Engineering Structures 302 (2024) 117329

Contents lists available at ScienceDirect

Engineering Structures

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

Modeling of forced-vibration systems using continuous-time state-space neural network

Hong-Wei Li [a,b], Yi-Qing Ni [a,b,*], You-Wu Wang [a,b], Zheng-Wei Chen [a,b], En-Ze Rui [a,b], Zhao-Dong Xu [c]

[a] Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China
[b] National Rail Transit Electrification and Automation Engineering Technology Research Center (Hong Kong Branch), Hong Kong, China
[c] School of Civil Engineering, China-Pakistan Belt and Road Joint Laboratory on Smart Disaster Prevention of Major Infrastructures, Southeast University, Nanjing, 210096, China

ARTICLE INFO

Dataset link: https://github.com/lihongweiseu/CSNN

Keywords:
Forced-vibration systems
Surrogate modeling
Machine learning
Continuous-time domain
State-space neural network

ABSTRACT

Dynamic analysis of forced-vibration systems in civil engineering could be computationally inefficient or even hard to converge if the systems are stiff or highly complicated. Rapid advances in machine learning make it possible to formulate surrogate models for forced-vibration systems using neural networks. The widely used neural networks such as the convolutional neural network (CNN), recurrent neural network (RNN), etc., usually require a constant sampling rate and data length, thus they are difficult to be implemented for real-time calculation of the dynamic system with varying sampling rates. Recently, the continuous-time state-space neural network (CSNN) has shown the capability to lift these restrictions and has been drawing growing attention from the community. In this paper, we propose a generalized CSNN model for various forced-vibration systems (linear and nonlinear). The CSNN model comprises two sets of independent neural networks aiming to compute the state derivative and system response, respectively. Both neural networks adopt linear and nonlinear layers in parallel, instead of only fully connected nonlinear layers as adopted in the literature. This configuration is aimed to enhance the CSNN model with its capability to recognize the linear and nonlinear behaviors of systems. Additionally, the bias options in the CSNN model are all turned off to improve the stability of the model in the long-term time-series forecast, premised on the assumption that the forced-vibration systems are dissipative systems without drift, which is the most common case in civil engineering. Integration on the state derivative at the current time step is executed to obtain the state at the next time step using the explicit 4th-order Runge–Kutta method. Both numerical and experimental illustrative examples are provided, demonstrating that the CSNN model can achieve high performance and training efficiency with a few hyper-parameters, and thus is highly promising for engineering applications.

1. Introduction

Dynamic systems in engineering fields might be intricate and exhibit significant nonlinearities [1,2]. Traditional approaches for system identification and evaluation often require us to gain as much as we can about the physical knowledge of the system. For example, the physical model of a hydraulic actuator-specimen system should reflect the dynamics of the servo-valve, actuator, specimen, and the control–structure interaction [3–5], leading to at least a fifth-order linear or nonlinear model, depending on the complexity of the specimen [6–8]. Another example is that the memory effect of some materials could be well interpreted by the fractional derivative order model, while solving it in the time domain is challenging [9,10]. Such physical models need to be carefully dealt with using sophisticated mathematical tools and thereby are not attractive to engineers. Furthermore, accurate physical models are typically not available. Although the finite element (FE) updating method could be utilized to generate a FE model as close as possible to the true physical system by fitting the measured system responses [11–13], excessive computational resources are needed to update the parameters and produce a high-fidelity FE model.

In order to improve the computing efficiency and reduce the cost to construct the physical models, extensive studies have been carried out

* Corresponding author at: Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China.
E-mail addresses: hong-wei.li@polyu.edu.hk (H.W. Li), ceyqni@polyu.edu.hk (Y.Q. Ni), youwu.wang@polyu.edu.hk (Y.W. Wang), zhengwei.chen@polyu.edu.hk (Z.W. Chen), enze-98.rui@connect.polyu.hk (E.Z. Rui), zhdxu@163.com (Z.D. Xu).

https://doi.org/10.1016/j.engstruct.2023.117329
Received 19 June 2023; Received in revised form 19 November 2023; Accepted 11 December 2023
Available online 23 December 2023

on building data-driven non-physical (i.e., "black-box") models, such as fuzzy rule-based models [14–16], Bayesian models [17–21], etc. Apart from these classical approaches, the research community has gained great progress in leveraging neural networks to identify or represent forced-vibration systems. For example, the state-space neural network which is essentially a particular version of the recurrent neural network (RNN) has been developed to represent forced-vibration systems in the discrete-time domain [22–25]. Long short-term memory (LSTM) neural network which is another version of RNN has been adopted to predict the dynamic responses of nonlinear systems [26–29]. Convolutional neural network (CNN) has been used for structural black-box modeling, damage detection, and loss data reconstruction [30–33]. Additionally, by adding essential physical information about the systems to LSTM and CNN, the enhanced neural networks PhyLSTM [34] and PhyCNN [35] have been formed for structural seismic response modeling. In the above studies, the neural networks only function at a specific sampling rate after they are trained because they were all constructed in the discrete-time domain. However, the non-uniform sampling strategy has often been required in some engineering problems to release the sensing/data processing burden [36,37] or comply with the physical requirements of the systems in concern [38,39]. Although resampling techniques can be adopted to unify the sampling rate, other challenges such as stability and precision issues may arise in this process. Therefore, making neural network models work independent of the sampling rate has practical significance. Recently, neural ordinary differential equations (NODE) for the formulation of continuous-time neural network models have been widely investigated [40,41]. The NODE method works regardless of the change of sampling rate and has exhibited high performance in learning the unmodeled nonlinear dynamics of systems [42–45]. Furthermore, the continuous-time state-space neural network (CSNN) has been developed based on the idea of the NODE method and has shown improved performance and efficiency in the identification of nonlinear systems [46,47], therefore has enormous potential for applications in modeling and response prediction of dynamic systems. How to construct CSNN models for various forced-vibration systems is an appealing research topic and deserves further investigation.

In this paper, we establish a generalized CSNN model for forced-vibration systems in civil engineering. The state vector is introduced in the CSNN model as the hidden variable, with the goal of learning the inherent dynamic characteristics of the system subject to different excitations. The CSNN model consists of two independent sets of neural networks, labeled as state and output calculators, which are used to compute the state derivative and output vectors, respectively. Both the state and output calculators adopt linear and nonlinear neural network layers in parallel, enabling the CSNN model to capture the linear and nonlinear components in the responses of the system. The integration operations from state derivative to state using the explicit 4th-order Runge–Kutta (RK4) method make the CSNN model independent of the data sampling rate. Additionally, the CSNN model does not require the input data length to be fixed. With the above features, the proposed CSNN model is highly flexible and has a strong capability to predict system responses in real time.

This paper is organized as follows. In Section 2, the CSNN modeling methodology is presented, and the features and potential applications of the CSNN model are discussed in detail. In Section 3, the performance of the CSNN model is evaluated through three illustrative examples, where the first two are purely numerical examples (linear and nonlinear cases), and the third one investigates a 6-story building using recorded response data from 21 seismic events occurring during 1987 to 2018. Section 4 summarizes the conclusions. The data and codes adopted in this study are publicly available on GitHub at https://github.com/lihongweiseu/CSNN.

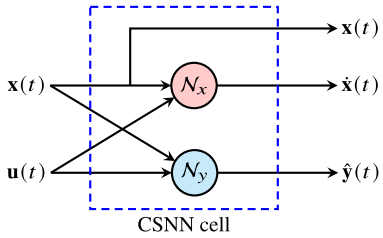

Fig. 1. Structure of the CSNN cell at time t.

2. Continuous-time state-space neural network (CSNN)

In this study, the forced-vibration system with excitations (*input vector*): $\mathbf{u}(t) \in \mathbb{R}^{n_u \times 1}$ and responses (*output vector*): $\mathbf{y}(t) \in \mathbb{R}^{n_y \times 1}$ is studied. Assuming that the true physical model of the system is completely unknown (black box), we will utilize the CSNN architecture to formulate a surrogate model of the system. The dynamic equation of an unknown system at any time t is constructed using a CSNN cell as shown in Fig. 1, given by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathcal{N}_x\left[\mathbf{x}(t), \mathbf{u}(t)\right], \\ \hat{\mathbf{y}}(t) &= \mathcal{N}_y\left[\mathbf{x}(t), \mathbf{u}(t)\right]. \end{aligned} \tag{1}$$

In Eq.(1), $\mathbf{x}(t) \in \mathbb{R}^{n_x \times 1}$ is the *state vector* which is used to learn the system behavior along the vibration trajectory; $\dot{\mathbf{x}}(t)$ is the *state derivative vector*; $\hat{\mathbf{y}}(t) \in \mathbb{R}^{n_y \times 1}$ is the *predicted output vector*; $\mathcal{N}_x(\cdot)$ and $\mathcal{N}_y(\cdot)$ are two neural networks, defined as the *state calculator* and *output calculator*, respectively. The initial state vector is assumed to be $\mathbf{0}$; therefore, the state vector is governed by

$$\mathbf{x}(t) = \int_0^t \mathcal{N}_x\left[\mathbf{x}(\tau), \mathbf{u}(\tau)\right] \mathrm{d}\tau. \tag{2}$$

Fig. 2 provides an illustration of CSNN for time-series modeling of a forced-vibration system, where

$$\mathbf{x}_i = \mathbf{x}(t_i),\ \mathbf{u}_i = \mathbf{u}(t_i),\ \hat{\mathbf{y}}_i = \hat{\mathbf{y}}(t_i),\ i = 0, 1, 2, \ldots, \tag{3}$$

$t_0 = 0$ s, and $[t_0, t_1, t_2, \ldots]$ is an ascending time series. The time interval between t_i and t_{i+1} is defined as $\Delta t_i = t_{i+1} - t_i$, which might vary with time. In general, the time interval is constant in the data processing for the sake of convenience. All the CSNN cells shown in Fig. 2 share the same model parameters.

