Neurocomputing 338 (2019) 139–153

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

DeepCrack: A deep hierarchical feature learning architecture for crack segmentation

Yahui Liu, Jian Yao*, Xiaohu Lu, Renping Xie, Li Li

Computer Vision and Remote Sensing (CVRS) Lab, School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, PR China

ARTICLE INFO

Article history:
Received 14 December 2017
Revised 10 January 2019
Accepted 15 January 2019
Available online 22 January 2019

Communicated by Dr. Yu Jiang

Keywords:
Convolutional neural network
Crack detection
Semantic segmentation
Hierarchical convolutional features
Guided filtering
Crack detection dataset

ABSTRACT

Automatic crack detection from images of various scenes is a useful and challenging task in practice. In this paper, we propose a deep hierarchical convolutional neural network (CNN), called as DeepCrack, to predict pixel-wise crack segmentation in an end-to-end method. DeepCrack consists of the extended Fully Convolutional Networks (FCN) and the Deeply-Supervised Nets (DSN). During the training, the elaborately designed model learns and aggregates multi-scale and multi-level features from the low convolutional layers to the high-level convolutional layers, which is different from the standard approaches of only using the last convolutional layer. DSN provides integrated direct supervision for features of each convolutional stage. We apply both guided filtering and Conditional Random Fields (CRFs) methods to refine the final prediction results. A benchmark dataset consisting of 537 images with manual annotation maps are built to verify the effectiveness of our proposed method. Our method achieved state-of-the-art performances on the proposed dataset (mean I/U of 85.9, best F-score of 86.5, and 0.1 s per image).

1. Introduction

Crack presents important information for the safety and durability of the man-made buildings. Hence, it is of great significance to detect and analyze cracks for the maintenance of these buildings. In practice, the demand for automatic crack detection has increased rapidly every year. A variety of traditional computer vision methods of crack detection from images have been proposed in the past decades. In general, these methods can be classified into two categories: local-feature-based and global-feature-based methods [1]. The former methods exploit the local features, such as intensity, gradient, local variance, and local texture anisotropy. The later methods track and extracts crack curves in an overall view through dynamic programming to optimize target functions based on certain criteria.

Since visually salient cracks are related to a lots of visual patterns, it is difficult to design a universal method to deal with cracks in various scenes. In other words, precisely detecting crack from natural images involves visual perception of various "levels" [2,3]. However, both the mentioned two kinds of crack detection methods can not satisfy this requirement so that they usually suffer a series of problems in practice. Recently, the supervised deep learning methods, such as Convolutional Neural Networks (CNNs), have achieved state-of-the-art performances in many high-level computer vision tasks, such as image recognition [4], object detection [5–7], and semantic segmentation [8–12]. These are powerful visual models that yield hierarchical features, which is an ideal method to aggregate multiple "levels".

In the conventional approaches, the main problem in crack detection is the way to deal with noises introduced by stains, spots, uneven illumination, blurring, and multiple scenes. Some methods make an assumption that there is a clear distinction between noises and cracks, such as Iterative Clipping Method (ICM) [13] which assumed that the intensities along cracks were usually darker than those of its surroundings. Li et al. [14] presented a Neighboring Difference Histogram Method (NDHM) to segment a crack image with a globally optimized threshold. However, the above assumption suffer issues when there are many dark spots and shadows in the test image. To improve the correctness and completeness of crack detection, the wavelet transforms based methods [15,16] were proposed to raise up the crack regions. However, they may not handle well the cracks with high curvatures or bad continuities because of the anisotropic characteristics of wavelets. The "FoSA" approach [1] was proposed to extend the NDHM with a stronger anti-noise capability and the "CrackTree" method [17] was designed to remove the pavement shadows. These methods showed outstanding precision and recall rates, but they are too complex and time-consuming. Some methods took both brightness and connectivity into consideration by

* Corresponding author.
E-mail address: jian.yao@whu.edu.cn (J. Yao).
URL: http://cvrs.whu.edu.cn/ (J. Yao)

https://doi.org/10.1016/j.neucom.2019.01.036
0925-2312/

measuring the image texture anisotropy, such as Conditional Texture Anisotropy (CTA) [18] and Free-Form Anisotropy (FFA) [19], which showed good results on crack segmentation but were sensitive to the edges which may enhance the noises sometimes. Then, saliency-based methods [20] were proposed to suppress noises. Hu et al. [21] used the Local Binary Pattern (LBP) method to analyze the basic local features to get good crack segmentation results, but the parameters need to be adjusted for each image. Zhang et al. [22] proposed using the Black Top-Hat (BTH) transformation and the threshold segmentation method to detect cracks from the concrete tunnel surface images but their method may encounter problems in uneven illumination images. More recently, Zhang et al. [23] established a Region of Aggregation (ROA) method to take multi-cues (cracks' spatial distribution, intensities, and geometric features) into account and a Region of Belief (ROB) concept for crack region growing. However, such method was only designed for thin cracks in 2–5 pixels wide.

Shortcomings of these traditional crack detections are obvious: each method was specifically designed for a specific database or scene. Once the dataset or scene are changed, the crack detectors tend to suffer failure. For example, the FoSA, CrackTree, and FFA methods can work well for thin cracks but fail to deal with wide cracks. The detector sometimes works, but sometimes fails, which means that the extracted features are not generalized well. The CNNs show powerful abilities that yield abundant hierarchical features, which can make breakthroughs in such applications.

Up till now, there are some attempts of applying CNNs on object segmentation. The first strategy utilizes separated mechanisms for feature extraction and image segmentation, in which it treats the CNNs as an assistant tool of the traditional computer vision methods. Some representative examples like [24,25] applied the CNNs to extract meaningful features, used superpixels to represent the structural pattern of the image, and then obtained the final labeling by aggregation of the classifier prediction. However, if there are errors in the initial superpixels, it may lead to poor predictions. The second strategy directly learns a nonlinear model from the images and ground truth label maps. Chen et al. [8,26] used a Conditional Random Field (CRF) to refine the low-resolution segmentation results obtained from a CNN, which means that they employed the CRF as a post-processing step (separated from the CNN training). Fully Convolutional Networks (FCN) [9] learns to upsample its feature maps – the outputs of the convolutional layers – to achieve pixel-wise segmentation, but can not produce very accurate labeling results. FCN showed that different stages of the convolutional layers obtained diverse meaningful features, for example, low layers kept more structure information, while the top layers obtained more abstract features for object recognition. Therefore, coarse feature maps of the top layers are not enough to obtain the refined segmentation results. To overcome it, Zheng et al. [10] developed an end-to-end network which jointly learned the parameters of the CNN and CRF in a unified architecture termed CRF-RNN. The predictive performance of FCN was improved further by CRF-RNN and fine-tuning it on large datasets [27]. The fact that joint training helps was also presented in other recent studies [28,29]. Meanwhile, the deconvolutional network [30] and its semi-supervised variant decoupled network [31] achieved better performances than FCN although at the cost of a more complex training. SegNet [12] proposed an idea of the encoder-decoder architecture, which discards the fully connected layers and shows the benefit of reducing the number of parameters significantly. Multi-scale deep architectures were also developed, which came into two main directions: (1) inputting images at several scales into the networks [11,29]; (2) combining feature maps from different layers of a deep architecture [32–34]. Their collective ideas were to learn multi-scale features, in which both local and global features will be kept.

With the development of deep learning, a few methods that applied CNNs to achieve crack detection has appeared sequentially. Some methods like [35–37] treated CNNs as a classifier to predict a label for each local patch, which was still far from pixel-wise segmentation. In addition, both the mentioned traditional methods and CNNs-based methods have not published an open crack detection dataset, so the performances of these methods are sometimes difficult to make a comparison. As shown in Fig. 1, we explore the meaningful features of each level layer in the following way: (1) predicting crack segmentation results with feature maps of each convolutional stage, termed as side output; (2) concatenating all side outputs to produce a final fused result; (3) supervising both side outputs and fused results by Deeply-Supervised Nets (DSN) [38], which forms an integrated direct supervision; (4) refining the final fused result by applying Guided Filtering (GF) [39]. The parallel architecture like HED [3] was applied to crack detection and achieved the state-of-the-art performance. DeepCrack takes full use of the advanced methods to achieve a deep end-to-end and pixel-wise crack segmentation architecture, which is the core topic of this paper. In addition, we built an open benchmark to evaluate the crack detection systems, in which multi-scale and multi-scene cracks are manually annotated.

In summary, our proposed crack segmentation method has the following contributions:

1. We developed an automatic crack segmentation method based on CNNs. It learns hierarchical features of cracks in multiple scenes and scales effectively. Then, both CRFs and GF methods are applied to refine the predictions of CNNs.
2. We explored the learning stage of CNN by using specially designed loss function to alleviate the imbalanced data distribution, in which the negative pixels are far more than the positive ones. To make the training effective, we apply DSN to facilitate the feature learning of each convolutional stage.
3. We established a public benchmark dataset with cracks in multi-scale and multi-scene to evaluate the crack detection systems. All of the crack images in our dataset are manually annotated. To our knowledge, this is the first open work in such field.
4. We established complete metrics to evaluate crack detection systems, including semantic segmentation evaluations, Precision-Recall curve, and Receiver Operating Characteristic (ROC) curve.

