COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Computer-Aided Civil and Infrastructure Engineering **32** (2017) 1025–1046

Structural Damage Detection with Automatic Feature-Extraction through Deep Learning

Yi-zhou Lin & Zhen-hua Nie

School of Mechanics and Construction Engineering, Jinan University & Key Lab of Disaster Forecast and Control in Engineering, Ministry of Education, Guangzhou, China

&

Hong-wei Ma*

Dongguan University of Technology, Dongguan, China & Department of Civil Engineering, Qinghai University, Xining, China

Abstract: *Structural damage detection is still a challenging problem owing to the difficulty of extracting damage-sensitive and noise-robust features from structure response. This article presents a novel damage detection approach to automatically extract features from low-level sensor data through deep learning. A deep convolutional neural network is designed to learn features and identify damage locations, leading to an excellent localization accuracy on both noise-free and noisy data set, in contrast to another detector using wavelet packet component energy as the input feature. Visualization of the features learned by hidden layers in the network is implemented to get a physical insight into how the network works. It is found the learned features evolve with the depth from rough filters to the concept of vibration mode, implying the good performance results from its ability to learn essential characteristics behind the data.*

1 INTRODUCTION

In recent years, the aging of large-scale structures has become an unavoidable issue, especially for these large-scale bridges performing an irreplaceable function in present-day society. Hence, ensuring the structural integrity and detecting the structural damage have attracted increasing attention by researchers in mechanics and civil engineering field (An et al., 2015; Mu and Yuen, 2016). In practice, a real bridge may be equipped with multiple sensors to monitor its vibration response. However, limited information can be obtained from time-domain data directly. Moreover, the form of excitation is almost impossible to get, giving rise to the difficulty to accurately estimate frequency response function (FRF) and vibration modes. As a result, the first and most important procedure is feature extraction (Amezquita-Sanchez and Adeli, 2015), that is, the problems may become more accessible, when the low-level sensor signals transform into significant and damage-sensitive features.

One way for feature extraction is computing dynamic signatures of the structure or designing their derivative to obtain a higher sensitivity of damage. Subsequently, damage detection can be implemented by monitoring the changes in these signatures. As a primary feature, natural frequency was investigated first due to the convenience in measurement. Cawley and Adams (1979) started related studies by directly dealing with damage detection based on the changes in the natural frequencies of the structure. However, Salawu (1997), presenting an overview of the publications of this topic before 1997, suggested that natural frequency changes may be confused by multiple damages. Although the application of natural frequency changes has been studied extensively, the damage detection methods based on

*To whom correspondence should be addressed. E-mail: *tmahw@jnu.edu.cn*.

DOI: 10.1111/mice.12313

frequency changes still have several limitations, such as the changes are usually so small to be buried by environmental and operational conditions, and natural frequencies seem "too general" for local damage. Unlike natural frequencies, mode shapes contain more local information, leading to a better sensitivity to local damage. Besides, they are less sensitive to environmental effects, such as the temperature, than natural frequencies are (Farrar and Iii, 1997). As a result, these methods that try to link damages and mode shapes or their derivatives have been developed. Pandey et al. (1991) first suggested that mode shape curvatures, the second derivations of mode shapes, have a good performance in damage localization. Dutta and Talukdar (2004) presented the mode shape curvatures are more sensitive to damage location than natural frequencies are, resulting in a better localization accuracy. The problem of the susceptibility to measurement noise of the modal curvatures was also discussed (Cao et al., 2014). Another derivative of mode shape is the modal strain energy, which is directly related to mode shape curvatures for beam-like structure. Shi et al. (2000) proposed a damage localization method based on modal strain energy change for beam, truss or frame type structures. These dynamic signatures also are the foundation of finite element (FE) model updating, playing a role as a metric to estimate the differences between FE model and real structure (Shabbir and Omenzetter, 2015). Nevertheless, in practice, the unreliability of damage identification using noisy measurements from accelerometers is still a problem (Adewuyi et al., 2009). In addition, from only response signals, it is difficult to precisely measure some of these signatures, such as FRF and vibration modes, giving rise to the requirement of an accurate FE model, which is also hard to be established.

In contrast to the approaches cited above, another methodology, called "Data-driven," extracts damage-sensitive features via only concentrating on the perturbation in sensor data instead of requiring a model of the structure. Some papers in this category used modern signal processing methods (Amezquita-Sanchez and Adeli, 2016; Huang et al., 2014) and tried to catch a sudden change in signals caused by the occurrence of damage. Hou et al. (2000) analyzed the characteristics of response signals under the wavelet transformation and found the wavelet coefficients can clearly show the moment when damage occurs. Law et al. (2005) proposed a concept of wavelet packet transform component energy and its sensitivity to local change in the system parameters was discussed. Another modern signal processing method, Hilbert-Huang transformation, was also introduced in structural health monitoring (Yang et al., 2004). In addition, the fusion of these signal processing methods was discussed by Li et al. (2017). Blachowski et al. (2017) proposed a damage localization method in truss structures using statistical signal processing. Another methodology in this category is treating the damage detection task as a pattern recognition problem based on time-series. Sohn and Farrar (2001) proposed a damage detection approach by modeling the dynamic signal using autoregressive moving-average (ARMA) model and monitoring the changes in model coefficients. Cavadas et al. (2013) present a data-driven method based on moving-load using moving principal component analysis and robust regression analysis, which detects damage by statistical features from time-domain data. However, the features used in above-maintained studies may not be optimized for damage, as a result of almost all of these methods are based on signal processing or statistics. They may also catch anomalies caused by other factors. Besides, even an experienced researcher may spend a long time to test which statistic index or signal transformation, even their derivative, is sensitive to structural damage.

On the other hand, artificial neural networks (ANN) have been utilized in structural damage detection field for many years (Adeli and Jiang, 2009). The features mentioned above are usually used as their inputs. Lee et al. (2005) presented a neural network–based damage detection method using the modal properties. Wavelet neural network (WNN), an adaptation of ANN using the concept of the wavelet, was employed to detect damage for high-rise buildings (Adeli and Jiang, 2005; Jiang and Adeli, 2007). Mehrjoo et al. (2008) used the neural network to overcome the problems caused by incompleteness of the measured data. Although partial successes were obtained, the performances of neural networks are influenced by input features greatly, that is, the drawbacks in input features, such as susceptibility to noise, may forbid the neural network to get better performance.

To address these limitations, this article uses the convolutional neural network (CNN), a type of neural network in deep learning field, as a feature extractor and a classifier to detect structural damage. Using this method, it is possible to detect damage from waveform signal directly without any hand-crafted features.

Deep learning is these methods that learning representations of data with multiple levels of abstraction via recomposing of multiple processing layers (LeCun et al., 2015). A CNN can make use of a large amount of data, which means the ability to improve itself continuously with more data. As a result, they have become pivotal methods of the concept of "big data." State of the art in various fields has been renewed by using deep learning algorithms in recent years, for example, in speech recognition, visual object recognition, object

detection, and many other domains. Especially, the CNN is most widely utilized in applications of speech recognition and computer vision. It is designed to process data that are in the form of multiple arrays, such as one-dimensional (1-D) for time-sequences and 2-D for signal data with multiple channels and images. LeCun et al. (1989) proposed the concept of CNN and successfully applied it to handwritten zip code recognition task. More recently, Krizhevsky et al. (2012) reported a large deep CNN for image classification to participate in the ImageNet LSVRC-2010 contest, and it outperformed the previous state of the art by reducing the error rate by 10%. Simonyan and Zisserman (2014) achieved new state-of-the-art results by utilizing a deeper CNN. To dealing with the audio data, Lee et al. (2009) applied CNN in the audio classification task. Abdel-Hamid et al. (2012) used CNN concept in hybrid NN–HMM model, which is a traditional robust method framework for speech recognition, and it also renewed the state of the art in this problem. In structural damage detection field, visual-based crack detection is an important topic (Chen et al., 2017), and CNN also achieves a good performance (Cha et al., 2017). The literature review mentioned above shows that the CNNs have a great potential to deal with these problems taking low-level sensor data as input and requiring better features to achieve the final goal, just like the structural damage detection is. It can learn hierarchical features automatically from examples in a data-specific way, an excellent property to tackle our issue.

This study presents a new approach for structural damage detection using a CNN with customized architecture. As a pilot study, a numerical model of simply supported Euler–Bernoulli beam is considered. Some new techniques of deep learning proposed by recent studies are used to improve its performance, resulting in an excellent accuracy, which is better than using wavelet packet transform component energy (Law et al., 2005) as the input feature.

However, neural networks are known as black boxes, that is, it is hard to understand how they work. To address this issue, this article conducts the visualization of hidden layers in the network by maximizing the activation of the target layer on the input. It is found that these features become more abstract layer by layer. What is more, they are understandable for humans and have a mechanical meaning to some extent. For instance, the layers in bottom play a role as band pass filters, which center in the nature frequencies of the structure. Multiband filters can be found in the middle layers. When the layers located deeper, it can be discovered that the network even has learned the concept of structural mode independently.

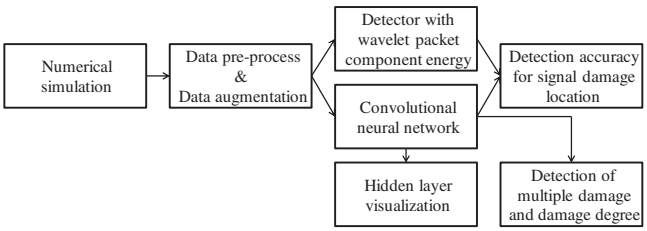

Fig. 1. The overview of the methodology.

It implies that the neural network can learn features of the structure automatically only from response data. Its findings, the structural modes, are helpful for damage detection task, which is also consistent with mechanical knowledge. Due to this promising property, this method has the potential to be used in more complex problems, in which hand-crafted features are hard to apply.

It is noteworthy that these findings are based on the data from a numerical model with different damage levels and locations. However, this condition is hard to be satisfied in practice. Further works are still needed to achieve real-world application with this method.

2 METHODOLOGY

This article presents an approach to detect structural damage with directly inputting low-level waveform signals. It can automatically extract features from the signals. In practice, low-level waveform signals are processed by signal processing, structural model estimation, or other hand-crafted feature-extraction methods. Then the representations of structure condition can be obtained. However, there are some drawbacks in this methodology as shown in Section 1. In this article, to address these issues, a CNN is designed to process low-level sensor data, to learn features, and to achieve damage detection simultaneously. Although damage localization is mainly discussed in this work, the ability to detect multiple damages and corresponding damage degree is also investigated.

