COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Computer-Aided Civil and Infrastructure Engineering **32** (2017) 805–819

Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network

Allen Zhang & Kelvin C. P. Wang*

School of Civil Engineering, Southwest Jiaotong University, Chengdu, China and School of Civil and Environmental Engineering, Oklahoma State University, OK, USA

Baoxian Li, Enhui Yang, Xianxing Dai & Yi Peng

School of Civil Engineering, Southwest Jiaotong University, Chengdu, China

Yue Fei, Yang Liu, Joshua Q. Li & Cheng Chen

School of Civil and Environmental Engineering, Oklahoma State University, OK, USA

Abstract: *The CrackNet, an efficient architecture based on the Convolutional Neural Network (CNN), is proposed in this article for automated pavement crack detection on 3D asphalt surfaces with explicit objective of pixel-perfect accuracy. Unlike the commonly used CNN, CrackNet does not have any pooling layers which downsize the outputs of previous layers. CrackNet fundamentally ensures pixel-perfect accuracy using the newly developed technique of invariant image width and height through all layers. CrackNet consists of five layers and includes more than one million parameters that are trained in the learning process. The input data of the CrackNet are feature maps generated by the feature extractor using the proposed line filters with various orientations, widths, and lengths. The output of CrackNet is the set of predicted class scores for all pixels. The hidden layers of CrackNet are convolutional layers and fully connected layers. CrackNet is trained with 1,800 3D pavement images and is then demonstrated to be successful in detecting cracks under various conditions using another set of 200 3D pavement images. The experiment using the 200 testing 3D images showed that CrackNet can achieve high Precision (90.13%), Recall (87.63%) and F-measure (88.86%) simultaneously. Compared with recently developed crack detection methods based on traditional machine learning and imaging algorithms, the CrackNet significantly outperforms the traditional approaches in terms of F-measure. Using parallel computing techniques, CrackNet is programmed to be efficiently used in conjunction with the data collection software.*

*To whom correspondence should be addressed. E-mail: *kelvin.wang@okstate.edu*.

DOI: 10.1111/mice.12297

1 INTRODUCTION

The automation of pavement crack detection generally requires robust algorithms of high level of intelligence. However, over the past decades, the private and public endeavors on worldwide basis vastly underestimated the challenges and difficulties in developing fully automated analysis tools for pavement cracks. The underlying difficulty is directly related to human's limitation in developing mathematical models simulating cognition capability.

Thresholding algorithms were proposed to find cracks by setting global or local thresholds (Cheng et al., 2003; Oliveira and Correia, 2009), with various limitations for images with complex illuminance. Segmentation-based methods were proposed to conduct analyses at small blocks (Kirschke and Velinsky, 1992; Huang and Xu, 2006; Ying and Salari, 2010), but were incapable of representing crack width accurately as the detection is conducted at block level instead of pixel level. Edge detectors were widely used to detect the edges of pavement cracks (Attoh-Okine and Ayenu-Prah, 2008; Santhi et al., 2012; Nisanth and Mathew, 2014), however with incapability in detecting complete crack profiles. Filter-based algorithms were developed to find cracks of anticipated responses (Zhang et al., 2013; Zalama et al., 2014), with limitations in detecting cracks of weak

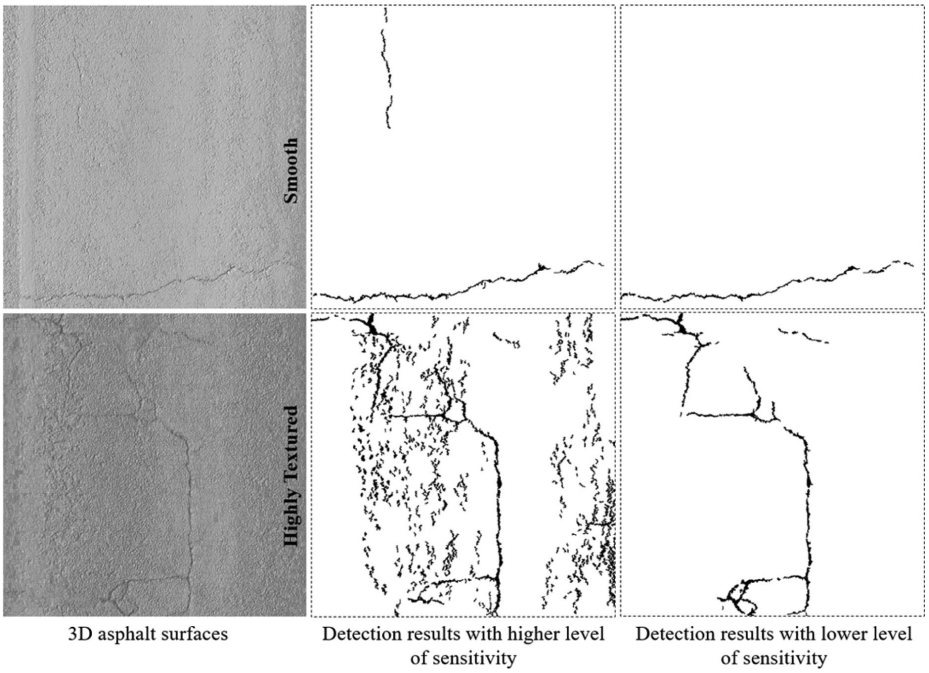

Fig. 1. Typical detection results on a smooth asphalt surface and a highly textured asphalt surface.

responses to predesigned filters. The *CrackTree* demonstrated a strong capability of detecting discontinued cracks (Zou et al., 2012), however without considering the actual crack width. Wavelet Transform was applied to decompose the original data into different frequency subbands. Consequently, cracks could be detected more easily following the assumption that cracks are primarily preserved in high frequency subbands (Zhou et al., 2006; Subirats et al., 2006; Wang et al., 2007). Nevertheless, the decomposition of original data into frequency domain adversely impacts the spatial entirety of pavement cracks, resulting in discontinued cracks.

The three-dimensional (3D) laser imaging technology has become the dominant approach to automated pavement data collection in recent years. Compared with two-dimensional (2D) pavement images, 3D pavement data are less vulnerable to lighting conditions and present more useful information as well as fewer noises in terms of crack detection (Wang, 2011; Zhang and Wang, 2017). Many studies were conducted to detect cracks specifically on 3D pavement surfaces. Depth-checking methods were proposed for crack detection using 3D pavement surface data (Jahanshahi et al., 2013; Ouyang and Xu, 2013), representing simple thresholding methodologies on the depth and length of a crack with limited successes particularly for complex and fine cracks. A modification of the dynamic optimization algorithm was implemented (Jiang and Tsai, 2016) to detect cracks on 3D pavement surfaces. Recently, the interactive cracking detection algorithm was proposed to detect cracks on 3D pavement surfaces with aid of the feedback from human operators (Zhang et al., 2016a). Hybrid procedures of matched filtering, tensor voting and minimum spanning tree were also developed for crack detection using 3D pavement data (Sollazzo et al., 2016). In addition, the 3D shadow modeling was proposed for detection of various descended patterns (i.e., cracks, joints, grooves, and potholes) on 3D pavement surface (Zhang et al., 2017). Although the recent crack detection algorithms demonstrate some successes, none of them have shown acceptable levels of precision and bias on consistent basis on a large set of diversified pavement data in various conditions.

The biggest challenge of automated crack detection is to consistently achieve high performance under various complex environments. A certain automated algorithm may yield detection results of satisfying accuracies on some particular roads, while resulting in completely unacceptable error rates on other roads. Such inconsistent performances may be frequently observed on asphalt surfaces where the textures and roughness levels are varying. Figure 1 shows detection results of the interactive cracking detection algorithm (Zhang et al., 2016a) on a smooth asphalt surface and a highly textured asphalt surface. It can be observed that the interactive cracking detection algorithm of high sensitivity yields many false-positive errors on the

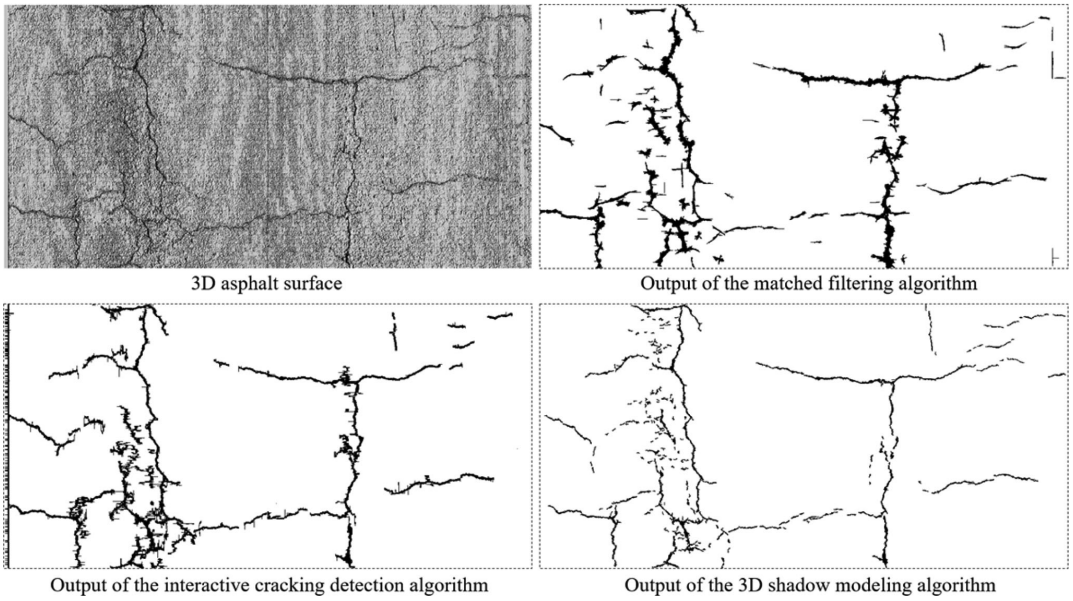

Fig. 2. Detection results of the matched filtering, interactive cracking detection, and 3D shadow modeling algorithms on a 3D pavement image.

