FROM CLASSICAL MODELS TO NEURAL OPERATORS: A STUDY ON MOORING TENSION PREDICTION FOR FLOATING OFFSHORE WIND TURBINE

Lang-Wei Zhong [1, 2], Yu-Ping Lan [1, 2], Xi-Wei Tang [1], Wei Huang [1, 2, *], Jian-Wen He [3] and Si-Wei Liu [4]

[1] *School of Civil Engineering, Sun Yat-Sen University, Guangzhou, China*
[2] *State Key Laboratory of Coastal and Offshore Engineering, Dalian University of Technology, Dalian, China*
[3] *Research Center of Submarine Pipeline and Cable, Research Institute of Tsinghua University in Shenzhen, Shenzhen, China*
[4] *Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China*
** (Corresponding author: E-mail: huangw288@mail.sysu.edu.cn)*

ABSTRACT

Offshore wind technology has emerged as a promising solution for harnessing wind resources in both shallow and deep waters. For floating offshore wind turbines, mooring systems play a critical role in maintaining station-keeping functions. Effective monitoring of mooring loads is essential to ensure safe and cost-efficient operation and maintenance. However, direct measurement of dynamic mooring line tensions is often costly and impractical. To address this challenge, this paper focuses on predicting mooring tensions using accessible motion data from the floating platform. We propose the use of deep learning algorithms, including deep neural networks and deep operator networks, to predict the mooring tension of floating offshore wind turbines based on six-degree-of-freedom (6-DOF) platform motions from OpenFAST or OrcaFlex under extreme sea conditions as input. We compare the prediction accuracy and generalization performance of various models, with the Transformer model under a local attention mechanism (LA-Transformer) and the multi-input deep operator network with an attention mechanism (MIONet-ATT) standing out as the top performers. Specifically, LA-Transformer inherits the precise and efficient foundational architecture of Transformer while pruning timing-independent components, achieving an optimal balance between computational complexity and accuracy. It is a novel model with outstanding precision and a moderate parameter size. MIONet-ATT, designed based on the concept of operator networks, offers better interpretability and generalization. Additionally, it demonstrates promising potential in non-extreme sea state applications.

ARTICLE HISTORY

Received:	16 February 2025
Revised:	7 August 2025
Accepted:	7 August 2025

KEYWORDS

Neural networks;
Deep operator networks;
Semi-submersible floating offshore wind turbine;
Moooring tension;
Transformer

1. Introduction

As the effects of global climate change and environmental degradation intensify, the development of renewable energy has emerged as a key priority for governments and research institutions worldwide. Wind energy, as a clean and inexhaustible resource, is particularly significant in supporting global sustainability initiatives [1, 2]. Offshore wind power presents unique benefits compared to onshore wind farms, including richer wind reserves, reduced noise disturbances, and no competition for land resources.

Offshore wind turbines are typically categorized into three groups depending on water depth: shallow (0–30m), intermediate (30–60m), and deep-water installations (beyond 60m). In deeper marine areas, floating offshore wind turbines (FOWTs) are deployed to maximize energy extraction. However, predicting the dynamic behavior of FOWTs remains a complex challenge due to their intricate coupling with oceanic conditions. Reliable response predictions are crucial for operational supervision, maintenance planning, and adaptive control strategies, all of which are vital for ensuring safe and efficient performance [3].

For example, anticipating mooring line tension with high accuracy allows for preemptive actions to mitigate risks such as structural damage or system failure, while also improving cost-effectiveness. Precise tension forecasting not only enhances the operational reliability of FOWTs but also extends their service life and optimizes energy production [4].

To assess the structural integrity of mooring systems and reduce the likelihood of degradation—including corrosion, wear, and fatigue—routine inspections are essential. At present, in-service physical examinations remain the most trusted method [5, 6]. However, this inspection-based approach for evaluating mooring system health is both expensive and time-consuming, similar as other topics like local scour of the monopile based on CFD-DEM [7, 8], dynamic analysis and fatigue analysis for TLP in time domain [9, 10]. As a result, researchers and industry professionals are increasingly shifting their attention toward digital real-time monitoring solutions.

