An efficient algorithm for architecture design of Bayesian neural network in structural model updating

Tao Yin*

School of Civil Engineering, Wuhan University, Wuhan 430072, P.R. China

&

Hong-ping Zhu

School of Civil Engineering & Mechanics, Huazhong University of Science and Technology, Wuhan 430074, P.R. China

Abstract: *There has been growing interest in applying the artificial neural network (ANN) approach in structural system identification and health monitoring. The learning process of neural network can be more robust when presented in the Bayesian framework, and rational architecture of the Bayesian neural network is critical to its performance. Apart from number of hidden neurons, the specific forms of the transfer functions in both hidden and output layers are also crucially important. To our best knowledge, however, the simultaneous design of proper number of hidden neurons, and specific forms of hidden- and output-layer transfer functions has not yet been reported in terms of the Bayesian neural network. It's even more challenging when the transfer functions of both layers are parameterized instead of using fixed shape forms. This paper proposes a tailor-made algorithm for efficiently designing the appropriate architecture of Bayesian neural network with simultaneously optimized hidden neuron number and custom transfer functions in both hidden and output layers. To cooperate with the proposed algorithm, both the Jacobian of network function and Hessian of the negative logarithm of weight posterior are derived analytically by matrix calculus. This is much more accurate and efficient than the finite difference approximation, and also vital for properly designing the Bayesian neural network architecture as well as further quantifying the confidence interval of network prediction. The validity and efficiency of proposed methodology is verified through probabilistic finite element (FE) model updating of a pedestrian bridge by using the field measurement data.*

*Corresponding author: Dr. Tao Yin, Associate Professor, Wuhan University, Wuhan, P.R. China. E-mail: tyin@whu.edu.cn

1 INTRODUCTION

The FE model updating method has become a prevalent technique utilized in structural system identification and health monitoring. The accuracy of FE model is essential for ensuring its successful implementation. However, due to assumption and uncertainty arisen from the theoretical hypothesis, boundary condition, and geometric and material properties, there is an unavoidable mismatching between the measured and model-predicted dynamic characteristics. Thus, the FE model must be adjusted to improve its matching quality, which is generally an inverse process and known as the FE model updating. There is a strong interest in developing the FE model updating methods based on vibration measurements over the past few decades (Friswell and Mottershead, 1995), which have been applied to a variety of structural systems and components, such as beams (Levin and Lieven, 1998; Teughels et al., 2003; Simoen et al., 2015), trusses and frames (Adeli and Cheng, 1993; Katafygiotis and Beck, 1998; Law et al., 2001; Adeli and Jiang, 2006; Yin et al., 2009; Yu and Yin, 2010; Yuen, 2010; Boulkaibet et al., 2015; Yin et al., 2017; Oh et al., 2017), bridges (Brownjohn, 2003; Jaish and Ren, 2007; Jensen et al., 2014; Shabbir and Omenzetter, 2015; Park, et al., 2017), highrise and historic buildings (Jiang and Adeli,

2005; Astroza et al., 2016; Torres et al., 2017), railway sleepers (Lam et al., 2014), pipelines (Zhu et al., 2008; Yin et al., 2017; Yin et al., 2019), aerospace structures (Mottershead et al., 2011; Stochino et al., 2017), etc. in diverse engineering fields.

Most of the methods mentioned above solve the inverse problem of adjusting model parameters by minimizing the difference between the model-predicted and measured dynamic properties. As repeated solution of large scale eigenvalue problem is generally required in this process, the computational cost will become unaffordable when dealing with complex models with large amount of degrees of freedom. This inverse problem can, however, be efficiently transformed to a forward one that is significantly easier to be handled by the ANN approach. Although not specifically developed for the model updating problem, the excellent capability of pattern matching makes the ANN approach to be a very promising tool for this purpose.

For system identification and health monitoring of structures, the multi-layer neural networks are widely utilized in the literature (Adeli 2001; Sohn, et al., 2004; Lam and Ng, 2008; Adeli and Jiang, 2009; Arangio and Beck, 2012; Sirca & Adeli, 2012; Hakim et al., 2015; Chang et al., 2018; Yin and Zhu, 2018), and currently, deep learning neural networks have also begun to be applied in this area (Abdeljaber et al., 2017, Cha et al., 2017, Lin et al., 2017, Grande et al., 2017; Gao et al., 2018; Wang et al., 2018; Yang et al., 2018). In this paper, the commonly used multi-layer feedforward neural networks are investigated, and they have been confirmed to be able to approximate any functional relationship between inputs and outputs with a single hidden layer (Cybenko, 1989). It's also well recognized that, for the complexity of network with single hidden layer, the number of hidden neurons have a significant impact on the ANN training process and the performance of trained ANN, especially for complex function fittings, such as the FE model updating problem. Too small a hidden neuron number will result in a poor-quality network that fails to reveal the essential characteristics of training data, whereas too large a number might cause the output of the neural network to fluctuate within the area between training data points. Thus, reasonably designing the network architecture with an appropriate complexity is essential to guarantee the successful implementation of ANN-based model updating. However, in practice, the ANN architecture is generally determined only by rule of thumb or experience, and few publications addressed the ANN design issue in the area of structural system identification and health monitoring (Lam, et al., 2006; Yuen and Lam, 2006; Lam and Ng, 2008; Arangio and Beck, 2012; Yin and Zhu, 2018).

It is noted that the traditional ANN approach simply minimizes the sum of squared errors between the network output and the target variables to estimate the network weights and biases from the training data. In order to get better performance, the learning process in a neural network can be elaborated in the Bayesian statistical framework by incorporating the prior information about the network parameters, leading to the concept of Bayesian neural network that is more robust in both the training and prediction process than the traditional ANN. Beginning with the early research activities relevant to the Bayesian neural network (Buntine and Weigend, 1991; MacKay, 1992), the application of Bayesian inference to the area of neural network research has received more and more attention (MacKay, 1994; Neal, 1996; Lampinen and Vethari, 2001; Barber, 2002; Lee, 2004; Arangio and Beck, 2012; Yin and Zhu, 2018). Due to the importance of neural network design, attention has been paid to the reasonable choice of the number of hidden neurons for the Bayesian neural network (Arangio and Beck, 2012). Apart from the number of hidden neurons, specific forms of transfer (or activation) functions and hyperparameters also have a non-negligible effect on the network performance (Lam and Ng, 2008; Snoek et al. 2012; Yin and Zhu, 2018). To the best of our knowledge, however, for Bayesian neural network, the simultaneous design of appropriate hidden neuron number, together with the specific forms of transfer functions in both the hidden and output layers has not been reported yet in previous research works. The goal becomes quite cumbersome and more challenging when the transfer functions of both layers are generalized to be a family of parameterized functions as compared to the fixed shape forms. In addition, the most of publications related to the predictive output distribution from Bayesian neural networks only considered the case of univariate output (Bishop, 2006; Iruansi, et al., 2012; Kocadağlı, 2014), while the predictive distribution with multivariate output was rarely involved. But the single-target network is not applicable for structural model updating as the number of candidate parameters to be refined should definitely exceed one. Furthermore, accurate estimation of the posterior probability of network architecture and the predictive distribution over trained network outputs is very dependent on the Hessian of the negative logarithm of the posterior of weight vectors as well as the Jacobian of network function. But the commonly used finite difference method does not meet the requirements of computational accuracy and efficiency, which is also vital for properly designing the architecture of the Bayesian neural network and further quantifying the uncertain of network prediction.

In this paper, an efficient and tailor-made algorithm is developed for Bayesian neural network with multiple target variables in terms of designing suitable class of network architectures for FE model updating. By treating the network design procedure as a combinatorial optimization problem, the proposed algorithm is intended to simultaneously determine the proper hidden neuron number and the suitable forms of parameterized hidden- and output-layer transfer functions. The analytically derived the

Jacobian and Hessian matrices guarantee the accuracy and efficiency of evaluating the posterior probability of network architectures and predictive distribution over trained network outputs. The validity and effectiveness of the developed methodology is fully demonstrated by probabilistic FE model updating of a 39-m-long pedestrian steel bridge in Wuhan, China, with fielding testing data.

2 THEORETICAL DEVELOPEMENT

ANN is powerful computational models inspired by biological neural network system to approximate functions which are generally unknown. The neural networks take the advantage of learning procedure with parallel and distributed processing for performance improvement. In the neural network learning process, transfer functions play a very important role, and properly choosing transfer functions in neural networks is of vital importance to their performance (Duch and Jankowski, 1997).

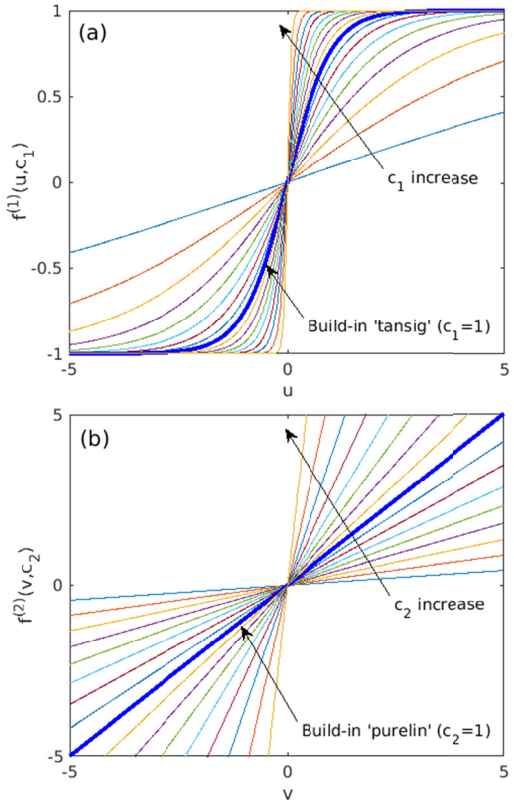

Fig. 1. Two custom transfer functions defined: (a) tansig-type for hidden layer; (b) purelin-type for output layer.

In this study, instead of utilizing the fixed shape transfer functions, i.e., the hyperbolic tangent sigmoid (tansig) and linear transfer functions (purelin) (Hagan et al., 1996), integrated in Matlab, as shown in Figure 1, the transfer functions employed in hidden and output layers are both generalized as a family of parameterized functions as

$$\mathcal{T}^{(1)}(u,c_1)=\frac{1-\exp(-2c_1u)}{1+\exp(-2c_1u)} \tag{1}$$

$$\mathcal{T}^{(2)}(v,c_2)=c_2v \tag{2}$$

where $\mathcal{T}^{(1)}$, $\mathcal{T}^{(2)}$ are transfer functions in hidden and output layers, and u, v denote arbitrary elements of the vector-valued inputs to the two layers. The two scaling parameters c_1 and c_2 control the slopes of transfer functions in both layers. In particular, $c_1=1$ and $c_2=1$ correspond to the build-in tansig and purelin in Matlab, respectively. Thus, the selection of hidden- and output-layer transfer functions in this paper is equivalent to determine suitable values of the two scaling parameters c_1 and c_2, where the flexibility and capability of network performance is expected to be improved as compared to the fixed shape functions.

In this study, the proposed algorithm emphasizes on designing the architecture of a feedforward Bayesian neural network with a single hidden layer, which specifically includes the simultaneous determination of suitable number of hidden neurons as well as appropriate values of scaling parameters corresponding to hidden- and output-layer transfer functions. Thus, instead of using the usual single index number, the class of network architectures is more conveniently defined to be a combination of these three design parameters by $\mathcal{A}_{N_H,c_1,c_2}$, and N_H denotes the number of hidden neurons. In this note, the overall network function with a single hidden layer can be given as:

$$\begin{aligned}\mathbf{y}\left(\mathbf{x},\mathbf{w};\mathcal{A}_{N_H,c_1,c_2}\right)&=\mathcal{T}^{(2)}(\mathbf{v}(\mathbf{x}),c_2)\\&=\mathcal{T}^{(2)}\left(\mathbf{W}^{(2)}\mathcal{T}^{(1)}(\mathbf{u}(\mathbf{x}),\boldsymbol{c_1})+\mathbf{b}^{(2)},c_2\right)\end{aligned} \tag{3}$$

where

$$\begin{aligned}\mathbf{u}(\mathbf{x})&=\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)}\in\mathbb{R}^{N_H\times1}\\\mathbf{v}(\mathbf{x})&=\mathbf{W}^{(2)}\mathcal{T}^{(1)}(\mathbf{u}(\mathbf{x}),c_1)+\mathbf{b}^{(2)}\in\mathbb{R}^{N_O\times1}\end{aligned} \tag{4}$$

and $\mathbf{x}\in\mathbb{R}^{N_I\times1}$, $\mathbf{y}\left(\mathbf{x},\mathbf{w};\mathcal{A}_{N_H,c_1,c_2}\right)\in\mathbb{R}^{N_O\times1}$ are vector-valued input and output of the neural network. N_I and N_O denote the number of neurons in both layers. The weight matrices $\mathbf{W}^{(1)}\in\mathbb{R}^{N_H\times N_I}$, $\mathbf{W}^{(2)}\in\mathbb{R}^{N_O\times N_H}$, and bias vectors $\mathbf{b}^{(1)}\in\mathbb{R}^{N_H\times1}$, $\mathbf{b}^{(2)}\in\mathbb{R}^{N_O\times1}$ together form the adaptive network parameters. The numbers 1 and 2 in parenthesis appearing in the superscripts denote the hidden and output layers, respectively. For convenience, all weights and biases can be stacked into a weight vector or network parameter vector $\mathbf{w}\in\mathbb{R}^{N_W\times1}$ as

$$\mathbf{w}^{\mathrm{T}}=\left[\mathrm{vec}\left(\mathbf{W}^{(1)}\right)^{\mathrm{T}}\left(\mathbf{b}^{(1)}\right)^{\mathrm{T}}\mathrm{vec}\left(\mathbf{W}^{(2)}\right)^{\mathrm{T}}\left(\mathbf{b}^{(2)}\right)^{\mathrm{T}}\right] \tag{5}$$

where $\mathrm{vec}(\cdot)$ represents the matrix vectorization by stacking the columns of a given matrix into a single column vector. $N_W=N_H(N_I+N_O+1)+N_O$ represents the total number of elements contained in network parameters $\mathbf{w}$.

