Deep Convolutional Neural Network for Structural Dynamic Response Estimation and System Identification

Rih-Teng Wu[1] and Mohammad R. Jahanshahi, A.M.ASCE[2]

Abstract: This study presents a deep convolutional neural network (CNN)-based approach to estimate the dynamic response of a linear single-degree-of-freedom (SDOF) system, a nonlinear SDOF system, and a full-scale 3-story multidegree of freedom (MDOF) steel frame. In the MDOF system, roof acceleration is estimated through the input ground motion. Various cases of noise-contaminated signals are considered in this study, and the conventional multilayer perceptron (MLP) algorithm serves as a reference for the proposed CNN approach. According to the results from numerical simulations and experimental data, the proposed CNN approach is able to predict the structural responses accurately, and it is more robust against noisy data compared with the MLP algorithm. Moreover, the physical interpretation of CNN model is discussed in the context of structural dynamics. It is demonstrated that in some special cases, the convolution kernel has the capability of approximating the numerical integration operator, and the convolution layers attempt to extract the dominant frequency signature observed in the ideal target signal while eliminating irrelevant information during the training process. **DOI: 10.1061/(ASCE)EM.1943-7889.0001556.** *© 2018 American Society of Civil Engineers.*

Introduction

Motivation and Related Work

During the last decades, many scientists have made efforts to propose effective structural health monitoring (SHM) techniques for civil buildings and structures. Estimation of structural vibration responses has been intensively investigated because they provide useful information for inferring the health state of a structure as well as inherent structural characteristics. Using nonlinear time-history analysis, the performance level of a structure under various earthquake intensities can be determined through maximum drift estimation. For instrumented structures, data recorded from acceleration sensors can be used to compute the fundamental frequencies and mode shapes. Changes in the identified characteristics serve as an indicator for health assessment.

Recently, advances in machine learning (ML) techniques have led to more opportunities for research in SHM. These ML approaches attempt to learn the underlying mechanisms from the available measurements and, subsequently, predict the possible outcomes given new input. More recently, the emergence of deep-learning approaches provide new opportunities for ML-based researches. Contrary to conventional ML techniques, deep learning usually refers to the establishment of a larger and deeper network with the aid of large amounts of data. Due to recent developments in computation and sensor technology, the accessibility of data acquisition is much easier compared with in the past and therefore increases the applicability of deep-learning approaches. This study proposes a deep convolutional neural network (CNN) to estimate the dynamic response of a linear single-degree-of-freedom (SDOF) system, a nonlinear SDOF system, and a multidegree of freedom (MDOF) 3-story steel frame. The conventional multilayer perceptron (MLP) approach is adopted to serve as a reference for the performance of the proposed deep CNN-based method.

MLP has been widely used to estimate the dynamic response or characteristics of a system. Masri et al. (1996) used MLP to model the dynamic response of linear and nonlinear systems. The network was trained using simulated vibration data to represent the behavior of the healthy structure, and damage detection was achieved later through the discrepancy between the network's prediction and the vibration response from a damaged structure. Masri et al. (2000) used a similar concept to that used by Masri et al. (1996) to model a four-degree-of-freedom dynamic system using MLP. The damage identification was conducted through the least-square error between the measured output from a damaged structure and the predicted response from the trained MLP with data from the healthy structure. Pei et al. (2005a) addressed how to fit a nonlinear function with MLP and used MLP to model a nonlinear dynamic system. Pei et al. (2005b), Pei and Smyth (2006a, b), and Pei et al. (2007, 2011) discussed the ability of MLP to approximate nonlinear functions in engineering mechanics. Also, those works were dedicated to developing constructive methods for the initialization of a MLP configuration where the MLPs were trained to predict the nonlinear restoring force by using displacement and velocity inputs.

Pei and Mai (2008), Pei et al. (2013), and Pei and Masri (2015) demonstrated the use of MLP to fit functions like first-order polynomial, exponential, and Gaussian function. Data from laboratory tests were used to validate the function fitting concept. Approximations of the displacement and acceleration transmissibility functions as well as the restoring force of a viscous fluid damper were presented. Xu et al. (2009) used MLP to estimate the acceleration response of a four-degree-of-freedom structure given the acceleration and excitation measurement at previous time steps. The input–output relationship of MLP was then adopted to estimate structural parameters such as mass and stiffness of the structure based on an equivalent autoregressive moving-average model. Wang (2011) localized and quantified damage in building structures through the use of MLP to identify the stiffness reduction in the damaged story given changes in the natural frequency.

[1]Ph.D. Student, Lyles School of Civil Engineering, Purdue Univ., West Lafayette, IN 47907 (corresponding author). Email: wu952@purdue.edu

[2]Assistant Professor, Lyles School of Civil Engineering, Purdue Univ., West Lafayette, IN 47907.

Note. This manuscript was submitted on December 10, 2017; approved on July 6, 2018; published online on October 31, 2018. Discussion period open until March 31, 2019; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Engineering Mechanics***, © ASCE, ISSN 0733-9399.

Cury and Crémona (2012) presented a damage classification scheme for a beam-type structure using MLP. Data histograms from the vibration signals, natural frequencies, and mode shapes served as inputs to the network. The effect of noise-contaminated signals was also discussed. Arangio and Beck (2012) incorporated Bayesian inference with MLP to achieve the damage detection, localization, and quantification for bridge structures. Similar to the scheme of Masri et al. (1996), damage was detected through the error between the measured data and prediction of a MLP trained by the undamaged structure. Suresh et al. (2012) used MLP to conduct the active control of a base-isolated building. They used MLP to approximate the nonlinear control law of the active controller and reduced the vibration response of the building by applying appropriate force achieved from the controller. Kao and Loh (2013) presented a health monitoring method for dam inspection using MLP. The long-term deformation of the dam was estimated through MLP given the information extracted from the water level and temperature distribution of the dam body. Razavi et al. (2014) used MLP to fit the load–deflection curve of a carbon fiber–reinforced polymer (CFRP) RC slab. Derkevorkian et al. (2015) combined MLP and an ordinary differential equation (ODE) solver to predict the relative displacement and velocity time history, where MLP was adopted to model the soil–structure interaction. The difference between the prediction of MLP and the measured data served as a tool to detect damage.

CNNs have been quite popular in speech recognition, image classification, and time-series modeling (LeCun and Bengio 1995). For instance, there havev been significant achievements in CNN-based approaches for speaker/gender identification and spoken-word/music recognition using one-dimensional (1D) acoustic signals (Längkvist et al. 2014). Due to its ability to learn the spatial invariant features automatically, CNN leads to a great success in object classification using large amounts of image data (LeCun et al. 2015; Krizhevsky et al. 2012). Recently, Cha et al. (2017) used CNN to detect cracks in concrete surfaces. Their results demonstrated that a CNN-based approach is capable of dealing with real-world challenges such as varying illumination conditions. Abdeljaber et al. (2017) proposed a CNN-based damage-detection approach for a planar steel frame. The CNN was used to extract the dominant features from 1D acceleration signals to perform damage detection and localization. Tompson et al. (2017) adopted CNN to accelerate two-dimensional (2D) and three-dimensional (3D) motion simulations of fluids and smoke, where the CNN was used to infer fluid pressure, which is the solution of the Poisson equation, which requires a large number of computation iterations. Chen and Jahanshahi (2018) used CNN to detect and localize cracks on the metallic surfaces of nuclear power plant components. Crack-detection results from different video frames were aggregated to enhance the final detection result. Atha and Jahanshahi (2018) proposed a CNN-based approach for corrosion detection on metallic surfaces. It was shown that the CNN is capable of learning appropriate classification features automatically without the need of human handcrafted features.

Although CNN has been explored in many research fields, most studies used CNNs as a feature extractor to conduct classification/recognition tasks. Only a few works used CNNs to perform regression or function approximation problems. This study presents a deep CNN-based approach for the response estimation in structural dynamic systems. Results from the CNN are compared with the results obtained from a MLP-based approach. The robustness of the proposed approach against noise is discussed through the use of various noise-contaminated vibration signals. Data used in this study consist of both numerical simulation and laboratory test data. Finally, the structural characteristics of the test steel frame are identified through the response predicted by the CNN model.

Contribution and Scope

To the best understanding of the authors, this paper presents the first attempt to adopt deep CNN for response estimation (regression) in structural dynamic problems. A deep CNN is used to estimate the vibration responses of a linear SDOF system, a nonlinear SDOF system, and a 3-story steel frame. The results of a MLP-based approach serve as a reference for the proposed CNN method, and the system identification of the steel frame is achieved through the estimations obtained from the CNN model. The rest of the paper is organized as follows: the "Methodology" section presents the formulation and configuration of MLP and CNN. The "Single-Degree-of-Freedom System" section discusses the analysis results from a linear and nonlinear SDOF system. The "Multidegree-of-Freedom System" section presents the results of vibration data collected from a shaking-table test of a 3-story steel frame where the CNN model was used to conduct the system identification of the frame. The "Physical Interpretation of Convolution Layer" section discusses the physical interpretations for the convolution layer. Concluding remarks and further discussions are addressed in the last section.

Methodology

Multilayer Perceptron

MLP, first inspired by the biological neural system, typically consists of one input layer, multiple hidden layers, and one output layer (Basheer and Hajmeer 2000). Fig. 1 depicts a MLP configuration with only one hidden layer. Each node in the hidden or output layer is fully connected with all the nodes in the previous layer, and these connections stand for the weighted combination of the nodes in the previous layer. For instance, consider the output value for node j indicated in Fig. 1 that is computed through the following two equations:

$$z_j = \sum (w_{ij} x_i + b) \tag{1}$$

$$y_j = f(z_j) \tag{2}$$

where x_i = all inputs; w_{ij} = weight between nodes i and j; b = bias term for regularization; z_j = temporary value; and y_j = output value for node j, which is computed by passing z_j through an activation function $f(\cdot)$.

The activation function allows MLP to deal with many problems across various research fields. It has been demonstrated that MLP is capable of approximating any linear or nonlinear function by

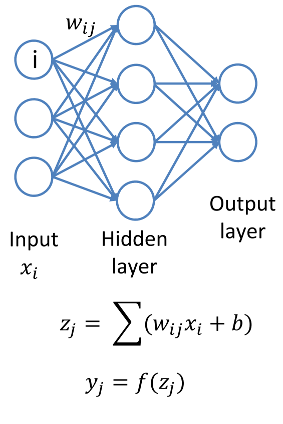


Fig. 1. Sample MLP configuration.

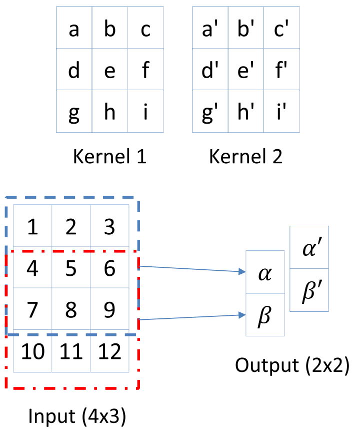


Fig. 2. Illustration of a convolution layer.

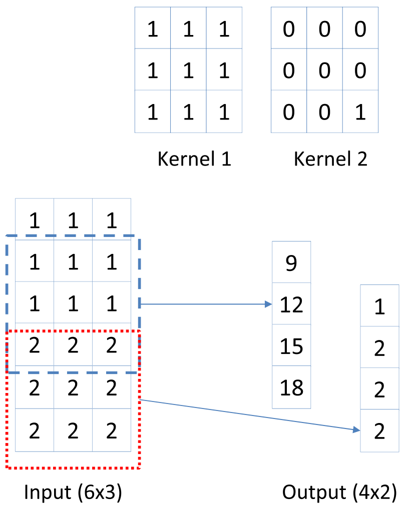


Fig. 3. Example of convolution operations.

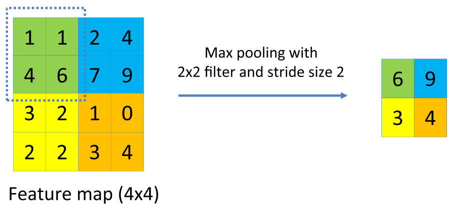


Fig. 4. Illustration of a pooling layer.

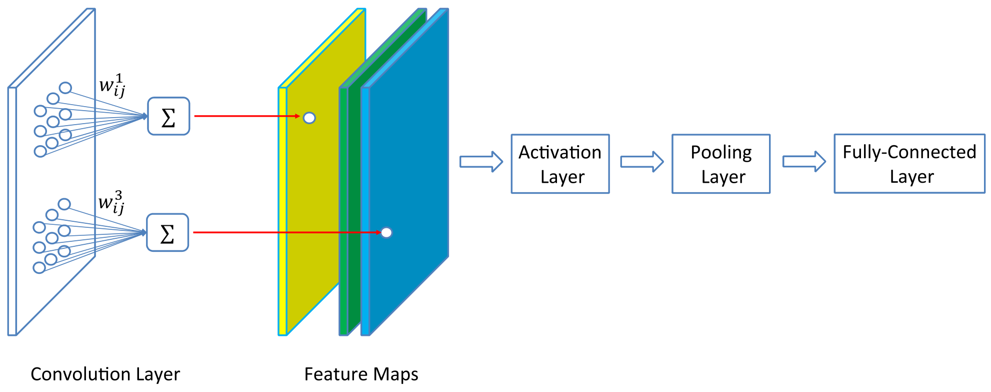


Fig. 5. Typical CNN configuration.

providing appropriate constraints. There are two activation functions considered in this study: hyperbolic tangent [$\tan h(x)$ in Eq. (3)] and the rectified linear unit (RELU) [Eq. (4)]

$$f(x) = \tan h(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{3}$$

$$f(x) = \max(0, x) \tag{4}$$

To establish a MLP model, a set of training data is required for the model to learn the explicit or implicit relationship between input and output data. The training of the MLP is conducted through the minimization of the difference between the prediction of the model and the desirable target value. The weights are usually optimized through the standard Levenberg–Marquardt back-propagation algorithm (Levenberg 1944; Marquardt 1963). The computation of gradients of the activation functions is involved in the optimization process, and therefore the back-propagation may favor the activation function whose gradient is of simpler form due to computation efficiency. The gradients of $\tan h(x)$ and RELU are given in Eqs. (5) and (6), respectively. Although the $\tan h(x)$ function is widely adopted in regression problems, the easy computation of the gradient of RELU has made RELU popular to use in recent years. After the training stage, the MLP model is evaluated through a test data set that was not included in the training process

$$f'(x) = 1 - \left(\frac{1-e^{-2x}}{1+e^{-2x}}\right)^2 = 1 - \tan h^2(x) \tag{5}$$

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In this study, the vibration signals (i.e., displacement, velocity, excitation, and acceleration time history) of the structures obtained from either numerical simulations or experimental data were used as input and output of the MLP model. The choice of inputs, outputs, and MLP configuration are discussed in the following sections.

Convolutional Neural Network

CNN, best known for its ability to extract features from raw data automatically (LeCun et al. 2015), usually consists of layers of convolution, activation, and pooling, as well as fully connected layers. The major differences between MLP and CNN are the convolution and pooling layers. To illustrate how the convolution layer works, Fig. 2 shows an example 4×3 input convolved with two kernels of size 3×3. The kernels slide over the whole input signal with a specified stride size (e.g., the stride size is 1 in Fig. 2), perform

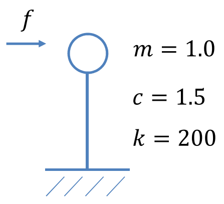
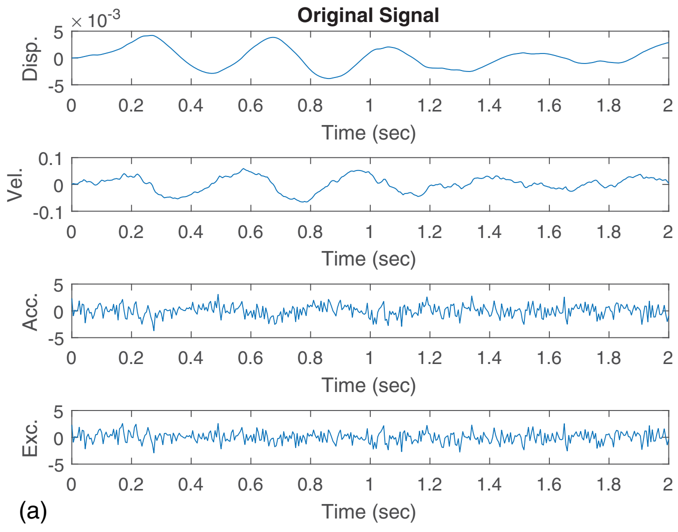


Fig. 6. Illustration of the SDOF system.

pointwise multiplications, and sum up all the values to obtain the output. For instance, the output of the first kernel is $\alpha = 1\times a + 2\times b + 3\times c + 4\times d + 5\times e + 6\times f + 7\times g + 8\times h + 9\times i$, and $\beta = 4\times a + 5\times b + 6\times c + 7\times d + 8\times e + 9\times f + 10\ \times g + 11\times h + 12\times i$, resulting a 2×1 output vector. The second kernel does the same operation but with different weight values, and hence the total output is of size 2×2. Fig. 3 demonstrates convolution operations. The 6×3 input is convolved with two kernels with size of 3×3, resulting in a 4×2 output. The dashed window and dotted window indicate the operations performed by Kernels 1 and 2, respectively. Typically, the output of each kernel is referred to as the feature map or feature channel.

Depending on the sizes of the input and kernel, a large number of feature maps could be obtained from the convolution layer. The pooling layer subsamples the feature map and extracts the dominant information in the map. Fig. 4 shows an example of a maximum pooling layer. The maximum pooling operation extracts the maximum value in the specified filter window and hence reduces the size of the feature map. For instance, in Fig. 4, the maximum value of six in the 2×2 window is extracted, and the pooling filter slides over the whole feature map with a stride of size 2 to obtain the 2×2 output.

Fig. 5 shows the configuration of a typical CNN network. In Fig. 5, the input data are convolved with three 3×3 filters.



