2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Efficient Dense-Dilation Network for Pavement Cracks Detection with Large Input Image Size

Kaige Zhang[1], Heng-Da Cheng[2] and Shan Gai[3]

Abstract— **Window-sliding/region-proposal based methods have been the popular approaches for object detection with deep convolutional neural networks. However, these methods are very inefficient when the input image size is large, such as pavement images (2000×4000-pixel) used for cracking detection. In this paper, we propose a solution to this problem by introducing a fully convolutional dense-dilation network and the corresponding training strategy. The network is trained with small image blocks, then works on full-size images, which only needs to forward once for the process. In the first phase, it trains a classification network which classifies a small image block as crack, sealed crack or background. In the second phase, the fully convolutional layer is employed to convert the classification network into a detection network that is insensitive to the input size. At last, via introducing the equivalent dense-dilation design, it transfers both the low-level and middle-level knowledge from the classification network to facilitate the end-to-end network refining and improve the crack localization accuracy. The proposed approach is validated on 600 pavement images (2000×4000-pixel) obtained by industry equipment and it achieves state-of-the-art performance comparing with that of the recently published works in efficiency and accuracy.**

I. Introduction

Automatic pavement cracks detection has been a challenge in intelligent pavement surface inspection systems [1]. Deep learning-based techniques show promising ability for solving multi-label object classification/detection problems; however, the input size is an issue which was rarely discussed in the references [2, 3]. In industry, the pavement images are usually captured with large image size, such as images of 2000×4000-pixel used in this paper, which makes the window-sliding and region-proposal based methods impractical because a lot of candidate regions need to be processed patch-by-patch; in such way, there would exist a lot of redundant convolution operations, which is one of the main reasons for the inefficiency.

In the early works of pavement crack detection, intensity-thresholding and edge-detection based methods were popular because they are fast and easy to implement [4]; however, they are sensitive to noise. More advanced computer vision-based methods have also been tried for solving this problem. Shi et al. [5] applied random structured forest to crack detection, where the integral channel features were used to discriminate the crack patches from non-crack patches based on a crack-token mapping strategy; and it used the distribution difference of the statistical feature histogram and statistical neighborhood histogram to identify false positives. However, the method cannot remove the noise connected to the true crack regions and such cases were quite common when applying their method to real pavement images as indicated in the experiments section. In [6], adaBoosting technique was used to combine a bunch of weak classifiers trained with Gabor features for crack detection; histograms of gradients (Hogs) of the image patch were also used to discriminate crack patches from background patches [7]. However, these methods extract some hand-crafted features which usually calculate some statistics at some local space but cannot well describe the varied structural information at a global view which is important for discriminating cracks from the noisy textures. Different from hand-crafted feature extractors, deep convolutional neural network (DCNN) can automatically learn the formats of the convolutional kernels at different scales through deep, multi-layer architecture[8]. Zhang et al. [9] designed a convolutional network for crack block and sealed-crack block pre-selection which works in a window sliding mode. Cha et al. [10] used DCNN to classify image patch as crack or non-crack patch which also works in window sliding mode to process a full-size image. The problem with those methods is: (1) the window-sliding based strategy is very inefficient because thousands of image patches/windows need to be processed patch-by-patch even for a single pavement image of 2000×4000-pixel; (2) the regular deep convolutional network built with fully connected layer as the classifier can only deal with image blocks of small, fixed size, e.g. 256×256-pixel in [10], and the outputs only give patch labels with low localization accuracy.