The overview of the proposed method, also shown in Figure 1, is organized as follows: (1) numerical simulation is conducted to obtain structural responses data; (2) the data are processed by preprocessing procedures including data augmentation, an operation to facilitate performance of the neural network by creating more data; (3) a deep CNN, as presented in Figure 2, is trained on the augmented data set; (4) for comparison, another detector using wavelet packet component energy as input feature is also trained; (5) their classification performances of single damage localization is

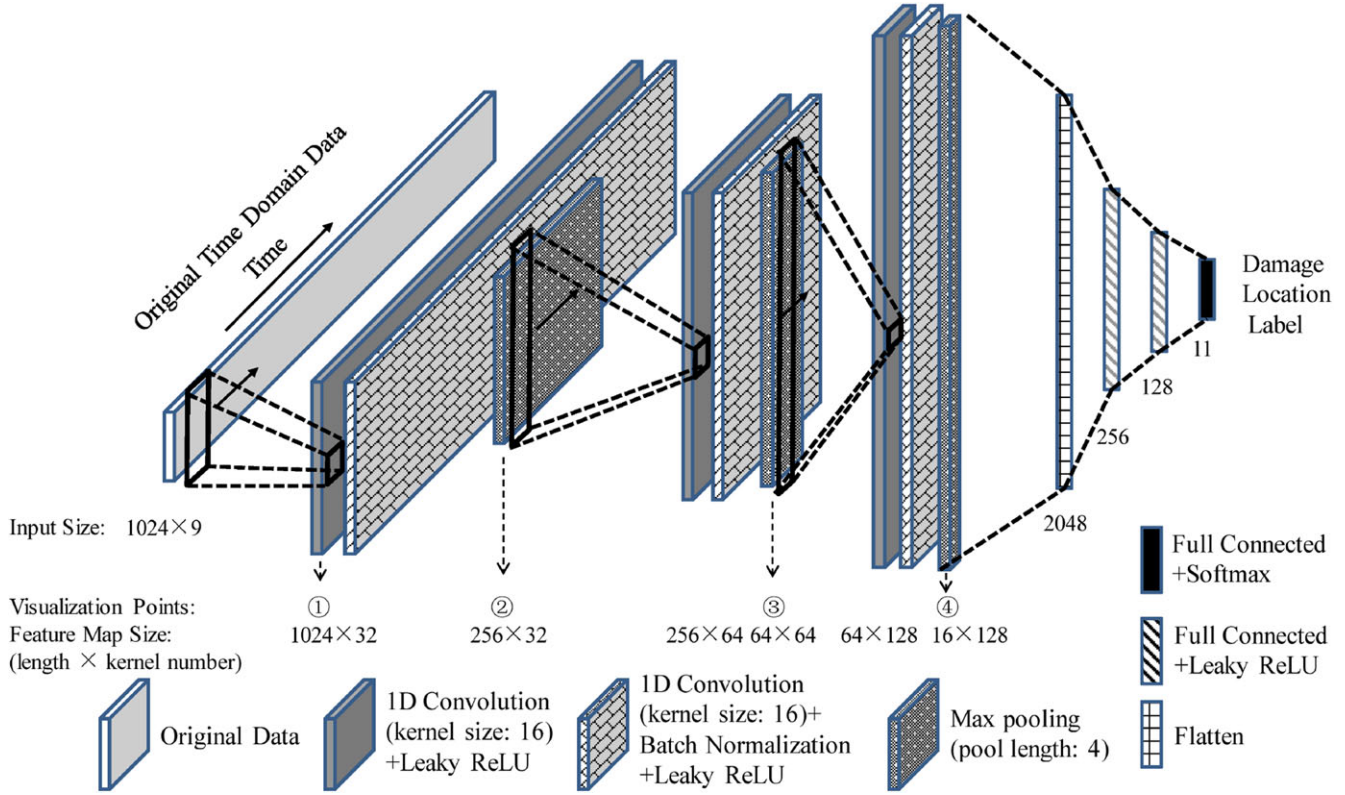

Fig. 2. Convolutional neural network architecture.

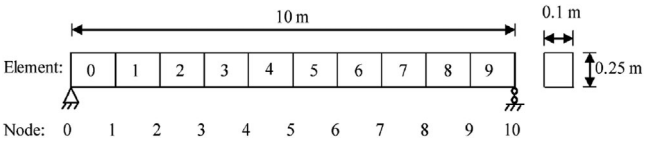

Fig. 3. The numerical model of the simply supported beam.

Table 1
The model natural frequency and damping ratio

Modal order	*1st*	*2nd*	*3rd*	*4th*	*5th*
Freq./Hz	5.79	23.16	52.13	92.77	145.29
Damp./%	1.39	0.39	0.27	0.29	0.37

estimated by a test procedure; (6) the application of the CNN to multiple damages detection is also discussed; and (7) to find a mechanical interpretation of how the neural network works, visualization of internal representations of the network is carried out.

2.1 Numerical simulation

A simply supported beam with a length of 10 m, subjected to burst random excitation, is considered in this study. It is simple so that its dynamic characteristics are clear, leading to convenience in validating the methodology with our knowledge, such as finding comprehensible features form the hidden layers of the CNN.

The beam has a rectangular cross-section with 0.1 m width and 0.25 m height, and it is divided into 10 equal Euler–Bernoulli beam elements, as shown in Figure 3. The material properties are listed as follows: the Young modulus is 206 GPa and the density is 7,900 kg/m^3. Rayleigh damping is used to simulate the damping effect, as shown in Equation 1

$$C = \alpha K + \beta M \tag{1}$$

where α and β used in this article are 7×10^{-6} s and 1.0 s^{-1}, respectively, resulting in a weak damping. Detailed dynamical properties of the numerical model, including natural frequencies and damping ratios of the first five orders, is shown in Table 1.

Damage is simulated by decreasing height of the target beam element. There are six damage conditions, including five damage levels and an intact condition. Five levels of damage are prepared by reducing the bending rigidity of the target element by 10%, 20%, 30%, 40%, and 50%, respectively, corresponding to reducing the

Table 2
The relative variation ratio of natural frequencies when element no. 4 is damaged

Damage level	*Modal order*				
	1st	*2nd*	*3rd*	*4th*	*5th*
10%	−0.38%	−0.71%	−0.36%	−0.09%	−0.36%
50%	−3.58%	−5.76%	−2.63%	−0.92%	−2.59%

height of the element to 96.5%, 92.8%, 88.8%, 84.3%, and 79.4%. Table 2 exhibits natural frequencies change slightly when the element no. 4 is damaged with level 10% and 50% (the bending rigidity of element no. 4 is 90% and 50% of the intact condition, respectively).

Random excitation, meaning the amplitude of the excitation on each time step follows Gaussian distribution, is applied on one of the nine nodes from no. 1 to no. 9 to simulate environmental load. It has equal intensity at different frequencies, leading to the excitation of multiple structure modes. However, the environmental load in real world does not have a spectrum with infinite bandwidth and the first several modes of a structure normally predominate the structural response. To simulate this phenomenon, a low-pass filter is applied on the random excitation. Concretely, an eighth-order Butterworth filter with normalized cutoff frequency 512 Hz is applied on 3-second-long random excitation with a standard deviation for 200 N.

Newmark's method (Newmark, 1959) is used to compute the response. Appropriate parameters are selected to obtain a second-order accuracy with no numerical damping. To get a reasonable accuracy, 2^{-13} s is chosen as the time step. However, such a small time step leads to a large volume of data, inducing computational issues for subsequent procedures. A down-sampling process, therefore, is required to reduce the final sampling frequency. The original computation results are down-sampled to 1,024 Hz, and only vertical accelerations from no. 1 to no. 9 node are measured, resulting in a data shape for $9 \times 3{,}072$ in each measurement.

On account of the nine positions for loading (node no. 1 to no. 9), 10 positions for damage (element no. 0 to no. 9), and six conditions for damage (one intact condition and five damage levels), the number of combinations is 459 ($9\,load\ positions \times (10\,damage\ locations \times 5\ damage\ levels + 1\ intact\ condition)$), because the damage location makes no sense for the intact condition. To generate a large enough data set, 15 calculations are carried out for each configuration; finally, the size of the whole data set is 6,885 (459×15).

2.2 Data augmentation

Getting more data is always the best way to make a neural network model generalize better, although the amount of data are usually limited in practice. One way to tackle this issue is creating fake data based on the true data set and adding it to the training set, which the term of "data augmentation" means.

Data augmentation is an efficient approach for different classification tasks. These state-of-the-art image recognition systems also used data augmentation techniques, such as randomly cropping a piece from the origin image, randomly rotating for a small angle, horizontal flipping, adjusting contrast, and so forth (Szegedy et al., 2015; Wu et al., 2015). For speech recognition tasks, injecting noise into the input is also a good way of data augmentation to improve the robustness (Sietsma and Dow, 1991).

During the training a neural network, a data batch, a subset of the whole data set, is usually utilized to

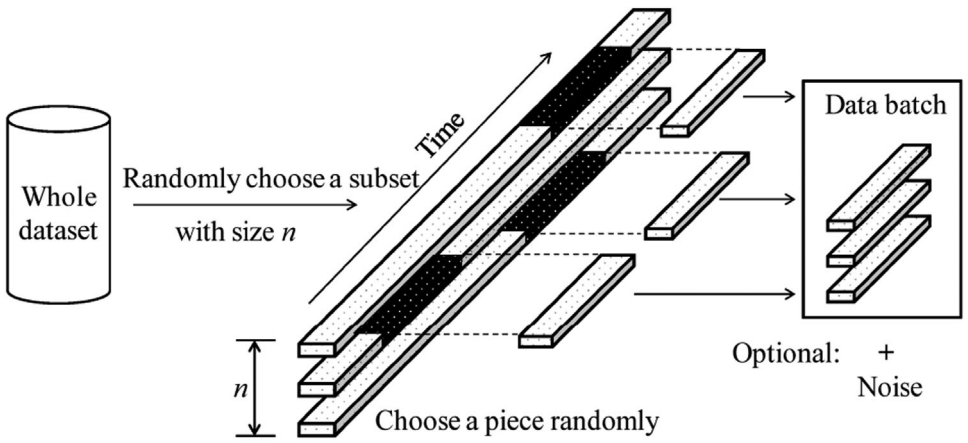

Fig. 4. Data augmentation procedure.

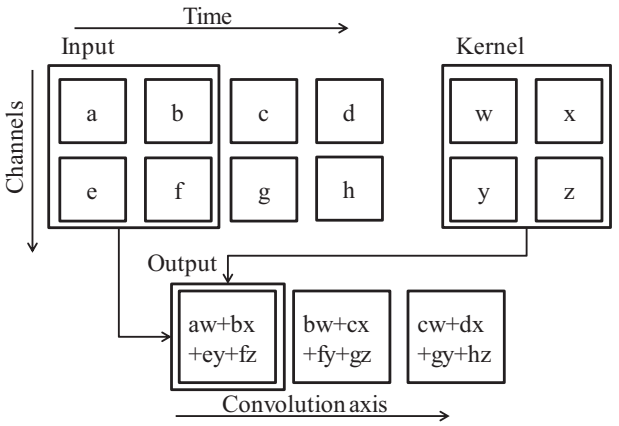

Fig. 5. An example of 1-D convolution along time axis.

calculate the gradient of the objective function with respect to the weights in the neural network in the current iteration step. The idea of data augmentation here is to identify valuable difficulties in every data batch in the training process by randomly sampling pieces from origin signals. Specifically, if the size of a data batch is n, this procedure randomly chooses n measurement data from the whole data set. Subsequently, pieces are randomly sampled from each selected datum, as illustrated in Figure 4. Here, the length of pieces is 1,024, a power of two, leading to a better computational efficiency. In other words, a part of the signal with 1-second length is cut out from the whole 3 seconds time sequence. It should be noticed that the data augmentation operation is carried out after a data normalization process, as demonstrated in Equation 2,

$$D_n = \frac{D_o - \overline{D_o}}{\sigma\left(D_o\right)} \tag{2}$$

where D_o is the origin data set, $\overline{D_o}$ and $\sigma(D_o)$ are the mean and standard deviation, respectively, and D_n is the normalized data set.

2.3 Noise simulation

In practical measurement or monitoring, noise is an inevitable factor, and the robustness to measurement noise should be discussed for every structural damage detection method.

Sensor noise, normally caused by various environmental factors, is simulated to estimate the robustness of the proposed method. It decays the signal by reducing the signal-to-noise ratio, leading to difficulties to identify valuable information. As a result, a neural network has to learn essential features against the interference of noise.

Concretely, signals from all the sensors are corrupted by adding white Gaussian noise. Noise level is determined by giving a standard deviation of white Gaussian noise, which is injected into the normalized data set, as shown in Equation 3 and Equation 4,

$$\hat{D_n} = D_n + \xi \tag{3}$$

$$\xi \sim N\left(0, \sigma^2\right) \tag{4}$$

where $\hat{D_n}$ is the noisy data set, and ξ is a random variable follows a Gaussian distribution with zero mean and σ^2 variance.

Here, σ is set to be 0.5, meaning the Gaussian distributed noise with 50% standard deviation is injected into the data set. It is an efficient way to assess the method by challenging such a high noise level.

2.4 Convolutional neural network

The only difference between a CNN (LeCun, 1989) and a simple neural network is that the former uses convolution instead of general multiplication in at least one of its layers. As a result, it makes CNNs specialized to process the data having a known grid-like topology, such as image data for 2-D and time-series data for one dimension. Notice that structural responses data are also time series to take advantage of the CNN.

