (c) with closing operation 1215 px correct detection (LoG edge detector)

562

(a) **(b)** **(c)**

Figure 5 Crack in the (a) ground truth, 2325 px, (b) without second level threshold operation 3672 pixels false detection (c) with second level threshold operation: 214 px false detection (Gaussian edge detector)



566

567 **Figure 6.** AlexNet DCNN architecture

568

569

570

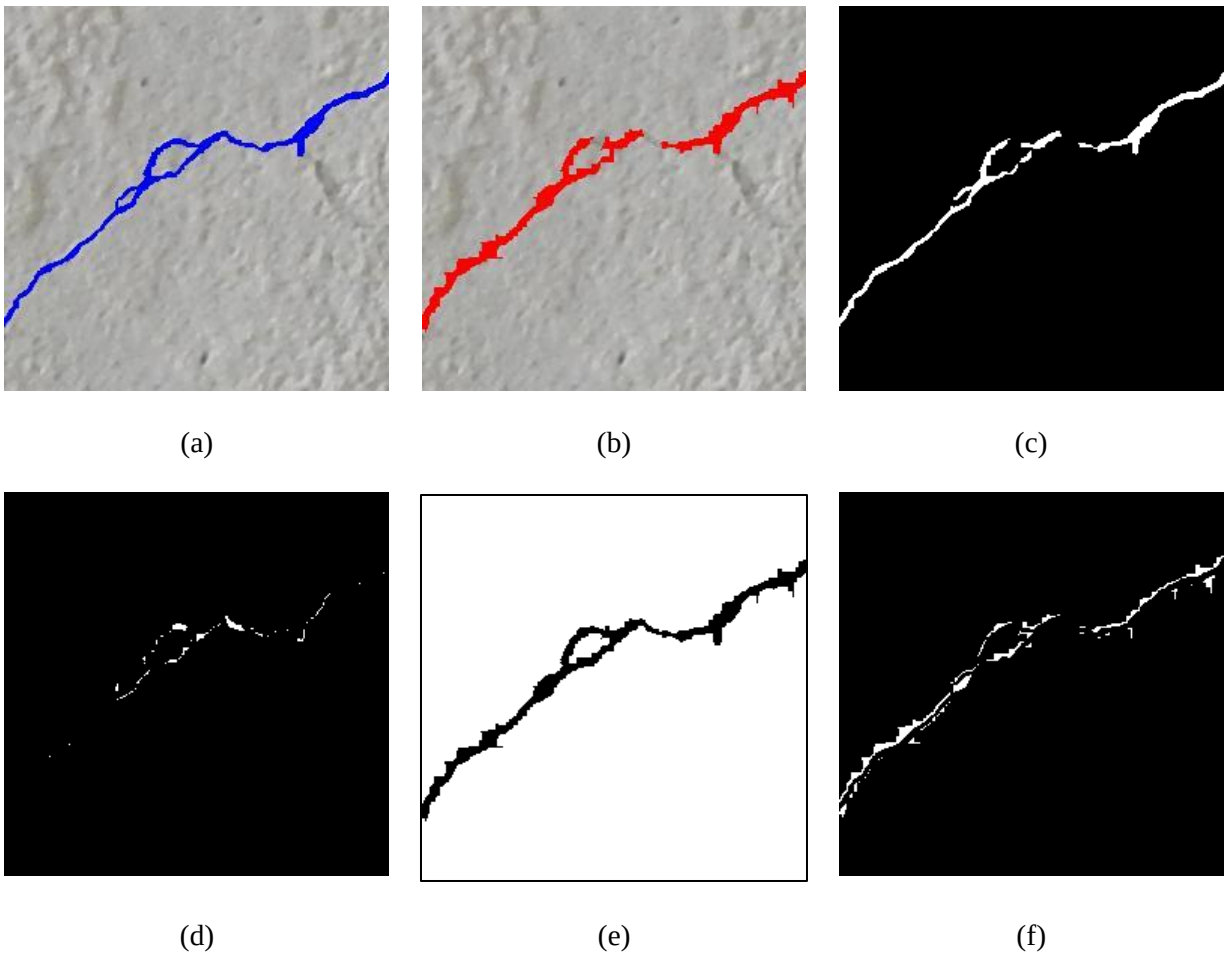
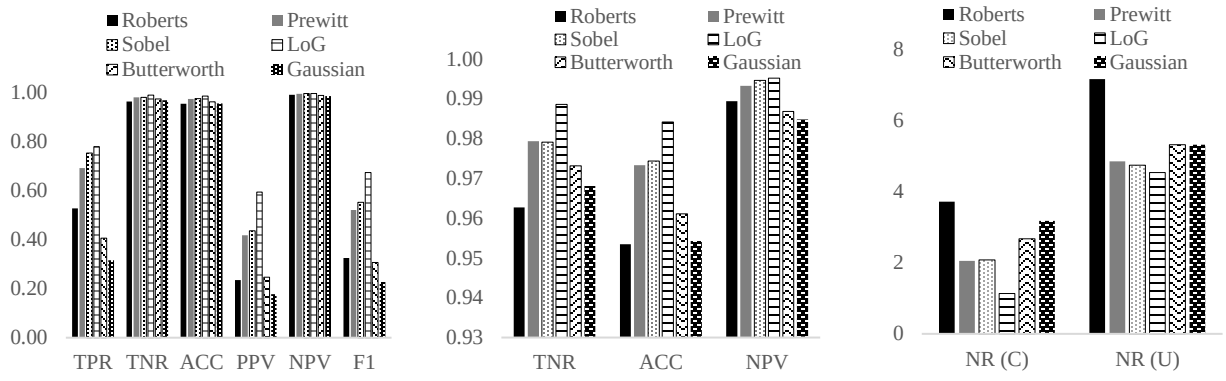