highly textured surface, while successfully finding all cracks on the smooth surface without any errors. On the other hand, the interactive cracking detection algorithm of low sensitivity achieves high accuracy on the highly textured surface, but fails to detect the fine crack on the smooth surface. This example illustrates challenges of many conflicting performance scenarios for traditional automated algorithms, such as the matched filtering (Zhang et al., 2013) and the 3D shadow modeling (Zhang et al., 2017), which require substantial manual assistance to obtain consistent results.

The conflicting performances of traditional automated algorithms may also be observed on the same road, or even on the same pavement image. Figure 2 shows the detection results of the matched filtering (Zhang et al., 2013), interactive cracking detection (Zhang et al., 2016a) and 3D shadow modeling (Zhang et al., 2017) algorithms on a specific 3D pavement image. The sensitivity levels of the three algorithms are progressively tuned to detect the fine cracks located at the top right corner of the image. However, the three algorithms all result in false-positive errors particularly at the left side of the image when the fine cracks start to be recognized.

The indistinctive boundary between fine cracks and local noises/textures indeed has created many difficulties for traditional automated algorithms. Fundamentally, the diversity of cracks and textures on asphalt pavement surfaces would require any automated algorithms to be completely self-adaptive and fully tuned on the basis of exhaustive examples. However, most existing automated algorithms for pavement crack detection were developed based on specific and uncomprehensive hypotheses and lacked the capability of learning from examples. This could be one of the reasons why the current automated algorithms have tangible difficulties in detecting pavement cracks at consistently high levels of precision and bias for a pavement network in varying conditions and with different texture characteristics (Zhang et al., 2016a). However, it is also true that humans are good at recognizing pavement cracks without hesitation and can achieve consistent results. From such a perspective, advanced machine learning techniques that simulate human cognition potentially provide pavement engineers a truly automated tool for production purposes.

Many successful applications of machine learning techniques were reported in the field of transportation engineering, such as traffic incident detection (Adeli and Samant, 2000), work zone capacity estimation (Adeli and Jiang, 2003), traffic flow forecasting (Jiang and Adeli, 2005), and traffic sign classification (Ciresan et al., 2012). There were also pioneering applications of machine learning techniques, such as Artificial Neural Network (ANN) and Support Vector Machine (SVM), in classifying cracks on pavement surfaces (Kaseko and Ritchie, 1993; Kaseko et al., 1994; Lee and Lee, 2004; Gavilan et al., 2011; Nejad and Zakeri, 2011; Marques, 2012; Daniel and Preeja, 2014). However, these studies generally represent only one or two layers of abstraction and cannot fully reflect the complexity of pavement surface.

In recent years, Deep Learning has been found to provide an opportunity to learn from experiences and understand complex problems based on a hierarchy of concepts (Goodfellow et al., 2016), where the concepts are defined and learned through increasing levels of abstraction (Murphy, 2012).

Particularly, deep Convolutional Neural Networks (CNN) have demonstrated successes in large-scale object recognition problems (Krizhevsky et al., 2012; Zeiler and Fergus, 2013; Sermanet et al., 2014; Szegedy et al., 2014; Simonyan and Zisserman, 2015; He et al., 2015). The two most significant properties of CNNs are space invariance and distortion invariance due to the sharing weights and pooling layers (LeCun et al., 1998), which contribute to their great potentials for vision tasks. However, due to the existence of pooling layers, the original data are progressively downsized through increasing levels of abstraction, resulting in loss of the original information. Therefore, most convolutional networks were to classify small image patches and could not achieve pixel-perfect accuracies. Some convolutional networks intended to assign class label to an individual pixel by treating the local context around the pixel as a whole patch (Tschopp, 2015; Maji et al., 2016), and may yield imprecise predictions around the target pixels (i.e., the crack pixels) due to overlaps. Other convolutional networks were also proposed for pixelwise classifications (Pinheiro and Collobert, 2015; Shelhamer et al., 2016), which all include pooling layers, and thus inevitably lose original data. In the All Convolutional Net, the pooling layers were replaced by convolution layers with larger strides (Springenberg et al., 2015), resulting in similar spatial reductions and data losses.

Pixel-perfect accuracy is critical for pavement crack detection, meaning the visible geometric features of cracks can be available regarding the shape, orientation, length, and width. First, the pixel-perfect accuracy can yield accurate identifications on the types of detected cracks (e.g., longitudinal cracks, transverse cracks, or alligator cracks). Second, the actual width, length and extent of a crack are important indicators to identify its severity level. Last, the accurate measurements of pavement cracks are very useful for timely monitoring on the developing behaviors of pavement cracks. Thus, the automated crack survey without a high level of pixel-perfect accuracy would become less useful. It is impossible for a human being to visually measure geometric features of cracks without resorting to measurement tools. Although current convolutional networks accomplished excellent performances in numerous vision tasks, it is challenging to apply them in pavement crack detection when pixel-level accuracy is fully considered.

Zhang et al. (2016b) proposed a CNN with four convolution layers, four max-pooling layers and two fully connected layers for crack detection. Nevertheless, the class label assigned to an individual pixel was still based on the local context around the pixel, which resulted in overestimation of crack width. It was shown in their study that deep CNN outperformed traditional machine learning techniques (i.e., SVM and Boosting method) for pavement crack detection. The classic architecture of AlexNet (Krizhevsky et al., 2012) was adopted to classify an input image as crack image or no-crack image (Some, 2016), and thus was incapable of detecting cracks at pixel level. Cha et al. (2017) also developed a deep CNN for piecewise classification on cracks. With the use of sliding window techniques, the class label was assigned to each small patch of size 256×256. It was demonstrated that their method achieved a high level of accuracy in classifying a small image patch as cracked or intact even under complex illuminations. However, the actual location of cracks in the small patch was not considered in their study, and the pixel-perfect accuracy was thus unattainable. Although the recent CNN-based methodologies have limitations in terms of pixel-perfect accuracy, they all reveal the great potential of deep CNN in detecting pavement cracks.

In this article, an efficient network architecture based on CNN is proposed for pavement crack detection on 3D asphalt surfaces with full considerations on pixel-perfect accuracy. Compared with traditional CNNs, the proposed network does not have any pooling layers. The pixel-level accuracy is achieved by the following regularizations. First, the spatial size of the input data is invariant through all layers. Second, the ground-truth of training data is prepared for pixel-to-pixel supervised learning. Last, an individual pixel is compared with its neighbors through the local connections provided at the convolution layer, and the final class score for an individual pixel is predicted by integrating and analyzing the multichannel responses evaluated at that pixel. Several thousand training data sets were prepared and manually processed by the research team for machine learning purposes. A total of 1,800 observations are used in the presented work of CrackNet in training, and another 200 observations are used for testing.

2 METHODOLOGY

2.1 Data preparation

All pavement surface data used in the article are 1-mm 3D data from the *PaveVison3D* system mounted in a Digital Highway Data Vehicle (DHDV) made by WayLink. The DHDV can scan the pavement surface

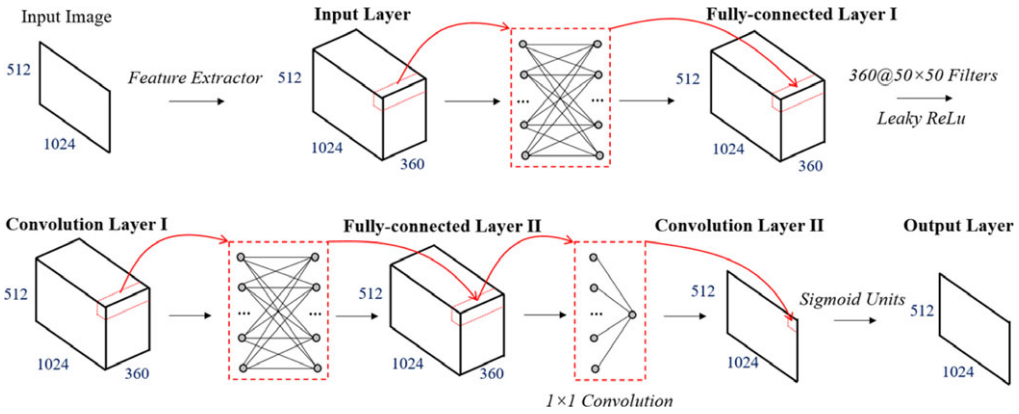

Fig. 3. Architecture of CrackNet.

at the data collection speed of 100 KPH (approximately 60 MPH) with full coverage for a 4 m wide lane at a 30 KHz scanning rate to ensure 1 mm resolution at the highway speed. This article uses the depth information contained in 3D data to train CrackNet. The original 3D image collected by the *PaveVision3D* system has a fixed size 4,096 × 2,048, representing approximately a 4 m wide and 2 m long pavement surface. To reduce computational overhead, the original 3D image is downsized to a 1,024 × 512 3D image by min-pooling techniques. In other words, the minimal value of every 4 × 4 image block is maintained on the downsized 3D image. Although the down-sampling procedure will lead to data loss, the min-pooling method increases the probability in maintaining cracks, given that the crack pixel has a lower elevation compared to the local surroundings.