The operations and neural network layers used in the state and output calculators are described in Fig. 3. For both the state and output calculators, the state and input vectors are concatenated first; then fed into a sequence of fully connected nonlinear layers and a single fully connected linear layer to calculate nonlinear and linear ingredients, respectively; and finally, the nonlinear and linear ingredients are summed up to generate the state derivative vector or output vector we need.

The graph representations of the state and output calculators are shown in Fig. 4, where the number of nonlinear layers in the state and output calculators is p and q, respectively. The hyperbolic tangent function, i.e., $\mathrm{Tanh}(\cdot)$, is adopted as the activation function in each nonlinear layer. Since the activation function $\mathrm{Tanh}(\cdot)$ could only generate the output greater than -1 and less than 1, the last nonlinear layer is followed by a fully connected linear layer to make the output vary in the entire real-number domain. Each layer contains several neurons, and the neuron number is equal to the size of the layer's output signal. There is only one linear layer in both the state and output calculators, thus the neuron numbers in their linear layers are fixed, which are n_x (state vector size) and n_y (output vector size), respectively. N sets of observed input and output vectors $\left[(\mathbf{u}_0, \mathbf{y}_0), (\mathbf{u}_1, \mathbf{y}_1), \ldots, (\mathbf{u}_{N-1}, \mathbf{y}_{N-1})\right]$ are used to train the CSNN model, i.e., to determine a set of the calculators' hyper-parameters (weights) which keep the errors between the

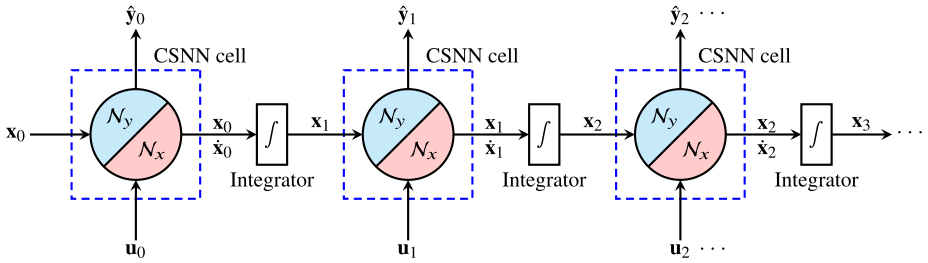

Fig. 2. Illustration of CSNN for time-series modeling of a forced-vibration system.

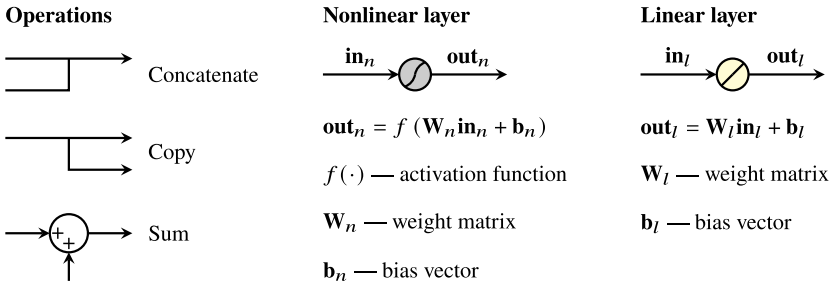

Fig. 3. Operations and layers used in the state and output calculators.

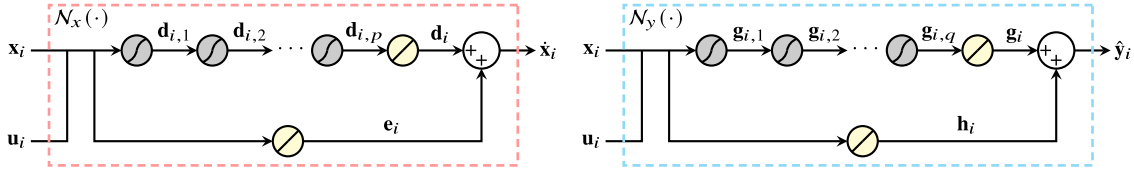

Fig. 4. Graph representations of the state and output calculators (left: state calculator; right: output calculator).

predictions $[\hat{\mathbf{y}}_0, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{N-1}]$ and observations $[\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{N-1}]$ minimum. The state vectors are treated as the hidden variables of the CSNN model.

It is worth mentioning that in Ref. [44], the authors established the linear state-space equations for a structure, and added a fully-connected neural network to the first equation which calculates the state derivative. This operation is similar to combining the nonlinear and linear neural network layers in parallel for the state calculator in this paper. The main difference is that, in Ref. [44], the linear state-space equations of the structure were known a priori and fixed, and the neural network was only used to capture the residual dynamics of the structure. Additionally, the state vector they used had explicit physical meanings (displacement and velocity). In this paper, we move a step forward to formulate the CSNN model for systems without any prior physical knowledge and let the model itself distinguish between the system's nonlinear and linear behaviors, where we do not assign specific physical meanings to the state vector of the CSNN model. In this way, the applicability of the CSNN model will not be restricted to certain problems.

The discrete form of the CSNN model is

$$\begin{aligned} \mathbf{x}_{i+1} &= \text{ODEint}\left[\mathcal{N}_x(\cdot), \mathbf{x}_i, \mathbf{u}_i, \mathbf{u}_{i+1}, \Delta t_i\right], \mathbf{x}_0 = \mathbf{0}, \\ \hat{\mathbf{y}}_i &= \mathcal{N}_y\left[\mathbf{x}_i, \mathbf{u}_i\right], \end{aligned} \tag{4}$$

where $\text{ODEint}\left[\mathcal{N}_x(\cdot), \mathbf{x}_i, \mathbf{u}_i, \mathbf{u}_{i+1}, \Delta t_i\right]$ refers to a numerical solver which integrates $\dot{\mathbf{x}}(t) = \mathcal{N}_x\left[\mathbf{x}(t), \mathbf{u}(t)\right]$ from the initial value $\mathbf{x}_i$ for the time interval Δt_i given the inputs $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ at t_i and t_{i+1}. The RK4 method is used in this paper. The time-moving-forward calculation of the solver is described in Algorithm 1, which is based on the procedure of the RK4 method for solving ordinary differential functions [48]. The difference is that the differential function is replaced by the neural network $\mathcal{N}_x(\cdot)$. The final slope used to calculate the state vector at t_{i+1} is estimated by a weighted average of the slope approximations at the beginning, midpoint, and end of the time interval. The midpoint input vector is required; however, it cannot be directly obtained. To address this issue, the average of the input vectors at the beginning and end of the time interval, i.e., $(\mathbf{u}_i + \mathbf{u}_{i+1})/2$, is used as the midpoint input vector. We have tested this treatment through several numerical cases with different time intervals and found that it could achieve high

Algorithm 1 Calculation of $\mathbf{x}_{i+1} = \text{ODEint}\left[\mathcal{N}_x(\cdot), \mathbf{x}_i, \mathbf{u}_i, \mathbf{u}_{i+1}, \Delta t_i\right]$ using the RK4 method.

