Engineering 7 (2021) 1786–1796

Contents lists available at ScienceDirect

Engineering

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

Research
Civil Engineering—Article

Real-Time Detection of Cracks on Concrete Bridge Decks Using Deep Learning in the Frequency Domain

Qianyun Zhang [a], Kaveh Barri [a], Saeed K. Babanajad [b], Amir H. Alavi [a,c,*]

[a] Department of Civil and Environmental Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA
[b] Wiss, Janney, Elstner Associates Inc., Northbrook, IL 60062, USA
[c] Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan 41354, China

ARTICLE INFO

Article history:
Received 18 April 2020
Revised 9 June 2020
Accepted 27 July 2020
Available online 19 November 2020

Keywords:
Crack detection
Concrete bridge deck
Deep learning
Real-time

ABSTRACT

This paper presents a vision-based crack detection approach for concrete bridge decks using an integrated one-dimensional convolutional neural network (1D-CNN) and long short-term memory (LSTM) method in the image frequency domain. The so-called 1D-CNN-LSTM algorithm is trained using thousands of images of cracked and non-cracked concrete bridge decks. In order to improve the training efficiency, images are first transformed into the frequency domain during a preprocessing phase. The algorithm is then calibrated using the flattened frequency data. LSTM is used to improve the performance of the developed network for long sequence data. The accuracy of the developed model is 99.05%, 98.9%, and 99.25%, respectively, for training, validation, and testing data. An implementation framework is further developed for future application of the trained model for large-scale images. The proposed 1D-CNN-LSTM method exhibits superior performance in comparison with existing deep learning methods in terms of accuracy and computation time. The fast implementation of the 1D-CNN-LSTM algorithm makes it a promising tool for real-time crack detection.

1. Introduction

According to the American Society of Civil Engineers (ASCE) report card [1], over 56 000 bridges in the United States are "structurally deficient." Economic and effective management of aging bridges is becoming a challenging task for the state departments of transportation and the government. Reliable condition evaluation of bridges is required before any decision-making on repairing and prevention. Although this concern motivates bridge owners to conduct frequent structural inspections, the application of conventional non-destructive evaluation (NDE) and visual inspection methods requires a point-by-point inspection of individual elements in order to discover the location of multiple defects, in addition to bridge closures due to inspectors' safety concerns. Furthermore, these methods are time-consuming, expensive, subjective, and highly dependent on inspectors' experience. Many research groups have been exploring alternative structural health monitoring (SHM) methods to deal with these limitations [2–4]. Although global SHM techniques seem promising, they are only capable of providing a coarse evaluation of structural behavior, and cannot provide detailed information [5,6]. In addition, interpreting the results collected by the SHM methods is a challenging task due to noise signals, sensor defects, or acquisition system settings.

In recent years, vision-based methods have been gaining more attention for civil infrastructure damage detection practices. A number of studies have been conducted to detect superficial defects such as cracks and corrosion. For example, segmentation [7], filtering [8,9], and stereovision-based methods [10] have been used to detect cracks and crack-like features in structural systems. Vision-based methods typically follow two steps to detect cracks [11]. In the first step, the image is filtered using a statistic filter, and crack features are locally extracted to fuse the image. The second step involves cleaning and linking the image segments to define the crack [11]. Shadow-removal algorithms have also been developed to remove the shadows from such images and pinpoint the crack [12,13]. However, bridge inspection data are collected in various situations and thus vary extensively. Issues such as noise caused by lighting conditions and distortion, dependence on prior knowledge, and the quality of image data are still challenging for reliable crack detection.

* Corresponding author.
E-mail address: alavi@pitt.edu (A.H. Alavi).

https://doi.org/10.1016/j.eng.2020.07.026

A viable solution to cope with these issues is to deploy machine learning (ML) methods. ML-based methods have been widely used in the areas of SHM and NDE [14–16]. These approaches are generally used to interpret the signal data collected from the testing systems; as a result, they are particularly used to provide useful information about the condition of structural systems. More recent efforts in this area have focused on integrating image feature extraction and ML techniques to develop novel SHM and NDE systems [17–19]. However, the use of over-extracted or false-extracted features often causes significant complexity in the model development. Convolutional neural networks (CNNs) can overcome this issue by extracting effective image features. CNNs are a class of deep learning algorithms inspired by the visual cortex of animals [20]. This method can efficiently capture the grid-like topology of images. It requires fewer computations due to sparsely connected neurons and the pooling process, which will shrink the image dimensions. Moreover, CNNs have shown reliable performance in differentiating among a large number of classes [21,26]. Due to their outstanding performance on image data, CNNs are becoming an efficient tool for the SHM systems [22–25]. Recently, Azimi and Pekcan [27] introduced a CNN approach for SHM that uses transfer learning techniques for compressed response data. Their CNN models were trained using acceleration response histories. The developed models were then validated by experimental data. Soukup and Huber-Mörk [28] proposed a CNN-based railway defect detection method. Cha et al. [29] proposed a deep learning crack damage detection system for concrete surfaces. Da Silva and de Lucena [30] developed a CNN image classifier for concrete crack detection. Although promising advancements have been made, the major downside of the existing deep learning-based SHM methods is their computational intensity and long training times. These limitations not only affect the methods' usefulness in practice, but also present a barrier to developing a real-time condition assessment system.

The present study presents a real-time deep learning approach for the detection of cracks on the bridge deck. To this aim, one-dimensional (1D)-CNN is integrated with an artificial recurrent neural network (RNN) architecture called long short-term memory (LSTM). The proposed 1D-CNN-LSTM algorithm is trained using images transferred into the frequency domain rather than the spatial domain. This strategy is adopted to reduce the computation time and improve the detection accuracy. The efficiency of the proposed system for real-time crack detection is discussed. This paper is organized as follows: Section 2 presents the details of the relevant deep learning methods. Section 3 provides an overview of the proposed approach and the database used for model calibration, and describes the preprocessing steps and network architecture. Section 4 provides the detection results and discusses the implementation of the proposed method on an unseen testing dataset. Section 5 provides a comparative study with existing deep learning methods and two commonly used edge detector methods. Further discussion and conclusions are presented in the last section.

2. Methods

2.1. Convolutional neural networks

Deep learning is a subset of ML that is capable of extracting higher level features from raw data using a multilayered artificial neural network structure. Deep learning is also known as deep neural learning or deep neural network. Among such techniques, CNN is the best-known deep learning architecture. CNNs are inspired by the biological processes of the animal visual cortex [31,32]. In visual processing, individual cortical neurons respond to specific respective fields. The respective fields of different neurons are then partially overlapped to cover the entire visual field. CNNs employ the mathematical operation convolution for general matrix multiplication in at least one of their layers [33]. A CNN usually consists of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer. Usually, the input layer is presented as a tensor of shape (number of images × image height × image width × image channels). Convolutional layers apply a number of filters to the local regions of inputs in order to extract feature maps of the images, with a shape (number of images × feature map height × feature map width × number of filters). For example, if the input consists of A images with size $M \times N$ pixels and C color channels, the shape of the input tensor will be $A \times M \times N \times C$. Supposing that the number of filters is k, the weight of filter i is W^i, b^i is the bias of filter i, x_s denotes the filter window patch, and a is the activation function (like a rectified linear unit (ReLU), sigmoid, and tanh), then the convolution of x_s, given filter i for each image, is defined as follows:

$$Z_{i,s} = a\left[\mathrm{sum}\left(W^i x_s\right) + b^i\right] \tag{1}$$

By sliding the filter window through each image with patch window size $f \times f \times C$ and stride size s (i.e., the filter moving steps in each direction) for all dimensions, the convolutional output is given size $N \times (\lfloor (M-f)/s+1 \rfloor) \times (\lfloor (N-f)/s+1 \rfloor) \times k$, where f is filter size, C is image channel number, $(\lfloor \cdot \rfloor)$ is the round-down function. Fig. 1 depicts the convolution processes of convolutional layers.

