Advanced Engineering Informatics 42 (2019) 100991

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Full length article

Sensor data reconstruction using bidirectional recurrent neural network with application to bridge monitoring

Seongwoon Jeong[a,*], Max Ferguson[a], Rui Hou[b], Jerome P. Lynch[b], Hoon Sohn[c], Kincho H. Law[a]

[a] Department of Civil and Environmental Engineering, Stanford University, 473 Via Ortega, Stanford, CA 94305-4020, USA
[b] Department of Civil and Environmental Engineering, University of Michigan, 2350 Hayward St., Ann Arbor, MI 48109-2125, USA
[c] Department of Civil and Environmental Engineering, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea

ARTICLE INFO

Keywords:
Sensor data reconstruction
Artificial neural network
Bidirectional recurrent neural network
Machine learning
Structural health monitoring
Smart structure

ABSTRACT

Sensors are now commonly employed for monitoring and controlling of engineering systems. Despite significant advances in sensor technologies and their reliability, sensor fault is inevitable. Sensor data reconstruction methods have been studied to recover the missing or faulty sensor data, as well as to enable sensor fault detection and identification. Most existing sensor data reconstruction methods use only the spatial correlations among the sensor data, but they rarely consider the temporal correlations among the data. Use of temporal correlations among the sensor data can potentially improve the accuracy for reconstructing the data. This paper presents a data-driven bidirectional recurrent neural network (BRNN) for sensor data reconstruction, taking into consideration the spatiotemporal correlations among the sensor data. The methodology is demonstrated using the sensor data collected from the Telegraph Road Bridge located along the I-275 Corridor in Michigan. The results show that the BRNN-based method performs better than other current data-driven methods for accurately reconstructing the sensor data.

1. Introduction

The last few decades have witnessed a growing use of sensors for the monitoring of engineering systems [1,2]. A sensor-based monitoring system can supplement subjective, qualitative and periodic manual inspection by collecting objective, quantitative and continuous data about the operating conditions of an engineering system. As engineering systems increasingly rely on sensor technology, effective management of a sensor network becomes important. However, due to problems such as noise, poor installation, and harsh environmental conditions, sensor fault is inevitable and even occurs frequently [3,4]. Faulty sensors and data loss could hinder the accurate assessment of the monitored system. Therefore, the ability to accurately estimate the data of a faulty sensor is desirable not only to ensure a healthy sensor network, but also to enhance the management of the monitored system.

Data collected by different sensors in a sensor network may have a certain level of correlation among each other. Sensor data reconstruction methods leverage such correlation to estimate the data of a target sensor based on the data collected by the other sensors [5,6]. There are at least two practical applications for sensor data reconstruction. First, when one or more sensors are faulty, sensor data reconstruction methods can be employed to recover the data of the faulty sensors. Second, when the sensors operate normally, sensor data reconstruction methods can be applied for anomaly detection by comparing the reconstructed data and the measured data [7]. Therefore, the ability to reconstruct sensor data accurately can potentially help engineers make better decisions on sensor network maintenance and on anomaly detection of the monitored system.

Statistical data reconstruction methods that use the spatial correlations among the sensor data have been proposed. For example, Kerschen et al. [8] have proposed a principal component analysis (PCA)-based method that extracts the PCA modes from the training data to perform detection, identification and reconstruction of a faulty sensor. Kullaa [9] has presented a minimum mean square error (MMSE)-based method to reconstruct sensor data using the covariance among the data from different sensors. Machine learning methods have also been applied to sensor data reconstruction using the spatial correlations among the sensor data. With the ability to learn the nonlinear pattern among the data, artificial neural networks (ANNs) have been employed for sensor data reconstruction. For example, feedforward neural network (FNN) has been employed for sensor data reconstruction by structuring a neural network to have the target sensor's data as

* Corresponding author.
E-mail address: seanjeong3@gmail.com (S. Jeong).

https://doi.org/10.1016/j.aei.2019.100991
Received 5 April 2019; Received in revised form 2 August 2019; Accepted 20 September 2019
Available online 18 November 2019

output and the other sensors' data as input [10–13]. These studies have shown that FNN can effectively learn the nonlinear spatial correlations among the data collected from spatially distributed sensors. There have also been efforts to use other machine learning methods for sensor data reconstruction problem. For example, Law et al. [14] discussed the use of support vector regression (SVR) to reconstruct missing sensor data collected from a bridge monitoring system. Most of current statistical and machine learning-based methods, however, have a limitation that they focus mainly on the spatial correlations among the sensor data.

Engineering systems are typically dynamic systems where the system state at a time step affects the system states at the subsequent time steps. Therefore, in addition to spatial correlations, sensor data collected from engineering systems often show temporal correlations. By leveraging both the spatial and the temporal correlations inherent in the data collected from different sensors, sensor data reconstruction can achieve higher accuracy comparing to the methods that consider only the spatial correlations [9]. However, relatively few attentions have considered spatiotemporal correlations for sensor data reconstruction. To learn complex, nonlinear spatiotemporal correlation among sensor data, machine learning models need to be specifically designed to handle sequential (time series) data. Recurrent neural network (RNN) is an ANN architecture specialized in learning temporal pattern from sequential data and has been employed in many applications, such as natural language processing [15,16] and stock price prediction [17]. Given the temporal correlations among the sensor data, RNN can improve the accuracy of sensor data reconstruction. For example, Moustapha and Selmic [18] have applied RNN to learn about the spatial and the temporal correlations among the sensor data for fault detection applications.

One limitation of RNN is that the method considers only the information from the past context. Future context, if available, can potentially help reconstruct sensor data more accurately. This paper discusses an ANN-based sensor data reconstruction method that considers both spatial and bidirectional temporal correlations among the sensor data. Specifically, bidirectional recurrent neural network (BRNN) [19], which is an ANN architecture designed to learn about the temporal behavior in both the positive time direction (i.e., past to present) and the negative time direction (i.e., future to present), is employed. Although BRNN requires a fixed endpoint in both time directions and, thus, is not suitable for online learning, BRNN is known to be useful for prediction problems that involve dynamic data with a fixed length [20]. Using numerical simulations, our preliminary work has demonstrated the effectiveness of BRNN for sensor data reconstruction and anomaly detection [21]. This paper extends our prior study by applying the BRNN-based sensor data reconstruction method on a bridge structure monitored with physical instrumentation of sensors.

This paper is organized as follows: Section 2 describes the BRNN-based method for sensor data reconstruction. In Section 3, the BRNN-base sensor data reconstruction method is demonstrated with vibration data collected from the Telegraph Road Bridge located along the I-275 Corridor in Michigan. The paper concludes with a brief summary and discussions in Section 4.

2. BRNN-based sensor data reconstruction method

This section describes the data-driven BRNN-based method for sensor data reconstruction. The basic assumption is that the system contains redundant information in the sensor network and that spatial and temporal correlations exist among the data collected from the sensors. The time series data of the (output) sensor of interest can be estimated using the time series data collected from other (input) sensors. Fig. 1 depicts the overall framework of the sensor data reconstruction process which consists of two main phases: (1) the training phase, and (2) the reconstruction phase. In the training phase, the time series sensor data collected from a "normal" state, where the monitored system is operating normally and the input and output sensors of interests work properly, is employed as the training dataset. The training data is preprocessed (i.e., normalized and scaled) and the normalization and scaling factors are stored for later use in the sensor data reconstruction phase. BRNN models are then trained using the preprocessed data. In the reconstruction phase, the data collected from the same set of input and output sensors is used as the testing dataset. The testing dataset is preprocessed using the normalization and scaling factors used in the training phase. The trained models are then applied to estimate time series data for the sensors of interest.

2.1. Data preprocessing and normalization

Let's consider an input-output time series sensor dataset $\mathbf{D} = (\mathbf{X}; \boldsymbol{y})$ of length T where $\mathbf{X}$ and $\boldsymbol{y}$ represent the data collected from N input sensors and a single output sensor, respectively. The input and output data are arranged as:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & \cdots & x_1^n & \cdots & x_1^N \\ \vdots & \ddots & \vdots & & \vdots \\ x_t^1 & \cdots & x_t^n & \cdots & x_t^N \\ \vdots & & \vdots & \ddots & \vdots \\ x_T^1 & \cdots & x_T^n & \cdots & x_T^N \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_t \\ \vdots \\ y_T \end{bmatrix}$$

where x_t^n and y_t denote the measurements at time t for an input sensor n and the output sensor, respectively. We further define $\boldsymbol{x}_t = (x_t^1, \cdots, x_t^n, \cdots, x_t^N)$, i.e., the t^{th} row of $\mathbf{X}$, to represent the measurements from all N input sensors at time t and $\boldsymbol{x}^n = (x_1^n, \cdots, x_t^n, \cdots, x_T^n)^T$, i.e., the n^{th} column of $\mathbf{X}$, to denote the time series measurements of length T for the n^{th} input sensor.

To make the data in approximately the same order of magnitude, the input and output time series data of length T are normalized to have zero mean and scaled within the range of $\langle -1, 1 \rangle$ as [22,23]:

$$x_t^n \leftarrow \frac{x_t^n - \bar{x}^n}{\max_t(|x_t^n|)} \quad \text{and} \quad y_t \leftarrow \frac{y_t - \bar{y}}{\max_t(|y_t|)} \tag{2}$$

where $\bar{x}^n$ and $\max_t(|x_t^n|)$ are, respectively, the mean and the maximum amplitude of the input data $\boldsymbol{x}^n$ for the input sensor n, and $\bar{y}$ and $\max_t(|y_t|)$ are, respectively, the mean and the maximum amplitude of the output data $\boldsymbol{y}$ for the output sensor. The mean and the absolute maximum values for each sensor are recorded for later use in the data reconstruction phase. Given the preprocessed time series dataset $\mathbf{D} = (\mathbf{X}; \boldsymbol{y})$, the sensor data reconstruction problem can be posed as finding the nonlinear regression relationships between the input data $\mathbf{X}$ and the output data $\boldsymbol{y}$. Once the supervised regression model is obtained based on the input and output data $\mathbf{D} = (\mathbf{X}; \boldsymbol{y})$, the trained model can be used to predict the data $\hat{\boldsymbol{y}}$ of the output sensor from the data collected from the input sensors.