Figure 7. Examples of metric, (a) ground truth, Cp=1,582 px, Up=63,954 px, (b) final binary image using Roberts edge detector, Cp=2276 px, Up=63,260 px (c) TP=1367 px, (d) FN=215 px, (e) TN=63,045 px, (f) FP=909 px (Robersts edge detector)

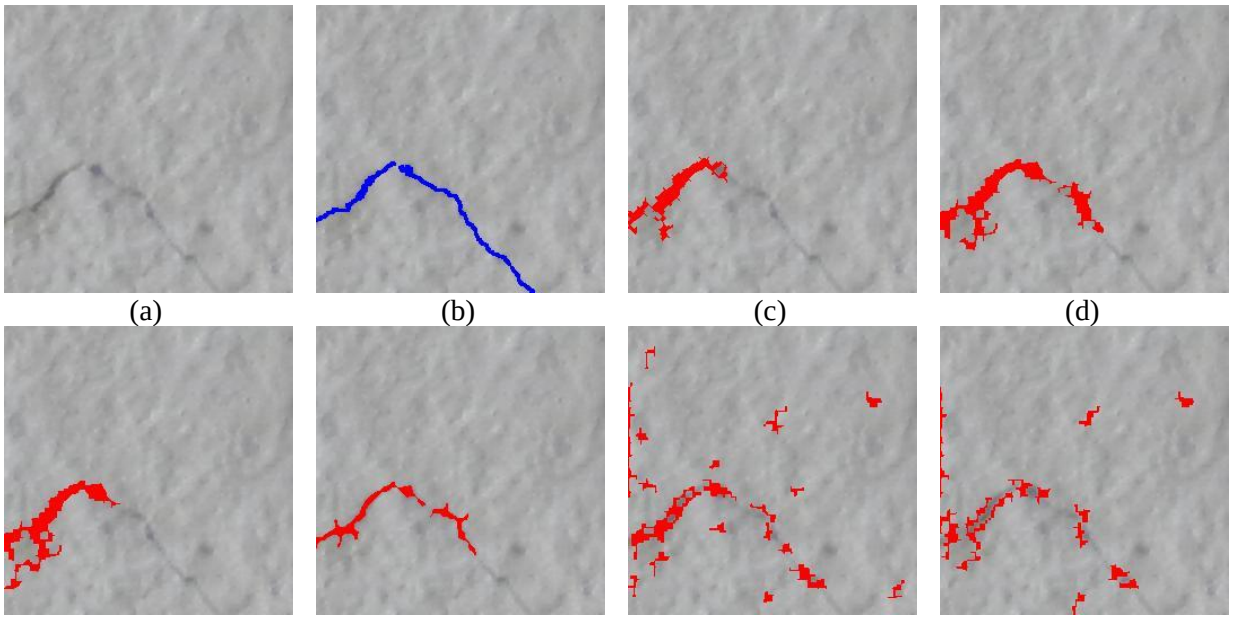


(a) (b) (c)

