DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection

Qin Zou, *Member, IEEE*, Zheng Zhang, Qingquan Li, Xianbiao Qi, Qian Wang, and Song Wang, *Senior Member, IEEE*

***Abstract*— Cracks are typical line structures that are of interest in many computer-vision applications. In practice, many cracks, e.g., pavement cracks, show poor continuity and low contrast, which bring great challenges to image-based crack detection by using low-level features. In this paper, we propose DeepCrack-an end-to-end trainable deep convolutional neural network for automatic crack detection by learning high-level features for crack representation. In this method, multi-scale deep convolutional features learned at hierarchical convolutional stages are fused together to capture the line structures. More detailed representations are made in larger scale feature maps and more holistic representations are made in smaller scale feature maps. We build DeepCrack net on the encoder–decoder architecture of SegNet and pairwisely fuse the convolutional features generated in the encoder network and in the decoder network at the same scale. We train DeepCrack net on one crack dataset and evaluate it on three others. The experimental results demonstrate that DeepCrack achieves *F*-measure over 0.87 on the three challenging datasets in average and outperforms the current state-of-the-art methods.**

***Index Terms*— Line detection, edge detection, contour grouping, crack detection, convolutional neural network.**

I. Introduction

CRACKS are common defects that can be found on surfaces of various types of physical structures, e.g., the road pavement [1], [2], the wall of nuclear power plants [3], the ceiling of tunnels [4], etc. Repairing cracks is an important task for preventing the expansion of harms and keeping the safety of engineering infrastructures. For example, a crack on the highway pavement will easily become a hole in just one rainy night, which will then be hazardous for high-speed vehicles. For a country like China or US, there are over 100,000 Km highway to be tested and maintained periodically. Automatic testing methods are greatly desired to improve the testing efficiency and reduce the cost. Crack is one of the most common defects. Fixing a crack before its deterioration can greatly reduce the cost of maintenance. Up to date, fully automatic crack detection from noise background is still a challenge.

As a crack is visually a linear/curvilinear structure, crack detection can be formulated as line detection, which is a fundamental problem in computer vision [5]–[7]. In visual perception, a crack can be characterized from two perspectives. From a global perspective, it looks like a one-pixel wide edge in the image, as it is thin and often holds jumping intensity to the background. From a local perspective, it is a line object that has a certain width. Accordingly, the crack detection methods can be roughly divided into two categories: edge-detection based ones and image-segmentation based ones. In the ideal case, if a crack has good continuity and high contrast, then traditional edge detection and image segmentation methods could detect it with high accuracy.

However, in practice cracks may constantly suffer from noise in the background, leading to poor continuity and low contrast. For example, in the pavement image shown in Fig. 1(a), impulse noises brought by the grain-like pavement texture break the crack and undermine its continuity, while the shadow reduces the contrast between the crack and the background. In addition, the direction of exposure may also impact the imaging quality of the crack. These complications commonly lead to degraded performance of the traditional low-level feature based crack detection methods.

In recent years, deep convolutional neural network (DCNN) has demonstrated state-of-the-art, human-competitive, and sometimes better-than-human performance in solving many computer vision problems, e.g., image classification [8], object detection [9], image segmentation [10], [11], etc. For line detection, DCNN-based methods have also been proposed for tasks such as edge detection [12], [13], contour detection [14], [15], boundary segmentation [16], [17] and so on. These deep architectures build high-level features from low-level primitives by hierarchically convolving the sensory inputs.

In particular, when using deep learning for edge detection, it has been observed that, the convolutional features become coarser and coarser in the convolving-pooling pipeline, and

Manuscript received March 1, 2018; revised September 15, 2018; accepted October 25, 2018. Date of publication October 31, 2018; date of current version November 21, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61872277, Grant 61301277, and Grant 91546106, in part by the National Key Research and Development Program of China under Grant 2016YFB0502203, and in part by the Hubei Provincial Natural Science Foundation under Grant 2018CFB482. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yonggang Shi. *(Corresponding author: Qingquan Li).*

Q. Zou, Z. Zhang, and Q. Wang are with the School of Computer Science, Wuhan University, Wuhan 430072, China (e-mail: qzou@whu.edu.cn; zhangzheng@whu.edu.cn; qianwang@whu.edu.cn).

Q. Li is with the Shenzhen Key Laboratory of Spatial Smart Sensing and Service, Shenzhen University, Shenzhen 518060, China (e-mail: liqq@szu.edu.cn).

X. Qi is with the Shenzhen Research Institute of Big Data, Shenzhen 518172, China (e-mail: qixianbiao@gmail.com).

S. Wang is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29200 USA (e-mail: songwang@cec.sc.edu).

Digital Object Identifier 10.1109/TIP.2018.2878966

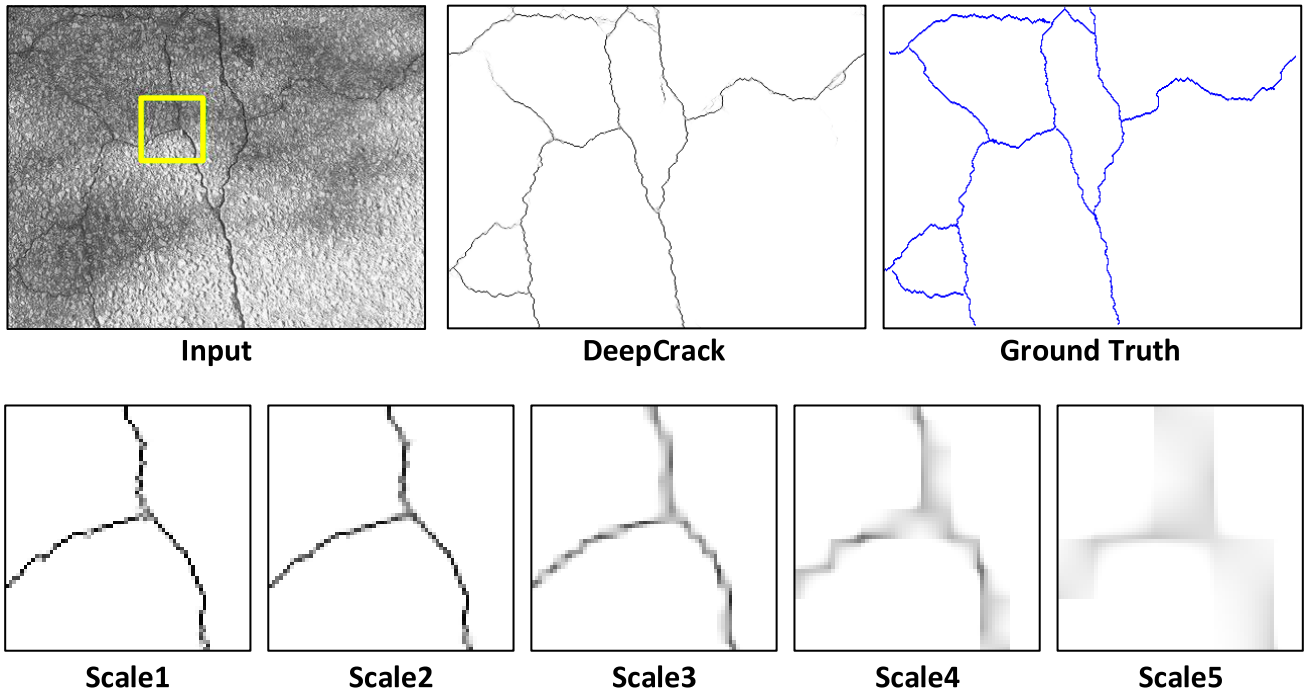


Fig. 1. A real example of crack detection using DeepCrack. The bottom row shows the feature maps generated by convolutional feature fusion at different scales in the DeepCrack net (for the image patch denoted by the rectangle in the input image).

the detailed features in larger-scale layers and the abstracted features in the smaller-scale layers can be fused together to improve the performance of edge detection [13], [18], [19]. When using deep learning for image segmentation, for example the SegNet [20], the convolutional features in the decoder network have been found to be useful to improve the performance of semantic image segmentation, and the indexing of pooling positions can further improve accuracy of boundary localization.

Inspired by these observations, we propose to fuse the convolutional features in both the encoder and decoder networks, and construct a new DeepCrack network for crack detection. We build the DeepCrack on the encoder-decoder architecture proposed in SegNet [20]. In SegNet, a convolution stage in the encoder network is corresponding to a convolution stage in the decoder network, at the same scale. In DeepCrack, we first pairwisely fuse the convolutional features of the encoder network and decoder network at each scale, which produces the single-scale fused feature map, and then combine the fused feature maps at all scales into a multi-scale fusion map for crack detection. An example is shown in Fig. 1, the bottom row shows the fused feature maps at different scales. The sparse feature in smaller scales and the continuous feature in larger scales are fused to get better crack-detection performance.

The contributions of this work lie in three-fold:

- Our main contribution is the design of a new neural network architecture for crack detection. This new network takes full use of the information of the encoder and decoder network, and builds a trainable end-to-end network for crack detection.
- In the proposed network, a convolutional layer of the encoder network and a convolutional layer of the decoder network at one same scale are fused to compute the training loss at the corresponding scale. The fusion of hierarchical convolutional features is found to be very effective for inferring the cracks out from the image background.
- Four datasets are constructed for performance evaluation, where one dataset containing 260 pavement images is used for training the network, and three others are used for test. For the three test datasets, two are pavement image datasets and one is stone surface image dataset. The ground-truth cracks are manually labeled by human expert, and the datasets are shared to the community to promote the research of crack detection. Extensive experiments are conducted and the results demonstrate the effectiveness of the proposed method.

The rest of this paper is organized as follows. Section II briefly reviews the related work. Section III describes the deep neural network architecture for crack detection. Section IV demonstrates the effectiveness of the proposed method by experiments. Finally, Section V concludes the paper.