2.2. Architecture of the BRNN model

The network architecture of a bidirectional recurrent neural network (BRNN) model with a single hidden layer is depicted as shown in Fig. 2. Here, this study follows the notations defined in [24] using arrows to describe the forward and backward direction. Fig. 2(a) shows the basic computational flow of the BRNN model. Note that the computations involve the information from both positive and negative time directions across all time steps $t = 1, \cdots, T$. As shown in Fig. 2(b), for each time step t, the BRNN model consists of three basic layers:

- An input layer with N input units containing the input data vector $\boldsymbol{x}_t$
- An output layer with a single output unit containing the predicted output $\hat{y}_t$
- A hidden layer consisting of M forward hidden units with values represented by $\overrightarrow{\boldsymbol{h}}_t = \{\overrightarrow{h}_t(1), \cdots, \overrightarrow{h}_t(M)\}$ and M backward hidden units with values represented by $\overleftarrow{\boldsymbol{h}}_t = \{\overleftarrow{h}_t(1), \cdots, \overleftarrow{h}_t(M)\}$

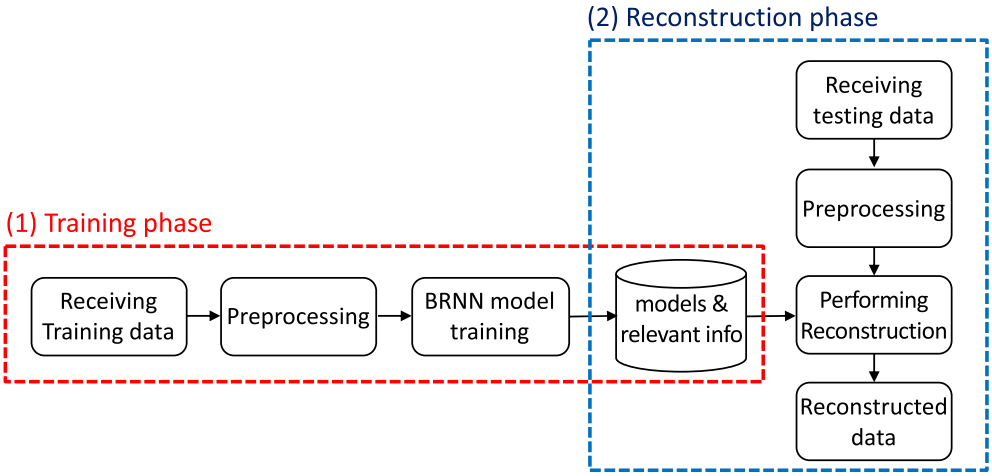


Fig. 1. Overview of sensor data reconstruction.

Here, the number of hidden units M needs to be chosen carefully. Too few hidden units would lead to an underfitting problem that the trained BRNN model is not able to adequately capture the relation among the data. On the other hand, too many hidden units would cause overfitting where the trained BRNN model may fit well for the training dataset but the model fails to generalize with other datasets.

One distinguished feature of BRNN is the cyclic computational flows among the computational units. The output of a computational unit at a time step becomes a part of the inputs for the units at a different time step. To consider both the past data and the future data when reconstructing the present data, as shown in Fig. 2(a), the BRNN model has connections in the positive time direction for the forward hidden units, as well as connections in the negative time direction for the backward hidden units. For each time step t, the computational units are interconnected as follows:

- Each of the M forward hidden units containing $\overrightarrow{\boldsymbol{h}}_t$ is connected to each of the input units containing $\boldsymbol{x}_t$ as well as the M forward hidden units containing $\overrightarrow{\boldsymbol{h}}_{t-1}$ of the previous time step $t-1$.
- Each of the M backward hidden units containing $\overleftarrow{\boldsymbol{h}}_t$ is connected to each of the input units containing $\boldsymbol{x}_t$ as well as the M backward hidden units containing $\overleftarrow{\boldsymbol{h}}_{t+1}$ of the next time step $t+1$.
- An output unit for $\hat{y}_t$ is connected to all the M forward and the M backward hidden units containing $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$, respectively.

In summary, as shown in Fig. 2(c), the BRNN is designed as a fully connected network within each time step and between the hidden layers of the adjacent time steps.

In Fig. 2(a), the variables, $\overrightarrow{\mathbf{U}}$, $\overleftarrow{\mathbf{U}}$, $\overrightarrow{\mathbf{W}}$, $\overleftarrow{\mathbf{W}}$, $\overrightarrow{\boldsymbol{v}}$ and $\overleftarrow{\boldsymbol{v}}$, denote the model weights connecting the units between the input to the hidden layers, the hidden to the hidden layers, and the hidden to the output layers. Using the input data $\boldsymbol{x}_t$ and the values $\overrightarrow{\boldsymbol{h}}_{t-1}$ from the previous time step, the values $\overrightarrow{\boldsymbol{h}}_t$ of the forward hidden unit are computed via an activation function $f(\cdot)$ as:

$$\overrightarrow{\boldsymbol{h}}_t = f(\overrightarrow{\mathbf{U}}\boldsymbol{x}_t + \overrightarrow{\mathbf{W}}\overrightarrow{\boldsymbol{h}}_{t-1} + \overrightarrow{\boldsymbol{b}}) \tag{3}$$

where $\overrightarrow{\boldsymbol{b}}$ is termed the bias for the forward hidden units. Similarly, the values $\overleftarrow{\boldsymbol{h}}_t$ of the backward hidden units are computed using the input $\boldsymbol{x}_t$ and the values $\overleftarrow{\boldsymbol{h}}_{t+1}$ from the next time step via the same activation function $f(\cdot)$ as:

$$\overleftarrow{\boldsymbol{h}}_t = f(\overleftarrow{\mathbf{U}}\boldsymbol{x}_t + \overleftarrow{\mathbf{W}}\overleftarrow{\boldsymbol{h}}_{t+1} + \overleftarrow{\boldsymbol{b}}) \tag{4}$$

where $\overleftarrow{\boldsymbol{b}}$ is the bias for the backward hidden units. In this study, the forward and backward hidden units employ the hyperbolic tangent function as the activation function to model the nonlinear relationships among the data:

$$f(z) = \tanh(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})} \tag{5}$$

Finally, the predicted output is computed using a linear function as:

$$\hat{y}_t = \overrightarrow{\boldsymbol{v}} \cdot \overrightarrow{\boldsymbol{h}}_t + \overleftarrow{\boldsymbol{v}} \cdot \overleftarrow{\boldsymbol{h}}_t + c \tag{6}$$

where c is the bias for the output unit. The weights and the biases represent the model parameters of the BRNN to be "trained" using the input and output data sets.

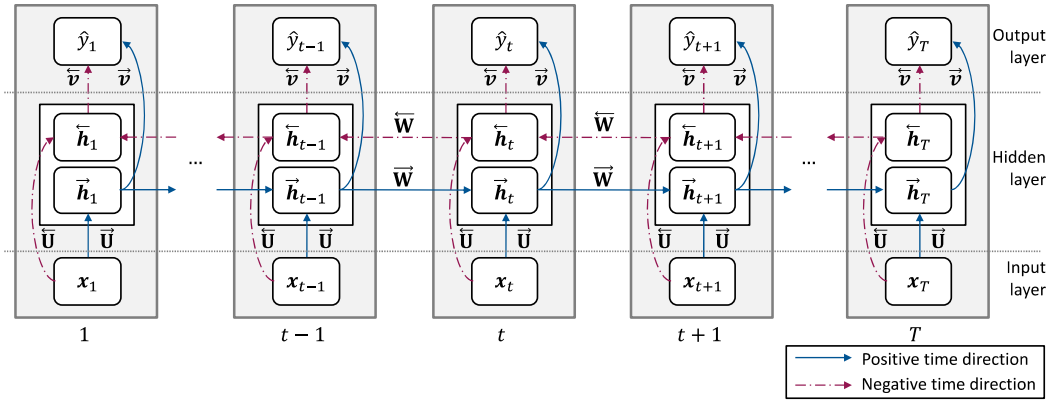
2.3. Training the BRNN model

The BRNN model is trained by iteratively adjusting its model parameters to minimize the loss function L that measures the difference between the output data $\boldsymbol{y} = (y_1, \cdots, y_t, \cdots, y_T)^T$ and the reconstructed output data $\hat{\boldsymbol{y}} = (\hat{y}_1, \cdots, \hat{y}_t, \cdots, \hat{y}_T)^T$. For initialization, the biases $\overrightarrow{\boldsymbol{b}}$, $\overleftarrow{\boldsymbol{b}}$ and c are set to zero and the elements for the weights $\overrightarrow{\mathbf{U}}$, $\overleftarrow{\mathbf{U}}$, $\overrightarrow{\mathbf{W}}$, $\overleftarrow{\mathbf{W}}$, $\overrightarrow{\boldsymbol{v}}$ and $\overleftarrow{\boldsymbol{v}}$ of the BRNN model are assigned randomly using a uniform distribution between $-1/\sqrt{l}$ and $1/\sqrt{l}$, where l is the size of a prior layer (where, in this case, $l = N$, the number of input sensors, for $\overrightarrow{\mathbf{U}}$ and $\overleftarrow{\mathbf{U}}$ and $l = M$, the number of hidden units, for $\overrightarrow{\mathbf{W}}$, $\overleftarrow{\mathbf{W}}$, $\overrightarrow{\boldsymbol{v}}$ and $\overleftarrow{\boldsymbol{v}}$) [25].

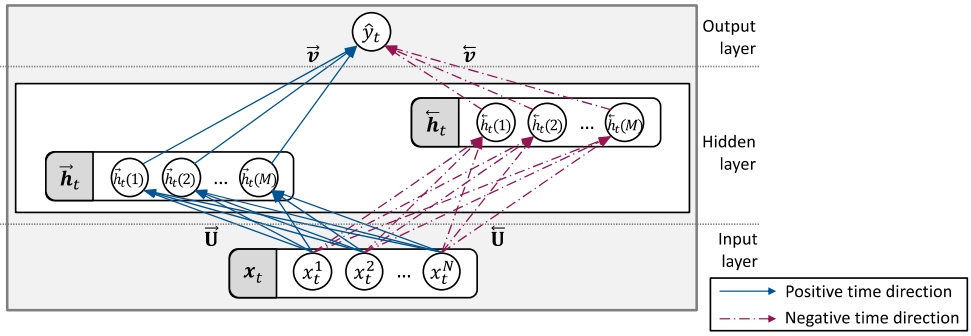
Once the model parameters are initialized, the backpropagation through time (BPTT) algorithm is employed for training the BRNN model [19,26]. For each iteration, the BPTT algorithm consists of three steps: forward pass, backward pass and parameter updating. In each iteration, $\overrightarrow{\boldsymbol{h}}_0$ of the forward hidden units and $\overleftarrow{\boldsymbol{h}}_{T+1}$ of the backward hidden units are initialized to be zero. In the forward pass, the predicted output $\hat{y}_t$, $t = 1, \cdots, T$, is calculated using the input data $\mathbf{X}$ via Eqs. (3), (4) and (6). In the backward pass, the loss function L is computed. In this study, the loss function L is computed using the mean-squared-error (MSE) as:

$$L = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{7}$$

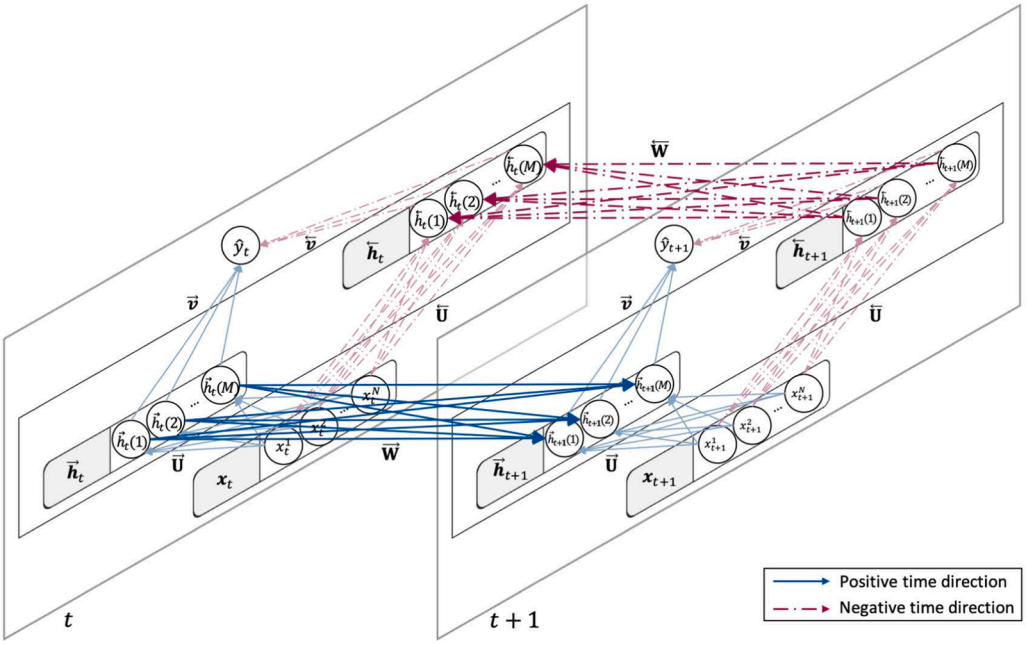
For each time step t, the gradient of the loss function L with respect to the predicted output $\hat{y}_t$ is computed and propagated forward and backward through time via the network to calculate the gradients of the loss function L with respect to the model parameters. The loss gradients for each model parameter are summed over all T time steps for parameter updating. The gradients of the loss function L computed with respect to the parameters are used to adjust the corresponding parameters using an optimization procedure, such as Adaptive Moment



(a) Overview of BRNN structure



(b) Organization of layers at a time step t



(c) Expanded view of interconnection between time steps t and $t + 1$

Fig. 2. Structure of bidirectional recurrent neural network with a single hidden layer and the computation flow of a forward pass.

