Computer-Aided Civil and Infrastructure Engineering **0** (2018) 1–17

A Fast Detection Method via Region-Based Fully Convolutional Neural Networks for Shield Tunnel Lining Defects

Yadong Xue* & Yicheng Li

Department of Civil Engineering, Tongji University, Shanghai, China

Abstract: *Tunnel lining defects are an important indicator reflecting the safety status of shield tunnels. Inspired by the state-of-the-art deep learning, a method for automatic intelligent classification and detection methodology of tunnel lining defects is presented. A fully convolutional network (FCN) model for classification is proposed. Information about defects, collected using charge-coupled device cameras, was used to train the model. The model's performance was compared to those of GoogLeNet and VGG. The best-set accuracy of the proposed model was over 95% at a test-time speed of 48 ms per image. For defects detection, image features were computed from large-scale images by the FCN and then detected using a region proposal network and position-sensitive region of interest pooling. Some indices (detection rate, detection accuracy, and detection efficiency, locating accuracy) were used to evaluate the model. The comparisons with faster R-CNN and a traditional method were conducted. The results show that the model is very fast and efficient, allowing automatic intelligent classification and detection of tunnel lining defects.*

1 INTRODUCTION

Subways are an important means of urban transportation. However, owing to poor geological conditions, lack of timely maintenance and damage incurred by humans, subway tunnels very often suffer from surface and internal structural damage with increasing operation time. The most prominent defects in urban subway shield tunnels are cracks and leakages. These defects reduce the strength of the concrete, dampen the tunnel lining, negatively affect the structure safety, and shorten the tunnel lifetime; consequently, these defects significantly increase safety-related risks of operating urban rail traffic. With a large number of urban subway construction sites and operation of train lines (more than 5,083 km in China), it is very urgent to develop new efficient structural defect detection and evaluation methods.

The traditional inspection of tunnel lining structures, in addition to the total station, has mainly amounted to manual inspection, sketching and/or taking photos. The advent of machine vision techniques offers new and more effective solutions. For example, automatic crack detection system based on linear array charge-coupled device (CCD) camera has been designed for detection of structural cracks (Yu et al., 2007). Complementary metal-oxide semiconductor (CMOS) industrial cameras have also been used in machine vision for detection of defects (Zhang et al., 2014). Given large amounts of imaging data, the next key issue is how to identify and analyze the defect efficiently, accurately, and intelligently.

A variety of image processing techniques (IPTs) have been developed and used for detection of defects in the field of civil engineering. Abdel-Qader et al. (2003) compared the performances of edge-detection algorithms (fast Harr transform [FHT], fast Fourier transform [FFT], Sobel filtering, Canny filtering, etc.) and proved FHT was the best algorithm for detection of cracks in bridges. Some IPTs, based on mathematical morphological arguments, have also been proposed (Iyer and Sinha, 2006; Landstrom and Thurley, 2012). In general, the implementation of these methods involves the following three steps: (1) contrast enhancement, (2) mathematical morphological processing, and (3) information extraction using linear filters. Other machine learning methods, such as artificial neural networks (ANN) (Adeli and Yeh, 1989; Eldin and Senouci, 1995; Jin and Zhou, 2014), support vector machine (SVM) (Qu et al., 2010), Adaboost (Cord and Chambon, 2012),

*To whom correspondence should be addressed. E-mail: *yadongxue@tongji.edu.cn*.

© 2018 *Computer-Aided Civil and Infrastructure Engineering*.
DOI: 10.1111/mice.12367

K-nearest neighbors algorithm (Lei and Zuo, 2009), grouping techniques (Yeum and Dyke, 2015) and Restricted Boltzmann Machine (Rafiei and Adeli, 2016, 2017; Rafiei et al., 2017) have also been used in the field of civil engineering for crack or damage detection and achieved some good results. However, a common problem with these methods is the inability to handle complex background images. The image background of subway tunnel lining is often very complicated. The advent of state-of-the-art deep learning techniques bears the promise to change this situation.

Deep learning has recently been at the forefront of machine learning and computer sciences, especially computer version. Although some of the methods were conducted in the 1980s and 1990s (convolutional neural network [CNN], LeCun and Bengio, 1995; LeCun et al., 1998; the prototype of Recurrent Neural Networks; Hopfield, 1982), a breakthrough in the application of these methods to computationally demanding problems were achieved only in 2006 (Hinton and Salakhutdinov, 2006). In general, deep learning refers to the approach in which many layers of nonlinear processing units are combined sequentially for feature extraction and transformation (LeCun et al., 2015). In the field of image processing, this approach has been used for image classification and object detection (including image segmentation). A classical problem in image processing is that of determining whether or not the image data contains some specific objects or features. While object detection amounts to identifying and locating specific objects, given a complete image. CNN has been widely used to solve problems of this sort. There is an authoritative competition named ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), which evaluates algorithms for object detection and image classification on big data. Many excellent classification models such as AlexNet (Krizhevsky et al., 2012), ZF (Zeiler and Fergus, 2014), VGG (Simonyan and Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), and ResNet (He et al., 2015) have been proposed and validated on ILSVRC series.

Recent state-of-the-art deep learning methods, such as R-CNN series (R-CNN, Girshick et al., 2014; Fast R-CNN, Girshick, 2015; Faster R-CNN, Ren et al., 2015), DarkNet (Redmon et al., 2016), SSD (Liu et al., 2016), and R-FCN (Dai et al., 2016), have demonstrated good performance on object detection tasks such as those in Pascal VOC challenge (Everingham et al., 2015).

The development of deep learning has led to technological progress in many other fields such as biology, agriculture, transportation and art, including traditional civil engineering. A deep CNN was used for detecting road cracks (Zhang et al., 2016). A classifier with a CNN model for detecting concrete cracks from images was built, which received an accurate result in the data set (Cha et al., 2017a). A novel structural damage detection approach using the deep CNN was proposed based on low-level waveform signals. (Lin et al., 2017). An efficient architecture of CNN (CrackNet) was proposed for 3D asphalt pavement surface pixel-wise classification (Zhang et al., 2017). Cha et al. (2017b) built a new model with region-based faster CNN (Faster R-CNN), which can be used to detect five different damage types including concrete crack, steel corrosion with two levels (medium and high), bolt corrosion, and steel delamination.

The subway tunnel lining images are usually affected by segment joints, pipelines, dust, illumination and other factors. The image scope contains the entire tunnel space, so the image background is complex and its size is very large. In this article, the main work is to explore a new way to process tunnel lining images more quickly, accurately, and intelligently.

The content of this article is structured as follows. Section 2 presents the data set with data sources. Section 3 introduces the methodology and technological process. Section 4 gives some model test results and comparative analysis. Section 5 gives some discussions and section 6 concludes this article.

2 DATA SETS

2.1 Data sources

The rise of deep learning is intimately related to the recent explosion in data (so-called big data). Many famous data sets, like ImageNet (Deng et al., 2009; Russakovsky et al., 2015) and Microsoft COCO (Lin et al., 2014), were conducted in this current trend of deep learning. However, there are no data sets that contain information about tunnel defects. Thus, it was crucial to collect a sufficient amount of data for this research. Data is “food” for deep learning. The more high-quality “food,” the better the performance of the model. In this article, an advanced Movable Tunnel Inspection (MTI) system was used to acquire tunnel lining surface images (Huang et al., 2017). The MTI system equipped with 6 high-resolution linear CCD cameras and 12 light-emitting diodes (LEDs) as sources of light, could scan over 13 meters of a tunnel’s surface at a time. Several detection tasks (Figure 1) were conducted in the Shanghai subway Nos. 1, 2, 4, 7, 8, 10, and 12. These tunnels were built in different years and the tunnel environment and lining disease conditions are different. Figure 2 shows example images acquired by the MTI-100.

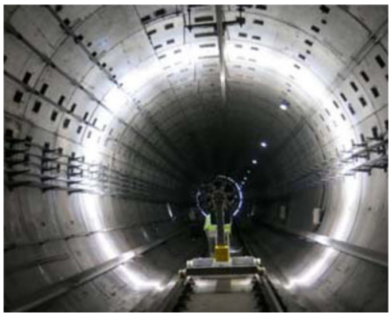


Fig. 1. MTI-100 subway tunnel inspection.

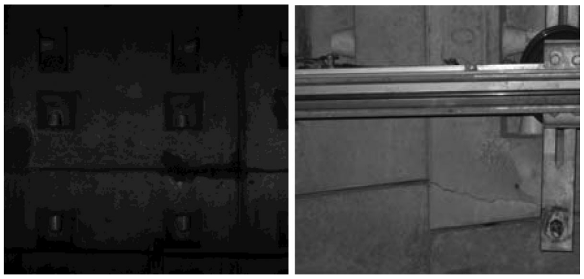


Fig. 2. Original lining image samples.

Table 1
Data sets for image classification

	Label	*Training set*	*Test set*	*Total*
Leakage	1	1,200	400	1,600
Crack	2	1,680	560	2,240
Segment joint	3	1,500	500	2,000
Pipeline	4	1,350	450	1,800
Lining	5	1,410	470	1,880
Sum		7,140	2,380	9,520

2.2 Data set for image classification

As a first step in defects detection, it is necessary to achieve an effective image classification of the defects. Consider the main defects and background interference of tunnel lining images, using five object categories including leakage, crack, segment joint, pipeline, and lining were defined. From the original inspection images, 9,520 images with 256 $\times$ 256 pixels were extracted. All these images were labeled manually, and each image contained only one ground truth label. The entire set of images was divided in two subsets: the training set and the test set. Table 1 lists the classification data set for this study.