2.2 Architecture

As illustrated in Figure 3, the proposed CrackNet is constituted by five layers: two fully connected layers, one convolution layer, one 1 × 1 convolution layer, and one output layer. The input of the proposed CrackNet are 360 feature maps generated by the feature extractor. Each feature map has the same size with the input image. The fully connected layers I and II provide full connections to previous layers following a pixel-independent manner. Thus, the connections are established across all channels at each pixel, and no two pixels are connected at each channel. Such connections can force the network to combine responses from all channels at each pixel. To emphasize space invariance, the weights for the established connections are shared at all pixels. In other words, the cross-channel connections established at each pixel use the same weights. From some perspective, the fully connected layers I and II can also be regarded as 1 × 1 convolution layers without dimensionality reduction.

Table 1
Number of parameters in CrackNet

Layer	*Number of parameters*
Fully connected layer I	129,600
Convolution layer I	900,000
Fully connected layer II	129,600
Convolution layer II	361 (including the bias)
Total	1,159,561

At convolution layer I, each channel of the output of the previous layer is convolved with an individual filter of size 50 × 50. The number of filters at convolution layer I therefore is 360. Following the convolutions, the Leaky Rectified Linear Unit (Leaky ReLu) is adopted to perform nonlinear transforms (Maas et al., 2013). The Leaky ReLu used in the article has a different leaky coefficient and is expressed as

$$h = \begin{cases} \boldsymbol{w}^T \cdot x, & \boldsymbol{w}^T \cdot x > 0 \\ 0.1\boldsymbol{w}^T \cdot x, & \text{else} \end{cases} \tag{1}$$

where h is the output; $\boldsymbol{w}^T$ is the weights; and $\boldsymbol{x}$ is the input.

The 1 × 1 convolution layer conducts cross-channel parametric pooling and integrating multiple channels into a single channel. Similarly, shared weights are also used in this layer for space invariance. In other words, identical weights are applied at each pixel to avoid spatial differences. Finally, the output layer uses a sigmoid unit to convert the output of convolution layer II to class scores whose values are between 0 and 1. Thus, the final output image contains predicted class scores for all individual pixels. In this article, class scores no smaller than 0.5 signify that the corresponding pixels are crack pixels, whereas class scores smaller than 0.5 indicate the

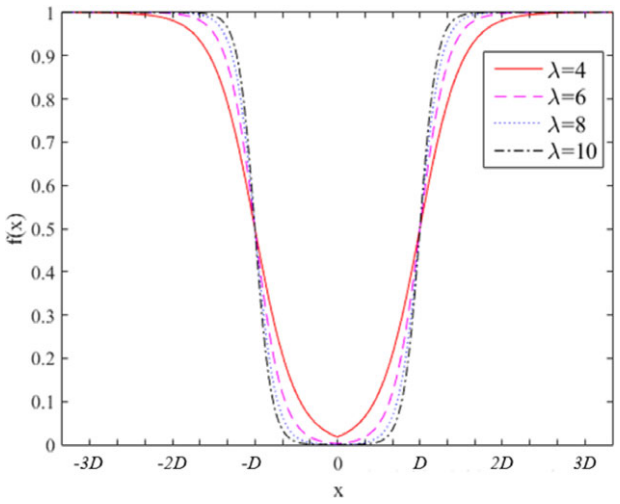

Fig. 4. Illustration of symmetric sigmoid curves with different values of parameter λ.

corresponding pixels are noncrack pixels. Table 1 shows that CrackNet has more than 1 million parameters to be learned.

The CrackNet intends to detect cracks with explicit requirements on pixel-level accuracy. First, the image height and width are invariant through all layers such that the error at each pixel can be learned efficiently following an end-to-end manner. Second, the feature extractor and convolution layer I provide local connections such that the relationships between crack pixels and their local surroundings can be learned by the CrackNet. Finally, the convolution layer II, fully connected layers I and II are deployed primarily for learning the complex difference between a crack pixel and a noncrack pixel based on the multichannel responses obtained at the same location.

2.3 Feature extractor

In this article, the feature extractor serves to generate feature maps and prepare the input of CrackNet. The feature extractor utilizes line filters oriented at various directions and with varied lengths as well as widths to enhance the contrast between cracks and the background. It can be considered as a feature detection layer with fixed operations and without learnable parameters. Each feature map, or equivalently each channel of the input of CrackNet is a filtered image processed by an individual line filter. In particular, there are 360 line filters used in feature extraction to generate 360 feature maps. The profile (cross section) of each line filter is a symmetric sigmoid curve developed in the article as

$$f\ (x) = 1 - 1/\left(1 + e^{\frac{-\lambda(D-|x|)}{D}}\right) \qquad (2)$$

Table 2
Sizes and rotation angles of line filters used by the feature extractor

Filter no.	*Filter width* s_x	*Filter length* s_y	*Rotation angle* θ
1-36	3	5	0°, 5°, 10°, 15°, . . . , 175°
37-72	5	5	0°, 5°, 10°, 15°, . . . , 175°
73-108	7	5	0°, 5°, 10°, 15°, . . . , 175°
109-144	9	5	0°, 5°, 10°, 15°, . . . , 175°
145-180	11	5	0°, 5°, 10°, 15°, . . . , 175°
181-216	3	10	0°, 5°, 10°, 15°, . . . , 175°
217-252	5	10	0°, 5°, 10°, 15°, . . . , 175°
253-288	7	10	0°, 5°, 10°, 15°, . . . , 175°
289-324	9	10	0°, 5°, 10°, 15°, . . . , 175°
325-360	11	10	0°, 5°, 10°, 15°, . . . , 175°

where x is the distance from the profile center; $f(x)$ is the function value at x; λ is the parameter to control the flatness at the center area and the steepness near the center area; and D is the distance from the profile center where the filter value equals to 0.5.

In Figure 4, the transition from edge area to the center area becomes more dramatic when parameter λ increases. In terms of crack detection, the symmetric sigmoid curve can be used to mimic the sharp changes of elevations between cracks and the background. The response of a crack to the filter can be distinctive if the profile of the crack is matched with the filter profile. The parameter λ is fixed as 6 in this article. Note that λ is not a sensitive parameter and contributes to small differences when it is greater than 4. In addition, the width of the profile is fixed as $4 \times D$ because the symmetric sigmoid curve becomes flat when the distance from center is greater than $2 \times D$.

Based on the defined profile, a single line filter is formulated by a collection of identical profiles along y axis. In other words, the profile defined in (2) is replicated along y axis. To yield nearly zero responses under noises, the line filter is shifted to have a zero mean. Table 2 shows the sizes and rotation angles of the 360 line filters used by the feature extractor. Five different widths are assigned to the 360 line filters to consider cracks of varying widths. In addition, two small lengths are used by the 360 line filters for matching with a crack at local regions. Finally, the 360 line filters are aligned at various orientations with a small fixed angle interval such that a crack fragment of arbitrary direction could be entirely or partially matched with one of the line filters. Figure 5 illustrates some representative line filters following the specifications in Table 2.

