Computer-Aided Civil and Infrastructure Engineering **33** (2018) 731–747

Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types

Young-Jin Cha*, Wooram Choi, Gahyun Suh & Sadegh Mahmoudkhani

Department of Civil Engineering, University of Manitoba, Winnipeg, MB, Canada

&

Oral Büyüköztürk

Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract: *Computer vision-based techniques were developed to overcome the limitations of visual inspection by trained human resources and to detect structural damage in images remotely, but most methods detect only specific types of damage, such as concrete or steel cracks. To provide quasi real-time simultaneous detection of multiple types of damages, a Faster Region-based Convolutional Neural Network (Faster R-CNN)-based structural visual inspection method is proposed. To realize this, a database including 2,366 images (with 500 × 375 pixels) labeled for five types of damages—concrete crack, steel corrosion with two levels (medium and high), bolt corrosion, and steel delamination—is developed. Then, the architecture of the Faster R-CNN is modified, trained, validated, and tested using this database. Results show 90.6%, 83.4%, 82.1%, 98.1%, and 84.7% average precision (AP) ratings for the five damage types, respectively, with a mean AP of 87.8%. The robustness of the trained Faster R-CNN is evaluated and demonstrated using 11 new 6,000 × 4,000-pixel images taken of different structures. Its performance is also compared to that of the traditional CNN-based method. Considering that the proposed method provides a remarkably fast test speed (0.03 seconds per image with 500 × 375 resolution), a framework for quasi real-time damage detection on video using the trained networks is developed.*

*To whom correspondence should be addressed. E-mail: *Young.Cha@umanitoba.ca*.

© 2017 *Computer-Aided Civil and Infrastructure Engineering*.
DOI: 10.1111/mice.12334

1 INTRODUCTION

Visual changes in civil infrastructures, such as cracks or corrosion, alert us to structural health conditions. Thus, visual inspection is the primary means of evaluating bridges' conditions (Ryan et al., 2012). Despite the critical roles of bridges in public safety and the economy, human-based visual inspection (which is significantly limited by the training of the inspector) is common, and consistency in quantitative evaluation and accessibility raises concerns about the safety of bridges (Graybeal et al., 2002). Because automation of visual inspections can address the limitations of the ordinary human-oriented visual approach, researchers have been attracted to the development of computer vision-based methods for structural damage detection.

Most studies have focused on image-processing techniques (IPTs) and testing their compatibility with real-world situations using pre- and postprocessing methods to detect one specific type of structural damage, such as concrete cracks (Abdel-Qader et al., 2003; Nishikawa et al., 2012), concrete spalling (German et al., 2012), steel cracks (Yeum and Dyke, 2015), loosened bolts (Park et al., 2015; Cha et al., 2016), road pavement cracks (Ying and Salari, 2010; Cord and Chambon, 2012; Zalama et al., 2014), underground concrete pipe cracks (Sinha et al., 2003), and potholes in asphalt pavement (Koch and Brilakis, 2011). Machine learnings have been applied for structural health monitoring problems (Adeli and Jiang, 2009; Siddique and Adeli, 2016; Rafiei et al., 2017). Some researchers tried to improve the efficiency and robustness of IPT-based methods in real-world conditions by applying machine learning techniques (Chen et al., 2012; O'Byrne et al., 2013; Liao and Lee, 2016; Wu et al., 2016; Mirzaei et al., 2016). Despite their improvements, these methods still require the pre- and postprocessing techniques, which are time consuming and can detect only one damage type.

To address the drawbacks of IPT-based damage detection techniques, a deep learning-based method using Convolutional Neural Networks (CNNs) for assessing concrete cracks was introduced by Cha et al. (2017). The main advantage is its automatic calculation of damage-sensitive features during training process of the CNN. They trained a CNN using cracked or intact concrete images with a resolution of 256 × 256 pixels and achieved approximately 98% accuracy. Then, they used a sliding window to provide evaluation images of sizes larger than 256 × 256 pixels, and showed that their method performed quite better than traditional IPT methods. This CNN-based method has the capacity to provide multiple classifications by adding new damage types, such as steel delamination, to its training set. Although they used a sliding window to localize the damage, it is difficult to find the optimal size of the sliding window because the testing images may have various sizes and scales.

To provide multiple objects detection and localization, Girshick et al. (2014) developed Region-based CNN (R-CNN), which takes images and object proposals from selective searches (Uijlings et al., 2013) as inputs and uses CNNs to extract features, and a simple novel regressor and support vector machines (SVMs) to localize and classify objects. The R-CNN significantly increased the accuracy of object detection, in comparison with CNN-based methods that use a sliding window, on the PASCAL visual object classes (VOCs) database (Everingham et al., 2007; Sermanet et al., 2014; Girshick et al., 2014), which is a benchmark in visual object category recognition and detection with a standard data set of images and annotation, and standard evaluation procedures. However, it was computationally slow, expensive, and difficult to use because R-CNN is not trained end-to-end, and three different training processes for CNNs, regressor, and SVM are required. Moreover, during the R-CNN's forward propagation, features extracted by CNNs are fed into the SVM and regressor, but computations are not shared, which results in a slow and costly process. To address the drawbacks of R-CNN, He et al. (2014) introduced a new CNN architecture called spatial pyramid pooling network (SPP-net). Despite an increase in the speed of object detection in comparison to the R-CNN, the SPP-net has a difficult training process similar to that of the R-CNN, and does not engage its pretrained CNNs in the training process, which limits its accuracy.

In 2015, Fast R-CNN was developed by Girshick to address the drawbacks of R-CNN and SPP-net. Fast R-CNN is trained end-to-end in one stage and shows higher speed and accuracy than both the R-CNN and SPP-net. Despite the better performance of Fast R-CNN, it is dependent on taking precomputed object proposals, as is the case with the R-CNN and SPP-net methods. Moreover, Fast R-CNN is slow and has limited accuracy because generating object proposals through an external method like selective search is time consuming, and is not optimized during the training process. To unify the object proposals creator and Fast R-CNN, Ren et al. (2016) proposed a region proposal network (RPN) to generate object proposals and introduced the Faster R-CNN method by combining RPN and Fast R-CNN to detect 20 VOCs. Faster R-CNN reduces computational costs by sharing features between RPN and Fast R-CNN, improves the accuracy by training the network end-to-end for both the object detector and proposal generator, and provides real-time object detection.

In this article, an architecture and data set are modified to classify five objects and train and validate the Faster R-CNN to provide quasi real-time, simultaneous, and automated vision-based structural surface damage detection of multiple damage types in videos. The five types of structural surface damage we choose as examples in our study are steel delamination, steel corrosion (medium and high levels), bolt corrosion, and concrete cracks. Based on this study, other damage types can be easily added using the Faster R-CNN-based method.

This article is organized as follows. In Section 2, an overview of the proposed method describes using RPN to generate object proposals, architecture for CNNs (Zeiler and Fergus, 2014), Fast R-CNN, and combining Fast R-CNN and RPN to create Faster R-CNN. In Section 3, the procedure for generating databases and training implementation are explained. In Section 4, the results of the training and validation processes, as well as testing, are discussed and compared with the CNNs-based method of Cha et al. (2017). In Section 5, a framework to make the Faster R-CNN compatible with quasi real-time damage detection is proposed. The article ends with a discussion of the findings in Section 6.

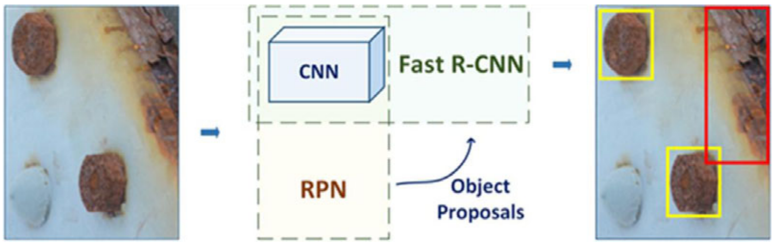

Fig. 1. The architecture of Faster R-CNN.

2 METHODOLOGY

To detect and localize multiple types of damage, the Faster R-CNN method is used for quasi real-time processing of images and videos. The overall schematic architecture of the Faster R-CNN is presented in Figure 1. The original Faster R-CNN was composed of RPN to provide object proposals in images and Fast R-CNN to improve the localization of the object proposals provided by the RPN and to classify objects in images. As shown in Figure 1, both RPN and Fast R-CNN networks use the same CNN to extract features from images. This Faster R-CNN is modified to achieve structural damage detection and localization. The details of the Faster R-CNN and its modification are explained in this section.

2.1 Convolutional neural networks

In this section, the fundamental theories and concepts of the CNN are presented. More details on convolutional (CONV) layers, including numerical examples, can be found in the research studies done by Ciresan et al. (2011) and Cha et al. (2017).