However, there are some deficiencies in the images acquired by CCD camera. First, CCD cameras required uniform illumination, which could not be ensured by 12 LEDs; thus, acquired images suffered from uneven brightness. Second, inhomogeneity of concrete lining and distractors associated with tunnel operation significantly increase the complexity of surface images. Third, the focusing operation of CCD camera influence the sharpness of data samples. Lastly, uneven speed distorts some images. Although all of these factors leave significant marks on the images, they also make the data sets very diverse, which is advantageous for avoiding overfitting and for increasing the model's robustness. Figure 3 shows the very diverse of data in the data sets. For each dimension, examples are also shown along the range of that property in Figure 3.

2.3 Data set for object detection

Data for object detection were also derived from images that were acquired by the MTI system. For this data set, entire camera images were used, with each image containing 3,000 $\times$ 3,724 pixels. In total, the data set contained 4,139 annotated images. Three classes including crack, leakage, and scratch were labeled (Figure 4a). Scratch is a common noisy pattern in the surface of tunnel lining, which is caused by human collision. Scratches are not defect of lining, but they are very similar to the appearance of cracks, adding scratches to the training set was beneficial for improving the detection ability of the model. For each image, the ground truth with class and bounding box were annotated. To annotate the bounding box, a rectangular box that contained the coordinates of upper left and lower right corners was used. In this data set, image could contain more than one ground truth label. Figure 4b shows some examples of such labeling. The set of images was divided into two subsets: the training set (containing 3,000 images) and the test set (containing 1,139 images).

3 METHODOLOGY

The line-scan CCD camera of MTI-100 has 7,500 pixels on one line. The width of the scanning direction can be set as needed. Therefore, the images acquired using MTI-100 are very large. In order to improve the efficiency and accuracy of image processing, a two-stage solution is proposed: (1) image classification and (2) defect detection and localization. A framework was developed based on R-FCN (Dai et al., 2016). This method was achieved with two parts, the backbone architecture and the head architecture. The backbone architecture is a Fully Convolutional Network (FCN) model to compute the feature maps. A FCN is a CNN in which only the last layer is fully connected; this layer will be removed and replaced when fine-tuning the network for object detection. The FCN model was pretrained with the data set

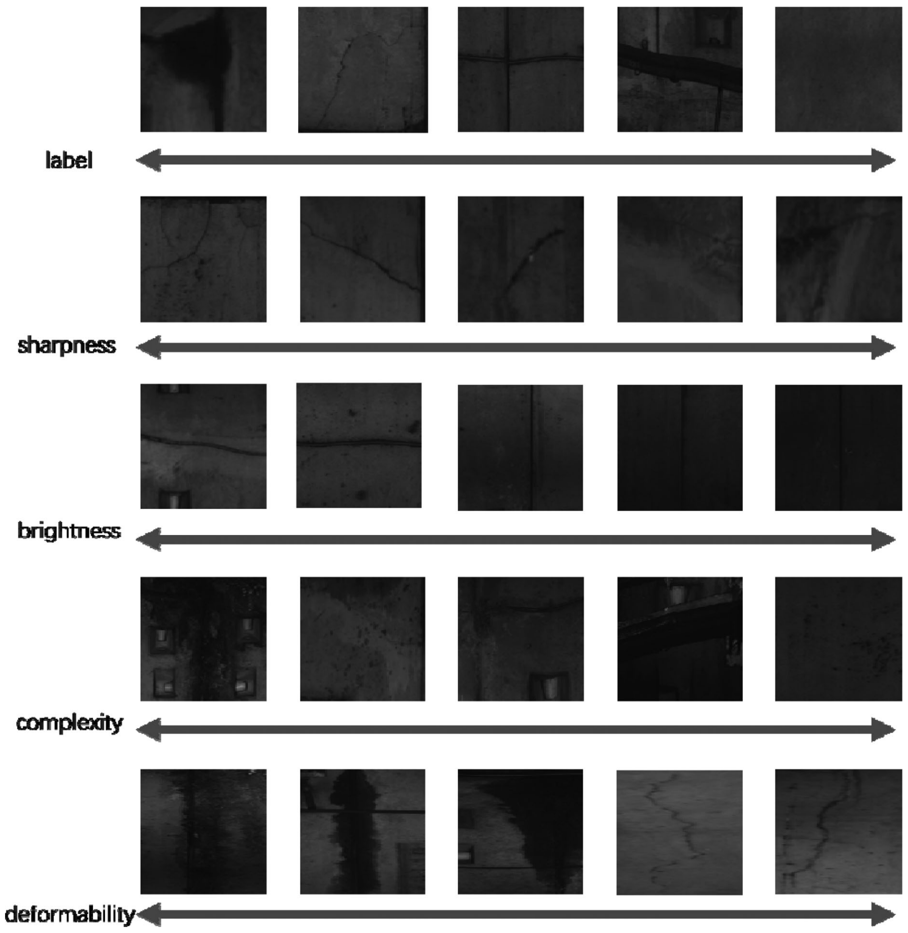


Fig. 3. The diversity of data in data set for image classification. Label shows five categories (leakage, crack, segment joint, pipeline, and lining) that were used to label the data set. Sharpness shows different images range from properly focused to very improperly focused ones. Brightness shows different brightness owing to uneven illumination. Complexity shows complex background of urban subway shield tunnels. Deformability shows image distortion caused by uneven speed.

for image classification. The head architecture includes region proposal network (RPN), position-sensitive RoI (Region of Interest) pooling, softmax and bounding box regression. RPN was used to generate region proposals, which might correspond to objects in the full image. Position-sensitive RoI pooling was used to explicitly encode positional information, and finally two loss functions (softmax and bounding box regression) were used to probabilistically determine the category of the defect and to precisely determine the location of the defect. Figure 5 schematically demonstrates the proposed method.

3.1 Backbone architecture

Compared with original neural networks, CNNs stand out because they have many hidden layers. The convolution layer and the pooling layer are the most common ones.

3.1.1 Convolution layer and pooling layer. The structure of the convolution layer was originally motivated by receptive fields in the visual system (Hubel and Wiesel, 1968). As opposed to full inter-layer connectivity in classical neural networks, the units in the convolution layer are sparsely connected (sometimes it is said that the units are locally connected). Because images are characterized by locality and invariance, the number of parameters can be significantly reduced by employing such connectivity. In a typical convolution layer, there are n kernels (each covering $l \times l$ units) convoluted with the $w \times h \times b$ feature map and slid over with the stride of s. Thus, the layer's output has the size of $(h - l)/(s + 1) \times (w - l)/(\mathrm{s} + 1) \times n$. Similar to classical neural networks, units in the convolution layer utilize a predefined activation function. Each convolution layer can be understood to perform the task of feature extraction on the image (a filter).

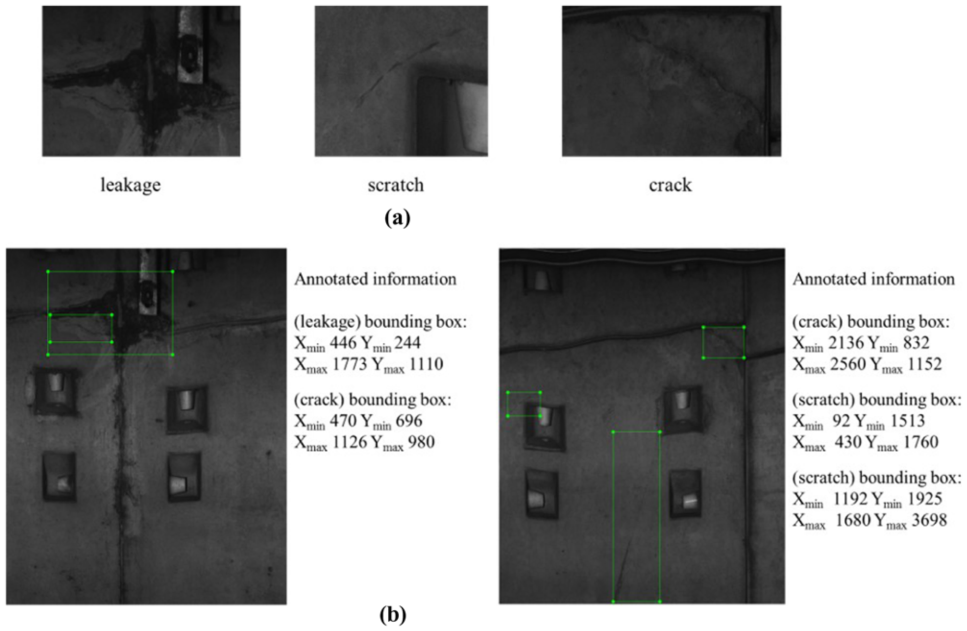

leakage
scratch
crack
(a)
Annotated information
(leakage) bounding box:
X_{min} 446 Y_{min} 244
X_{max} 1773 Y_{max} 1110
(crack) bounding box:
X_{min} 470 Y_{min} 696
X_{max} 1126 Y_{max} 980
Annotated information
(crack) bounding box:
X_{min} 2136 Y_{min} 832
X_{max} 2560 Y_{max} 1152
(scratch) bounding box:
X_{min} 92 Y_{min} 1513
X_{max} 430 Y_{max} 1760
(scratch) bounding box:
X_{min} 1192 Y_{min} 1925
X_{max} 1680 Y_{max} 3698
(b)

Fig. 4. Examples of labeling for object detection.