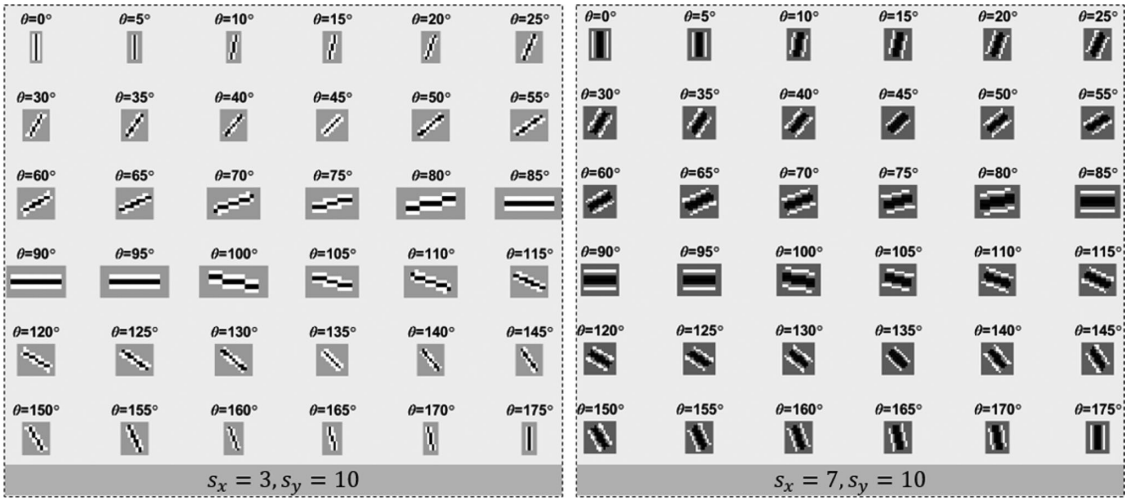

Fig. 5. Representative line filters used by the feature extractor.

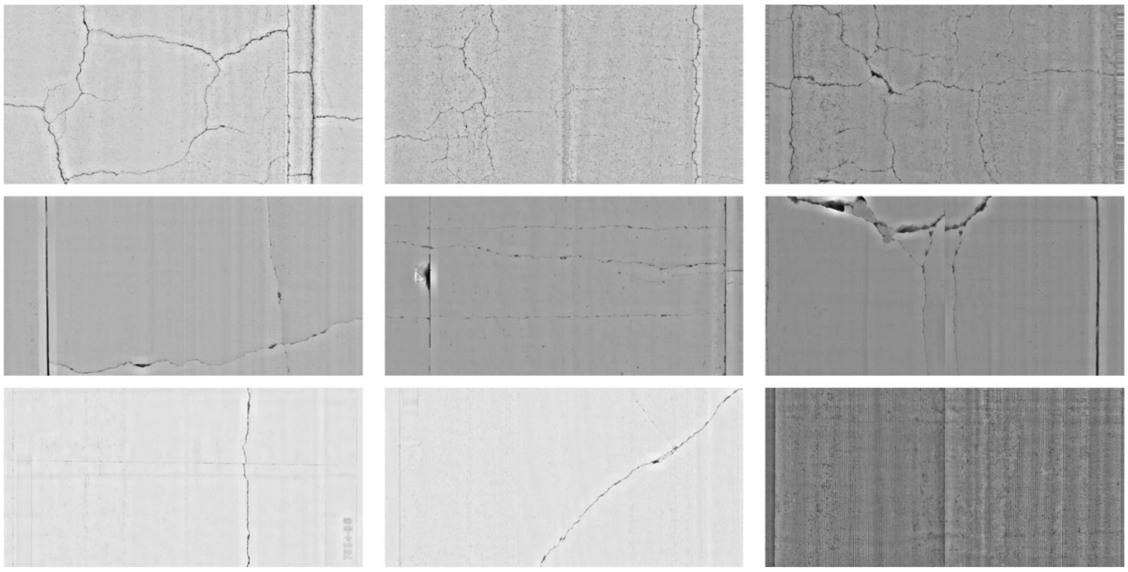

Fig. 6. Representative 3D pavement images from the image library.

3 IMAGE LIBRARY FOR TRAINING AND TESTING

An image library is established to feed CrackNet with diverse examples for supervised learning. The image library has more than 5,000 3D pavement images at 1 mm resolution representing diversified variations of cracks and pavement surface textures. All 3D pavement images in the library are collected in the last 5 years, and at various locations as well as different collection speeds ranging from 20 to 60 MPH. There is no overlap between any two images, and no more than 100 images are from the same road. All types of cracks with various severity levels defined in the LTPP protocol (Miller and Bellinger, 2014) can be found in the image library. The ground-truths of cracks on all images are manually marked with close supervision on pixel-perfect accuracies by multiple teams. A three-round inspection is conducted to ensure the ground-truths are accurate at pixel level. For the first round, several well-trained operators manually mark the cracks on provided 3D pavement images with full resolution. For the second round, several other well-trained operators examine and refine the ground-truths for correcting errors and reducing subjectivity. Finally, the ground-truths are further inspected and verified by experts. The entire process of preparing ground-truths of cracks is completed on continuing basis for nearly 1 year. Figure 6 illustrates several

typical 3D pavement images from the established image library.

Two thousand asphalt surface images are randomly selected from the image library for training and testing of CrackNet. The 2,000 asphalt surface images represent various textures and different mix types, including Hot Mix Asphalt (HMA) and Warm Mix Asphalt (WMA). In particular, 1,800 images are used as training data, and the other 200 images are considered as testing data.

4 TRAINING

4.1 Cost function

The sigmoid units at the output layer produce predicted values between 0 and 1 at all individual pixels. In this article, the target values for a background pixel and a crack pixel are set as 0 and 1, respectively. Given the ground-truth of cracks, the target values for all pixels of an image can be determined directly. Subsequently, Cross Entropy is employed as the cost function to measure the similarity between the predicted values and target values (Goodfellow et al., 2016; Nielsen, 2017):

$$C = \sum_{i=1}^{n} [y_i \ln a_i + (1 - y_i) \ln (1 - a_i)] \tag{3}$$

where y_i is the target value at ith pixel; a_i is the predicted value at ith pixel; and n is the number of pixels of the image.

Cross Entropy is used as the cost function for two reasons. First, it improves the learning speed. Second, it drives the network to learn at a rate controlled by the similarity between the predicted values and the target values. In other words, the network can learn faster when the errors are larger.

4.2 Learning method

The objective of training is to minimize the cost function using gradient-based optimization. Mini-batch Gradient Descent, a variation of Stochastic Gradient Descent, is implemented here to update parameters at each iteration. In addition, Backpropagation with the use of Momentum is applied to compute gradients. Thus, the learning method for each iteration is

$$\begin{cases} \Delta \boldsymbol{W}_{i+1} = 0.9 \cdot \Delta \boldsymbol{W}_i - \varepsilon \cdot \left(\frac{\partial C}{\partial \boldsymbol{w}} | \boldsymbol{w}_i\right)_{\boldsymbol{B}_i} \\ \boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \Delta \boldsymbol{W}_{i+1} \end{cases} \tag{4}$$

where i is the iteration index; $\boldsymbol{w}_i$ is the weights learned at ith iteration; $\Delta \boldsymbol{W}_i$ is the momentum variable at ith iteration; ε is the learning rate; $\boldsymbol{B}_i$ represents the ith mini-batch; and $(\frac{\partial C}{\partial \boldsymbol{w}} | \boldsymbol{w}_i)_{\boldsymbol{B}_i}$ is the average gradient over $\boldsymbol{B}_i$ evaluated at $\boldsymbol{w}_i$.

In this article, the size of mini-batch and the learning rate are manually adjusted through the iterations. The mini-batch size for an iteration ranges from 10 to 30, whereas the learning rate for an iteration is between 0.001 and 0.05. In particular, smaller batch sizes and larger learning rates are preferred in the beginning for faster convergence. Larger batch sizes and smaller learning rates are used subsequently for fine-tuning.

4.3 Normalized initialization and dropout

Due to large number of parameters, the initialization of parameters becomes important. A good strategy for parameter initialization is to keep similar variances of activation values and backpropagated gradients across all layers. In the article, the Normalized Initialization (Glorot and Bengio, 2010) is adopted to initialize parameters such that the variances of activation values and backpropagated gradients can be maintained efficiently.

In addition, another efficient technique called "Dropout" is implemented in the article to reduce the risk of overfitting (Hinton et al., 2012). The Dropout technique randomly omits each hidden neuron with a probability of 0.5. The omitted hidden neurons have zero output values, and thus will not be activated in both forward and backward passes. The Dropout technique prevents complex co-adaptions of neurons and forces neurons to be more independent instead of relying on other neurons.