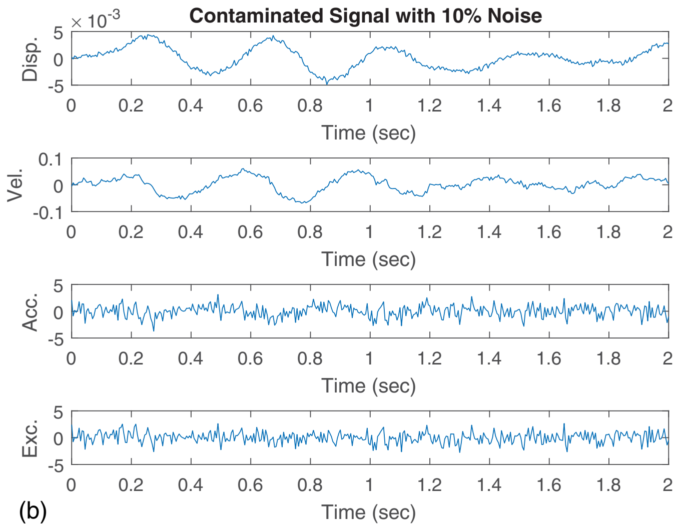
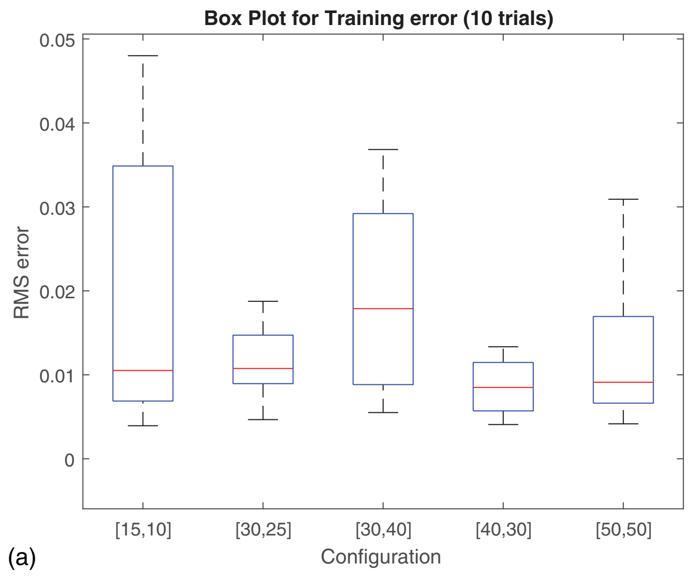


Fig. 7. Vibration signals [i.e., displacement (Disp.), velocity (Vel.), acceleration (Acc.), and excitation (Exc.)] of the linear SDOF system: (a) original signals; and (b) signals contaminated with 10% noise.



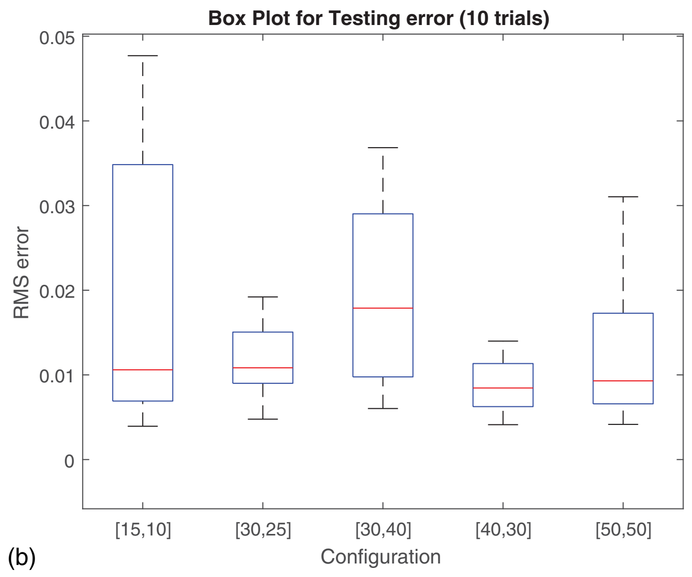


Fig. 8. Error variation for different MLP configurations: (a) training error; and (b) test error.

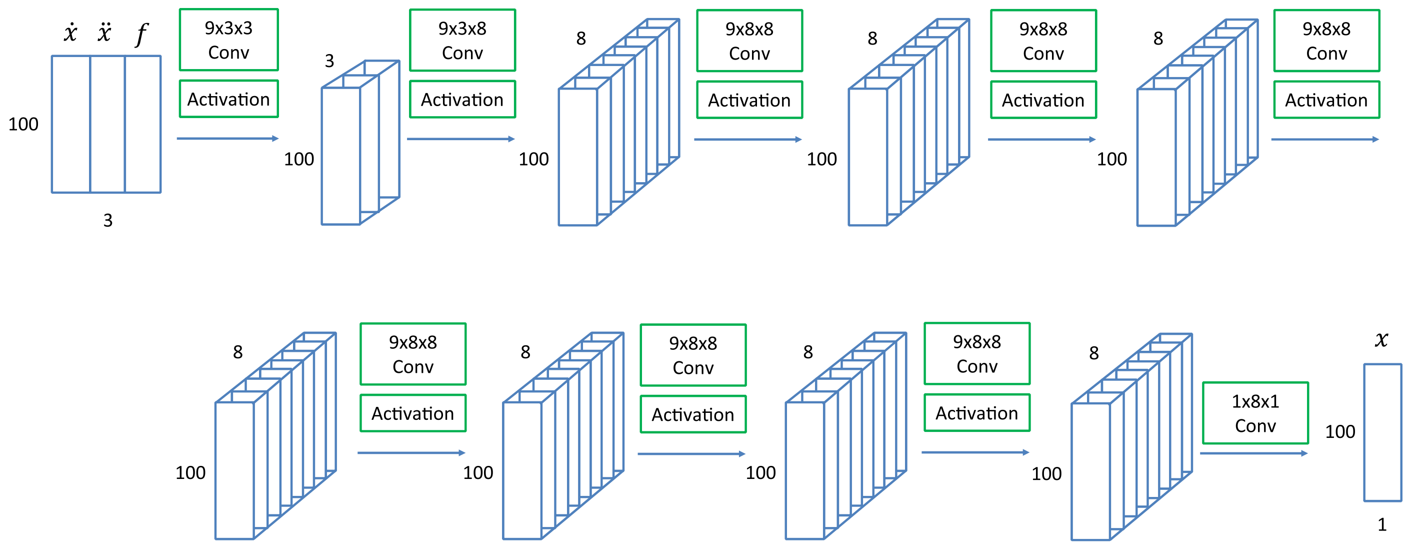


Fig. 9. CNN configuration for SDOF system.

During the convolution, the filter scans through the input data with a user-specified stride size (Cha et al. 2017). The inputs within the filter window are multiplied with the weights (e.g., w_{ij}^1) and summed to obtain the feature value in the next layer. Therefore, all the inputs share the same weights to form one feature map. As shown in Fig. 5, three feature maps are obtained after the input layer is convolved with three filters. The pooling layer serves as a subsampling mechanism, which is used to reduce the output dimensionality but keep the most salient information at the same time. This information is then passed into the fully connected layer to conduct the classification or regression tasks. The configuration in Fig. 5 is simply for illustration purposes. For deep CNNs, the network usually consists of multiple convolution and pooling layers stacked together to form a large network. Although pooling layer is usually required for image classification tasks, it may not be necessary in the context of regression problem because it only extracts the dominant information and eliminates the details that might be crucial for the regression problem. Similar to MLPs, in order to introduce nonlinearity in the system, activation functions are incorporated into CNNs. In this study, there are two activation functions considered [$\tan h(x)$ in Eq. (3) and RELU in Eq. (4)], and the pooling layer is removed from the network.

Similar to the training of MLP, the CNN model is established through the training and the test data sets. The weights in the filters are optimized through the minimization of the difference between the prediction of the model and the desirable target value. The optimization algorithm is the back-propagation approach (LeCun et al. 1998). After training, the CNN model is evaluated through the test data set.

The differences between the configurations of MLP and CNN provide insights for why CNN is potentially more suitable for response estimation in structure dynamics. The convolution layers in

Table 1. RMS error: MLP-tan h results ($\dot{x}$, $\ddot{x}$, $f \Rightarrow x$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.012	0.012	—	—
1	0.037	0.037	0.035	0.035
2	0.073	0.073	0.069	0.069
5	0.181	0.181	0.170	0.170
10	0.356	0.356	0.332	0.332
30	0.812	0.812	0.730	0.730

Table 2. RMS error: CNN-tan h results ($\dot{x}$, $\ddot{x}$, $f \Rightarrow x$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.027	0.026	—	—
1	0.031	0.030	0.029	0.027
2	0.040	0.039	0.031	0.030
5	0.076	0.075	0.044	0.043
10	0.146	0.146	0.068	0.067
30	0.380	0.379	0.135	0.134

Table 3. RMS error: MLP-RELU results ($\dot{x}$, $\ddot{x}$, $f \Rightarrow x$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
1	0.007	0.007	—	—
1	0.049	0.049	0.047	0.047
2	0.072	0.072	0.068	0.068
5	0.176	0.176	0.165	0.165
10	0.356	0.356	0.332	0.332
30	0.817	0.817	0.736	0.736

Table 4. RMS error: CNN-RELU results ($\dot{x}$, $\ddot{x}$, $f \Rightarrow x$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.025	0.024	—	—
1	0.029	0.029	0.026	0.026
2	0.044	0.043	0.036	0.036
5	0.081	0.081	0.052	0.052
10	0.146	0.146	0.069	0.069
30	0.381	0.381	0.139	0.138

CNN may serve as filters to remove noise more efficiently. Moreover, contrary to MLP, CNN is capable of dealing with various sizes of inputs while avoiding drastically increasing the number of weights in the network. For instance, in a MLP with one hidden layer of 15 nodes, the number of weights will increase from 45 to 1,500 if the input size changes from 3 to 100. In contrast, the number of CNN weights will only depend on the size of convolution kernels, and hence it is not sensitive to changes in input size.

This study uses vibration signals (i.e., the displacement, velocity, excitation, and acceleration time history) of the structures obtained from either numerical simulation or experimental data as the input and output of the CNN model. The choice of inputs, outputs, and CNN configuration are discussed in detail in the following sections.

Single-Degree-of-Freedom System

In this section, MLP and CNN are used to estimate the response of a linear and a nonlinear SDOF system adopted from Masri et al. (1996). The performances of MLP and CNN with respect to noise-contaminated input and output signals are compared and discussed. For the linear SDOF system, there are two cases of input–output relationship considered: (1) use of acceleration, velocity, and excitation to estimate displacement; and (2) use of only excitation to estimate acceleration. For the nonlinear SDOF system, the following input–output relationship is considered: use displacement and velocity to predict restoring force. In this study, five noise levels (i.e., 1%, 2%, 5%, 10%, and 30%) are added to the signal to investigate the robustness of MLP and CNN against noisy data.

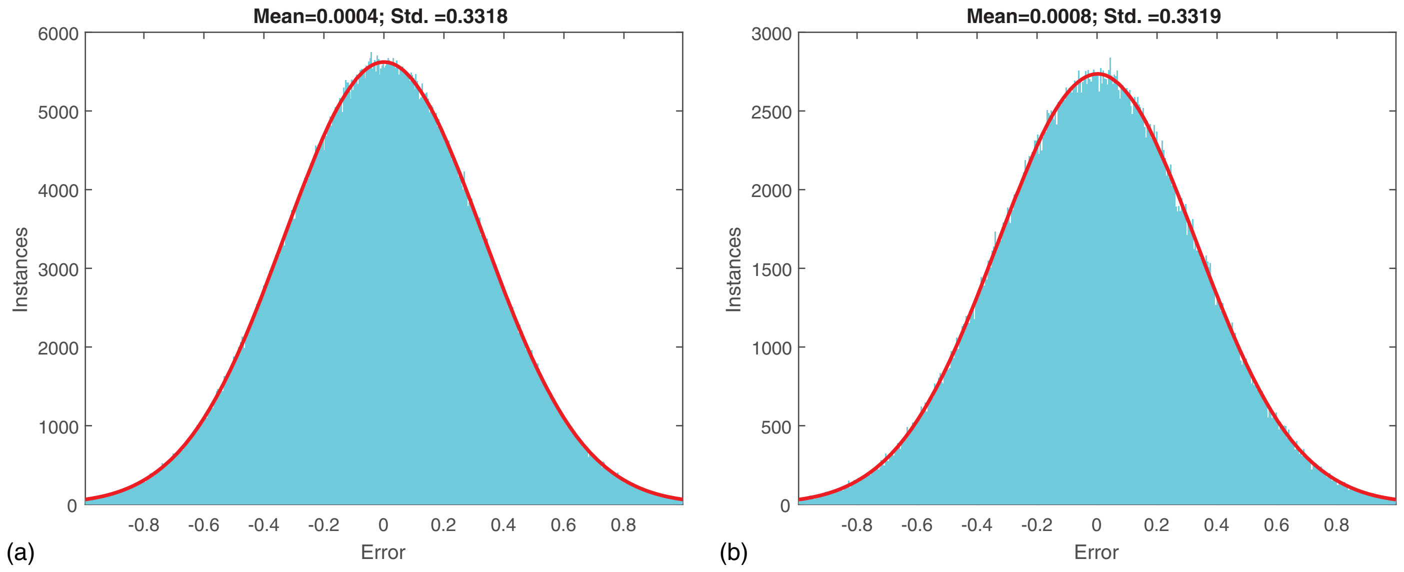


Fig. 10. MLP error histogram for tan h with 10% noise: (a) training error histogram; and (b) test error histogram.

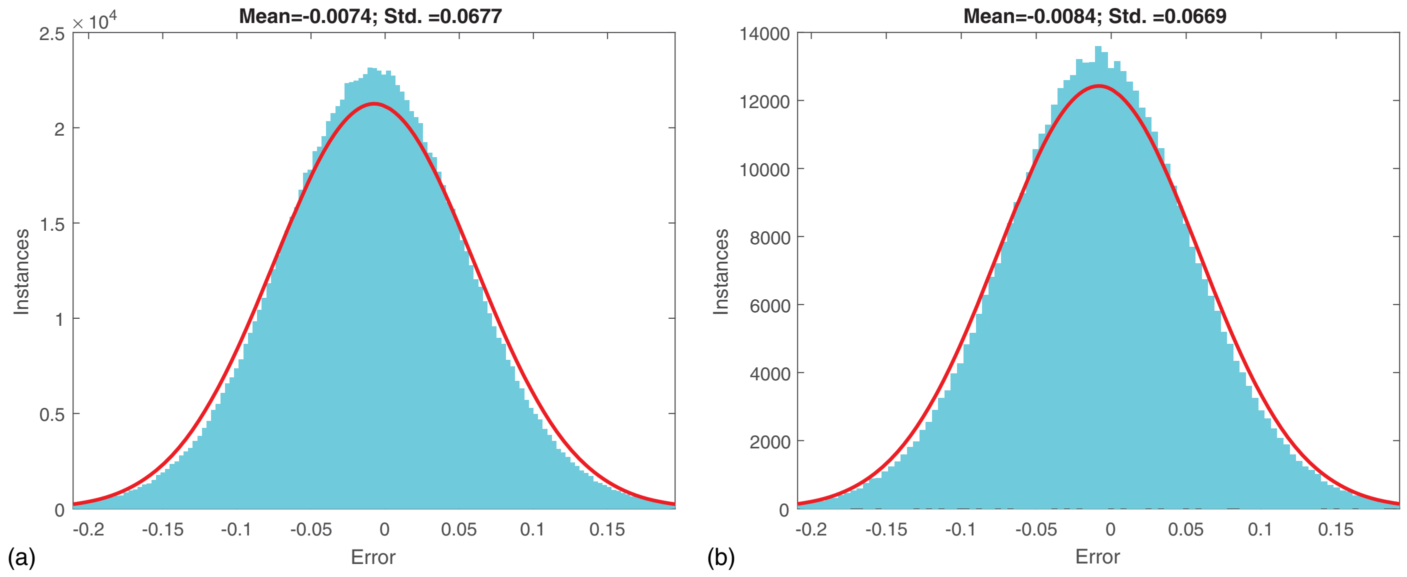


Fig. 11. CNN error histogram for tan h with 10% noise: (a) training error histogram; and (b) test error histogram.

Linear SDOF System

Consider the following linear SDOF system:

$$m\ddot{x} + c\dot{x} + kx = f \tag{7}$$

where $m = 1.0$ is the mass; $k = 200$ is the stiffness; $c = 1.5$ is the damping, and f = excitation. The input excitation is white noise with zero mean and variance of 1. Fig. 6 illustrates this SDOF system. The sampling frequency and data length is set to 200 Hz and 8,300 s, respectively. Therefore, a total of 1,660,000 data points are generated using the standard Newmark-Beta (Newmark 1959) numerical integration method. Fig. 7 depicts an example of the generated original vibration signals as well as the 10% noise-contaminated signals.

Preliminary Test for MLP Configuration

In this section, a preliminary test is conducted to determine the appropriate configuration for MLP. Masri et al. (1996) selected two hidden layers with 15 and 10 hidden nodes, denoted as [15,10] as the configuration for MLP. To investigate the sensitivity of different configurations, five configurations (i.e., [15,10], [30,25], [30,40], [40,30], [50,50]) were selected to perform 10 repeated trials for each configuration. The preliminary test used acceleration, velocity, and excitation at time k to estimate displacement at time k [i.e., $\dot{x}(k)$, $\ddot{x}(k)$, $f(k) \Rightarrow x(k)$, pointwise estimation]. The data sets were divided randomly into a 70% training set and 30% test set. Before training, the output signals were scaled to make it comparable with the inputs, and the inputs are normalized to $[-1, 1]$. No noise was added for the preliminary test. Root-mean square (RMS) error indicator was selected for performance evaluation in this study.