Calculate the slope (state derivative) at t_i: $\mathbf{k}_1 = \mathcal{N}_x(\mathbf{x}_i, \mathbf{u}_i)$

Use $\mathbf{k}_1$ to calculate the temporary state at the interval midpoint: $\mathbf{x}_{i+0.5} = \mathbf{x}_i + \frac{\mathbf{k}_1}{2}\Delta t_i$

Use $\mathbf{x}_{i+0.5}$ to calculate the slope at the interval midpoint: $\mathbf{k}_2 = \mathcal{N}_x\left(\mathbf{x}_{i+0.5}, \frac{\mathbf{u}_i + \mathbf{u}_{i+1}}{2}\right)$

Use $\mathbf{k}_2$ to calculate the temporary state at the interval midpoint: $\mathbf{x}'_{i+0.5} = \mathbf{x}_i + \frac{\mathbf{k}_2}{2}\Delta t_i$

Use $\mathbf{x}'_{i+0.5}$ to calculate the slope at the interval midpoint: $\mathbf{k}_3 = \mathcal{N}_x\left(\mathbf{x}'_{i+0.5}, \frac{\mathbf{u}_i + \mathbf{u}_{i+1}}{2}\right)$

Use $\mathbf{k}_3$ to calculate the temporary state at t_{i+1}: $\mathbf{x}'_{i+1} = \mathbf{x}_i + \mathbf{k}_3\Delta t_i$

Use $\mathbf{x}'_{i+1}$ to calculate the slope at t_{i+1}: $\mathbf{k}_4 = \mathcal{N}_x\left(\mathbf{x}'_{i+1}, \mathbf{u}_{i+1}\right)$

Use the average slope to calculate the state at t_{i+1}: $\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)\Delta t_i$

stability and precision. It should be stressed, though, that this is only a numerical observation without rigorous proof. In addition to the RK4 method, the Euler method is another possible option. Nevertheless, the Euler method is not recommended due to its low precision, particularly when using relatively large time intervals.

Algorithm 1 essentially demonstrates how the CSNN model is implemented to predict the system responses in the discrete-time domain. Based on Algorithm 1, the training of the CSNN model is performed following the procedure described in Algorithm 2. Adam algorithm is adopted to optimize the model parameters in this study, and alternative algorithms such as stochastic gradient descent (SGD) algorithm and L-BFGS algorithm can also be adopted. Furthermore, multiple independent input–output datasets (batched data) can be considered in the training and prediction processes of the CSNN model.

Algorithm 2 Training process of the CSNN model.

Set the total training number: N_T
Calculate the number of data points for training: N
Initialize the model parameters (weights) with random values
Set $j = 1$

while $j \leq N_T$ **do**
  Initialize the loss function and state: $\mathcal{L} = 0$, $\mathbf{x}_0 = \mathbf{0}$
  Set $i = 0$
  Calculate the predicted output: $\hat{\mathbf{y}}_i = \mathcal{N}_y(\mathbf{x}_i, \mathbf{u}_i)$
  Calculate the loss function: $\mathcal{L} = \mathrm{RMS}(\mathbf{y}_i, \hat{\mathbf{y}}_i)$ ▷ mean squared error (squared L2 norm)

  while $i < N - 1$ **do**
    Calculate the state $\mathbf{x}_{i+1}$ using the RK4 method (Algorithm 1)
    Calculate the predicted output: $\hat{\mathbf{y}}_{i+1} = \mathcal{N}_y(\mathbf{x}_{i+1}, \mathbf{u}_{i+1})$
    Accumulate the loss function: $\mathcal{L} = \mathcal{L} + \mathrm{RMS}(\mathbf{y}_{i+1}, \hat{\mathbf{y}}_{i+1})$
    Set $i = i + 1$
  end while

  Calculate the gradient of the loss function $\mathcal{L}$ with respect to the model parameters (back-propagation)
  Update model parameters to reduce the loss function $\mathcal{L}$ ▷ Adam algorithm is adopted in this study
  Set $j = j + 1$
end while

In general, there are two features/advantages of the CSNN model:

1. The effectiveness of a trained CSNN model is not influenced by different data sampling rates/time intervals. Other neural networks that can deal with time-series problems, e.g., RNN, LSTM, etc., work for a specific time interval after they are trained [49,50], and they need to be trained again each time as the time interval changes.
2. The CSNN model could be formed as a single-time-step moving-forward model, with one input and one output (one-to-one) at each time point. Consequently, the CSNN model can be used to predict the system responses in real time regardless of the input data length. Other neural networks such as the physics-guided convolutional neural network (PhyCNN) proposed in Ref. [35], are only valid for a fixed input data length due to the adoption of fully connected layers. Moreover, the CSNN model has fewer hyper-parameters to train because it adopts identical model parameters for all time steps. This model configuration gives the CSNN model a considerable capability to avoid or mitigate the overfitting issue.

Additionally, compared to the existing CSNN models reported in the literature, the CSNN model proposed in this paper has the following refinements which are made specifically for the modeling of forced-vibration systems in civil engineering:

1. The hyper-parameters of the existing CSNN models include both weights and biases, while all the bias options in the state and output calculators of the proposed CSNN model are turned off. This modification is based on the fact that almost all forced-vibration systems in civil engineering are dissipative systems without drift, which means the systems' responses vanish gradually after the input signal decreases to zero. Turning off the bias options is a straightforward way to satisfy this requirement. Despite that there is no rigorous proof provided in this paper regarding the necessity and sufficiency of this modification, the authors have tried to train the CSNN models with the bias options turned off for the illustrative examples that will be presented later and found that better performance and faster training speed were achieved compared to the situation where the bias options were turned on. Therefore, we believe this modification is reasonable for the modeling of forced-vibration systems in civil engineering.
2. In literature, the state vectors of CSNN models normally had physical meanings, and their initial values were carefully handled. However, we do not assign physical meanings to the state vector of the proposed CSNN model, and the state vector is merely treated as the hidden variable to process data-driven modeling of forced-vibration systems. To guarantee that zero input series generate zero output series, the initial state vector is assumed to be zero in this paper.
3. Nonlinear neural network layers were usually used alone in previous studies since they are very powerful at fitting the training datasets, and it appeared unnecessary to adopt linear layers. However, this strategy normally results in a large number of hyper-parameters and might be inefficient for time-series problems. Instead of using nonlinear layers alone, the proposed CSNN model adopts both linear and nonlinear neural network layers, enabling the model to capture and balance linear and nonlinear components in the system responses. We will demonstrate through illustrative examples that, with this configuration, the CSNN model can achieve good performance using a relatively small number of hyper-parameters.

For the last feature, here we propose an approximate method to quantify the linear and nonlinear components $\hat{\mathbf{y}}_{l,i}$ and $\hat{\mathbf{y}}_{n,i}$ in the predicted system output $\hat{\mathbf{y}}_i$. The formulas to calculate $\hat{\mathbf{y}}_{l,i}$ and $\hat{\mathbf{y}}_{n,i}$ are given by

$$\begin{aligned} &\mathbf{x}_{l,i+1} = \mathrm{ODEint}\left[\mathcal{N}_{lx}(\cdot), \mathbf{x}_i, \mathbf{u}_i, \mathbf{u}_{i+1}, \Delta t_i\right], \mathbf{x}_{l,0} = \mathbf{0}, \\ &\hat{\mathbf{y}}_{l,i} = \mathcal{N}_{ly}\left[\mathbf{x}_{l,i}, \mathbf{u}_i\right], \hat{\mathbf{y}}_{n,i} = \hat{\mathbf{y}}_i - \hat{\mathbf{y}}_{l,i}, \end{aligned} \tag{5}$$

where $\mathcal{N}_{lx}(\cdot)$ and $\mathcal{N}_{ly}(\cdot)$ are the linear state and output calculators which are formed by removing all the nonlinear layers in $\mathcal{N}_x(\cdot)$ and $\mathcal{N}_y(\cdot)$, and $\mathbf{x}_{l,i}$ is the linear state. In Eq.(5), ODEint takes $\mathbf{x}_i$ rather than $\mathbf{x}_{l,i}$ as the initial state to ensure that the calculation of $\mathbf{x}_{l,i+1}$ is stable. The root mean square (RMS) of the linear and nonlinear components can be obtained using the following equations:

$$\mathbf{RMS}_l = \sqrt{\frac{1}{N}\sum_{i=0}^{N}\hat{\mathbf{y}}_{l,i}^2}, \ \mathbf{RMS}_n = \sqrt{\frac{1}{N}\sum_{i=0}^{N}\hat{\mathbf{y}}_{n,i}^2}. \tag{6}$$

And then, the *nonlinear ratio* $\mathbf{R}_n$ is defined as

$$\mathbf{R}_n = \frac{\mathbf{RMS}_n}{\mathbf{RMS}_l + \mathbf{RMS}_n} \times 100\% \tag{7}$$

to roughly indicate the proportion of the nonlinear component in the predicted system output. Note that $\mathbf{RMS}_l$, $\mathbf{RMS}_n$ and $\mathbf{R}_n$ have the same dimension as the system output, i.e., $\mathbf{RMS}_l$, $\mathbf{RMS}_n$, $\mathbf{R}_n \in \mathbb{R}^{n_y \times 1}$, and the calculations of Eqs.(6) and (7) are operated elementwise.

The features listed above make the proposed CSNN model workable for various problems within the domain of civil engineering such as system modeling, identification, and structural health monitoring (SHM). We list some possible applications of the proposed CSNN model in the following.