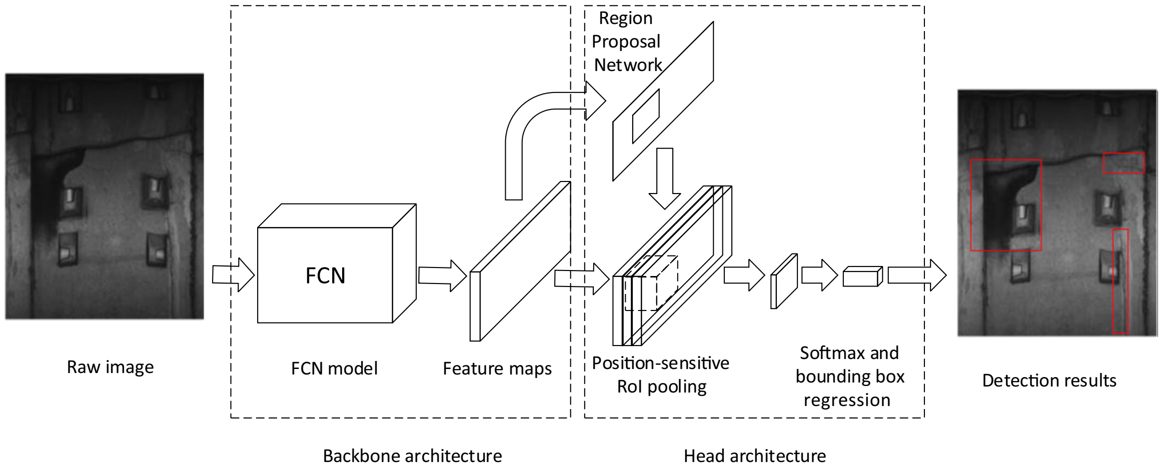
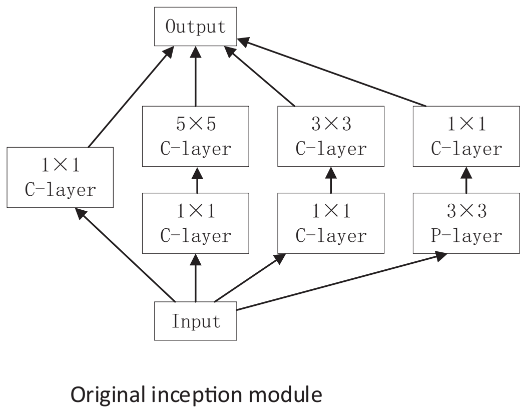


Fig. 5. Schematic of the proposed method.

The pooling layer is also called the subsampling layer, and usually follows the convolution layer. In the pooling layers, there are no weights and activation functions in the units. Two computational approaches, mean pooling and max pooling, can be used on the pooling layer. Mean pooling refers to calculating the average value while max pooling refers to calculating the maximum value. Pooling layers are used to reduce the number of parameters in the model, and to achieve some special functions, such as translational invariance, scaling and distortion, reducing overfitting, etc.

3.1.2 Model in detail. Inspired by GoogLeNet and VGG, the FCN model was designed. The well-known advantages of GoogLeNet are its inception module and fully convolutional structure, while the advantage of VGG is that it uses small convolution kernels in the best-set model. Thus, the inception module and the 7 × 7 convolutional layer in the first layer of GoogLeNet were improved. For the inception module, two 3 × 3 convolutional kernels were used to replace the 5 × 5 one (shown in Figure 6). As for the 7 × 7 convolutional layer, four 3 × 3 convolutional layers were used to replace it. Table 2 lists the architecture characteristics of



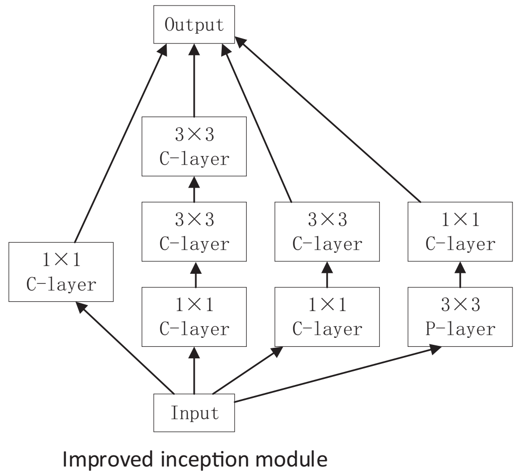


Fig. 6. Original inception module and improved inception module. Here, C denotes a convolutional layer and P denotes a pooling layer. The input feature map is calculated by four branches and then concatenated in the output.

Table 2
Characteristics of the model's architecture

Type	*Size/stride*	*Output size*	*Depth*
Convolution	3×3/1	224 × 224 × 32	1
Convolution	3×3/1	224 × 224 × 32	1
Max pooling	3×3/2	112 × 112 × 32	0
Convolution	3×3/1	112 × 112 × 64	1
Convolution	3×3/1	112 × 112 × 64	1
Max pooling	3×3/2	56 × 56 × 64	0
Convolution	3×3/1	56 × 56 × 198	1
Max pooling	3×3/2	28 × 28 × 198	0
Improved inception		28 × 28 × 256	3
Improved inception		28 × 28 × 256	3
Max pooling	3×3/2	14 × 14 × 480	0
Improved inception		14 × 14 × 512	3
Improved inception		14 × 14 × 512	3
Improved inception		14 × 14 × 528	3
Improved inception		14 × 14 × 832	3
Max pooling	3×3/2	7 × 7 × 832	0
Improved inception		7 × 7 × 832	3
Improved inception		7 × 7 × 1,024	3
Avg pooling	7×7/1	1 × 1 × 1,024	0
Dropout		1 × 1 × 1,024	0
Fully connected		1 × 1 × 1,000	1
Softmax		1 × 1 × 5	0

the proposed model (the "depth" denotes the number of the layers with trainable parameters).

3.2 Region proposal network

Region proposal is a region in which an object can be potentially located. In large-scale image processing, generation of region proposals is an important step toward detection of defects. These regions are usually extracted using methods such as selective search (SS) (Uijlings et al., 2013) and edge boxes (EB) (Zitnick and Dollár, 2014), to name a few. However, these approaches cannot be incorporated into the end-to-end deep learning model; in addition, they are time-consuming. A RPN was proposed in 2015 (Ren et al., 2015). This network can accept images of any size as input, and it outputs a set of proposed rectangular objects. The procedure is shown schematically in Figure 7. An anchor box is selected as a template of the object's region, and it is slid over the feature map in the FCN model. The anchor box is defined in terms of two parameters: the aspect ratio and the scale. In this article, three scales for the anchor box were considered. A faster R-CNN used an anchor box with the ratios of 1:2, 1:1, and 2:1 (Ren et al., 2015). As cracks are usually quite elongated, the aspect ratios of 1:3 and 3:1 were also considered. There are two stages in the RPN. In the training stage, the predicted bounding box is calculated by a two-class classification layer and a bounding box regression layer. The loss function here combines the location information and softmax loss. In the test stage, the predicted bounding box is calculated with an object score. Thus, by setting a threshold of object score, a certain number region proposal can be gotten. The region with a high score will be computed in a RoI-wise subnet in Sections 3.3 and 3.4.

3.3 Position-sensitive RoI pooling

In image classification, fully convolutional networks are usually deep for translational invariance. However, in the object detection, translation variance is needed. A state-of-the-art method to overcome this dilemma is position-sensitive RoI pooling (Dai et al., 2016).

