Renewable Energy 174 (2021) 218–235

Contents lists available at ScienceDirect

Renewable Energy

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

Sequence-based modeling of deep learning with LSTM and GRU networks for structural damage detection of floating offshore wind turbine blades

Do-Eun Choe [a, *], Hyoung-Chul Kim [b], Moo-Hyun Kim [c]

[a] Dept. of Civil Eng., New Mexico State Univ., 3035 South Espina St., Las Cruces, NM, 88003, USA
[b] Center for Energy and Environmental Sustainability (CEES), Prairie View A&M Univ., Roy G. Perry College of Engineering, Prairie View, TX, 77446, USA
[c] Dept. of Ocean Eng., Texas A&M Univ., 727 Ross Street, College Station, TX, 77843, USA

A R T I C L E I N F O

Article history:
Received 27 October 2020
Received in revised form
7 March 2021
Accepted 6 April 2021
Available online 16 April 2021

Keywords:
Offshore wind energy
Machine learning
Deep learning
Long-short-term memory
Gated recurrent unit

A B S T R A C T

This paper proposes and tests a sequence-based modeling of deep learning (DL) for structural damage detection of floating offshore wind turbine (FOWT) blades using Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks. The complete framework was developed with four different designs of deep networks using unidirectional or bidirectional layers of LSTM and GRU networks. These neural networks, specifically developed to learn long-term and short-term dependencies within sequential information such as time-series data, are successfully trained with the sensor signals of damaged FOWT. The sensor data were simulated due to the limited availability of field data from damaged FOWTs using multiple computational methods previously validated with experimental tests. The simulations accounted for the damage scenarios with various intensities, locations, and damage shapes, totaling 1320 damage scenarios. Both the presence of damage and its location were detected up to an accuracy of 94.8% using the best performing model of the selected network when tested for independent signals. The *K*-fold cross-validation accuracy of the selected network is estimated to be 91.7%. The presence of damage itself was detected with an accuracy of 99.9% based on the cross-validation regardless of the damage location. Structural damage detection using deep learning is not restricted by the assumptions of the systems or the environmental conditions as the networks learn the system directly from the data. The framework can be applied to various types of civil and offshore structures. Furthermore, the sequence-based modeling enables engineers to harness the vast amounts of digital information to improve the safety of structures.

Published by Elsevier Ltd.

1. Introduction

The rapid increase in demand for clean and sustainable energy has promoted the development of technology and commercialization of the floating offshore wind turbine (FOWT). The development of FOWT intends to harvest larger amounts of energy resources from deeper water compared to the current fixed-based offshore wind turbine. Its various advantages over the on-land or fixed-based offshore wind turbine have made FOWT more appealing than ever before, particularly with the successful commissions of projects such as WindFloat in Portugal [1] and Hywind in Scotland [2,3]. However, FOWT is in the infancy stage of commercialization. This is due to challenges that complicate the predictive outcomes for its highly nonlinear system, which involves the uncertainties inherent in the system, environment, and operational conditions. The uncertainties inherent in the floating system includes larger inertia loading by motions, potential instability by blade pitch control, potential high fluctuation of power output due to motions etc. The modeling uncertainties increase as the hydrodynamics of floating structures and the turbine aerodynamics are coupled in the system. In addition, the uncertainties during operation and monitoring increase due to the longer physical distance of FOWT from the coast compared to the current fixed-based offshore wind turbine. These disadvantages result in higher operation and maintenance costs [4,5]. Structural Health Monitoring (SHM) has been discussed as a vital option that may significantly

* Corresponding author.
E-mail addresses: dchoe@nmsu.edu (D.-E. Choe), khc4156@gmail.com (H.-C. Kim), m-kim3@tamu.edu (M.-H. Kim).

https://doi.org/10.1016/j.renene.2021.04.025
0960-1481/Published by Elsevier Ltd.

improve the safety and the cost-efficiency of FOWT systems, which service lives are longer than typical offshore structures [6]. By continuously monitoring the sensor signals from the tower or blade, the on-site maintenance and inspection schedule can be optimized [7]. Early detection of structural damage could prevent catastrophic failures and secondary damage leading to exorbitant repair costs [8].

SHM methods based on the modal properties as damage sensitive features have been suggested considering the characteristics of the FOWT structures. The structural members of FOWT are typically slender, e.g., blades and towers and experience highly nonlinear environmental conditions including extreme winds and waves [9,10]. However, Tcherniak et al. [11] reported that the application of modal-property-based SHM methods to FOWT structure is limited due to several violations of the assumptions of modal-property-based methods in the FOWT system and environments.

Several non-modal-property-based methods have recently been developed for wind turbine applications. Dervilis et al. [12] investigated SHM of wind turbine blades using Artificial Neural Network (ANN), which is a traditional shallow neural network, non-deep-learning technique. It was reported that the damage could be detected before a visible crack in the structure was observed. Häckell et al. and Tsiapoki et al. [6] [[,13] introduced and validated a three-tier modular SHM framework with real field wind turbine data. They analyzed training and testing data using a combination of machine learning, damage parameters, and probabilistic modeling to determine if the damage was present for on-land wind turbines. Wang et al. [14] introduced a deep autoencoder for identifying impending wind turbine blade breakages based on supervisory control and data acquisition data, which were collected from four wind farms located in China. However, these methods have been developed for on-land turbines, which may not fully consider the nature of FOWT behavior, such as the dynamic coupling of the blades' aerodynamics and the hydrodynamics involved in floating body motion. Therefore, reliable SHM methods for FOWT are critically needed for the safe, reliable, and cost-effective maintenance of FOWT systems. In an effort to develop a reliable SHM method, we propose a sequence-based modeling of Deep Learning (DL) using Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. This approach has not been previously applied to the problem in structural engineering, but its potential for providing multiple advantages is high.

DL is a subarea of Machine Learning (ML), which is again a subarea of Artificial Intelligence (AI). DL uses deep layers of neural networks to learn the system directly from the data, which has drastically improved the performance from traditional shallow neural networks. Traditional neural network, typically called as Artificial Neural Network (ANN) [15], is a single hidden-layered neural network that feed the input only forward. Multiple layered neural networks, deep learning, enabled a fine tuning of the associated parameters of the network by its back propagation as well as by increasing the model complexity with the deeper layer [16]. In addition, the traditional neural network cannot learn from a sequential data as an input. Recurrent Neural Network (RNN) is one of deep learning method that learns from sequential data. Recent development of LSTM neural network, a type of RNN, improved the learning performance by introducing "*forget gate*" [17].

The advantage of DL methods for the maintenance of FOWT is that it is less restricted by the assumption of the system and the environmental conditions because the deep network learns the system directly from the data. While the modal-property-based methods carry various assumptions for their applications, whereas the DL method does not. For instance Operational Modal Analysis (OMA) [18] assumes the system as a linear, stationary, and observable to be applicable. The substructure of FOWT a floating system rather than stationary. In addition, the coupled dynamics of the hydrodynamics of floating body and the turbine aerodynamic is highly nonlinear, which does not respect the linearity assumption of OMA. The suggested DL method may potentially be applicable to various types of future structures with dissimilar environmental and mechanical conditions.

Beside the specific DL applications, various data-drive modeling methods has been applied to solve the wind energy problems [19–24]. Data-driven modeling is a broader definition than ML and refers the methods that are informed by data rather than the theories behind the system regardless of the methods including various ML, ANN, and DL that utilize multiple layers of neural networks.

Both LSTM and GRU networks are types of Recurrent Neural Networks (RNNs) that are specifically developed to learn long-term and short-term relationships within the sequence data often used for text recognition, weather predictions, or medical detections. These networks were chosen to design the deep networks and train the sensor signals of FOWT in our research. It has been reported that it is practically difficult to train the traditional RNN as the gradients tend to either vanish or diverge [25]. LSTM neural network was first introduced by Hochreiter & Schmidhuber [17] and was followed by many variations of the algorithm. LSTM algorithm introduced a few unique functions including "*forget gate*", which allows the network to efficiently capture the sequential dependency and differentiated LSTM from the traditional RNNs. GRU neural network was recently developed by Cho et al. [26] This application has been rapidly adopted due to its simpler structure and faster training time. Overall, it has demonstrated comparable or superior performance to the LSTM network. Bahdanau et al. [27] reported that LSTM and GRU units performed comparably to each other based on their machine translation experiments; however, evidence showing that this applies to other research areas are lacking. Chung et al. [28] tested both LSTM and GRU for polyphonic music and speech signals, where the GRU performed slightly better but comparable to LSTM. Other than this, reports addressing the performance of LSTM and GRU are scarce.

In this paper, the sequence-based modeling of DL is proposed and tested for structural damage detection of FOWT blades. The following section describes basic properties of the FOWT and the numerical modeling of the FOWT system and various damage scenarios. The next section describes the principle of the primary unit of LSTM and GRU networks for the sequence-based modeling, including conceptual and mathematical details of how the LSTM and GRU network learns the long-term dependency from sequential data. The section, entitled *Framework*, provides the details and visual representations of the entire DL process within its subsections. The last two sections of the paper present the results of the damage detection of FOWT blades using the proposed deep network and provide a conclusion for our research findings.

2. Structural damage of FOWT blades

2.1. Simulation of OC4-DeepCwind semi-submersible FOWT

In this study, OC4-DeepCwind semi-submersible FOWT [29] with NREL 5 MW baseline turbine is used as shown in Fig. 1. The turbine is a three-bladed semi-submersible type FOWT with the capacity of 5 MW. Each blade length is 61.5 m with hub diameter of 3 m, which makes the rotor diameter of 126 m. It is designed to operate the turbine between 3 m/s~ 25 m/s for its cut-in and cut-off wind speeds and the rated tip speed is 80 m/s. Hub height is 90 m and the center of mass of the turbine is −0.2 m, 0.0, 64.0 m in three global coordinate directions. The mass of rotor, nacelle, and tower

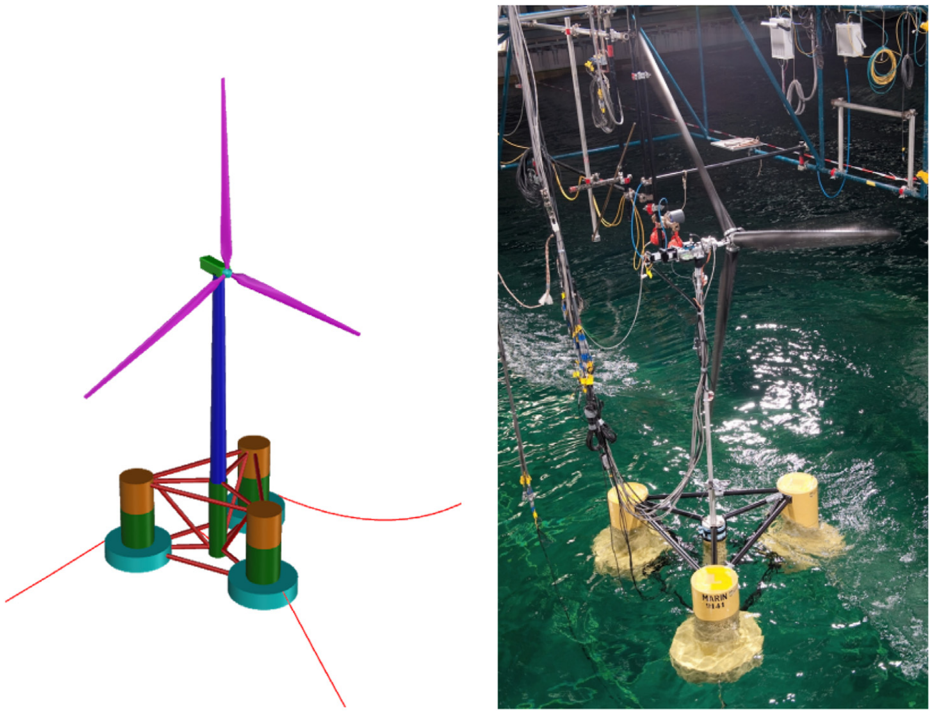


Fig. 1. DeepCWind OC4 semi-submersible FOWT: Simulation model (left) and actual model testing [30] (right).

are measured to be 110, 000 kg, 240, 999 kg, & 347,460 kg respectively. The detailed dimension, parameters, and properties of the turbine, platform, and mooring are given in Kim et al. [10].

To ensure the simulated data to represent the responses of the actual wind turbine, the floating platform, and the mooring system, two software, FAST [31] and Orcaflex [32], are coupled with additional sets of connected software. Fig. 2 shows the structure of the simulation software. The integration of the set of software, FAST and OrcaFlex, have been verified with actual testing data [30,33,34]. In this coupled model, OrcaFlex calculates the hydrodynamics of the substructures, passing mooring tension, hydrodynamic coefficients, and hydrodynamic forces to FAST. FAST calculates the displacements and velocities of the platform and passes back into OrcaFlex at each time step. For the mooring dynamics in the time domain analysis, the lumped mass element method was used. The detailed description of the coupled process between OrcaFlex and FAST can be found in Masciola et al. [32] The hydrodynamic coefficients, such as radiation damping and added mass, and first and second order wave excitation forces are obtained based on the potential theory in the frequency domain by WAMIT [35] and subsequently entered into the time-domain analysis. The effects of radiation damping force from added mass and damping coefficients are included using the impulse response function in the form of a convolution integral. Also, the viscous drag force is added using the Morison equation for pontoons and platform columns.