By utilizing the neural network method, the FE model updating problem investigated in the study is equivalent to predict multiple target variables $\mathbf{t}$ representing the model-updating parameters from a vector $\mathbf{x}$ of inputs representing the modal characteristics by adjusting the adaptive network parameters $\mathbf{w}$. The network input $\mathbf{x}$, i.e., the modal properties, is obtained through eigenvalue analysis based on the structural FE model as follows

$$\mathbf{x} = \mathcal{G}(\mathbf{t}) + \boldsymbol{\eta} \tag{6}$$

where $\mathcal{G}$: $\mathbb{R}^{N_O} \to \mathbb{R}^{N_I}$ denotes the FE model that accepts the model parameters $\mathbf{t}$ extracted from the model parameter space generated with uniform-distribution assumption to predict the modal parameters $\mathbf{x}$. $\boldsymbol{\eta} \in \mathbb{R}^{N_I \times 1}$ is the sample noise vector, coming from a zero-mean Gaussian with varying levels of standard deviations $\{0, a_1, \cdots, a_{N_S}\}$, added into the calculated modal parameters $\mathbf{x}$ for ensuring a robust network after training. N_S is the total number of noise levels considered. Let $\mathcal{D}_M^{(0)} = \left\{\{\mathbf{x}_1^{(0)}, \mathbf{t}_1\}, \cdots, \{\mathbf{x}_M^{(0)}, \mathbf{t}_M\}\right\}$ denote M sets of input-output training samples obtained by FE analysis without noise, the full set of training data is the gathering of training data generated by imposing various levels of noise on the noise-free training input samples, i.e.,

$$\mathcal{D}_N = \left\{\mathcal{D}_M^{(0)}, \mathcal{D}_M^{(a_1)}, \cdots, \mathcal{D}_M^{(a_{N_S})}\right\} \tag{7}$$

where $\mathcal{D}_M^{(a_i)} = \left\{\{\mathbf{x}_1^{(a_i)}, \mathbf{t}_1\}, \cdots, \{\mathbf{x}_M^{(a_i)}, \mathbf{t}_M\}\right\}$ for $i = 1, \cdots, N_S$, and $N = M(N_S + 1)$. In this study, the design of Bayesian neural network aims to identify the most probable architecture class by utilizing the expanded training data $\mathcal{D}_N$ from N_A prescribed classes of network architectures.

To begin with, the target vector $\mathbf{t}$ are approximated as independent Gaussian distribution, which is conditional on the input vector $\mathbf{x}$ and network weights $\mathbf{w}$, with an $\mathbf{x}$-dependent mean provided by the network function in Eq. (3) and shared Gaussian noise parameter β (Bishop, 2006). In this note, the conditional distribution of the target vector in terms of the architecture class $\mathcal{A}_{N_H,c_1,c_2}$ is defined as

$$p\left(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta, \mathcal{A}_{N_H,c_1,c_2}\right) = \mathcal{N}\left(\mathbf{t}|\mathbf{y}\left(\mathbf{x}, \mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right), \beta^{-1}\mathbf{I}_{N_O}\right) \tag{8}$$

where $\mathbf{I}_{N_O} \in \mathbb{R}^{N_O \times N_O}$ is an identity matrix. In addition, the prior distribution for the uncertain network parameters $\mathbf{w}$ is chosen as a Gaussian:

$$p\left(\mathbf{w}|\alpha, \mathcal{A}_{N_H}\right) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}_{N_W}\right) \tag{9}$$

where $\mathbf{I}_{N_W} \in \mathbb{R}^{N_W \times N_W}$ is also an identity matrix. N_W denotes the dimension of the weight vector for this class of network architectures. Based on the conditional distribution of the target vector provided by Eq. (8), the likelihood function is conveniently constructed by utilizing the full set of input-output training data $\mathcal{D}_N$ as

$$\begin{aligned} &p\left(\mathcal{D}_N|\mathbf{w}, \beta, \mathcal{A}_{N_H,c_1,c_2}\right) \\ &= \prod_{n=1}^{N} \mathcal{N}\left(\mathbf{t}_n|\mathbf{y}(\mathbf{x}_n, \mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}), \beta^{-1}\mathbf{I}_{N_O}\right) \end{aligned} \tag{10}$$

By employing the Bayes' theorem, the posterior distribution of the uncertain weight vector $\mathbf{w}$ with the network architecture $\mathcal{A}_{N_H,c_1,c_2}$ is given by

$$\begin{aligned} &p\left(\mathbf{w}|\mathcal{D}_N, \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right) \\ &= \frac{p\left(\mathcal{D}_N|\mathbf{w}, \beta, \mathcal{A}_{N_H,c_1,c_2}\right)p\left(\mathbf{w}|\alpha, \mathcal{A}_{N_H}\right)}{p\left(\boldsymbol{\mathcal{D}}_N|\alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right)} \end{aligned} \tag{11}$$

It is seen from Eq. (11) that the Bayesian neural network in this paper is related to the concept of automatic relevance determination (ARD) (Yuen and Mu, 2015; Mu and Yuen, 2016). For the investigated problem, there is only one hyperparameter utilized as a regularization factor, instead of associating each model parameter with an individual hyperparameter. It is noted that the posterior distribution in Eq. (11) is not Gaussian since the network function given in Eq. (3) is nonlinearly dependent on weight parameters $\mathbf{w}$. However, one can achieve a local Gaussian approximation by employing the Laplace approximation as

$$\begin{aligned} &p\left(\mathbf{w}|\mathcal{D}_N, \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right) \\ &= \mathcal{N}\left(\mathbf{w}|\widehat{\mathbf{w}}, \mathbf{A}^{-1}\left(\widehat{\mathbf{w}}; \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right)\right) \end{aligned} \tag{12}$$

where $\widehat{\mathbf{w}}$ is the local maximum obtained through the usual nonlinear optimization algorithm by maximizing the logarithm of the posterior distribution in Eq. (11), i.e.,

$$\begin{aligned} \widehat{\mathbf{w}} &= \arg\max_{\mathbf{w}}\left\{\ln\left[p\left(\mathbf{w}|\mathcal{D}_N, \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right)\right]\right\} \\ &\propto \arg\max_{\mathbf{w}}\left\{-\beta\mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) - \alpha\|\mathbf{w}\|^2\right\} \end{aligned} \tag{13}$$

and $\mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$ denotes the sum-of-squares error function between the neural network output and target, i.e.,

$$\mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = \sum_{n=1}^{N}\left\|\mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right\|^2 \tag{14}$$

where $\mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = \mathbf{y}\left(\mathbf{x}_n, \mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) - \mathbf{t}_n$, and $\|\cdot\|$ represents the usual Euclidean norm. The matrix $\mathbf{A}\left(\widehat{\mathbf{w}}; \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right) \in \mathbb{R}^{N_W \times N_W}$ represents the Hessian of the negative logarithm of the posterior evaluated at the maximum of the posterior $\widehat{\mathbf{w}}$ for the given class of network architectures $\mathcal{A}_{N_H,c_1,c_2}$, i.e.,

$$\begin{aligned} &\mathbf{A}\left(\widehat{\mathbf{w}}; \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right) \\ &= -\nabla\nabla\ln\left[p\left(\mathbf{w}|\mathcal{D}_N, \alpha, \beta, \mathcal{A}_{N_H,c_1,c_2}\right)\right]\Big|_{\mathbf{w}=\widehat{\mathbf{w}}} \\ &= 1/2\left(\beta\mathbf{H}\left(\widehat{\mathbf{w}}; \mathcal{A}_{N_H,c_1,c_2}\right) + \alpha\mathbf{I}_{N_W}\right) \end{aligned} \tag{15}$$

where $\mathbf{H}\left(\widehat{\mathbf{w}}; \mathcal{A}_{N_H,c_1,c_2}\right) \in \mathbb{R}^{N_W \times N_W}$ is the Hessian matrix of $\mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$ evaluated at $\widehat{\mathbf{w}}$, i.e.,

$$\mathbf{H}\left(\hat{\mathbf{w}};\mathcal{A}_{N_H,c_1,c_2}\right)=\begin{bmatrix}\mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} & \mathbf{H}_{14}\\ & \mathbf{H}_{22} & \mathbf{H}_{23} & \mathbf{H}_{24}\\ & & \mathbf{H}_{33} & \mathbf{H}_{34}\\ \text{sym} & & & \mathbf{H}_{44}\end{bmatrix}_{\mathbf{w}=\hat{\mathbf{w}}} \tag{16}$$

where $\mathbf{H}_{11}\in\mathbb{R}^{N_IN_H\times N_IN_H}$, $\mathbf{H}_{12}\in\mathbb{R}^{N_IN_H\times N_H}$, $\mathbf{H}_{13}\in\mathbb{R}^{N_IN_H\times N_HN_O}$, $\mathbf{H}_{14}\in\mathbb{R}^{N_IN_H\times N_O}$, $\mathbf{H}_{22}\in\mathbb{R}^{N_H\times N_H}$, $\mathbf{H}_{23}\in\mathbb{R}^{N_H\times N_HN_O}$, $\mathbf{H}_{24}\in\mathbb{R}^{N_H\times N_O}$, $\mathbf{H}_{33}\in\mathbb{R}^{N_HN_O\times N_HN_O}$, $\mathbf{H}_{34}\in\mathbb{R}^{N_HN_O\times N_O}$ and $\mathbf{H}_{44}\in\mathbb{R}^{N_O\times N_O}$ are the submatrices of the symmetric Hessian and derived analytically in the Appendix by using Kronecker products and matrix calculus. It's much more accurate and efficient than using the finite difference method to approximated the Hessian, which is vital important for network architecture selection.

In the evidence framework, following the similar procedure as Bishop (2006), the point estimates for hyperparameters α and β can be obtained with the Laplace approximation by maximizing the log evidence

$$\begin{aligned}\ln\left[p\left(\mathcal{D}_N|\alpha,\beta,\mathcal{A}_{N_H,c_1,c_2}\right)\right]&\\ \simeq NN_O\ln\beta-\beta J\left(\hat{\mathbf{w}};\mathcal{A}_{N_H,c_1,c_2}\right)&\\ -NN_O\ln(2\pi)+W\ln\alpha-\alpha\|\hat{\mathbf{w}}\|^2&\\ -\ln\left|\mathbf{A}\left(\hat{\mathbf{w}};\alpha,\beta,\mathcal{A}_{N_H,c_1,c_2}\right)\right|&\end{aligned} \tag{17}$$

at $\hat{\mathbf{w}}$ with respect to α and β, and one can obtain

$$\hat{\alpha}=W/\left(\|\hat{\mathbf{w}}\|^2+\mathrm{tr}\left[\mathbf{A}^{-1}\left(\hat{\mathbf{w}};\hat{\alpha},\hat{\beta},\mathcal{A}_{N_H,c_1,c_2}\right)\right]\right)$$
$$\hat{\beta}=\left(NN_O-\hat{\alpha}\|\hat{\mathbf{w}}\|^2\right)/J\left(\hat{\mathbf{w}};\mathcal{A}_{N_H,c_1,c_2}\right)$$

Instead of marginalizing over all possible values of hyperparameters, the evidence $p\left(\mathcal{D}_N|\mathcal{A}_{N_H,c_1,c_2}\right)$ can be conveniently approximated by substituting their point estimates into the evidence $p\left(\mathcal{D}_N|\alpha,\beta,\mathcal{A}_{N_H,c_1,c_2}\right)$ as

$$\begin{aligned}p\left(\mathcal{D}_N|\mathcal{A}_{N_H,c_1,c_2}\right)\simeq&\\ p\left(\mathcal{D}_N|\hat{\mathbf{w}},\hat{\beta},\mathcal{A}_{N_H,c_1,c_2}\right)p\left(\hat{\mathbf{w}}|\hat{\alpha},\mathcal{A}_{N_H}\right)&\\ (2\pi)^{W/2}\left|\mathbf{A}\left(\hat{\mathbf{w}};\hat{\alpha},\hat{\beta},\mathcal{A}_{N_H,c_1,c_2}\right)\right|^{-1/2}&\end{aligned} \tag{18}$$

