CrackGAN: Pavement Crack Detection Using Partially Accurate Ground Truths Based on Generative Adversarial Learning

Kaige Zhang, Yingtao Zhang, and H. D. Cheng

Abstract—**Fully convolutional network is a powerful tool for per-pixel semantic segmentation/detection. However, it is problematic when coping with crack detection using partially accurate ground truths (GTs): the network may easily converge to the status that treats all the pixels as background (BG) and still achieves a very good loss, named "All Black" phenomenon, due to the unavailability of accurate GTs and the data imbalance. To tackle this problem, we propose crack-patch-only (CPO) supervised generative adversarial learning for end-to-end training, which forces the network to always produce crack-GT images while reserves both crack and BG-image translation abilities by feeding a larger-size crack image into an asymmetric U-shape generator to overcome the "All Black" issue. The proposed approach is validated using four crack datasets; and achieves state-of-the-art performance comparing with that of the recently published works in efficiency and accuracy.**

Index Terms—**Pavement crack detection, fully convolutional networks, generative adversarial learning, and partially accurate GTs.**

I. INTRODUCTION

AUTOMATIC pavement crack detection is a challenging task in intelligent pavement surface inspection system [1]. It is also a research topic for more than three decades. However, industry-level pavement crack detection task is still not well solved: many published references have reported good results on specific crack datasets [2]; however, the methods failed when processing industrial pavement images of which the cracks were thin and the precise pixel-level ground truths (GTs) were difficult to obtain [3], [4]. Recently, fully convolutional network (FCN) [5], trained in end-to-end for pixel-level object segmentation/detection, was applied to pavement crack detection [4], [6]. However, it suffered from the "All Black" issue when processing industrial images: the network converged to the status that treated all the pixels as background (BG) [4]; and similar issue was also reported in [6] where the FCN failed to detect thin cracks.

It is known that deep learning is a data driven approach which heavily relies on the training data with accurate GTs. Due to the domain sensitivity (i.e., the performance of a "well-trained" network may decrease when utilizing the datasets obtained from different road sections and/or during different periods), it is necessary to *manually* mark the GTs to re-train the models for new pavement crack detection tasks. In industry, the pavement images are captured using a camera mounted on top of a vehicle running on the road. Under

such setting, most cracks are very thin and crack boundaries are vague, which makes the annotation of pixel-level GTs very difficult. Instead of the labor-intensive per-pixel crack annotation, marking the cracks as 1-pixel curves is more feasible and preferable in practice because of its simplicity and low labor-cost, and such GT is named *labor-light* GT. However, such GTs may not completely match the cracks at pixel-level accurately; i.e., they are partially accurate GTs, and that makes the loss computation inaccurate. Moreover, as a long-narrow target, a crack can only occupies a very small area in a full image. Since patch-wise training is equivalent to loss sampling in FCN [5], directly training FCN for pixel-level crack detection makes the training set heavily imbalanced; moreover, such problem cannot be handled by simply rebalancing the data via loss function since the GTs are not accurate. The observation is that the network will simply converge to the status that treats the entire crack image as BG (labeled with zero), and still can achieve a good detection accuracy (BG-samples dominate the accuracy calculation). It is named "All Black" problem which is pretty common in industrial pixel-level pavement crack detection.

In general, the existing computer vision-based crack detection approaches could be grouped into two categories: rule-based and machine learning-based methods. Rule-based methods try to extract some pre-defined features to identify the cracks. Cheng et al. [7] proposed a fuzzy logic-based intensity thresholding method for crack segmentation based on the assumption that crack pixels were darker than BG pixels; however, the method failed when processing crack images with low foreground-background contrast. Wang et al. [8] introduced a wavelet-based edge detection algorithm, and the drawback was that it could not handle the cracks with high curvature or low continuity well. Oliveira et al. [9] proposed a dynamic thresholding method for crack detection based on information entropy, which was sensitive to noise. Zou et al. [10] designed an intensity-difference measuring function to find an optimal threshold for crack segmentation; however, the robustness was poor and the method was easy to fail when working on different datasets. Many works introduced some crack linking method to enhance the crack continuity [11]-[16]. However, these methods did not solve the problem well and usually produced intolerable false positives for linking together the noises. In addition, Tsai et al. [17] performed a comprehensive study on the performances of six low-level image segmentation algorithms, and Abdel-Qader et al. [18] discussed different edge detectors, including Sobel, Canny, and

fast Haar transformation [19]. The rule-based approaches are easy to implement; however, they are sensitive to noise, which results in poor generalizability.

Machine learning-based methods have attracted increasing attentions during the past two decades. These methods perform crack detection following two steps: feature extraction and pattern classification. Cheng et al. [20] and Oliveira et al. [21] utilized mean and variance of an image block as the features to train classifiers for pavement crack detection. However, the good performances heavily relied on complex post processing. Hu et al. [22] and Gavilan et al. [23] utilized textural information to set up the feature vectors and employed support vector machine (SVM) for the classification. However, they could not handle the problem well when processing the images with complex pavement textures. Zalama et al. [24] employed Gabor filters for feature extraction and AdaBoosting [24] for crack identification; and Shi et al. [2] combined multi-channel information to set up the feature vector, and employed random structure forest [25] for crack-token mapping. These methods tried to solve the problem by extracting some hand-crafted features and training a classifier to discriminate cracks from the noisy BG; however, they did not address the issue well because the hand-crafted feature descriptors usually calculated statistics locally and lacked good global view, even the statistics from different locations were combined together. Thus, they could not represent the global structural pattern well which was important to discriminate cracks from the noisy textures.

As one of the most important branches in machine learning, deep learning has achieved great success during the past ten years, and it is the most promising way to solve challenging object detection problems, including pavement crack detection. Initially, deep learning-based object detection methods relied on window-sliding or region-proposal; and these methods tried to find a bounding box for each possible object in an image. R-CNN (region-based convolutional neural networks) [26] was the early work which utilized selective search [27] to generate candidate regions, and then sent the regions into a CNN for classification. Based on R-CNN, Cha et al. [28] designed a convolutional network for pavement crack detection which worked with window-sliding mode. Zhang et al. [3] employed a CNN for pre-classification which removed most of the noise areas before performing crack and sealed crack detection. Problems of these methods were: (1) window-sliding-based strategy was impractical due to the huge time complexity, especially when processing large images [4]; (2) traditional region-proposal methods [27] were unable to select good candidate regions from the noisy pavement images, and it was also inefficient because a great number of candidate regions had to be processed for a full-size image. Zhang et al. [29] employed parallel processing to improve the computation efficiency of region-based methods; however, the computation and resource costs were expensive. Zhang et al. [4] addressed the computational issue by generalizing a classification network to an end-to-end detection network which minimized the redundant convolutional operations. FCN is a one-stage pixel-level semantic segmentation method without window-sliding. Recently, Yang et al. [6] employed FCN for pixel-level crack detection and achieved good results on concrete-wall images and pavement images with clear cracks; the method failed to detect thin cracks. Moreover, the method relied on accurate pixel-level GTs which were labor-intensive and often infeasible under industrial setting. In addition, deep learning-based crack detection articles have been keeping on appearing. Chen et al. [30] and Park et al. [31] proposed NB-CNN and patch-CNN for crack detection, respectively; however, the networks could only process fixed input size images which limited the practical application. Tong et al. [32] utilized DCNN for crack length estimation; and Hoang et al. [33] employed CNN and edge detector for crack recognition. Gopalakrishnan et al. [34] used transfer learning for pavement distress detection with a DCNN. Zou et al. [35] and Yang et al. [36] introduced DCNNs for crack detection with hierarchical feature learning. The methods were either based on the traditional classification network with fully connected layers which only could handle fixed input-size images, or based on the FCN architecture which relied on the accurate, labor-intensive GTs.

In this paper, we propose CrackGAN for pavement crack detection with the following contributions: (1) it solves a practical and essential problem,"All Black" issue, existing in deep learning-based pixel-level crack detection methods; (2) it proposes the crack-patch-only (CPO) supervised adversarial learning and the asymmetric U-Net architecture to perform the end-to-end training; (3) the network can be trained with partially accurate GTs generated by labor-light method which can reduce the workload of preparing GTs significantly; (4) furthermore, it can solve data imbalance problem which is the byproduct of the proposed approach. Moreover, even the network is trained with small image patches and partially accurate GTs, it can deal with full-size images and achieve great performance.

