8-2019

Deep Learning for Crack-Like Object Detection

Kaige Zhang
Utah State University

DEEP LEARNING FOR CRACK-LIKE OBJECT DETECTION

by

Kaige Zhang

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Heng-Da Cheng, Ph.D. Major Professor	Lie Zhu, Ph.D. Committee Member
Vicki Allan, Ph.D. Committee Member	Curtis Dyreson, Ph.D. Committee Member
Haitao Wang, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Deep Learning for Crack-Like Object Detection

by

Kaige Zhang, Doctor of Philosophy

Utah State University, 2019

Major Professor: Heng-Da Cheng, Ph.D.
Department: Computer Science

Computer vision-based crack-like object detection has many valuable applications, such as pavement surface inspection, underground pipeline inspection, bridge cracking monitor, railway track assessment, etc. However, in most of the contexts, the cracks appear as thin, irregular long-narrow objects, and often are buried into complex, textured background with high diversity which make the crack detection very challenging.

During the past a few years, the deep learning technique has achieved great success and has been utilized for solving a variety of object detection problems successfully. However, using deep learning for high accurate crack localization is non-trivial. First, region-based object detection cannot locate cracks accurately, and it is also very inefficient. Second, the networks are facing severe data imbalance issue inherent in crack-like object detection which can fail the training. Third, deep learning-based methods are also domain sensitive, which can result in poor model generalization ability. Fourth, deep learning is a data driven method that relies on a large amount of manually labeled ground truths (GTs) for the training which is labor-intensive and even infeasible, especially for annotating pixel-level GTs.

Focusing on the aforementioned problems, this research proposes the following solutions concerning crack localization accuracy and computational efficiency. First, we introduce a

deep classification network for crack-region preselection, which solves the noise problem that has troubled the researchers for many years. Second, we generalize a deep classification network to an object detection network with fully convolutional network (FCN) which boosts the computational efficiency via preventing the redundant convolutional operations existing in region-based object detection methods. Third, in order to improve crack localization accuracy, we propose a dense-dilation network for knowledge transfer, which succeeds the end-to-end training. Fourth, to address "All Black" issue caused by the data imbalance inherent in FCN-based crack detection, we introduce generative adversarial learning to perform the end-to-end training, and the method is also robust to the biased GTs. At last, we propose a self-supervised crack detection approach based on cycle-consistent generative adversarial networks, of which the training does not need manually annotating GTs, and it has potential to realize true fully automatic crack detection without human's intervention.

The proposed methods are validated on crack detection datasets from multiple sources, including asphalt pavement images, concrete road surface images, bridge crack images, building surface crack images, etc. The experiments demonstrate the effectiveness and superiority of the proposed approach.

(114 pages)

PUBLIC ABSTRACT

Deep Learning for Crack-Like Object Detection

Kaige Zhang

Cracks are common defects on surfaces of man-made structures such as pavements, bridges, walls of nuclear power plants, ceilings of tunnels, etc. Timely discovering and repairing of the cracks are of great significance and importance for keeping healthy infrastructures and preventing further damages. Traditionally, the cracking inspection was conducted manually which was labor-intensive, time-consuming and costly. For example, statistics from the Central Intelligence Agency show that the world's road network length has reached 64,285,009 km, of which the United States has 6,586,610 km. It is a huge cost to maintain and upgrade such an immense road network. Thus, fully automatic crack detection has received increasing attention.

With the development of artificial intelligence (AI), the deep learning technique has achieved great success and has been viewed as the most promising way for crack detection. Based on deep learning, this research has solved four important issues existing in crack-like object detection. First, the noise problem caused by the textured background is solved by using a deep classification network to remove the non-crack region before conducting crack detection. Second, the computational efficiency is highly improved. Third, the crack localization accuracy is improved. Fourth, the proposed model is very stable and can be used to deal with a wide range of crack detection tasks. In addition, this research performs a preliminary study about the future AI system, which provides a concept that has potential to realize fully automatic crack detection without human's intervention.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Heng-Da Cheng, for his strong support during my Ph.D. studies. I am very grateful to my committee members, Dr. Curtis Dyreson, Dr. Vicki Allan, Dr. Lie Zhu, and Dr. Haitao Wang for their great comments, advice, and contributions to this research.

I would like to thank all the friends at Utah State University and the colleagues at CVPRIP group.

I would like to thank my wife, Jing Han, for her warm encouragement and moral support at all the time. My sincere appreciation goes to my parents and parents-in-law, who helped me with taking care of my child. Thanks to my daughter, Crystal, she is my endless source of power.

Kaige Zhang

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Cracks are common defects on surfaces of man-made structures such as pavements, bridges, walls of nuclear power plants, ceilings of tunnels, etc. Timely discovering and repairing of the cracks are of great significance and importance for keeping healthy infrastructures and preventing further damages. With the increasing demand of computer-aided intelligent infrastructure monitor/inspection systems, such as pavement surface inspection [1], underground pipeline inspection [2], bridge cracking monitor [3], railway track assessment [4], etc., fully automatic crack detection has gained more and more attention in the past ten years. However, the task is non-trivial because in most contexts, the cracks appear as thin, irregular long-narrow objects, and are often buried into complex and textured background which make the task very challenging. In addition, in many applications, the image is large, such as pavement images of 2048×4096-pixel for road surface inspection, which requires the method to have high computational efficiency. Thus, a labor-light, computational efficient crack detection approach with high crack localization ability has great significance.

This research focuses on crack detection using deep learning. The dissertation is organized as follows. In Chapter 1, we review the previous works on crack detection and recent development on deep learning-based object detection; then we discuss the issue of using deep learning for crack detection. In Chapter 2, we present the work which utilizes a deep classification network to assist crack detection [5, 6]. In Chapter 3, we generalizes the classification network to a detection network with FCN which boosts the computation efficiency for processing large-size image, and also improves the localization accuracy [7]. In Chapter 4, we introduce generative adversarial learning to solve a practical problem, "All Black" issue, caused by the data imbalance inherent in crack-like object detection; in the meanwhile, the method is robust to biased GTs which frees the engineers from annotating the labor-intensive pixel-level GTs and greatly reduced the workload [8, 9]. In Chapter 5,

we discuss a self-supervised structure learning approach with cycle-consistent generative adversarial networks [10]; the network can be trained without using paired data which further reduces the work load. It is an important research direction for future crack detection systems.

Since pavement crack detection is one of the most difficult tasks in crack-like object detection, we discuss the methods based on solving practical problems from industrial pavement crack detection. Nonetheless, the methods are also verified using a variety of public datasets and can be applied to other crack detection applications.

1.1 Crack Detection

Traditional low-level image processing techniques have been deeply explored for crack detection in the past thirty years. Intensity-thresholding-based crack segmentation was quite popular in the early times because it was fast and straightforward. The goal was to find a proper threshold using grayscale information. Chan et al. [11] applied a thresholding method [12] to pavement crack detection which calculated the optimal threshold based on an ideal image model of which only object and background exist. Cheng et al. [13] proposed a fuzzy logic–based thresholding method which transformed the intensity domain into the fuzzy domain using the maximum entropy; they also designed a complete pavement distress detection system [14]. Oliveira and Correia [15] developed a dynamic thresholding method based on entropy; Kirschke and Velinsky [12] applied a histogram-based thresholding to crack segmentation. Koutsopoulos et al. [16] tested Otsu thresholding method [17] and the relaxation method [18]. Alekseychuk [2] developed a dynamic optimization method for crack-like object segmentation. Zou et al. [19], working on the assumption that crack intensity was usually consistent and lower than the background, designed an intensity difference-measuring function to find the threshold. These methods were sensitive to noise and would fail when the background contains complex textures.

More sophisticated feature extractions have also been used for crack segmentation. These methods usually work in two steps: feature extraction and pattern classification, with the first step being the key in segmentation. Abdel-Qader [3] discussed different edge

detectors for crack identification, such as Sobel, Canny and fast Haar transformation [20], finding that they were too sensitive to noise. Hu and Zhao [21] developed a crack descriptor based on a reduced local binary pattern (LBP) subset that assumed that cracks could be extracted using edge, corner and plain area information, overlooking the complexity of background textures. Zalama et al. [22] utilized visual features extracted by Gabor filters for crack detection; and they employed adaBoosting with a set of weak classifiers (each corresponding to a distinct filter/feature-extractor) for feature extraction. Shi et al. [23] used a integral channel feature for crack token mapping; and named CrackForest, it applied random structured forest [24,25] for crack-patch identification, where the distribution differences of the statistical feature histogram and statistical neighborhood histogram were used to discriminate cracks from false positives. Petrou and Kittler [26] designed a Walsh function-based texture descriptor for crack segmentation. Wang et al. [27] proposed a so-called trous edge detection algorithm for crack segmentation. Zhou et al. [28] selected a bunch of statistical features based on wavelet coefficients for crack segmentation. Nejad and Zakeri [29] developed an expert system for pavement distress detection based on wavelet theory. However, these methods were unable to separate crack pixels from the complicated background textures accurately because of being not able to catch sufficient global structural information.

Because a segmented result usually contains many disjoint crack fragments due to the intensity inconsistency along a crack, many methods have employed some linking operations to enhance the continuity. Zou et al. [19] utilized minimal spanning tree to generate continuous crack curves. Vaheesan et al. [30] used Hough transformation to enhance the connectivity of crack fragments. Zou et al. [19] employed tensor-voting for crack enhancement. Based on endpoint and orientation information, Song et al. [31] proposed a dual-threshold linking method to defragment crack segments. By checking the connectivity along eight directions, Cheng et al. [32] grouped the related crack fragments into a full crack. Formulate the problem as a trajectory-tracking, Huang and Xu [33] connected preselected crack seeds to build crack curves for crack detection. The segmentation method with post-processing

(noise removal and crack continuity enhancement) achieved a certain degree of success; however, it did not solve the problem well: the complex post-processing usually introduced quite a bunch of manually-tunable parameters, which was easy to fail when processing images from different sources.

1.2 Deep Learning for Object Detection

Over the last ten years, deep learning has achieved great success [34, 35], in which an artificial neural network (ANN) with multiple hidden layers is used. The deep network architectures can be deep belief network, deep recurrent neural network, deep convolutional neural network (DCNN), etc. For computer vision, DCNN is the main architecture, and it showed great advantages dealing with computer vision tasks with the following advantages. First, the convolutional kernels are learned automatically, so they have high flexibility and can mine useful information associated with the tasks by designing an objective function properly. Second, via deep architectures, the network can better catch global information via combining knowledge from different levels/scales; this is hard to achieve using traditional handcrafted feature extractors. Third, the deep architecture combined with different activation functions among the hidden layers provides the ability to model complex non-linear relation between input and output which can leverage the performance of processing complex pattern classification problems. With the aforementioned merits, DCNN has improved state-of-the-arts in various visual object detection benchmarks.

Object detection is to determine what objects are in an image (object classification) and where the object is (object localization). One of the most notable success in deep learning is the AlexNet [36], which won the ImageNet Large Scale Visual Recognition Challenge 2012 [37]. After that, a lot of related works were done by utilizing a deep classification network [36]. RCNN was the early work for object detection based on a classification network, it relied on region-proposal [38, 39] to find possible image regions and introduced a CNN to each region for feature extraction, and then sent the extracted features to a classifier such as SVM to determine what object it contains. Spatial Pyramid Pooling (SPP) network [40] calculated the CNN representation only once on the entire image and

used the features to find the representation of each patch; it needed a fixed input size due to the fully connected layers and introduced a special pooling layer after the last convolution layer. Fast R-CNN [39] improved SPP net by enabling the gradients propagation through spatial pooling and made the network possible to be trained end-to-end, and it added a bounding box regression to the network training. Faster RCNN [41] improved Fast RCNN by replacing selective search with a small CNN called Region Proposal Network to generate regions of interest (ROI); it introduced anchor box and utilized different scales and aspects of ratios to generate candidate regions. R-FCN [42] introduced FCN into Faster RCNN and used position sensitive convolution for the classification; it still followed the two-stage detection ides, region-proposal and classification, but all the convolution operations were shared. Single shot detector (SSD) [43] conducted the two-stage processing in a single shot, simultaneously predicting the bonding box and class label; it introduced non-maximum suppression to group highly overlapped bounding box into a single box. Similarly, YOLO [44] divided an image into S×S grids and each grid predicted an object. An improved version of YOLO, YOLO9000 [45], utilized 9000 object classes to train the model and introduced fully convolution and multiscale prediction to improve the computation efficiency and detection accuracy, respectively. Mask-RCNN [46] improved the detection accuracy by removing the Softmax layer which eliminates the competition between different classes and achieved state-of-the-art performance.

1.3 Deep Learning for Crack Detection

Based on a classification CNN, [47] used per-pixel window-sliding for crack detection which is extremely inefficient and could cost tens of days to process a large size image (2048×4096-pixel). Rather than the per-pixel window-sliding, [5] utilized a CNN for pre-selection which divides a full-size image into a few image blocks, and find and remove most of the background/non-crack area before performing crack segmentation within a crack patch/region. Similarly, Cha et al. [11] proposed a CNN for block-level crack detection without post processing which cannot localize the cracks accurately. Zhang et al. [48] proposed a crack detection method, CrackNet, with hand-crafted convolutional kernels which

also relies on window-sliding to process full-size images. Processing large-size images with window-sliding are very inefficient, because a great number of image blocks need to be processed, even introducing parallel processing [48]. To solve the computation efficiency problem, Zhang et al. [7] generalized a classification network to a detection network with fully convolutional design; the authors also detailed the inefficiency reason of the window-sliding based detection, and improved the computation efficiency by tens of times via eliminating the redundant convolution and performing crack detection end-to-end in one stage. While the computation efficiency has been greatly improved, localization accuracy is not satisfactory. Zhang et al. [7] tried to improve the crack localization accuracy by performing an end-to-end training, it failed due to the biased GTs and data imbalance inherent in crack-like object detection. At the same time, Yang et al. [49] proposed to use a fully convolutional network for pixel-level crack detection based on the images with high resolution and quality, and with carefully annotated GTs; however, it failed to detect thin cracks which are quite common in industrial pavement images. Targeting on building an industrial pavement crack detection system, [8, 9] introduced generative adversarial learning and an asymmetric U-shape network to handle the dada imbalance and biased-GT. A recent work [10] proposed a self-supervised structure learning method for crack detection based on cycle-consistent generative adversarial networks that does not need to manually prepare pixel-level GTs for training and it achieved comparable results with supervised methods. It could be an important research direction of automatic crack detection system in the future.

CHAPTER 2

PRESELECTION WITH DEEP CLASSIFICATION NETWORK

2.1 Background

As been discussed in the introduction, the main problem troubled the industrial application of crack detection is the noise issue which has not been solved using traditional image segmentation. It is difficult to find a reliable threshold for crack segmentation: the result either produced many crack fragments with a lot of noise, or missed most of the cracks. In the meanwhile, when trying to perform noise removal, it may remove the true crack fragments; and when performing crack defragments, it may link the noises together and create undesirable false positives.

Before deep learning, the researchers had tried to solve the problem by designing various hand-crafted feature extractors and applying some machine learning techniques to train a model for crack/non-crack classification. Shi et al. [23] introduced random structured forest [24] for crack detection where the integral channel features of 3 colors, 2 magnitudes and 8 orientations were used to discriminate crack patches from non-crack patches based on a crack-token mapping strategy; and it used the distribution differences of a statistical feature histogram and a statistical neighborhood histogram to identify the false positives. It achieved very good results when processing the images captured with cellular phone where the images were with high quality, and the BGs (BGs) were smooth. However, the performance decreased dramatically when processing the industrial images. In addition, the method cannot remove the noise connected to the true crack regions and such cases were quite common in crack detection. In summary, the hand-crafted feature descriptors calculated statistical features at some local scope, then employed machine learning to discriminate cracks from BGs. However, they were not able to solve the problem well because hand-crafted features can only collect information from a very limited space and lack of

the global information, even the statistics at different locations were combined to build an integrated feature vector, such as Hog [50]; thus, they cannot represent the structural information well which was important to discriminate cracks from noisy textures. A more intuitive explanation is that the fake crack-debris produced by the complicated BG textures would probably generate the same feature descriptors, such as histograms of oriented gradients that will confuse the classifier and fail the job.

Different from traditional pattern classification methods, DCNN performs feature extraction and machine classification at one stage with the following advantages. First, the multiple convolutional layers with different activations can model the high non-linearity; and the convolutional kernels are learned automatically, so they have high flexibility and can mine more useful information. Second, via deep architectures, the network can better catch global structural information by combining knowledge from different levels/scales; and this is hard to achieve using traditional handcrafted feature extraction methods.

2.2 Related Works

2.2.1 Deep convolutional neural network

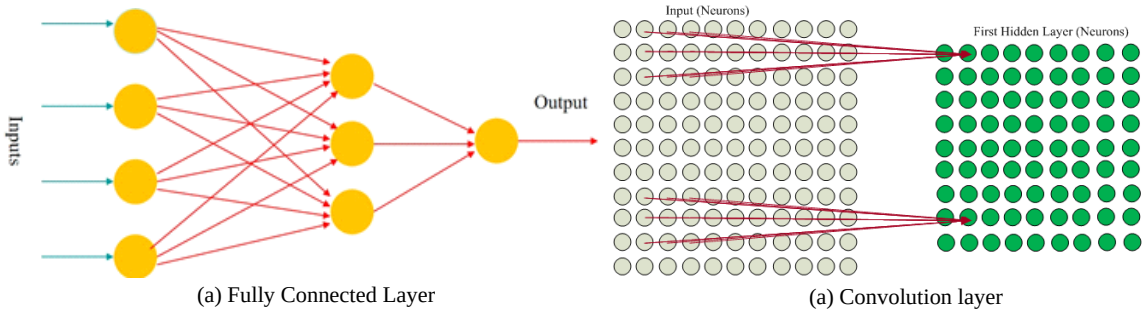


Fig. 2.1: Illustration of fully connected layer and convolutional layer.

In DCNNs, two types of layers contribute the main functions: convolutional layer and fully connected layer. Take a two-layer network as example. For fully connected layer, Fig. 2.1 (a), each input neuron connects to every neuron in the next layer with some trainable

weights; for the convolutional layer, Figure 2.1 (b), the main differences are the local-connection design and weights sharing. Each neuron in the next layer only connects with the neurons in a local region. Moreover, for image processing problems, the local weights are just the same as 2-D filters operating the convolution operations on the input neurons at a specific stride, and outputting the neuron-values of the next layer. Here, "deep" means that there are multiple layers, and the convolutional layers usually appear at the lower layers that serve as feature extractor; the fully connected layers are at the end serving as the classifier that outputs the labels or membership values. Other auxiliary designs may be included between layers, such as MaxPooling [36], ReLu [51], Dropout [52, 53], etc.

With such setting, DCNN can extract representative features for different tasks such as image classification, object detection, image segmentation, etc.

2.2.2 Transfer learning

In deep learning, the most challenging thing is to train the network. A deep neural network usually has tens of millions of parameters, e.g. the AlexNet designed [36] has 60 million and the VGG [54] has 138 million; therefore, it requires a large amount of labeled data for training. In practice, however, it is rare to have datasets of sufficient size. Fortunately, people find that through transferring the knowledge learned from different resources and fine-tuning the network based on limited data, it can still get the network converged. Depending on the situations between the source and target domains and tasks, there can be different transfer learning strategies according to what knowledge to transfer and how to transfer. Following the definition by Pan Yang [55], given a source domain D_s and a learning task T_s, and a target domain D_t and a learning task T_t, transfer learning intends to improve the learning of the target predictive function $f_t(\cdot)$ in D_t by applying the knowledge in D_s and T_s, where $D_s \neq D_t$, or $T_s \neq T_t$. Depending on the situation between the source and target domains and tasks, inductive transfer learning and transductive transfer learning are defined as follows [55]. Inductive transfer learning: In this setting, the target and source tasks are different, regardless of the similarity between the source and target domains. This means that the aim of inductive transfer learning is to improve the learning

of $f_t(\cdot)$ in D_t applying the knowledge in D_s and T_s, where $T_s \neq T_t$.

Transductive transfer learning: Transductive transfer learning corresponds to the cases where the source and target tasks are identical, but the source and target domains are different. In other words, transductive transfer learning intends to improve the learning of $f_t(\cdot)$ in D_t via applying the knowledge in D_s and T_s, where $D_s \neq D_t$ and $T_s = T_t$. Yosinski et al. [56] discussed the transferability of knowledge from different layers in deep convolutional neural network. In general, low-level convolutional layers learn more generic features with strong transferability, and higher-level layers learn knowledge more specific to the task. In this Chapter, we transfer the generic knowledge from a pre-trained network using ImageNet data [37] and fine-tune the network to classify crack and non-crack blocks for pre-selection.

2.2.3 LASSO

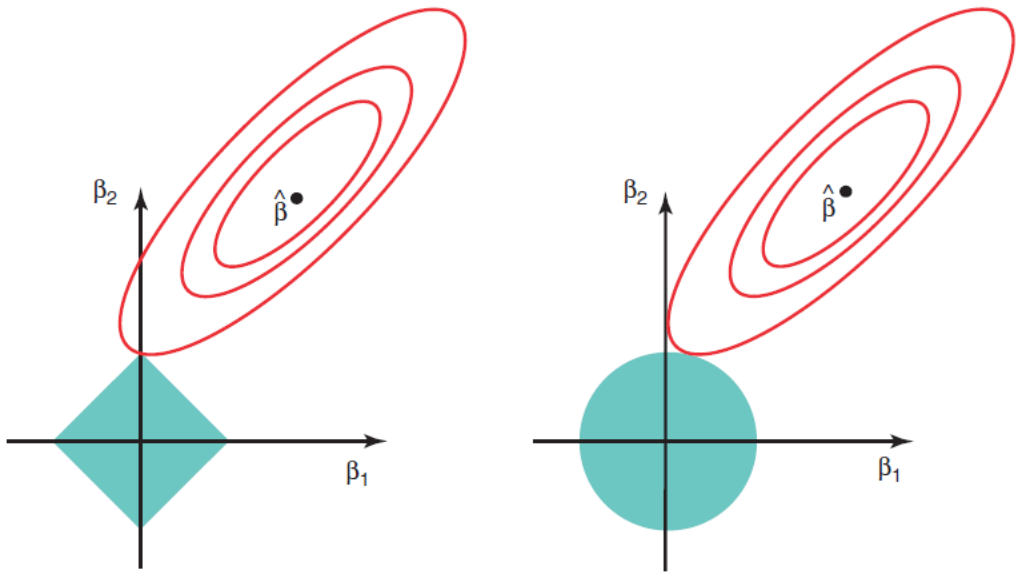


Fig. 2.2: Illustration of the difference between ridge regression and LASSO. (Figure from [57])

For machine learning, a good model is very important. Linear regression is the most popular and widely used model. Suppose there are n observations $\{x_i, y_i\}_{i=1}^{n}$. Each ob-

servation includes a response variable y_i and a column vector x_i of p predictors. The mathematical expression of the ordinary linear regression model is

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i \tag{2.1}$$

The goal is to find the "best" coefficients β that minimize the objective function S as follows:

$$S(\beta) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right|^2 \tag{2.2}$$

Ridge regression improved the ordinal linear regression by making a L2-norm constraint to the objective, and the objective function became

$$S(\beta) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right|^2 + \lambda \|\beta\|^2 \tag{2.3}$$

Least absolute shrinkage and selection operator (LASSO) [57] replaced the constraint with L1-norm, and the objective function became

$$S(\beta) = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right|^2 + \lambda \|\beta\| \tag{2.4}$$

Comparing with ridge regression, LASSO tends to force the coefficients of the least influential features to be exactly zero. Take a two-dimensional feature as example, in Figure 2.2, the blue diamond denotes the LASSO constraint with L1 penalty and the blue circle is the ridge regression with L2 penalty, respectively. The optimization is equal to expending the red ellipse until it touches the L1 or L2 constraint area. From Figure 2.2, the coefficient estimations of LASSO and ridge regression are given by the point at which the ellipse first touches the diamond/circle region. For LASSO, the first contacted point will be the one located at some axes where some of the coefficients are zero; while for ridge regression, the point will be somewhere at the boundary of the circle. Thus, LASSO could be used for variable/feature selection.

2.3 Proposed Method

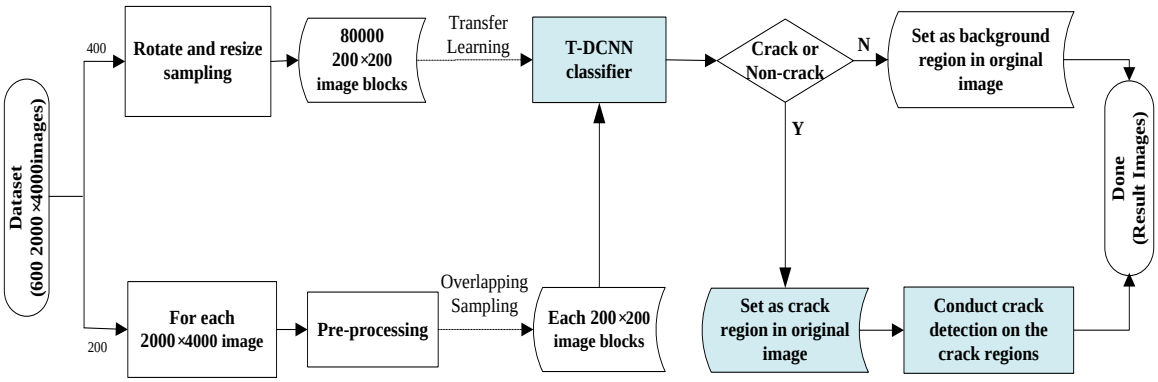
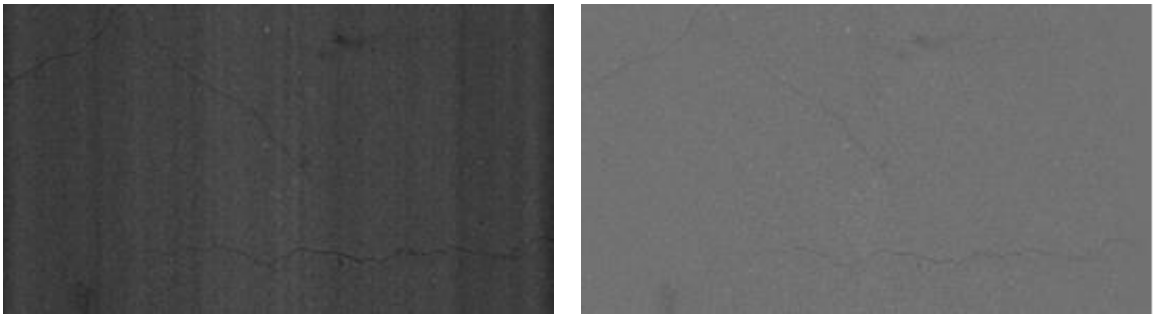


Fig. 2.3: Flowchart of the proposed method

Deep learning has showed great superiority in solving complicated pattern classification problems. The proposed method introduces a deep classification network to perform the preselection which screens out most noisy BG-areas before crack detection. First, a column-wise preprocessing is conducted to eliminate the illuminance unbalance; second, transfer learning are used to train a DCNN to divide an image into crack and non-crack regions; in this way, the possible noises from the BG are removed according to the region label. Focusing on the crack regions, a block-wise segmentation algorithm is developed to find the possible crack pixels with less noise. At last, tensor voting is employed to improve the crack continuity and extract the crack curves as the detection results. A flowchart of the proposed approach is shown in Figure 2.3.