1. Construction of surrogate models for dynamic systems with comprehensive physical models, e.g., the fractional derivative order systems, where the time domain analysis requires a large amount of computation at each time step and the time interval must be very small to ensure convergence [51–53].

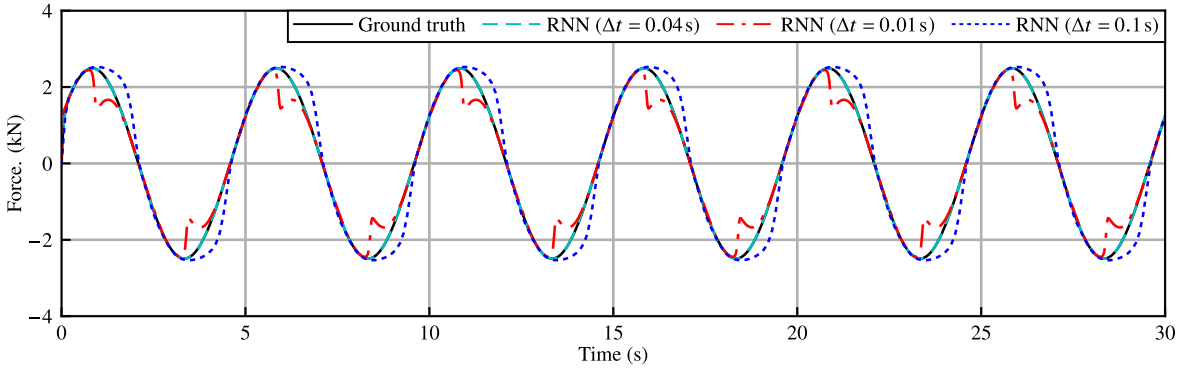

Fig. 5. Ground truth and RNN model prediction of the Zener system force response excited by $u_d(t) = 0.8\ \sin(0.4\pi t)$ (mm).

2. Data-driven modeling of unknown dynamic systems. The CSNN model can be used for dynamic modeling of devices such as the damper, isolation bearing, etc., filled with a newly invented material, solely based on the experimental results. In this way, the effort to determine a delicate model with clear physical meanings could be saved.
3. Online prediction of system responses. The CSNN model can be implemented for SHM tasks, where the target system responses are calculated and evaluated in real time based on the sensed data.

3. Illustrative examples

Three illustrative examples are given in this section. The first two are purely numerical examples. The third one uses the recorded monitoring data of a 6-story hotel building in San Bernardino, CA. The CSNN models in these illustrative examples are coded in Python utilizing the deep-learning library PyTorch. The data and codes used in this paper are publicly available on GitHub at https://github.com/lihongweiseu/CSNN.

3.1. Illustrative example 1: the shear Zener model of viscoelastic dampers

The Zener model [10,54] describing the shear behavior of viscoelastic dampers is expressed as

$$\frac{G_1+G_2}{\eta}F_d(t)+\dot{F}_d(t)=\frac{A_dG_1G_2}{h_d\eta}u_d(t)+\frac{A_dG_1}{h_d}G_1\dot{u}_d(t), \tag{8}$$

where $F_d(t)$ and $u_d(t)$ are the shear force and displacement of the damper, which are treated as the output and input; A_d and h_d are the shear area and height of the damper; the other parameters are shear moduli. The parameter values we use in this example are listed below [10].

$$G_1 = 4.79\,\text{MPa},\ G_2 = 0.479\,\text{MPa},\ \eta = 0.248\,\text{MPa},\ A_d = 6\times10^3\,\text{mm}^2,\ h_d = 10\,\text{mm}. \tag{9}$$

Eq. (8) represents a linear first-order differential system. We first use a one-to-one RNN model with 1 layer and 10 hidden states to fit the system responses. A sinusoidal excitation $u_d(t) = 0.8\ \sin(0.4\pi t)$ (mm) is adopted as the training input, where the time interval is $\Delta t = 0.04$ s and the time duration is 30 s, leading to 751 training datasets. The RNN model is trained 6000 times. Fig. 5 shows the ground truth and RNN model prediction of the system force response (output) excited by the training input. The RNN model performs well using the training time interval $\Delta t = 0.04$ s. Nevertheless, it fails to fit the data when the time interval changes to $\Delta t = 0.01$ s and $\Delta t = 0.1$ s. Similar results can be obtained when using the LSTM model to fit the system responses. To summarize, both RNN and LSTM models necessitate a fixed time interval, which renders them utterly impractical for dealing with dynamic time-series problems.

Next, we utilize the CSNN model to capture the system's dynamic behavior. Nine band-limited white noises (BLWN) with different time intervals, signal powers, and cut-off frequencies as listed in Table 1 are generated as the input displacement signals, which all have a time duration of 100 s. To obtain the BLWN signals, first, the MATLAB function wgn(·) is used to create the white Gaussian noise samples with different signal powers, and then the MATLAB low pass filters with different cut-off frequencies are acting on these noise samples. The first 40 s of the nine input displacement signals (cases 1–9) are shown in Fig. 6. Only the first 10 s of case 1 are used to train the CSNN model.

The sizes of the input and output are $n_u = 1$ and $n_y = 1$, respectively. The configuration of the CSNN model is described in Table 2. Two hidden states ($n_x = 2$) are used in the CSNN model. The state calculator $\mathcal{N}_x(\cdot)$ includes two nonlinear layers ($p = 2$) with two neurons ($n_d = 2$) in each layer, and the output calculator $\mathcal{N}_y(\cdot)$ includes one nonlinear layer ($q = 1$) with one neuron ($n_g = 1$). The model configuration shown in Table 2 is determined by comparing the model performance after training under different configurations, which are formed by setting all the coefficients (n_x, p, n_d, q and n_g) as 1 and gradually increasing them. Although the neuron numbers are the same for each nonlinear layer in this example, they could be set to different values. The total training number is set to $N_T = 6,000$.

The Pearson correlation coefficients r ($-1 \le r \le 1$, abbreviated as Pearson corr.) between the ground truth and CSNN model prediction of the force response, as well as the nonlinear ratios for the nine cases are listed in Table 3. Pearson corr. reflects the linear correlation or similarity between two signals, and higher Pearson corr. means higher correlation ($r = 1$ represents a perfect correlation, i.e., the two signals are exactly the same). The Pearson corr. listed in Table 3 are very close to 1, indicating an excellent performance of the CSNN model. It can be observed from Fig. 6 and Table 3 that when the input amplitude is higher than that considered in the training dataset, the Pearson corr. slightly decreases with the increase of the input amplitude (cases 2, 6, 7, and 8). Therefore, the input signals with a wider amplitude range are preferred for training the model, which should better cover the highest input amplitude that could possibly occur in the testing dataset. The nonlinear ratios listed in Table 3 are less than 35% (4.47% ∼ 34.90%), therefore the linear layers of the CSNN model have a larger impact on the system predictions than the nonlinear layers. Since the studied system Eq.(8) is a linear system, an ideal CSNN model should mute the nonlinear layers, i.e., minimize the absolute values of the nonlinear weights, to make the nonlinear ratios close to zero. Although we have tried to increase the training times and found that the nonlinear ratios could hardly be further decreased, the precision of the CSNN model is acceptable for this illustrative example. The ground truth and CSNN model prediction of the force response for case 6 are depicted in Fig. 7, where we can see that they coincide quite well.

It should be remarked that some model configurations could cause the model unstable if the model is insufficiently trained. For a brief

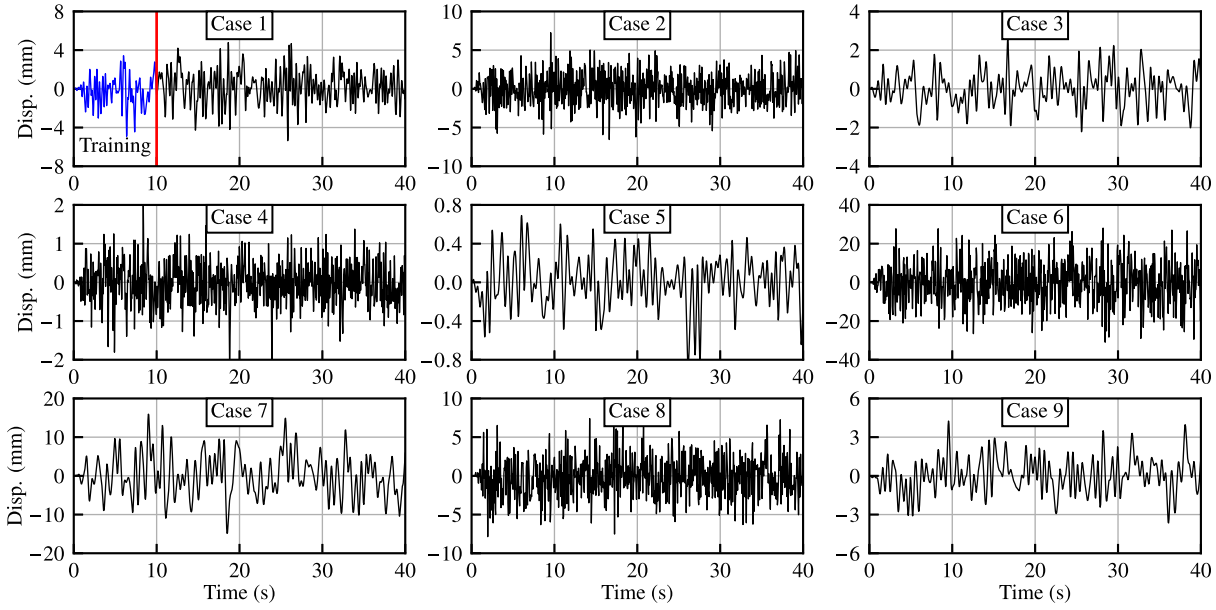

Fig. 6. Time histories (first 40 s) of the input signals for illustrative example 1.