Estimation (Adam) [27]. Additionally, learning rate, which limits the amount of adjustments per each iteration, is defined. The learning rate can affect the number of iterations in the training process and the accuracy of the BRNN model. A higher learning rate can speed up the training, while a lower learning rate likely improves the accuracy of the BRNN model. The three steps of forward pass, backward pass and parameter updating iterate until either the loss L converges or the number of epochs (i.e., the number of times that the entire datasets are being processed) reaches a prescribed limit. The maximum number of epochs can affect the accuracy of the BRNN model. While the training error often decreases as the number of epochs increases, too many epochs may cause overtraining such that the accuracy of the BRNN model for other datasets decreases. It should be noted that in the BRNN model, the weights $\overrightarrow{\mathbf{U}}$, $\overleftarrow{\mathbf{U}}$, $\overrightarrow{\boldsymbol{v}}$ and $\overleftarrow{\boldsymbol{v}}$ and the biases $\overrightarrow{\boldsymbol{b}}$, $\overleftarrow{\boldsymbol{b}}$ and c are the same for every time step, and the weights $\overrightarrow{\mathbf{W}}$ and $\overleftarrow{\mathbf{W}}$ are the same for every forward and backward time step. Details on the forward pass,

backward pass and parameter update for BRNN can be found in [26,28].

One issue of training the BRNN is the significant demand on the computational and memory requirements due to the chains of connections linking the computational units. For unidirectional RNN, truncated BPTT (TBPTT) methods are often used to reduce the computational cost and memory usage by limiting the number of time steps for which the loss is propagated [29,30]. However, implementing the TBPTT strategy for BRNN is not practical because the hidden units along either the positive or the negative time direction need to be processed over all time steps in order to compute the estimated outputs. Instead, this work adopts a batch learning approach [30]. As summarized in Algorithm 1, the approach defines a batch size B which is the maximum number of consecutive data points to be used in an iteration (corresponding to one cycle of forward pass, backward pass and parameter updating for that batch). That is, the training dataset $\mathbf{D} = (\mathbf{X}; \boldsymbol{y})$ with T consecutive data points is divided into $\lfloor T/B \rfloor$ batches so that the batches $\mathbf{D}_1, \ldots, \mathbf{D}_{\lfloor T/B \rfloor - 1}$ each contains B consecutive data points whereas $\mathbf{D}_{\lfloor T/B \rfloor}$ contains the remaining $T - (\lfloor T/B \rfloor - 1) \times B$ consecutive data points (see step 3 of Algorithm 1). It should be noted that if the last batch $\mathbf{D}_{\lfloor T/B \rfloor}$ is too small, it can be simply discarded. As shown from steps 5 to 9 in Algorithm 1, each batch is trained independently. The training continues until either the maximum number of epochs R is reached or the loss change ΔL is lower than a predefined threshold ΔL_{thres}, as shown in step 4. With this approach, BRNN can be trained with a large amount of training data without causing excessive memory usage by limiting the number of times that the loss is propagated.

Algorithm 1.. Training of BRNN model

Input:
- $\mathbf{D} = (\mathbf{X}; \boldsymbol{y})$: preprocessed training dataset
- T: number of training data points
- B: batch size
- R: number of maximum epochs
- ΔL_{thres}: minimum value of loss change
- η: learning rate

1: Initialize the BRNN model's weights and biases
2: Initialize epoch $r \leftarrow 0$, loss in the previous step $L_{\text{prev}} \leftarrow \infty$, and change of loss $\Delta L \leftarrow \infty$
3: Divide $\mathbf{D}$ into $\lfloor T/B \rfloor$ batches: $\mathbf{D} = \{\mathbf{D}, \ldots, \mathbf{D}_{\lfloor T/B \rfloor}\}$
4: **while** $(r < R)$ **or** $(\Delta L > \Delta L_{\text{thres}})$ **do**
5: Set $L_{\text{curr}} \leftarrow 0$
6: **for each** $\mathbf{D}_i = (\mathbf{X}_i; \boldsymbol{y}_i)$, $i = 1, \ldots, \lfloor T/B \rfloor$ **do**
7: Forward pass: compute $\hat{\boldsymbol{y}}_i$ using input $\mathbf{X}_i$ according to Eqs. (3), (4) and (6)
8: Backward pass: (a) compute the loss L between output $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ using the MSE function in Eq. (7), and (b) compute the gradients of the loss with respect to the BRNN parameters
9: Parameter updating: update the weights and biases based on the gradients of the loss with respect to the parameters and the learning rate η using Adam [27]
10: $L_{\text{curr}} \leftarrow L_{\text{curr}} + L$
11: **end for**
12: Update $r \leftarrow (r + 1)$; $L_{\text{curr}} \leftarrow \frac{L_{\text{curr}}}{\lfloor T/B \rfloor}$; $\Delta L \leftarrow |L_{\text{prev}} - L_{\text{curr}}|$; $L_{\text{prev}} \leftarrow L_{\text{curr}}$
13: **end while**

2.4. Sensor data reconstruction

Let's consider a test dataset $\mathbf{D}' = (\mathbf{X}'; \boldsymbol{y}')$ collected from the same N input sensors and the same output sensor as in the trained model with T' consecutive data points per each sensor. The length of the data points (i.e., the sensor data) should span over a certain time period sufficient to establish the temporal correlations among the sensor data. The sufficient period may preclude true real-time data reconstruction, but the actual duration is relatively short. As shown in the case study example described later, 100 consecutive data points, which can be collected in 0.5 s at a sampling rate of 200 Hz, are normally sufficient.

The sensor data reconstruction is performed as follows. Using the normalization and scaling factors recorded in the training phase, the data $\mathbf{D}'$ is first normalized and scaled using Eq. (2). The trained BRNN model then takes the input data $\mathbf{X}'$ to compute the prediction $\hat{\boldsymbol{y}}'$ for the output sensor's data according to Eqs. (3), (4) and (6). Here, the values for $\overrightarrow{\boldsymbol{h}}_0$ of the forward hidden units and $\overleftarrow{\boldsymbol{h}}_{T'+1}$ of the backward hidden units are set to zeros. It should be noted that the reconstructed data for the first few and the last few (e.g., 10) time steps can be erroneous due to the effect of the zero initialization. The accuracy of the reconstructed data can be evaluated by measuring the discrepancy between the measured output $\boldsymbol{y}'$ and the predicted output $\hat{\boldsymbol{y}}'$, for example, using the root-mean-square-error (RMSE) as:

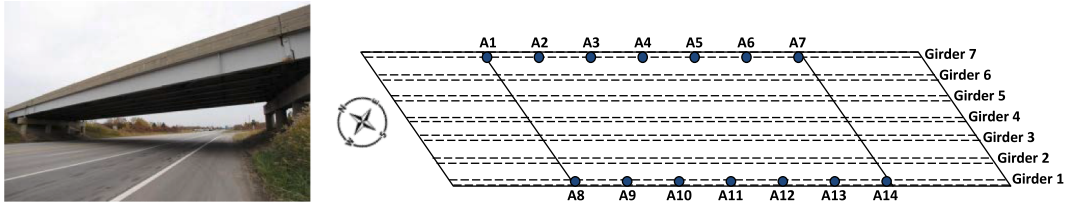
$$\epsilon = \sqrt{\frac{\sum_{t=1}^{T'} (\hat{y}_t' - y_t')^2}{T'}} \tag{8}$$

The trained BRNN model can have at least two different uses. First, if the output sensor is known to be faulty, the faulty measurement data can be replaced by the reconstructed data. The reconstructed data can be further utilized to estimate the health condition (e.g., operational frequency and mode shapes) of the monitored system even when some of the sensors are faulty. Second, if the output data possesses the possibility of anomalies, the testing error ϵ computed by Eq. (8) can be used for validating the output sensor by using, for example, the analytical redundancy approach [6]. This study focuses on reconstructing the data of faulty sensors as discussed next in Section 3.

3. Evaluation of the method on a monitored bridge structure

In the case study, the BRNN-based sensor data reconstruction method is demonstrated using the sensor data collected on the Telegraph Road Bridge located in Monroe, Michigan. The Telegraph Road Bridge, as shown in Fig. 3(a), is an overpass bridge situated along the I-275 corridor in Michigan. The bridge has been instrumented with a sensor network for structural health monitoring research purposes since early 2011 by the research team at the University of Michigan [31,32]. As shown in Fig. 3(b), vertical vibration responses are collected at 14 sensor locations on the two exterior girders. The accelerometers measure the vertical vibration responses for one-minute duration every 2 h at 200 Hz sampling rate.

This section discusses the training and the application of the BRNN method for sensor data reconstruction. The vibration response datasets (see Fig. 4) collected during the month of July 2014, which includes



(a) Side view (b) Accelerometer layout

Fig. 3. Telegraph Road Bridge (Monroe, Michigan).

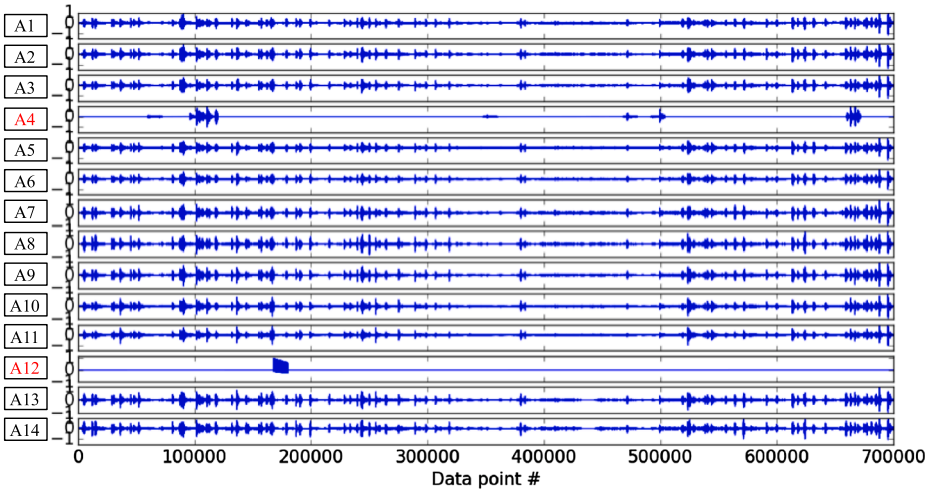


Fig. 4. Vertical acceleration measurement of the Telegraph Road Bridge (July 2014).

