COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Computer-Aided Civil and Infrastructure Engineering **00** (2018) 1–17

Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces with a Recurrent Neural Network

Allen Zhang

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

Kelvin C. P. Wang*

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

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

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

Enhui Yang

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

&

Shi Qiu

School of Transportation Engineering, Beijing University of Technology, China

Abstract: *A recurrent neural network (RNN) called CrackNet-R is proposed in the article for fully automated pixel-level crack detection on three-dimensional (3D) asphalt pavement surfaces. In the article, a new recurrent unit, gated recurrent multilayer perceptron (GRMLP), is proposed to recursively update the internal memory of CrackNet-R. Unlike the widely used long short-term memory (LSTM) and gated recurrent unit (GRU), GRMLP is intended for deeper abstractions on the inputs and hidden states by conducting multilayer nonlinear transforms at gating units. CrackNet-R implements a two-phase sequence processing: sequence generation and sequence modeling. Sequence generation is specifically developed in the study to find the best local paths that are most likely to form crack patterns. Sequence modeling predicts timely probabilities of the input sequence being a crack pattern. In terms of sequence modeling, GRMLP slightly outperforms LSTM and GRU by using only one more nonlinear layer at each gate. In addition to sequence processing, an output layer is proposed to produce pixel probabilities based on timely probabilities predicted for sequences. The proposed output layer is critical for pixel-perfect accuracy, as it accomplishes the transition from sequence-level learning to pixel-level learning. Using 3,000 diverse 3D images, the training of CrackNet-R is completed through optimizing sequence modeling, sequence generation, and the output layer serially. The experiment using 500 testing pavement images shows that CrackNet-R can achieve high Precision (88.89%), Recall (95.00%), and F-measure (91.84%) simultaneously. Compared with the original CrackNet, CrackNet-R is*

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

DOI: 10.1111/mice.12409

about four times faster and introduces tangible improvements in detection accuracy.

1 INTRODUCTION

Three-dimensional (3D) laser imaging technology has become popular for automated data collection in recent years (Wang, 2011; Wang and Smadi, 2011). Compared with two-dimensional (2D) pavement images, 3D pavement surface 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; Zhang et al., 2017a; Zhang et al., 2017b). Many algorithms based on traditional image processing were proposed for automated crack detection on 3D pavement data, including depth-checking methods (Jahanshahi et al., 2013; Ouyang and Xu, 2013), modified dynamic optimization algorithm (Jiang and Tsai, 2016), interactive crack detection algorithm (Zhang et al., 2016a), hybrid crack detection procedures (Sollazzo et al., 2016), and the 3D shadow modeling (Zhang et al., 2017a). However, traditional algorithms have many limitations in generalizing the complexity of crack detection on diverse pavement surfaces, and lack the capability of learning from example data (Zhang et al., 2017b).

Machine learning, particularly deep learning techniques have been widely applied in recent engineering practices (Adeli and Samant, 2000; Adeli and Jiang, 2003; Jiang and Adeli, 2005; Panakkat and Adeli, 2009; Rafiei et al., 2017). Artificial neural network (ANN) and support vector machine (SVM) were also adopted in early studies for pavement crack classification (Kaseko et al., 1994; Lee and Lee, 2004; Nejad and Zakeri, 2011; Daniel and Preeja, 2014). Nevertheless, the traditional learning models normally represent only one or two layers of abstraction, resulting in inadequate model capacity. Recent developments based on deep convolutional neural networks (CNNs) have demonstrated a more effective approach to automated crack detection (Zhang et al., 2016b; Cha et al., 2017a; Eisenbach et al., 2017; Zhang et al., 2017b). The faster region-based CNN (Cha et al., 2017b) can even detect multiple types of damages, including concrete cracks, steel corrosion, bolt corrosion, and steel delamination. It was shown (Zhang et al., 2016b; Zhang et al., 2017b) that deep CNNs significantly outperform traditional machine learning techniques, such as SVM and Boosting. Using invariant data size through all layers, the CNN-based CrackNet (Zhang et al., 2017b) was demonstrated to be efficient in detecting cracks on various 3D asphalt surfaces with high level of pixel-perfect accuracy.

The most promising feature of CNN is to detect objects through small local receptive fields (Lecun et al., 1998; Lecun et al., 2015). Nevertheless, cracks are difficult to be distinguished from other noise patterns when they are analyzed at small local receptive fields due to lack of global context. Figure 1 shows that the crack and the noise pattern are nearly indistinguishable when they are viewed through a small local receptive field. The typical errors resulted from CrackNet are false-positive errors due to complex local textures and false-negative errors occurring at hairline cracks (Zhang et al., 2017b). The primary reason might be that the local receptive fields used in CrackNet are insufficient to comprehend the difference between noise patterns and hairline cracks.

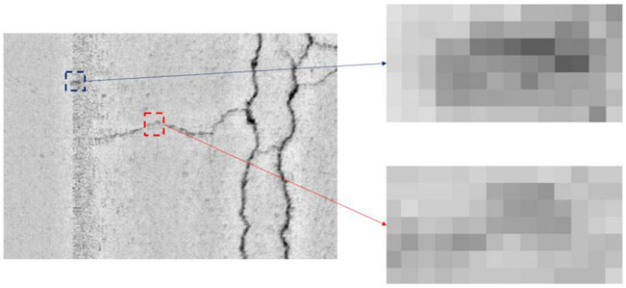

Fig. 1. Illustration of the local regions of a crack and a noise pattern.

For complex cases, the separation of cracks and noise patterns only can be resolved under a large context. The pooling layers used in traditional CNNs progressively downsize the data. Consequently, filters of same size used at deeper layers can be associated with larger context. Nevertheless, the pooling layers adversely impact pixel-perfect accuracy due to loss of original information. For explicit requirements on pixel-perfect accuracy, the local receptive field needs to be augmented for larger context (Luo et al., 2016). The straightforward option is to increase the sizes of filters used at convolutional layers. However, the computational cost significantly rises with large filter sizes. Another approach to increasing local receptive field for larger context is to stack more convolutional layers using small filters (Simonyan and Zisserman, 2015; He et al., 2015a), which reduces computations (Simonyan and Zisserman, 2015). Nevertheless, it only increases the local receptive field size linearly in theory, while introducing difficulties in training due to deeper architecture. More important, a considerable portion of the rectangle receptive field used in traditional CNNs may be ineffective (Luo et al., 2016). Regarding crack detection, a large rectangle receptive field may include excessive background pixels, resulting in attenuated local motifs. The local receptive field problem associated with CNNs creates many challenges in detecting cracks under larger context for higher confidence level.

Recurrent neural networks (RNNs) conduct sequence-based learning, and have been widely adopted in many applications (Liang and Hu, 2015; Tripathi et al., 2016; Milan et al., 2016). Different from forward neural networks, RNNs can address the flow of internal memory and process arbitrary input sequences (Hochreiter and Schmidhuber, 1997; Schmidhuber, 2014). Pavement cracks can be considered as a sequence of pixels that formulates a descended pattern on pavement surface. RNNs are therefore suitable for such an application. The concept of sequence rooted in RNNs is beneficial to pavement crack detection for two reasons. First, a long sequence of pixels reflects a large context and hence increases the confidence level. For instance, a 20-pixel-long crack segment may represent all useful information within a 20×20 local region. If the sequence is purely generated along the crack segment, many useless background pixels within the local region can be excluded from the sequence to avoid pattern attenuation and reduce computations. Second, group behavior can be emphasized through the concept of sequence. Pixels of a hairline crack normally present indistinctive elevation differences compared with the local surroundings. However, the hairline crack becomes recognizable when corresponding pixels are united as a whole. In other words, the hairline crack is more likely to be detected correctly on the basis of group behavior. RNNs dynamically update their internal memories based on previous inputs, placing an explicit bias on group behavior instead of individual behaviors. Further, RNNs can process variable-length sequences (Goodfellow et al., 2016). Fittingly pavement cracks have varying lengths.

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014; Chung et al., 2014) are the two popular recurrent units. However, LSTM and GRU only implement one-layer nonlinear transforms on the inputs and hidden states at most gating units, often leading to insufficient abstraction. Although LSTM and GRU can be stacked in a multilayer manner for deeper abstraction, the additional level of abstraction is not applied to the inputs directly. Alternatively, adding more nonlinear layers at transition stages potentially improves the overall performance by virtue of deeper abstractions on the inputs and hidden states (Pascanu et al., 2014; Salehinejad et al., 2018). Thus, this article proposes gated recurrent multilayer perceptron (GRMLP) to support multilayer abstractions on the inputs and hidden states directly at most gating units. Individual multilayer perceptrons (MLPs) are deployed at most gating units of the proposed GRMLP, and can consist of arbitrary numbers of nonlinear layers to provide various degrees of abstractions on the inputs and hidden states.

The proposed GRMLP differs from the Deep Transition RNNs (DT-RNNs) (Pascanu et al., 2014) and recurrent multilayer perceptrons (RMLPs) (Puskorius and Feldkamp, 1994; Parlos et al., 1994). A major difference is that the DT-RNNs and RMLP do not have specific gating units, and the transitions in DT-RNNs and RMLP are completed mostly through MLPs. The success of LSTM and GRU has demonstrated that well-designed gating units are important for learning long-term dependencies and tackling gradient vanishing problems (Chung et al., 2014).