2. Proposed method

2.1. Model architecture

We formulate crack segmentation as a binary image labeling problem, where "0" and "1" refer to "non-crack" and "crack", respectively. Such application is a task that requires both high-level features and low-level cues [40]. Our architecture, as showed in Fig. 1, aggregates hierarchical features acquired from multiple layers. We use the 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG-16 net [41] designed for object classification. The fully connected layers and fifth pooling layer are discarded due to the following reasons: (1) we expect the meaningful side-output with different scales, and a layer after the fifth pooling yields a too small output plane (the interpolated prediction feature map is too fuzzy to generate a refined result); (2) the fully connected layers are computationally intensive, which is memory/time-consuming [3]. Each convolutional layer is comprised of convolution, batch normalization [42] and Rectified Linear Unit (ReLU) [43]. Here, the convolution is a process with a filter bank to produce a set of feature maps. The batch normalization is applied to reduce internal covariate shift. The ReLU layer

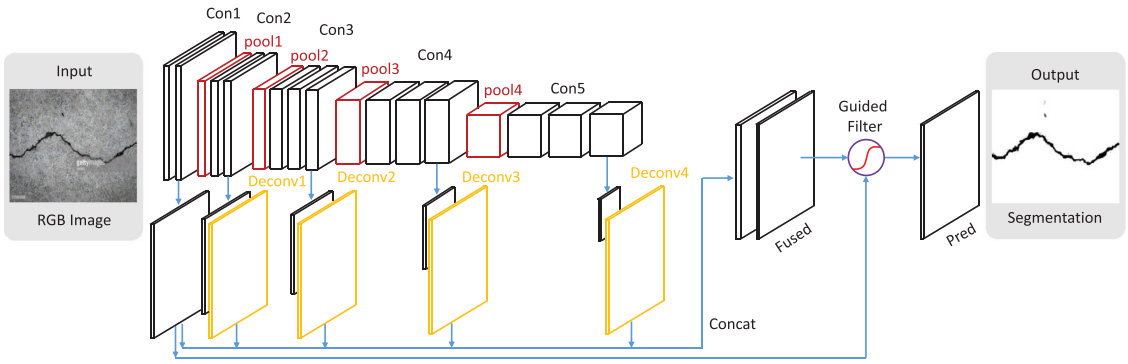


Fig. 1. An illustration of our proposed DeepCrack architecture. In this architecture, there are no fully connected layers, the side-output layers are inserted after the convolutional layers, deep supervision is applied at each side-output layer and then all of them are concatenated to form a final fused output. In this way, the final output layer acquires multi-scale and multi-level features as the plane size of the input of side-output layers becomes smaller and the receptive field size becomes larger. The fused prediction is refined by guided filtering with the first side-output layer.

computes the activation function $\max(0, x)$, which makes the networks able to learn a non-linear task. The spatial pooling is carried out by four max-pooling layers, which follow some of the convolutional layers in each stage (not all the convolutional layers are followed by plane size reduction operation, respectively conv1_2, conv2_2, conv3_3 and conv4_3). The plane size reduction operation is achieved by a stride 2 block: a Max-pooling with 2×2 pixel filter. It is used to achieve translation invariance over small spatial shifts in the image, which can also reduce the parameter size of the networks. The side-output features are obtained by a convolutional layer with a kernel size 1 and a number of output N. Except the first side-output layer, other four side-output layers are followed by deconvolutional layers, which are applied to upsample the plane size of the feature maps to be the same as the input image. Then, the upsampled feature maps are concatenated to form final features, which are followed by a convolutional layer and a softmax layer. The output of the softmax layer is a N-channel map of the probabilities where N is the number of classes ($N = 2$ in our application). According to the prediction of the softmax layer, we can get a predicted label for each pixel by a fixed threshold.

For a same input image, there is a fact that most predictions of lower convolutional stages preserve well the crack region boundaries but are sensitive to local noise, such as dark spots and dirt. On the contrary, predictions of deeper convolutional stages show better anti-noise abilities but fail in preserving the segmentation boundaries. Therefore, it is a trade-off strategy to linearly fuse the predictions of different convolutional stages. We explore further and first propose a novel method to refine the fused prediction by applying Guided Filtering [39], named as *guided feathering*. First, a binary mask is generated by the fused prediction. Then, the side-output of conv1_2 is set as a guidance map. The guided filter achieves the final refined prediction by well preserving the crack regions and removing the noises in the low level prediction. Comparing to the CRF method, such technique is faster and more efficient.

Our architecture consists of main three parts: (1) the convolutional layers which correspond to the first 13 convolutional layers in the VGG-16 net [41]; (2) the side-output layers; (3) the refinement module.

Fig. 2 shows the main architecture of our proposed model, in which the details on each operations are presented. There are some differences with the VGG-16 networks: (1) we insert batch normalization (BN) layer [42] between the convolutional layer and ReLU layer [43], which is applied to improve model generalization; (2) the pool5 layer and fully connected layers are discarded. Hence, it is a fully convolutional network.

The side-output features are obtained by a convolutional layer with a kernel size 1 and a number of output N (N=2). It can be treated as a linear fusion method and outputs a prediction map. Given the differences of receptive field size, deeper predictions are less affected by noises while lower ones present more detailed boundaries. The final fused result is a trade-off of these side-output predictions, as showed in Fig. 2.

We apply two methods to refine the fused prediction: Conditional Random Field (CRF) and Guided Filtering (GF) [39]. Here, the CRF-based methods is similar to CRF-RNN [10]. Fig. 3 presents the guided filtering process, in which the predictions of fused and side-output 1 are input p and guidance I, respectively. The parameters are $r = 5, \epsilon = 1e-6$ for the guided filter.

Our method has several advantages: (1) applying the whole image to train and generate pixel-wise predictions in an end-to-end manner, which is easy-to-use in practice; (2) the hierarchical feature learning in a network, which has been proved beneficial in some computer tasks [11,33,44]; (3) refining the fused predictions with an efficient guided filtering; (4) the computation efficiency due to discarding the fully connected layers (reducing the parameters of VGG-16 nets significantly from 134M to 14.7M).

2.2. Loss formulation

Most of the notations and formulations of the proposed method follow those of HED [3], but with some differences. For crack segmentation, we define the training set by $\mathbf{S} = \{(\mathbf{I}_n, \mathbf{G}_n), n = 1, \ldots, N\}$, where the image sample $\mathbf{I}_n = \{I_j^{(n)}, j = 1, \ldots, |\mathbf{I}_n|\}$ denotes the original input image and $\mathbf{G}_n = \{G_j^{(n)}, j = 1, \ldots, |\mathbf{G}_n|\}, Y_j^{(n)} \in \{0, 1\}$. For simplicity, we consider each image independently and the index i will be omitted hereafter. Each side-output layer during the training stage, as showed in Fig. 4, will be evaluated respectively. The goal of the training is to learn a model that minimizes the differences between the final prediction of the network and the ground truth. To learn meaningful features for crack segmentation, we apply DSN [38] to supervise each side-output layer. Each side-output layer can be treated as a pixel-wise classifier with the corresponding weights $\mathbf{w} = \{(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(M)})\}$, where M is the number of side-output layers. We denote all the parameters of the network as $\mathbf{W}$, and then the loss function is formulated as

$$\begin{aligned} \mathcal{L}_{\text{side}}(\mathbf{I}, \mathbf{G}, \mathbf{W}, \mathbf{w}) &= \sum_{m=1}^{M} \alpha_m \ell_{\text{side}}(\mathbf{I}, \mathbf{G}, \mathbf{W}, \mathbf{w}^{(m)}) \\ &= \sum_{m=1}^{M} \alpha_m \Delta(\mathbf{P}^{(m)}, \mathbf{G}, \mathbf{W}, \mathbf{w}^{(m)}), \end{aligned} \tag{1}$$

where ℓ_{side} refers to the image-level loss function for side-output, $\mathbf{P} = \{P_j, j = 1, \ldots, |\mathbf{I}|\}, P_j \in \{0, 1\}$ refers to the predicted results of

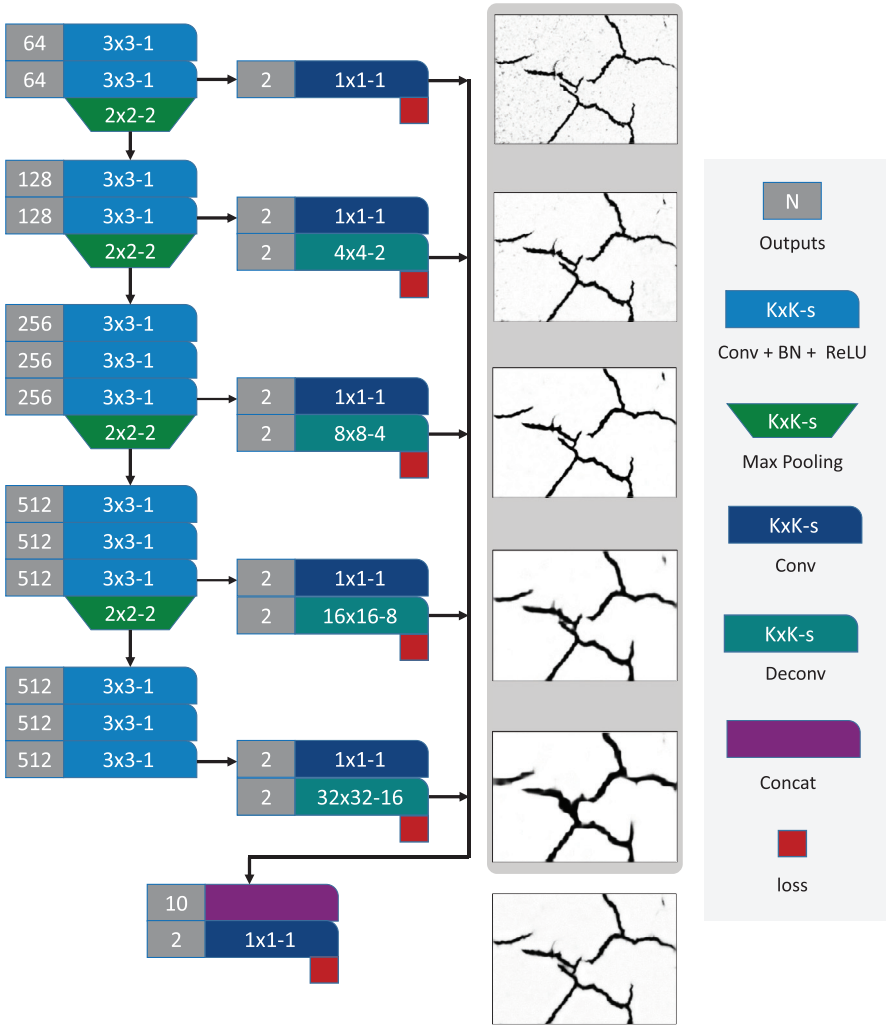


Fig. 2. The details of our convolutional network.