2.1.1 Convolutional layer. A CONV layer has the main computational role in CNNs and includes a set of filters (kernels) with learnable weights. Filters have the same depth as the input of their layer, and a smaller width and height than the input. For example, the filters of the first CONV layer of a CNN that takes RGB images (which have depth 3) as input have depth 3. Each filter slide on the input and dot product (convolution) is calculated between the filter and its respective field on the input. Sliding step is defined by a stride that has an effect on output size and computational costs. Higher strides may decrease output size and computational cost, but might lead to lost features. The computed dot product values related to the channels of each filter are added together with a bias to produce the responses of each filter. These responses are stacked to produce the spatial output of the CONV layer. CONV layer outputs feature its size, depending on stride, input size, and filter size, and may be less than input size. As reducing output size may lead to lost features, symmetrical zeros can be added into the input to maintain output size.

2.1.2 Max pooling layer. A max pooling layer performs downsampling by decreasing the spatial size of its input. Downsampling reduces computational costs and the probability of overfitting. A max pooling layer slides a window (filter) on the input and outputs the maximum value from its respective field.

2.1.3 Fully connected layer. Unlike a CONV layer, which connects to a local spatial region of its input, a fully connected (FC) layer connects to all neurons in its previous layer. This layer, such as a layer in a regular multilayer artificial neural networks, is a vector that performs dot product and adds a bias on its inputs in each neuron.

2.1.4 Softmax layer. A softmax layer is a well-known multiclass classifier in CNNs that can predict the class of its input. Softmax normally takes features from a FC layer, calculates the probabilities of each individual class, and then outputs the class with the highest probability as the classification results.

2.2 Region proposal networks

The role of a RPN is to propose object proposals, and the schematic architecture of a RPN is presented in Figure 2. The RPN takes images as inputs and outputs a set of rectangular object proposals, including the probability of being an object in each proposal. RPN uses a CNN to extract a feature map (outputs of the last layer of CNN) and slide another CONV layer on the map. The CONV layer is followed by a rectified linear unit (ReLU) activation function (Nair and Hinton, 2010), which provides nonlinearity and increases the speed of convergence (Krizhevsky et al., 2012). This CONV, followed by ReLU, maps the features of each sliding window to a vector, which is fed into regression and softmax layers. Regression and softmax layers predict the coordinates of the multiple bounding boxes and the

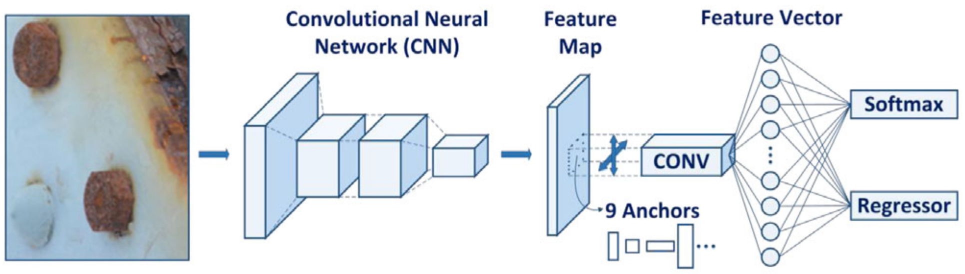

Fig. 2. The schematic architecture of the RPN.

probability of being an object in each box, respectively. The details of the RPN are provided below.

The core element of the RPN is the feature map after CNN presented in Figure 2. To generate object proposals, each corresponding spatial window of the sliding CONV is associated with nine rectangular boxes called anchors, as shown in Figure 2. Based on the work of Ren et al. (2016), the recommended number of anchors is nine. The nine anchors are composed of three different widths and heights, and are expressed as eight constants (i.e., center of the sliding CONV window: (x_a, y_a), width and height: (w_a^k, h_a^k)) for nine anchors, where k is three. Thus, the combinations of the three widths and heights provide nine anchors. To calculate the overlap between an anchor and a ground truth, the Intersection-over-Union (IoU) concept is used. An anchor is labeled as positive if its IoU ratio with a ground truth is the highest, if there are multiple ground truths, or if it is greater than 0.7 (Ren et al., 2016). Boxes with every IoU ratio lower than 0.3 are labeled as negative (background). The other anchors are not used for the training process.

This sliding CONV, followed by ReLU, is fed into a FC layer (feature vector; Figure 2). Using the vector and initial weights, the softmax layer calculates two outputs for each of the nine generated boxes at each sliding CONV window, which are the probability of being an object in the box or just being part of the background (having no object). The probability of objectness, which is calculated by softmax layer for each bounding box, is between zero and one, and is updated during training process to minimize its difference from one or zero for positive or negative anchors, respectively. A detailed description of the softmax layer and its updating process can be found in the research of Cha et al. (2017). The regression layer, which is the regressor introduced by Girshick et al. (2014), predicts the center coordinates, width, and height of a bounding box, and is trained to map the predicted box to a ground truth box.

RPN is trained end-to-end, for both classification and regression layers, by mini-batch gradient descent (MBGD). The MBGD computes gradients for one image per iteration by randomly selecting 256 mini-batches (half positive and half negative anchors) to train RPN using the following loss function:

$$L(p_i, p_i^*, t_i, t_i^*) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{1}{N_{reg}} \sum_{j \in \{x,y,w,h\}} p_i^* L_{reg}(t_{i,j}, t_{i,j}^*) \tag{1}$$

where i is related to each anchor in the mini-batch, and p_i^* and p_i are the ground truth label and predicted probability of being an object in the anchor, respectively. Note that p_i^* is zero or one for negative or positive labels of an anchor, respectively. In Equation (1), when an anchor is negative, just the term including L_{cls} is active to improve classification. For normalization of classification and regression terms in multitask loss, the constants N_{cls} and N_{reg} are used (Ren et al., 2016), and are equal to the mini-batch size and number of anchors divided by 10, respectively. The variables $t_{i,j}$ and $t_{i,j}^*$ in Equation (1) are used to define the geometrical differences between the predicted bounding box and anchor, as well as the ground truth box and the anchor, and is calculated as:

$$\begin{bmatrix} t_{i,x}, t_{i,y} \\ t_{i,w}, t_{i,h} \\ t_{i,x}^*, t_{i,y}^* \\ t_{i,w}^*, t_{i,h}^* \end{bmatrix} = \begin{bmatrix} (x_i - x_{i,a})/w_{i,a}, (y_i - y_{i,a})/h_{i,a} \\ \log(w_i/w_{i,a}), \log(h_i/h_{i,a}) \\ (x^* - x_{i,a})/w_{i,a}, (y^* - y_{i,a})/h_{i,a} \\ \log(w^*/w_{i,a}), \log(h^*/h_{i,a}) \end{bmatrix} \tag{2}$$

where (x_i, y_i), $(x_{i,a}, y_{i,a})$, and (x^*, y^*) are center coordinates of the predicted bounding for the ith anchor, the anchor, and the ground truth with the highest IoU with the anchor, respectively. Variables (w_i, h_i), $(w_{i,a}, h_{i,a})$, and (w^*, h^*) are the width and height of predicted bounding box, anchor, and ground truth box, respectively. Figure 3 shows an example of the geometry of an anchor, a predicted bounding box, and a ground truth box, respectively.

The predicted bounding box parameters are trained to improve their overlap with those of the ground truth

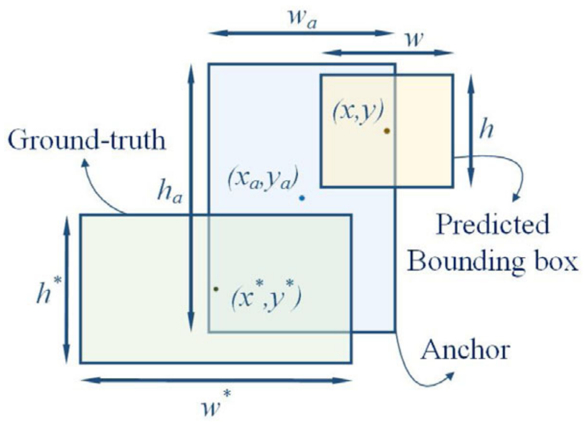

Fig. 3. The geometry of an anchor, a predicted bounding box, and a ground truth box.

boxes. The log loss function is used as the classification loss function, L_{cls}, and regression loss function, L_{reg} is

$$L_{reg}(X_1, X_2) = \begin{cases} 0.5(X_1 - X_2)^2 & \text{if} |X_1 - X_2| < 1 \\ |X_1 - X_2| - 0.5 & \text{otherwise} \end{cases} \quad (3)$$

where X_1 and X_2 are example variables. As can be seen in Equations (1)–(3), the variables of Equation (2) are used in Equation (3) to calculate the geometrical differences between the predicted bounding box and the ground truth box to use as the regression loss. These differences are minimized in the training process to improve the overlap between the predicted bounding box and ground truth box. A detailed description of the updating process of bounding boxes can be found in the research of Girshick et al. (2014).

2.3 Fast R-CNN

The schematic architecture of Fast R-CNN, which localizes and classifies objects in images, is shown in Figure 4. The Fast R-CNN takes precomputed object proposals from RPN and uses CNNs, like RPN, to extract a features map from the input image. Features bound by object proposals are called a region of interest (RoI). In the RoI pooling layer (Figure 4), precomputed object proposals are overlaid on the feature map. RoI pooling takes RoIs and applies max pooling operation to extract a fixed-size feature vector from each RoI. These vectors are fed into FC layers, followed by two regression and softmax layers, to calculate the location of bounding boxes and classify objects in the boxes.