4.4 Parallel computing

Except the output layer, all other layers of CrackNet require intensive computations on 360 images of size 1,024 × 512 for both forward and backward passes. Therefore, parallel computing techniques is applied to improve the computational efficiency of CrackNet such that the training of CrackNet on the basis of 1,800 example images can become manageable. There have been pioneering applications of parallel computing techniques in large-scale engineering computations (Adeli and Kamal, 1989, 1992a, b; Adeli and Kumar, 1995; Schrefler et al., 2000; Hsieh et al., 2002; Torbol, 2014; Ponz-Tienda et al., 2016; Wu et al., 2016). All these studies demonstrated substantial gains of time efficiency through well-designed parallel processing. The fundamental challenge of parallel computing is to reformulate the problem and develop parallel algorithms such that the parallel machines can be fully utilized (Adeli and Vishnubhotla, 1987). As shown in Figure 3, the structure of CrackNet is compatible with parallel computing using the Graphics Processing Unit (GPU). For the forward pass to produce the output, the computational tasks at all elements or pixels of the output

data at each layer can be operated simultaneously. With respect to the backward pass to compute gradients, the computational tasks at all elements of the input data at each layer can also be executed concurrently. Given that the summation of partial gradients is executed at all elements simultaneously, race condition problems will occur when multiple GPU threads try to write at the same memory address concurrently. To avoid race condition problems, the *atomic add* function, a built-in function of the *CUDA* programming platform (NVidia, 2017), can be used to sum the partial gradients at each element without interference from other elements. In general, for both forward and backward passes, the computation tasks at each element of each layer in CrackNet can be considered as completely independent with the use of *atomic add* function. Therefore, the forward and backward passes are both implemented in a massively parallel manner using a single or multiple GPU devices for up to two to three orders of computational improvements versus CPU based serial code implementation of the same algorithms. The CrackNet is programmed under C++ environment and with the use of CUDA C platform, but without using the NVidia cuDNN library. Using 2 GPU devices (2 NVidia GeForce GTX TITAN Black), the average processing time of the forward and backward passes for a single image of size 1,024 $\times$ 512 are 5.37 and 18.51 seconds, respectively. CrackNet may require more computations than traditional deep CNNs, such as the networks proposed in Zhang et al. (2016b) and Cha et al. (2017). First, the input data of traditional deep CNNs are normally small image patches, and the data depths at hidden layers may not exceed 360. Second, the data size in traditional deep CNNs is continuously reduced through pooling layers or convolutional layers with stride greater than 1. Last, CrackNet provides full connections across all channels at each pixel, resulting in intensive computations. However, it will be demonstrated in the next section that the fully connected layers improve the accuracy of CrackNet. In addition, the time cost due to the use of invariant spatial size through all layers is also necessary, as it is beneficial for learning errors at pixel level.

4.5 Training result

Fully connected layers I and II establish full connections across all channels at each individual pixel, which might not be necessary. To justify the use of fully connected layers, a shallower network without fully connected layers I and II is trained for a comparison with CrackNet. This shallower network, denoted as CrackNet-1, shares the same architecture with CrackNet but only consists of three layers by excluding the two fully connected layers. To avoid overfitting problem, the 1,800 training images are divided into two subsets. First, 300 images are randomly selected from the 1,800 images and serve as validation data which do not participate in learning errors and tuning parameters. Second, the other 1,500 images are all involved in the learning process. For time efficiency, the performances of the two networks on 300 validation images are evaluated at every 50 iterations instead of each iteration to inspect if overfitting occurs. In the meantime, the learned parameters are saved at every 50 iterations for final selection of optimal parameters that yield the best performances on the 300 validation images. The timely performances of the two networks on validation data also provide a guide to manually tune hyper parameters, including learning rate and mini-batch size.

Precision and Recall are two commonly used indicators for evaluating crack detecting algorithms (Fawcett, 2006; Zhang et al., 2016a, 2017). Precision refers to the percentage of crack pixels classified correctly with respect to all detected pixels, whereas Recall represents the percentage of crack pixels classified correctly with respect to all true crack pixels. Precision and Recall frequently conflict with each other. For instance, a high level of sensitivity may yield high Recall but low Precision. On the other hand, a low level of sensitivity could result in high Precision but low Recall. Therefore, it is challenging to achieve high Precision and high Recall simultaneously. F-measure is the harmonic mean of Precision and Recall (Fawcett, 2006), reflecting the accuracy of an algorithm more appropriately. A high F-measure can be achieved only when the Precision and Recall are both high.

The training is completed after 700 iterations, which takes roughly 9 days on two NVidia GeForce GTX TITAN Black cards in the same computer. Figure 7 shows the cost function values for the 700 iterations and the overall F-measures on validation data elevated at every 50 iterations. It is clearly demonstrated in Figure 7 that CrackNet yields better performance than CrackNet-1 in terms of both cost function and overall F-measure, indicating the importance of fully connected layers. After 500 iterations, the cost function values for the two networks are significantly reduced and only oscillate in a small range, implying that the two networks can finally yield similar outputs for most training examples. In addition, the overall F-measure on validation data is improved progressively, indicating that the model does not result in overfitting problems. Particularly, the highest overall F-measure 89.54% for CrackNet is observed at the 650th iteration. Therefore, the parameters saved at the 650th iteration are considered as optimal. Using the optimal parameters, the overall Precision, Recall, and

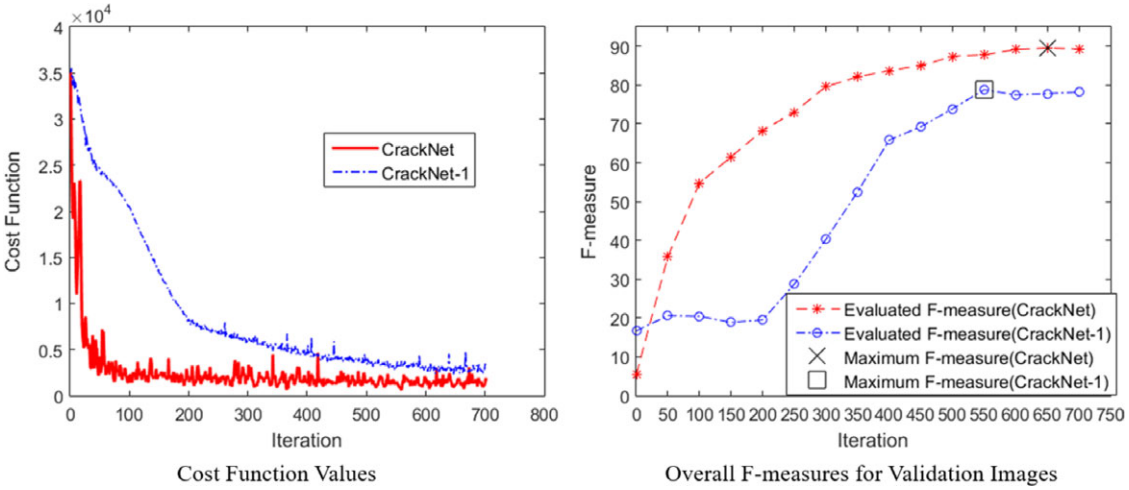

Fig. 7. Illustration of training progress.

F-measure achieved by CrackNet for all 1,800 training images are 89.15%, 87.16%, and 88.14%, respectively.

5 TESTING AND EVALUATION

After the training, CrackNet is applied to the 200 testing images for further validations. Figure 8 illustrates several typical testing images correctly classified by CrackNet. As shown in Figure 8, the CrackNet can detect cracks with varying widths, severity levels and contexts. Compared with the ground-truth of cracks, the CrackNet also achieves a high level of pixel-perfect accuracy. To further evaluate the CrackNet, a recent crack detection method, 3D shadow modeling (Zhang et al., 2017) is implemented for a comparison study. The projection angle (61 degree) used in 3D shadow modeling is optimized for highest overall F-measure on all 200 testing images. In addition, a SVM-based method is also evaluated in the article to compare CrackNet with traditional machine learning approach. To conduct pixel-level classification using the SVM-based method, a sliding window of size 15 × 15 is centered at each pixel and continuously scans the local context around each pixel. The same local features as proposed in (Marques, 2012), including minimum value, mean, variance, third-, fourth-, and fifth-order moments, are extracted at each pixel. With extracted features, the SVM classifier using Radial Basis Function (RBF) is trained via the *LIBSVM* program (Chang and Lin, 2011). During the training, fivefold Cross Validation and Grid Search are implemented for faster search and lower risk of overfitting (Hsu et al., 2010). For simplicity, the SVM-based method is called Pixel-SVM in this article.

Table 3
Overall Precision, Recall, and F-measure of CrackNet, 3D shadow modeling, and Pixel-SVM

Method	*Precision (%)*	*Recall (%)*	*F-measure (%)*
Pixel-SVM	51.28	82.90	63.36
3D shadow modeling	66.84	77.19	71.64
CrackNet	90.13	87.63	88.86

The Precisions, Recalls, and F-measures of CrackNet, 3D shadow modeling, and Pixel-SVM for all 200 testing images are shown in Figure 9. Table 3 lists the overall Precisions, Recalls, and F-measures achieved by the CrackNet, 3D shadow modeling, and Pixel-SVM. According to Figure 9 and Table 3, the Pixel-SVM produces low Precisions but high Recalls. The 3D shadow modeling yields Precisions and Recalls of similar levels. However, CrackNet can achieve higher Precisions and Recalls concurrently. The performances of CrackNet on training and testing images are found to be similar, further indicating the overfitting problem is avoided. In terms of the overall F-measure, 3D shadow modeling is slightly better than Pixel-SVM. However, CrackNet significantly outperforms both Pixel-SVM and 3D shadow modeling according to the overall F-measure. Particularly, the overall Precision for CrackNet is roughly 90%, implying its noticeable accuracy at pixel level.

Figure 10 shows the performances of Pixel-SVM, 3D shadow modeling, and CrackNet on several representative testing images. It is found that Pixel-SVM results in serious overestimations on the crack width due to substantial overlaps. Such overestimations explain why the Precision for Pixel-SVM is comparatively low.