Following the Bayes' theorem, the probability of network architecture class $\mathcal{A}_{N_H,c_1,c_2}$ conditional on the training data $\mathcal{D}_N$ can be calculated to determine the most plausible architecture class for Bayesian neural network within the full set of potential architecture classes, and this yields:

$$\begin{aligned}&p\left(\mathcal{A}_{N_H,c_1,c_2}|\mathcal{D}_N,\mathcal{U}\right)\\ &=\frac{p\left(\mathcal{D}_N|\mathcal{A}_{N_H,c_1,c_2},\mathcal{U}\right)p\left(\mathcal{A}_{N_H,c_1,c_2}|\mathcal{U}\right)}{\sum_{j=1}^{N_A}p\left(\mathcal{D}_N|\mathcal{A}_{N_H,c_1,c_2},\mathcal{U}\right)p\left(\mathcal{A}_{N_H,c_1,c_2}|\mathcal{U}\right)}\end{aligned} \tag{19}$$

where $\mathcal{U}$ denotes the judgment on the initial plausibility of the network architecture class, and j is the index of one of the N_A candidate network architecture classes. As there is generally no idea about the suitable network architecture for a given problem at the beginning, it's just assumed that each class of architectures has an equal prior possibility $1/N_A$. In this study, the optimal class of network architectures is achieved by maximizing the posterior $p\left(\mathcal{A}_{N_H,c_1,c_2}|\mathcal{D}_N,\mathcal{U}\right)$, which is equivalent to maximizing the evidence $p\left(\mathcal{D}_N|\mathcal{A}_{N_H,c_1,c_2},\mathcal{U}\right)$ with respect to the number of hidden neurons N_H, and scaling parameters c_1 and c_2 of hidden- and output-layer transfer functions simultaneously. By assuming further that the probability distribution is solely specified by the class of network architectures $\mathcal{A}_{N_H,c_1,c_2}$, the user's preference $\mathcal{U}$ can thus be dropped from the notation hereafter for brevity. Instead of directly utilizing the evidence in Eq. (18), it's more convenient to maximize the following log-evidence form

$$\begin{aligned}\ln\left[p\left(\mathcal{D}_N|\mathcal{A}_{N_H,c_1,c_2}\right)\right]\simeq NN_O\left[\ln\hat{\beta}-\ln(2\pi)-1\right]&\\ +W\ln\hat{\alpha}-\ln\left|\mathbf{A}\left(\hat{\mathbf{w}};\hat{\alpha},\hat{\beta},\mathcal{A}_{N_H,c_1,c_2}\right)\right|&\end{aligned} \tag{20}$$

which depends solely on the hyperparameter estimates and the corresponding Hessian matrix. In this study, for the given training data $\mathcal{D}_N$, the class of network architectures to be selected from the entire prescribed set is the one having the highest value of log-evidence given in Eq. (20).

It is noted that the evidence of network architecture class given in Eq. (18) is conceptually equivalent to that given in Beck and Yuen (2004) with respect to a set of parameterized FE models, and both includes two terms. The first term named as likelihood factor favors more complex model parameterization scheme, whereas the second term, i.e., the Ockham factor, ensures a resultant model that fits the data with a suitable complexity by imposing a penalty against such parameterization complexity. In this note, however, it seems to be some counterintuitive that why the present study intends to select the proper values for scaling parameters c_1, c_2 of hidden- and output-layer transfer functions besides of the number of hidden neurons N_H, as the complexity of network architecture solely depends on N_H but not on c_1 and c_2. This is understandable that, although c_1 and c_2 do not affect the network complexity, they do have a direct impact on the maximum a posteriori estimation of network parameters $\hat{\mathbf{w}}$ as seen from Eq. (13), the Hessian in Eq. (16), and the point estimation of the hyperparameters $\hat{\alpha}$ and $\hat{\beta}$. As a result, the values of so-called likelihood factor, Ockham factor as well as the log evidence are obviously dependent on both c_1 and c_2 as well. Thus, instead of the penalty on model complexity, the Ockham factor should be more generally understood as a penalty against the uncertainty of identified model parameters, which will be explained later. This also clearly interprets the importance and significance of this study to simultaneously select the hidden neuron number and proper scaling parameters of transfer functions in both layers.

Algorithm 1 Proposed architecture design algorithm for Bayesian neural network

1. **Input:** Initialize the temporary maximum number of hidden neurons to be N_{H0}, the hidden- and output-layer scaling parameters $c_{1,0}=1$ and $c_{2,0}=1$.

2. Generate two sequences of scaling parameters $\{c_{1,1}, \cdots, c_{1,i} = c_{1,0}, \cdots, c_{1,N_{c_1}}\} \in \mathbb{R}^{N_{c_1} \times 1}$ and $\{c_{2,1}, \cdots, c_{2,i} = c_{2,0}, \cdots, c_{2,N_{c_2}}\} \in \mathbb{R}^{N_{c_2} \times 1}$.
3. Set the index of the main loop $k = 1$.
4. **While** convergence criterion is not satisfied (main loop)
5. **For** $i = 1$ to N_{H0} (inner loop 1)
6. Calculate the log evidence $\ln[p(\mathcal{D}_N|\mathcal{A}_{i,c_{1,k},c_{2,k}})]$ with i hidden neurons and scaling parameters $c_{1,k}$, $c_{2,k}$.
7. Update i to $i+1$, and calculate $\ln[p(\mathcal{D}_N|\mathcal{A}_{i+1,c_{1,k},c_{2,k}})]$, which is compared with $\ln[p(\mathcal{D}_N|\mathcal{A}_{i,c_{1,k},c_{2,k}})]$.
8. If $\ln[p(\mathcal{D}_N|\mathcal{A}_{i+1,c_{1,k},c_{2,k}})] < \ln[p(\mathcal{D}_N|\mathcal{A}_{i,c_{1,k},c_{2,k}})]$, output $n_{H0} = i$ for the given $c_{1,k}, c_{2,k}$, and end this inner loop.
9. Otherwise, if $i < N_{H0}$, increase i by 1 and go to Step 5.
10. If $i = N_{H0}$, set $n_{H0} = N_{H0}$ and end this inner loop.
11. **End for**
12. Set hidden neuron number and output-layer scaling parameter to be n_{H0} and $c_{2,k}$ recorded in the previous inner loop, and initialize the hidden-layer scaling parameter as $c_{1,1}$.
13. **For** $i = 1$ to N_{c_1} (inner loop 2)
14. Define an integer number $N_p > 1$, and calculate N_p consecutive log-evidence values for $n_{H0}, \cdots, n_{H0} + N_p - 1$ hidden neurons, respectively,
$$\ln[p(\mathcal{D}_N|\mathcal{A}_{n_{H0},c_{1,i},c_{2,k}})], \cdots, \ln\left[p\left(\mathcal{D}_N\middle|\mathcal{A}_{n_{H0}+N_p-1,c_{1,i},c_{2,k}}\right)\right]$$
15. **End for**
16. Find both the optimized n_{H1} and $c_{1,i}$ with respect to the maximum log-evidence value:
$$(n_{H1,\text{best}}, c_{1,\text{best}}): = \arg\max_{(n_{H1},c_{1,i})} \left\{\ln[p(\mathcal{D}_N|\mathcal{A}_{n_{H1},c_{1,i},c_{2,k}})], \text{for } n_{H1} = n_{H0}, \cdots, n_{H0} + N_p - 1 \text{ and } c_{1,i} = c_{1,1}, c_{1,2}, \cdots, c_{1,N_{c_1}}\right\}$$
17. Set hidden neuron number and hidden-layer scaling parameter to be $n_{H1,\text{best}}$ and $c_{1,\text{best}}$ recorded in the previous inner loop, and initialize the output-layer scaling parameter as $c_{2,1}$.
18. **For** $i = 1$ to N_{c_2} (inner loop 3)
19. Calculate N_p ($N_p > 1$) consecutive log-evidence values for $n_{H1,\text{best}}, \cdots, n_{H1,\text{best}} + N_p - 1$ hidden neurons, respectively,
$$\ln[p(\mathcal{D}_N|\mathcal{A}_{n_{H1,\text{best}},c_{1,\text{best}},c_{2,i}})], \cdots, \ln\left[p\left(\mathcal{D}_N\middle|\mathcal{A}_{n_{H1,\text{best}}+N_p-1,c_{1,\text{best}},c_{2,i}}\right)\right]$$
20. **End for**
21. Find both the optimized n_{H2} and $c_{2,i}$ wth respect to the maximum log-evidence value:
$$(n_{H2,\text{best}}, c_{2,\text{best}}): = \arg\max_{(n_{H2},c_{2,i})} \left\{\ln[p(\mathcal{D}_N|\mathcal{A}_{n_{H2},c_{1,\text{best}},c_{2,i}})], \text{for } n_{H2} = n_{H1}, \cdots, n_{H1} + N_p - 1 \text{ and } c_{2,i} = c_{2,1}, c_{2,2}, \cdots, c_{2,N_{c_2}}\right\}$$
22. If $k > 1$, test if the condition $c_{1,\text{best}} = c_{1,\text{old}}$, and $c_{2,\text{best}} = c_{2,\text{old}}$ is met; If yes, stop the while loop and go to Step 28.
23. Otherwise, save the optimized scaling parameters in the current main loop, i.e., $c_{1,\text{old}} = c_{1,\text{best}}$, and $c_{2,\text{old}} = c_{2,\text{best}}$.
24. Set the main loop index $k = k + 1$.
25. Update the temporary maximum hidden neuron number to be $N_{H0} = \max\{N_{H0}, n_{H2,\text{best}}\}$.
26. Update the scaling parameters to be $c_{1,k+1} = c_{1,\text{best}}$, $c_{2,k+1} = c_{2,\text{best}}$, and go to Step 5.
27. **End while**
28. **Output:** simultaneously optimized hidden neuron number $n_{H2,\text{best}}$, and hidden- and output-layer scaling parameters $c_{1,\text{best}}$ and $c_{2,\text{best}}$.

It should be pointed out that if the 'best' class of network architectures is identified by directly comparing log-evidence for all classes with Eq. (20), the computational burden for such an exhaustive way would be unaffordable. Thus, instead of directly picking up the 'best' one, this paper formulates the selection process as an optimization problem, which is solved very efficiently by a properly designed algorithm tailor-made for handling this issue, where the number of hidden neurons together with the scaling parameters of transfer functions in both layers is simultaneously identified. Considering a sequence of discrete sample values with a total number of N_{c_1} and N_{c_2} assigned to the scaling parameters c_1 and c_2, respectively, the proposed algorithm consists of one main loop and three inner loops. The main loop controls the convergence of the whole algorithm, and the three inner loops correspond to sequential estimation for N_H, and the combinations $\{N_H, c_1\}$ and $\{N_H, c_2\}$, respectively. One can refer to Algorithm 1 for more detailed information. It is also illustrated in a flowchart as shown in Figure 2, providing a more intuitive representation of the entire flow of the proposed algorithm. In this figure, the main steps of Algorithm 1 are identified, and the inner loops are referred by different colors for clarity. It is also noted that the order of last two inner loops can be exchanged, leading to two different search strategies I and II, i.e.,$\{N_H, c_1\} \to \{N_H, c_2\}$ and $\{N_H, c_2\} \to \{N_H, c_1\}$, which are also indicated in the flowchart.

Denote $\mathcal{A}^*_{N_H,c_1,c_2}$ to be the class of Bayesian neural network architectures with the simultaneously optimized hidden neuron number, transfer-function scaling parameters for both hidden and output layers obtained through the proposed design algorithm. By substituting the point estimate of hyperparameters $\hat{\alpha}^*$ and $\hat{\beta}^*$ achieved at the maximum a posteriori estimation $\widehat{\mathbf{w}}^*$ corresponding to the optimized class of network architectures into Eq. (13), the posterior probability distribution of network parameters can be approximated as Gaussian:

$$p(\widehat{\mathbf{w}}^*|\mathcal{D}_N, \hat{\alpha}^*, \hat{\beta}^*, \mathcal{A}^*_{N_H,c_1,c_2}) = \mathcal{N}\left(\mathbf{w}^*|\widehat{\mathbf{w}}^*, \mathbf{A}^{-1}(\widehat{\mathbf{w}}^*; \hat{\alpha}^*, \hat{\beta}^*, \mathcal{A}^*_{N_H,c_1,c_2})\right) \quad (21)$$

where the superscript $*$ denotes the quantities corresponding to the optimized Bayesian neural network architecture.

The predictive distribution of network output can be further achieved by marginalizing over the posterior distribution of network weights provided in Eq. (21) as

$$p(\mathbf{t}|\mathbf{x}, \mathcal{D}_N, \mathcal{A}^*_{N_H,c_1,c_2}) = \int p(\mathbf{t}|\mathbf{x}, \mathbf{w}^*, \hat{\beta}^*, \mathcal{A}^*_{N_H,c_1,c_2})$$

$$p\left(\mathbf{w}^*|\mathcal{D}_N,\hat{\alpha}^*,\hat{\beta}^*,\mathcal{A}^*_{N_H,c_1,c_2}\right)\mathrm{d}\mathbf{w}^* \quad (22)$$

It is worth noting that the integration form in Eq. (22) is analytically difficult to handle because of the nonlinear dependence of network model $\mathbf{y}(\mathbf{x},\mathbf{w};\mathcal{A}_{N_H,c_1,c_2})$ on the high-dimensional network weights $\mathbf{w}$. To make progress, an approximation approach is generally employed to approximate the integral with a finite sum as

$$p(\mathbf{t}|\mathbf{x},\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}) \simeq \frac{1}{L}\sum_{i=1}^{L} p\left(\mathbf{t}|\mathbf{x},\mathbf{w}_i^*,\hat{\beta}^*,\mathcal{A}^*_{N_H,c_1,c_2}\right) \quad (23)$$

where $\{\mathbf{w}_i^*\}$ denotes a sample of network parameters drawn from the posterior distribution $p\left(\mathbf{w}^*|\mathcal{D}_N,\hat{\alpha}^*,\hat{\beta}^*,\mathcal{A}^*_{N_H,c_1,c_2}\right)$. It is noted that drawing such sample of parameter vectors representative of the posterior distribution is generally not easy. To fulfill this purpose, a Markov chain Monte Carlo technique is generally employed. However, the main shortcoming of this method is the considerable amount of computational cost especially for the situation of high-dimensional network parameter space.