Fig. 8 shows the variations and box plots of training and testing errors of the five MLP configurations using the 10 repeated trials. The training and testing error of each configuration are almost identical, which demonstrates the capability of generalization achieved

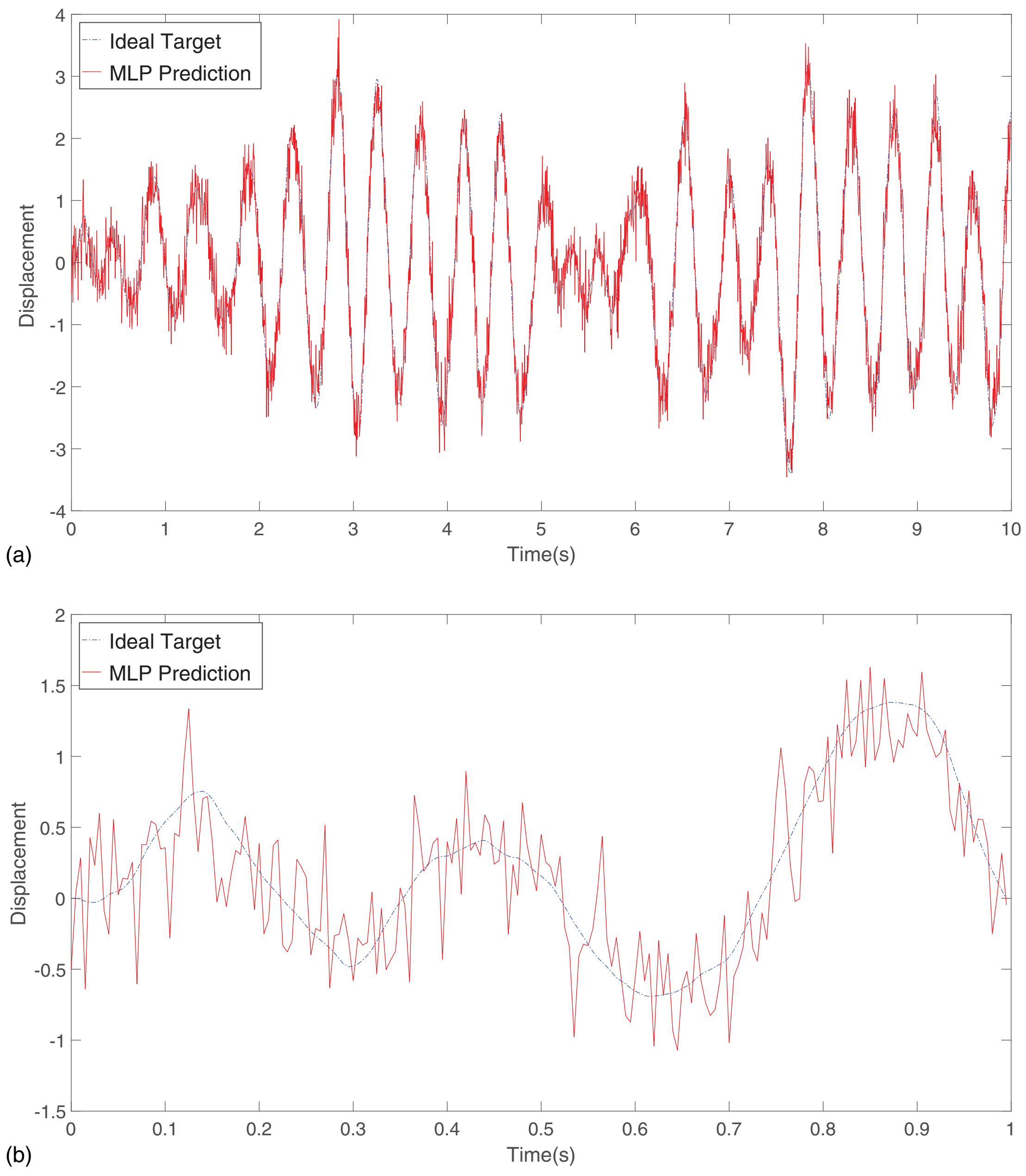


Fig. 12. MLP prediction versus ideal target (tan h with 10% noise): (a) 10 s of the estimated displacement; and (b) first 1 s of estimated response.

by MLP. Fig. 8 clearly indicates that [40,30] outperforms other configurations with a smaller variation and lower median of RMS error for both training and test data sets. As a result, this study uses [40,30] as the MLP configuration to conduct pointwise prediction for SDOF systems.

CNN Configuration for SDOF System

Fig. 9 depicts the proposed CNN configuration for SDOF system. As mentioned previously, the pooling layer is deployed in this study for conducting response estimation. The input to the network is three 100×1 vectors, which stand for velocity, acceleration, and excitation, respectively. The output of network is a 100×1 displacement vector. The first convolution layer is designed to be of size $9 \times 3 \times 3$. The intuition behind this is that the convolution layer can act as a filter to remove the noise from the input signals. There was no overlap between each sample, and hence a total of $1{,}660{,}000/100 = 16{,}600$ samples were used for training and testing. The data sets were divided randomly into 70% training set and 30% test set. Before training, the outputs were scaled to make them comparable with the inputs, and the inputs were normalized to $[-1, 1]$. For the case using excitation to predict acceleration ($f \Rightarrow \ddot{x}$), the input to the network was a 100×1 vector, and the first and second convolution layers in Fig. 9 were replaced with two filters size of $9 \times 1 \times 1$ and $9 \times 1 \times 8$, respectively.

Results and Discussions

For real-world problems, the presence of noise is inevitable during the data-collection process. The source of noise may come from the sensor itself, ambient vibration, or even temperature variations. It is impossible to have a clean (i.e., ideal) signal for real-world

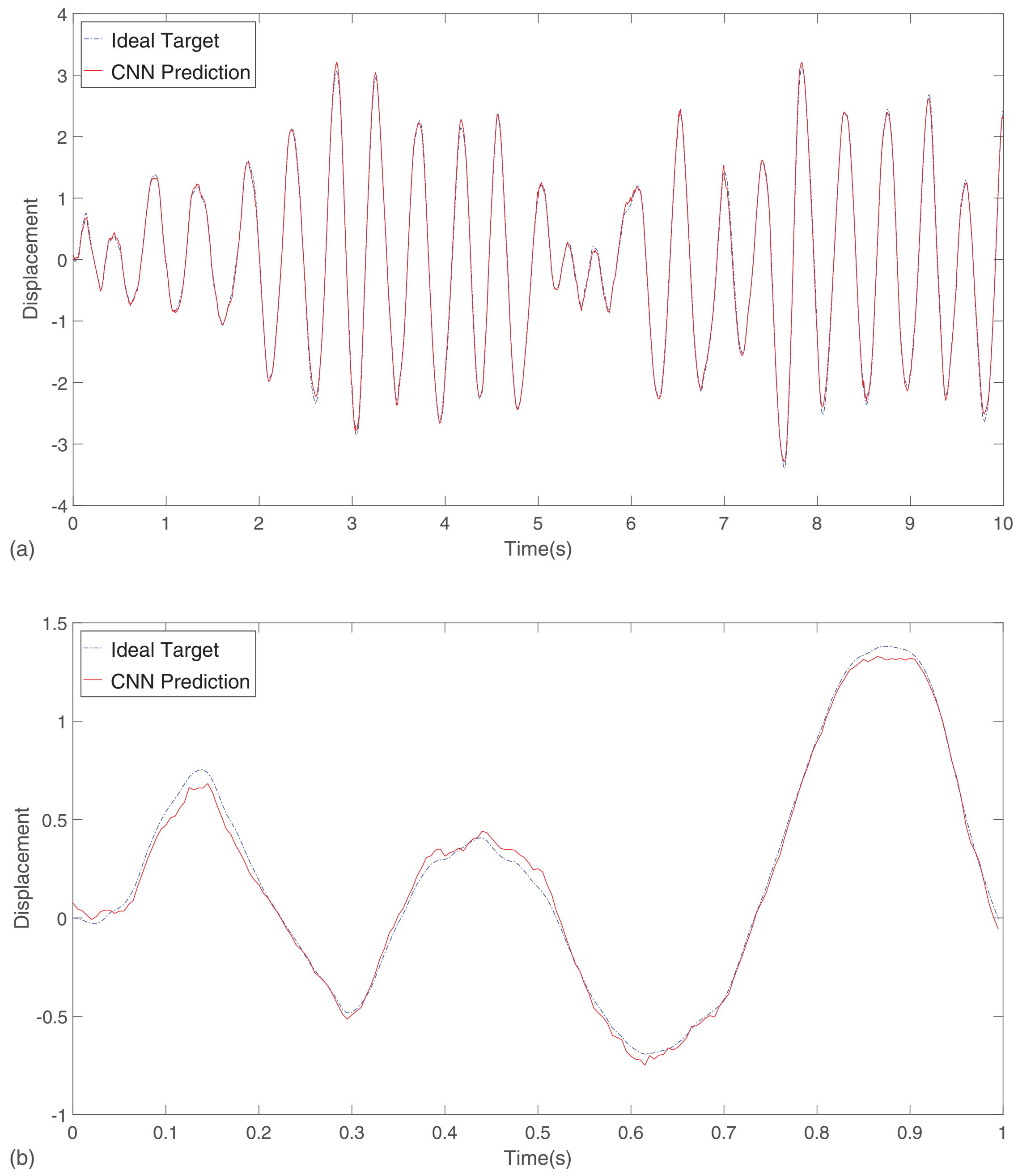


Fig. 13. CNN prediction versus ideal target (tan h with 10% noise): (a) 10 s of the estimated displacement; and (b) first 1 s of the estimated response.

applications. In this study, five noise levels (i.e., 1%, 2%, 5%, 10%, and 30%) were considered to simulate practical situations. The noise was added independently to each vibration signal, and each noise signal was obtained by multiplying the noise level with the standard deviation of each vibration signal (Cury and Crémona 2012). Each network was trained with the noisy input and noisy output data.

Tables 1–4 provide the RMS error of using velocity, acceleration, and excitation to predict the displacement, for MLP-tan h, CNN-tan h, MLP-RELU, and CNN-RELU, respectively. The second and third columns give the error with respect to the noisy target, and the fourth and fifth columns list the error with respect to the ideal target. In general, both algorithms achieve good estimations with a low RMS error even if the signal is contaminated with noise. The performances of the two activation functions, tan h and RELU, are similar. Also, the training and testing RMS errors for both MLP and CNN are very similar, which means that there is no overfitting for the two models (overfitting occurs when the learning model performs well on the training data but poor on the test data). Because the RMS errors with respect to the ideal target are always smaller than the RMS errors with respect to the noisy target, this demonstrates that both MLP and CNN are capable of learning the true response given noisy input and output data. For the ideal case (i.e., 0% noise), MLP outperforms CNN. However, for noisy cases, CNN achieves lower RMS errors for all cases. Compared with the performance of MLP, Tables 1 and 2 demonstrate that CNN reduces the test RMS error for 1%, 2%, 5%, 10%, and 30% noisy cases by 22%, 56%, 75%, 80%, and 82%, respectively. This indicates that CNN is more robust against noise.

As a representative case, Figs. 10 and 11 show the error distribution of 10% noise data, tan h, for MLP and CNN, respectively. The gray solid line is the fitted normal distribution curve superimposed on the histogram for comparison reasons. The test error standard deviation for CNN is 0.0669, which is five times lower than the test error standard deviation for MLP (i.e., 0.3319). In other words, the error distribution of CNN is more centered to zero than that of MLP. Figs. 12 and 13 depict the prediction of 10% noisy data with tan h activation function for MLP and CNN with respect to the ideal target, respectively. The dashed line indicates the ideal target whereas the solid line is the prediction from the learning model. The MLP prediction captures the wave shape of the ideal target but there exist some small unwanted high-frequency oscillations. On the contrary, the CNN predictions almost match with the ideal target. Fig. 14 shows the test RMS error variation versus different noise levels for both MLP and CNN. As the noise level increases, the RMS error of CNN increases much more slowly than the MLP.

Another case considered in the linear SDOF system is to use excitation f to estimate the acceleration $\ddot{x}$. This is of particular interest because it is difficult to acquire information on displacement responses in real world. Tables 5–8 present the RMS error of using excitation to predict the acceleration for MLP-tan h,

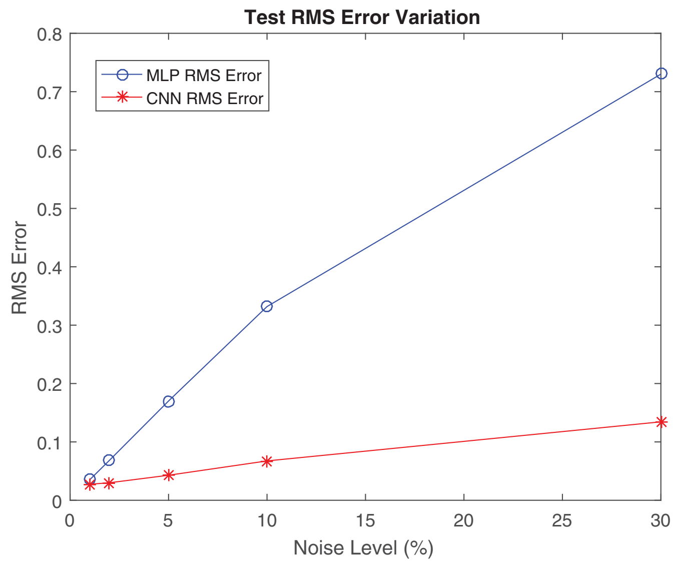


Fig. 14. Test RMS error variations for MLP and CNN ($\dot{x}$, $\ddot{x}$, $f \Rightarrow x$; tan h).

Table 5. RMS error: MLP-tan h results ($f \Rightarrow \ddot{x}$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.531	0.531	—	—
1	0.532	0.532	0.532	0.532
2	0.529	0.529	0.529	0.528
5	0.531	0.531	0.528	0.528
10	0.523	0.523	0.513	0.513
30	0.660	0.660	0.582	0.582

Table 6. RMS error: CNN-tan h results ($f \Rightarrow \ddot{x}$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.492	0.490	—	—
1	0.493	0.492	0.493	0.491
2	0.490	0.489	0.490	0.489
5	0.492	0.490	0.489	0.488
10	0.488	0.486	0.477	0.476
30	0.632	0.632	0.551	0.549

Table 7. RMS error: MLP-RELU results ($f \Rightarrow \ddot{x}$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.531	0.531	—	—
1	0.532	0.532	0.532	0.532
2	0.529	0.529	0.529	0.529
5	0.531	0.530	0.528	0.528
10	0.523	0.523	0.513	0.513
30	0.660	0.660	0.582	0.582

Table 8. RMS error: CNN-RELU results ($f \Rightarrow \ddot{x}$)

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.497	0.495	—	—
1	0.498	0.496	0.498	0.496
2	0.496	0.494	0.495	0.494
5	0.497	0.495	0.494	0.493
10	0.493	0.491	0.482	0.481
30	0.637	0.637	0.557	0.555

CNN-tan h, MLP-RELU, and CNN-RELU, respectively. In this case, there is still no overfitting for either MLP or CNN models because the RMS errors for training and testing are similar. Also, there is no obvious discrepancy between the performances of tan h and RELU activation functions. The RMS error with respect to the ideal target is slightly smaller than the error with respect to the noisy target. Compared with the performance of MLP, Tables 5 and 6 indicate that CNN reduces the test RMS error for 1%, 2%, 5%, 10%, and 30% noisy cases by approximately 6%–8%. Although CNN outperforms MLP, the reduction in the RMS error is not as significant as the case of using $\dot{x}$, $\ddot{x}$, f to predict x.

As a representative case, Figs. 15 and 16 depict the error distribution of 10% noise data, using tan h activation function, for MLP and CNN, respectively. The gray solid line is the fitted normal distribution curve superimposed on the histogram for comparison reasons. The test error standard deviation of CNN is 0.4756, which is lower than the test error standard deviation of MLP (i.e., 0.5133).

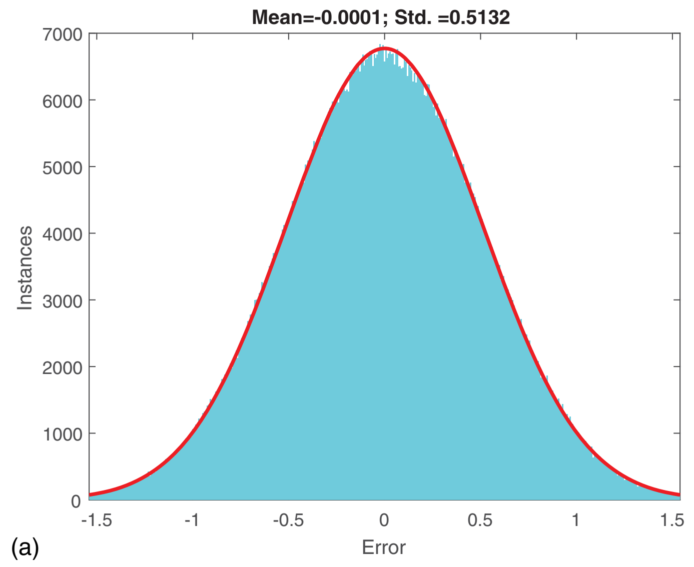
Figs. 17 and 18 present the prediction of 10% noise data with tan h activation function for MLP and CNN with respect to the ideal target, respectively. The dashed line indicates the ideal target, while the gray line is the prediction from the learning model. In this case, the CNN matches slightly better with the ideal target compared with the prediction of MLP by a 7% reduction of test RMS error. Fig. 19 shows the test RMS error variation versus different noise levels for both MLP and CNN. The RMS error for CNN is always lower than MLP for all noise levels.