In this study, based on development of the original CNN-based CrackNet (Zhang et al 2017b), the described CrackNet-R is based on GRMLP to detect pavement cracks at pixel level on 3D asphalt surfaces with tangible improvements on detection accuracy and processing speed. This article proposes a systematic methodology to train an RNN for pixel-level pavement crack detection. The proposed CrackNet-R consists of three components: sequence generation, sequence modeling, and an output layer. The sequence generation method developed in the article strives to capture potential crack fragments by inspecting all possible orientations. Based on the proposed GRMLP, sequence modeling treats each generated sequence as an input sequence of pixels, and subsequently predicts timely probabilities of the input sequence being a crack pattern. Unique strategies are used to train the sequence modeling such that it will be capable of identifying crack patterns of variable lengths. The output layer is proposed in the article specifically for pixel-perfect accuracy. The output layer predicts pixel probabilities based on sequence probabilities, resulting in enhanced pixel-perfect accuracy.

2 DATA PREPARATION

The asphalt pavement surface data used in the article are 1-mm 3D data from the *PaveVison3D* system by WayLink, which can scan the pavement surface at the data collection speed of 60 MPH with full coverage for a 4-m-wide lane. The research team built an image library consisting of more than 6,500 3D pavement images and corresponding ground-truth images of pavement cracks through a laborious processing in a 12-month period of time with more than six people. Figure 2 shows a 3D virtual pavement surface rendered via the OpenGL platform based on the elevation information of collected pavement surface area.

Figure 3 shows several typical 3D pavement images and corresponding ground-truths of pavement cracks from the image library. Four thousand asphalt surface images are selected from the image library for the

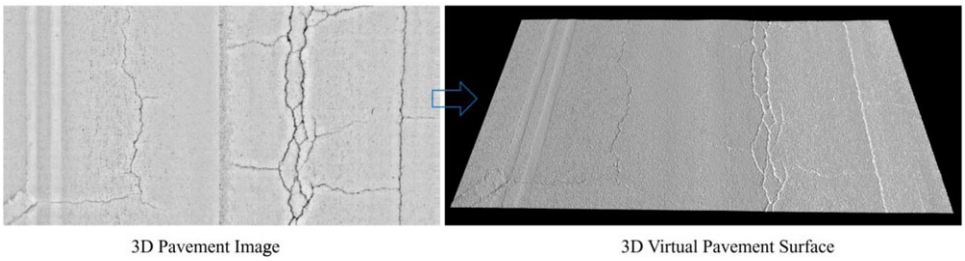

Fig. 2. Illustration of rendered 3D virtual pavement surface.

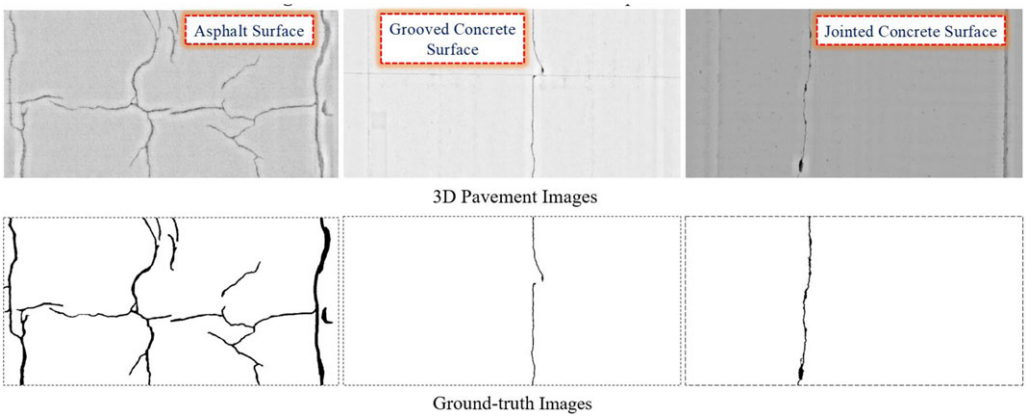

Fig. 3. Representative 3D pavement images and ground-truth images from the image library.

training and testing of CrackNet-R. 500 images are randomly selected as testing data. Another set of 500 images is also randomly selected as validation data. The remaining 3,000 images are used for training.

To reduce the computational overhead, the original 3D image of the 4,096 × 2,048 size is downsized to a 1,024 × 512 3D image by min-pooling techniques, which outputs the minimal one of nearby values at a certain location. Given that a crack pixel has a lower elevation compared to the local surroundings, the min-pooling method is more likely to make fine or hairline cracks visible in the downsized data sets. In this article, all cracks on the 4,000 original images are verified to be recognizable on the 4,000 downsized images. It will be demonstrated that CrackNet-R can detect a large percentage of cracks (roughly 95%) on downsized images, implying the down-sampling procedure does not have a significant impact on the detection accuracy.

3 METHODOLOGY

CrackNet-R illustrated in Figure 4 represents two-phase sequence processing: sequence generation and sequence modeling. Sequence generation is to find a rational path at each pixel that may form a crack pattern. Each path assembles multiple pixels and is considered as a sequence of connected pixels. On the other hand, sequence modeling is to take the features of all pixels included in a sequence as input and predict how likely the corresponding sequence forms a crack pattern. If sequence modeling is considered in the time domain, a new pixel is added to the sequence at each time step. The number of time steps thus equals to the number of pixels included in the input sequence. The timely sequence is then defined in the article as the input sequence till current time step. For instance, timely sequence at time k means the sequence from the first input pixel to the kth input pixel. Accordingly, the timely probability in Figure 4 is defined as the probability that the timely sequence or the input sequence till current time step forms a crack pattern. In addition, for each time step, or equivalently for each pixel included in the sequence, the feature vector of the corresponding pixel will be extracted and provided as the input data of the network at the corresponding time step.

An output layer is also proposed for pixel-perfect accuracy to produce pixel probability based on timely probabilities predicted for sequences. Figure 5 shows the process to yield the final pixel probability based on corresponding timely probabilities. Let u and v represent the pixel coordinates. Suppose many timely probabilities are predicted at pixel (u, v) during sequence modeling, such as the highlighted probabilities

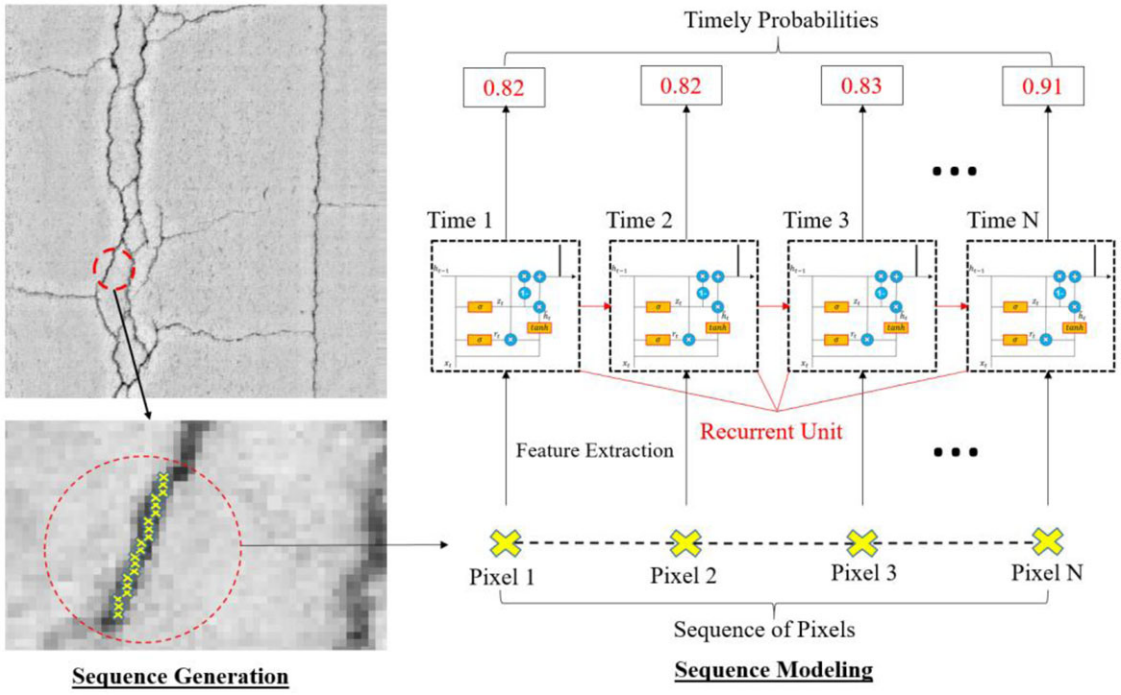

Fig. 4. Illustration of two-phase sequence processing in CrackNet-R.