Convolutional layers usually have a pooling layer. The pooling layers are applied to reduce the dimension of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer [20]. The pooling process can compute the maximum or average, depending on the expectation of the outputs. Max pooling computes the maximum value from each local cluster of the previous layer and passes it to the next layer. Average pooling passes the mean value of the local cluster of the previous layer to the next layer. For example, max pooling of the x_s patch can be denoted as follows:

$$\mathrm{pool}_s = \max(x_s) \tag{2}$$

According to previous studies [34], max pooling provides better performance than average pooling on image datasets. The last layers of a CNN are fully connected layers, which compute the class scores. Fully connected layers connect every neuron in one layer to every neuron in another layer. The flattened matrix goes through a fully connected layer to classify the images. Widely used CNN structures normally consist of input layers followed by several convolutional layers, pooling layers, fully connected layers, and output layers [35–37]. Fig. 2 shows an example of a CNN architecture.

Depending on the convolution dimension and direction, there are 1D, two-dimensional (2D), and three-dimensional (3D) CNNs. 1D-CNNs conduct convolution calculation in one dimension (along one axis), while 2D- and 3D-CNNs calculate convolutional values in two and three directions, respectively. In this study, a 1D-CNN is applied to extract and learn features of the flattened image frequency signals. Fig. 3 shows a simple example of 1D convolution.

2.2. Long short-term memory

RNNs are a class of deep learning that is suitable for sequential data. General RNNs have short-term memory issues. If a sequence is very long, it will be difficult for RNNs to carry information from the earlier steps to the later steps. In other words, if the processing signal is very long, RNNs may lose some important information from the beginning. Moreover, RNNs suffer from the vanishing

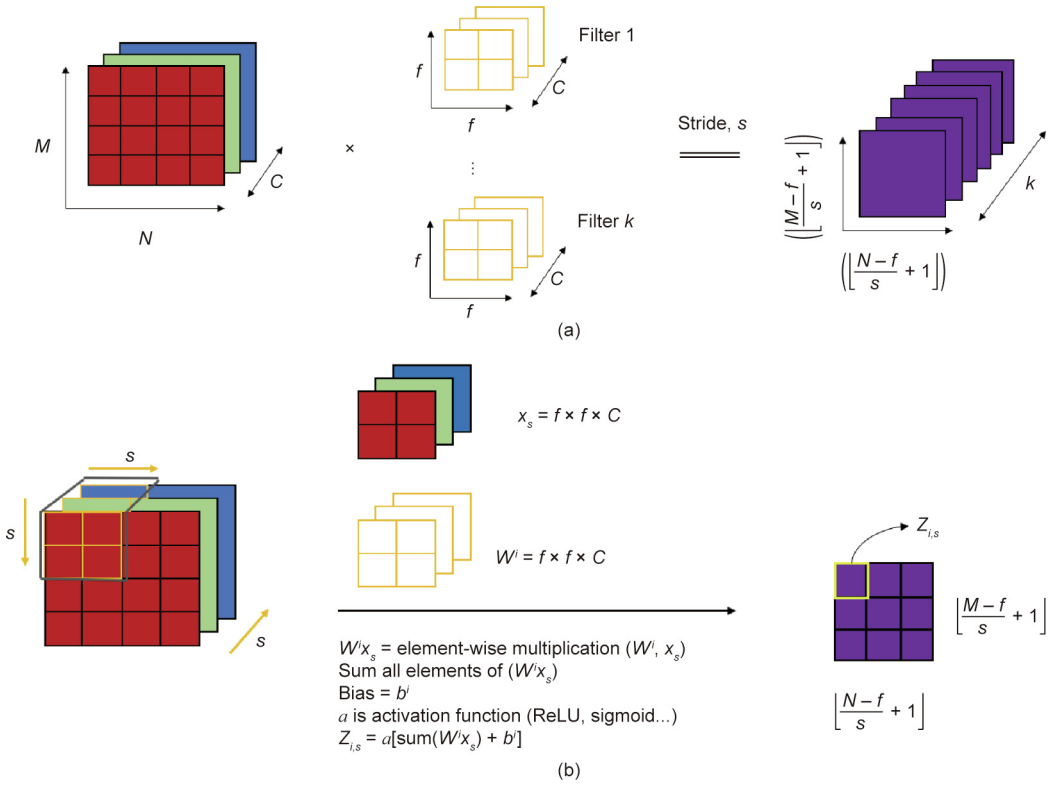

Fig. 1. Details of the convolutional layers. (a) Overall convolution process for each image through convolutional layer; (b) detailed convolution process for each individual filter.

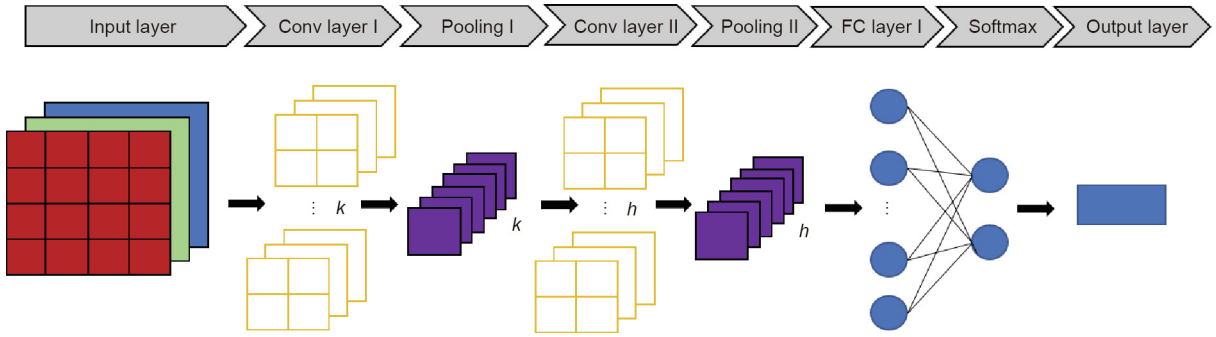

Fig. 2. An example of a CNN architecture. *k* and *h* are the number of filters in different layers; Conv means convolutional.

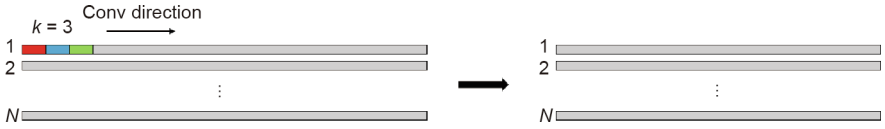

Fig. 3. 1D convolution for *N* examples.

gradient problem during back propagation. For example, if the gradient value becomes extremely small, the learning process would not be improved significantly. In typical RNNs, small gradients can stop the layers' learning process. To solve these issues, an LSTM approach [38] can be used. LSTM is an invariant RNN architecture with internal gates that can be used to regulate the flow of information. LSTM consists of three thresholding structures to filter out empty inputs and redundant information and fuse similar information. Fig. 4 shows the working mechanism of LSTM. C_{t-1} and C_t are the cell states in the sequence, which act as conveyor belts to pass on the information. Three gates are designed to add information to or remove it from the cell state. Gates are composed of a sigmoid neural net layer and a pointwise multiplication operation. The output of the sigmoid layer is between 0 and 1 to describe how much of each component should be passed through. The first gate is called the "forget gate," and decides what information needs to be removed. For example, if the output number equals 0, this component should be completely removed [38]. Eq. (3) shows the calculation. The second gate is used to decide what information needs to be added to the cell state. This consists of the updated output of the sigmoid layer and a new candidate created by a tanh layer. The combination of these two is added to