In this paper, we propose an efficient dense-dilation network for pavement crack detection that is insensitive to input size and it only needs to forward once to process a full-size image without window sliding. In order to successfully build an end-to-end crack detection network, an image-patch classification network is trained as the base network; then, it is converted to the detection network by replacing the fully connected layer with the equivalent convolutional layers as present in section two. To solve the hard-to-converge problem which we encountered in the end-to-end training mode, the equivalent dense-dilation design is introduced to enable the transfer of both low and middle-level knowledge learned from the classification network for fine-tuning and finally to improve the crack localization accuracy. The whole process is a one-stage strategy which is free of window-sliding and

[1]Kaige Zhang, PhD Candidate, is with Dept. of Computer Science, Utah State University, Logan 84322 USA kg.zhang@aggiemail.usu.edu

[2]Heng-Da Cheng, Professor, is with Dept. of Computer Science, Utah State University, Logan, 84322 USA hd.cheng@aggiemail.usu.edu

[3]Shan Gai is with Dept. of Information Engineering, Nanchang Hangkong University, Nanchang, China gaishan@nchu.edu.cn

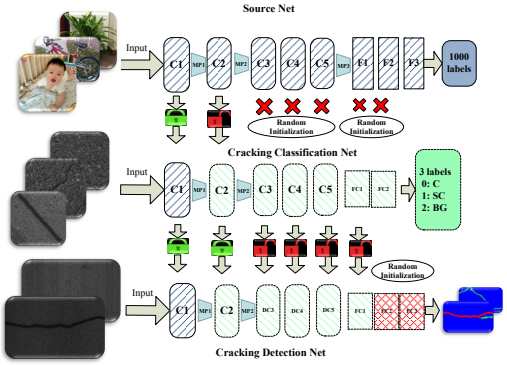

Fig. 1. Overview of the proposed method: network at the top is the source net trained with ImageNet; the network in the middle is the crack block classification net and the network at the bottom is the detection net working on full-size image; the layers filled with green patterns are for fine-tuning, those filled with red patterns are trained from scratch (C=crack; SC=sealed crack; BG=background).

free of region proposal; besides, it is more efficient because of the prevention of redundant convolutional operations. The effectiveness of the proposed approach is demonstrated by the experimental results on 600 pavement images obtained by industry equipment.

II. Proposed Method

A. Overview

In our early attempts, it is found that directly training a fully convolutional network (FCN) [11] for end-to-end per-pixel detection was not satisfactory. There are several reasons, and the most possible ones are: (1) Crack, as a thin-long target, can only occupy a very small area in the full image comparing to the background regions; with the fact that patch-wise training is equivalent to the loss sampling in FCN [11], directly applying FCN to full size cracking images for pixel-level detection makes the training set heavily unbalanced. (2) The precise ground truths of pavement crack images are inherently difficult to obtain since the crack boundaries are vague, and that makes it impossible for per-pixel ground truth marking; therefore, the community usually marks the cracks by 1-pixel curves and sets them as the ground truths. However, they cannot perfectly match the actual cracks at pixel level, that makes the loss computation inaccurate.

Based on these analyses, this work starts with training a classification network as the base net which classifies image block of 227×227-pixel as crack, sealed crack or background; the training set is composed of 60,000 image blocks of the three categories (20,000 instances for each). However, such classification net can only process image blocks with fixed size, 227×227-pixel. To build a detection network working on full-size pavement images, the fully connected layers are replaced with the equivalent convolutional layer that releases the fixed-input-size constraint. While the classification net working on full-size image only produces output with low-resolution and low localization accuracy, the next procedure targets on increasing the localization accuracy with the end-to-end refining. As discussed later, the localization-accuracy decrease mainly comes from the convolution/pooling layers working at the strides larger than one, especially those high-level layers with larger receptive field, it reduces the convolutional strides directly and introduce the equivalent dense-dilation design to remedy the affection of stride changes on forward computing. It shows that this equivalent dense-dilation design can enable the transfer of middle-level knowledge learned from the classification network which is an important factor to succeed the convergence of the end-to-end refining and finally improve the crack localization accuracy. Fig. 1 is an overview of the process.

B. Training a Classification Network

Comparing with the task of crack detection, training the network as an image-block classifier is much easier. It can be viewed as a weakly supervised learner which serves as an important role in this work for transforming the classification net into the final detection net.