2.3.1 Preprocessing

Different from the images captured by cellular phones [19, 23], the data used in this work are pavement images captured by an industrial line-scan camera which scans road surface at 4.096-width and produces a 2048×4096-pixel image every 2048 scans (i.e., 1-pixel represents 1 mm^2 road area). Due to the uneven lighting, the illuminances along different locations can be different, and that create non-uniform intensities at different columns, as shown in Figure 2.4(a). However, the intensities along the same column are quite similar



(a) (b)

Fig. 2.4: Illuminance rebalance. (a): A low-resolution pavement image captured using a vehicle running at 100km/h. (b): The illuminance balanced image (right).

and the mean value can be used to represent the illuminance, i.e., the average intensity of each column can be mapped to the same value for illuminance balancing. Such column-wise illuminance-rebalancing can be realized using Eqs. (2.5) and (2.6) as follows [22]:

$$I_{ij}^{'} = \bar{G}\frac{I_{ij}}{A_i^k} \qquad i = 1, 2, \ldots, M; j = 1, 2, \ldots, N \tag{2.5}$$

where I_{ij} = the gray value of the k-th image at column i and row j; k = image number; $I_{ij}^{'}$ = modified pixel value; M = number of image columns; N = number of rows; and $\bar{G}$ = predefined average gray value for each column (128 is used) after the preprocessing.

$$A_i^k = \alpha A_i^{k-1} + (1-\alpha)\, a_i^k \qquad i = 1, 2, \ldots, M \tag{2.6}$$

where $\alpha = \beta\,(k - 1/k)$ = variable to control the weight of the current image, which is also the latest-measured illuminance information; β = empirical constant (0.75 is used); A_i^k = weighted average intensity value of the i-th column utilized to calculate the balanced intensity of the k-th image; and a_i^k = average value of the i-th column in the k-th image. See Figure 2.4 (b) for the result after the preprocessing.

2.3.2 Preselection with transfer learning

Transfer leaning is introduced to train a deep convolutional neural network for the classification of crack and non-crack blocks, which screens out most noisy BG areas. A

deep convolutional neural network, AlexNet, with five convolutional layers and two fully connected layers is used. The convolutional layers are trained to find proper kernels for feature extraction automatically; the multilayer deep architecture is used to catch the global structural pattern by combining knowledge from different levels. The generic knowledge from ImageNet pre-trained model is transferred for training.

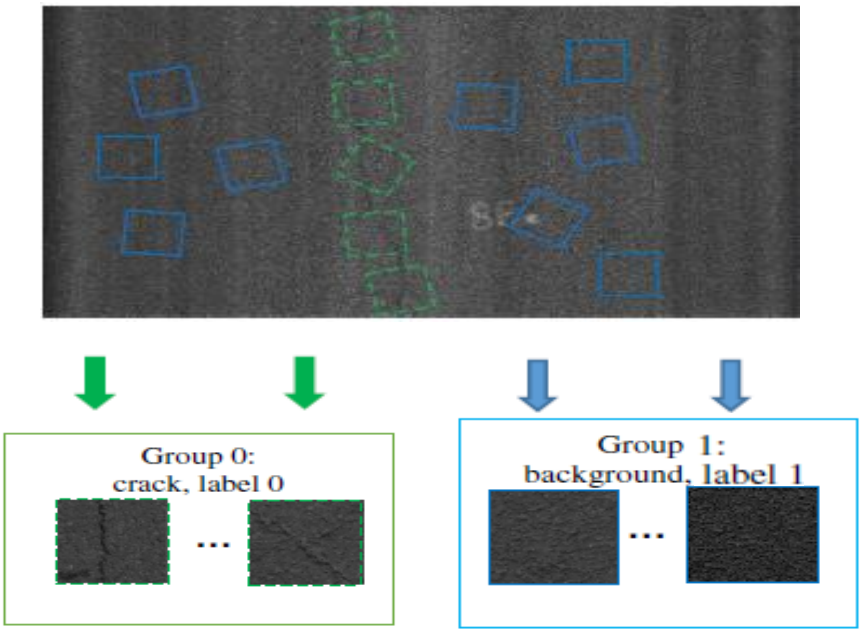


Fig. 2.5: Illustration of Image-block generation

Image-block generation: To train a deep convolutional neural network, a relatively large dataset is needed. In the experiment, 600 pavement images (2048×4096-pixel each) with low similarity are selected from 20,000 images. Among them, 400 images are used to obtain the training set of 30,000 crack blocks and 30,000 BG blocks sized 400×400-pixel each. The images are further used to generate the 200×200-pixel sub-blocks by cropping the center square area to perform a two-step pre-selection. The other 200 images are used to obtain the test set of 20,000 crack blocks and 20,000 BG blocks. As shown in Figure 2.5, the image blocks are generated as follows:

- For each image of 2048×4096-pixel, cracks are manually marked by 1-pixel curves;

- Crack blocks are sampled from the original images along the crack curves, and the BG blocks are sampled randomly from the areas far from the crack curves;
- To reduce similarity, the distance d between two block samples satisfies $d > w/2$, where w = side length of the sample block.

Image resize and image rotation are used to augment the data and increase the diversity based on the following knowledge: (1) cracks are direction invariant because that a crack is still a crack when it is changed to another direction; and 2) different cracks may have different widths, and the BG textures might have different coarse levels; therefore, the resized images (90%, 95%, 100%, 105% and 110% of the original images) can also be used to generate crack blocks.

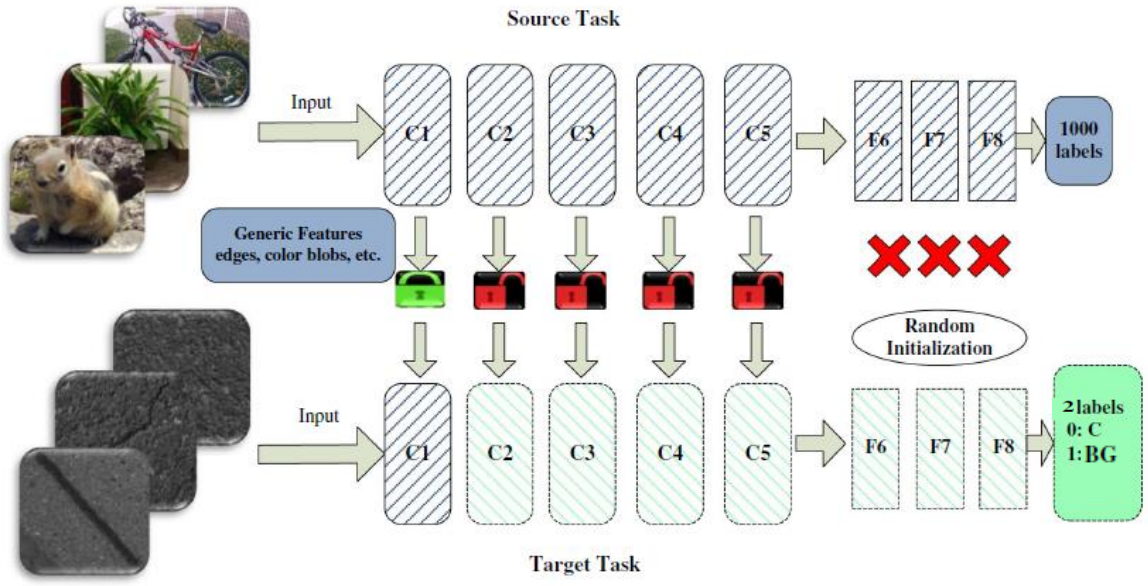


Fig. 2.6: Transfer of generic knowledge based on ImageNet data (C1, C2, C3, C4, and C5 = five convolution layers; F6, F7, and F8 = three fully connected layers; C = crack, BG = BG).

Network architecture: A DCNN usually contains tens of millions parameters. Directly learning such huge number of parameters is very challenging, especially when the dataset is not large enough. It has been verified that the knowledge learned from one task can

be used to ease the network training of other tasks [55, 58]; that is, the learning of a task can be improved by using the knowledge learned from other tasks, and the improvement is significant in large-data driven tasks such as deep learning. Typically, the generic features, such as edge and color blobs, occur at high probability regardless of the exact cost function and image data [56]. In DCNNs, low-level convolutional layers learn more generic features with strong transferability and higher level layers learn knowledge more specific to the task [55]. In this work, the generic knowledge in the first convolutional layer is transferred from the pre-trained model, AlexNet, based on the following considerations:

- The crack patterns are relatively simple which can be represented by the generic knowledge. For example in Figure 2.7, many feature maps show crack patterns similar to those in the original images which support this assumption;
- The crack patterns have little similarity with the natural objects (e.g., dog, cat); therefore, middle-level and high-level knowledge is less useful for crack detection and is not transferred; and
- By transferring the generic features from the first layer of the source task, the network needs not to relearn the basic features that make the network training easier.

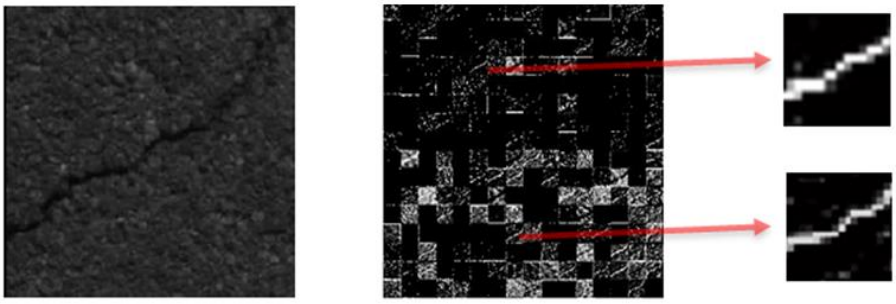


Fig. 2.7: Feature maps of a sample image. Left: an image block. Right: feature maps after the process of first convolutional layer.

In Figure 2.6, the source network is pre-trained using the data [36] with Caffe [59]. The model takes a square (227×227) image block as input and produces a probability distribution

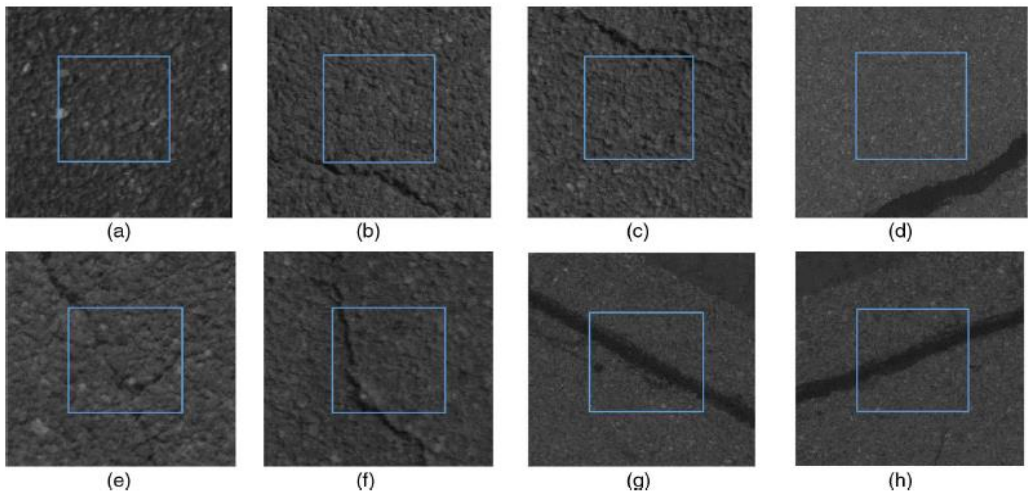


Fig. 2.8: Classification results for different image blocks; squares indicate mask blocks: (a - d) BG; (e - h) crack.

over the 1,000 object classes. The network is mainly composed of five convolutional layers (C1, C2, C3, C4 and C5) followed by three fully connected layers (F6, F7 and F8). The five convolutional layers serve as a feature extractor; after each convolutional layer, a max-pooling [36] operation with kernel size of 3×3. The three fully connected layers serve as a classifier to output the class labels by computing $Y_6 = \sigma(W_6 Y_5 + B_6)$, $Y_7 = \sigma(W_7 Y_6 + B_7)$, and $Y_8 = \phi(W_8 Y_7 + B_8)$, where Y_k denotes the output of the k-th layer, W_k and B_k are trainable parameters of the k-th layer; and $\sigma(X[i]) = max(0, X[i])$ and $\phi(X) = e^{X[i]} / \sum_j e^{X[i]}$ are ReLU and SoftMax, respectively. The stochastic gradient descent (SGD) [60] is used to update the parameters during training. In the target task, pavement image blocks are input into the network, and resized to 227×227-pixel for transfer learning. As shown in Figure 2.6, the first lock on the left shows that the generic features learned from the first convolutional layer C1 are transferred directly and kept unchanged during training; the other four locks indicate that the parameters from C2, C3, C4 and C5 are copied from the source network but allowed to change during training, which means that the higher level knowledge is relearned for the target task; finally, the parameters of the last three fully connected layers, F6, F7 and F8, are randomly initialized and trained by SGD.

Two-step pre-selection: Based on the transfer learning, a pavement image is first divided into smaller blocks, which are sent to the DCNN for classification. As shown in Figure 2.8, if a crack traveling through the center area of a block (the square area in the middle of the block with side length $w_c = w/2$, where w is the side length of the image block), the block is classified as a crack block; and if the crack is near the border of the image block, it is classified as BG. Specifically, a 200×200-pixel image block with a crack traveling outside the center area with w_c =100 is classified as a BG block. In addition, the size of the image block can influence the classification performance: the smaller it is, the less global information it catches; this makes the classification more difficult and increases computation complexity because more blocks need to be processed from a full-size image. The larger the image block is, the more BG regions are involved, which on the other hand makes the pre-classification less useful. Moreover, when applying transfer learning, if the image was too large, crack information could be lost when resizing to 227×227-pixel. Based on these considerations, a two-step pre-classification is designed. First, a mask block is defined as the center square area of the image block; this is a smaller image region with side length $w_c = w/2$ (Figure 2.8), and it is used to locate the crack region in a mask image. The mask image is defined as a binary image with the same size as the original image which utilized ones to represent crack regions and zeros to represent the BG regions. Considering the efficiency and accuracy, a two-step classification is performed as follows: at the first step, 400×400-pixel image blocks are sampled from the original image in the left-to-right and top-to-bottom order, and they are input into the DCNN sequentially with the step-size d_s of 100 pixels of which overlapping exists between adjacent samples. If the image block is classified as crack, the related mask block area in the mask image is set as 1; the BG blocks are initialized as 0s. Second, sampling of 200×200-pixel blocks with a step-size of 50 pixel is conducted focusing only on the crack regions produced in the first step to obtain more accurate areas with mask blocks of 100×100-pixel. In this way, the related mask images are generated, refer to Figure 2.9 (c), and most noisy BG regions can be detected and discarded before later processing. The cracks are separated and located in 100×100-pixel

area to assist crack segmentation and extraction. Figure 2.9 shows the related results.

2.3.3 Crack curve extraction

After pre-classification, cracks can be located in 100×100-pixel square areas. To obtain more accurate crack locations to facilitate calculation of cracking extent [1, 61, 62], a crack curve detection is applied to extract the crack curves. First, a block-wise segmentation method is designed to produce the binary image; then tensor voting is used to extract the final crack curves.

Image segmentation: Image segmentation clusters pixels into groups and assigns each group a label. In this Chapter, the pixels in a pavement image are grouped into cracks and BG. A block-wise segmentation method is developed and applied to each crack block after the pre-selection to find the possible crack pixels. Segmentation is realized by using a thresholding method where a linear regression model is used to find the proper threshold for each block. In detail, 2,000 low similarity image blocks, 100×100-pixel each, are collected; the related statistics and the best threshold for each block are then calculated and used to fit a linear model as discussed subsequently. The best threshold is defined as the one that maximizes the F1 measure [63] of the segmented result against the GTs of each image block; it can be obtained automatically by comparing the segmented results using different threshold values with the GTs (the values from M-30 to M+30 are used, M is the mean value of the block); the threshold value maximizing the F1 measure is selected as the best threshold for that block. The related statistics are mean (M), standard deviation (SD), smoothness (SM), third momentum (TM), and uniformity (UF). The linear model is

$$T = \beta_0 + \beta_1 M + \beta_2 SD + \beta_3 SM + \beta_4 TM + \beta_5 UF \tag{2.7}$$

where β_0, β_1, $\cdots$, β_5 = parameters to be determined by least square formula. It is well known that the stability and prediction accuracy of a linear model are heavily related to the selected predictors [64]; here, the least absolute shrinkage and selection operator (LASSO) [57] are used. The constraint of the least squares is

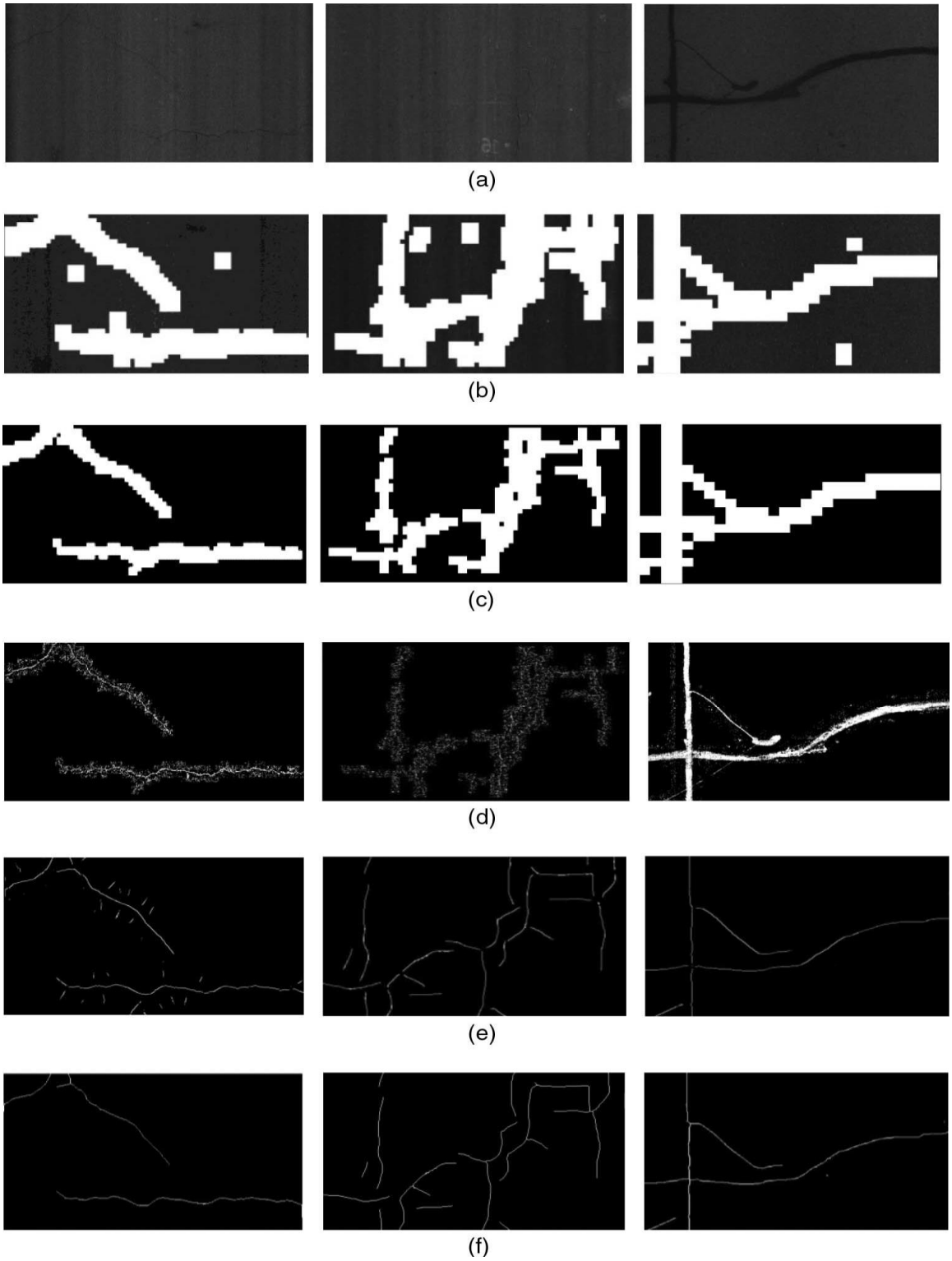


Fig. 2.9: Step-by-step results for proposed method: (a) original images; (b) results after first step of pre-selection; (c) mask images after second step of pre-selection; (d) results for block-wise segmentation; (e) results for curve detection; (f) final results for detected crack curves.

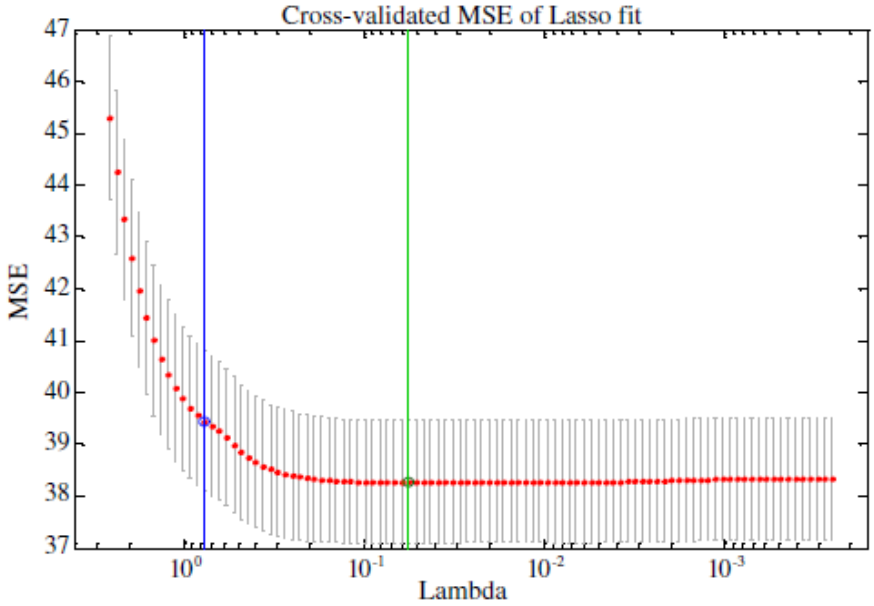


Fig. 2.10: Best lambda selected by cross-validated mean square error (MSE) based on minimum-plus-one standard error formula (solid line to left = lowest value achieved by MSE; solid line to right represents minimum-plus-one standard error, which is also the lambda chosen by LASSO).

$$min \sum_{i=1}^{n} [T_i - (\beta_0 + \beta_1 M + \beta_2 SD + \beta_3 SM + \beta_4 TM + \beta_5 UF)]^2 \tag{2.8}$$

subject to

$$\sum_{j=1}^{5} \|\beta_j < s\| \tag{2.9}$$

where s is constant; and $s \geq 0$.By using the Lagrange multiplier, it is equivalent to minimizing the residual sum of the squares plus a penalty term as

$$min \sum_{i=1}^{n} [T_i - (\beta_0 + \beta_1 M + \beta_2 SD + \beta_3 SM + \beta_4 TM + \beta_5 UF)]^2 + \lambda \tag{2.10}$$

where $\lambda \geq 0$. When s is small (or equivalently λ is large), some of the regression coefficients will approach to 0, which serves to eliminate the predictors that are not significant. As shown in Figure 2.10, λ is set as 0.729 by the minimum-plus-one standard error formula [65],

achieving a good R2 (0.68). Finally, M, SD and UF are selected as the most significant predictors, and the prediction model is

$$T = 1.11 * M - 1.21 * SD + 0.33 * UF - 5.52 \tag{2.11}$$

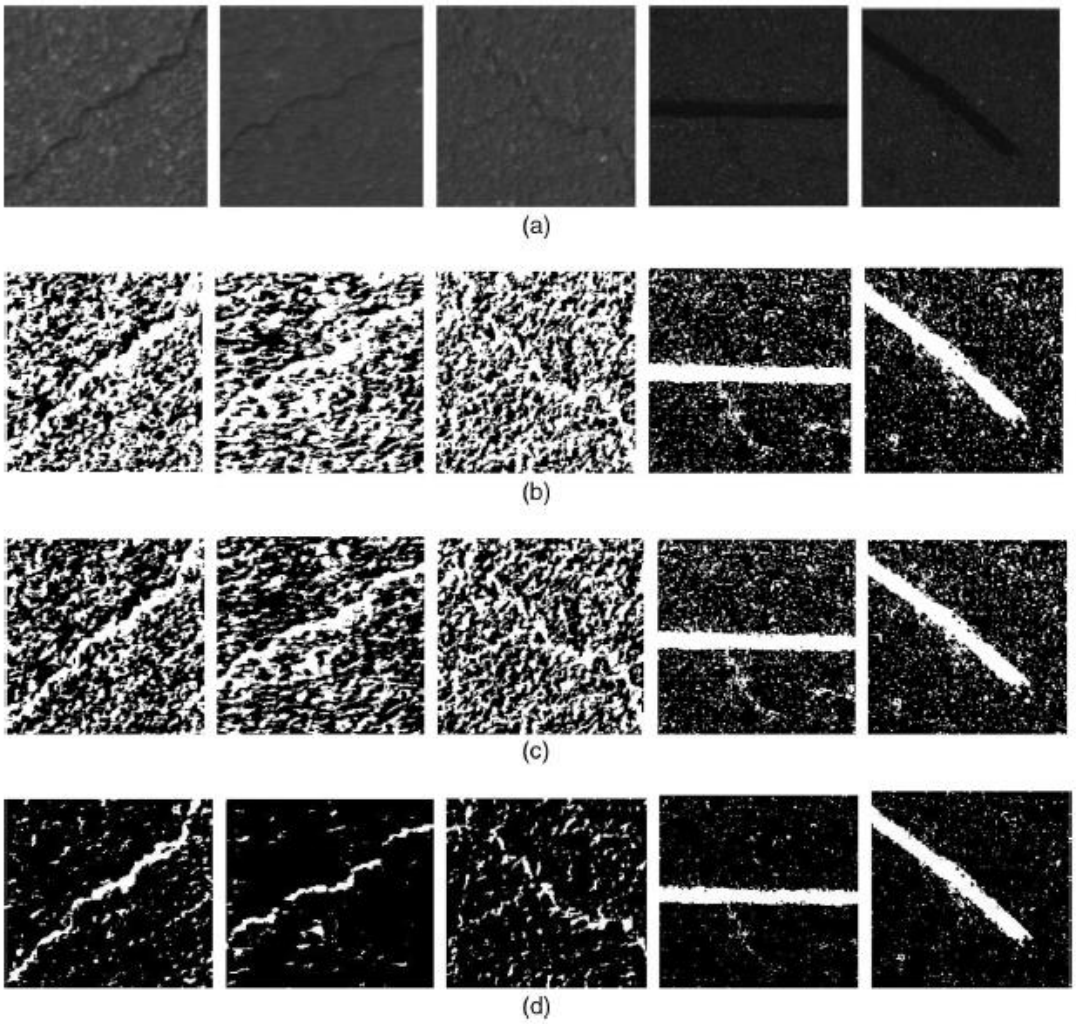


Fig. 2.11: Comparison of segmentation methods on image blocks: (a) original crack image blocks; (b) segmentation results using [17]; (c) segmentation results using [19]; and (d) segmentation results for proposed method.