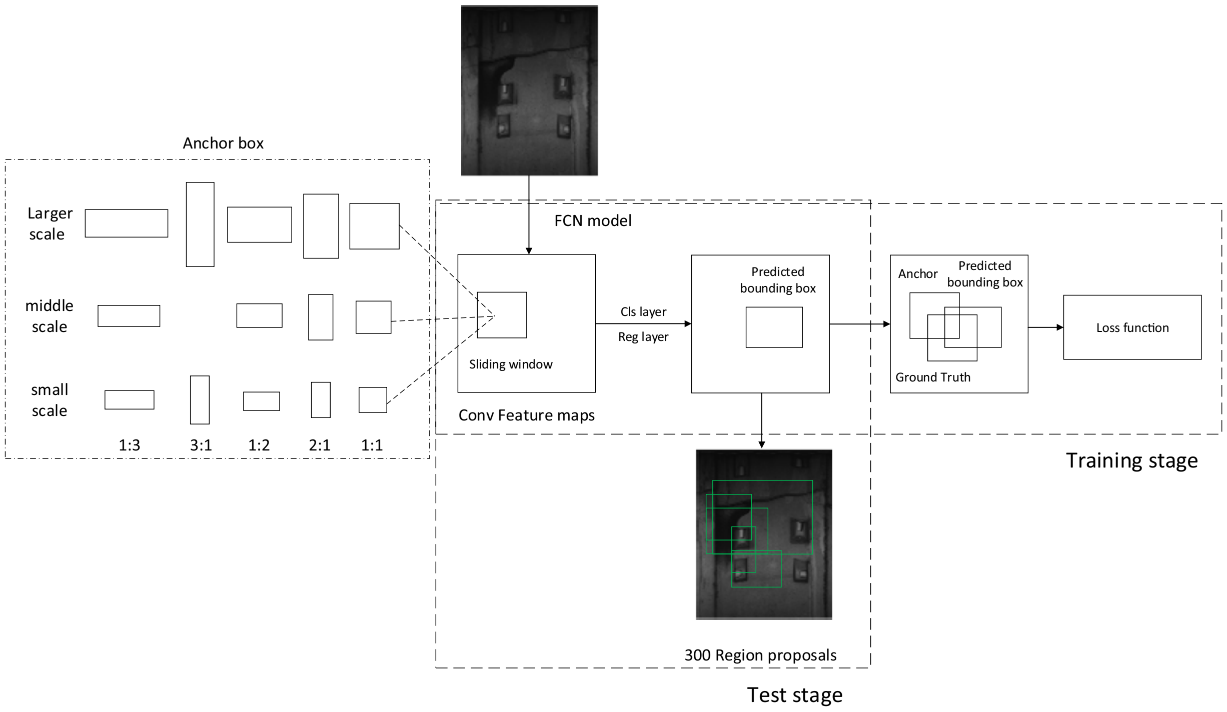


Fig. 7. Region proposal network.

Figure 8 shows a simple example of position-sensitive RoI pooling for softmax. Based on the feature map extracted by the FCN model, a convolution layer with n (Equation (1)) kernels was used to compute the position-sensitive feature map. The feature map was spread to k^2 ($k = 3$ in this example). For example, the first one contains top-left information about each object. Then, a RoI pooling layer (Girshick, 2015) (a single degree pyramid pooling; He et al., 2015) was used to divide each $w \times h$ RoI rectangle into $k \times k$ bins by a regular grid with the size of $w/k \times h/k$. Following Equation (2), an array of $k \times k \times n$ was obtained. Because each region has two corners, for the bounding box regression there would be $4n$ convolution kernels, yielding an array of $k \times k \times 4n$.

$$n = (c + 1) \times k^2 \tag{1}$$

Here, n is the number of kernels for softmax, c is the number of object categories, and k is the size of the score map.

$$r_l(i, j) = \sum_{(x,y)\in bin(i,j)} z_{i,j,c}(x_0 + x, y_0 + y)\big/ m \tag{2}$$

Here, $r_l(i, j)$ is the number in l dimensions in the (i, j)th bin, $z_{i,j,c}$ is one of n score maps, (x_0, y_0) is the top-left corner of an RoI, m is the number of pixels in the bin, and l is the category label, including background ($l = 0$).

In this article, to obtain accurate results, the size of the score map was set to 7, and three categories were considered. Thus, there were 196 convolution kernels for the softmax function and 784 kernels for the bounding box regression.

3.4 Softmax and bounding box regression

Softmax is a classifier for predicting and ranking the RoIs. Equations (3) and (4) define the output of softmax. The loss function of softmax is the cross-entropy loss and is defined in Equation (5).

$$r_l = \sum\nolimits_{i,j} r_l\,(i, j) \tag{3}$$

$$s_l = \frac{e^{r_l}}{\sum_{l\cdot=0}^{l} e^{r_{l\cdot}}} \tag{4}$$

$$L_{cls}(s, l) = -\log s_l \tag{5}$$

In the above, r_l is an l-dimensional vector for each RoI, s_l is an l-dimensional vector of a discrete probability distribution over the different categories. $L_{cls}(s,l)$ is the classification loss function.

The bounding box and its output were computed using Equation (3). Because there are $4n$ kernels, the output is a predicted tuple $t^l = (t^l_x, t^l_y, t^l_w, t^l_h)$, and the ground truth regression

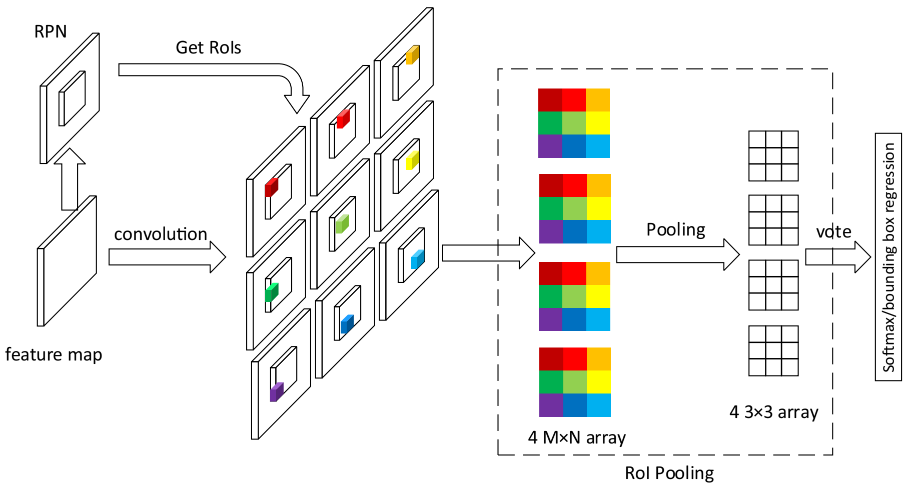


Fig. 8. Example of position-sensitive RoI pooling.

target tuple is $v^l = (v^l_x, v^l_y, v^l_w, v^l_h)$. Equation (6) defines the loss function of the bounding box regression.

$$L_{bbreg}(t^l, v^l) = \sum_{i\in\{x,y,w,h\}} \text{smooth}_{L_1}(t^l_i - v^l_i) \qquad (6)$$

in which $\text{smooth}_{L_1}(x)=\begin{cases} 0.5x^2 & \text{if } |x|<1 \\ |x|-0.5 & \text{otherwise} \end{cases}$.

In this article, a multitask loss L on each RoI was used for classification and bounding box regression, as shown in Equation (8):

$$L(s, l, t^l, v) = L_{cls}(s, l) + \lambda\,[l > 0]\, L_{bbreg}(t^l, v^l) \quad (7)$$

in which $\lambda = 1$ when $l > 0$.

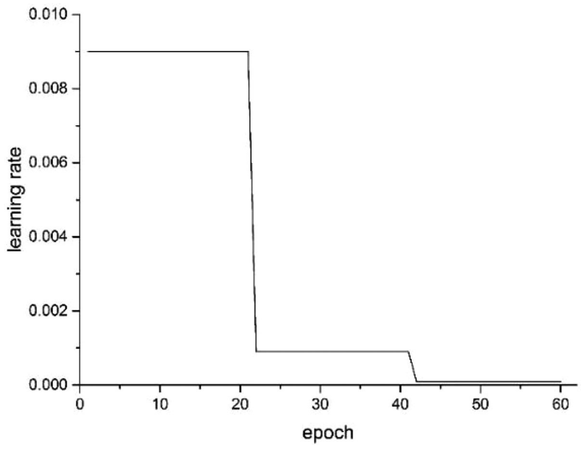


Fig. 9. Learning rate curve of model training.

4 EXPERIMENTS

The experiments were conducted on a computer with one Intel Core i7-5820K CPU, 64 GB Random Access Memory and three GeForce GTX 1080 GPU (24GB graphics memory). The proposed method was implemented based on the deep learning framework Caffe (Jia et al., 2014). The calculation software environment was set with python 2.7.12, CUDA 8.0 and cuDNN5.0.

The method consists of two parts: one is to use small-size images (256 × 256 pixels) to train the classification function of the network; the other is to use large-size images (3,000 × 3,724 pixels) to train the detection and localization function of the network.

4.1 Experiments on feature map classification

The preparation of the samples, especially the naked eye identification and labelling, is tedious and meticulous. Although there are nearly 10,000 samples for classification (Table 1), it is still not enough for the training of deep learning models. To achieve the robustness and avoid overfitting, k-fold validation ($k = 5$) and data augmentation (shift, shear and zoom, no rotation) were used in the training and validation process.

4.1.1 Comparisons with AlexNet and GoogLeNet. The proposed model was compared with the top-performance models in ISVRC2012 (AlexNet) and ISVRC2014 (GoogLeNet). To evaluate the accuracy of the proposed model, the same hyperparameters were used for training. Nesterov's optimal gradient method (Sutskever et al., 2013) was used to train the model with the minibatch size of 40 in training and 20 in test. Figure 9 shows the learning rate during model training.