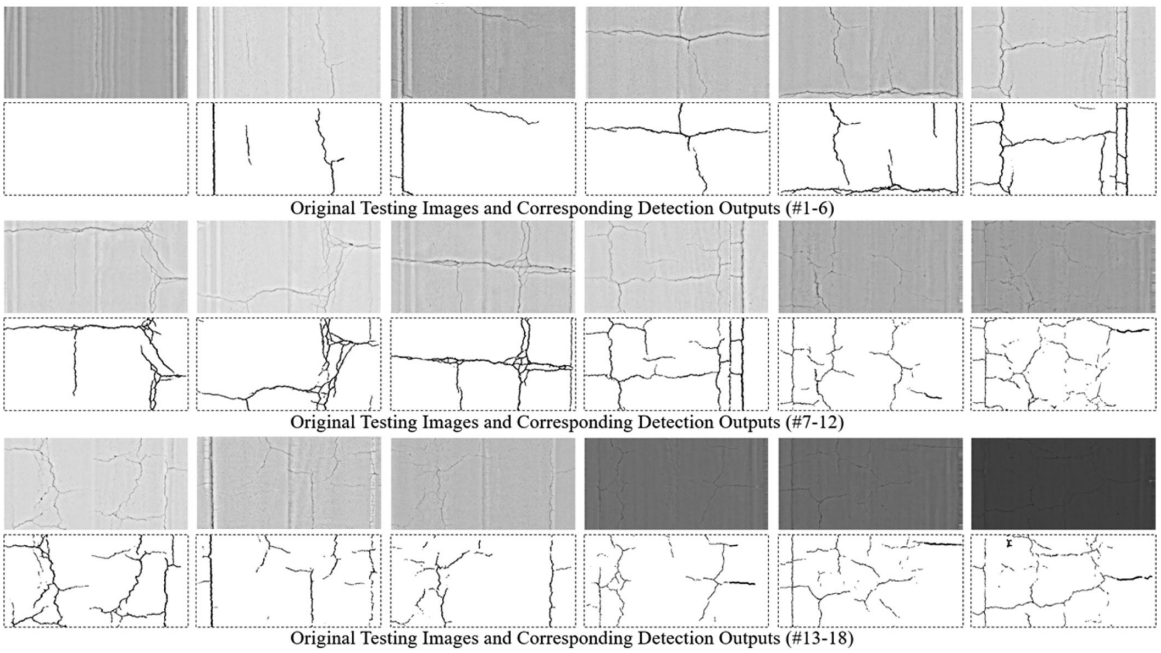

Fig. 8. Typical testing images correctly classified by the CrackNet.

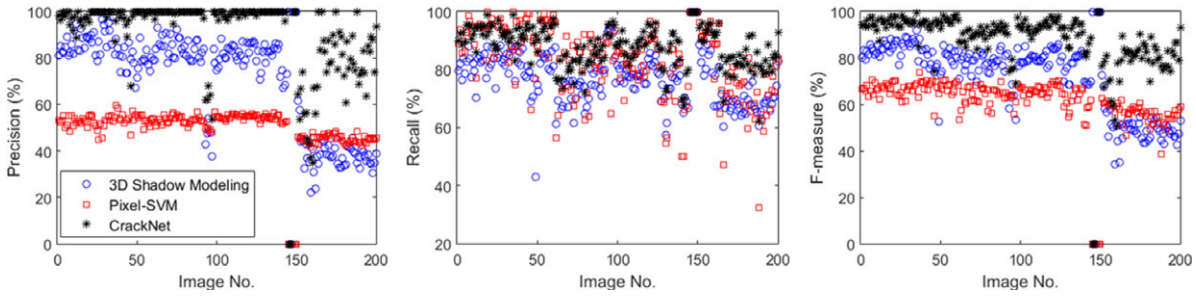

Fig. 9. Illustration of Precision, Recall, and F-measure of CrackNet, 3D shadow modeling, and Pixel-SVM.

The 3D shadow modeling can lead to a higher level of pixel-perfect accuracy, but is sensitive to local noises. Compared with both Pixel-SVM and 3D shadow modeling, CrackNet is more robust in suppressing noises and detecting fine cracks. The efficiency of CrackNet is highlighted that automated algorithms based on deep-learning techniques have better potential compared to traditional algorithms with shallow level of abstraction and limited learning capability.

6 DISCUSSION

Figure 11 shows several testing images with typical errors resulted from the CrackNet. The false-positive errors are highlighted in the dashed rectangles, whereas the false-negative errors are indicated in the dashed circles. The false-negative errors universally occur at local regions of very fine cracks or hairline cracks. There are several possible reasons for missing hairline cracks. First, CrackNet may be trained to be conservative on hairline cracks such that complex textures on rough surfaces (e.g., open-graded asphalt surfaces) will be prevented from being misclassified. As shown in Table 3, the overall Recall is slightly lower than the overall Precision, implying the conservativeness of CrackNet. Second, the hairline cracks are degraded during down-sampling. Last, the feature extractor does not grasp hairline cracks sufficiently. As an image block with hairline cracks can have multiple crack pixels as a whole, it is highly possible that the differences between hairline cracks and the background become more distinctive when they are analyzed at block level. Therefore, for pixel-level detection, CrackNet potentially has more difficulties in finding hairline cracks successfully. To reduce false-negative errors resulted from CrackNet, one

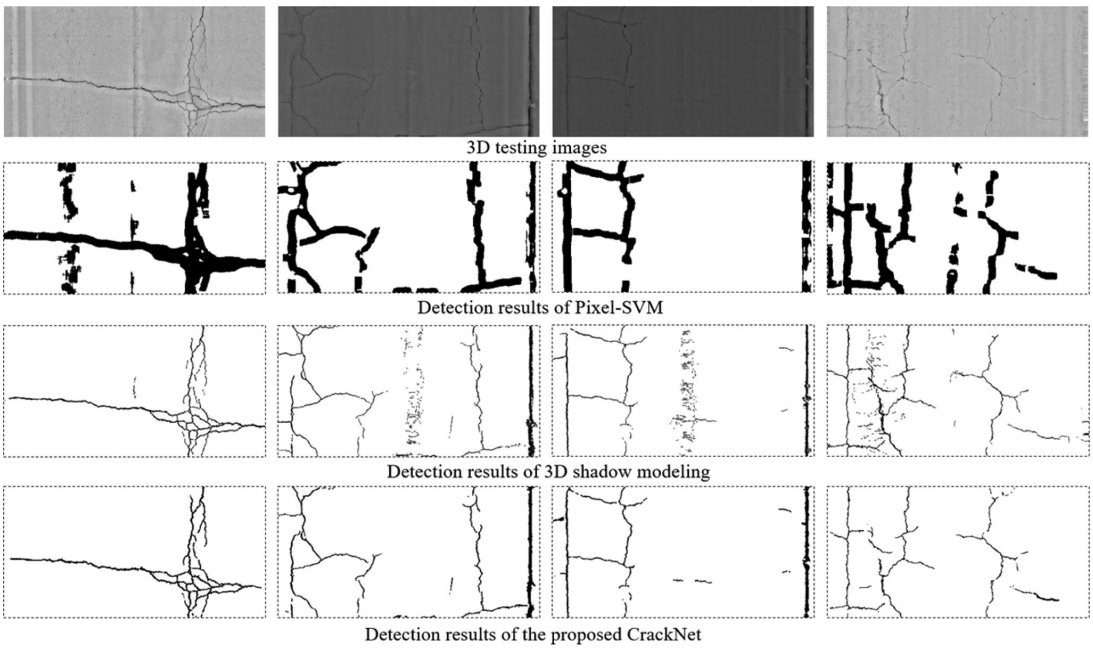

Fig. 10. Typical comparison between Pixel-SVM, 3D shadow modeling, and CrackNet.

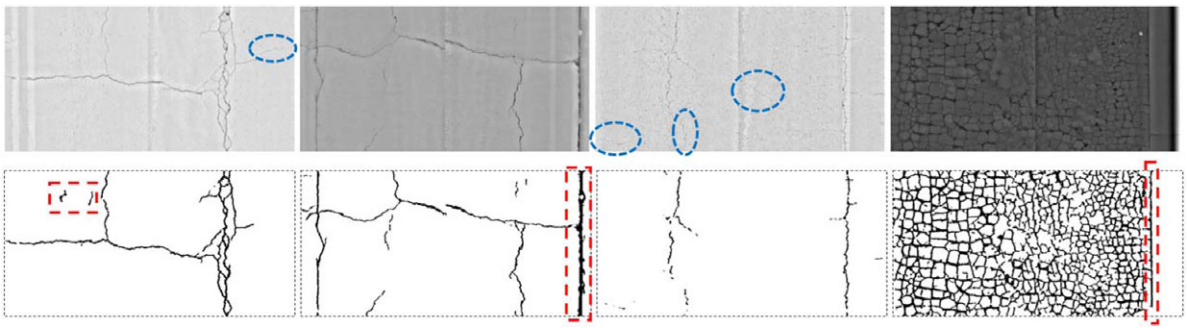

Fig. 11. Representative errors of CrackNet on testing images.

of the possible solutions is to increase the resolution of the input data such that the continuity of fine cracks is improved. The second possible resolution is to increase the number of line filters used by the feature extractor, and align the line filters at more possible orientations. Alternatively, other preprocessing techniques can be applied to further enhance the contrast between fine cracks and the background.