In this article, a CNN is designed for feature extraction and damage localization task at once. Typically, a convolutional network may consist of multiple convolutional layers. Furthermore, a convolutional layer usually contains three stages: convolution, nonlinear activation, and pooling. To achieve better performance, a straightforward and efficient method of regularization named "Batch Normalization" (BN) (Ioffe and Szegedy, 2015) is also employed. These normalization layers are inserted before the nonlinear activation stage in some convolutional layers, as shown in Figure 2. All these components will be introduced in the following sections.

As illustrated in Figure 2, the particular architecture of the network is demonstrated as follows: (1) a convolutional layer with nonlinear activation is followed by another one with a normalization layer before nonlinear activation, then a pooling layer is added; (2) the same substructure is replicated twice; (3) the output of previous architecture is flattened into a 1-D vector; (4) two full connected (FC) layers are used to access the output layer. There are a total of 1,072,267 parameters and eight nonlinear hidden layers in this deep CNN.

2.4.1 Convolution and pooling. The convolution and pooling are the most important and unique operations in a CNN.

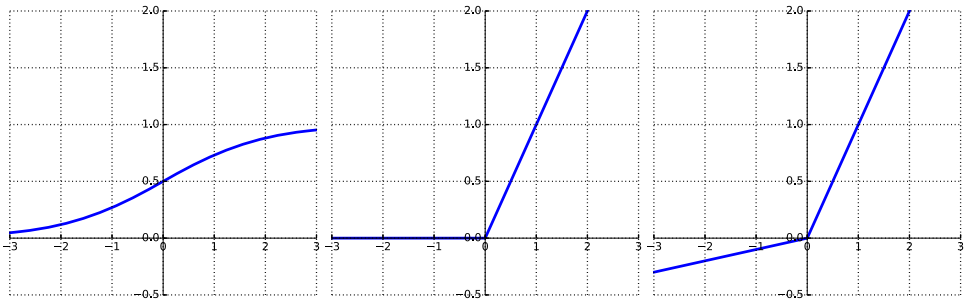

Fig. 6. Sigmoid (a), ReLU (b), and leaky ReLU (c) ($\alpha = 0.1$).

Generally, convolution is an operation on two real-valued functions, such as Equation 5.

$$f(i) = \int_{-\infty}^{\infty} s(n)k(i-n)\,dn \tag{5}$$

In CNN terminology, the first function (for instance, the function s in Equation 5) is usually known as the input and the second (e.g., the function k in Equation 5) is known as the kernel or filter. Besides, the output here is referred as feature map. Nevertheless, data processed on a computer, both time and sensor data are discrete. Assuming the functions s and k are defined on integer domain, discrete convolution then can be defined as Equation 6, where v_s and v_k are the max valid indexes in function S and K, respectively, assuming there is always zero when the index is out of the valid range.

Due to convolution being commutative, the two representations in Equation 6 are equivalent. However, the latter one is more appropriate for machine learning algorithm implementation, because in CNN context, the kernel K, usually a multidimensional array of parameters, is much smaller than input S, leading to a much smaller max valid index in the kernel. In other words, that the v_k is much smaller than v_s brings convenience in algorithm implementation.

Furthermore, in the latter term of Equation 6, the input S is flipped relative to the kernel K, that is, when n increases, the index into the kernel increases, but the index into the input S decreases; it brings the property of commutativity to convolution. Nevertheless, the commutativity is only useful for mathematical derivation but not necessary for neural network implementation. As a result, a version of convolution without flipping, which actually should be called as cross-correlation, is adopted by most machine learning libraries, as shown in Equation 7.

$$F(i) = \sum_{n=1}^{v_s} S(n)\,K(i-n) = \sum_{n=1}^{v_k} S(i-n)\,K(n) \tag{6}$$

$$F(i) = \sum_{n=1}^{v_k} S(i+n)\,K(n) \tag{7}$$

However, it is still called "convolution" in machine learning field. This article follows this convention. This adaptation makes sense because appropriate parameters in the kernel will be learned by the learning algorithm whether the input is flipped or not.

Equation 7 shows the 1-D convolution with single channel in machine learning, however, can be extended to multichannels situation by simply applying convolution on multiple channels in parallel, then adding the results of those channels up. Figure 5 demonstrates what happened in 1-D convolution with two channels along time axis. In this example, the input is a sequence of vectors with length for two, and the kernel has a kernel size for two. The output is computed by multiplying the kernel by corresponding part of the input and adding the products up through channels. Finally, this operation is implemented step by step along the convolution axis (time axis). In this article, 1-D convolution with multiple channels is used in all the convolutional layers.

As mentioned above, a pooling operation is a part of a typical convolutional layer. It provides a statistical summary of the nearby output elements.

One type of pooling operation is the max pooling (Zhou and Chellappa, 1988). It is similar with convolution, however, instead of matrix multiplication, it picks out the maximum within its "kernel size," called as pool length here, step by step. Besides, pool stride is the length of gaps between every two neighbor steps. It is usually set to be the same of pool length, whereas the similar parameter in convolution is configured to be one. For instance, assuming the output before a max pooling layer is a vector like $[1, 2, 3, 4]$ and both pool length and pool stride are two, then the output of pooling will be $[2, 4]$.

The pooling operation mainly brings two benefits: first, it helps make the representation more invariant to the small variance of the input. On account of the result of pooling being a statistical summary, a small variance in input may change its statistic characters slightly. It is a useful property to make a neural network more robust to detect whether some feature is present or not. Second, the pooling operation reduces the size of feature map, which is essential to improve the computational efficiency of the network.

This work employs max pooling, and pool stride is set to be four.

2.4.2 Leaky rectified linear unit (ReLU). Activation functions are the essential part of a neural network, which transforms a network from a linear mapping to a nonlinear one, resulting in a higher representation capacity.

Historically, the sigmoid function, having the form $\sigma(x) = 1/(1 + e^{-x})$ as shown in Figure 6a, had frequently been used for its excellent properties, such as easy to calculate the derivatives and the range of its output is between zero and one. However, its defects make it rarely used in recent years. For instance, the sigmoid function "kills" gradients. When its output is close to either zero or one, the gradient is always approximate to zero, meaning parameters are hard to move anymore in back-propagation. It is a dangerous property, especially when the neural network becomes deeper and deeper, leading to the difficulties in neural network training.

The ReLU (Glorot et al., 2011; Nair and Hinton, 2010) was then proposed and has become very popular recently. It has a simple form as $f(x) = \max(0, x)$, as shown in Figure 6b. This form mainly brings two merits: first, gradients can be retained instead of vanishing at the whole right side of the function. This property brings a better training speed than using sigmoid functions (Krizhevsky et al., 2012). Second, it is cheaper in computation. Nevertheless, the ReLU may "die" in training process, that is, it is possible that some weights are updated by a large gradient, then their activations become zero and never activate again because the gradient of the left side of ReLU is always zero.

The leaky rectified linear units (leaky ReLU) (Maas et al., 2013) is a variant of ReLU designed to solve the "dying ReLU" problem. To gain an insight into what is different, Figure 6 is prepared. In Figure 6c, it can be found that the right part has a slope. In fact, its mathematical expression is shown in Equation 8,

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{8}$$

where α is a parameter with small value. When a unit is not active, the nonzero gradient brings the ability for escaping from "dying ReLU." Although Figure 6c shows a big α (0.1) for clarity, $\alpha = 0.01$ is more widely used in machine learning applications. In this article, the leaky ReLU is applied as nonlinearity, and the parameter α is 0.01 in every activation function.

2.4.3 Batch normalization. BN is also a type of layer that is specialized to solve the "internal covariate shift" problem in neural network training (Ioffe and Szegedy, 2015). In brief, the distribution of internal activations will continuously vary with the changes of network weights during training, which makes the learning algorithm have to fit these unstable distributions in every training step, leading to a low convergence rate. The BN layer is designed to alleviate this problem with a cheap computational cost.

The details of the algorithm are shown in Equation 9 to Equation 12.

$$\mu_D = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{9}$$

$$\sigma_D^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_D)^2 \tag{10}$$

$$\hat{x_i} = \frac{x_i - \mu_D}{\sqrt{\sigma_D^2 + \epsilon}} \tag{11}$$

$$y_i = \gamma \hat{x_i} + \beta \tag{12}$$

where ϵ is a small number, which is used against numerical problem of division by zero. During training, for every mini-batch of data, $D = \{x_1, \ldots, x_m\}$, the algorithm calculates the mean and variance, then shifts and scales the origin data x_i to zero-mean and one variance. Finally, it introduces two learnable parameters, γ and β, to hold the model flexibility.

The BN layer keeps its output following a similar distribution. As a result, the difficulty of training in the next layer is reduced, giving rise to a fast convergence. In practice, it is found that the BN not only brings fast convergence but also enhances the generalization.

A total of three BN layers are inserted in the utilized network architecture to promote the training progress, as illustrated in Figure 2.

2.5 Structural damage detection

In this section, the introduced CNN is applied to structural damage detection task. Although damage localization is mainly discussed, the ability of the CNN to detect multiple damages and corresponding damage degree is also investigated.

Table 3
The detailed configuration of the convolutional neural network architecture

Layer	*Type*	*Input shape*	*Output shape*	*Kernel num.*	*Kernel size*	*Stride*	*Padding*	*With BN*	*Activation*
1	Convo.	(1024, 9)	(1024, 32)	32	16	1	Same	False	Leaky ReLU
2	Convo.	(1024, 32)	(1024, 32)	32	16	1	Same	True	Leaky ReLU
3	Pooling	(1024, 32)	(256, 32)	None	4	4	Valid	False	None
4	Convo.	(256, 32)	(256, 64)	64	16	1	Same	False	Leaky ReLU
5	Convo.	(256, 64)	(256, 64)	64	16	1	Same	True	Leaky ReLU
6	Pooling	(256, 64)	(64, 64)	None	4	4	Valid	False	None
7	Convo.	(64, 64)	(64, 128)	128	16	1	Same	False	Leaky ReLU
8	Convo.	(64, 128)	(64, 128)	128	16	1	Same	True	Leaky ReLU
9	Pooling	(64, 128)	(16, 128)	None	4	4	Valid	False	None
10	Flatten	(16, 128)	(2048)	None	None	None	None	None	None
11	FC	(2048)	(256)	None	None	None	None	False	Leaky ReLU
12	FC	(256)	(128)	None	None	None	None	False	Leaky ReLU
13	Softmax	(128)	(11)	None	None	None	None	None	None

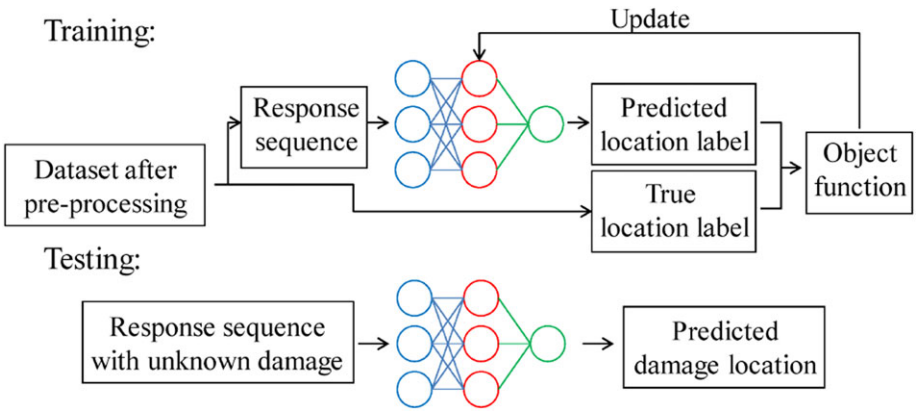

Fig. 7. Overview of the damage detection procedure.