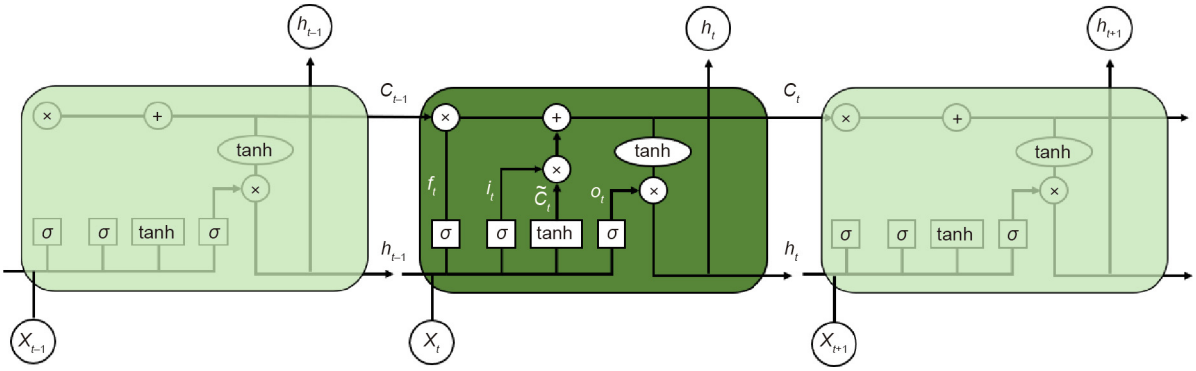

Fig. 4. Mechanism of LSTM.

update the state, as shown in Eqs. (4)–(6). The last step is to decide what information needs to be output. The cell state is passed through a sigmoid layer and a tanh layer to generate the final output, as presented in Eqs. (7) and (8).

$$f_t = \sigma\left(W_f[h_{t-1}, X_t] + b_f\right) \tag{3}$$

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i) \tag{4}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, X_t] + b_C) \tag{5}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{6}$$

$$o_t = \sigma(W_o[h_{t-1}, X_t] + b_o) \tag{7}$$

$$h_t = o_t \tanh(C_t) \tag{8}$$

where f_t and i_t are the sigmoid layer outputs; $\tilde{C}_t$ is the state that needs to be added; $\sigma()$ is the sigmoid function; C_t is the current state; o_t is the sigmoid output of the cell state; h_t is the output of the current time step; W_f, W_i, W_C, and W_o are the layer weights; h_{t-1} and X_t are the output of the previous time step and the input of the current step, respectively; and b_f, b_i, b_C, and b_o are the bias terms [38]. The LSTM algorithm has been applied to deal with long sequence signals such as natural languages [39], heart rate signals [40], and speech signals [41].

3. Proposed method for the real-time detection of concrete cracks

As discussed in the previous section, CNNs are powerful for feature extraction. However, feature fusion plays a more important role in overall model performance. Fully connected layers are widely used to simply combine extracted features with adjusted weights. Even though fully connected layers have decent performance on feature hybridizing, they are often insufficient to extract high-level information. To tackle this issue, this study integrates 1D-CNN with LSTM as a feature fusion layer. LSTM has proven to be efficient in hybridizing features for the long-term dependence of sequential data [42–48]. In bridge inspection, the input concrete deck images are single images captured at a certain moment rather than in a sequence. Therefore, in this study, LSTM is applied to the feature level instead of the input level. During the 1D-CNN feature extraction stage, the convolutional kernels are used to scan the entire image frequency vector to extract the features of all the objects in the image. Setting the kernel moving strides to be smaller than the kernel size itself ensures that there will be overlapping parts for each scanning area. Therefore, the feature blocks extracted by the convolutional kernels are strongly dependent on each other, and can be treated as sequential data inputs for the LSTM layer to conduct feature fusion. A fully connected layer is then applied to further fuse sufficient features obtained from the previous layers. Fig. 5 presents the framework of the proposed method for the real-time detection of concrete cracks on bridges. As shown in this figure, the first step is to collect a database that includes thousands of images of cracked and non-cracked concrete bridge decks. The database is then divided into training, validation, and testing subsets. The training and validation datasets are passed to the preprocessing stage, where the images are transformed into the frequency domain. Arguably, the edge shape of the surface cracks corresponds to high frequencies. Therefore, a high-pass filter (HPF) is applied to filter out the low frequencies corresponding to the background. After filtering, the image frequency matrices are flattened into vector frequency signals. These vectors are used to train the proposed 1D-CNN-LSTM algorithm. The developed method is applied to test images by conducting a sliding window through the whole image. The local window with cracks is kept in the output image.

3.1. Database

The 1D-CNN-LSTM models are developed using a database containing 4800 images of manually labeled cracked and non-cracked concrete bridge decks [54]. The database includes cracks as narrow as 0.06 mm and as wide as 25 mm. The size of the images is 256 × 256 pixels. In order to improve the detection accuracy, images are broken into sub-images with 64 × 64 pixels. Out of the 4800 available images, 4300 images are cropped into 17 200 small images. Images that are blurry or that include corner cracks are eliminated. Finally, 16 789 images are kept as the datasets for this study. The remaining 500 bridge deck images are randomly stitched into 20 images sized 1280 × 1280 pixels for testing the generalization capacity of the developed classifier.

3.2. Preprocessing of data in the frequency domain

In many studies, images are processed in the spatial domain. That is to say, images are processed as they are, without further preprocessing. In the spatial domain, the values of the pixels change with respect to the scene, and image processing is based on the pixel values. An arguably more efficient approach to process images is to transform them into the frequency domain [49]. Images can be transformed from the spatial domain to the frequency domain by conducting a discrete Fourier transform (DFT). In the frequency domain, the value and location are represented by sinusoidal relationships that depend upon the frequency of a pixel occurring within an image. In this domain, a pixel location

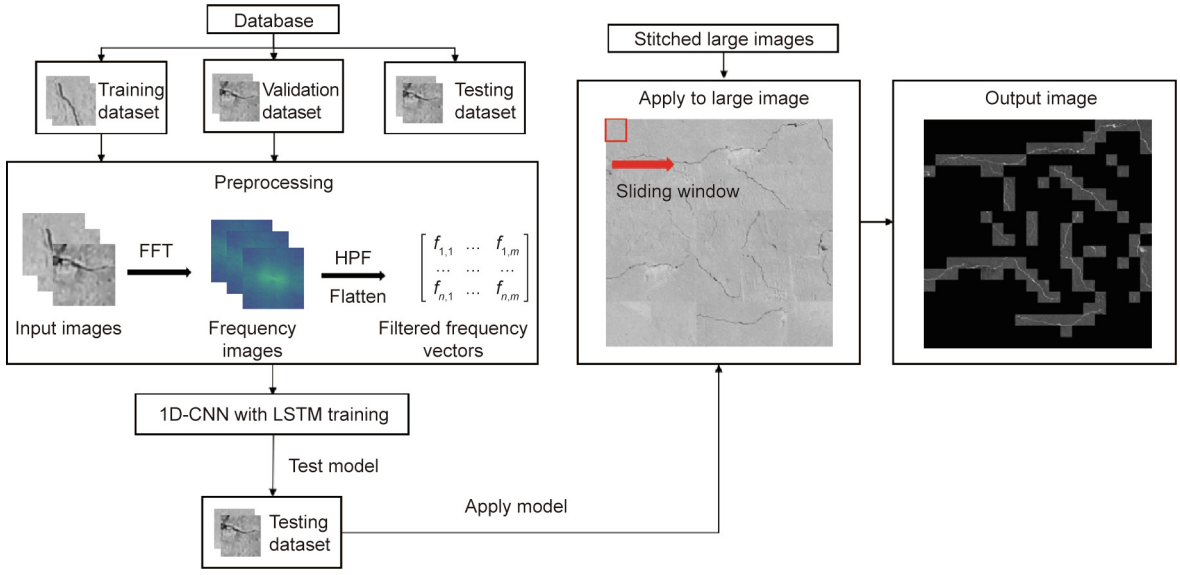

Fig. 5. The framework of the proposed method. FFT: fast Fourier transform.