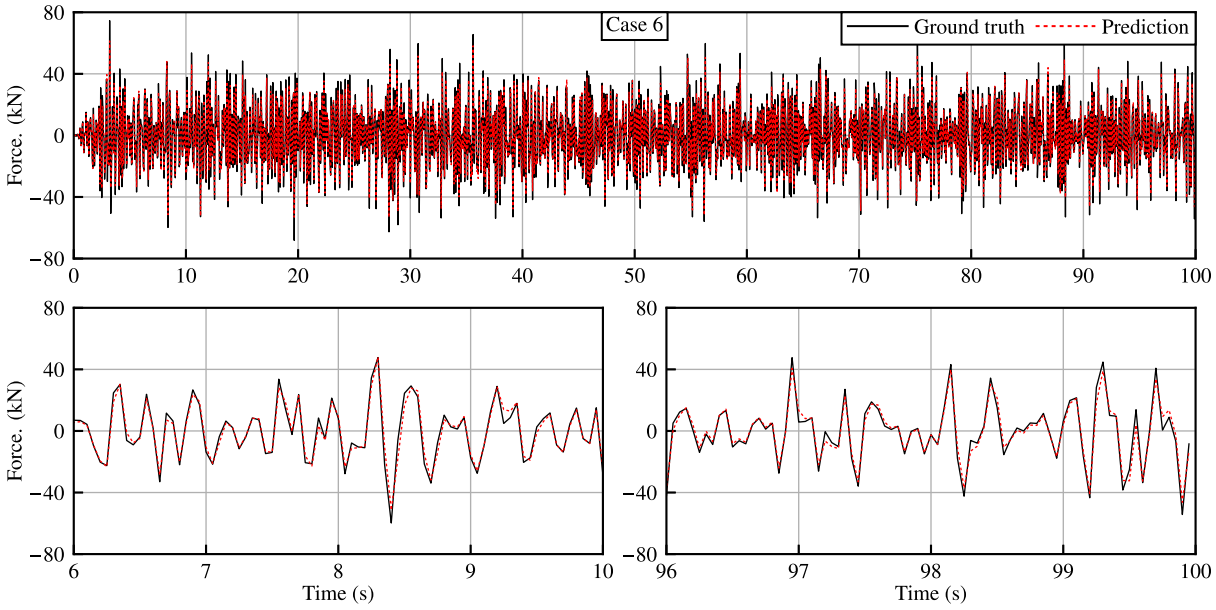

Fig. 7. Ground truth and CSNN model prediction of the force response for case 6 of illustrative example 1.

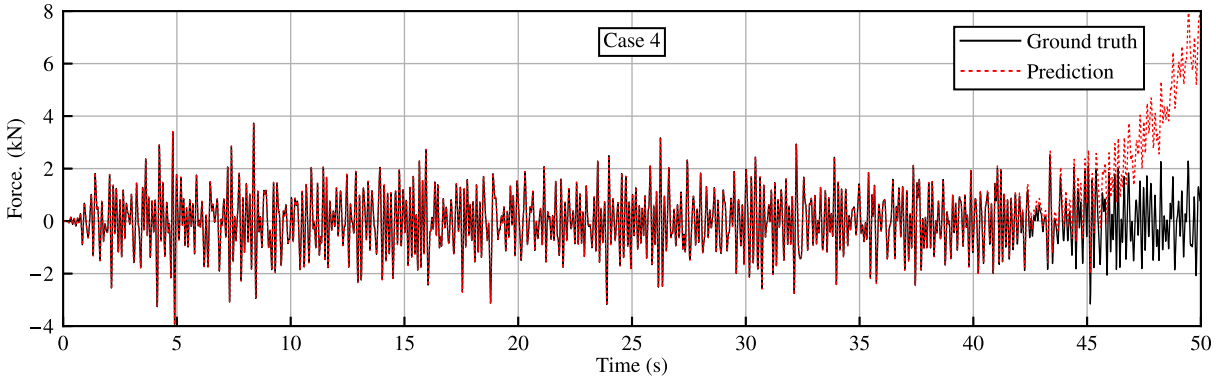

Fig. 8. Ground truth and CSNN model prediction of the force response for case 4 of illustrative example 1 ($p = 1$, trained 3000 times).

Table 1
Nine cases of BLWN displacement inputs.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9
Time interval (s)	0.01	0.002	0.002	0.002	0.002	0.05	0.05	0.05	0.05
Signal power (mm^2)	32	128	128	8	8	128	128	8	8
Cut-off frequency (Hz)	4	8	2	8	2	8	2	8	2

Table 2
Configuration of the CSNN model for illustrative example 1.

Size of state: n_x	Nonlinear layer number of $\mathcal{N}_x(\cdot)$: p	Neuron number of each nonlinear layer in $\mathcal{N}_x(\cdot)$: n_d	Nonlinear layer number of $\mathcal{N}_y(\cdot)$: q	Neuron number of each nonlinear layer in $\mathcal{N}_y(\cdot)$: n_g
2	2	2	1	1

Table 3
Pearson corr. between the ground truth and CSNN model prediction of the force response and the nonlinear ratios for the nine cases of illustrative example 1.

Case no.	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9
Pearson corr.	0.999942	0.999449	0.999955	0.999959	0.999934	0.985037	0.993983	0.999438	0.999915
Nonlinear ratio (%)	29.32	32.55	33.86	34.90	34.01	10.75	8.40	16.89	4.47

demonstration, Fig. 8 shows the results for case 4 after the model is trained 3000 times, where only one nonlinear layer is used in the state calculator and all other configuration settings stated in Table 2 remain unchanged. It can be seen that the CSNN model becomes unstable at about 45 s. And increasing the training times to 6000 would make the model stable (in the range of $0 \sim 100$ s).

In the following two illustrative examples, the CSNN model and the PhyCNN model proposed in Ref. [35] are compared. To make the comparison as fair as possible, we use the same training/testing datasets and total training numbers (3000 times for illustrative example 2, and 10,000 times for illustrative example 3) as used in Ref. [35]. All the training and testing data used in these two illustrative examples were downloaded from the GitHub project of Ref. [35] at https://github.com/zhry10/PhyCNN. The detailed descriptions of the selected input signals (earthquakes) and how the training and testing datasets are determined can be found in Ref. [35], which are omitted in this paper.

3.2. Illustrative example 2: a nonlinear system subjected to acceleration excitations

A single degree-of-freedom (DOF) nonlinear system subjected to acceleration excitations is investigated in this illustrative example. The nonlinear system is expressed as follows:

$$m\ddot{y}(t) + c\dot{y}(t) + k_1 y(t) + k_2 y^3(t) = -m\ddot{u}_g(t), \tag{10}$$

where the parameter values are mass $m = 1$ kg, damping coefficient $c = 1$ Ns/m, linear stiffness coefficient $k_1 = 20$ N/m, and nonlinear stiffness coefficient $k_2 = 200$ N/m^3 [35]. The database contains 99 independent seismic acceleration histories ($\ddot{u}_g(t)$) and the corresponding system displacement responses ($y(t)$). The first 10 datasets labeled as cases 1–10 are used for training and the rest (cases 11–99) are used for testing. Each input sequence is sampled at 20 Hz ($\Delta t = 0.05$ s) with a time duration of 50 s, leading to 1001 data points. The PhyCNN model has the following limitations. It is executed at the specific time interval $\Delta t = 0.05$ s and does not work for other time intervals. Additionally, it takes all 1001 acceleration data points in 50 s as the single input vector to calculate the 1001-point system responses. The input sequence with a different time duration must be cut off or padded to be a sequence of 50 s. The proposed CSNN model is not subject to these limitations. It is a one-to-one neural network model, and a well-trained CSNN model can be used to predict the system response with different time intervals and time durations.