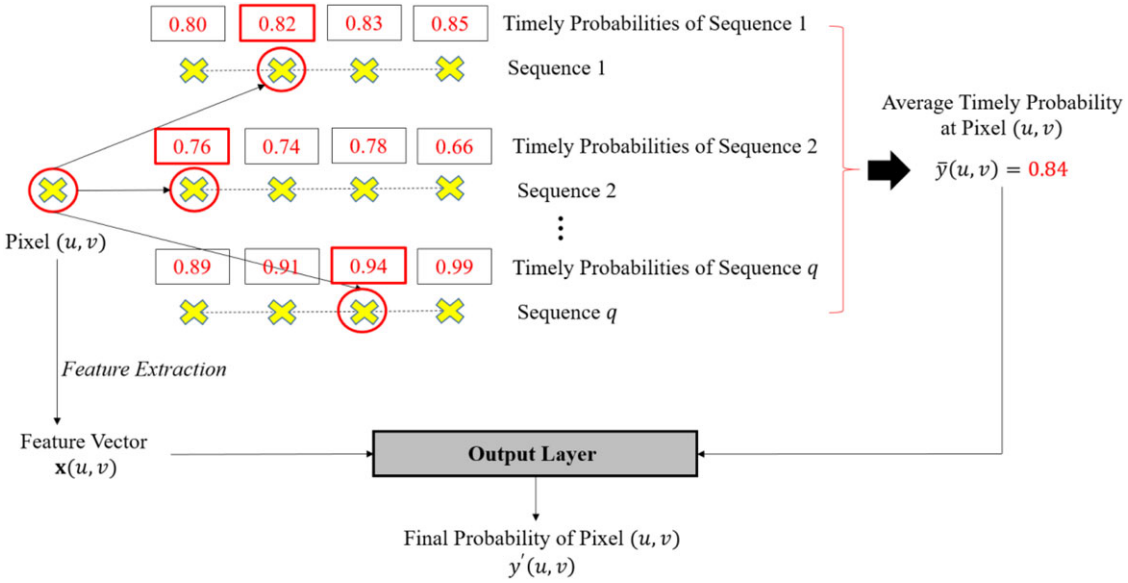

Fig. 5. Illustration of producing final pixel probability through output layer.

0.82, 0.76, and 0.94. The average timely probability predicted at pixel (u, v) is then computed. Subsequently, the output layer predicts the final pixel probability by taking the average timely probability and feature vector of pixel (u, v) as inputs. Based on pixel probabilities, CrackNet-R is able to learn errors at pixel level instead of sequence level, leading to enhanced pixel-perfect accuracy.

3.1 Sequence generation

With respect to an individual pixel, the difference between a crack pixel and a noise pixel can be indiscernible even by human eyes. As shown in Figure 1, the crack and the noise pattern are indistinguishable when they are viewed at small local regions. However, they become fully separable by human eyes under a larger

context. The underlying principle is that a crack perceived by human eyes must consist of a sequence of neighboring pixels, and formulate a clear linear pattern whose length exceeds certain perceived "values." Thus, crack detection must combine multiple pixels together and hence conduct classifications based on their group characteristics. Otherwise, noise patterns will be difficult or even impossible to be discriminated. Thereby, the sequence generator is developed here to unite successive pixels in a manner that includes as many crack pixels as possible. The ideal case is that the entire crack can be united and analyzed as a whole.

The objective of sequence generation in the study is to find the best local path at each individual pixel that is most likely to form a crack pattern. The best local path for an individual pixel is defined in the article as the best oriented line segment that connects the largest number of crack pixels. However, it is impossible to know whether an individual pixel belongs to a crack or not at the initial phase of detection. As the crack has lower elevation compared with the local surroundings, the best local path is then redefined as the best oriented line segment with fixed length that presents the lowest average elevation. In the article, an oriented line segment $P(u, v, l, \theta)$ is specified with its origin (u, v), length l, and orientation θ:

$$P\ (u, v, l, \theta) = \{(u_i',\ v_i') \mid\ u_i' =\ u + icos\theta,\ v_i' =\ v + isin\theta, 0 \le i < l\} \quad (1)$$

In Equation (1), the oriented line segment $P(u, v, l, \theta)$ is defined in a discrete manner, as the 3D pavement images are essentially digital images. For digital images, the variable i in Equation (1) only takes integer values within the specified range for a unit sampling interval (1 pixel) along the orientation θ. Denote $\bar{h}(u, v, l, \theta)$ as the average elevation of all sample points along the oriented line segment $P(u, v, l, \theta)$, then the best orientation $\hat{\theta}$ for pixel (u, v) that yields the lowest average elevation can be expressed as

$$\hat{\theta} = \text{argmin}_\theta\left(\bar{h}\,(u, v, l, \theta)\right) \quad (2)$$

Consequently, the best local path $S(u, v, l)$ for pixel (u, v) is identified as

$$S\ (u, v, l) =\ P\left(u, v, l, \hat{\theta}\right) \quad (3)$$

The length l in this study is initialized as 20-pixel-long for image size 1,024 × 512. The optimization of length l will be discussed in Section 4.2. With fixed length l, the best local path is purely dependent on the best orientation $\hat{\theta}$. An approach to determining $\hat{\theta}$ is to compute the average elevation $\bar{h}(u, v, l, \theta)$ along each possible orientation. Particularly, a sufficiently small angle interval can be used to rotate the fixed-length line segment to various possible orientations and subsequently compute corresponding average elevations. The average elevation $\bar{h}(u, v, l, \theta)$ is computed at every 1°. As a result, the maximum translation in either u or v axis will be less than $2l \cdot \sin(0.5^\circ)$, or roughly 0.35 pixel. In other words, for digital images, all possible orientations can be considered using such a small angle interval. According to Equations (1)–(3), the best local paths for all pixels are generated by treating each pixel as the origin of a local path. Subsequently, each generated local path is a sequence of connected pixels and will be provided for sequence modeling.

3.2 Sequence modeling

Sequence modeling is to predict the probability of the input sequence being a crack pattern. Given an input sequence, features of each pixel included in the input sequence are extracted and provided as the input vector for each time step. Figure 4 shows that the number of input feature vectors extracted for the input sequence is equal to the input sequence length l. The RNN receives the input vector and then transmits a scalar output probability at each time step. The recurrent unit is a key structure repeatedly used in the RNN and serves as the data interpreter at each time step. Denote $X = \{x_1, x_2, \ldots, x_n\}$ as the input feature vectors for n time steps, $Y = \{y_1, y_2, \ldots, y_n\}$ as output probabilities for the n time steps, and $Z = \{z_1, z_2, \ldots, z_n\}$ as the target probabilities for the n time steps. Let ω be the parameters of the recurrent unit. Then, the sequence modeling is subject to an optimization problem:

$$\hat{\omega} = \text{argmax}_\omega(p(Y = Z|X, \omega)) \quad (4)$$

where $p(\cdot)$ represents the probability that the corresponding event is true, and $\hat{\omega}$ denotes optimal parameters of the recurrent unit.

Figure 6 illustrates the method to produce the feature vector of an individual pixel in the sequence, or equivalently for a single time step. The elements of the feature vector of pixel (u, v) are differences between the elevation of pixel (u, v) and the local average elevations sampled along evenly-oriented line segments. The local circle around pixel (u, v) with radius ϕ is not considered in feature extraction. Such a consideration can reduce the risk of contrast attenuation occurring for wide cracks, as wide cracks have similar elevations at small local neighborhoods. The oriented line segments involved in feature extraction intend to represent the local surroundings. The oriented line segments are evenly aligned and distributed in the spatial domain for isotropic comparison. In addition, the local elevation

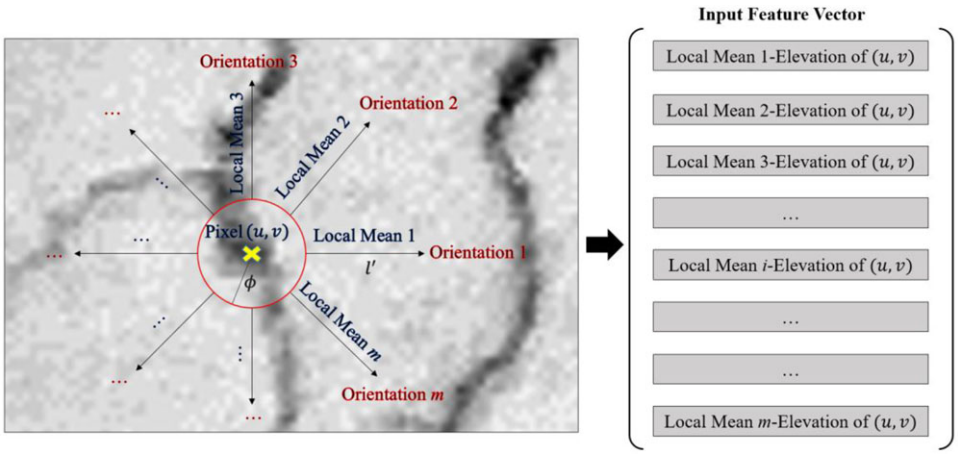

Fig. 6. Illustration of feature extraction.

difference instead of the original elevation is considered in feature extraction to emphasize local contrast and enhance spatial invariance. On the one hand, local elevation difference purely conveys local contrast. On the other hand, local elevation difference potentially discards pavement unevenness occurring at different locations.

Different from Equation (1), an oriented line segment used for feature extraction is specified as:

$$\begin{aligned} P'\ (u, v, \phi, l', \theta_i') &= \{ (u_j',\ v_j') \mid \\ u_j' &= u + (j + \phi)\, cos\theta_i', \\ v_j' &= v + (j + \phi)\, sin\theta_i', 0 \le j < l' \} \end{aligned} \quad (5)$$

where $P'(u, v, \phi, l', \theta_i')$ represents the ith oriented line segment around pixel (u, v) used for feature extraction, ϕ is the radius of the local circle, l' is the length of the corresponding line segment, and θ_i' represents the orientation of the corresponding line segment.

Given that $\overline{h'}(u, v, \phi, l', \theta_i')$ is the average elevation of $P'(u, v, \phi, l', \theta_i')$, and m orientations are considered for feature extraction, the feature vector for pixel (u, v) can be identified as

$$x\ (u, v) = \{x\,(u, v, i) \mid i\ =\ 1,\ 2,\ 3, \ldots,\ m\} \quad (6)$$

subject to

$$x\ (u, v, i) = \overline{h'}\ (u, v, \phi, l', \theta_i') - I\,(u, v) \quad (7)$$

where $x(u, v, i)$ is the ith element in $x(u, v)$, and $I(u, v)$ is the original elevation of pixel (u, v).