Figure 10 shows the training accuracy curves of the three networks. Table 3 compares the performance in terms of accuracy and test time. For AlexNet, the maximal accuracy was 91.43% and was obtained at the

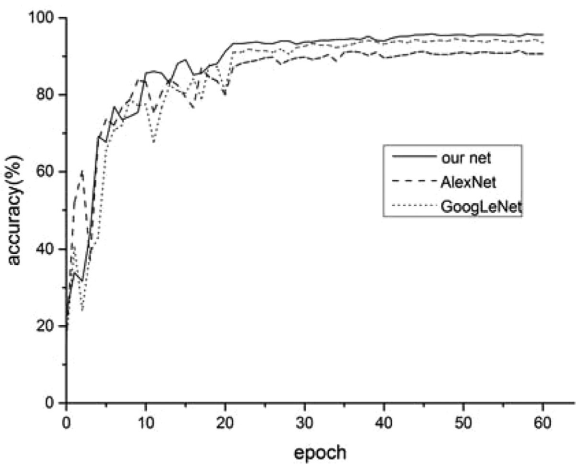


Fig. 10. Accuracy curve of model training.

Table 3
Performance of networks

	Accuracy	*Test time (per image) (average time for all operations)*
AlexNet	91.43%	46 ms
GoogLeNet	94.58%	50 ms
Our net	95.84%	48 ms

57th epoch, at a test time of 46 ms. For GoogLeNet, the maximal accuracy was 94.58% at the 49th epoch, at a test time of 50 ms. For the proposed model, the accuracy was 95.84% and was obtained at the 58th epoch, at a test time of 48 ms.

Confusion matrices were constructed to analyze the number of correct classifications for each image category. The rows and columns of these confusion matrices represent the labels of different categories. The *i*th value in the *j*th column indicates the times the image with the ground truth label *i* was classified by the model into *j*th category. The diagonal elements in these confusion matrices report correct classifications. Tables 4–6 show the confusion matrix for AlexNet, GoogLeNet, and the proposed model.

4.1.2 Comparisons with VGG16. The model was compared with another model that performed exceptionally in ISVRC2014—VGG16 model. Although the hyperparameters and some other parameters were extensively varied in VGG16 model, this model did not exhibit convergence when applied to this data set (Table 1). In the training phase, the performance of VGG was very slow. The reason may be due to too many parameters in VGG16, which makes it difficult to train. For users and researchers who do not have much experience with parametric tuning of CNNs, the complexity of training can become a big obstacle. Thus, for engineering applications the proposed model appears to be more suitable.

Table 4
Confusion matrix for the test set, for AlexNet

Label	*Leakage*	*Crack*	*Joint*	*Pipeline*	*Lining*	*Accuracy*
Leakage	352	18	11	3	16	88.00%
Crack	13	505	6	3	33	90.18%
Segment joint	10	28	443	13	6	88.60%
Pipeline	4	6	6	432	2	96.00%
Lining	1	18	3	4	444	94.46%

Table 5
Confusion matrix for the test set, for GoogLeNet

Label	*Leakage*	*Crack*	*Joint*	*Pipeline*	*Lining*	*Accuracy*
Leakage	389	4	1	4	2	97.25%
Crack	9	541	7	1	2	96.61%
Segment joint	13	8	459	13	7	91.80%
Pipeline	15	7	6	419	3	93.11%
Lining	1	21	3	2	443	94.26%

Table 6
Confusion matrix for the test set, for the proposed model

Label	*Leakage*	*Crack*	*Joint*	*Pipeline*	*Lining*	*Accuracy*
Leakage	391	5	1	3	0	97.75%
Crack	4	555	1	0	0	99.11%
Segment joint	7	14	464	12	3	92.80%
Pipeline	6	6	12	425	1	94.44%
Lining	7	14	3	0	446	94.89%

4.1.3 Classification results. A fraction of images was randomly drawn from the test set, and predictions of the proposed model on these images are shown in Figure 11.

4.2 Experiments on defect detection

Combining the RPN and position-sensitive RoI pooling, the method for detecting tunnel lining defects was set up. With a pretrained FCN model, the learning rate was set as 0.001 for first 10k iterations and 0.0001 for next 20k iterations. Approximate joint training was used to train this end-to-end model, which could train RPN and the detection method at the same time.

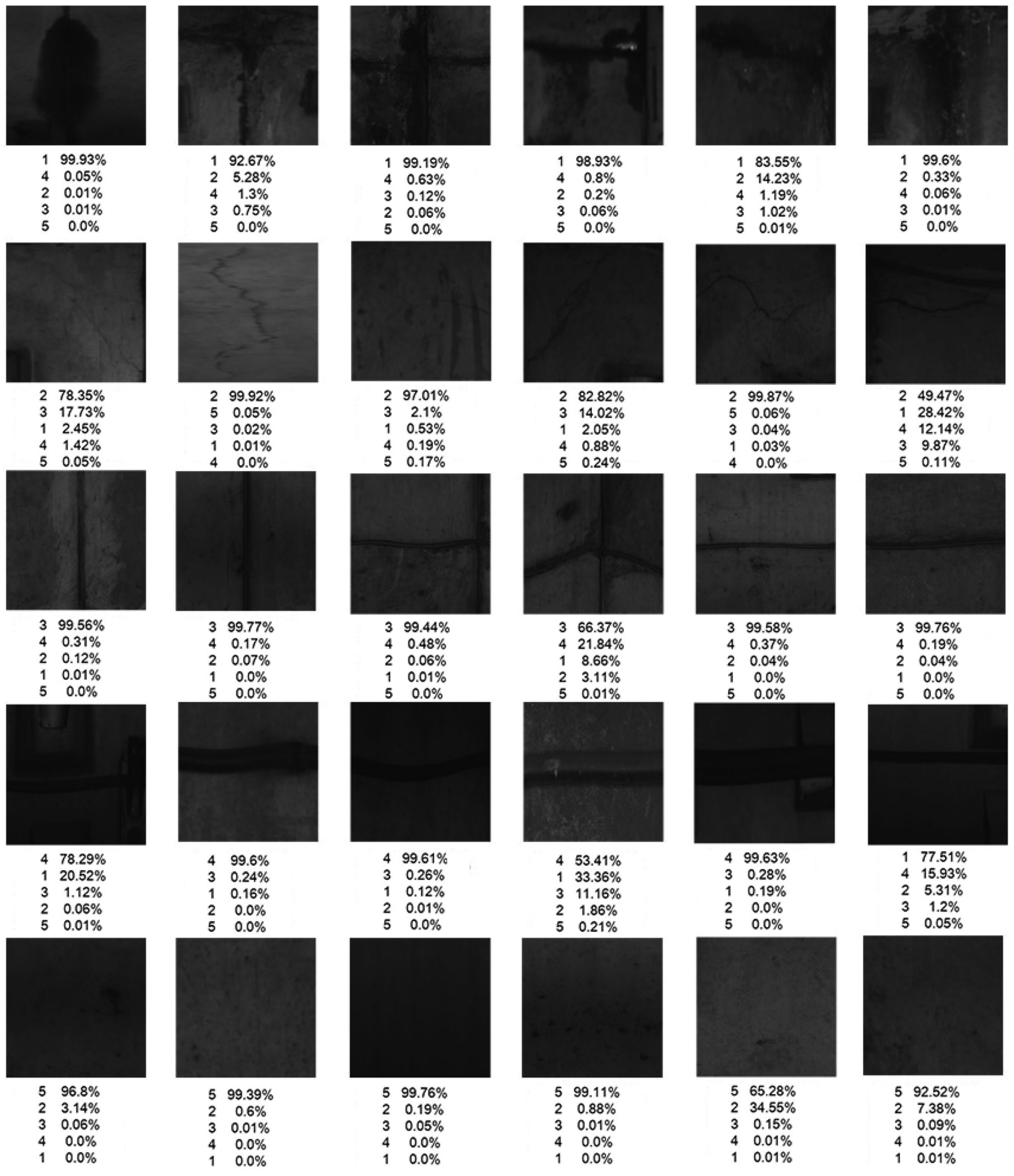


Fig. 11. Some classification results of the proposed model.

4.2.1 Detection performance accuracy. The performance on detection task was evaluated in terms of three indicators:

- Detection rate: the ratio of the correct detection number to the number of defects labeled in the test set.
- Detection accuracy: the average probability for each type of detected region of a certain category.
- Detection efficiency: the average detection speed of the test data set images.

4.2.2 Locating accuracy. Accurately locating the defect is crucial to the safety of the tunnel lining structure.

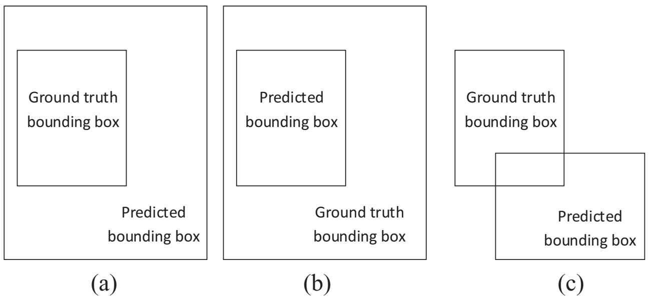

Ground truth bounding box
Predicted bounding box
(a)
Predicted bounding box
Ground truth bounding box
(b)
Ground truth bounding box
Predicted bounding box
(c)

Fig. 12. Three scenarios of the location.

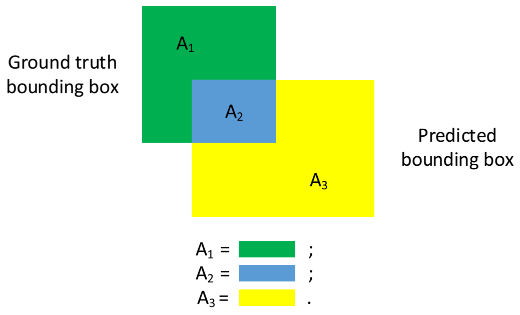


Fig. 13. Diagram of location accuracy.