In this paper, the predictive distribution given in Eq. (22) is achieved in a significantly more efficient way as compared to the Monte Carlo method. Assuming that the covariance of posterior distribution of uncertain network weights to be small (Bishop, 2006), the linear approximation of the nonlinear network function is achieved through taking a Taylor series expansion around the maximum a posteriori estimation $\widehat{\mathbf{w}}^*$ and solely keeping the linear terms, i.e.,

$$\begin{aligned} &p\left(\mathbf{t}|\mathbf{x},\mathbf{w}^*,\hat{\beta}^*,\mathcal{A}^*_{N_H,c_1,c_2}\right)\\ &\simeq \mathcal{N}\left(\mathbf{t}|\mathbf{y}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right)\right.\\ &\left.+\,\mathbf{G}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right)\Delta\mathbf{w}^*,\hat{\beta}^{*\,-1}\mathbf{I}_{N_O}\right) \end{aligned} \quad (24)$$

where $\Delta\mathbf{w}^* = \mathbf{w}^* - \widehat{\mathbf{w}}^*$, and the $\mathbf{x}$-dependent Jacobian matrix $\mathbf{G}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right)$ of network function analytically evaluated at $\widehat{\mathbf{w}}^*$ is expressed as following:

$$\mathbf{G}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right) = [\mathbf{G}_1 \quad \mathbf{G}_2 \quad \mathbf{G}_3 \quad \mathbf{G}_4]|_{\mathbf{w}^*=\widehat{\mathbf{w}}^*} \quad (25)$$

$\mathbf{G}_1 \in \mathbb{R}^{N_O\times N_I N_H}$, $\mathbf{G}_2 \in \mathbb{R}^{N_O\times N_H}$, $\mathbf{G}_3 \in \mathbb{R}^{N_O\times N_H N_O}$, and $\mathbf{G}_4 \in \mathbb{R}^{N_O\times N_O}$ are the submatrices of the $\mathbf{x}$-dependent Jacobian matrix and given analytically in Appendix.

Thus, the predictive distribution over the Bayesian neural network output with optimized architecture given in Eq. (22) can be further approximated as a multivariate Gaussian:

$$\begin{aligned} &p\left(\mathbf{t}|\mathbf{x},\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right) =\\ &\mathcal{N}\left(\mathbf{t}|\mathbf{y}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right),\hat{\beta}^{*\,-1}\mathbf{I}_{N_O}+\mathbf{G}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right)\right.\\ &\left.\mathbf{A}^{-1}\left(\widehat{\mathbf{w}}^*;\hat{\alpha}^*,\hat{\beta}^*,\mathcal{A}^*_{N_H,c_1,c_2}\right)\mathbf{G}^{\mathrm{T}}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right)\right) \end{aligned} \quad (26)$$

whose mean is given by the network function $\mathbf{y}\left(\mathbf{x},\widehat{\mathbf{w}}^*;\mathcal{A}^*_{N_H,c_1,c_2}\right)$ for the optimized Bayesian neural network architecture with the most probable network weights $\widehat{\mathbf{w}}^*$. It should be noted that the covariance matrix in Eq. (26) is the combination of the uncertainty arisen from intrinsic noise on target variable and uncertainty of network parameters.

Within the framework of the proposed methodology, the procedure for performing the probabilistic FE model updating is straightforward. When the measured modal parameters $\mathbf{x}_t$ are available, the distribution of updated model parameters can be conveniently achieved with the predictive distribution over the trained network output provided in Eq. (26). Then, one can further obtain the mathematical expectation of modal parameters predicted from the updated model by an integral with respect to the predictive distribution of updated model parameters as

$$\mathrm{E}\left[\mathcal{G}(\mathbf{t});\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right] = \int_{\boldsymbol{\Theta}} \mathcal{G}(\mathbf{t})p(\mathbf{t}|\mathbf{x}_t,\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2})\mathrm{d}\mathbf{t} \quad (27)$$

where $\mathcal{G}(\mathbf{t})$ represents the modal properties calculated from the structural model by accepting model parameters $\mathbf{t}$ through modal analysis procedure. $\mathbf{x}_t$ denotes the measured modal properties employed for the model updating, and $\boldsymbol{\Theta}$ is the model parameter space of the updated model.

It is apparent that Eq. (27) depending on the predictive distribution in Eq. (26) is generally difficult to handle analytically. To make progress, the Monte Carlo simulation technique is adopted herein to predict the distribution of modal properties. The integral in Eq. (27) is approximated with the Monte Carlo sampling technique by generating a sequence of vectors $\{\mathbf{t}_1\},\cdots,\{\mathbf{t}_K\}$ which forms a stationary Markov chain. Thus, the expected value in Eq. (27) and corresponding variance-covariance matrix can be approximated as, respectively,

$$\mathrm{E}\left[\mathcal{G}(\mathbf{t});\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right] \approx \frac{1}{K}\sum_{i=1}^{K}\mathcal{G}(\mathbf{t}_i) \quad (28)$$

and

$$\begin{aligned} \boldsymbol{\Sigma}\left(\mathcal{G}(\mathbf{t});\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right) &\approx \frac{1}{K}\sum_{i=1}^{K}\mathcal{G}(\mathbf{t}_i)\mathcal{G}(\mathbf{t}_i)^{\mathrm{T}}\\ -\mathrm{E}\left[\mathcal{G}(\mathbf{t});\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right]&\mathrm{E}\left[\mathcal{G}(\mathbf{t});\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right]^{\mathrm{T}} \end{aligned} \quad (29)$$

where $\{\mathbf{t}_i\}$ represents a model parameter sample drawn from the predictive distribution over the trained network output $p\left(\mathbf{t}|\mathbf{x}_t,\mathcal{D}_N,\mathcal{A}^*_{N_H,c_1,c_2}\right)$ with optimized Bayesian neural network architecture based on the updated FE model. From Eqs. (28) and (29), the statistical properties of predicted modal parameters from updated model is achieved, which can be utilized for assessing the validity of updated model.

(29)

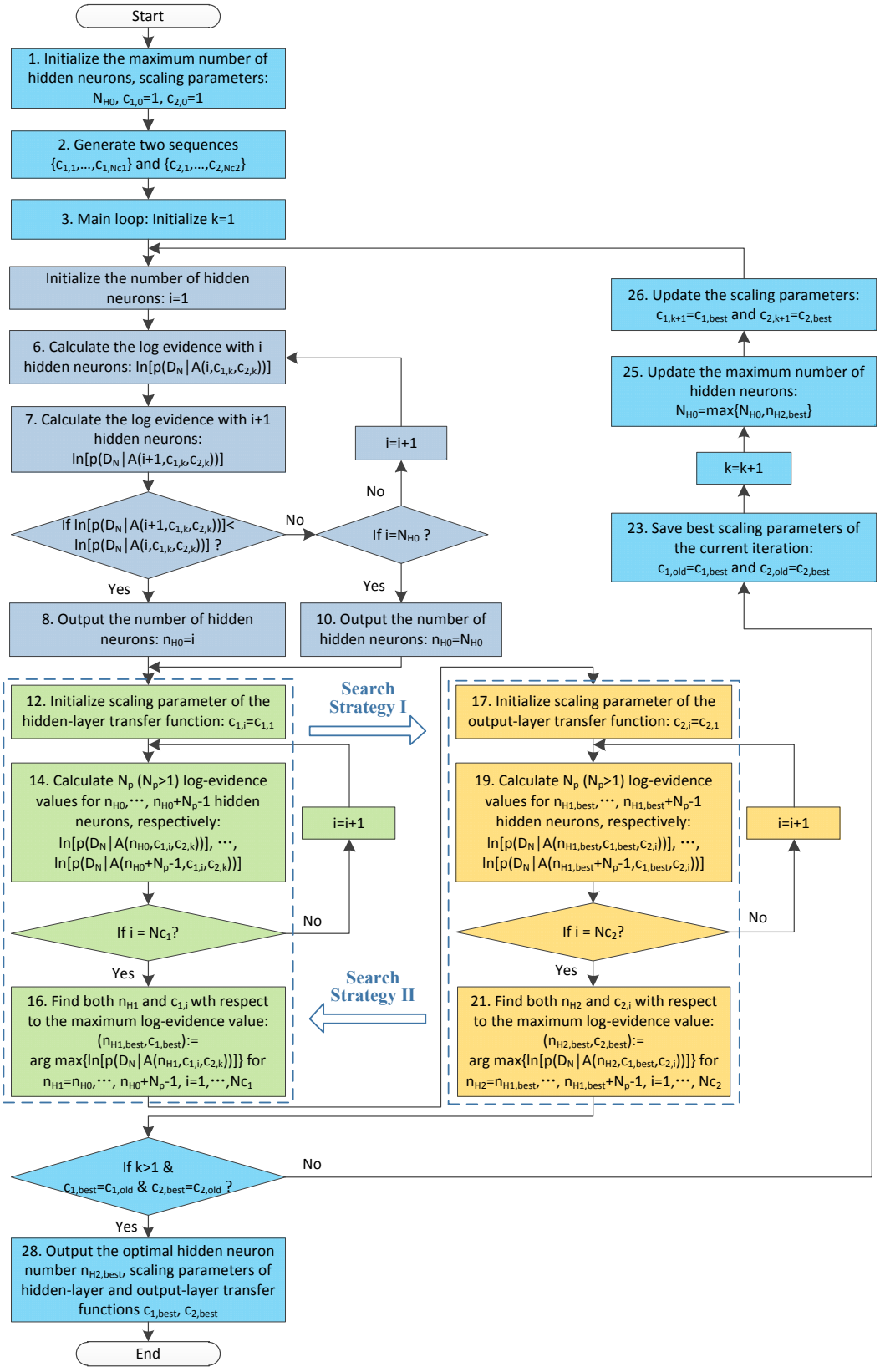

Fig. 2. Flowchart of the proposed architecture design algorithm for the Bayesian neural network.

3 MODEL UPDATING OF A PEDESTRIAN BRIDGE

Fig. 3. Xima Road pedestrian steel bridge: (a) Baidu Map© view shows the pedestrian bridge and its vicinity; (b) elevation view; (c) connection detail between strut, top chord and diagonal chord; (d) longitudinal view.

In this section, the proposed methodology is verified through model updating of a 39-m-long pedestrian steel bridge, located at Xima Road, Jiang'an District, Wuhan, China, as shown in Figure 3. Except for the cross-section of the strut being the circular tube, other non-bridge-deck members, including the upper chords, bottom chords, and diagonal chords, are all in the form of hollow rectangles in cross section. Table 1 lists the geometrical and material properties of the pedestrian bridge, most of which are obtained from the design drawings. By utilizing these physical parameters provided in this table, the initial FE model of the pedestrian bridge is established. Figure 4 shows the structural FE model with a total number of 68 elements and 30 nodes. Each chord and strut is discretized into one single beam element while the bridge deck is modeled as plate elements. The bridge deck is composed of two layers of steel panels with a thickness of 12 mm located at both the top and bottom, respectively. For the established FE model, it is much more convenient to treat the bridge deck as plate elements with uniform thickness regardless of its hollowness in the middle layer, and thus the material properties of the bridge deck is involved in the updating parameters later for compensating the influence of this approximation.

Table 1
Geometrical and material properties employed for modeling the pedestrian bridge.

Parameter descriptions	*Initial values*
Bridge span	39 m
Bridge width	4.3 m
Bridge height	3.4 m
Young's modulus of steel members	2.06×10^{11} N/m^2
Mass density of steel members	7.85×10^{3} kg/m^3
Poisson's ratio of steel members	0.3
Outer dimension of top chord section (Rectangular hollow)	0.4×0.3 m
Thickness of top chord section	0.012 m
Outer dimension of bottom chord section (Rectangular hollow)	0.4×0.3 m
Thickness of bottom chord section	0.012 m
Outer dimension of diagonal chord (Rectangular hollow)	0.25×0.3 m
Thickness of diagonal chord	0.012 m
Outer radius of strut section (Circular tube)	0.09 m
Thickness of strut section	0.012 m
Thickness of bridge deck	0.2 m
Young's modulus of bridge deck	2.06×10^{11} N/m^2
Mass density of bridge deck	7.85×10^{3} kg/m^3
Poisson's ratio of bridge deck	0.3

Based on the initial FE model shown in Figure 4, the first four natural frequencies and mode shapes of the pedestrian bridge model are obtained from the modal analysis process, including the 1st and 2nd vertical modes and the 1st and 2nd torsional modes as shown in Figure 5. It should be noted that the lateral modes is not involved in the model updating procedure, which is intended to match the sensor layout (see yellow arrows in Figure 4) in the field testing.