In addition, the performance of other thresholding methods, including Otsu [17] and Zou et al. [19], is compared; both achieved lower R2 (0.30 for Otsu (1975) and 0.41 for Zou et

al. [19]). The related segmentation results are shown in Figure 2.10. Focusing on the crack block-regions, the best thresholds are calculated using Eq. (2.11), and the thresholding is conducted to obtain the segmented blocks and to produce the segmented binary images as in Figure 2.9 (d).

Crack curve extraction: As shown in Figure 2.9 (d), the segmented results may contain many discontinuous crack fragments and noises; therefore, tensor voting based curve detection [66] is used to connect the crack fragments along some curves and produce crack curves with less noise. Notice that there are some small gaps in the curves generated by using the local maximum voting strength [67]. Therefore, a gap filling operation using the morphology close operation [20] is applied to connect cracks segments if the nearest distance among them is less than 10 pixels. Figure 2.9 shows the final results after this operation.

2.4 Experiments

The experiments are performed using an HP Z220 workstation with 8G of memory; an Nvidia Quadro K4000 GPU is used for training and testing. The system is built with the Caffe software package; and MATLAB R2014a is used.

2.4.1 Dataset and metrics

The dataset is obtained from the images captured by a line-scan camera mounted at a height of 2.3 m on the top of a vehicle; the camera can scan a 2×4-m road area to generate a 2,000×4,000-pixel image (i.e., a road area of 1 mm2 corresponds to 1 pixel). The vehicle can work at 100km/h. Detection performance are compared with that of four methods in terms of recall, precision and F1-measure. Precision and recall can be computed using true positive (TP), false negative (FN) and false positive (FP) as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.13}$$

2.4.2 Fine-tuning details

The source network in Figure 2.6 is pre-trained with Caffe using ImageNet data. Transfer learning is realized via the following fine-tuning strategy. The input image block is resized to 227×227 pixels, and the generic knowledge is transferred by copying the parameters, which are unchanged during training, of the first convolution layer from the source task. The base learning rate $base_lr$ is 0.001, the test iteration is 100, and min-batch is 400. The learning policy is that, for every step-size (100 is used) iteration, the learning rate decreases to

$$lr_{step} = base_lr * gamma^{floor(iter/step)} \tag{2.14}$$

where lr_{step} = learning rate after $iter$ iterations; $iter$ = current iteration time; $step$ = step size; $base_lr$ = base learning rate; and $gamma$ = decreasing factor (set as 0.2). Weight decay, which is used to update the backpropagated gradient, is set as 0.005 and all weights of the convolutional layers are copied from the pre-trained source network. The parameters of the first convolutional layer are unchanged during training, whereas the parameters of the second through fifth convolutional layers are retrained; the weights of the last three fully-connected layers are set randomly and trained from scratch. The output labels of the last layer are 0 and 1, which stands for the C (crack) and BG (BG), respectively. The maximum iteration/epoch is set to 40,000. Finally, an overall accuracy of 0.953 is obtained after 40,000 iterations on the test set; however, the network reaches its best performance after about 8,000 iterations and changes little from 8,000 to 40,000 iterations.

2.4.3 Performance

For evaluation of detection, the proposed method is compared with traditional edge detection methods, Canny, CrackIT and CrackForest. In the experiments, the ground truths (GTs) of 300 test images are manually marked. As shown in Figure 2.12 (c), Canny segmentation cannot obtain any intuitive cracking information because it is too sensitive to noise; and it achieves high recall (0.999) but very low precision (0.001). CrackIT cannot detect cracks because of the low resolution and low contrast of the industrial pavement image,

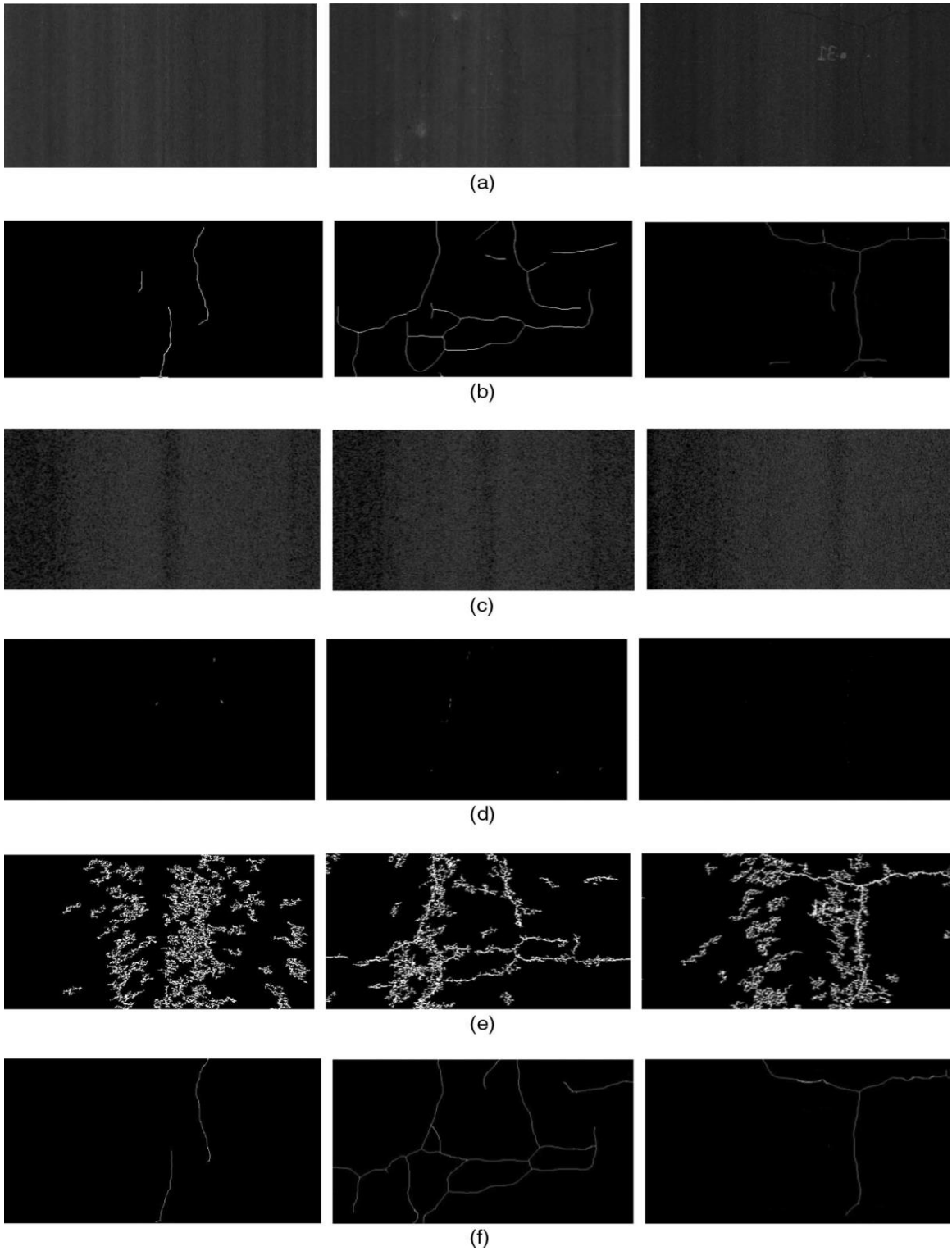


Fig. 2.12: Comparison of results using actual industry images: (a) original pavement images; (b) manually marked GTs; (c) Canny; (d) CrackIT; (e) CrackForest; (f) proposed method.

Table 2.1: Performance evaluation of the method based on pre-selection

Method	Recall	Precision	F1-measure
Canny	**0.999**	0.001	0.002
CrackIT	0.891	0.252	0.490
CrackForest	0.851	0.515	0.641
Proposed method	0.951	**0.847**	**0.895**

Figure 2.12(d); consequently, both recall and precision are very low (0.001). CrackForest shows good detection property via its crack mapping strategy using random forest; however, it cannot cope with noise, especially the noise connected to the true crack regions; also, it achieves low precision (0.515). Table 2.1 summarizes the statistical results.

2.5 Summary

In this Chapter, a deep classification network is introduced to perform a pre-selection which removes most BG areas before crack segmentation based on a window-sliding strategy. We have present that the generic knowledge learned from ImageNet contains rich structural information and it can be used to extract crack patterns for crack/non-crack classification. The pre-classification with DCNN outperformed the hand-crafted feature extraction method, which also indicated that the global information is important to discriminate cracks from complicated BG textures. However, the computational efficiency remains an issue.

CHAPTER 3
TRANSFORM A CLASSIFICATION NET TO DETECTION NET

3.1 Background

In Chapter 2, we introduced a DCNN to perform a pre-selection before crack detection, and achieved very good result. It is a region-based object detection method where window-sliding is needed when processing large input images. However, the window-sliding based processing is very inefficient when dealing with large images because a lot of image blocks have to be processed in sequence. Such problem is a serious issue which impeded the applications of deep learning based object detection, but it is rarely mentioned in the references [38, 39]. For example, the time cost of processing an industrial pavement image, 2048$times$4096-pixel, is larger than 10 seconds using state-of-the-art method Fast-RCNN [39], which is far from the real-time processing (0.1 seconds per image). What is more, in order to achieve good crack localization accuracy, the method employed complex post processing to extract a crack curve for the detection, which was expensive as well. In this Chapter, we explore the reason of the inefficiency, and propose a one-state crack detection method to improve the computation efficiency by more than 10 times; and it also proposes a training strategy to improve the localization accuracy.

3.2 Related Works

3.2.1 Fully convolutional network (FCN)

A regular DCNN involves two types of layers, convolutional layer and fully connected layer. Before FCN, the DCNN based methods are restrained by the condition that the input image has to be a fixed size [38], and once the training is finished, it can only deal with images with the specific size, or the inputs have to be resized. However, it turns out that the fully connected layer can be grouped as a special case of the convolutional

layer with kernel size equal to the entire input dimension which can be used to address the issue. Long et al. [69] proposed the fully convolutional network design for per-pixel semantic segmentation. It implements an end-to-end, pixel-by-pixel supervised learning mechanism by introducing some up-sampling strategy to remedy the resolution loss. Instead of paying attention to building an end-to-end network focusing on the up-sampling, we introduce the fully convolutional design to convert a classification net into a detection network that can work on the full-size images seamlessly. More important, our work also shows that the semantic segmentation network is the same as classification network working under larger field of view, which could be used to explain many of the phenomenon with fully convolutional layers such as the boundary vagueness in semantic segmentation [69], blurry of generated images using GAN [70], etc.

3.2.2 Dilated convolution

The dilated convolution was originally designed for end-to-end semantic segmentation by aggregating contextual information at multi-scales [71–73]. As shown in Figure 3.1, in image processing problems, it considers 2-D discrete convolution: Let $F : \mathbb{Z}^2 \to \mathbb{R}$ be a 2-D discrete function with domain $\Omega_f = [-f, f]$ which can be viewed as the input image; and let $K : \mathbb{Z}^2 \to \mathbb{R}$ be another 2-D function with $\Omega_f = [-f, f]$, that can be viewed as the convolutional kernel, assuming $n < m$. The discrete convolution operator * can be expressed as

$$F * K(m, n) = \sum_{i=-r}^{r} \sum_{j=-r}^{r} F(m - i, n - j)K(i, j) \tag{3.1}$$

For the dilated convolution, the kernel function is changed to

$$K^d_{i,j} = \begin{cases} K_{i/s,j/s}, & \text{if } s \text{ divides } i \text{ and } j; \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

From Figure 3.1, with dilated convolution, the receptive field is expanded directly which can involve much more context information; and it is reported that via such way, the network can achieve better performance for semantic segmentation problems [71]. However, instead

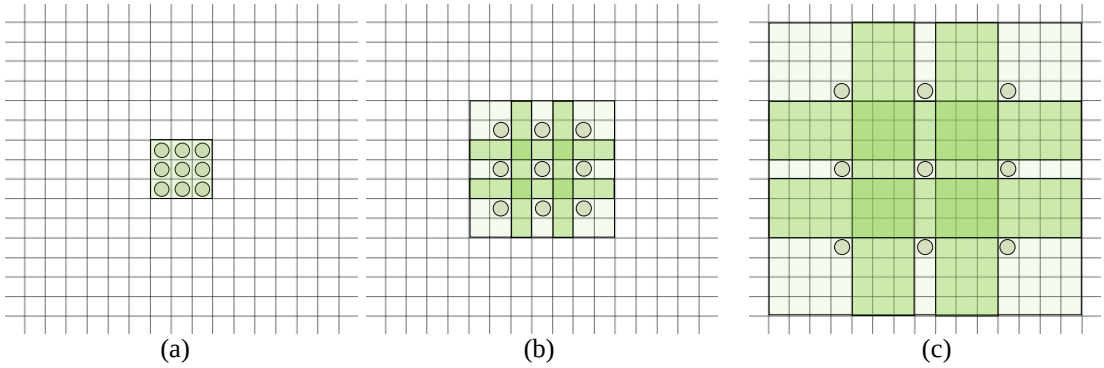


Fig. 3.1: Dilated convolution with kernels of 3×3: (a) without dilation; (b) 2-dilated convolution; (c) 3-dilated convolution. (Figure from [71]).

of emphasizing the significance of enlarging the context coverage in an end-to-end training mode by the original authors, our work presents that the dilated convolution is an equivalent implementation of the standard convolutions with dense inter-sampling under multi-layer CNN context, and that can be used to enable the transfer of middle level knowledge and facilitate the end-to-end training.

3.3 Proposed Method

In the early attempts, it was found that directly training a fully convolutional network (FCN) [69] for end-to-end per-pixel detection was not satisfactory. There were several reasons; the most possible ones were: 1) crack, as a thin-long target, can only occupy a very small area in the full image comparing to the BG. with the fact that patch-wise training is equivalent to loss sampling in FCN [69], directly applying FCN to pavement cracking images for pixel-level detection is actually operating on a heavily unbalanced training set. 2) the precise GTs of pavement crack images are inherently difficult to obtain since the cracks are thin and the crack boundaries are vague, that makes it impossible for per-pixel GT annotation; therefore, the community usually marks the cracks by 1-pixel curves and set them as the GTs. While such GT is more than welcome by the engineers because of the light labor-cost; however, they cannot perfectly match the actual crack at pixel-level that makes the computation inaccurate.

This work starts from training a classification network to classify image blocks of

227×227-pixel as cracks or BG. It is trained using the transfer learning, and the training set is composed of 40,000 image blocks of the two categories (20,000 instances for each). While the original classification network can only process image blocks with a fix size, consequently, the first task is to enable the network to work on full size pavement images; and this is realized by replacing the fully connected layer with fully convolutional layer in conventional DCNNs. While directly applying this classification network on full-size image can only produce a low resolution output with poor localization accuracy, the second task focuses on increasing the output resolution that will facilitate the refining later. The resolution decrease mainly comes from the operations working at the stride larger than one, including convolution and max-pooling; therefore, it reduces the strides, and introduces the dilated-layer to remedy the affection of stride changes. At last, an end-to-end network refining is conducted under a larger field of view to reduce the localization uncertainty and achieve more accurate detection results. Figure 3.2 is an overview of such design.

3.3.1 Training a classification network

Comparing with the task of high accurate crack localization, training the network as an image-block classifier is much easier. The classification can be viewed as a weakly supervised detection, and it plays the key for transforming the classification network into the final detection network. The data used to train the deep classification network are prepared following the data augmentation in Chapter 2: first, we manually marked 400 original pavement images of size 2000×4000-pixel; secondly, we sample the image blocks (227×227-pixel) of crack along the corresponding marked curves, the BG blocks are sampled on the locations far from the cracking curves. The distance d between two block-samples along the curves satisfies $d > w/2$, w is the side length of the sample block. Besides, it also employed image resizing and image rotating to increase data diversity.

Transfer learning is an effective method for training deep networks [56,58] and is utilized here. In essence, the DCNN involves a lot of filters which conduct convolution operations on various input images for feature extraction; however, each of the convolution kernels/filters actually operates on the entire image (or the entire feature maps) with the same format

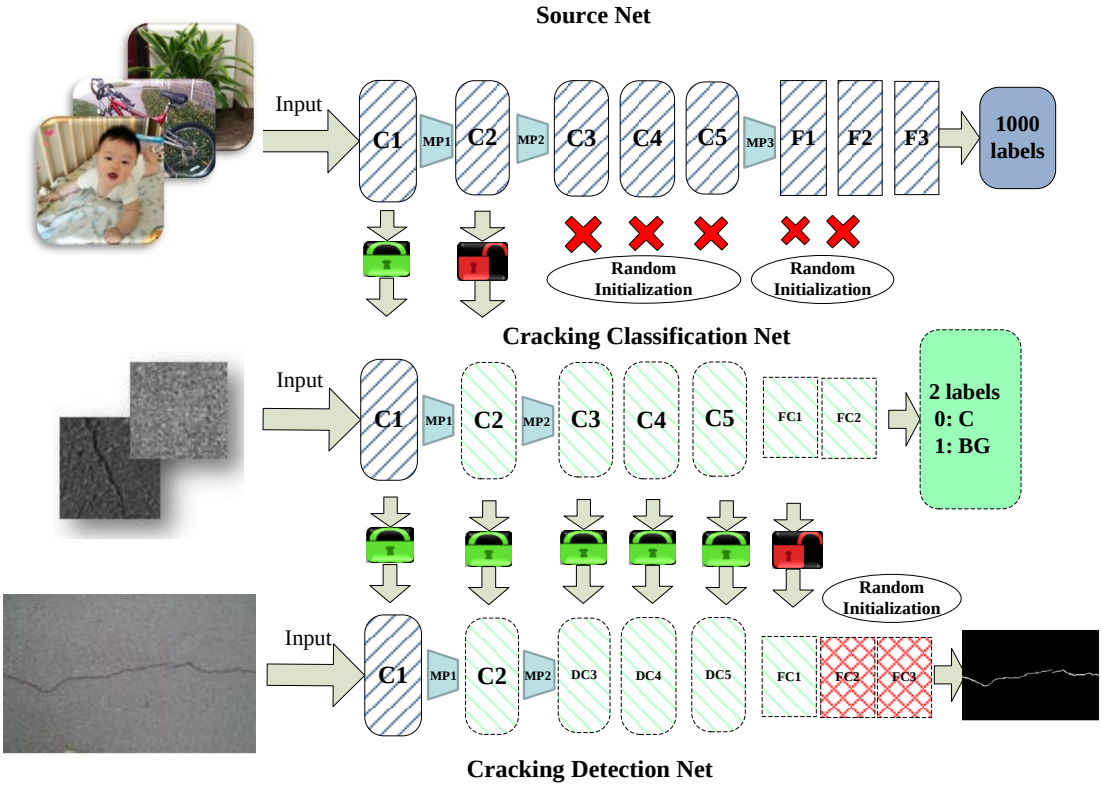


Fig. 3.2: Overview of the proposed approach: network at the top is the source net trained with ImageNet; network in the middle is the crack block classification net trained with transfer learning and network at bottom is the proposed dilation net for crack detection (C=crack; BG=BG).

during the training which makes the convolutional layer have more chance to extract the common features or knowledge. Specifically, for the lower convolutional layers, they usually learn generic knowledge that has strong transferability. Therefore, as in Chapter 2, a pre-trained model is used as the source network; and the generic knowledge from low level convolutional layers are transferred by initializing the parameters using the pre-trained model. The network involves five convolutional layers C1, C2, ..., C5 and three fully connected layers F1, F2 and F3; between C1 and C2, C2 and C3, and C5 and F1, there are three max-pooling layers, respectively. In the experiments, it re-implemented the network by: (1) transferring the generic knowledge by copying the weights of the first convolutional layer; (2) removing the max-pooling layer between C5 and F1; 3) Removing F3 and halving the output channels of C3, C4, C5, C6, F1 and F2 for parameter reduction. The weights of C2 are copied from the source net and retrained, weights of C3-C5, F1 and F2 are randomly initialized and trained. The inputs are image blocks with the labels of crack and BG. In the last layer, Softmax layer [36] is employed to calculate the loss, and send the gradient information back to the former layers for parameter updating via BP algorithm [60]. In this way, it can create a well-trained classification network.

3.3.2 Crack detection network

An ideal method using a classification network to perform detection is the patch-wise per-pixel labeling which has almost been abandoned due to its huge time complexity caused by the sliding window process that conducts the forward computation at every pixel location patch-by-patch. Researchers have tried to find a tradeoff by using region proposal and stride window-sliding [38, 39, 44, 68]; however, it is still very inefficient since the region based forward process has to be performed many times, especially for processing large images. In this Chapter, we propose a solution by introducing equivalent convolutional layers to replace the fully connected layers and designed an equivalent dense dilation layer to transform a classification network into an end-to-end detection network.

Working on full-size image with FCN: Fully connected layer requires a fixed input dimension and that has caused much inconvenience in dealing with object detection using

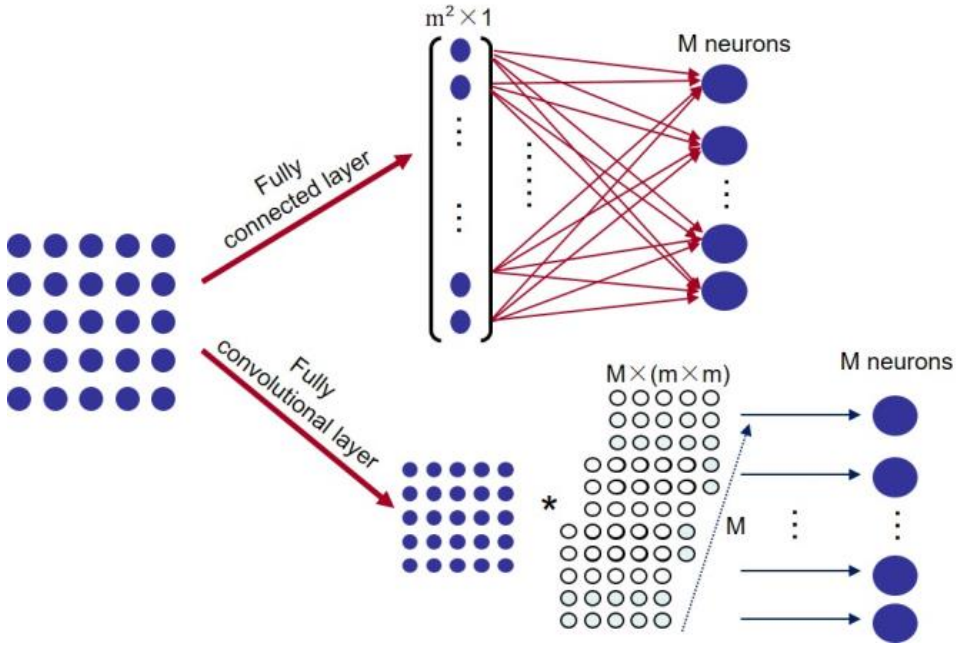


Fig. 3.3: Fully connected layer is a special case of the convolutional layer.

large input images. Fortunately, it is found that a fully connected layer can be replaced by an equivalent convolutional layer. As in Figure 3.3, assume that there was a layer with $m \times m \times 1$-dimension feature map as input, and the neuron number of the next layer is M. For the fully connected layer, the $m \times m$ feature map will be treated as an $m^2 \times 1$ vector with each element as an input neuron, and they are connected to every neuron in the next layer independently, refer to Figure 3.3. Then, for the equivalent convolutional layer, it can configure M convolutional kernels, each with $m \times m \times 1$-dimension, and conducts the convolutional calculation on the entire input respectively; and the output would be M-dimension feature map but each of the map is 1×1-dimension. It is a different implementation, but the function is the same, including input dimension, trainable parameters and output dimension. Here, the dimension property of a layer is defined as a vector with 3 elements representing input, trainable parameter and output dimension; D_{fconn} and D_{fconv} denotes the dimension of a fully connected layer and the dimension of a fully convolutional layer, respectively. Eqs. (3.3) and (3.4) indicate the dimension equality of the two implementations

as illustrated in Figure 3.3.

$$D_{fconn} = (m^2 \times 1, m^2 \times M, M) \tag{3.3}$$

$$D_{fconv} = (m \times m, M \times m^2, M) \tag{3.4}$$

In the classification network, the input of the first fully connected layer F1 is the feature map of 6×6×128 dimension from C5 ($m = 6$), and the output is 2048 neurons ($M = 2048$); i.e., we can apply 2048 convolution kernels of 6×6×128-dimension on the feature maps and that will produce 2048×1-dimension output which is a perfect match with the original F1 layer. For F2 layer, since the input has a 1×2048 dimension "feature-maps", it can configure a 1×1×2048-dimension kernel and set the output as 3-dimension vector that stands for the three categories. Please find the related network settings and the dimension changes in Table 3.1. Since the convolutional operation does not require a fixed input size, the modified network can work on the full image directly without retraining the parameters.

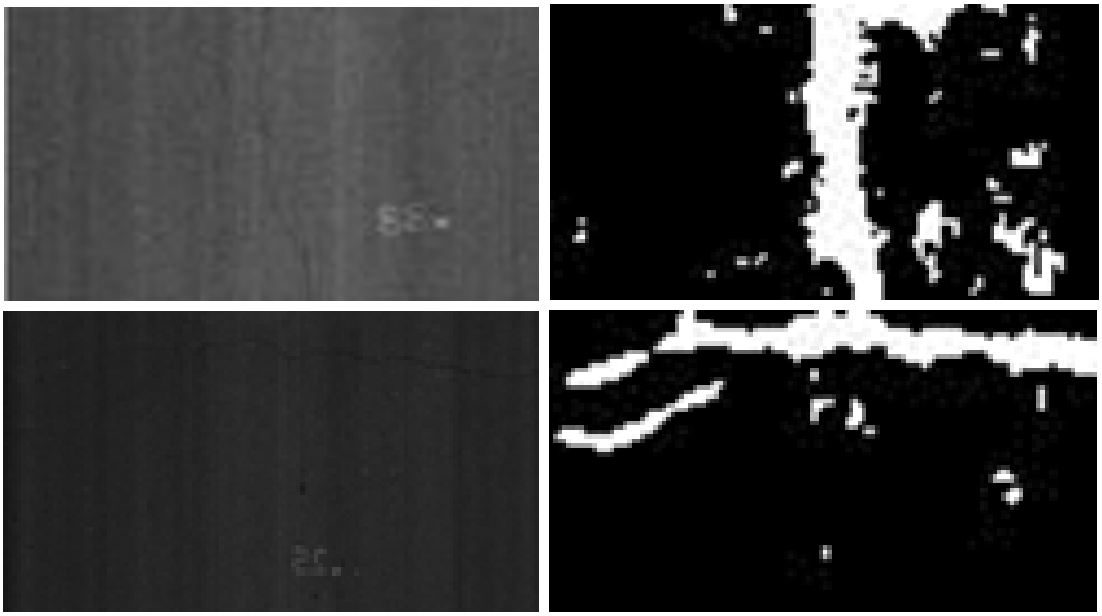


Fig. 3.4: Sample images and the outputs produced by Naive detection network. Left column: the original image. Right column: outputs of the Naive detection network.