is represented by its x- and y-frequencies, and its value is represented by an amplitude. Images can be transformed to the frequency domain in order to determine which pixels contain more important information and whether repeating patterns occur. In other words, in the frequency domain, we deal with the rate at which the pixel values are changing in the spatial domain. Since the frequencies of images relate to the pixel value changing rate, the images' frequency components are divided into two parts: high-frequency components, which correspond to edges in an image; and low-frequency components, which correspond to smooth regions. Many researchers have applied this property to image filtering, compression, and reconstruction [50–52]. The DFT is a sampled Fourier transform; therefore, it does not contain all the frequencies forming an image, but only contains a set of samples that is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, so the images in the spatial and Fourier domains are of the same size. For an image with size $M \times N$, the 2D DFT of the image can be presented as follows:

$$F(k,\ l) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,\ y)\exp\left[-i2\pi\left(\frac{kx}{M}+\frac{ly}{N}\right)\right] \tag{9}$$

where $f(x,\ y)$ is the image pixel in the spatial domain, and $\exp\left[-i2\pi\left(\frac{kx}{M}+\frac{ly}{N}\right)\right]$ is the basis function corresponding to each point $F(k,\ l)$ in the frequency domain. The equation can be interpreted as follows: The value of each point $F(k,\ l)$ is obtained by multiplying the spatial image with the corresponding base function and summing the result.

On the other hand, images in the frequency domain can be transformed back into the spatial domain. The inverse Fourier transform is given by the following equation:

$$f(x,\ y) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F(k,\ l)\exp\left[i2\pi\left(\frac{kx}{M}+\frac{ly}{N}\right)\right] \tag{10}$$

For large images, a fast Fourier transform (FFT) usually reduces the dimension complexity and computation time. The FFT produces complex values, which include the real and imaginary part or the magnitude and phase. In image processing, only the magnitude is usually displayed, because it contains most of the geometric structure information of the spatial domain. The magnitude and phase can be presented as follows:

$$|F(k,\ l)| = \sqrt{\mathrm{Re}(k,\ l)^2 + \mathrm{Im}(k,\ l)^2} \tag{11}$$

$$\phi(k,\ l) = \tan^{-1}\left[\frac{\mathrm{Im}(k,\ l)}{\mathrm{Re}(k,\ l)}\right] \tag{12}$$

where $|F(k,\ l)|$ is the magnitude, $\phi(k,\ l)$ is the phase, and $\mathrm{Re}(k,\ l)$ and $\mathrm{Im}(k,\ l)$ are the real part and imaginary part of the FFT output. $F(k,\ l)$ has a low frequency at the corners of the image but high-frequency areas in the center, which is inconvenient to interpret. Thus, the zero frequency will usually be shifted to the center. Fig. 6 shows the shift of the frequency image center [53].

Converting images into the frequency domain can significantly speed up the CNN training. Training the network with raw images, as is done in the spatial domain by 2D convolution, is not an efficient solution for bridge deck data. Instead, the training is performed in the frequency domain, which is more like 1D signal pattern recognition. Therefore, in the preprocessing stage, images in RGB channels are read into a grayscale image matrix first. Then, FFT is applied to each image in the image matrix. The center of the frequency images is shifted, and the magnitude of each image is calculated. Fig. 7 demonstrates the significant difference between the frequency distribution of images with cracks and without cracks. As shown in Fig. 7, the frequency spectrum has line-shaped sparks for the image with cracks but does not have such sparks for the image without cracks. Low frequencies corresponding to smooth background regions are eliminated using an HPF. Obviously, the main difference between the crack and non-crack images is in the high-frequency domain. A frequency filter ratio equal to 0.5 means that only the top 50% high frequencies are kept. Ratios from 0.2 to 0.7 are tested to obtain the best performance. Finally, 0.5 is selected to define the threshold. Any frequency higher than the threshold is kept, and all low frequencies are replaced with 0. After filtering, a frequency amplitude matrix is calculated for each image and flattened to a frequency amplitude vector. Fig. 8 shows the original and filtered amplitudes of crack and non-crack images. The preprocessed image data are reshaped into a matrix with shape (number of images $\times$ length of flattened

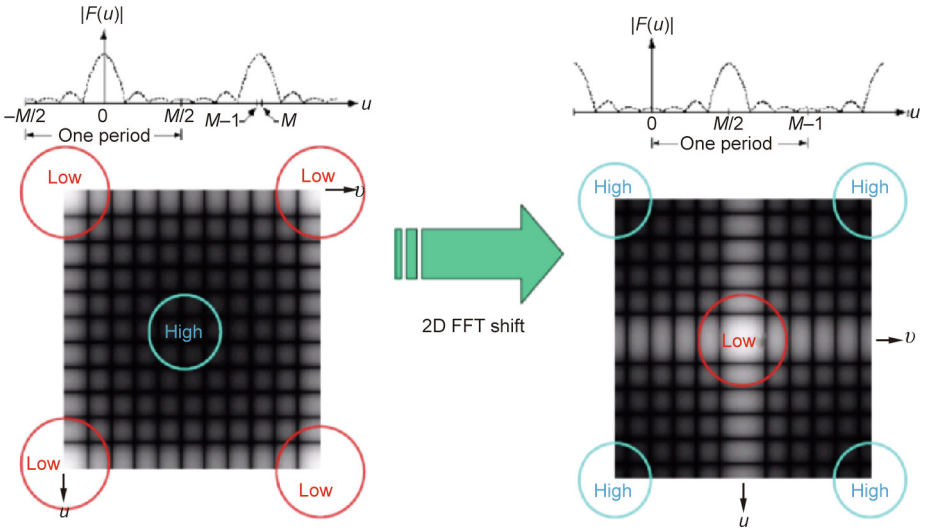

Fig. 6. A 2D image frequency center shift, where u and v are spatial frequency.

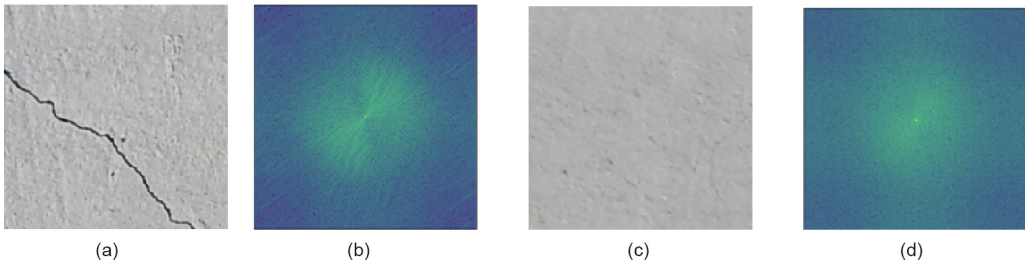

Fig. 7. (a) Crack image; (b) frequency spectrum for crack image; (c) non-crack image; (d) frequency spectrum for non-crack image.