The rest of the paper is organized as follows: In section II, it discusses the related works. In section III, it introduces the proposed method. In section IV, it describes the evaluation metrics and the experimental results. At the end, it provides the conclusion.

II. RELATED WORKS

In this section, it discusses the techniques related to the proposed method.

A. Generative Adversarial Networks

Goodfellow et al. [37] proposed generative adversarial network (GAN) which could be trained to generate real-like images by conducting a max-min two-player game. Based on GAN, Mirza et al. [38] proposed conditional GAN which introduced additional information (the condition) to the generator for producing specific outputs according to the input condition. While GAN is difficult to train, Radford et al. [40] proposed deep convolutional generative adversarial network (DC-GAN) which configured the generator with convolutional layers, and the training became easier and more stable. Based on conditional GAN, Isola et al. [39] set up the generator with an encoding-decoding network, then the GAN became an image-to-image translation network. Inspired by these works,

we formulate the crack detection as an image-to-image translation problem, and introduce generative adversarial loss to regularize the objective function to overcome the "All Black" issue using partially accurate GTs generated by labor-light method.

B. Transfer Learning in DCNN

Transfer learning has been widely used for training deep convolutional neural networks, which intends to transfer knowledge learned in previous tasks to make the training easier [42]. Depending on situations, there are different transfer learning strategies according to "what knowledge to transfer" and "how to transfer the knowledge". Yosinski et al. [43] discussed the knowledge transferability of different layers in deep neural networks. Oquab et al. [44] transferred the mid-level knowledge for nature image processing. Zhang et al. [3] transferred the generic knowledge learned from ImageNet [45] to ease the training of a crack detection network. Zhang et al. [4] also transferred the mid-level knowledge via introducing a dense-dilation layer into FCN to improve crack localization accuracy. The proposed approach employed transfer learning to train the prototype of the encoding network, and also transferred the knowledge from a pre-trained DC-GAN to provide the generative adversarial loss for the end-to-end training.

C. Fully Convolutional Network

Regular DCNN usually employed convolutional layers for feature extraction and fully connected layers for classification [46]. Interestingly, it turned out that the fully connected layer could be considered as a special case of the convolutional layer with kernel size equal to the input size [4]. Long et al. [5] proposed the fully convolutional network (FCN) for per-pixel semantic segmentation. Based on FCN, Chen et al. proposed DeepLab model [47] for multi-scale semantic segmentation; Ronneberger et al. [48] proposed U-Net architecture for medical image segmentation. Xie et al. [49] employed FCN for contour detection; Yu et al. [50] proposed dilated convolutional design for multi-scale context aggregation. To improve the computation efficiency, Zhang et al. [4] generalized a patch-based classification network to be a detection network for crack detection where FCN was employed. The proposed approach introduces FCN to extend the U-Net to the asymmetric U-Net, which provided the network with the translation ability of both crack and BG images. The FCN design also enables the patch-based CrackGAN (trained with small image patches) to work on the full-size images seamlessly.

III. PROPOSED METHOD

Fig.1 is the overview of the proposed method. D is a pre-trained discriminator obtained directly from a pre-trained DC-GAN using crack-GT patches only. Such pre-trained discriminator will force the network to always generate crack-GT images, which is the most important factor to overcome the "All Black" issue. The pixel-level loss is employed to ensure that the generated crack patterns are the same as that of the
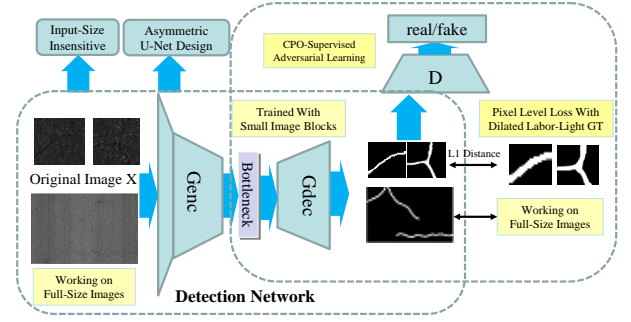

Fig. 1: Overview of the proposed method.

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. After training, the generator itself will serve as the crack detection network. In addition, the network is designed as a fully convolutional network which can process full-size images after the patch-based training. Finally, the overall objective function is:

$$L_{final} = L_{adv} + \lambda L_{pixel} \tag{1}$$

where L_{adv} is the adversarial loss generated by the pre-trained discriminator and L_{pixel} is the pixel-level loss computed with L1-distance.
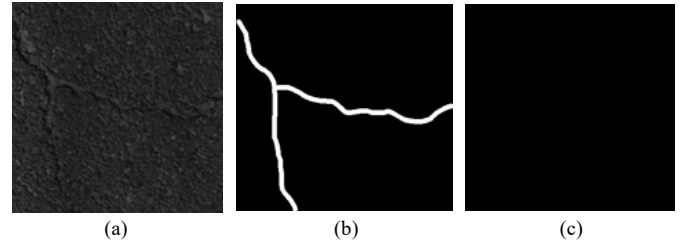

Fig. 2: "All Black" issue encountered when using FCN-based method for pixel-level crack detection: (a) the industrial pavement crack image; (b) the dilated GT-image utilized in the training; and (c) the detection result with the "well-trained" U-Net (see Fig. 3).
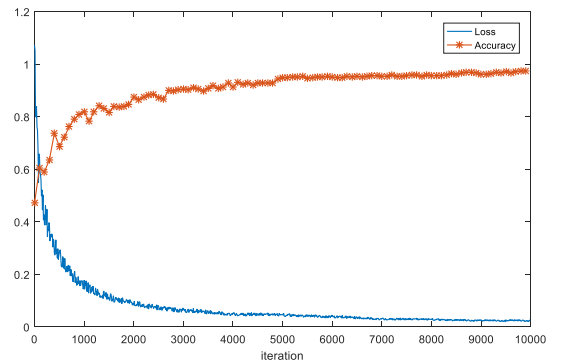

Fig. 3: The loss and accuracy curves when training a regular U-Net using industrial pavement images with partially accurate GTs.

A. "All Black" issue

This work results from addressing a practical engineering issue that the authors have encountered in industry. At the early attempts, we trained an FCN [5] for pixel-level pavement crack detection based on the data and GTs [4]. However, the results were not satisfactory: the networks were easily to converge to BG even there were cracks. The most possible reasons were: (1) most cracks in the industrial pavement images were thin and the crack-boundaries were vague, that made it very difficult for per-pixel GT annotation; in practice, the engineers just marked the cracks with 1-pixel curves for simplicity, and they were used as the GTs that were only partially accurate. Such GTs could not match the actual cracks at pixel-level well, which made the loss computation inaccurate, and failed the task. (2) Crack, as a long-narrow object, only occupies a very small area in a full image; and since patch-wise training was equivalent to loss sampling in FCN [5], training an FCN end-to-end with pavement crack images actually worked on extremely imbalanced dataset. Even the network simply classified all the pixels as BG, it still achieved quite a "good" accuracy (since BG pixels dominate the whole images), that was the "All Black" issue. As shown in Fig. 3, during training, the loss decreases rapidly and approaches to a very low value; however, in Fig. 2, the detection results are all blacks (i.e., all BGs). Moreover, it is worth to mention that other FCN architectures also encounter such problem; here it just takes U-Net as an example. Since the GTs are only partially accurate, existing approaches for solving data imbalance cannot work here.
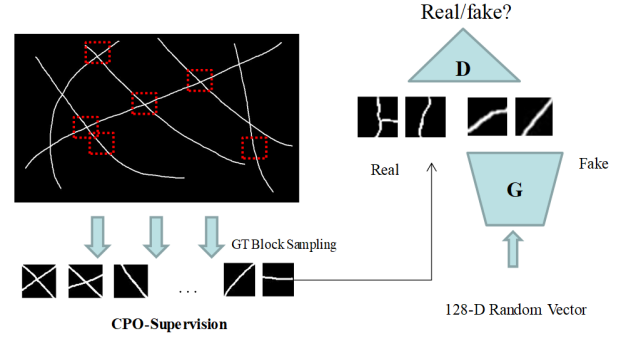
B. CPO-supervision and one-class discriminator