The dynamics due to the elasticity of the tower and blade in the time-domain simulation are implemented using ElastoDyn, a subroutine of FAST. In ElastoDyn, the blade model is a straight and

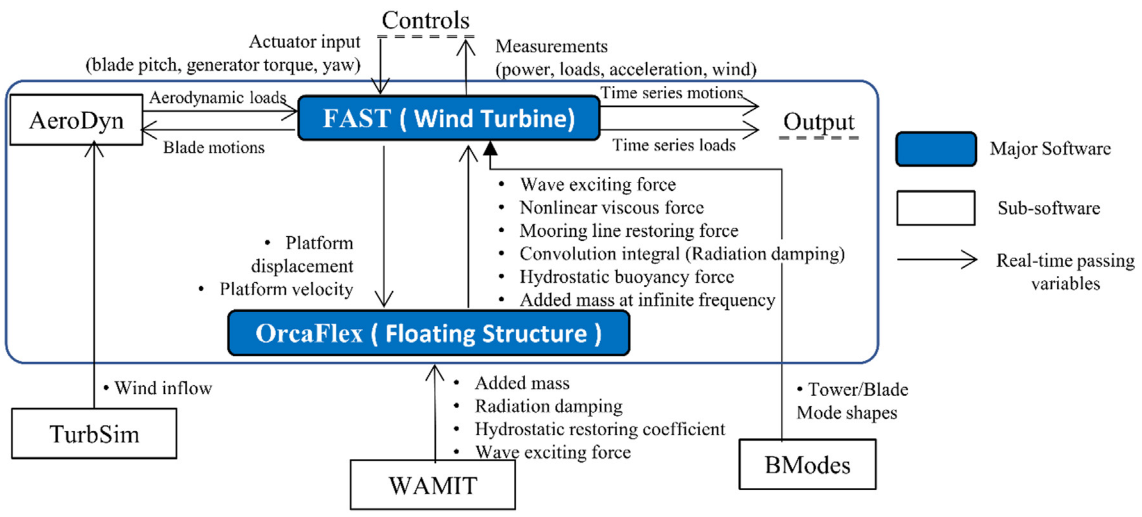

Controls
Actuator input
(blade pitch, generator torque, yaw)
Measurements
(power, loads, acceleration, wind)
AeroDyn
Aerodynamic loads
Blade motions
FAST (Wind Turbine)
Time series motions
Time series loads
Output
• Wave exciting force
• Nonlinear viscous force
• Mooring line restoring force
• Convolution integral (Radiation damping)
• Hydrostatic buoyancy force
• Added mass at infinite frequency
• Platform displacement
• Platform velocity
OrcaFlex (Floating Structure)
• Wind inflow
TurbSim
• Added mass
• Radiation damping
• Hydrostatic restoring coefficient
• Wave exciting force
WAMIT
• Tower/Blade Mode shapes
BModes
Major Software
Sub-software
Real-time passing variables

Fig. 2. Simulation software structures.

isotropic cantilever beam structure with distributed mass and bending stiffness. Blade mode shape is pre-calculated using finite element method (FEM) code, specified as a polynomial coefficient, and entered into the FAST-time domain simulation. BModes [36] is the FEM program code that provides dynamically coupled modes for a beam. The mode-shape polynomial equations are used as shape functions in a nonlinear beam model according to the Rayleigh-Ritz method. Deflections of the beam's node points are calculated using a linear summation of normal mode shapes. In this study, the first and second flap modes, the first edge mode for blade, the first two fore-aft mode, and the side-to-side modes for the tower are included [37].

$$u(z,t) = \sum_{a=1}^{n} \phi_a(z) q_a(t) \tag{1}$$

where $u(z,t)$ is the lateral deformation at time t and location z. $\phi_a(z)$ and $q_a(t)$ are the normal mode shape and generalized coordinate for normal mode a, respectively.

The full-field wind is generated using the *Turbsim* program [38], a stochastic, full-field, turbulent-wind simulator that integrates various wind spectra and turbulence models. For the implementation of turbulence, the *Kaimal* turbulence model is used. The Kaimal model is defined in IEC 61400–1 2nd ed. [39], and 3rd ed. [40], which is used for generating of full-filed wind file or hub-height wind file including wind turbulence in the TurbSim [38], which is the pre-processor program of FAST-OrcaFlex. In this study, the accelerations of fore-aft and side-to-side directions are obtained from the numerical sensors, and the acceleration results are used for the structural damage detection. For simplicity, only the parking condition (blade-fixed) is considered, and the blade rotation and control are not considered. Nineteen numerical sensors per blade are evenly distributed along the blade length, and twenty sensors at the towers are assumed. In this paper, accelerometer is used to measure vibration, which has been mainly used in long structures such as bridges [41–44], offshore structure [45], and wind turbine blade and tower [9,46–49] for the industrial and research purposes. In the case of accelerometer, it must be properly installed so that all the interested directions of acceleration can be measured. For example, if users want to measure the three axes of acceleration of flap, edge, and axial directions in the wind blade, three single-axis accelerometers must be installed for each direction. If the multi-axis accelerometer is used, one accelerometer is enough to measure all the directions of acceleration at the specific blade length.

2.2. Damage modeling

In the present study, the damaged condition is modeled by reducing the bending stiffness and the mass of the damaged materials [8,50,51]. During the simulations of FEM analyses and time-domain analyses, the distribution of mass and the stiffness of the damaged blades were entered along with the blade length. The bending stiffness is represented by the product of Young's modulus (E) and the area moment of inertia of the beam cross-section (I). The undamaged properties of mass and the bending stiffness are given in Jonkman et al. [29].

The damaged condition is modeled by various damage locations, shapes, and intensity. The blade model of FAST is divided into ten blade segments. This study aims to use DL to find the damage location in one of the ten segments. To create the various damage conditions, the blade structure of each segment is divided into two elements at 19 distinct locations (Fig. 3). At each damage location, six different levels of stiffness reductions (5%, 10%, 15%, 20%, 25 & 30%) and the corresponding mass reductions are modeled. At each damage location and each damage intensity, ten different damage shapes defined by the edge side width of the damaged blade, *w*, are modeled (Fig. 3). The right top picture in Fig. 3 shows an example damage model when the damage is located in the normalized scale of 0.46, where 1 represents the blade tip with the width *w* of 0.006 (0.006 × blade length 61.5 m). A total of 1200 damage scenarios are simulated to be used for structural damage detection at ten locations. Damage location is expressed as an integer from 1 to 10 so that the blade root element and the blade tip element are 1 and 10, respectively as shown in Fig. 3. The reduced stiffness and mass distribution for the damaged materials were replaced with the non-damaged values of the simulation model. To maintain the same number of undamaged FOWT, 120 additional simulations were performed with randomly generated wind speeds and turbulence using a coefficient of variance of 0.25 for wind speed and 0.125 for wind turbulence.

3. Principle of sequence-based modeling for damage detection of FOWT

To model and predict the structural damage status of FOWT blades, we used both LSTM and GRU neural networks. In this section, the principle of the sequence-based modeling is presented through the primary units of LSTM and GRU neural networks.

The goal of the prediction in the sequence-based modeling of this research is to identify the damaged status $\boldsymbol{Y} = \{y_1, y_2, \cdots, y_{11}\}$ indicating undamaged status (y_1) or one of ten damaged locations (y_2 to y_{11}) based on the collected signals of the numerical accelerometers $\boldsymbol{X}$, located at multiple, n, points of the wind turbine tower and blades of FOWT.

In sequence-based modeling, the model predicts y_i based on n sets of sequential data $\boldsymbol{x}_i$, which may typically be n set of numerical values in traditional modeling. In other words, instead of the numerical values of input sets $\boldsymbol{x}$, the long-term or short-term relationships within the sets of sequence vectors, $\boldsymbol{x}$, becomes the basis of the predictions. In this study, an input $\boldsymbol{x}_i$ is represented by a $[n \times t]$ matrix consisting of n sets of sensor signals recorded for the length of t. The training input $\boldsymbol{X}$ can be represented by $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_i, \cdots, \boldsymbol{x}_m\}$, where m is the total number of observations. The number of sensors, n, is also referred to as the number of *features* in DL. It is ideal to maintain identical time steps, t, for all observations, $\boldsymbol{x}_1$.through $\boldsymbol{x_m}$. If not, the timestep variation of the data sets within each batch must be minimized to reduce the amount of the padding or truncation of data.

Both LSTM and GRU neural networks are types of RNNs and are known to capture long-term dependencies. In RNNs, the models are trained to maximize the conditional log-likelihood function of [26].

$$max \frac{1}{m} \sum_{i=1}^{m} logP_{\boldsymbol{\theta}}(y_i | \boldsymbol{x}_n) \tag{2}$$

where $\boldsymbol{\theta}$ is a set of unknown parameters. The conditional probability that y_i is classified as y given the set of input $\boldsymbol{x}_i = \{x_{1,1,i}, x_{1,2,i}, \cdots x_{1,t,\ i}, x_{2,1,i}, x_{2,2,i}, \cdots x_{n,t,\ i}\}$ is defined as:

$$P(y_i = y | \ \boldsymbol{x}_{\boldsymbol{i}}\ ) = g(\boldsymbol{h_t},\ \boldsymbol{c_t}) \tag{3}$$

where $\boldsymbol{h_t} = f(\boldsymbol{h_{t-1}}, \boldsymbol{x_t}, \boldsymbol{c})$ for GRU or $\boldsymbol{h_t} = f(\boldsymbol{c_t}(\boldsymbol{h_{t-1}}, \boldsymbol{x_t}))$ for LSTM.

The hidden state$\boldsymbol{h_t}$, and cell state, $\boldsymbol{c_t}$, with associated parameters,$\boldsymbol{\theta}$, are to be trained using already-known training data sets of $\boldsymbol{X}$–$\boldsymbol{Y}$, which is the main goal of the DL modeling. Fig. 4 shows the primary unit of LSTM and GRU cells that will repeat its process for each data timestep to train the cell and hidden states.

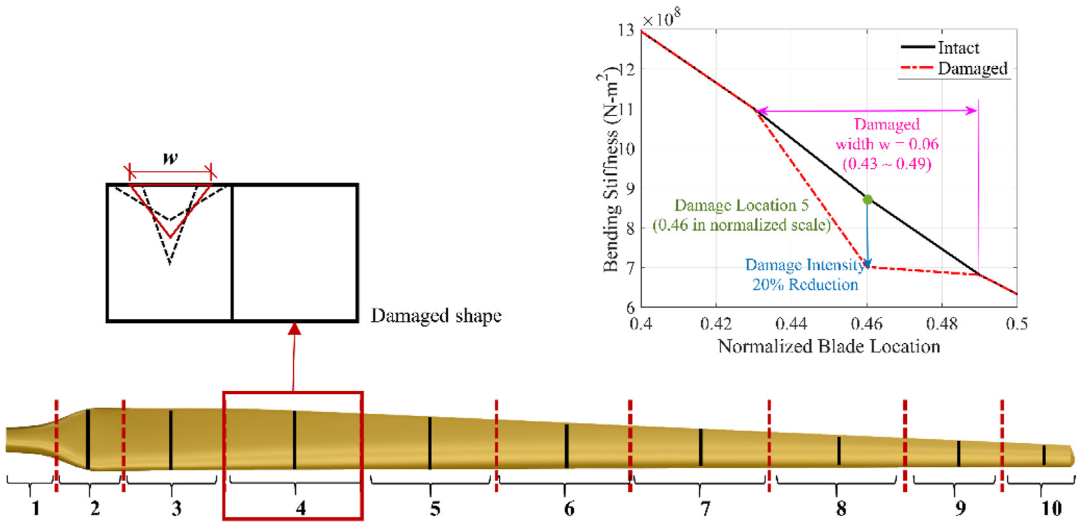


Fig. 3. Illustration of damage location index and sample damage modeling.

The principal differentiator between LSTM network and the traditional RNN is that a few gates, such as *'forget gate'*, are used at each timestep to control the passing of information along with the sequences, which capture long-term dependencies at higher accuracy [17]. The diagram on the left side of Fig. 4 shows the step-by-step prediction mechanism of an LSTM unit. Upon the input of each data $\boldsymbol{x}_i$, the recurrent gate, also called the *forget gate*, first decides how much of the information to drop from the previous cell state, $\boldsymbol{c}_{t-1}$. The new *cell state*, $\boldsymbol{c}_t$, is then updated based on the modified previous cell status, $\boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}$, the new *input gate*, $\boldsymbol{i}_t$, and *cell candidate*, $\boldsymbol{g}_t$. The new hidden status, $\boldsymbol{h}_t$, is updated based on the new cell state, $\boldsymbol{c}_t$, and the output gate, $\boldsymbol{o}_t$. The gates can be represented as follows [52,53].

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{g}_t) \tag{4a}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{4b}$$

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{V}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right), \tag{4c}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{V}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i), \tag{4d}$$

$$\boldsymbol{g}_t = \sigma(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{V}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c), \tag{4e}$$

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{V}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \tag{4f}$$

where $\boldsymbol{W}$, $\boldsymbol{V}$, & $\boldsymbol{b}$ are the unknown model parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}\}$, where $\boldsymbol{W}$ are the matrix of *input weights*, $\boldsymbol{V}$ are the matrix of *recurrent weights*, $\boldsymbol{b}$ are a vector of *bias*, σ represents a logistic sigmoid function, $\sigma(x) = (1+e^{-x})^{-1}$, and $\odot$ represents an element-wise product.