Before implementing a detection network for large size images, we perform an analysis of the resolution changes under large input. Assume that a classification net is trained based on images with size $m \times m$-pixel, now we directly apply the convolutional network on images with sizes $M \times M$-pixel, where $M > m$. At the first glance, it tends to give us an illusion that the network realized a 227-time down-sampling process since it is fed a 227×227-pixel image and produces a classification label. If so, the classification net working on full-size image tended to produce an output with resolution $= (2000 \times 4000)/227 = 8.8 \times 17.6$. However, it is not true. As in Table 3.1, it got a 63×125 output. Indeed, under larger field of view, such FCN based design actually realized multiple-spot detection at different locations. The number of the spots is decided by the convolution and pooling with stride larger than one. Besides, such process on the full image does not involve redundant convolution operations other than patch wise pixel-labeling working in the sliding-window mode. While it only gives low quality output with very low resolution, it is called "Naive Detection Net", i.e., it needs to be improved. Refer Figure 3.4 for the result produced by the Naive Detection Network.

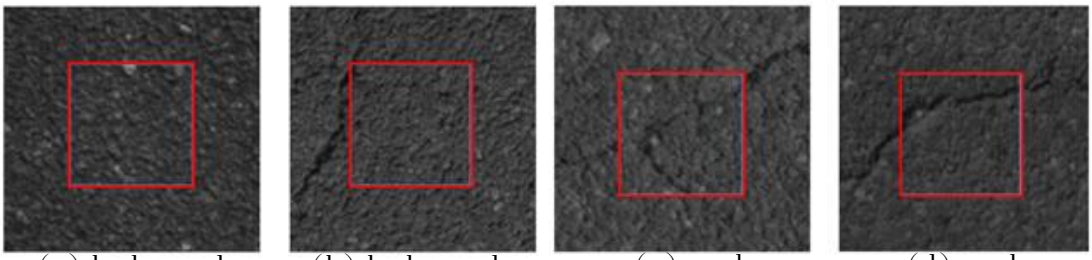


Fig. 3.5: Localization uncertainty of the classification network.

Localization uncertainty analysis: In addition to the resolution loss inherited from the classification net, the low localization accuracy in Figure 3.4 is also caused by the localization uncertainty of the classification net as in Figure 3.5. For classifying the image blocks, it is found that the network did a very good job when dealing with the blocks when a crack passing through the block center or the blocks without cracks. However, they become ambiguous for the blocks with cracks near the border, refer Figure 3.5 for the cracks located

outside the center-square areas. It is called localization uncertainty of the classification net; and if the localization uncertainty can be reduced, the detection performance will be improved.

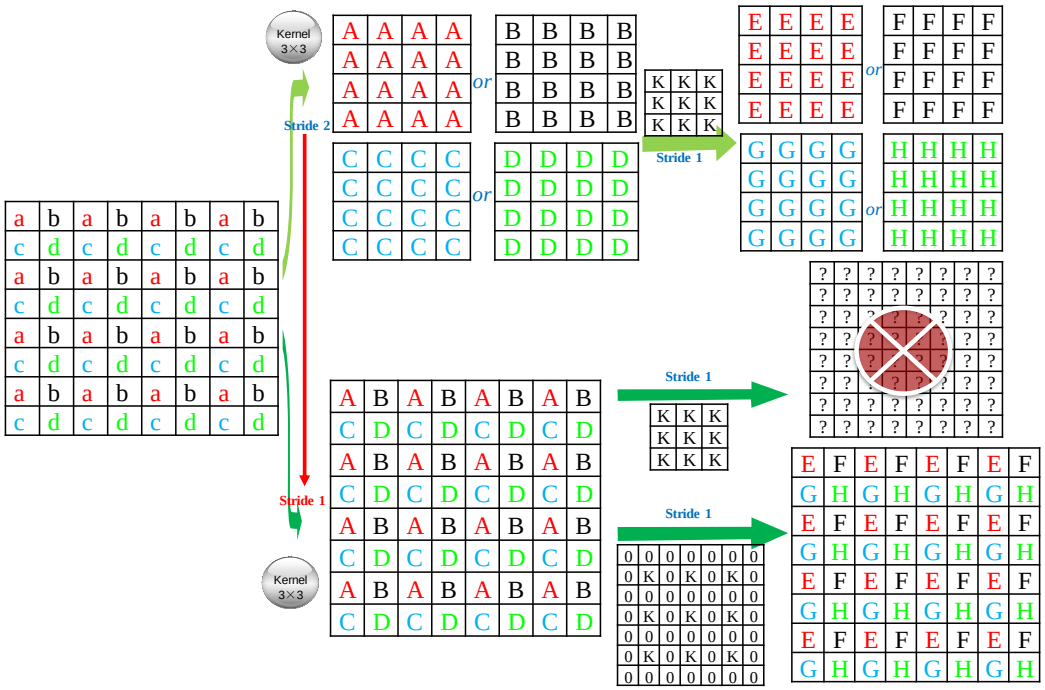


Fig. 3.6: Equivalent dilated convolution when stride changed.

Equivalent Dense-Dilation layers: In the early days of deep learning, researchers tried substantial efforts to make the network converge, such as max pooling [36], dropout [53], increasing convolution/pooling stride [36], etc. Among them, the large strides configured within convolution and pooling layers significantly reduce the memory cost by cutting down the output dimensions. However, it also discards a significant amount of information which is important to high resolution object detection. In this part, it presents the dilated convolution can be implanted into the classification net to reduce the resolution loss, and it will not damage the forward propagation pipeline of the original classification network, and the parameters/knowledge can be reused in the detection network. As in Figure 3.6, there is an input image for a network having two convolutional layers; the first layer conducts convolution on the input image with a kernel of 3×3 at stride 2. "A", "B", "C" and "D"

denotes that the outputs obtained by the convolutions operated at the corresponding locations “a”, “b”, “c” and “d”, respectively. Intuitively, with stride 2, the convolution will be conducted on locations centered with “a” or “b” or “c” or “d”, depending on the starting point (padding the related values for the elements near borders); the output is showed in Figure 3.6 top-middle. While the direct way to eliminate coverage loss is reducing the stride to one; however, it involves additional outputs which disturbs the output order, as shown in Figure 3.6. Obviously, we cannot continue the forward propagation with original kernels for a desired output in the subsequent layers. However, as shown in Figure 3.6, the messed outputs still have the information except for interpreting the additional values equal to the convolution results operated on the other starting points. In the next layer, if it interpolated 0s between each pair of elements of the original kernel and conducted the convolution operation at stride 1, it will output the same values as the original convolution operation, only with the additional interpolated items. It turns out that the information is just the ones at the missed locations under the large-stride operations that will improve the resolution. Besides, the same idea can also be generalized to any pooling layers by picking up the elements at those sparsely located spots and conducting the related pooling operations. In addition, once a stride reduction applied to a specific layer, all the subsequent layers should take the dilated operation at the related stride size; and the dilation stride sizes have to accumulate along with each reduction from the previous layers. For example, assume there were three convolution layers in a network with the convolutional stride sizes as 2, 2 and 1, respectively; if the stride from first layer is reduced to 1, the dilation strides of second and third layers will be set as 2; if both first and second layer strides are reduced to 1, the dilation sizes of second and third layers have to be 2 and 4. As an exception, if the convolution kernel is 1×1 with stride 1, such as FC2 in the classification network, the dilation is not necessary.

Based on the above discussion, it reduced the stride of MP2 to 1, and set the dilation levels of C3-C5 to 2; and reduced the stride of C5 to 1, set the stride of FC1 to 4 (note that we did not make change for C1, C2 and MP1 because they have relative small receptive fields

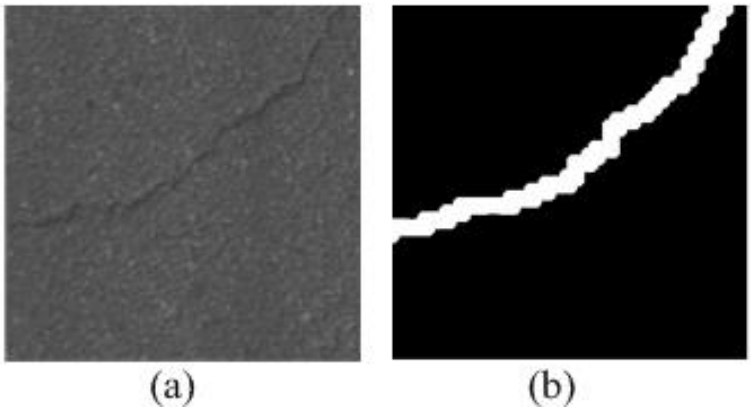


Fig. 3.7: Dilated GT. (a) An image block of 400×400 pixels; (b) dilated GT.

which will make little difference). Finally, it will produce the output of 250×500-dimension. The dilated network will not improve the performance of network radically since it is only an equivalent conversion of the classification net. However, the increased output resolution is important for making the proposed network converge under larger field of view.

3.3.3 Network refining under larger field of view

As discussed, the low quality outputs result from the low resolution and the localization uncertainty. In this part, we conduct an end-to-end refining process under larger field of view to reduce the localization uncertainty and improve the detection results. It is a fine-tuning strategy that will not retrain the parameters of C1-C5 which have learned rich knowledge in the classification mode.

Firstly, the larger field of view is set by utilizing images of 400×400-pixel as the training data which is determined experimentally. As mentioned at the beginning of the method part, training an end-to-end detection net with 1-pixel-width crack curve as GT is problematic due to the high risk of mismatch. Other than using the 1-pixel-width GT as the label, we use the k-dilated ground truth images as the label images ("disk" structuring element was used for the dilation) and it is as the "ground truth with-tolerance", see Figure 3.7. That is, as long as the detected crack pixel falls in the dilated range of the 1-pixel GT, then it will be treated as true positive. The dilation size also indicates the uncertainty level.

Based on such settings, it re-implemented the last layer FC2 by setting the convolutional kernels with 3×3 of 1024 channels and added another layer FC3 with kernels of size 1×1 and 3 output channels representing 3 categories. Then it performs the fine-tuning under larger field of view with images of 400×400-pixel and the related label images (note that the label images will be resized to 50×50 because there is an 8-time resolution loss from the first two layers that will lead to a 50×50 output dimension). During training, the parameters of C1-C5 will not change. Besides, only crack and sealed crack images are involved because the BG samples are taken into consideration automatically under the FCN mode [69]). Figure 3.8 are the detection results of the sample images in Figure 3.5. It can be seen that the refining procedure reduces the localization uncertainty level and the false positives are in much smaller sizes which could be further removed using simple denoising processes [20].

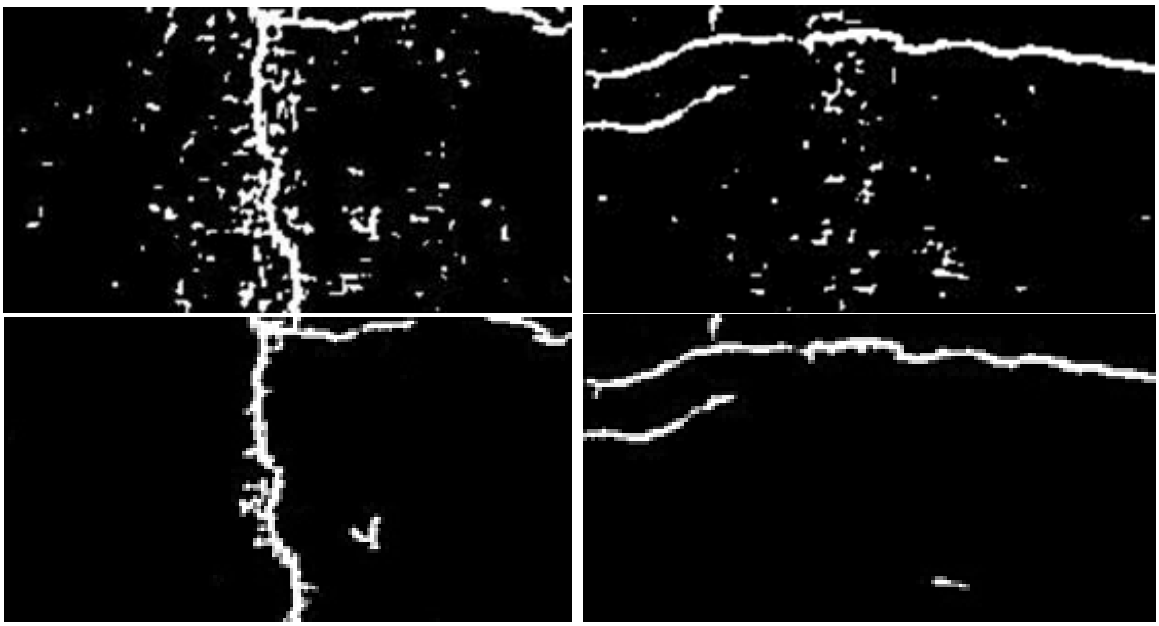


Fig. 3.8: Detection results of the images in Figure 3.5. First row: the outputs of the refined network. Second row: the final results after removing small noisy points.

3.4 Experiments

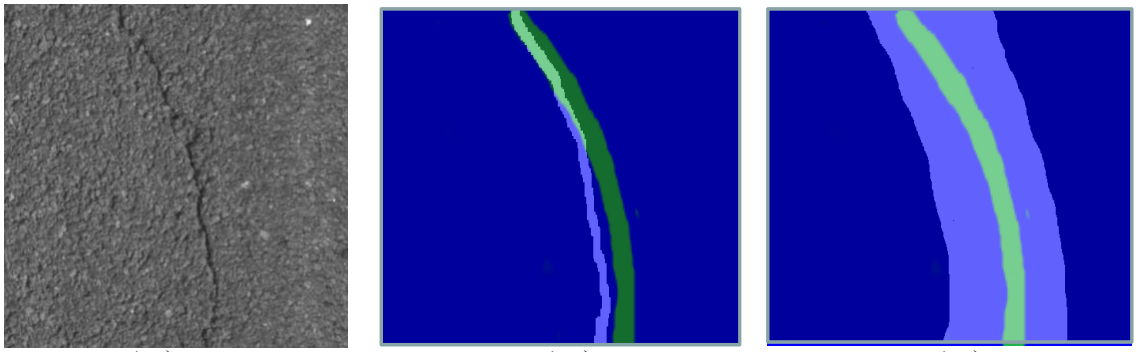


Fig. 3.9: Pixel-level mismatching: (a) a crack image; (b) detection result overlapped with the 3-width GTs image; and (c) detection result overlapped with the 12-width GT image. The transparent areas represent the n-dilated GT cracks with different n and the green areas represent the detected crack.

3.4.1 Dataset and metrics

The data are obtained with industry equipment, the camera is set on the top of a vehicle running at 100km/h which can scan 4.096-meter width road and produce an image of 2048×4096-pixel for every 2048 line-scans in 0.2048 seconds (i.e., 1 pixel represents 1×1 mm_2 road area). The cracks are usually thin, and it is difficult to precisely mark the ground truth at pixel level since the crack width is tiny and the boundary is vague; thus, the GTs are marked using 1-pixel/n-pixel curves representing the cracks (labor-light GTs). Such GTs may not match the true crack locations at pixel-level, processing of such images is more challenging. 400 low-resolution images from industry are used for the experiments. The training-test ratio is 2:1.

Different from most object detection tasks [80], the IOU (intersection over union) metric computed using the related regions is not suitable for evaluating the crack detection algorithms. As shown in Figure 3.9, crack, as a thin-long target can only occupy a very small area and the image consists mainly of BG pixels [81,82]; with the fact that the precise pixel-level GTs are hard to obtain for low-resolution pavement crack images, it is infeasible to obtain the accurate intersection area. For the second and third images in Figure 3.9, it is obvious that the detection results are very good; however, the IoU values for (b) and (c) are very low, 0.13 and 0.2, respectively. According to [82], it uses Hausdorff Distance for

the overall evaluation of the crack localization accuracy: For two sets of points A and B, the Hausdorff distance can be calculated with:

$$H(\mathrm{A},\mathrm{B}) = max[h(\mathrm{A},\mathrm{B}), h(\mathrm{A},\mathrm{B})] \tag{3.5}$$

where

$$h(\mathrm{A},\mathrm{B}) = max_{a\in\mathrm{A}} min_{b\in\mathrm{B}} \|a-b\| \tag{3.6}$$

The penalty if defined as:

$$h_p(\mathrm{A},\mathrm{B}) = 1/(|\mathrm{A}|)\sum_{a\in\mathrm{A}} sat_u min_{b\in\mathrm{B}} \|a-b\| \tag{3.7}$$

Here, u is the upper limit used to directly get rid of the false positives that are far away from the GTs. Instead of setting u as 1/5 of the image width [82], it is set as 50-pixel to emphasize the localization accuracy by eliminating the influence of possible noises from the large BG areas. Assume A is the detected crack set and B is the ground truth set, the overall score is:

$$score_{BH}(A,B) = 100 - \frac{BH(A,B)}{u} \times 100 \tag{3.8}$$

where

$$BH(A,B) = max[h_p(A,B), h_p(B,A)] \tag{3.9}$$

With Eq. (3.8), it can obtain the Hausdorff Distance score (HD-score) which can reflect the overall crack localization accuracy. The metric is insensitive to foreground-BG imbalance existing in most long-narrow object detection tasks.

It also introduces the region based precision rate (p-rate) and recall rate (r-rate) for the evaluation which can measure the false-detection severity and the missed-detection severity, respectively. As present in Figure 3.10, a crack image of 400×400-pixel is divided into small image patches (50×50-pixel); if there was a crack detected in a patch, marked as red "1s", it is a positive. In the same way, for GT images, if there is a marked crack curve in a patch, it is a crack patch. Then the TP_{region}, FP_{region} and FN_{region} can be obtained by counting

Table 3.1: Performance evaluation of the dilated FCN

Method	p-rate	r-rate	HD-score
Method in Chap. 2	**0.951**	0.847	95
Method in [11]	0.520	0.969	65
Proposed	0.921	**0.970**	**96**

the corresponding square regions, and further be used to calculate the p-rate and r-rate:

$$p_{region} = \frac{TP_{region}}{TP_{region} + FP_{region}} \tag{3.10}$$

$$r_{region} = \frac{TP_{region}}{TP_{region} + FN_{region}} \tag{3.11}$$

Then the region based F1 score ($F1_{region}$) can be computed as:

$$F1_{region} = \frac{2 * p_{region} * r_{region}}{p_{region} + r_{region}} \tag{3.12}$$

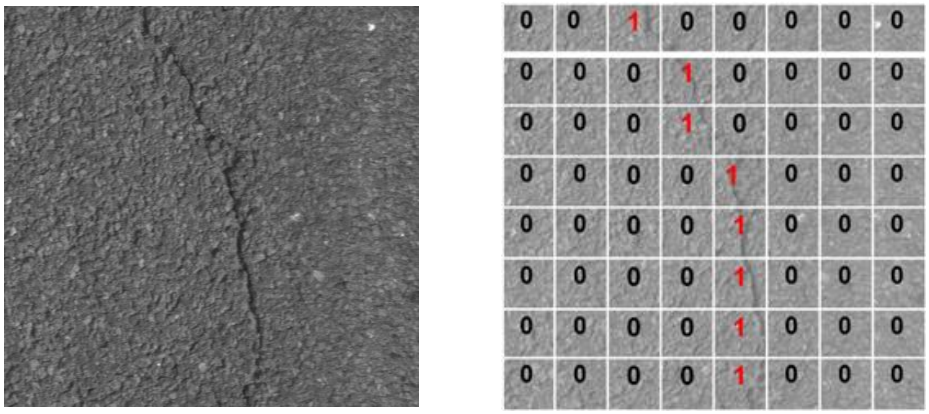


Fig. 3.10: Region-based evaluation for computing r-rate and p-rate: (a) the original crack image block; (b) an illustration of counting the crack and non-crack regions. The squares with label "1" are the crack regions, and with label "0" are BG regions.

3.4.2 Experimental results

The proposed method indeed is still a classification network. The only difference is it introduced a second stage, end-to-end fine-tuning, to improve the localization accuracy.

Therefore, we compare the method with the window-sliding based method in terms of HD-score and computation efficiency to validate the effectiveness of the proposed method. Method in [11] is a region based method and window-sliding is the strategy to perform crack detection on full-size images; it cannot accurately locate the cracks since it only based on the image block label for cracking detection, that results in the low HD-score of 65 and the low p-rate, 0.520. The time complexity is high, which takes 10.2 seconds to process an image with 2000×4000 pixels. Similarly, the method in Chapter 2 is also based on window-sliding, and is with complex post processing to improve the accuracy; it takes 12.3 seconds to process a full-size image. Instead, the newly proposed method conducts the forward computation only once for the detection without introducing redundant convolutions; and it is efficient and only takes 1.2 seconds and the localization accuracy is also very good. Refer Table 3.1 for the results.

3.4.3 Summary

In this Chapter, we propose a solution for pavement crack detection with large input image. It overcomes the computational efficiency issue of window-sliding based detection. Specifically, we the equivalent convolution layers were introduced to realize multi-spot dense detection without redundant convolution operations; and to overcome the convergence problem encountered in the end-to-end training, the equivalent dense-dilation layers were designed and implanted into the pre-trained classification network to enable the transfer of middle-level knowledge for transfer learning. The experiments demonstrated the effectiveness of the proposed approach on industrial pavement images. Moreover, the proposed approach also performed a study on the problem: what is the difference between a DCNN-based classification net and detection/segmentation net. It indicated that the detection and segmentation network is a classification network working on larger field of view with the receptive field as the input size (without using the up-sampling/decoding layers). In the end, it should be mentioned that the effectiveness of the end-to-end fine-tuning depends on the dilation scale of the GTs and the training image size; and it is easy to suffer from the imbalance problem if the parameters are not properly selected.

CHAPTER 4
CRACK DETECTION WITH GENERATIVE ADVERSARIAL LEARNING

4.1 Background

In Chapter 3, we have proposed an end-to-end crack detection network and developed the equivalent dense dilation layer for transfer learning which improved the crack localization accuracy. However, the training was unstable, and the image size and dilation scale of the GTs in the training were determined by trial and error. In the experiments, it was found that the network is easily to converge to the status that treated all the pixels as background (BG) which can still achieve a very good loss; we named it "All Black" phenomenon. Recently, Yang et al. [49] applied a FCN based network to crack detection and achieved very good results on the data collected from website. However, the method was suffering from the "All Black" issue when dealing with the images obtained from industrial settings due to the unavailability of precise GTs and data imbalance inherent in crack-like object detection. In this Chapter, we propose a solution for this problem by introducing an asymmetric U-shape network and crack-patch-only (CPO) supervised generative adversarial learning.

4.2 Related Works

4.2.1 Generative adversarial networks

Generative adversarial networks (GAN) [74] is an important progress of deep learning in recent years; and it proposes a novel learning strategy that can be used to train different models by conducting a max-min two-player game. The original GAN utilized two regular neural networks (multi-layer perception) to play a contest in terms of data distribution: a generator was trained to map from a latent space to a distribution of interest, while a discriminator was trained to distinguish between the data created by the generator and the data from the real distribution. Assume the generator is to learn a distribution p_g over

data x, and a prior distribution on input noise $p_z(z)$. Then the mapping from noise to the data space is $G(z;\theta_g)$, where G is the function representing the generator. Then, another neural network $D(x;\theta_d)$ gets a data with the same dimension as x and outputs a single scalar representing the probability of the real data rather than from a fake case. D and G are trained with the following value function $V(G,D)$:

$$min_D max_G V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{4.1}$$

From Figure 3.1, with dilated convolution, the receptive field is expanded directly which can involve more context information; and it is reported that via such way, the network can achieve better performance for semantic segmentation problems [71]. However, our work presents that the dilated convolution is an equivalent implementation of the standard convolutions with dense inter-sampling under multi-layer CNN context which instead can be used to enable the transfer of middle-level knowledge and facilitate the end-to-end training.

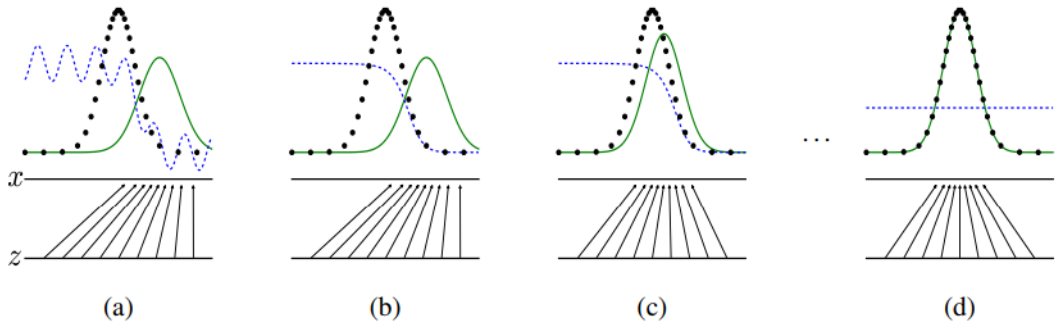


Fig. 4.1: The generative networks are trained to minimize the distribution difference between the real data and that from the generator. (Figure from reference [74]).

It has been proved that by conducting the minmax game, the global goal of the optimization is $p_g = p_{data}$. An illustration of this theory can be found in Figure 4.1 where the inputs are one dimensional data following some distribution as indicated by the black

dash curve, the green solid curve denotes the distribution of the generated data. It can be observed that the distribution of the generated data finally approached to the real data distribution. While GAN is known to be difficult to train, Radford et al. [75] proposed DC-GAN that replaces the fully connected layers with convolutional/de-convolutional layers in the generator, and it is trained to optimize the convolutional kernels to generate real-like images, which also makes the training easier and more stable.