Finally, if pixel (u, v) is the kth pixel in the sequence, then the input vector at the kth time step x_k is defined as

$$x_k = x\,(u, v) \quad (8)$$

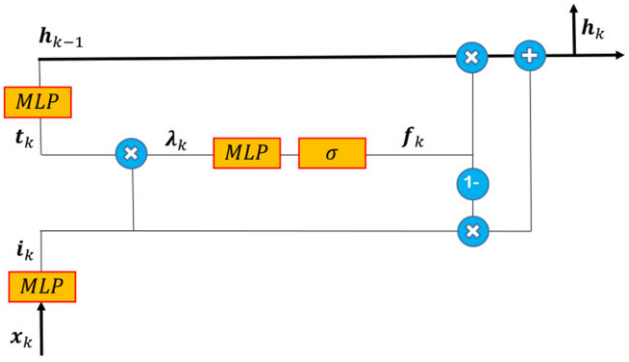

Fig. 7. Illustration of proposed GRMLP.

In this study, 30 orientations distributed in every 12° are used to extract features for reasonable computational efficiency. In addition, the length of the oriented line segment l' and the radius of the local circle ϕ are initialized as 20 and 0, respectively. The optimization of l' and ϕ will be discussed in Section 4.2.

LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014; Chung et al., 2014) are the two widely used recurrent units. Sophisticated gating units performing affine transforms and nonlinear activations are designed in both LSTM and GRU to address long-term dependencies. However, most gates used in LSTM and GRU only conduct one-layer affine transform on input vectors and previous hidden state vectors before the nonlinear activations, which may result in shallow abstraction. This article proposes GRMLP in Figure 7 that conducts deeper abstractions on input vectors and previous hidden state vectors by using MLPs at most gates. GRMLP consists of four gates, including input gate, transform gate, forget gate, and output gate. The

input gate interprets the input vector provided at each time step, and is defined as

$$i_k = F_i\ (x_k, \Omega_i) \tag{9}$$

where i_k is the output of the input gate, $F_i(\cdot)$ represents the MLP operation applied at the input gate, and Ω_i is the parameters of the corresponding MLP. Similarly, the transform gate transforms the previous hidden state vector via the MLP operation, and is defined as

$$t_k = F_t\ (h_{k-1}, \Omega_t) \tag{10}$$

where t_k is the output of the transform gate, $F_t(\cdot)$ represents the MLP operation used at the transform gate, h_{k-1} is the hidden state vector at previous time step, and Ω_t is the parameters of the MLP used at the transform gate.

The forget gate first merges the output vectors of the input and transform gates into one vector via element-wise multiplication, and then transforms the merged vector via MLP operation. Finally, element-wise logistic sigmoid function is used to produce the output vector of the forget gate whose values are between 0 and 1. The forget gate of the proposed GRMLP is defined in the article as

$$\lambda_k = i_k\ \otimes t_k \tag{11}$$

$$f_k = \sigma\ [F_f\ (\lambda_k,\ \Omega_f)] \tag{12}$$

where f_k is the output of the forget gate, $F_f(\cdot)$ represents the MLP operation used at the forget gate, Ω_f is the parameters of the corresponding MLP, λ_k is the Hadamard product of i_k and t_k, $\sigma[\cdot]$ denotes the element-wise logistic sigmoid function, and $\otimes$ denotes element-wise multiplication.

Eventually, the output gate produces the hidden state vector for current time step based on the following equation:

$$h_k = (1 - f_k)\ \otimes i_k + f_k \otimes h_{k-1} \tag{13}$$

where h_k is the hidden state vector at current time step.

With respect to crack detection, if h_{k-1} carries the probability information of how likely the previous sequence forms a crack pattern, the vector i_k is anticipated to present the likelihood of the new pixel added at current time step being a crack pixel. Consequently, the output of the forget gate f_k defines the amount of existing memory to be forgotten, and the amount of new content to be added. The multiplication shown in Equation (11) is expected to yield negative values if the existing content and new content belong to different classes, or positive values if the existing content and new content belong to the same class. Based on Equation (13), the internal memory represented by the hidden state h_k can be updated in a timely manner when the proposed GRMLP is applied recursively at each time step.

In this article, two typical architectures of the proposed GRMLP are adopted for sequence modeling, denoted as GRMLP-A and GRMLP-B. In Figure 8, GRMLP-A only has one nonlinear layer at most gates, which is considered as the simplest architecture of GRMLP. GRMLP-B is designed for a higher model capacity by using two nonlinear layers at most gates. The size of vectors used in GRMLP is $m \times 1$, where m is the number of orientations used for feature extraction. Different from LSTM and GRU, parametric rectified linear unit (PRELU; He et al., 2015b) is the primary nonlinear activation function used in the GRMLP. The hyperbolic tangent and logistic sigmoid functions extensively used in LSTM and GRU are highly possible to yield zero gradients during back-propagation due to saturation. Nevertheless, PRELU avoids zero gradients effectively (He et al., 2015b). In addition, the nonlinearity introduced by PRELU is learnable by tuning the parameters of PRELU. In this study, the LSTM and GRU are implemented for comparison study.

The hidden state h_k is considered as the output vector of the recurrent unit at time k. However, as shown in Figure 4, it is anticipated the output at time k is a scalar probability. Therefore, additional weight vector w_{hy} $(m \times 1)$ and bias b_y are used here to convert h_k to a scalar value, and then logistic sigmoid function is applied to the scalar value for nonlinear activation:

$$y_k = \sigma\ [(w_{hy} \cdot h_k + b_y)] \tag{14}$$

where y_k is the timely output probability at time k. Given a series of input vectors, the recurrent unit is repeatedly applied to produce a series of timely probabilities, indicating how likely the timely input sequence forms a crack pattern.

3.3 Output layer for pixel-perfect accuracy

To achieve pixel-perfect accuracy, it is proposed to produce pixel probabilities based on timely probabilities predicted for sequences. The pixel probability represents the likelihood of a pixel being an element of the crack. Two procedures are proposed to generate pixel probabilities based on timely probabilities of sequences. First, the average timely probability predicted at each pixel is computed, as a single pixel may be included in multiple sequences. Second, the output layer predicts the final probability of each pixel by considering the average timely probability as well as the feature vector of

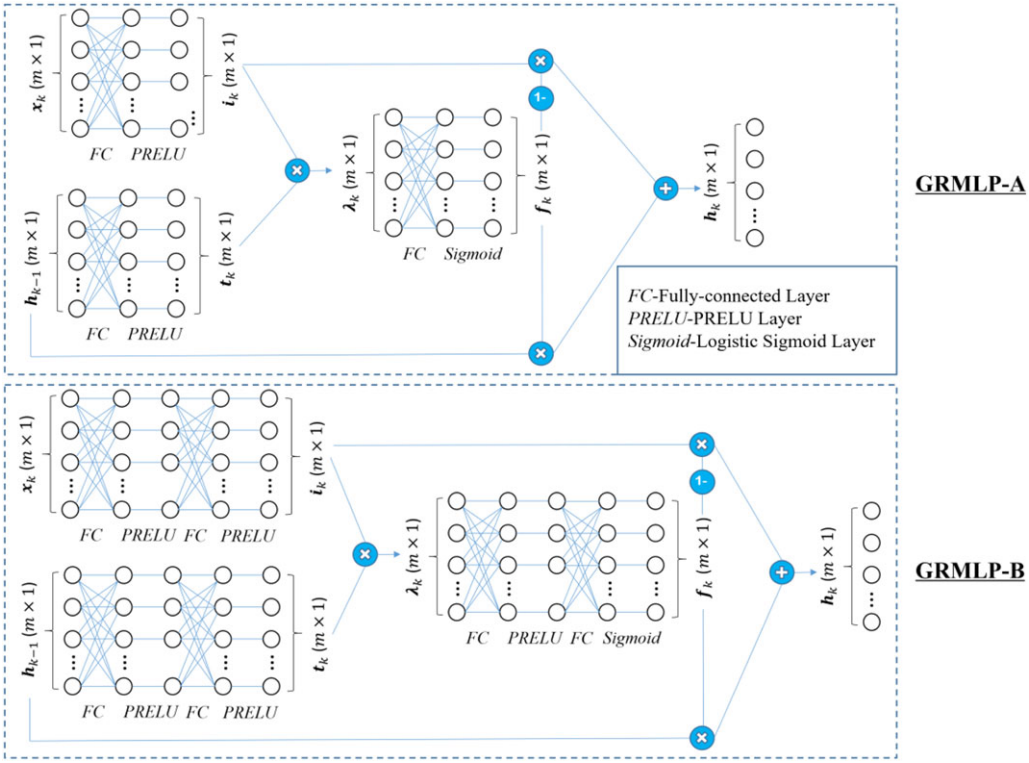

Fig. 8. Illustration of GRMLP-A and GRMLP-B.

the corresponding pixel. The average timely probability produced at pixel (u, v) is defined as

$$\bar{y}\ (u, v) = \frac{1}{q} \sum_{i=1}^{q} y_{k_i'}\ (u_i', v_i',\ l) \tag{15}$$

subject to

$$\forall i, (u, v) := s_{k_i'}$$

where $s_{k_i'}$ is the k_i'th element in sequence $S(u_i', v_i',\ l)$; $y_{k_i'}(u_i', v_i',\ l)$ is the timely output probability at time k_i' for sequence $S(u_i', v_i',\ l)$; q is the number of sequences that include pixel (u, v).