For each RoI, the regressor outputs four parameters that represent the center coordinates (T_x^u, T_y^u) of object bounding boxes, as well as their height (T_h^u) and width (T_w^u); softmax outputs the probability ($P = (P_0, P_1, \ldots, P_k)$) of $k+1$ classes (k training class + 1 background class). If the IoU ratio between a RoI and a ground truth is more than 0.5 (Girshick, 2015), its label is positive. By contrast, the label of the RoI is background if the IoU is between 0.1 and 0.5.

MBGD is used to train end-to-end Fast R-CNN using Equation (4) as loss function where u and v represent the label and coordinates (center coordinates, height, and width) of the bounding box of each ground truth, $v = (v_x, v_y, v_h, v_w)$. Parameter u is one or zero for positive or background RoIs, respectively. L_{cls} is log loss and operates as the softmax's loss, whereas L_{reg} (Equation (3)) works as regression loss. Two images per iteration and 64 RoIs as mini-batches (25% positive and 75% negative) for each image are randomly selected to train Fast R-CNN (Girshick, 2015).

$$L(P, u, T^u, v) = L_{cls}(P, u) + [u \geq 1] \sum_{i \in \{x,y,w,h\}} L_{reg}(T_i^u, v_i) \quad (4)$$

2.4 ZF-Net architecture of CNN

The same architecture of CNN for RPN and Fast R-CNN should be used to share their features for Faster R-CNN. There are many well-known architectures for CNNs (i.e., VGG-16 (Simonyan and Zisserman, 2014), Microsoft ResNet-152 (He et al., 2016), and GoogleNet (Szegedy et al., 2015)). However, ZF-net (Zeiler and

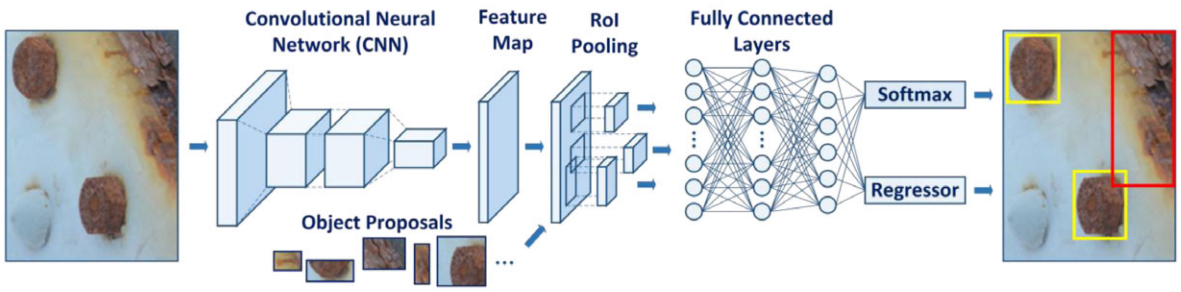

Fig. 4. The schematic architecture of the Fast R-CNN.

Table 1
The detailed specifications of RPN's layers

Layer	*Type*	*Depth*	*Filter size*	*Stride*	*Layer*	*Type*	*Depth*	*Filter size*	*Stride*
1	CONV+ReLU	96	7 × 7	2	7	CONV+ReLU	384	3 × 3	1
2	LRN	–	–	–	8	CONV+ReLU	384	3 × 3	1
3	Max pooling	96	3 × 3	2	9	CONV+ReLU	256	3 × 3	1
4	CONV+ReLU	256	5 × 5	2	10	Sliding CONV+ReLU	256	3 × 3	1
5	LRN	–	–	–	11	FC	256	–	–
6	Max pooling	256	3 × 3	2	12	Softmax & Regressor	–	–	–

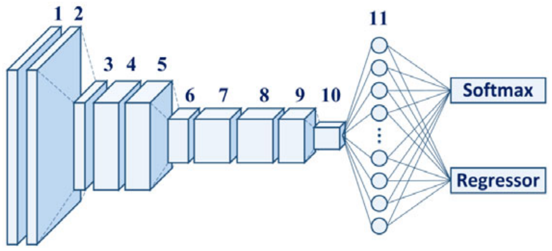

Fig. 5. The modified architectures of ZF-net for RPN.

Fergus, 2014) has the fastest training and testing speed and can be used for real-time detection, as demonstrated in many studies (Ren et al., 2016; Li et al., 2016). Zeiler and Fergus (2014) introduced their network (ZF-net) with eight weighted CONV and FC layers, and won the Large-Scale Visual Recognition Challenge 2013 for its classification (Russakovsky et al., 2015). The architecture and layer specifications of ZF-net are composed of 13 layers with CONV, local response normalization (LRN), max pooling, FC, and softmax layers. All CONV layers are followed by a ReLU activation function and have zero padding, in the way that keeps their constant spatial resolution. Max-pooling layers perform spatial pooling using 3 × 3 filters with a two-pixel stride and zero padding equal to 1 pixel. These CONV, LRN, and pooling layers are followed by three FC layers and a softmax layer, which are customized for the 1,000 class ImageNet data set.

To develop a Faster R-CNN-based method to detect multiple types of structural damage, the original ZF-net architecture is modified here for RPN and Fast R-CNN. To develop RPN, the original ZF-net is modified. The last max-pooling and FC layers of ZF-net are replaced by the sliding CONV, followed by a FC layer (feature vector) with 256 in depth, and its softmax layer is replaced with softmax and regression layers as shown in Figure 5. The details of the ZF-net-based RPN are presented in Table 1.

To modify for Fast R-CNN, the last max-pooling layer of ZF-net is replaced by the RoI pooling layer. To prevent overfitting during the training process, dropout layers with a threshold of 0.5 are added between the first and second FC layers and between the second and third FC layers of ZF-net. The depth of the last FC layer is modified to six for the five damage types and background to ensure compatibility with the problem. The softmax layer is replaced with softmax and regression layers as shown in Figure 6. The details of the ZF-net-based RPN are presented in Table 2.

2.5 Faster R-CNN by sharing CNN between RPN and Fast R-CNN

In Faster R-CNN, computations of the CNN for feature extraction are shared between RPN and Fast R-CNN, as shown in Figure 6. Also, as can be seen in the previous section, the first nine layers of both RPN and Fast

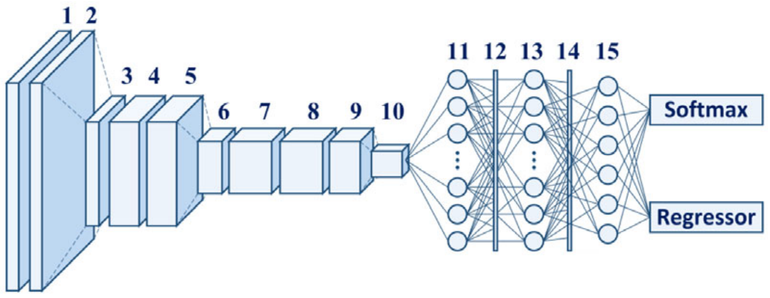

Fig. 6. The modified architectures of ZF-net for Fast R-CNN.

Table 2
The detailed specifications of Fast R-CNN's layers

Layer	*Type*	*Depth*	*Filter size*	*Stride*	*Layer*	*Type*	*Depth*	*Filter size*	*Stride*
1	CONV+ReLU	96	7 × 7	2	9	CONV+ReLU	256	3 × 3	1
2	LRN	–	–	–	10	RoI pooling	256	–	–
3	Max pooling	96	3 × 3	2	11	FC+ReLU	4,096	–	–
4	CONV+ReLU	256	5 × 5	2	12	Dropout	–	–	–
5	LRN	–	–	–	13	FC+ReLU	4,096	–	–
6	Max pooling	256	3 × 3	2	14	Dropout	–	–	–
7	CONV+ReLU	384	3 × 3	1	15	FC+ReLU	6	–	–
8	CONV+ReLU	384	3 × 3	1	16	Softmax & Regressor	–	–	–

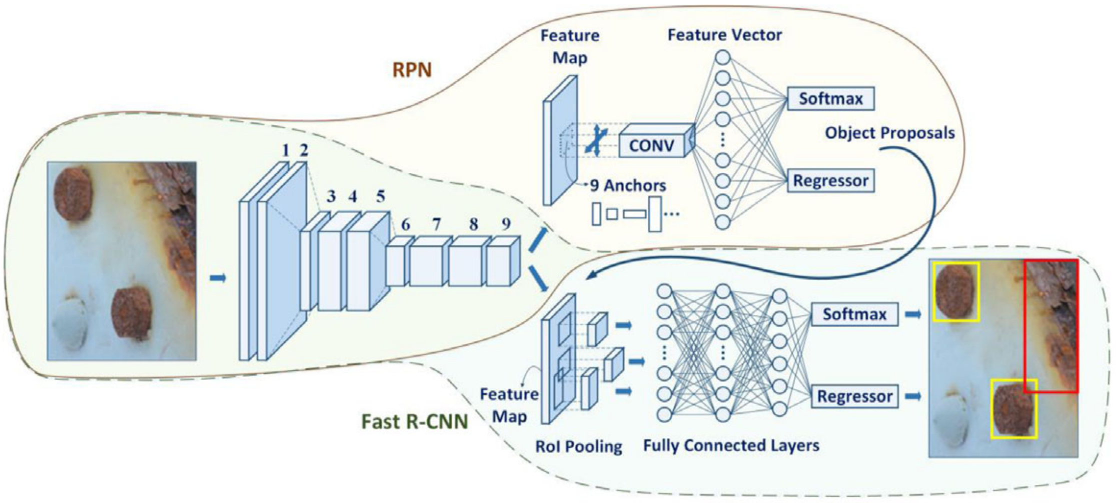

Fig. 7. The schematic architecture of the Faster R-CNN.