Training a deep classification net needs a relatively large dataset. It is prepared following the strategy of block sampling with data augmentation in reference [9]. Then, transfer learning is used to train the classification network. In essence, the DCNN involves lots of filters which conduct convolution operations on the input image for feature extraction; however, each convolution kernel/filter actually operates on the entire image/feature map with the same format during training, which makes the convolution layers have more chance to extract those common features from the image. Especially, for the low-level convolutional layers, they usually learn generic features that have strong transferability [12]. In our case for detecting cracks and sealed cracks, ImageNet pre-trained model, AlexNet[8], is used as the source network. The proposed approach re-implemented the network by: (1) transferring the generic knowledge via copying weights of the first convolutional layer and keeping them unchanged during training; (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 labels of crack, sealed crack or background, and Softmax with cross-entropy loss [8] is used. In this way, it can create a classification net which is easier to train. See Fig. 1.

C. Crack Detection Network

1) Working on full-size image with FCN: To build the detection network working on the full-size image, the fully connected layers at the end of the network are replaced with the equivalent convolutional layers as in Fig. 2. In detail, 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. Then, for the equivalent

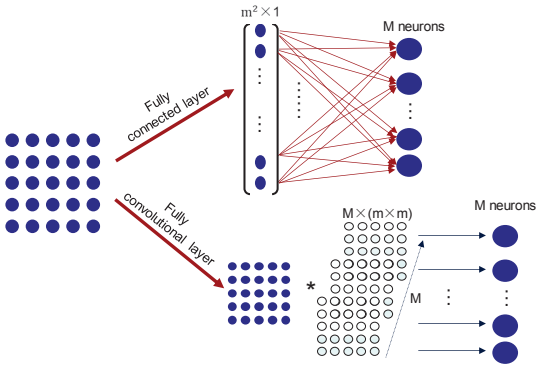

Fig. 2. Fully connected layer is a special case of the convolutional layer.

convolutional layer, it can configure M convolutional kernels, each with $m \times m \times 1$-dimension, and conducts the convolution computing on the entire input respectively; then, the output would be M-dimension feature map but each of the map is 1×1-dimension. As described above, the base network built from AlexNet [8] has input of $6 \times 6 \times 128$ dimensions for the first fully connected layer F1 (m = 6), and the output is 2048 neurons (M=2048); thus, it can configure 2048 convolution kernels of 66128 on the feature maps that will produce 2048×1-dimension output which is a perfect match with the original F1 layer. For F2, it can configure a $1 \times 1 \times 2048$-dimension kernel and set the output as 3-dimension vector that stands for the three categories. In this way, the modified network can work on the full-size image directly without other modification, which serves as the prototype of the detection network.

2) Resolution analysis under larger field of view: Before finalizing the end-to-end detection net for full-size pavement image processing, we perform an analysis of the resolution changes under larger field of view. Assume that the classification net is trained based on images with size n×n-pixel, now we directly apply the convolutional network on images with sizes N×N-pixel, where N>n; we define such situation, especially when $M \gg m$, as under larger field of view. At the first glance, it tends to give us an illusion that the classification 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 the full image tended to produce an output with (2000×4000) / 227 = **8×16**. However, it is not true (the output dimension is **63×125**). Indeed, under larger field of view, it actually realized multiple-spot detections at different locations with all convolutional operations shared at each layer. The number of the spots is decided by the convolution and pooling operations with stride larger than one; i.e., the convolutional layers with stride equal to one will not reduce the resolution (ignoring the border under larger field of view). Besides, other than the equivalent implementation of multi-spot detection with window sliding-based method, such process with full-size image as input does not involve redundant convolutional operations; similarly, redundant convolutions also exist in region proposal based method because of the overlapping among the proposed regions [2, 13]. We argue that, in addition to the time cost of region proposal/window sliding process, this redundant convolutions is one of the key factor affecting the detection efficiency with large input image-size; besides, the DCNNs working with large input image-size is more able to make full use of the parallel-computing capability of GPU. While it only gives the output with low resolution and low localization accuracy, it is named Naive Detection Net. Fig. 3 shows the results of two full-size images produced by the Naive Detection Net.

