Identification of Structural States by Acceleration Data based on a Convolutional Neural Network

Kitami Institute of Technology ○ Student member Youqi Zhang
Kitami Institute of Technology Member Yasunori Miyamori
Kitami Institute of Technology Member Takehiko Saito
Kitami Institute of Technology Member Shuichi Mikami

1. Introduction

As the aging bridges have become an unavoidable problem, the study on bridge structural health monitoring is greatly necessary for the purpose of detecting potential structural damage and determining the current structural state.

The classical structural damage detection methodologies[1-4] have the same identification logic which needs to define some damage indicators or thresholds artificially. Structural damages are indicated when some indicators emerge singular values. The process of damage detection shows forward logic relationship between monitoring data and damage information. Obviously, the forward logic methodologies require lots of handicraft low-level features, which may unable to capture more vital information of the structure.

With the rapid development of calculation capacity of computer hardware, deep learning[5] shows its advantages dealing with capturing deep features for classification tasks. Machine learning based methods avoid the artificial feature extraction process, and link the input and output data end-to-end by a computer-generated calculation model.

Under this circumstance, we attempt to apply deep learning method to automatically and efficiently extract the in-depth structural features and estimate the structural states. Thus, in this study identification of structural states is carried out only by utilizing a convolutional neural network (CNN)[6] and raw vibration data of a simply supported steel girder bridge.

2. Data generation

2.1 Vibration experiment

In order to apply the machine learning methods to the bridge, vibration experiment was conducted to build a large size database. The 6.45m-long steel bridge was shown in Fig. 1. In total 15 accelerometers were installed on the upper flange to collect the vibration data. Sampling frequency is 10000 Hz. Local structural states (mass and stiffness) were changed by fixing additional element on the lower or upper flange of the bridge, as shown in Fig. 2 and 3. The additional element 1 (5.11 kg) is a steel plate on the lower flange which was fixed by 2 clamps. The additional element 2 (3.21 kg) is an actuator which was installed on the upper flange fixed by magnet. Notes that the actuator was not used as exciter in this study.

The experiment consists of 6 scenarios. Scenarios 1-4 were applied by additional element 1 in 4 different positions.

Figure 1　Steel bridge

Figure 2　Additional element 1

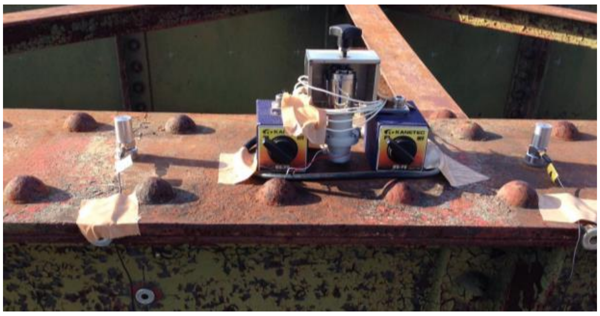

Figure 3　Additional element 2

Scenario 5 was applied by additional element 2. Scenario 6 is a structural state with no additional element. Vertical hammer impulse excitation was applied on 14 positions of the upper and lower flanges. The experimental layout is shown in Fig. 5.

2.2 Database

The database contains in total 8595 measurements of 0.6s-long free damped vibration data which are divided into 6 categories corresponding to the 6 experimental scenarios. Raw acceleration data is directly used without any filtering pre-processing. Each measurement consists 90,000 acceleration data samples (15 channels×6000 samples). The detail

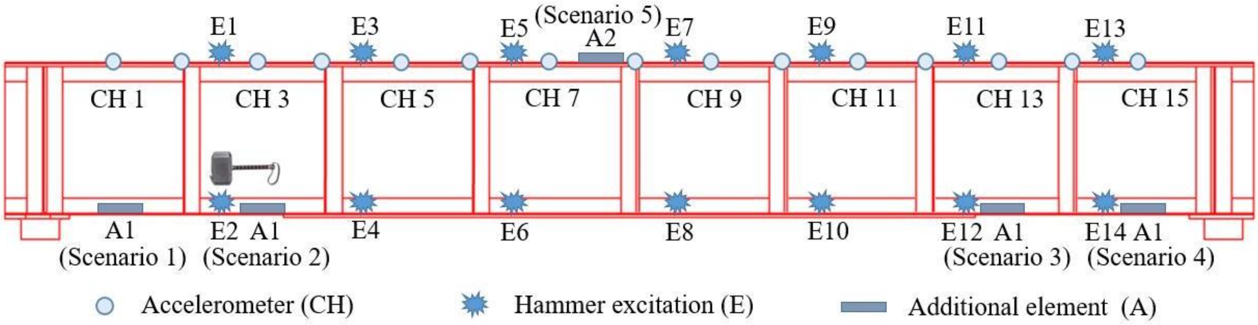

Figure 4　Experimental layout

distribution information is shown in Table 1. One channel waveform of a measurement is shown as an example in Fig. 5.