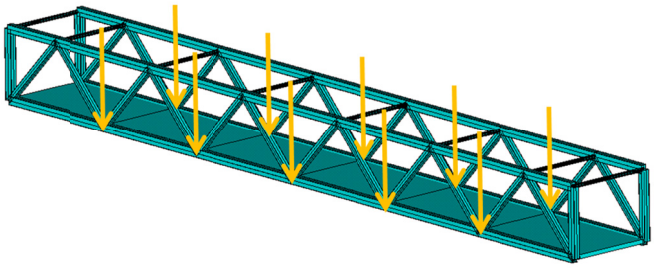

Fig. 4. FE model of the pedestrian steel bridge (yellow arrows: sensor layout in the field testing).

As shown in Figure 4, there are a total of ten sensors involved in the field testing, which are placed at the intersections of the diagonal and lower chords along the both sides of the pedestrian bridge except for the supporting points at both end of the bridge. These uniaxial sensors are oriented to measure the vertical vibration of the bridge structure, which is consistent with the fact that only vertical and torsional modes obtained from the FE model are employed for comparison purpose.

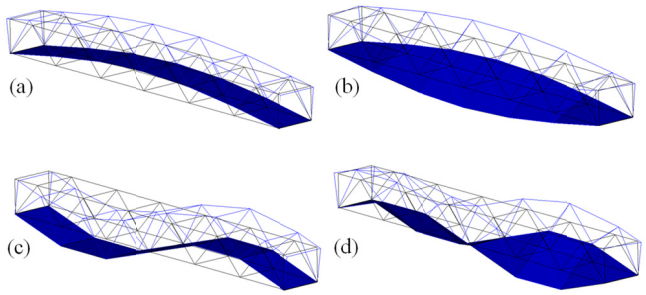

Fig. 5. Calculated mode shapes from the initial FE model of the pedestrian bridge: (a) mode 1 (2.90 Hz); (b) mode 2 (5.53 Hz); (c) mode 3 (8.77 Hz); (d) mode 4 (10.61 Hz).

Fig. 6. Experiment configuration: (a) signal conditioning box and laptop implemented with data acquisition software; (b) sensor layout on the bridge deck; (c) sensor connected with shielded cable; (d) sensor fixed close to bottom chord joint through magnetic base.

The experiment equipment and associated configuration for the field testing of the pedestrian bridge are shown in Figure 6, where the measurement is performed in the ambient environment. The 16-channeled (Type: KT6016-IEPE) signal conditioning box and laptop implemented with data acquisition software (Type: YE7600) are shown in Figure 6(a), and the layout of sensors configured on the bridge deck is provided in Figure 6(b). The ambient vibration response are acquired synchronously through ten piezoelectric accelerometers (Type: KT11000L) with sensitivity around 10V/g attached close to the intersection of diagonal and bottom chords through magnetic base, and the responses are then transferred to the conditioning box through shielded cables (referring to Figures 6(c) and (d)). The sampling frequency is set to 100 Hz, and the dynamic responses are measured with duration about 30 minutes. The modal parameters including natural frequencies and mode shapes are identified by the frequency domain decomposition (FDD) method.

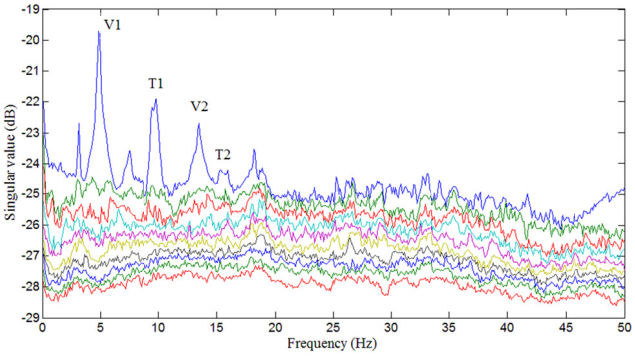

Fig. 7. Singular value plot of the pedestrian bridge.

The singular-value spectrum calculated using the measured time-domain responses from all the ten channels is shown in Figure 7. The labeled spectral peaks in this figure indicate the four interested modes. The labels 'V1', 'V2', 'T1' and 'T2' indicated in this figure stand for the 1st and 2nd vertical and the 1st and 2nd torsional modes of the pedestrian bridge, respectively.

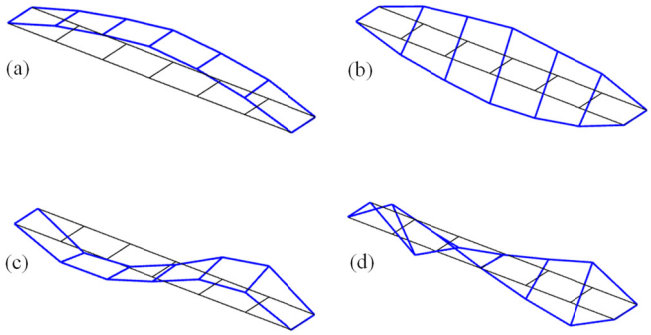

Fig. 8. Identified mode shapes from the field testing: (a) mode 1 ('V1', 4.86 Hz); (b) mode 2 ('T1', 9.69 Hz); (c) mode 3 ('V2', 13.53 Hz); (d) mode 4 ('T2', 15.54 Hz).

Figure 8 shows the identified four mode shapes by the FDD approach related to the FE mode shapes provided in Figure 5. It is clear that the degree of agreement between the measured mode shapes and their calculated counterparts is quite good, and the values of Modal Assurance Criterion (MAC) for each mode are 0.9962, 0.9915, 0.9526 and 0.8543, respectively, which are all close to 1. However, by comparing the values of natural frequencies provided in Figures 5 and 8, the differences between the measured and FE calculated natural frequencies are more obvious. This

implies that it is necessary to refine the initial structural model to ensure a good match between the calculated natural frequencies and the measured values.

Table 2
Definition of FE model parameters to be updated.

Parameter names & descriptions	*Initial values*
E_D (Young's modulus of bridge deck)	2.06×10^{11} N/m^2
m_D (Mass density of bridge deck)	7.85×10^{3} kg/m^3
μ_S (Poisson's ratio of steel members)	0.3
μ_D (Poisson's ratio of bridge deck)	0.3
m_S (Mass density of steel members)	7.85×10^{3} kg/m^3
E_S (Young's modulus of steel members)	2.06×10^{11} N/m^2

It is noted that the geometric parameters of the pedestrian steel bridge are taken from the design drawings and are therefore relatively reliable. The updating parameters for this pedestrian bridge are thus taken from the material parameters, which are believed to be more uncertainty than the geometric parameters. It is mentioned previously that the material of bridge deck is assumed to be homogeneous along the thickness direction in the initial FE model regardless of the fact that middle layer of bridge deck is hollow. Thus, the Young's modulus and mass density of the bridge deck material are more uncertain and are firstly considered as the model parameters for updating. In addition, by noting the fact that the Poisson's ratio also have a non-negligible effect on the structural dynamics, thus the Poisson's ratio of steel members and bridge deck material are also involved for updating. Table 2 shows the list of candidate model parameters selected to be updated for the pedestrian bridge by the proposed methodology.

Table 3
FE model updating results for the pedestrian bridge.

Scaling factors	*Initial values*	*Updated values*	*Standard deviations*
θ_1 (scaling for E_D)	1	0.7175	0.0693
θ_2 (scaling for m_D)	1	0.3128	0.0519
θ_3 (scaling for μ_S)	1	0.9203	0.0258
θ_4 (scaling for μ_D)	1	1.1049	0.0595
θ_5 (scaling for m_S)	1	0.9851	0.0191
θ_6 (scaling for E_S)	1	1.0242	0.0139

Instead of directly refining the physical parameter values as listed in Table 2, dimensionless scaling factors corresponding to these parameters as defined in Table 3 are more convenient to be updated since different orders of parameter magnitude will usually lead to numerical difficulties. θ_1 to θ_6 listed in Table 3 represent the six non-dimensional scaling factors to be actually updated. In addition, for structural model updating, one should be aware that not all candidate parameters for updating possess the same degree of modeling error. This implies that some parameters should be updated more intensely than others, which is conveniently done by setting the range of values for each individual parameter. For the pedestrian bridge investigated in this study, as stated before, parameters related to material properties, i.e., the Young's modulus, Poisson's ratio, and mass density of the bridge deck should be updated more intensely than others, and thus assigned with a relatively larger parameter range. In contrast, material parameters of the steel members are believed to be relatively more certain, and are specified to be in a smaller range of values. Following this thought, for updating the pedestrian bridge model, the lower and upper bounds of the scaling factors shown in Table 3 are defined to be [0.1 0.1 0.8 0.5 0.9 0.9] and [1.2 1.2 1.2 **1**.5 1.1 1.1], respectively.

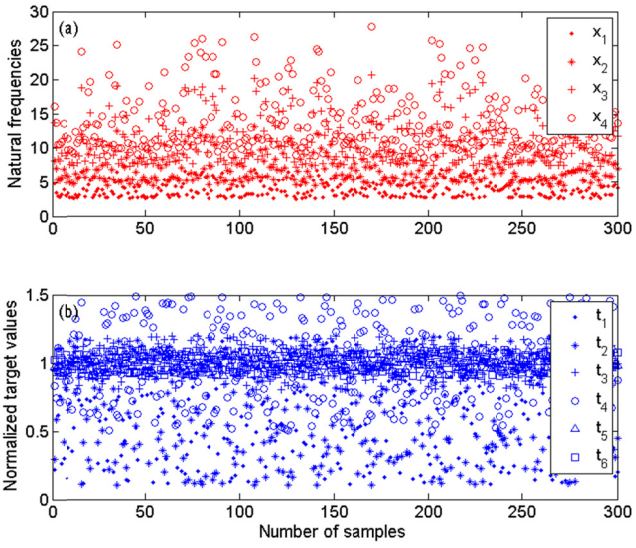

Fig. 9. Noise-free training samples generated: (a) training input samples and (b) training output samples.

In this study, the four natural frequencies as shown in Figures 5 and 8 are utilized as input variables feed to the neural network. Since one generally has no idea about specific values of scaling factors to be adjusted before updating the model, the uniform distribution should be more reasonably assumed. Figure 9 shows the generated input-output training data $\mathcal{D}_M^{(0)}$ with a sample size of M=300 in the absence of sample noise and within the prescribed range of parameter space. It is clear that the higher-order frequency samples have greater variability than the lower-order ones. Based on $\mathcal{D}_M^{(0)}$, the full set of training data $\mathcal{D}_N$ is obtained with Eq. (7) by combining the training data generated with three levels of noise (1%, 3% and 5%) imposed on the noise-free training input samples (see Figure 9(a)) respectively. The overall size of training data $\mathcal{D}_N$ is thus four times that of $\mathcal{D}_M^{(0)}$.

Providing the training data $\mathcal{D}_N$, the most probable class of architectures $\mathcal{A}_{N_H,c_1,c_2}^*$ for the Bayesian neural network can be determined by maximizing the log-evidence in Eq. (20) before applying the trained network to the FE model updating process. Prior to utilize the proposed network design algorithm, some control parameters should be

determined. In this study, the value of maximum number of hidden neurons $N_{H,\max}$ is set to be 15. In addition, the function $\tan(\pi\gamma/180)$ with angle value γ ranging from 5 to 85 with a step length of 5 is utilized to generate the sequence of discrete values assigned to the scaling parameters c_1 and c_2 as shown in Figure 1. Although this choice is not unique, it can ensure the slope of the resulting transfer-function curves change more evenly and is thus expected to be more representative in the defined parameter space spanned by c_1 and c_2. Thus, the total number of sample points for the two scaling parameters is $N_{c_1} = N_{c_2} = 17$. Although other ranges for these three design parameters can be taken, the proposed algorithm does not aim to find a unique global maximum by searching the entire parameter space, but rather to obtain the 'best' result within a prescribed parameter range efficiently.

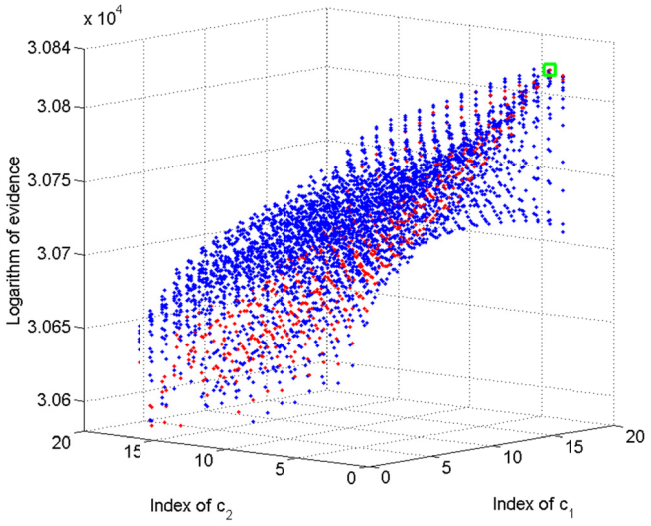

Fig. 10. Scatter plot of exhaustive search history (blue point maker: exhaustive search points; green square marker: maximum value; red point marker: exhaustive search points corresponding to the optimal number of hidden neurons).