GRU neural network is first introduced by Cho et al. [26] The cell structure is presented on the right side of Fig. 4. The GRU unit includes two separate gates similar to the forget gate of LSTM, *update gate*,$\boldsymbol{z}_t$, and *reset gate*,$\boldsymbol{r}_t$, while the overall structure is simpler than LSTM. The reset gate, $\boldsymbol{r}_t$, decides how much information to *drop* from the previous information and creates a state candidate, $\tilde{\boldsymbol{h}}_t$, with the given input data $\boldsymbol{x}_t$. For instance, if the reset gate is closed to 0, this means the entire information from the previous steps are ignored. The update gate$\boldsymbol{z}_t$, decides how much information from the previous hidden state to keep. The update gate will be mostly active with the data that shows long-term dependency, while the reset gate will be active with a short-term dependency of the data. The hidden state is updated by the state candidate and the update gate. After the end of the sequence of the data, the summary of the input, **c**, is calculated. The gates can be represented as follows [28].

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (1-\boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t \tag{5a}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{V}_c(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})), \tag{5b}$$

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{V}_r \boldsymbol{h}_{t-1}), \& \tag{5c}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{V}_z \boldsymbol{h}_{t-1}) \tag{5d}$$

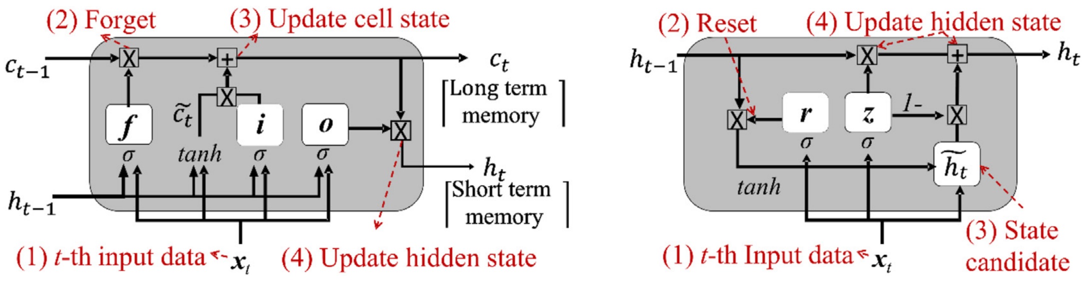


Fig. 4. Primary units of LSTM (left) & GRU (right) cells.

4. Framework

In this section, we present the framework of the sequence-based prediction using a DL technique structured with two most advanced RNNs, LSTM and GRU network, based upon the time-series information of structural behavior. We tested the proposed framework for the structural damage detection of FOWT blades. Fig. 5 illustrates the framework used in this paper: data collection, data preprocess, sequence input, LSTM/GRU layers, fully connected layer, softmax layer, output layer, and prediction. The subsequent subsections provide details of the methods used in each step. Each step of the entire framework for damage detection is coded, processed, modeled, and computed using MATLAB software.

4.1. Data collections

A total of 13,200 sets of simulations were performed: 120 sets of damaged FOWTs at each of the ten different locations with various damage levels and shapes, totaling 1200 damage scenarios, and an additional 120 sets of undamaged FOWTs detailed in earlier sections of this study. Each set of raw data is composed of 20,000 timestep signals (1000 s) for a total of 154 sensor signals (features) including 20 sensors at FOWT tower recorded in two local coordinate directions (x_{tw}, y_{tw}), 19 sensors at each of three FOWT blades recorded in two local coordinate directions (x_{bl}, y_{bl}). For the purpose of efficient training of the deep networks, only 2–12 features out of a total of 154 features were selected and tested based on the domain engineering judgment. After numerous tests, 3 types of selections are reported in this paper: (1) 12 features at tower and blade in 2 directions, (2) 2 features (sensor signals) recorded at the blade tip in 2 coordinate directions; and (3) 5 features including 2 tower sensors and 3 blade sensors. The optimum length of the sequences, *t*, per single observation was determined after testing various lengths within the range of 300–1000 timesteps collected out of 20,000 total timesteps. After multiple trial-and-errors, we found that sequence data containing less than 500 timesteps significantly decreased the performance while the performance was slightly improved by sequence data comprised of up to 900 timesteps. Therefore, data A, C, E, and F collected 900 timesteps per each feature within a single observation, and B and D collected 500 timesteps considering a large amount of data in these sets shown in Table 1.

Table 1 shows the sets of data used in this paper. Each feature set is tested for two variations. Data sets A, C, and E were collected using an identical timeline for all observations starting from 400 s (8000 timesteps). Data set B, D, and F were collected at multiple time durations per damage scenario. These variable timelines were randomly chosen using the condition that the signals have at least a 50% overlap between the sequences. The overlapping technique is commonly used in image processing DL applications. For instance, data C includes a total of 1320 observations, collecting only one sequence per feature for each damage scenario. Data set D includes a total of 6600 observations, collecting five sequences per feature at random time durations for each damage scenario (Table 1). Fig. 6 shows a sample data out of 1320 scenarios used for data set F, displaying the full-length raw data from two sensors. The figure also presents the data slicing process with 50% of overlap.

Once the group of data sets, A to F, were prepared, the data is then divided into two groups of training and testing sets: 80% for training sets and 20% for testing sets. The common split of the data in DL is either 70 to 30 (training:testing) or 80 to 20 (training:testing). In this research, the 80:20 ratio was chosen in order to secure a sufficient number of training sets. For *K*-fold cross validation, the data split is redefined for the validations of the final network, which is detailed in a later section.

4.2. Data pre-treatment: normalization, wavelet filter, & denoise

Data normalization is critical to obtain an unbiased model and to train the network efficiently. One assumes that *n* features of input sequence data with various units, dimensions are used to train a deep network and one of the feature sequences contains significantly higher numbers due to the engineering unit used for the measurement. Thus, the network will rely on this particular feature sequence compared to other features. To avoid this problem, data normalization is commonly recommended to improve the performance of the DL process. This is usually accomplished uniformly between 0 and 1 or in a zero-centered transformation. In the research reported here, the sets of data are normalized with center 0 and standard deviation 0, called *z-score standardization*, which is one of the commonly used data normalization methods. The transformation is performed with the equation, $x_{t,\ norm} = (x_t - \bar{x})/\nu$, where $x_{t,\ norm}$ is the data in the transformed space, x_t is the data in the original space, is $\bar{x}$ the mean of a time series of $\boldsymbol{x}_i$, and ν is

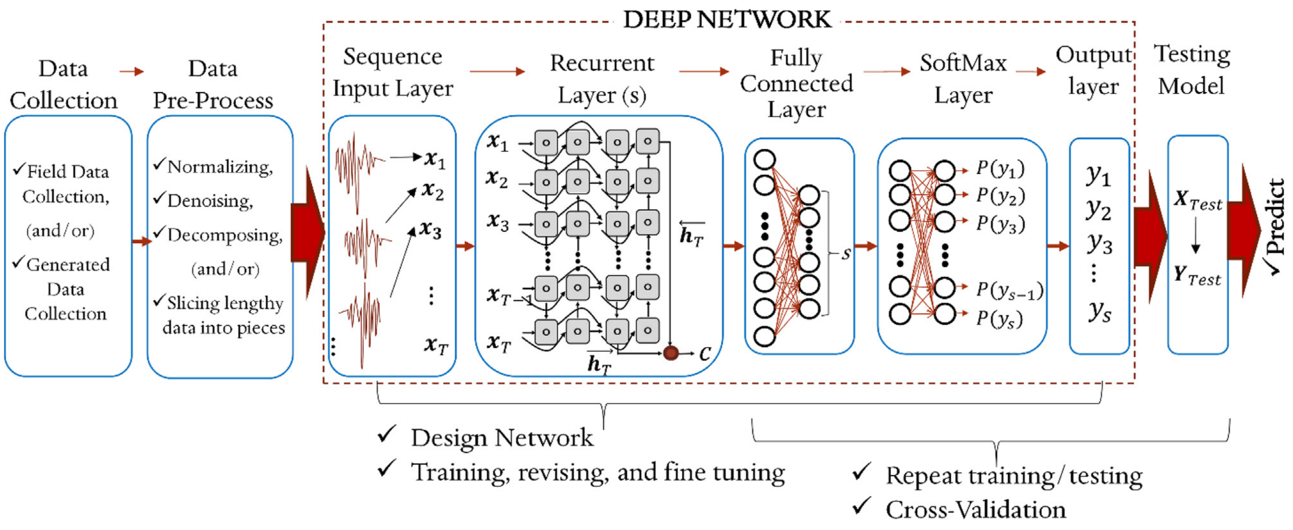


Fig. 5. Framework of sequence-based modeling of deep learning for structural damage detection.

Table 1
List of collected data sets.

Set	Number of (n)	Length of input x, (t)	Size of single input x, ($n \times t$)	Number of classes, (s)	Damage scenarios per location	Samples per scenario	Extracted timeline	Number of observations, (m)
A	12	900	12×900	11	120	1	Identical	1320
B	12	500	12×500	11	120	3	Random	3960
C	5	900	5×900	11	120	1	Identical	1320
D	5	500	5×500	11	120	5	Random	6600
E	2	900	2×900	11	120	1	Identical	1320
F	2	900	2×900	11	120	5	Random	6600

the standard deviation of the time series signal. This common approach of data normalization is used in statistical analysis as well as DL techniques for effective training of the networks.

Wavelet decomposition filters [54] are also tested for the performance improvement of training and predictions. The wavelet filter extracts additional feature vectors, which are added to the original data to train the model. The model performance trained with the added features from the wavelet filters was then compared to the performance with non-treated data. The technique has been successfully used for the recent study of signal classifications [55–57], although none of these studies were associated with DL classification. It is reported to improve the performance of predictions for the noisy data. Lately, Yildirim [58] reported improved prediction accuracies for LSTM network classification by using wavelet-filtered electrocardiogram signals as an added feature vector. The wavelet filter bank is composed of high-pass and low-pass filters (HP and LP), which are repeated for i levels. The filters are represented as $H = \sum_{k=-\infty}^{\infty} s[k]\phi_h[2n-k]$, $L = \sum_{k=-\infty}^{\infty} s[k]\phi_g[2n-k]$, where s is the input signal and $\phi_h$and $\phi_g$are low-pass and high-pass filters, respectively. The output of HP filter at each level i is detailed coefficients (cDi). The output of LP is approximate coefficients (cAi), which are passed to HP and LP filters at the next level. In our research, the added features up to 3rd level of detailed coefficient (cD1, cD2, & cD3) from the high pass filter are tested for improvement of the performance. We used the Daubechies dB6 wavelet member [59] of the wavelet family for the filter. The technique did not significantly improve the performance in this study, which we expected as the data used in this research is free from noise. This finding is detailed in later sections of this paper. Fig. 7 shows a sample observation, $\boldsymbol{x}_i$, for data with five features.

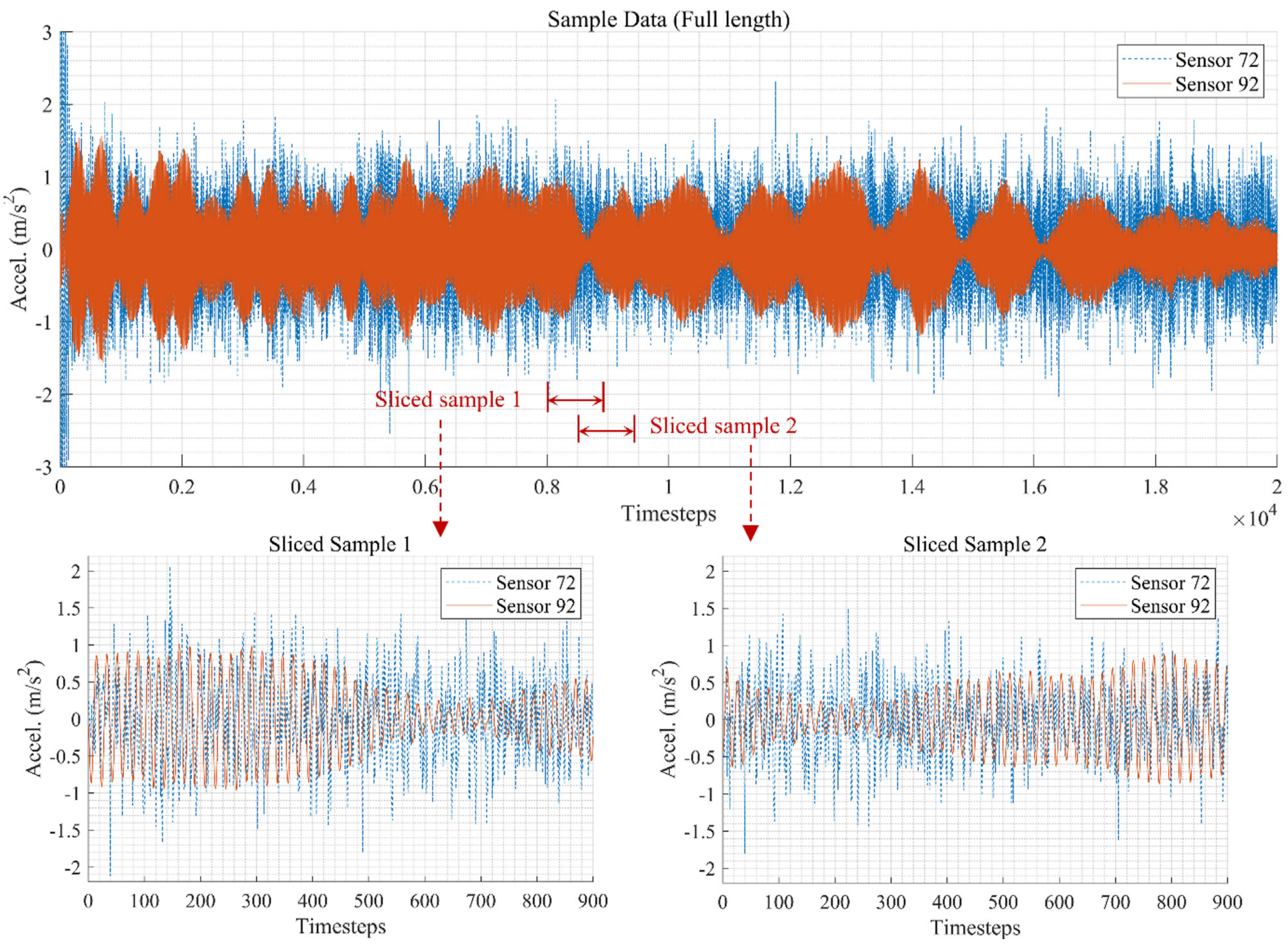


Fig. 6. A sample data out of total 1320 scenarios used for data set F; sliced samples for training.