We adopt random selection of the data to give a more reliable result. For splitting the data, 80% of the dataset is chosen as train set, and the other 20% of the dataset is divided into test set. Note that there is no overlap between train set and test set.

Table 1　Distribution of database

	Amount		Amount
Scenario 1	1279	Scenario 4	1286
Scenario 2	1414	Scenario 5	1181
Scenario 3	1362	Scenario 6	2073
In total			8595

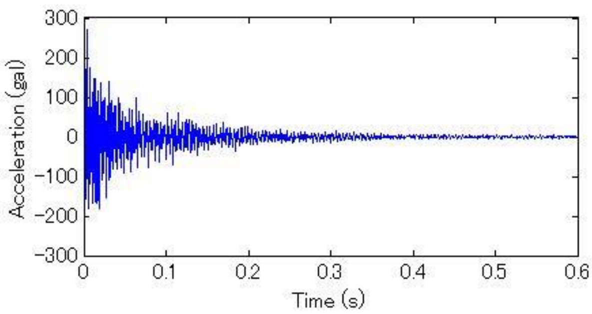

Figure 5　Waveform of one channel of a sample

3. Methodology

In this study, a CNN is designed to classify the data corresponding to the 6 states of the bridge. In order to have a better comprehension of the performance or capacity of the CNN, a fully connected neural network (FCNN)[7] is also designed to make a comparison with the CNN.

3.1 Convolutional neural network

Typical CNN consists of input layer, convolutional layers, pooling layers, flatten layer, fully connected layers, and a softmax layer.

The algorithm of 1-D convolution is shown in Eq. 1. The detail calculation example was shown in Fig 6. The significance of convolution operation is to extract local features of the data.

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

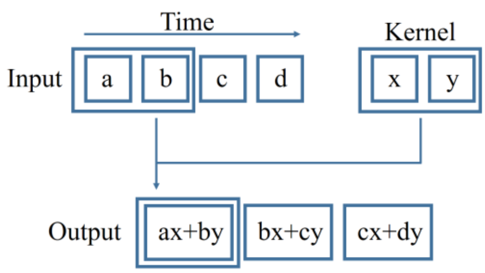

Figure 6　Example of 1-D convolution

Batch normalization[8] operation is also conducted after convolution in this study. During training, for every batch of data, the algorithm calculates the mean and variance, then shifts and scales the origin data to zero-mean and one variance. The operation solved the internal covariate shift problem and increase the convergence rate.

1-D Max pooling is applied after batch normalization in this study. This operation picks out the maximum within its kernel size, step by step. Fig. 7 shows an example of the operation. It makes the neural network to detect more features and reduces the size of data which can improve the computational efficiency of the natural network.

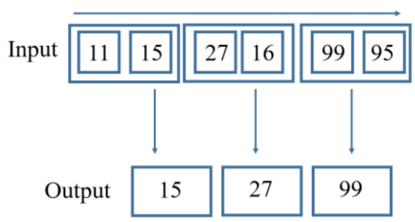

Figure 7　Example of 1-D max pooling

Flatten layer is designed before fully connected layers. The purpose for flatten is to reshape the input matrix into a long vector which can be inputted into the fully connected layers.

Dropout operation[9] is utilized after flatten operation. In brief, it inactivates some neurons during training and reactivates those neurons during test. The operation can increase the convergence rate.

Fully connected layers are the fundamental structure of neural network. The neurons in fully connected layer are linked with all the neurons in forward layer, and calculated by the basic equation as Eq. 2. In Eq.2, *y* means output, *x* means input, *w* means weight, *b* means bias, and *f* means activation function.