Regular FCN-based methods may only produce all-black images as the detection results [4], [6]. In order to address this problem, it adds a new constraint, generative adversarial loss, to regularize the objective function, which will make the network always generate crack-GT detection result; accordingly, the training data are prepared with crack patches only (i.e., CPO-supervision), without involving any non-crack patches and "all black" patches. As shown in Fig. 1, the adversarial loss is provided by a one-class discriminator obtained via pre-training the DC-GAN [40] only with crack-GT-like patches. It is well-known that the DC-GAN can generate real-like images from random noise by conducting the training with a max-min two-player game, in which a generator is used to generate real-like images and a discriminator is used to distinguish between real and fake images. As verified in [37], it was 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 alternatively [53]:
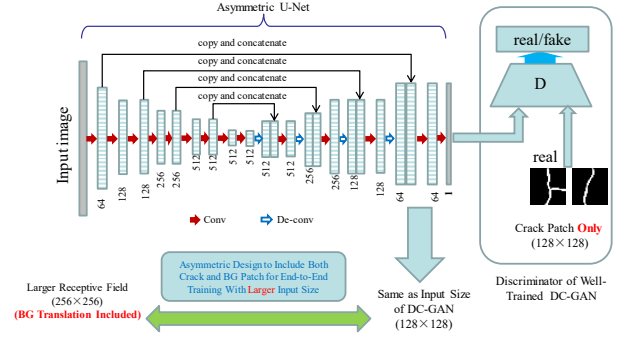
$$\max_D V(D,G) = E_{x \sim p_d(x)}[logD(x)] \\ + E_{z \sim p_d(z)}[log(1 - D(G(z)))] \tag{2}$$

$$\max_G V(D,G) = E_{z \sim p_d(z)}[log(D(G(z)))] \tag{3}$$

where x is the image from the real data (crack-GT-like patches) with distribution $p_d(x)$; z is the noise vector generated

Fig. 4: Pre-train a one-class DC-GAN with augmented GTs based on CPO-supervision. The real crack-GT data are augmented with manually marked "crack" curves.

Fig. 5: Asymmetric U-Net with larger input image (under larger field of view) with CPO-supervision.

randomly from Gaussian distribution $p_d(z)$; and D is the discriminator and G is the generator set up with convolutional and deconvolutional kernels, respectively. In practice, whether a sample is real or fake depends on the data setting. In accordance with the CPO-supervision, only crack-GT patches are contained in the real image-set for training the DC-GAN. With such setting, the discriminator will only recognize crack-GT patch as real and treat all-black patch as fake, which prevents the network to generate all-black (fake) image as the detection result, thus overcoming the "All Black" issue. Such discriminator is named one-class discriminator. In the implementation, the crack-GT data are further augmented by manually marking a bunch of "crack" curves and sampling the patches accordingly, as indicated in Fig. 4.

In Fig. 5, after training, the discriminator of the well-trained DC-GAN is concatenated to the end of the asymmetric U-Net generator to provide the adversarial loss for end-to-end training. Since the output of the generator serves as a fake image, the adversarial loss is:

$$L_{adv} = -E_{x \in I}[logD(G(x))] \tag{4}$$

Here, different from pre-training the DC-GAN in Eq. (2), x is the crack-patch, and I is the training set containing crack patches only; G is set up with the asymmetric U-Net architecture illustrated in Fig. 5, and D is the pre-trained one-class discriminator.
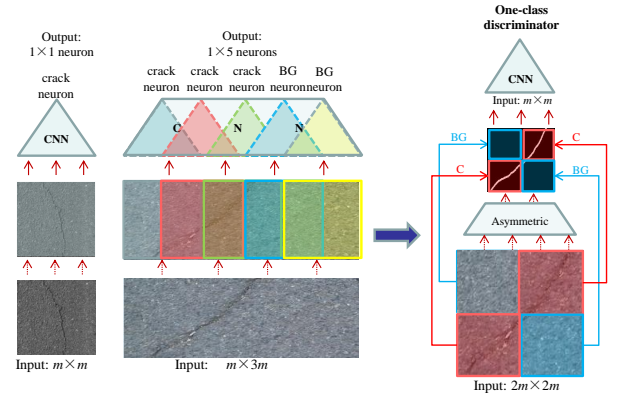
C. Asymmetric U-Net for BG-image translation

In subsection III-B, it introduced the CPO-supervision and generative adversarial learning to force the network to always generate crack-GT patches and prevent the "All Black" phenomenon. However, for a crack detection system, it should be able to process both crack and non-crack/BG pavement images. Normally, the discriminator should treat all-black patch as real to represent the BG-image translation result, such as directly applying the original pix2pix GAN [39]; unfortunately, treating all-black patch as real will encourage the network to generate all-black images as the detection results which is against solving the "All Black" issue. In order to include the translation of BG-image with CPO-supervision, it replaces the regular U-Net generator in the original pix2pix GAN with the proposed asymmetric U-shape generator which inputs a larger size *crack* patch (256×256) and outputs a smaller *crack-GT* image (128×128) for the end-to-end training. In accordance with the CPO-supervision, the larger input image has to be a crack image so that the *correct* output will always be a crack-GT patch recognized by the discriminator as real. With such setting, the network is able to translate both crack image and BG-image after the training. It is detailed later.

Receptive field analysis under larger field of view: To understand how the asymmetric design is able to include BG-image translation ability by only using crack samples for the training, it first performs a receptive field analysis under larger field of view. In Fig. 6, there is a DCNN network, as a classification network, with an $m \times m$ image patch as input, and the output is a single neuron representing the class label of the input image patch. When the same DCNN network is fed a larger size input image, it will output multiple neurons, and each neuron represents a class label of the corresponding image patch of size $m \times m$ "sampled" from the larger input image. For example, when the network's input is an image of $m \times 3m$, the output has five neurons (the number of neurons depends on the down-sampling rate of the DCNN) which represent class labels of five image patches including both crack and non-crack samples of size $m \times m$ (from left to right, the first three neurons represent crack-samples and the last two represent BG-samples). Indeed, under the multi-layer convolutional mode, each neuron actually has a receptive field with a 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 [4]. 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 receptive field of the original image translation network).

According to the analysis, as in Fig. 6, when a crack image with the size larger than the input-size of the discriminator is input to the asymmetric U-Net, and passes through the network; the network will produce a downsampled image patch that exactly *matches the input-size* of the discriminator. The output will be treated as a single image by the one-class

Fig. 6: Receptive field analysis under larger field of view: with a larger input image, the CNN realizes multi-spot sampling with the same receptive field. At the right, the asymmetric network architecture includes the image translation of both crack and BG samples; however, the training data contains crack patch only.

discriminator for the generative adversarial learning which still maintains the working mechanism of COP-supervision. However, since the network is trained to translate a larger crack image to a downsampled crack-GT image, it includes the translation of both crack and non-crack image samples inherently. In this way, the network can be trained to process both crack and BG images. Refer to Fig. 6.
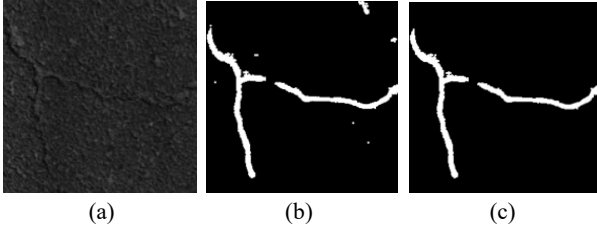
D. L1 loss with dilated GTs

It introduces the CPO-supervised generative adversarial learning and the asymmetric U-Net to prevent the "All Black" phenomenon; however, it is only an image-level supervision that does not specify the exact location of the cracks in the generated image. As analyzed before, one of the reasons for the "All Black" issue is the pixel-level mismatching due to the inaccurate GTs. Thus, it introduces the dilated-GT to specify a relatively larger crack area to ensure that it covers the actual crack locations, and if a detected crack pixel is in the dilated area, it is treated as a true positive. The experiments demonstrate that by combining the CPO-supervised adversarial loss and the loosely-supervised L1 loss, the network can be trained to generate cracks in the expected locations. Following [3], it marks the cracks with 1-pixel-width curves, and crops crack patches and partially accurate crack-GT patches from the original pavement images and the images with partially accurate GT, respectively. Then the partially accurate 1-pixel-width GTs were dilated three times using a disk structure with radius of 3 to generate the dilated GTs which are used to provide the loosely supervised pixel-level loss:

$$L_{pixel} = -E_{x \in I, y}[\|y - G(x)\|_1] \tag{5}$$

where x is the input crack patch; y is the dilated GT; I is the dataset of larger size crack patch (256×256 comparing with the output size of 128×128) used for end-to-end training; G is the asymmetric U-Net; and D is the discriminator.