2.5.1 Single damage localization. Figure 2 shows an overview of the CNN. It contains two parts: convolutional layers and FC layers. These convolutional layers are designed to extract features from local regions of the original data and to repeat the feature-extraction process from the previous feature map. To achieve this goal, two convolutional layers are stacked to get a good performance in nonlinear transformation, then a pooling layer follows to get a statistical summary of the feature map. All the convolutional layers operate convolution along the time axis. And, the kernel number increases along with the depth to hold enough capability of retaining information of the feature map because the pooling operation always reduces the size of the feature map along the time axis. The same substructure is replicated twice to obtain an implicit hierarchical feature structure in the neural network. On the other hand, the top FC layers work as a traditional classifier, that is, they take the feature map from previous convolutional layers as input and map it to the probabilities of every final category. More details are listed in Table 3.

As illustrated in Figure 7, there are two phases in this method: training phase and testing phase. In the training phase, the neural network is updated by reducing the difference between the predicted damage location and the real damage location. An objective function is utilized to estimate the difference between the prediction and the true answer, and its derivatives on model parameters are used to update the neural network. On the other hand, in the testing phase, the trained neural network receives new responses data, which are never seen by it in training. Then a prediction of damage location is made. Furthermore, an estimation of the model performance can be conducted by checking whether the prediction is correct.

Concretely, the accelerations of vertical direction are considered as input data, and more details are presented as follows. As mentioned in Section 2.1, the data set contains a total of 6,885 measurements, and each

Table 4
Configurations of training process

Name	*Value*	*Description*
Batch size	128	The size of data batch used in every training iteration
Steps per epoch	128	The number of steps in a single training epoch
Initial learning rate	5×10^{-6}	The initial learning rate of Adam algorithm
β_1	0.9	A parameter of Adam, weight of the momentum term
β_2	0.999	A parameter of Adam, controlling the decay of learning rate
Patience	300	A parameter of early stopping, how long the performance does not increase is acceptable

measurement has a shape for $9 \times 3{,}072$. After shuffling the whole data set, 70% (4,820 samples) of it is used as the training set, the testing set takes 20% (1,377 samples), and the validation set occupies 10% (688 samples). All these data sets are processed by a normalization and a data augmentation procedure as discussed in Section 2.2. As a result, the network receives a batch of pieces of the original response sequence in every training iteration. Each piece has a shape for $9 \times 1{,}024$.

As a supervised learning algorithm, the CNN needs a known label for every datum. In this section, the labels are the damage locations, which are encoded into one-hot form. It means the label vector is generated by the rule that the vector has all zero elements except the position i, where i is the number of damaged element, as shown in Figure 3. In this way, each element in the label vector represents the probability of whether its position is the true damage location, that is, the element with value one in the true label vector means its corresponding beam element is damaged with 100% possibility and the other zero elements mean their positions have 0% possibility of damage. This representation is required because the CNN uses Softmax layer as the output layer. It squashes the input to a vector of which each element is in a range from zero to one. These elements are then treated as the probability of damage predicted by the CNN. In other words, taking a sample as input, the network outputs a conditional probability distribution of damage location given the input sample.

Categorical cross entropy is used as objective function to estimate the difference between the true damage location and the predicted damage location. It measures the difference between two probability distributions p and q over the same underlying set of events. When distributions p and q are discrete, it is defined as Equation 13,

$$H(p, q) = -\sum_{x} p(x) \log(q(x)) \tag{13}$$

where $H(p, q)$ is the cross entropy, x is the valid discrete value of the distributions. In this article, x is the index of elements in the label vector, because a valid index represents a valid discrete damage location. $p(x)$ is the xth element in the true label vector, meaning the true possibility of the existence of damage in xth element. Similarly, $q(x)$ represents the xth element in the prediction vector, indicating the predicted possibility of the existence of damage in xth element.

To minimize the output of the objective function, an adaptation of mini-batch stochastic gradient descent

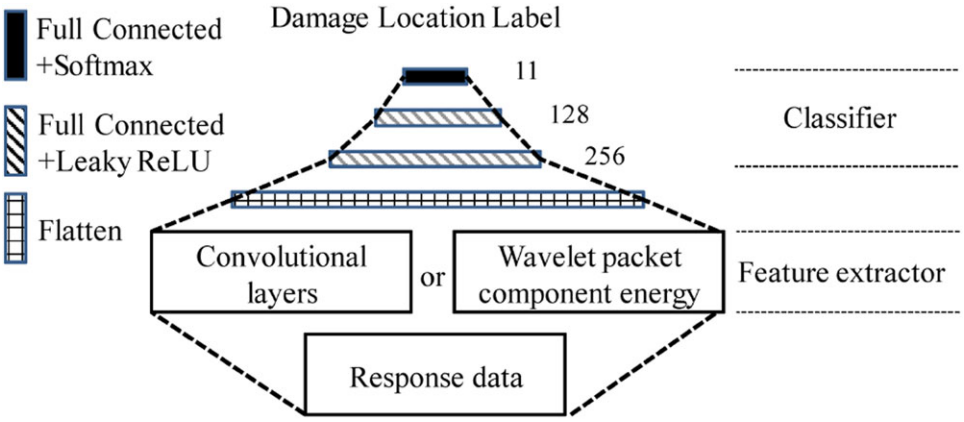

Fig. 8. Comparing features by the same classifier.

algorithm, named Adam (Kingma and Ba, 2014), is employed as the optimization algorithm. It uses both an adaptive learning rate and a momentum term during the training, leading to a better performance. Early stopping technology is utilized to control the training process to avoid overfitting. It records the recent best performance of the neural network on validation set during training. If the performances of further several iteration epoches all do not exceed the recent best, then it stops the training process. Detailed configurations of training are summarized in Table 4.

2.5.2 Multiple damages identification. In this section, the application of CNN to multiple damages identification is discussed. Both damage level and damage location are considered by designing a new representation of damage label and using our CNN to regress it. Specifically, it is investigated that at most three damages exist and the possible damage levels are the five levels mentioned in Section 2.1.

Almost the same neural network architecture described in Section 2.5.1 and all the configurations of training are still used in this section, except three aspects: the strategy of responses data generation, the output layer of the CNN, and the objective function.

To obtain responses data with multiple damages, multiple stiffness degradations of the beam elements are simulated on the simply supported beam model mentioned in Section 2.1. The strategy of computation is described as follows: (1) randomly choosing the number of damages in range from zero to three; (2) randomly selecting the beam elements to be decayed according to the number of damages; (3) randomly choosing the damage level for each selected element in the five damage levels as described in Session 2.1; (4) randomly choosing a load position, then computing the structural response using this configuration of damage; and (5) repeating the process (1) to (4) 10,000 times to get enough data. As a result, the data set consists of 10,000 examples including the ones with intact condition, single damage, and multiple damages. Following the setting in Section 2.5.1, the 70%, 20%, and 10% of the data set are used as training set, testing set, and validation set respectively.

The last layer of the CNN is also modified to fit the requirement of detecting multiple damages. Instead of Softmax activation, the last layer now uses linear activation to fit a new definition of label and the output size is changed to 10. Now the label is a vector consisting of 10 elements. Each element represents the damage level of corresponding beam element. For example, if a label vector has all zero elements except that the second element is 0.1 and the fourth is 0.3, it means the second and fourth element of the beam are damaged with damage level 10% and 30%, respectively, and the others are intact.

Similarly, the objective function also needs a modification to follow the change in label definition. Mean squared error is used in this task as shown in Equation 14.

$$L\left(y_p, y_t\right) = \frac{1}{k}\sum_{n=1}^{k}\left(y_p\left(n\right) - y_t\left(n\right)\right)^2 \tag{14}$$

where y_p and y_t are the label of prediction and the true label, respectively, and k is the length of the label vector, which is 10 in this case.

Having been trained, a CNN can give a prediction label of which each element indicates a predicated damage level of its corresponding beam element, leading to multiple damages identification. Similarly, the method is tested on both noise-free and noisy data set.

2.6 Visualization for physical interpretation

Deep neural networks have shown their excellent performances in a variety of fields. However, they are "black boxes" and humans may find it hard to understand how they work. It is vital for researchers to get qualitative interpretations to these systems so that an estimation can be carried out on their reliability. In structural damage detection task, meaningful indexes have been discussed by previous works, such as vibration modes and their derivatives, as mentioned in Section 1. These indexes map the structural response into a new representation to more efficiently show the healthy condition of the structure. If similar representations are utilized inside the neural network, it can be more easily understood in a physical view.

How to visualize a hidden layer in a neural network to get its intermediate representations? A straightforward idea to solve this problem is generating a synthetic signal that maximally activates the target hidden neuron (Erhan et al., 2009). The activation value of a neuron can be regarded as a score to evaluate how similar is the input signal to the feature learned by the neural. In other words, the signal that makes the target neuron most active is the one most similar to the learned feature. As a result, it is chosen as an intuitive expression for this neuron.

Specifically, Simonyan et al. (2013) used an additional ℓ_2 norm penalty term to forbid the amplitude of the input signal increase infinitely, as shown in Equation 15,

$$\hat{x} = \arg\max_{I}\ s_{ij}(I, \theta) - \lambda\|I\|_2^2 \tag{15}$$

Table 5
Scenario setting

		Feature extractor	
Scenario no.		*Convolutional net.*	*Wavelet packet*
Noise	Noise free	1	2
	With noise	3	4

where s_{ij} is the score or the activation value of ith neural in layer j in a neural network, I and θ are the input signal and model parameters, respectively.

In this work, the CNN is utilized. As a result, activations have a form of feature maps (refer to Figure 2 for insight) instead of a single value. A criterion with a form of single value is, thus, needed to estimate the activity level of the whole feature map and mean value is considered here. A modification is therefore introduced that using the mean of the ith feature map in layer j instead of the activation value s_{ij}, as shown in Equation 16,

$$\hat{x} = \arg\max_{I} \overline{S_{ij}(I,\theta)} - \lambda \|I\|_2^2 \tag{16}$$

$$x^* = \frac{\hat{x} - \overline{\hat{x}}}{\sigma(\hat{x})} \tag{17}$$

where $S_{ij}(I,\theta)$ is the ith feature map in layer j and the weight parameter λ here is set to be 10 in this article. In addition, a normalization process is carried out to ensure the same scale in visualization results, as shown in Equation 17. Subsequently, intuition into how the neural network works can be found by observing these results through depth of the network.

3 RESULTS

As mentioned in Section 1, there are defects in these hand-crafted features for structural damage detection. Some of them are susceptible to noise or hard to measure from responses data directly, and some are maybe too general instead of concentrating on local damage. Besides, designing these features costs a considerable amount of time and requires expert knowledge.

The primary purpose of this work is extracting features of the structure condition from responses data directly by the learning algorithm. These features are specialized for damage detection by supervised training to locate the damage. However, how to estimate the extracted features is an important topic. To tackle this issue, the classification accuracy is considered,

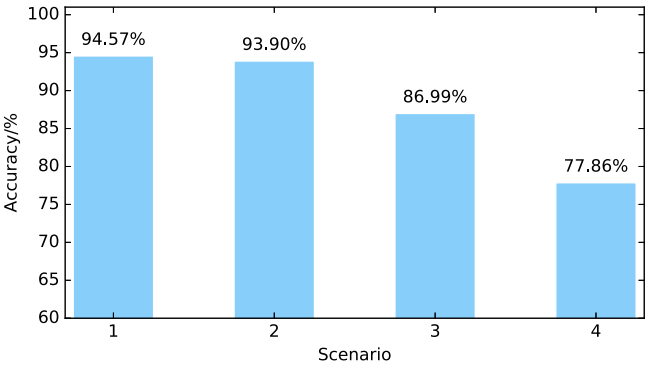

Fig. 9. Accuracy of scenarios defined in Table 5.

and the visualization of hidden layers is implemented to get qualitative interpretations into these features as well.

In this section, the performances of features given by the proposed method and a powerful hand-crafted feature extractor, the wavelet packet transform component energy (Law et al., 2005), are compared. Although directly comparing the performance of features is hard, the following approach is considered. Ordinarily, shallow neural networks have been used as classifiers with hand-crafted features in structural damage detection field for years. The proposed architecture also can be recognized as a combination of two parts: the convolutional layers at the bottom can be regarded as a hierarchical feature extractor, and the top FC neural network plays a role as a classifier. In this way, the comparison can be implemented by replacing the bottom feature extractor with the hand-crafted features, which is the wavelet packet transform component energies in this article, as shown in Figure 8. More details will be discussed in the next subsection.