The output layer proposed in the article takes both the feature vector and average timely probability produced at each pixel as inputs, and transforms them into a scalar value. Subsequently, logistic sigmoid function is applied to the transformed value to yield the final pixel probability.

$$y'\ (u, v) = \sigma\ [(w_{xy'} \cdot x\ (u, v) + \omega_{yy'} \bar{y}\ (u, v) + b_{yy'})] \tag{16}$$

where $y'(u, v)$ is the final probability of pixel (u, v); $w_{xy'}$ is a $m \times 1$ weight vector; $\omega_{yy'}$ and $b_{yy'}$ are two scalar parameters.

The parameters of the output layer, including $w_{xy'}$, $\omega_{yy'}$, and $b_{yy'}$, are shared at all pixels to address space invariance. The output layer is critical to pixel-perfect accuracy. First, it correlates pixel probabilities with timely probabilities predicted by the RNN. It will be discussed in Section 4 that the timely target probabilities for sequences of more noncrack pixels will be comparatively lower. Therefore, well-trained RNN is expected to produce lower timely probabilities for sequences of more noncrack pixels. In other words, the average timely probability evaluated at a noncrack pixel is more likely to be lower. As a result, crack pixels and noncrack pixels become separable through averaging timely output probabilities. Second, the output layer further emphasizes pixel-perfect accuracy through the use of feature vector differing from pixel to pixel. Last, the output layer provides the media through which the errors can be learned at pixel level instead of sequence level. Existence of the output layer helps the network to learn an optimal boundary in terms of extracted features as well as average timely probability such that the gap between crack pixels and non-crack pixels is maximized.

4 TRAINING

The training of CrackNet-R is to optimize sequence modeling, sequence generation, and the output layer sequentially.

4.1 Optimization of sequence modeling

In this study, the timely target probability z_k defines the desired likelihood that the sequential pixels from Time 1 (the very beginning) to Time k yield a crack pattern. Considering the sequence of pixels starting from pixel (u, v) in time domain, the sequence of pixels from Time 1 to Time k starting from pixel (u, v) can be described as $P(u, v, k, \hat{\theta})$ according to Equations (1) and (3). Denote $N_{(u,v)}^{(k)}$ as the number of crack pixels in timely input sequence $P(u, v, k, \hat{\theta})$, the timely target probability z_k for the origin pixel (u, v) is defined in the article as

$$z_k = N_{(u,v)}^{(k)}/k \tag{17}$$

The timely target probability defined in Equation (17) is a critical concept specifically proposed for identifying crack patterns of variable lengths. First, in Equation (17), the timely target probability is directly associated with the number of crack pixels included in the timely input sequence. If crack pixels in a timely input sequence are no fewer than noncrack pixels, the corresponding timely input sequence is considered as a crack pattern, and the timely target probability will thus be no smaller than 0.5. Otherwise, the timely input sequence is considered as a noncrack pattern, and the corresponding timely target probability is smaller than 0.5. The timely target probability is greater when more crack pixels are included in the timely input sequence. On the contrary, the timely target probability is lower when more noncrack pixels are included. Second, given a sequence with fixed length l, there will be l timely input sequences of increasing lengths through l time steps. The RNN will thus learn to predict timely probabilities for variable-length timely input sequences. The negative effect of generating fixed-length sequences is therefore mitigated. In other words, the RNN trained in such a manner is equally applicable to variable-length sequences. In addition, the timely target probability is purely dependent on the timely input sequence instead of an individual pixel. Such a specification can force the RNN to learn a fundamental principle that a sequence of pixels is considered as a crack pattern if half or more than half of the pixels are crack pixels, no matter how long the sequence is.

The cross entropy is adopted in the study as the cost function to measure the dissimilarity between timely output probabilities and timely target probabilities (Goodfellow et al., 2016):

$$C = -\frac{1}{n}\sum_{k=1}^{n}[z_k ln y_k + (1 - z_k)\ln(1 - y_k)] \tag{18}$$

where n is the number of time steps or the length of the input sequence.

This cost function represents the average error over all time steps, and is defined for an individual sequence. However, numerous sequences will be generated for a mini-batch of input images or even a single input image. Therefore, the training objective is to minimize the average of cost function values for all generated sequences. According to Equation (4), the optimization of sequence modeling is to maximize the similarity between timely output probabilities and timely target probabilities. The cost function as defined in Equation (18) can serve as a measure of the dissimilarity between timely output probabilities and timely target probabilities. Therefore, the optimization of sequence modeling equivalently is to minimize this cost function. Mini-batch Gradient Descent, a variation of Stochastic Gradient Descent, is implemented to optimize the parameters of recurrent unit recursively through iterations.

Four RNNs using LSTM, GRU, GRMLP-A, and GRMLP-B are, respectively, denoted as CrackNet-LSTM, CrackNet-GRU, CrackNet-GRMLP-A, and CrackNet-GRMLP-B. The four RNNs are trained individually for a comparison study. The 3,000 training images are all stochastically involved in the learning process to optimize the parameters of the recurrent unit. The 500 validation images do not participate in the learning process, and are used to monitor the network performance and inspect if overfitting problem occurs. For time efficiency, the performances of the four networks on 500 validation images are evaluated at every 25 iterations instead of each iteration. In the meantime, the learned parameters are saved at every 25 iterations for final selection of optimal parameters that yield the best performances on the 500 validation images. The accuracy of modeling an individual sequence is defined in the article as

$$\mathcal{H} = \frac{1}{n}\sum_{k=1}^{n} q^{(k)} \tag{19}$$

subject to

$$q^{(k)} = \begin{cases} 0, & \left(y_k \geq \frac{1}{2},\ t_k < \frac{1}{2}\right) or\ \left(y_k < \frac{1}{2},\ t_k \geq \frac{1}{2}\right) \\ 1, & \left(y_k \geq \frac{1}{2},\ t_k \geq \frac{1}{2}\right) or\ \left(y_k < \frac{1}{2},\ t_k < \frac{1}{2}\right) \end{cases} \tag{20}$$

where $\mathcal{H}$ represents the modeling accuracy for an individual sequence; $q^{(k)}$ is an indicator that signifies whether the timely input sequence at time k is classified correctly. A zero value of $q^{(k)}$ means the timely input sequence is classified incorrectly. On the other hand, the timely input sequence is classified correctly when $q^{(k)}$ is equal to 1.

In the article, the average modeling accuracy over all generated sequences is used to evaluate the network performance during training. The four networks are trained identically through 2,000 iterations. Figure 9

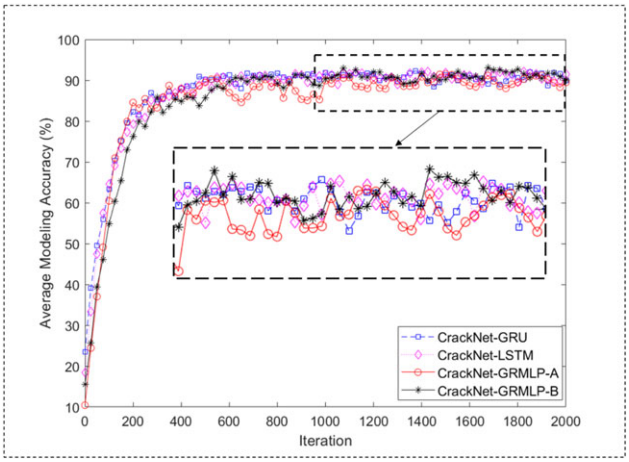

Fig. 9. Evolution of modeling accuracies on validation data.

shows the evolution of modeling accuracies on the 500 validation images through 2,000 iterations. The performances of the four networks are similar. However, compared with CrackNet-GRMLP-A, the CrackNet-GRU, and CrackNet-LSTM yield generally higher modeling accuracies on validation data. The CrackNet-GRMLP-B results in lower modeling accuracies for the first 1,000 iterations in comparison with CrackNet-GRU and CrackNet-LSTM. Nevertheless, CrackNet-GRMLP-B progressively behaves better than the other three networks after 1,000 iterations. The highest modeling accuracies achieved by CrackNet-GRU, CrackNet-LSTM, CrackNet-GRMLP-A, and CrackNet-GRMLP-B are 92.29%, 92.17%, 91.51%, and 93.06%, respectively. CrackNet-GRMLP-B slightly outperforms the other three networks. The efficiency of CrackNet-GRMLP-B implies that the modeling accuracy can potentially be improved by using deeper architectures with more nonlinear layers at gating units. The trained CrackNet-GRMLP-B is christened as CrackNet-R for discussions in the rest of the article.

4.2 Optimization of sequence generation

Before sequence modeling, the sequence generator and the feature extractor use initialized parameters to serially produce sequences of pixels and sequences of feature vectors. Such procedures can be considered as subtasks in sequence generation to prepare input vectors for sequence modeling. If the parameters for sequence generation can also be optimized to produce more effective input sequences, the network performance potentially can be further improved. Table 1 summarizes the initial values of parameters for sequence generation. For digital images, the parameters in Table 1 should all take nonnegative integer values, leading to substantially reduced searching space. However, the searching space is still infinite if these parameters are not confined within manageable ranges. Thus, appropriate range of each parameter is estimated before the optimization for manageable searching space. The recommended ranges of all parameters for sequence generation are shown in Table 1.