576 **Figure 8** Results of the studied edge detectors on the sub-images in the C class (a) TRP, PPV, and F1 (b)

577 TNR, ACC, and NPV, (c) NR in C and U classes.

578

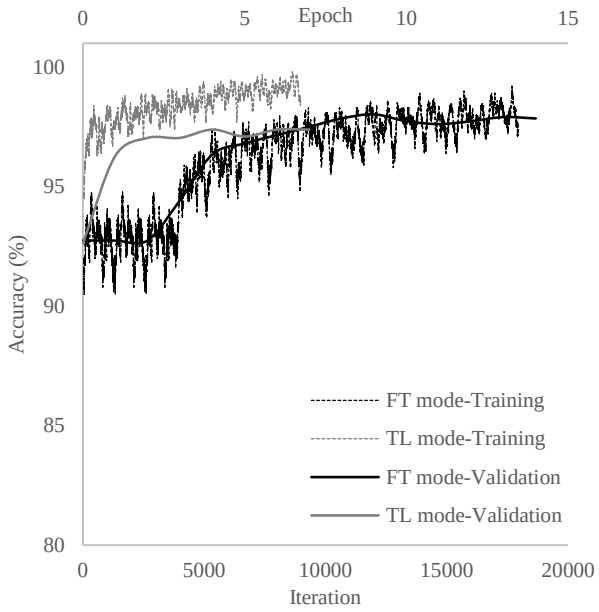


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 9 An example of edge detector performance on a 0.02 mm crack (a) original image, (b) GT=1145 px, (c) Roberts, TPR=39% (d) Prewitt, TPR=60%, (e) Sobel, TPR=55%, (f) LoG, TPR=71%, (g) Butterworth, TPR=38%, (h) Gaussian, TPR=17%