The figure shows the raw signal (top left), normalized signal (top right), wavelet-filtered added feature (bottom left) for level 1 (cD1) only, and the normalized added features (bottom right). Fig. 8 shows the sample added features for levels 1 to 3 (cD1, cD2, & cD3) in addition to the original signal.

While the wavelet filter method is to add more information to help finding important information from noisy data, the wavelet denoise technique is to delete unimportant information from those. We also performed wavelet denoise using Donoho and Johnstone's universal threshold to quell the data noise [60]. Assuming that a time-series input $\boldsymbol{x}_n$ is expressed as $\boldsymbol{x}_n = f(i) + \sigma\varepsilon(i)$, the objective of denoising is to surpass the noise of $\boldsymbol{x}_n$ and recover $f(i)$. The denoising algorithm is composed of three steps: First, decompose the time series input with a chosen wavelet member and a chosen level of decomposition. Second, apply soft thresholding to the detailed coefficients (cD*i*) of each level with a chosen threshold method. Third, reconstruct the time-series based on the original cA*i* at the last level *i* and the detailed coefficients of cD1 to cD*i* at the levels 1 to *i* modified from the previous step. The model trained with denoised input data is then compared to the model trained by untreated data. Although field data may include a significant level of noise, our simulated data generates much less noise. In fact, the data denoise technique we used did not improve model training in our study. Fig. 9 shows the sample denoised data compared to the original signal through levels 1, 2, and 3.

4.3. Recurrent layers of LSTM and GRU neural networks

The sensor signals from FOWT are entered one at a time as input sets into the LSTM/GRU cell discussed in the previous section (Fig. 4). Fig. 10 shows the continuous process of the data input in the sequence $t = 1$ to $t = T$ with various configurations. The layer can be in a single direction or bi-directional, which can be layered deeper by stacking layers. The standard unidirectional LSTM/GRU network layer is shown in Fig. 10 (top left). Bidirectional LSTM (BiLSTM) layers scan the data once in forward and next in backward to combine both information to predict the structural damage location as shown in Fig. 10 (lower left). The bi-directional LSTM/GRU enables the efficient capture of both the past and future contexts. The equation of the bidirectional LSTM can be represented as the same equations used for the gates and the status, $\boldsymbol{h_t}$,$\boldsymbol{f_t}$, $\boldsymbol{i_t}$, & $\boldsymbol{o_t}$ as in equation (4), repeated for both forward and backward directions, $\overrightarrow{\boldsymbol{h_t}}$,$\overrightarrow{\boldsymbol{f_t}}$, $\overrightarrow{\boldsymbol{i_t}}$, & $\overrightarrow{\boldsymbol{o_t}}$ and $\overleftarrow{\boldsymbol{h_t}}$,$\overleftarrow{\boldsymbol{f_t}}$, $\overleftarrow{\boldsymbol{i_t}}$, & $\overleftarrow{\boldsymbol{o_t}}$, respectively. Finally, both sets

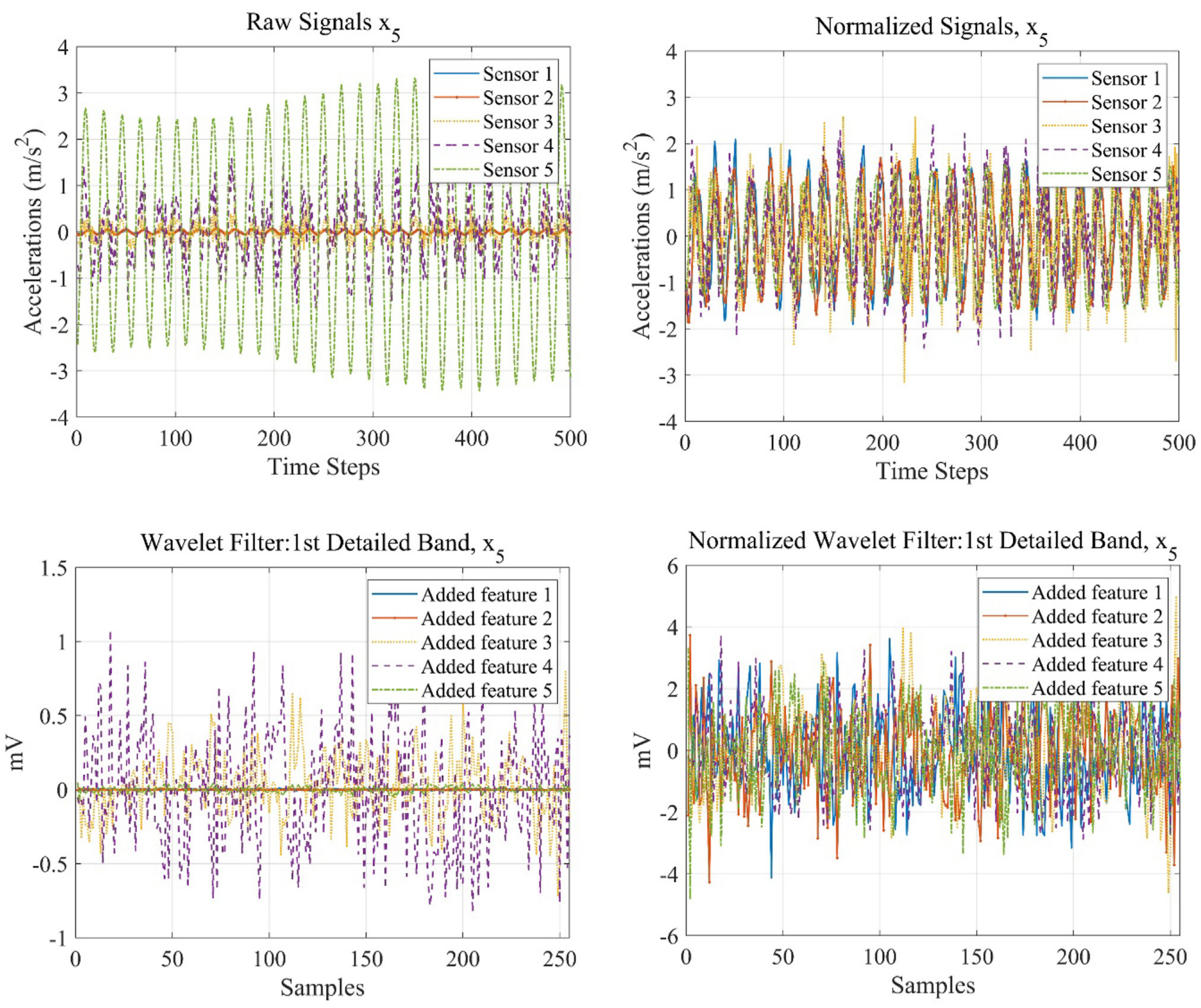


Fig. 7. Sample raw and treated data. Observation 5; damage location at 2.

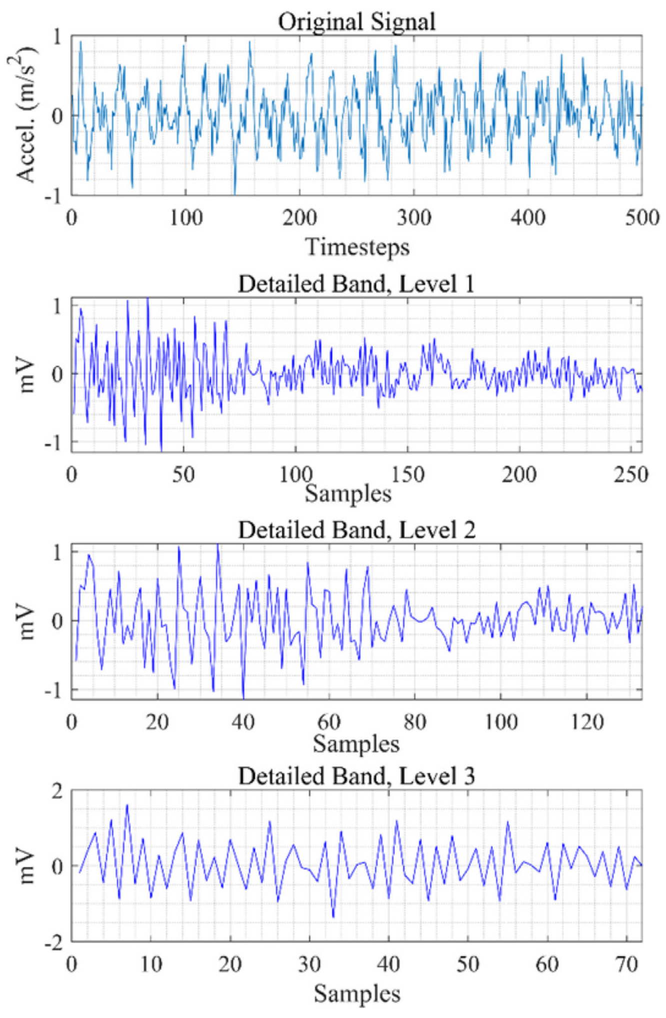


Fig. 8. Sample wavelet filtered signals: Original and 1st to 3rd detailed band.

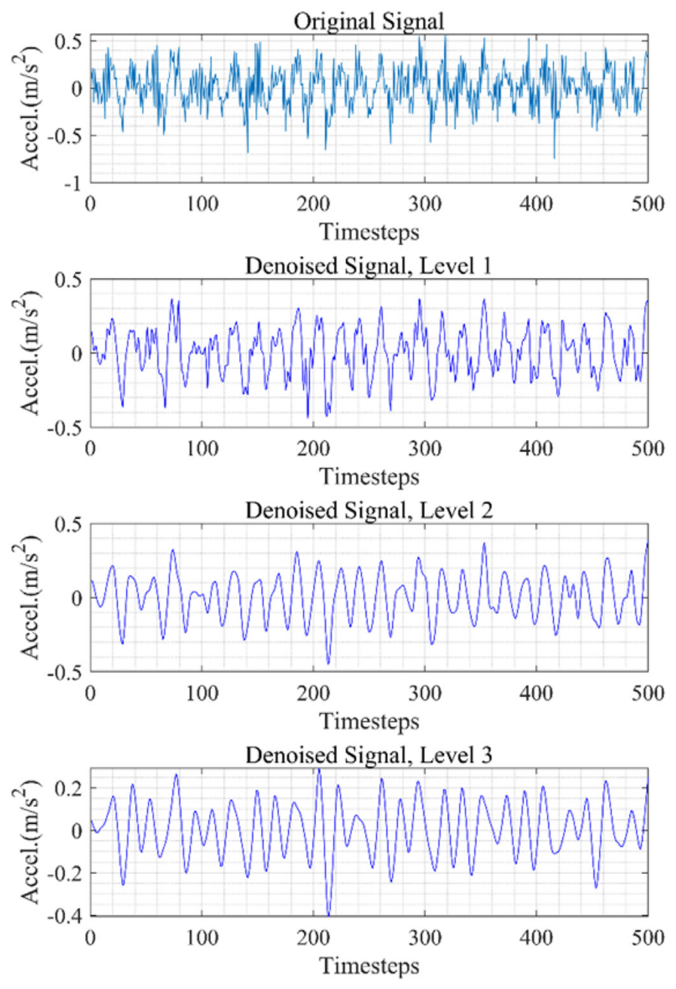


Fig. 9. Sample data denoise level 1 to 3 using Daubechies dB6 wavelet & Donoho and Johnstone's universal threshold.

of information are used to predict the damage location by $y_T = \sigma_f(\overrightarrow{\boldsymbol{h_t}}, \overleftarrow{\boldsymbol{h_t}})$, where σ_f represents the function used to estimate the output [53,61].

A deeper network of either unidirectional or bidirectional of LSTM or GRU network can be designed as needed. The right side of Fig. 10 shows a deeper BiLSTM network layer where two layers of BiLSTM networks are stacked together. The hidden output of one BiLSTM layer is propagated through t in the same way as a single BiLSTM, and at the same time, the output of the one BiLSTM layer is used as the input data to the next BiLSTM. Every hidden layer of the second BiLSTM layer receives an input sequence, which includes the output sequences of both forward and backward layers from the previous layer. In this research, the complexity of GRU network is not further developed because a single layer of unidirectional GRU performed better than the most complex form of LSTM layer and the techniques of bi-directional or double-layered approaches did not improve the performance of GRU for the data used in this research.

Computationally, the modeling of damage detection using deep neural networks can be done by either *sequence-to one regression*, which is the prediction from the given sequence sets to the damage location as a *continuous* number, or *sequence classification*, which is the prediction from the sequences to the damage location as a *discontinuous* number representing the segment number of the damaged location. In this paper, we chose to perform the *sequence classification* of the damaged segment expressed as a category variable.

4.4. Proposed networks for structural damage detection of FOWT blades

In this research, we designed and tested numerous neural networks to find the structural damage of FOWT blades. Four deep networks are presented in this paper, as shown in Fig. 11. Each proposed network contains different recurrent layers: (I) standard LSTM, (II) standard GRU, (III) single layer of BiLSTM, and (IV) stacked BiLSTM, which details and selections were presented in the previous section. More than a single layer of unidirectional GRU led to an unnecessary increase in network complexity that led to overfitting data. Therefore, only single-layered GRU networks are presented in this paper.