II. Related Work

A. Line Detection

Line detection is a fundamental problem in computer vision. In a broad sense, line detection includes the edge/contour detection and line object detection. When edges and contours can be built and perceived on the gradient, the detection of them could be treated as line object detection or line grouping in the gradient map. In the past several decades, the research in edge and contour detection has experienced three main stages.

The first stage is featured by computing the first order or second order gradients on the pixel intensity, where a representative in this stage is the Canny edge detector [21].

In the second stage, edge detection and contour grouping are featured by energy minimization methods and middle-level feature learning algorithms. The global Pb [22] is a representative learning method for edge detection, and sketch token [23] and structure edge detector [24] have promoted the learning ability to a peak in this stage. While for contour detection, the ratio contour [25], level set [26], [27] and untangling cycles [28] are part of the representatives, which model the line clutters with graph, and minimize the energy function to infer out the contour. In the third stage, the detection of edges and contours is featured by deep learning, e.g., the deep learning method for edge detection [12], [13], [29], contour detection [14], [15], and boundary segmentation [11], [16], [17]. In [29], line sections are predicted from image patches under a deep learning framework, and a multi-scale version was constructed for edge detection. In [30], DCNN feature abstraction and neighbor search are combined together to handle edge detection and line object extraction. In [12], the edge is detected by a deep convolutional network in an end-to-end manner. The convolutional features in multiple convolutional stages are found to be useful for improving the edge detection results. Similarly, in [13], richer convolutional features generated by a fully convolutional network are fused to further improve the performance.

A number of line object detection methods have also been developed for different application purposes. In [31], a path voting based method was proposed for wire-line detection from vessel X-ray image. The minimal paths were calculated on image patches, and were aggregated to construct a line probability map. In [32], road network extraction from satellite images was studied by regression learning and optimization. In [33], edge detector was built on CNNs, and used to provide information for semantic image segmentation. The convolutional features in different scales were also investigated for some other applications, e.g., video segmentation [19] and symmetry detection [34].

B. Crack Detection

Under a normal illuminance, a crack is generally darker than the background. Therefore, the image thresholding is a straightforward way for crack detection. For example in [35], the threshold value was figured out by examining the difference between the cracks and their neighboring non-crack pixels. In [36], the threshold value was calculated in a heuristic way. However, pavement shadows and uneven illuminations would undermine the robustness of the thresholding-based methods. As the crack is thin and displays as an edge, many methods stemmed from edge detection and wavelet transformation have been developed for crack detection [37]–[40]. However, the edge information would easily be tangled by heavy noise.

As a branch of energy minimization methods, minimal path searching has also been studied for crack detection. In [35] and [41], seed-growing methods built on minimal path searching were proposed for pavement crack detection. In [42], minimal path searching was performed in a path-voting way. In [43], the minimal path searching was used to track cracks in complex background. In these minimal-path-based methods, the main limitation is that the seed points for path tracking should be set in advance.

Machine learning based methods have also been investigated for crack detection. In [2], deep convolutional neural network was used to classify the image patches into crack blocks and non-crack ones. In [4], the detection of bridge cracks was studied by using a modified active contour model and greedy search-based support vector machine. In [3], fully convolutional neural networks were studied to infer cracks of nuclear power plant using multi-view images. Many other methods were also proposed for crack detection, e.g., the saliency detection method [44], the structure analysis methods by using the minimal spanning tree [45] and the random structure forest [46]. Generally, deep learning based methods produce better results than traditional methods. However, there still lacks investigation on end-to-end trainable CNN models for robust crack detection.

III. DeepCrack Network

In this section, we introduce first the architecture of the DeepCrack, then the design of the loss function, and finally the difference of DeepCrack with other deep convolutional networks.

A. Network Architecture

The DeepCrack network is built on the SegNet network [20]. SegNet is a deep convolutional encoder-decoder architecture designed for pixel-wise semantic segmentation, which contains an encoder network and a corresponding decoder network. The encoder network is inspired by the convolutional layers in the VGG16 network [47], which consists of 13 convolutional layers and 5 down-sampling pooling layers. The decoder network also has 13 convolutional layers, and each decoder layer has a corresponding layer in the encoder network. Thus, the encoder network is almost symmetric to the decoder network, where the only difference is that, the first encoder layer, i.e., the first convolution operation, produces a multi-channel feature map, and the corresponding last decoder layer, i.e., the last convolution operation, produces a c-channel feature map, with c the number of classes in the image segmentation task.

After each convolution operation, a batch-normalization step is applied to the feature maps. The max-pooling operation with a stride larger than 1 can reduce the scale of feature maps while not causing translation variance over small spatial shifts, but the sub-sampling will cause a loss of spatial resolution, which may lead to the bias of boundaries. To avoid the absence of detail representation, max-pooling indices are used to capture and record the boundary information in the encoder feature maps when sub-sampling is performed. Then, in the decoder network, the corresponding decoder layer uses the max-pooling indices to perform non-linear up-sampling. This up-sampling step will produce sparse feature maps. However, compared with continuous and dense feature maps, the sparse feature maps obtain more precise location of region boundaries.

Meanwhile, due to the nature of hierarchical learning of deep convolutional neural networks, multi-scale convolutional

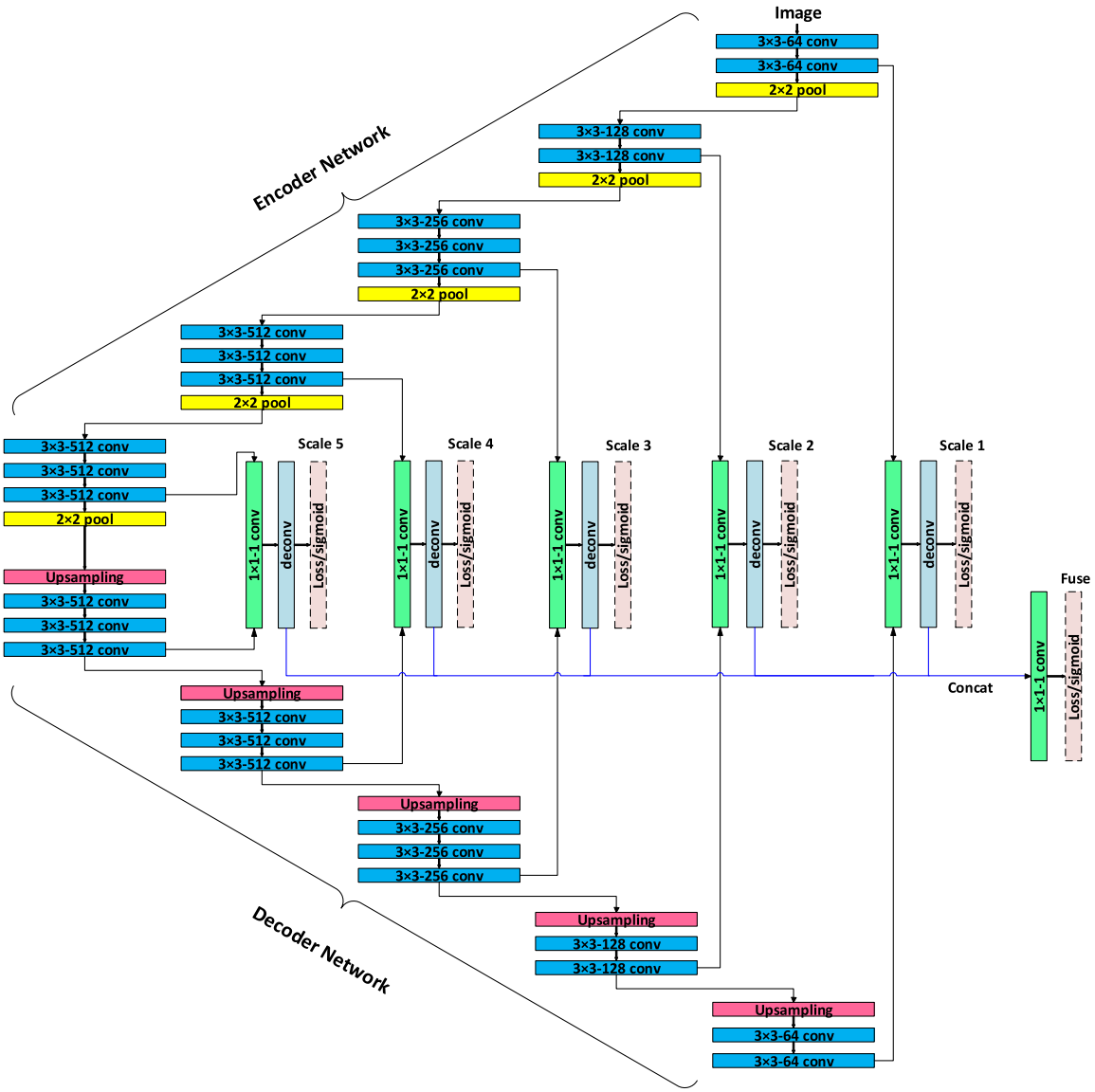


Fig. 2. An illustration of the DeepCrack network. The feature maps of the encoder network and decoder network are pairwisely connected and fused at each convolution stage, which produces fused maps of different scales. At each scale, the pixel-wise prediction loss is calculated by a skip-layer fusion procedure, independently. Meanwhile, the fused maps at all scales are concatenated and fused to product a multi-scale fusion map, which is the output of the DeepCrack network. This output is a crack probability map for crack detection.

features can be learnt in the form of increasingly larger receptive fields in the down-sampled layers. The fusion of the multi-scale convolutional features has been proved to be useful for improving the performance of line detectors [13], [16], [18]. In this work, we consider the scale changes caused by both the pooling operation and upsampling operation, and build the DeepCrack on the SegNet's encoder-decoder architecture. In SegNet, there exist five different scales, which correspond to 5 down-sampling pooling layers. In order to utilize both sparse and continuous feature maps in each scale, the DeepCrack conducts a skip-layer fusion to connect the encoder network and decoder network. As illustrated in Fig. 2, the convolutional layer before the pooling layer at each scale in the encoder network is concatenated to the last convolutional layer at the corresponding scale in the decoder network. The skip-layer fusion handles the concatenated convolutional features with a sequence of operations.