Nonlinear SDOF System

System Description

Consider the following nonlinear SDOF system:

$$g(x, \dot{x}) = ax + bx^3 + c\dot{x} \qquad (8)$$



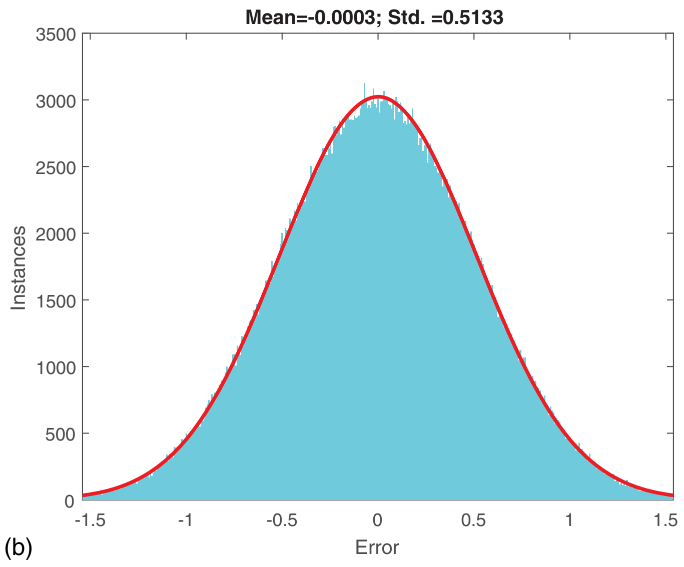
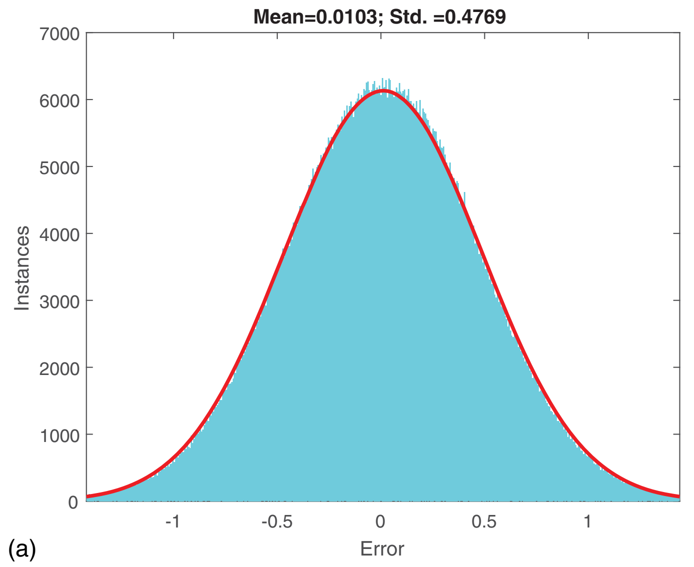


Fig. 15. MLP error histogram for 10% noise: (a) training error histogram; and (b) test error histogram.



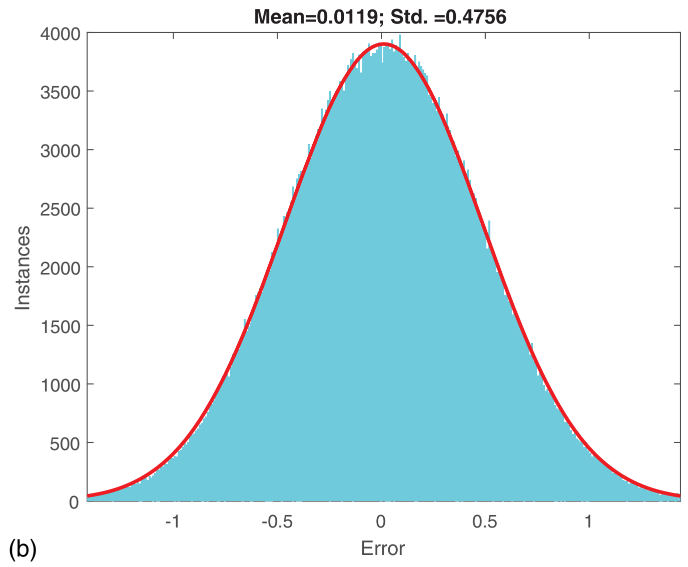


Fig. 16. CNN error histogram for 10% noise: (a) training error histogram; and (b) test error histogram.

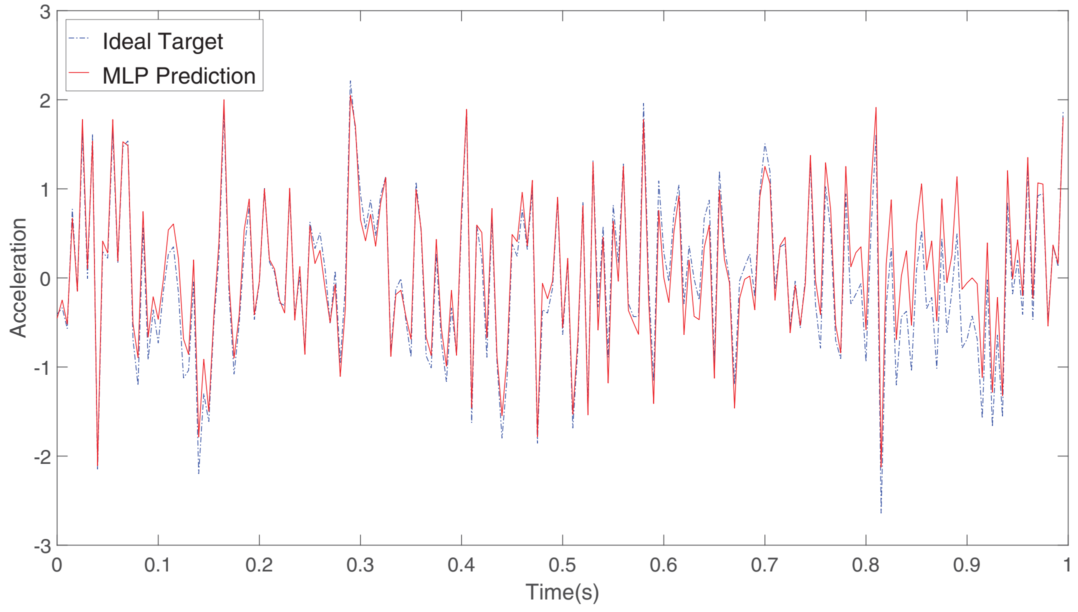


Fig. 17. MLP prediction versus ideal target (10% noise).

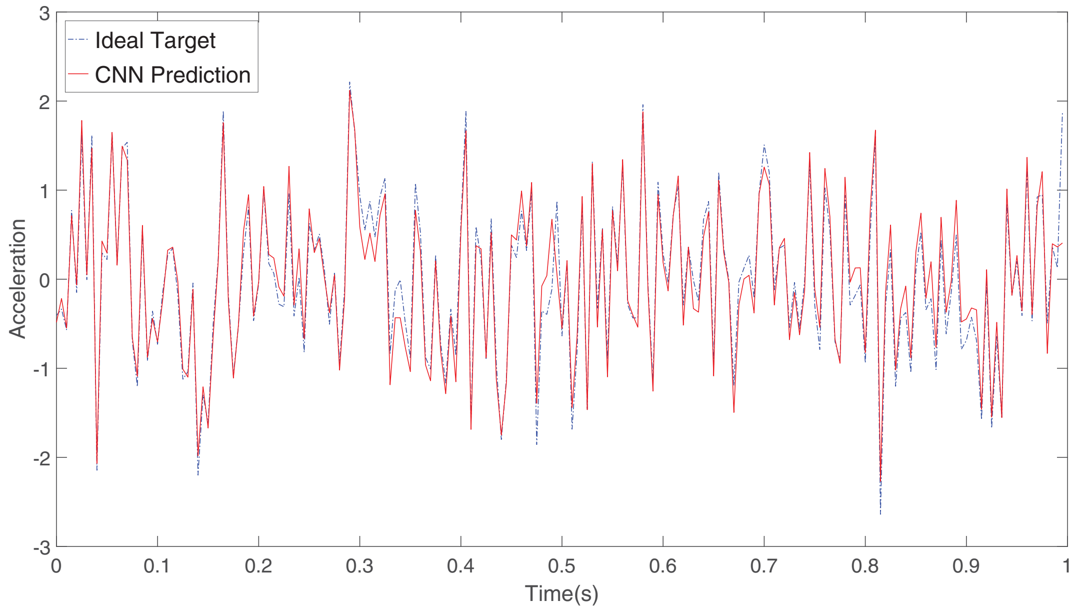


Fig. 18. CNN prediction versus ideal target (10% noise).

where $a = (2\pi)^2$ is the linear stiffness coefficient; $b = 10$ is the nonlinear stiffness coefficient; $c = 1.25$ is the damping coefficient; and $g(x, \dot{x})$ = restoring force. The input excitation is the white-noise signal with zero mean and variance large enough to drive the system into the nonlinear phase. Similar to the linear SDOF system, the sampling rate and the data length are 200 Hz and 8,300 s, respectively. The Runge-Kutta method is used to generate the numerical vibration responses. Fig. 20 plots the restoring force versus the displacement and the velocity time history. Nonlinear response behavior is observed from the response plot.

In this case, the displacement and the velocity were used to predict the restoring force [i.e., $x, \dot{x} \Rightarrow g(x, \dot{x})$]. Similar to the linear SDOF system, configuration [40,30] was selected for MLP model. The input to MLP was the displacement and velocity at time k, and its output was the restoring force at time k. The data set was randomly divided into 70% training and 30% test data sets. Before training, the output was scaled to make it comparable with the inputs, and the inputs were normalized to $[-1, 1]$.

In terms of the CNN model, the input to the network is two 100×1 vectors, and the first and second convolution layer in Fig. 9 are replaced with two filters with size $9 \times 2 \times 2$ and $9 \times 2 \times 8$, respectively. There is no overlapping between each sample, and hence a total of $1{,}660{,}000/100 = 16{,}600$ samples were used for training and testing. The data sets were divided randomly into a

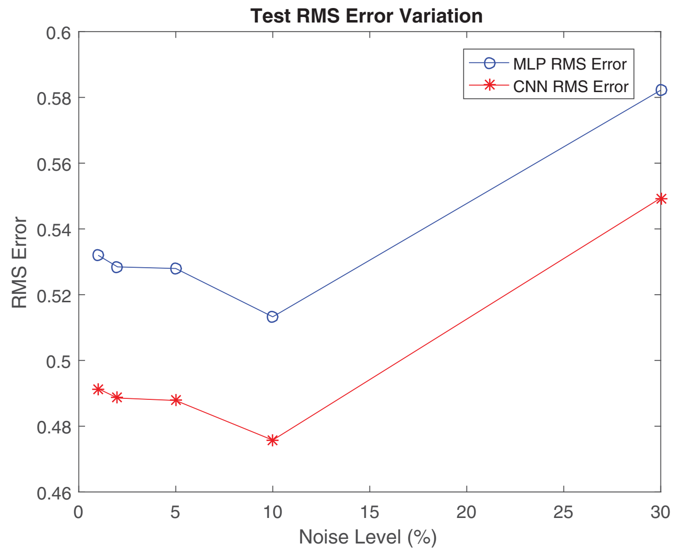


Fig. 19. Test RMS error variation ($f \Rightarrow \ddot{x}$).

70% training set and 30% test set. Before the training of CNN, the output was scaled to make it comparable with the inputs, and the inputs were normalized to $[-1, 1]$.

Results and Discussions

Five noise levels (i.e., 1%, 2%, 5%, 10%, and 30%) were considered for performance evaluation purposes. Tables 9–12 show the RMS error of using displacement and velocity to predict the restoring force for MLP-tan h, CNN-tan h, MLP-RELU, and CNN-RELU, respectively. In general, both MLP and CNN algorithms achieve small RMS errors for estimating the restoring force under the presence of noise. No overfitting phenomenon were observed for either MLP and CNN models because the RMS errors of training and test data sets are close. For the ideal case (i.e., noise level 0%), MLP outperformed CNN. For noise levels greater than 1%, CNN was superior to MLP with smaller RMS errors with respect to

Table 9. RMS error: nonlinear SDOF MLP-tan h results

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.060	0.059	—	—
1	0.105	0.100	0.094	0.089
2	0.158	0.158	0.127	0.127
5	0.391	0.392	0.313	0.313
10	0.752	0.753	0.581	0.581
30	2.155	2.152	1.609	1.606

Table 10. RMS error: nonlinear SDOF CNN-tan h results

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.213	0.179	—	—
1	0.186	0.166	0.180	0.159
2	0.163	0.137	0.133	0.100
5	0.308	0.292	0.200	0.173
10	0.521	0.514	0.208	0.188
30	1.500	1.501	0.451	0.439

Table 11. RMS error: nonlinear SDOF MLP-RELU results

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.251	0.250	—	—
1	0.104	0.104	0.093	0.092
2	0.249	0.249	0.231	0.230
5	0.399	0.398	0.323	0.322
10	0.788	0.788	0.626	0.626
30	2.181	2.175	1.644	1.640

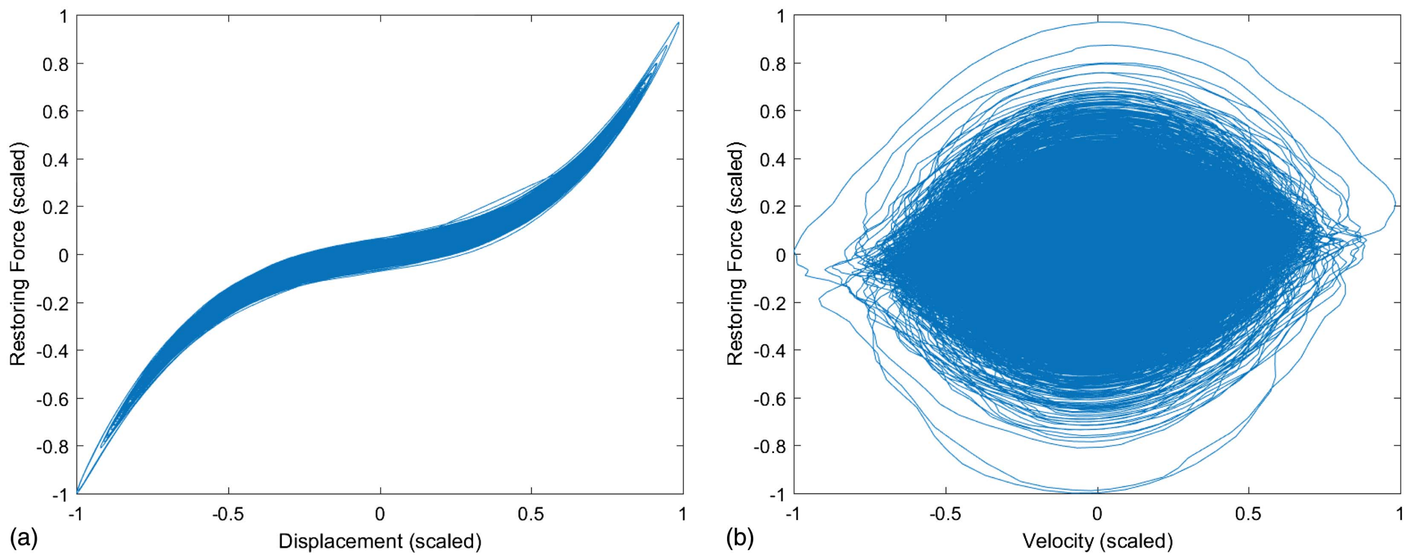


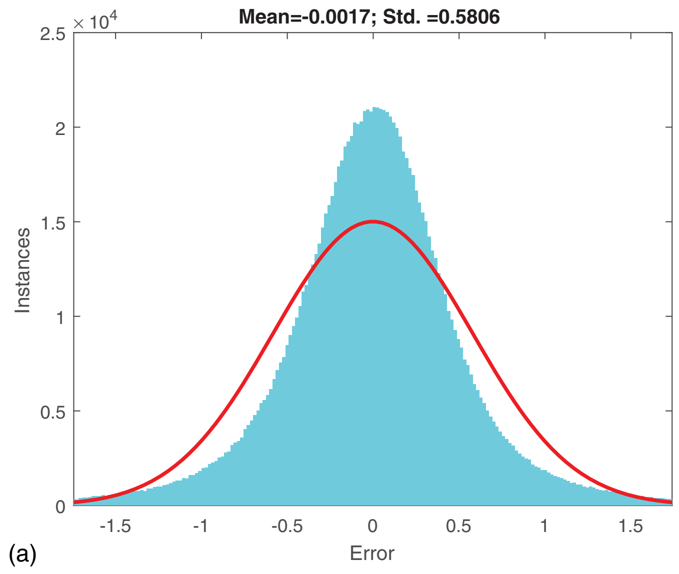
Fig. 20. (a) Nonlinear restoring force versus displacement; and (b) restoring force versus velocity.

Table 12. RMS error: Nonlinear SDOF CNN-RELU results

Noise (%)	Training	Testing	Training (with respect to ideal)	Testing (with respect to ideal)
0	0.252	0.217	—	—
1	0.140	0.140	0.132	0.132
2	0.150	0.150	0.117	0.117
5	0.279	0.279	0.152	0.151
10	0.517	0.519	0.198	0.200
30	1.500	1.504	0.451	0.450

the ideal target. Compared with the performance of MLP, Tables 9 and 10 indicate that CNN reduces the test RMS error for 2%, 5%, 10%, and 30% noisy cases by 21%, 45%, 68%, and 73%, respectively. According to the comparison between the RMS errors with respect to the noisy target and the ideal target, both MLP and CNN are capable of eliminating the effect of noise and learning to estimate the ideal target. The performances of tan h and RELU function are also close to each other.