R-CNN have the same specifications, and their computations can be shared. Figure 7 shows the eventual architecture of Faster R-CNN produced by a combination of Fast R-CNN and RPN and sharing these nine layers.

A four-step training process is used to fine tune the parameters of the RPN and Fast R-CNN. In the first step, RPN is trained with initial weights, and object proposals are prepared for Fast R-CNN. Then, Fast R-CNN is initialized with the trained weights from step one. In step three, RPN is initialized with the final weights of the previous step and trained again. For the last step, Fast R-CNN takes the object proposals generated in step three and is trained with the initial parameters trained in step three. As RPN may produce more than 2,000 object proposals for an image, which causes costly computations and may decrease the accuracy of object detection (Girshick, 2015; Ren et al., 2016; Fan et al., 2016), outputs of RPN are sorted based on the score of its softmax layer, and the first 2,000 objects proposals (if there are more than 2,000 generated proposals) with the highest scores are fed into Fast R-CNN in the second stage of training. Using the methods of Ren et al. (2016) for the fourth stage of training as well as the testing stage, the first 300 object proposals with the highest scores are used to increase the speed of detection.

3 DATABASE AND IMPLEMENTATION DETAILS

3.1 Training, validation, and testing databases

To develop a database containing steel delamination, steel corrosion (medium and high), bolt corrosion, and concrete cracks, 297 images (with a resolution of 6,000 × 4,000 pixels) are collected using a Nikon D5200 and D7200 DSLR cameras. Images are taken under different lighting conditions at a distance of 1.0–1.5 m from the camera. The images are taken from two bridges and a building complex of the University of Manitoba in Winnipeg, Manitoba, Canada. These images are cropped to 500 × 375 pixels, and 2,366 cropped images

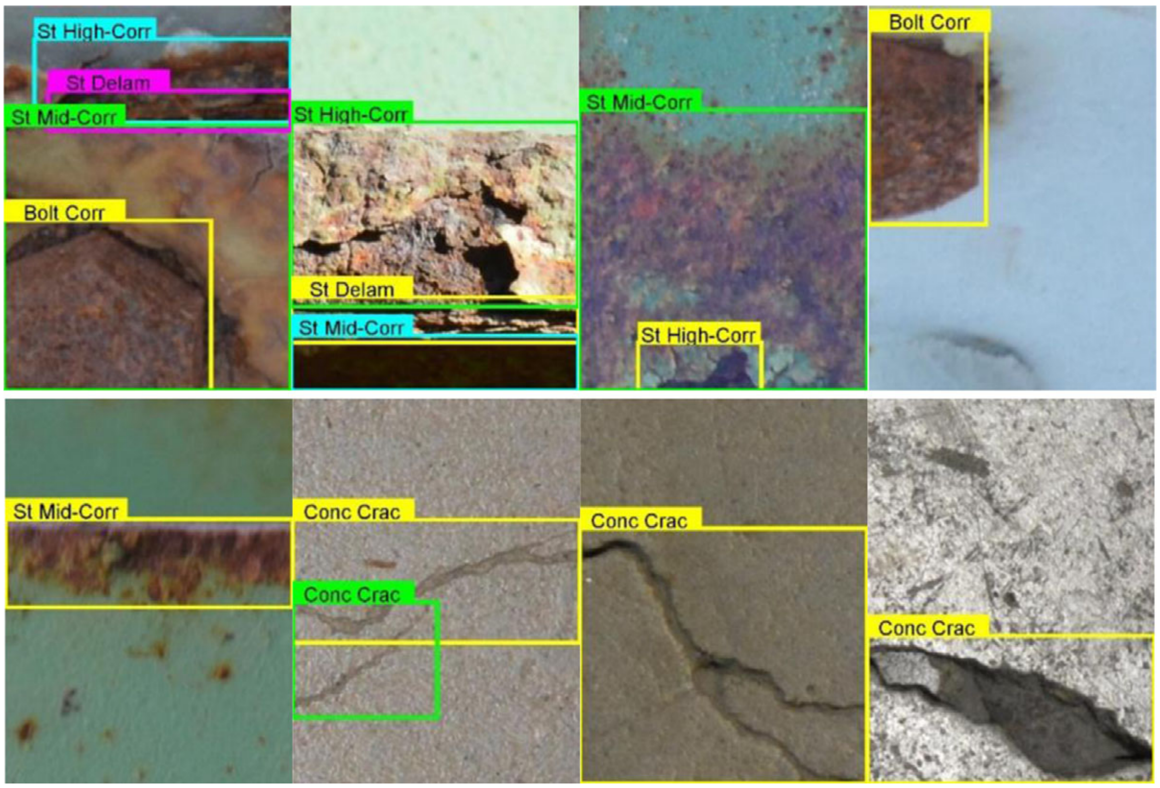

Fig. 8. Sample images with corresponding bounding boxes and labels.

that include the structural damage types are chosen to generate the database.

To annotate the labels of the objects (damage type) and the coordinates of their corresponding bounding boxes in images, a code in a MATLAB environment is developed to specify them manually. During the annotation process, the labels and bounding boxes for 4,695 objects are specified in the 2,366 images. Figure 8 shows the examples of annotated images. It should be mentioned that corrosion under the covered steel sections that cause color change and cover deformation is considered to be medium corrosion. In contrast, a corrosion is deemed to be high corrosion when the cover is removed and a rough corroded steel surface is observed.

To generate the testing data set, images are randomly selected from the annotated images so that the selected testing set contained at least 30% of images of each damage type, and simultaneously, included approximately 30% objects of each damage type. The remaining images not selected for the testing data set are used to create a training and validation data set. Data augmentation is a way to increase the performance of CNNs and can reduce the probability of overfitting, in addition to dropout, on small databases. Following the work of He et al. (2014), we perform horizontal flipping (Figure 9) on the training and validation set for data augmentation. Table 3 shows the detailed proportions of the testing, training, and validation sets after data augmentation.

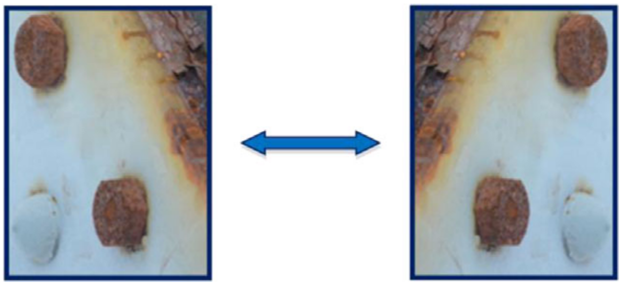

Fig. 9. Horizontal flipping for data augmentation.

3.2 Implementation details

All experiments are performed using the open source Faster R-CNN library (Ren et al., 2016), MATLAB 2016a, CUDA 8.0, and CUDNN 5.1 on a computer with a Core i7-6700k @4 GHz CPU, 32 GB DDR4 memory, and 8 GB memory ASUS Turbo GeForce GTX 1080 graphics processing unit (GPU). The layers of the CNN and FC layers are initialized by zero-mean Gaussian

Table 3
The proportion of training, validation, and testing sets

	Training and validation		*Testing*	
Damage class	*Objects*	*Number of images*	*Objects*	*Number of images*
Medium steel corrosion	623	285	274	124
Steel delamination	759	286	309	124
High steel corrosion	667	287	296	124
Concrete cracks	407	348	167	152
Bolt corrosion	822	447	371	189

distribution with standard deviations of 0.01 and 0.001, respectively. The RPN and Fast R-CNN networks are trained with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005 for 80,000 and 40,000 iterations, respectively.

Average precision (AP) is often used to evaluate the performance of an object detector (Girshick et al., 2014; He et al., 2014; Girshick, 2015; Ren et al., 2016), and summarizes the precision/recall curve by calculating the area under the curve (Everingham et al., 2010). For a given class, precision and recall are defined as the proportions of true positives in the retrieved results and the proportions of true positives that are from the positive class, respectively. Mean AP (mAP) is defined as the average of calculated APs for all classes. Detailed descriptions of precision/recall curve and AP can be found in the paper by Everingham et al. (2010).