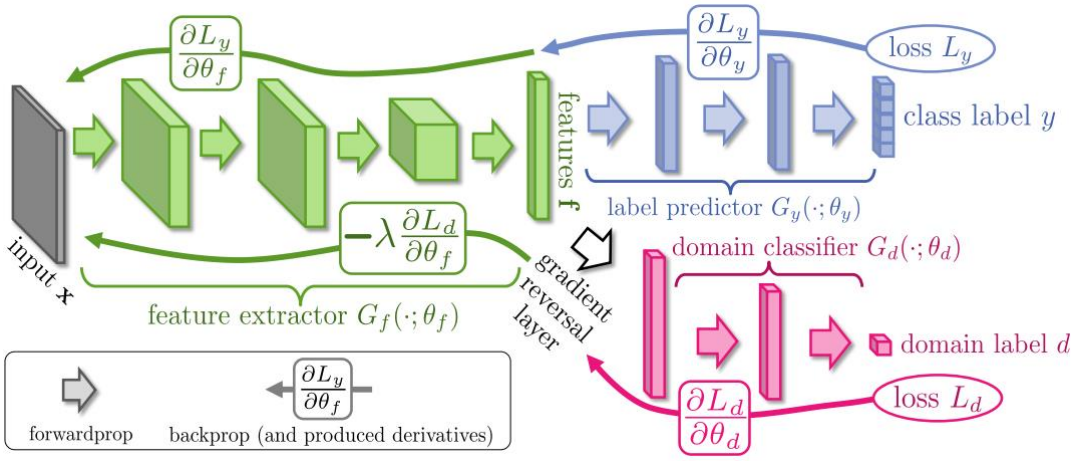


Fig. 4.2: Illustration of deep domain adaptation (Figure from reference [76])

4.2.2 Deep domain adaptation

Domain adaptation is a research field related to transfer learning. The algorithm is designed to transfer two or multiple domains to a common domain. With the development of deep learning, deep domain adaptation was proposed where deep neural networks are trained for domain adaptation [76]. The most common context is that only one of the domains has label while the data from other domains have no labels but only domain information. The goal is to transform the data from the no-label domains to the labeled domain so that the domain difference will not influence the performance. As illustrated

in Figure 4.2, a neural network is served as the feature extractor which encodes an input image to a feature vector; a classifier at the top is used to predict class label and another classifier at the bottom is used to output a value representing how likely is the data from that of the labeled domain. The networks are trained with back-propagation; however, the main difference is that the domain classifier will back-propagate the reversed gradient to update the parameters of the feature extractor so that the data from different domains cannot be distinguished at the feature space. In that way, the data will be mapped to the space with the same domain. In this Chapter, we use a pre-trained classifier to replace the discriminator in GAN for domain adaptation.

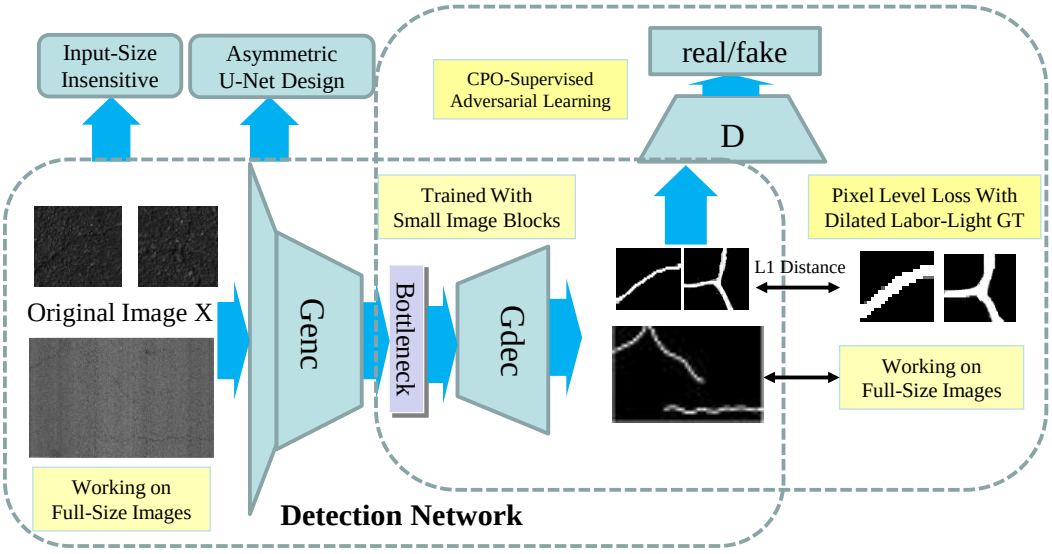


Fig. 4.3: Overview of the proposed method

4.3 Proposed Method

Figure 4.3 is an overview of the proposed method; it has three main novelties: (1) crack-patch-only (CPO) supervised adversarial learning; (2) weakly supervised pixel-level

loss obtained with dilated GTs; and (3) asymmetric U-Net design. In Figure 4.3, D is a pre-trained discriminator; it is obtained directly from the discriminator of a pre-trained DC-GAN, where the network is trained with crack-like GT patches only. Such pre-trained discriminator will force the network to always generate crack-GT, which is the most important factor to overcome the "All Black" issue. There, the pixel-level loss is employed to ensure that the generated crack patterns are the same as that of the input patch via optimizing the L1 distance based on the dilated GTs. Since the network is trained with crack-patch only, the asymmetric U-shape architecture is introduced to enable the translation abilities of both crack and non-crack images as detailed later; and it is implemented by adding two additional layers to the encoding part of the generator. After the training, the generator will serve as the detector to be used for crack detection; and it is designed as a fully convolutional network which can process full-size images after the patch-based training, as indicated in Figure 4.3. Finally the overall objective function is:

$$L_{final} = L_{adv} + L_{pixel} \tag{4.2}$$

where L_{adv} is the loss generated by the COP adversarial learning and L_{pixel} is the pixel-level loss obtained using the dilated GT images. The L_{adv} forces the network to generate crack images; the L_{pixel} directs the network to generate cracks at the expected locations; and the CPO supervision relies on the asymmetric U-Net.

4.3.1 "All Black" issue

As discussed before, directly training a FCN [69], including U-Net [77], for end-to-end per-pixel crack detection using industrial pavement images cannot achieve the expected results. It has been analyzed that the reason is that the cracks from industrial images are very thin and the precise-GTs are not available. When conducting the end-to-end training with FCN, the network will simply classify all the pixels as background and still achieve a quite "good" accuracy (background pixels dominate the whole images). We name this phenomenon "All Black" problem. Refer Figure 4.4 and Figure 4.5 for such phenomenon.

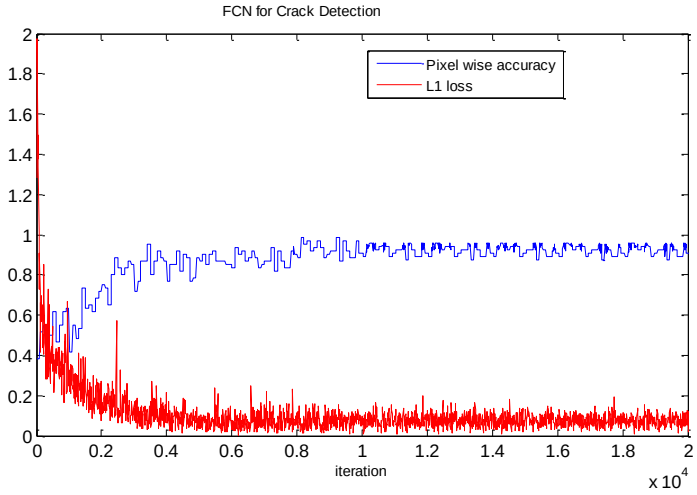


Fig. 4.4: The loss and accuracy curves under the regular FCN

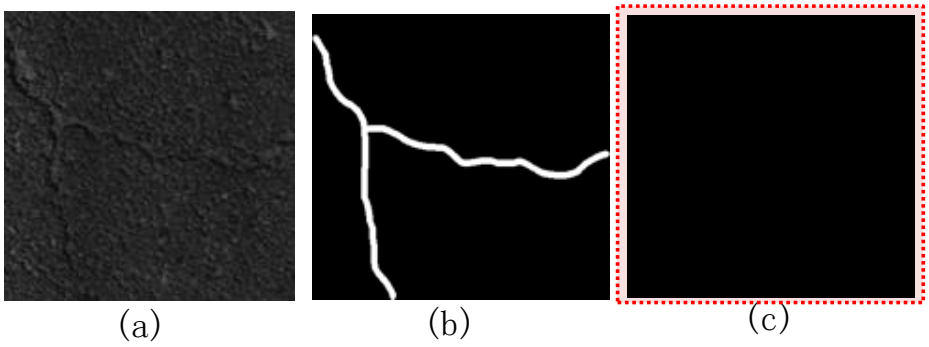


Fig. 4.5: "All Black" problem encountered when using FCN based method for pixel-level crack detection on low-resolution images: (a) the low-resolution image captured under industry setting; (b) the GT image and (c) the detection result with "well-trained" FCN (refer Figure 4.4).

Figure 4.4 shows that during the training, the loss decreased rapidly and reached to a very low value; however, as shown in Figure 4.5, the detection result of a crack image is all black (background). Moreover, it is worth to mention that similar phenomenon is also reported in [49] where the FCN based-method fails to detect thin cracks; here, it only takes the U-Net as an example.

4.3.2 CPO-supervision

Regular FCNs only produce all-black image as the detection result. In order to solve such problem, we add another constraint, generative adversarial loss, to train the detection network that forces the network to always generate crack-like detection result. It employed DC-GAN [75] to realize such goal. It is well-known that DC-GAN can generate real-like images from even random noise by conducting the training with a max-min two-player game. As has been verified [74], it is better for G to maximize $log(D(G(z)))$ instead of minimizing $log(1 - D(G(z)))$. Therefore, the actual optimization strategy is to optimize the following two objectives [78] alternatively:

$$\max_D V_D(D, G) = E_{x \sim p_d(x)}[logD(x)] + E_{z \sim p_d(z)}[log(1 - D(G(z)))] \tag{4.3}$$

$$\max_G V_G(D, G) = E_{z \sim p_d(z)}[log(D(G(z)))] \tag{4.4}$$

While our target is to force the network to produce crack-like images and overcome the "All Black" issue, it only uses the crack-GT patches as the real samples for the training. Since the output of the detection network (i.e., generator) serves as the "fake" image, the adversarial loss is:

$$L_{adv} = -E_{x \in I}[logD(G(x))] \tag{4.5}$$

where x is the input crack patch, I is the dataset with crack patches, G is configured with the asymmetric U-Net as discussed later, and D is the discriminator. The crack-GT patches are augmented by manually marking a bunch of "crack-curve" images and the discriminator is obtained from a pre-training a DC-G, refer to Figure 4.6. Instead of generating full-size

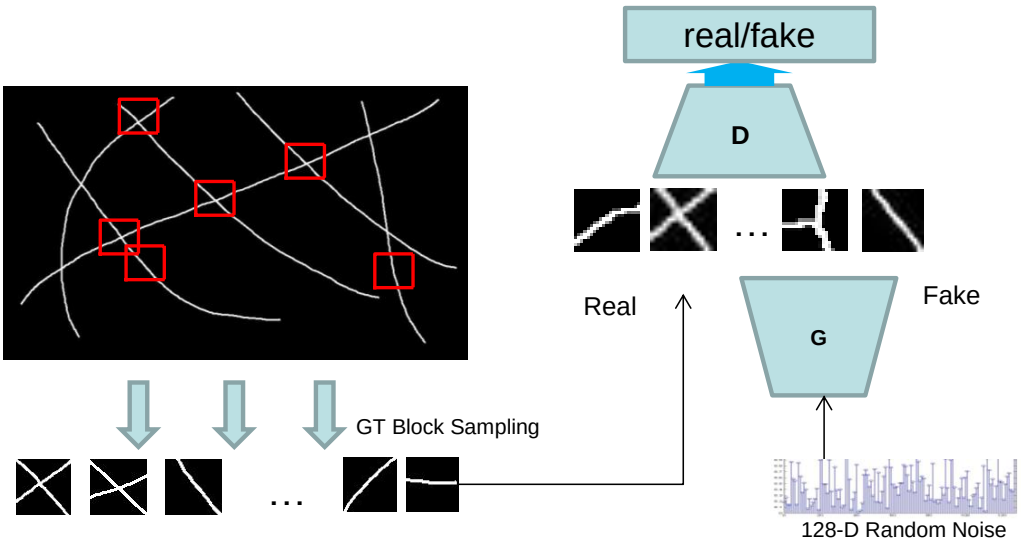


Fig. 4.6: Pre-train a DC-GAN with augmented GT images based on CPO-supervision: the real GT set is augmented with manually marked "crack" curves to ensure the crack patterns have high diversity.

images, the network is trained to generate GT-like image patches; and such network is easy to train and it can work on larger size images seamlessly.

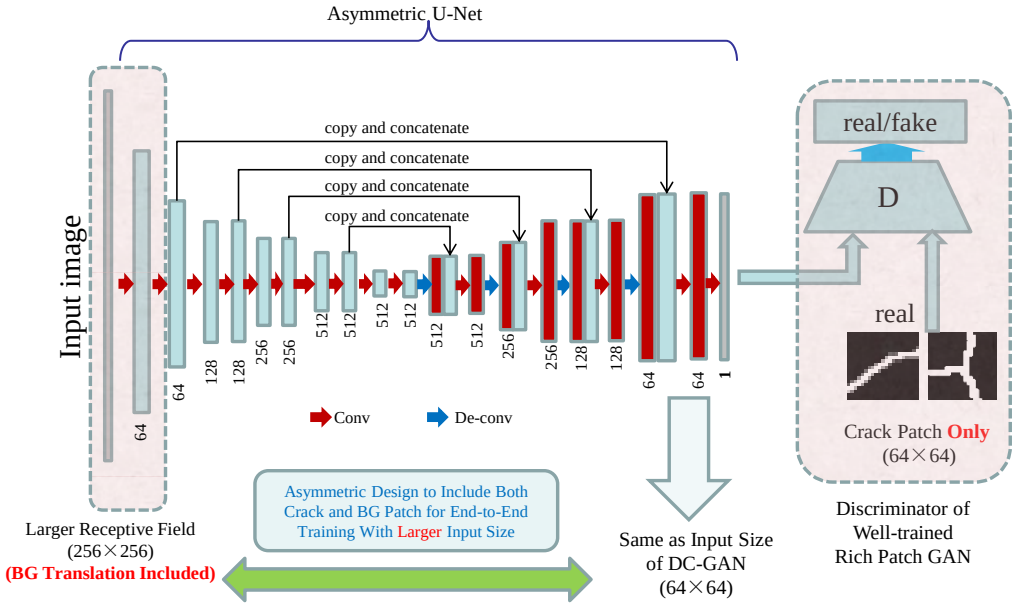


Fig. 4.7: Asymmetric U-Net architecture under larger receptive with CPO-supervision.

4.3.3 Asymmetric U-Net generator

The crack detection is formulated as a conditional image generation problem. In the last subsection, it introduced the CPO-supervised adversarial learning to force the network to always generate GT-like crack patches and overcame the "All Black" problem. It should be noticed that the discriminator only recognizes crack-GT patches as "real" without considering the background patches. Normally, the discriminator should also treat "all black" patches as real for the background (BG) translation; however, treating "all black" patches as real will encourage the network to generate all black images as the detection results which cannot solve the "All Black" issue. In order to include the translation of background

samples, we propose the asymmetric U-Net architecture that inputs a larger size crack image (256×256) to the asymmetric U-shape network (A-U-Net) while outputs a smaller size image (64×64) for the end-to-end training. The larger input image has to be a crack image so that the correct output will always be a crack-like patch recognized by the discriminator as real; but it includes the translation of both crack and BG samples, refer to Figures 4.7 and 4.8.

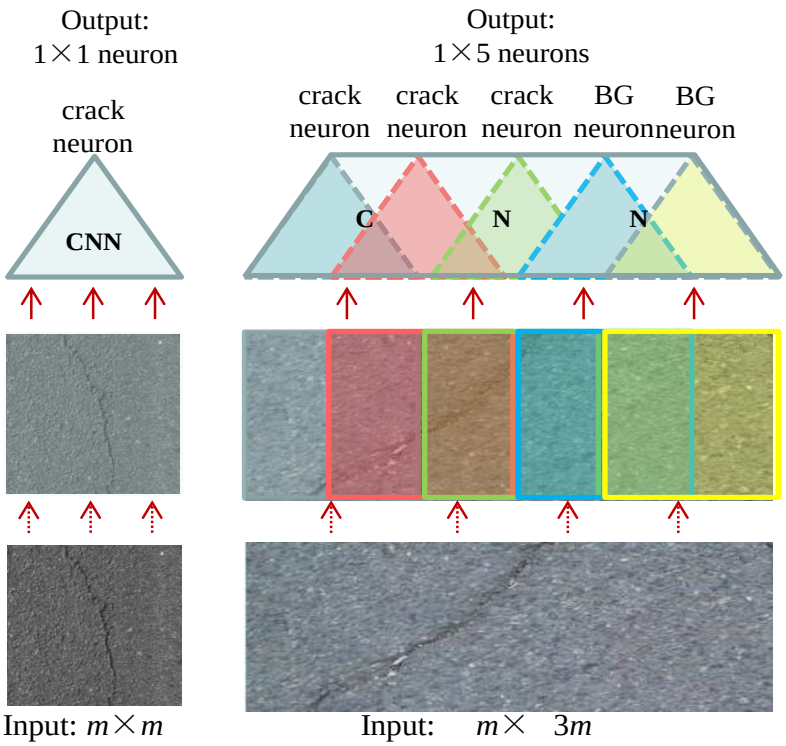


Fig. 4.8: With a larger input image, the CNN realizes multi-spot sampling with the same receptive field. At the right side, the first three neurons represent three crack samples while the last two are background (BG) neurons.

Receptive field under larger field of view: To better understand how the asymmetric design is able to include the BG translation ability by using crack samples only, we first

perform a receptive field analysis under larger field of view. In Figure 4.8 left, assume that there is a DCNN network, such as a classification network, with an $m \times m$ image patch as input, the output is a single neuron representing the class label of the input patch. When the same DCNN is fed a larger size input image (i.e., $M \times M, M > m$), it will output multiple neurons (the number of neurons depends on the down-sampling rate of the DCNN); each neuron represents a class label of the corresponding image patch with size $m \times m$ "sampled" from the larger input image. As shown in Figure 4.8, when the network's input is an image of $m \times 3m$, the output consists of five neurons which represent class labels of the five image patches including both crack and non-crack samples with size $m \times m$ (from left to right, the first three are neurons representing crack patches and last two are neurons representing background labels). Indeed, under the multi-layer fully convolutional mode, each neuron actually has a receptive field with specific size; since the convolutional layer is input-size insensitive, operating the network under larger receptive field actually realized a multi-spot image sampling with the image size equal to the receptive field of the neuron. Thus, when performing an image translation using a deep convolutional neural network with a larger input image, the process is equal to translating multiple smaller image samples at the same time (the size is equal to the actual receptive field of the image translation network).

Based on the analysis, we design an asymmetric U-Net which inputs a crack image with the size larger than the input-size of the discriminator, then it is passed through the network to produce a down-sampled image patch to match the input-size of the discriminator so that it can be used by the discriminator to produce the adversarial loss for generative adversarial learning. While the discriminator only treated crack-GT patch as real, it will force the network to generate crack patch as the detection result to prevent the "All Black" issue. Since the network is trained to translate a larger crack image to a down-sampled crack image, it includes the translation of both crack and non-crack image patches. In this way, the network can be trained to translate both crack and background images for crack detection. Refer to Figure 4.7 and Figure 4.8.

4.3.4 L1 loss with dilated GTs

It introduced the CPO-supervised adversarial loss and the A-U-Net to force the network to generate crack GT patches; however, it was image-level weak supervision that does not specify where the crack should appear in the generated image. What we want is to provide more information so that the network knows where the generated cracks should be. As mentioned before, training a pixel-level end-to-end detection network using 1-pixel crack curves (labor-light GTs) has high risk of encountering the mismatching problem. Instead, it specifies a relatively larger crack area to make sure it covers the actual crack locations, and if the detected crack fell into the specified area, it gives a positive reward. The experiments demonstrated that by combining the two kinds of supervision information, CPO adversarial leaning and the loosely specified crack areas, the network can generate cracks at the expected locations. Specifically, it dilated the GT thrice using a disk structure with radius 3. Following [5], it marked the GT image with 1-pixel-width crack and sampled crack patches and crack-GT patches from the original crack images and the marked GT images, respectively. Then the 1-pixel-width crack curves in the sampled crack-GT patches are dilated and they are used to provide the weakly supervised pixel-level loss:

$$L_{pixel} = -E_{x \in I, y}[\|y - G(x)\|_1] \tag{4.6}$$

where x is the input crack image; y is the dilated-GT image; I is the dataset of larger size crack patches (256×256 comparing with the output size of 64×64) used for end-to-end training; G is the A-U-Net and D is the discriminator. Overall, the final objective function is:

$$L_{final} = L_{adv} + \lambda L_{pixel} \tag{4.7}$$

The pixel loss is normalized during training and λ can be determined via grid search. Figure 4.9 shows the detection results of some sample images. Moreover, once the training is finished, the discriminator is no longer needed and the A-U-Net itself will serve as the detection network which translates the original image to the result image.

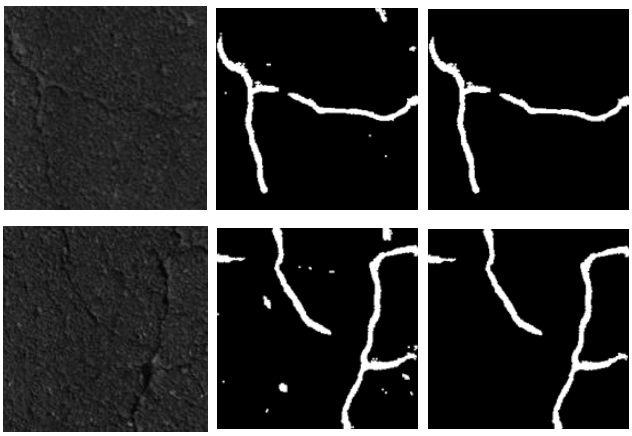


Fig. 4.9: Detection results of the final A-U-Net. First column: image samples from industry. Second column: the outputs of the network. Third column: the final results after removing the isolated noise areas.

4.3.5 Working on full-size images

Notice that the network is trained with small image blocks; however, under industry settings, the image size is much larger, such as pavement images with 2048×4096 pixels. A common solution to process large image is to sample smaller image blocks from the full-size image and process patch-by-patch, named window-sliding strategy [11]; however, it is inefficient [7]. Since the asymmetric U-Net is designed as a fully convolutional network and the crack patterns are scale insensitive, it can work on arbitrarily larger size images which realizes the same function as window-sliding sampling [7]. In addition, such fully convolutional processing mechanism does not involve redundant convolutions as discussed in [7]. Refer experiments part for the detection results of directly applying the A-U-Net on full-size pavement images.

4.3.6 Implementation details

Network architecture: Figure 4.7 illustrates the architecture of the A-U-Net. It sets the convolutional kernel of the first layer as 7×7 kernels with stride 2 followed by a rectified

linear unit (ReLU) [51]; then it applied a 3×3 convolutional layer with stride 2 followed by a ReLU layer. These two layers serve as the asymmetric part of the A-U-Net which realizes a 4-time down-sampling and reduce the larger input image to the output feature maps with the same size as the output of the A-U-Net. Then the remaining layers of the encoding and decoding parts are set up according to the regular U-Net architecture [77]. For the remaining layers of the encoding part, it consists of four repeated two-convolutional layers with 3×3 kernels and stride 2; and each convolutional layer is followed by a ReLU layer. After each of the first three convolutional layers, it doubles the number of convolutional channels. For the decoding part, it consists of four 3×3 de-convolutional layers that up-samples the feature maps; the input of each deconvolution layer is the output of the last layer with the corresponding feature map from the encoding part, then followed by a regular convolutional layer. After the last de-convolutional layer, another regular convolutional layer followed by a Tanh activation layer [35] is used to translate the 64-channel feature maps to the 1-channel image and it is compared with the dilated GT image for L1 loss computing according to Eq. (4.5). In summary, the network architecture is as follows. The encoding part:

C_64_7_2 - ReLU - C_64_3_2 - ReLU - C_128_3_1 - ReLU - C_128_3_2 - ReLU - C_256_3_1 - ReLU - C_256_3_2 - ReLU - C_512_3_1 - ReLU - C_512_3_2 - ReLU - C_512_3_1 - ReLU - C_512_3_2 - ReLU

The decoding part:

DC_512_3_2 - ReLU - C_512_3_1 - ReLU - DC_256_3_2 - ReLU - C_256_3_1 - ReLU - DC_128_3_2 - ReLU - C_128_3_1 - ReLU - DC_64_3_1 - ReLU - C_64_3_1 - ReLU - C_1_3_1 - Tanh

Here, the naming rule follows the format: "layer type_channel number_kernel size_stride"; "C" denotes convolution; "DC" is de-convolution; and Tanh is the Tanh activation layer. For instance, "C_64_7_2" means that the first layer is a convolutional layer and the number of channels is 64, the kernel size is 7 and the stride is 2.

Network training: The training is a two-stage strategy which employs transfer learning at two places: it pre-trains a DC-GAN and reused the generator; and it also pre-trains the

encoder part of the generator with a classification network. The DC-GAN is trained with the augmented GT patches of 64×64-pixel, aiming at training a discriminator with strong crack-pattern recognition ability to provide the adversarial loss at the second stage. A total of 60000 crack patches with various patterns are used, the cracks are dilated twice using a "disk" structure with radius 3. The other settings follow [75]: the Adam optimizer [79] is used, the learning rate is set as 0.0002, the parameters for momentum updating are 0.9, batch size is 128 and the input "noise" vector is with 128 dimensions. A total of 100 epochs (each epoch is total images/batch size = 60000/128 iterations) are run to obtain the final model. Then the well-trained discriminator was concatenated to the end of A-U-Net to provide the adversarial loss at the second stage, refer Figure 4.7 and Eq. (4.5).

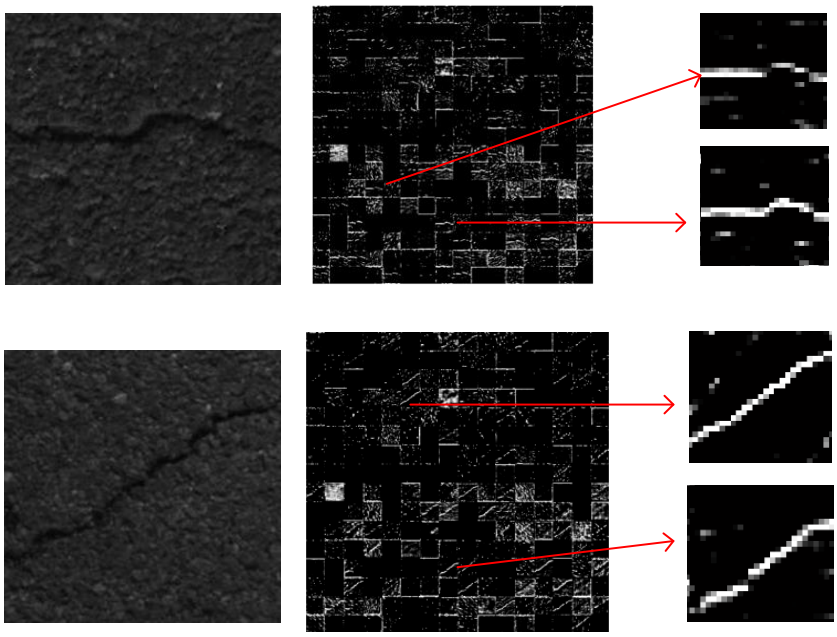


Fig. 4.10: Weakly supervised learning is able to learn crack pattern information. Left side: the image blocks sent to the classification network. Middle: the feature maps after the first convolutional layer. Right side: the feature maps with the similar crack patterns to the original image blocks.