700,000 data points per sensor are employed in the study. It should be noted that Sensors A4 and A12 are down most of the time on that month. As to be discussed in Section 3.1–3.4, to evaluate the BRNN-based method, the first 400,000 data points of the functioning sensors (i.e., all sensors except for A4 and A12) are employed. Section 3.5 then discusses using all of 700,000 data points of the functioning sensors as the input data to recover the missing data for Sensors A4 and A12.

This section is organized as follows. In Section 3.1, the effects of batch size, training dataset size and testing dataset size to the BRNN-based sensor data reconstruction method are investigated. The results are then used to select the batch size, the training dataset size and the testing dataset size in the subsequent sections. In Section 3.2, the BRNN-based method is compared with other existing sensor data reconstruction methods to assess the effectiveness of the BRNN-based method. Section 3.3 examines the effect of using a subset of sensors as input sensors. Section 3.4 discusses the robustness of the BRNN-based method against the environmental effects (e.g., temperature change) using datasets collected from the different months over a one-year period. Finally, Section 3.5 discusses the recovery of the data of the faulty sensors (i.e., sensors A4 and A12) for the month of July 2014 by utilizing the intact datasets collected during different time periods. For the training and testing of the BRNN models, this study uses an open source machine learning framework, namely PyTorch [33]. In addition, a desktop computer (Intel Core i7-7820X 3.60 GHz; 32 GB RAM; NVIDIA GTX 1080 Ti) is used for the evaluation.

3.1. Effects on data size of training and testing

This section describes the training of BRNN models for sensor data reconstruction and discusses the effect of batch size, training dataset size and testing dataset size on the data reconstruction accuracy and computational times. Data collected from the twelve functioning sensors (i.e., Sensors A1 to A14 but without Sensors A4 and A12) are considered to assess the effects of nearby sensors. Two BRNN models for two different output sensors are considered for the testing purpose. The first model has Sensor A1 located near the support as the output sensor, while all other sensors are the input sensors. The second model has Sensor A11 located at the middle of a girder as the output sensor, while all other sensors are considered as the input sensors. The hyperparameters for training the BRNN model are selected heuristically as follows:

- Number of forward hidden units: 50
- Number of backward hidden units: 50
- Maximum number of epochs: 200
- Learning rate: 10^{-3}

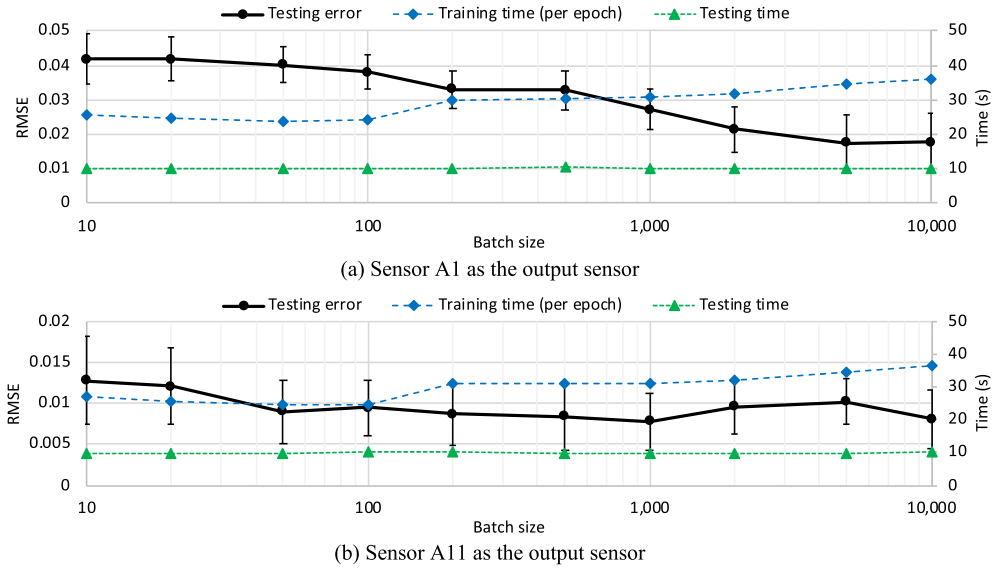
3.1.1. Effects of batch size

To study the effect of the batch size, the experiments use the first 100,000 data points as the training dataset. The trained model is then tested with five datasets, each has 40,000 consecutive data points (per sensor) selected from the data points 200,001 to 400,000. As discussed in Section 2.3, a batch learning approach [30] is adopted to decrease the computational and memory demands for training the BRNN models. For implementation, it is important to choose an appropriate batch size. If the batch size is too small, sensor data reconstruction would be inaccurate due to the effects from the initial hidden units with zero values and the temporal correlations. However, if the batch size is too large, model training would require significant memory usage (that does not necessarily lead to better accuracy).

Two BRNN models, one with A1 and the other with A11 as the output sensors, are trained with different batch sizes ranging from 10 to 10,000. Fig. 5 shows the mean and standard deviation of the testing errors (i.e., RMSE between the actual and reconstructed values) from the five testing datasets, the training time per epoch and the reconstruction time with respect to the different batch sizes for Sensors A1 and A11. The testing error for Sensor A1 is relatively low when the batch size is equal to or higher than 5000, while the testing error for Sensor A11 is relatively low when the batch size is equal to or higher than 50. In addition, the training times increase as the batch size increases. It can also be seen that the computing time for testing is almost the same for all batch sizes. Based on the results, the batch size of 5000 is selected and used in the study of other effects to achieve a good balance of the training time and the data reconstruction accuracy.

3.1.2. Effects of training data sizes

Next, we examine the effect of the training dataset size on data reconstruction. The two BRNN models, one with A1 and the other with A11 as the output sensors, are trained with datasets of different sizes selected from the first 10,000 to the first 140,000 data points, while the batch size is kept at 5000 data points. For each training dataset, the normalization and scaling factors obtained from the largest dataset with 140,000 data points are used so that the testing errors can be compared. Similar to the previous example, five testing cases each has 40,000 consecutive data points per sensor (selected from the data points 200,001 to 400,000) are used. Fig. 6 shows the mean and standard deviation of the testing errors, the training time per epoch and the testing time with respect to the different training data size. As can be seen in the figure, the training time increases linearly as the training data size increases, while the testing times are very much the same



(a) Sensor A1 as the output sensor

(b) Sensor A11 as the output sensor

Fig. 5. Testing error, training time and reconstruction time with respect to different batch sizes.

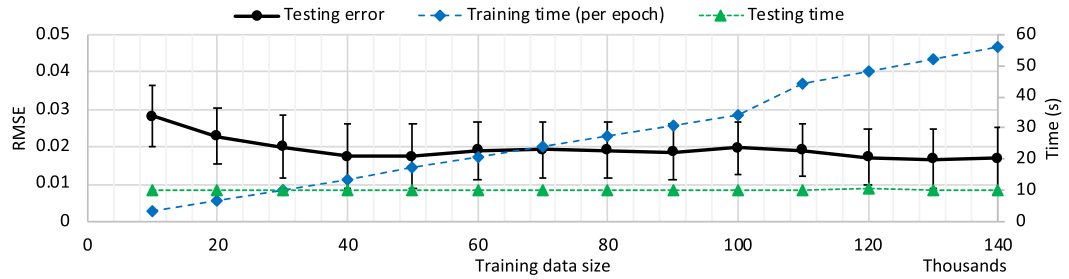
regardless of the training data size. For both BRNN models, the errors decrease until the training sizes reach about 40,000 data points; beyond that, little improvements (sometimes worsen) are observed. Based on the results, the training dataset size of about 40,000 data points would achieve a good accuracy for sensor data reconstruction.

3.1.3. Effect of testing data size

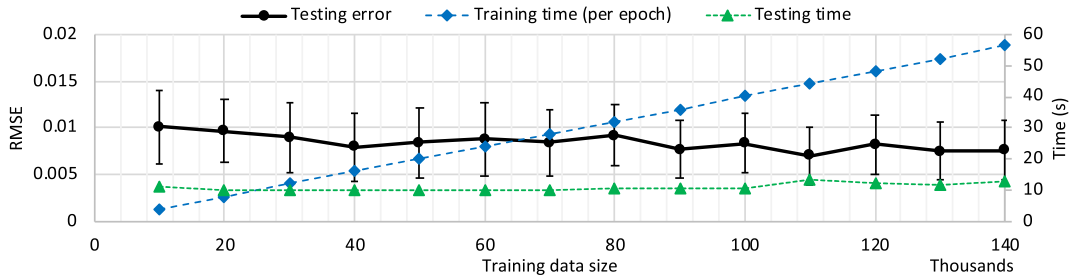
The last example examines the effect of the size of the testing dataset on data reconstruction. Since the initial hidden units (i.e., $\overrightarrow{\boldsymbol{h}}_0$ and $\overleftarrow{\boldsymbol{h}}_{T'+1}$) of BRNN models are set to zero, the testing error could be high when the testing dataset does not contain sufficient data points. The sensor data reconstructions are performed using the two BRNN models constructed with a training dataset containing 100,000 data points (from the data points 1 to 100,000) and a batch size of 5000. Again, five testing cases each of which has 40,000 consecutive data points per sensor (selected from the data points 200,001 to 400,000) are employed. To see the effects due to the initial hidden units, the sizes of the testing datasets range from 10 (that produces 4000 testing datasets per each test case) to 40,000 (i.e., one testing dataset for the test case). Fig. 7 shows the mean and standard deviation of the testing errors with respect to the sizes of the testing datasets. The results show that the errors are relatively consistent when the testing dataset sizes reach about 100 data points for both BRNN models with Sensor A1 and Sensor A11 as the output sensor. Based on the results, the testing dataset size of 100 data points or larger would be sufficient to mitigate the effect of initial hidden units with zero values.