On the other hand, false-positive errors generated by CrackNet can be roughly divided into two types. The first type of false-positive errors, such as those marked on the first image in Figure 11, is caused by local noises of small extent. Such errors can be eliminated using traditional postprocessing methods, such as length filtering that removes patterns whose lengths are smaller than certain thresholds. Generally speaking, errors associated with CrackNet are small in number and consistent in variety compared to those associated with traditional methods.

In Figure 11, the second type of false-positive errors mainly results from shoulder drop-off (highlighted on the second image), pavement edge (highlighted on the last image), or any other patterns that appear to be similar to cracks. This type of errors is challenging to be eliminated due to complex similarities with pavement cracks. However, if the receptive field at convolution layers can be increased, there may be higher probability of eliminating this type of false-positive errors. In addition, increasing the receptive field may also lead to lower risks in generating the first type of false-positive errors.

The feature extractor conducts fixed operations to prepare the input of CrackNet and includes no learnable parameters. Although it reduces parameters and

training difficulty, it could be revised as a learnable layer to enhance the learning capability of CrackNet. In addition, CrackNet also has limitations for concrete pavement surfaces. However, the current version of CrackNet demonstrates a successful approach to applying learning techniques for pavement crack detection on asphalt surfaces with explicit requirements on pixel-perfect accuracy.

The training and testing images used in the article are 3D surface data. However, the architecture of CrackNet may also be suitable for other types of data, such as 2D pavement images. First, the feature extractor is a general-purpose procedure to enhance contrast between the crack and the background, given that the crack pixel has a lower intensity or elevation value. Second, all subsequent layers of CrackNet also implement general-purpose operations without constraints on data types. The only matter is to adjust the filter sizes according to the size of input data. If appropriate data are used for training, it is likely that CrackNet still can yield similar performance for other types of data.

7 CONCLUSIONS

In this article, an efficient network architecture based on Convolutional Neural Network (CNN) christened as CrackNet is described for the automated detection of pavement cracks on asphalt surfaces. Different from traditional CNNs, CrackNet does not have any pooling layers that downsize the outputs of previous layers. Regardless of the data depth, the data width and height are invariant through all layers to achieve pixel-perfect accuracy. The input of CrackNet are feature maps generated by the feature extractor using proposed line filters. The output of CrackNet are the predicted class scores for all individual pixels. CrackNet uses more than one million parameters and consists of one general convolution layer, one 1×1 convolution layer, two fully connected layers and one output layer.

Using 1,800 image data sets of asphalt surfaces randomly selected from the 3D image library established by the research team, CrackNet is trained on two GPU devices recursively with 700 iterations. The training of CrackNet is successfully completed by the use of various efficient learning techniques, including Mini-batch Gradient Descent, Momentum, Cross Entropy, Normalized Initialization, and Dropout. Then 200 testing images from the image library are processed by the trained CrackNet. The overall Precision, Recall, and F-measure of CrackNet on the 200 testing images are 90.13%, 87.63%, and 88.86%, respectively. It is demonstrated in the comparison study that the CrackNet significantly outperforms 3D shadow modeling which is a competitive crack detection algorithm without learning capability, and Pixel-SVM, which is based on traditional machine learning algorithms. The article reveals that applications of deep-learning techniques may provide much better solutions than traditional crack detection algorithms with shallow level of abstraction and limited learning capability.

CrackNet in its current version requires substantial processing time and potentially has more difficulties in detecting hairline cracks successfully. However, the distinctive advantage of CrackNet is that it detects cracks at pixel level instead of block level. Pixel-perfect accuracy desired in automated pavement survey can thus be enhanced through CrackNet. The image library for training and testing purposes includes more than 5,000 3D pavement images collected from diverse pavement sections. Manual preparation of the ground-truths for thousands of images with explicit requirements on pixel-perfect accuracy demands a huge amount of labor and time. However, if the complexity and diversity of pavement surface are fully considered, it is worthwhile to use a great number of labeled examples for comprehensive learning and low risk of overfitting. The feature extractor used in the article can be considered as a feature detection layer with fixed operations and without learnable parameters. Such fixed operations may not be robust enough to extract hairline cracks or suppress complex noises desirably. However, according to current performance, the CrackNet is still highly efficient even with a fixed feature extraction layer. For future developments, the feature extractor can be redesigned as a learnable layer for enhanced learning capability.

Improvements of CrackNet are continuing at rapid pace by the authors of this article, including identifying joints on concrete pavements, grooves, and shoulder drop-offs. A larger volume of test images in varying conditions is to be used to further verify the performance of CrackNet and make network refinements. Continuing advances of GPU have allowed the training of CrackNet to be conducted in an efficient manner. It is anticipated that future production-level software solutions to the problem of automated crack survey will be based on deep-learning techniques such as CrackNet due to their robust performance in terms of precision and bias, and their consistency across different pavement conditions.

ACKNOWLEDGMENTS

The authors wish to thank Michael Ohara, Ruxin Yan, Te Pei, Zhixing Ma, Xiaoli Xu, Guolong Wang, and Shihai Ding for their help in preparing the image library for training and testing CrackNet. Partial financial support

by the Federal Aviation Administration Grant 13-G-013 was appreciated for the presented work. In the past few years, FHWA provided various project and technical support to the OSU research team.

REFERENCES

Adeli, H. & Jiang, X. (2003), Neuro-fuzzy logic model for freeway work zone capacity estimation, *Journal of Transportation Engineering*, **129**(5), 484–93.

Adeli, H. & Kamal, O. (1989), Parallel structural analysis using threads, *Computer-Aided Civil and Infrastructure Engineering*, **4**(2), 133–47.

Adeli, H. & Kamal, O. (1992a), Concurrent analysis of large structures—I. Algorithms, *Computers and Structures*, **42**(3), 413–24.

Adeli, H. & Kamal, O. (1992b), Concurrent analysis of large structures—II. Applications, *Computers and Structures*, **42**(3), 425–32.

Adeli, H. & Kumar, S. (1995), Concurrent structural optimization on massively parallel supercomputer, *Journal of Structural Engineering*, **121**(11), 1588–97.

Adeli, H. & Samant, A. (2000), An adaptive conjugate gradient neural network-wavelet model for traffic incident detection, *Computer-Aided Civil and Infrastructure Engineering*, **15**(4), 251–60.

Adeli, H. & Vishnubhotla, P. (1987), Parallel processing, *Computer-Aided Civil and Infrastructure Engineering*, **2**(3), 257–69.

Attoh-Okine, N. & Ayenu-Prah, A. (2008), Evaluating pavement cracks with bidimensional empirical mode decomposition, *EURASIP Journal on Advances in Signal Processing*, v2008, 1–7.

Cha, Y. J., Choi, W. & Buyukozturk, O. (2017), Deep learning-based crack damage detection using convolutional neural networks, *Computer-Aided Civil and Infrastructure Engineering*, **32**(5), 361–78.

Chang, C. C. & Lin, C. J. (2011), LIBSVM: a library for support vector machines, *ACM Transactions on Intelligent Systems and Technology*, **2**(3), 1–27.

Cheng, H.D., Shi, J. & Glazier, C. (2003), Real-time image thresholding based on sample space reduction and interpolation approach, *Journal of Computing in Civil Engineering*, **17**(4), 264–72.

Ciresan, D., Meier, U., Masci, J. & Schmidhuber, J. (2012), Multi-column deep neural network for traffic sign classification, *Neural Networks*, **32**, 333–38.

Daniel, A. & Preeja, V. (2014), A novel technique for automatic road distress detection and analysis, *International Journal of Computer Applications*, **101**(10), 18–23.

Fawcett, T. (2006), An introduction to ROC analysis, *Pattern Recognition Letters*, **27**(8), 861–74.

Gavilan, M., Balcones, D., Marcos, O., Llorca, D. F., Sotelo, M. A., Parra, I., Ocana, M., Aliseda, P., Yarza, P. & Amirola, A. (2011), Adaptive road crack detection system by pavement classification, *Sensors Journal*, **11**(10), 9628–57.

Glorot, X. & Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *Journal of Machine Learning Research*, **v9**, 249–56.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), Deep learning, MIT Press. Available at: http://www.deeplearningbook.org/, accessed March 20, 2017.

He, K., Zhang, X., Ren, S. & Sun, J. (2015), Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **37**(9), 1904–16.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. R. (2012), Improving neural networks by preventing co-adaptation of feature detectors, *arXiv*, **1207**, 0580, 1–18.

Hsieh, S., Yang, Y. & Hsu, P. (2002), Integration of general sparse matrix and parallel computing technologies for large–scale structural analysis, *Computer-Aided Civil and Infrastructure Engineering*, **17**(6), 423–38.

Hsu, C. W., Chang, C. C. & Lin, C. J. (2010). A practical guide to support vector classification. Available at: http://www.csie.ntu.edu.tw/˜cjlin/papers/guide/guide.pdf, accessed June 30, 2017.