Ren et al. (2016) acknowledged that in their study, they did not choose initial parameters of Faster R-CNN carefully. Although Fan et al. (2016) studied the influence of these parameters on the performance of Faster R-CNN, they did not introduce a straightforward way to choose these parameters, so we had to find them via trial-and-error. To find the optimal anchor sizes and ratios, we examined 27 combinations of ratios from nine different ratios (0.2, 0.35, 0.5, 0.85, 1, 1.15, 1.7, 1.85, and 2) and two combinations of sizes from six different anchor sizes (96, 128, 192, 256, 384, and 512). In these experiments, the validation set is randomly chosen for each case from the training and validations set so that the selected validation testing set contained at least 30% of images from each damage type. Ren et al. (2016) labeled a RoI as the calculated class by the softmax layer of Fast R-CNN if the calculated probability of softmax for the RoI was 0.6 or more (for each class among the 20 classes of VOC database). If the probability was lower than 0.6 for the 20 classes of VOC, or was more than 0.6 for classifying as a background, Faster R-CNN did not consider the RoI to be the result of object detection. We found, via trial-and-error, that increasing this probability causes higher mAP in our problem, so we use the probability of 0.9, instead of 0.6, to create the outputs of our experiments.

Inputs to Faster R-CNN can be scaled to decrease the inappropriate effects scale variations in objects (Ren et al., 2016; Fan et al., 2016), although we do not apply any scaling on our images, which are taken from an approximately constant distance between the damages and camera and have insignificant scale invariances. Ren et al. (2016) applied scaling of 600 on the shorter side of images (with the average resolution of 500 × 375), and slid the sliding CONV layer of RPN with stride 16 on feature map. To maintain a constant spatial resolution, they performed convolution on the sliding window using a 3 × 3 filter with a 1-pixel stride and zero padding equal to 1 pixel. As reducing the sliding step of the CONV may lead to a higher accuracy (Ren et al., 2016), we choose stride 16 via trial-and-error. It should be noted that we input images without scaling them, thus our stride is lower in comparison to the default stride of Faster R-CNN. In our trial-and-error process, we found that if we applied scaling, mAP could decrease as our database is scale invariant.

4 EXPERIMENTS

4.1 Training, validation, and test results

The network is trained using the four-step training strategy for 40 cases, and its precision is evaluated with the test set. The training time using GPU mode is approximately 4 hours, and using CPU mode would be approximately 4.5 days. The time required to evaluate a 500 × 375-pixel image is 0.03 seconds in GPU mode and 0.45 seconds in CPU mode. We record the AP for the five types of damages and mAP (Figure 10 and Table A1).

The highest AP for concrete cracks is 90.9% in Case 29, although the APs for the other classes in these cases may be lower than their average. To ensure a reasonable balance among APs, we choose Case 29, which had the highest mAP (87.8%) and APs with 90.6, 83.4, 82.1,

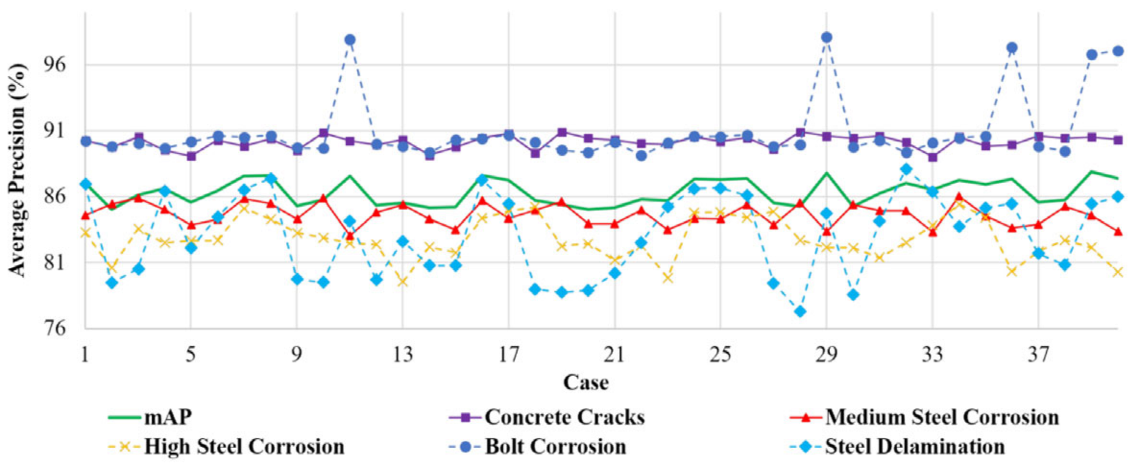

Fig. 10. The performance of the network for the testing set.

98.1, and 84.7 for concrete cracks, medium steel corrosion, high steel corrosion, bolt corrosion, and steel delamination, respectively. Case 29 has anchors with the ratios of 0.20, 0.85, and 1.70 and the sizes of 96, 192, and 384.

4.2 Testing new images

To provide a clear understanding of how the trained Faster R-CNN detects structural damage, we perform new tests using 11 new 6,000 × 4,000-pixel images. These images are taken under various lighting conditions that were studied in previous traditional CNN-based method (Cha et al., 2017) at a distance of 1.0–1.5 m from the camera. To ensure the same precision produced for Case 29 in Table A1, each image is inputted into the network in 128 separate parts (with 500 × 375 resolution), and its output parts are reassembled. The outputs show mAP with 89.7% and APs with 94.7, 91.8, 86.1, 90.9, and 85.2 for concrete cracks, medium steel corrosion, high steel corrosion, bolt corrosion, and steel delamination, respectively, which are similar to the APs of Case 29. The input and output images are shown in Figures 11–13.

Figure 11 includes images with steel damage in intense lighting. Although the most damaged parts are detected in Figure 11a, three instances of bolt corrosion are incorrectly classified to medium steel corrosion (indicated with a solid black box). Here, corroded bolts in intensive sunlight are suitably detected, although rotating the angle of the camera is required to detect all steel-corroded locations. Both Figures 11b and c show suitable results in intensive light and shadow, but there are four incorrectly detected areas as corroded bolts (indicated with solid black boxes), as well as areas in intensive sunlight with distances more than 1.5 m from the camera that are not detected. These problems can be resolved by rotating the angle of the camera with a movable unmanned aerial vehicle (UAV).

Images including concrete cracks are shown in Figure 12. Concrete cracks are successfully detected under different lighting conditions, although one incidence of incorrect detection is shown in Figure 12a, which are under intensive lighting conditions and the cracks are smaller than the training set. In Figure 12b, one location is misclassified as steel delamination. In Figures 12c and d, two locations with patterns similar to cracking are detected as concrete cracks, respectively. However, these errors are minor compared to the overall success of detection and can be resolved by a larger training database or rotating the angle of the camera.

Figure 13 includes images of steel damage under uniform lighting conditions and, similar to Figure 12, damage is satisfactorily detected using the Faster R-CNN. In Figure 13a, at some locations, corroded bolts are detected as steel corrosion. Moreover, a corroded steel area with a distance less than 1.0 m to the camera is not detected successfully. In Figure 13c, one bolt corrosion is incorrectly detected, in addition to some minor areas that are not detected or wrongly detected. These errors can be minimized by developing a bigger training database. In Figure 13d, in addition to good performance in detecting areas that include corroded or delaminated steel, there is one incorrectly detected area in the lower right corner. The image in Figure 13d also contains two intact bolts that are not detected because our network is trained to detect just corroded bolts (in addition to the other mentioned damage types).

Despite minor errors, the results demonstrate the suitable performance of our Faster R-CNN-based method in the autonomous visual inspection of civil infrastructure. These minor errors may be caused by