3.2. Comparison of other sensor data reconstruction methods with BRNN

As discussed in the introduction section, other data-driven sensor data reconstruction methods, such as principal component analysis (PCA) [8], minimum mean square error (MMSE) estimation [9], support vector regression (SVR) [14], feedforward neural network (FNN) [12] and recurrent neural network (RNN) [18], have been proposed. In this section, the BRNN-based sensor data reconstruction method is

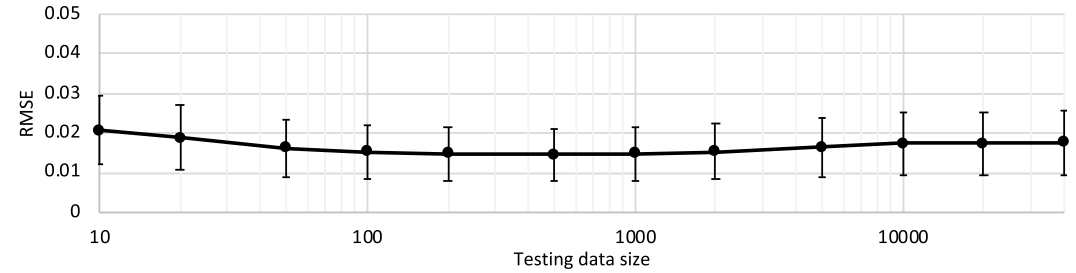


(a) Sensor A1 as the output sensor

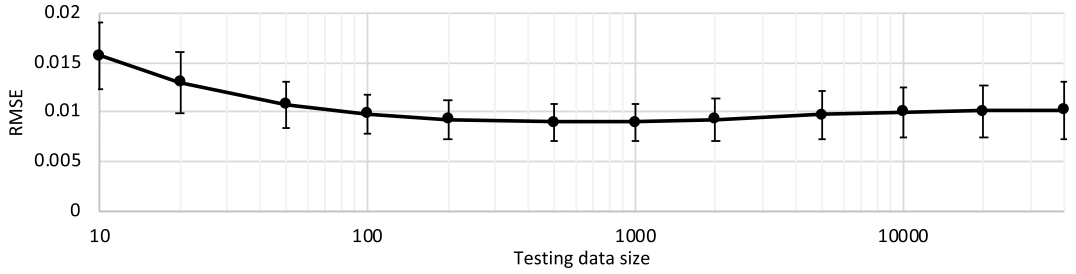


(b) Sensor A11 as the output sensor

Fig. 6. Testing error, training time and reconstruction time with respect to different training data sizes.



(a) Sensor A1 as the output sensor



(b) Sensor A11 as the output sensor

Fig. 7. Testing error with respect to different testing data sizes.

Table 1
Hyperparameters of the ANN models.

	FNN and RNN	BRNN
Number of hidden layers	1	1
Number of hidden units	100	100 (50 forward and 50 backward hidden units)
Activation function for hidden layer	Hyperbolic tangent function	Hyperbolic tangent function
Activation function for output layer	Linear function	Linear function
Optimization algorithm	Adam	Adam
Loss function	MSE	MSE
Learning rate	10^{-3}	10^{-3}
Maximum number of epochs	200	200
Batch size	5000	5000

evaluated by comparing with the existing methods. Two BRNN models for reconstructing the data of Sensor A1 and Sensor A11 are created for the testing purposes. For comparison, PCA, MMSE, SVR, FNN and RNN models corresponding to the two BRNN models are also created. As shown in Table 1, the neural network models, namely FNN and RNN models are created and trained using the same hyperparameters as for the BRNN model.

For evaluation, the data collected from the twelve operating sensors (i.e., all sensors but Sensors A4 and A12) are used. The PCA model is fitted with 100,000 data points (from the data points 1 to 100,000) for each sensor by projecting the data onto a lower dimensional subspace (i.e., principal subspace) which maximizes the variance of the projected data [8,34]. Here, an M-dimensional principal subspace is the M eigenvectors corresponding to the M largest eigenvalues of the data covariance matrix [34]. The MMSE estimator, which is a linear function that minimizes the MSE between the measurement data and the estimated data, is obtained based on the same 100,000 data points for each sensor [9]. The SVR model is fitted with the same 100,000 data points for each sensor by finding the fittest function that has at most ε deviation (i.e., a predefined deviation limit) from the measurement data. The three ANN models are also trained using the same 100,000 data points for each sensor. For evaluation, the testing dataset, which has 40,000 data points per sensor, is selected from the data points 100,001 to 140,000. Fig. 8 shows the training and the testing errors for the ANN models, including FNN, RNN and BRNN models, with respect to the training epoch. Once the ANN models are trained sufficiently, the testing error decreases as the training error decreases.

Fig. 9 shows the samples of sensor data reconstruction results using all five methods tested. It should be noted that Fig. 9 shows the measurement data collected over a period of one second (i.e., 200 data points) out of the entire reconstructed 40,000 data points so that the plots can clearly show the comparison between the original data and the reconstructed data. As shown in Fig. 9(a), the BRNN model reconstructs the data for Sensor A1, which locates near the support of the bridge, most accurately among the tested methods. For Sensor A11, which locates at the center of the bridge, all methods work well, but, as shown in Fig. 9(b), the BRNN- and RNN-based methods show slightly better fit comparing to the other three methods. Table 2 provides further information on the reconstruction results, including the computation time and the testing errors. One notable result is that the testing error for the sensor at the center of the bridge (i.e., Sensor A11) is generally lower than the sensor located near the support of the bridge (i.e., Sensor A1). This is because the sensor located near the support has only one adjacent sensor, which means that there exists less spatial information for estimating the sensor data. Nevertheless, the BRNN models consistently yield smaller testing error than all other methods, because BRNN is able to take both spatial and bidirectional temporal correlation into account. One tradeoff for the BRNN models is the amount of training time required. Once BRNN model is trained, reconstruction process can be executed very efficiently even though the computing time remains higher than the other methods.

3.3. Selection of input sensors

So far, the sensor data reconstruction is demonstrated using all available sensors as input sensors, which assumes that all sensors can contribute to the reconstruction of output sensor's data. However, in reality, only a subset of sensors is contributing to the data reconstruction process. It would be more practical to employ the subset of sensors that pose most contributions since this would reduce the size of the BRNN model, particularly, when there are many sensors in the sensor network. Furthermore, by using a subset of sensors as input sensors, sensor data reconstruction model can still be used even when some of

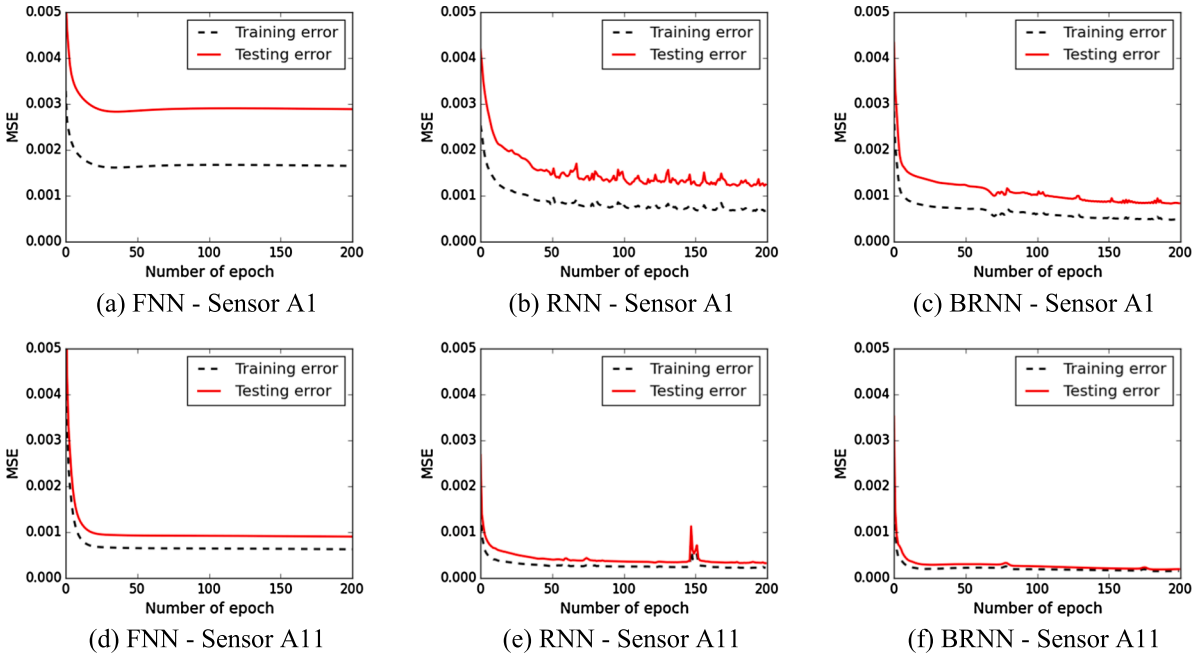


Fig. 8. Training error and testing error with respect to the training epoch.

other sensors in the network are faulty. Since the BRNN-based sensor data reconstruction method is based on the spatiotemporal correlations, one plausible approach is to choose the sensors that have high spatial correlations with the targeted output sensor as the input sensors.

Fig. 10 shows the covariance matrix among the data from the twelve sensors in the network. Here, the covariance matrix is computed using 100,000 data points (from the data points 1 to 100,000) for each sensor. From the covariance matrix, it can be seen that the data from sensors along the same girder are highly correlated, while the data from the sensors on different girders are not correlated well. For example, Sensor A1 is highly correlated with Sensors A2 to A7 which are located on the same girder but the sensor shows very little correlation with Sensors A8 to A14 which are located on a different girder. The matrix also shows that the data from adjacent sensors (e.g., Sensors A1 and A2) on the same girder are more correlated than the data from distant sensors (e.g., Sensors A1 and A7) even on the same girder.

To study the effect of using only the sensors that have high covariance with the target sensor, sensor data reconstruction for sensor A1 is performed with two BRNN models. The first model has all available sensors (i.e., Sensors A2, …, A14) as inputs, while the second model uses only the highly correlated sensors (i.e., Sensors A2, …, A7) as inputs. Both models are constructed based on the hyperparameters as shown in Table 1. The two models are trained using the first 100,000 data points (from the data points 1 to 100,000) for each sensor. For testing, 40,000 data points (from the data points 100,001 to 140,000) are employed. Fig. 11 shows the reconstructed data for each model. The reconstruction errors of the two models are shown in Table 3. The results show that the model constructed using the data from the highly correlated subset of sensors is as accurate as the model constructed using the data from all the sensors.

3.4. Environmental effect

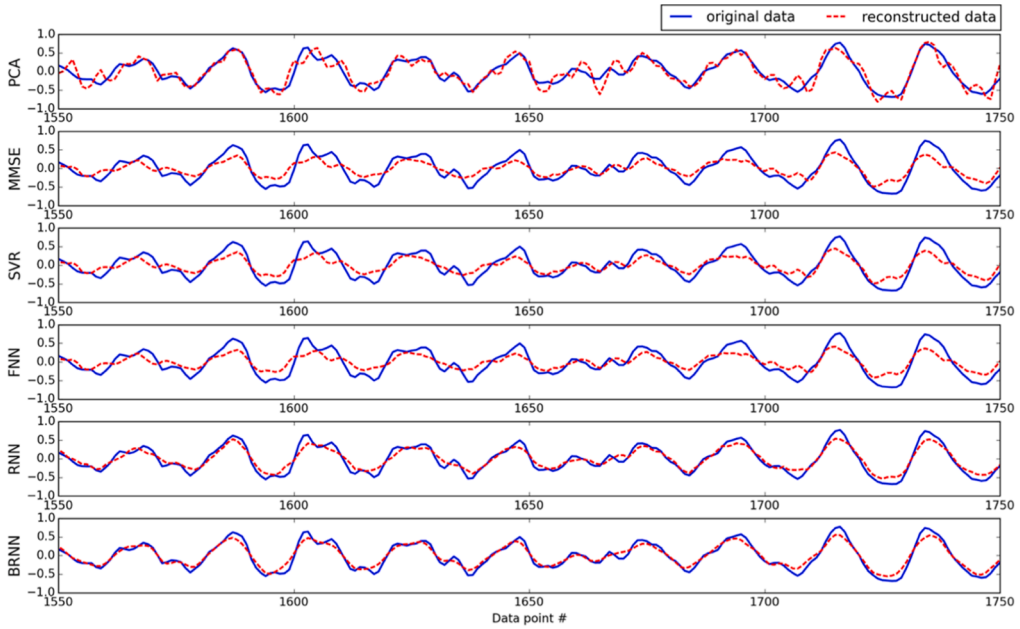
For bridges which are in service, environmental condition (e.g., temperature) varies throughout the year [35]. In this section, a study is conducted to see whether the BRNN models trained with data from a certain month can be used to reconstruct the data for other months. For this study, the data collected from April 2014 to February 2015 is used to consider the change of temperature in a year. Note that, the data from March 2014 and August 2014 are not used due to the quality of the data. For this study, a BRNN model with Sensors A8, A9 and A10 as input sensors and Sensor A11 as output sensor is constructed based on the hyperparameters defined in Table 1. Furthermore, Sensors A12, A13 and A14 located on the same girder of Sensor A11 are not used as input sensors because the sensors malfunctioned frequently during the same period.