Huang, Y. X. & Xu, B. G. (2006), Automatic inspection of pavement cracking distress, *Journal of Electronic Imaging*, **15**(1), 013017.1–013017.6.

Jahanshahi, M. R., Jazizadeh, F., Masri, S. F. & Becerik-Gerber, B. (2013), Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor, *Journal of Computing in Civil Engineering*, **27**(6), 743–54.

Jiang, C. & Tsai, Y. (2016), Enhanced crack segmentation algorithm using 3D pavement data, *Journal of Computing in Civil Engineering*, **30**(3), 04015050.1–04015050.10.

Jiang, X. & Adeli, H. (2005), Dynamic wavelet neural network model for traffic flow forecasting, *Journal of Transportation Engineering*, **131**(10), 771–79.

Kaseko, M. S. & Ritchie, S. G. (1993), A neural network-based methodology for pavement crack detection and classification, *Transportation Research Part C: Emerging Technologies*, **1**(4), 275–91.

Kaseko, M. S., Lo, Z. P. & Ritchie, S. G. (1994), Comparison of traditional and neural classifiers for pavement-crack detection, *Journal of Transportation Engineering*, **120**(4), 552–69.

Kirschke, K. R. & Velinsky, S. A. (1992), Histogram-based approach for automated pavement-crack sensing, *Journal of Transportation Engineering*, **118**(5), 700–10.

Krizhevsky, A., Sutskever, I. & Hinton, G. (2012), ImageNet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, **v2**, 1097–105.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86**(11), 2278–324.

Lee, B. J. & Lee, H. D. (2004), Position-invariant neural network for digital pavement crack analysis, *Computer-Aided Civil and Infrastructure Engineering*, **19**(2), 105–18.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013), Rectifier nonlinearities improve neural network acoustic models, in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia.

Maji, D., Santara, A., Mitra, P. & Sheet, D. (2016), Ensemble of deep convolutional neural networks for learning to detect retinal vessels in fundus images, *arXiv*, **1603**, 04833, 1–4.

Marques, A. G. C. S. (2012), Automatic road pavement crack detection using SVM. Thesis presented to Instituto Superior Técnico, Lisbon, Portugal, in partial fulltime of the requirements for the degree of Master of Science.

Miller, J. S. & Bellinger, W. Y. (2014), *Distress Identification Manual for the Long-Term Pavement Performance Program*, Federal Highway Administration, Washington DC.

Murphy, K. P. (2012), *Machine Learning: A Probabilistic Perspective*, The MIT Press, Cambridge, Massachusetts.

Nejad, F. M. & Zakeri, H. (2011), An optimum feature extraction method based on wavelet-radon transform and dynamic neural network for pavement distress classification, *Expert Systems with Applications*, **38**(8), 9442–60.

Nielsen, M. (2017), Improving the way neural networks learn. Available at: http://neuralnetworksanddeeplearning.com/chap3.html, accessed February 15, 2017.

Nisanth, A. & Mathew, A. (2014), Automated visual inspection on pavement crack detection and characterization, *International Journal of Technology and Engineering System*, **6**(1), 14–20.

NVidia (2017), CUDA C programming guide. Available at: http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz4eceVT29C, accessed March 1, 2017.

Oliveira, H. & Correia, P. L. (2009), Automatic road crack segmentation using entropy and image dynamic thresholding, in *Proceedings of the 17th European Signal Processing Conference*, Glasgow, Scotland, 622–26.

Ouyang, W. & Xu, B. (2013), Pavement cracking measurements using 3D laser-scan images, *Measurement Science and Technology*, **24**(10), 105204.1–105204.9.

Pinheiro, P. O. & Collobert, R. (2015), From image-level to pixel-level labeling with convolutional networks, *arXiv*, **1411**, 6228, 1–9.

Ponz-Tienda, J. L., Salcedo-Bernal, A. & Pellicer, E. (2016), A parallel branch and bound algorithm for the resource leveling problem with minimal lags, *Computer-Aided Civil and Infrastructure Engineering*, **32**(6), 474–98.

Santhi, B., Krishnamurthy, G., Siddharth, S. & Ramakrishnan, P. K. (2012), Automatic detection of cracks in pavements using edge detection operator, *Journal of Theoretical and Applied Information Technology*, **36**(2), 199–205.

Schrefler, B. A., Matteazzi, R., Gawin, D. & Wang, X. (2000), Two parallel computing methods for coupled thermohydromechanical problems, *Computer-Aided Civil and Infrastructure Engineering*, **15**(3), 176–88.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. & LeCun, Y. (2014), OverFeat: integrated recognition, localization and detection using convolutional networks, *arXiv*, **1312**, 6229, 1–16.

Shelhamer, E., Long, J. & Darrell, T. (2016), Fully convolutional networks for semantic segmentation, *arXiv*, **1605**, 06211.

Simonyan, K. & Zisserman, A. (2015), Very deep convolutional networks for large-scale image recognition, *arXiv*, **1409**, 1556, 1–14.

Sollazzo, G., Wang, K. C. P., Bosurgi, G. & Li, Q. (2016), Hybrid procedure for automated detection of cracking with 3D pavement data, *Journal of Computing in Civil Engineering*, **30**(6), 04016032.1–04016032.12.

Some, L. (2016), Automatic image-based road crack detection methods. Thesis presented to KTH Royal Institute of Technology, Stockholm, Sweden, in partial fulltime of the requirements for the degree of Master of Science.

Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. (2015), Striving for simplicity: the all convolutional net, *arXiv*, **1412**, 6806, 1–14.

Subirats, P., Dumoulin, J., Vinvent, L. & Barba, D. (2006), Automation of pavement surface crack detection using the continuous wavelet transform, in *Proceedings of International Conference on Image Processing*, Atlanta, Georgia, 3037–40.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2014), Going deeper with convolutions, *arXiv*, **1409**, 4842v1, 1–12.

Torbol, M. (2014), Real-time frequency-domain decomposition for structural health monitoring using general-purpose graphic processing unit, *Computer-Aided Civil and Infrastructure Engineering*, **29**(9), 689–702.

Tschopp, F. (2015), Efficient convolutional neural networks for pixelwise classification on heterogeneous hardware systems, *arXiv*, **1509**, 03371, 2–23.

Wang, K. C. P. (2011), Elements of automated survey of pavements and a 3D methodology, *Journal of Modern Transportation*, **19**(1), 51–57.

Wang, K. C. P., Li, Q. & Gong, W. (2007), Wavelet-based pavement distress image edge detection with à trous algorithm, in *Transportation Research Record 2024*, TRB, National Research Council, Washington DC, 73–81.

Wu, Q., Cole, C. & McSweeney, T. (2016), Applications of particle swarm optimization in the railway domain, *International Journal of Rail Transportation*, **4**(3), 167–90.

Ying, L. & Salari L. (2010), Beamlet transform-based technique for pavement crack detection and classification, *Computer-Aided Civil and Infrastructure Engineering*, **25**(8), 572–80.

Zalama, E., Gomez-Garcia-Bermejo, J., Medina, R. & Llamas, J. (2014), Road crack detection using visual features extracted by Gabor filters, *Computer-Aided Civil and Infrastructure Engineering*, **29**(5), 342–58.

Zeiler, M. D. & Fergus, R. (2013), Visualizing and understanding convolutional networks, *arXiv*, **1311**, 2901, 1–11.

Zhang, A., Li, Q., Wang, K.C.P. & Qiu, S. (2013), Matched filtering algorithm for pavement cracking detection, in *Transportation Research Record 2367*, TRB, National Research Council, Washington DC, 30–42.

Zhang, A. & Wang, K. C. P. (2017), The fast prefix coding algorithm (FPCA) for 3D pavement surface data compression, *Computer-Aided Civil and Infrastructure Engineering*, **32**(3), 173–90.

Zhang, A., Wang, K. C. P. & Ai, C. (2017), 3D shadow modeling for detection of descended patterns on 3D pavement surface, *Journal of Computing in Civil Engineering*, **31**(4), 04017019.1–04017019.13.

Zhang, A., Wang, K. C. P., Ji, R. & Li, Q. (2016a), Efficient system of cracking-detection algorithms with 1-mm 3D-surface models and performance measures, *Journal of Computing in Civil Engineering*, **30**(6), 04016020.1–04016020.16.

Zhang, L., Yang, F., Zhang, Y. D. & Zhu, Y. J. (2016b), Road crack detection using deep convolutional neural network, in *Proceedings of International Conference on Image Processing*, Phoenix, AZ, **v2016**, 3708–12.

Zhou, J., Huang, P. S. & Chiang, F. P. (2006), Wavelet-based pavement distress detection and evaluation, *Optical Engineering*, **45**(2), 027007.1–027007.10, 2006.

Zou, Q., Cao, Y., Li, Q. Q., Mao, Q. Z. & Wang, S. (2012), CrackTree: automatic crack detection from pavement images, *Pattern Recognition Letters*, **33**(3), 227–38.