As a representative case, Figs. 21 and 22 present the error distribution of 10% noise data with tan h as activation function for MLP and CNN, respectively. The solid gray line is the fitted normal distribution curve superimposed on the histogram for comparison reasons. The test error standard deviation for CNN is 0.1869, which is three times lower than the test error standard deviation of MLP (i.e., 0.5810). This means that the error distribution of CNN is more centered to zero than that of MLP. Figs. 23 and 24 show the response prediction of 10% noise data, tan h activation function, for MLP and CNN with respect to the ideal target, respectively. The gray line is the predicted response, and the dashed line is the ideal target. The CNN prediction almost matches with the ideal target



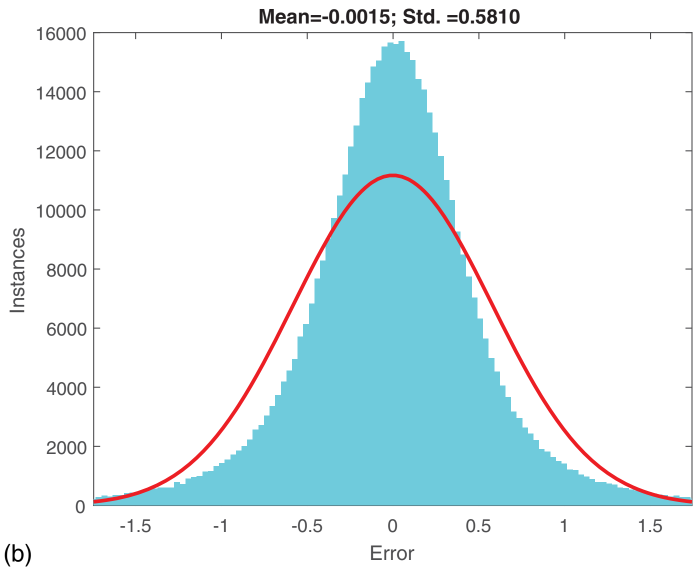
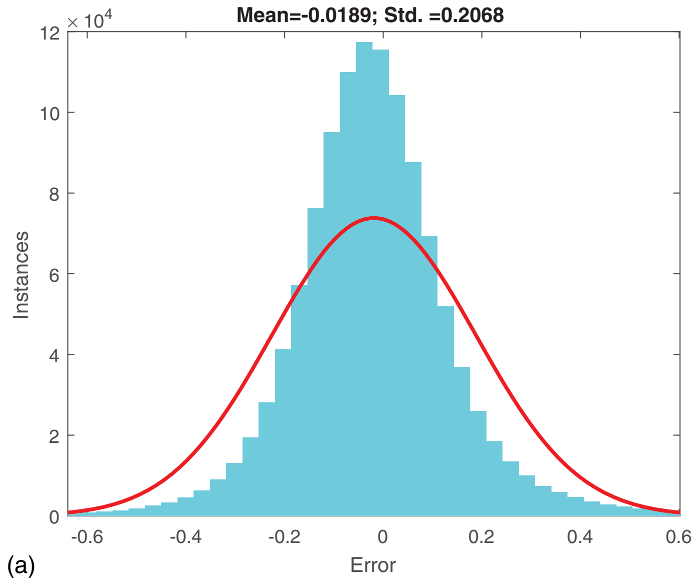


Fig. 21. MLP error histogram for tan h with 10% noise: (a) training error histogram; and (b) test error histogram.



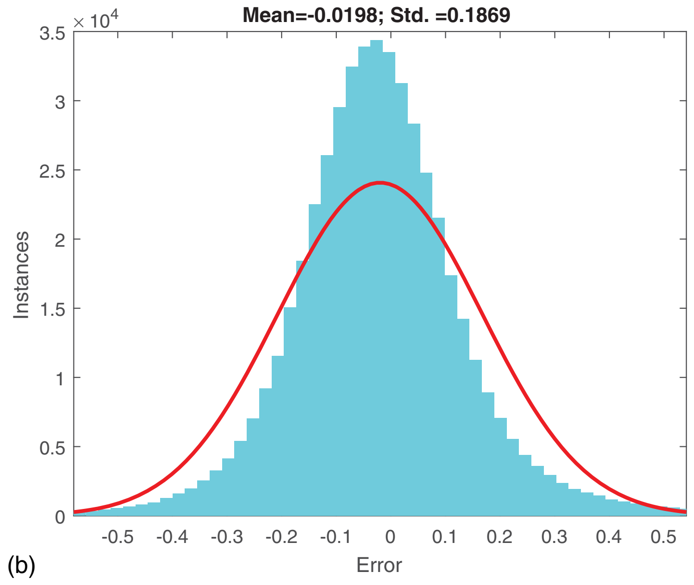


Fig. 22. CNN error histogram for tan h with 10% noise: (a) training error histogram; and (b) test error histogram.

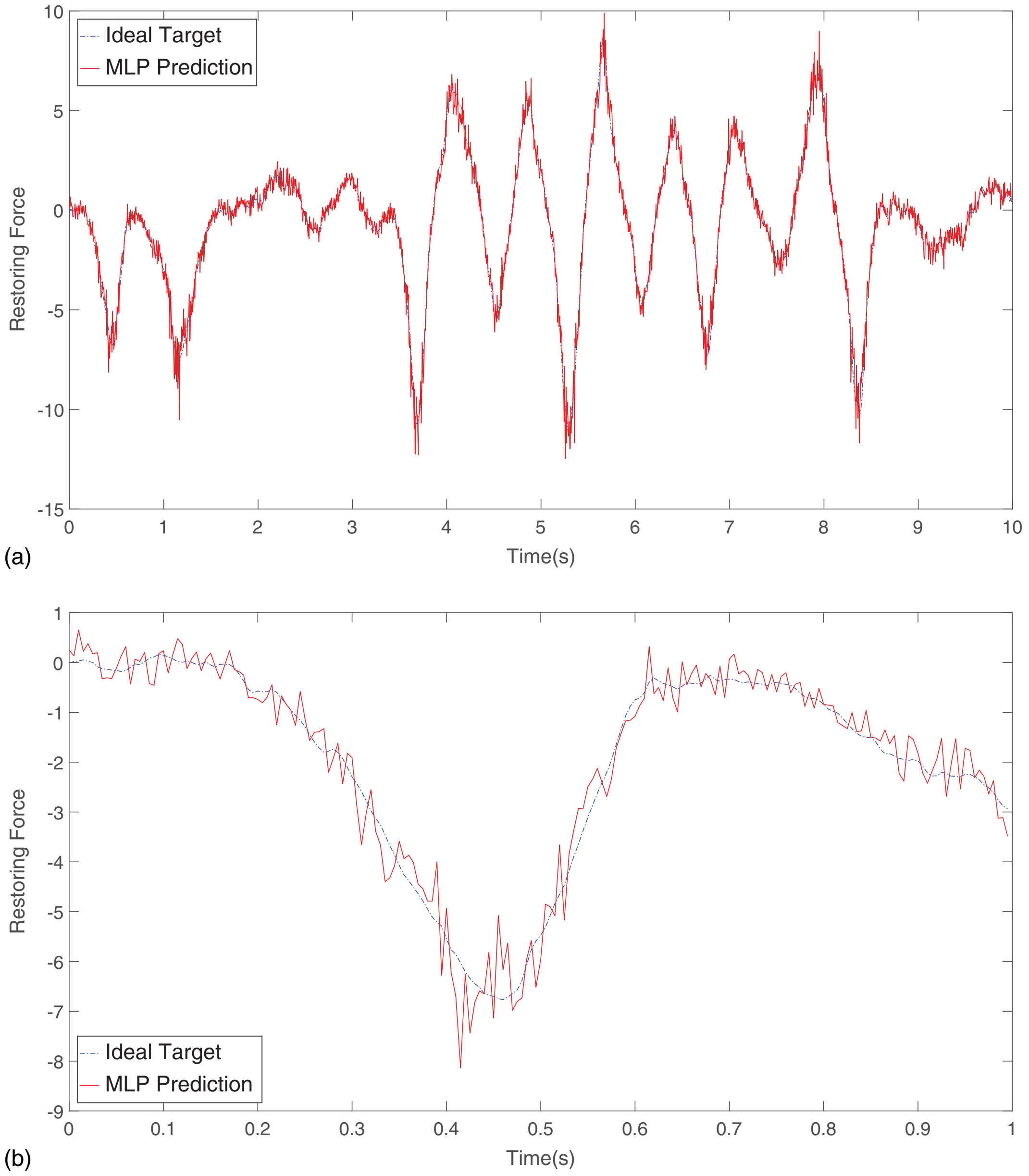


Fig. 23. MLP prediction versus ideal target (tan h with 10% noise): (a) 10 s of the estimated restoring force; and (b) first 1 s of the estimated response.

response whereas the MLP prediction captures the target wave shape but exhibits some undesirable high-frequency oscillations. Fig. 25 depicts the test RMS error variation versus different noise levels for both MLP and CNN. This figure indicates that the RMS errors of CNN increase much more slowly than the RMS errors of MLP as the noise level increases.

To further test the ability of the proposed CNN approach in generalization, the nonlinear SDOF responses were resimulated using a random excitation with length of 50 s. Similarly, the displacement, velocity, and restoring force responses were contaminated with 10% noise. The trained CNN model with 10% noisy measurements was used to predict the noise-free restoring force given the noisy displacement and velocity measurements. Fig. 26 shows the prediction from CNN and the target restoring force. The CNN model accurately captures the dynamic behavior of the system even when the responses are generated using a new excitation. The overall RMS error between CNN prediction and target response is 0.1706, which is close to the test RMS error of 0.188 indicated in Table 10. This further demonstrates the trained CNN model generalizes very well on new data. When applying new data, the scaling factors for input and output measurements must be identical with the scaling factors used during training stage.

Multidegree-of-Freedom System

In this section, vibration responses from a 3-story steel frame tested on a shake table are used to establish the MLP and CNN models to predict the dynamic behavior of the structural system. The objective is to predict the roof acceleration $\ddot{x}_3$ given only the information of the ground excitation f. The prediction from CNN model is further used to perform the system identification of the steel frame.

To account for the complexity in the real MDOF system, the input to both MLP and CNN models is a 3-s excitation signal, which is a 600×1 vector for a sampling frequency of 200 Hz. The output of the network is a 1×1 roof acceleration signal at

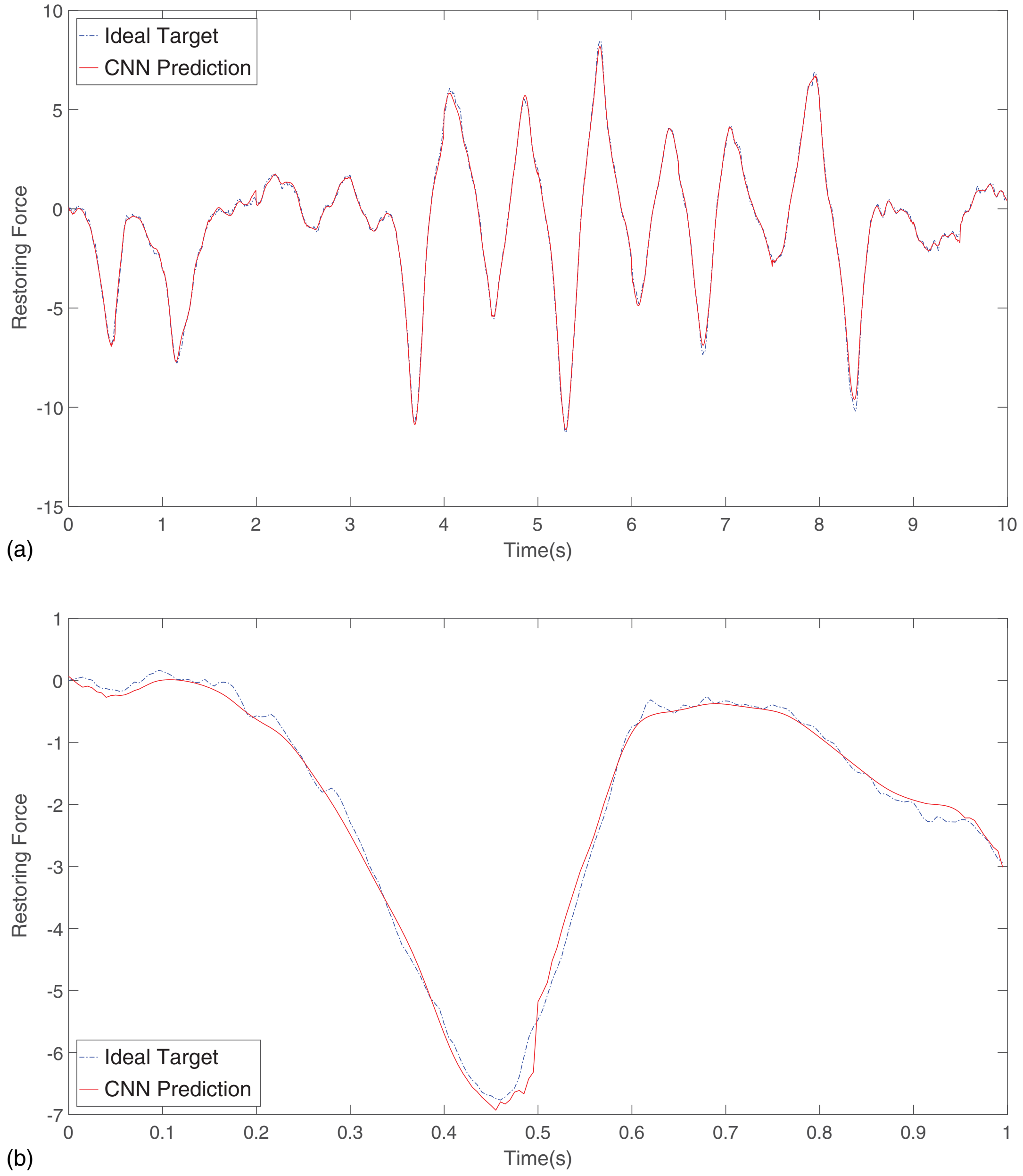


Fig. 24. CNN prediction versus ideal target (tan h with 10% noise): (a) 10 s of the estimated restoring force; and (b) first 1 s of the estimated response.

the end of the corresponding 3-s time interval. Each input sample has 599 points that overlap with each other. For instance, the input and output for the first sample are $f(600), f(599), \ldots, f(1)$, and $\ddot{x}_3(600)$, respectively. The input and output for the second sample are $f(601), f(600), \ldots, f(2)$, and $\ddot{x}_3(601)$, respectively. Therefore, the first 599 points of roof acceleration are discarded during the training and testing. The MLP configuration is selected as [40,30], and the proposed CNN configuration is shown in Fig. 27. The data set consists of 197,612 samples, and it was randomly divided into 85% training and 15% test data sets. Before training, the input and output were scaled to $[-1, 1]$. The tan h activation function was used in this case.

Shake-Table Test Setup

Fig. 28 shows the full-scale 3-story steel frame built at the National Center for Research on Earthquake Engineering (NCREE) in Taiwan. The frame was designed as a moment-resisting frame with each floor height equal to 3 m. The length, width, and thickness of the steel floor slab are 3, 2, and 3 cm, respectively. The dimensions of each beam and column is H150 × 150 × 7 × 10 mm. Accelerometers and LVDT sensors were installed at each floor slab to measure the floor acceleration and floor displacements, respectively. The steel frame was excited with the El Centro earthquake, random excitation, and the TCU076 ground motion recorded during the 1999 Chi-Chi earthquake in Taiwan. In this study, only the responses recorded in the transverse direction of the frame were used for the analysis.

Shake-Table Test Results

Table 13 presents the training and test RMS error for both MLP and CNN models. The RMS errors for CNN is 19.38% lower than the errors of MLP. Because the errors for training and testing are the

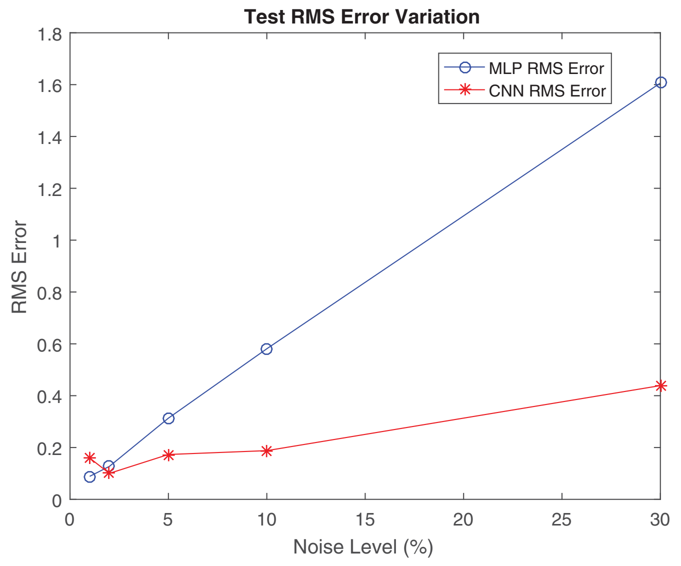


Fig. 25. Test RMS error variation.

same, there is no overfitting for both the MLP and CNN models. Figs. 29 and 30 show the error distribution for MLP and CNN predictions, respectively. The solid gray line is the fitted normal distribution curve superimposed on the histogram for comparison reasons. The test error standard deviation for CNN is 0.1042, which is 19% lower than the test error standard deviation of MLP (i.e., 0.1287). It is seen that the CNN prediction errors are more centered to zero than those of MLP. Figs. 31 and 32 show a 10-s predicted response versus target response for MLP and CNN, respectively. The dashed line is the target response whereas the gray line is the estimation from the prediction models. The CNN estimation matches slightly better with the target response.

System Identification

This section demonstrates that the prediction from the established CNN model can be further used to perform system identification. Fig. 33 shows the frequency response of the recorded NCREE roof acceleration data and the predicted roof acceleration from CNN. Table 14 presents the identified first three natural frequencies in the transverse direction from the NCREE measurements and CNN

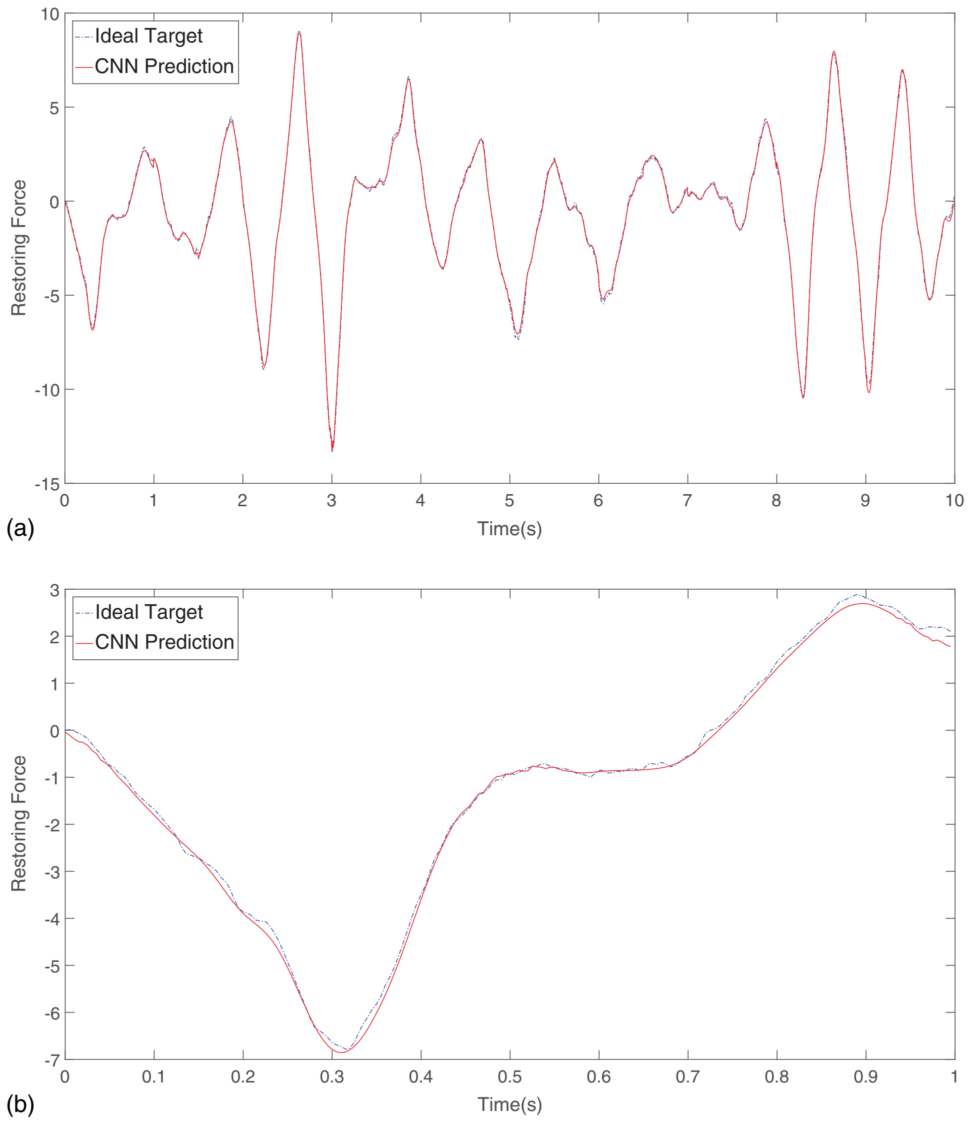


Fig. 26. CNN prediction versus ideal target using measurements from new excitation: (a) 10 s of the estimated restoring force; and (b) first 1 s of the estimated response.