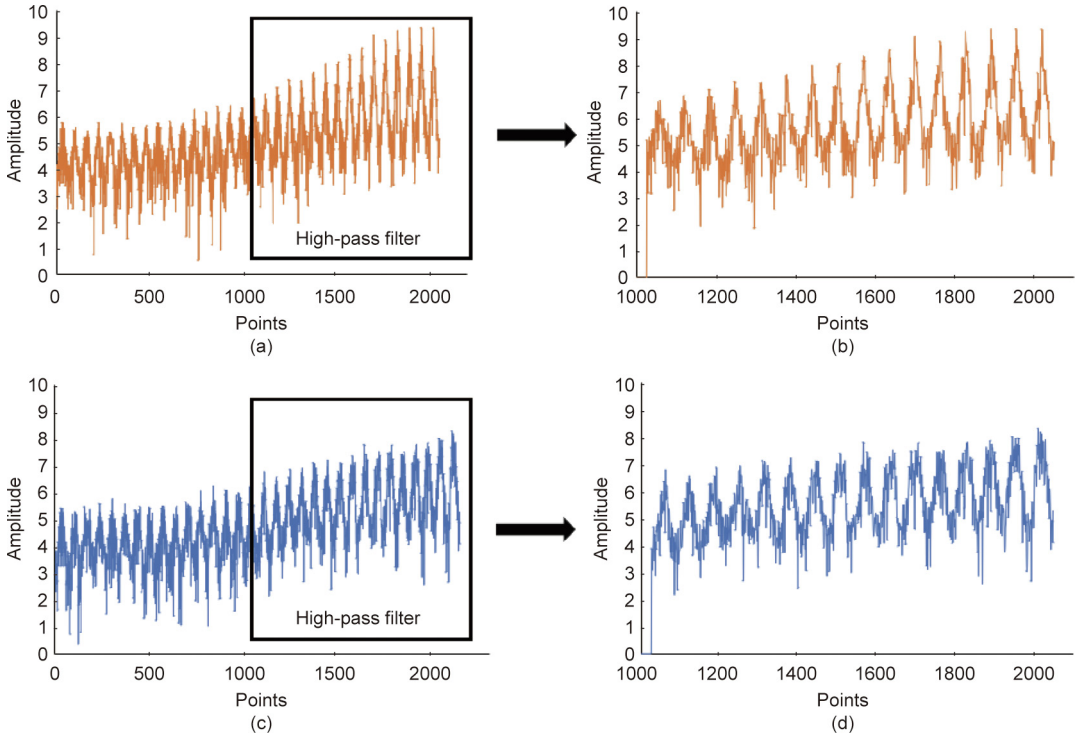

Fig. 8. (a) Amplitude for crack image; (b) filtered amplitude for crack image; (c) amplitude for non-crack image; (d) filtered amplitude of non-crack image.

frequency vector × 1). The reshaped data are used to develop the 1D-CNN with LSTM as the input layer.

3.3. The 1D-CNN-LSTM model architecture

The optimal 1D-CNN-LSTM architecture is selected via an extensive trial-and-error approach and developed using TensorFlow modules [55]. The optimal network consists of an input layer, four sets of convolutional layers followed by a max pooling layer, an LSTM layer, two fully connected layers, and an output layer. For each convolutional layer, batch normalization is applied. The ReLU function is used as the activation function for the convolutional layers. Sigmoid and softmax functions are considered as the activation functions for the first and second fully connected layers, respectively. The total network parameters are 230 082, and the trainable parameters are 229 378. For the training process, a stochastic gradient descent is chosen as the optimizer with a minibatch size of 32 out of 16 789 images. An adaptive and logarithmically decreasing learning rate is adopted to speed up the convergence. The initial learning rate and weight decay are 0.1 and 0.0001, respectively. A momentum value of 0.9 is applied to avoid overfitting. Furthermore, dropout is added after the LSTM layer and first fully connected layer. The dropout ratio is set to 0.5. Early stopping is also added to the network, which stops training when the model performance does not improve significantly. Table 1 shows the architecture of the proposed 1D-CNN-LSTM model.

4. Crack-detection results

4.1. Performance analysis

The ratio of the crack to non-crack images in the database is 1:2. The preprocessed database of 16 789 manually labeled images is randomly divided into the training, validation, and testing sets with respective percentages of 70%, 15%, and 15%. Fig. 9 shows the loss and accuracy for the training and validation procedure. The proposed method yields high accuracy on the training and validation data. The maximum training and validation accuracy are, respectively, 99.05% and 98.9%, achieved at the 49th and 40th epochs. The testing accuracy is 99.25%. The best model is saved to be applied to the unseen testing dataset. The simulations are conducted on a desktop computer with Intel® Xeon® CPU E5-1650 v4 @ 3.60 GHz, NVIDIA Quadro K420 GPU, and 31.9 GB RAM. The total training time is 1 h 12 min 4 s.

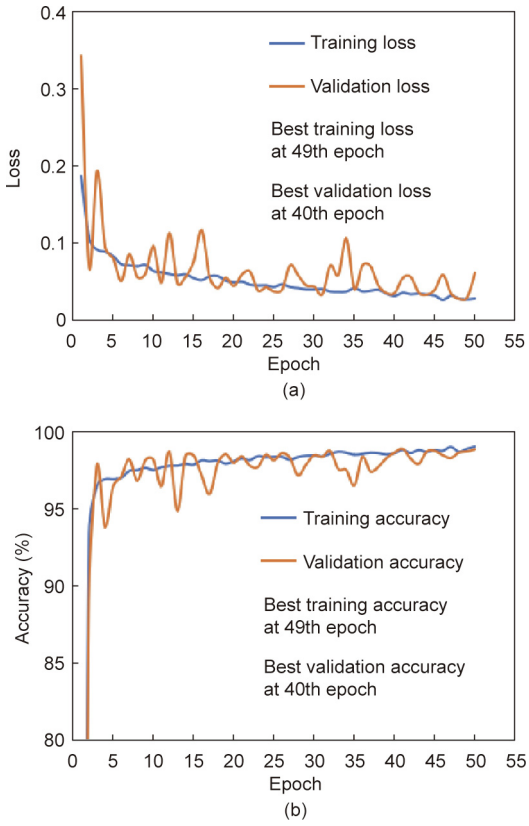

Fig. 9. (a) Loss and (b) accuracy of the developed 1D-CNN-LSTM model.

4.2. Implementation of the 1D-CNN-LSTM model

An implementation running code is developed in Python 3.7 for concrete bridge crack detection. Following the procedure described in Section 3, large-scale images are broken into small image groups with a size of 64 × 64 pixels. Each image group is transformed into the frequency domain and filtered with the same scale HPF. The developed model is then applied as a local window sliding through each image group to classify small images in the group as crack or

Table 1
The architecture of the 1D-CNN-LSTM model.

Layer (type)		Output shape	Number of parameters	Filter size	Number of filters	Stride	Activation function
Layer 1	conv1d_1 (Conv1D)	(Input#, 4096, 32)	128	3	32	1	–
	BN_1 (Batch Normalization)	(Input#, 4096, 32)	128	–	–	–	–
	activation_1 (Activation)	(Input#, 4096, 32)	0	–	–	–	ReLU
	Maxpooling_1 (MaxPooling)	(Input#, 2047, 32)	0	4	–	2	–
Layer 2	conv1d_2 (Conv1D)	(Input#, 2047, 64)	6 208	3	64	1	–
	BN_2 (Batch Normalization)	(Input#, 2047, 64)	265	–	–	–	–
	activation_2 (Activation)	(Input#, 2047, 64)	0	–	–	–	ReLU
	Maxpooling_2 (MaxPooling)	(Input#, 511, 64)	0	4	–	4	–
Layer 3	conv1d_3 (Conv1D)	(Input#, 511, 128)	24 704	3	128	1	–
	BN_3 (Batch Normalization)	(Input#, 511, 128)	512	–	–	–	–
	activation_3 (Activation)	(Input#, 511, 128)	0	–	–	–	ReLU
	Maxpooling_3 (MaxPooling)	(Input#, 127, 128)	0	4	–	4	–
Layer 4	conv1d_4 (Conv1D)	(Input#, 127, 128)	49 280	3	128	1	–
	BN_4 (Batch Normalization)	(Input#, 127, 128)	512	–	–	–	–
	activation_4 (Activation)	(Input#, 127, 128)	0	–	–	–	ReLU
	Maxpooling_4 (MaxPooling)	(Input#, 31, 128)	0	4	–	4	–
Layer 5	lstm_1 (LSTM)	(Input#, 128)	131 584	–	–	–	–
Layer 6	dense_1 (Dense)	(Input#, 128)	16 512	–	–	–	Sigmoid
Layer 7	dense_2 (Dense)	(Input#, 2)	258	–	–	–	Softmax