Weights and biases are trainable variables, and activation function is manually assigned.

$$y = f(\sum u \times w + b) \quad (2)$$

Relu function[10] is chose as the activation function in this study, as shown in Eq. 3, Fig. 8. It solves the vanishing gradient problem, and adds the nonlinear attribute to the neural network.

$$y = \max(0, u) \quad (3)$$

The softmax layer is arranged as the last layer to output the final result of the classification. The detail algorithm is shown in Eq. 4. The possibilities of all the prediction candidates are evaluated, and the candidate with highest possibility will be output as the final result.

$$y_i = \frac{\exp(u_i)}{\sum_{i=1}^{n} \exp(u_i)} \quad (4)$$

The difference between the labels and neural network outputs will be evaluated by the loss function. The loss function in this study is cross entropy, as shown in Eq. 5. E means the loss, N means the number of training data, x means samples, y means the actual value (label), and a means the output of neural network (prediction).

$$E = -\frac{1}{N}\sum_x [y\ln a + (1-y)\ln(1-a)] \quad (5)$$

After calculating the loss, an optimizer will be utilized to reduce the difference between the neural network outputs and the labels. This is the core process of training.

3.2 Fully connected neural networks

FCNN are also designed to classify the data of 6 states of the bridge. The FCNN has obvious characteristics that all hidden layers in the network are fully connected layers, which is introduced in Section 3.1.

The detailed structure of neural network in this study is shown in Table 3. There are 3 fully connected hidden layers in the network.

4. Results

In this paper, we utilized the 1-D conversion natural network to identify all the structural states. The batch size is set to 512. The dropout ratio is 0.25. The learning rate of Adam optimizer[11] is set to 0.001. For the FCNN, the basic calculation modules are same to the CNN that, batch size:512, activation function: Relu; loss function: cross entropy; optimizer: Adam; and learning rate: 0.001. The detailed structures of CNN and FCNN is shown in Table 2 and 3.

The CNN and FCNN were designed by Python 3.5 in Ubuntu 16.4 OS environment, based on the Tensorflow[12] framework and Keras API.

The FCNN method got the lower classification accuracy 87.45% after training for 200 epochs. It means the FCNN could extract important vibration information of the bridges, but cannot achieve efficient computing in terms of the performance.

From Table 3, it can be found that the 5-layer FCNN has more than 7 million parameters to be trained. Comparing to the CNN (0.4 million parameters, shown in Table 2), the

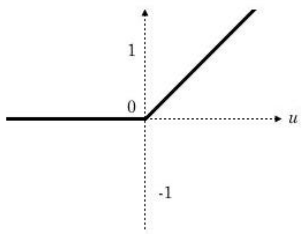

Figure 8　Relu activation function

Table 3　Structure of the fully connected neural network

Layer	Output Shape	Parameter
Input Layer	90000×1	0
Dense1 (Relu)	80×1	7,200,080
Dense2 (Relu)	40×1	3240
Dense3 (Relu)	20×1	1640
Dense4(Softmax)	6	246
Total parameters		7,205,206