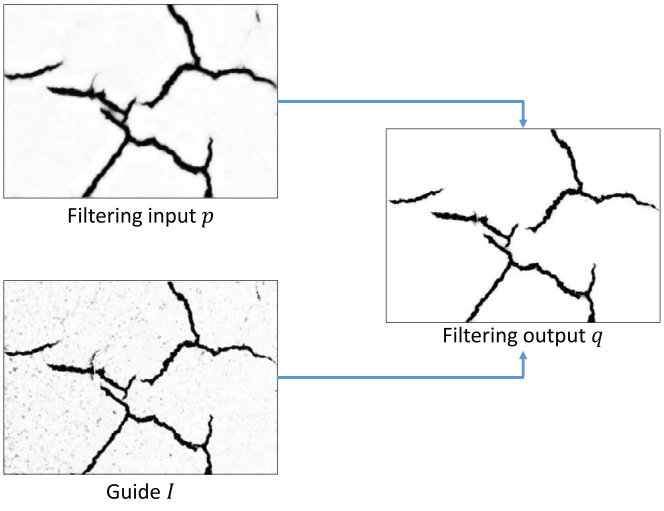


Fig. 3. The illumination of guided filtering refinement module.

the m-th side-output layer, which is upsampled to the raw image size when necessary, α_m is a hyper-parameter denoted as the loss weight for each side-output layer. In our image-to-image training, the modified cross-entropy function Δ is defined as

$$\begin{aligned}\Delta = &- \sum_{j\in\mathbf{G}_+} w_0 \log \Pr(P_j = 1|\mathbf{I}, \mathbf{W}, \mathbf{w}) \\ &- \sum_{j\in\mathbf{G}_-} w_1 \log \Pr(P_j = 0|\mathbf{I}, \mathbf{W}, \mathbf{w}),\end{aligned} \tag{2}$$

where we denote $|\mathbf{G}|, |\mathbf{G}_+|, |\mathbf{G}_-|$ as the total number of all pixels, all positive pixels and all negative pixels for an input image $\mathbf{I}$, respectively. w_0 and w_1 are the class loss weights for corresponding non-crack and crack pixels, respectively. $\Pr(\cdot)$ refers to the probability of positive or negative for a pixel in the predicted map. Let C_0 and C_1 be the total numbers of non-crack (negative) pixels and crack (positive) ones in the total training set, respectively. Given that over 95% of the ground truth are non-crack, the simple cross-entropy loss can cause training difficulties due to the saturation behavior of the activation function. Therefore, we need to weight the loss differently which termed *class balancing* [12]. We set $w_0 = 1.0$ (when pixel j is a negative) and $w_1 = \frac{C_0}{C_1}$ (when pixel j is a positive) to achieve such goal.

Each the side-output layer can be applied to generate a prediction map, which contributes to the corresponding side-output loss. The side-output layers are concatenated to a final fused prediction

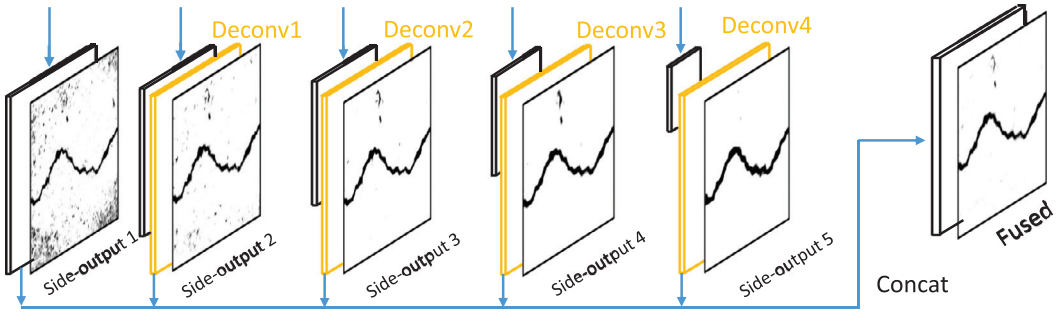


Fig. 4. An illustration of side-output predictions. The predictions consist of "side-output m" ($m = 1, \ldots, 5$). Each side-output can produce the loss termed as $\alpha_m \Delta(\mathbf{P}^{(m)}, \mathbf{G}, \mathbf{W}, \mathbf{w}^{(m)})$. The final fused prediction "Fused" also produces the loss termed as $\mathcal{L}_{\text{fuse}}$.

Fig. 5. Some representative samples in our benchmark database. The images were obtained by main two ways: (1) we downloaded from the Internet; (2) we took some photos of the real cracks personally.

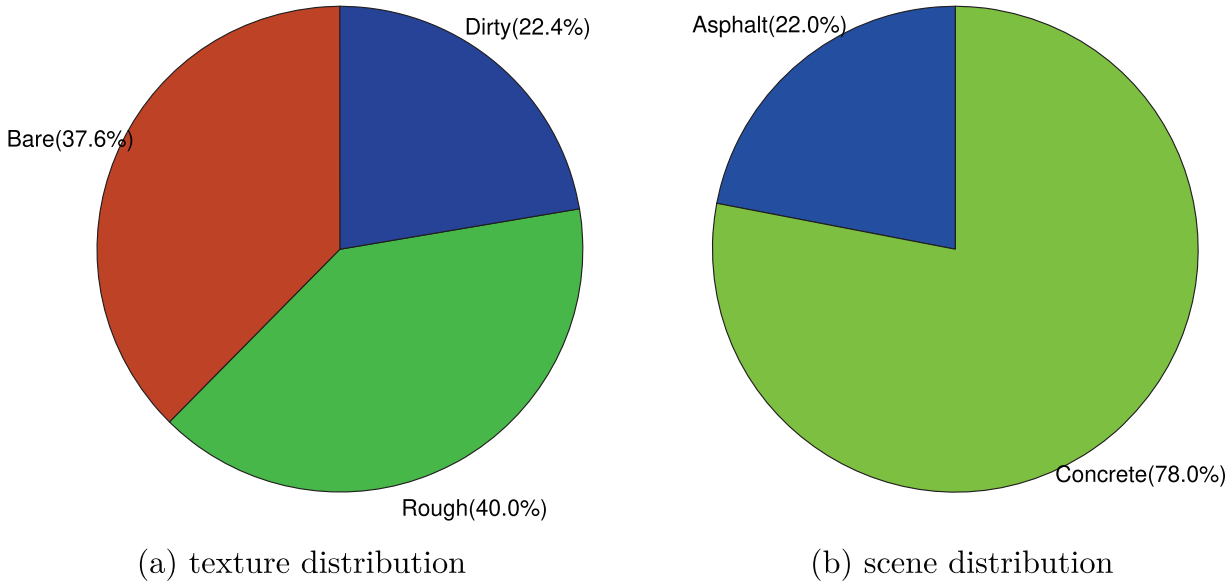


Fig. 6. Statistics of textures and scenes distribution of our database.

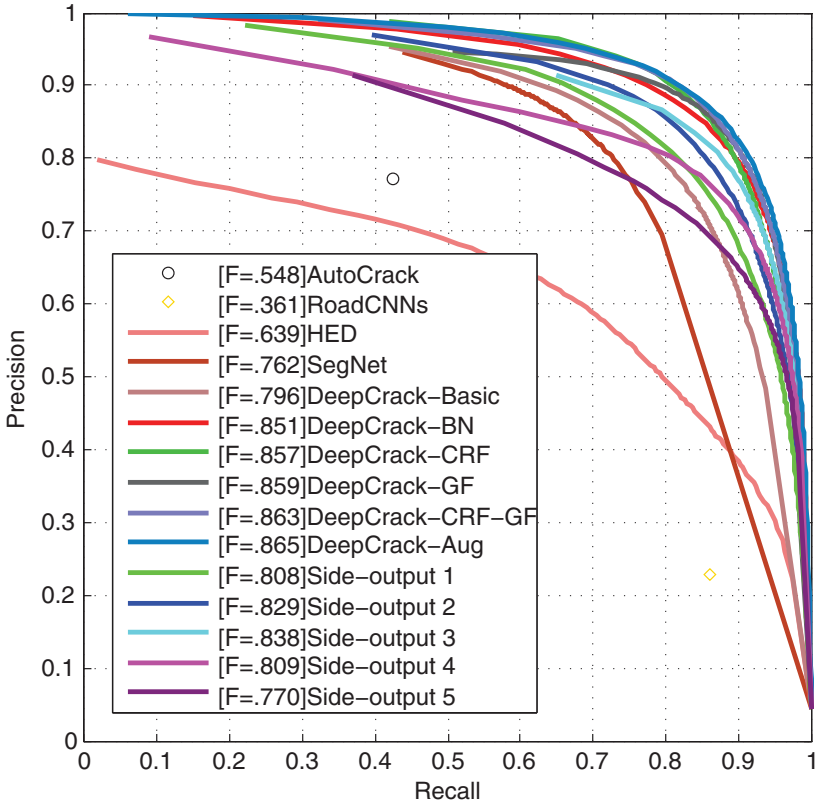


Fig. 7. The Precision-Recall (PR) curve for crack segmentation on our database.

which produces the fused loss termed $\mathcal{L}_{\text{fuse}}$ as

$$\begin{aligned}\mathcal{L}_{\text{fuse}}(\mathbf{I},\mathbf{G},\mathbf{W}) = &-\sum_{j\in\mathbf{G}_+} w_0 \log \Pr(P_j=1|\mathbf{I},\mathbf{W}) \\ &-\sum_{j\in\mathbf{G}_-} w_1 \log \Pr(P_j=0|\mathbf{I},\mathbf{W}),\end{aligned} \tag{3}$$

where $\mathbf{I}$, $\mathbf{G}$, w_0 and w_1 denote the same meanings with those in Eq. (2). Therefore, our overall loss function becomes

$$\mathcal{L} = \mathcal{L}_{\text{side}}(\mathbf{I},\mathbf{G},\mathbf{W},\mathbf{w}) + \mathcal{L}_{\text{fuse}}(\mathbf{I},\mathbf{G},\mathbf{W}). \tag{4}$$

2.3. Model parameters

We trained our network using the publicly available *Caffe* [45] library and built it on top of the implementations of FCN [9], DSN [38], HED [3] and SegNet [12]. We used the Stochastic Gradient Descent (SGD) [46] method to optimize our model. The model parameters we tuned are: the size of input image ($544\times384\times3$), the size of ground truth ($544\times384\times1$), the size of mini-batch (1), the learning rate (1e−4), the loss weight for each side-output layer and the final fused layer (1.0), the momentum (0.9), the weight decay (2e-4), and the training iterations