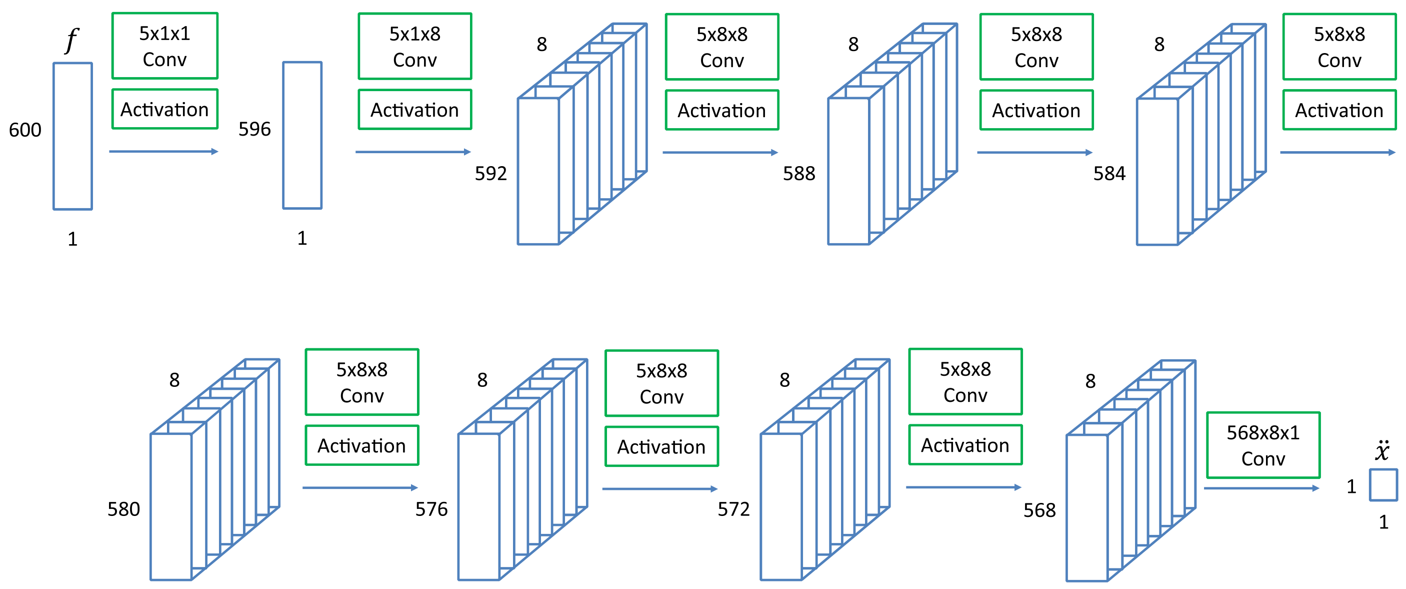


Fig. 27. CNN configuration for MDOF system.

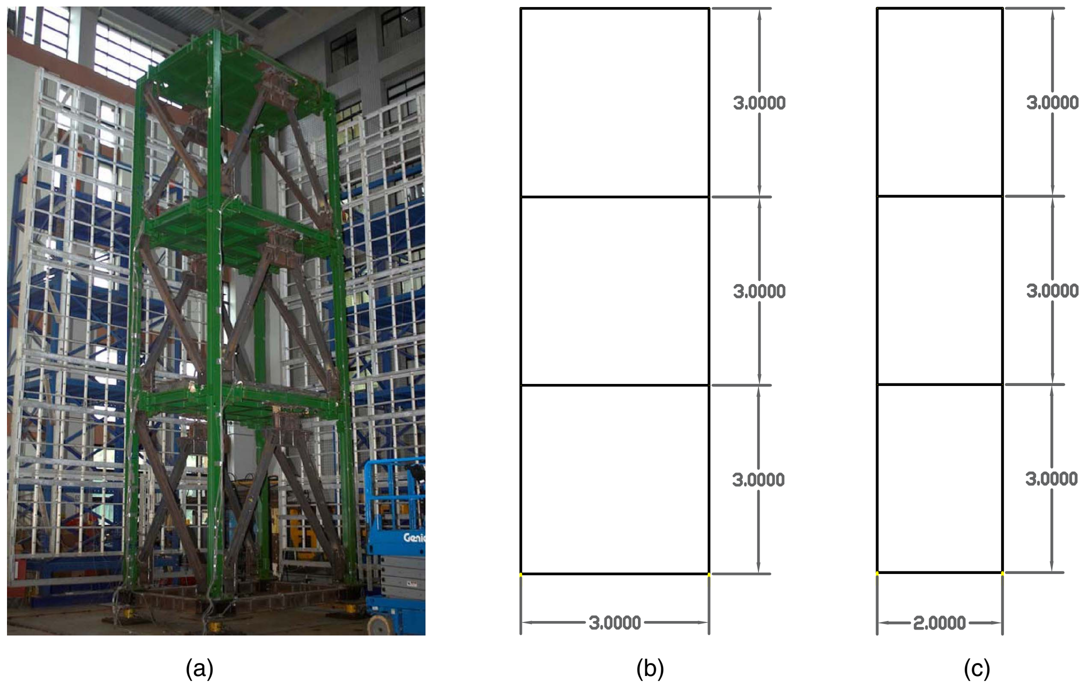


Fig. 28. Three-story steel frame at NCREE: (a) frame; (b) front view; and (c) side view (unit = meters). [Image (a) courtesy of NCREE.]

estimations. The identified mode frequencies from CNN are very close to the ones obtained from NCREE data. The identification error is below 8% for setting the NCREE data as the ground truth.

Table 13. RMS error: MLP and CNN

Method	Training	Testing
MLP	0.129	0.129
CNN	0.104	0.104

Physical Interpretation of Convolution Layer

Although the learning capability of conventional MLP has been demonstrated in several studies, how to interpret the physical meaning of the MLP model remains a challenging problem, and hence the MLP is often considered a black-box methodology. CNN, however, has a higher interpretability over the MLP algorithm. For instance, studies in image classifications have shown that the convolution layer in CNN acts as a feature detector to extract spatial invariant features like edges and color contrast (Krizhevsky et al. 2012). The convolution kernels become useful filters after the network is trained using large amounts of data. In this section, the physical interpretation of the convolution layer for regression problems is provided. The authors show that the integration operator can be approximated

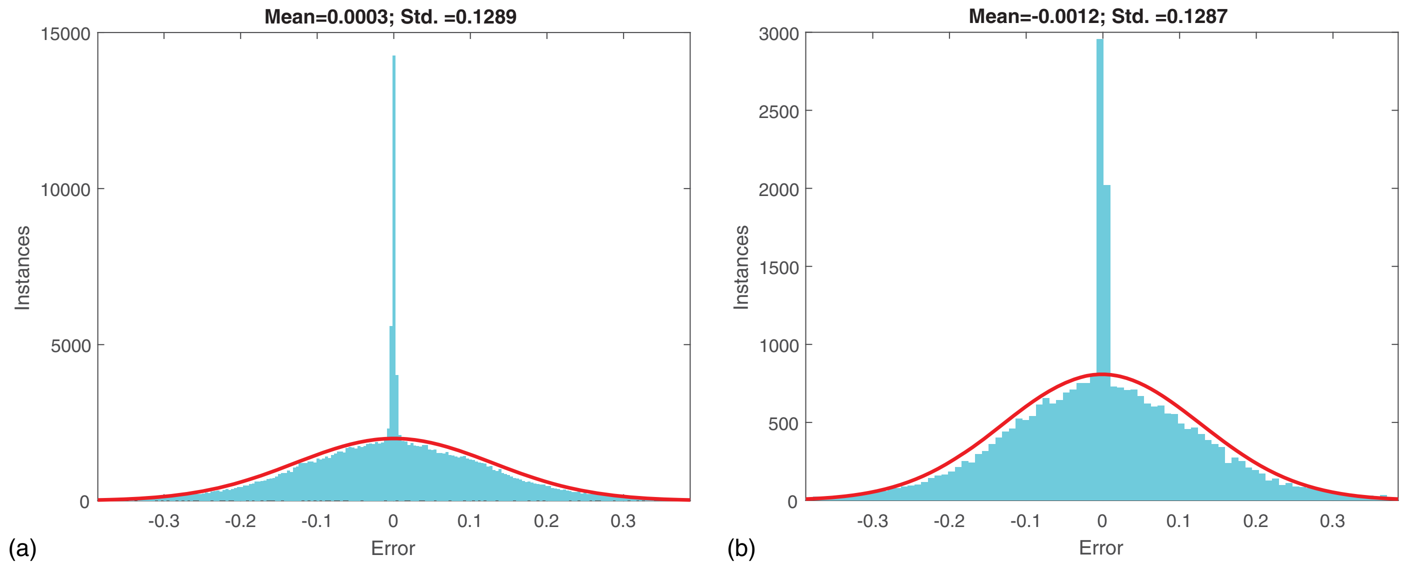


Fig. 29. MLP error histogram: (a) training error histogram; and (b) test error histogram.

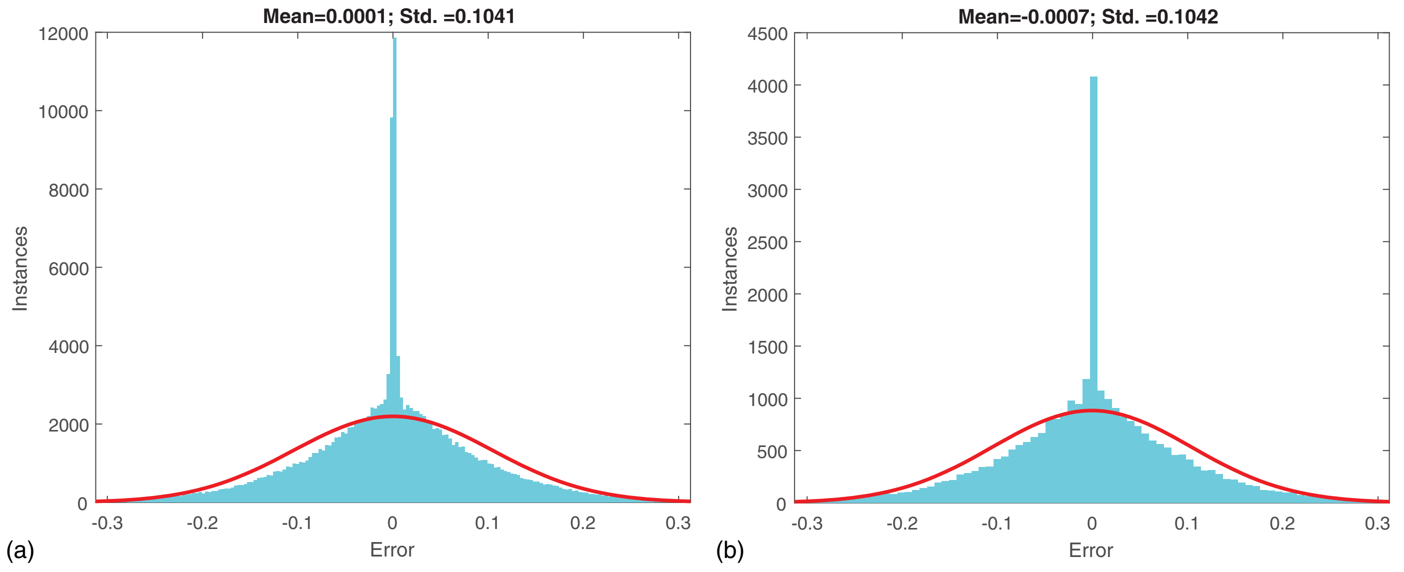


Fig. 30. CNN error histogram: (a) training error histogram; and (b) test error histogram.

through the convolution kernels. Also, the outputs of the convolution layer preserve the dominant signature in the frequency domain. The CNN models trained using 1%, 5%, 10%, and 30% noisy data with $\tanh$ activation function in the "Single-Degree-of-Freedom System" section have been selected as examples to illustrate the interpretation.

Approximate the Integration Operator with the Convolution Layer

In this section, it will be shown that for cases where integration is involved in the response estimation (e.g., $\dot{x}$, $\ddot{x}$, $f \Rightarrow x$), the convolution layer acts as an integration operator. Consider the example in the "Single-Degree-of-Freedom System" section, where velocity, acceleration, and excitation were used to estimate the displacement (i.e., $\dot{x}$, $\ddot{x}$, $f \Rightarrow x$). It is reasonable to assume that the CNN model attempts to conduct the numerical integration given the available data. The integration of a time-series signal $s(t)$ over a time interval T is given as follows:

$$g(t) = \int_{t-T}^{t} s(\tau)d\tau \tag{9}$$

where $g(t)$ = integration of $s(t)$. Eq. (9) can be written in terms of convolution operation

$$g(t) = \int_{t-T}^{t} s(\tau)d\tau = \int_{-\infty}^{\infty} s(\tau)\mathbb{1}_{[t-T,t]}d\tau = s(t) * h(t) \tag{10}$$

where $\mathbb{1}_{[t-T,t]}$ = indicator function, $h(t) = \mathbb{1}_{[t-T,t]}$; and an asterisk = convolution operation. Applying the Fourier transform to $g(t)$ gives

$$F\{g(t)\} = F\{s(t)\}F\{h(t)\} = S(\omega)H(\omega) \tag{11}$$

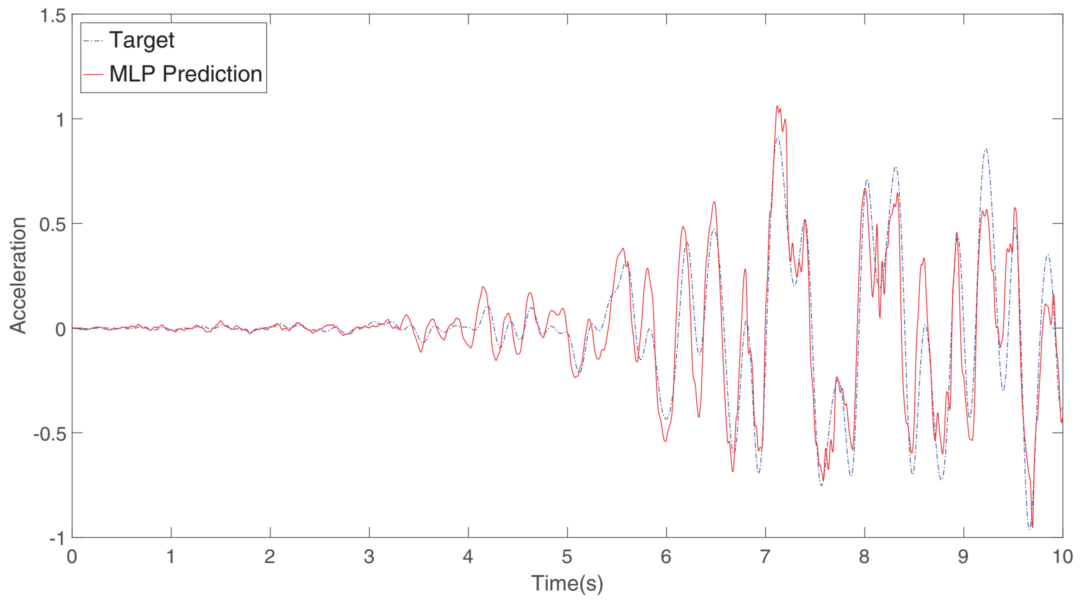


Fig. 31. MLP prediction versus target.