Overall, the final objective function is:

$$L_{final} = L_{adv} + \lambda L_{pixel} \tag{6}$$

(a)   (b)   (c)

Fig. 7: Detection results of CrackGAN: (a) industrial pavement image suffered from "All Black" issue; (b) detection result of CrackGAN; (c) the final result image after removing the isolated noises.

(a)   (b)

Fig. 8: Weakly supervised learning is able to learn rich crack pattern information: (a) the image patches inputs into the classification network; (b) the feature maps after the first convolutional layer.
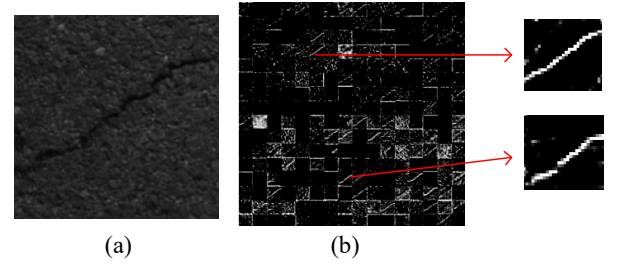
The pixel-level loss is normalized during training and $\lambda = 0.30$ is determined via grid search with step size 0.05. Fig. 7 shows the detection result of a sample image. Moreover, once the training is finished, the asymmetric U-Net generator itself will serve as the detection network to translate the original pavement image to the result image.

E. Working on full-size images

Notice that the network is trained with small image patches; however, under industry settings, the image size is much larger (2048×4096 pixels). A traditional solution to process large input image is to sample it into smaller image patches from the full-size image and do the processing patch-by-patch, named window-sliding strategy [28], [3]; however, it is very inefficient [4]. In our approach, the asymmetric U-Net is designed as a fully convolutional network, and it can work on images of arbitrary sizes seamlessly. In addition, such fully convolutional processing mechanism is quite efficient, which does not involve redundant convolutions as discussed in [4].

F. Implementation details

Network architecture: Fig. 5 presents the architecture of the asymmetric U-Net. The first layer is configured with 7×7 convolutional kernels with stride 2 and is followed by a rectified linear unit (ReLU) [52]; and it serves as the asymmetric part of the U-Net generator, which realize a 2-time downsampling of the larger input images and output the feature maps with the same size as the final output of the asymmetric U-Net. Then the remaining layers of the encoding and decoding parts are following the regular U-Net architecture [48]. The encoding part consists of four repeated convolutional layers with 3×3 kernels and the stride is 2; and each convolutional layer is followed by a ReLU layer. After each of the first three convolutional layers, the number of convolutional channels is doubled. The decoding part consists of four 3×3 deconvolutional layers that up-samples the feature maps; the input of each de-convolutional layer is the output of the last layer concatenated 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 with Tanh activation [46] is utilized to translate the 64-channel feature map to the 1-channel image, and it is compared with the dilated-GT for L1 loss computation according to Eq. (5). In summary, the network architecture is as follows. The encoding part:

C_64_7_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
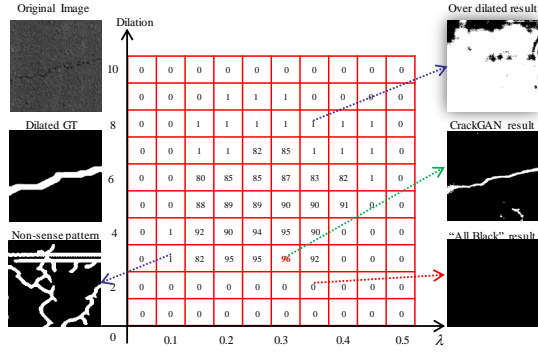The decoding part:
DC_512_3_2 - ReLU - C_512_3_1 - ReLU - DC_256_3_2 - ReLU - C_256_3_1 - ReLU - DC_128_3_2 - ReLU - C_128_3_1 - ReLU - DC_64_3_1 - ReLU - C_64_3_1 - ReLU - C_1_3_1 - Tanh
Here, the naming rule follows the format: "layer type_channel number_kernel size_stride". "C" denotes convolution; "DC" is de-convolution; and Tanh is the Tanh activation. 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, the one-class discriminator and the encoding part of the generator. First, the DCGAN is trained with the crack-GT patches of 128×128-pixel as described in subsection III-B, aiming at training a discriminator with strong crack-pattern recognition ability to provide the adversarial loss for the end-to-end training at the second stage. A total of 60,000 dilated crack-GT patches with various crack patterns are used. The other training settings follow [40]: the Adam optimizer [54] is used, the learning rate is 0.0002, the parameters for momentum updating are 0.9, the batch size is 128 and the input "noise" vector is 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 is concatenated to the end of the asymmetric U-Net to provide the adversarial loss at the second stage. Refer Fig. 5 and Eq. (4).

Inspired by [3], it also pre-trains the encoding part of the generator under the classification setting. Zhang et al. [3] showed that by performing an image-block classification task, the network was able to extract the relevant crack patterns; and the learned knowledge could be transferred to ease the training of an end-to-end detection network [4]. Fig. 8 is the low-level feature maps of a classification network trained with crack and non-crack patches [3]. The classification network is configured by adding a fully connected layer at the end of the encoding part (bottleneck) of the asymmetric U-Net and the output dimension is 2 representing crack and non-crack with labels 0 and 1. The training samples are crack

Fig. 9: Grid search board with HD-scores utilized to determinate the optimal parameters λ and dilation-scale. The optimal parameters are determined according to the best HD-score. Testing results of the three main failure cases are also present in the figure including the "All Black" result, over-dilated result, and the non-sense patterns which overlooked the pixel-level loss.
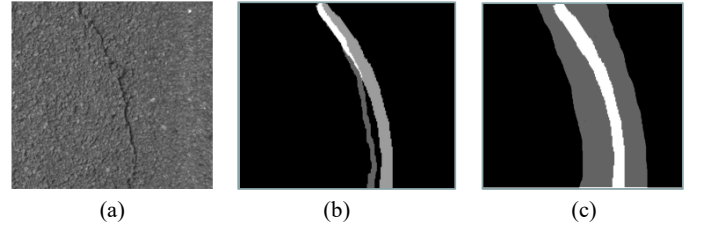
and non-crack patches of 256×256. It shows that the network extracted same crack pattern as the original image, i.e., the network is able to learn useful information with the weakly supervised information, crack/non-crack image labels only. Then the well-trained parameters are used to initialize the encoding part of the generator for the end-to-end training; and the other settings are same with the DC-GAN except replacing the generator with the asymmetric U-Net and changing the objective function according to Eq. (6).

Parameter selection: The parameters, λ in Eq. (6) and the dilation scale, are determined by grid search. From the grid search board in Fig. 9, the dilation scale (number of dilation times with the disk structure) should be between 3 and 7, and the λ should be between 0.15 and 0.4 for an effective detection. Dilation scale less than 3 will cause the "All Black" issue due to the pixel-level mismatching and the data imbalance, while too big dilation (>8) could not provide meaningful crack location information and would produce useless output. In addition, a larger λ tends to fail the task, but a small λ ($0.15 < \lambda < 0.4$) can work well, which indicates that the adversarial loss is very important to succeed the training under the industrial setting. It can be observed from the grid board that the HD-score is either a good one (>80) or a very small one (0 or 1) which indeed represents the two different model statuses, well-trained or failed. However, the causes of the failures could be grouped into three main cases: over-dilated, "All Black" problem, or the useless output pattern which overlooks the pixel-level loss with a small λ, as indicated in Fig. 9.
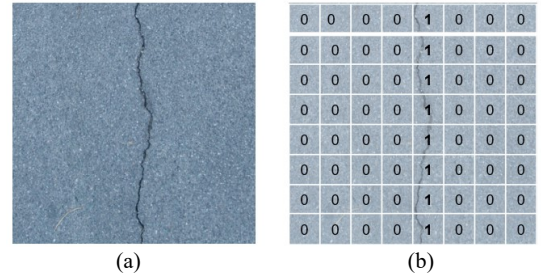
IV. EXPERIMENTS

A. Dataset and Metrics

CFD [2], the CrackGAN dataset (CGD) collected by the authors, and dataset [16] are utilized for evaluation. CFD contains 118 pavement crack images (480×320-pixel each) obtained by people standing on the road using an iPhone, the ground truths are carefully marked at pixel level which is labor-intensive. The image quality is high and the background is smooth and clean. CGD is a dataset with 400 pavement