The PhyCNN model consists of 5 convolution layers, 5 rectified linear unit (ReLU) activation functions (following at the end of each convolution layer), and finally, 3 fully connected linear layers. Each convolution layer has 64 filters and 50 kernels. For the CSNN model, we use the same model configuration as we used in illustrative example 1 (Table 2). As a result, the CSNN model has significantly fewer parameters than the PhyCNN model.

The Pearson corr. between the ground truth and model prediction of the system response for the 99 cases are displayed in Fig. 9. The PhyCNN model fits the training cases well except for case 8 and case 10. However, the PhyCNN model exhibits inconsistent performance for the testing cases, where the minimum Pearson corr. is 0.57781 (case 94). In contrast, the CSNN model achieves good performance for both training and testing cases. Therefore, the CSNN model is more stable than the PhyCNN model for this nonlinear illustrative example. The ground truth and model prediction of the displacement responses for cases 30 and 94 are shown in Fig. 10. It is apparent from Fig. 10 that the CSNN model predicts the system responses better than the PhyCNN model.

Fig. 11 is the scatter plot of the nonlinear ratio over the prediction's RMS for the 99 cases. The nonlinear ratios vary from 45.25% (case 52) to 85.05% (case 29). A trend that the nonlinear ratio increases with the prediction's RMS appears obviously. Therefore, it can be inferred that the nonlinear layers of the CSNN model contribute more when the system response is higher, which is in accord with the pattern that the nonlinear term $k_2 y^3(t)$ of the true model Eq.(10) becomes increasingly dominant as the system response increases.

3.3. Illustrative example 3: a 6-story hotel building with recorded seismic responses

In this illustrative example, a 6-story hotel in San Bernardino, California, built in 1970, is investigated. The recorded ground and hotel roof accelerations from 21 seismic events that occurred between 1987 to 2018 are selected as the input–output database for the training and testing of the CSNN model. The first 15 cases are used for training and the rest 6 cases are used for testing. Each acceleration sequence is sampled at 50 Hz ($\Delta t = 0.02$ s) with a time duration of 49.98 s, leading to 2500 data points. The PhySNN model is deployed with the input and output sizes being set to 2500 each. The configuration of the CSNN model used in this illustrative example is listed in Table 4.

Fig. 12 shows the Pearson corr. between the measurement and model prediction of the building response for the 21 cases. Similar results as Fig. 9 are observed in Fig. 12. The PhyCNN works very well for the training cases, while its performance noticeably degrades for the testing cases. However, the CSNN model achieves the same performance level for both training and testing cases, within a relatively small

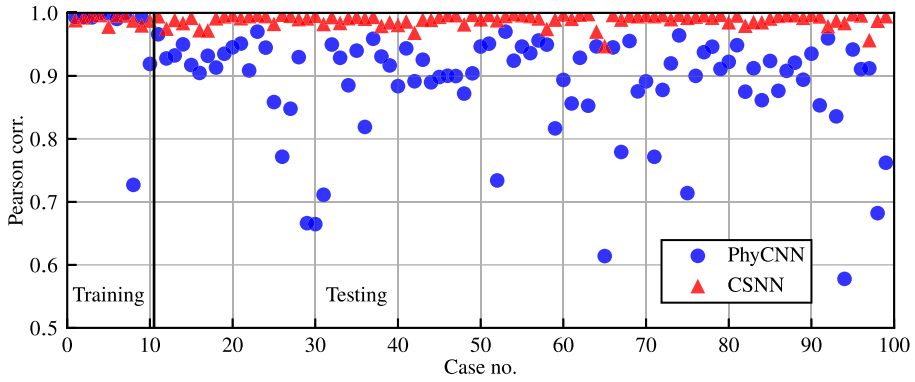

Fig. 9. Pearson corr. between the ground truth and model prediction of the system response for the 99 cases of illustrative example 2.

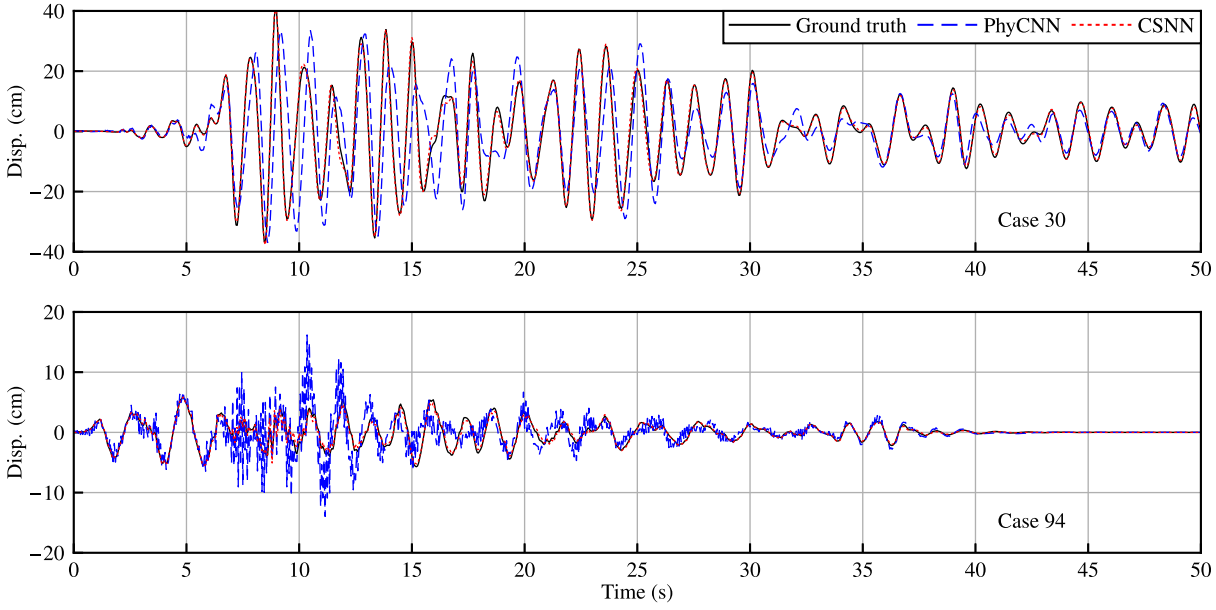

Fig. 10. Ground truth and model prediction of the displacement responses for cases 29, 30, 65, and 94 of illustrative example 2.

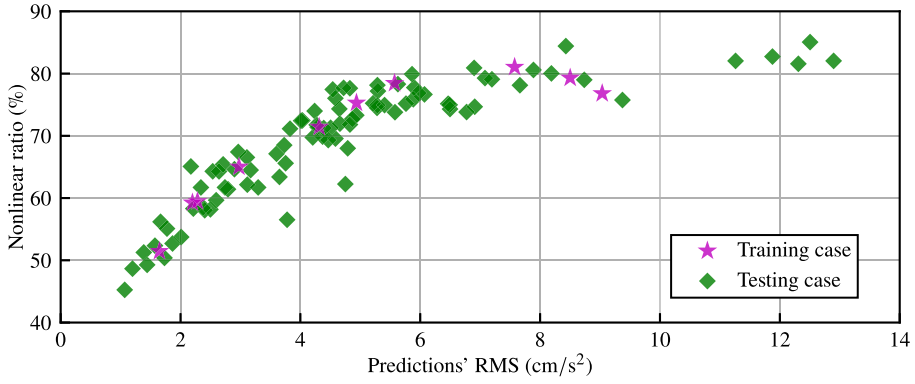

Fig. 11. Nonlinear ratios for the 99 cases of illustrative example 2.

Table 4
Configuration of the CSNN model for illustrative example 3.

Size of state: n_x	Nonlinear layer number of $\mathcal{N}_x(\cdot)$: p	Neuron number of each nonlinear layer in $\mathcal{N}_x(\cdot)$: n_d	Nonlinear layer number of $\mathcal{N}_y(\cdot)$: q	Neuron number of each nonlinear layer in $\mathcal{N}_y(\cdot)$: n_g
3	1	3	1	1

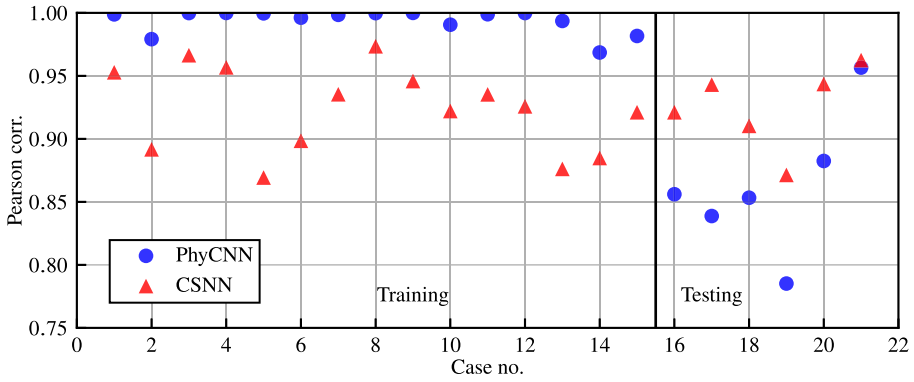

Fig. 12. Pearson corr. between the measurement and model prediction of the building response for the 21 cases of illustrative example 3.