non-crack images. In order to obtain more continuous cracks, an overlapped local sliding window is applied. Fig. 10 provides a brief description of the sliding process. As shown in the Fig. 11, the detected cracks are more continuous after using the overlapped sliding window. Small images with cracks are kept as the output and restored to their original location in the large-scale image. All other small images without cracks are eliminated. Fig. 12 shows the crack-detection implementation framework. As discussed before, 500 images sized 256 × 256 pixels are stitched into 20 large images sized 1280 × 1280 pixels to test the generalization of the trained model. For each image, the time for output generation is merely 5–7 s. Fig. 13 presents the crack-detection results for two of the tested images. The implementation accuracy is calculated as follows:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \times 100\% \tag{13}$$

$$\mathrm{ER} = 100\% - \mathrm{ACC} \tag{14}$$

where ACC and ER are the accuracy percentage and error rate, respectively; TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives, respectively. Referring to Fig. 13, the implementation accuracies are 98.5% and 97.75%; accordingly, the error rates are 1.5% and 2.25%.

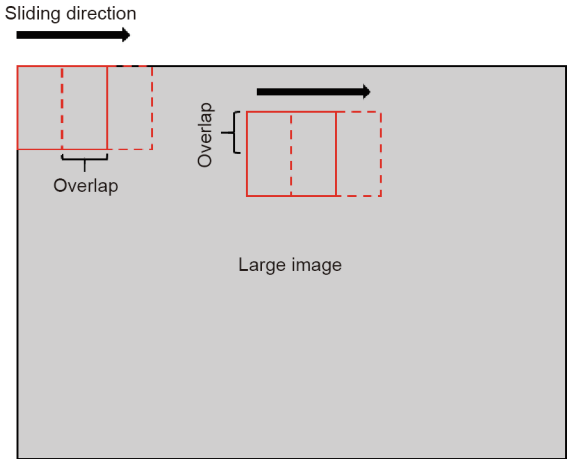

Fig. 10. A schematic representation of the overlapped sliding window.

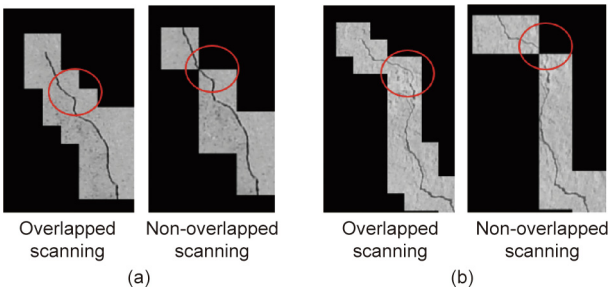

Fig. 11. Detected cracks using the overlapped and non-overlapped windows.

5. Comparative study

In order to provide insight into the implementation speed and accuracy of the proposed 1D-CNN-LSTM method, it was compared with existing deep learning methods and two commonly used edge detectors: namely, the Canny [56] and Sobel [57] detectors. Almost all existing studies in the area of deep learning for crack detection are based on the use of classic CNN structures and training images in the spatial domain [29,30]. Therefore, a 2D-CNN network similar to the model recently proposed by Cha et al. [29] was also developed in this study. The same database was used for the training, validation, and testing of the 2D-CNN method. The details of the 2D-CNN architecture are shown in Table 2. The total parameters and trainable parameters are 896 770 and 896 322, respectively. Table 3 summarizes the comparative study results. As seen in this table, the proposed method not only provides better detection performance, but also requires significantly fewer total and trainable parameters. The most important observation here is that the computation cost of the 2D-CNN method is about 129% and 710% higher than the proposed 1D-CNN-LSTM approach for the calibration and implementation phases, respectively. The slow processing speed of the 2D-CNN and other existing deep-learning-based methods pertains to the fact that they deal with images in the spatial domain. Also, applying the HPF in the preprocessing stage reduces redundant data information, leading to faster processing. The fast implementation of 1D-CNN-LSTM makes it ideal for the real-time detection of cracks on bridge decks. Furthermore, Fig. 14 shows a simple comparison of the 1D-CNN-LSTM, Canny, and Sobel edge detectors on one of the testing samples. As seen in this figure, 1D-CNN-LSTM notably outperforms the edge

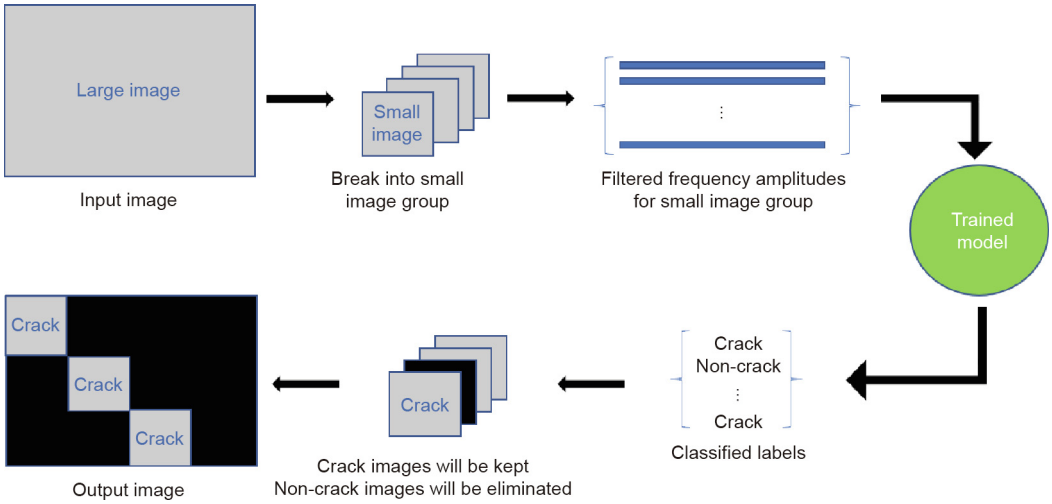

Fig. 12. The 1D-CNN-LSTM implementation process for concrete crack detection.

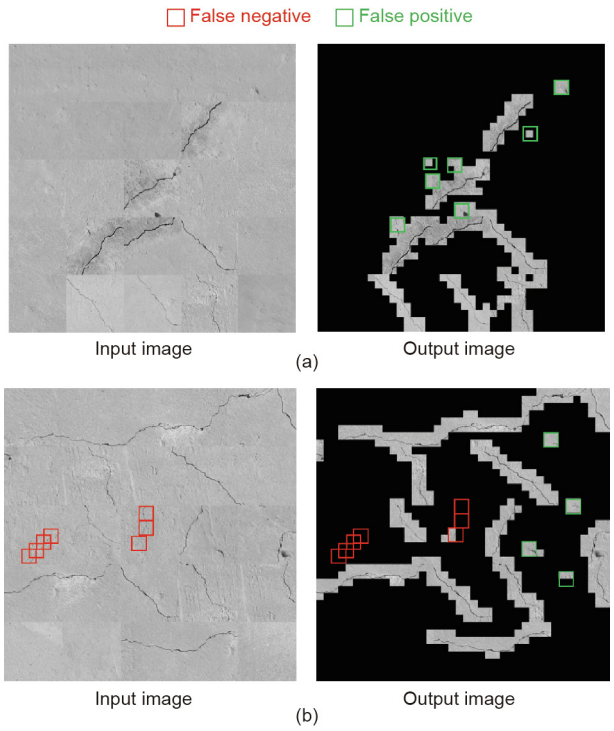

Fig. 13. Crack-detection results for two of the tested images.

detecting methods. In fact, the Canny method can hardly detect any cracks, due to the uneven concrete surface, while the partially detected cracks by the Sobel method are heavily affected by the noisy background.