Inspired by [5], it also pre-trains the encoding part of the generator under the classification setting. Zhang et al. [5] showed that by performing an image-block classification task, the network can still extract the relevant crack patterns; and the learned knowledge could be transferred to ease the training of an end-to-end detection network. Figure 4.10 is the low-level feature maps of a classification network trained with crack and non-crack patches; the classification network is configured by adding a fully connected layer at the end of the encoding part (bottleneck) of the A-U-Net and set the output dimension as two representing the crack/non-crack with label 0/1. The training samples are crack and non-crack patches of 256×256. It shows that the network extracted the same crack pattern as the original image which means that the network is able to learn useful information with the weakly supervised information, crack/non-crack image labels only. Then the well-trained parameters are used to initialize the corresponding encoding part of the generator at the end-to-end training stage; and the other settings are the same with the DC-GAN except replacing the generator with the asymmetric U-Net and setting the objective function with Eq. (4.7).

4.4 Experiments

4.4.1 Dataset and metrics

The proposed method is compared with six crack detection methods on two crack datasets: the public dataset CFD [23] captured with a cellphone and another dataset collected by the authors using a industrial line-scan camera. CFD is a public dataset with pixel-level GTs. It consists 118 pavement crack images of 320×480-pixel captured using an iPhone5 with focus of 4mm, aperture of f/2.4 and exposure time of 1/134s. The images were obtained by people standing on the road and holding an iPhone near the road. The image quality is high and the backgrounds are smooth and clean, and the pixel-level GTs are carefully marked. As mentioned in Chapter 3, the cracks from industrial images are usually thin, and it is difficult to mark the GTs at pixel-level since the crack width is tiny and the boundary is vague. Thus, the GTs are marked using 1-pixel curves representing the

Table 4.1: Quantitative evaluations of CrackGAN on CFD dataset

Methods	p-rate	r-rate	F1-score	HD-score
CrackIT	88.02%	45.11%	59.65%	21
MFCD	80.90%	87.47%	84.01%	85
CrackForest	85.31%	90.22%	87.69%	88
[11]	68.97%	**98.21%**	81.03%	70
FCN [49]	86.01%	92.30%	89.04%	88
U-Net	88.01%	90.02%	89.01%	90
CrackGAN	**89.21%**	96.01%	**91.28%**	**95**

Table 4.2: Quantitative evaluations of CrackGAN on industrial dataset

Methods	p-rate	r-rate	F1-score	HD-score
CrackIT	89.10%	2.52%	4.90%	9
CrackForest	31.10%	98.01%	47.22%	63
[11]	69.20%	**98.30%**	81.22%	64
FCN-VGG [49]	0.00%	0.00%	N/A	N/A
U-Net	0.00%	0.00%	N/A	N/A
CrackGAN	**89.21%**	96.01%	**91.28%**	**95**

cracks (labor-light GTs). While GTs may not match the true crack locations at pixel-level, processing of such images is more challenging. In the same way, the region based p-rate, r-rate, and HD-score are used as the metrics for the evaluation.

4.4.2 Overall performance

The proposed method is compared with CrackIT-v1 [83], MFCD [84], CrackForest [23], method in [11], FCN-VGG [49] and U-Net [77]. CrackIT and MFCD are state-of-the-art methods using traditional image processing techniques; and CrackForest is state-of-the-art machine learning method before deep learning appears. References [11], FCN-VGG [49], and U-Net [77] are deep learning methods. [11] is a block-level detection method based on the window-sliding strategy; FCN-VGG [49] is trained with the carefully marked pixel-level GTs (labor-intensive) for pixel-level detection; and U-Net is also the pixel-level detection method based on fully convolutional strategy. Figure 4.11 and Table 4.1 are the detection results on CFD; and Figure 4.12 and Table 4.2 are the results on the industrial images. It can be observed from Figure 4.11 and Figure 4.12 that CrackIT missed most crack

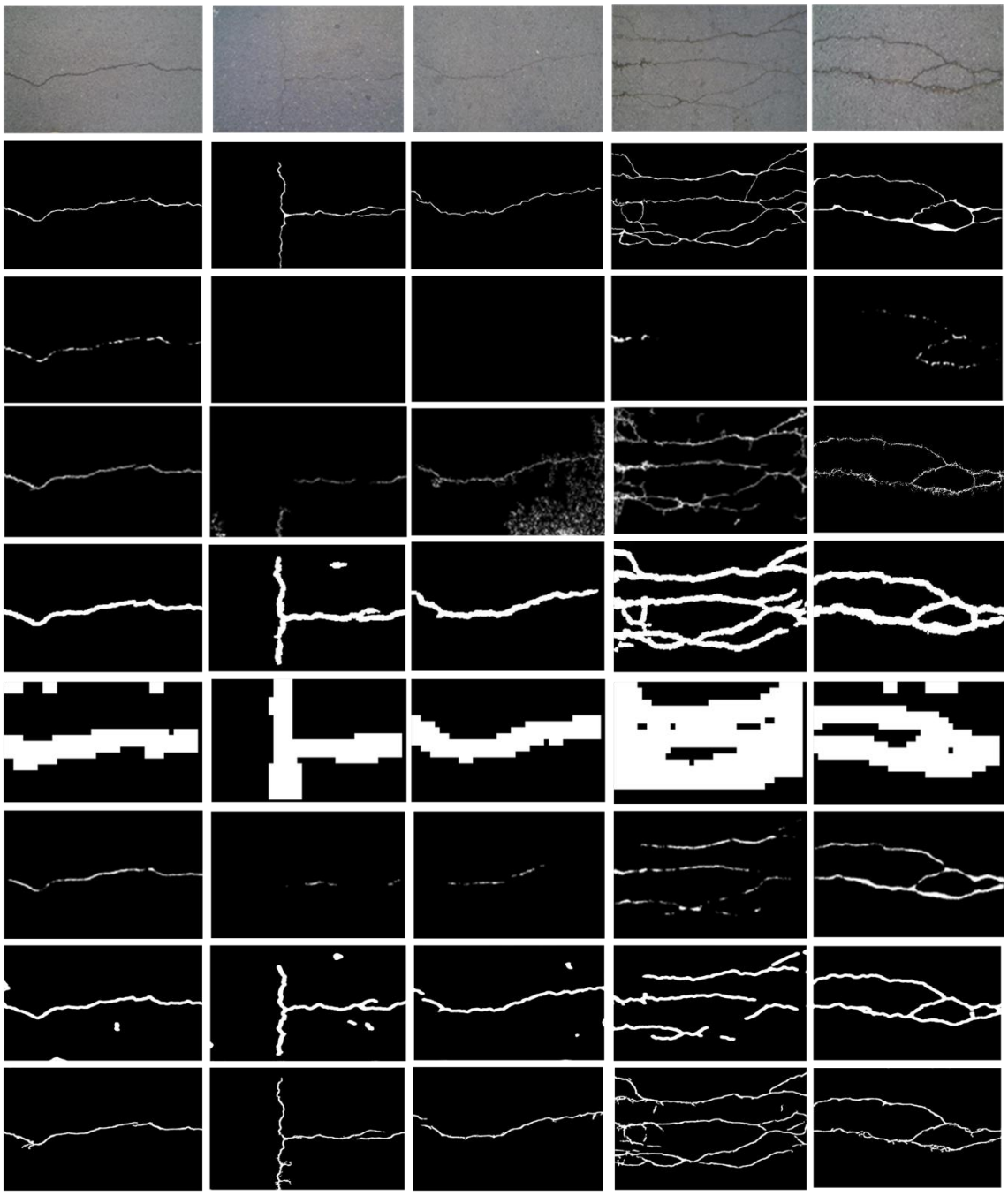


Fig. 4.11: Comparisons on CFD. From top to bottom are: original images, ground truth images and the detection results of CrackIT, MFCD, CrackForest, [11], FCN-VGG, U-Net and CrackGAN, respectively.

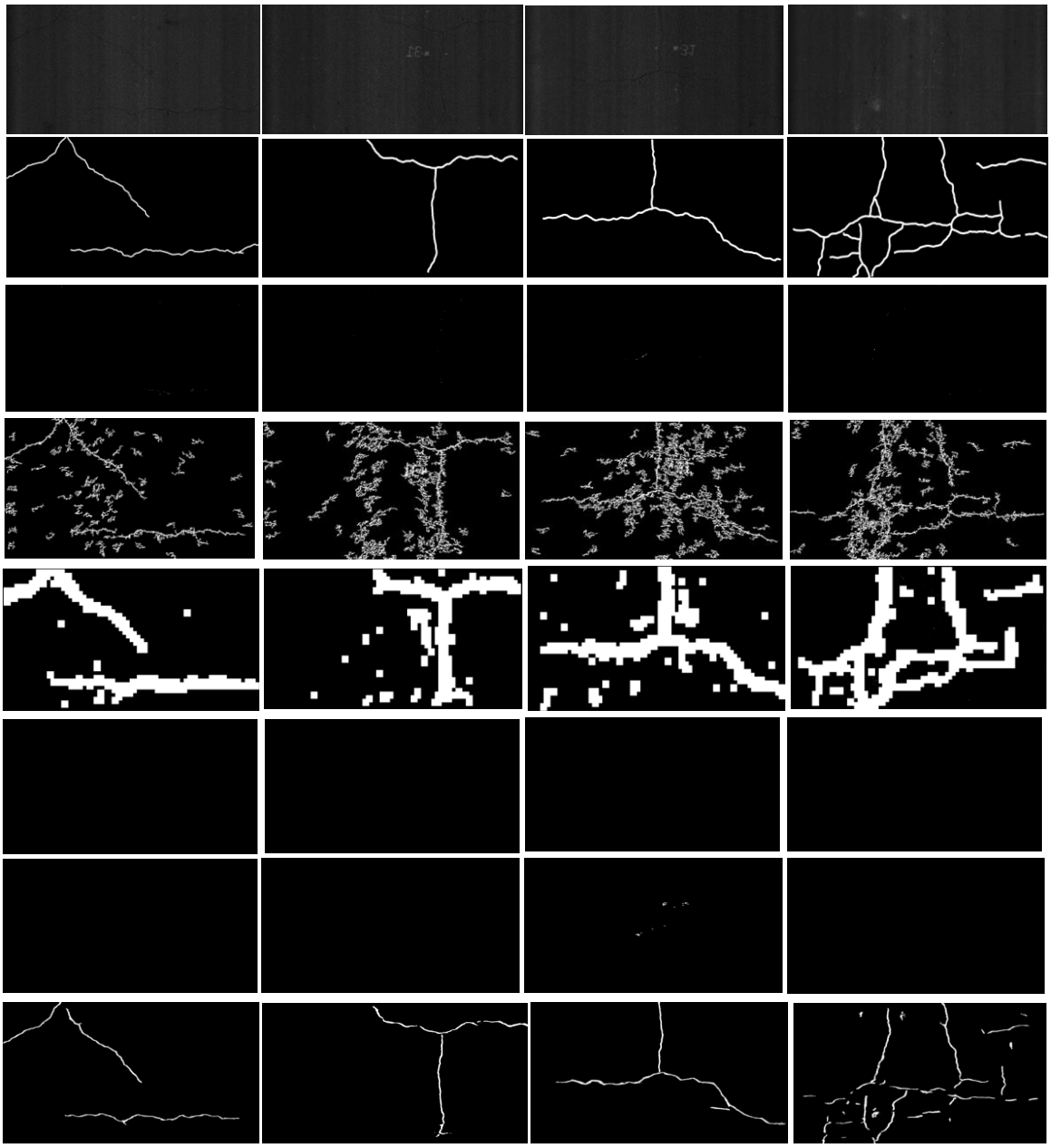


Fig. 4.12: Comparisons on the data from industry. From top to bottom are: original image, ground truth images and the detection results of CrackIT, CrackForest, [11], FCN-VGG ("All Black"), U-Net ("All Black") and CrackGAN, respectively.

Table 4.3: Comparisons of computational efficiency

Method	Time	Method	Time
CrackIT	6.1 s	FCN	2.8 s
CrackForest	4.0 s	U-Net	2.2 s
[11]	10.2 s	**CrackGAN**	**1.6 s**

pixels and cannot even detect any cracks in the second and third images where the cracks are thin; and it also fails to detect the cracks from the industrial images in Figure 4.12. Therefore, it achieves very low recall rate, 0.4511, and a low HD-score, 21. For MFCD, it has good detection ability and can detect most of the cracks on CFD; however, it did not solve the noise problem well and achieved relatively low precision rates and HD-score. CrackForest can most of the cracking locations and achieves good recall reates 0.9022 and 0.9801 on CFD and industry data, respectively; however, it is not good at noise removal on the industry images due to the complicated pavement textures. For the region based deep learning approach [11], it cannot accurately locate the cracks since it only produces the image block labels and obtained the low p-rates on both datasets, 0.6897 and 0.6920, respectively. The FCN-VGG and the U-Net [77] performed very well on the CFD dataset because the cracks are clear. However, as shown in Figure 4.12 seventh and eighth rows, they encountered the "All Black" issue when dealing with the industrial images due to the data imbalance and unavailability of precise pixel-level GTs. The proposed CrackGAN overcame the "All Black" problem and achieved very good performance on both CFD and the industrial datasets. Table 4.1 and Table 4.2 show the quantitative results.

4.4.3 Computation efficiency

In addition to the crack localization accuracy, it also compared the computation efficiency. The average processing times of the methods dealing with a large size image of 2048×4096-pixel is shown in Table 4.3. CrackIT-v1 [83] takes 6.1 seconds based on a patch-wise processing; and CrackForest [23] takes a relative less time (4.0 seconds) via using the parallel computing to implement the random forest for image patch classification. The above three methods are conducted with Matlab 2016b on HP 620 workstation with 32G

memory and twelve i7 cores. For the remaining deep learning methods, they are conducted on the same computer but running on the Nvidia 1080Ti GPU with Pytorch. Method in [11] takes 10.2 seconds since it uses the window-sliding method. FCN-VGG [23], U-Net [77] and CrackGAN take much less time due to the fully convolutional network architecture; moreover, the CrackGAN takes less time (i.e., 1.6 seconds) than U-Net because it cuts off the last couple of de-convolutional layers for the asymmetric U-Net design.

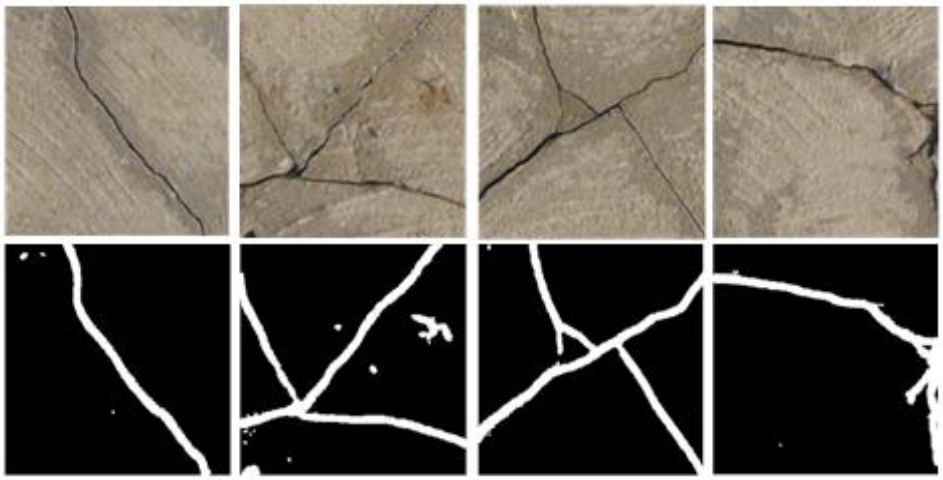


Fig. 4.13: Detection results on concrete wall and concrete pavement images. First row: original concrete wall images. Second row: corresponding results of CrackGAN. Third row: original concrete pavement images. Fourth row: the corresponding results of CrackGAN

In addition to the pavement crack detection using public dataset CFD and the industrial images, the proposed method is also able to handle other crack detection problems. Figure 4.13 provide the detection results on dataset [23] which contains concrete pavement images and concrete wall images.

4.4.4 Summary

In this Chapter, we have proposed a novel deep generative adversarial network, named

CrackGAN, for crack detection using labor-light GTs. The proposed method solved a practical issue, "All Black" problem, existing in deep learning based pixel-level crack detection methods. It proposes an asymmetric U-Net architecture and a CPO-supervised generative adversarial learning strategy to generate expected crack patterns and implicitly enables both crack and background image translation abilities to overcome the data imbalance problem. The experimental results demonstrate the effectiveness of the proposed method, especially for dealing with pavement crack detection from the industrial settings.

CHAPTER 5
SELF-SUPERVISED STRUCTURE LEARNING FOR CRACK DETECTION

5.1 Background

Through Chapter 1 to Chapter 4, we have proposed a suit of algorithms for high accurate crack localization and the problem has been well-solved with supervised deep learning. However, it should be noticed that these methods rely on a reasonable amount of paired training data obtained by making non-trivial GTs manually; that is expensive, especially for pixel-level annotations; and they also face a common problem, overfitting, which appears as poor model generalizability, and that makes the actual workload much heavier because it needs to re-mark the pixel-level GTs and re-train the model for new data. Thus, it is worth to rethink the automatic crack detection by taking into account the labor-cost for preparing GTs. As a pre-exploration of the future AI-based crack detection system, in this Chapter, we discuss a self-supervised structure learning network which can be trained without using paired data, even without using GTs. It is achieved by training an additional reverse network to translate the output back to the input, simultaneously. First, a labor-free structure library is prepared and set as the target domain for structure learning. Then a dual-network is built with two generative adversarial networks (GANs): one is trained to translate a crack image patch (X) to a structural patch (Y), and the other is trained to translate Y back to X, simultaneously. The experiments demonstrate that with such settings, the network can be trained to translate a crack image to the GT-like image with similar structure patterns, and it can be used for crack detection. The proposed approach is validated on four crack datasets and achieves comparable performance with that of state-of-the-art supervised approaches.

5.2 Related Works

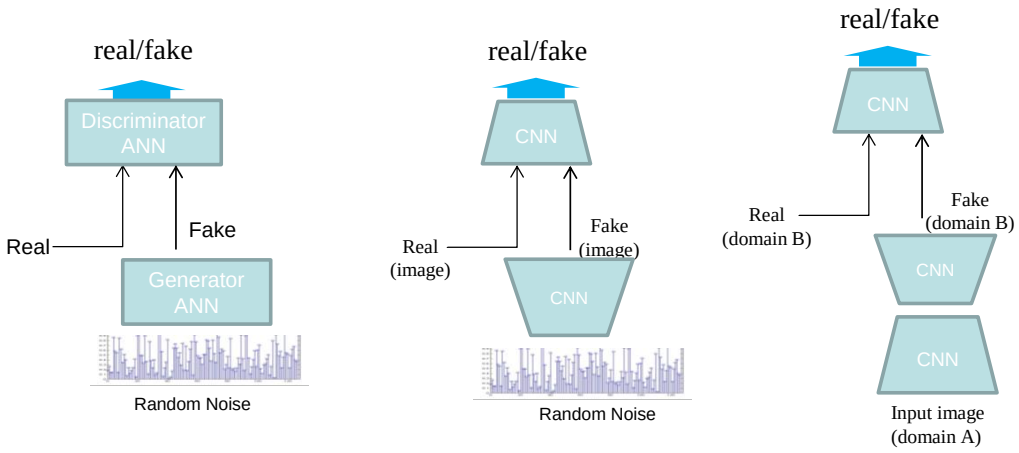


Fig. 5.1: Architectures of three different GANs: (a) Original GAN; (b) DC-GAN; (c) imageto-image translation GAN.

5.2.1 Image-to-image translation GAN

The original GAN configures the generator with fully connected layers which can generate images from random noises. While the original GAN is difficult to train, Radford et al. [75] proposed DC-GAN which replaced the fully connected layers with convolutional/deconvolutional layers to set up the generator; and it was trained to optimize the convolutional kernels to generate real-like images where the training became much easier. In order to make GAN more controllable, Mirza and Osindero [85] proposed conditional GAN which used additional information in the inputs so that the network could be trained to generate specific outputs according to the input condition. Based on conditional GAN and DC-GAN, Isola et al. [70] set up the condition with an input image that made the network become an image-to-image translation network trained with paired images that broadened the application contexts of supervised GAN. Recently, Zhu et al. [86] proposed the cycle-consistent adversarial networks that could be trained without using paired data for image style transfer. Based on these works, we formulate crack detection as an image-

to-image translation problem, and introduce the cycle-consist adversarial learning to train the networks without using manually labeled GTs. Refer Figure 5.1 for different GAN architectures.

5.2.2 Structure learning

Structure learning tries to learn a mapping function between inputs and structure patterns which can represent the inputs; it has been used to address computer vision problems. For example, Kontschieder et al. [25] utilized structure learning to assist semantic image labelling; Dollar et al. [24] employed structure learning for edge/contour detection. Since cracks exhibit strong structural patterns, such as straight lines, cross shapes, T-junctions, Y-junctions, etc., structure learning is also used for crack detection [24]. For instance, Shi et al. [23] introduced structure learning for crack detection where a random decision forest is trained to map the crack patch to a structure patch. These works train a cluster to group the structure patches into a limited number of representative patterns; then, with the paired image-GT data, a random forest was trained to find the mapping relationship from input images to structure patches. Different from these works, we use the cycle-consistent generative adversarial learning to find the source-target mapping relationship between two domains automatically; and the target domain (structure library) is obtained without manually marking GTs.

5.2.3 Self-supervised learning

Instead of spending a lot of time on preparing GTs, unsupervised/self-supervised learning receives increasing attention in recent years. While there is no clear line between unsupervised learning and self-supervised learning, two main settings are used in existing self-supervised learning: one is to learn features using an unsupervised pre-text task, and then applies the learned knowledge to other tasks [87]; and another is to introduce external data and relates the images to the introduced data for supervising the training. Doersch et al. [88] used the similarity measurements among randomly cropped image patches from a full-size image to supervise the learning for context prediction; Noroozi et al. [89] trained

a CNN to solve Jigsaw puzzle problem to provide self-supervision, and the real task was to learn visual representations. Sohn et al. [90] introduced unsupervised domain adaptation for face recognition from unlabeled video frames where still labeled images were used to pre-train the network. The method performs the learning by using a dual-network that translates the output back to the input for self-supervision; and as an assistant factor, it introduces a labor-free structure library for structure learning which makes the forward network generate consistent structure patterns as the original cracks for detection.

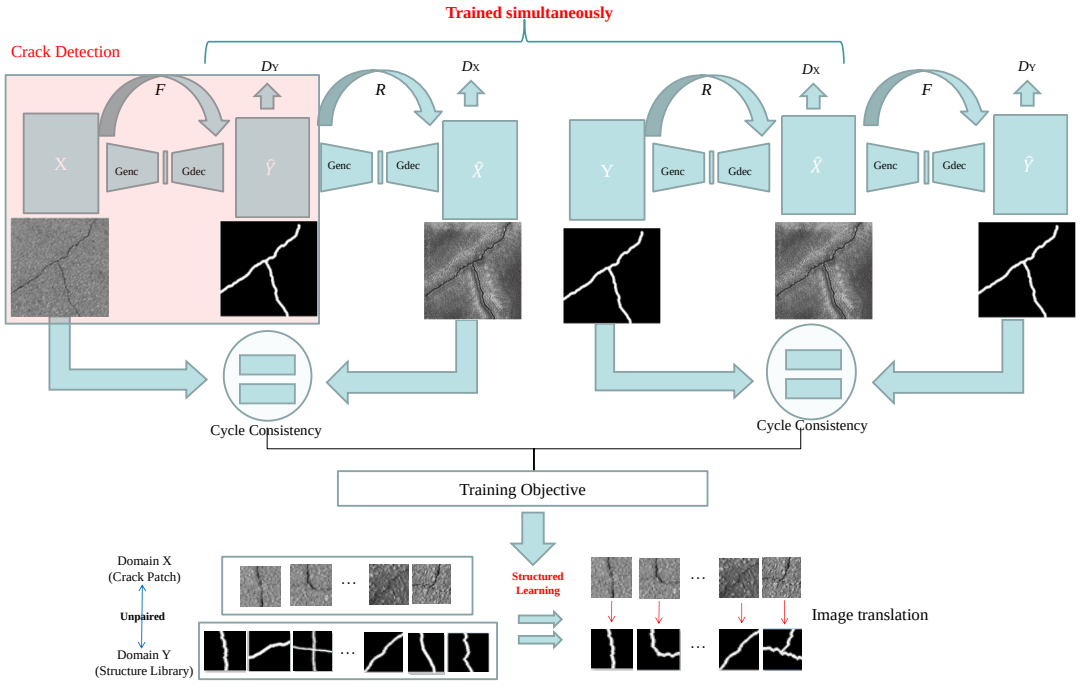


Fig. 5.2: Overview of the proposed approach with cycle-GAN

5.3 Proposed Method

In this section, it details the proposed method. The target is to learn an image-to-image translation/generation network that can translate crack image to GT-like image with

similar structure pattern, i.e., using structured curves to represent cracks. Two separate datasets/domains are needed to train the network: a crack-image set X and a structure library Y with images $\{x_i\}$ and $\{y_i\}$. As in Figure 5.2, the network consists of two image-to-image translation GANs: a forward GAN F translates images from X to Y (F: X$\sim$Y), and a reverse GAN R translates image from Y to X (R: Y$\sim$X). Two discriminators D_x and D_y are introduced: D_x aims to distinguish between images $\{x_i\}$ and the generated/translated images $\{R(y_i)\}$ with forward adversarial loss L_{advf}, and D_y aims to distinguish between $\{y_i\}$ and $\{F(y_i)\}$ with reverse adversarial loss L_{advr}; and D_y is designed as a pre-trained one-class discriminator working under larger field of view to succeed the training. Meanwhile, two cycle-consistent losses with L1-distance formulas are used in the forward GAN denoted as $L1_{fc}$ and in the reverse GAN denoted as $L1_{rc}$. Overall, the objective function is:

$$L = (L_{advf} + L_{advr}) + \lambda(L1_{fc} + L1_{rc}) \tag{5.1}$$

where λ controls the relative weight between the adversarial loss and the cycle-consistent loss which can be determined by grid search. In addition, the networks are designed as fully convolutional networks trained with small image patches; however, it can process images with arbitrary sizes.