In order to avoid overfitting or underfitting the training data, it is critical to optimize the network complexity. A less complex model would underfit the data, which results in a high prediction error in both training and testing sets. However, an overly complex networks tend to overfit the data resulting in a low prediction error in training sets while producing a high prediction error in the testing sets. The ideal model produces a similar error rate in the predictions for both training and testing sets. The network hyperparameters are detailed in the later section of this paper.

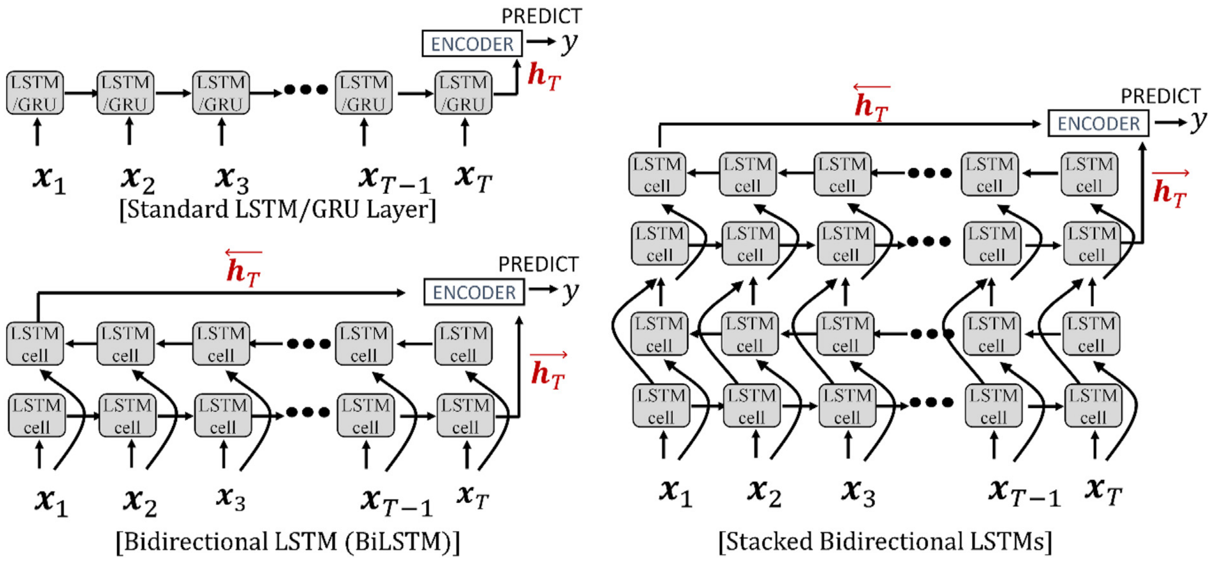

PREDICT
ENCODER
[Standard LSTM/GRU Layer]
[Bidirectional LSTM (BiLSTM)]
[Stacked Bidirectional LSTMs]
LSTM /GRU
LSTM cell

Fig. 10. LSTM, GRU, BiLSTM, and double-layered BiLSTM network layers.

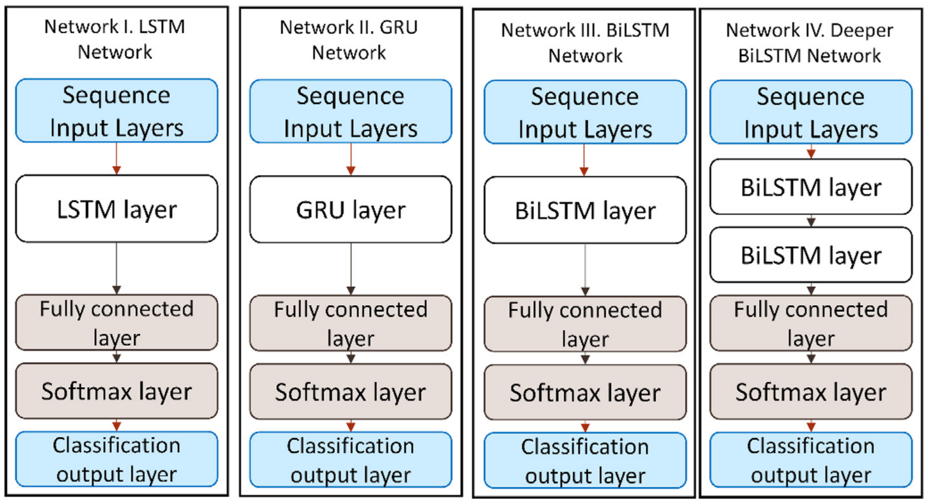


Fig. 11. Proposed networks for deep learning of structural damage detection of FOWT blades.

4.5. Fully connected & Softmax layer

The fully connected layer (Fc) follows the recurrent layers of LSTM/GRU/BiLSTM where the outputs of these recurrent layers are connected to every activated unit of the next layer. Softmax layer. In this research, the fully connected layer multiplies the input data to the weight, $\boldsymbol{W}$, from the last iteration of the previous recurrent layer and adds the *bias*, $\boldsymbol{b}$. This layer outputs the classification based on the trained recurrent layer given input data.

The Softmax layer provides a vector, $\boldsymbol{S}(y_i)$, as an outcome that includes the probabilities of each target class based on the weights from the last step of the previous layer: $\boldsymbol{S}(y_i) = P(y = y_i|\boldsymbol{x}_i, \boldsymbol{\theta})$, where $i = 1$ to 11. In this research, the Softmax function calculates the probabilities that the damage has occurred at the location $y_i = 1 \sim s$ given the input sensor data ($\boldsymbol{x}_i$) where $s = 11$ is the size of the vector $\boldsymbol{S}$ and the number of the class, representing the damage status: no damage (ND) status and possible damage locations 1 to 10 defined in FOWT blades. The probabilities are between 0 and 1 and later used to determine the damage location. Typical Softmax function is used, which calculates the ratio of the exponential of the out value from Fc layer to the sum of the exponential values of all out values from the Fc layer.

4.6. Fine-tuning network

In order to train an unbiased model with the highest accuracy and the simplest structure, it is necessary to fine tune the hyper-parameters such as the *number of hidden unit* and a *learning rate*.

The *number of hidden units* is the size of the hidden status vector per batch. A larger number of hidden units reflects increasing model complexity, and a network with an excessive number of hidden units for a given system and data attribution may result in the overfitting of data. In the damage detection models, it was defined between 64 and 256 units after testing the training progress. The selected number of hidden units for each case and the selection progress will be shown in the later section with the results of the network training.

Learning rate (LR) is a hyperparameter used in the stochastic gradient descent optimization algorithm to find the unknown parameters$\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}\}$, of the model discussed in equations (4) and (5). This controls how much to change the model parameters $\boldsymbol{\theta}$, upon the estimated error when the model weights are updated. In fact, this is the most critical hyperparameter during the training. A lower value of LR tends to overfit the model by leading it to the local minimum, whereas, a larger value of LR may not stabilize the model. The optimum LR varies depending on the model complexity and the data attributions. This should be explored in a large interval and then refined using a smaller interval. LR can be scheduled to change over the iterations. In this research, the LRs were fine-tuned and tested for each different network and data set to find the best network.

Batch size is the number of data sets that will be scanned at each iteration out of the total number of observations used for the training data. In our research, a batch size of 50 was used for all cases after testing various numbers ranging between 20 and 200. For example, if one of training data has 1000 observations, then the entire data set is divided into 20 group of 50 observations. Then, only 50 data are scanned at each *iteration*. It takes 20 iterations to scan the entire observation, which is called an *epoch*. In this case, one epoch is equal to 20 iterations. In the network training, the maximum epoch also defines when to stop the network training since excessive iterations after the model to find the optimum status may also cause overfitting problems. It is desirable to have the maximum epoch be within a reasonable range. The settings of hyperparameters are related to one another. For instance, a lower setting of LR requires longer training, meaning a higher value for the maximum epoch. In this research, the maximum epoch is defined within the 200 to 600 range depending on model complexity and defined learning rates.

4.7. Testing, cross-validation, & estimators of prediction error and accuracy

The trained networks are tested using independent sensor signals not employed during the training process. The data split ratio of 80–20 is used for training and testing sets. However, accuracies obtained from the testing sets may differ slightly from the true value of prediction accuracies depending on the number of observations used during training and testing or on the magnitude of bias incorporated into the model. Once the model complexity and hyperparameters are fine-tuned, the cross-validation was performed for the selected models for each network-data set.

Cross-validation is critical to obtain an unbiased model and to allow the estimation of prediction errors and accuracies, which more closely represent the true value. In this section, the *K*-fold cross-validation method and estimators of the prediction error and accuracy used for structural damage detection in FOWTs are presented.

In the machine learning and statistical predictions, there are several cross-validation methods and variations such as *leave-p-out, hold-out, and K-fold* cross-validations. Cross-validation was studied earlier by Stone [62] which was a *leave-one-out* method in the context of regression, which is the same as a leave-p-out method when $p = 1$. Geisser [63,64] has studied *V*-fold method. Leave-*p*-out method is an exhaustive validation method which tests and validates all possible ways to split the data. Leave-one-out validation results in the same computation as *K*-fold using $K = m$ where *m* is the number of observations, which is also considered as an exhaustive method. Hold-out testing is known for its inefficient use of the data because it does not account for the variance with respect to the training set reported by Dietterich [65]. In the same report, it was concluded that *K*-fold is a more appropriate method when comparing one algorithm to another. We used *K-fold cross-validation* which has been shown to be one of the most reliable and non-exhaustive validation methods for estimating an unbiased model [66]. In addition, Blum et al. [67] reported that the *K*-fold method provides more accurate estimates of prediction errors compared to hold-out testing.

K-fold cross-validation splits the training data set into *K* sets with a size of m/K, where m is the total number of observations. $K-1$ sets are used for training network and one set is used for validation, which independent training process is repeated for *K* times by choosing a different validation set through $k = 1$ through $k = K$. Model accuracy is calculated by averaging *K* training accuracies and the most accurately trained model is used for prediction.

Fig. 12 visualizes the iteration of repeated training for *K*-fold cross-validation. In this research, $K = 8$ is used. The eight (*K*) sets of data is divided into two groups, seven set for the training and one set for the validation. The split of eight fold is randomly selected within the group such that $(\boldsymbol{X}, \boldsymbol{Y}) = (\{\boldsymbol{X}_k, \boldsymbol{U}_k\}\ \{\boldsymbol{Y}_k, \boldsymbol{V}_k\})$, where $\boldsymbol{X}_k$ and $\boldsymbol{Y}_k$ represent *k*-th *training* set of the input signals (*X*) and corresponding classifications (*Y*) with a size of $m-(m/K)$, and $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ represent *k*-th *validation* set with a size of m/K.

The cross-validation estimator, *CV,* [66] is used for the estimator of the prediction error, *PE*, given algorithm $A(.)$ and the sets of data $(\boldsymbol{X}, \boldsymbol{Y})$ in the cross-validation of the structural damage detection model of FOWT. The estimator of the prediction error is represented as follows:

$$PE[A(), (\boldsymbol{X}, \boldsymbol{Y})] = CV = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{(m/K)}\sum_{\boldsymbol{x}_i, y_i \in \boldsymbol{U}_k, \boldsymbol{V}_k} L[A(\boldsymbol{X}_k, .), (\boldsymbol{x}_i,\ y_i)]\right) \tag{6}$$

where $A(\boldsymbol{a}, \boldsymbol{b})$ is the return of a function by algorithm *A* trained by the set $\boldsymbol{a}$ upon the new input of $\boldsymbol{b}$ and $L[.]$ is a component loss which represents the misclassifications of each observation. In this research, the loss is defined as a weighted classification error:

$$L[A(\boldsymbol{a}, .), (\boldsymbol{x}_i,\ y_i)] = w_i I\{A(\boldsymbol{a}, \boldsymbol{x}_i) \neq y_i\} \tag{7}$$

where w_i is the weight which sums to 1, $\sum_{i=1}^{n} w_i = 1$ and $I\{\}$ is the indicator function: $I\{x\} = 1$ if *x* is true and otherwise, $I\{x\} = 0$. In the structural damage detection, $I\{\} = 0$ if the damage status of FWOT blades detected from the deep neural network model is the same as the actual damage status, meaning no loss. Otherwise, the value returns to 1 with given weight. Note that $A(.)$ is trained by $\boldsymbol{X}_k$ and tested for $\boldsymbol{x}_i, \in \boldsymbol{U}_k$ in equation (7).