6. Conclusions

In this study, a new concrete crack-detection method was presented that integrates 1D-CNN, LSTM, and a learning process in the image frequency domain. Thousands of images of cracked and non-cracked concrete bridge decks were used to train, validate, and test the proposed 1D-CNN-LSTM algorithm. The accuracy of the 1D-CNN-LSTM model was found to be 99.05%, 98.90%, and 99.25%, respectively, on the training, validation, and testing datasets. The implementation framework was able to successfully deploy the well-trained model for the detection of cracks on unlabeled large-scale images with high accuracy. Although the rate of false positives and negatives was satisfactory in the implementation, the performance of the model can still be improved by including more training data. The proposed method was compared with an existing deep-learning-based method and two edge detectors. The comparison results showed that the performance

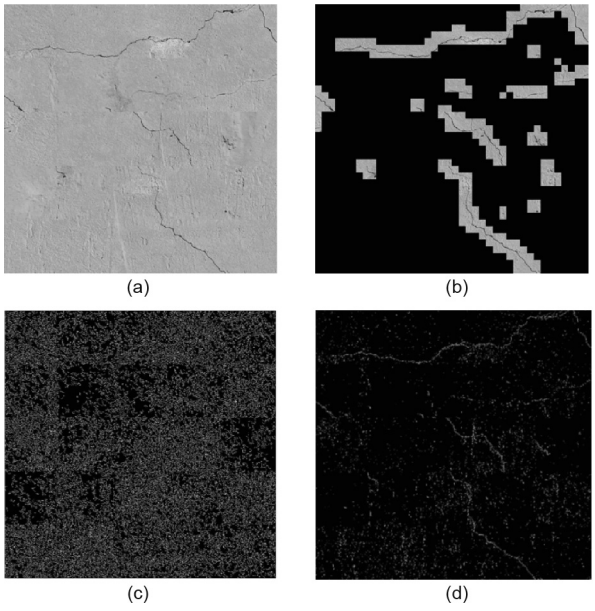

Fig. 14. (a) Original image; (b) output by the proposed 1D-CNN-LSTM method; (c) output by the Canny edge detector; (d) output by the Sobel edge detector.

Table 2
The architecture of the classical 2D-CNN network for crack detection.

Layer (Type)		Output shape	Number of parameters	Filter size	Number of filters	Stride	Activation function
Layer 1	conv2d_1 (Conv2D)	(Input#, 64, 64, 32)	320	3	32	1	–
	BN_1 (Batch Normalization)	(Input#, 64, 64, 32)	128	–	–	–	–
	activation_1 (Activation)	(Input#, 64, 64, 32)	0	–	–	–	ReLU
	Maxpooling_1 (MaxPooling)	(Input#, 63, 63, 32)	0	2	–	1	–
Layer 2	conv2d_2 (Conv2D)	(Input#, 63, 63, 64)	18 496	3	64	1	–
	BN_2 (Batch Normalization)	(Input#, 63, 63, 64)	265	–	–	–	–
	activation_2 (Activation)	(Input#, 63, 63, 64)	0	–	–	–	ReLU
	Maxpooling_2 (MaxPooling)	(Input#, 31, 31, 64)	0	2	–	2	–
Layer 3	conv2d_3 (Conv2D)	(Input#, 31, 31, 128)	73 856	3	128	1	–
	BN_3 (Batch Normalization)	(Input#, 31, 31, 128)	512	–	–	–	–
	activation_3 (Activation)	(Input#, 31, 31, 128)	0	–	–	–	ReLU
	Maxpooling_3 (MaxPooling)	(Input#, 7, 7, 128)	0	4	–	4	–
	Flatten ()	(Input#, 6272)	0	–	–	–	–
Layer 4	dense_1 (Dense)	(Input#, 128)	802 944	–	–	–	Sigmoid
Layer 5	dense_2 (Dense)	(Input#, 2)	258	–	–	–	Softmax

Table 3
A comparison of the 2D-CNN and 1D-CNN-LSTM methods.

Method	Total parameters	Trainable parameters	Training accuracy	Validation accuracy	Testing accuracy	Training time	Implementation output time
Standard 2D-CNN	896 770	896 322	98.52%	98.12%	97.80%	2 h 45 min 16 s	38–59 s per image
1D-CNN-LSTM	230 082	229 378	99.05%	98.90%	99.25%	1 h 12 min 4 s	5–7 s per image

of 1D-CNN-LSTM is superior to the compared methods. One of the significant advantages of the 1D-CNN-LSTM method over other studied CNN algorithms is its significantly faster training and implementation time. This issue is critical for real-time concrete crack detection, especially for autonomous bridge inspection using unmanned aerial vehicles. This observation clearly validates the efficiency of data preprocessing in the frequency domain. Future research can focus on developing models using larger and more diverse datasets. Integrating other efficient deep learning methods such as Yolo with LSTM in the frequency domain can also be a suitable topic for future research.

Ethical statement

Authors state that the research was conducted according to ethical standards.

This research does not involve human participants and/or animals. I confirm that this work is original and has not been published elsewhere. This submission is approved by all the authors and, if accepted, will not be published elsewhere in the same form, in English or in any other language, without the written consent of the copyright holder.

Acknowledgments

The research reported on in this paper was conducted under a project sponsored by the Impactful Resilient Infrastructure Science and Engineering (IRISE) public/private research consortium. At the time of publication, the consortium included the Pennsylvania Department of Transportation, the Federal Highway Administration (*ex officio*), Allegheny County, the Pennsylvania Turnpike Commission, Golden Triangle Construction, and Michael Baker International. IRISE was established in the Civil and Environmental Engineering Department in the University of Pittsburgh's Swanson School of Engineering to study problems related to transportation infrastructure durability and resiliency.

Compliance with ethics guidelines

Qianyun Zhang, Kaveh Barri, Saeed K. Babanajad, and Amir H. Alavi declare that they have no conflict of interest or financial conflicts to disclose.

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of any member of the IRISE research consortium at the time of publication. This report does not constitute a standard, specification, or regulation.