5.3.1 Data preparation

The training data consist of two parts: the crack patches and a structure library. The structure library contains a large number of binary image patches of various structure patterns; the crack patches are prepared by cropping crack blocks of 64×64 randomly from the full-size crack images according to [5]. Different from other learning-based detection approaches [23, 49], no GTs were involved. Instead, a labor-free structure library is employed to assist training. First, a bunch of GT-like images are collected from the public datasets. In this work, object boundary annotations of VOC object detection dataset [80], object contour annotations of Berkeley segmentation dataset [91], and GTs of public crack detection datasets [19, 49] are used. As shown in Figure 5.3, the annotations are the curves

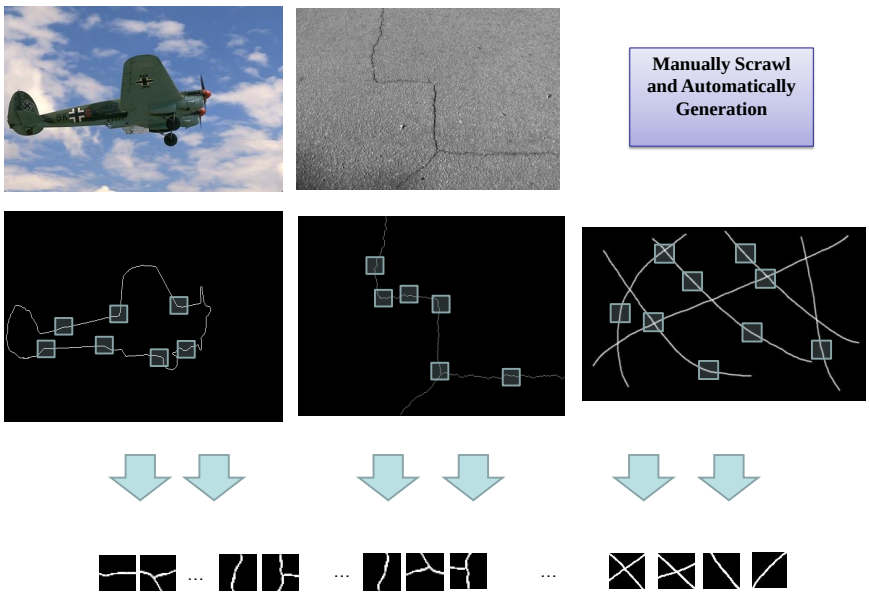


Fig. 5.3: Preparing a labor-free structure library

representing boundaries of objects or cracks; the structure library consists of image patches of 64×64 sampled from the annotations. The sampling process is: it first skeletons the annotations as 1-pixel curves, and then samples image patches of size 64×64 along the curves. To increase the diversity of the structure patterns, the distance d between each pair of sampled patches satisfies $d > s/2$, and s is the side length of the squared structure patch. Then the data are further augmented using image rotation and distortion [5]. As in [5], the structure patches are dilated using a disk structure with radius equal to 3.

5.3.2 Objective

As present in Figure 5.2, it introduces a dual-network and performs training by checking the consistency of both forward and backward cycles. The objective function contains two parts: the adversarial loss and the cycle consistency loss. The adversarial loss helps the detection network to generate GT-like structure images, and the cycle consistency loss makes sure that the generated structure patterns are consistent with the input cracks.

Adversarial loss: It is well-known that by conducting a max-min two-player game, the generative adversarial network can be trained to generate real-like images even from noise. In practice, it is realized by optimizing the following two objectives alternatively [78]:

$$\max_{D} \underset{D}{V}(D,G) = E_{x\sim p_d(x)}[logD(x)] + E_{z\sim p_d(z)}[log(1-D(G(z)))] \tag{5.2}$$

$$\max_{G} \underset{G}{V}(D,G) = E_{z\sim p_d(z)}[log(D(G(z)))] \tag{5.3}$$

where D is the discriminator; G is the generator; z is the noise vector inputting into the generator; and x is the real image in the training set. G generates images $G(z)$ similar to images from X, and D aims to distinguish between generated samples $G(z)$ and real samples x. D tries to maximize Eq. (5.2) while G is tries to maximize Eq. (5.3); therefore, the operation of G is equivalent to performing minimization of Eq. (5.2), which would result in a competition between G and D, i.e., the adversarial learning. The original GAN [74] configured the networks with fully connected layers was hard-to-train. In our case, the generator is an encoding/decoding U-Shaped network which inputs an image and generates another image (as the fake image); thus, z is a crack patch from the training data which serves as the input condition (refer to conditional GAN) that makes the generator produce a specific image related to the input, and x is image from the structure library as the real sample. In the same way, the reverse network translates a structured image, including the generated structured image by the forward network, to a crack patch for adversarial learning. The adversarial loss can help generator to create structure images via requiring the outputs to have identical distribution as the training data. However, the structure library may have different distribution with the expected GTs in terms of the structure patterns. Thus, different from the original cycle-GAN, the adversarial loss of the forward GAN is provided by a pre-trained DC-GAN whose discriminator is reused, serving as a domain adaptor. Refer network training section for details.

Cycle-consistent loss: While the adversarial loss can help generator to create structure

images, it cannot guarantee the consistency of structured patterns between the output and the input, i.e., the adversarial loss alone is inadequate for translating a crack patch to the desired structure patch, vice versa. However, researchers found that by introducing extra cycle-consistency constraints, the network can be trained to realize the abovementioned goal [86]. As in Figure 5.2, for each sample x from dataset X, after a cycle processing, the network should be able to bring x back to the original patch, i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \sim x$. In the same way, for each structure image y from the structure library Y, after a cycle processing, the network should also be able to bring y back to the original image, i.e., $y \rightarrow R(y) \rightarrow F(R(y)) \sim y$. Such constraints can be formulated to the following cycle-consistency loss:

$$L_{cyc}(F, R) = E_{x \sim p_d(x)}[\|R(F(x)) - x\|_1] + E_{y \sim p_d(y)}[\|F(R(y)) - y\|_1] \tag{5.4}$$

5.3.3 Network architecture

As in Figure 5.2, the entire network includes two image-to-image translation GANs: a forward GAN F that inputs a crack image and generates a structured image; and a reverse GAN R that inputs a structured image and generates a crack image; F and R use the same network architecture where each GAN includes two parts: the generator and discriminator. The generator is a U-shape [77] image-to-image translation network and the discriminator is a classification network with a larger receptive field that is realized by adding another stride convolution layer to the original discriminator, named one-class discriminator.

U-Net generator: The U-Net includes three components: encoding layers, decoding layers and skipping layers. The encoding layers extract useful information with stride convolutions and produce some representative feature maps/vectors; the decoding layers reconstruct an image with the same size of the input image from the encoded feature maps; and the skipping layers are the feature maps from the encoder that are reused as the inputs of the decoding layers to provide more detailed information for image generation, refer to Figure 5.4. The encoder consists of four repeated convolution layers with 3×3 kernels; after each of the first three convolutional layers, the number of feature map channels is doubled.

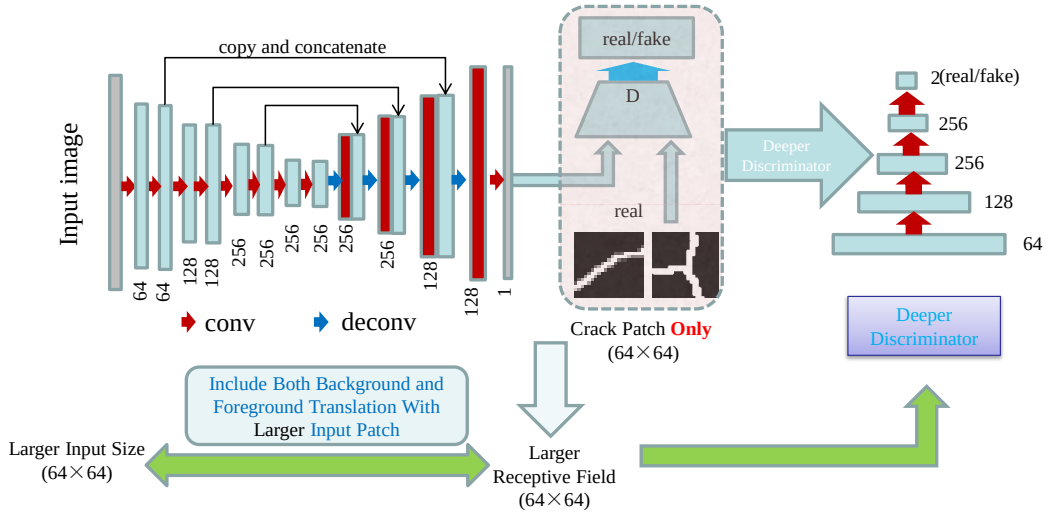


Fig. 5.4: Network architecture of the forward GAN

For the decoder, it consists of four 3×3 deconvolution layers that "up-sample" the feature maps; the input of each deconvolution layer is the output of the last layer and the skipping layers from the encoding part. Each convolution or deconvolution layer is followed by a ReLU layer [51]. After the last deconvolution layer, another regular convolution layer with 3×3 kernels is configured to translate the 128-channel feature maps to the output image, and it is compared with the dilated GT for L1 loss computation according to Eq. (5.4). In summary, the architecture of the generator is as follows.

The encoding part:

C_64_3_2 - ReLU - C_64_3_1 - ReLU - C_128_3_2 - ReLU - C_128_3_1 - ReLU - C_256_3_2 - ReLU - C_256_3_1 - ReLU - C_256_3_2 - ReLU - C_256_4_1 - ReLU

The decoding part:

DC_256_3_2 - ReLU - DC_256_3_2 - ReLU - DC_128_3_2 - ReLU - DC_128_3_2 - ReLU - C_1_3_1

The naming rule follows the format "layer type_kernel number_kernel size_stride"; "C" denotes convolution and "DC" is deconvolution. For instance, "C_64_3_2" means that the first

layer is a convolution layer, the number of kernel is 64, the kernel size is 3 and the stride is 2.

Discriminator with Larger FOV: Before introducing the architecture of one-class discriminator, we explore the regular discriminator concerning receptive field of view which refers to the discriminator used in the original image-to-image translation GANs [70] where the output of discriminator contains multiple labels. In convolutional GAN, a generator is usually configured with a series of deconvolution layers to generate images from random noise, and the discriminator is set up with a classifier to discriminate the generated image and real image. With such setting, the discriminator can only process image patches with the same size as the generator's outputs. Isola et al. [70] configured the generator and discriminator with fully convolutional networks where the input can be arbitrary size. However, when it inputs a larger size image, the output of the discriminator will have multiple labels, which indeed does not change the input size of each sample, i.e., the receptive field of each output neuron/label is the same. For example in Figure 5.5, originally the discriminator, CNN-1, is designed to discriminate a patch of $m \times m$ as real or fake; with such setting, when the generator is fed in a real crack image patch with larger size, e.g. $2m \times 2m$, the discriminator will output a 2×2 matrix where each label corresponds to an image sample of $m \times m$ from the larger input image. As illustrated in CNN-2, the $m \times m$ image samples include two different types of images: the image patch containing a crack pattern or the "all black" patch; and they are both considered as real images since they are all sampled from the real larger structure images. It has been discussed that the crack like object detection confronts serious data imbalance problem because the background usually dominates the image area; treating both crack patch and "all black" patch as real can weaken the foreground-background distinguishing ability and fail the task.

Here, we use a discriminator which takes a larger size image as input but treat it as a single image for discrimination, i.e. each larger input corresponds to a single label. The benefit of such design is that the discriminator will only treat the structured patch (crack patch) as real and force the generator to pay more attention on the crack areas in a larger

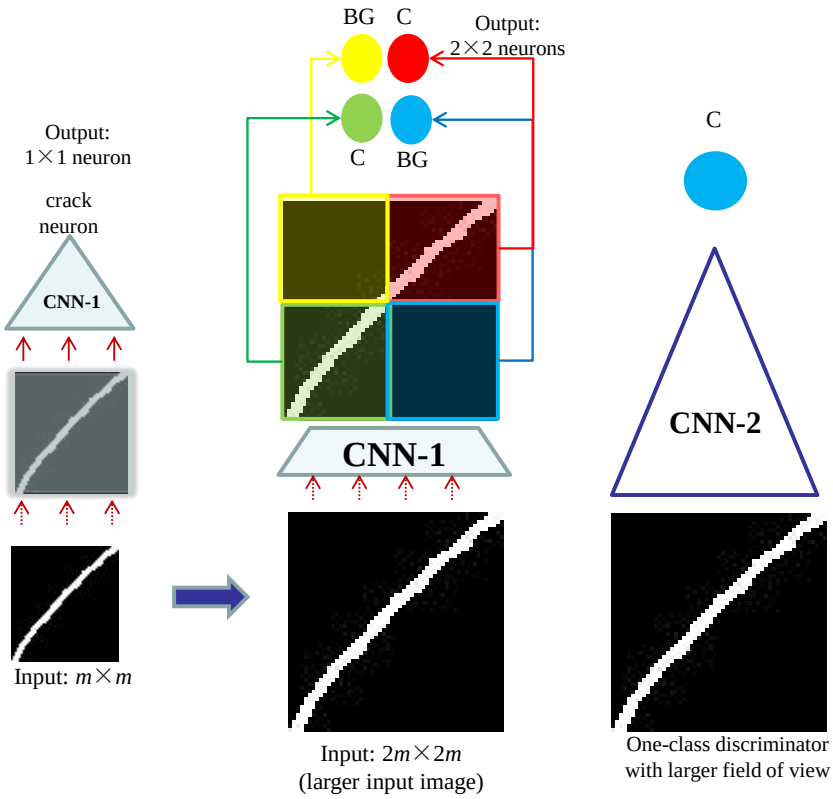


Fig. 5.5: One-class discriminator

input image to avoid background area dominating the training samples, thus to overcome the data imbalance. It is realized by explicitly adding a stride convolution layer to the discriminator; then the discriminator will only outputs a single label/neuron for each input image, it is named one-class discriminator. In detail, the discriminator consists of five 3×3 convolution layers and a fully connected layer followed by a Softmax layer with 2-channel output representing the class label of a 64×64 image. In summary, the discriminator is:
C_64_3_2 - ReLU - C_128_3_2 - ReLU - C_256_3_2 - ReLU - C_256_3_2 - ReLU - FC_4_2 -SM
The "FC" is fully connected layer and "SM" is Softmax layer; and they together serve as a classifier to determine if the input image is real, and to provide the adversarial loss. For the reverse network, it uses the same architecture where the input is a structure patch and the output is a real-like crack image with relevant background textures. The crack images from the training data are considered as real and the images generated by the U-shape generator are considered as fake.

5.3.4 Network training

Parameter initialization: It has present that the generic knowledge learned from large scale data such as ImageNet [37] can be used to extract crack pattern and is also beneficial to network training. As has discussed in Chapter 4, the output after conducting convolution using a pertained image classification network on ImageNet shows strong crack pattern the same as the original crack (see Figure 5.6). Here, we adopt the same routine and initialize parameters of the forward generator using a pre-trained model [5]. In addition, different from the original cycle-GAN, it also pre-trains a DC-GAN with the structure library as discussed in Chapter 4 (refer Figure 4.6); then the discriminator of the well-trained DC-GAN is transferred and concatenated at the end of the generator in the forward GAN to provide the generative adversarial loss and update the parameters via backpropagation; therefore, the pre-trained discriminator indeed severs as a domain adaptor that forces the generated image to become a structured patch. For the discriminator in the reverse GAN, it is trained from scratch.

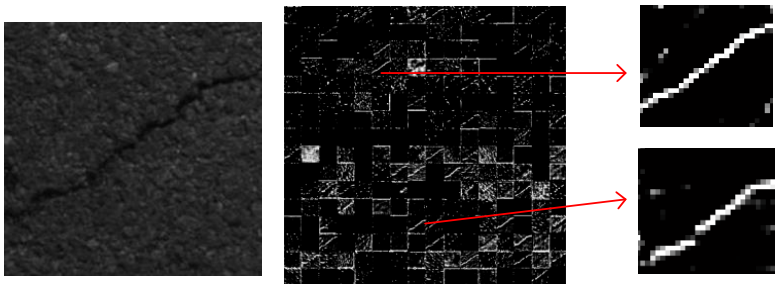


Fig. 5.6: Knowledge learned from a classification task has strong transferring ability and can be used for parameter initialization to ease the training on different tasks. Left: original crack image block. Middle: feature maps extracted using low level convolutional kernels of pre-trained model.

Training detail: After initialization, the parameters of the generators and the discriminator in the reverse GAN are updated alternatively following Algorithm 1. The Adam [79] optimizer is used, the learning rate is set as 0.0002. Since the two cycles are trained simultaneously, and the cycle-consistency restriction is applied to both the forward and reverse GANs, the training samples include both real crack patches and generated crack patches, and the structure patches from the library and the generated structure patches. Thus, an additional buffer is needed to store the generated images during the training; 50 previously generated images are stored; and the batch size is 6 due to memory limitation. A total of 100 epochs were run and the model parameters are saved every 5 epochs. In this way, the networks can be trained to generate similar structure patterns according to the input cracks: the forward GAN is trained to translate the original crack patches to the structure patches; and the reverse GAN is trained to translate the structured patches to crack images. Refer Figure 5.7 for the results of such translation on the test set. In experiments, the validation results are saved every twenty-iteration to monitor the training; finally, all the saved models are tested to select the best one as the detection network. After the training, the reverse GAN and the discriminator of the forward GAN are no longer used, and the generator of the forward GAN alone serves as the detection network that translates a pavement image

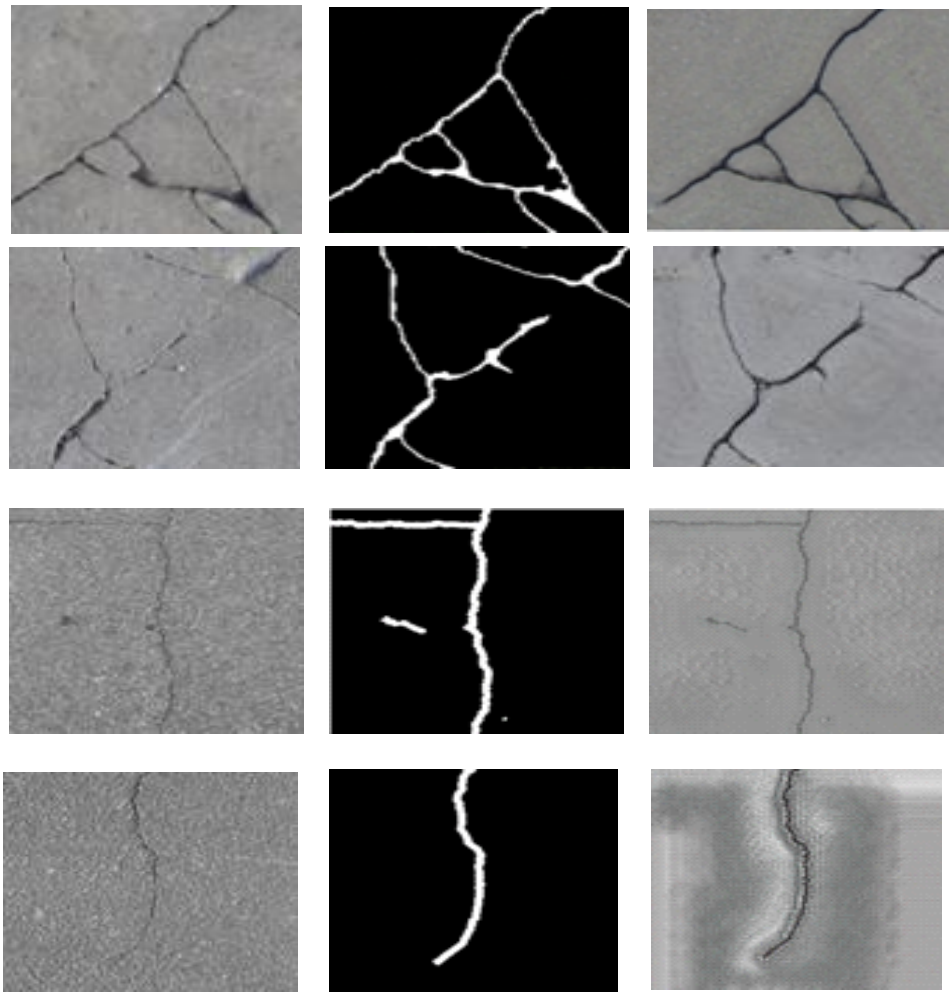


Fig. 5.7: Results from the image-to-image translation network. Left column: the test images from [49] and [19]. Middle column: the corresponding translation results of the forward GAN. Right column: the reconstructed results by the reverse GAN using the translated results.

into the GT-like image for crack detection.

5.4 Experiments

The experiments are conducted with an HP workstation with Intel i7 CPU and 32G memory. An Nvidia 1080Ti GPU was configured and PyTorch was used to perform the training and testing.

5.4.1 Datasets and metrics

The comparisons are present on the two representative datasets: the public dataset CFD [23] captured with a cellphone and the data from industry. As mentioned in Chapter 4, the industrial data have no precisely marked pixel-level GTs; thus, the GTs are roughly marked using 1-pixel curves which are used only for crack/non-crack block sampling.

In addition, another two datasets are also employed, CrackTree dataset [19] and FCN dataset [49]. CrackTree is a pavement crack detection dataset with 206 pavement images of 600×800-pixel. The accurate GTs are also available. FCN dataset is also a public dataset composed of 800 images. The images are collected from multiple sources including historical crack images from internet, newly captured crack images of concrete walls from buildings and the pavement crack images. While the multi-source images are taken at different distances with different cameras, the resolution levels ranges from 72 dpi to 300 dpi. The GTs are manually marked at pixel-level.

For the evaluation metrics, the region based p-rate, r-rate, and HD-score are used as the metrics for the evaluation. Refer Chapter 3 for the detailed definitions.

5.4.2 Performance evaluation

Since the proposed approach is a self-supervised/unsupervised method, it is compared with the rule based approaches, which could be viewed as unsupervised methods, and the machine learning based methods. The rule based approaches are CrackIT [83] and MFCD [84]; and the machine learning methods are CrackForest [23], and the CrackGAN in Chapter 4.

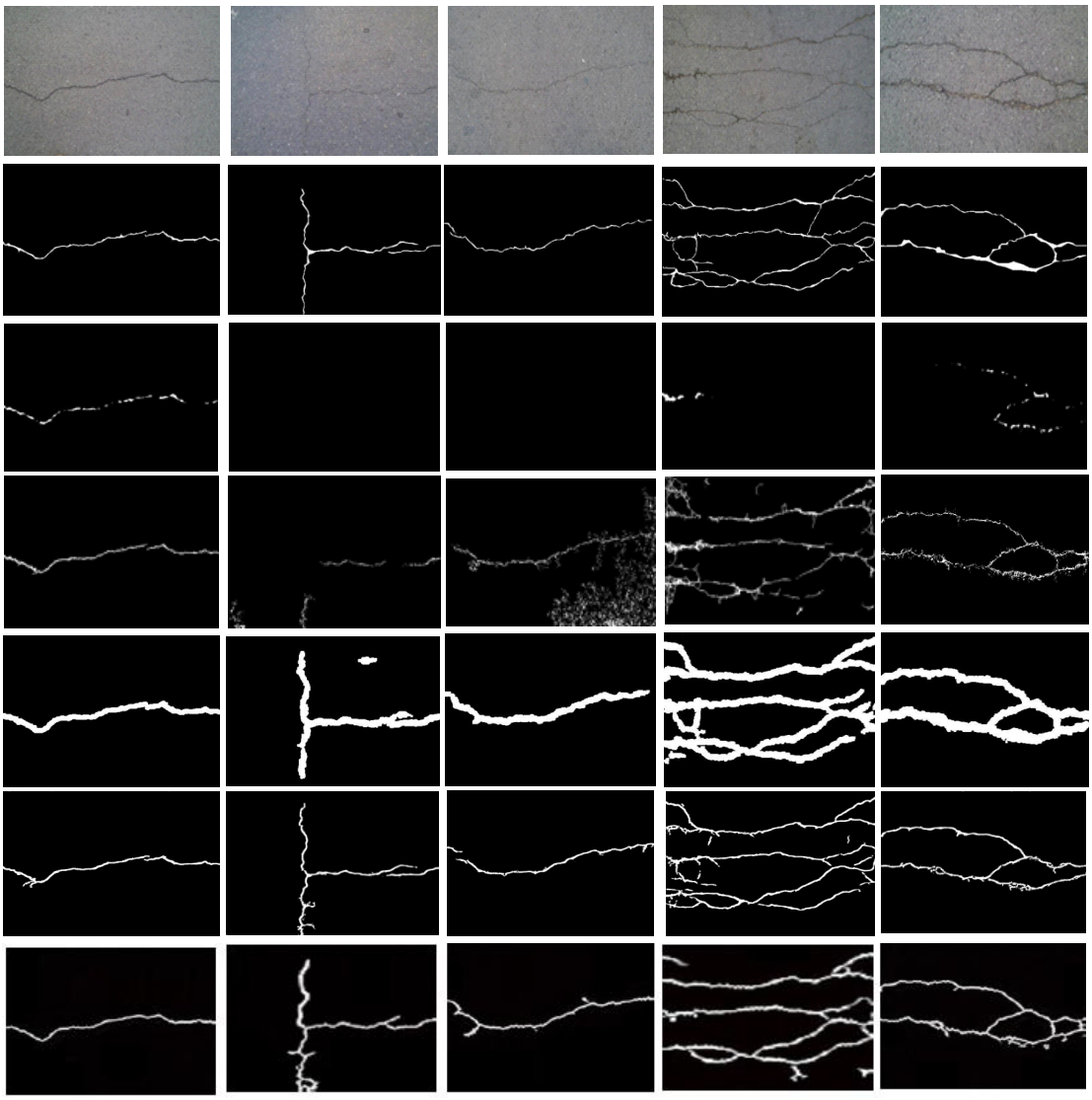


Fig. 5.8: Comparisons on CFD data. From top to bottom are: original image, GTs, and the detection results of CrackIT, MFCD, CrackForest, CrackGAN and the proposed method, respectively.

Table 5.1: Quantitative evaluation of self-supervised GAN on CFD

Methods	p-rate	r-rate	F1-score	HD-score
CrackIT	88.02%	45.11%	59.65%	21
MFCD	80.90%	87.47%	84.01%	85
CrackForest	85.31%	90.22%	87.69%	88
CrackGAN	88.03%	96.11%	91.88%	96
Proposed	88.01%	93.02%	89.01%	92

Table 5.2: Quantitative evaluation of self-supervised GAN on industrial data

Methods	p-rate	r-rate	F1-score	HD-score
CrackIT	89.10%	2.52%	4.90%	9
MFCD	50.90%	87.47%	64.35%	67
CrackForest	31.10%	98.01%	47.22%	63
CrackGAN	89.21%	96.01%	91.28%	95
Proposed	78.01%	86.02%	81.81%	78

Results on CFD: The sample images with the detection results and the quantitative measurements in terms of precision, recall, F-measure and Hausdorff score are present in Figure 5.8 and Table 5.1, respectively. The CFD dataset provides pixel-level GTs; thus, all the provided detection results including the supervised machine learning methods are from the experiments using CFD data; and the training and test samples split rate is 2:1. From Figure 5.8, it can be observed that CrackIT can detect the clear cracks with distinct crack width such as the cracks in the first image, but it misses most thin cracks in the second and third images. MFCD shows strong crack detection ability and can detect most cracks. However, it may introduce noises when dealing with images having non-smooth background as in the third image. For CrackForest, it achieves very good results using its own data but the performance decreases on different data. For CrackGAN, it has very good detection results and achieved the best performance. From Table 5.1 and Figure 5.8, the proposed method outperforms the "unsupervised" methods CrackIT and MFCD and achieves comparable results as the best supervised methods; however, the advantage is that it is a self-supervised method trained with a labor-free structure library only, without using expensive GTs.