Similarly, the estimator of the prediction accuracy given the network algorithms and the sets of numerical sensor data of FOWT can be calculated as follows:

$$PA[A(), (\boldsymbol{X}, \boldsymbol{Y})] = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{(m/K)}\sum_{\boldsymbol{x}_i, y_i \in \boldsymbol{U}_k, \boldsymbol{V}_k} I\left\{A(\boldsymbol{X}_k, \boldsymbol{x}_i) = y_j\right\}\right) \tag{8}$$

5. Results of damage detection

5.1. Summary results

Table 2 shows the results of the damage detection with the selected models for four proposed networks (Fig. 11) with various sets of sensor data (Table 1). The general model selection

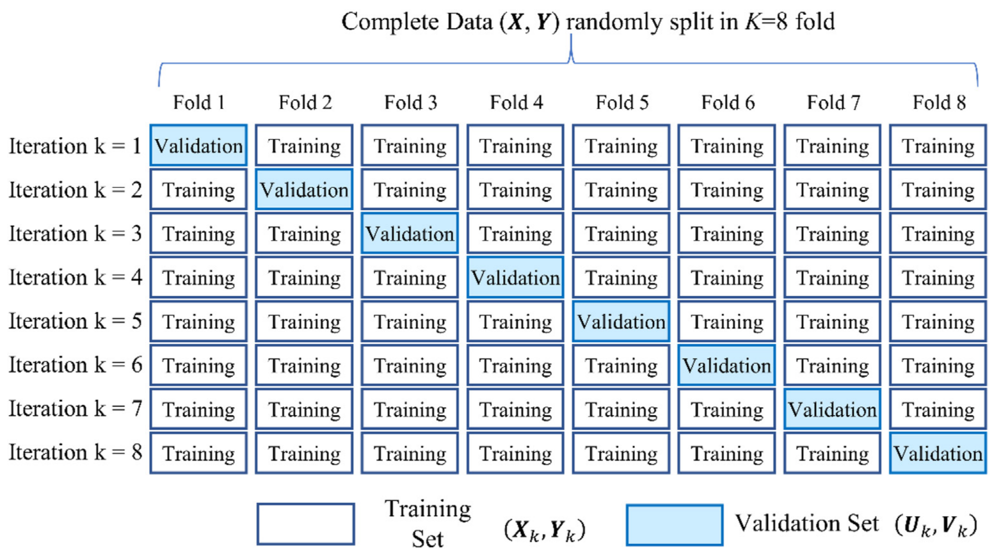


Fig. 12. Iterations k = 1~K in K-fold cross-validation with K = 8.

philosophy is applied to this research that an unbiased model with the highest accuracy and with the simplest structures are to be selected. The selected model (D-II) exhibits 91.7% of the cross-validation accuracy based on the estimator of K-fold iterations and was trained with network design II using data sets with five features (sensors), data D. The detailed K-fold cross-validation results of the selected D-II model is shown in Table 3, where the best network prediction reached 94.8%. The learning rate was scheduled to set the initial value of 0.007 with a drop rate of 0.3 every 100 epoch up to the maximum epoch of 600 in this case. In all cases, the best results are obtained using normalized data input sequences with no other data treatments.

It is found that the performance of the single-layer unidirectional GRU network (network II) was superior to even the most complex LSTM network, the double-layered bidirectional LSTM networks (network IV). In most of the data selections, the performance of the GRU network demonstrated a 10–30% gain in accuracy compared to the other networks tested. The accuracies reached 89.7% using a 12-feature selection, 91.7% with a 5-feature selection, and 81.8% with a 2-feature selection. It is also observed that the selection of more than five features does not improve performance.

Except for data sets A and F, the bidirectional layer of LSTM (BiLSTM, network III) slightly improves the performance of LSTM networks (network I); however, the results are comparable to one another. It was also found that the double-layered BiLSTM network performs better than the single-layered BiLSTM with most of the data selections except for data B. However, the computational time takes roughly 300% longer compared to the single-layered BiLSTM, and the performance improvement is a marginal percentage: 2%

Table 2
Results of the selected models for all proposed networks and data.

Case	Data	Network	Prediction Accuracy ($PA[\cdot]$)	Average Training Accuracy	Initial LR	LR Drop (rate/freq.)	# of Feature	# of Hidden Unit	Batch Size	Max Epoch
A-I	A	I	0.628	0.831	0.001	0.5/200	12	128	50	350
A-II		II	0.757	0.834	0.0008	0.5/200		200	50	220
A-III		III	0.600	0.833	0.001	0.5/100		128	50	350
A-IV		IV	0.619	0.887	0.0005	0.5/100		128×2	50	250
B–I	B	I	0.746	0.906	0.001	0.5/200	12	128	50	300
B-II		II	0.867	0.947	0.0005	0.5/200		128	50	600
B-III		III	0.770	0.957	0.001	0.5/100		128	50	300
B-IV		IV	0.735	0.926	0.001	0.5/100		128×2	50	300
C–I	C	I	0.678	0.941	0.002	0.5/200	5	96	50	600
C-II		II	0.640	0.959	0.002	0.5/200		96	50	600
C-III		III	0.695	0.982	0.007	0.5/100		96	50	450
C-IV		IV	0.686	0.998	0.007	0.5/100		96×2	50	400
D-I	D	I	0.640	0.998	0.007	0.3/100	5	96	50	600
D-II		II	0.917	0.997	0.007	0.3/100		96	50	600
D-III		III	0.819	0.998	0.007	0.3/100		96	50	600
D-IV		IV	0.848	0.980	0.007	0.3/100		96×2	50	450
E-I	E	I	0.665	0.947	0.005	0.5/100	2	128	50	600
E-II		II	0.711	0.867	0.001	0.5/200		200	50	600
E-III		III	0.674	0.915	0.001	0.5/100		200	50	600
E-IV		IV	0.699	0.991	0.001	0.5/100		128×2	50	600
F–I	F	I	0.731	0.991	0.001	0.5/200	2	128	50	600
F-II		II	0.818	0.999	0.001	0.5/200		200	50	600
F-III		III	0.434	0.986	0.005	0.5/100		128	50	600
F-IV		IV	0.625	0.994	0.005	0.5/100		96×2	50	450

Table 3
Results of *K*-fold cross-validation of the selected model of D-II.

Data	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	Best Accuracy	*PA[·]*
Validation	0.895	0.932	0.948	0.925	0.918	0.910	0.892	0.918	0.948	0.917
Training	0.999	0.998	1.000	0.999	0.999	0.998	0.999	0.999	1.000	–

increase in data A and C; 3% increase in data D, 19% increase in data F.

It is also observed that unnecessary increases in model complexity result in lower performance. This is particularly clear when the model is trained using a small number of features. For instance, the training model with data set F (at the smallest number of the features) shows an accuracy of 73.1% with a single-layer LSTM network (F–I); however, double-layered BiLSTM network (F-IV) decreases the performance accuracy to 62.5% for the same data set.

5.2. Summary of the selected model, D-II

The *K*-fold iterations of the selected model of D-II is displayed in Table 3. The prediction accuracy (*PA*[.]), 91.7%, is estimated based on equation (8). The best model exhibits 94.8% of accuracy at $k = 3$. Note that *PA*[.] is defined only based on the accuracies of validation sets after being trained by the training sets and, therefore, the table cell that represents *PA*[.] on the bottom row remains blank in the table. In this *K*-fold cross-validation, out of total 6600 observations (m), the number of observations used during each training (m-m/K) is 5775 and the number of observations for the validations (m/K) is 825. The entire training process is repeated $K = 8$ times as shown in Fig. 12. A single set of observation includes the signals from 5 sensors of 500 timesteps each. The K fold is randomly split and, therefore, the total numbers of observation for each class may not be exactly the same.

Fig. 13 and Fig. 14 show the *confusion matrix* of the selected model of case D-II when $k = 2$ & $k = 3$, which compares the detected damage status to the true damage status. ND stands for no damage (ND) status, and the numbers 1 to 10 represent the location of the damages. Diagonal boxes display the numbers of true classifications for 11 different damage states. The lower and upper white triangles show the numbers of false classifications. The bottom line of the confusion matrix shows the prediction accuracies (%) of each true damage group and the bottom right corner box displays the overall prediction accuracy. Similarly, the right-side column of the confusion matrix displays the prediction accuracies (%) of each predicted damage group. The prediction accuracies reached 93.2% ($k = 2$) and 94.8% ($k = 3$) for determining both the presence and location of the damage. This is displayed in the box at the bottom right corner of Figs. 13 and 14. In the model at $k = 3$, out of 5.2% of total misclassifications, 3.1% of misclassifications locate the damage within one-segment distance. The damage location of only 2.1% of the entire data set is located more than one-segment distance from the true damage location.

The presence of damage itself was detected with higher accuracy (99.9%) regardless of the identification of its location in this model. The damage status 1 to 10 indicates that damage (D) existed while ND indicates no damage. From the network $k = 3$ (Fig. 14), only 1 sample out of 825 samples tested provided a false identification of damage, and this was at location 1 and predicted as ND from the network. The rest of the 824 samples show true D-D or true ND-ND regardless of the damage location. Thus, the results suggest that the model is 99.9% accurate for detecting the presence of damage. Similarly, the confusion matrix of network $k = 2$ shows only 2 misclassifications out of 825 observations. This suggest that

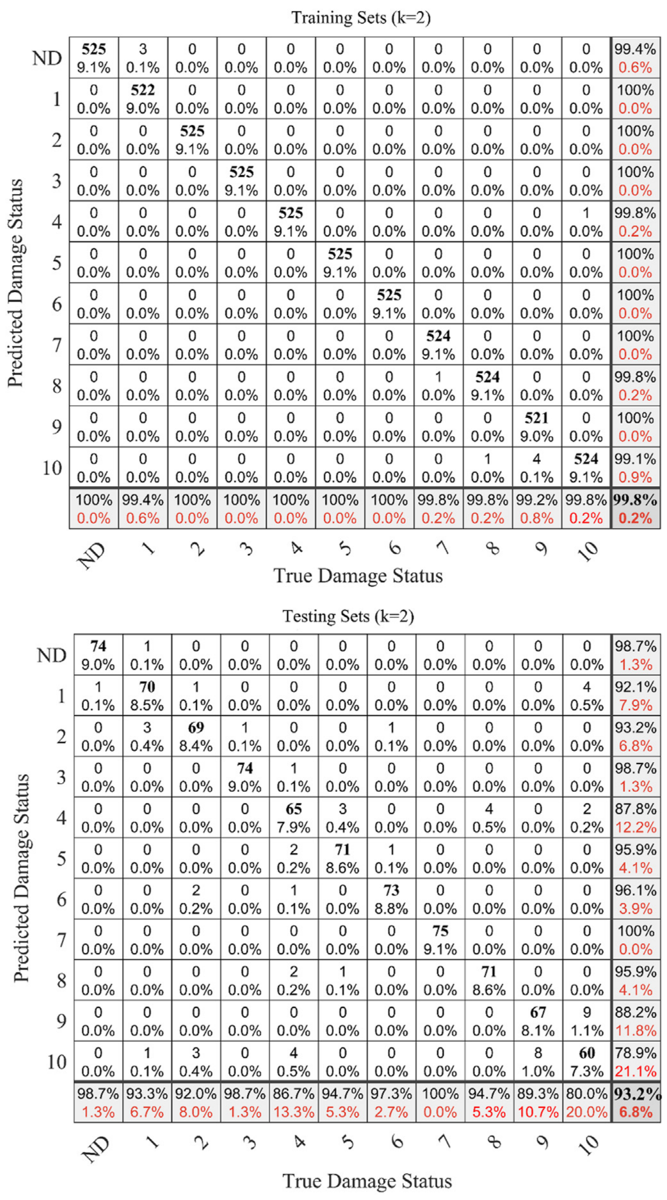
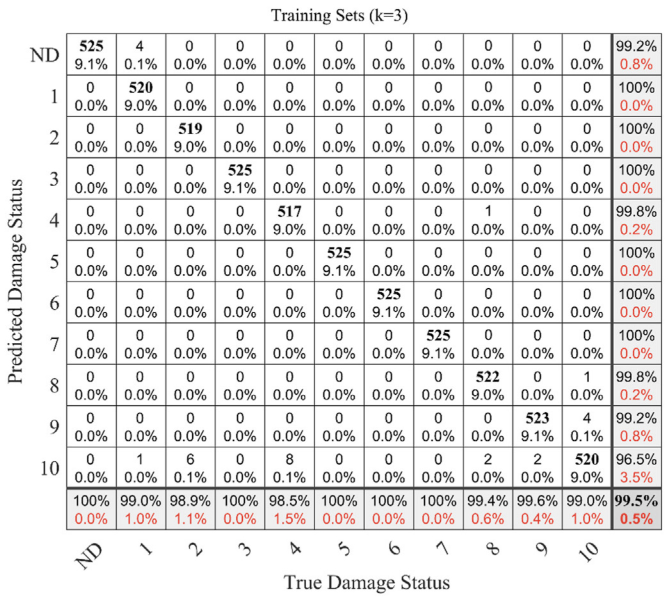


Fig. 13. Confusion matrix of the selected model, $k = 2$.

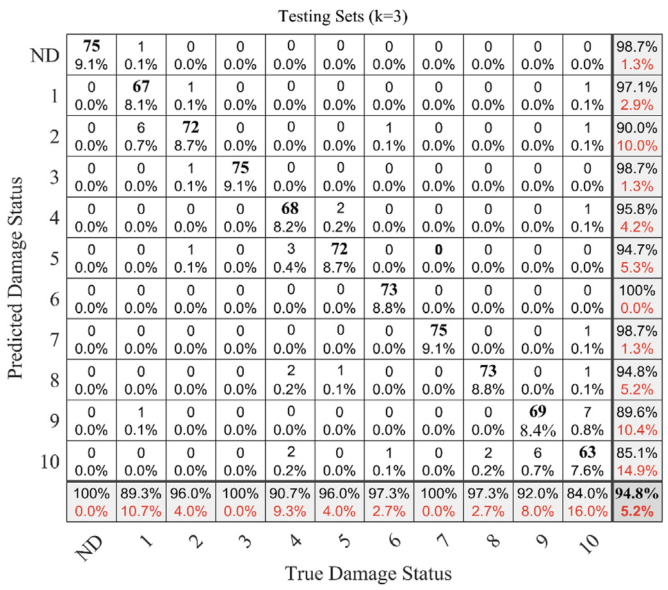
the model detected the presence of damage with 99.8% accuracy.