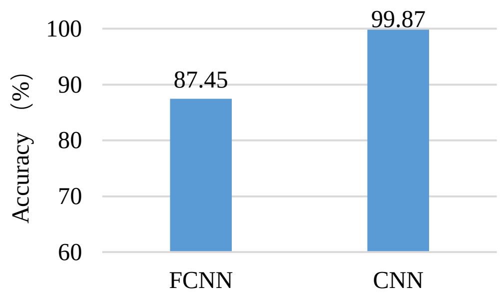

Figure 9　Accuracies of the classification

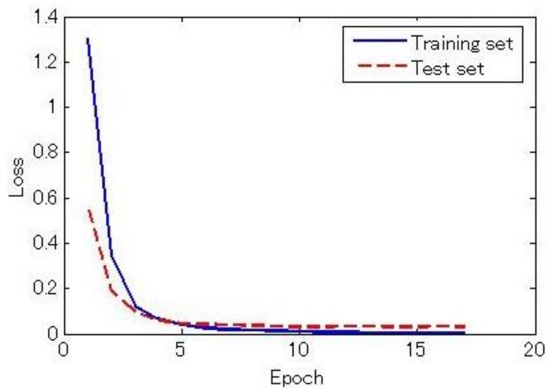

Figure 10　Loss of CNN in every epoch

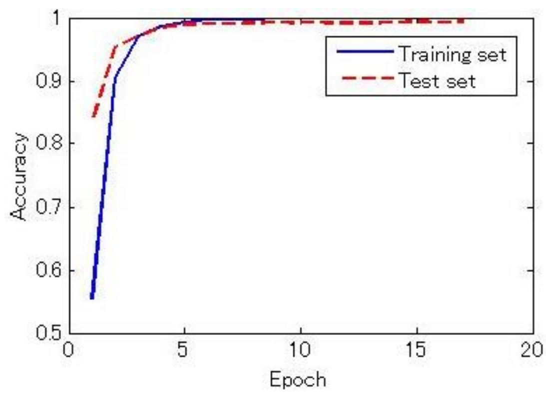

Figure 11　Accuracy of CNN in every epoch

Table 2　Structure of the convolutional neural network

Layer (type)	Output Shape	Kernel number	Kernel size	Stride	Padding	Activation	Parameter
Input Layer	90000×1	None	None	None	None	None	0
Reshape	6000×15	None	None	None	None	None	0
Conv1D	5991×5	5	10	1	Valid	Relu	755
Batch normalization	5991×5	None	None	None	None	None	20
MaxPooling1D	1997×5	None	3	3	Valid	None	0
Flatten	9985	None	None	None	None	None	0
Dropout (0.25)	9985	None	None	None	None	None	0
Dense	40	None	None	None	None	Relu	399,440
Dense	6	None	None	None	None	Softmax	246
Total parameters							400,461

Table 4　Confusion matrix and accuracy of all scenarios

Count		Predicted additional element location							
		S1	S2	S3	S4	S5	S6	Total	Accuracy
Actual additional element location	S1	1277	1	1	0	0	0	1279	99.84 %
	S2	0	1411	0	0	0	3	1414	99.79 %
	S3	1	1	1360	0	0	0	1362	99.85 %
	S4	3	0	1	1282	0	0	1286	99.69 %
	S5	0	0	0	0	1181	0	1181	100 %
	S6	0	0	0	0	0	2073	2073	100 %
	Total	1281	1413	1362	1282	1181	2076	8595	Overall: 99.87%

calculation cost of FCNN is high, while the convergence speed is much slower than CNN.

The CNN got the better performance with 99.87% accuracy. The confusion matrix and accuracy in every category is shown in Table 4. There are only 11 errors comparing to the large number of database (11/8595). Figures 11 and 12 show the convergence speed of the CNN model, and it shows that the network can get a very high accuracy after training for only 5 epochs. Compared to the performance obtained by FCNN, CNN achieves the higher accuracy and convergence of speed.

5. Conclusions

Different bridge states are classified by a CNN and a FCNN in this study. CNN shows overwhelming superiority to FCNN method, considering on classification accuracy, convergence speed, calculation cost and efficiency.

High accuracy of states identification by CNN (99.87%, 8584/8595) shows the feasibility of applying CNN method in structural health monitoring field. Future work will focus on identification of bridge states in multiple local stiffness changes and distinguishing the severity of different damages.

Acknowledgement

This research was supported by JSPS Grant-in-Aid for Scientific Research(C) Grant Number 15K06176. The contributions of Mr. Masato Nakano, Ms. Sanae Suzuki, and Ms. Yukina Hashizume were greatly acknowledged.

Reference

1) Salawu O. S.: Detection of Structural Damage Through Changes in Frequency: A Review, Engineering Structures, No. 19(9), pp.718-723, 1997.
2) Hou Z. K., Noori M. N., Amand R. S.: Wavelet-Based Approach for Structural Damage Detection, Journal of Engineering Mechanics, No. 126(7), pp.677-683, 2000.
3) Yang J. N., Lei Y., Lin S., et al: Hilbert-Huang Based Approach for Structural Damage Detection, Journal of Engineering Mechanics, No. 130(1), pp.85-95, 2004.
4) Shi Z. Y., Law S. S., Zhang L. M.: Structural damage localization from modal strain energy change, Journal of Sound & Vibration, No. 218(5), pp.825-844, 1998.
5) LeCun Y., Bengio Y., Hinton G. E.: Deep learning, Nature, No. 521, pp.436–444, 2015.
6) Krizhevsky A., Sutskever I., Hinton G. E.: ImageNet Classification with Deep Convolutional Neural Networks, Communications of the Acm, No. 60(2), 2012.
7) Nielsen H.: Theory of the Backpropagation Neural Network. Neural Networks, No.1(1), pp.593-605, 1988.
8) Sergey I., Christian S.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, International Conference on Machine Learning, pp.448-456, 2015.
9) Nitish S., Geoffrey H., Alex K., et al: Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, No. 15.1, pp.1929-1958, 2014.
10) Maas A. L., Hannun A. Y., and Ng A. Y.: Rectifier Nonlinearities Improve Neural Network Acoustic Models, 30th International Conference on machine Learning, 2013.
11) Kingma D. and Ba J.: Adam: a Method for Stochastic Optimization, Computer Science, 2014.
12) Abadi M., Barham P., Chen J., et al: Tensorflow: a System for Large-scale Machine Learning, Google Research, 2015.