Results on industry dataset: The detection results on the data from industry are present

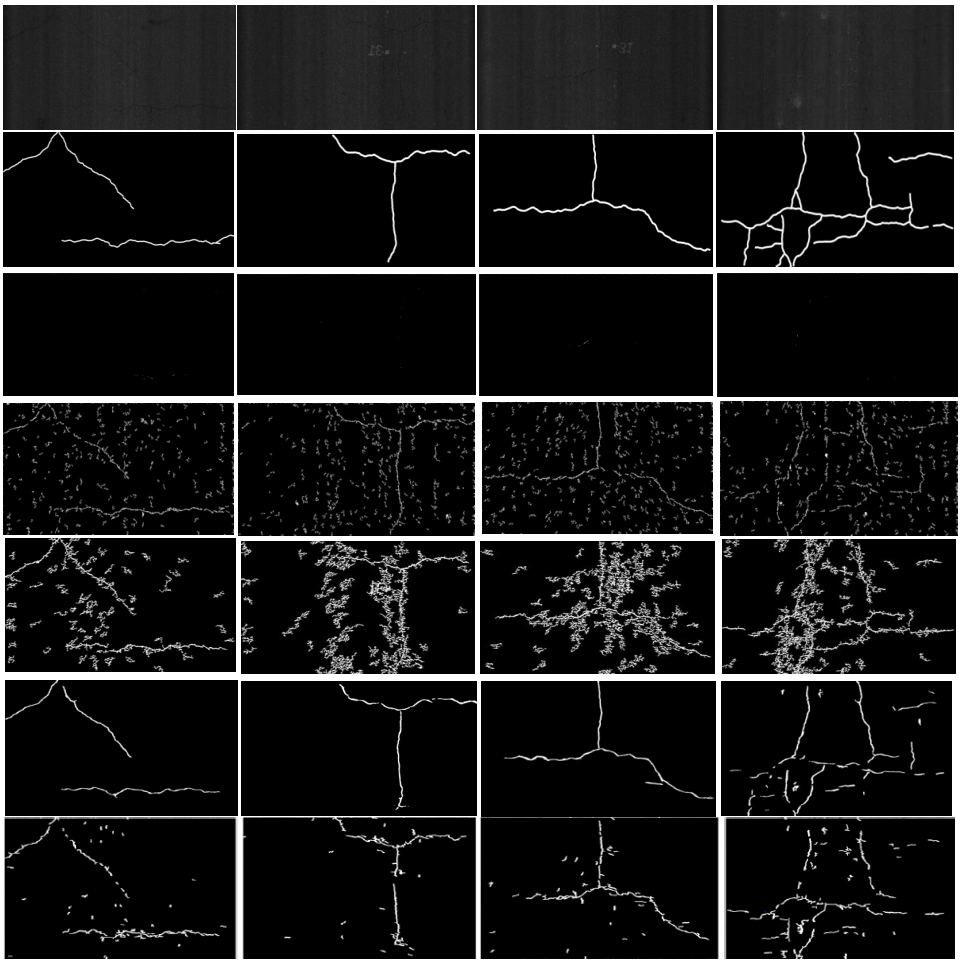


Fig. 5.9: Comparison of detection results on industrial data. From top to bottom are: original image, GTs, and the detection results of CrackIT, MFCD, CrackForest, CrackGAN and the proposed method, respectively.

in Table 5.2 and Figure 5.9. Different from the CFD data, there is no precise pixel-level GTs for this dataset. The biased/inaccurate 1-pixel GTs are not suitable for end-to-end training, and it will easily run into the "all black" local minimal due to the pixel-level mismatching and data imbalance; therefore, the results of the end-to-end training methods are the ones trained using the mixed datasets of CFD, CrackTree and FCN of which the pixel-level GTs are given. As shown in Figure 5.9, CrackIT shows the similar results as using CFD data, but most cracks are missed from the industrial data. MFCD detects most cracks but also introduces a lot of noise due to the complicated pavement texture from the background. For CrackGAN it achieved best results because of the supervised end-to-end training. The proposed method trained with the assistance of the structure library achieved comparable results with the CrackGAN without labor-cost GTs.

5.4.3 Ablation study

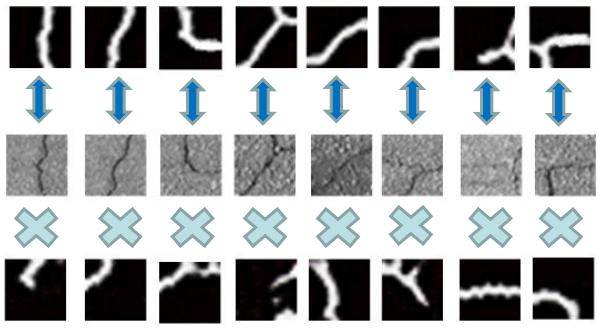


Fig. 5.10: Testing results on training set with and without cycle consistent loss. The top row is the generated images with the proposed setting, and the bottom row is the generated images without cycle consistent loss.

Ablation of cycle-consistency loss: GAN has been proved to perform the learning via reducing the distribution difference between real data and generated data. As reported in [86], the distribution consistency could be any permutation of the images which cannot

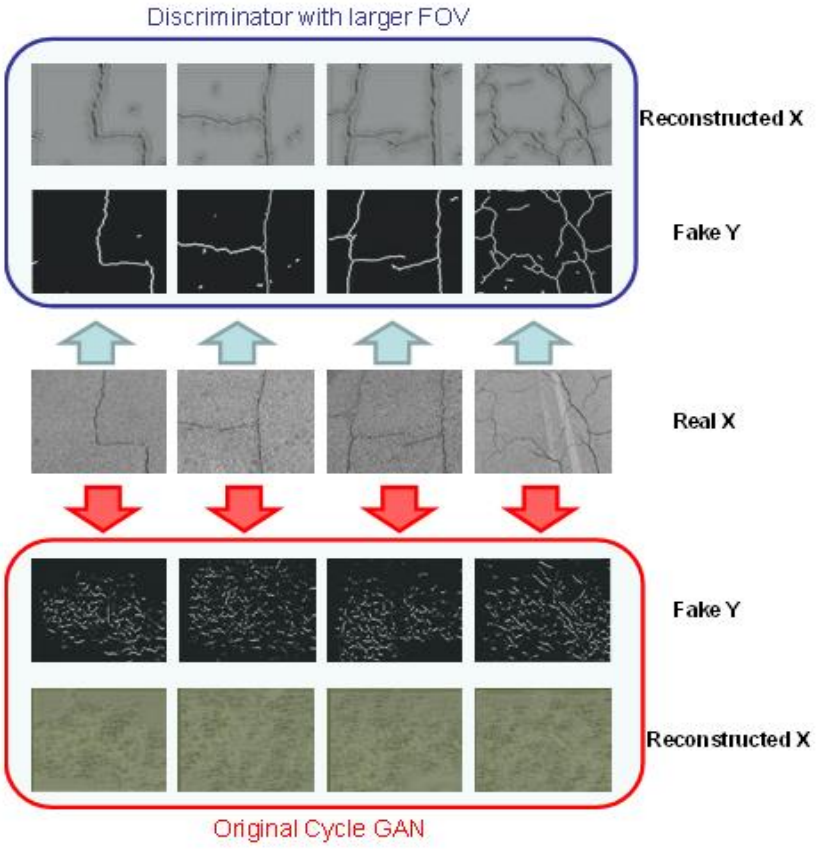


Fig. 5.11: Experiments with and without one-class discriminator. Images at the top are with the proposed setting and the images on the bottom are results with the original cycle-GAN.

guarantee the consistency of input-output patterns. In Figure 5.10, the images on the bottom row are the generated image patches by the network trained without cycle-consistency loss while the top row are image patches with both the adversarial loss and the cycle-consistency loss. It can be seen that without the cycle-consistency loss, the network can generate the structured image patches; however, the generated structure patterns are quite random that do not match the input crack patterns well, which cannot be used for crack detection. Notice that the testing is conducted on the training dataset directly.

Ablation of one-class discriminator: One of the main differences from the original GAN is the modified discriminator which works with a larger input image (larger than the receptive field of the generator). While it has been discussed that with the original design, the discriminator would treat both the structured patch and the "all black" patches as real where the structured patches represent the crack patches and the "all black" patches represent the background patches, respectively. From Figure 5.11, it can be observed that without one-class discriminator, the network cannot generate expected results, only produce some nonsensical textures. A possible explanation is that when the discriminator treated both structured patch and "all black" patch as real, its function was greatly weakened. In contrast, the network trained with one-class discriminator can handle the problem properly.

5.4.4 Summary

In this Chapter, we discussed a self-supervised structure learning approach for crack detection without relying on GTs, which has potential to realize real fully automatic crack detection that does not need to manually mark GTs for the training. It formulated crack detection as an unsupervised structure learning problem by introducing a labor-free structure library to assist the training of cycle-GAN. It proposed to use discriminator with larger field of view, which only treats the crack patches as real, to overcome the data imbalance problem inherent in crack-like object detection. The method is validated using multiple crack detection datasets. Moreover, the proposed method combines domain adaptation and generative adversarial networks to handle object detection, which is a general means that can be applied to other computer vision problems.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

In this dissertation, we have explored and designed deep learning algorithms for crack-like object detection. Specifically, we have solved four important issues using deep learning for industrial crack detection, which have filled the gap between academic research and industrial application. The contributions of our work are as follows.

In Chapter 2, we employed a deep classification network for crack-region pre-selection which improved the performance by removing most of the noisy regions before crack detection.

In Chapter 3, we first generalized a classification network to a detection network which realized a one-stage crack detection network for large size input images. The method boosts the computation efficiency by more than ten times faster than the window-sliding based strategy, and makes deep learning feasible for real-time industrial crack detection.

In Chapter 4, we introduced generative adversarial learning to solve an important problem inherent in crack-like object detection, data imbalance. The network is robust to biased GTs, which greatly reduced the workload of preparing GTs for the training. It is of great significance for saving labor-cost and reducing budget in industry applications.

In Chapter 5, we proposed a self-supervised method for crack detection. The method introduced cycle-consistent generative adversarial networks that does not need to manually preparing GTs for the training, which minimizes the human intervention; and it is an important research direction of the future system for crack detection.

The dissertation focuses on deep learning for crack-like object detection problem based on the pavement crack detection; nonetheless, the proposed method can be used for multi-class object detection via setting the Softmax layer as multi-channel outputs, such as locating road marks [69, 92], discriminating crack and sealed cracks [5], etc.

We have proposed to use generative adversarial learning for crack detection. However, it should be mentioned that, the networks detect the cracks as n-width crack curves which will miss the crack width information. In practical applications, many of the protocols prefer to use crack width to estimate cracking severity which might be an issue for the proposed methods, and it needs to be further discussed.

REFERENCES

[1] N. F. Hawks, and T. P. Teng. *Distress identification manual for the long-term pavement performance project*. National academy of sciences, 2014.

[2] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: an algorithm based on minimal path selection," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2718-2729, 2016.

[3] I. Abdel, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," *J. Comput. Civil Eng.*, vol. 17, no. 4, pp. 255-263, 2003.

[4] P. Lad, and M. Pawar, "Evaluation of railway track crack detection system," *in Proc., IEEE ROMA*, 2016, pp. 1-6.

[5] K. Zhang, H. D. Cheng, and B. Zhang, "Unified approach to pavement crack and sealed crack detection using pre-classification based on transfer learning," *J. Comput. Civil Eng.*, vol. 32, no. 2 (04018001), 2018.

[6] K. Zhang, and H. D. Cheng, "A novel pavement crack detection approach using pre-selection based on transfer learning," *in Proc. of the 9th International Conference on Image and Graphics*, pp. 237-283: Springer, 2017.

[7] K. Zhang, H. D. Cheng, and S. Gai, "Efficient Dense-Dilation Network for Pavement Crack Detection with Large Input Image Size," *in Proc. IEEE ITSC*, 2018, pp. 884-889.

[8] Central Intelligence Agency, "The world factbook." *https://www.cia.gov/library/publications/resources/the-world-factbook/*, accessed Jan. 2017.

[9] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," *in Proc., IEEE ICIP*, 2016, pp. 3708-3712.

[10] M. Gavilan, D. Balcones, D. F. Llorca et al., "Adaptive road crack detection system by pavement classification," *Sensors*, vol. 11, no. 10, pp. 9628–9657, 2011.

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

[12] K. R. Kirschke, and S. A. Velinsky, "Histogram-based approach for automated pavement crack sensing." *Journal of Transportation Engineering*, vol. 118, no. 5, pp. 700-710, 1992.

[13] H. D. Cheng, J. Chen, C. Glazier, and Y. G. Hu, "Novel approach to pavement crack detection based on fuzzy set theory," *J. Comput. Civil Eng.*, vol. 13, no. 4, pp. 270-280, 1999.

[14] H. D. Cheng, and M. Miyojim, "Novel system for automatic pavement distress detection." *J. Comput. Civil Eng.*, vol. 12, no. 3, 145-152, 1998.

[15] H. Oliveira, and P. L. Correia, "Automatic road crack segmentation using entropy and dynamic thresholding," *in Proc. of 17th European Signal Processing Conf.*, 2009, pp. 622-626.

[16] H. N. Koutsopoulos, I. E. Sanhouri, and A. B. Downey, "Analysis of segmentation algorithms for pavement distress images," *Journal of Transportation Engineering*, vol. 119, no. 6, pp. 868-888, 1993.

[17] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. on Syst. Man Cybern.*, vol. 9, no. 1, 62-66, 1979.

[18] A. Rosenfield, and R. C. Smith, "Thresholding using relaxation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, no. 5, pp. 598-606, 1979.

[19] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "CrackTree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227-238, 2012.

[20] R. C. Gonzalez, R. E. Woods, and S. L. Steven. *Digital image processing using Matlab*, Addison-Wesley, 2009.

[21] Y. Hu, and Zhao, C. X., "A local binary pattern based methods for pavement crack detection," *J. Pattern Recognit. Res.*, vol. 5, no. 1, pp. 140-147, 2010.

[22] E. Zalama, J. Bermejo, R. Medina, and J. Llamas, "Road crack detection using visual features extracted by Gabor filters," *Computer-aided Civil and Infrastructure Engineering*, vol. 29, no. 5, pp. 342-358, 2014.

[23] Y. Shi, L. M. Cui, Z. Qi et al., "Automatic road crack detection using random structured forest," *IEEE Trans. on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3434-3445, 2016.

[24] P. Dollár, and C. L. Zitnick, "Structured forests for fast edge detection," *in Proc., IEEE ICCV*, 2013, pp. 1841–1848.

[25] P. Kontschieder, S. R. Bulo, and M. Pelillo, "Structured class-labels in random forest for semantic image labelling," *in Proc., IEEE ICCV*, 2011, pp. 2190-2197.

[26] M. Petrou, and J. Kittler, "Automatic surface crack detection on textured materials." *J. Mater. Process. Technol.*, vol. 56 (1-4), pp. 158-167, 1996.

[27] K. Wang, Q. Li, and W. Gong, "Wavelet-based pavement distress image edge detection with a trous algorithm," *Transp. Res. Rec.*, vol. 6 (2024), pp. 24-32, 2000.

[28] J. Zhou, P. S. Huang, and F. P. Chiang, "Wavelet-based pavement distress detection evaluation," *Optic Engineering*, vol. 45, no. 2, pp. 1-10, 2006.

[29] F. M. Nejad, and H. Zakeri, "A comparison of multiresolution methods for detection and isolation of pavement distress," *Journal of Expert Systems with Applications*, vol. 38, no. 3, pp. 2857-2872, 2011.

[30] K. Vaheesan, C. Chandrakumar, and M. Rahman, "Tiled fuzzy Hough transform for crack detection," *in Proc., 12th International Conference on Quality Control by Artificial Vision*, pp. 1-6: SPIE(953411), 2015.

[31] H. Song, W. Wang, F. Wang et al., "Pavement crack detection by ridge detection on fractional calculus and dual-thresholds," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 4, pp. 19-30, 2015.

[32] H. D. Cheng, J. Wang, Y. Hu et al., "Novel approach to pavement cracking detection based on neural network," *Transp. Res. Rec.*, vol. 1764, pp. 119–127, 2001.

[33] Y. Huang, and B. Xu, "Automatic inspection of pavement cracking distress," *Journal of Electronic Imaging*, vol. 15, no. 1, pp. 17-27, 2006.

[34] G. E. Hinton, D. Li, Y. Dong et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82-97, 2012.

[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, pp. 436-444, 2015.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural network," *in Proc., NIPS*, 2012, pp. 1-9.

[37] J. Deng, W. Dong, R. Socher et al., "ImageNet: A large-scale hierarchical image database," *in Proc., IEEE CVPR*, 2009, pp. 248-255.

[38] R. Girshick, J. Donahue, T. Darrell et al., "Rich feature hierarchies for accurate object detection and semantic segmentation," *in Proc. IEEE CVPR*, 2014.

[39] R. Girshick, "Fast R-CNN," *in Proc., IEEE ICCV*, 2015, pp. 1440-1448.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *in Proc. ECCV*, pp. 346-361: Springer, 2014.

[41] S. Ren, K. He, R. Girshick et al., "Faster R-CNN: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[42] J. Dai, Y. Li, K. He et al., "R-FCN: Object detection via region-based fully convolutional networks," *arXiv preprint arXiv:1605.06409*, 2016.

[43] W. Liu, D. Anguelov, D. Erhan C. Szegedy et al., "SSD: single shot multi-box detector," *in Proc., ECCV*, pp. 21-37: Springer, 2016.

[44] J. Redmon, S. Divvala, R. Girshick et al., "You only look once: unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2015.

[45] J. Redmon, and A. Farhadi, "YOLO9000: Better, faster, stronger," *in Proc., IEEE CVPR*, 2017, pp. 6517-6525.

[46] K. He, G. Gkioxari, P. Dollar et al., "Mask R-CNN," *arXiv preprint arXiv:1703.06870*, 2017.

[47] K. Zhang, H. D. Cheng, "Asymmetrical U-Net: A Bias-Tolerant Crack Detection Network Using Generative Adversarial Learning," submitted to CVPR 2019.

[48] A. Zhang, C. P. Kelvin, B. Wang et al., "Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network," *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 32, no. 10, pp. 805-819, 2017.

[49] X. Yang, H. Li, Y. Yu et al., "Automatic pixel-level crack detection and measurement using fully convolutional network," *Comput.-Aided Civ. Infrastructure. Eng.*, vol. 33, no. 12, pp. 1090-1109, 2018.

[50] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection," *in Proc., IEEE CVPR*, 2005.

[51] V. Nair, and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," *in Proc., ICML*, 2010.

[52] N. Srivastava, G. E. Hinton, A. Krizhevsky et al. "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[53] L. Wan, M. Zeiler, S. Zhang et al., "Regularization of neural networks using DropConnect," *in Proc. ICML*, 2013.

[54] K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR, abs/1409.1556*, 2014.

[55] S. J. Pan, and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345-1359, 2010.

[56] J. Yosinski, J. Clune, Y. Bengio et al., "How transferable are features in deep neural networks?" *in Proc. NIPS*, 2014.

[57] R. Tibshirani, "Regression shrinkage and selection via lasso," *Journal of Royal Statistic Society*, vol. 58, no. 1, pp. 267-288, 1996.

[58] M. Oquab, L. Bottou, I. Laptev et al., "Learning and transferring mid-level image representations," *in Proc., IEEE CVPR*, 2014, pp. 1717-1724.

[59] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev et al., "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv: 1408. 5093*, 2014.

[60] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[61] W. Wang, "Protocol based pavement cracking measurement with 1 mm 3D pavement surface model." *Ph.D. thesis*, Oklahoma State Univ., Stillwater, 2015.

[62] N. F. Hawks, and T. P. Teng, *Distress identification manual for the long-term pavement performance project*. National academy of sciences, 2014.

[63] D. Power, "Evaluation: from precision, recall and F-measure to ROC, Informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, 2011.

[64] S. J. Sheather, *A modern approach to regression with R.* Springer, New York, 2009.

[65] J. Friedman, Hastie, T., and Tibshirani, R., "Regularization paths for generalized linear models via coordinate descent," *J. Stat. Software*, vol. 33, no. 1, pp. 1-22, 2010.

[66] G. Medioni, and Tang, C., "Tensor Voting: Theory and Applications," *in Proc., 12th Congress Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificial*, 2000.

[67] T. Linton, "Tensor Voting," *https://www.mathworkcia.gov/library/publications/resources*, accessed 2017.

[68] P. Sermanet, D. Eigen, X. Zhang et al., "Overfeat: integrated recognition, localization and detection using convolutional networks," *arXive prepring arXiv:1312.6229*, 2014.

[69] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *in IEEE CVPR*, 2015.

[70] P. Isola, J. Y. Zhu, T. Zhou et al, "Image-to-image translation with conditional adversarial networks," *in Proc. IEEE CVPR*, 2017.

[71] F. Yu, and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *in Proc. ICLR*, 2016.

[72] L. C. Chen, G. Papandreou, I. Kokkinos et al., "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *in Proc. Int. Conf. Learn. Represent.*, 2015.

[73] L. Chen, G. Papandreou, I. Kokkinos et al., "DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *arXiv preprint arXiv:1606.00915*, 2017.

[74] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," *in Proc. NIPS*, 2014.

[75] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *in Proc. ICLR*, 2016.

[76] Y. Ganin, and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[77] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *in Proc. Med. Image Comput. Comput.-Assist, Intervention*, 2015, pp. 234–241.

[78] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning GAN for pose-invariant face recognition," *in Proc. IEEE CVPR*, 2017.

[79] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[80] M. Everingham, L. V. Gool, C. K. I. William et al., "The PASCAL Visual Object Classes Challenge 2011 Results," *Online Available: http://www.pascalnetwork.org/challenges/VOC/voc2011/workshop/index.html*, accessed 2017.

[81] Y. C. Tsai, V. Kaul, and R. M. Mersereau, "Critical assessment of pavement distress segmentation methods," *Journal of Transportation Engineering*, vol. 136, no. 1, pp. 11-19, 2010.

[82] Y. C. Tsai, and A. Chatterjee, "Comprehensive, quantitative crack detection algorithm performance evaluation system," *J. Comput. Civ. Eng.*, vol. 31, no. 5 (04017047), 2017.

[83] H. Oliveira, and P. L. Correia, "CrackIT-An image processing toolbox for crack detection and characterization," *in Proc., IEEE ICIP*, 2014.

[84] H. Li, D. Song, Y. Liu, and B. Li, "Automatic Pavement Crack Detection by Multi-Scale Image Fusion," *IEEE Trans. Intell. Transp. Syst., DOI: 10.1109/TITS.2018.2856928*, 2018.

[85] M. Mirza, and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[86] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *in Proc. IEEE ICCV*, 2017.

[87] C. Doersch, and A. Zisserman, "Multi-task self-supervised visual learning," *arXiv preprint arXiv:1708.07860*, 2017.

[88] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," *in Proc. IEEE ICCV*, 2015.

[89] M. Noroozi, and P. Favaro, "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles," *in Proc. ECCV*, 2016.

[90] K. Sohn, S. Liu, G. Zhong et al., "Unsupervised domain adaptation for face recognition in unlabeled videos," *in Proc. IEEE CVPR*, 2017, pp. 3210–3218.

[91] D. Martin, C. Fowlkes, D. Tal et al., "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," *in Proc. IEEE ICCV*, 2001.

[92] H. Maeda, Y. Sekimoto, T. Seto et al., "Road damage detection and classification using deep neural networks with smartphone images," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 12, pp. 1127-1141, 2018.

CURRICULUM VITAE

Kaige Zhang

EDUCATION

Ph.D., Computer Science — Jan. 2015 - present
Utah State University, USA
Dissertation: "Deep Learning for Crack-Like Object Detection"

M.S., Signal & Information Processing — Sep. 2011 - Mar. 2014
Harbin Engineering University, China

B.S., Electronical Engineering — Sep. 2007 - July 2011
Harbin Institute of Technology, China

RESEARCH EXPERIENCE

USU, CVPRIP Lab — Jan. 2015 - present
Deep Learning for Crack-Like Object Detection

- developed a pavement cracks detection algorithm with modified fully convolutional neural network;
- proposed a solution to solve the efficiency problem in deep learning-based object detection;
- implemented the crack detection algorithm.

Hikvision Research USA — May 2018-Aug. 2018

- Pose-robust face recognition based on deep learning

HEU, Signal Processing Lab — Sep. 2007 - July 2011

- designed an aircraft trajectory tracking algorithm based on generalized Kalman filter and build the ADS-B message processing system;
- cooperated with a partner and implemented the ADS-B message decoding, object tracking, and report generalization functions with FPGA + DSP6000.

HIT, Radar Signal Processing Lab Sep. 2011 - Mar. 2014

- studied different radar signal processing methods, e.g. matched filtering, 2-D Fourier transformation, DFT, IDFT, pulse compression, interpolating with sinc function, etc.
- implemented Spatial Frequency Interpolation algorithm for image reconstruction with spotlight SAR data using Matlab.

TEACHING EXPERIENCE

- taught recitation class for CS1400 (Introduction to Computer Science with Python) at Fall 2018; served as teaching assistant for CS2420 (Data Structure and Algorithms), CS1410 (C++2), and CS3810 (Computer System Organization and Architecture).

PROFESSIONAL SERVICES

- Session Chair of IEEE International Conference of Intelligent Transportation System 2018 (Pavement Perception Session).
- Reviewer for International Journal of Pavement Research and Technology (Elsevier), Automation in Construction (Elsevier), IEEE Access, IEEE ITSC2018; Co-reviewer for ECCV2018 (Primary reviewers: Qinxun Bai)

PUBLICATIONS

- **Kaige Zhang**, H. D. Cheng and Boyu Zhang, Unified approach to pavement crack & sealed crack detection using pre-classification based on transfer learning, Journal of Computing in Civil Engineering, ASCE, vol. 32, no. 2, 2018..
- **Kaige Zhang**, H. D. Cheng and Shan Gai, Efficient Dense-Dilation Network for Pavement Cracks Detection with Large Input Image Size, IEEE ITSC 2018.
- **Kaige Zhang** and H. D. Cheng, Self-Supervised Structure Learing for Crack Detection Based On Cycle-GAN, Journal of Computing in Civil Engineering, 2019. (Accepted)
- **Kaige Zhang** and H. D. Cheng, A novel pavement crack detection approach using pre-selection based on transfer learning, ICIG 2017, LNCS, Springer.
- **Kaige Zhang** and H. D. Cheng, CrackGAN: A Labor-Light Crack Detection Approach Using Industrial Pavement Images Based on Generative Adversarial Learning, submitted to IEEE Transactions on Intelligent Transportation Systems. (Major revision)
- Shan Gai, **Kaige Zhang**, Vector Extension of Quaternion Wavelet Transform and Its Application to Color Image Denoising, IET Signal Processing (DIO: 10.1049/iet-spr.2018.5127).
- Xiao Ma, **Kaige Zhang** and Xuan Yang, Pose-invariant facial expression recognition based on 3D morphable model and generative adversarial learning, ICIG 2019, LNCS, Springer.
- Yulong Qiao, **Kaige Zhang** and Chaozhu Zhang, Research on Trajectory Tracking Using ADS-B Messages, ICIC Express Letters, vol. 8, no. 1, pp. 319-326, 2014.
- **Kaige Zhang**, Yulong Qiao and Chaozhu Zhang, Trajectory tracking using adaptive multi-model filtering method in ADS-B system, IEEE Proc. of Robot, Vision and Signal Processing, 2013, Japan.

- Chunyan Song, Yu-Long Qiao, **Kaige Zhang**, Infrared and Visible Image Fusion Based on Oversampled Graph Filter Banks, submitted to IEEE Access.