References

[1] ASCE's 2017 infrastructure report card: bridges [Internet]. Reston: American Society of Civil Engineers; 2017 [cited 2017 Oct 11]. Available from: https://www.infrastructurereportcard.org/cat-item/bridges/.
[2] Spencer BF Jr, Hoskere V, Narazaki Y. Advances in computer vision-based civil infrastructure inspection and monitoring. Engineering 2019;5(2):199–222.
[3] Bao Y, Chen Z, Wei S, Xu Y, Tang Z, Li H. The state of the art of data science and engineering in structural health monitoring. Engineering 2019;5(2):234–42.
[4] Fujino Y, Siringoringo DM, Ikeda Y, Nagayama T, Mizutani T, Fujino ZY, et al. Research and implementations of structural monitoring for bridges and buildings in Japan. Engineering 2019;5(6):1093–119.
[5] Ansari F. Sensing issues in civil structural health monitoring. New York: Springer; 2005.
[6] Babanajad SK, Zhan Y, Taylor T, Ansari F. Virtual reference approach for dynamic distributed sensing of damage in large structures. J Aerosp Eng 2017;30(2):1–12.
[7] Iyer S, Sinha SK. A robust approach for automatic detection and segmentation of cracks in underground pipeline images. Image Vis Comput 2005;23(10):921–33.
[8] Salman M, Mathavan S, Kamal K, Rahman M. Pavement crack detection using the Gabor filter. In: Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems; 2013 Oct 6–9; Hague, the Netherlands; 2014.
[9] Talab AMA, Huang Z, Xi F, HaiMing L. Detection crack in image using Otsu method and multiple filtering in image processing techniques. Optik 2016;127(3):1030–3.
[10] Shan B, Zheng S, Ou J. A stereovision-based crack width detection approach for concrete surface assessment. KSCE J Civ Eng 2016;20(2):803–12.
[11] Sinha SK, Fieguth PW. Automated detection of cracks in buried concrete pipe images. Autom Construct 2006;15(1):58–72.
[12] Zou Q, Cao Y, Li Q, Mao Q, Wang S. CrackTree: automatic crack detection from pavement images. Pattern Recognit Lett 2012;33(3):227–38.
[13] Fujita Y, Hamamoto Y. A robust automatic crack detection method from noisy concrete surfaces. Mach Vis Appl 2011;22(2):245–54.
[14] Plehiers PP, Symoens SH, Amghizar I, Marin GB, Stevens CV, Van Geem KM. Artificial intelligence in steam cracking modeling: a deep learning algorithm for detailed effluent prediction. Engineering 2019;5(6):1027–40.
[15] Shang C, You F. Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the big data era. Engineering 2019;5(6):1010–6.
[16] Liu SW, Huang JH, Sung JC, Lee CC. Detection of cracks using neural networks and computational mechanics. Comput Methods Appl Mech Eng 2002;191(25–26):2831–45.
[17] Moon H, Kim J. Intelligent crack detecting algorithm on the concrete crack image using neural network. In: Proceedings of the 28th International Symposium on Automation and Robotics in Construction; 2011 Jun 29–Jul 2; Seoul, Republic of Korea; 2011.
[18] O'Byrne M, Ghosh B, Schoefs F, Pakrashi V. Regionally enhanced multiphase segmentation technique for damaged surfaces. Comput Aided Civ Infrastruct Eng 2014;29(9):644–58.
[19] Prasanna P, Dana KJ, Gucunski N, Basily BB, La HM, Lim RS, et al. Automated crack detection on concrete bridges. IEEE Trans Autom Sci Eng 2016;13(2):591–9.
[20] Ciresan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J. Flexible, high performance convolutional neural networks for image classification. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence; 2011 Jul 16–22; Barcelona, Spain; 2011.
[21] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 2012;1:1097–105.
[22] Abdeljaber O, Avci O, Kiranyaz MS, Boashash B, Sodano H, Inman DJ. 1-D CNNs for structural damage detection: verification on a structural health monitoring benchmark data. Neurocomputing 2018;275:1308–17.
[23] De Oliveira MA, Monteiro AV, Filho JV. A new structural health monitoring strategy based on PZT sensors and convolutional neural network. Sensors 2018;18(9):1–21.
[24] Khodabandehlou H, Pekcan G, Fadali MS, Pekcan G, Fadali MS. Vibration-based structural condition assessment using convolution neural networks. Struct Contr Health Monit 2018;26(2):e2308.
[25] Tabian I, Fu H, Khodaei ZS. A convolutional neural network for impact detection and characterization of complex composite structures. Sensors 2019;19(22):4933.
[26] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521(7553):436–44.
[27] Azimi M, Pekcan G. Structural health monitoring using extremely compressed data through deep learning. Comput-Aided Civ Infrastruct Eng 2020;35:597–614.
[28] Soukup D, Huber-Mörk R. Convolutional neural networks for steel surface defect detection from photometric stereo images. In: Bebis G, Boyle R, Parvin B, Koracin D, McMahan R, Jerald J, editors. Advances in visual computing. Cham: Springer; 2014. p. 668–77.
[29] Cha YJ, Choi W, Büyüköztürk O. Deep learning-based crack damage detection using convolutional neural networks: deep learning-based crack damage detection using CNNs. Comput Aided Civ Infrastruct Eng 2017;32(5):361–78.
[30] Da Silva WRL, de Lucena DS. Concrete cracks detection based on deep learning image classification. Proceedings 2018;2(8):489.
[31] Matsugu M, Mori K, Mitari Y, Kaneda Y. Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Netw 2003;16(5–6):555–9.
[32] Fukushima K. Neocognitron. Scholarpedia 2007;2(1):1717.
[33] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521:436–44.
[34] Scherer D, Muller A, Behnke S. Evaluation of pooling operations in convolutional architectures for object recognition. In: Proceedings of the 20th International Conference on Artificial Neural Networks; 2010 Sep 15–18; Thessaloniki, Greece; 2010.
[35] Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw 2015;61:85–117.
[36] Donahue J, Jia Y, Vinyals O, Hoffman J. DeCAF: a deep convolutional activation feature for generic visual recognition. In: Proceedings of the 31st International Conference on Machine Learning; 2014 Jun 21–26; Beijing, China; 2014. p. 647–55.
[37] Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. Overfeat: integrated recognition, localization and detection using convolutional networks. 2013. ArXiv:1312.6229.
[38] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.

[39] Gers FA, Schmidhuber E. LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Trans Neural Netw 2001;12(6):1333–40.
[40] Swapna G, Kp S, Vinayakumar R. Automated detection of diabetes using CNN and CNN-LSTM network and heart rate signals. Procedia Comput Sci 2018;132:1253–62.
[41] Zhao J, Mao X, Chen L. Speech emotion recognition using deep 1D & 2D CNN LSTM networks. Biomed Signal Process Control 2019;47:312–23.
[42] Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J. LSTM: a search space odyssey. IEEE Trans Neural Netw Learn Syst 2017;28(10):2222–32.
[43] Wang J, Zhang J, Wang X. Bilateral LSTM: a two-dimensional long short-term memory model with multiply memory units for short-term cycle time forecasting in re-entrant manufacturing systems. IEEE Trans Ind Inf 2018;14(2):748–58.
[44] Tsironi E, Barros P, Weber C, Wermter S. An analysis of convolutional long short-term memory recurrent neural networks for gesture recognition. Neurocomputing 2017;268:76–86.
[45] Oehmcke S, Zielinski O, Kramer O. Input quality aware convolutional LSTM networks for virtual marine sensors. Neurocomputing 2018;275:2603–15.
[46] Zhou X, Hu B, Chen Q, Wang X. Recurrent convolutional neural network for answer selection in community question answering. Neurocomputing 2018;274:8–18.
[47] Sainath TN, Vinyals O, Senior A, Sak H. Convolutional longshort-term memory fully connected deep neural networks. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing; 2015 Apr 19–24; South Brisbance, QLD, Australia; 2015.
[48] Liu T, Bao J, Wang J, Zhang Y. A hybrid CNN–LSTM algorithm for online defect recognition of CO_2 welding. Sensors 2018;18(12):4369.
[49] Das A. Interpretation and processing of image in frequency domain. In: Guide to signals and patterns in image processing. Berlin: Springer; 2015. p. 93–147.
[50] Chang SG, Yu B, Vetterli M. Adaptive wavelet thresholding for image denoising and compression. IEEE Trans Image Process 2000;9(9):1532–46.
[51] Zhang M, Gunturk BK. Multiresolution bilateral filtering for image denoising. IEEE Trans Image Process 2008;17(12):2324–33.
[52] Portilla J, Strela V, Wainwright MJ, Simoncelli EP. Image denoising using scale mixtures of Gaussians in the wavelet domain. IEEE Trans Image Process 2003;12(11):1338–51.
[53] Gonzalez RC, Woods RE, Masters BR. Digital image processing. 3rd edition. Upper Saddle River: Prentice Hall; 2008.
[54] Maguire M, Dorafshan S, Thomas R. SDNET2018: a concrete crack image dataset for machine learning applications. Logan: Utah State University; 2018.
[55] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. 2016. arXiv:1603.04467.
[56] Canny J. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 1986;8(6):679–98.
[57] Sobel I, Feldman G. A 3×3 isotropic gradient operator for image processing. A talk at the stanford Artificial Project. 1968:271–2.