Table 1
Initial values and recommended ranges of parameters for sequence generation

Procedure	*Pixel sequence generation*	*Feature extraction*		
Parameters	l	ϕ	l'	m
Initial value	20	0	20	30
Recommended range	[15, 25]	[0, 10]	[5, 20]	[20, 180]

Importantly, the number of orientations m used to collect local background information determines the size of input feature vector, and hence impacts the parametric structure of the recurrent unit. Any different value of m requires retraining of the recurrent unit, which is computationally costly. More importantly, the parameter m is insensitive as increasing m does not yield a big difference once it has exceeded certain values. Per experience the elevations of local background are grasped sufficiently when m is equal to 30. Therefore, the parameter m is not considered in optimization. Based on the recommended ranges in Table 1, there are 1,936 possible combinations of parameter values. The objective of optimizing sequence generation is to yield maximal modeling accuracy on validation data. Thus, the average modeling accuracy on validation data under each possible combination of parameter values is evaluated in this study to attain the best set of parameters for sequence generation. However, if all 500 validation images are used to measure the average modeling accuracy resulted from a single set of parameter values, the entire optimization task is equivalent to processing 0.97 million images or 508 billion sequences, which would be too time consuming. To reduce the computational cost, 100 images randomly selected from the 500 validation images are employed as a subset of validation data to measure the average modeling accuracy under each possible set of parameter values. Before optimization, the 100 randomly selected images are verified again to have cracks of various severity levels. The average modeling accuracies on 100 validation images resulted from 1,936 possible sets of parameter values are in Figure 10. Note that the vertical shift of modeling accuracies is primarily due to changes of parameter l'.

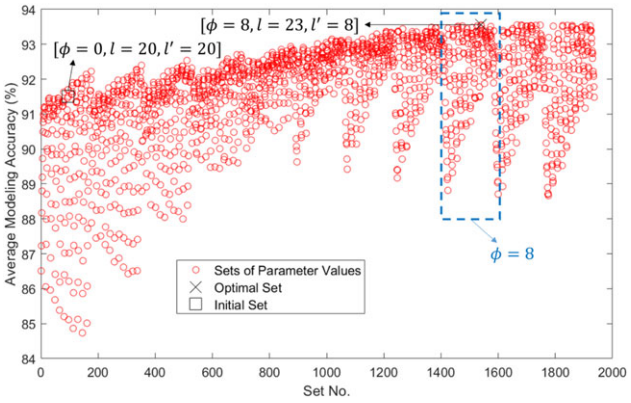

Fig. 10. Average modeling accuracies under possible sets of parameter values.

It is found that similar accuracy levels can be achieved for sequences with varying lengths, indicating that the trained recurrent unit is applicable to variable-length sequences. The initial set of parameter values $[\phi = 0, l = 20, l' = 20]$ yields modeling accuracy 91.51% on the 100 validation images, whereas the optimal set of parameter values $[\phi = 8, l = 23, l' = 8]$ can achieve modeling accuracy 93.56% on the 100 validation images. In addition, the optimal set is observed within the estimated searching space rather than at the boundary, implying that other sets out of the estimated searching space may not yield better performances. With the optimal set of parameter values, the average modeling accuracy on all 500 validation images is 94.98%, introducing roughly 2% improvement compared with the average modeling accuracy 93.06% resulted from the initial parameter values.

Figure 11 demonstrates typical performances of optimized sequence processing. The average timely probability map represents the average timely probabilities computed at all pixels per Equation (15). In the average timely probability map, a higher intensity value corresponds to a higher average timely probability. In Figure 11, crack pixels universally result in higher average timely probabilities, indicating that the two-phase sequence processing is trained successfully.

4.3 Optimization of output layer

Similar to Equation (18), cost function C' for training the output layer is identified as

$$C' = -\frac{1}{UV}\sum_{u=1}^{U}\sum_{v=1}^{V}\begin{bmatrix} z'(u,v)\ln(y'(u,v)) + \\ (1-z'(u,v))\ln(1-y'(u,v)) \end{bmatrix} \tag{21}$$

where U and V are the image width and height, respectively; $z'(u, v)$ is the target probability for pixel (u, v).

The target probability for a crack pixel is 1.0, whereas the target probability for a noncrack pixel is 0.0. Performance indicators, Precision, Recall, and F-measure, are adopted in the article to measure pixel-level detection accuracy (Fawcett, 2006; Zhang et al., 2016a). The Precision can be considered as *cost*, whereas Recall can be considered as *benefit*. F-measure is the harmonic mean of Precision and Recall (Fawcett, 2006), reflecting the accuracy of an algorithm. Similarly, the optimization of output layer still utilizes Mini-batch Gradient Descent algorithm, and is completed after 500 iterations. The network performance on 500 validation images is evaluated at every 25 iterations. Parameters of the output layer are also saved at every 25 iterations.

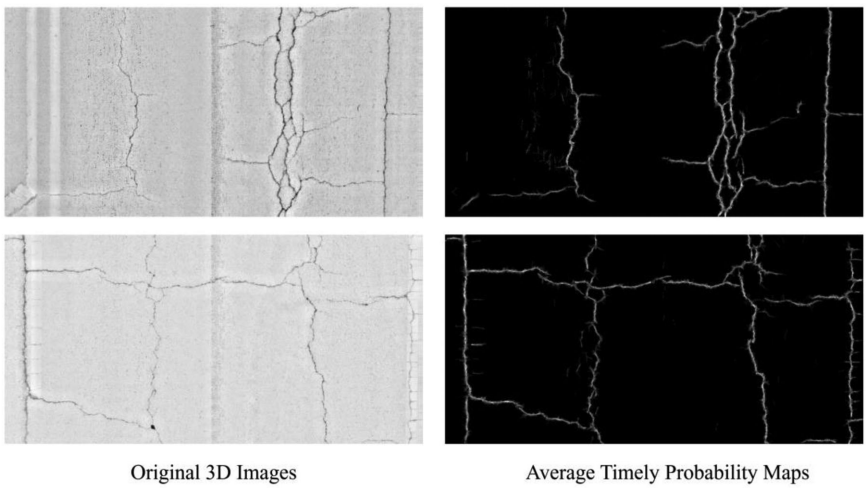

Fig. 11. Illustration of typical performance of two-phase sequence processing.

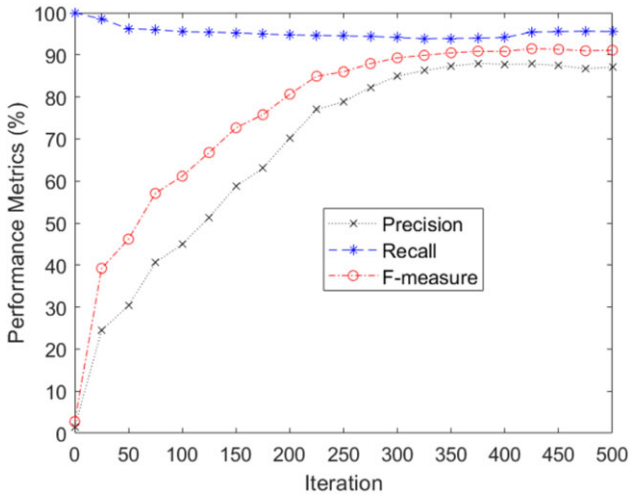

Fig. 12. Network performances on validation data during the training of output layer.

In Figure 12, CrackNet-R initially reaches nearly 100% Recall, while producing almost 0% Precision. In other words, most noncrack pixels are misclassified as crack pixels in the beginning. Later on, CrackNet-R progressively converges and yields high Recalls and Precisions concurrently. The optimal network performance on validation data is observed at the 425th iteration, quantified as Precision 87.83%, Recall 95.43%, and F-measure 91.47%. Figure 13 illustrates typical performances of the trained CrackNet-R. It is found that CrackNet-R is consistently efficient for varying asphalt surfaces with diverse types of cracks.

Table 2
Overall Precision, Recall, and F-measure of CrackNet, CrackNet-GRU, and CrackNet-R

Network	*Precision (%)*	*Recall (%)*	*F-measure (%)*
CrackNet	83.89	89.41	86.57
CrackNet-GRU	86.93	94.54	90.58
CrackNet-R	88.89	95.00	91.84

5 TESTING AND EVALUATION

The trained CrackNet-R is applied to the 500 testing images for further validation. The original CrackNet (Zhang et al., 2017b) and CrackNet-GRU are also evaluated for comparison. The CrackNet-GRU is trained identically with CrackNet-R via the optimizations of sequence modeling, sequence generation, and the output layer. The Precisions, Recalls, and F-measures achieved by CrackNet, CrackNet-GRU, and CrackNet-R for the 500 testing images are illustrated in Figure 14. Table 2 shows the overall Precisions, Recalls, and F-measures achieved by the three networks. The CrackNet-R behaves slightly better than CrackNet-GRU in terms of both Precision and Recall. The overall F-measure produced by CrackNet-R is 1.26% higher than the F-measure yielded by CrackNet-GRU. Compared with original CrackNet, CrackNet-R performs noticeably better in terms of either Precision or Recall. Particularly, the CrackNet-R yields tangibly higher Recall, indicating more cracks are detected by CrackNet-R. The improvement in terms of F-measure is over 5%, leading to many visible differences. Figures 15 and 16 show the