Fig. 10: Illustration of pixel-level mismatching: (a) a crack image; (b) detection result overlapped with GT-image dilated once using a disk structure with radius 3; and (c) detection result overlapped with GT-image dilated four times. The transparent areas represent the dilated GT-cracks with different dilation scales and the green areas represent the detected cracks.
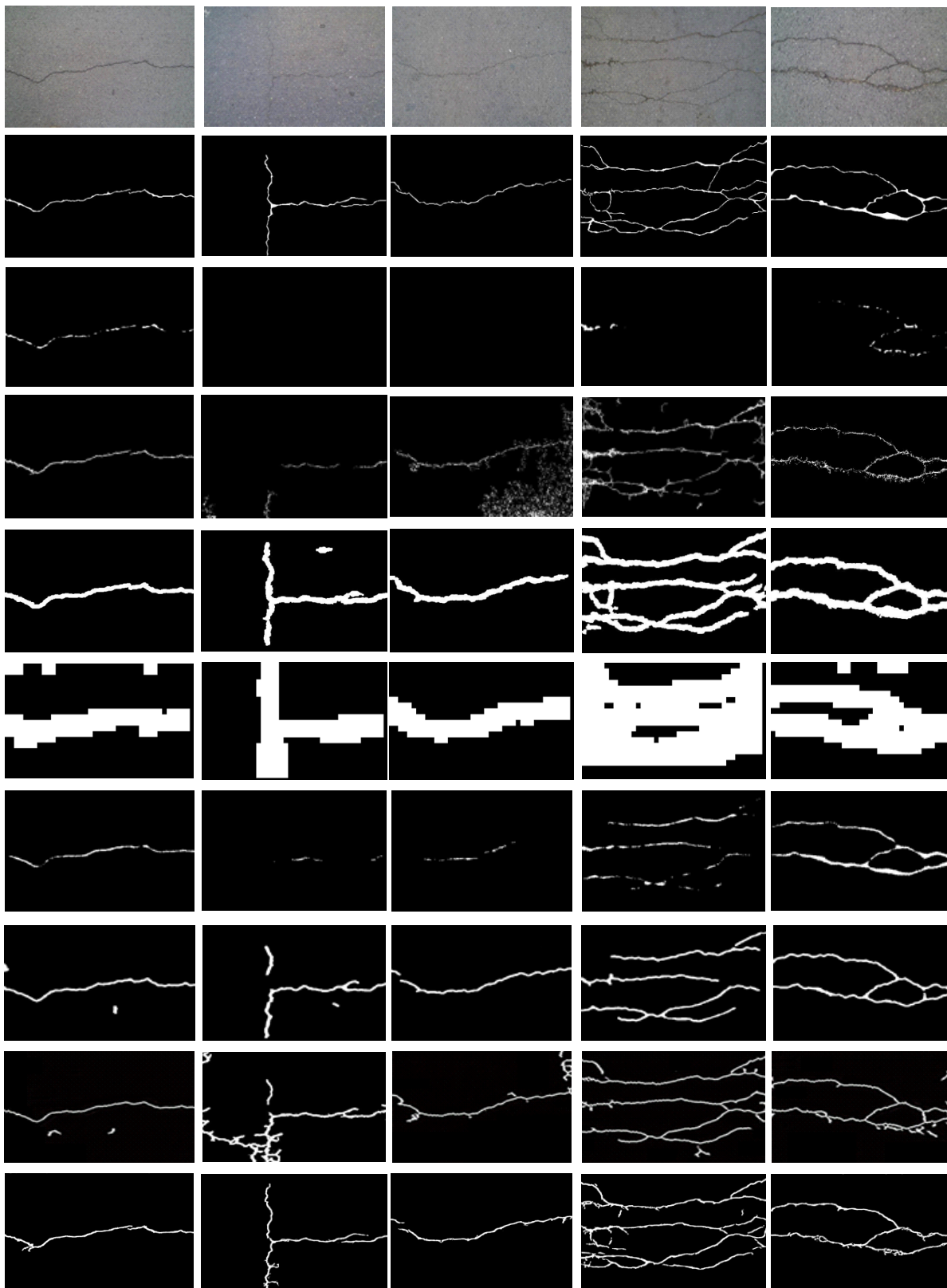

Fig. 11: Region-based evaluation: (a) the original crack image; (b) illustration of counting the crack and non-crack regions. The squares with label "1s" are the crack regions, and with label "0s" are BG-regions.

crack images (2048×4096-pixel each) collected by the authors using a line-scan industrial camera mounted on the top of a vehicle running at 100km/h; and the camera scans 4.096-meter width road surface and produces a pavement image of 2048×4096-pixel for every 2048 line-scans (i.e., 1 pixel represents 1×1 mm$_2$ area). Most of the cracks are thin, and sometimes even hard to be recognized by human. Furthermore, it is infeasible to obtain the accurate GTs at pixel-level; thus, the cracks are represented by 1-pixel curves roughly marked by the engineers in a labor-light way, and it is named partially accurate GTs). However, such GTs may not match the true crack locations accurately, and processing them is much more challenging. The proposed algorithm can achieve the best results using both "accurate" and "partially accurate" datasets that demonstrate its robustness as well. For CFD and CGD, the data are augmented following [3] to facilitate the training; and the training-test ratio is 2:1. Dataset [16] has industrial images from five different capture systems: Aigle-RN has 38 images with annotation, ESAR has 15 images with annotations, and LCMS has 5, LRIS has 3 and Tempest has 7 images with annotations, respectively. The GTs are marked at pixel level. To our best knowledge, it is the only public pavement crack dataset from industry; and it contains relatively few images, they are used for testing only.

Different from most object detection tasks [55], the intersection over union (IOU) is not suitable for evaluating crack detection algorithms [56]. As shown in Fig. 10, crack, as a long-narrow target, only occupies a very small area, and the image consists mainly of BG pixels [56]. With the fact that the

Fig. 12: Comparison of the detection results on CFD using different methods. From top to bottom are: original images, GT images, results of CrackIT, results of MFCD, results of CrackForest, results of [28], results of FCN-VGG, results of DeepCrack, results of Pix2pix GAN, and results of CrackGAN, respectively.

precise pixel-level GTs are difficult to obtain, it is impossible to obtain the accurate intersection area. As shown in Fig. 10 (b) and Fig. 10 (c), it is obvious that the detection results are very good; however, the IOU values are very low, 0.13 and 0.2, respectively. According to [56], it employs Hausdorff distance to evaluate the crack localization accuracy. For two sets of points *A* and *B*, the Hausdorff distance can be calculated with:

$$H(A, B) = max[h(A, B), h(A, B)] \qquad (7)$$

where

$$h(A, B) = max_{a \in A} min_{b \in B} \|a - b\| \qquad (8)$$

The penalty is defined as:

$$h_p(A, B) = 1/(|A|) \sum_{a \in A} sat_u min_{b \in B} \|a - b\| \qquad (9)$$

Here, parameter *u* is the upper limit of the saturation function *sat* which is 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 [56], it is set as 50-pixel to emphasize the localization accuracy by eliminating the influence of possible noises from the large BG areas. *A* is the detected crack set and *B* is the GT set, the overall score is:

$$score_{BH}(A, B) = 100 - \frac{BH(A, B)}{u} \times 100 \qquad (10)$$

where

$$BH(A, B) = max[h_p(A, B), h_p(B, A)] \qquad (11)$$

The Hausdorff distance score (HD-score) can reflect the overall crack localization accuracy, and it is insensitive to the foreground-background imbalance inherent in long-narrow object detection.