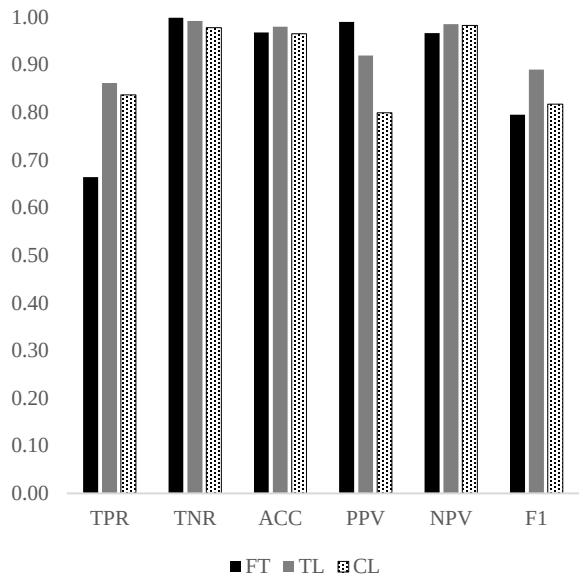
584



585

586 **Figure 10.** DCNN accuracy during training and validation

587



588

589 **Figure 11** Metrics for the DCNN in FT, TL, and CL modes

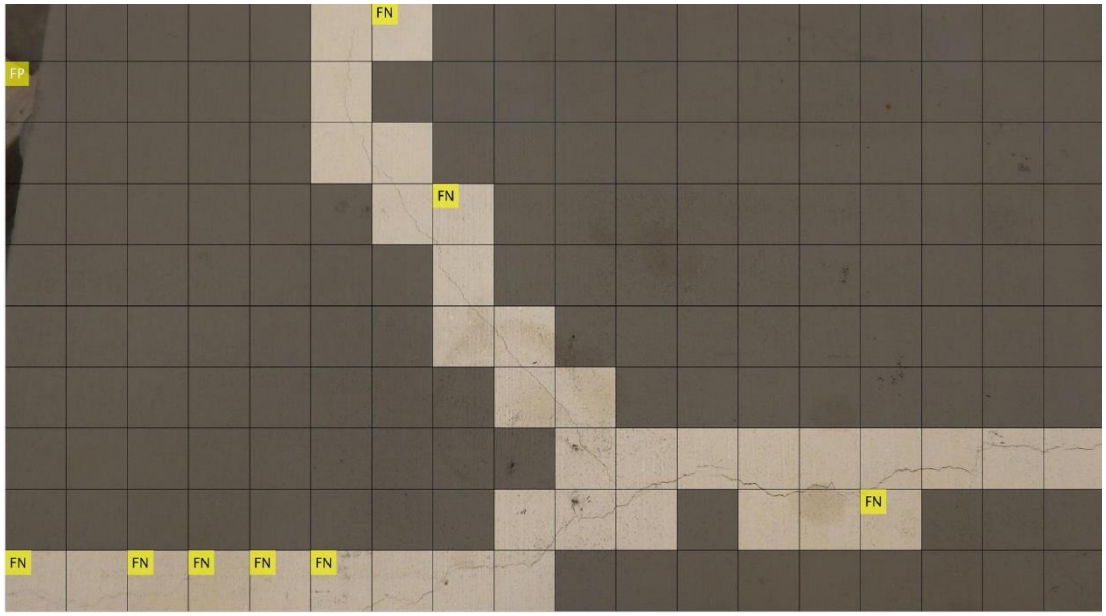
590

(a) (b) (c)

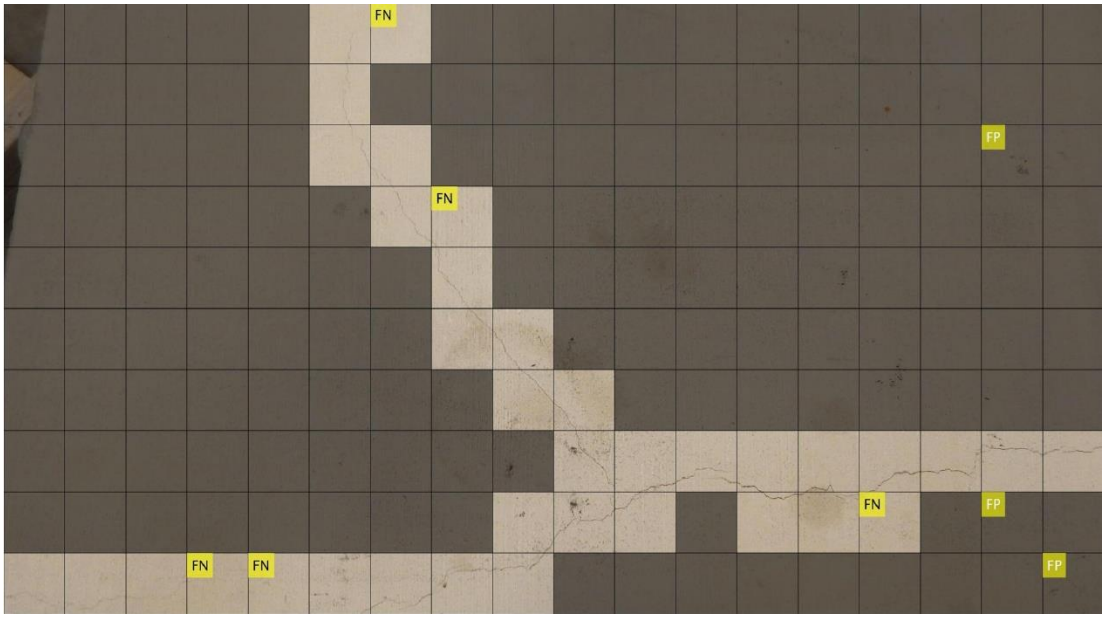
591 **Figure 12.** DCNN results for a crack of width 0.08 mm: (a) FT mode, (b) TL mode, and (c) CL mode