Figure 3 illustrates the skip-layer fusion in details. First, the feature maps from encoder network and decoder network are concatenated, followed by a 1×1 *conv* layer which decreases the multi-channel feature maps to 1 channel. Then, in order to calculate pixel-wise prediction loss in each scale, a *deconv* layer is added to up-sample the feature map and a *crop* layer is used to crop the up-sampling result into the size of the input image. After these operations, we can get the prediction maps of each scale with the same size of the ground-truth crack maps. The prediction maps generated in the five different scales are further concatenated, and

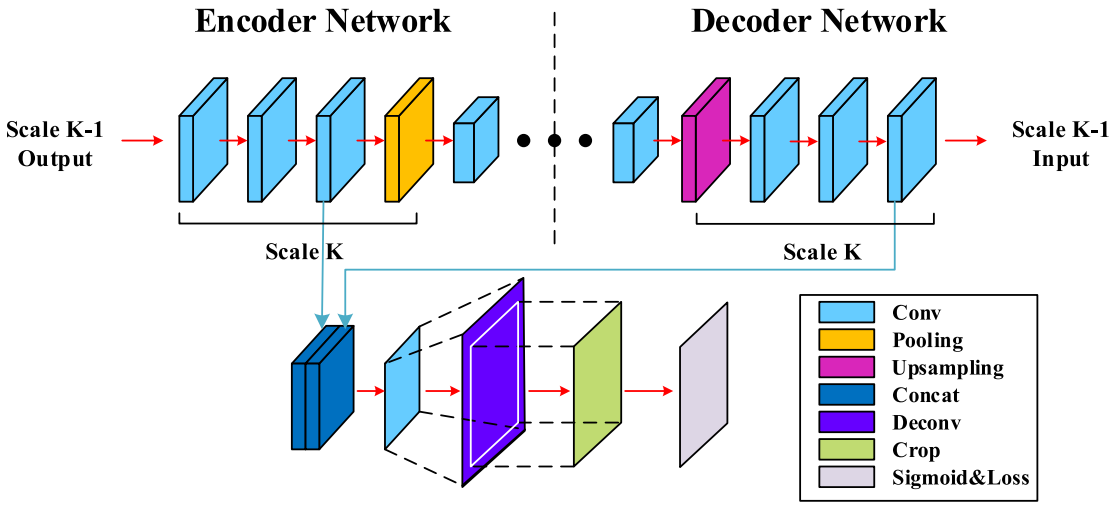


Fig. 3. An illustration of skip-layer fusion at scale K. In each scale, the last *conv* layer in the encoder network and the last *conv* layer in the decoder network are concatenated, followed by a 1×1 *conv* layer with 1-channel output. Then, a *deconv* layer is used to up-sample the feature map. After cropped into the size of the label map, the output is passed to a sigmoid cross-entropy layer to calculate the loss.

a 1×1 *conv* layer is added to fuse the outputs at all scales. As last, we can obtain the prediction maps at each skip-layer fusion and the overall fused layer in the end.

B. Loss Function

Given a training data set containing N images as $S = \{(X^n, Y^n), n = 1, ..., N\}$, where $X^n = \{x_i^{(n)}, i = 1, ..., I\}$ denotes the raw input image, $Y^n = \{y_i^{(n)}, i = 1, ..., I, y_i^{(n)} \epsilon \{0, 1\}\}$ denotes the ground-truth crack label map corresponding to X^n, I denotes the number of pixel in every image, our goal is to train the network to produce prediction maps approaching the ground truth. In the encoder-decoder architecture, let K be the number of convolution stages, then at the stage k, the feature map generated by the skip-layer fusion can be formulated as $F^{(k)} = \{f_i^{(k)}, i = 1, ..., I\}$, where $k = 1, ..., K$. Further, the multi-scale fusion map can be defined as $F^{fuse} = \{f_i^{fuse}, i = 1, ..., I\}$.

Different from semantic segmentation on Pascal VOC, there are only two classes in crack detection, which can be seen as a binary classification problem. We adopt a cross entropy loss to measure the prediction error. Generally, the ground-truth crack pixels stand as a minority class in the crack image, which makes it an imbalance classification or segmentation. Some works [12], [13] deal with this problem by adding larger weights to the minority class. However, in crack detection, we find that larger weights adding to the cracks will result in more false positives. Thus, we define the pixel-wise prediction loss as

$$l(F_i; W) = \begin{cases} log(1 - P(F_i; W)), & \text{if } y_i = 0, \\ log(P(F_i; W)), & \text{otherwise,} \end{cases} \quad (1)$$

where F_i is the output feature map of the network in pixel i, W is the set of standard parameters in the network layers, and $P(F)$ is the standard sigmoid function, which transforms the feature map into a crack probability map. Then, the total loss can be formulated as

$$\mathcal{L}(W) = \sum_{i=1}^{I} (\sum_{k=1}^{K} l(F_i^{(k)}; W) + l(F_i^{fuse}; W)). \quad (2)$$

C. Comparison With Other Architectures

The proposed DeepCrack has two main differences with the original SegNet [20]. First, the original SegNet has no connection between the convolutional features in the encoder network and decoder network, which would cause sparse outputs. In DeepCrack, skip-layer fusion is applied to connect the encoder network and decoder network. Second, the original SegNet is designed for semantic segmentation, which sets up a softmax loss layer to measure the prediction error in each object channel. While in the DeepCrack network, the output is a 1-channel prediction map that indicates the probability of each pixel belonging to the crack by using a cross-entropy loss. DeepCrack is also quite different with U-Net [11]. U-Net performs skip-layer fusion by copying convolution layers in an early stage as a part of a corresponding later stage in the main network, which results in a sole loss. DeepCrack performs skip-layer fusion at each stage independently and assigns it a loss, which leads to multiple losses, and to effective capturing information of thin objects at each scale. Compared with DeepEdge [29], DeepContour [14] and N^4-Fields [30] which perform convolution on image patches, DeepCrack performs convolution on the whole image and generates results in an end-to-end manner.

We also compare the DeepCrack network with two end-to-end deep edge detection architectures, i.e., HED [12] and RCF [13]. Both HED and RCF have their main architectures built on VGG16, which is similar to the encoder network in DeepCrack. Besides the lack of the *pool*5 layer, RCF changes the stride of *pool*4 layer to 1 and uses the atrous algorithm to fill the holes. In the five convolution stages, HED connects the last convolution layers in each scale to produce the fused

prediction map, while RCF connects all convolution layers in each scale at first, and then fuses multi-scale feature maps. Different from HED and RCF which do not have a corresponding decoder network, the proposed DeepCrack pairwisely fuses convolutional features in the encoder network and decoder network at the same scale. Due to the absence of sparse and non-linear up-sampling features in the decoder network, feature maps generated by HED and RCF are often continuous and dense, which would lead to inaccurate localization and error prediction. We will illustrate this point in the experiment.

IV. Experiments and Results

In this section, we first introduce the experimental settings, and then report crack detection results obtained by DeepCrack and the comparison methods. At last, we investigate the performance of DeepCrack at different settings.

A. Experimental Settings

1) Implementations Details: We implement our network using the publicly available Caffe [48] which is well-known in this community. In our network, batch normalization is used after each convolutional layer in both the encoder and decoder network, which is convinced to speed the convergence in training process. The weights of *conv* layer in the entire network are initialized by the 'msra' method and the biases are initialized to 0. The up-sampling operation in decoder network is achieved by using the pooling indices stored in max-pooling layer, and in the skip-layer fusion is conducted by bi-linear interpolation. In training, the initial global learning rate is set to 1e-5 and will be divided by 10 after every 10k iterations. The momentum and weight decay are set to 0.9 and 0.0005, respectively. The stochastic gradient descent method (SGD) is employed to update the network parameters with mini-batch size of 2 in each iteration. We train the network with 100k iterations in total. All experiments in this paper are performed by using a single GeForce GTX TITAN-X GPU.

2) Datasets[1]*:* Four crack datasets are used in this study, in which the pavement crack dataset CrackTree260 is used for training the deep networks, and the other three ones are used for test. The images in the test datasets share the same size of 512×512. The ground-truth cracks are annotated by four persons using a specialized labeling tool.

- *CrackTree260* It contains 260 road pavement images - an expansion of the dataset used in [45]. These pavement images are captured by an area-array camera under visible-light illumination. We use all 260 images for training. Data augmentation has been performed to enlarge the size of the training set. We rotate the images with 9 different angles (from 0-90 degrees at an interval of 10), flip the image in the vertical and horizontal direction at each angle, and crop 5 subimages (with 4 at the corners and 1 in the center) on each flipped image with a size of 512×512. After augmentation, we get a training set of 35,100 images in total.
- *CRKWH100* It contains 100 road pavement images captured by a line-array camera under visible-light illumination. The line-array camera captures the pavement at a ground sampling distance of 1 millimeter.
- *CrackLS315* It contains 315 road pavement images captured under laser illumination. These images are also captured by a line-array camera, at the same ground sampling distance.
- *Stone331* It contains 331 images of stone surface. When cutting the stone, cracks may occur on the cutting surface. These images are captured by an area-array camera under visible-light illumination. We produce a mask for the area of each stone surface in the image. Then the performance evaluation can be constrained in the stone surface.

3) Evaluation Metrics: For each image, $Precision$ and $Recall$ can be calculated by comparing the detected cracks against the human annotated ground truth. Then, the F-measure ($2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$) can be computed as an overall metric for performance evaluation. Specifically, three different F-measure-based metrics are employed in the evaluation: the best F-measure on the data set for a fixed threshold (ODS), the aggregate F-measure on the data set for the best threshold on each image (OIS), and the average precision (AP), which is equivalent to the area under the precision-recall curve [24]. Considering that cracks have a certain width, a detected crack pixel is still taken as a true positive if it is no more than 2 pixels away from human annotated crack curves.