In addition, the region-based precision rate (p-rate) and recall rate (r-rate) are used for evaluation, which can measure the false-detection severity and the missed-detection severity, respectively. In Fig. 11, a pavement image of 400×400-pixel is divided into small image patches (50×50-pixel); if there is a crack detected in a patch, marked as "1s", it is positive. In the same way, for GT images, if there is a marked curve in a patch, it is a crack patch. Then the region based true positive (TP), false positive (FP) and false negative (FN) can be obtained by counting the corresponding squares, and further be used to calculate the region based precision and recall rates:

$$P_{region} = \frac{TP_{region}}{TP_{region} + FP_{region}} \qquad (12)$$

$$R_{region} = \frac{TP_{region}}{TP_{region} + FN_{region}} \qquad (13)$$

Then the region-based F1 score can be computed as:

$$F1_{region} = \frac{2 * P_{region} * R_{region}}{P_{region} + R_{region}} \qquad (14)$$

TABLE I: Quantitative evaluations on CFD

Methods	P_{region}	R_{region}	$F1_{region}$	*HD-score*
CrackIT	**88.05**%	45.11%	59.65%	21
MFCD	80.90%	87.47%	84.05%	85
CrackForest	85.31%	90.22%	87.69%	88
[28]	68.97%	**98.21**%	81.03%	70
FCN-VGG [6]	86.01%	92.30%	89.04%	88
DeepCrack	88.03%	94.11%	90.96%	94
Pix2pix GAN	88.01%	90.02%	89.01%	90
CrackGAN	88.03%	96.11%	**91.89**%	**96**

TABLE II: Quantitative evaluation on CGD

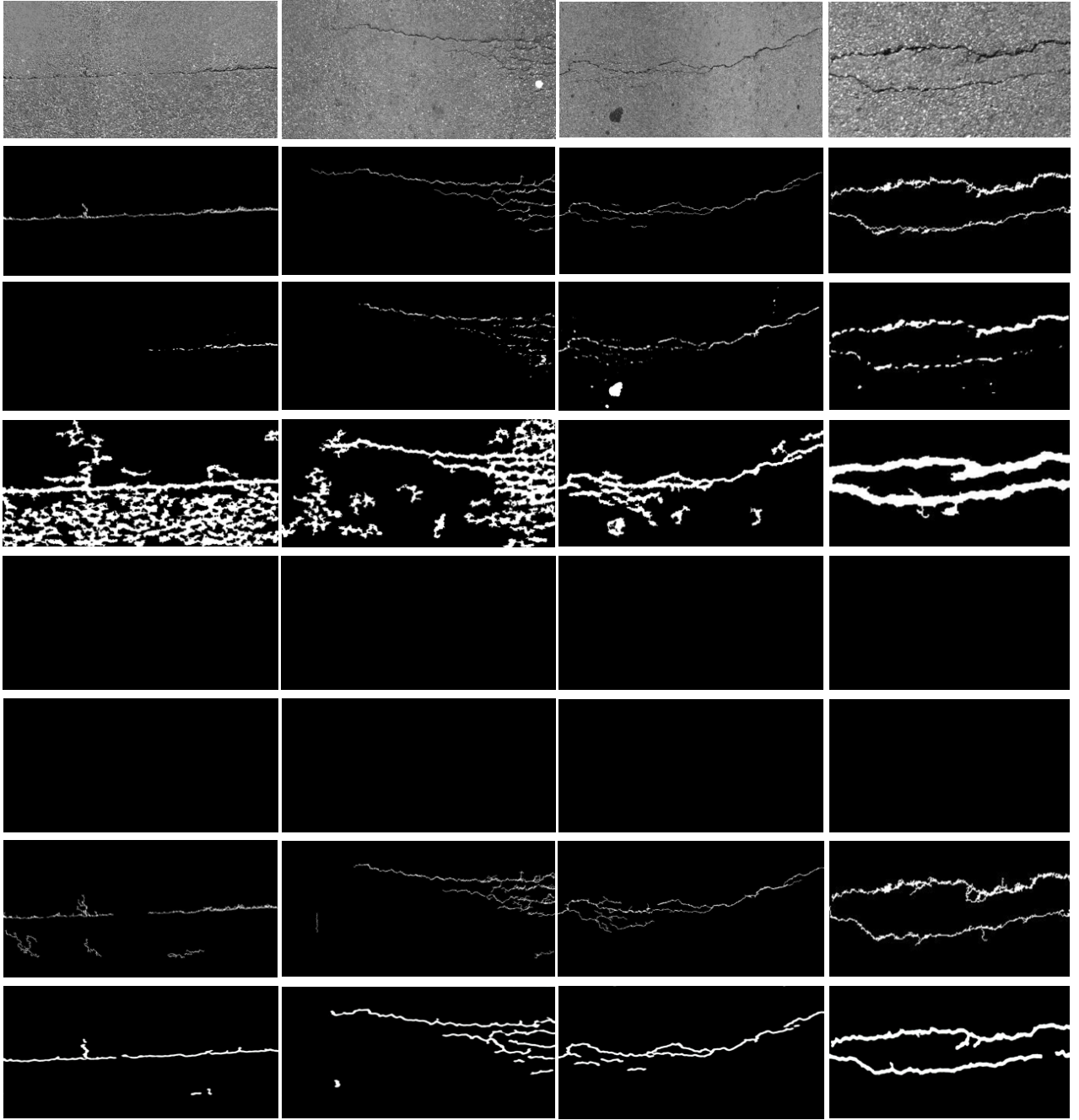| Methods	P_{region}	R_{region}	$F1_{region}$	*HD-score*
CrackIT	**89.10**%	2.52%	4.90%	9
CrackForest	31.01%	98.01%	47.22%	63
[28]	69.20%	**98.30**%	81.22%	64
FCN-VGG [6]	0.00%	0.00%	N/A	N/A
DeepCrack-1	37.01%	97.01%	53.57	65
DeepCrack-2	0.00%	0.00%	N/A	N/A
Pix2pix GAN	0.00%	0.00%	N/A	N/A
CrackGAN	87.01%	96.01%	**91.28**%	**96**

B. Overall Performance

The comparisons are performed on CFD [2], CGD, and dataset [16] to justify the state-of-the-art performance.

CFD: The proposed method is compared with CrackIT-v1 [57], MFCD [58], CrackForest [2], [28], FCN-VGG [6], Pix2pix GAN (with U-Net as the generator) [39], and Deep-Crack [35] on CFD; and the related results are shown in Fig. 12 and Table I. CrackIT introduced the traditional mean and standard deviation (STD) for crack patch selection, and utilized some post-processing for pixel-based crack detection. However, the features with mean and STD are not able to select the crack patches well, especially when the cracks are thin; thus, the false negative rate is high, and it cannot even detect any cracks in the second and third sample images from Fig. 12. MFCD developed a complex path verification algorithm to link candidate crack seeds for the detection; however, it might also connect the false positives and generate fake cracks. As shown in Fig. 12, it produces many noises in the third image with non-smooth background. CrackForest employed integral channel information with 3 colors, 2 magnitudes and 8 orientations for feature extraction and applied random forest for crack token mapping; and the histogram difference between crack and non-crack regions was used for noise removal. As shown in Fig. 12, it achieves very good results on the images whose backgrounds are smooth and clean. However, the performance deteriorates when processing the industrial images as shown in Figs. 13 and 14. [28] was a patch-level crack detection method which trained a deep classification network for crack and non-crack patch classification; it could not provide accurate crack locations as shown in Fig. 12. FCN-VGG was a pixel-level crack detection method of which the accurate pixel-level GTs were needed to train the FCN-based network end-to-end. Similar to the results reported in the original papers, it failed when detecting thin cracks. DeepCrack achieves very good results on CFD due to the multi-scale hierarchical fusion; however, the training relied on accurate GTs and the method would fail easily when the GTs are biased. Pix2pix GAN [39]

Fig. 13: Comparison of the detection results on CGD. From top to bottom are: original images, GT images, results of CrackIT, results of CrackForest, results of [28], results of FCN-VGG, results of DeepCrack-1, results of DeepCrack-2, results of Pix2pix GAN, and results of CrackGAN, respectively.

Fig. 14: Comparison of the detection results on dataset [16]. From top to bottom are: original images, GT images, results of CrackIT, results of CrackForest, results of FCN-VGG, results of Pix2pix GAN, results of MPS [16], and results of CrackGAN, respectively

was an image-to-image translation network with U-Net as the generator which introduced generative adversarial learning for image style translation originally. However, as discussed in section III-C, the discriminator would treat both crack and non-crack as real which immediately would weaken the crack-patch generation ability, that makes the network similar to the regular U-Net; therefore, it achieves similar results as FCN-based methods, and also encounters "All Black" problem as shown in Figs. 13 and 14. CrackGAN introduces CPO-supervision and the asymmetric U-Net architecture to build the one-class discriminator for generative adversarial learning,

which enhances the crack patch discrimination ability by treating the all-black patch as fake images to avoid the data imbalance problem inherent in crack-like object detection, and finally improves the crack detection ability, especially for thin and tiny crack detection. As shown in Fig. 12 and Table I, it achieves the best results.