The accuracy of locating is evaluated by establishing quantitative indicators. Generally, there are three scenarios (Figure 12) about the result of ground truth and prediction. The Figure 12a shows that although the predicted bounding box contains the whole defect (ground truth bounding), the boundaries are too broad. The Figure 12b is the opposite of Figure 12a. Figure 12c is between 12a and 12b. To evaluate the accuracy of bounding box, two indexes were defined from the position relationship between the predicted bounding box and the ground truth. The green box in Figure 13 refers to the ground truth, the yellow box refers to the predicted bounding box, and the blue box refers to the overlapping area. The index η_1 refers to the ratio of the overlapping area to the ground truth area (Equation (8)). The greater the value of η_1, the more accurate the defect location. The index η_2 refers to the ratio of the superfluous area to the predicted bounding box area (Equation (9)). The smaller the value of η_2, the more accurate the defect location.

$$\eta_1 = \frac{A_2}{A_1 + A_2} \tag{8}$$

$$\eta_2 = \frac{A_3}{A_2 + A_3} \tag{9}$$

The trained model was tested on the test set that contains 1,139 images. There are 1,867 defects in the test data set images. The detection rate was 94.4%, the detection accuracy was 85.6%, and the detection efficiency was 0.266 seconds per image. To the locating accuracy, average η_1 is 0.874, and η_2 is 0.062. Figure 14 shows some defect detection results.

4.2.3 Comparisons with the traditional method and Faster R-CNN. Image processing based on traditional image algorithms has many successful application cases. In the early stage of tunnel defect image processing, the authors also adopted traditional method and programmed software. A typical traditional detection method combines the HOG feature and SVM classifier (Dalal and Triggs, 2005). The definition of HOG is locally normalized histogram of gradient orientation in dense overlapping grids; it is a local region descriptor. With deep learning algorithm, Faster R-CNN (Ren et al., 2015) got very good performance in object detection task. These two models were tested with the data sets, and the results are shown in Table 7. These results show the advantages of deep learning methods comparing to the traditional one. Traditional methods rely on manual designed features (means feature engineering). Because of complex lining background and photos noises, the HOG feature didn't perform very well and the traditional method didn't receive a good detection rate; however, the two deep learning methods got very high detection rates. Moreover, the detection efficiency of traditional method is very slow compared to the deep learning methods.

By comparing with Faster R-CNN, it was found that the two methods got similar results in the detection rate and detection accuracy. For the detection efficiency, the proposed method was about 33% higher than Faster R-CNN. The locating accuracy of the method was also a little better than Faster R-CNN for the position-sensitive RoI pooling and the score map.

5 DISCUSSION

In this section, some important features of the proposed model, including robustness, adaptability, sensitivity, etc. will be discussed in detail.

5.1 Robustness and adaptability

The robustness and adaptability of the method were evaluated with respect to the translation of images, variation in the image scale, blurring and deformation of images (Koziarski and Cyganek, 2017).

- **Image translation**. Figure 15 shows the same defects at different locations in the image. The image size is 3,000 × 3,724 pixels. The confidence was quite high

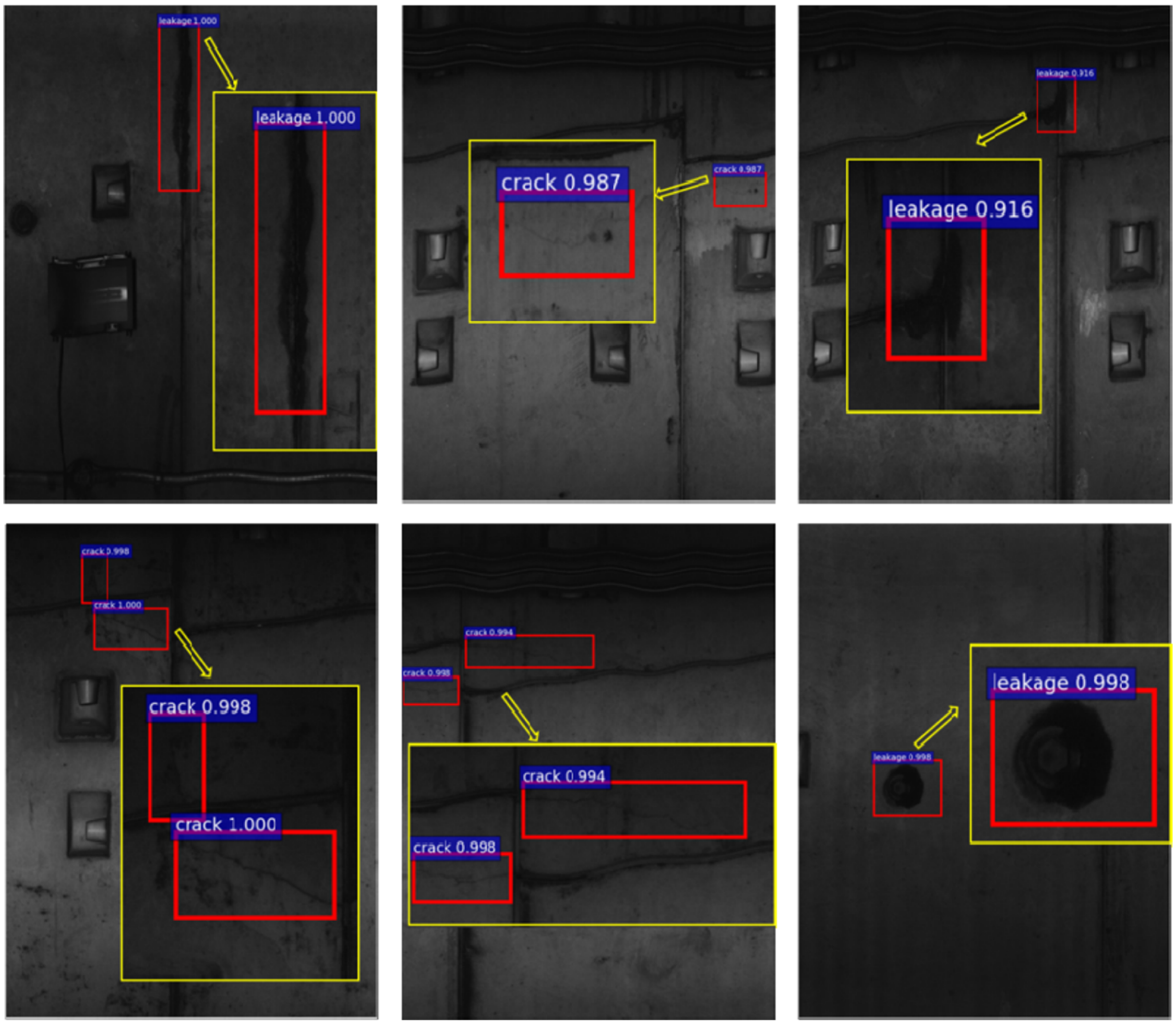


Fig. 14. Some detection results of the proposed model.

Table 7
Comparison of the results with the traditional method and Faster R-CNN

Method	*Detection rate*	*Detection accuracy*	*Detection efficiency*	η_1	η_2
Proposed method	94.4%	86.6%	0.266 s	0.874	0.062
Faster R-CNN	94.0%	87.3%	0.396 s	0.803	0.125
Traditional method	49.5%		63.340 s	0.646	0.227

for the two cracks. Regardless of translations, defects were always detected.

- **Image scale variations.** Figure 16 shows different scales for the same defects (from left to right: 3,000 × 3,724 pixels, 1,700 × 2,000 pixels, and 1,200 × 1,450 pixels). All of the defects were detected with a very high confidence, except for the rightmost image, for which the location seems to be inaccurate a little.
- **Image blurring.** Figure 17 shows images that were processed using Gaussian blurring. From left to right, the blurring radii were 0, 5, and 8. Each of these images features two cracks, one crack is clearly visible while the other is not very clear because its width is very small. Although the image in the middle is already quite blurred, two cracks were still detected by the model. However, for the rightmost image, the network detected only one crack.

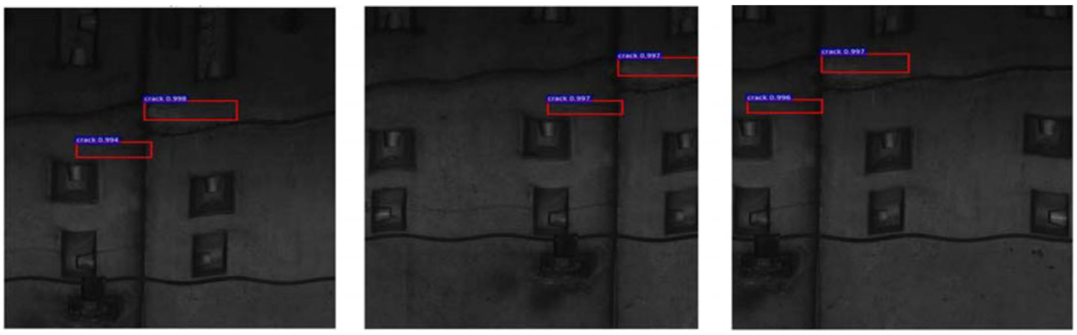


Fig. 15. Robustness with respect to image translation. Due to the shaking of the MTI, the segment joints in the image appear as wavy lines (straight lines in fact), but this still does not affect the detection effect of the model.