3.1 Scenarios and configuration

To estimate the learned feature, two schemes are considered. The first is comparing performance with another feature extractor. A comparison between the proposed method and the wavelet packet transform component energy is then implemented. As shown in Figure 8, the two feature extractors are judged by the same neural network classifier, that is, the better features will lead to a better performance. Second, its robustness to noise is investigated. Both original and noisy data are arranged in the comparative tests. The definition of noise level here is already discussed in Section 2.3. All the scenario plans are summarized in Table 5.

Here are some details about the parameters in the wavelet packet analysis used in this study. Db2

Table 6
The accuracy of scenario 1

		Predicted damage location												
Count		*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Total*	*%*
Actual damage location	0	745	13	26	5	0	0	0	0	0	0	1	790	94.30
	1	9	677	13	6	3	1	0	0	0	0	2	711	95.22
	2	5	9	769	4	3	0	0	0	0	0	0	790	97.43
	3	4	3	14	728	8	0	1	0	0	1	1	760	95.79
	4	1	7	10	10	629	2	4	0	1	1	3	668	94.16
	5	1	0	2	5	4	646	3	8	2	0	3	674	95.85
	6	0	0	0	5	8	6	638	13	19	1	1	691	92.33
	7	0	0	2	4	3	5	13	694	5	8	3	737	94.17
	8	0	0	0	0	2	4	10	16	716	14	3	765	93.59
	9	0	0	0	0	0	1	7	20	7	762	0	797	95.61
	10	4	4	23	6	8	2	8	2	4	5	743	809	91.84
	Total	769	713	859	773	668	667	684	753	754	792	760	8192	Overall: 94.57

Table 7
The accuracy of scenario 2

		Predicted damage location												
Count		*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Total*	*%*
Actual damage location	0	792	26	3	4	2	0	0	0	0	0	2	829	95.54
	1	4	615	2	2	6	2	0	0	0	0	9	640	96.09
	2	3	7	772	7	3	1	0	1	0	0	3	797	96.86
	3	0	1	0	751	2	2	0	0	0	0	5	761	98.69
	4	0	1	1	5	627	2	1	0	0	1	3	641	97.82
	5	1	0	0	0	0	682	0	0	1	0	2	686	99.42
	6	0	0	0	2	2	18	638	4	0	5	9	678	94.10
	7	0	0	0	0	0	1	4	680	6	3	8	702	96.87
	8	0	0	0	1	0	9	0	3	721	8	16	758	95.12
	9	0	0	0	0	0	2	3	3	10	732	2	752	97.34
	10	12	33	18	48	29	65	9	9	21	22	682	948	71.94
	Total	812	683	796	820	671	784	655	700	759	771	741	8192	Overall: 93.90

wavelet (Daubechies and Heil, 1992) is used as wavelet generating function, and four is selected as the level of wavelet packet decomposition; as a result, there are 16 component coefficients per signal. Finally, the original responses data, containing nine channels, can be represented by a vector of wavelet packet transform component energies with length for 144 (16×9).

In the scenarios considering noise, white Gaussian noise is injected in all three data sets, that is, the network is trained by noisy data and also is tested by noisy data.

As for the conjuration of computing platform, all the numerical experiments are performed on the same software and hardware platform. MATLAB is used to program the structural response numerical simulation program. Python package Tensorflow and Keras (Chollet et al., 2015) are employed to establish the neural network architecture with GPU acceleration. On the other hand, the hardware platform has the specification that the processor and video card type are Inter Core i7-4720HQ and NVIDA GTX970M, respectively.

The training of CNN is accelerated by GPU in all scenarios. Meanwhile, the computation of wavelet packet transform component energy is executed on the CPU, leading to a much lower computational efficiency. Detailed numbers are listed as follows: single training epoches of all the CNNs cost the same time, 13 seconds, however, different numbers of epoch are consumed to achieve the best performance in validation. In single damage localization task, the scenario with noise needs less time to achieve the best. Concretely, 954 and 737 epoches are expanded in noise-free and noisy situation, respectively. Similarly, the situation with noise in

Table 8
The accuracy of scenario 3

		Predicted damage location												
Count		*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Total*	*%*
Actual damage location	0	677	22	59	3	4	0	0	0	0	0	2	767	88.27
	1	17	626	6	8	14	1	0	1	10	2	1	686	91.25
	2	29	13	711	8	7	7	0	4	1	2	0	782	90.92
	3	20	26	13	706	14	6	9	0	5	3	0	802	88.03
	4	1	26	30	7	547	16	5	4	6	5	0	647	84.54
	5	1	10	6	5	30	621	17	21	14	11	1	737	84.26
	6	2	2	3	14	3	17	574	14	40	29	2	700	82.00
	7	2	1	4	1	8	9	5	631	11	58	0	730	86.44
	8	0	4	0	1	7	7	9	14	717	29	0	788	90.99
	9	3	0	1	0	1	3	2	26	13	706	0	755	93.51
	10	13	26	29	13	22	4	5	17	21	38	610	798	76.44
	Total	765	756	862	766	657	691	626	732	838	883	616	8192	Overall: 86.99

Table 9
The accuracy of scenario 4

		Predicted damage location												
Count		*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*Total*	*%*
Actual damage location	0	632	73	19	6	9	8	2	1	2	5	9	766	82.51
	1	26	579	18	6	9	7	4	1	2	1	14	667	86.81
	2	33	29	678	10	12	7	2	6	1	1	8	787	86.15
	3	18	12	32	616	14	10	10	5	5	5	7	734	83.92
	4	11	22	10	14	577	30	9	5	1	2	7	688	83.87
	5	6	15	14	6	13	618	22	6	5	5	11	721	85.71
	6	6	10	13	3	14	19	551	19	4	4	5	648	85.03
	7	2	7	6	2	7	19	21	594	21	15	4	698	85.10
	8	4	2	1	3	14	30	16	38	588	57	15	768	76.56
	9	7	4	8	5	7	23	14	22	23	619	10	742	83.42
	10	69	138	68	22	46	86	54	88	46	30	326	973	33.50
	Total	814	891	867	693	722	857	705	785	698	744	416	8192	Overall: 77.86

multiple damages identification spends 1,609 epoches, less than 3,462 epoches costed in the noise-free situation. It may be more difficult in training on a noisy data set, so that a further current best performance is harder to get than in the case without noise. As for the detector with wavelet package component energy, it consumes 190 seconds per epoch in training for the low efficiency of computing wavelet package on CPU; 63 and 91 epoches are spent in noise-free and noisy situation, respectively.

The testing processing takes much less time than training does. Every CNN has the same speed of testing, 8.92×10^{-4} seconds per example. Meanwhile, a speed for 2.07×10^{-2} seconds per example is measured in the cases of detectors with wavelet package component energy.

3.2 Single damage localization accuracy

Damage localization is concentrated and the neural network should "observe" the responses data, then "diagnose" the damage position. A total of 11 location categories are prepared. Specifically, categories from no. 0 to no. 9 represent the damages are located on element no. 0 to no. 9, and category no. 10 represents the intact condition.

Figure 9 shows the accuracies of the four scenarios. It can be found that the CNN achieves a high accuracy, 94.57%, in the noise-free situation. And the feature extractor of wavelet packet energy also obtains a good result for 93.90% accuracy. When white noise is injected into the data set, the CNN still gets a considerable accuracy for 86.99%. It largely exceeds the

accuracy, 77.86%, in scenario 4. It seems that the wavelet packet transform component energy is more sensitive to noise.

In addition, more details can be found in Table 6 to Table 9. A total of 8,192 examples are randomly chosen from the testing data set to estimate the model performance. These tables show how models predict damage location in a statistical view. Table 6 shows the neural network makes good predictions and only a small number of mistakes, which are mainly neighboring the true damage location. As same as the previous scenario, scenario 2 is also carried out without noise, and Table 7 presents its performance. It can be observed that errors occur also in adjacent elements of the diagonal, implying the locations near the true damage location are more error prone. Moreover, detecting the intact condition seems more difficult that only 71.94% accuracy is obtained. Overall, the performance of this scenario is similar with scenario 1, except the poor accuracy of detecting intact condition. When noise is considered, the CNN still performs well as detailed in Table 8. Excepting the errors near the true damage location, the other errors mainly appear in the elements on antidiagonal of the table, meaning the network predicts the wrong damage positions at the symmetric position to the true damage position. Besides, the lower accuracy of detecting intact condition indicates the noise makes the gap between the intact condition and the other damage conditions ambiguous, leading to the difficulty in identifying the intact condition. As for scenario 4, a worse performance is shown in Table 9. The noise deteriorates the performance of wavelet packet transform component energy largely in detecting the intact condition.

To get a further intuition, a visualization technology called "t-SNE" (Maaten and Hinton, 2008) is employed to visualize the distribution of predicted damage location label. The t-SNE is designed to create a 2- or 3-D map of high-dimensional data, trying to hold both local and global data structure at the same time. It means nearby points in original high-dimensional space are still close in new low-dimensional space and the far away points are still far. It is a useful technique to reveal data structure in a single map.

Four 2-D maps of predicted location, distribution for the four scenarios are displayed in Figure 10 to Figure 13 and each figure shows 256 points downsampled form the test data set. The two dimensions called "t-SNE axis-1" and "t-SNE axis-2" in these figures are created by the t-SNE algorithm to find the best representation of the relationship of data in the original high dimension. It can be identified in Figure 10 that points in the same class are close to each other and there are clear gaps between each two clusters. It also can be observed that a few points are in the wrong cluster, such as the point of 10 in the cluster of zeros and the points of three in the cluster of two. In scenario 2, different classes are approximately divided as illustrated in Figure 11. Nevertheless, there are some confusions in some clusters. Especially category 10, the intact condition, is dispersed in multiple clusters. In scenario 3, the performance of the CNN is influenced by noise. And, the corresponding 2-D map, Figure 12, shows the points with the same categories gather in the same cluster, although a few points are at the wrong place. Besides, the points in close categories are more fallible, such as the points of other categories in the cluster of zero. Finally, in scenario 4, the cluster of 10 with such a wide range in Figure 13 makes the identification hard, indicating the poor accuracy.

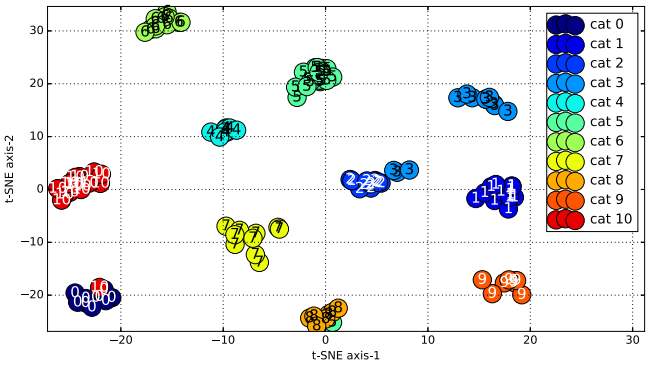

Fig. 10. 2-D visualization of prediction for scenario no. 1.

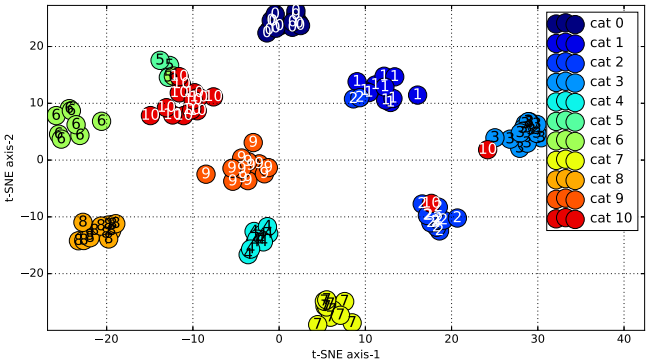

Fig. 11. 2-D visualization of prediction for scenario no. 2.