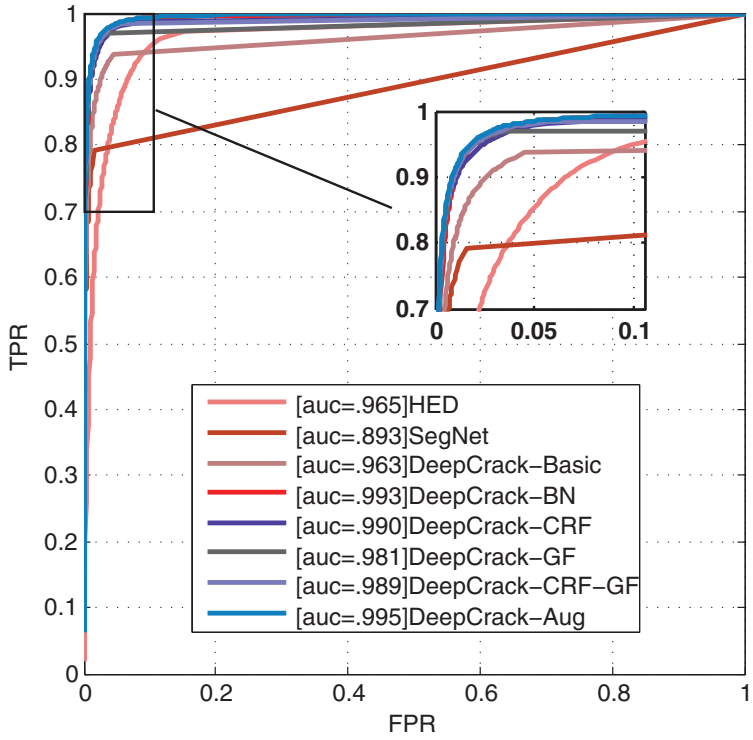


Fig. 8. The ROC curve for crack segmentation on our database. Our methods DeepCrack-Aug achieves the best AUC score.

(2e5; reduce learning rate by 1/5 after 5e4). In later experiments, we fixed the values of all parameters discussed above.

Significantly, our proposed network are efficiently trained without utilizing any pre-trained models for main two reasons:

1. Our task aims to distinguish only two classes (i.e., crack and non-crack), which is easier than general semantic segmentation issues (e.g., 21 classes for PASCAL VOC [47]). In addition, there are great differences about the semantic categories between PASCAL VOC dataset and our crack detection dataset, which lead to a result that initializing the proposed network with the pre-trained models makes few effects.
2. The usage of batch normalization, side output supervision and elaborate loss function improve the convergence and accuracy of the proposed network.

2.4. Data augmentation

Data augmentation has been proven to be a crucial technique in deep networks [3,4]. We rotated the images to 8 different angles (every 45° in [0°, 360°)) and cropped the largest rectangle in the rotated image (without the blank regions produced by rotation). We also flipped the images at each angle horizontally. Hence, the dataset were augmented by 16 times. In the training stage, we applied both the raw images and augmented ones to train networks. We resize the input images at 256×256 because of the rotation transformation, which is different from the training without data augmentation.

All models were trained and tested on a single NVIDIA TITAN X. For a 544×384 input image, inference time is about 0.1 s. So, testing is efficient.

3. Experiments

3.1. Benchmark

In previous studies, every method build a specific small evaluation dataset to verify its effectiveness. It makes the comparison of different methods very difficult. Hence, a good crack detection dataset will deal with this problem well. Therefore, we have established an open benchmark database[1] that can provide empirical basis for research on crack detection and segmentation. The dataset is consist of 537 RGB color images with manually annotated segmentations. Some representative images and their corresponding segmentations are shown in Fig. 5, respectively. All of the segmentations were issued by presenting the subject with a binary image. The images were divided into two main subsets: a training set with 300 images and a testing set with 237 ones. Each image is made available to a pixel-wise segmentation map, which presents to be a mask exactly covering the crack regions. All of the images are of a fixed size of 544×384 pixels.

Table 1
The percentages of crack pixels and non-crack ones.

	Crack pixels (%)	Non-crack pixels (%)
Training	2.91	97.09
Test	4.33	95.67
Total	3.54	96.46

Table 1 displays the percentages of crack pixels and non-crack ones of the database, which accords with the fact that the crack regions only occupy a small proportion of the images. We chose crack images in various scenes and scales to universally represent the characteristics of cracks. Fig. 6 shows the statistics for the semantic annotation of the major textures and scenes distribution. For the bare type, it presents as a clean and smooth texture of the background. Therefore, there is a remarkable contrast between crack and non-crack regions. For the rough type, it presents as a cratered or rough surface. For the dirty type, there are plenty of spots and stains distributing in the image. The contrast between crack and non-crack regions is lower for the latter two types. Asphalt and concrete, which are commonly used in man-made

[1] Available at: https://github.com/yhlleo/DeepCrack.

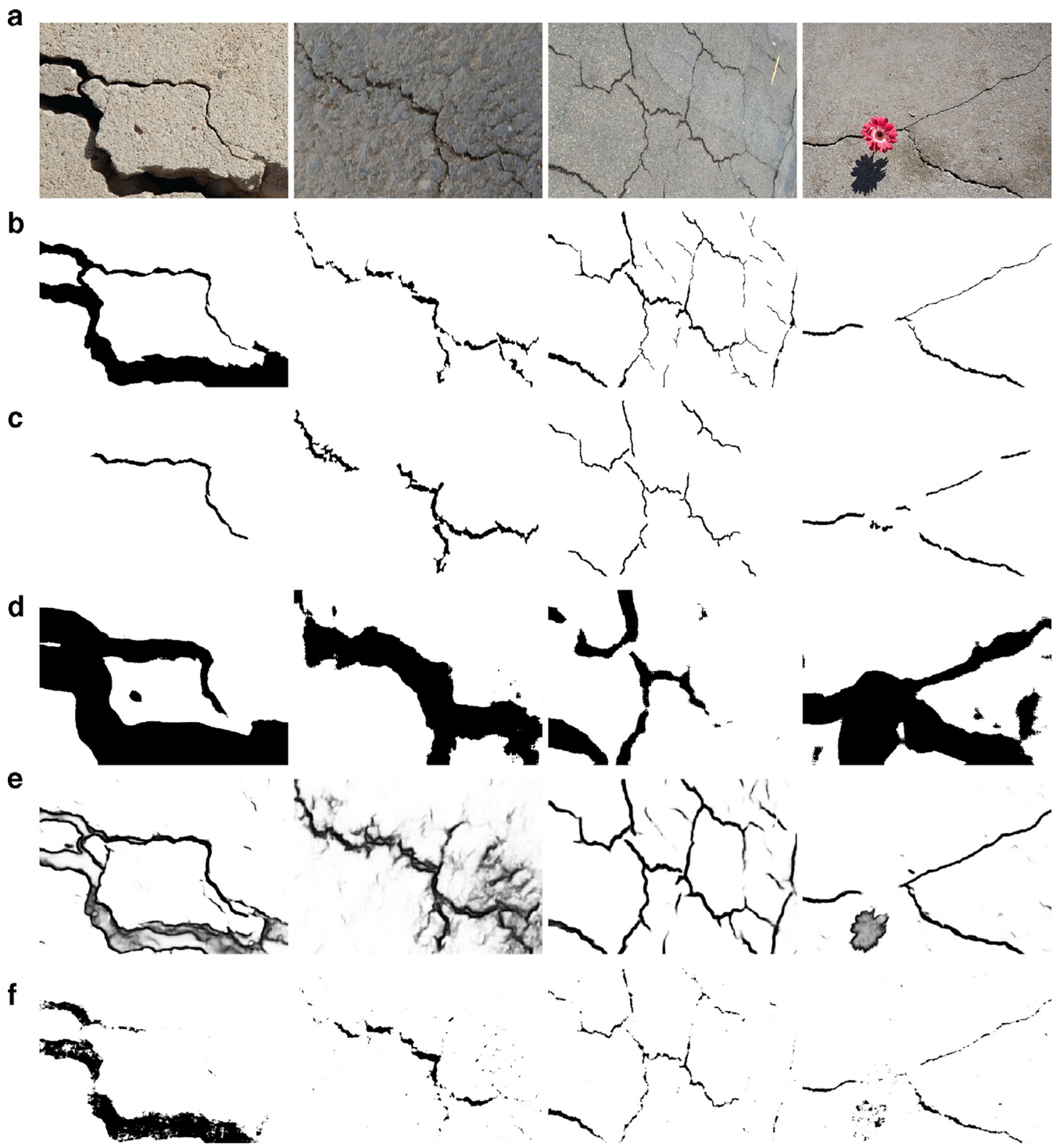


Fig. 9. Results on several samples with thin cracks of our Crack Detection Database. In each column we present (a) original images, (b) ground truth, (c) AutoCrack [22], (d) RoadCNNs [37], (e) HED [3], (f) SegNet [12], respectively.

buildings, are two major scenes of the database. The width of cracks is in ranges from 1 pixel to 180 ones in the database. So, multiple textures, scenes and scales make the crack segmentation a challenging task on our built database.

There is another public benchmark database for asphalt pavement crack detection [48]. The benchmark[2] contains a few annotated images(less than 40) and is specially designed for thin cracks in 2–5 pixels wide. We realized that the thin cracks with several pixels width and dashed-line shape are different from the wider crack regions. Post-processing, such as length constraint, curvature and geometric features etc., which are broadly applied in traditional methods [22,48], is requisite to obtain continuous and complete thin crack segments. However, it is the weakness of deep convolutional neural networks.

[2] Dataset: http://perso.lcpc.fr/sylvie.chambon/FISSURES/Datasets.html.