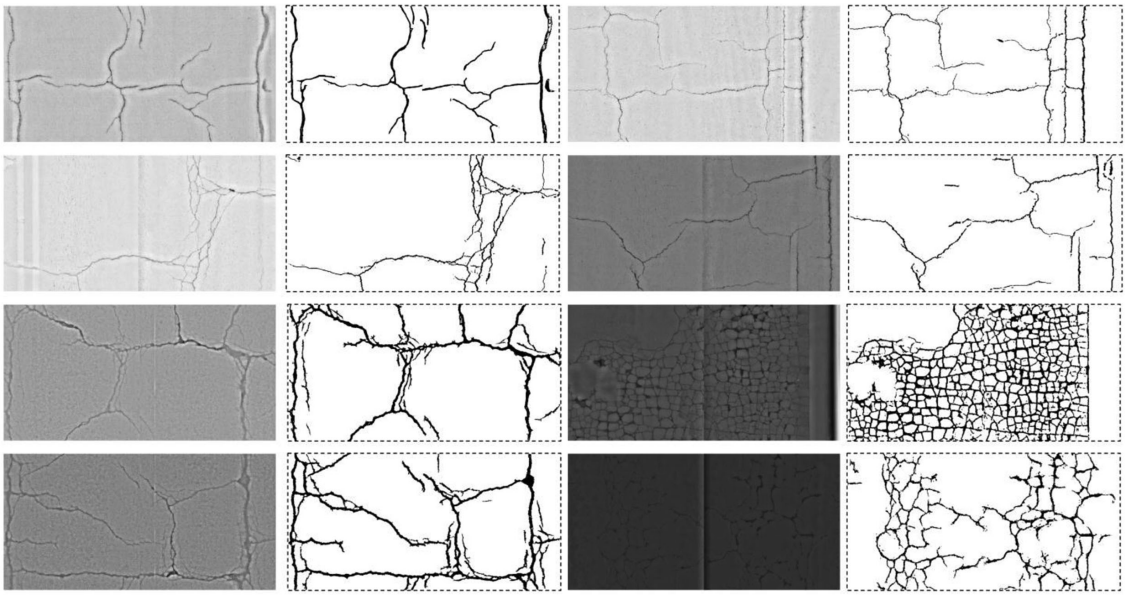

Fig. 13. Typical performances of CrackNet-R.

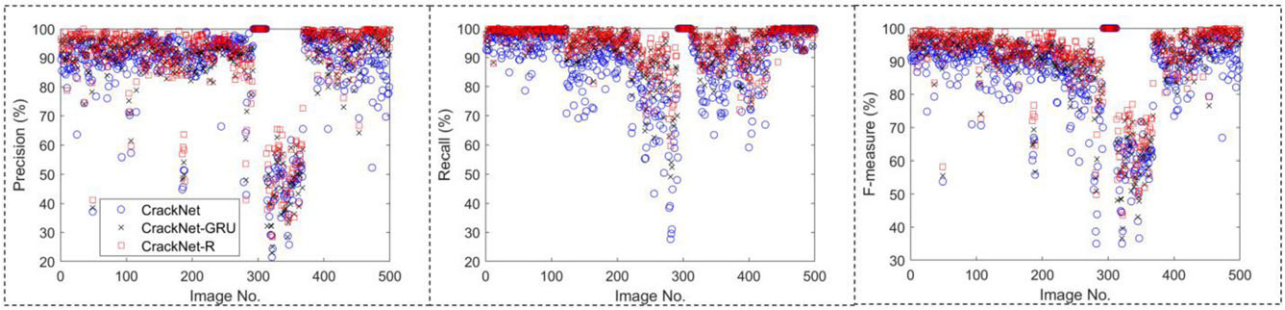

Fig. 14. Illustration of Precision, Recall, and F-measure of CrackNet, CrackNet-GRU, and CrackNet-R.

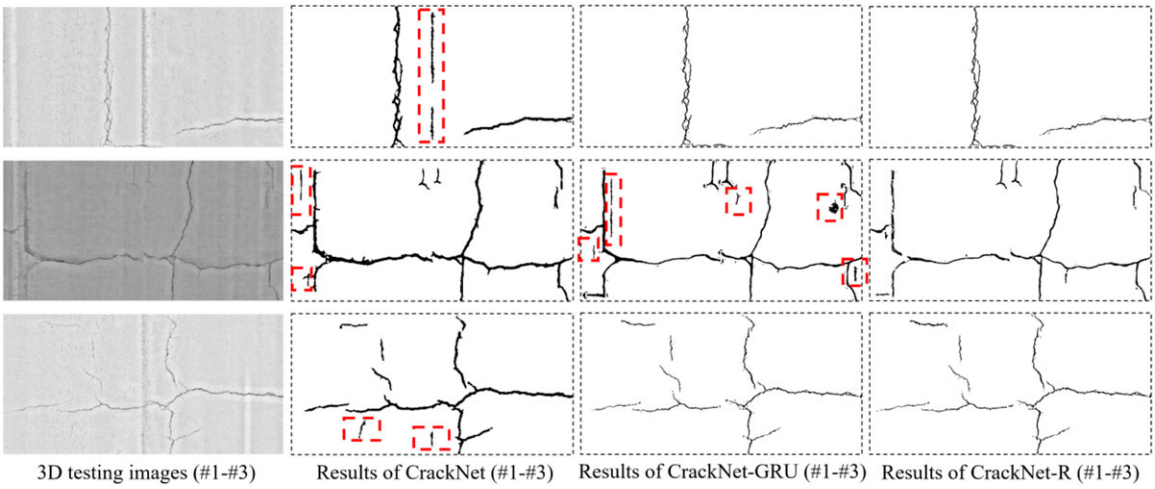

Fig. 15. Illustration of false-positive errors reduced by CrackNet-R.

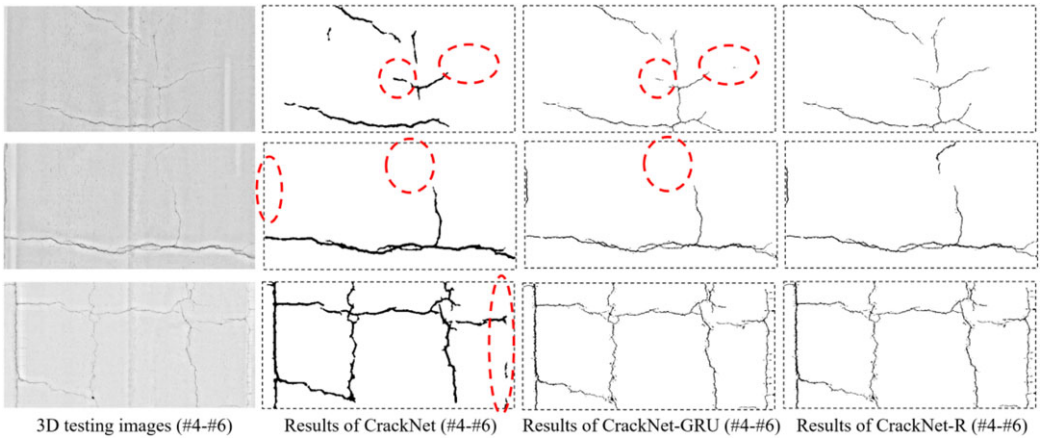

Fig. 16. Illustration of false-negative errors reduced by CrackNet-R.

typical improvements achieved by CrackNet-R, where the false-positive errors are highlighted in dashed rectangles, and the false-negative errors are highlighted in dashed circles. As shown in Figure 15, CrackNet-R has a stronger capability of suppressing noises. In Figure 16, CrackNet-R is also more robust than original CrackNet and CrackNet-GRU in detecting fine or hairline cracks. Table 3 shows the average processing speeds of CrackNet and CrackNet-R. CrackNet-R is found to be roughly four times faster than CrackNet. The efficiency of CrackNet-R reveals the advantages of sequence-based learning for pixel-level crack detection.

Table 3
Average processing time of CrackNet and CrackNet-R

Network	*Hardware device*	*Input image size*	*Processing time (seconds)*
CrackNet	Dual GPUs: Two NVidia GeForce GTX 1080 Ti	1,024 × 512	2.894
CrackNet-R			0.713

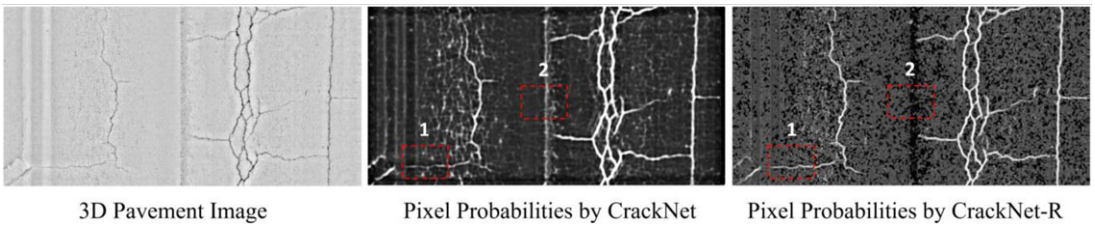

Fig. 17. Pixel probabilities produced by CrackNet and CrackNet-R.