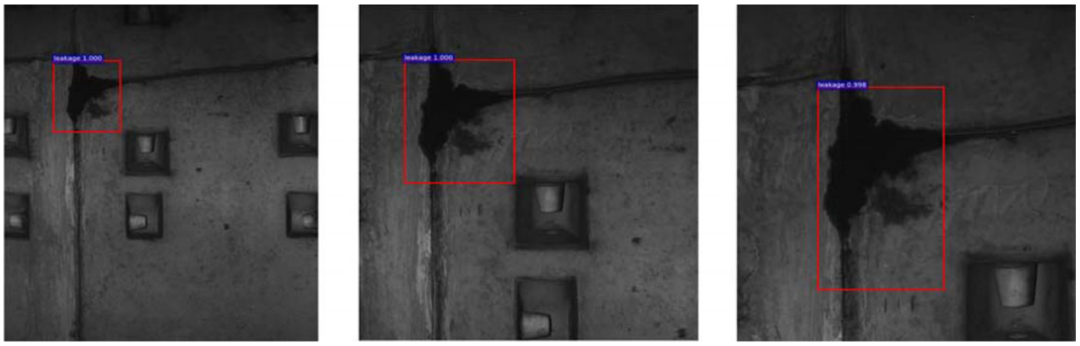


Fig. 16. Robustness with respect to the defect image scale.

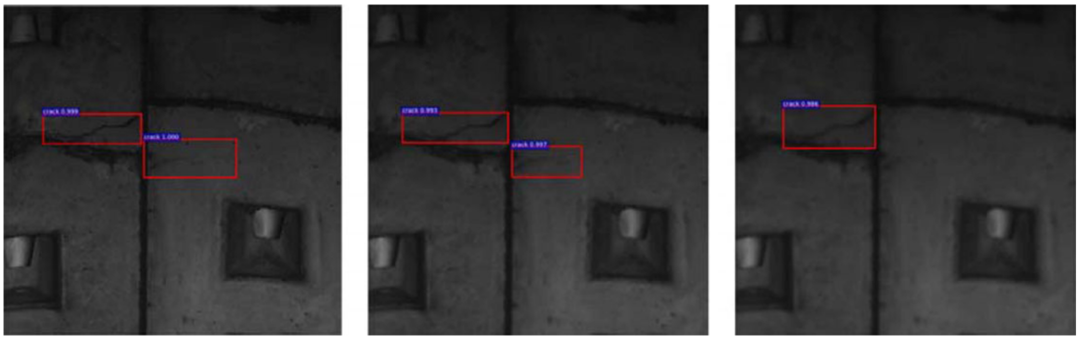


Fig. 17. Robustness with respect to image blurring.

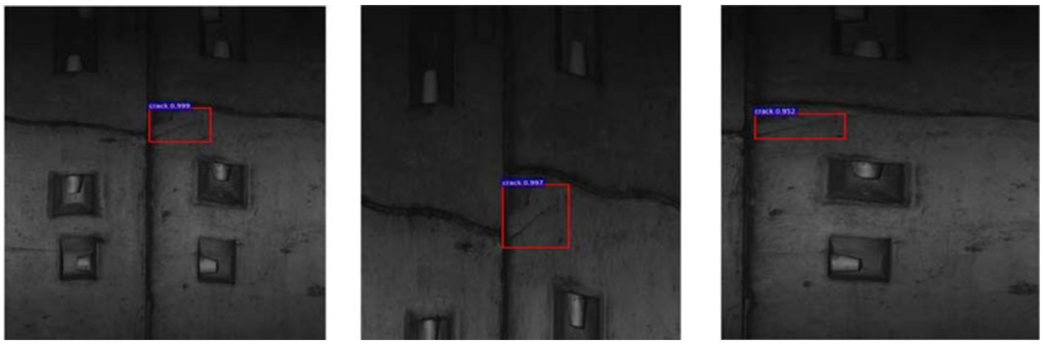


Fig. 18. Robustness with respect to image deformation.

Table 8
Properties and characteristics of crack

	Length		*Width*	
Crack	*Max*	*Min*	*Max*	*Min*
mm	479.66	76.85	2.03	0.29
pixel	1,654	265	7	1

Table 9
Properties and characteristics of leakage

	Area		*Morphologic ratio*	
Leakage	*Max*	*Min*	*Max*	*Min*
mm	443309.92	2109.31	113.40	0.29
pixel	5,235,552	25,081	3,711/277	255/874

- **Image deformation.** Figure 18 shows images that were deformed by uneven stretching. The leftmost image shows the original image. The middle and the rightmost images show the original image after application of horizontal (middle) and vertical (rightmost) stretching. The confidence is still quite high, and the locations of cracks are also quite accurately detected.

These results suggest that the proposed network is characterized by good robustness and adaptability. In certain scenarios, the defects are nearly invisible to the naked eye, but the model is still able to detect them.

5.2 Sensitivity of the method

Traditional methods of object detection rely on manual designed features, thus its sensitivity is subject to the designed feature threshold (Laefer et al., 2014), while the sensitivity of deep learning methods is associated with the training samples. In other words, if there are some defects with certain physical properties and characteristics in the training set, the method can detect similar defects in application. In this article, the physical properties and characteristics for crack and leakage in the data set are shown in Tables 8 and 9. The length and width of crack and the area and morphologic ratio of leakage are listed. The morphologic ratio refers to the

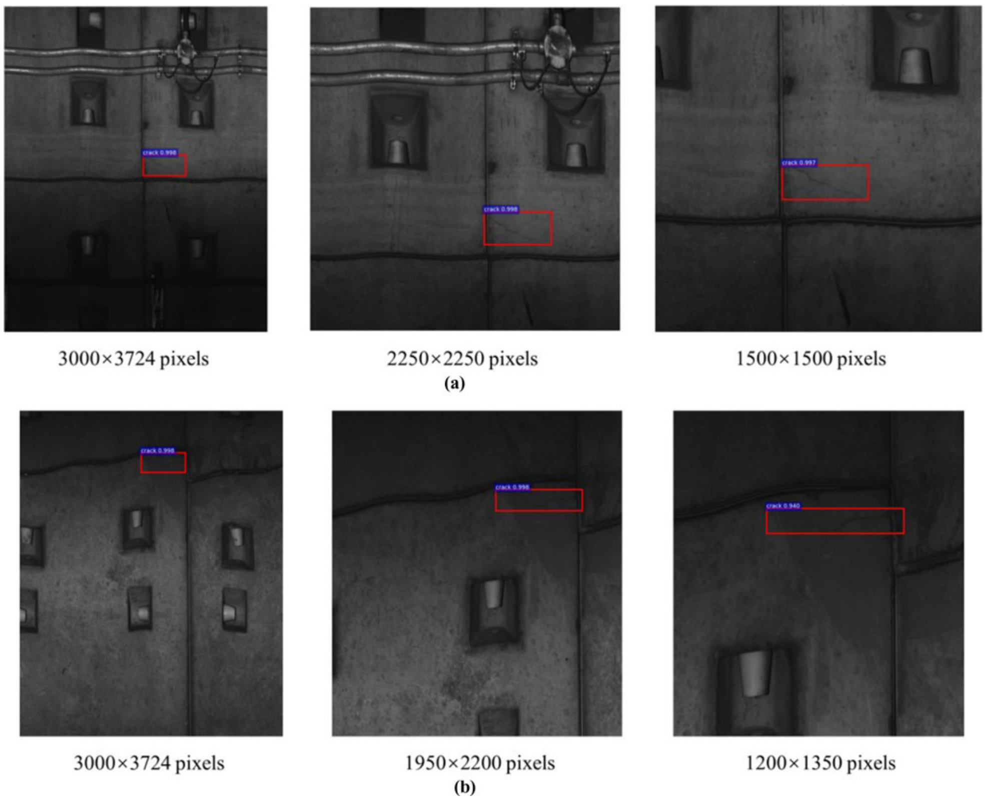


Fig. 19. Location accuracy of different image scale with small objects.

ratio of the maximum horizontal pixel number to the maximum vertical pixel number of a leakage area.

5.3 Effect of image size on small object location

The images acquired by the line-scan camera are continuous and can be divided into images of different sizes as required. For the same size of defect, the effect of different image size was tested. The results show that the detection accuracy was almost similar, while the location accuracy trended to be more accurate with smaller image size (the ratio of defect to image size larger). Figure 19 shows two examples. In the Figure 19a, η_1 was 0.863, 0.925, 0.964 from left to right; η_2 was 0 all. In the Figure 19b, η_1 was 0.816, 0.894, 0.912 from left to right; η_2 was 0.063, 0, 0 correspondingly. This provides some hints for ways to improve defect detection and location performance in the future.

6 CONCLUSIONS

The detection of urban shield tunnel lining will get a huge amount of images. In order to efficiently, accurately, and intelligently process the images, a two-step solution based on deep learning is proposed in this article: (1) considering that most of the lining images are defect-free, therefore a FCN network model is first established to classify images quickly and (2) for the images containing defects, an image processing model based on R-FCN is established to detect and locate the defects.