3.3 Multiple damages identification results

A trial of multiple damages identification is also conducted. As described in Section 2.5.2, at most three damages are introduced into the numerical model, and both situations whether considering noise are tested. To achieve multiple damages identification, a modification is conducted on the label definition that each element of the label vector indicate the damage level of its

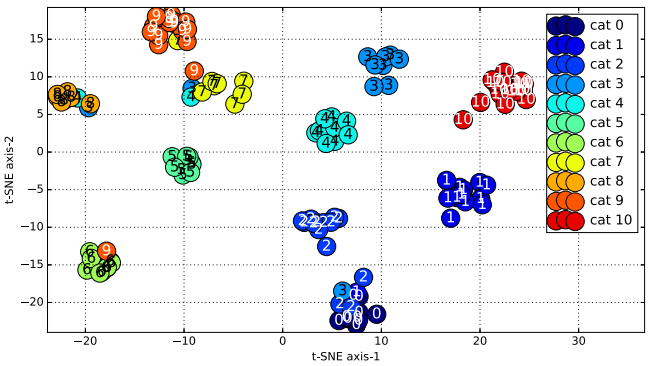

Fig. 12. 2-D visualization of prediction for scenario no. 3.

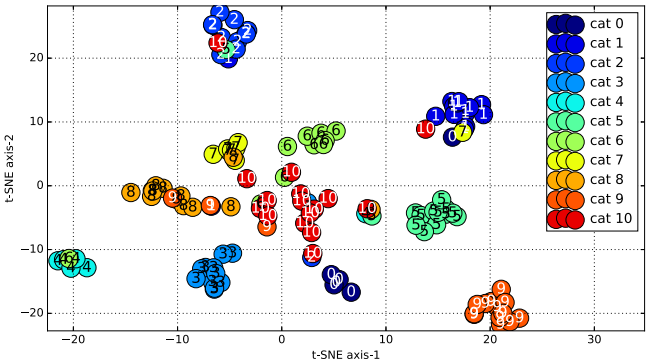

Fig. 13. 2-D visualization of prediction for scenario no. 4.

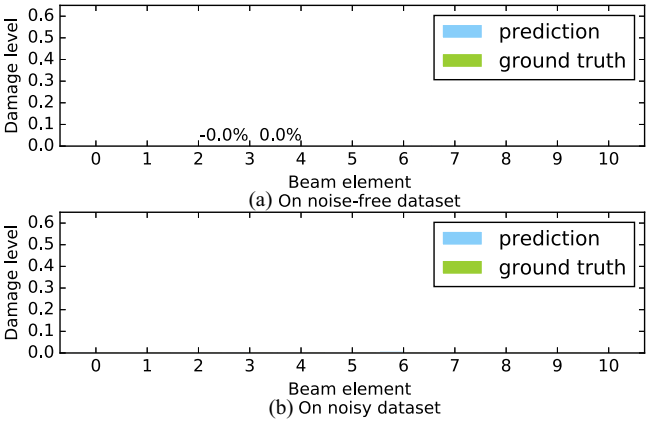

Fig. 14. Prediction of damage level for zero damage case on noise-free (a) and noisy data set (b).

corresponding beam element. Subsequently, mean squared error is utilized as the objective function to measure the distance between the true label and the predicted label.

The performance of the trained neural network is tested on the testing data set. A tiny mean squared error, 1.27×10^{-4}, is achieved on the noise-free situation, whereas the one for 8.33×10^{-4} is obtained on the noisy situation. To get a further insight in the identifi-

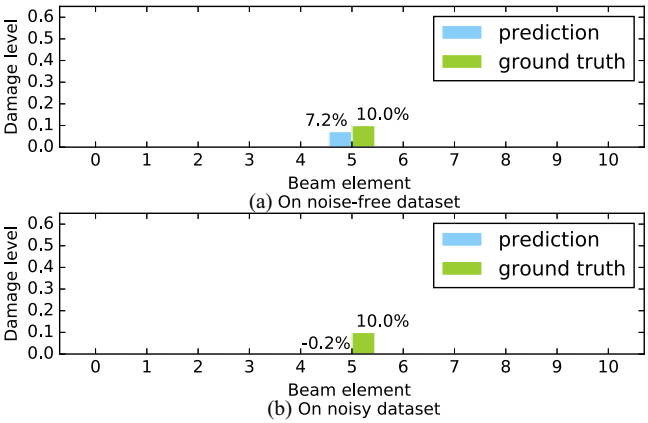

Fig. 15. Prediction of damage level for single damage case on noise-free (a) and noisy data set (b).

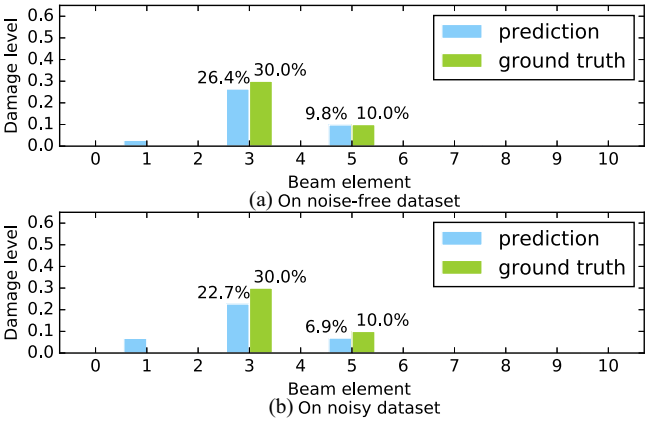

Fig. 16. Prediction of damage level for two damages case on noise-free (a) and noisy data set (b).

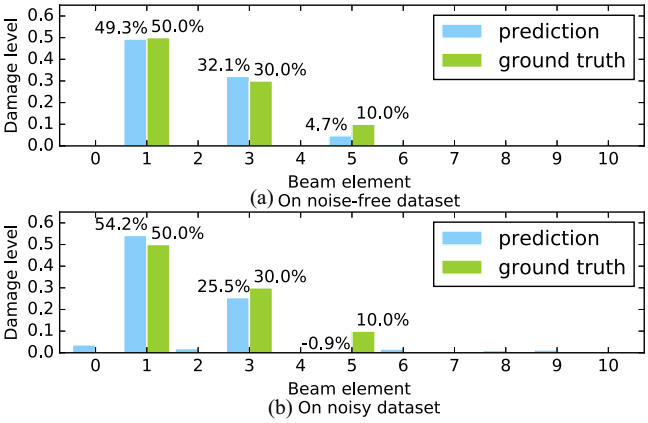

Fig. 17. Prediction of damage level for three damages case on noise-free (a) and noisy data set (b).

cation performance, four examples are prepared to test the learned neural network, including situations containing zero to three damages and low to high damage level.

Figure 14 to Figure 17 show comparisons between the prediction of damage label and the true damage label.

In Figure 14, both models on the two noise conditions exhibit an excellent prediction on an example of intact condition: both predictions of damage level are close to zero.

The prediction of a single tiny damage for level 10% is demonstrated in Figure 15. An excellent result, only 2.8% error and correct localization, is obtained by the neural network trained on the noise-free data set. However, when noise is considered, it appears to be confused, judging the example with tiny damage being in the intact condition.

When another damage with level 30% is added on element 3, as shown in Figure 16, the noise-free model also gives a good prediction, that is, correct localization and little error on level 30% damage are achieved. A reasonable result is obtained in the noisy case that the error on element 3 is less than 8%, nevertheless, the tiny damage is still underestimated and a little mistake is made on element 1.

In the case of three damages as illustrated in Figure 17, the noise-free model achieves an almost perfect prediction except the underestimation on damage level 10% still exists. Although the model trained on noisy data set still cannot recognize tiny damage on element 3, it gives an excellent prediction on the other damages.

In general, the CNN also achieves a reasonable result on the multiple damages identification task. Despite the difficulty in detecting tiny damage, our method shows its excellent performance of damage localization and damage degree prediction. In addition, the reliability is also exhibited by the models giving low damage estimation on all the intact elements.

3.4 Physical interpretation through visualization

As discussed in Section 2.6, although the CNN shows its excellent performance, it is inscrutable for humans, leading to distrust in its reliability. In other words, one may not understand how a neural network works and be not clear when it will make mistakes. This problem is particularly severe when a neural network works on a responsible task, just as structural damage detection. Hence, the effort of finding physical interpretations to help the human understand how the neural network works is significant.

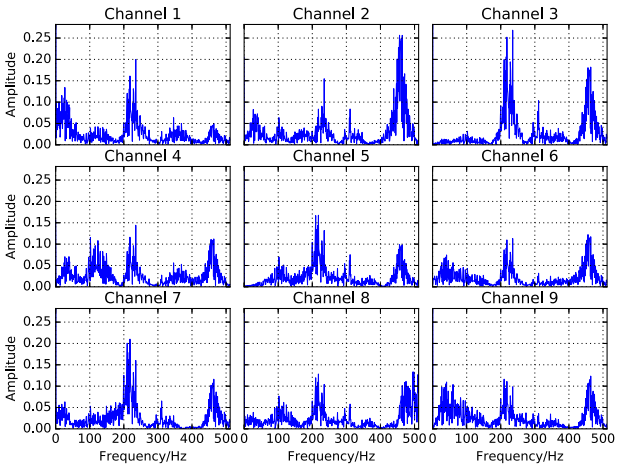

Fig. 18. Spectrum of the learned feature in visualization point 1, the peaks are centered at 52.1 Hz, 210.1 Hz, and 484.7 Hz, the third, sixth, and ninth natural frequency of the structure.

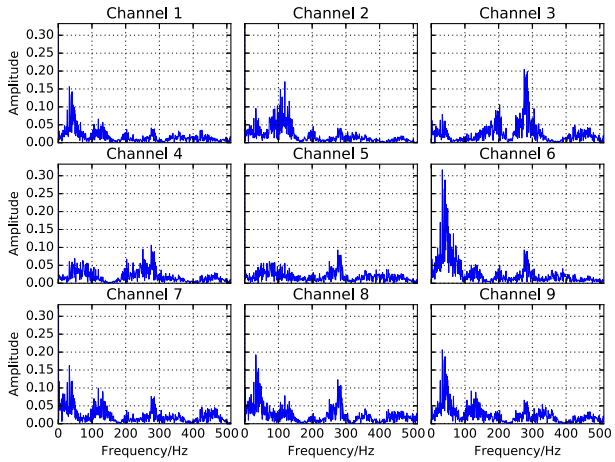

Fig. 19. Spectrum of the learned feature in visualization point 1, the peaks are centered at 52.1 Hz and 287.7 Hz, the third and seventh natural frequency.

In this section, using the method mentioned in Section 2.6, an interpretation of our CNN is demonstrated by showing what features are learned by each hidden layer. The representations of the learned features are obtained by maximizing the average of the target feature map over the input, that is, the optimized input makes the target neuron most active. If the input deviates from the optimized state, the corresponding activation value of the neuron will decline rapidly. Specifically, the CNN in scenario 3 is discussed in detail here for its robustness to noise, implying the network has learned some essential characters behind the data. From bottom to top, four visualization points in our CNN are chosen, as displayed in Figure 2. On account of the difficulty in directly getting insight from time-sequence data

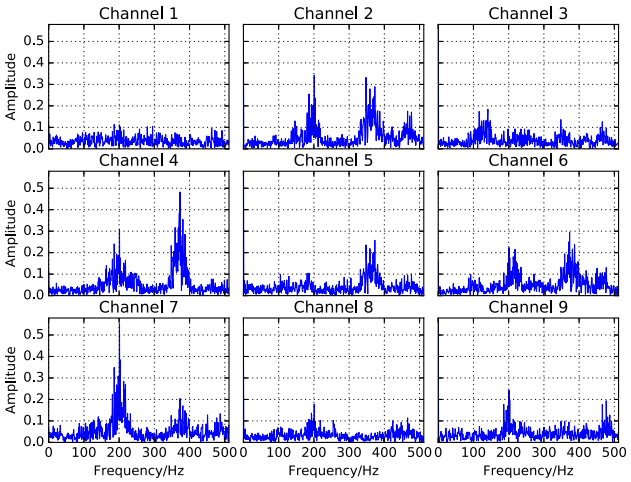

Fig. 20. Spectrum of the learned feature in visualization point 2, the peaks are centered at 210.1 Hz and 379.0 Hz, the sixth and eighth natural frequency of the structure.