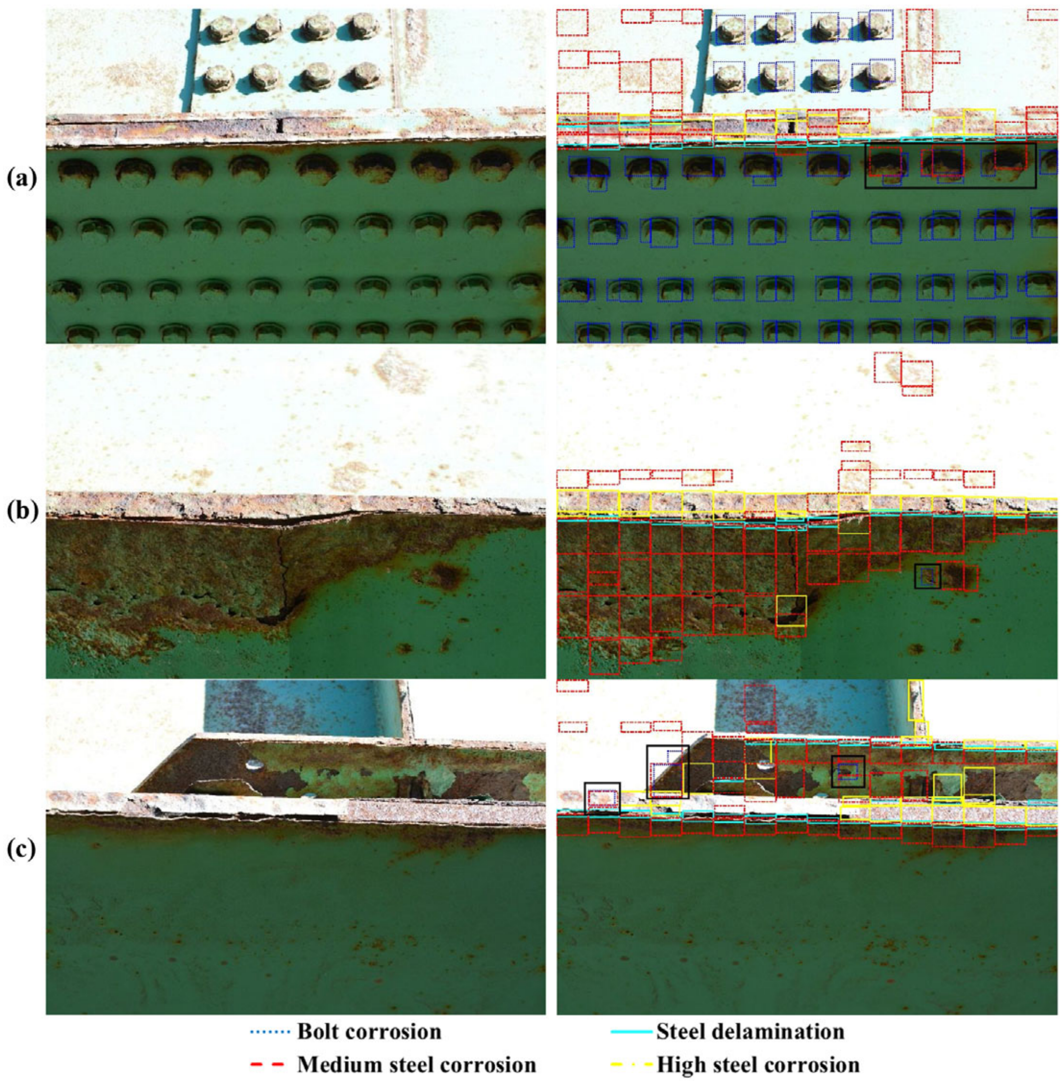

Fig. 11. Detected steel damages in intensive lighting; new testing images (left) and output of the Faster R-CNN (right).

camera angle, the small training database, or distances that are too close to or far from the camera. Therefore, in future studies, a larger database with a wider range of distances between structural damages and the camera will be generated to improve the method's capacity and generalization. Additionally, the network will be used with a UAV to solve the problem of the camera's angle.

4.3 Comparative study

To compare the performances of the proposed Faster R-CNN-based approach and the traditional CNN-based crack detection method (Cha et al., 2017), the same original training images for concrete crack detection from the authors' previous works (Cha et al., 2017) were used. For a comparative test, a new 6,000 $\times$ 4,000-pixel image is taken using a Nikon 5200 DSLR camera from the Engineering Information and Technology Complex at the University of Manitoba. This image contained a concrete surface with cracks.

As can be seen in Figure 14, the traditional CNN-based method also shows good results with high accuracy as claimed in the previous work (Cha et al., 2017). The new Faster R-CNN based method has also similar results, but it provides more optimized bounding boxes for damage localization. The Faster R-CNN uses various different bounding boxes during the preparation of the training data for cracks as opposed to the fixed sliding window size of the traditional CNN-based method. Moreover, the RPN of the Faster R-CNN uses anchors having nine different bounding boxes to localize the object instead of fixed sliding window that the traditional

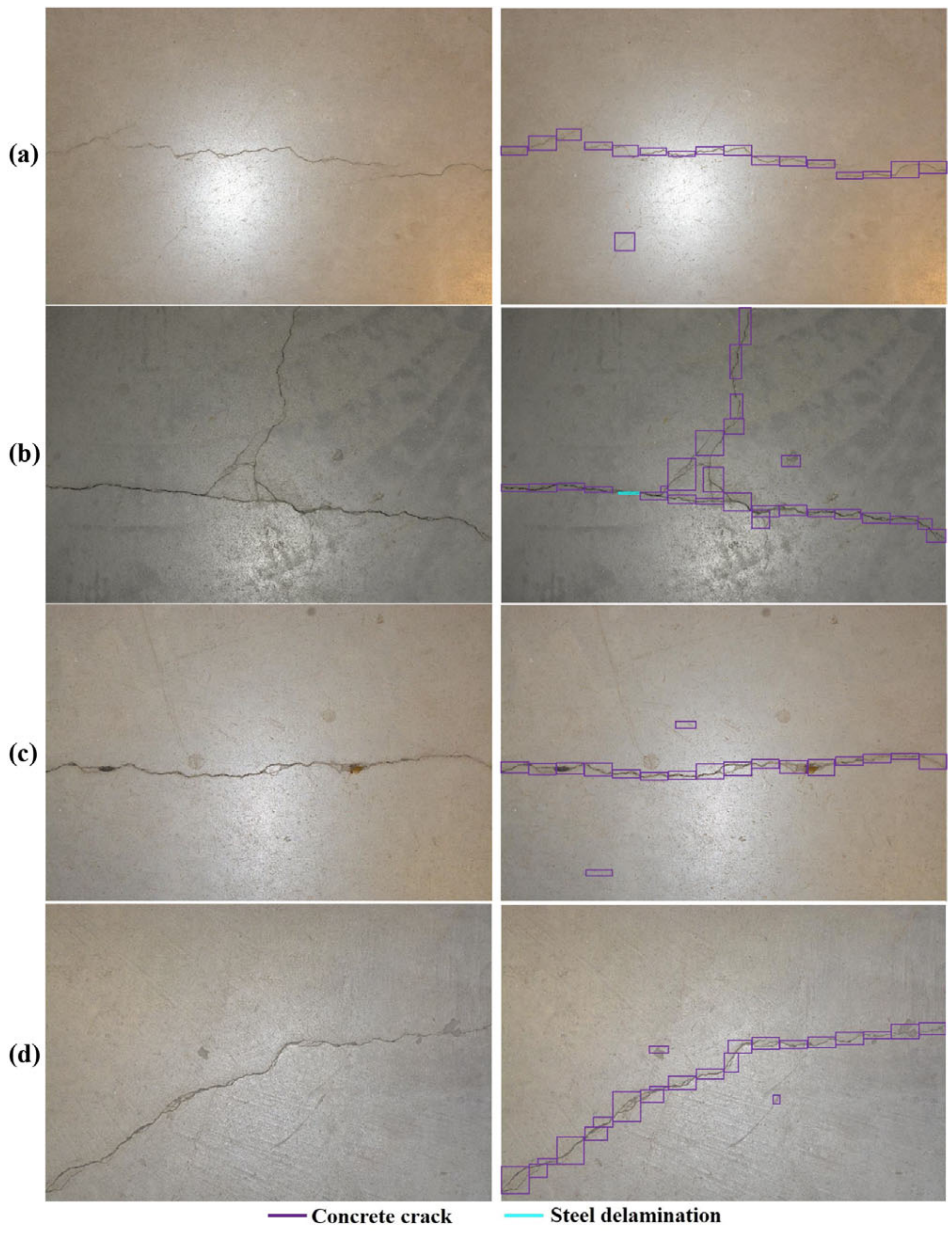

Fig. 12. Detected concrete cracks; new testing images (left) and output of the Faster R-CNN (right).

CNN based method (Cha et al., 2017) used. Therefore, the flexible sizes of the bounding boxes for the Faster R-CNN during the preparation of the training data and the localization process through the RPN detect more varying lengths and shapes of cracks. Thus, Faster R-CNN provides better results with the advantage of flexible bounding box technique in terms of damage localization.

The Faster R-CNN has faster processing of the testing images than the traditional CNN because the original testing input image is only processed one time by the CNN (i.e., layers 1–9) and the Faster R-CNN applies bounding boxes (i.e., anchors) for the localization to the trained "feature map" (Figure 7). However, the traditional CNN-based method applies "sliding window" to the original image many times implementing the processing of the entire network for each sliding window, and the sliding windows are overlapped 50% for the detection of crack in edges (Cha et al., 2017). Moreover, it is not easy to determine the sliding window size because