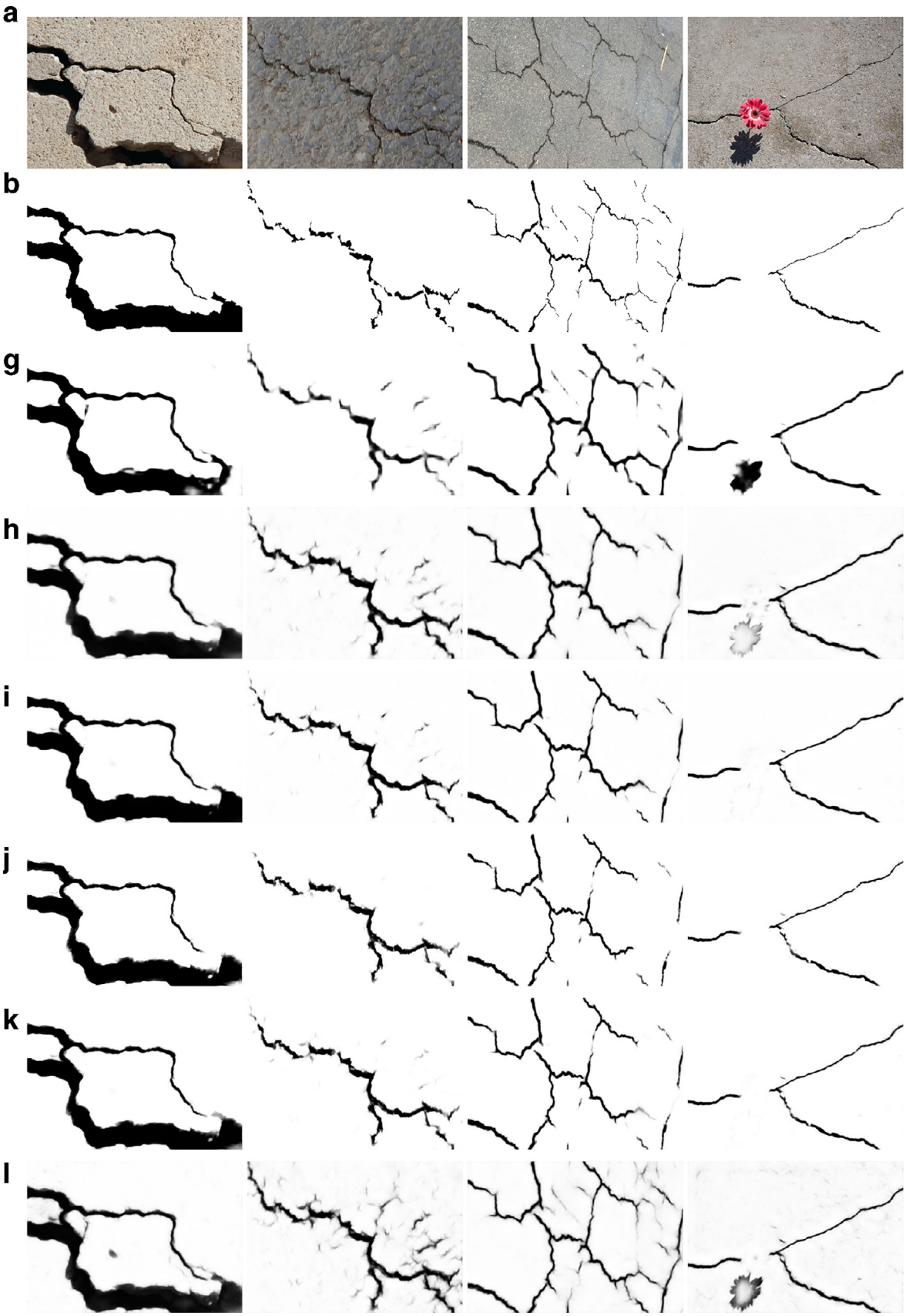


Fig. 10. Results on several samples with thin cracks of our Crack Detection Database. In each column we present (a) original images, (b) ground truth, (g) DeepCrack-Basic, (h) DeepCrack-BN, (i) DeepCrack-CRF, (j) DeepCrack-GF, (k) DeepCrack-CRF-GF and (l) DeepCrack-Aug, respectively.

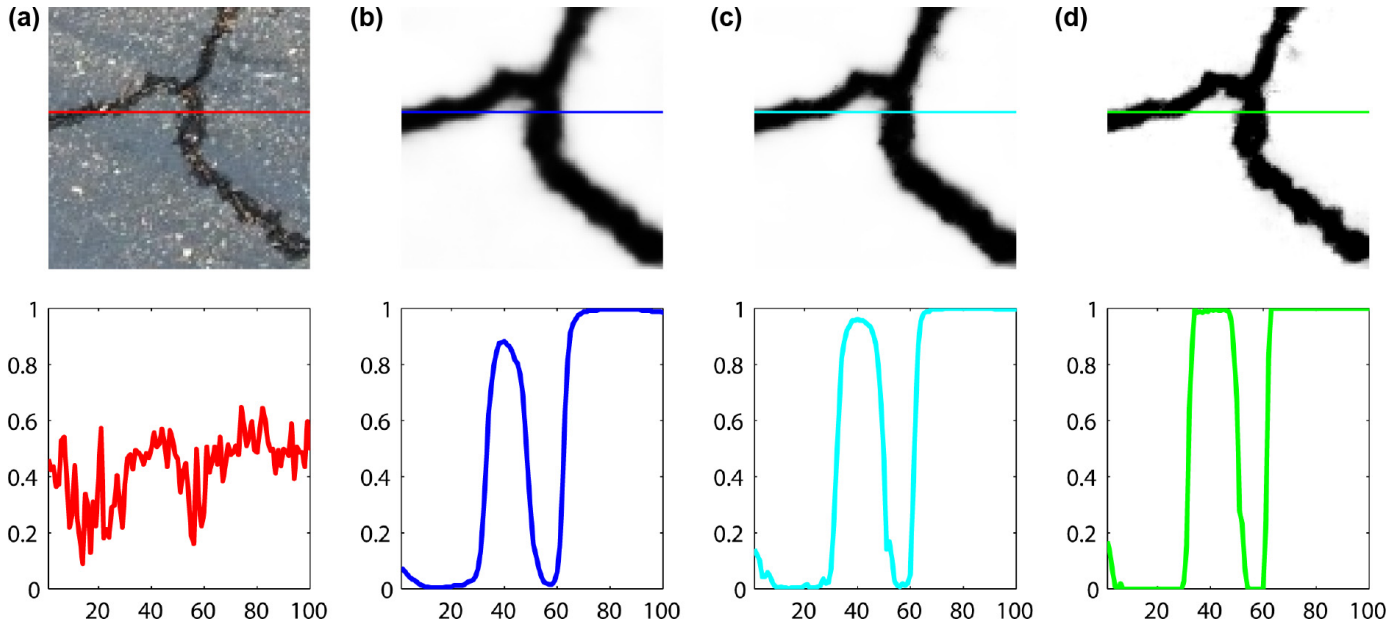


Fig. 11. An illustration of the different predictions. The first row images are original image (a), fused prediction (b), CRF refined result (c), and guided filtering refined result (d), respectively. The second row are plotting maps of the pixel value along the color lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

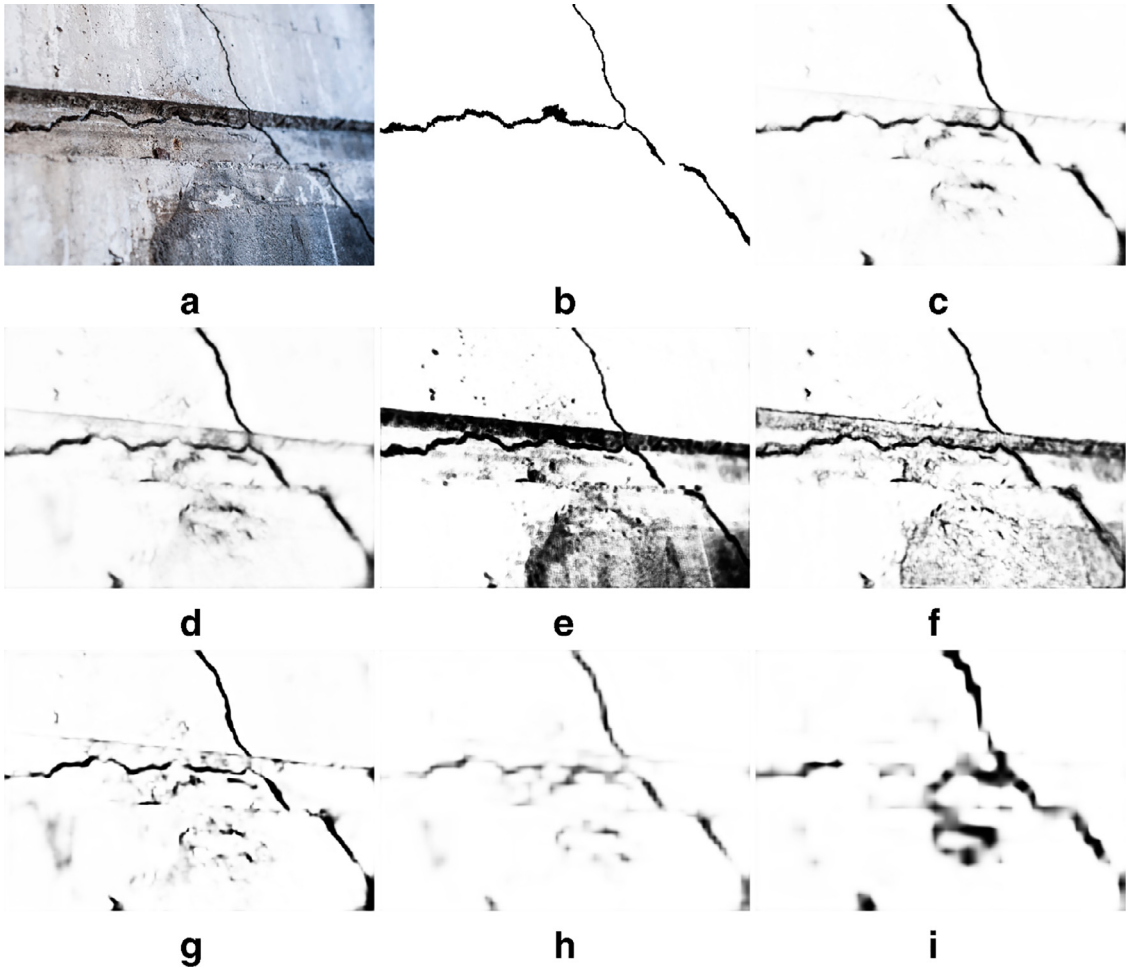


Fig. 12. The probability maps generated by DeepCrack-CRF: (a) the test sample; (b) ground truth, (c) CRF refined result, (d) linear fusion map with (e)–(i) (side-output 1–5), respectively. It implies that learned features are distinct in each stage. More detailed local features are retained in the lower level layers. Meanwhile more abstract features are represented in the deeper layers. The fused result and refined prediction show better performances.