Fig. 12 shows the temperature change measured by a thermistor on the Telegraph Road Bridge during the target period. BRNN models are created for each of the 10 selected months as highlighted in Fig. 12. Each BRNN model is trained with 100,000 data points per sensor collected from the corresponding month. In addition, a yearly model is created and trained using the training data containing 100,000 data points per sensor that concatenates five datasets collected from five different months, including April, July, October, December 2014 and February 2015. Using 11 BRNN models (i.e., 10 monthly models and a yearly model), sensor data reconstructions are performed for 10 testing datasets, each of which has 40,000 data points per sensor collected from one of the 10 selected months. Fig. 13 shows the testing errors of the 11 BRNN models for the 10 testing datasets. It can be seen that the testing errors of all 11 BRNN models are quite similar for each month. This result indicates that, for the data considered at this bridge site, the BRNN-based method appears to be robust against the change of environmental conditions.

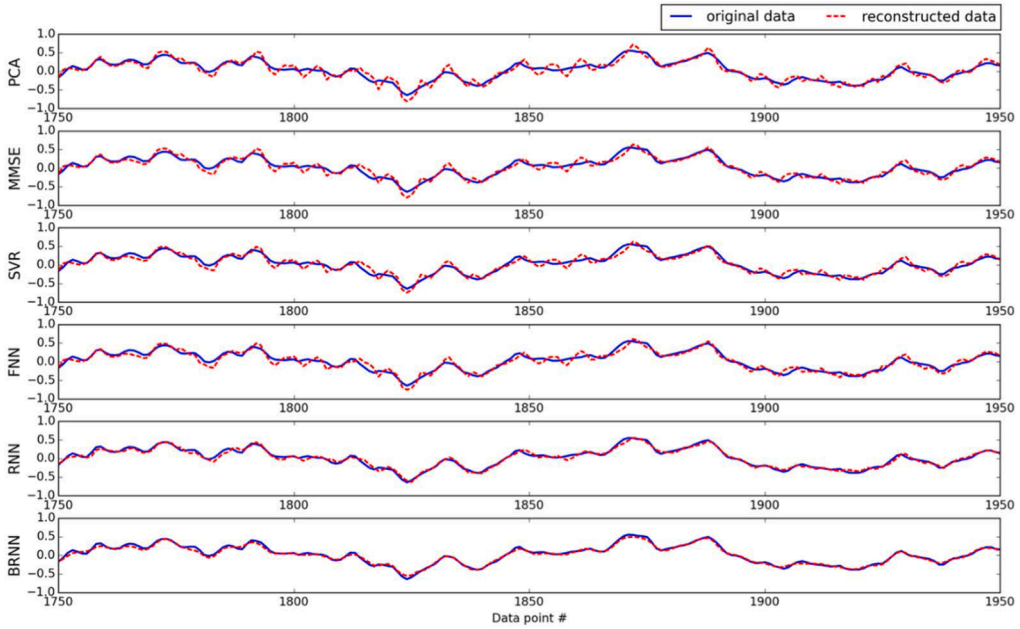
3.5. Recovery of missing data

Sensor data reconstruction can be used for recovering missing data when some sensors are faulty. As shown in Fig. 4, Sensors A4 and A12 were not working properly in the month of July 2014 and the data is clearly erroneous. The missing data can be recovered using BRNN-based sensor data reconstruction method. The basic idea is to train a BRNN model using good quality datasets collected from other dates. In this section, the data of Sensors A4 and A12 in July 2014 is recovered using the two BRNN models trained with the data of October 2014 and July 2012, respectively, when the input sensors and output sensors operated normally.

Fig. 14 depicts the sensor data reconstruction process for recovering the missing data of July 2014 by training two BRNN models corresponding to the two target sensors A4 and A12, respectively. The first BRNN model is constructed with input sensors A2, A3, A5, A6 and A7,



(a) Sensor A1



(b) Sensor A11

Fig. 9. Sensor data reconstruction with the different methods.

Table 2
Comparison of computing time and testing error of different sensor data reconstruction methods.

	PCA	MMSE	SVR	FNN	RNN	BRNN
Training time per epoch (sec/epoch)	–	–	–	0.189	23.962	34.449
Total training time (sec)	0.188	0.021	11.2	37.8	4792.4	6889.8
Testing time (sec)	1.512	0.004	3.024	0.040	4.741	8.639
Testing RMSE for Sensor A1	0.085	0.052	0.052	0.053	0.035	0.029
Testing RMSE for Sensor A11	0.030	0.028	0.029	0.030	0.018	0.014

which are located along the same girder as the output sensor A4. This first model is trained using 100,000 data points collected in October 2014 for each sensor when all of input and output sensors were operating properly. Once the BRNN model is trained, the data (i.e., the 700,000 data points per sensor) collected by Sensors A2, A3, A5, A6 and A7 in July 2014 is inputted to the model to reconstruct the data for Sensor A4. The recovered dataset of Sensor A4 is shown in Fig. 14.

Similarly, the second BRNN model is constructed with input sensors A8, A9, A10, A11, A13 and A14 and an output Sensor A12. This second BRNN model is trained using 100,000 data points per sensor, collected in July 2012 when the input and output sensors were operating properly. The data (i.e., the 700,000 data points per sensor) collected by Sensor A8, A9, A10, A11, A13 and A14 in July 2014 is then inputted to the BRNN model to recover the data for Sensor A12 in July 2014. The recovered dataset of Sensor A12 is also shown in Fig. 14.

The recovered data, which now includes the data for all 14 sensors, can be further analyzed to check the integrity of the target system. For example, the bridge's operational mode shapes can now be computed using the data collected from the functional sensors and the recovered

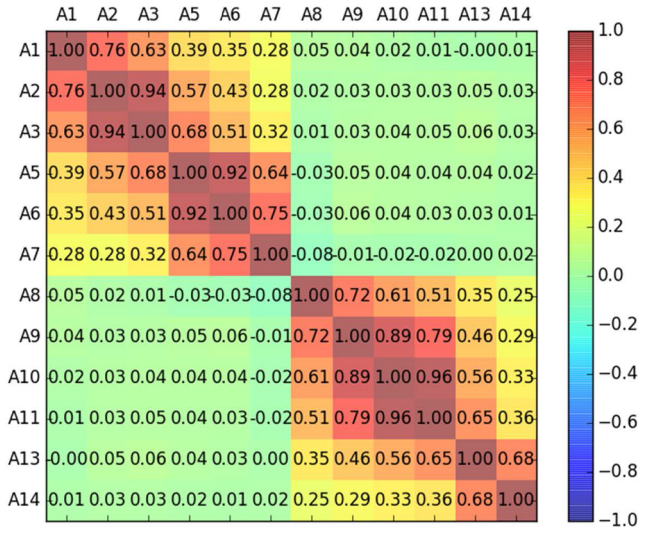


Fig. 10. Covariance matrix of sensor data collected from the Telegraph Road Bridge.

data for the two faulty sensors A4 and A12 for July 2014, for example, using the frequency domain decomposition method [36]. Fig. 15 shows the first five modal frequencies and the operational mode shapes computed with the recovered sensor data along with the corresponding modal frequencies and operational mode shapes experimentally measured and reported in [37]. As shown in Fig. 15, the modal frequencies and the operational mode shapes computed using the recovered datasets match very well with the bridge's original mode shape and frequencies obtained experimentally. It should be noted that the third mode shape computed from the reconstructed dataset looks different from the one obtained from the earlier experimental study not because the reconstructed data is incorrect but because the dataset used in this study does not include sensor data collected at the center of the bridge.

For comparison, the data recovery process is repeated using five other data reconstruction methods, including PCA, MMSE, SVR, FNN and RNN. Fig. 16 shows the singular values of the power spectral density (PSD) matrix, which is computed using the frequency domain decomposition method, from the recovered datasets [36]. As shown in the figure, all methods produce the same modal frequencies. The singular values show some differences in their values among the methods for frequencies over 33 Hz. Fig. 17 shows the operational mode shapes of Girder 7 obtained using the recovered datasets. The mode shape values at the location of the target sensor (i.e., Sensor A4) are different among the tested methods. In summary, while the data reconstructed by all the methods are sufficient to estimate the same results for the modal frequencies, the sensor data reconstructed by the BRNN method appears to be useful for estimating the singular values for high frequencies and for calculating operational mode shapes.

Table 3
Comparison of computing time and testing error of different sensor data reconstruction methods.

	Model 1	Model 2
Input sensors	All sensors (A2, …, A14)	Highly correlated sensors (A2, …, A7)
Testing RMSE for Sensor A1	0.0288	0.0292

4. Summary and discussion

This paper presents a data-driven method for accurate sensor data reconstruction by leveraging the spatiotemporal correlations among the data collected by a network of sensors. This work employs BRNN to learn about the spatiotemporal behavior of a system in both positive (from past to present) and negative (from future to present) time direction. The BRNN-based sensor data reconstruction method consists of two phases: the training phase and the reconstruction phase. In the training phase, a BRNN model is trained with a dataset containing a sequence of synchronized data for the input and the output sensors. In the reconstruction phase, the trained BRNN model takes the input sensors' data and reconstructs the data of a target output sensor.

The BRNN method for sensor data reconstruction is demonstrated and validated with vibration response data collected from the Telegraph Road Bridge in Michigan. Based on the results in the case study, the following findings are worth mentioning. The batch size for the training phase should be carefully chosen because a small batch size may lead to inaccurate data reconstruction, while a large batch size would require significant memory usage and training time. For the evaluation with the data from the Telegraph Road Bridge, the batch size of 5000 provides a good balance between data reconstruction accuracy and training time. From the tests performed, the testing errors decrease very quickly until the training data size increases to certain level (e.g., 40,000); after that level, the rate of improvement on accuracy becomes slow even with an increase of data size while the training time continues to increase in proportion to the size of the training dataset. For practical implementation, the size of the training dataset should be selected to balance the time required in the training and the desired accuracy for the data reconstruction. In addition, the data size for the testing dataset should not be too small to avoid degradation in the prediction accuracy due to the effect of the hidden units at the two ends of the neural network. The results show that the BRNN-based method yields more accurate estimation than the other data reconstruction methods, such as PCA, MMSE, SVR, FNN and RNN. The results also show that the input sensors for data reconstruction can be chosen based on their correlation with the target output sensor. Furthermore, it is seen that the BRNN-based sensor data reconstruction method is robust against environmental effects in that a BRNN model trained with datasets from a certain month can be used to recover faulty datasets of other months.