Prior to apply the proposed design algorithm, for verification purpose, the exhaustive search method is first employed over the above-defined parameter space to find the optimal solution. The exhaustive search history is shown in Figure 10, and each discrete point corresponds to a parameter combination $\{N_H, c_1, c_2\}$. The total number of log-evidence evaluations of Eq. (20), i.e., the number of discrete points, is $N_{H,\max}N_{c_1}N_{c_2}$ =15×17×17=4335. The optimal point (the maximum log-evidence value is 30825.9286) found by the exhaustive search is marked by a green square to indicate the most probable class of architectures $\mathcal{A}^*_{N_H,c_1,c_2}$ in the present study. The identified optimal hidden neuron number $N_{H,\text{best}}$ is 12, and the simultaneously optimized scaling parameters $c_{1,\text{best}}$ and $c_{2,\text{best}}$ are tan(80π/180)=5.6713 and tan(5π/180)=0.0875, respectively. In addition, the red points indicate the exhaustive results corresponding to the optimal hidden neuron number 12, where the importance and necessity of optimizing both transfer-function parameters is clearly interpreted from wide distribution of these discrete points in the scaling parameter space.

It should be noted that Lam and Ng (2008) proposed an algorithm for selecting the hidden-layer transfer functions (tansig and satlins) and the hidden neuron number, which is recently employed by Yin and Zhu (2018) to the Bayesian neural network. This algorithm simply traverses tansig and satlins in outer loop and each hidden neuron number in inner loop. This would be very time consuming since for each transfer function specified by the outer loop, each inner loop is required to be executed until the optimal N_H is found. Applying this algorithm to the present study, it takes about $N_{H,\max}N_{c_1}N_{c_2}$ log-evidence evaluations, and the performance is very similar to the exhaustive method as shown in Figure 10. In contrast, the reported algorithm in this paper requires only about $N_k[N_H + N_p(N_{c_1} + N_{c_2})]$ log-evidence evaluations, where the number of consecutive log-evidence values N_p is set to be 2 in this study, and the number of main-loop iterations N_k required for achieving convergence is also generally a very small number. Obviously, the proposed tailor-made algorithm is expected to be significantly more efficient by comparing with that used in Lam and Ng (2008) and Yin and Zhu (2018) especially for large numbers of $N_{H,\max}$, N_{c_1} and N_{c_2}.

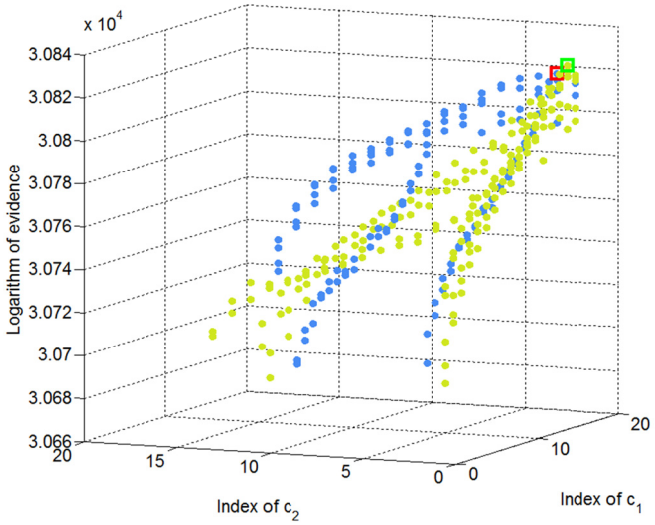

Fig. 11. Scatter plot of the iteration history by applying the proposed design algorithm (blue point and red square maker: history and maximum of search strategy I; green point and green marker: history and maximum of search strategy II).

By adopting the proposed design algorithm, the optimized number of hidden neurons together with the two scaling parameters of hidden- and output-layer transfer functions is simultaneously identified, where the initial

estimation of the temporary maximum hidden neuron number N_{H0} is 10. Figure 11 shows the search history of the proposed algorithm with search strategies I and II in terms of a scatter plot, where the corresponding optimal points found are marked by red and green squares, respectively. It's apparent that two 'best' points found by the two strategies are actually very close to each other, while the latter is a bit better. More importantly, the 'best' point marked by green square completely coincides with the previous optimal point found by the exhaustive search method in Figure 10. It is apparent that the developed algorithm efficiently locates the optimal value within the pre-defined parameter space by evaluating the objective function with only a small number of times (about 240 with $N_k = 3$) without searching the entire parameter space with 4335 times. This clearly indicates the significant effectiveness of the proposed algorithm on optimizing the architecture of the Bayesian neural network.

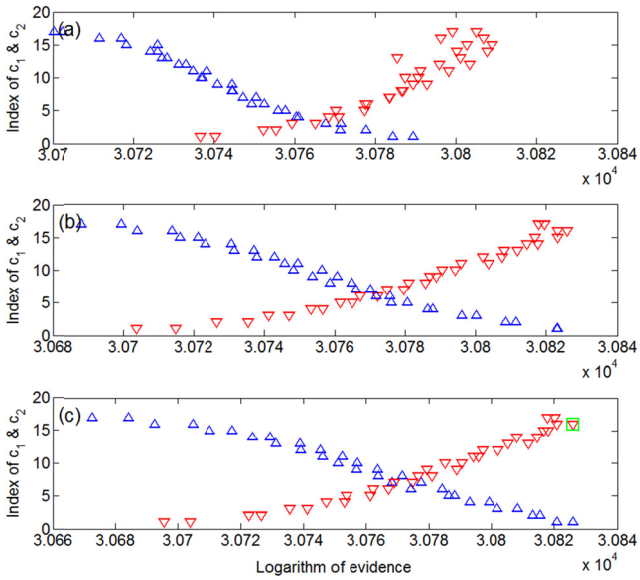

Fig. 12. Iteration history of the proposed design algorithm: (a) to (c) denote results of the 1st, 2nd and 3rd main loops (red downward -pointing triangle: c_1; blue upward-pointing triangle: c_2; green square marker: maximum point).

To further interpret the optimization process of the proposed algorithm, the iteration history of search strategy II in terms of two transfer-function scaling parameters and the number of hidden neurons are shown in Figures 12 and 13, respectively, where the three main loops are provided separately in both two figures for the sake of clarity. Specifically, in Figure 12, the blue points represent the iteration history for optimizing the scaling parameter c_2 of the output-layer transfer function, while the red points stand for the subsequent iteration process for optimizing the hidden-layer scaling parameter c_1. It is found from the results of first main loop as presented in Figure 12(a) that, in the process of optimizing the parameter c_2, the log-evidence value gradually increases with the decrease of c_2, whereas the observation is opposite when optimizing c_1. In addition, the maximum log evidence obtained by the optimization of c_1 is better than that of c_2. This implies that, through the optimization of c_2 and c_1 in turn, the current optimal value of the objective function is improved. Furthermore, by comparing the optimization history for each main loop, it is obvious that the maximum log-evidence value achieved by each round of main loop is improved relative to its previous round, which reflects the validity of the proposed algorithm. Also, as shown in Figure 12(c), the 'best' point obtained in the last round of main loop (see the green square marker) is exactly the same to the optimal value found by the exhaustive search approach as illustrated in Figure 10.

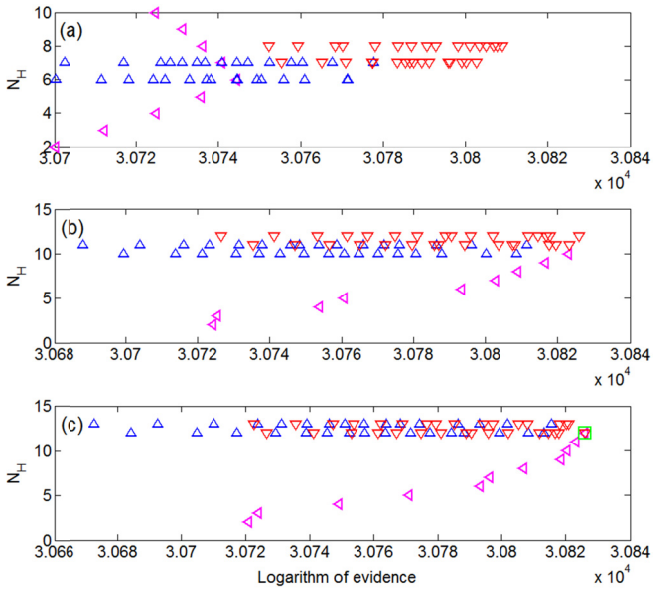

Fig. 13. Iteration history of the proposed algorithm for the number of hidden neurons: (a) to (c) denote the results of 1st, 2nd, and 3rd main loops (pink left-pointing triangle: N_H; red downward-pointing triangle: c_1; blue upward-pointing triangle: c_2; green square marker: maximum point).

Figure 13 shows the iteration history in terms of number of hidden neurons N_H with respect to each main loop for demonstration purpose, and it attempts to reflect the optimization process of the proposed algorithm from another perspective. It is clearly seen that, in the first inner loop within each main loop for optimizing N_H (as labeled by pink color triangle), the log-evidence value either increases first, then decreases or gradually increases with the increase of iterative step, indicating the importance of the number of hidden neurons to be optimized. The 'best' N_H value achieved in this inner loop is then involved in optimization process of subsequent inner loops for sequentially optimizing the combinations $\{N_H, c_2\}$ and $\{N_H, c_1\}$ with N_p consecutive log-evidence values. In

addition, it is also very clear that the alternate optimization of scaling parameters c_2 and c_1 leads to a significant improvement of the log-evidence value, which fully demonstrates the necessity of simultaneously optimizing the hidden- and output-layer transfer functions. Moreover, it should be pointed out that, although the upper limit of N_H ($N_{H,\max}$) is set to 15 for the exhaustive method in Figure 10, the maximum N_H actually handled by proposed design algorithm is only 13 as seen from Figure 13(c), indicating the efficiency of the proposed search strategy.

Based on the trained Bayesian neural network with optimized architecture and the four natural frequencies obtained from field testing (see Figure 8), the initial FE model of the pedestrian bridge is updated by adjusting the six scaling factors in Table 3. In the framework of proposed probabilistic model updating procedure, the distribution of these scaling factors is conveniently achieved from the predictive distribution over the trained network output with Gaussian approximation as provided in Eq. (26).

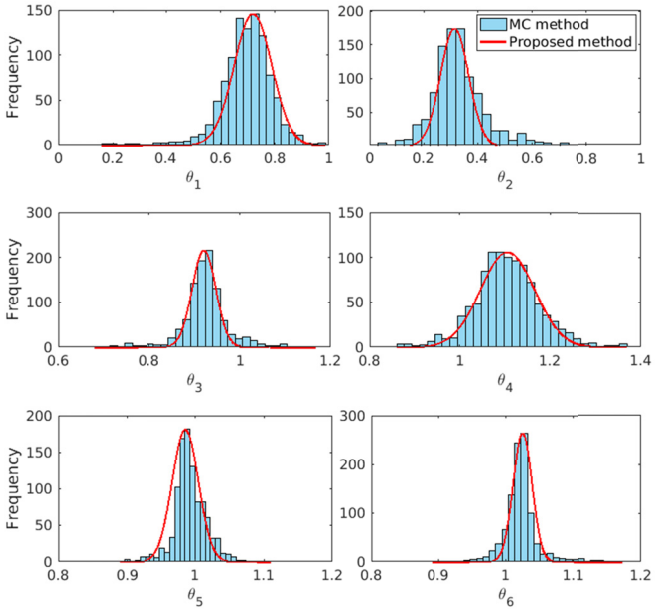

Fig. 14. Comparison of marginal distributions of updated model parameters obtained by Monte Carlo method and proposed method with Gaussian approximation.

The marginal predictive distribution of the six uncertain model parameters are obtained by the proposed method and shown in Figure 14 as red solid lines, while the histogram plot by employing the Monte Carlo technique in Eq. (23) is also provided herein for comparison. It is clear that, for all scaling factors, matching between the proposed Gaussian approximation of the marginal predictive distribution and corresponding Monte Carlo results is reasonably good. It implies that the marginal predictive distribution of each updating parameter can indeed be approximated by Gaussian. In addition, one can easily see from this figure that the uncertainty for the identified material parameters of bridge deck is obviously greater than others. This is consistent with actual situation since the inhomogeneous property of bridge deck along the thickness direction is not considered in initial FE model, and the corresponding material properties are believed to be more uncertain. Moreover, the most probable values and associated standard deviations of model updating results are also calculated and given in Table 3 for clarity purpose. It is clear from this table that the amount of adjustment for the Young's modulus and mass density of bridge deck is much larger than other modeling parameters, and the significant reduction of mass density actually reflects the large hollow space existing inside the bridge deck. Also, since the material properties of steel members are believed to be more reliable in the initial FE model as above mentioned, they thus get less adjustment with relatively low uncertainty.