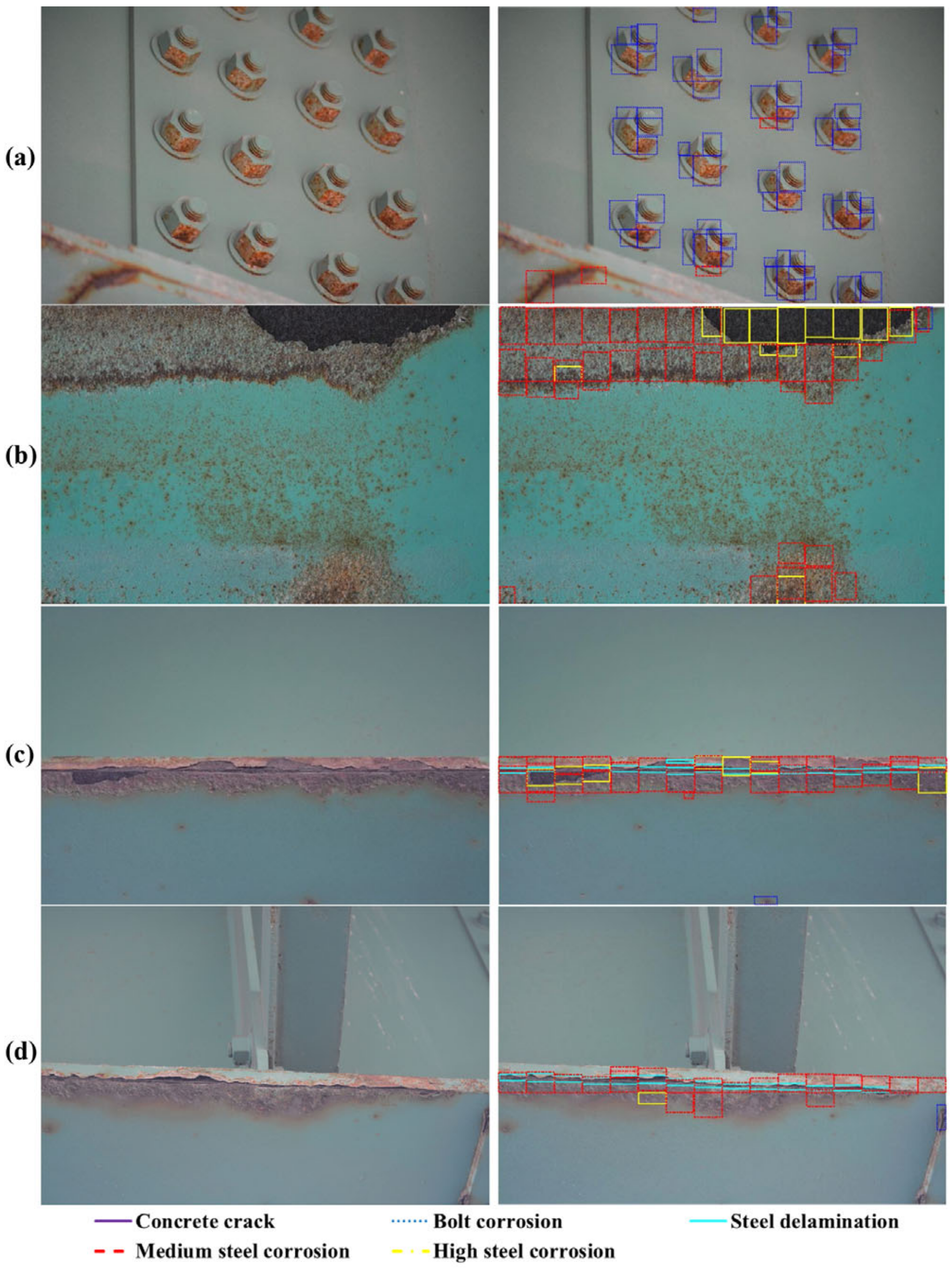

Fig. 13. Detected steel damages under normal light; new testing images (left) and output of Faster R-CNN (right).

the input testing images may have different scales and sizes with different distances between the camera and the object.

5 TESTING ON VIDEO

The trained Faster R-CNN achieved 0.03 seconds per 500 × 375 resolution image in the detection of structural damage, which provides enough speed for quasi real-time detection. This test speed provided an evaluation of 1.4 frames per second if we had 1,920 × 1,080-pixel frames, including ten 500 × 375-pixel frames. To make the Faster R-CNN compatible with the videos, we use videos with a 30.00 frame rate and 1,920 × 1,080 sizes. Figure 15 shows a sample of 28 sequential frames (frames 35–62) from the provided video for concrete detection.

This method accomplished quasi real-time autonomous damage detection if a fixed camera is used with a desktop computer. It can solve the problem of

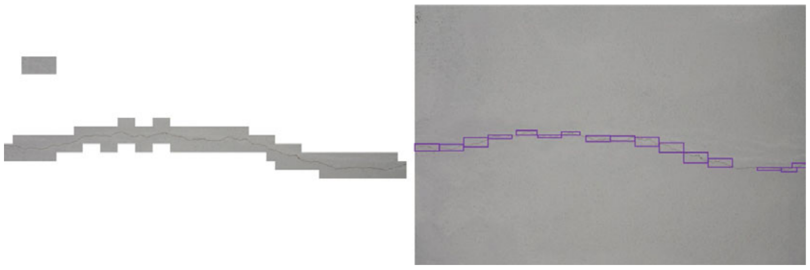

Fig. 14. Detected concrete cracks by the CNN (left) and the Faster R-CNN (right).

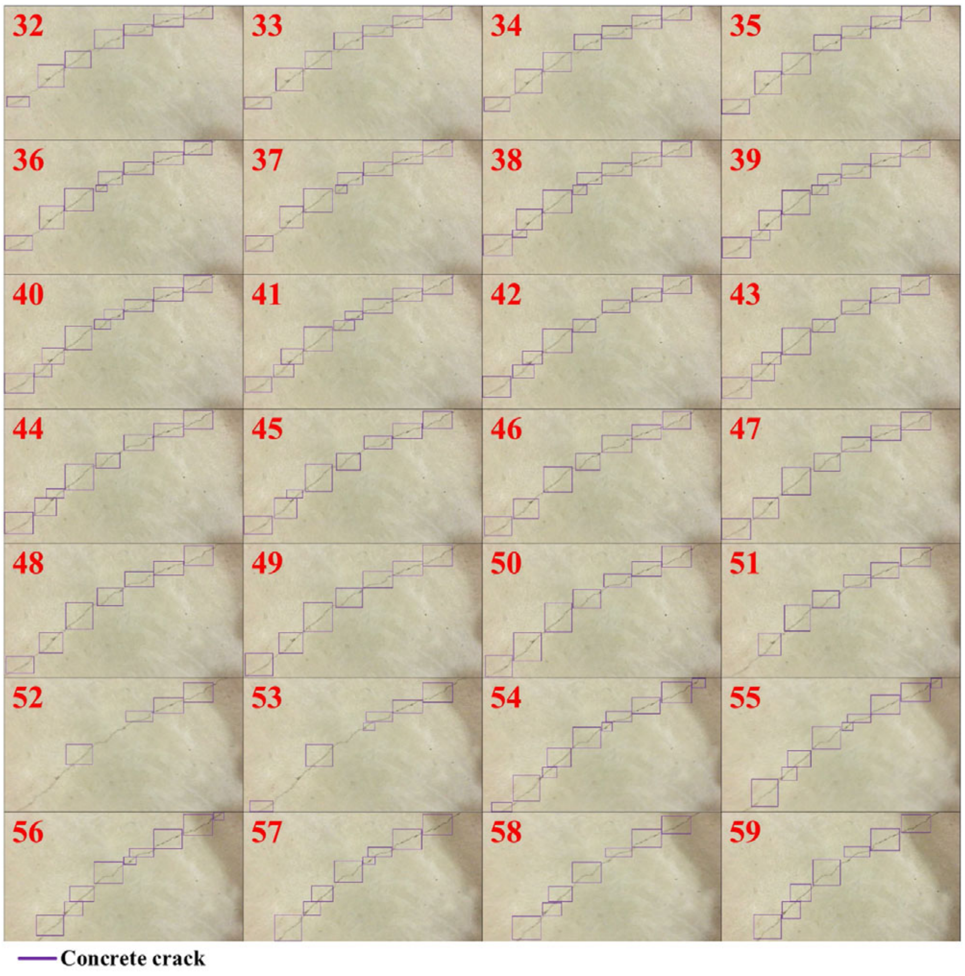

Fig. 15. A sample of 28 sequential frames.

camera angle limitations and replace human-based visual inspection using UAVs, even though it would not be quasi real-time processing.

6 CONCLUSIONS

Our previous method used traditional CNN with a sliding window to localize the damage, but it is difficult to determine the size of the window if the testing input images have different sizes and scales, and it may require a tremendous amount of background images to detect multiple damage types for robust results. To overcome these obstacles, we developed a structural damage detection method based on Faster R-CNN to detect five types of surface damages: concrete cracks, steel corrosion (medium and high levels), bolt corrosion, and steel delamination. Nikon D5200 and D7200

DSLR cameras were used to collect 297 images with 6,000 × 4,000 resolution. The images were cropped into 500 × 375-pixel images. Among the cropped images, 2,366 images containing the five types of structural surface damages were selected, and the location of each damage and its corresponding label was specified. The testing database was generated using random selection of these annotated images. To find the maximum accuracy, the initial parameters of the method to generate object proposals were selected via trial-and-error. Based on the best case (i.e., Case 29), the test results showed 94.7%, 91.8%, 86.1%, 90.9%, and 85.2% AP for the five damage classes, respectively, and an 89.7% mAP. The images not selected for the testing data set were doubled by data augmentation and used to create a training and validation data set. The robustness of the network was investigated and validated using 11 new images collected from different structures. In addition, the performance of the trained network was compared with the CNN-based method of Cha et al. (2017) for concrete crack detection. Due to different amounts of data in the training sets for each method, it is not easy to conclude superiority in either method for detection and localization of multiple types of damage. However, it can be claimed that Faster R-CNN has better computation efficiency based on the optimized architecture of the network, and it provides more flexible sizes of the bounding boxes to accommodate for different sizes and scales of the input images. In addition, a video-based damage detection framework using the Faster R-CNN was developed, which can provide quasi real-time, autonomous vision-based structural damage detection.

In the future, more samples with a wider range of distances between the damages and camera will be provided and added to the database to increase the accuracy and robustness of the proposed method, extensive comparative studies will be performed, and the Faster R-CNN will be used with UAVs to replace human-oriented inspection with an autonomous visual inspection.