4) Comparison Methods: We compare the performance of DeepCrack with current state-of-the-art methods. In these methods, the CrackTree is a traditional low-level feature based method, and the other ones are deep learning based methods.

- HED [12]. It fuses multi-scale convolutional features by using the last convolutional feature map at each stage in VGG16. We train HED on CrackTree260.
- RCF [13]. It fuses multi-scale convolutional features by using all convolutional feature maps at each stage in VGG16. We train RCF on CrackTree260.
- SegNet [20]. It achieves an end-to-end learning and segmentation by sequentially using an encoder network and a decoder network. We train SegNet on CrackTree260.
- SRN [34]. It is originally designed for end-to-end object symmetry detection, which uses the similar feature fusion strategy in HED. We train SRN on CrackTree260.
- U-Net [11]. It performs skip-layer fusion for end-to-end boundary segmentation and formulates the training target with one single loss. We train U-Net on CrackTree260.
- SE [24]. It learns edges and line structures using the random decision forests. We train SE on CrackTree260 by using a number of 8 decision trees and default parameters released by [24].
- CrackTree [45]. It is a method specifically designed for pavement crack detection. The edge-length threshold for graph construction is 10, and the tree-pruning threshold is 50, for all test images.
- CrackForest [46]. It uses SE architecture to generate the crack map, and post-processes the crack map to obtain the final crack.
- DeepCrack. DeepCrack is trained on CrackTree260.

Note that, the results generated by RCF, HED, SRN and SE are thick crack maps, as shown in Fig. 4, which require to be

[1]https://sites.google.com/site/qinzoucn

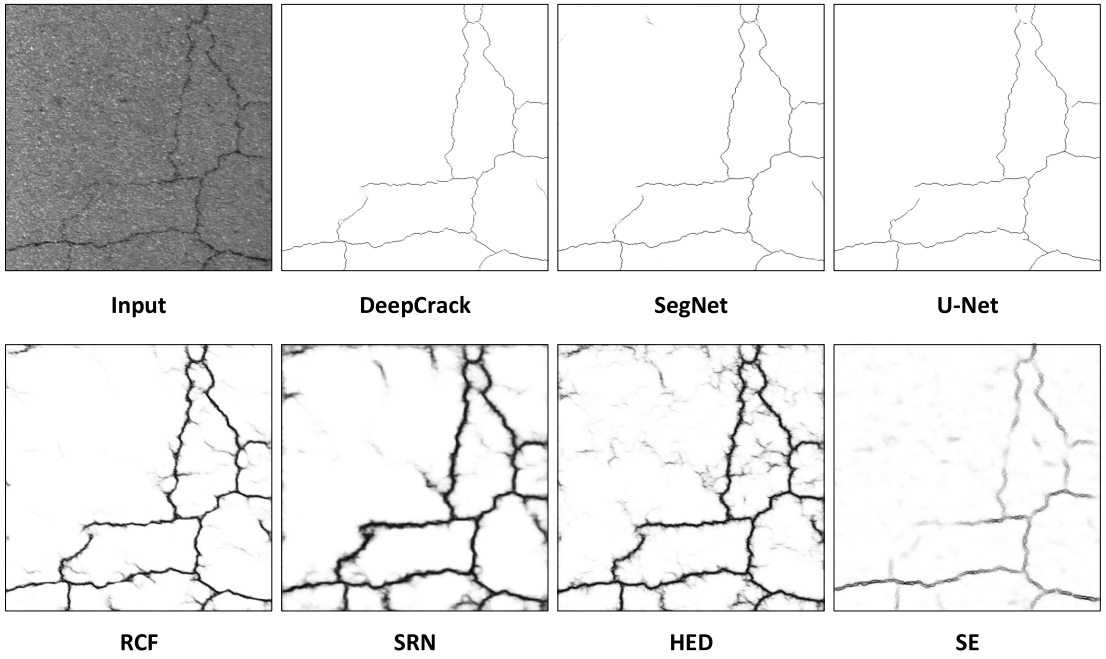


Fig. 4. Crack maps produced by different methods. Note that, DeepCrack, SegNet [20] and U-Net [11] produce thin crack maps, and RCF [13], SRN [34], HED [12] and SE [24] produce thick crack maps.

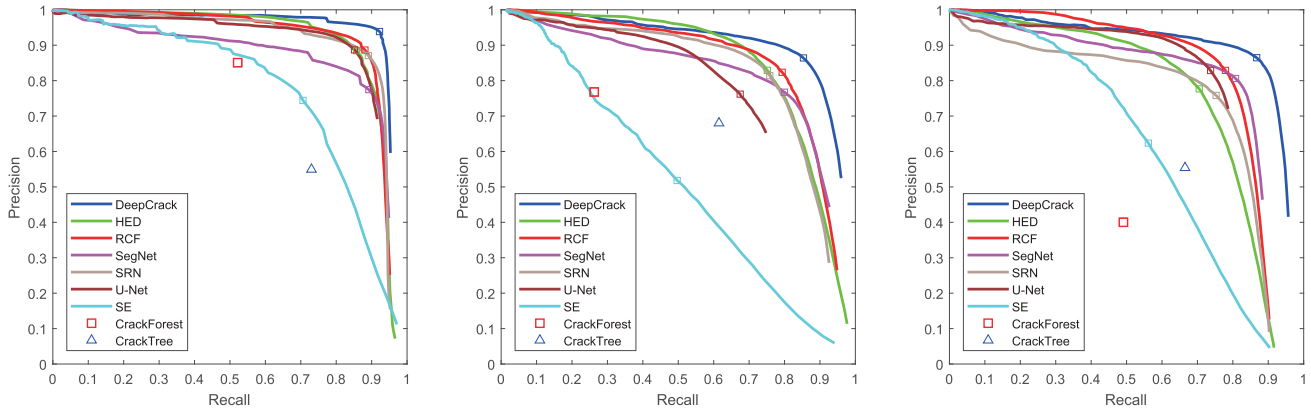


Fig. 5. Precision-Recall curves on the three test datasets. (a) CRKWH100. (b) CrackLS315. (c) Stone331.

post-processed. As in these methods, we employ the standard non-maximum suppression (NMS) [24] to thin the soft crack maps, and take the post-processed results in the comparison. For the results generated by DeepCrack, they are already thin crack maps and can be directly evaluated.

B. Overall Performance

Figure 5 shows the precision-recall curves of nine methods on the three test datasets, where six methods are deep-learning-based. A small rectangle has been plotted at the position corresponding to the best F-measure for each curve. As the CrackTree and CrackForest methods produce hard crack curves, they are marked as point (denoted by a triangle) on the chart using the average precision and recall values.

1) CRKWH100: It can be seen from Fig. 5(a) that, DeepCrack holds a curve most close to the up-right corner in the chart, and achieves the best precision and recall values, as denoted by the best F-measure rectangle. The performances of RCF, SRN and HED are very close. SegNet shows the lowest performance among these deep learning methods, which indicates that the combination of convolutional features in different scales is an effective way to improve the crack-detection performance. Note that, the deep learning based methods achieve significant boost performance over the low-level feature based methods - CrackTree, CrackForest and SE.

Table I shows the quantitative results of the comparison methods. The best result is achieved by DeepCrack, with an ODS F-measure value of 0.9095. Comparing to RCF, SRN, HED and U-Net, there are 4.74%, 4.93%, 6.92% and 6.36% performance improvement on ODS, respectively. Although the performance of SegNet is relatively lower than other deep learning methods, it still achieves ODS value of 0.8184.

TABLE I
QUANTITATIVE EVALUATION OF DIFFERENT METHODS ON THE THREE TEST DATASETS

Methods	CRKWH100			CrackLS315			Stone331			FPS
	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP	
HED [12]	0.8403	0.8851	0.9096	0.7633	0.7976	0.8289	0.7186	0.7627	0.7582	25
RCF [13]	0.8621	0.8909	0.9079	0.7878	0.8158	0.8285	0.7889	0.8287	0.8198	20
SegNet [20]	0.8184	0.8518	0.8496	0.7610	0.7951	0.7798	0.7938	0.8151	0.7874	7
SRN [34]	0.8602	0.8906	0.9037	0.7549	0.7885	0.7950	0.7353	0.7764	0.7409	17
U-Net [11]	0.8459	0.8539	0.9021	0.6718	0.7025	0.7401	0.7570	0.7763	0.8088	6
SE [24]	0.6888	0.7434	0.7646	0.4586	0.5210	0.4946	0.5574	0.6229	0.6050	5
CrackForest [46]	0.6468	0.6468	-	0.3917	0.3917	-	0.4410	0.4410	-	4
CrackTree [45]	0.6269	0.6269	-	0.6429	0.6429	-	0.6041	0.6041	-	0.5
DeepCrack	**0.9095**	**0.9170**	**0.9315**	**0.8449**	**0.8671**	**0.8772**	**0.8559**	**0.8751**	**0.8883**	6

Among the methods, none of the three low-level feature based methods - CrackTree, CrackForest and SE achieves an ODS value over 0.7, while the value of SE is 0.6888 and CrackTree obtains the lowest ODS value 0.6269.

2) CrackLS315: Images in this dataset are captured under laser illumination, which makes them more different with the training images than that in CRKWH100. The precision-recall curves are shown in Fig. 5(b). DeepCrack achieves the best performance on CrackLS315. HED, SRN, RCF and SegNet both show commendable results, while RCF has better performance than HED, SRN and SegNet. It can be observed from Table I that, the ODS of DeepCrack reaches up to 0.8449 that outperforms all compared methods. RCF holds an ODS value of 0.7878, which ranks the second. The ODS of HED, SRN, SegNet and U-Net, are 8.16%, 9.00%, 8.39% and 17.31% lower than the results of DeepCrack, respectively. And compared with CrackTree, DeepCrack obtains an improvement of 20.20% in terms of ODS. The performance of SE suffers a surprising decline on this dataset, which only holds an ODS value of 0.4586.