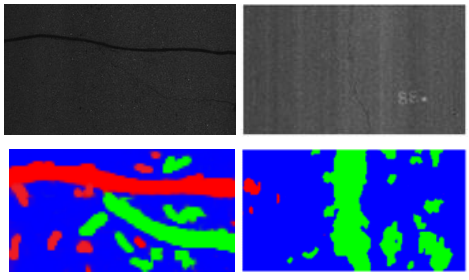

Fig. 3. Low quality detection results with Naive Detection Net.

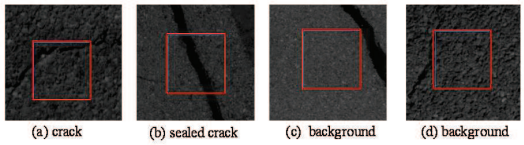

Fig. 4. Localization uncertainty of the classification net.

3) Localization uncertainty: In addition to the resolution loss inherited from the classification net, the localization uncertainty is another factor that results in the low detection accuracy. As shown in Fig.4, in classifying the image blocks, it is found that the network did very well when dealing with the blocks with crack/sealed crack passing through the block center or the blocks without cracks. However, it becomes ambiguous for the blocks with cracks near the border. It is named localization uncertainty of the classification net, and if the uncertainty can be reduced, the detection performance will be improved. As mentioned before, we removed the last max-pooling layer from the original AlexNet that occupies a large receptive field was for the same reason; besides, the end-to-end refining with equivalent dilation design is also targeting on further improving the localization accuracy.

4) Equivalent dense-dilation layers: It is known that knowledge transfer is very useful for training deep neural networks [12]. In this section, we developed the equivalent dense-dilation design to transfer both low and middle-level knowledge from the classification network to facilitate the end-to-end refining of the detection network. The dilated convolution (also named atrous convolution) is originally designed for end-to-end semantic segmentation by aggregating contextual information at multi-scales [14]. In image processing problems, it considers 2-D discrete convolution: Let F:$\mathbb{Z}^2 \rightarrow \mathbb{R}$ be a 2-D discrete function with domain $\Omega_f = [-f, f]$ which can be viewed as the input image; and let K: $\mathbb{Z}^2 \rightarrow \mathbb{R}$ be another 2-D function with domain $\Omega_k = [-k, k]$

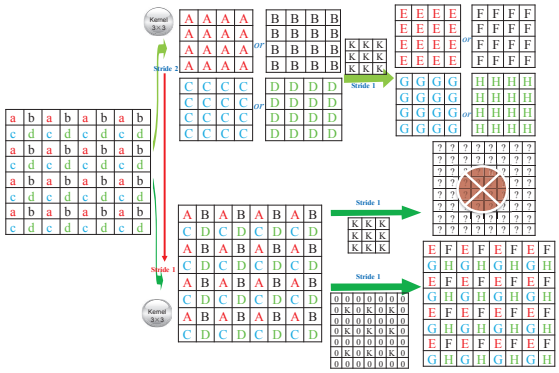

Fig. 5. Equivalent dense-dilated convolution when stride changed.

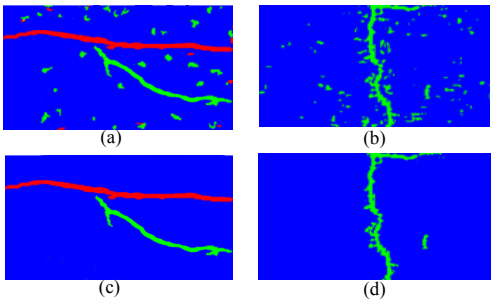

Fig. 6. Detection results of the images in Fig. 3: (a) and (b) are the outputs of the refined network; (c) and (d) are the final results after removing small noisy regions.

that can be viewed as the convolutional kernel, assuming n<m. The discrete convolution operator * can be expressed as:

$$F * K(m,n) = \sum_{i=-r}^{r} \sum_{j=-r}^{r} F(m-i, n-j) K(i,j) \quad (1)$$

For the dilated convolution, the kernel function is changed to:

$$K_{i,j}^{d} = \begin{cases} K_{i/s,j/s}, & if\ s\ divides\ i\ and\ j; \\ 0, & otherwise. \end{cases} \quad (2)$$