REFERENCES

Abdel-Qader, I., Abudayyeh, O. & Kelly, M. E. (2003), Analysis of edge-detection techniques for crack identification in bridges, *Journal of Computing in Civil Engineering*, **17**(4), 255–63.

Adeli, H. & Jiang, X. (2009), *Intelligent Infrastructure—Neural Networks, Wavelets, and Chaos Theory for Intelligent Transportation Systems and Smart Structures*, CRC Press, Taylor & Francis, Boca Raton, FL.

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**, 361–78.

Cha, Y.-J., You, K. & Choi, W. (2016), Vision-based detection of loosened bolts using the Hough transform and support vector machines, *Automation in Construction*, **71**(2), 181–88.

Chen, P.-H., Shen, H.-K., Lei, C.-Y. & Chang, L.-M. (2012), Support-vector-machine-based method for automated steel bridge rust assessment, *Automation in Construction*, **23**, 9–19.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011), Flexible, high performance convolutional neural networks for image classification, in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, 16–22 July 2011, 1237–42.

Cord, A. & Chambon, S. (2012), Automatic road defect detection by textural pattern recognition based on AdaBoost, *Computer-Aided Civil and Infrastructure Engineering*, **27**(4), 244–59.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J. & Zisserman, A. (2010), The Pascal visual object classes (VOC) challenge, *International Journal of Computer Vision*, **88**(2), 303–38.

Everingham, M., Zisserman, A., Williams, C. K., Van Gool, L., Allan, M., Bishop, C. M., Chapelle, O., Dalal, N., Deselaers, T. & Dorkó, G. (2007), The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. Available at: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/, accessed June, 2017.

Fan, Q., Brown, L. & Smith, J. (2016), A closer look at Faster R-CNN for vehicle detection, in *Proceedings of 2016 IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, 19–22 June 2016, 124–29.

German, S., Brilakis, I. & DesRoches, R. (2012), Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments, *Advanced Engineering Informatics*, **26**(4), 846–58.

Girshick, R. (2015), Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 07–13 December 2015, 1440–48.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 23–28 June 2014, 580–87.

Graybeal, B. A., Phares, B. M., Rolander, D. D., Moore, M. & Washer, G. (2002), Visual inspection of highway bridges, *Journal of Nondestructive Evaluation*, **21**(3), 67–83.

He, K., Zhang, X., Ren, S. & Sun, J. (2014), Spatial pyramid pooling in deep convolutional networks for visual recognition, in *Proceedings of the 13th European Conference on Computer Vision*, Zurich, Switzerland, 6–12 September 2014, 346–61.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 12 December 2016, 770–78.

Koch, C. & Brilakis, I. (2011), Pothole detection in asphalt pavement images, *Advanced Engineering Informatics*, **25**(3), 507–15.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in *Proceedings of the Neural Information Processing Systems Conference*, Stateline, NV, 3–8 December 2012.

Li, C., Kang, Q., Ge, G., Song, Q., Lu, H. & Cheng, J. (2016), Deep: learning deep binary encoding for multi-label classification, in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Las Vegas, NV, 26 June–1 July 2016, 744–51.

Liao, K.-W. & Lee, Y.-T. (2016), Detection of rust defects on steel bridge coatings via digital image recognition, *Automation in Construction*, **71**(2), 294–306.

Mirzaei, G., Adeli, A. & Adeli, H. (2016), Imaging and machine learning techniques for diagnosis of Alzheimer disease, *Reviews in the Neurosciences*, **27**(8), 857–70.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel, 21–24 June 2010, 807–14.

Nishikawa, T., Yoshida, J., Sugiyama, T. & Fujino, Y. (2012), Concrete crack detection by multiple sequential image filtering, *Computer-Aided Civil and Infrastructure Engineering*, **27**(1), 29–47.

O'Byrne, M., Schoefs, F., Ghosh, B. & Pakrashi, V. (2013), Texture analysis based damage detection of ageing infrastructural elements, *Computer-Aided Civil and Infrastructure Engineering*, **28**(3), 162–77.

Park, J., Kim, T. & Kim, J. (2015), Image-based bolt-loosening detection technique of bolt joint in steel bridges, in *Proceedings of the 6th International Conference on Advances in Experimental Structural Engineering (6AESE)*, Urbana–Champaign, IL, 1–2 August 2015.

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

Ren, S., He, K., Girshick, R. & Sun, J. (2016), Faster R-CNN: towards real-time object detection with region proposal networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **39**(6), 1137–49.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015), ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, **115**(3), 211–52.

Ryan, T., Mann, J., Chill, Z. & Ott, B. (2012), *Bridge Inspector's Reference Manual (BIRM)*, Report, Federal Highway Administration (FHWA), Report No. FHWA NHI 12–049, 2012.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. & Lecun, Y. (2014), Overfeat: integrated recognition, localization and detection using convolutional networks, in *Proceedings of the International Conference on Learning Representations (ICLR2014)*, Banff, Canada, 14–16 April 2014.

Siddique, N. H. & Adeli, H. (2016), Brief history of natural sciences for nature-inspired computing in engineering, *Journal of Civil Engineering and Management*, **22**(3), 287–301.

Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, 7–9 May 2015.

Sinha, K., Fieguth, P. W. & Polak, M. A. (2003), Computer vision techniques for automatic structural assessment of underground pipes, *Computer-Aided Civil and Infrastructure Engineering*, **18**(2), 95–112.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 7–12 June 2015, 1–9.

Uijlings, J. R., Van De Sande, K. E., Gevers, T. & Smeulders, A. W. (2013), Selective search for object recognition, *International Journal of Computer Vision*, **104**(2), 154–71.

Wu, L., Mokhtari, S., Nazef, A., Nam, B. & Yun, H.-B. (2016), Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment, *Journal of Computing in Civil Engineering*, **30**(1), 04014118, 1–19.

Yeum, C. M. & Dyke, S. J. (2015), Vision-based automated crack detection for bridge inspection, *Computer-Aided Civil and Infrastructure Engineering*, **30**(10), 759–70.

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

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

Zeiler, M. D. & Fergus, R. (2014), Visualizing and understanding convolutional networks, in D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (eds.). *Computer Vision – ECCV 2014. Lecture Notes in Computer Science*, vol. 8689, Part 1, 818–33, Springer, Cham, Switzerland.

Table A1
The performance of the network for the testing set

		Average precision (%)				
Case	*mAP (%)*	*Concrete cracks*	*Medium steel corrosion*	*High steel corrosion*	*Bolt corrosion*	*Steel delamination*
1	87.1	90.3	84.6	83.3	90.2	87.0
2	85.0	89.8	85.4	80.6	89.8	79.5
3	86.1	90.5	85.9	83.5	90.1	80.5
4	86.6	89.5	85.0	82.5	89.7	86.4
5	85.6	89.1	83.8	82.7	90.2	82.1
6	86.5	90.3	84.2	82.7	90.6	84.5
7	87.6	89.8	85.9	85.1	90.5	86.5
8	87.6	90.4	85.5	84.3	90.6	87.4
9	85.3	89.5	84.3	83.3	89.7	79.7
10	85.8	90.8	85.9	82.9	89.7	79.5
11	87.6	90.2	83.0	82.5	98.0	84.2
12	85.4	89.9	84.8	82.4	90.0	79.7
13	85.5	90.3	85.4	79.5	89.8	82.6
14	85.2	89.2	84.3	82.2	89.4	80.8
15	85.2	89.8	83.5	81.7	90.3	80.8
16	87.6	90.5	85.7	84.4	90.4	87.3
17	87.2	90.8	84.3	84.9	90.7	85.5
18	85.7	89.3	85.0	85.2	90.1	79.0
19	85.4	90.9	85.6	82.3	89.5	78.7
20	85.0	90.4	83.9	82.4	89.4	78.9
21	85.2	90.3	83.9	81.2	90.2	80.2
22	85.8	90.0	85.0	82.3	89.1	82.5
23	85.7	90.0	83.5	79.8	90.1	85.2
24	87.4	90.5	84.3	84.8	90.6	86.6
25	87.3	90.2	84.3	84.8	90.6	86.7
26	87.4	90.5	85.4	84.4	90.7	86.1
27	85.5	89.6	83.8	84.9	89.8	79.4
28	85.3	90.9	85.5	82.7	90.0	77.3
29	87.8	90.6	83.4	82.1	98.1	84.7
30	85.3	90.4	85.4	82.1	89.8	78.6
31	86.3	90.6	84.9	81.4	90.3	84.1
32	87.0	90.1	84.9	82.5	89.3	88.1
33	86.5	89.0	83.3	83.8	90.1	86.4
34	87.2	90.5	86.0	85.4	90.5	83.7
35	86.9	89.9	84.5	84.4	90.6	85.2
36	87.3	89.9	83.6	80.3	97.4	85.5
37	85.6	90.6	83.9	81.8	89.8	81.7
38	85.7	90.5	85.3	82.7	89.5	80.8
39	87.9	90.5	84.6	82.1	96.8	85.5
40	87.4	90.3	83.4	80.3	97.1	86.0
Average	86.3	90.2	84.6	82.8	91.0	83.1