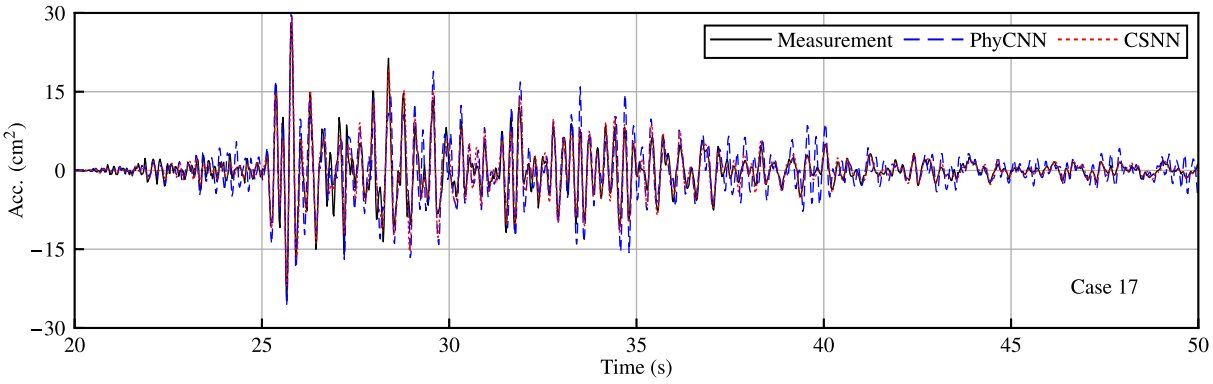

Fig. 13. Measurement and model prediction of the building roof acceleration response for case 17 of illustrative example 3.

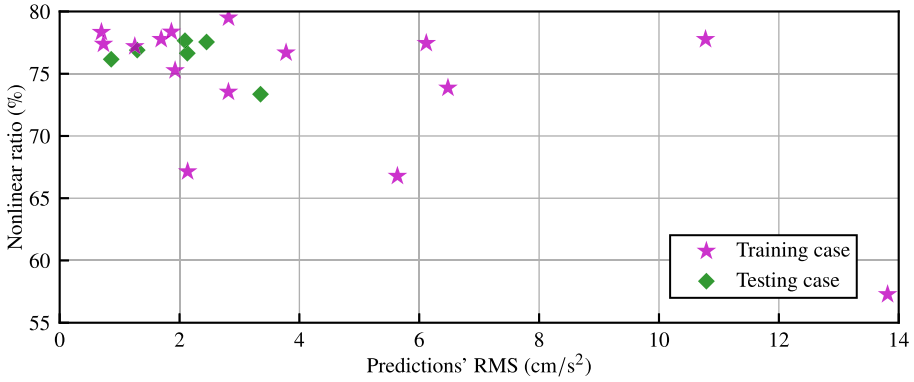

Fig. 14. Nonlinear ratios for the 21 cases of illustrative example 3.

variation range ($0.86242 \leq r \leq 0.97244$) compared to the PhyCNN model ($0.78518 \leq r \leq 0.99998$). Furthermore, the CSNN model performs better than the PhyCNN model for all six testing cases. The measurement and model prediction of the roof acceleration response for case 17 are shown in Fig. 13, where the CSNN model prediction agrees with the measured response better than the PhyCNN model. Fig. 14 is the scatter plot of the nonlinear ratio over the prediction's RMS for the 21 cases. The nonlinear ratios reach a relatively high level, varying from 57.28% (case 9) to 79.50% (case 11), and there is no obvious pattern in the variations of the nonlinear ratio. Therefore, the structure exhibits considerable nonlinearities for all cases.

Both illustrative examples 2 and 3 show that the CSNN model can represent a system's underlying dynamic properties and adapt to various excitation scenarios. The PhyCNN model adopts the neural network with a much larger scale than the CSNN model. The consequence is that the PhyCNN model tends to overfit the training datasets, and meanwhile performs erratically for the testing datasets, as shown in Figs. 9 and 12. Additionally, the PhyCNN model requires some quite strict preconditions, i.e., the time duration, time interval, and the number of input points must be fixed, which significantly diminishes the operability of the PhyCNN model. Contrarily, there are no such preconditions in the CSNN model, making it highly adaptive for modeling and prediction under a variety of circumstances.

4. Conclusion

This paper presented a novel CSNN architecture to elicit data-driven surrogate models for forced-vibration systems. The CSNN model includes state and output calculators, which are used to compute the state derivative and output vectors, respectively. Both state and output calculators consist of several nonlinear neural network layers and a linear neural network layer in parallel. The RK4 method is utilized to

update the state vector and calculate the output vector. The execution of the CSNN model does not require the time interval and time duration of the input data to be fixed, and therefore is highly flexible and adaptive.

Three illustrative examples were provided to demonstrate the effectiveness of the proposed CSNN model. Illustrative example 1 indicates that the CSNN model has a great capability to represent the dynamic behavior of a linear system using very limited training datasets, even though the testing input signals are substantially different from the training input signals. Illustrative examples 2 and 3 compare the CSNN model with the PhyCNN model for the response prediction of a highly nonlinear numerical system and a 6-story building with measured data, respectively. The results show that the CSNN model achieves better and more stable performance than the PhyCNN model. It is important to remark that the influences of structural uncertainty and noise of measured signals are not considered in this paper, while they are core issues that need to be addressed to guarantee the robustness of the CSNN model. Based on the current work, we believe that the proposed CSNN model has great potential, and more in-depth studies will be undertaken in our follow-up work to further explore the CSNN model for various system modeling/identification and structural health monitoring problems, with the presence of uncertainties and sensor noises.

CRediT authorship contribution statement

Hong-Wei Li: Methodology, Data curation, Software, Formal analysis, Writing – original draft, Visualization. **Yi-Qing Ni:** Conceptualization, Validation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **You-Wu Wang:** Validation, Supervision. **Zheng-Wei Chen:** Validation, Visualization. **En-Ze Rui:** Data curation, Visualization. **Zhao-Dong Xu:** Supervision, Funding acquisition.

Declaration of competing interest

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

Data availability

The raw and processed data required to reproduce the above findings are available to download from GitHub at https://github.com/lihongweiseu/CSNN.

Acknowledgments

The work described in this paper was supported by a grant from the Research Grants Council (RGC) of the Hong Kong Special Administrative Region (SAR), China (Grant No. PolyU 152308/22E), the Hong Kong Polytechnic University's Start-up Fund for RAPs under the Strategic Hiring Scheme (Grant No. P0046770), and a grant from the National Key Research and Development Program of China (Grant No. 2019YFE0121900). The authors also appreciate the funding support by the Innovation and Technology Commission of Hong Kong SAR Government to the Hong Kong Branch of National Engineering Research Center on Rail Transit Electrification and Automation (Grant No. K-BBY1).