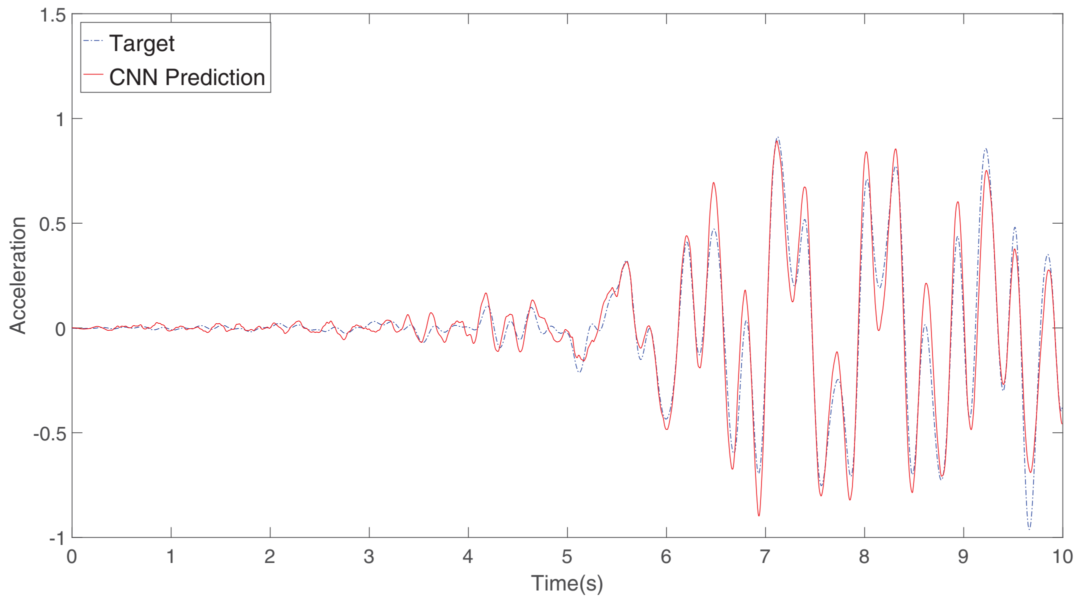


Fig. 32. CNN prediction versus target.

where F = Fourier transform; and $S(\omega)$ and $H(\omega)$ = Fourier transforms of $s(t)$ and $h(t)$, respectively. Convolution in time domain equals the multiplication in frequency domain. The theoretical displacement time series $x(t)$ can be derived from setting the signal $s(t)$ as the acceleration $\ddot{x}(t)$ and applying the convolution operation twice

$$x(t) = \ddot{x}(t) * h(t) * h(t) \tag{12}$$

Next, consider the first output channel in the first convolution layer shown in Fig. 9. The authors assume that the convolution layers in CNN attempt to estimate the ground-truth displacement time history. The estimated displacement time series $\hat{x}(t)$ from CNN is expressed

$$\hat{x}(t) = \dot{x}(t) * w_1 + \ddot{x}(t) * w_2 + f(t) * w_3 \tag{13}$$

where $w_1(9 \times 1)$, $w_2(9 \times 1)$, and $w_3(9 \times 1)$ = first, second, and the third column of the first kernel in the convolution layer, respectively. By using the equation of motion [i.e., Eq. (7)] as well as (10), Eq. (13) can be arranged

$$\begin{aligned}\hat{x}(t) &= \ddot{x}(t) * (h(t) * w_1 + w_2) + f(t) * w_3 \\ &= \ddot{x}(t) * (h(t) * w_1 + w_2) + [\ddot{x}(t) * (m + ch(t)) + k\hat{x}(t)] * w_3 \\ &= \ddot{x}(t) * [h(t) * w_1 + w_2 + mw_3 + ch(t) * w_3] + k\hat{x}(t) * w_3\end{aligned} \tag{14}$$

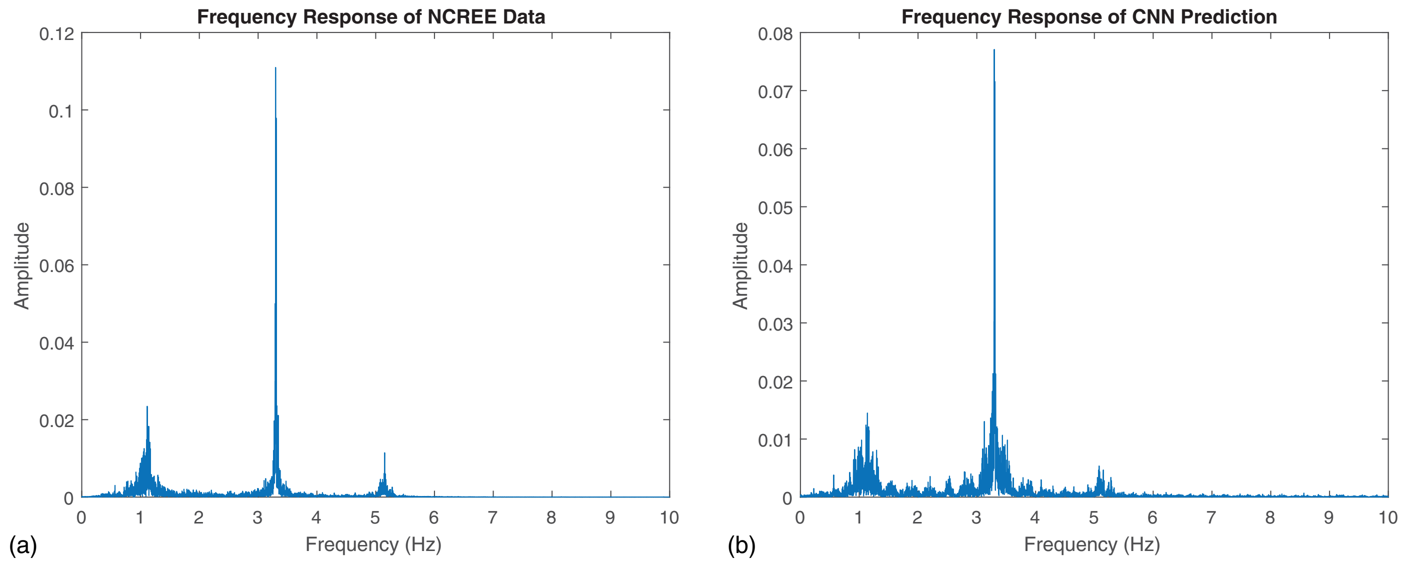


Fig. 33. Frequency response of (a) NCREE data; and (b) CNN prediction.

Table 14. Identified natural frequencies in transverse direction

Mode	NCREE (Hz)	CNN (Hz)	Error (%)
1	1.113	1.140	2.43
2	3.297	3.297	0.00
3	5.513	5.081	−7.84

Introducing the delta function $\delta(t)$ for the left of Eq. (14) yeilds

$$\hat{x}(t) * [\delta(t) - kw_3] = \ddot{x}(t) * [h(t) * w_1 + w_2 + mw_3 + ch(t) * w_3] \tag{15}$$

To see whether or not the convolution operation in this CNN model is able to approximate the integration operator, Fourier transform is applied to both Eqs. (12) and (15). The Fourier transform of Eq. (12) is

$$X(\omega) = \ddot{X}(\omega)H^2(\omega) \tag{16}$$

and the Fourier transform of Eq. (15) is

$$\begin{aligned}\hat{X}(\omega) &= \ddot{X}(\omega)\frac{[H(\omega)W_1(\omega) + W_2(\omega) + mW_3(\omega) + cH(\omega)W_3(\omega)]}{1 - kW_3(\omega)}\\ &\equiv \ddot{X}(\omega)H_{\mathrm{CNN}}(\omega)\end{aligned} \tag{17}$$

where capital letters denote the corresponding functions after the transformation; ω = variable in the frequency domain (rads); and $H_{\mathrm{CNN}}(\omega)$ = filter corresponding to the CNN kernel.

As a result, $H^2(\omega)$ was compared with $H_{\mathrm{CNN}}(\omega)$ in the frequency domain to see if they have similar frequency responses. The trained values of w_1, w_2, and w_3 of the CNN kernels for 1%, 5%, 10%, and 30% noise levels are listed in Table 15.

Because the kernels are not flipped during the operation in the convolution layer, set $h(t) = \mathbb{1}_{[t-9,t]}$ and flip $h(t)$ to compute $H^2(\omega)$. Fig. 34 shows the frequency responses $H^2(\omega)$ and $H_{\mathrm{CNN}}(\omega)$ computed using 256-point Fourier transform. The solid line is the $H^2(\omega)$, and the other lines are the $H_{\mathrm{CNN}}(\omega)$ obtained using 1%, 5%, 10%, and 30% noise levels in data. The correlation coefficient R between each $H_{\mathrm{CNN}}(\omega)$ and $H^2(\omega)$ is listed in the figure. It is demonstrated that $H_{\mathrm{CNN}}(\omega)$ is similar to $H^2(\omega)$, which means that the CNN kernel is performing operations to approximate the integration operator. The $H_{\mathrm{CNN}}(\omega)$ for 1% noise data has the highest correlation coefficient (i.e., $R = 0.98$) with the theoretical $H^2(\omega)$. As the noise level increases, $H_{\mathrm{CNN}}(\omega)$ gradually deviates from the analytical frequency response, which leads to decreasing R values (i.e., $R = 0.93$, $R = 0.90$, and $R = 0.85$ for 5%, 10%, and 30% noise data, respectively). This is expected because the convolution layer needs to perform filtering in addition to the integration operation to eliminate noise. The subsequent layers will further fine-tune the processed data. As shown in Fig. 34, the CNN

Table 15. Values of CNN kernels for 1%, 5%, 10%, and 30% noisy data

Noise level											
1%			5%			10%			30%		
w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3	w_1	w_2	w_3
0.23	0.14	−0.36	0.21	0.15	−0.34	0.17	0.17	−0.31	0.11	0.17	−0.27
0.07	0.06	−0.01	0.06	0.07	0.01	0.03	0.08	0.02	−0.04	0.10	0.05
0.52	0.12	−0.05	0.51	0.13	−0.03	0.50	0.14	−0.02	0.43	0.10	−0.01
0.15	0.35	−0.27	0.15	0.35	−0.26	0.15	0.36	−0.25	0.15	0.33	−0.24
0.26	−0.33	0.12	0.26	−0.33	0.12	0.28	−0.34	0.10	0.35	−0.37	0.06
0.12	−0.07	0.25	0.13	−0.07	0.26	0.15	−0.08	0.25	0.21	−0.11	0.23
0.39	−0.05	0.06	0.40	−0.05	0.07	0.41	−0.06	0.07	0.42	−0.05	0.07
−0.22	0.14	−0.22	−0.21	0.14	−0.21	−0.21	0.14	−0.20	−0.17	0.15	−0.18
−0.11	−0.41	0.23	−0.10	−0.40	0.24	−0.10	−0.41	0.24	−0.04	−0.39	0.25

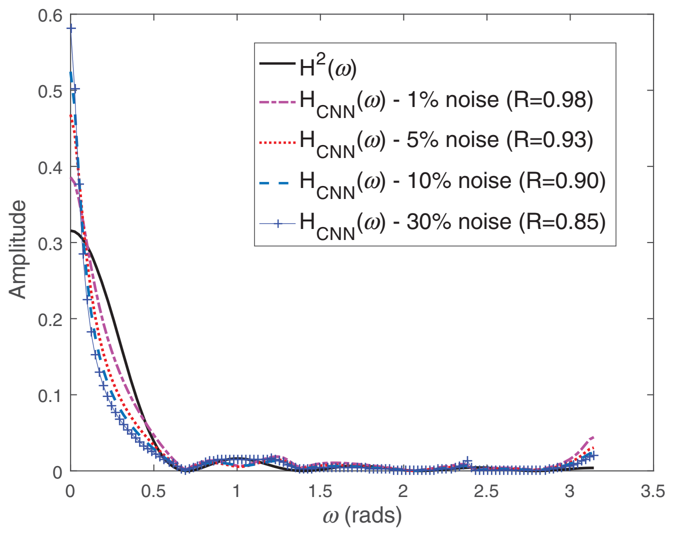


Fig. 34. Frequency responses of $H^2(\omega)$ and $H_{\text{CNN}}(\omega)$ for 1%, 5%, 10%, and 30% noise level in data.

kernel $H_{\text{CNN}}(\omega)$ filters out more high-frequency signal than $H^2(\omega)$. The authors are discussing the first output channel of the first convolution layer in the CNN model. The displacement $\hat{x}(t)$ is not the final estimation of the CNN model, and hence $H_{\text{CNN}}(\omega)$ is not identical with $H^2(\omega)$. The $\hat{x}(t)$ is passed through all the other layers in the CNN to obtain the final estimation.

Dominant Feature Extraction

In this section, the authors show that the CNN extracts the useful information from the input and preserves the dominant characteristic of the signal throughout the whole network. Fig. 35 depicts the time series of the CNN estimation, the ideal target, and their corresponding fast Fourier transform (FFT). A total of 5 s of the time series with a 200-Hz sampling rate is shown, and hence there is a total of 1,000 data points. The ideal target signal has a dominant frequency located at 2.15 Hz, and so does the estimation from CNN. The authors observe that this frequency signature is preserved in every output of the convolution layers.

Fig. 36 shows the time series of input signals, FFT of the input signals, and FFT of the first convolution layer (i.e., Layer 2) output. Fig. 37 shows the FFT of the second to last convolution layer (i.e., Layer 17) output. According to Fig. 36, the input signals have the same dominant frequency feature observed in the ideal target except for the third input signal (i.e., excitation). During the network training, CNN tries to extract the useful feature and eliminate the irrelevant information. As shown in Figs. 36 and 37, every output channel in Layers 2 and 17 exhibited the dominant frequency signature. From the inputs to the output, the network maintains the useful signature and eliminates the irrelevant high-frequency signals to achieve the estimation.

Concluding Remarks

This paper presented a deep CNN–based approach for the vibration response estimation of a linear SDOF system, a nonlinear SDOF system, and a 3-story steel frame tested by NCREE in Taiwan. For the SDOF system, five noise levels (i.e., 1%, 2%, 5%, 10%, and 30%) were considered to account for the measurement in real-world conditions. According to the linear SDOF results of using velocity, acceleration, and excitation to predict displacement, the proposed CNN approach achieved lower RMS errors for all noise levels compared with the MLP model proposed in previous studies. There were no overfitting phenomena for either MLP or CNN methods, and both methods were capable of predicting the ideal target given the noisy input and output training data. The error distribution of CNN was more centered to zero compared with MLP. The CNN-predicted time history matched with the ideal target pretty well, whereas the MLP-predicted time history captured the general trend but exhibited some undesirable high-frequency oscillations. As the noise level increased, the prediction errors increased faster for MLP with respect to the CNN, which means CNN is more robust than the conventional MLP. In the case of using excitation to predict acceleration, the CNN RMS errors were slightly lower than the MLP RMS error.

For the results of nonlinear SDOF system, the CNN-based approach worked better than the MLP for noise levels greater than 1%.

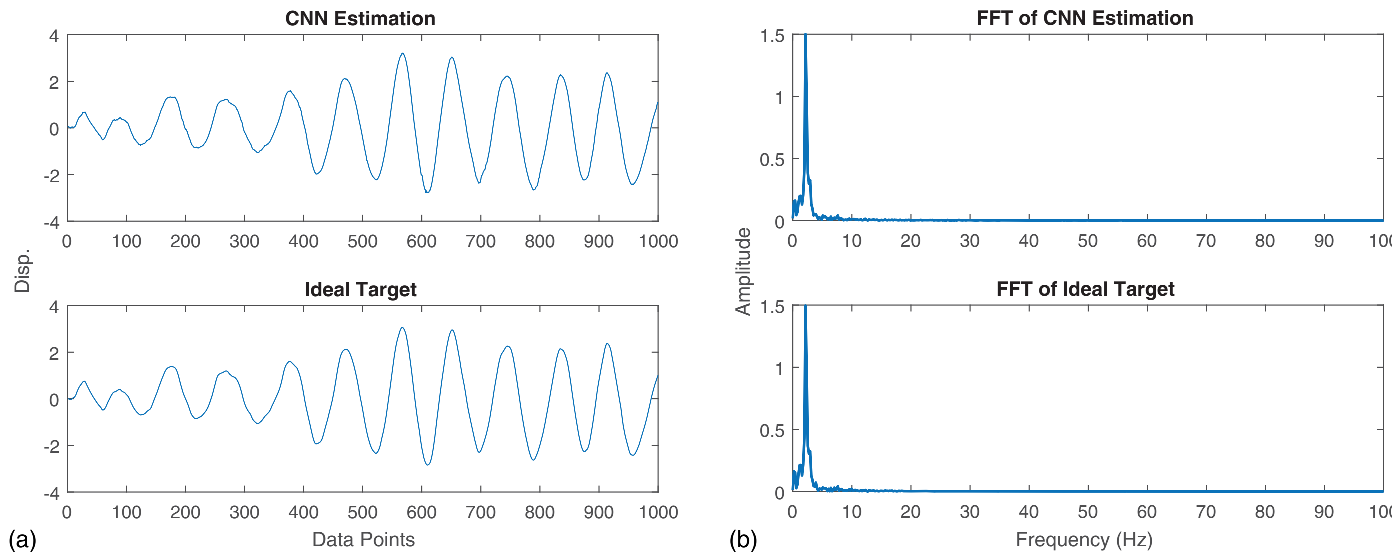


Fig. 35. Time series and frequency responses: (a) time series of CNN estimation and ideal target; and (b) FFT of CNN estimation and ideal target.