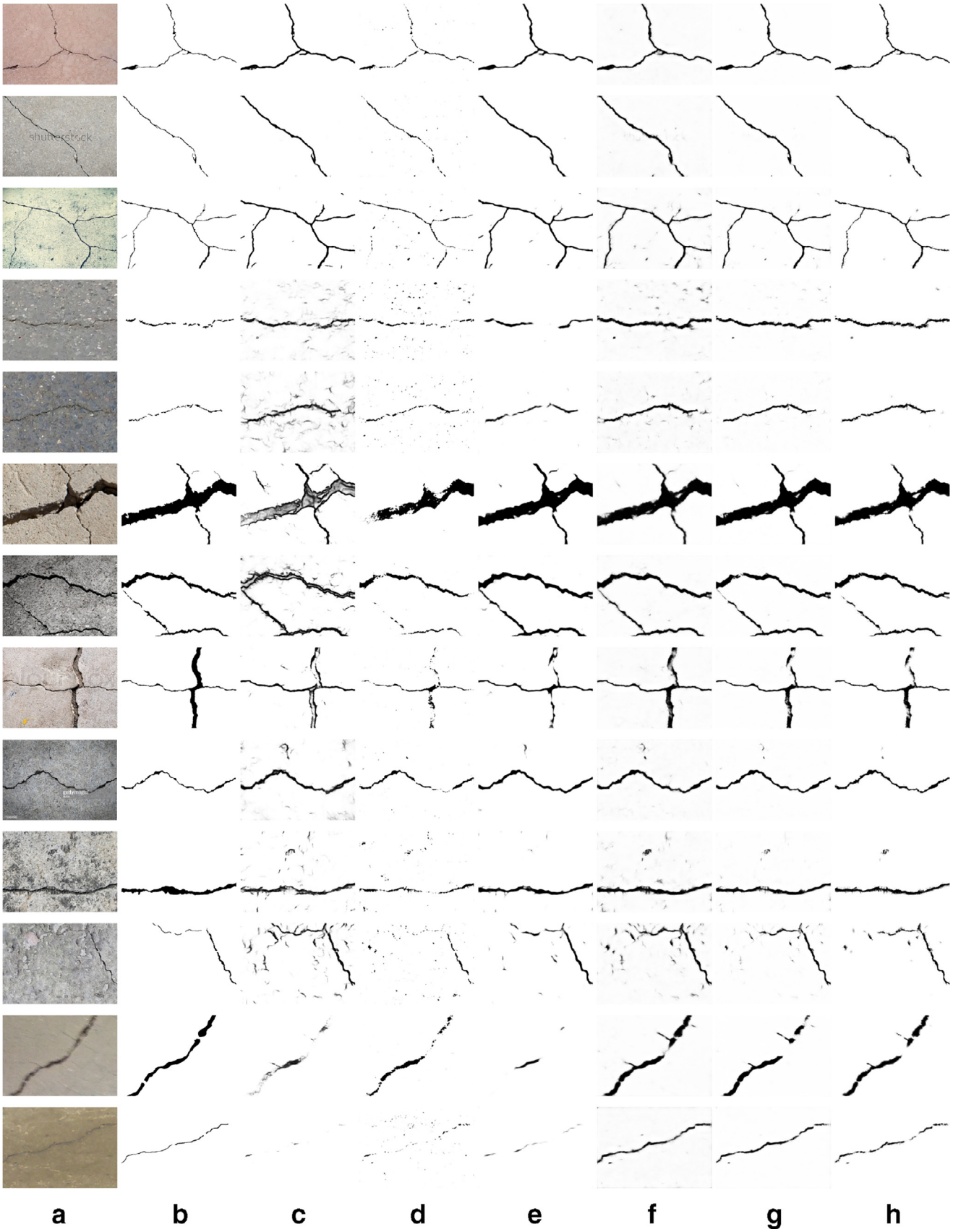


Fig. 13. Several samples with cracks in various scenes. The columns are: (a) original image, (b) ground truth, (c) HED [3], (d) SegNet [12], (e) DeepCrack-Basic, (f) DeepCrack-BN, (g) DeepCrack-CRF, and (h) DeepCrack-GF.

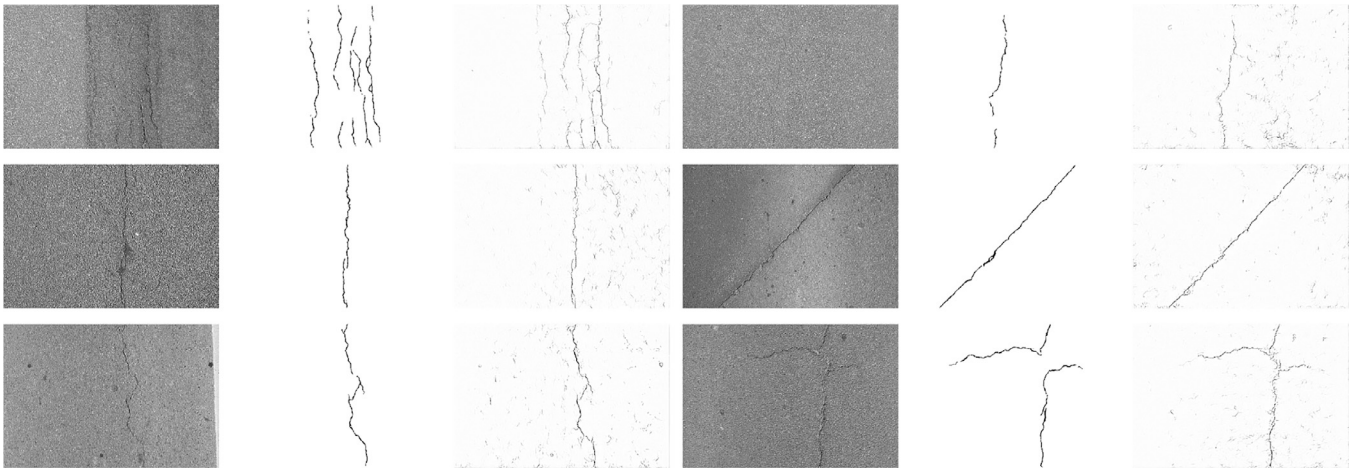


Fig. 14. We apply the DeepCrack-BN to extract reliable hypothesis crack regions on the dataset [48]. Predictions of DeepCrack-BN with darker pixels are more likely to be crack pixels. Inference time is about 0.3 s for the test samples with the size of 512×768 pixels.

3.2. Metrics

We performed the major experiments on our established benchmark database. To evaluate our work, we introduce three metrics of common semantic segmentation evaluations [9,12]. Let n_{ij} be the number of pixels of the class i predicted to be the class j, where there are n_{cls} different classes, and $t_i = \sum_j n_{ij}$ be the total number of pixels of the class i (both true positives and false positives are included). Then, we compute:

(1) Global accuracy (G), which measures the percentage of the pixels correctly predicted: $\Sigma_i n_{ii}/\Sigma_i t_i$,
(2) Class average accuracy (C), which means the predictive accuracy over all classes: $(1/n_{\text{cls}})\Sigma_i n_{ii}/t_i$,
(3) Mean intersection over union (I/U) over all classes: $(1/n_{\text{cls}}) \sum_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$.

In addition to evaluate the semantic segmentation, three common metrics in the crack detection field are computed as

(1) Precision (P) $= \frac{\#\text{TruePositives}}{\#\text{TruePositives}+\#\text{FalsePositives}}$,
(2) Recall (R) $= \frac{\#\text{TruePositives}}{\#\text{TruePositives}+\#\text{FalseNegatives}}$,
(3) F-score (F) $= \frac{2PR}{R+R}$.

Given that P-R metrics take no account of #TrueNegatives, we introduce a classical metric Receiver Operating Characteristic (ROC) curve. The performances of the mentioned methods in our paper are calculated the scores, respectively. For the ROC curve, we calculate three metrics:

(1) True Positive Rate, TPR $= \frac{\#\text{TruePositives}}{\#\text{TruePositives}+\#\text{FalseNegatives}}$,
(2) False Positive Rate, FPR $= \frac{\#\text{FalsePositives}}{\#\text{FalsePositives}+\#\text{TrueNegatives}}$,
(3) The Area Under the ROC Curve, AUC.

3.3. Evaluation

We trained our networks with six strategies: (1) DeepCrack-Basic used the HED [3] architecture with our loss function and was trained with original 300 training images; (2) DeepCrack-BN is the same as DeepCrack-Basic but adding batch normalization layers before each activation operation; (3) DeepCrack-CRF is the same as DeepCrack-BN but adding CRF after the network; (4) DeepCrack-GF is the same as DeepCrack-BN but applying the refining module of guided filtering, as showed in Fig. 1; (5) DeepCrack-CRF-GF linearly combined the prediction of DeepCrack-CRF and DeepCrack-GF, which was formulated as

$$\mathbf{P} = \beta \mathbf{P}_{\text{CRF}} + (1-\beta)\mathbf{P}_{\text{GF}}, \tag{5}$$

where $\mathbf{P}$ refers to the prediction map and β is the balancing weight which was set as 0.5; (6) DeepCrack-Aug is the same as DeepCrack-BN but trained with the augmented data of 9.6 k images. DeepCrack-CRF and DeepCrack-Aug networks were fine-tuned by the trained DeepCrack-BN model. To make the experiments convincing, we compared our method with other four typical methods: (1) AutoCrack [22], a traditional artificial designed detector; (2) RoadCNNs [37], a latest CNNs-based classifier for patch classification; (3) HED [3], an edge detection network achieving the state-of-the-art performances; (4) SegNet [12], a latest semantic segmentation network. We fine-tuned the HED and SegNet networks with their original architectures and loss functions on our augmented dataset.