5.3. Selection of model complexity

In addition to the depth of the network layers, model complexity is also determined by the number of hidden units. A higher number of hidden units increases the model complexity which may lead to overfit the data and produce a biased model. On the other hand, an insufficient number of hidden units decreases the performance of the model. The relationship between model complexity, errors, and model bias (underfitting or overfitting) was


Training Sets (k=3)

Predicted Damage Status \ True Damage Status	ND	1	2	3	4	5	6	7	8	9	10	
ND	**525** 9.1%	4 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.2% 0.8%
1	0 0.0%	**520** 9.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
2	0 0.0%	0 0.0%	**519** 9.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	**525** 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**517** 9.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	99.8% 0.2%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**525** 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**525** 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**525** 9.1%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**522** 9.0%	0 0.0%	1 0.0%	99.8% 0.2%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**523** 9.1%	4 0.1%	99.2% 0.8%
10	0 0.0%	1 0.0%	6 0.1%	0 0.0%	8 0.1%	0 0.0%	0 0.0%	0 0.0%	2 0.0%	2 0.0%	**520** 9.0%	96.5% 3.5%
	100% 0.0%	99.0% 1.0%	98.9% 1.1%	100% 0.0%	98.5% 1.5%	100% 0.0%	100% 0.0%	100% 0.0%	99.4% 0.6%	99.6% 0.4%	99.0% 1.0%	**99.5% 0.5%**


Testing Sets (k=3)

Predicted Damage Status \ True Damage Status	ND	1	2	3	4	5	6	7	8	9	10	
ND	**75** 9.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.7% 1.3%
1	0 0.0%	**67** 8.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	97.1% 2.9%
2	0 0.0%	6 0.7%	**72** 8.7%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	90.0% 10.0%
3	0 0.0%	0 0.0%	1 0.1%	**75** 9.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.7% 1.3%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**68** 8.2%	2 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	95.8% 4.2%
5	0 0.0%	0 0.0%	1 0.1%	0 0.0%	3 0.4%	**72** 8.7%	0 0.0%	**0** 0.0%	0 0.0%	0 0.0%	0 0.0%	94.7% 5.3%
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**73** 8.8%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	100% 0.0%
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**75** 9.1%	0 0.0%	0 0.0%	1 0.1%	98.7% 1.3%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.2%	1 0.1%	0 0.0%	0 0.0%	**73** 8.8%	0 0.0%	1 0.1%	94.8% 5.2%
9	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	**69** 8.4%	7 0.8%	89.6% 10.4%
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.2%	0 0.0%	1 0.1%	0 0.0%	2 0.2%	6 0.7%	**63** 7.6%	85.1% 14.9%
	100% 0.0%	89.3% 10.7%	96.0% 4.0%	100% 0.0%	90.7% 9.3%	96.0% 4.0%	97.3% 2.7%	100% 0.0%	97.3% 2.7%	92.0% 8.0%	84.0% 16.0%	**94.8% 5.2%**

Fig. 14. Confusion matrix of the selected model, $k = 3$.

discussed in the earlier section.

Fig. 15 shows the training progress of Network II trained with set D by decreasing the number of hidden units: (a) 256 units; (b) 128 units; (c) 96 units and (d) 64 units. Left side figures of (a), (b), (c), and (d) show the model accuracies of training sets (solid line) and validation sets (triangle mark) during training progress. The figures on the right-side show the loss function. At this stage, all models use a constant learning rate of 0.001 to determine model complexity. The ideal progress is to show the validation accuracy as close as possible to the training accuracy throughout the progression while maintaining the highest level of accuracy. Although an increase in the number of hidden units leads to similar validation accuracies, 87.33%, 86.75%, 88.92% and 88.92%, the gap between the training (solid line) and validation (triangle mark) accuracies increases over iterations in figure (a) and (b). In addition, the loss functions increase over iterations in the two figures of (a) and (b). These results indicate that the model slightly overfits the training data. In contrast, Fig. 15 (c) and (d) show stabilized loss compared to (a) and (b) and their validation accuracies are close to the training accuracies. Fig. 15 (c) shows the accuracy level of 88.92% and (d) show 88.90%. Using these results, we chose to further investigate models with 64 and 96 hidden units and to fine-tune our approach as detailed in the following section.

5.4. Improving accuracy

After the model complexity is chosen based on the depth of network layers and the number of hidden units, networks were fine-tuned to achieve the highest accuracy. As detailed in earlier sections, model hyperparameters such as learning rate, minimum batch size, and maximum number of epochs were controlled in order to improve accuracy. The most critical hyperparameter to improve the model is the learning rate (LR) which controls the changes of the unknown parameters of $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{V}, \boldsymbol{b}\}$. The expression of the changes in the stochastic optimization algorithm is $\boldsymbol{\theta}_{\boldsymbol{i}+1} = \boldsymbol{\theta}_{\boldsymbol{i}} - \alpha\nabla l(\boldsymbol{\theta}_{\boldsymbol{i}})$ where $\alpha > 0$ is the learning rate and $l(\boldsymbol{\theta}_{\boldsymbol{i}})$ is the loss function from the entire data set. Note that $\boldsymbol{\theta}$ is updated at each batch, a subset of the entire dataset, while $l(\boldsymbol{\theta}_{\boldsymbol{i}})$ uses the entire data used for training. A batch size of 50 was used for our research. The dynamics between the hyperparameters allows the model to use a fixed batch size at this reasonable level. We proceeded to control the learning rate for each network design. Table 4 shows the fine-tuning progress of the selected model D-II. The test set A is for the network with 64 hidden units and set B is the progress for the network with 96 hidden units. Fine-tuning of the network resulted in the highest testing accuracy for test B5 which reached 92.7% and used for the final configuration of D-II model. The details of 24 fine-tuned network models (A-I to F-IV) were summarized in Table 2.

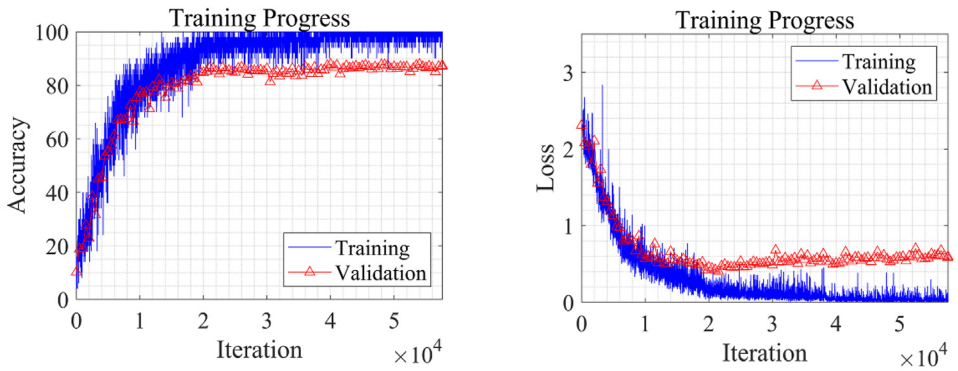
5.5. Effects of wavelet filters and denoise

In this section, the effects of added features from the wavelet filter and the denoise technique are presented. Data treatments other than data normalization did not significantly improve the results of the training and predictions. Instead, they interfered with the training process. Since the data is simulated and lacks the noise expected from the field sensors, we expected this outcome. However, the results are presented here because data treatments are a critical part of the framework for increasing the performance of models and will play a significant role to train a network with field data. The results do not exclude the needs of the wavelet filters and wavelet denoise techniques when the network is trained with field data or when the data type is dissimilar to the data used in this research.

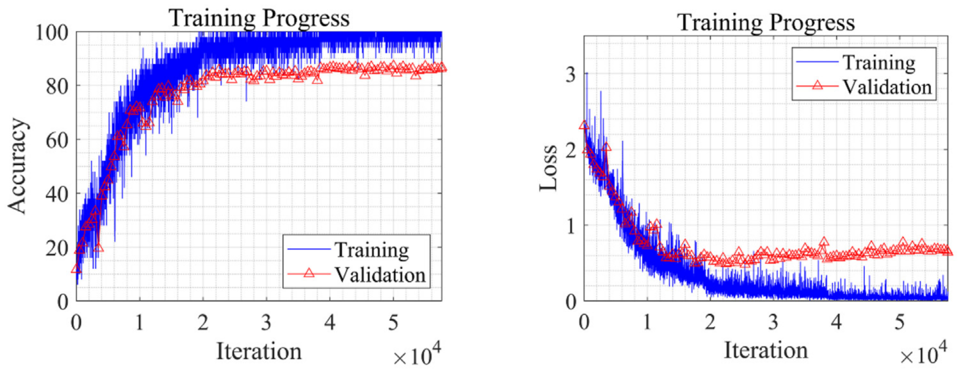
Table 5 shows the results of model testing and training accuracy after fine-tuning. Samples of treated data were presented in Figs. 7, Figure 8, and Fig. 9. Results from models trained by denoised data are listed according to the level of denoise, 1 to 3, and compared with the results of the untreated data shown in the top four rows of the table. It was observed that the denoise process removes necessary information from the signals in this research. Therefore, the performance of the model decreases as the denoise level increases. This may be due to the fact that the training data is simulated and without noise. The bottom three rows of Table 5 show the results of the fine-tuned model trained by data sets with added features from the wavelet filter. cD*i*, which indicates the detailed coefficient at the level of *i*, the output from the HP filters at each level. For the type of data used in our research, added features did not improve the training results.

6. Conclusion

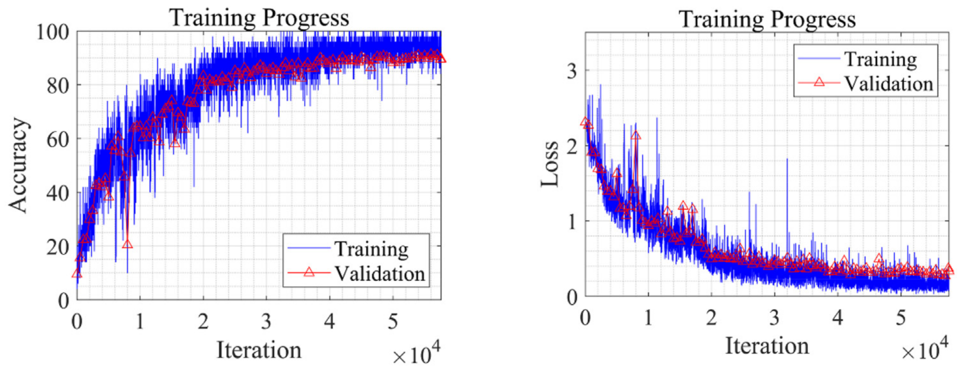
A sequence-based modeling and prediction method of DL was proposed and tested for structural damage detection of FOWT



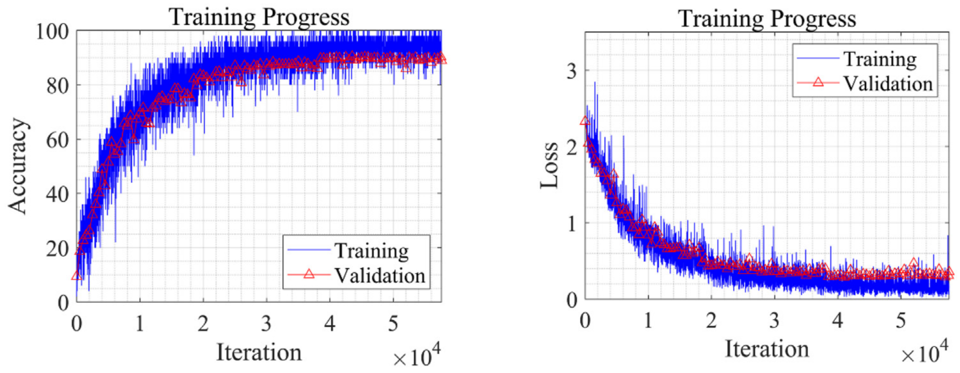
(a) 256 hidden units, validation accuracy 87.33 %; training accuracy of 95.73%



(b) 128 hidden units, validation accuracy 86.75 %; training accuracy of 98.35%



(c) 96 hidden units, validation accuracy 88.92 %; training accuracy of 99.69%



(d) 64 hidden units, validation accuracy 88.90 %; training accuracy of 95.17%

Fig. 15. Comparison by the number of hidden units.

Table 4
Accuracies by learning rate (LR) during the fine-tuning.

Test Network	Testing Accuracy	Training Accuracy	Initial LR	LR Drop (rate/freq.)	# of Hidden Unit	Batch Size	Max Epoch
A1	0.777	0.847	0.0005	0.5/200	64	50	600
A2	0.850	0.920	0.0008	0.5/200	64	50	600
A3	0.889	0.952	0.001	0.5/200	64	50	600
A5	0.897	0.999	0.0035	0.4/100	64	50	600
A6	0.886	0.998	0.007	0.3/100	64	50	600
B1	0.825	0.880	0.0005	0.5/200	96	50	600
B2	0.880	0.942	0.001	0.5/E200	96	50	600
B3	0.889	0.997	0.0035	0.4/E100	96	50	600
B4	0.917	0.998	0.005	0.5/E100	96	50	600
B5	0.948	0.997	0.007	0.3/E100	96	50	600

Table 5
Training results of wavelet filters and denoise for case D-II.