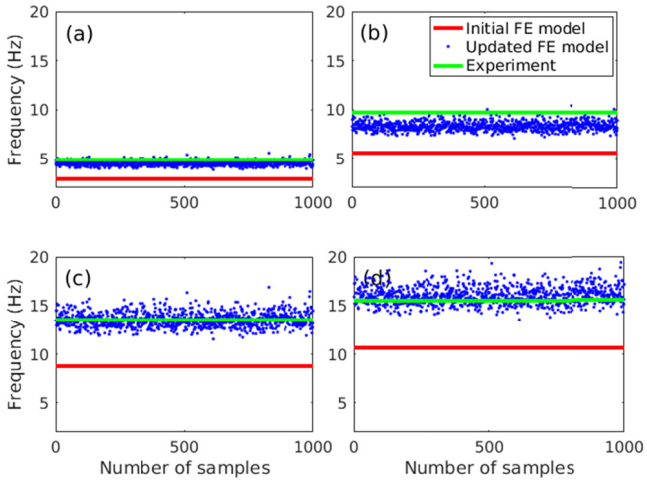

Fig. 15. Distribution of natural frequencies predicted from the updated FE model: (a) to (d) modes 1 to 4.

Table 4
Measured and model-predicted natural frequencies with corresponding standard deviations (Hz).

Mode No.	*Experiment*	*Initial FE model*	*Updated FE model*	*Standard deviations*
1	4.86	2.90	4.48	0.24
2	9.69	5.53	8.34	0.46
3	13.53	8.77	13.56	0.75
4	15.54	10.61	16.02	0.89

Figure 15 shows the samples of predicted natural frequencies through the updated FE model by utilizing the Monte Carlo technique to sample the predictive distribution in Eq. (27), where results measured from the field testing and those predicted from the initial FE model are also presented for comparison. It is clear that, by employing the proposed model updating procedure, the improvement of frequency prediction accuracy for the refined FE model is very obvious. It is also intuitively seen that the higher the order of mode, the greater the uncertainty of the natural frequency prediction results. In addition, the statistical

properties of predicted natural frequencies from the updated FE model are obtained by utilizing Eqs. (28) and (29) based on the samples in Figure 15. The quantitative results are shown in Table 4 together with the measured and predicted natural frequencies from the initial FE model. It is clear from this table that the refined FE model is apparently more precise than its original form. Furthermore, it is also verified from this table that the standard deviation increases with the increase of mode order, indicating that the higher order modes are more uncertain than the lower ones, which coincides with the general knowledge. More importantly, the standard deviation values shown in Table 4 can be further utilized for constructing error bars to estimate confidence intervals of refined FE models, which is crucial for evaluating the quality of updated model in practice.

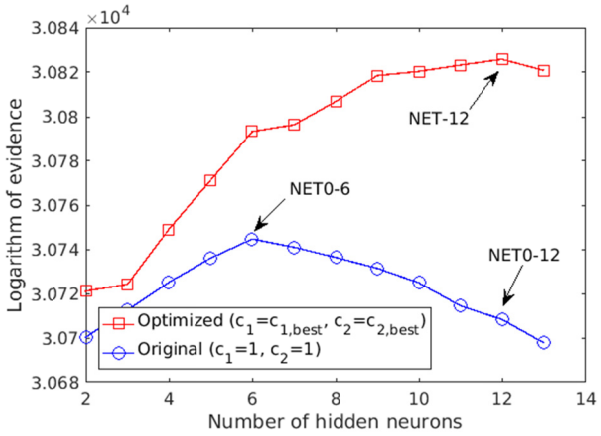

Fig. 16. Log evidence with the increase of N_H for the Bayesian neural networks with optimized and original c_1,c_2.

To investigate the performance of Bayesian neural network with optimized architectures obtained through the proposed design algorithm, Figure 16 shows the log-evidence values of network architecture with optimized ($c_1 = c_{1,\text{best}}$, $c_2 = c_{2,\text{best}}$) and original ($c_1 = 1, c_2 = 1$) scaling parameters c_1, c_2 from the perspective of the change of hidden neuron number N_H. It is clear that, for any of the two network architectures, the log-evidence value increases first and then decreases with the increase of N_H, and this phenomenon is consistent with Lam and Ng (2008). The peak points of both curves indicate the optimal value of N_H with respect to these two network architectures. Here are three networks compared, denoted by NET-12, NET0-6, and NET0-12, respectively. As labeled in this figure, NET-12 represents the optimal network architecture with $c_1 = c_{1,\text{best}}$, $c_2 = c_{2,\text{best}}$, and $N_{H,\text{best}} = 12$. While NET0-6 and NET0-12 stand for the original network architecture ($c_1 = 1, c_2 = 1$) with optimized hidden neuron number $N_H = 6$ and specified number $N_H = 12$, respectively. The first two networks are intended to compare the neural network possessing simultaneously optimized c_1, c_2 and N_H with the original network by only optimizing N_H. The last one is employed to compare the effects of optimizing both c_1 and c_2 in terms of same number of hidden neurons N_H.

It is noted here that the neural network training process is susceptible to initialization values assigned to the weight parameters. The performance of trained networks with different initial weights may not be completely identical, so the results presented above are obtained by averaging multiple repetitions to reduce the impact of different initial weights on the difference in trained networks. In order to facilitate the comparison of the three networks as shown in Figure 16, the results obtained from these specific networks with corresponding architectures under various initial weights are provided in Figures 17 and 18 from a statistical point of view.

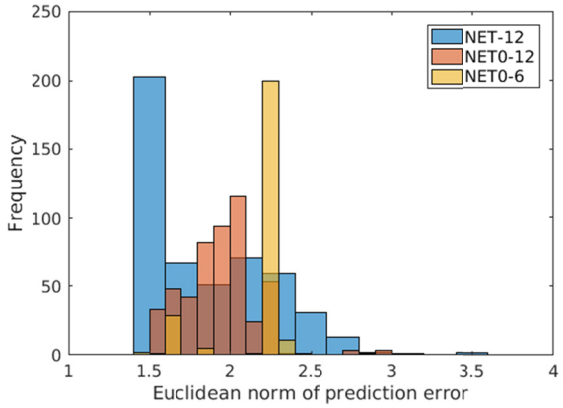

Fig. 17. Histogram of Euclidean norm of prediction error vector in natural frequencies (Hz) for networks with optimized and original c_1, c_2.

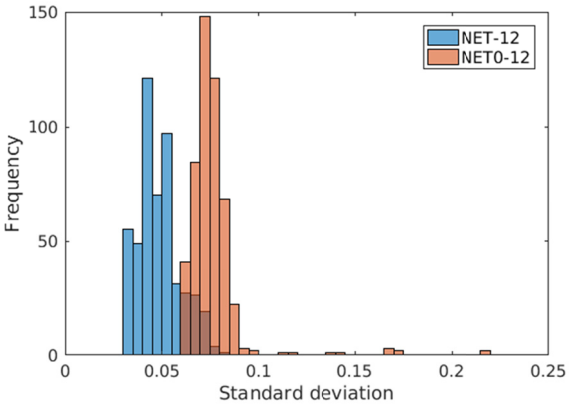

Fig. 18. Histogram of mean value of standard deviations of all identified model parameters for networks with optimized and original c_1, c_2.

Figure 17 shows the histogram plot of Euclidean norm of difference between the measured and predicted natural frequency vector from the updated FE model. It's clear that the matching of NET-12 as well as NET0-12 are obviously better than NET0-6. This is because the network complexity of the former two is relatively larger, which allows them to better fit the data relative to the simple one.

On the other hand, to investigate the effect of optimizing the two transfer-function scaling parameters, Figure 18 shows the histogram plot of mean value of standard deviations of all identified model updating parameters for NET-12 and NET0-12, which possess the same complexity in terms of same hidden neuron number. It is very obvious that the standard deviation values of NET-12 with optimized c_1 and c_2 are less than those of NET0-12 without optimization, implying that the uncertainty of updated model parameters for NET-12 is reduced. It is mainly due to the contribution of simultaneously optimized c_1 and c_2 to the uncertainty reduction of network weight parameters according to Eq. (26). This also reinforces the statement made above that the Ockham factor actually imposes penalty on the uncertainty of identified model parameters. Although not shown here, it is also found that the uncertainty of updated parameters of NET0-12 is larger than NET0-6. This is expected, as the increase of N_H or network architecture complexity leads to an increase in the Ockham factor, which is reflected in the amplified uncertainty of the updated model parameters. Thus, by optimizing the hidden neuron number N_H together with the hidden- and output-layer scaling parameters c_1 and c_2, the accuracy and uncertainty of the updated structural model can be improved, respectively.

4 CONCLUSIONS

For probabilistic FE model updating based on dynamic measurements, this paper develops an efficient and tailor-made algorithm utilized for designing the architecture of Bayesian neural network through simultaneous determination of the proper hidden neuron number and the specific forms of parameterized hidden- and output-layer transfer functions. The validity and efficiency of the proposed methodology is fully demonstrated through the FE model updating conducted for a pedestrian bridge with the field testing data.

The obtained results clearly reveal that the proposed design algorithm allows one to achieve the optimized architecture of Bayesian neural network with appropriate hidden neuron number and suitable forms of transfer functions in both hidden and output layers simultaneously, through a very effective and mathematically rigorous way. It is emphasized herein that this algorithm is far more effective than that used in the authors' recent publication (Yin and Zhu, 2018), which essentially follows an exhaustive search strategy. It is also noted that the proposed method is not intended to find a unique global maximum by searching the entire parameter space, but rather to locate the 'optimal' result within a specified parameter range efficiently. By applying the optimized network achieved through the proposed algorithm to the structural model updating, the accuracy as well as uncertainty of the updated structural model is found to be both improved. In addition, it is clear from the results of predicted natural frequencies based on the updated model that the refined FE model is more precise than its initial form. One can further find out that the higher the mode order, the larger the associated uncertainty. This coincides with the common knowledge that the high-order modes are generally more uncertain than the low-order ones, indicating the rationality of the model updating results obtained by the proposed methodology. Furthermore, it is shown that the obtained standard deviations of predicted modal parameters can be further utilized to predict error bars that quantify the confidence intervals for the updated structural models, which provides a key reference point for evaluating the quality of the refined FE model. Also, the uncertainty of the predicted modal parameters is especially critical for structural health monitoring and damage detection based on FE model updating. This is because excessive prediction uncertainty would overwhelm the changes of modal parameters induced by the actual damage and significantly reduce the reliability of damage detection results. Moreover, it should be pointed out that although this paper mainly concentrates on the neural network with a single hidden layer, in principle, the proposed design algorithm should be extended to the neural networks with multiple hidden layers, such as the deep neural networks. This is deserved to be further investigated in the future.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support provided by the National Natural Science Foundation of China (Grants No. 51778506, 51838006), and a scholarship from the China Scholarship Council (File No. 201806275091) while the first author visiting the Center for Engineering Dynamics and Institute for Risk and Uncertainty in the University of Liverpool. Our special thanks are due to Professor Au SK, University of Liverpool, for many useful discussions. Finally, we would like to thank the Editor and the anonymous reviewers for their constructive comments and valuable suggestions to improve the quality of the article.

REFERENCES

Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M. & Inmand, D. J. (2017), Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks, *Journal of Sound and Vibration*, **388**(3), 154–70.

Adeli, H. & Cheng, N.T. (1993), Integrated genetic algorithm for optimization of space structures, *Journal of Aerospace Engineering, ASCE,* **6**(4), 315–28.

Adeli, H. (2001), Neural networks in civil engineering: 1989−2000, *Computer-Aided Civil and Infrastructure Engineering*, **16**(2), 126–42.

Adeli, H. & Jiang, X. (2006), Dynamic fuzzy wavelet neural network model for structural system identification, *Journal of Structural Engineering, ASCE*, **132**(1), 102–11.

Adeli, H. & Jiang, X. (2009), Intelligent infrastructure – neural networks, wavelets, and chaos theory for intelligent transportation systems and smart structures, CRC Press, Taylor & Francis, Boca Raton, Florida.

Arangio, S. & Beck, J. L. (2012), Bayesian neural networks for bridge integrity assessment, *Structural Control and Health Monitoring*, **19**(1), 3–21.

Astroza, R., Nguyen, L.T. & Nestorovic, T. (2016), Finite element model updating using simulated annealing hybridized with unscented Kalman filter, *Computers & Structures*, **177**, 176–91.

Barber, D. (2002), Bayesian methods for supervised neural networks, Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge.

Beck, J.L. & Yuen, K.V. (2004), Model selection using response measurement: A Bayesian probabilistic approach, *Journal of Engineering Mechanics, ASCE*, **130**(2), 192–203.

Bishop, C. M. (2006), Pattern recognition and machine learning, Springer: Berlin.

Brownjohn, J. M. W., Moyo, P., Omenzetter, P. & Lu, Y. (2003), Assessment of highway bridge upgrading by dynamic testing and finite-element model updating, *Journal of Bridge Engineering*, **8**(3), 162–72.

Boulkaibet, I., Mthembu, L., De Lima Neto, F. & Marwala, T. (2015), Finite element model updating using fish school search and volitive particle swarm optimization, *Integrated Computer-Aided Engineering*, **22**(4), 361–76.

Buntine, W. & Weigend, A. (1991), Bayesian back-propagation, *Complex Systems*, **5**, 603–43.