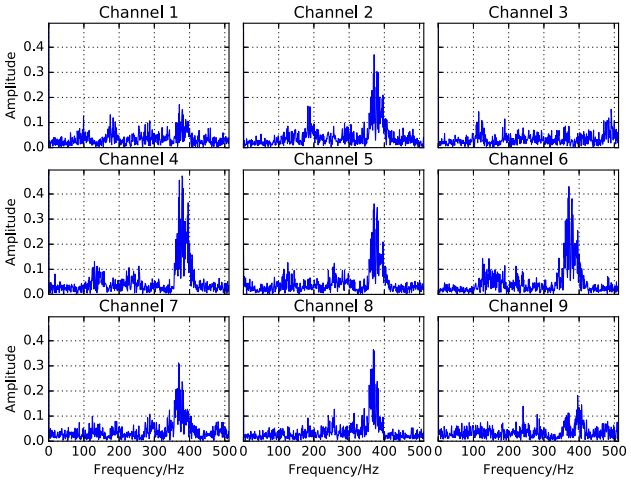

Fig. 21. Spectrum of the learned feature in visualization point 2, the peak is centered at 379.0 Hz, the eighth natural frequency.

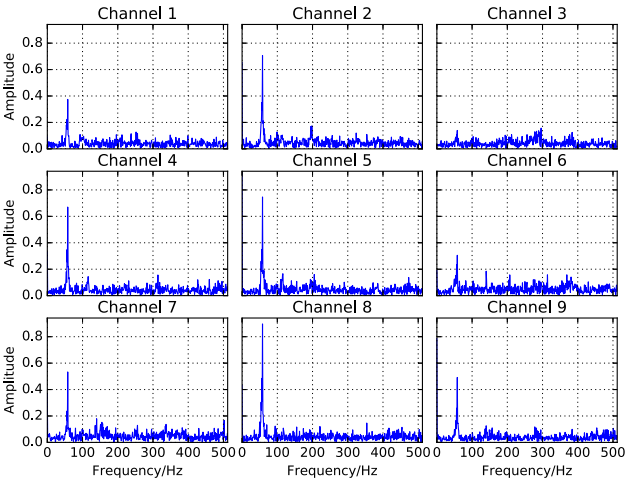

Fig. 22. Spectrum of the learned feature in visualization point 3, the peak is centered at 52.1 Hz, the third natural frequency of the structure.

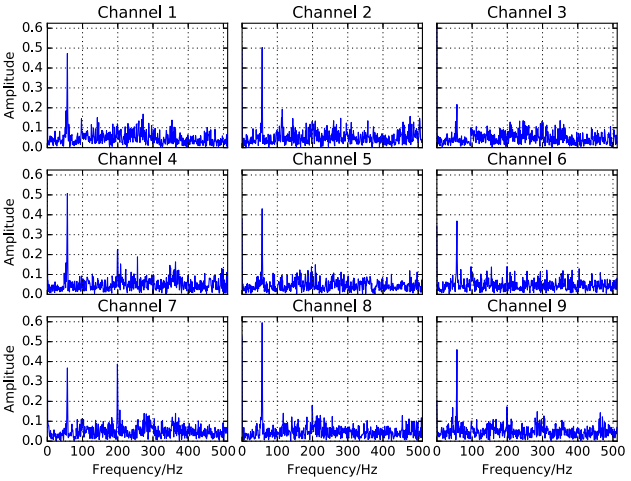

Fig. 23. Spectrum of the learned feature in visualization point 3, the peaks are centered at 52.1 Hz and 210.1 Hz, the third and sixth natural frequency.

for humans, all the representations of the learned features are transformed into frequency domain.

In the bottom of the network, both layers at visualization points 1 and 2 in Figure 2 have 32 kernels. Some of these kernels are visualized and demonstrated first. Figure 18 and Figure 19 illustrate the spectrums of two kernels at visualization point 1. Both figures show that the kernels in the first convolutional layer seem to have learned a band-pass filter. Moreover, although the bandwidths are wide and many burrs can be found in a large range of frequency, the peaks are centered at the natural frequency of the structure. Concretely, Figure 18 shows the third, sixth, and ninth natural frequency (52.1 Hz, 210.1 Hz, and 484.7 Hz) and Figure 19 shows a band-pass filter with two peaks centering on the third and seventh natural frequency (52.1 Hz and 287.7 Hz). Besides, there is also all-pass filter in this layer, which means the origin signals can be received by subsequent layers. In the next visualization point, it is obvious that better filters are learned as displayed in Figure 20 and Figure 21. Specifically, both single-band and multiband filters have a narrower bandwidth, and burrs are less than the last layer.

It, therefore, can infer that the bottom layers have learned elemental features of the data, which are the main frequency components in this case. Furthermore, it can be found that the learned features become more distinct in the deeper layer.

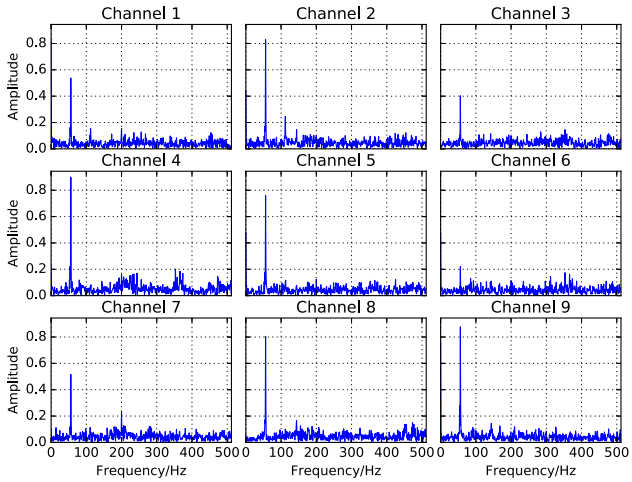

Fig. 24. Spectrum of the learned feature in visualization point 4, the third natural frequency can be found at 52.1 Hz.

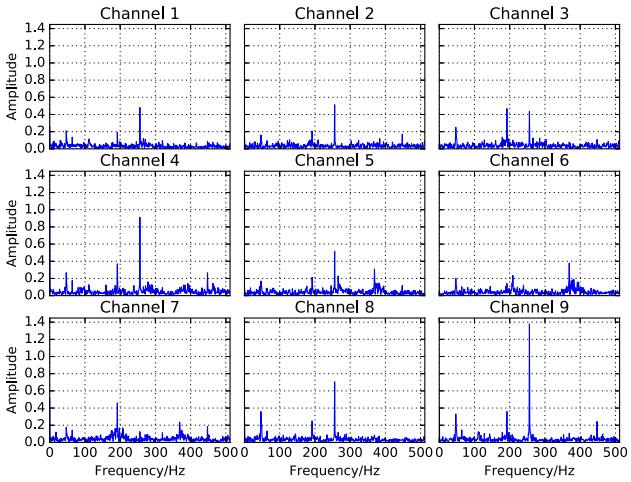

Fig. 25. Spectrum of the learned feature in visualization point 4, the third, sixth, and seventh natural frequency can be found at 52.1 Hz, 210.1 Hz, and 287.7 Hz.

Subsequently, some kernels in visualization point 3 are illustrated in Figure 22 and Figure 23. The number of kernels in this layer, a total of 64, is more than the last layer. In general, the learned features become more distinct than point 2. The bandwidth is so narrow that the peak becomes a "line" in Figure 23. The single-band filter also has a narrow bandwidth as illustrated by Figure 22. Furthermore, it is a significant relationship between different channels in Figure 22: if the steeple in every figure is picked out, redrawn with appropriate signs and connected, it would appear a shape of the third mode of the structure. To confirm this assumption, the corresponding optimized time–sequence data are visualized as a video. According to this video, it is found that the structure dominantly vibrates in the third mode, though some high-frequency components also can be observed. Figure 26 shows a comparison between the shape of the structure in one frame of the video and the true third mode. They are extremely similar.

It is reasonable to suppose not only the features learned by previous layers are refined further, but also the links between channels are intensified in the middle layers. As a result, the learned features become high level and understandable with mechanical meaning, such as the concept of third mode.

Finally, the kernels at visualization point 4, the top of the convolutional layers, show their more abstract and high-level features. As shown in Figure 24, the third mode filter also can be found in this layer and the bandwidth is narrower. Figure 25 demonstrates a multi-band filter holding three bands of frequency with narrow bandwidths. Besides, different weights of amplitude can be found in different channels. It implies a concept of "mode combination" may be learned in this layer and

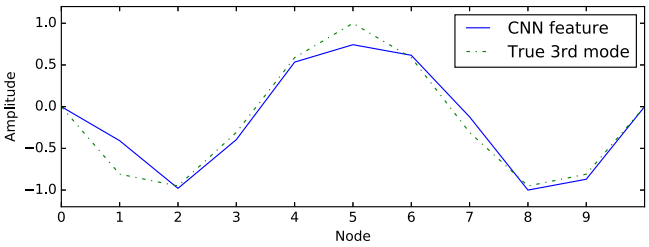

Fig. 26. A shape of third mode in a kernel representation.

these features are specialized in recognizing a combination of specific modes, that is, if the balance of each component changes in the input data, the activation of this neuron will also have a change.

In this layer, the learned features become high level and understandable with mechanical meaning. In mechanics, the response of a linear system is a linear combination of its modes. This mechanism hides behind the data. However, the network has learned it independently to achieve the damage localization, such as it has learned the concept of mode combination described previously. Notice that all these features are learned on the noisy data set; it only uses these essential features that can fight against the noise. That is the reason why the neural network can keep a reasonable performance on the noisy data set.

It is noteworthy that the other features not shown in this section have similar property with the ones that have been shown. For instance, most of these features are also band pass filters but have different weights of amplitude in different channels or have different levels of coarseness.

Table 10
Hierarchical features learned by deep learning models in different tasks

Goal	*Original input*	*Low-level feature*	*Medium-level feature*	*High-level feature*
Structural damage detection	Structure responses	Frequency bands	Vibration modes	Combinations of modes
Speech recognition	Speech records	Frequency bands	Phonemes	Words
Image recognition	Pixels	Edges	Object parts	Object models

In brief, the learned features show a characteristic of hierarchy: they become more and more distinct and abstract layer by layer and even mechanical concepts are learned in the deep layer. Moreover, the features from high layers are constituted by the ones from low layers in a nonlinear way, so that abstract features can be learned. This implies that a deep neural network can independently learn essential characters hidden behind the data to achieve its goal. Table 10 may lend further insight into how deep learning models work with learned hierarchical features by comparing these features between different tasks, such as speech and image recognition.

4 DISCUSSION AND CONCLUSION

In this article, a novel structural damage detection approach using the deep CNN is proposed. It can automatically extract features from low-level waveform signals instead of relying on hand-crafted features. Interesting results are obtained: (1) the CNN shows its excellent accuracy, even though the data set is noisy; (2) a reasonable performance is also observed in the multiple damages identification; and (3) by visualizing hidden layers, it is found that hierarchical features are learned layer by layer and even mechanical concepts are learned in deep layers, such as vibration modes and their combination.

This study extends the application of neural network method in structural damage detection. As mentioned in Section 1, neural network approaches are applied in structural damage detection field for years. However, almost all of these researches are based on hand-crafted features from mechanics or statistics and use a neural network as a classifier or a regressor to achieve damage detection. In contrast, this article only uses a CNN to link the responses data and damage information. It is a fresh attempt at applying neural network technology in this issue.