We tried to binarize the probability maps with variant global thresholds. Fig. 7 shows the Precision-Recall curve generated by the threshold segmentation method. According to it, we can get the statistics, including inference time (*Times*), the best thresholds (T) and the mentioned metrics (*Metrics*), as showed in Table 2. Here, we calculated the G, C and I/U by having the same threshold as the best F-score value. Several test samples are shown in Figs. 9 and 10. Compared with the other four methods, our architecture shows noticeable improvement of performances. AutoCrack shows poor performances, when it encountered wider or weaker cracks. RoadCNNs is a patch-based classification method with CNNs which achieved very rough segmentations. HED was designed for edge detection which leads to the facts that the results of thin cracks are pretty good but very unfortunate for wider cracks. This is the main reason HED achieved poor performances. Therefore, there are some intrinsic differences between crack detection and edge detection. Batch normalization, which is treated as a regularizer, can reduce over-fitting for the network and boost performances a lot. Both conditional random fields and guided image filtering methods can be implemented to refine the dense predictions, but the latter is more faster and more efficient in such application. Fig. 11 presents the details on the predictions of DeepCrack-BN, DeepCrack-CRF and DeepCrack-GF, in which DeepCrack-GF achieves the sharpest boundaries. DeepCrack-CRF-GF shows a slight improvement in mean I/U and F-score, comparing to DeepCrack-CRF and DeepCrack-GF. DeepCrack-Aug was trained with augmented data and achieves the best F-score, which indicates that it is possible to increase the performances further with more annotated data. Besides, Table 2 shows that our proposed optimization method with both GF and CRF can slightly improve the model performances (GF improves about 0.8, CRF improves 0.6, GF-CRF improves 1.2). Compared with only augmenting the training

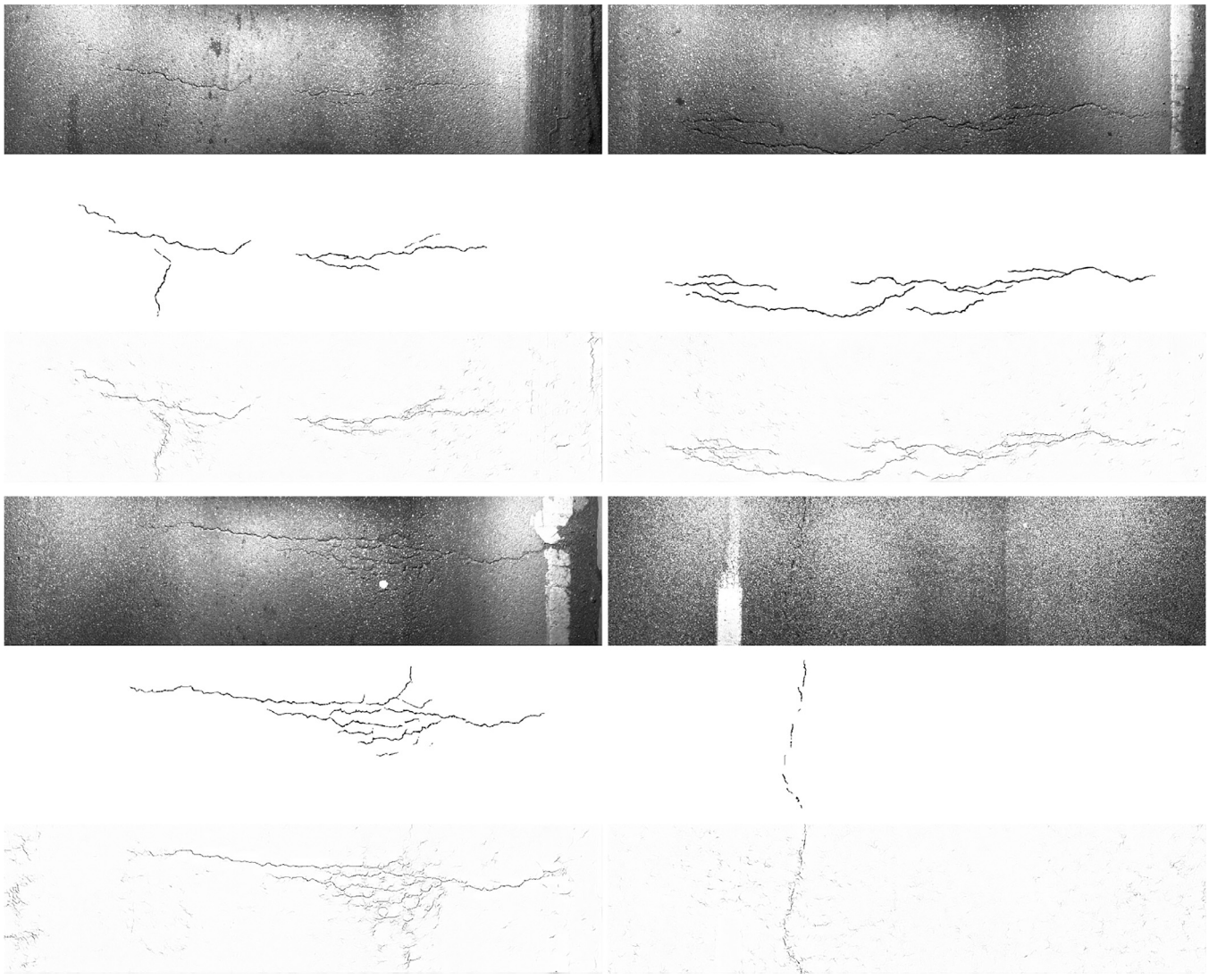


Fig. 15. We apply the DeepCrack-BN to extract reliable hypothesis crack regions on the dataset [48]. Predictions of DeepCrack-BN with darker pixels are more likely to be crack pixels. Inference time is about 0.8 s for the test samples with the size of 480×1920 pixels.

Table 2
Comparison of different methods on our testing database.

Methods	Times	T	Metrics					
	(ms)		G	C	I/U	P	R	F
AutoCrack [22]	–	–	97.0	70.9	67.3	77.2	42.4	54.8
RoadCNNs [37]	–	–	86.9	86.5	54.2	22.9	86.0	36.1
HED [3]	56	0.70	95.8	87.7	70.0	59.4	69.1	63.9
SegNet [12]	184	0.91	98.0	82.3	78.5	79.7	72.9	76.2
DeepCrack-Basic	74	0.86	97.8	92.5	80.1	79.4	79.9	79.6
DeepCrack-BN	109	0.87	97.8	96.6	81.3	83.9	86.3	85.1
DeepCrack-CRF	400+	0.90	98.2	95.4	83.6	**86.8**	84.6	85.7
DeepCrack-GF	118	0.75	**98.6**	95.0	**85.9**	85.2	86.6	85.9
DeepCrack-CRF-GF	400+	0.80	98.5	95.4	85.4	86.6	85.9	86.3
DeepCrack-Aug	109	0.86	97.5	**97.0**	80.2	86.1	**86.9**	**86.5**

dataset 16 times (Aug improves 1.4), its clear that our proposed refining post-processing methods are effective. Compared GF and CRF, the former method achieves better performances with lower computation. In addition, we applied both GF and CRF methods to refine the results of DeepCrack-Aug, in which we achieved a slight improvement on the performance (P=0.869, R=0.864 and F=0.867).

Table 3 displays the statistics of different level convolutional layers, including each side-output and the fused-output. Both two types of metrics present two similar tendencies: (1) all metrics are increasing gradually from the low level layers to the middle level ones first, then it turns into decreasing from the middle level layers to the high level ones; (2) both the simplest fusion (linear fusion) and guided filtering refinement improve the performances. It is logical that the low level layers represent more local features

Table 3
Results of each side-output and fused-output on our testing dataset in DeepCrack-GF.

Outputs	Metrics					
	G	C	I/U	P	R	F
Side-output 1	95.9	95.7	73.0	76.7	84.6	80.5
Side-output 2	97.1	95.8	77.7	77.9	87.1	82.3
Side-output 3	97.7	96.2	80.5	75.2	**91.0**	82.3
Side-output 4	97.1	96.2	77.9	75.1	87.4	80.7
Side-output 5	96.8	95.4	76.4	69.1	86.3	76.7
Fused results	97.8	**96.6**	81.3	83.9	86.3	85.1
Refined results	**98.6**	95.0	**85.9**	**85.2**	86.6	**85.9**

The fused results achieve best global accuracy, mean I/U, precision and F-score.

with a smaller receptive field. More non-crack pixels were predicted to be crack pixels, termed *false positives*, and lesser crack pixels were predicted to be non-crack pixels, termed *false negatives*, in such low level layers. It indicates that the low level features are susceptible to noise, such as spots, stains and other objects similar to cracks. With the layers becoming deeper, the meaningful local features learned from bigger receptive fields are more abstract, which is good to decrease false positives but is inferior to increase false negatives. Therefore, higher level features show more anti-noise capabilities. Fused hierarchical features can be treated as neutralization which aggregates the multiple level features from coarse to fine. Therefore, fused results show better integrated performances. Fig. 12 presents the outputs of different convolutional stages of a same sample, which verifies our analysis.

The ROC metrics are shown in Fig. 8. Though there are only small differences in the ROC curve, all of our DeepCrack networks achieves the better performances of AUC > 0.98. Compared with the other two methods (HED [3] and SegNet [12]), our architecture shows obvious improvements.

Some more test samples in specific scenes, including thin, wide, with stains and blurring, are respectively evaluated and presented in Fig. 13. Our method shows better performances on visual effects in these experiments.

As previously mentioned, post-processing, such as length constraint, curvature and geometric features etc., which are broadly applied in traditional methods [22,48], is requisite to obtain continuous and complete thin crack segments. However, it is the weakness of deep convolutional neural networks. We randomly choose half annotated images from [48] to fine-tuning our trained model DeepCrack-BN and obtain the predictions shown in Figs. 14 and 15. The results demonstrate that our CNNs-based methods can provide the reliable hypothesis for thin cracks with several pixels width and dashed-line shape.

4. Conclusion

Crack detection is a viable research. Unlike other detection tasks, segmenting the refined crack regions in pixel-wise is better than predicting bounding-boxes in practice. Our work makes contributions to propose a CNN-based learning method for semantic segmentation and establish a challenging benchmark dataset with multi-scene and multi-scale cracks.