3) Stone331: It can be seen from Fig. 5(c), DeepCrack outperforms other comparison methods, which has an ODS value of 0.8559. Surprisingly, the second rank is achieved by SegNet with an ODS value of 0.7938, which achieves a weak improvement of 0.52% than RCF. The other deep learning methods, such as HED, SRN, and U-Net, obtain better performance than traditional low-level feature based methods.

We also make visual comparisons on the results. In Fig. 6, crack-detection results of six typical input images are given for the proposed method and the comparison methods. In the first two columns, the input images selected from CRKWH100 contain shadows and obvious noise. DeepCrack can still generate a crack map very close to the ground truth. In the middle two columns, two images are selected from CrackLS315, one contains tiny cracks and the other contains cracks embedded in the road lane, which can hardly be observed without a careful inspection. It can be seen that, all these methods can detect the tiny crack. However, except for DeepCrack, the other methods produce many false detections. For the stone surface images in the last two columns, DeepCrack obtains crack-detection results close to the ground truth, while the comparison methods suffer from many false positives.

Among the deep models, RCF, HED and SRN produce thick crack maps, while DeepCrack, SegNet and U-net produce thin crack maps, as illustrated by Fig. 4. One possible reason is that, the bone networks of DeepCrack, SegNet and U-Net contain an almost symmetrical decoder network corresponding to the encoder network. The decoder network explicitly up-samples the feature maps stage by stage, resulting in an output with the same size of the input, which can help recover thin crack structures, as imposed by the ground truth. The bone networks of RCF, HED and SRN do not contain a decoder network, thus are less capable of producing thin cracks.

In Fig. 6, DeepCrack, SegNet and U-Net are observed to be able to suppress more background artifacts than HED, RCF, SRN, which indicates that the decoder network can also improve the precision of crack prediction. As the DeepCrack fuses low-level and high-level features in the convolution stages of different scales, it can further improve the precision of crack extraction and robustness of background-artifacts suppression. U-net also adopts skip-layers, but it applies one sole loss working on the final prediction, which makes it hard to converge and easily to product incomplete prediction.

In summary, better results can be achieved by the DeepCrack, which fuses the multi-scale convolutional features in both the encoder and decoder networks. More results of the proposed method have been shown in Fig. 11.

C. Constructing DeepCrack With Different Scales

In previous work, the experimental results demonstrate that DeepCrack has notable advantages over other compared methods in crack detection. In this part, we study the effect of fusing the multi-scale convolutional features. Specifically, we want to know how important each scale is in the multi-scale fusion architecture. So at each time, we remove one scale connection of all five scales, and re-train the modified model with the same parameter setting. We repeat this modification five times and get five 'incomplet' multi-scale DeepCrack models. Finally, we test these models on the above three test datasets.

It can be seen from Fig. 7, removing a skip-layer connection of any scale will result in a decreased performance. It indicates that each scale makes a contribution to improve the final results. Meanwhile, the connection in scale one is observed to have significant contribution to the final result. It is because that, the scale one has the same resolution with the input image and holds most of the crack details.

For further exploration, we set different weights to different scales to test how it influences the performance of DeepCrack.

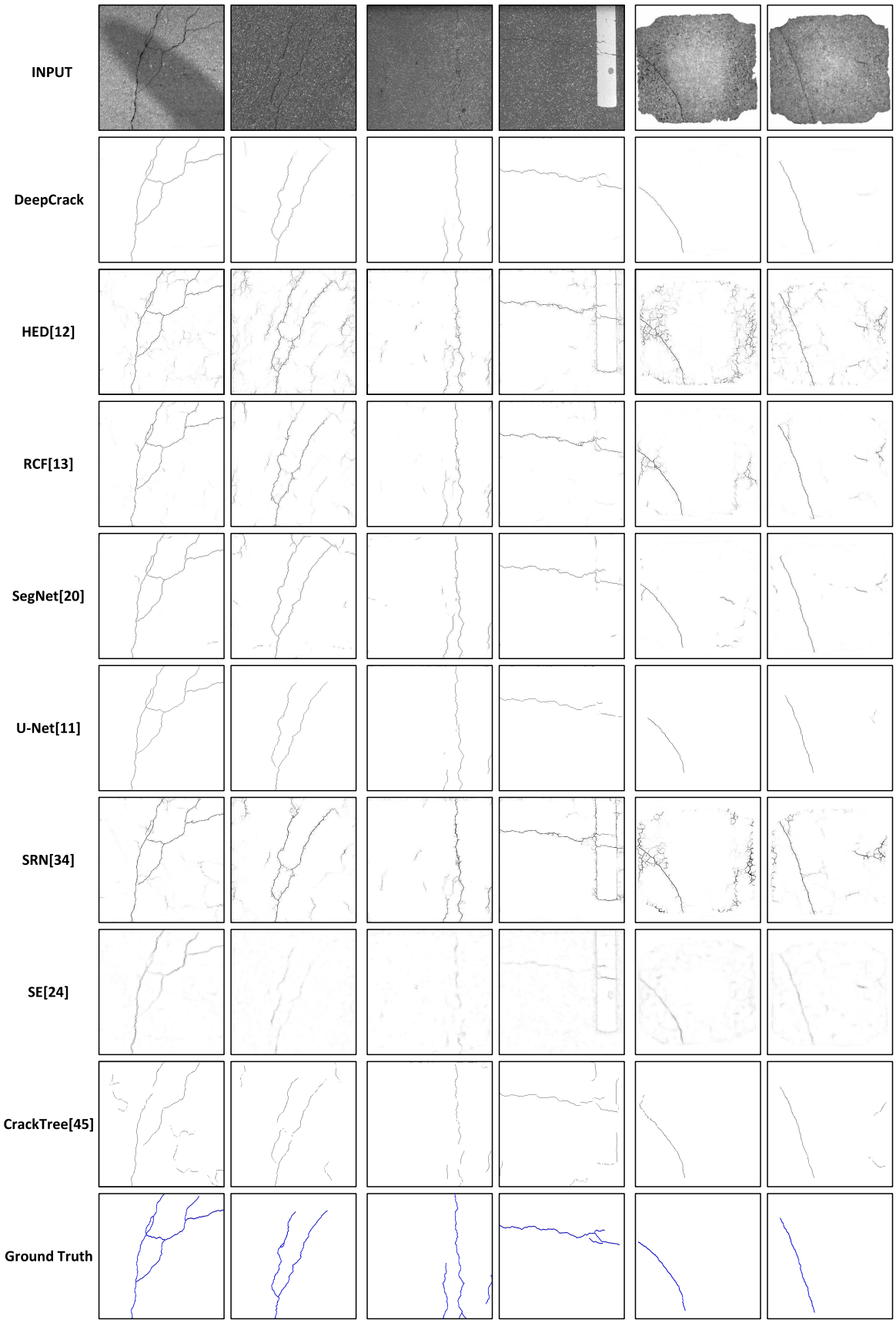


Fig. 6. Comparison of results obtained by different methods on six sample images (from left to right) selected from CRKWH100, CrackLS315 and Stone331, respectively (with two images from each dataset). Note that, the results of HED, SRN, RCF and SE have been post-processed by NMS. The ground-truth cracks have been highlighted in blue.

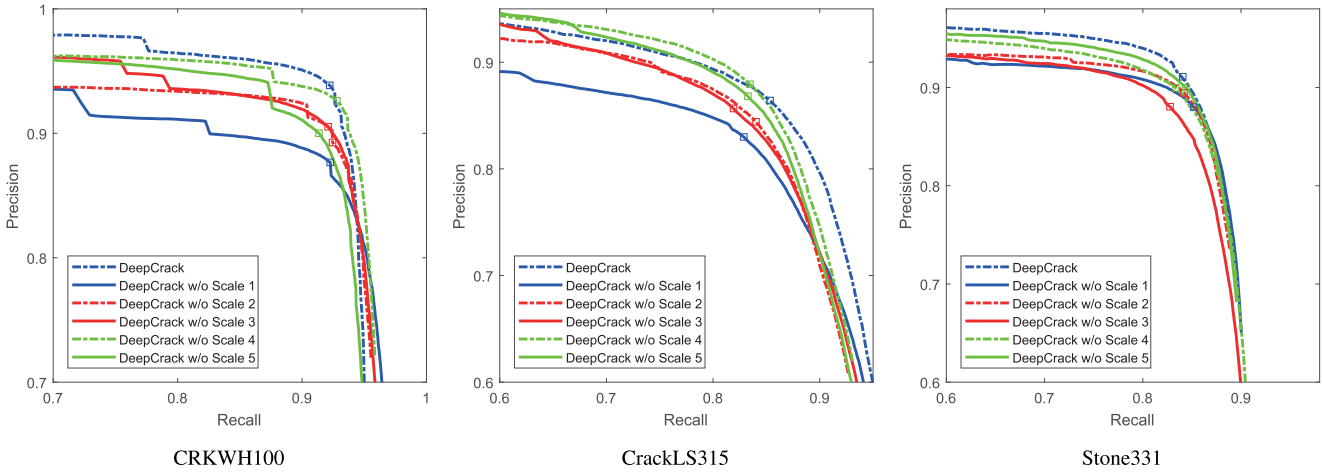


Fig. 7. Comparison of DeepCrack with its modified versions by removing the information from a convolution scale.