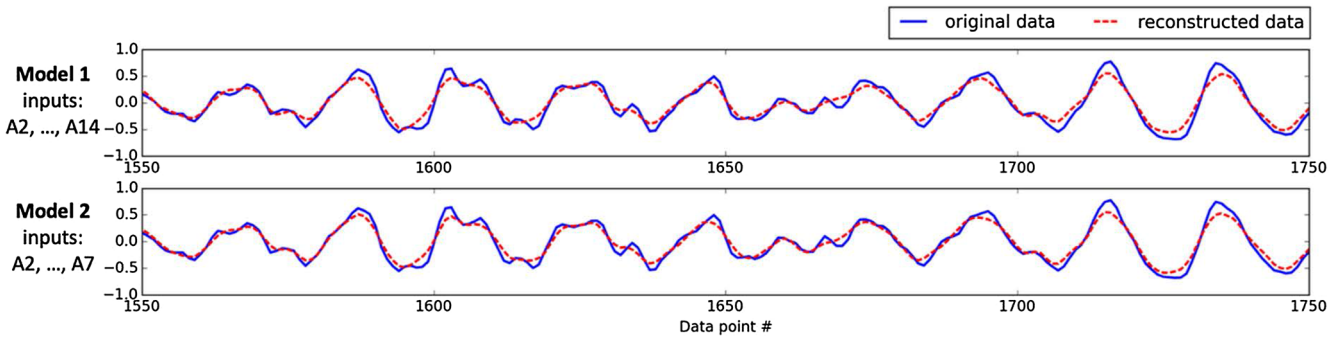


Fig. 11. Sensor data reconstruction for Sensor A1 with different input sensors.

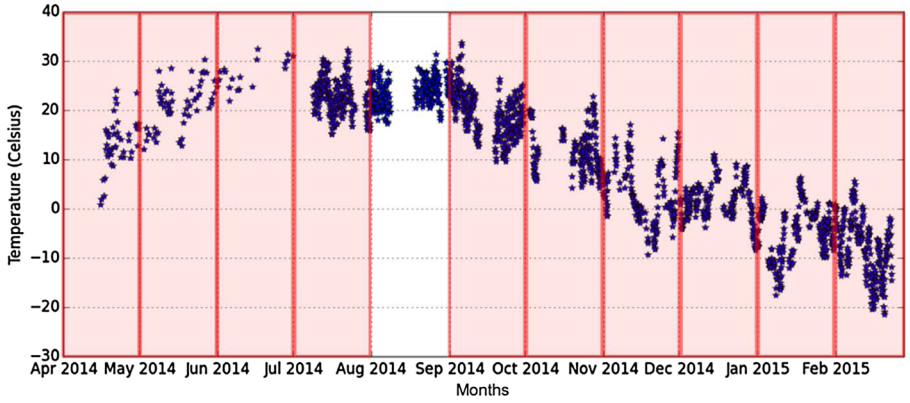


Fig. 12. Temperature change measured on the Telegraph Road Bridge from April 2014 to February 2015 (Months selected for analysis are highlighted with boxes).

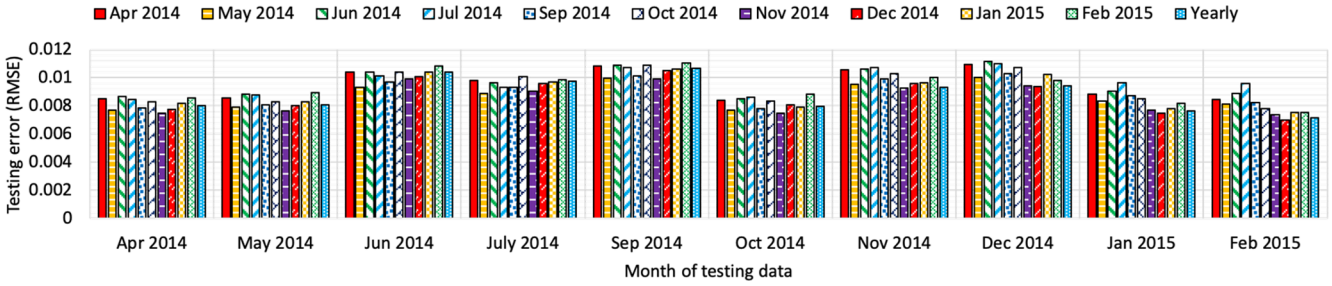


Fig. 13. Testing errors for testing sets collected from different months: 10 BRNN models trained with monthly data and 1 BRNN model trained with yearly data are used.

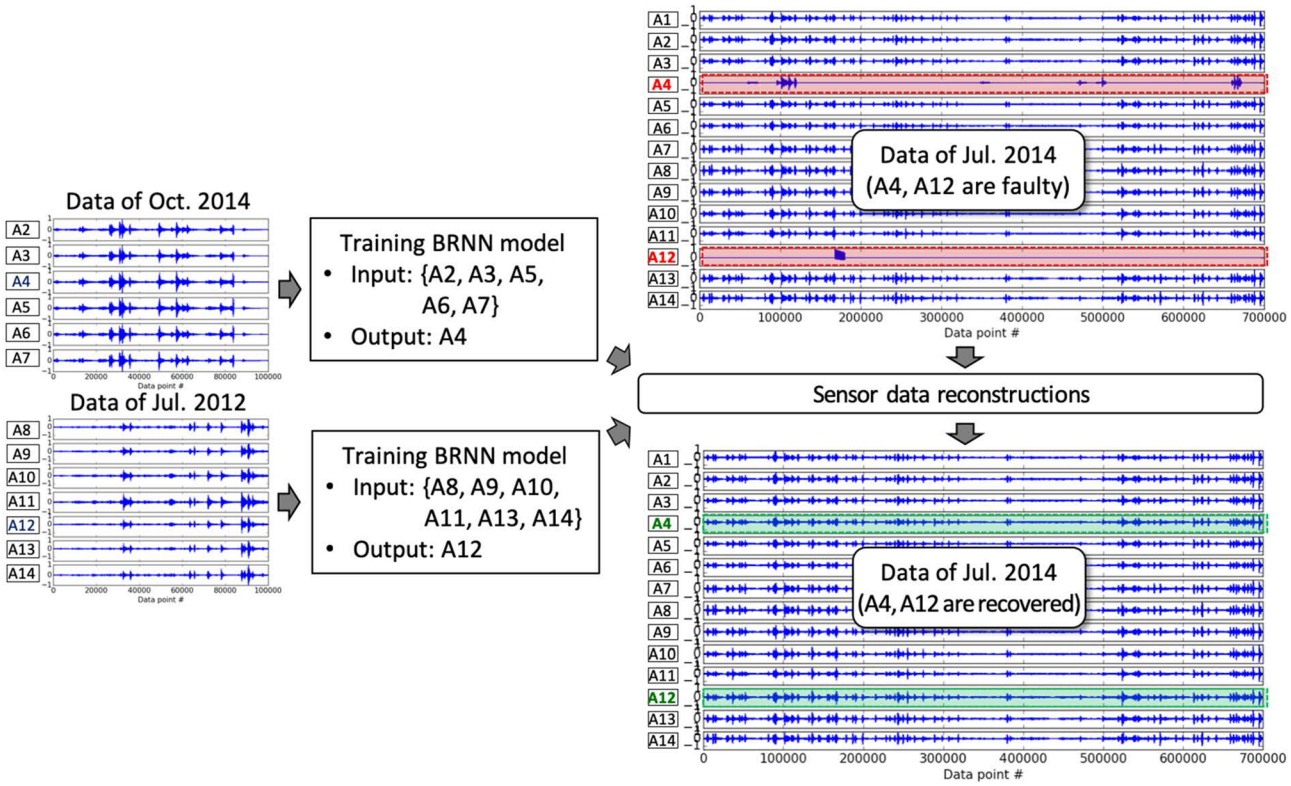
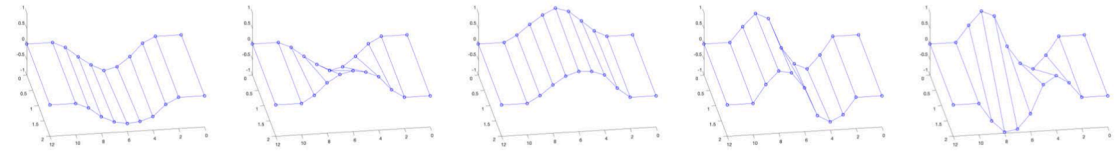


Fig. 14. Recovery of missing data using BRNN models trained with data from different months.

In this paper, the BRNN-based sensor data reconstruction method is tested using vibration response data collected from a bridge structure. The method can potentially be applied to different types of sensor data as well as to different engineering applications, as long as there exists sufficient spatiotemporal correlations among the sensor data. In this study, the BRNN method assumes that the data is collected from a network of synchronized sensors of the same type that operate at the same sampling rate. Future works may consider the application of the



$f_1 = 2.4902$ Hz $f_2 = 2.9297$ Hz $f_3 = 4.8340$ Hz $f_4 = 8.2031$ Hz $f_5 = 9.0332$ Hz

(a) Modal frequency and operational mode shapes computed from the reconstructed dataset

$f_1 = 2.4414$ Hz $f_2 = 2.9297$ Hz $f_3 = 4.8828$ Hz $f_4 = 8.0078$ Hz $f_5 = 9.082$ Hz

(b) Modal frequency and operational mode shapes obtained from experiments [37]

Fig. 15. First five modal frequencies and operational mode shapes of the Telegraph Road Bridge.

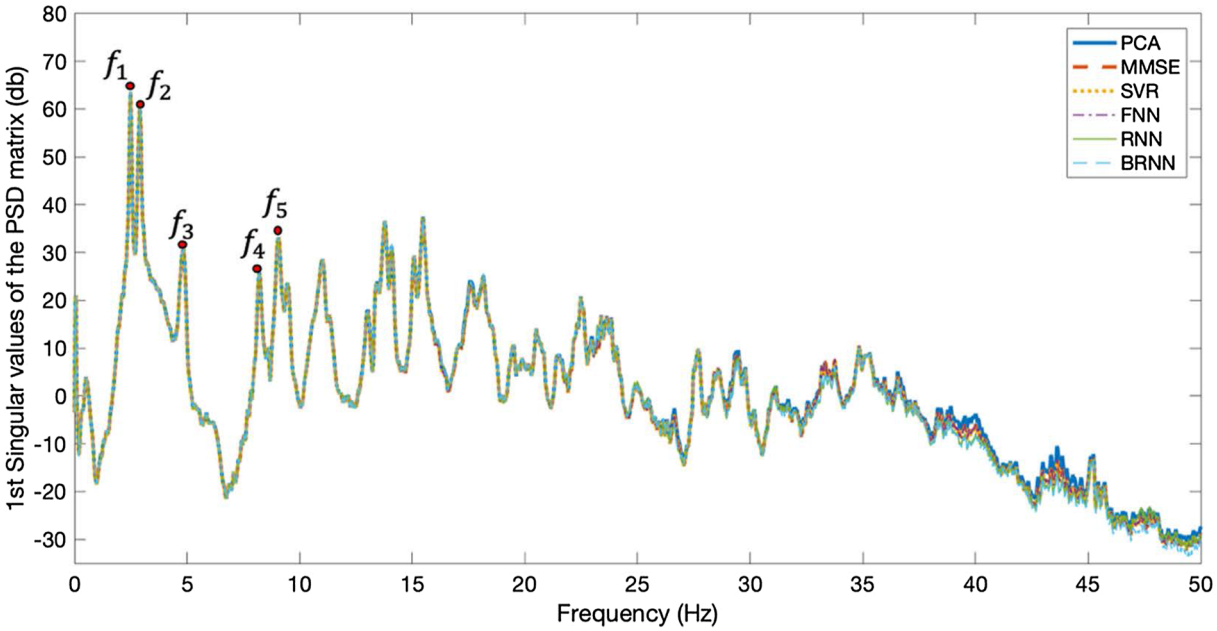


Fig. 16. Singular values of the power spectral density matrix of the recovered data ($f_1 = 2.4902$, $f_2 = 2.9297$, $f_3 = 4.8340$, $f_4 = 8.2031$, $f_5 = 9.0332$).