It is worth to mention that in Tables I, II, and III, some p-rates and r-rates of the CrackGAN are not the maximum values, but they do not affect the state-of-the-art performance. For example in Table I, CrackIT achieved best p-rate (88.05%) even it missed quite a lot of cracks; because the precision is

TABLE III: Quantitative evaluation on dataset [16]

Methods	P_{region}	R_{region}	$F1_{region}$	*HD-score*
CrackIT	**90.33%**	4.22%	8.06%	11
CrackForest	36.21%	**97.21%**	52.76%	65
MPS[16]	79.01%	84.20%	81.52%	82
FCN-VGG [6]	0.00%	0.00%	N/A	N/A
Pix2pix GAN	0.00%	0.00%	N/A	N/A
CrackGAN	86.53%	94.20%	**91.29%**	**95**

TABLE IV: Comparisons of computational efficiency

Method	Time	Method	Time
CrackIT	6.1 s	DeepCrack	2.4 s
CrackForest	4.0 s	Pix2pix GAN	2.3 s
[28]	10.2 s	**CrackGAN**	**1.6 s**
FCN-VGG	2.8 s		

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

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

Fig. 15: Crack detection on concrete pavement images and concrete wall images: (a) and (b) are concrete wall images; (c) and (d) are concrete pavement images; (e)-(h) are the detection results by CrackGAN.
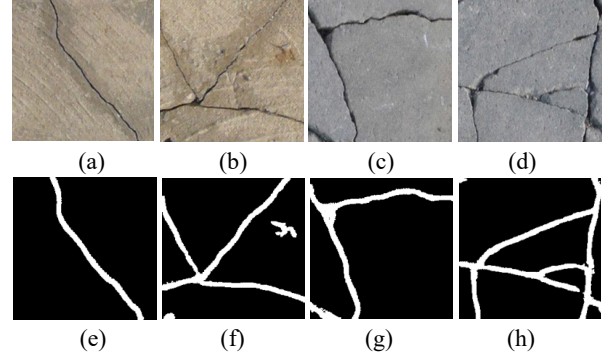
calculated with TP/(TP+FP), if FP is small; even FN is very large, the p-rate can still be large. Similarly, [28] achieved very good r-rate (98.21%) even the patch level detection will cause a lot of false positives, because the recall rate is calculated with TP/(TP+FN) which does not take into account the FP. Therefore, only p-rate or r-rate cannot represent the performance of state-of-the-art crack detection algorithm. Refer Table I for the quantitative results.

CGD: The related results on CGD are shown in Fig. 13 and Table II. Similar to the results on CFD, CrackIT misses most cracks and MFCD introduces many noises because of the thin cracks and textured background. CrackForest introduces many noises, among which quite a lot of them connected to the true crack regions; and it was because the method utilized the distribution differences of statistical histogram and statistical neighborhood histogram of the positive regions for noise removal, which did not consider the removal of the noise connected to the true positives. Therefore, it achieves a low p-rate, 31.01%. Same as the results on CFD, [28] could not give accurate crack locations, and achieves a low p-rate, 69.20%. Suffering from the "All Black" issue, the FCN-VGG recognizes all crack and non-crack patches as background and produces all-black images as the results. For a fair evaluation, we conduct the comparison with the latest work DeepCrack on two different settings: one (DeepCrack-1) exactly follows the original paper trained with CrackTree data [10], and another (DeepCrack-2) re-trains the model using the industrial dataset. As present in Table II and Fig. 13, DeepCrack-1 introduces unacceptable noises due to the performance degradation on different domains as discussed at the beginning of this work; and DeepCrack-2 encounters the "All Black" problem. As discussed in section III-C, with the default settings, the discriminator in the original Pix2pix GAN will recognize both crack-GT and all-black-GT as real which damages the crack-GT generation ability and makes it like a regular U-Net; thus, it also produces all-black images as the results. By introducing the CPO-supervision and the adversarial learning with asymmetric U-Net generator, the model can be trained to generate crack-like results without losing the BG translation ability, and finally overcome the "All Black" issue. As shown in Fig. 13, it can detect thin cracks from the pavement images obtained from industrial settings. Refer Table II for quantitative results.

Dataset [16]: Fig. 14 and Table III present the results of CrackIT, CrackForest, MPS [16], FCN, Pix2pix GAN, and CrackGAN. Similar to the results on CGD, CrackIT missed quite a lot of cracks due to the drawback of feature extraction and post-processing. CrackForest could not remove the noises connected to the true crack regions and achieves a very low

p-rate. MPS [16] is a traditional image processing method based on minimal path selection; it performed the detection by following three steps, endpoint selection, minimal path estimation, and minimal path selection. It achieved good results as shown in Fig. 14 and Table III; however, it utilized quite a few tunable parameters and post-processing procedures that needed extra works manually. As discussed before, FCN-VGG and Pix2pix GAN fail the task due to the "All Black" issue. Instead, CrackGAN can properly handle the "All Black" problem and achieves the best performance.

In addition to pavement crack detection, the proposed method can also deal with other crack detection tasks; Fig. 15 provides crack detection results on concrete pavement images and concrete wall images based on the model trained with dataset [6] and the labor-light GTs.

C. Computational Efficiency

In addition to the detection accuracy, it also compared the computation efficiency of the methods with public testing codes. The average processing times for processing a full-size image of 2048×4096-pixel are present in Table IV. CrackIT-v1 [57] takes 6.1 seconds based on a patch-wise processing; and CrackForest [2] takes a relative less time (4.0 seconds) via using the parallel computing to implement the random forest for image patch classification. The two methods are implemented with Matlab-2016b on HP 620 workstation with 32G memory and twelve i7 cores. For the deep learning methods, they are implemented with the same computer but run on an Nvidia 1080Ti GPU with Pytorch. [28] takes 10.2 seconds because it is based on the window-sliding. FCN [6], DeepCrack, Pix2pix GAN, and CrackGAN take much less time due to the FCN architecture; moreover, the CrackGAN takes much less time (i.e., 1.6 seconds) because it cuts off the last de-convolutional layer for the asymmetric U-Net design.

V. Conclusion

In this work, we propose a novel deep generative adversarial network, named CrackGAN, for pavement crack detection. The method solves a practical and essential problem, "All Black" issue, existing in FCN-based pixel-level crack detection when using partially accurate GTs. More important, the network can solve crack detection tasks in a labor-light way. It can reduce the workload of preparing GTs significantly, and create the new idea for object detection/segmentation using partially accurate GTs. In addition, it can also solve the data imbalance problem which is the byproduct of the proposed approach. Moreover, the network is trained with small image patches, but can deal with any size images. The experiments demonstrate the effectiveness and superiority of the proposed method, and the proposed approach achieves state-of-the-art performance comparing with the recently published works.

Moreover, the theoretical analysis of neuron's property concerning receptive field can be employed to explain many phenomena in deep learning, such as the boundary vagueness in semantic segmentation [5], blurry of the generated images with GAN [39], [41], etc., which have not been explained clearly yet. We believe that the analysis of each neurons property discussed in this paper could become a routine for designing effective neural networks in the future.

References

[1] N. F. Hawks, and T. P. Teng, "Distress identification manual for the long-term pavement performance project," SHRP-P-338, National academy of sciences, Washington, DC, 2014.

[2] Y. Shi, L. Cui, F. Meng and Z. S. Chen, "Automatic road crack detection using random structured forest," IEEE Trans. Intell. Transp. Syst., vol. 17, no. 12, pp. 34343445, 2016.

[3] 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.

[4] 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, Maui.

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

[6] X. Yang, H. Li, Y. Yu, X. Luo, and X. Yang, "Automatic PixelLevel Crack Detection and Measurement Using Fully Convolutional Network," Comput.-Aided Civ. Infrastructure. Eng., vol. 33, no. 12, pp. 1090-1109, 2018.

[7] H. D. Cheng, J. Chen, C. Glazier, and Y. Hu, "Novel approach to pavement crack detection based on fuzzy set theory," J. Comput. Civil Eng., vol. 13, no. 4, pp. 270280, 1999.

[8] K. Wang, Q. Li, and W. Gong, "Wavelet-based pavement distress image edge detection with trous algorithm," Transp. Res. Rec., vol. 2024, pp. 2432, 2000.

[9] H. Oliveira, and P. L. Correia, "Automatic road crack segmentation using entropy and dynamic thresholding," in Proc. 17th European Signal Processing Conf., Glasgow, 2009.

[10] Q. Zou, Y. Cao, Q. Li, and S. Wang, "CrackTree: Automatic crack detection from pavement images," Pattern Recognition Lett., vol. 33, no, 3, pp. 227238, 2012.