On the other hand, it provides a new framework to tackle this problem. To solve a scientific problem, traditional methodology tends to analyze it from "reason" to "result," then leading to "application." In other words, researchers usually interpret the data ("result") by ensuring the principle ("reason") using their knowledge and logic, then use these principles to solve problems ("application"). However, the real world is complex, and a phenomenon is always caused by many factors. Instead of considering some of these factors and establishing a hypothesis, another methodology is considering all the factors implicitly by directly linking data ("result") and problem-solving ("application"). A relationship can be established between arbitrary two or more factors by learning algorithm, if only relating data have been prepared. The learned relationship then can be used in various applications. This is the key idea in the term of "big data." This article follows this methodology. Specifically, the network achieves structural damage detection without knowledge from human experts, only depending on the data.

The proposed method has demonstrated its remarkable performance and provides a new framework for future studies to detect structural damage.

First of all, excellent classification accuracy is obtained even in the situation with noise. Such a good accuracy is vital for a detection system. In addition, a reasonable performance is also achieved in the multiple damages identification.

Second, the neural network can automatically extract features of the structure from low-level sensor data directly. As a result, a lot of time could be saved. Moreover, the learned features are optimized to facilitate the final classification accuracy instead of being limited in specific physical definition. For instance, the concept of vibration mode is found in the high layer of convolutional layers, however, extracting vibration modes is not the goal of the network, it just found some specific modes are helpful for damage detection in this case. Besides, even these vibration mode features are used more complexly in the later layers, that is, they are combined and transformed nonlinearly in the subsequent layers. Instead, human researchers usually derive new indices from some extant concepts, then validate their performance.

Third, the methodology, learning from data, has a good extensibility. Notice that the network does not know any structure information but responses data, in other words, it can be applied to any structure.

Similarly, although damage localization is concentrated in this work, the method also can be easily generalized to detection task as shown in Section 3.3. Besides, though the linear structure is discussed in this article and hand-crafted features still have a reasonable performance, there are other situations: assuming the problem becomes more complex and the existing human knowledge might not be able to represent its characters perfectly, the neural network may still keep the potential to achieve a good result, because of its ability of self-evolution by constantly learning from data. Nevertheless, human experts are still significant. A supervisor is necessary for any artificial intelligence system, and human experts should validate whether the system is irregular or not. Moreover, scientists might be able to get insight into complex problems from a neural network by visualizing layers in it.

Future work will focus on the problems caused by the dependence on numerical model. The proposed method needs a large amount of data to involve different damage scenarios. However, a real-world structure only has one life-cycle and the data from various damage scenarios are unavailable. In addition, almost all the structures on active duty are in a condition near intact, leading to scarcity of the data carrying explicit and distinct damage information. An accessible approach to deal with this situation may be modeling an accurate FE model of the intact structure and using this model to generate data from different damage scenarios. Nevertheless, distinction between the FE model and the real-world structure always exists. This means a CNN has to learn from FE model and predict on real-world data, even the data sets yield different distributions. It is a problem in transfer learning domain, a topic of machine learning, because the data sets for learning and predicting have different distributions. This problem may be the key point to apply this method on real-world structure.

ACKNOWLEDGMENTS

The authors acknowledge the financial supports from the National Natural Science Foundation of China (11402098), International Science and Technology Cooperation Fund of Qing Hai Province (2014-HZ-822), Science and Technology Plan of Guang Dong Province (2013B021500008), and National Key Laboratory Open Fund of China (SV2014-KF-16).

REFERENCES

Abdel-Hamid, O., Mohamed, A., Jiang, H. & Penn, G. (2012), Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition, in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan, 4277–80.

Adeli, H. & Jiang, X. (2005), Dynamic wavelet neural network for nonlinear identification of highrise buildings, *Computer-Aided Civil and Infrastructure Engineering*, **5**, 316–30.

Adeli, H. & Jiang, X. (2009), *Intelligent Infrastructure: Neural Networks, Wavelets, and Chaos Theory for Intelligent Transportation Systems and Smart Structures*, CRC Press, Boca Raton, FL.

Adewuyi, A. P., Wu, Z. S. & Serker, N. H. M. K. (2009), Assessment of vibration-based damage identification methods using displacement and distributed strain measurements, *Structural Health Monitoring*, **8**(6), 443–61.

Amezquita-Sanchez, J. & Adeli, H. (2015), Feature extraction and classification techniques for health monitoring of structures, *Scientia Iranica. Transaction A, Civil Engineering*, **22**(6), 1931.

Amezquita-Sanchez, J. P. & Adeli, H. (2016), Signal processing techniques for vibration-based health monitoring of smart structures, *Archives of Computational Methods in Engineering*, **23**(1), 1–15.

An, Y., Spencer, B. F. & Ou, J. (2015), A test method for damage diagnosis of suspension bridge suspender cables, *Computer-Aided Civil & Infrastructure Engineering*, **10**, 1–14.

Blachowski, B., An, Y., Spencer, B. F. & Ou, J. (2017), Axial strain accelerations approach for damage localization in statically determinate truss structures, *Computer-Aided Civil & Infrastructure Engineering*, **32**(4), 304–18.

Cao, M., Xu, W., Ostachowicz, W. & Su, Z. (2014), Damage identification for beams in noisy conditions based on teager energy operator-wavelet transform modal curvature, *Journal of Sound & Vibration*, **333**(6), 1543–53.

Cavadas, F., Smith, I. F. C. & Figueiras, J. (2013), Damage detection using data-driven methods applied to moving-load responses, *Mechanical Systems & Signal Processing*, **39**(1-2), 409–25.

Cawley, P. & Adams, R. D. (1979), The location of defects in structures from measurements of natural frequencies, *Journal of Strain Analysis for Engineering Design*, **14**(2), 49–57.

Cha, Y.-J., Choi, W. & Buyukozturk, O. (2017), Deep learning-based crack damage detection using convolutional neural network, *Computer-Aided Civil & Infrastructure Engineering*, **32**(3), 2013–14.

Chen, F., Jahanshahi, M. R., Wu, R. & Joffe, C. (2017), A texture-based video processing methodology using Bayesian data fusion for autonomous crack detection on metallic surfaces, *Computer-Aided Civil & Infrastructure Engineering*, **32**(4), 271–87.

Chollet, F., et al. (2015), Keras. Available at: https://github.com/fchollet/keras, accessed October 2017.

Daubechies, I. & Heil, C. (1992), *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Dutta, A. & Talukdar, S. (2004), Damage detection in bridges using accurate modal parameters, *Finite Elements in Analysis & Design*, **40**(3), 287–304.

Erhan, D., Bengio, Y., Courville, A. & Vincent, P. (2009), Visualizing higher-layer features of a deep network, University of Montreal, **1341**, 3.

Farrar, C. R. & Iii, G. H. J. (1997), System identification from ambient vibration measurements on a bridge, *Journal of Sound & Vibration*, **205**(1), 1–18.

Glorot, X., Bordes, A. & Bengio, Y. (2011), Deep sparse rectifier neural networks, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, University of Lleida, Lleida (Catalonia), Spain, 315–23.

Hou, Z. K., Noori, M. N. & Amand, R. S. (2000), Wavelet-based approach for structural damage detection, *Journal of Engineering Mechanics*, **126**(7), 677–83.

Huang, Y., Beck, J. L., Wu, S. & Li, H. (2014), Robust Bayesian compressive sensing for signals in structural health monitoring, *Computer-Aided Civil & Infrastructure Engineering*, **29**(3), 160–79.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: accelerating deep network training by reducing internal covariate shift, *Computer Science*.

Jiang, X. & Adeli, H. (2007), Pseudospectra, music, and dynamic wavelet neural network for damage detection of highrise buildings, *International Journal for Numerical Methods in Engineering*, **71**(5), 606–629.

Kingma, D. P. & Ba, J. (2014), Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, **25**(2), 2012.

Law, S. S., Li, X. Y., Zhu, X. Q. & Chan, S. L. (2005), Structural damage detection from wavelet packet sensitivity, *Engineering Structures*, **27**(9), 1339–48.

Le, Q. V. (2011), Building high-level features using large scale unsupervised learning, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, BC, Canada, 8595–98.

LeCun, Y. (1989), Generalization and network design strategies in Pfeifer R., Schreter Z., Fogelman F., and Steels L. (eds.),*Connectionism in Perspective*, Elsevier, Zurich, Switerland, 143–55.

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

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989), Backpropagation applied to handwritten zip code recognition, *Neural Computation*, **1**(4), 541–51.

Lee, H., Pham, P. T., Yan, L. & Ng, A. Y. (2009), Unsupervised feature learning for audio classification using convolutional deep belief networks, in *Advances in Neural Information Processing Systems 22: Conference on Neural Information Processing Systems 2009. Proceedings of A Meeting Held 7-10 December 2009*, Vancouver, British Columbia, Canada, 1096–1104.

Lee, J. J., Lee, J. W., Yi, J. H., Yun, C. B. & Jung, H. Y. (2005), Neural networks-based damage detection for bridges considering errors in baseline finite element models, *Journal of Sound & Vibration*, **280**(3-5), 555–78.

Li, Z., Park, H. S. & Adeli, H. (2017), New method for modal identification of super high-rise building structures using discretized synchrosqueezed wavelet and Hilbert transforms, *The Structural Design of Tall and Special Buildings*, **26**(3), https://doi.org/10.1002/tal.1312.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013), Rectifier nonlinearities improve neural network acoustic models, in *Proceedings ICML*, **30**(1).

Maaten, L. v. d. & Hinton, G. (2008), Visualizing data using t-SNE, *Journal of Machine Learning Research*, **9**(Nov), 2579–605.

Mehrjoo, M., Khaji, N., Moharrami, H. & Bahreininejad, A. (2008), Damage detection of truss bridge joints using artificial neural networks, *Expert Systems with Applications*, **35**(3), 1122–31.

Mu, H. & Yuen, K. (2016), Ground motion prediction equation development by heterogeneous Bayesian learning, *Computer-Aided Civil & Infrastructure Engineering*, **31**(10), 761–76.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning*, (ICML-10), Haifa, Israel, 807–14.

Newmark, N. M. (1959), A method of computation for structural dynamics, *Journal of the Engineering Mechanics Division*, **85**(1), 67–94.

Pandey, A. K., Biswas, M. & Samman, M. M. (1991), Damage detection from changes in curvature mode shapes, *Journal of Sound Vibration*, **145**(2), 321–32.

Salawu, O. S. (1997), Detection of structural damage through changes in frequency: a review, *Engineering Structures*, **19**(9), 718–23.

Shabbir, F. & Omenzetter, P. (2015), Particle swarm optimization with sequential niche technique for dynamic finite element model updating, *Computer-Aided Civil & Infrastructure Engineering*, **30**(5), 359–75.

Shi, Z. Y., Law, S. S. & Zhang, L. M. (2000), Structural damage localization from modal strain energy change, *Journal of Sound & Vibration*, **126**(12), 825–44.

Sietsma, J. & Dow, R. J. F. (1991), Creating artificial neural networks that generalize, *Neural Networks*, **4**(1), 67–79.

Simonyan, K., Vedaldi, A. & Zisserman, A. (2013), Deep inside convolutional networks: visualising image classification models and saliency maps, arXiv preprint arXiv:1312.6034.

Simonyan, K. & Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.

Sohn, H. & Farrar, C. R. (2001), Damage diagnosis using time series analysis of vibration signals, *Smart Material Structures*, **10**(3), 446–51.

Szegedy, C., Liu, W., Jia, Y. Sermanet, P., Reed, S., Anguelov, D. & Rabinovich, A. (2015), Going deeper with convolutions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, 1–9.

Wu, R., Yan, S., Shan, Y., Dang, Q. & Sun, G. (2015), Deep image: scaling up image recognition, arXiv preprint arXiv:1501.02876, **7**(8).

Yang, J. N., Lei, Y., Lin, S. & Huang, N. E. (2004), Hilbert-Huang based approach for structural damage detection, *Journal of Engineering Mechanics*, **130**(1), 85–95.

Young-Jin, C. & Oral, B. (2015), Structural damage detection using modal strain energy and hybrid multiobjective optimization, *Computer-Aided Civil and Infrastructure Engineering*, **30**(5), 347–58.

Zhou, Y. T. & Chellappa, R. (1988), Computation of optical flow using a neural network, in *IEEE International Conference on Neural Networks*, San Diego, CA, 71–78.