TABLE II
PERFORMANCE OF DEEPCRACK BY SETTING DIFFERENT WEIGHTS TO THE LOSS AT DIFFERENT SCALES

$\{\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, \alpha^{(4)}, \alpha^{(5)}\}$	CRKWH100			CrackLS315			Stone331		
	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP
{1, 1/3, 1/9, 1/27, 1/81}	0.8892	0.9015	0.9209	0.8269	0.8510	0.8747	0.7737	0.8010	0.8062
{1, 1/2, 1/4, 1/8, 1/16}	0.9017	0.9110	0.9279	0.8327	0.8554	0.8770	0.8464	0.8667	0.8667
{1, 1, 1, 1, 1}	0.9095	0.9170	0.9315	0.8449	0.8671	0.8772	0.8559	0.8751	0.8883
{1, 2, 4, 8, 16}	0.9012	0.9081	0.9311	0.8328	0.8585	0.8758	0.8494	0.8726	0.8868

For a clear presentation, we rewrite the loss function Eq. (2) as:

$$\mathcal{L}(W) = \sum_{i=1}^{I}(\sum_{k=1}^{K} \alpha^{(k)} \cdot l(F_i^{(k)}; W) + l(F_i^{fuse}; W)), \quad (3)$$

where $\alpha^{(k)}$ denotes the weight placed on the scale k ($1 \leq k \leq 5$). While it is very difficult to find an optimal parameter setting, we choose several representative parameter settings to explore the influence of different weights on the scales. The parameter settings of $\alpha^{(k)}$ and the corresponding results are listed in Table II. Four equal-ratio series are used, with the ratio of 1/3, 1/2, 1, and 2, respectively. In the first two cases, larger weights are set to smaller scales, while in the last case, larger weights are set to larger scales. In Table II we can see that, the ratio of 1/3 produces lower results than the ratio of 1/2, and the ratio of 1/2 produces lower results than the ratio of 1. It indicates that, the information fused at larger scales do make some contributions to the final results. When giving larger weights to smaller scales, a ratio of 2 brings no performance improvement, and on the contrary leads to a subtle lower performance than the standard version, i.e., a ratio of 1. It simply indicates that the scale one holds a dominant influence on predicting the cracks and setting larger weights on other scales will do no good to improve the performance.

D. Training DeepCrack With and Without Pre-Trained Model

In this part, we study with experiments to find whether it is better to train DeepCrack from the pre-trained model or from scratch. As a matter of fact, great difference exists between the crack images and the natural images, especially the nature images of ImageNet and Passcal. The natural images are often colorful and contain visually recognizable object(s), while the crack images are always grayscale and often contain heavy impulse noises. We compare the results of DeepCrack trained from scratch and fine-tuned on pre-trained SegNet model on PASCAL VOC2012. The results have been plotted in Fig. 8. It shows that the model trained from scratch obtains better performance than that trained from pre-trained model, on all the three test datasets. It may be because that the pre-trained model is well fit for nature image segmentation and is impossible or very difficult to be fine-tuned for crack detection.

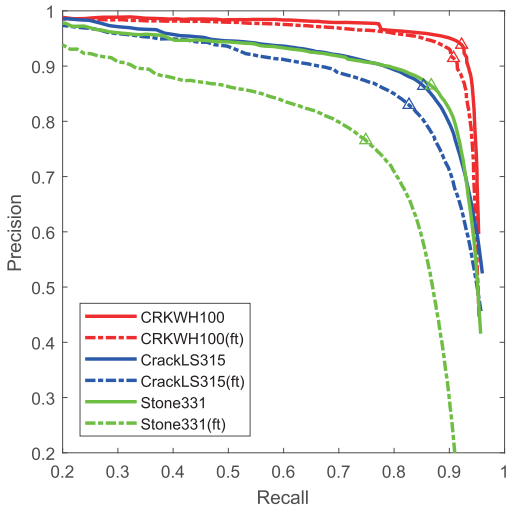


Fig. 8. DeepCrack results with and without pre-trained model. Note that, the '(ft)' denotes the version with fine tune.

TABLE III
QUANTITATIVE EVALUATION OF DEEPCRACK BY USING DIFFERENT SETTINGS

Methods	CRKWH100			CrackLS315			Stone331		
	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP
DeepCrack-reduce(20%)	0.9042	0.9127	0.9355	0.8322	0.8581	0.8742	0.8006	0.8302	0.8368
DeepCrack-noise(20%)	0.9002	0.9090	0.9290	0.8347	0.8577	0.8704	0.8531	0.8718	0.8795
DeepCrack-bias(4 pixels)	0.8674	0.8901	0.8981	0.7622	0.8055	0.7961	0.7973	0.8299	0.8290
DeepCrack-bias(6 pixels)	0.6938	0.7246	0.7942	0.5872	0.6353	0.6456	0.5712	0.6391	0.5947
DeepCrack-upsample(bilinear)	0.8824	0.8959	0.9190	0.7826	0.8174	0.8530	0.7156	0.7683	0.7771
DeepCrack	0.9095	0.9170	0.9315	0.8449	0.8671	0.8772	0.8559	0.8751	0.8883

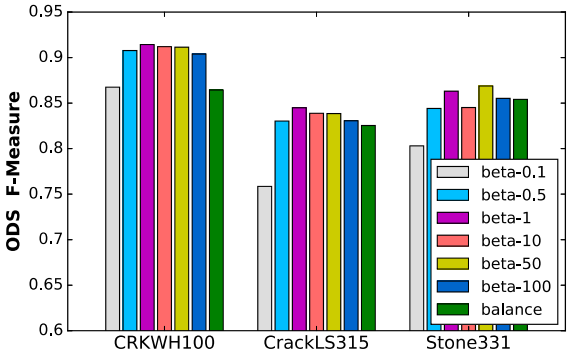


Fig. 9. Different loss weights on the crack and the background.

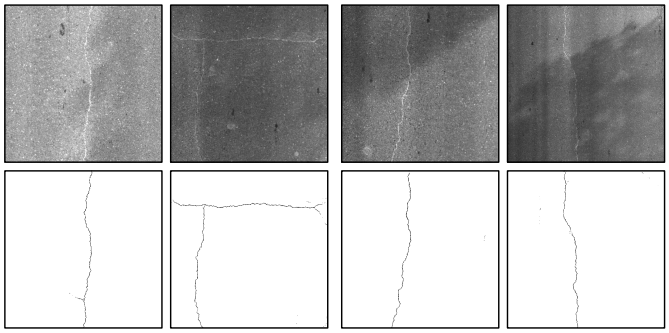


Fig. 10. Detection of bright cracks using DeepCrack.

E. Influence of Uncorrect Labels and Upsampling Strategies

We also conduct several experiments to explore the sensitivity of DeepCrack to noisy ground-truth crack labeling. First, for each image, we randomly reduce 20% of the ground-truth crack pixels, and add 20% of noisy ground-truth crack pixels, and name the retrained models as 'DeepCrack-reduce (20%)' and 'DeepCrack-noise (20%)', respectively. Second, for each image, we random shift the crack labeling left, right, up and down, with 4 and 6 pixels, and name the retrain models as 'DeepCrack-bias (4 pixels)' and 'DeepCrack-bias (6 pixels)', respectively. The test results are presented in the Table III.

From Table III we can see, on CRKWH100 and CrackLS315, reducing 20% of the ground-truth pixels or adding 20% noisy crack labels has very little influence on DeepCrack's performance. On Stone331, adding noisy crack labels will bring little affect, but reducing ground-truth crack labels leads to a declined performance. The results show that DeepCrack is generally not sensitive to noisy crack labels, and is less sensitive on CRKWH100 and CrackLS315 than on Stone331. The reason may be that, the DeepCrack is trained on pavement images, therefore it is more robust in handling pavement images than in handling stone images.

From Table III we can also see, shifting the crack labels with 4 pixels leads to largely decreased performance on all three datasets, and shifting with 6 pixels leads to even worse results. It indicates that the proposed method is sensitive to the spatial bias of ground truth.

To explore the influence of max-pooling indices used in the upsampling operation, we replace the max-pooling indices with bilinear interpolation for upsampling the feature maps. We retrain the model and predict cracks on the three test datasets. As shown in Table III, the upsampling with bilinear interpolation results in a declined performance as compared to the case with max-pooling indices (in the last row of Table III). It indicates that the max-pooling indices are helpful to locate the crack pixels in the upsampling procedure.

F. Different Weights on the Crack and Non-Crack Background

In section III-B, we formulate the loss function by adding the same weight to the crack label and the background, although the distribution of them are imbalanced. We will show the advantage of this setting with experiments. Specifically, we re-define the weighted loss function as Eq. (4),

$$l(F_i; W) = \begin{cases} \dfrac{2\alpha}{\alpha+\beta} \cdot log(1 - P(F_i; W)), & \text{if } y_i = 0, \\ \dfrac{2\beta}{\alpha+\beta} \cdot log(P(F_i; W)), & \text{otherwise,} \end{cases} \tag{4}$$

where α and β are different weights adding to the background and the cracks, respectively. We set the label of pixel belonging to background as $y_i = 0$ and crack as $y_i = 1$. For a convenient comparison, we set the weight of background α to be 1 and set different values of β with {1, 10, 50, 100}. Notice that, when $\beta = 1$, the weighted loss function is equivalent to the loss function defined by Eq. (1), which is called the 'standard weight'. We also make comparison with the balance weight setting used in [12], where α is the number of ground-truth crack pixels and β is the number of background pixels. We call it the 'balance weight'.