In the classification network, a variety of settings are employed to assist the training, such as max pooling [8], drop out [15], increasing convolution/pooling stride [8], etc. Among them, the large strides configured within convolution and pooling layers significantly reduce the output dimension/resolution and discard lots of information that is trivial for classification, e.g. the localization information may be ignored by the max-pooling layer; however, the information can be important for high resolution object detection. In this part, we introduce the equivalent dense-dilation design and implant it into the classification net seamlessly to reduce the resolution loss and also to reduce the localization information loss. It will not damage the forward propagation pipeline of the original classification net, hence, reusing/transfer the parameters/knowledge for end-to-end network refining.

As shown in Fig. 5, 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 are 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 elements for the elements near borders); the output is showed in Fig. 5 at the 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 Fig. 5. Obviously, we cannot continue the forward propagation with original kernels for a desired output in the subsequent layers. However, as shown in Fig. 5, 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 additional information is just the outputs operated at the missed locations under the large-stride; and it can carry useful information that could be transferred at the end-to-end fine-tuning stage to improve the localization accuracy. 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.

Moreover, it can be found that once a stride reduction was applied to a specific layer, all the subsequent layers should take the dilated operation at the related stride size in order to maintain the original forward-computing logic; and the dilation stride size have to accumulate along with each reduction from the previous layers. For example, assuming 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 should 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 3. As an exception, if the convolution kernel is 1×1 with stride 1, such as FC2 in the classification net, the dilation is trivial.

Based on the above discussion, it reduced the stride of the second max pooling layer (MP2) to 1, and set the dilation levels of C3-C5 to 2; and reduced the stride of C5 to 1, set the dilated stride of FC1 to 4 (note that we did not make change for C1, C2 and MP1 because they have relatively small receptive fields which will make little difference). Finally, it will produce the output of 250×500-dimension. This dense-dilation network is an equivalent conversion of the classification which improves the output resolution and enables the transfer of middle level knowledge; and it serves as an important factor making the subsequent network-refining converge under larger field of view.

D. End-to-End Refining with Dilated Ground-Truth

In this part, it conducts an end-to-end refining process under larger field of view to reduce the localization uncertainty