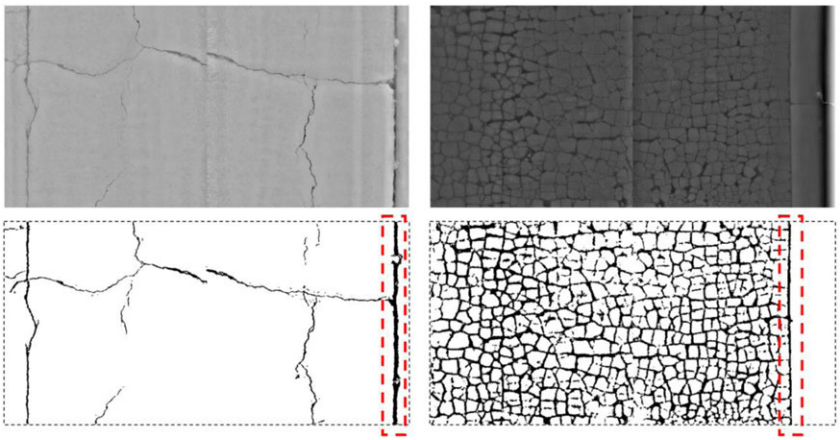

Fig. 18. Typical errors resulted from CrackNet-R.

6 DISCUSSION

Figure 17 illustrates the final pixel probabilities produced by CrackNet and CrackNet-R for a typical 3D pavement image. It is found that CrackNet-R yields similar pixel probabilities at all cracking areas. In Figure 17, pixel probabilities within region 1 show that CrackNet-R results in higher pixel probabilities for fine cracks in comparison with CrackNet. In addition, the outputs within region 2 also indicate that CrackNet-R yields significantly lower pixel probabilities at noise areas. The local receptive field used in CNN can include many unnecessary background pixels, which may result in distractions from patterns of interest. On the contrary, given a rational input sequence, the RNN can focus on the input sequence and determine the pattern type of the input sequence. Particularly, the dynamic change of internal memory of the RNN can further reflect some imperceptible differences between cracks and noise patterns. Such advantages of RNN may explain why the proposed CrackNet-R is more robust than CrackNet in detecting fine cracks and eliminating noises.

In Figure 18, similar to CrackNet, CrackNet-R also yields false-positive errors for cases of shoulder drop-off (highlighted on the first image) and pavement edge (highlighted on the second image). These noise patterns are very challenging to be excluded due to their complex similarities with pavement cracks. To distinguish these noise patterns, the network should have a strong understanding of the global context, a much needed future development for CrackNet-R. The sequence generator used in the article generates sequences of fixed length. Variable-length sequences potentially can yield substantially higher detection accuracy. Therefore, a future development can focus on generating variable-length sequences to fully follow the characteristics of

pavement cracks. Additionally, the sequence of pixels in the article is purely an oriented line segment with fixed length. If free-form paths with arbitrary lengths could be generated at each pixel, the gap between pavement cracks and noise patterns would be further augmented.

Finally, the proposed GRMLP in the article is not exclusive to crack detection. Instead, the proposed GRMLP is applicable to other problems as it consists of general-purpose operations to add new content and erase old memory.

7 CONCLUSIONS

This article describes a systematic approach to training a RNN called CrackNet-R for detecting pavement cracks on 3D asphalt surfaces with explicit requirement on pixel-perfect accuracy. A new recurrent unit, namely GRMLP, is proposed and developed in the article to operate the internal memory of CrackNet-R and conduct sequence modeling. Compared with the widely used LSTM and GRU, the proposed GRMLP presents many structural differences. The major difference is that the proposed GRMLP employs individual MLPs at most gating units to accomplish deeper abstractions on the inputs and hidden states. Consequently, the proposed GRMLP is capable of modeling highly nonlinear transitions from inputs to internal memories. With respect to pavement crack detection, it is shown in the article that the proposed GRMLP slightly outperforms LSTM and GRU by using only one more nonlinear layer at each gating unit.

CrackNet-R consists of three components: sequence generation, sequence modeling, and an output layer. Sequence generation is to generate a sequence of connected pixels at each pixel along the best orientation where the lowest average elevation is observed. Subsequently, the proposed GRMLP is adopted to implement sequence modeling and predict the timely probabilities of input sequences being crack patterns. Unique strategies are developed in the article to train sequence modeling such that the network can handle variable-length sequences. The output layer is specifically for pixel-perfect accuracy. The output layer produces pixel probabilities based on timely probabilities predicted for sequences.

The training of CrackNet-R is completed by optimizing sequence modeling, sequence generation, and the output layer in a serial manner with 3,000 training images. Based on the optimization of sequence modeling and sequence generation, the average modeling accuracy of CrackNet-R on 500 validation images is 94.98%, reflecting the efficiency of sequence processing. Finally, the output layer is also trained using Mini-Batch Stochastic Gradient Descent algorithm.

The experiment using 500 testing images shows that the overall Precision, Recall, and F-measure achieved by the trained CrackNet-R are 88.89%, 95.00%, and 91.84%, respectively, all better than those of CrackNet. The improvements in terms of Precision and Recall achieved by CrackNet-R are 5.00% and 5.59%, respectively. Such improvements indicate that CrackNet-R can detect more cracks, while suppressing noises more successfully. In addition, the CrackNet-R is roughly four times faster than CrackNet.

Compared with pavement crack detection methods using deep CNNs, such as the original CrackNet by the authors, CrackNet-R presents a fundamental change in methodology and emphasizes the concept of sequence. The efficiency of CrackNet-R reveals the potential of sequence-based learning in detecting pavement cracks at pixel level. It is anticipated that future solutions based on advanced sequence modeling can introduce substantial benefits for automated pavement crack detection.

ACKNOWLEDGMENTS

The study presented in the article was partially supported by the U.S. Federal Aviation Administration (FAA) Grant 13-G-013, and also by the National Natural Science Foundation of China (No. U1534203). The U.S. company WayLink Systems Corp. provided various resources in technical and data support for the presented work.

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. & 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.

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

Cha, Y. J., Choi, W., Suh, G. & Mahmoudkhani, S. (2017b), Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, *Computer-Aided Civil and Infrastructure Engineering*, https://doi.org/10.1111/mice.12334.

Cho, K., van Merrienboer, B., Bahdanau, D. & Bengio, Y. (2014), On the properties of neural machine translation: encoder-decoder approaches, *arXiv*, **1409**, 1259, 1–9.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014), Empirical evaluation of gated recurrent neural networks on sequence modeling, *arXiv*, **1412**, 3555, 1–9.

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

Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., Ebersbach, D., Stoeckert, U. & Gross, H. M. (2017), How to get pavement distress detection ready for deep learning? A systematic approach, in *Proceedings of the International Joint Conference on Neural Networks*, Anchorage, AK, 2039–47.

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

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

He, K., Zhang, X., Ren, S. & Sun, J. (2015a), Deep residual learning for image recognition, *arXiv*, **1512**, 3385, 1–12.

He, K., Zhang, X., Ren, S. & Sun, J. (2015b), Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, *arXiv*, **1502**, 01852, 1–11.

Hochreiter, S. & Schmidhuber, J. (1997), Long short-term memory, *Neural Computation*, **9**(8), 1735–80.

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., 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.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), Deep learning, *Nature*, **521**(7553), 436–44.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86**(11), 2278–2323.

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.

Liang, M. & Hu, X. (2015), Recurrent convolutional neural network for object recognition, in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, 3367–75.

Luo, W., Li, Y., Urtasun, R. & Zemel, R. (2016), Understanding the effective receptive field in deep convolutional neural networks, *arXiv*, **1701**, 4128, 1–9.

Milan, A., Rezatofighi, S. H., Dick, A., Schindler, K. & Reid, I. (2016), Online multi-target tracking using recurrent neural networks, *arXiv*, **1604**, 3635, 1–9.

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.

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

Panakkat, A. & Adeli, H. (2009), Recurrent neural network for approximate earthquake time and location prediction using multiple seismicity indicators, *Computer-Aided Civil and Infrastructure Engineering*, **24**(4), 280–92.

Parlos, A. G., Chong, K. T. & Atiya, A. F. (1994), Application of the recurrent multilayer perceptron in modeling complex process dynamics, *IEEE Transactions on Neural Networks*, **5**(2), 255–66.

Pascanu, R., Gulcehre, C., Cho, K. & Bengio, Y. (2014), How to construct deep recurrent neural networks, *arXiv*, **1312**, 6026, 1–13.

Puskorius, G. V. & Feldkamp, L. A. (1994), Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks, *IEEE Transactions on Neural Networks*, **5**(2), 279–97.

Rafiei, M. H., Khushefati, W. H., Demirboga, R. & Adeli, H. (2017), Supervised deep restricted Boltzmann machine for estimation of concrete, *ACI Materials Journal*, **114**(2), 237–44.

Salehinejad, H., Sankar, S., Barfett, J., Colar, E. & Valaee, S. (2018), Recent advances in recurrent neural networks, *arXiv*, **1801**, 01078, 1–20.

Schmidhuber, J. (2014), Deep learning in neural networks: an overview, *arXiv*, **1404**, 7828, 1–88.

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.

Tripathi, S., Lipton, Z. C., Belongie, S. & Nguyen T. (2016), Context matters: refining object detection in video with recurrent neural networks, *arXiv*, **1607**, 4648, 1–12.

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. & Smadi, O. (2011), Automated imaging technologies for pavement distress surveys, *Transportation Research Circular No. E-C156*, TRB, National Research Council, Washington DC.

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), 173–190.

Zhang, A., Wang, K. C. P. & Ai, C. (2017a), 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, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu Y., Li, J. Q. & Chen, C. (2017b), Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network, *Computer-Aided Civil and Infrastructure Engineering*, **32**(10), 805–19.

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*, **v2016**, 3708–12.