It can be seen from Fig. 9, DeepCrack equipped with small weight (β <1) will have decreased performance, which indicates that it is not good to allocate more weight to non-crack background. When larger weights are set to the crack, lower ODSs can be observed on CRKWH100 and

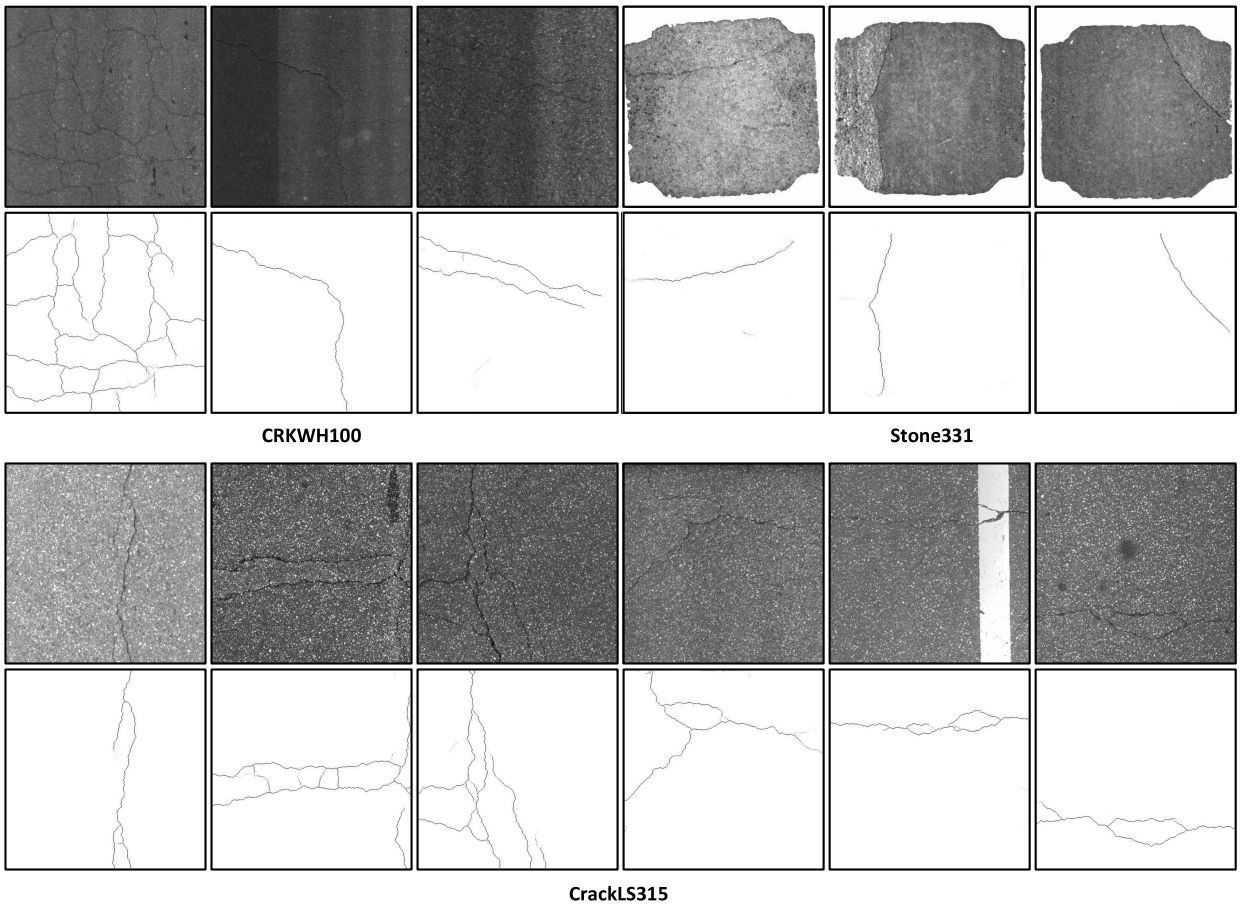


Fig. 11. More results of DeepCrack on the three test datasets. Full resolution results can be accessed at our website.

CrackLS315. Compared with the balance weight, the standard weight generally obtains higher ODS. The reason is that, when a larger weight is given to the crack, false negative prediction will receive heavier punishment. As a result, more pixels will be predicted as crack, while not bringing much impact on the whole loss. Thus, when placing larger weights to the crack, the overall performance will not be improved but get undermined.

On the Stone331, irregular variations of ODS can be observed, as compared with that on CRKWH100 and CrackLS315. It may be because the DeepCrack model is trained on pavement images, the rule got on pavement images is not strictly consistent with that on stone images. Such results indicate that adding different weights to crack and non-crack background will not guarantee a stable improvement on DeepCrack's performance.

G. Detection of Bright Cracks

In the experiments we find that, the DeepCrack model trained on CrackTree260 cannot detect bright cracks. The reason we guess is that there are very few bright cracks in the training dataset. To justify this point, we inverse the brightness of the training images, such that cracks in them will have higher intensity than the background and display as bright cracks. We retrain DeepCrack with the new training dataset. We select four original images from CRKWH100 dataset that contain bright cracks, and perform crack detection using the newly trained model. The results are shown in Fig. 10. It can be seen that, DeepCrack can well handle the bright cracks.

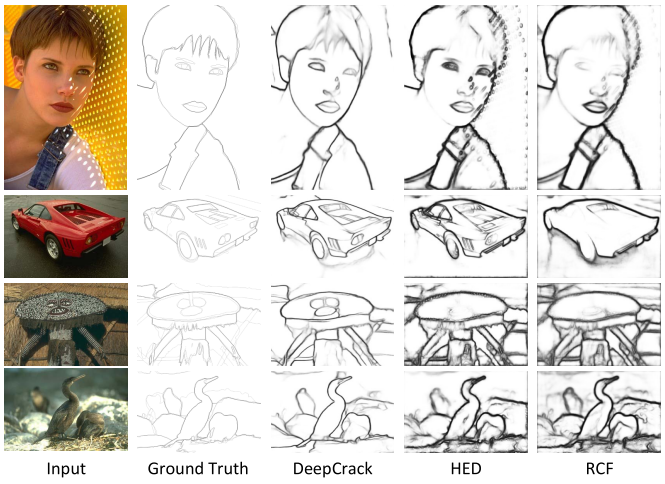


Fig. 12. Sample edge-detection results on BSDS500.

H. Running Efficiency

The proposed DeepCrack, as well as HED, SRN, RCF and SegNet, does not have fully connected layers, which

leads to largely reduced weight parameters. In the test, these networks do not have to suffer the heavy computation load of gradient calculation as in the training. Thus, DeepCrack and the four others can efficiently predict crack maps. It can be seen from Table I (last column), DeepCrack handles images of 512×512 at a speed of 6 FPS, exactly 0.153 second per image. SegNet is a little faster than DeepCrack, which needs 0.141 second per images. With less network layers, HED, RCF and SRN can achieve even faster speeds at about 25 FPS, 20 FPS and 17 FPS. For the traditional methods, SE and CrackForest can process about 5 images and 4 images in one second, respectively and CrackTree needs 2 second to process one image in average. Note that, the running time for HED, RCF, SRN and DeepCrack is based on a GeForce GTX TITAN-X GPU, and the running time for SE, CrackTree and CrackForest is based on a 2.3GHz E5-2630 CPU.

V. Conclusion

In this work, a novel end-to-end trainable convolutional network - DeepCrack - was proposed for crack detection. In DeepCrack, convolutional features at each scale were pair-wisely fused, and the fused feature maps at all scales were further fused into a multi-scale feature-fusion map for crack detection. For performance evaluation, four crack datasets were constructed. Under the same evaluation protocol, one dataset was used for training, and the other three datasets were used for test. Experimental results showed that, the proposed DeepCrack achieved over 0.87 ODS F-measure value on the test datasets in average, and outperformed the competing methods that do not have a decoder network. It indicates that the convolutional features in the encoder and decoder networks are both useful for crack detection. Experimental results also showed that the DeepCrack was not sensitive to noisy crack labeling and could well handle bright cracks.

Appendix
DeepCrack's Performance on Other Tasks

We also examine the capability of DeepCrack on two other line-detection tasks. One is for edge detection, and the other is for vessel detection.

A. Edge Detection

On BSDS500, we augmented the 300 training images to train DeepCrack, and used the other 200 images for test. In Fig. 13, we can see DeepCrack got an ODS of 0.778, which is slightly higher than DeepEdge and DeepContour, but lower than RCF (with NMS) and HED (with NMS). However, DeepCrack obtains better results than RCF and HED on some images, for example the ones shown in Fig. 12. From Fig. 12 we can see, DeepCrack produces clean edge maps while the HED and RCF produce thick ones. And the DeepCrack is found to be talent in detecting thin edges, and would omit fine structures, which leads to relatively lower recall in edge detection. However, this characteristic makes DeepCrack more suitable for crack detection from noise and grain-like texture background.

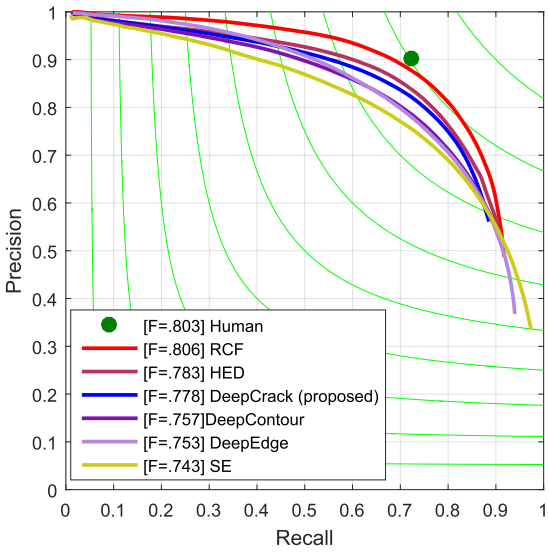


Fig. 13. Edge-detection performance on BSDS500.

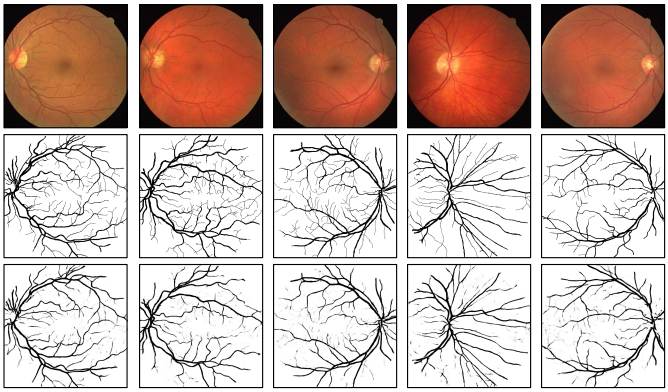


Fig. 14. Results on DRIVE dataset. Row 1: retinal vessel images. Row 2: ground truth labeled by human expert. Row 3: results produced by DeepCrack.