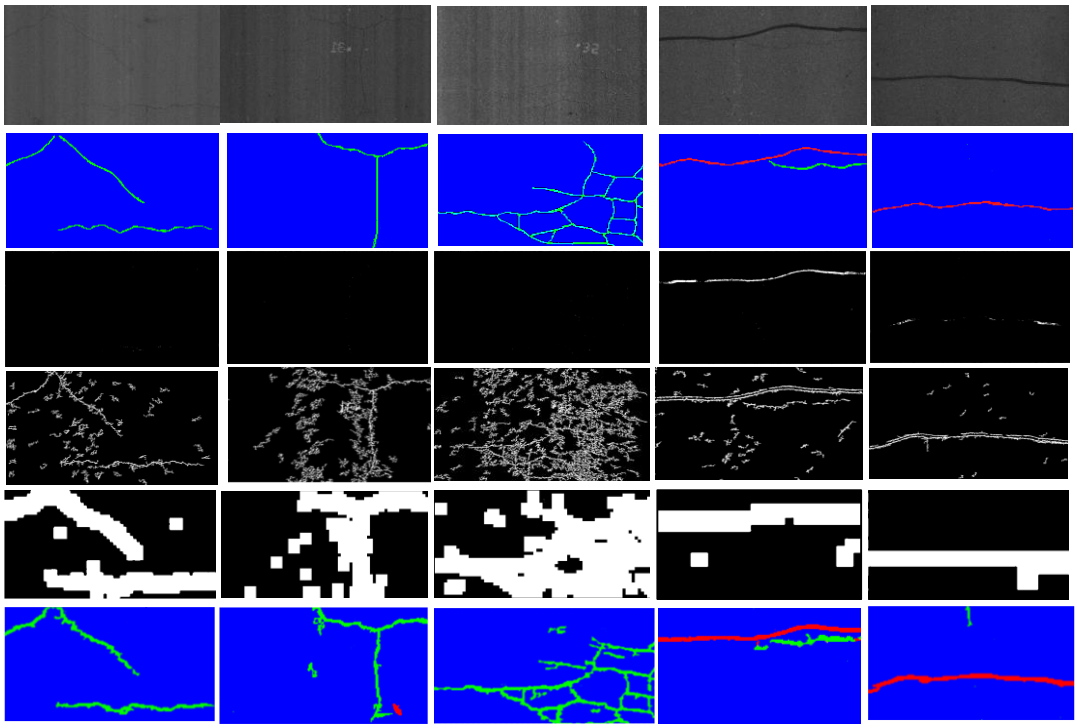

Fig. 7. Comparisons of different crack detection algorithms (from top to down: original images, ground truths, results of CrackIT, results of CrackForest, results of [10] and results of the proposed method.

and improve the detection results.

First, the larger field of view is set by utilizing images of 400×400-pixel as the training data which is an experimentally determined size. As mentioned before, training an end-to-end detection net with 1-pixel-width crack curve as ground truth is problematic for the high risk of mismatch. Other than using the 1-pixel-width ground truth as the label, it uses the k-dilated ground truth images as the label images ("disk" structuring element was used for the dilation). That is, as long as the detected crack pixel falls in the dilated range of the 1-pixel ground truth, it will be treated as true positive.

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 relevant label images. Besides, only crack and sealed crack images are involved in the training set because the background samples are automatically taken into consideration under the FCN mode [11]). Fig. 6 is the detection results of the sample images in Fig. 3. It can be seen that the refining procedure reduces the localization uncertainty level and the false positives are in smaller sizes which could be further removed using simple denoising processes [16], see Fig. 6 (c) and (d).

III. Experiments

The detection net is built with the open-source deep learning framework, Caffe [17]; the server with Nvidia GTX-1080-Ti GPU is used to conduct the experiments with Matlab 2014a.

A. Dataset and Metrics

The data is collected using a line-scan camera. The camera is set on top of a vehicle running at 100km/h which can scan 4-meter-width road surface and produce a 2000×4000-pixel image every 2000 line-scans, and that is 1 pixel representing 1×1 mm^2 road area. There, 600 images with low similarity are utilized for the experiments; among them, 200 with crack, 200 with sealed crack and 200 without cracks. The training-test split is 1:1.

Different from most nature image object detection tasks [18], the traditional IOU (intersection over union) computed using number of the related pixels are not suitable for evaluating the localization accuracy of crack detection algorithms because the crack is thin and the image consists mainly of background pixels [19]. As [19], the buffered Hausdorff distance is used to penalty the detected crack pixels and calculated the overall score to evaluate the localization accuracy. The precision and recall rate based on skeletonized detection results are also reported for evaluation since the precision rate can reflect the false-detection severity and the recall rate can reflect the missed-detection severity.

B. Experimental Results

Three state-of-the-art approaches are tested for comparison: CrackIT [20], CrackForest [5] and approach from [10]. CrackForest and [10] are learning-based methods with [10]

TABLE I
CRACK DETECTION PERFORMANCE

Method	p-rate	r-rate	HD-score	C or SC?
CrackIT	0.901	0.001	9	No
CrackForest	0.303	0.981	51	No
Method in [10]	0.520	0.969	65	No
Proposed	**0.921**	**0.970**	**96**	**Yes**

as a deep learning approach; CrackIT is a more traditional image-processing-based approach.

As shown in Fig. 7 (third row) and Table 1, the CrackIT cannot even detect any cracks in the first three images, because it performed complex denoise processes that removed all the cracking information. It can only handle the images having distinct, wide cracks with smooth background, such as the fourth image with a distinct sealed crack; therefore, achieving very low recall rate (0.001), and low *HD*-score (9). For CrackForest, it can identify most of the cracking locations (*r*-rate=0.981); however, it is not good in noise removal, especially the noise connected to the true cracking regions, which causes very low p-rate (0.303) and low *HD*-score (51). For the region based deep leaning approach [10], 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. Also, those three methods do not explore discriminating cracks from sealed cracks which is an important issue in pavement surface inspection systems. The proposed method can locate the cracks with very good accuracy and can discriminate the cracks from sealed cracks, which is a multi-label [21] detection approach.