In this article, two data sets were built based on the images that were acquired using the MTI-100 with six high-resolution linear CCD cameras. An improved FCN was proposed and designed, and its efficacy, accuracy, and difficulty of training were evaluated and compared with those of GoogLeNet, AlexNet, and VGG. The proposed network demonstrated the accuracy of 95.84%, which was higher than AlexNet (91.43%), GoogLeNet (94.58%), and VGG16 (no convergence) on the same data set. The test time of the proposed model was 48 ms, compared with 50 ms (GoogLeNet) and 46 ms (AlexNet) for other networks.

The feature map in the deep layers was computed using the trained FCN model. Based on the RPN and position-sensitive RoI pooling, a method for detecting tunnel lining defects was set up. The performance of the proposed method was evaluated on large-scale images, each containing $3{,}000 \times 3{,}724$ pixels. The detection rate was 94.4%, the accuracy was 86.6%, and the efficiency was 0.266 seconds. The location accuracy η_1 is 0.874; η_2 is 0.062. Some comparsions were conducted with Faster R-CNN and the traditional method (HOG+SVM), and the results suggest that the method is accurate and efficient.

Although nearly 10,000 samples have been collected in this study, they are still not enough for the deep-learning models. In the past, the authors carried out a lot of highway tunnel inspection and got a large number of images of various types of defects. Considering the similar features of cracks and seepage in shield tunnel and road tunnel, it is planned to extend the images of road tunnel as samples to the model. In addition, MTI-100 is being used to get more shield tunnel lining images.

ACKNOWLEDGMENTS

The work presented in this article was supported by the Science and Technology Commission of Shanghai Municipality (Grant No. 16DZ1200402), the National Natural Science Foundation of China (Grant No. 41072206), and the Fundamental Research Funds for the Central Universities (Grant No. 0200219209, 0250219107).

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. & Yeh, C. (1989), Perceptron learning in engineering design, *Microcomputers in Civil Engineering*, **4**(4), 247–56.

Cha, Y. J., Choi, W. & Buyukozturk, O. (2017a), Deep learning-based crack damage detection using convolutional neural network, *Computer-Aided Civil and Infrastructure Engineering*, **32**(3), 2013–14.

Cha, Y. J., Choi, W., Suh G., Mahmoudkhani, S. & Buyukozturk, O. (2017b), Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, *Computer-Aided Civil and Infrastructure Engineering*, https://doi.org/10.1111/mice.12334.

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.

Dai, J., Li, Y., He, K. & Sun, J. (2016), R-FCN: object detection via region-based fully convolutional networks, arXiv, 1605, 06409, 1–11.

Dalal, N. & Triggs, B. (2005), Histograms of oriented gradients for human detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2005*, IEEE, San Diego, pp. 886–93.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009), Imagenet: a large-scale hierarchical image database, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2009*, IEEE, Miami, pp. 248–55.

Eldin, N. N. & Senouci, A. B. (1995), A pavement condition-rating model using backpropagation neural networks, *Computer-Aided Civil and Infrastructure Engineering*, **10**(6), 433–41.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J. & Zisserman, A. (2015), The pascal visual object classes challenge: a retrospective, *International Journal of Computer Vision*, **111**(1), 98–136.

Girshick, R. (2015), Fast R-CNN, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015*, IEEE, Boston, pp. 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 2014*, IEEE, Columbus, pp. 580–87.

Gonzalez, R. C. & Woods, R. E. (2007), *Digital Image Processing* (3rd Edition), Pearson Prentice Hall, Upper Saddle River, NJ.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(9), 1904–16.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*, IEEE, Las Vegas, pp. 770–8.

Hinton, G. E. & Salakhutdinov, R. R. (2006), Reducing the dimensionality of data with neural networks, *Science*, **313**(5786), 504–07.

Hopfield, J. J. (1982), Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, **79**(8), 2554–58.

Huang, H. W., Sun, Y., Xue, Y. D. & Wang, F. (2017), Inspection equipment study for subway tunnel defects by greyscale image processing, *Advanced Engineering Informatics*, **32**, 188–201.

Hubel, D. H. & Wiesel, T. N. (1968), Receptive fields and functional architecture of monkey striate cortex, *The Journal of Physiology*, **195**(1), 215–43.

Iyer, S. & Sinha, S. K. (2006), Segmentation of pipe images for crack detection in buried sewers, *Computer-Aided Civil and Infrastructure Engineering*, **21**(6), 395–410.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. & Darrell, T. (2014), Caffe: convolutional architecture for fast feature embedding, in *Proceedings of the 22nd ACM International Conference on Multimedia*, ACM, Orlando, pp. 675–78.

Jin, T. & Zhou, Z. Y. (2014), Leakage detection method for piping network based on BP neural network, *Applied Mechanics and Materials*, **470**(2014), 738–42.

Koziarski, M. & Cyganek, B. (2017), Image recognition with deep neural networks in presence of noise—dealing with and taking advantage of distortions, *Integrated Computer-Aided Engineering*, **24**(4), 337–49.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in *Proceedings of the Advances in Neural Information Processing Systems 25*, NIPS, Stateline, pp. 1097–105.

Laefer, D., Truong-Hong, L., Carr, H. & Singh, M. (2014), Crack detection limits in unit based masonry with terrestrial laser scanning, *NDT & E International*, **62**, 66–76.

Landstrom, A. & Thurley, M. J. (2012), Morphology-based crack detection for steel slabs, *IEEE Journal of Selected Topics in Signal Processing*, **6**(7), 866–75.

LeCun, Y. & Bengio, Y. (1995), Convolutional networks for images, speech, and time series, in M. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, vol. 3361, MIT Press, Cambridge, MA, pp. 255–58.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), Deep learning, *Nature*, **521**(7553), 436–44.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86**(11), 2278–324.

Lei, Y. & Zuo, M. J. (2009), Gear crack level identification based on weighted K nearest neighbor classification algorithm, *Mechanical Systems and Signal Processing*, **23**(5), 1535–47.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L. (2014), Microsoft COCO: Common Objects in Context, in *Proceedings of the 13th European Conference on Computer Vision*, Zurich, pp. 740–55.

Lin, Y. Z., Nie, Z. H. & Ma, H. W. (2017), Structural damage detection with automatic feature-extraction through deep learning, *Computer-Aided Civil & Infrastructure Engineering*, **32**(12): 1025–46.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. & Berg, A. C. (2016), SSD: single shot multibox detector, in *Proceedings of the 14th European Conference on Computer Vision*, Springer International, Amsterdam, pp. 21–37.

Qu, Z., Feng, H., Zeng, Z., Zhuge, J. & Jin, S. (2010), A SVM-based pipeline leakage detection and pre-warning system, *Measurement*, **43**(4), 513–19.

Rafiei, M. H. & Adeli, H. (2016), A novel machine learning model for estimation of sale prices of real estate units, *Journal of Construction Engineering and Management*, **142**(2), 04015066.

Rafiei, M. H. & Adeli, H. (2017), A novel machine learning-based algorithm to detect damage in high-rise building structures, *The Structural Design of Tall and Special Buildings*, **26**(18), https://doi.org/:10.1002/tal.1400.

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.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016), You only look once: unified, real-time object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016*, IEEE, Las Vegas, pp. 779–88.

Ren, S., He, K., Girshick, R. & Sun, J. (2015), Faster R-CNN: towards real-time object detection with region proposal networks, in *Proceedings of the Advances in Neural Information Processing Systems*, NIPS, Montreal, pp. 91–9.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. & Berg, A. C. (2015), Imagenet large scale visual recognition challenge, *International Journal of Computer Vision*, **115**(3), 211–52.

Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 [cs.CV].

Sutskever, I., Martens, J., Dahl, G. E. & Hinton, G. E. (2013), On the importance of initialization and momentum in deep learning, in *Proceedings of the 30th International Conference on Machine Learning*, ICML, Atlanta, pp. 1139–47.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D. & Rabinovich, A. (2015), Going deeper with convolutions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015*, IEEE, Boston, pp. 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.

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.

Yu, S. N., Jang, J. H. & Han, C. S. (2007), Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel, *Automation in Construction*, **16**(3), 255–61.

Zeiler, M. D. & Fergus, R. (2014), Visualizing and understanding convolutional networks, in *Proceedings of the 13th European Conference on Computer Vision*, Springer International, Zurich, pp. 818–33.

Zhang, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y. & Chen, C. (2017), Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network, *Computer-Aided Civil and Infrastructure Engineering*, **32**(10): 805–19.

Zhang, L., Yang, F., Zhang, Y. D. & Zhu, Y. J. (2016), Road crack detection using deep convolutional neural network, in *Proceedings of the 2016 International Conference on Image Processing*, IEEE, Phoenix, pp. 3708–12.

Zhang, W., Zhang, Z., Qi, D. & Liu, Y. (2014), Automatic crack detection and classification method for subway tunnel safety monitoring, *Sensors*, **14**(10), 19307–28.

Zitnick, C. L. & Dollár, P. (2014), Edge boxes: locating object proposals from edges, in *Proceedings of the 13th European Conference on Computer Vision*, Springer International, Zurich, pp. 391–405.