592

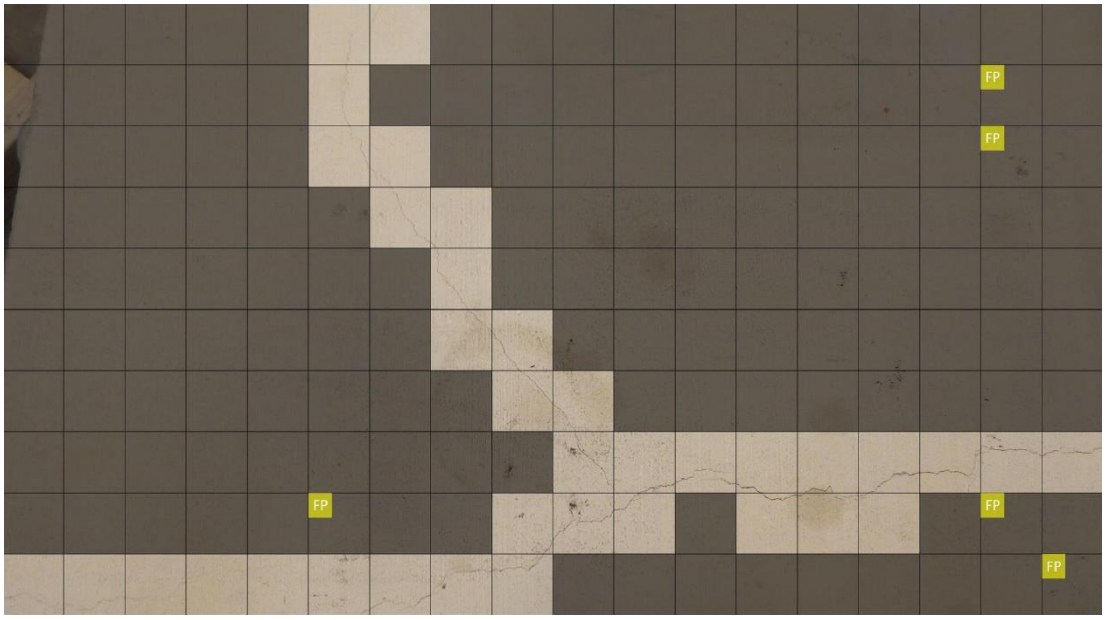
593



(a)



(b)

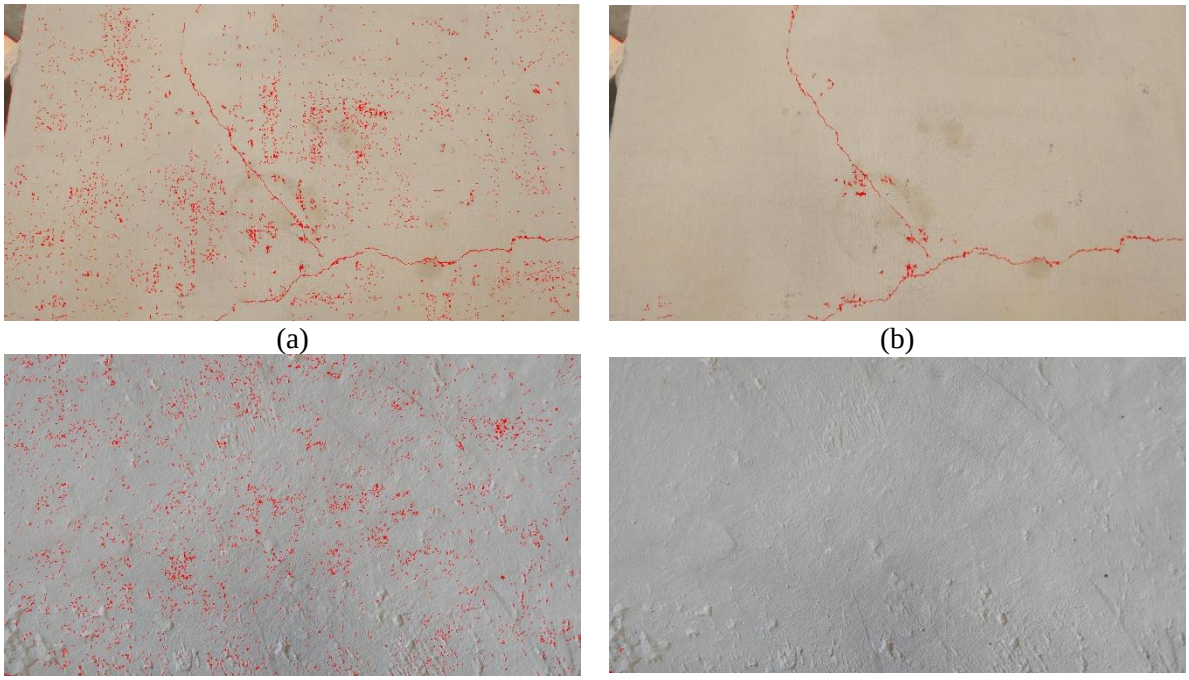


(c)

Figure 13. Results of (a) fully trained DCNN crack detection, (b) transfer learning DCNN, and (c) classifier DCNN for crack detection on the original full scale images in the testing dataset

(a) (b) (c)

Figure 14 Examples of FNs in the U class images (a) non-crack edge, (b) different surface finish, (c) noise due to the coarse concrete surface



(a) (b)

(c) (d)

Figure 15 Combination of DCNN and edge detectors (a) the superimposed image with crack using LoG on all sub-images, (b) the superimposed image with crack without using LoG on U class sub-images, (c) the superimposed image without crack using LoG on all sub-images, (d) the superimposed image without crack without using LoG on U class sub-images.