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

Chang, C. M., Lin, T. K. & Chang, C. W. (2018), Applications of neural network models for structural health monitoring based on derived modal properties, *Measurement*, **129**, 457–70.

Cybenko, G. (1989), Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, **2**(4), 303–14.

Duch, W. & Jankowski, N. (1997), New neural transfer functions, *Applied Mathematics and Computer Science*, **7**(3), 639–58.

Friswell, M. I. & Mottershead, J. E. (1995), Finite element model updating in structural dynamics, Kluwer Academic Press, Dordrecht.

Gao, Y. Q. & Mosalam, K. M. (2018), Deep transfer learning for image-based structural damage recognition, *Computer-Aided Civil and Infrastructure Engineering*, **33**(9), 748–68.

Grande, Z., Castillo, E., Mora, E. & Lo, H. K. (2017), Highway and Road Probabilistic Safety Assessment Based on Bayesian Network Models, *Computer-Aided Civil and Infrastructure Engineering*, **32**(5), 379–96.

Hagan, M. T., Demuth, H. B., Beale, M. H. (1996), Neural Network Design, PWS Publishing, Boston, MA.

Hakim, S. J. S., Razak H. A. & Ravanfar S. A. (2015), Fault diagnosis on beam-like structures from modal parameters using artificial neural networks, *Measurement*, **76**, 45–61.

Iruansi, O., Guadagnini, M., Pilakoutas, K. & Neocleous, K. (2012), Predicting the shear resistance of RC beams without shear reinforcement using a Bayesian neural network, *International Journal of Reliability and Safety*, **6**(1–3), 82–109.

Jaish, B. & Ren, W. X. (2007), Finite element model updating based on eigenvalue and strain energy residuals using multiobjective optimisation technique, *Mechanical Systems and Signal Processing*, **21**(5), 2295–317.

Jensen, H. A., Millas, E., Kusanovic, D. & Papadimitriou, C. (2014), Model-reduction techniques for Bayesian finite element model updating using dynamic response data, *Computer Methods In Applied Mechanics and Engineering*, **279**, 301–24.

Jiang, X. & Adeli, H. (2005), Dynamic wavelet neural network for nonlinear identification of highrise buildings, *Computer-Aided Civil and Infrastructure Engineering*, **20**(5), 316–30.

Katafygiotis, L. S. & Beck, J. L. (1998), Updating models and their uncertainties. II: Model identifiability, *Journal of Engineering Mechanics, ASCE*, **124**(4), 463–67.

Kocadağlı, O. & Aşıkgil, B. (2014), Nonlinear time series forecasting with Bayesian neural networks, *Expert Systems with Applications*, **41**(15), 6596–610.

Lam, H. F., Hu, Q. & Wong, M.T. (2014), The Bayesian methodology for the detection of railway ballast damage under a concrete sleeper, *Engineering Structures*, **81**, 289–301.

Lam, H. F. & Ng, C. T. (2008), The selection of pattern features for structural damage detection using an extended Bayesian ANN algorithm, *Engineering Structures*, **30**(10), 2762–70.

Lam, H. F., Yuen, K. V. & Beck, J. L. (2006), Structural health monitoring via measured Ritz vectors utilizing artificial neural networks, *Computer-Aided Civil and Infrastructure Engineering*, **21**(4), 232–41.

Lampinen, J. & Vethari, A. (2001), Bayesian approach for neural networks—review and case studies, *Neural Networks*, **14**(3), 257–74.

Law, S. S., Chan, T.H.T. & Wu, D. (2001), Super-element with semi-rigid joints in model updating, *Journal of Sound and Vibration*, **239**(1), 19–39.

Lee, H. K. H. (2004), Bayesian nonparametrics via neural networks, ASA-SIAM Series on Statistics and Applied Probability.

Levin, R. I. & Lieven, N. A. J. (1998), Dynamic finite element model updating using neural networks, *Journal of Sound and Vibration*, **210**(5), 593–607.

Lin, Y. Z., Nie, Z. H. & Ma, H. W. (2017), Structural damage detection with automatic feature-extraction through deep learning, *Computer-Aided Civil and Infrastructure Engineering*, **32**(12), 1025–46.

MacKay, D. J. C. (1992), A practical Bayesian framework for back-propagation networks, *Neural Computation*, **4**(3), 448–72.

MacKay, D. J. C. (1994), Bayesian methods for backpropagation networks, Model of Neural Networks III, Chapter 6. Springer: Berlin, 211–54.

Mottershead, J. E., Link, M. & Friswell, M. I. (2011), The sensitivity method in finite element model updating: A tutorial, *Mechanical Systems and Signal Processing*, **25**(7), 2275–96.

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

Neal, R. M. (1996). Bayesian learning for neural networks, Vol. 118, Lecture Notes in Statistics, Springer: Berlin.

Oh, B. K., Kim, D. & Park, H. S. (2017), Modal response-based visual system identification and model updating methods for building structures, *Computer-Aided Civil and Infrastructure Engineering*, **32**(1), 34–56.

Park, Y. S., Kim, S., Kim, N. & Lee, J. J. (2017), Finite element model updating considering boundary conditions using neural networks, *Engineering Structures*, **150**, 511–19.

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

Simoen, E., De Roeck, G. & Lombaert, G. (2015), Dealing with uncertainty in model updating for damage assessment: A review, *Mechanical Systems and Signal Processing*, **56-57**, 123–49.

Sirca Jr, G. F. & Adeli, H. (2012), System identification in structural engineering, *Scientia Iranica*, **19**(6), 1355–64.

Snoek, J., Larochelle, H., Adams, R.P. (2012), Practical Bayesian optimization of machine learning algorithms, In *Advances in neural information processing systems*, **25**, 2960–8.

Sohn, H., Farrar, C. R., Hernez, F. M., Czrnecki, J. J., Shunk, D. D., Stinemates, D. W., et al. (2004), A review of structural health monitoring literature: 1996–2001. Report no. LA-13976-MS. Los Alamos (NM): Los Alamos National Laboratory.

Stochino, F., Cazzani, A., Poppi, S., Turco, E. (2017), Sardinia radio telescope finite element model updating by means of photogrammetric measurements, *Mathematics and Mechanics of Solids*, **22**(4), 885–901.

Teughels, A., De Roeck, G. & Suykens, J. A. K. (2003), Global optimization by coupled local minimizers and its application to FE model updating, *Computers and Structures*, **81**(24–25), 2337–51.

Torres, W., Almazán, J. L., Sandoval, C. & Boroschek, R. (2017), Operational modal analysis and FE model updating of the Metropolitan Cathedral of Santiago, Chile, *Engineering Structures*, **143**, 169–88.

Wang, N. N., Zhao, Q. G., Li, S. Y., Zhao, X. F. & Zhao, P. (2018), Damage classification for masonry historic structures using convolutional neural networks based on still images, *Computer-Aided Civil and Infrastructure Engineering*, **33**(12), 1073–89.

Yang, X. C., Li, H., Yu, Y. T., Luo, X. C., Huang, T. & Yang, X. (2018), Automatic pixel-level crack detection and measurement using fully convolutional network, *Computer-Aided Civil and Infrastructure Engineering*, **33**(12), 1090–109.

Yin, T., Lam, H. F., Chow, H. M. & Zhu, H. P. (2009), Dynamic reduction-based structural damage detection of transmission tower utilizing ambient vibration data, *Engineering Structures*, **31**(9), 2009-19.

Yin, T., Jiang, Q. H. & Yuen, K. V. (2017), Vibration-based damage detection for structural connections using incomplete modal data by Bayesian approach and model reduction technique, *Engineering Structures*, **132**(1), 260–77.

Yin, T., Yuen, K. V., Lam, H. F. & Zhu, H. P. (2017), Entropy-based optimal sensor placement for model identification of periodic structures endowed with bolted joints, *Computer-Aided Civil and Infrastructure Engineering*, **32**(12), 1007–24.

Yin, T. & Zhu, H. P. (2018), Probabilistic damage detection of a steel truss bridge model by optimally designed Bayesian neural network, *Sensors*, **18**(10), 3371, doi:10.3390/s18103371.

Yin, T, Zhu, H. P. & Fu, S. J. (2019), Damage identification of periodically-supported structures following the Bayesian probabilistic approach, *International Journal of Structural Stability and Dynamics*, 19, 1940011.

Yu, L. & Yin, T. (2010), Damage identification in frame structures based on FE model updating, *Journal of Vibration and Acoustics*, **132**(5), 051007.

Yuen, K. V. (2010), Efficient model correction method with modal measurement, *Journal of Engineering Mechanics, ASCE*, **136**(1), 91–9.

Yuen, K.V. & Lam, H. F. (2006), On the complexity of artificial neural networks for smart structures monitoring, *Engineering Structures*, **28**(7), 977–84.

Yuen, K.V. & Mu, H.Q. (2015), Real-time system identification: an algorithm for simultaneous model class selection and parametric identification, *Computer-Aided Civil and Infrastructure Engineering*, **30**(10), 785–801.

Zhu, X. Q., Hao, H. & Peng, X. L. (2008), Dynamic assessment of underwater pipeline systems using

statistical model updating, *International Journal of Structural Stability and Dynamics*, **8**(2), 271–97.

APPENDIX

The submatrices of the Hessian in Eq. (16) is derived analytically by

$$\mathbf{H}_{11} = \nabla_1 \nabla_1^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_1 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_1 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{12} = \nabla_1 \nabla_2^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_1 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_2 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{13} = \nabla_1 \nabla_3^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla \mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}_n), c_1\right)\left(\mathbf{x}_n^{\mathrm{T}} \otimes \mathbf{I}_{N_H}\right)\right]^{\mathrm{T}} \otimes \mathbf{e}_n^{\mathrm{T}}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) + 2\sum_{n=1}^{N} \left[\nabla_1 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_3 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{14} = \nabla_1 \nabla_4^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_1 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_4 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{22} = \nabla_2 \nabla_2^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_2 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_2 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{23} = \nabla_2 \nabla_3^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla \mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}_n), c_1\right)\right]^{\mathrm{T}} \otimes \mathbf{e}_n^{\mathrm{T}}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) + 2\sum_{n=1}^{N} \left[\nabla_2 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_3 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{24} = \nabla_2 \nabla_4^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_2 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_4 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{33} = \nabla_3 \nabla_3^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_3 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_3 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{34} = \nabla_3 \nabla_4^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_3 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_4 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

$$\mathbf{H}_{44} = \nabla_4 \nabla_4^{\mathrm{T}} \mathcal{J}\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = 2\sum_{n=1}^{N} \left[\nabla_4 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)\right]^{\mathrm{T}} \nabla_4 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right)$$

and

$$\nabla_1 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}_n), c_2\right)\mathbf{W}^{(2)} \nabla \mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}_n), c_1\right)\left(\mathbf{x}_n^{\mathrm{T}} \otimes \mathbf{I}_{N_H}\right)$$

$$\nabla_2 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}_n), c_2\right)\mathbf{W}^{(2)} \nabla \mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}_n), c_1\right)$$

$$\nabla_3 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}_n), c_2\right)\left(\mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}_n), c_1\right)^{\mathrm{T}} \otimes \mathbf{I}_{N_O}\right)$$

$$\nabla_4 \mathbf{e}_n\left(\mathbf{w}; \mathcal{A}_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}_n), c_2\right)$$

where ∇_1, ∇_2, ∇_3 and ∇_4 denote the derivative with respect to weight vectors $\mathrm{vec}\left(\mathbf{W}^{(1)}\right)$, $\mathbf{b}^{(1)}$, $\mathrm{vec}\left(\mathbf{W}^{(2)}\right)$, and $\mathbf{b}^{(2)}$, respectively. The derivatives of transfer functions can be obtained analytically from Eqs. (1) and (2). $\otimes$ denotes the Kronecker products.

The submatrices of the Jacobian in Eq. (25) are given analytically as

$$\mathbf{G}_1 = \nabla_1 \mathbf{y}\left(\mathbf{x}, \mathbf{w}^*; \mathcal{A}^*_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}), c_{2,\mathrm{best}}\right)\mathbf{W}^{*(2)} \nabla \mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}), c_{1,\mathrm{best}}\right)\left(\mathbf{x}^{\mathrm{T}} \otimes \mathbf{I}_{N_H}\right)$$

$$\mathbf{G}_2 = \nabla_2 \mathbf{y}\left(\mathbf{x}, \mathbf{w}^*; \mathcal{A}^*_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}), c_{2,\mathrm{best}}\right)\mathbf{W}^{*(2)} \nabla \mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}), c_{1,\mathrm{best}}\right)$$

$$\mathbf{G}_3 = \nabla_3 \mathbf{y}\left(\mathbf{x}, \mathbf{w}^*; \mathcal{A}^*_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}), c_{2,\mathrm{best}}\right)\left(\mathcal{T}^{(1)}\left(\mathbf{u}(\mathbf{x}), c_{1,\mathrm{best}}\right)^{\mathrm{T}} \otimes \mathbf{I}_{N_O}\right)$$

$$\mathbf{G}_4 = \nabla_4 \mathbf{y}\left(\mathbf{x}, \mathbf{w}^*; \mathcal{A}^*_{N_H,c_1,c_2}\right) = \nabla \mathcal{T}^{(2)}\left(\mathbf{v}(\mathbf{x}), c_{2,\mathrm{best}}\right)$$