B. Vessel Detection

Vessel detection/segmentation is an important task in medical image processing. We run the proposed DeepCrack on the DRIVE dataset [49] for retinal vessel detection. The DRIVE contains 20 images for training and 20 images for test. Since the number of training samples is too small, we random select 15 images from the test set and add them to the training set. Then, the remaining 5 images are used for test, as shown in the top row of Fig. 14. We conduct data augmentation to the 35 training images, where one image is augmented into 54 images, and a number of 1,890 images are used to train the DeepCrack. The results are displayed in Fig. 14. It is surprising that the DeepCrack model trained on such a small-scale dataset presents very good performance on detecting the main blood vessels structures. Some small vessel branches are found to be missed. We think this would be solved by giving enough training data to DeepCrack.

Acknowledgement

The authors would like to thank Yuanhao Yue and Qin Sun from Wuhan University for their help in labeling the crack ground-truth and plotting some of the figures.

REFERENCES

[1] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth, "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure," *Adv. Eng. Inform.*, vol. 29, no. 2, pp. 196–210, 2015.

[2] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2016, pp. 3708–3712.

[3] S. J. Schmugge, L. Rice, J. Lindberg, R. Grizziy, C. Joffey, and M. C. Shin, "Crack segmentation by leveraging multiple frames of varying illumination," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 1045–1053.

[4] Z. Qu, L. Bai, S.-Q. An, F.-R. Ju, and L. Liu, "Lining seam elimination algorithm and surface crack detection in concrete tunnel lining," *J. Electron. Imag.*, vol. 25, no. 6, p. 063004, 2016.

[5] J. Geusebroek, A. W. M. Smeulders, and H. Geerts, "A minimum cost approach for segmenting networks of lines," *Int. J. Comput. Vis.*, vol. 43, no. 2, pp. 99–111, 2001.

[6] A. Sironi E. Türetken, V. Lepetit, and P. Fua, "Multiscale centerline detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1327–1341, Jul. 2016.

[7] Z. Zhang, F. Xing, X. Shi, and L. Yang, "Semicontour: A semi-supervised learning approach for contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 251–259.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[9] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.

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

[12] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1395–1403.

[13] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5872–5881.

[14] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2015, pp. 3982–3991.

[15] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 193–202.

[16] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, "Convolutional oriented boundaries," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 580–596.

[17] A. Khoreva, R. Benenson, M. Omran, M. Hein, and B. Schiele, "Weakly supervised object boundaries," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 183–192.

[18] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Convolutional channel features," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 82–90.

[19] A. Khoreva, R. Benenson, F. Galasso, M. Hein, and B. Schiele, "Improved image boundaries for better video segmentation," in *Proc. Eur. Conf. Comput. Vis. Workshops*, Nov. 2016, pp. 773–788.

[20] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[21] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[22] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[23] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3158–3165.

[24] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015.

[25] S. Wang, T. Kubota, J. M. Siskind, and J. Wang, "Salient closed boundary extraction with ratio contour," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 546–561, Apr. 2005.

[26] P. Martin, P. Refregier, F. Goudail, and F. Guerault, "Influence of the noise model on level set active contour segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 799–803, Jun. 2004.

[27] C. Li, C. Xu, C. Gui, and M. D. Fox, "Level set evolution without re-initialization: A new variational formulation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2005, pp. 430–436.

[28] Q. Zhu, G. Song, and J. Shi, "Untangling cycles for contour grouping," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[29] G. Bertasius, J. Shi, and L. Torresani, "DeepEdge: A multi-scale bifurcated deep network for top-down contour detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4380–4389.

[30] Y. Ganin and V. S. Lempitsky, "N^4-fields: Neural network nearest neighbor fields for image transforms," in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 536–551.

[31] V. Bismuth, R. Vaillant, H. Talbot, and L. Najman, "Curvilinear structure enhancement with the polygonal path image—Application to guide-wire segmentation in X-ray fluoroscopy," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent. (MICCAI)*, 2012, pp. 9–16.

[32] A. Sironi, V. Lepetit, and P. Fua, "Multiscale centerline detection by learning a scale-space distance transform," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2697–2704.

[33] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4545–4554.

[34] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye, "SRN: Side-output residual network for object symmetry detection in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 302–310.

[35] Q. Li, Q. Zou, D. Zhang, and Q. Mao, "FoSA: F^* seed-growing approach for crack-line detection from pavement images," *Image Vis. Comput.*, vol. 29, no. 12, pp. 861–872, 2011.

[36] M. Kamaliardakani, L. Sun, and M. K. Ardakani, "Sealed-crack detection algorithm using heuristic thresholding approach," *J. Comput. Civil Eng.*, vol. 30, no. 1, p. 04014110, 2014.

[37] P. Subirats, J. Dumoulin, V. Legeay, and D. Barba, "Automation of pavement surface crack detection using the continuous wavelet transform," in *Proc. Int. Conf. Image Process.*, Oct. 2006, pp. 3037–3040.

[38] G. Zhao, T. Wang, and J. Ye, "Anisotropic clustering on surfaces for crack extraction," *Mach. Vis. Appl.*, vol. 26, no. 5, pp. 675–688, 2015.

[39] M. Salman, S. Mathavan, K. Kamal, and M. Rahman, "Pavement crack detection using the Gabor filter," in *Proc. IEEE Conf. Intell. Transp. Syst.*, Oct. 2013, pp. 2039–2044.

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

[41] 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.

[42] Q. Zou, Q. Li, F. Zhang, Z. Xiong, and Q. Wang, "Path voting based pavement crack detection from laser range images," in *Proc. Int. Conf. Digit. Signal Process.*, Oct. 2016, pp. 432–436.

[43] V. Kaul, A. Yezzi, and Y. C. Tsai, "Detecting curves with unknown endpoints and arbitrary topology using minimal paths," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 10, pp. 1952–1965, Oct. 2012.

[44] W. Xu, Z. Tang, J. Zhou, and J. Ding, "Pavement crack detection based on saliency and statistical features," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2013, pp. 4093–4097.

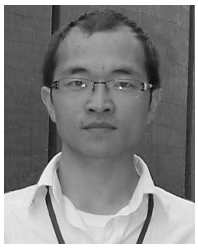
[45] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "CrackTree: Automatic crack detection from pavement images," *Pattern Recognit. Lett.*, vol. 33, no. 3, pp. 227–238, 2012.

[46] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3434–3445, Dec. 2016.

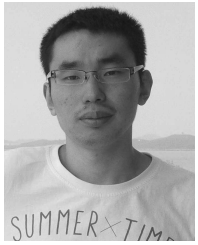
[47] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[48] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678.

[49] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. Van Ginneken, "Ridge-based vessel segmentation in color images of the retina," *IEEE Trans. Med. Imag.*, vol. 23, no. 4, pp. 501–509, Apr. 2004.

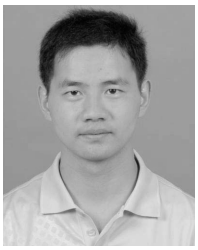


Qin Zou (M'13) received the B.E. degree in information engineering and the Ph.D. degree in photogrammetry and remote sensing (computer vision) from Wuhan University, China, in 2004 and 2012, respectively. From 2010 to 2011, he was a Visiting Ph.D. Student with the Computer Vision Lab, University of South Carolina, USA. He is currently an Associate Professor with the School of Computer Science, Wuhan University. His research activities involve computer vision, pattern recognition, and machine learning. He is a member of the ACM. He was a co-recipient of the National Technology Invention Award of China 2015.



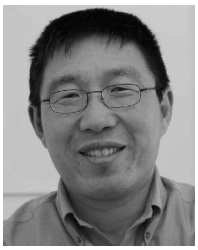
Zheng Zhang received the B.S. degree in computer science from Wuhan University, China, in 2015, where he is currently pursuing the master's degree with the School of Computer Science. He received the first prize from the China Undergraduate Contest in Internet of Things in 2015. His research interest includes deep learning and its applications in image classification and retrieval.

Qingquan Li received the Ph.D. degree in geographic information science and photogrammetry from the Wuhan Technical University of Surveying and Mapping, China. From 1988 to 1996, he was an Assistant Professor with Wuhan University, where he became an Associate Professor. Since 1998, he has been a Professor with Wuhan University. He is currently the President and a Professor with Shenzhen University, China. He is a Professor with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University. He is also the Director of the Shenzhen Key Laboratory of Spatial Smart Sensing and Service, Shenzhen University. He is an Academician of the International Academy of Sciences for Europe and Asia. His research areas include precision engineering survey, pattern recognition, and intelligent transportation systems.



Xianbiao Qi received the B.E. degree in information engineering and the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in 2008 and 2015, respectively. He was an Intern with the Web Search and Mining Group, Microsoft Research Asia, from 2011 to 2012. He was also a Researcher with the University of Oulu, Finland, from 2014 to 2016. He held a post-doctoral position with the Department of Computing, The Hong Kong Polytechnic University, from 2016 to 2018. He is currently a Research Scientist with the Shenzhen Research Institute of Big Data. His current research interests include face analysis, object detection, and scene text detection.

Qian Wang received the Ph.D. degree from the Illinois Institute of Technology, USA. He is currently a Professor with the School of Computer Science, Wuhan University. His research interests include search and computation outsourcing security, wireless systems security, big data security and privacy, and applied cryptography. He is an Expert of the national 1000 Young Talents Program of China. He received the National Science Fund for Excellent Young Scholars of China. He was a recipient of the 2016 IEEE Asia-Pacific Outstanding Young Researcher Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY.



Song Wang (M'02–SM'13) received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign (UIUC) in 2002. From 1998 to 2002, he was a Research Assistant with the Image Formation and Processing Group, Beckman Institute, UIUC. In 2002, he joined the Department of Computer Science and Engineering, University of South Carolina, where he is currently a Professor. His research interests include computer vision, medical image processing, and machine learning. He is a Senior Member of the IEEE Computer Society. He is currently serving as the Publicity/Web portal Chair for the Technical Committee of Pattern Analysis and Machine Intelligence, IEEE Computer Society. He is currently serving as an Associate Editor for PATTERN RECOGNITION LETTERS.