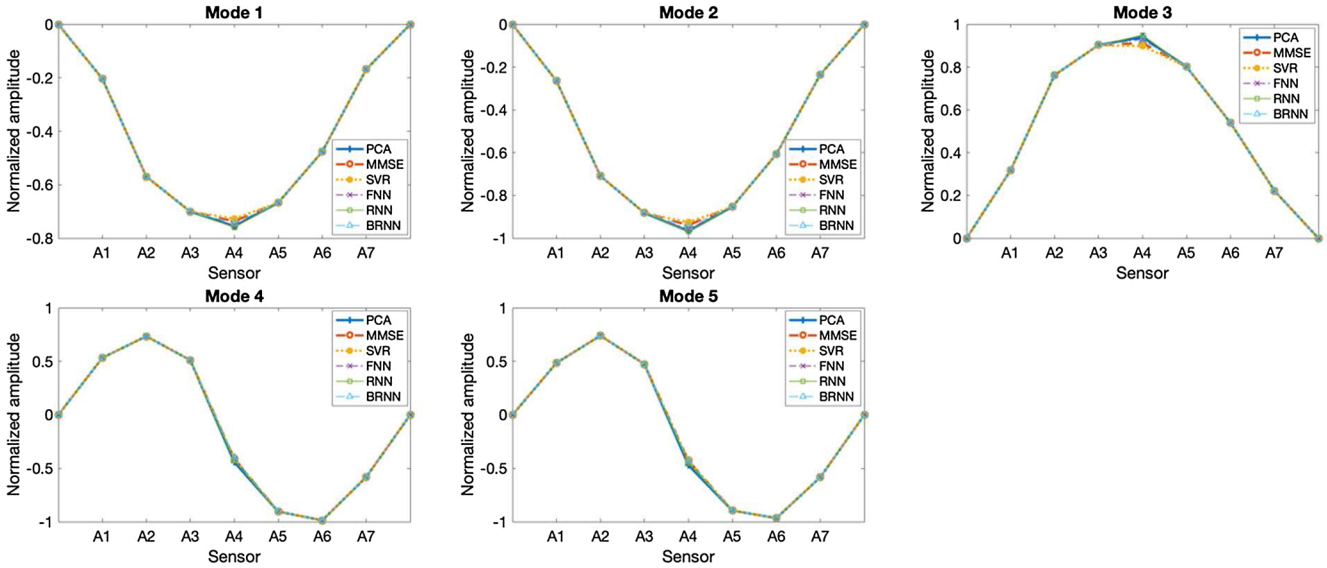


Fig. 17. Operational mode shapes of Girder 7 computed using the recovered data.

BRNN-based reconstruction method to a network of sensors that collect data at different sampling rate. Last but not least, while the BRNN model discussed in this study includes only a single hidden layer, the model can be extended to a deep neural network with multiple hidden layers, which may improve the accuracy of the data reconstruction method.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The research is supported by a collaborative project funded by the US National Science Foundation (Grant No. ECCS-1446330 to Stanford University and Grant No. CMMI-1362513 and ECCS-1446521 to the University of Michigan). This research is also supported by a Grant No. 13SCIPA01 from Smart Civil Infrastructure Research Program funded by Ministry of Land, Infrastructure and Transport (MOLIT) of Korea government and Korea Agency for Infrastructure Technology Advancement (KAIA). The authors thank the Michigan Department of Transportation (MDOT) for access to the Telegraph Road Bridge and for offering support during installation of the wireless monitoring system. Any opinions, findings, conclusions or recommendations expressed in this paper are solely those of the authors and do not necessarily reflect the views of NSF, MOLIT, MDOT, KAIA or any other organizations and collaborators.

References

[1] V. Venkatasubramanian, R. Rengaswamy, S. Kavuri, K. Yin, A review of process fault detection and diagnosis: Part III: process history based methods, Comput. Chem. Eng. 27 (3) (2003) 327–346.
[2] H. Sohn, C. Farrar, F. Hemez, J. Czarnecki, A Review of Structural Health Review of Structural Health Monitoring Literature 1996–2001, Los Alamos National Laboratory, Los Alamos, NM, 2002.
[3] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S.Z.S. Nair, E. Kohler, G. Pottie, M. Hansen, M. Srivastava, Sensor network data fault types, ACM Trans. Sens. Networks (TOSN) 5 (3) (2009) 25.
[4] Y. Zhang, Scalable data management and data-driven analytics for structural condition assessment using structural monitoring data, PhD Thesis University of Michigan, Deparment of Civil and Environmental Engineering, 2016.
[5] S. Qin, W. Li, Detection, identification, and reconstruction of faulty sensors with maximized sensitivity, AIChE J. 45 (9) (1999) 1963–1976.
[6] A. Pouliezos, G. Stavrakakis, Analytical redundancy methods, in: Real Time Fault Monitoring of Industrial Processes, Dordrecht, Springer, 1994.
[7] R.A.K.P. Kromanis, Data-driven approaches for measurement interpretation: analysing integrated thermal and vehicular response in bridge structural health monitoring, Adv. Eng. Inf. 34 (2017) 46–59.
[8] G. Kerschen, P. De Boe, J. Golinval, K. Worden, Sensor validation using principal component analysis, Smart Mater. Struct. 14 (1) (2005) 36–42.
[9] J. Kullaa, Sensor validation using minimum mean square error estimation, Mech. Syst. Sig. Process. 24 (5) (2010) 1444–1457.
[10] X. Xu, J. Hines, R. Uhrig, Sensor validation and fault detection using neural networks, in: Maintenance and Reliability Conference (MARCON 99), 1999.
[11] I. Eski, S. Erkaya, S. Savas, S. Yildirim, Fault detection on robot manipulators using artificial neural networks, Rob. Comput. Integr. Manuf. 27 (1) (2011) 115–123.
[12] K. Smarsly, K.H. Law, Decentralized fault detection and isolation in wireless structural health monitoring systems using analytical redundancy, Adv. Eng. Softw. 73 (2014) 1–10.
[13] K. Dragos, K. Smarsly, Distributed adaptive diagnosis of sensor faults using structural response data, Smart Mater. Struct. 25 (10) (2016) 105019–105115.
[14] K. Law, S. Jeong, M. Ferguson, A data-driven approach for sensor data reconstruction for bridge monitoring, in: 2017 World Congress on Advances in Structural Engineering and Mechanics, 2017.
[15] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE international conference on acoustics, speech and signal processing, 2013.
[16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint, vol. arXiv: 1406.1078, 2014.
[17] A. Rather, A. Agarwal, V. Sastry, Recurrent neural network and a hybrid model for prediction of stock returns, Expert Syst. Appl. 42 (6) (2015) 3234–3241.
[18] A.I. Moustapha, R.R. Selmic, Wireless sensor network modeling using modified recurrent neural networks: application to fault detection, IEEE Trans. Instrum. Meas. 57 (5) (2008) 981–988.
[19] M. Schuster, K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[20] Z. Lipton, J. Berkowitz, C. Elkan, A critical review of recurrent neural networks for sequence learning, ArXiv Preprint, p. arXiv:1506.00019, 2015.
[21] S. Jeong, M. Ferguson, K. Law, Sensor data reconstruction and anomaly detection using bidirectional recurrent neural network, in: SPIE Smart Structures/NDE Conference, Denver, CO, USA, 2019.
[22] Y. LeCun, L. Bottou, G. Orr, K. Müller, Efficient backprop, in: Neural networks: Tricks of the trade, Berlin, Heidelberg, Springer, 1998, pp. 9–50.
[23] A. Ng, J. Ngiam, C. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, S. Tandon, UFLDL tutorial, [Online]. Available: < http://deeplearning.stanford.edu/tutorial/ > [Accessed 15 January 2018].
[24] C. Manning, R. Socher, G. Fang, R. Mundra, CS224n: Natural Language Processing with Deep Learning, Lecture note, 2017.
[25] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: The Thirteenth International Conference on Artificial Intelligence and Statistics, 2010.
[26] P. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.
[27] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: The 3rd International Conference for Learning Representations, 2015.
[28] M. Hagan, H. Demuth, M. Beale, O. De Jess, Neural network design, 2014.
[29] R. Williams, J. Peng, An efficient gradient-based algorithm for on-line training of recurrent network trajectories, Neural Comput. 2 (4) (1990) 490–501.
[30] I. Sutskever, Training recurrent neural networks, PhD Thesis Department of Computer Science, University of Toronto, 2013.
[31] Y. Zhang, S. O'Connor, G. van der Linden, A. Prakash, J. Lynch, SenStore: a scalable cyberinfrastructure platform for implementation of data-to-decision frameworks for infrastructure health management, J. Comput. Civil Eng. 30 (5) (2016) 04016012.
[32] S. O'Connor, Y. Zhang, J. Lynch, M. Ettouney, P.O. Jansson, Long-term performance assessment of the Telegraph Road Bridge using a permanent wireless monitoring system and automated statistical process control analytics, Struct. Infrastruct. Eng. 13 (5) (2017) 604–624.
[33] PyTorch, "PyTorch," [Online]. Available: < http://pytorch.org/ > [Accessed 1 February 2018].
[34] C. Bishop, Pattern recognition and machine learning, springer, 2006.
[35] H. Sohn, Effects of environmental and operational variability on structural health monitoring, Philos. Trans. Roy. Soc. Lond. A: Math., Phys. Eng. Sci. 365 (1851) (2007) 539–560.
[36] R. Brincker, L. Zhang, P. Andersen, Modal identification of output-only systems using frequency domain decomposition, Smart Mater. Struct. 10 (3) (2001) 441–445.
[37] A. Mosavi, H. Sedarat, S. O'Connor, A. Emami-Naeini, V. Jacob, A. Krimotat, J. Lynch, Finite element model updating of a skewed highway bridge using a multivariable sensitivity-based optimization approach, in: Nondestructive Characterization for Composite Materials, Aerospace Engineering, Civil Infrastructure, and Homeland Security, San Diego, CA, USA, 2012.