606 **Table 1.** Number of cracked and sound sub-images in training, validation, and testing datasets

Dataset	No of Original Images	C	U	Total
Training	81	1129	11680	12809
Validation		125	1646	1771
Testing	19	319	3101	3420

607

608 **Table 2.** Number of Cp and Up pixels in the testing dataset

Dataset	Cp	Up	Cp Ratio (%)
im1	18835	11777645	0.16
im2	13952	11782528	0.12
im3	67548	11728932	0.57
im4	13472	11783008	0.11
im5	46192	11750288	0.39
im6	46372	11750108	0.39
im7	46658	11749822	0.40
im8	37572	11758908	0.32
im9	42675	11753805	0.36
im10	88321	11708159	0.75
im11	2693	11793787	0.02
im12	1264	11795216	0.01
im13	3336	11793144	0.03
im14	0	11796480	0.00
im15	5995	11790485	0.05
im16	4203	11792277	0.04
im17	0	11796480	0.00
im18	4953	11791527	0.04
im19	1304	11795176	0.01

609

610

611

612 **Table 3.** Summary of edge detector performance on sub-images in the C class

Domain	Edge Detector	TPR	TNR	ACC	PPV	NPV	F1	MCW (mm)	T (s)
Spatial	Roberts	0.53	0.96	0.95	0.23	0.99	0.32	0.40	5.15
	Prewitt	0.69	0.98	0.97	0.42	0.99	0.52	0.20	4.13
	Sobel	0.76	0.98	0.97	0.44	0.99	0.56	0.20	4.64
	LoG	0.79	0.99	0.98	0.60	1.00	0.68	0.10	3.79
Frequency	Butterworth	0.41	0.97	0.96	0.25	0.99	0.31	0.20	5.76
	Gaussian	0.32	0.97	0.95	0.18	0.98	0.23	0.20	5.70

613

614

615 **Table 4** Summary of edge detector performance on sub-images in the U class

Domain	Edge Detector	TNR	T (s)
Spatial	Roberts	0.93	5.46
	Prewitt	0.95	4.71
	Sobel	0.95	4.83
	LoG	0.95	4.05
Frequency	Butterworth	0.95	5.98
	Gaussian	0.93	5.86

616
617

618

619 **Table 5.** Summary of DCNN results

Mode	TP	FN	TN	FP	TPR	TNR	ACC	PPV	NPV	F1	MCW (mm)	Time (s)
FT	212	107	3099	2	0.66	1.00	0.97	0.99	0.97	0.80	0.08	2.65
TL	275	44	3077	24	0.86	0.99	0.98	0.92	0.99	0.89	0.04	2.81
CL	267	52	3034	67	0.84	0.98	0.97	0.80	0.98	0.82	0.08	2.75

620

621

622 **Table 6.** Comparison of DCNN and edge detection performance considering sub-images

Method		TPR	TNR	ACC	PPV	NPV	F1	MCW (mm)	Time (s)
DCNN	FT mode	0.66	1.00	0.97	0.99	0.97	0.80	0.08	2.65
	TL mode	0.86	0.99	0.98	0.92	0.99	0.89	0.04	2.81
	CL mode	0.84	0.98	0.97	0.80	0.98	0.82	0.08	2.75
Edge Detector	Roberts	0.53	0.96	0.95	0.23	0.99	0.32	0.40	5.30
	Prewitt	0.69	0.98	0.97	0.42	0.99	0.52	0.20	4.42
	Sobel	0.76	0.98	0.97	0.44	0.99	0.56	0.20	4.74
	LoG	0.79	0.99	0.98	0.60	1.00	0.68	0.10	3.92
	Gaussian	0.41	0.97	0.96	0.25	0.99	0.31	0.20	5.87
	Butterworth	0.32	0.97	0.95	0.18	0.98	0.23	0.20	5.78

623

624

625

626