References

[1] Khalil H. Nonlinear systems. 3rd ed.. Upper Saddle River, New Jersey: Prentice Hall; 2002.
[2] Li H-W, Wang F, Ni Y-Q, Wang Y-W, Xu Z-D. An adaptive and robust control strategy for real-time hybrid simulation. Sensors 2022;22(17):6569.
[3] Dyke S, Spencer Jr B, Quast P, Sain M. Role of control-structure interaction in protective system design. J Eng Mech 1995;121(2):322–38.
[4] Stehman M, Nakata N. IIR compensation in real-time hybrid simulation using shake tables with complex control-structure-interaction. J Earthq Eng 2016;20(4):633–53.
[5] Li H-W, Maghareh A, Montoya H, Uribe JWC, Dyke SJ, et al. Sliding mode control design for the benchmark problem in real-time hybrid simulation. Mech Syst Signal Process 2021;151:107364.
[6] Maghareh A, Silva CE, Dyke SJ. Parametric model of servo-hydraulic actuator coupled with a nonlinear system: Experimental validation. Mech Syst Signal Process 2018;104:663–72.
[7] Silva CE, Gomez D, Maghareh A, Dyke SJ, Spencer Jr BF. Benchmark control problem for real-time hybrid simulation. Mech Syst Signal Process 2020;135:106381.
[8] Li H-W, Maghareh A, Wilfredo Condori Uribe J, Montoya H, Dyke SJ, Xu Z. An adaptive sliding mode control system and its application to real-time hybrid simulation. Struct Control Health Monit 2022;29(1):e2851.
[9] Monje CA, Chen Y-Q, Vinagre BM, Xue D-Y, Feliu-Batlle V. Fractional-order systems and controls: Fundamentals and applications. london: Springer; 2014.
[10] Li H-W, Gomez D, Dyke SJ, Xu Z-D. Fractional differential equation bearing models for base-isolated buildings: Framework development. J Struct Eng 2020;146(2):04019197.
[11] Skolnik D, Lei Y, Yu E, Wallace JW. Identification, model updating, and response prediction of an instrumented 15-story steel-frame building. Earthq Spectra 2006;22(3):781–802.
[12] Song W, Dyke S. Real-time dynamic model updating of a hysteretic structural system. J Struct Eng 2014;140(3):04013082.
[13] Alkayem NF, Cao M, Zhang Y, Bayat M, Su Z. Structural damage detection using finite element model updating with evolutionary algorithms: A survey. Neural Comput Appl 2018;30(2):389–411.
[14] Chen M-Y, Linkens DA. Rule-base self-generation and simplification for data-driven fuzzy models. Fuzzy Sets and Systems 2004;142(2):243–65.
[15] Solomatine D, See LM, Abrahart R. Data-driven modelling: Concepts, approaches and experiences. Practical Hydroinf 2009;17–30.
[16] Kerr-Wilson J, Pedrycz W. Generating a hierarchical fuzzy rule-based model. Fuzzy Sets and Systems 2020;381:124–39.
[17] Yan G, Sun H, Büyüköztürk O. Impact load identification for composite structures using Bayesian regularization and unscented Kalman filter. Struct Control Health Monit 2017;24(5):e1910.
[18] Chen Z, Zhang R, Zheng J, Sun H. Sparse Bayesian learning for structural damage identification. Mech Syst Signal Process 2020;140:106689.
[19] Ni YQ, Wang YW, Zhang C. A Bayesian approach for condition assessment and damage alarm of bridge expansion joints using long-term structural health monitoring data. Eng Struct 2020;212:110520.
[20] Wang YW, Ni YQ, Wang X. Real-time defect detection of high-speed train wheels by using Bayesian forecasting and dynamic model. Mech Syst Signal Process 2020;139:106654.
[21] Wang YW, Ni YQ, Zhang QH, Zhang C. Bayesian approaches for evaluating wind-resistant performance of long-span bridges using structural health monitoring data. Struct Control Health Monit 2021;28(4):e2699.
[22] Suykens JA, De Moor BL, Vandewalle J. Nonlinear system identification using neural state space models, applicable to robust control design. Internat J Control 1995;62(1):129–52.
[23] Zamarreño JM, Vega P. State space neural network. Properties and application. Neural networks 1998;11(6):1099–112.
[24] Rangapuram SS, Seeger MW, Gasthaus J, Stella L, Wang Y, Januschowski T. Deep state space models for time series forecasting. In: Advances in neural information processing systems, vol. 31, Curran Associates, Inc.; 2018, p. 1–10.
[25] Pulido B, Zamarreño JM, Merino A, Bregon A. State space neural networks and model-decomposition methods for fault diagnosis of complex industrial systems. Eng Appl Artif Intell 2019;79:67–86.
[26] Wang Y. A new concept using LSTM neural networks for dynamic system identification. In: 2017 American control conference. IEEE; 2017, p. 5324–9.
[27] Gonzalez J, Yu W. Non-linear system modeling using LSTM neural networks. IFAC-PapersOnLine 2018;51(13):485–9.
[28] Zhang R, Chen Z, Chen S, Zheng J, Büyüköztürk O, Sun H. Deep long short-term memory networks for nonlinear structural seismic response prediction. Comput Struct 2019;220:55–68.
[29] Yue Z, Ding Y, Zhao H, Wang Z. Mechanics-guided optimization of an LSTM network for real-time modeling of temperature-induced deflection of a cable-stayed bridge. Eng Struct 2022;252:113619.
[30] Lei X, Sun L, Xia Y. Lost data reconstruction for structural health monitoring using deep convolutional generative adversarial networks. Struct Health Monit 2021;20(4):2069–87.
[31] Li Y, Ni P, Sun L, Zhu W. A convolutional neural network-based full-field response reconstruction framework with multitype inputs and outputs. Struct Control Health Monit 2022;29(7):e2961.
[32] Ghiasi A, Moghaddam MK, Ng C-T, Sheikh AH, Shi JQ. Damage classification of in-service steel railway bridges using a novel vibration-based convolutional neural network. Eng Struct 2022;264:114474.

[33] Chen L, Chen W, Wang L, Zhai C, Hu X, Sun L, et al. Convolutional neural networks (CNNs)-based multi-category damage detection and recognition of high-speed rail (HSR) reinforced concrete (RC) bridges using test images. Eng Struct 2023;276:115306.
[34] Zhang R, Liu Y, Sun H. Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. Comput Methods Appl Mech Engrg 2020;369:113226.
[35] Zhang R, Liu Y, Sun H. Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. Eng Struct 2020;215:110704.
[36] Yang Y-C, Nagarajaiah S. Output-only modal identification by compressed sensing: Non-uniform low-rate random sampling. Mech Syst Signal Process 2015;56:15–34.
[37] Fan X-P. Bridge extreme stress prediction based on Bayesian dynamic linear models and non-uniform sampling. Struct Health Monit 2017;16(3):253–61.
[38] Gomozov O, Trovão JPF, Kestelyn X, Dubois MR. Adaptive energy management system based on a real-time model predictive control with nonuniform sampling time for multiple energy storage electric vehicle. IEEE Trans Veh Technol 2016;66(7):5520–30.
[39] Zou L, Wang Z-D, Zhou D-H. Moving horizon estimation with non-uniform sampling under component-based dynamic event-triggered transmission. Automatica 2020;120:109154.
[40] Chen RTQ, Rubanova Y, Bettencourt J, Duvenaud DK. Neural ordinary differential equations. In: Advances in neural information processing systems, vol. 31, Curran Associates, Inc.; 2018, p. 1–13.
[41] Massaroli S, Poli M, Park J, Yamashita A, Asama H. Dissecting neural ODEs. In: Advances in neural information processing systems, vol. 33, Curran Associates, Inc.; 2020, p. 3952–63.
[42] Quaglino A, Gallieri M, Masci J, Koutník J. SNODE: Spectral discretization of neural ODEs for system identification. 2020, p. 1–15, arXiv preprint arXiv:1906.07038.
[43] Alvarez VMM, Rosca R, Falcutescu CG. DyNODE: Neural ordinary differential equations for dynamics modeling in continuous control. 2020, p. 1–9, arXiv preprint arXiv:2009.04278.
[44] Lai Z, Mylonas C, Nagarajaiah S, Chatzi E. Structural identification with physics-informed neural ordinary differential equations. J Sound Vib 2021;508:116196.
[45] Rahman A, Drgoňa J, Tuor A, Strube J. Neural ordinary differential equations for nonlinear system identification. 2022, p. 1–6, arXiv preprint arXiv:2203.00120.
[46] Forgione M, Piga D. Continuous-time system identification with neural networks: Model structures and fitting criteria. Eur J Control 2021;59:69–81.
[47] Beintema GI, Schoukens M, Tóth R. Continuous-time identification of dynamic state-space models by deep subspace encoding. In: The eleventh international conference on learning representations. 2023, p. 1–15.
[48] Butcher JC. Numerical methods for ordinary differential equations. 3rd ed.. John Wiley & Sons; 2016.
[49] Sahin SO, Kozat SS. Nonuniformly sampled data processing using LSTM networks. IEEE Trans Neural Netw Learn Syst 2018;30(5):1452–61.
[50] Yuan X, Li L, Wang K, Wang Y. Sampling-interval-aware LSTM for industrial process soft sensing of dynamic time sequences with irregular sampling measurements. IEEE Sens J 2021;21(9):10787–95.
[51] Li H-W, Gomez D, Dyke SJ, Xu Z-D, Dai J. Investigating coupled train-bridge-bearing system under earthquake-and train-induced excitations. J Vib Acoust 2021;143(5).
[52] Li H-W, Xu Z-D, Gomez D, Gai P-P, Wang F, Dyke SJ. A modified fractional-order derivative Zener model for rubber-like devices for structural control. J Eng Mech 2022;148(1):04021119.
[53] Li H-W, Xu Z-D, Wang F, Gai P-P, Gomez D, Dyke SJ. Development and validation of a nonlinear model to describe the tension–compression behavior of rubber-like base isolators. J Eng Mech 2023;149(2):04022104.
[54] Pritz T. Analysis of four-parameter fractional derivative model of real solid materials. J Sound Vib 1996;195(1):103–15.