[11] Y. Huang, and B. Xu, "Automatic inspection of pavement cracking distress." J. Electron. Imaging, vol. 15, no. 1, pp. 17-27, 2006.

[12] G. Li, S. He, Y. Ju, and K. Du, "Long-distance precision inspection method for bridge cracks with image processing," Automat. Constr., vol. 41(Supplement C), pp. 8395, 2014.

[13] J. Chen, M. Su, R. Cao, S. Hu, and C. Lu, "A self-organizing map optimization based image recognition and processing model for bridge crack inspection," Automation in Construction, vol. 73(Supplement C), pp. 5866, 2007.

[14] L. Wu, S. Mokhtari, A. Nazef, B. Nam, and B. Yun, "Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment," J. Comput. Civil Eng., vol. 30, no. 1 (04014118), 2016.

[15] T. S. Nguyen, S. Begot, F. Duculty, and M. Avila, "Free-form anisotropy: A new method for crack detection on pavement surface images," in Proc. IEEE ICIP, 2011, pp. 1069-1072.

[16] 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, Oct. 2016.

[17] Y. C. Tsai, V. Kaul, and R. M. Mersereau, "Critical assessment of pavement distress segmentation methods," J. Transp. Eng., vol. 136, no. 1, pp. 11-19, 2010.

[18] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of edge-detection techniques for crack identification in bridges," J. Comput. Civ. Eng., vol. 17, no. 4, pp. 255263, 2003.

[19] R. C. Gonzalez, R. E. Woods, and S. L. Steven, Digital image processing using MATLAB, Addison-Wesley, Boston, 2009.

[20] H. D. Cheng, J. Wang, Y. Hu, C. Glazier, X. Shi, and X. Chen, "Novel approach to pavement cracking detection based on neural network," Transp. Res. Rec., vol. 1764, pp. 119127, 2001.

[21] H. Oliveira, and P. L. Correia, "Automatic road crack detection and characterization," IEEE Trans. Intell. Transp. Syst., vol. 14, no. 1, pp.155-168, 2013.

[22] Y. Hu, C. X. Zhao, and H. N. Wang, "Automatic pavement crack detection using texture and shape descriptors," IETE Tech. Rev., vol. 27 (2010), pp. 398-405, 2010.

[23] M. Gavil, D. Balcones, O. Marcos, D. F. Llorca, M. A. Sotelo, I. Parra, M. Oca, P. Aliseda, and P. Yarza, "Adaptive road crack detection system by pavement classification," Sensors, vol. 11, no. 10, pp. 2857, 2011.

[24] E. Zalama, J. Gomez-Garcia-Bermejo, R. Medina, and J. Llamas, "Road crack detection using visual features extracted by Gabor filters," Comput.-Aided Civ. Infrastruct. Eng., vol. 29, no. 5, pp. 342-358, 2014.

[25] P. Dollr, and C. L. Zitnick, "Structured forests for fast edge detection," in Proc. IEEE ICCV, 2013, pp. 18411848.

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

[27] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. "Selective search for object recognition," Int. J. Comput. Vision, vol. 104, no. 2, pp. 154-171, 2013.

[28] Y. J. Cha, W. Choi, and O. Bykztrk, "Deep learning-based crack damage detection using convolutional neural networks," Comput.-Aided Civ. Infrastruct. Eng., vol. 32 (2017), pp. 361378, 2017

[29] A. Zhang, C. P. Kelvin, B. Wang, E. Li, X. Yang, Y. Dai, Y. Fei, and C. Chen, "Automated Pixel - Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network," Comput.-Aided Civ. Infrastruct. Eng., vol 32 (2017), pp. 805-819, 2017.

[30] F. C. Chen, and M. R. Jahanshahi, "NB-CNN: Deep learning-based crack detection using convolutional neural network and Nave Bayes data fusion," IEEE Transactions on Industrial Electronics, vol. 65, no. 5, pp. 4392-4400, 2018.

[31] S. Park, S. Bang, and H. Kim, "Patch-Based Crack Detection in Black Box Images Using Convolutional Neural Networks," Journal of Computing in Civil Engineering, vol. 33, no. 3 (040190170), 2019.

[32] Z. Tong, J. Gao, Z. Han, and Z. Wang, "Recognition of asphalt pavement crack length using deep convolutional neural networks," Road Materials and Pavement Design, vol. 19, no. 6, pp. 334-1349, 2018.

[33] N. D. Hoang, Q. L. Nguyen, and V. D. Tran, "Automatic recognition of asphalt pavement cracks using metaheuristic optimized edge detection algorithms and convolution neural network," Automation in Construction, vol. 94, pp. 203-213, 2018.

[34] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection," Construction and Building Materials, vol. 157, pp. 322-330, 2017.

[35] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, S. Wang, "DeepCrack: Learning hierarchical convolutional features for crack detection," IEEE Transactions on Image Processing, vol. 28, no. 3, pp. 1498-512, 2019.

[36] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, H. Ling, "Feature pyramid and hierarchical boosting network for pavement crack detection," arXiv preprint arXiv:1901.06340. 2019.

[37] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Proc. NIPS, 2014.

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

[39] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in Proc. IEEE CVPR, 2017.

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

[41] 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.

[42] 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.

[43] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, How transferable are features in deep neural networks? in Proc. NIPS, Montreal, Canada, 2014.

[44] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations," in Proc. IEEE CVPR, New York, 2014.

[45] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, "ImageNet: A Large-Scale Hierarchical Image Database," in Proc. IEEE CVPR, 2009.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural network," in Proc. NIPS, Harrahs Lake Tahoe, 2012.

[47] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in Proc. Int. Conf. Learn. Represent., 2015.

[48] 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. 234241.

[49] S. Xie, and Z. Tu, "Holistically-nested edge detection," in Proc. IEEE ICCV, 2015.

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

[51] Y. Liu, M. M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in Proc. IEEE CVPR, 2017.

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

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

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

[55] M. Everingham, L. Van Gool, C. K. I. Williams, J.Winn, and A. Zisserman, The PASCAL Visual Object Classes Challenge 2011 Results. Online Available: http://www.pascalnetwork.org/challenges/VOC/voc2011, accessed 2018.

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

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

[58] 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.

Yingtao Zhang received her M.S. degree in Computer Science from Harbin Institute of Technology, Harbin, China, in 2004, and the Ph.D. degree in Pattern Recognition and Intelligence System from Harbin Institute of Technology, Harbin, China, in 2010. Now, she is an associate professor at School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. Her research interests include pattern recognition, computer vision, and medical image processing.

Heng-Da Cheng received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1985, under the supervision Prof. K. S. Fu. He is currently a Full Professor with the Department of Computer Science, and an Adjunct Full Professor with the Department of Electrical Engineering, Utah State University, Logan, UT. He is an Adjunct Professor and a Doctorial Supervisor with the Harbin Institute of Technology. He is also a Guest Professor with the Institute of Remote Sensing Application, Chinese Academy of Sciences, Wuhan University, and Shantou University, and a Visiting Professor of Northern Jiaotong University, Huazhong Science and Technology University, and Huanan Normal University.

He has authored over 350 technical papers and is the Co-Editor of the book entitled *Pattern Recognition: Algorithms, Architectures, and Applications* (World Scientific Publishing Company, 1991).

His research interests include image processing, pattern recognition, computer vision, artificial intelligence, medical information processing, fuzzy logic, genetic algorithms, neural networks, parallel processing, parallel algorithms, and VLSI architectures.

Dr. Cheng was the General Chair of the 11th Joint Conference on Information Sciences (JCIS) (2008), the tenth JCIS (2007), the Ninth JCIS (2006), and the Eighth JCIS (2005). He served as a Program Committee Member and the Session Chair for many conferences, and as a Reviewer for many scientific journals and conferences. He has been listed in Who's Who in the World, Who's Who in America, and Who's Who in Communications and Media.

Dr. Cheng is also an Associate Editor of *Pattern Recognition*, *Information Sciences*, and *New Mathematics and Natural Computation*.

Kaige Zhang received the B.S. degree in electronic engineering from Harbin Institute of Technology, Harbin, China, in 2011, and M.S. degree in signal and information processing from Harbin Engineering University, Harbin, China, in 2014. He is currently pursuing the Ph.D. degree in computer science with Utah State University. His research interests include computer vision, machine learning, and the applications on intelligent transportation systems, precision agriculture and biomedical data analytic.