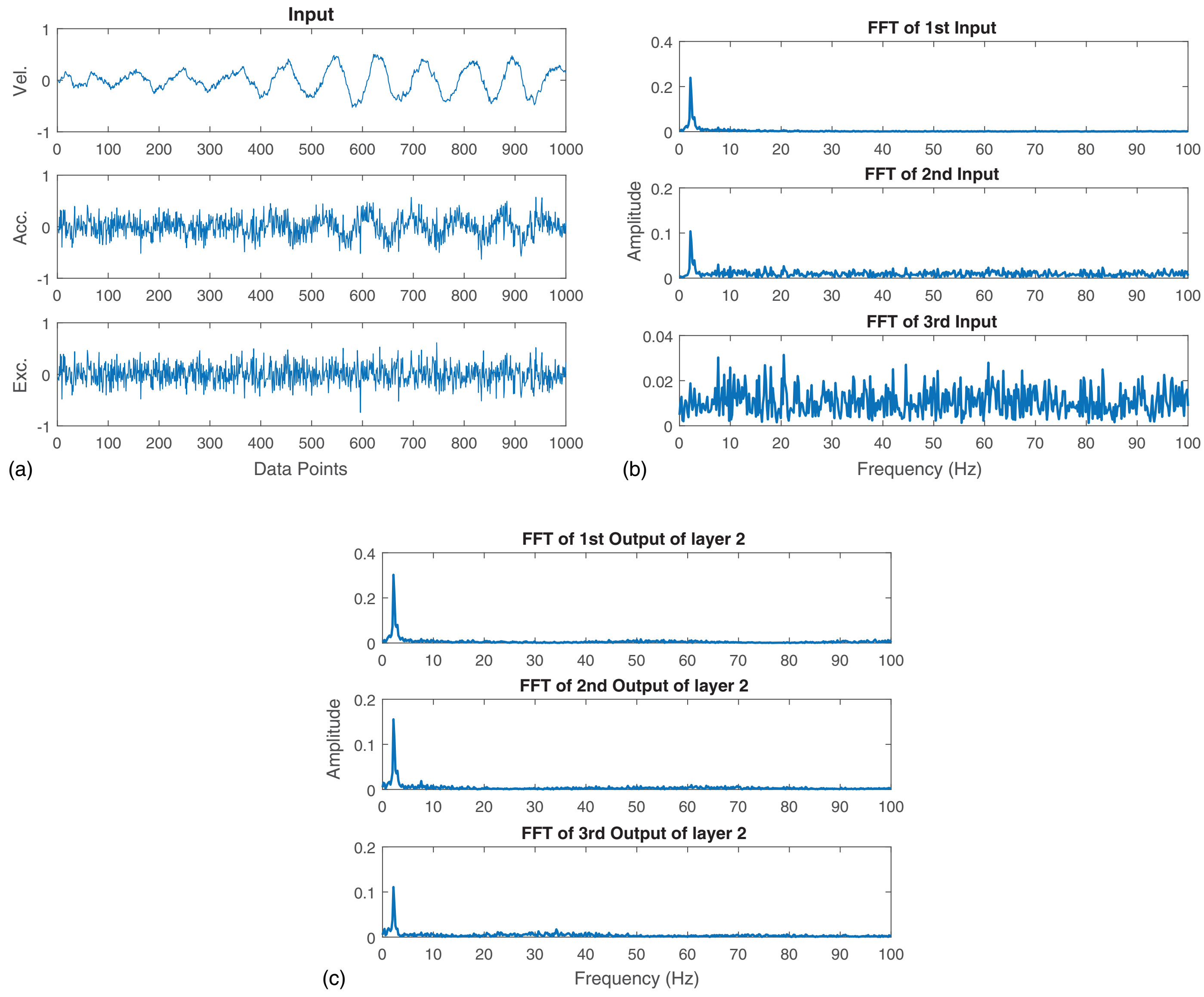


Fig. 36. Time series and frequency responses: (a) time series of input signals; (b) FFT of input; and (c) FFT of Layer 2 output.

Similarly, no overfitting was observed for either MLP or CNN methods, and the two approaches were able to learn the true behavior from noisy training data. The error distribution of the CNN prediction was more centered to that of MLP prediction, and the predicted response time history from CNN matched better with the ideal response. The RMS error for CNN increased much more slowly than the error for MLP as the noise level increased.

In the case of the MDOF system, the proposed CNN approach was validated through shake-table tests on a full-scale 3-story steel frame. Ground excitation was used to estimate the roof acceleration response. From the RMS error of CNN and MLP, it was shown that the CNN prediction error was lower than the MLP by 19.38%. The error distribution of CNN prediction was more centered to zero, and the predicted time history from CNN matched slightly better with the target response. Further, the established CNN model was used to identify the natural frequencies of the steel frame. The difference between the identified frequencies from the CNN prediction and the NCREE measurements was below 8%.

In terms of the training time for model establishment, the conventional MLP is more computationally efficient than the deep CNN. Table 16 tabulates the training times of MLP and CNN models for one noise level. The conventional MLP runs roughly 10 times faster than the CNN. Although there was no significant effect observed for the selection of activation functions, RELU runs faster than tan h by approximately 25%. The MLP algorithm was implemented in MATLAB R2016a, and the CNN algorithm was implemented using the MatConvNet (Vedaldi and Lenc 2015) library. A NVIDIA GeForce GTX 1070 GPU was to enhance the computation efficiency for both the MLP and the CNN. As discussed previously, if the network configuration is fixed, the number of weights in MLP will increase drastically as the number of input increases, whereas the number of weights in CNN will remain the same. This illustrates a potential advantage of CNN over MLP when implemented in a sensor network for field applications. As the number of sensor increases, the increase in MLP training time should be more prominent and eventually exceed the training time required by CNN. This will need further justification because there are other factors that may affect the training time, e.g., number of multiplication operations, network configuration, and chosen optimization algorithm.

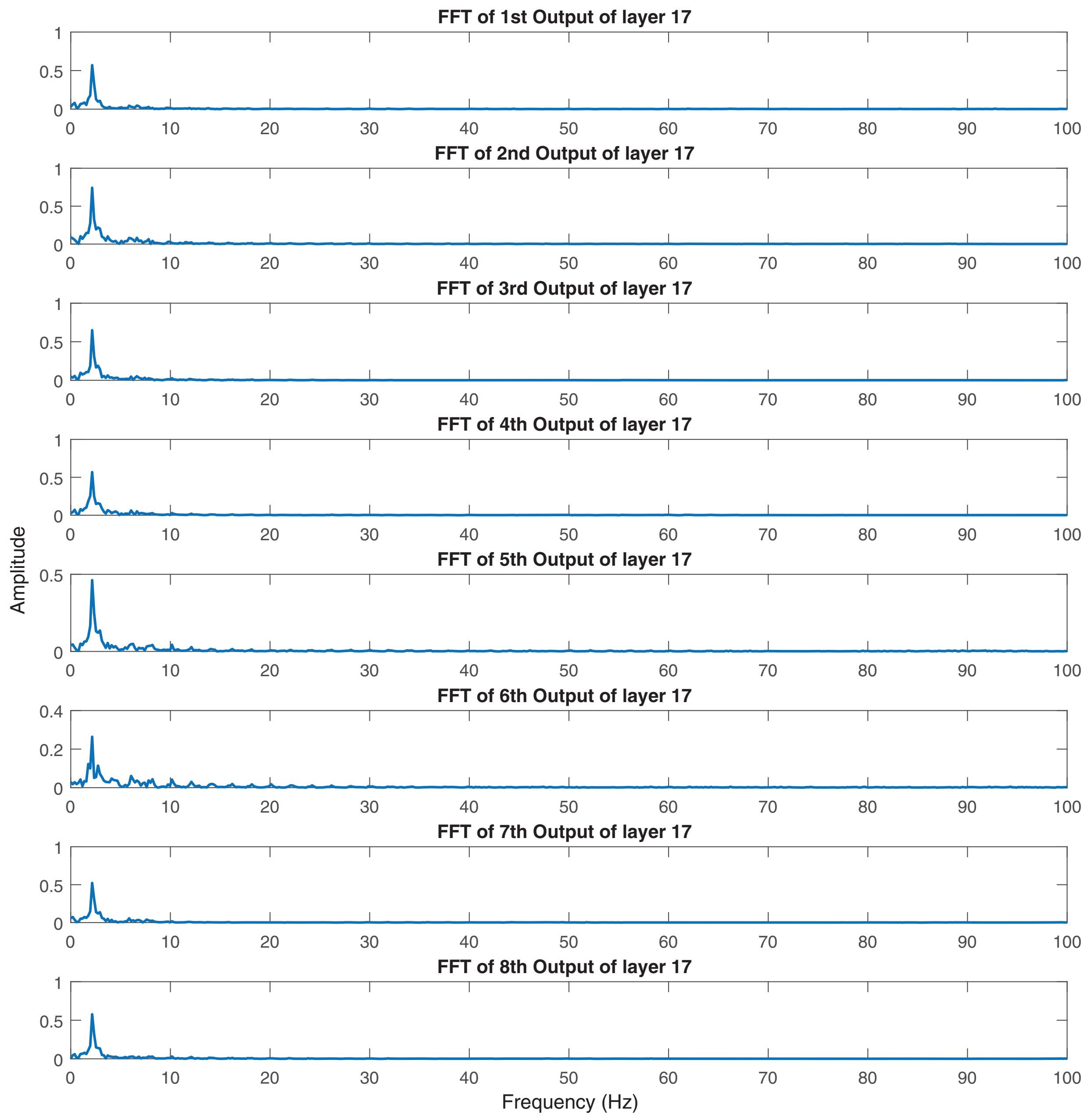


Fig. 37. FFT for the output of the 17th layer.

Table 16. Model training time comparison between MLP and CNN

Case	CNN-tan h training time (h)	CNN-RELU training time (h)	MLP-tan h training time (h)	MLP-RELU training time (h)
SDOF ($\dot{x}$, $\ddot{x}$, $f \Rightarrow x$)	2.0	1.6	0.2	0.1
SDOF ($f \Rightarrow \ddot{x}$)	1.9	1.5	0.1	0.05
Nonlinear SDOF [x, $\dot{x} \Rightarrow g(x, \dot{x})$]	2.0	1.5	0.2	0.14
MDOF ($f \Rightarrow \ddot{x}_3$)	13.0	—	1.5	—

The "Physical Interpretation of Convolution Layer" section discussed the physical interpretation of the convolution layer. The authors have shown that the integration operator can be approximated by the convolution layer. This is achieved through the comparison of frequency responses of the theoretical integration operator and those of the convolution kernel. The frequency response of the convolution kernel appears to be similar to the integration operator, and the convolution kernel attempts to eliminate more high-frequency signals because the CNN model was trained using noise-contaminated data. Moreover, the convolution layers eliminated the irrelevant information and preserved the dominant frequency signature throughout the whole network. All the output signals from the convolution layers exhibited the dominant frequency component that appeared in the ideal target signal.

In general, although both MLP and the proposed CNN approach estimated the dynamic response for the systems of interest, the proposed CNN approach is more robust against noise-contaminated data. The convolution layers act as filters, which is an advantage of using CNN. Therefore, the proposed CNN is more desirable for the response estimation of a real structure. Although the conventional MLP is more computational efficient, it remains challenging to interpret the physical meaning in the MLP model. The CNN, however, has a higher interpretability to understand the underlying mechanism during the training process, and therefore it is more favorable than the conventional MLP algorithm. As part of the future work, the authors plan to interpret the performance of convolution layers for a MDOF system.

Acknowledgments

The authors would like to thank Dr. Ting-Yu Hsu from the National Taiwan University of Science and Technology in Taiwan for providing the shake-table data collected at NCREE.

References

Abdeljaber, O., O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman. 2017. "Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks." *J. Sound Vib.* 388: 154–170. https://doi.org/10.1016/j.jsv.2016.10.043.

Arangio, S., and J. L. Beck. 2012. "Bayesian neural networks for bridge integrity assessment." *Struct. Control Health Monit.* 19 (1): 3–21. https://doi.org/10.1002/stc.420.

Atha, D. J., and M. R. Jahanshahi. 2018. "Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection." *Struct. Health Monit.* 17 (5): 1110–1128. https://doi.org/10.1177/1475921717737051.

Basheer, I. A., and M. Hajmeer. 2000. "Artificial neural networks: Fundamentals, computing, design, and application." *J. Microbiol. Methods* 43 (1): 3–31. https://doi.org/10.1016/S0167-7012(00)00201-3.

Cha, Y.-J., W. Choi, and O. Büyüköztürk. 2017. "Deep learning-based crack damage detection using convolutional neural networks." *Comput.-Aided Civ. Infrastruct. Eng.* 32 (5): 361–378. https://doi.org/10.1111/mice.12263.

Chen, F. C., and M. R. Jahanshahi. 2018. "NB-CNN: Deep learning-based crack detection using convolutional neural network and naïve Bayes data fusion." *IEEE Trans. Ind. Electr.* 65 (5): 4392. https://doi.org/10.1109/TIE.2017.2764844.

Cury, A., and C. Crémona. 2012. "Pattern recognition of structural behaviors based on learning algorithms and symbolic data concepts." *Struct. Control Health Monit.* 19 (2): 161–186. https://doi.org/10.1002/stc.412.

Derkevorkian, A., M. Hernandez-Garcia, H.-B. Yun, S. F. Masri, and P. Li. 2015. "Nonlinear data-driven computational models for response prediction and change detection." *Struct. Control Health Monit.* 22 (2): 273–288. https://doi.org/10.1002/stc.1673.

Kao, C.-Y., and C.-H. Loh. 2013. "Monitoring of long-term static deformation data of Fei-Tsui arch dam using artificial neural network-based approaches." *Struct. Control Health Monit.* 20 (3): 282–303. https://doi.org/10.1002/stc.492.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. "Imagenet classification with deep convolutional neural networks." In *Proc., 26th Annual Conf. on Neural Information Processing Systems*, 1106–1114. Lake Tahoe, NV: NIPS.

Längkvist, M., L. Karlsson, and A. Loutfi. 2014. "A review of unsupervised feature learning and deep learning for time-series modeling." *Pattern Recognit. Lett.* 42: 11–24. https://doi.org/10.1016/j.patrec.2014.01.008.

LeCun, Y., and Y. Bengio. 1995. "Convolutional networks for images, speech, and time-series." In *The handbook of brain theory and neural networks*. Cambridge, MA: MIT Press.

LeCun, Y., Y. Bengio, and G. Hinton. 2015. "Deep learning." *Nature* 521 (7553): 436–444. https://doi.org/10.1038/nature14539.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-based learning applied to document recognition." *Proc. IEEE* 86 (11): 2278–2324. https://doi.org/10.1109/5.726791.

Levenberg, K. 1944. "A method for the solution of certain non-linear problems in least squares." *Q. Appl. Math.* 2 (2): 164–168. https://doi.org/10.1090/qam/10666.

Marquardt, D. 1963. "An algorithm for least-squares estimation of nonlinear parameters." *SIAM J. Appl. Math.* 11 (2): 431–441. https://doi.org/10.1137/0111030.

Masri, S. F., M. Nakamura, A. G. Chassiakos, and T. K. Caughey. 1996. "Neural network approach to detection of changes in structural parameters." *J. Eng. Mech.* 122 (4): 350–360. https://doi.org/10.1061/(ASCE)0733-9399(1996)122:4(350).

Masri, S. F., A. W. Smyth, A. G. Chassiakos, T. K. Caughey, and N. F. Hunter. 2000. "Application of neural networks for detection of changes in nonlinear systems." *J. Eng. Mech.* 126 (7): 666–676. https://doi.org/10.1061/(ASCE)0733-9399(2000)126:7(666).

Newmark, N. M. 1959. "A method of computation for structural dynamics." *J. Eng. Mech. Div.* 85 (3): 67–94.

Pei, J.-S., and E. C. Mai. 2008. "Constructing multilayer feedforward neural networks to approximate nonlinear functions in engineering mechanics applications." *J. Appl. Mech.* 75 (6): 061002. https://doi.org/10.1115/1.2957600.

Pei, J.-S., E. C. Mai, J. P. Wright, and S. F. Masri. 2013. "Mapping some basic functions and operations to multilayer feedforward neural networks for modeling nonlinear dynamical systems and beyond." *Nonlinear Dyn.* 71 (1–2): 371–399. https://doi.org/10.1007/s11071-012-0667-9.

Pei, J.-S., E. C. Mai, J. P. Wright, and A. W. Smyth. 2007. "Neural network initialization with prototypes—Function approximation in engineering mechanics applications." In *Proc., Int. Joint Conf. on Neural Networks*. NJ: IEEE.

Pei, J.-S., and S. F. Masri. 2015. "Demonstration and validation of constructive initialization method for neural networks to approximate nonlinear functions in engineering mechanics applications." *Nonlinear Dyn.* 79 (3): 2099–2119. https://doi.org/10.1007/s11071-014-1797-z.

Pei, J.-S., and A. W. Smyth. 2006a. "New approach to designing multilayer feedforward neural network architecture for modeling nonlinear restoring forces. I: Formulation." *J. Eng. Mech.* 132 (12): 1290–1300. https://doi.org/10.1061/(ASCE)0733-9399(2006)132:12(1290).

Pei, J.-S., and A. W. Smyth. 2006b. "New approach to designing multilayer feedforward neural network architecture for modeling nonlinear restoring forces. II: Applications." *J. Eng. Mech.* 132 (12): 1301–1312. https://doi.org/10.1061/(ASCE)0733-9399(2006)132:12(1301).

Pei, J.-S., J. P. Wright, S. F. Masri, E. C. Mai, and A. W. Smyth. 2011. "Toward constructive methods for sigmoidal neural networks—Function approximation in engineering mechanics applications." In *Proc., Int. Joint Conf. on Neural Networks*. NJ: IEEE.

Pei, J.-S., J. P. Wright, and A. W. Smyth. 2005a. "Mapping polynomial fitting into feedforward neural networks for modeling nonlinear dynamic systems and beyond." *Comput. Methods Appl. Mech. Eng.* 194 (42–44): 4481–4505. https://doi.org/10.1016/j.cma.2004.12.010.

Pei, J.-S., J. P. Wright, and A. W. Smyth. 2005b. "Neural network initialization with prototypes—A case study in function approximation." In *Proc., Int. Joint Conf. on Neural Networks*. NJ: IEEE.

Razavi, S. V., M. Jumaat, and H. E.-S. Ahmed. 2014. "Load-deflection analysis of CFRP strengthened RC slab using focused feed-forward time delay neural network." *Concr. Res. Lett.* 5 (3): 858–872.

Suresh, S., S. Narasimhan, and S. Nagarajaiah. 2012. "Direct adaptive neural controller for the active control of earthquake-excited nonlinear base-isolated buildings." *Struct. Control Health Monit.* 19 (3): 370–384. https://doi.org/10.1002/stc.437.

Tompson, J., K. Schlachter, P. Sprechmann, and K. Perlin. 2017. "Accelerating Eulerian fluid simulation with convolutional networks." Preprint, submitted July 13, 2016. http://arxiv.org/abs/1607.03597.

Vedaldi, A., and K. Lenc. 2015. "Matconvnet—Convolutional neural networks for MATLAB." In *Proc., ACM Int. Conf. on Multimedia*. New York: ACM.

Wang, B.-S. 2011. "Identifying damage of the benchmark structure by using artificial neural network methods." *Adv. Mater. Res.* 219–220: 312–317. https://doi.org/10.4028/www.scientific.net/AMR.219-220.312.

Xu, B., A. Gong, J. He, and S. F. Masri. 2009. "A novel time-domain structural parametric identification methodology based on the equivalency of neural networks and ARMA model." In Vol. 79 of *Proc., 5th Int. Conf. on Emerging Intelligent Computing Technology and Applications*, 888–897. Berlin, Heidelberg: Springer.