We present a deep hierarchical features learning architecture, named DeepCrack, for crack segmentation, which is inspired by an edge detection network [3]. We keep the DSN module to provide integrated direct supervision for multi-level features learning, apply batch normalization [42] to reduce internal covariate shift, and modify the cross-entropy loss function for our application, and attempt to refine the predictions, such as conditional random fields and guided filtering. The guided filtering method is faster and more efficient than the conditional random fields. We established an open crack detection dataset to evaluate our method and compared with the up-to-date methods. Experimental results demonstrate that our method has achieved comparable performance with the state-of-the-art methods.

In the future, we will plan to exploit a better strategy to merge the features of side-output layers. More images of false crack regions will be added to the current benchmark database, which will make the database more comprehensive. Some more detailed metrics, such as accuracies on specific scenes and scales, will be proposed.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Project No. 41571436), the National Key Research and Development Program of China (Project No. 2017YFB1302400), and the Hubei Province Science and Technology Support Program, China (Project No. 2015BAA027).

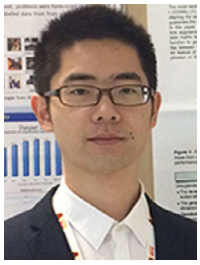
References

[1] Q. Li, Q. Zou, D. Zhang, Q. Mao, FoSA: F* seed-growing approach for crack-line detection from pavement images, Image Vis. Comput. 29 (12) (2011) 861–872.
[2] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, J. Physiol. 160 (1) (1962) 106–154.
[3] S. Xie, Z. Tu, Holistically-nested edge detection, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1395–1403.
[4] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.
[5] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.
[6] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.
[7] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2015, pp. 91–99.
[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Semantic image segmentation with deep convolutional nets and fully connected CRFs, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (2017) 834–848.
[9] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431–3440.
[10] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P.H. Torr, Conditional random fields as recurrent neural networks, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1529–1537.
[11] D. Eigen, R. Fergus, Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2650–2658.
[12] V. Badrinarayanan, A. Handa, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2017) 2481–2495.
[13] H. Oh, N.W. Garrick, L.E. Achenie, Segmentation algorithm using iterative clipping for processing noisy pavement images, in: Proceedings of the International Conference Imaging Technologies: Techniques and Applications in Civil Engineering, 1998.
[14] Q. Li, X. Liu, Novel approach to pavement image segmentation based on neighboring difference histogram method, in: Proceedings of the IEEE Congress on Image and Signal Processing, 2, 2008, pp. 792–796.
[15] J. Zhou, P.S. Huang, F.-P. Chiang, Wavelet-based pavement distress detection and evaluation, Opt. Eng. 45 (2) (2006). 027007–027007.
[16] S. Wu, Y. Liu, A segment algorithm for crack dection, in: Proceedings of the IEEE Symposium on Electrical & Electronics Engineering, 2012, pp. 674–677.
[17] Q. Zou, Y. Cao, Q. Li, Q. Mao, S. Wang, CrackTree: automatic crack detection from pavement images, Pattern Recognit. Lett. 33 (3) (2012) 227–238.
[18] F. Roli, Measure of texture anisotropy for crack detection on textured surfaces, Electron. Lett. 32 (14) (1996) 1274–1275.
[19] T.S. Nguyen, S. Begot, F. Duculty, M. Avila, Free-form anisotropy: a new method for crack detection on pavement surface images, in: Proceedings of the IEEE International Conference on Image Processing (ICIP), 2011.
[20] W. Xu, Z. Tang, J. Zhou, J. Ding, Pavement crack detection based on saliency and statistical features, in: Proceedings of the IEEE International Conference on Image Processing (ICIP), 2013.
[21] Y. Hu, C.-x. Zhao, A local binary pattern based methods for pavement crack detection, J. Pattern Recognit. Res. 5 (1) (2010) 140–147.
[22] W. Zhang, Z. Zhang, D. Qi, Y. Liu, Automatic crack detection and classification method for subway tunnel safety monitoring, Sensors 14 (10) (2014) 19307–19328.

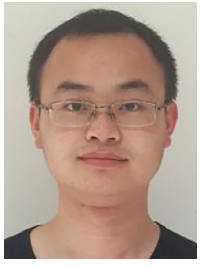
[23] D. Zhang, Q. Li, Y. Chen, M. Cao, L. He, B. Zhang, An efficient and reliable coarse-to-fine approach for asphalt pavement crack detection, Image Vis. Comput. 57 (2017) 130–146.
[24] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 35 (8) (2013) 1915–1929.
[25] C. Couprie, C. Farabet, L. Najman, Y. LeCun, Indoor semantic segmentation using depth information, International Conference on Learning Representations (ICLR) (2013).
[26] G. Papandreou, L.-C. Chen, K. Murphy, A.L. Yuille, Weakly-and semi-supervised learning of a DCNN for semantic image segmentation, Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (2015) 1742–1750.
[27] M. Everingham, S.A. Eslami, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective, Int. J. Comput. Vis. (IJCV) 111 (1) (2015) 98–136.
[28] A.G. Schwing, R. Urtasun, Fully connected deep structured networks, CoRR, arXiv:1503.02351 (2015).
[29] G. Lin, C. Shen, I. Reid, et al., Efficient piecewise training of deep structured models for semantic segmentation, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016).
[30] H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1520–1528.
[31] S. Hong, H. Noh, B. Han, Decoupled deep neural network for semi-supervised semantic segmentation, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2015, pp. 1495–1503.
[32] W. Liu, A. Rabinovich, A.C. Berg, Parsenet: looking wider to see better, International Conference on Learning Representations (ICLR) Workshop (2016).
[33] B. Hariharan, P. Arbeláez, R. Girshick, J. Malik, Hypercolumns for object segmentation and fine-grained localization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 447–456.
[34] T. Kong, A. Yao, Y. Chen, F. Sun, Hypernet: towards accurate region proposal generation and joint object detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016).
[35] D. Soukup, R. Huber-Mörk, Convolutional neural networks for steel surface defect detection from photometric stereo images, in: Proceedings of the International Symposium on Visual Computing, 2014, pp. 668–677.
[36] X. Gibert, V.M. Patel, R. Chellappa, Deep multi-task learning for railway track inspection, IEEE Transactions on Intelligent Transportation Systems 18 (2016) 153–164.
[37] L. Zhang, F. Yang, Y.D. Zhang, Y.J. Zhu, Road crack detection using deep convolutional neural network, in: Proceedings of the IEEE International Conference on Image Processing (ICIP), 2016, pp. 3708–3712.
[38] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, Z. Tu, Deeply-supervised nets, in: Proceedings of the AISTATS, 2, 2015, p. 6.
[39] K. He, J. Sun, X. Tang, Guided image filtering, IEEE Trans. Pattern Anal. Mach. Intell. 35 (6) (2013) 1397–1409.
[40] J. Yang, B. Price, S. Cohen, H. Lee, M.-H. Yang, Object contour detection with a fully convolutional encoder-decoder network, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) (2016).
[41] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, International Conference on Learning Representations (ICLR) (2015).
[42] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, International Conference on Machine Learning (ICML) (2015).
[43] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of International Conference on Machine Learning (ICML), 2010, pp. 807–814.
[44] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: Proceedings of the European Conference on Computer Vision (ECCV), 2014, pp. 346–361.
[45] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.
[46] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT, 2010, pp. 177–186.
[47] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[48] S. Chambon, J.-M. Moliard, Automatic road pavement assessment with image processing: review and comparison, Int. J. Geophys. 2011 (2011) 1–20.

Yahui Liu received the B.E. degree and M.E. degree from the School of Remote Sensing and Information Engineering at Wuhan University, China in 2015 and 2018, respectively. Now, he is pursuing a Ph.D. degree at University of Trento and FBK, Trento, Italy. He is the author of several international conference papers. Currently, his research focuses on include image segmentation, image generation and deep learning.

Jian Yao received the B.Sc. degree in Automation in 1997 from Xiamen University, China, the M.Sc. degree in Computer Science from Wuhan University, China, and the Ph.D. degree in Electronic Engineering in 2006 from The Chinese University of Hong Kong. From 2001 to 2002, he has ever worked as a Research Assistant at Shenzhen R&D Centre of City University of Hong Kong. From 2006 to 2008, he worked as a Postdoctoral Fellow in Computer Vision Group of IDIAP Research Institute, Martigny, Switzerland. From 2009 to 2011, he worked as a Research Grantholder in the Institute for the Protection and Security of the Citizen, European Commission Joint Research Centre (JRC), Ispra, Italy. From 2011 to 2012, he worked as a Professor in Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences, China. Since April 2012, he has been a Hubei "Chutian Scholar" Distinguished Professor with School of Remote Sensing and Information Engineering, Wuhan University, China, and the director of Computer Vision and Remote Sensing (CVRS) Lab (CVRS Website: http://cvrs.whu.edu.cn/), Wuhan University, China. He has published over 90 papers in international journals and proceedings of major conferences and is the inventor of over 20 patents. His current research interests mainly include computer vision, image processing, machine learning, LiDAR data processing, robotics, etc.



Xiaohu Lu received the B.E. degree and M.E. degree from the School of Remote Sensing and Information Engineering at Wuhan University, China in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of Civil, Environmental and Geodetic Engineering, Ohio State University, Columbus, USA. His research interests include image processing, LiDAR data processing, vanish point detection, and edge detection.



Renping Xie received the B.E. degree from Shanxi University of Finance & Economics in June 2012. He is a successive postgraduate and doctoral program graduate student majoring in Photogrammetry and Remote Sensing in Wuhan University. He has published several international conference papers and journal papers, and is the inventor of several patents. His current research interests include SLAM (simultaneous localization and mapping), Robotics, Image Processing, etc.

Li Li received the B.E. degree and M.E. degree from the School of Remote Sensing and Information Engineering at Wuhan University, China in 2013 and 2016, respectively. He is pursuing a Ph.D. degree at School of Remote Sensing and Information Engineering, Wuhan University, China. He has published several international conference papers and journal ones, and he is the inventor of several patents. Currently he mainly works on Image Mosaicking, LiDAR Data Processing, Robotics, Machine Learning, etc.