With single GPU, [10] takes 9.20 seconds to process a full image of 2000×4000-pixel which is inefficient due to the sliding window working mechanism (with 50-pixel as the sliding stride). Instead, the proposed method conducts the forward process on the full-size image only once, which is more efficient; it takes 1.20 seconds to process a full image with the same GPU. Moreover, with the sliding stride of 50-pixel, [10] actually outputs 40×80 labels only, that is far less than the proposed method with the output resolution of 250×500.

IV. CONCLUSION

In this paper, we propose the equivalent dense-dilation network for pavement crack detection with large input image. To overcome the inefficiency problem with window-sliding detection, we replace the fully connected layer with the equivalent convolution layer which realizes the multi-spot dense detection without redundant convolution operations. To overcome the hard-to-converge problem encountered in the end-to-end training mode, the equivalent dense-dilation design is implanted into the pre-trained classification network to enable the transfer of middle-level knowledge for transfer learning. The effectiveness of the proposed approach is validated on 600 pavement images of 2000×4000 pixels, and it achieves state-of-the-art performance comparing with the most recently published works in accuracy and efficiency. Moreover, the proposed approach also performed a study of the problem: what is the difference between a DCNN-based classification net and detection/segmentation net; it shows that the detection/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 future, we will develop a domain robust network for pavement cracks detection.

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. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proc. of CVPR, 2014.

[3] R. Girshick, Fast R-CNN. In Proc. of ICCV, 2015.

[4] Y. C. Tsai, V. Kaul and R. M. Mersereau. Critical assessment of pavement distress segmentation methods. Journal of Transportation Engineering, 136 (1), 11-19, 2010.

[5] Y. Shi, L. M. Cui, Z. Q. Qi, F. Meng and Z. S. Chen. Automatic road crack detection using random structured forest. IEEE Transactions on Intelligent Transportation Systems, 17 (12), 3434-3445, 2016.

[6] 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, 29 (5), 342-358, 2014.

[7] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu. Road crack detection using deep convolutional neural network. In Proc. of ICIP, Phoenix, 2016.

[8] A. Krizhevsky, I. Sutskever and G. E. Hinton. ImageNet classification with deep convolutional neural network. In Proceedings of International Conference on Neural Information Processing Systems, Harrah's Lake Tahoe, 2012.

[9] K. Zhang, H. D. Cheng and B. Zhang. Unified approach to pavement crack and sealed crack detection using pre-classification based on transfer learning. Journal of Computing in Civil Engineering, 32(2): 04018001, 2018.

[10] Y. J. Cha, W. Choi and O. Bykztrk. Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. Computer-Aided Civil and Infrastructure Engineering, 32, 2017.

[11] J. Long, E. Shelhamer and T. Darrell. Fully convolutional networks for semantic segmentation. In Proc. of CVPR, 2015.

[12] J. Yosinski, J. Clune, Y. Bengio, H. Lipson. How transferable are features in deep neural networks? In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, Canada, 2014.

[13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In Proc. of ICLR, 2014.

[14] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In Proc. of ICLR, 2016.

[15] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929C1958, 2014.

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

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. ArXiv preprint arXiv: 1408. 5093, 2014.

[18] 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.

[19] Y. C. Tsai and A. Chatterjee. Comprehensive, quantitative crack detection algorithm performance evaluation system. Journal of Computing in Civil Engineering, 31(5): 04017047, 2017.

[20] H. Oliveira and P. L. Correia. CrackIT-An image processing toolbox for crack detection and characterization. In Proc. of ICIP, Paris, 2014.

[21] Y. Zhu and A. Elgammal. A Multilayer-Based Framework for Online Background Subtraction with Freely Moving Cameras. In Proc. of ICCV 2017.