Treatment Method	Levels	Testing Accuracy	Training Accuracy	Initial LR	LR Drop (rate/freq.)	# of Hidden Unit	Batch Size	Max Epoch
Untreated		0.927	0.997	0.007	0.3/100	96	50	600
Wavelet Denoise	Level 1	0.848	1.000	0.007	0.3/100	96	50	600
	Level 2	0.803	0.999	0.007	0.3/100	96	50	600
	Level 3	0.450	1.000	0.007	0.3/100	96	50	600
Wavelet Filter	cD1	0.830	0.979	0.001	0.9/100	128	50	600
	cD1 & cD2	0.823	0.991	0.005	0.3/100	200	50	600
	cD1, cD2, & cD3	0.843	0.974	0.005	0.3/100	200	50	600

blades, using uni/bi-directional LSTM and GRU neural networks. The complete framework was developed and presented. A total of 1200 damage scenarios were simulated with various shapes, intensities, and locations of 5 MW semisubmersible FOWT blades, in addition to 120 simulations of undamaged FOWT. Various data treatment methods were applied and tested to improve the performance of the network. We found that the wavelet filter and denoise treatments did not improve or interfere with the network training performance in this research. This may be because the simulated data does not likely contain the noises commonly expected from the field data. Therefore, the data treatments should not be excluded if a network is trained by field data or if the data attribution is dissimilar to the one used in this research.

Four different deep network designs were proposed and tested. Overall, the network with a single GRU unit (Network II) performed better than the other networks in training numerical sensor signals from the FOWTs. However, the results from the deep network with a stacked BiLSTM network (Network IV) demonstrated that this configuration is comparable to one with GRU networks depending on data attribution. The difference in performance between the networks with single and double layers of BiLSTM was not significant.

The presence of structural damage was successfully detected using the selected network (Network II) with a 99.9% accuracy, which was tested with independent testing signals and repeated for *K* randomly-split groups during the *K*-fold cross-validation. The best network during *K*-fold validation reached 100% accuracy. The overall damage status, which indicates both the presence and its location, was detected with an accuracy of 91.7.% based on *K*-fold cross-validation. The testing accuracy of the best model reaches 94.8% in the detection of damage status at $k = 3$. Out of 5.2% of incorrect predictions in this model, 3.1% fall in the damage location within 1-segment distance from the true damage location. Only 2.1% of the testing signals are misclassified more than 1 segment distance from the true damage location. *K*-fold cross-validation was performed to obtain unbiased results and to calculate the estimators of the prediction accuracies. The prediction of damage status was not significantly improved after using more than five features (sensors). The future work of this research will be a systematic DL monitoring of the entire FOWT system. In addition to the structural failure of blades, holistic system failure scenarios will be considered including structural, mechanical, and operational failures. Alternative DL methods will also be considered.

The methodology and framework of the sequence-based modeling provided in this paper can be applied to various engineering research fields especially in data-driven modeling. We believe that sequence-based modeling will enable the engineers to advance the technologies and knowledge in various engineering fields by harnessing the massive amounts of digital informational currently being cumulated.

Funding acknowledgements

This material is based upon work supported by the National Science Foundation (NSF) under Grant #2118586. This material is also based upon work supported by the National Science Foundation under Grant # 1914692.

Data accessibility statement

Some or all data, models, or code generated or used during the study will be available through the funding agency (National Science Foundation, United States) and are also available from the corresponding author by request.

CRediT authorship contribution statement

Do-Eun Choe: Conceptualization, Methodology, Formal analysis, Funding acquisition, Investigation, Software, Project administration, Validation, Visualization, Writing – original draft, Writing – review & editing. **Hyoung-Chul Kim:** Data curation, Software, Validation, Writing – original draft, Writing – review & editing. **Moo-Hyun Kim:** Software, Writing – review & editing, Supervision.

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.

References

[1] Y. Liu, S. Li, Q. Yi, et al., Developments in semi-submersible floating foundations supporting wind turbines: a comprehensive review, Renew. Sustain. Energy Rev. 60 (2016) 433–449.

[2] M. Froese, World's first floating wind farm delivers promising results, World's first floating wind farm delivers promising results, https://www.windpowerengineering.com/worlds-first-floating-wind-farm-delivers-promising-results/, 2018.

[3] Allen HL, Goupee AJ, Viselli AM, et al. Experimental comparison of an annular floating offshore wind turbine hull against past model test data. J. Offshore Mech. Arctic Eng.; 142.

[4] A.C. Levitt, W. Kempton, A.P. Smith, et al., Pricing offshore wind power, Energy Pol. 39 (2011) 6408–6421.

[5] C. Moné, M. Hand, M. Bolinger, et al., Cost of Wind Energy Review, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2015, 2017.

[6] M.W. Häckell, R. Rolfes, M.B. Kane, et al., Three-tier modular structural health monitoring framework using environmental and operational condition clustering for data normalization: validation on an operational wind turbine system, Proc. IEEE 104 (2016) 1632–1646.

[7] Z. Hameed, Y. Hong, Y. Cho, et al., Condition monitoring and fault detection of wind turbines and related algorithms: a review, Renew. Sustain. Energy Rev. 13 (2009) 1–39.

[8] Griffith DT, Yoder N, Resor B, et al. Structural Health and Prognostics Management for Offshore Wind Turbines: an Initial Roadmap. *SAND2012-10109 Sandia Natl Lab*.

[9] W. Weijtjens, R. Shirzadeh, G. De Sitter, et al., Classifying resonant frequencies and damping values of an offshore wind turbine on a monopile foundation for different operational conditions, Proc. European Wind Energy Association, EWEA (2014).

[10] H.-C. Kim, M.-H. Kim, D.-E. Choe, Structural health monitoring of towers and blades for floating offshore wind turbines using operational modal analysis and modal properties with numerical-sensor signals, Ocean. Eng. 188 (2019), 106226.

[11] D. Tcherniak, S. Chauhan, M.H. Hansen, Applicability Limits of Operational Modal Analysis to Operational Wind Turbines, in: Structural Dynamics and Renewable Energy, ume 1, Springer, 2011, pp. 317–327.

[12] N. Dervilis, M. Choi, S. Taylor, et al., On damage diagnosis for a wind turbine blade using pattern recognition, J. Sound Vib. 333 (2014) 1833–1850.

[13] S. Tsiapoki, M.W. Häckell, T. Grießmann, et al., Damage and ice detection on wind turbine rotor blades using a three-tier modular structural health monitoring framework, Struct. Health Monit. 17 (2018) 1289–1312.

[14] L. Wang, Z. Zhang, J. Xu, et al., Wind turbine blade breakage monitoring with deep autoencoders, IEEE Trans Smart Grid 9 (2016) 2824–2833.

[15] M.H. Hassoun, Fundamentals of Artificial Neural Networks, MIT press, 1995.

[16] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Network. 61 (2015) 85–117.

[17] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.

[18] C. Rainieri, G. Fabbrocino, Operational Modal Analysis of Civil Engineering Structures vol. 142, Springer N Y, 2014, p. 143.

[19] P. Qian, X. Ma, P. Cross, Integrated data-driven model-based approach to condition monitoring of the wind turbine gearbox, IET Renew. Power Gener. 11 (2017) 1177–1185.

[20] H. Zhao, H. Liu, W. Hu, et al., Anomaly detection and fault analysis of wind turbine components based on deep learning network, Renew. Energy 127 (2018) 825–834.

[21] S. Bogoevska, M. Spiridonakos, E. Chatzi, et al., A data-driven diagnostic framework for wind turbine structures: a holistic approach, Sensors 17 (2017) 720.

[22] S. Yin, G. Wang, H.R. Karimi, Data-driven design of robust fault detection system for wind turbines, Mechatronics 24 (2014) 298–306.

[23] J. Steiner, R. Dwight, A. Viré, Data-driven Turbulence Modeling for Wind Turbine Wakes under Neutral Conditions, IOP Publishing, 2020, 062051.

[24] H.-J. Kim, B.-S. Jang, C.-K. Park, et al., Fatigue analysis of floating wind turbine support structure applying modified stress transfer function by artificial neural network, Ocean. Eng. 149 (2018) 113–126.

[25] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Network. 5 (1994) 157–166.

[26] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. ArXiv Prepr ArXiv14061078..

[27] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv Prepr ArXiv14090473..

[28] Chung J, Gulcehre C, Cho K, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv Prepr ArXiv14123555..

[29] J. Jonkman, S. Butterfield, W. Musial, et al., Definition of a 5-MW reference wind turbine for offshore system development, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009.

[30] A.J. Coulling, A.J. Goupee, A.N. Robertson, et al., Validation of a FAST semi-submersible floating wind turbine numerical model with DeepCwind test data, J. Renew. Sustain. Energy 5 (2013), 023116.

[31] J. Jonkman, M. Buhl, New Developments for the NWTC's Fast Aeroelastic HAWT Simulator, 2004, p. 504.

[32] M. Masciola, A. Robertson, J. Jonkman, et al., Investigation of a FAST-OrcaFlex Coupling Module for Integrating Turbine and Mooring Dynamics of Offshore Floating Wind Turbines, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2011.

[33] F. Wendt, A. Robertson, J. Jonkman, et al., Verification and Validation of the New Dynamic Mooring Modules Available in FAST V8, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2016.

[34] A. Coulling, A. Goupee, A. Robertson, et al., Validation of a FAST Floating Wind Turbine Model Using Data from the DeepCwind Semi-submersible Model Tests, 2013.

[35] Lee C, Newman J. The Computation of Second-Order Wave Loads..

[36] G. Bir, User's guide to BModes (software for computing rotating beam-coupled modes), National Renewable Energy Lab.(NREL), Golden, CO (United States), 2005.

[37] X. Qu, Y. Li, Y. Tang, et al., Comparative study of short-term extreme responses and fatigue damages of a floating wind turbine using two different blade models, Appl. Ocean Res. 97 (2020), 102088.

[38] B.J. Jonkman, TurbSim User's Guide: Version 1.50, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2009.

[39] International Electrotechnical Commission. Wind Turbine Generator Systems-Part 1: Safety Requirements. IEC 61400-1..

[40] International Electrotechnical Commission. Wind Turbines-Part 1: Design Requirements. IEC 614001 Ed 3.

[41] J.P. Conte, X. He, B. Moaveni, et al., Dynamic testing of Alfred Zampa memorial bridge, J. Struct. Eng. 134 (2008) 1006–1015.

[42] Brownjohn J, Pavic A, Carden E, et al. Modal Testing of Tamar Suspension Bridge.

[43] F.M.R.L.D. Magalhães, Operational modal analysis for testing and monitoring of bridges and special structures, 2010.

[44] White JR. Operational Monitoring of Horizontal axis Wind Turbines with Inertial Measurements..

[45] F. Liu, H. Li, S.-L.J. Hu, Stochastic Modal Analysis for a Real Offshore Platform, 2015, pp. 12–14.

[46] C. Ruzzo, G. Failla, M. Collu, et al., Operational modal analysis of a spar-type floating platform using frequency domain decomposition method, Energies 9 (2016) 870.

[47] Devriendt C, Elkafafy M, De Sitter G, et al. Continuous Dynamic Monitoring of an Offshore Wind Turbine on a Monopile Foundation. ISMA2012..

[48] G. De Sitter, W. Weijtjens, M. El-Kafafy, et al., Monitoring Changes in the Soil and Foundation Characteristics of an Offshore Wind Turbine Using Automated Operational Modal Analysis, Trans Tech Publ, 2013, pp. 652–659.

[49] C. Devriendt, P.J. Jordaens, G. De Sitter, et al., Damping estimation of an offshore wind turbine on a monopile foundation, IET Renew. Power Gener. 7 (2013) 401–412.

[50] M.M. Shokrieh, R. Rafiee, Simulation of fatigue failure in a full composite wind turbine blade, Compos. Struct. 74 (2006) 332–342.

[51] S. Kim, D.E. Adams, H. Sohn, et al., Crack detection technique for operating wind turbine blades using Vibro-Acoustic Modulation, Struct. Health Monit. 13 (2014) 660–670.

[52] Gers FA, Schmidhuber J, Cummins F. Learning to Forget: Continual Prediction with LSTM..

[53] R. Zhao, R. Yan, J. Wang, et al., Learning to monitor machine health with convolutional bi-directional LSTM networks, Sensors 17 (2017) 273.

[54] S.G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. 11 (1989) 674–693.

[55] R.J. Martis, U.R. Acharya, L.C. Min, ECG beat classification using PCA, LDA, ICA and discrete wavelet transform, Biomed. Signal Process Contr. 8 (2013) 437–448.

[56] M. Thomas, M.K. Das, S. Ari, Automatic ECG arrhythmia classification using dual tree complex wavelet based features, AEU-Int J Electron Commun 69 (2015) 715–721.

[57] S. Sahoo, B. Kanungo, S. Behera, et al., Multiresolution wavelet transform based feature extraction and ECG classification to detect cardiac abnormalities, Measurement 108 (2017) 55–66.

[58] Ö. Yildirim, A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification, Comput. Biol. Med. 96 (2018) 189–202.

[59] I. Daubechies, Ten Lectures on Wavelets, SIAM, 1992.

[60] D.L. Donoho, J.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, Biometrika 81 (1994) 425–455.

[61] Cui Z, Ke R, Pu Z, et al. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. ArXiv Prepr ArXiv180102143..

[62] M. Stone, Cross-validatory choice and assessment of statistical predictions, J R Stat Soc Ser B Methodol 36 (1974) 111–133.

[63] S. Geisser, A predictive approach to the random effect model, Biometrika 61 (1974) 101–107.

[64] S. Geisser, The predictive sample reuse method with applications, J. Am. Stat. Assoc. 70 (1975) 320–328.
[65] T.G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Comput. 10 (1998) 1895–1923.
[66] Y. Bengio, Y. Grandvalet, No unbiased estimator of the variance of k-fold cross-validation, J. Mach. Learn. Res. 5 (2004) 1089–1105.
[67] A. Blum, A. Kalai, J. Langford, Beating the Hold-Out: Bounds for K-fold and Progressive Cross-Validation, 1999, pp. 203–208.