Previous approaches to calculating the dynamic response of mooring lines have largely relied on numerical simulation techniques, exemplified by the contributions of Hall et al. [1], Chen et al. [2], and Zhang et al. [3], which encompass methods such as finite elements [11-14], finite difference, and lumped mass models. However, a significant challenge for these numerical methods remains achieving a balance between computational accuracy and efficiency, rendering them inadequate for the task of providing high-precision, real-time predictions of mooring tension. In their work, Sidarta et al. [4] leveraged an Artificial Neural Network (ANN) to forecast mooring line tension, utilizing vessel motion data as input parameters. Upon training, this ANN model demonstrated the capability to predict mooring line tensions across diverse sea conditions. Qiao et al. [5] observed that identical vessel motions could result in differing mooring line tensions, attributed to the vessel's reciprocating movements. This realization underscored the importance of capturing the temporal sequence dynamics between vessel motion and mooring line tension. To address this, they employed a Long Short-Term Memory (LSTM) network, a potent deep learning architecture, to develop a real-time predictor for mooring line responses. Recognizing that mooring line tension often exhibits both low-frequency and wave-frequency components, Wang et al. [6] adopted a preprocessing step involving a low-pass filter to segregate the cable tension and platform motion into their respective wave-frequency and low-frequency constituents within the training dataset. Subsequently, they utilized two separate LSTM models to train and predict the low-frequency and wave-frequency components of the cable tension, ultimately combining these predictions. Huang, Huang and Tang [15] have applied the LSTM method to predict the mooring tensions of the spread catenary and taut mooring system considering the dynamic stiffness of the polyester ropes, Tang et al. predicted the mooring tension of the FOWT combined the CNN-LSTM-ATT method and Chebyshev polynomials [16].

In a comparative analysis, Payenda et al. [17] evaluated three recurrent neural network models for predicting mooring cable tension in semi-submersible floating wind turbines, concluding that the Bidirectional LSTM model exhibited superior performance. In mostly previous studies, the input data for ANNs in the field of mooring line tension prediction were the six-degree-of-freedom (6-DOF) responses of the floater. The responses of floating platform motions can be readily obtained in practice through a global positioning system (GPS), which offers high accuracy and stability, or by utilizing sensors such as gyroscopes, accelerometers, magnetometers, or other direct and convenient measurement devices [18].

We believe that the task of predicting mooring line tension based on platform motion falls into the category of short-term dependent time series prediction. Although the platform's reciprocating motion may lead to different mooring line tensions at the same position, the current mooring line tension is predominantly influenced by recent motions rather than those from the distant past. As a result, LSTM is undoubtedly the benchmark model for this prediction task and remains the mainstream approach in existing literature. In addition to LSTM, other models such as one-dimensional convolutional neural networks (CNNs), GRU, BiLSTM and modified Transformers are also suitable for short-term dependencies. It is worth noting that while native Transformers are primarily designed for handling long-term dependencies[9], they can be adapted to short-term tasks by limiting attention windows or adjusting network architectures. Furthermore, hybrid models combining these techniques may offer additional benefits [19]. Besides these deep neural network models, we also explore the use of neural operators like DeepONet for this prediction task. Neural operators are designed to learn mappings between functional spaces and offer strong generalization capabilities. Compared to the criticized "black box property" of traditional deep neural network models, deep operator networks are considered to have better interpretability [20]. In recent years, this method has gained increasing application in the field of ocean engineering. Cao et al. [21] conducted a comprehensive evaluation of three neural operators - the Deep Operator Network (DeepONet), Fourier Neural Operator (FNO), and Wavelet Neural Operator (WNO) - for predicting floating structures' dynamic responses. Their findings demonstrated that these neural operators outperform GRU models in terms of prediction accuracy.

Subsequent research has further advanced this field. Zhao et al. [22] developed a fully adaptive time-frequency coupling forecasting model (FTD) by integrating deep operator networks with a self-attention mechanism. This innovative approach exhibited superior prediction accuracy and generalization capability compared to the original DeepONet model. In another study, Zhang et al. [23] proposed a Multiple-Input Operator Network (MIONet), which demonstrated enhanced accuracy and broader applicability when benchmarked against both LSTM and DeepONet models. Similarly, Zhao et al. [22] confirmed that the Multi-Branch DeepONet Model achieves higher precision in ship motion prediction, reinforcing the effectiveness of neural operator-based approaches in marine engineering applications.

In recent years, the prediction of mooring line tension has been extensively studied in ocean engineering, with various deep learning models being employed for this purpose. Long Short-Term Memory (LSTM) networks have emerged as the most prevalent approach, alongside Gated Recurrent Units (GRU) and Transformer architectures. Earlier research also utilized Backpropagation (BP) neural networks for this task. While newer approaches like Deep Operator Networks (DeepONet) have begun gaining attention in ocean engineering applications, their specific application to mooring line tension prediction remains unexplored in the literature.

This study begins by examining the fundamental paradigms of deep neural networks before delving into the methodology of deep operator networks. Building on this foundation, the paper proposes a novel operator network architecture that incorporates dual branches for displacement and velocity inputs. The research systematically compares the predictive accuracy and generalization capabilities of traditional deep learning models with those of neural operator approaches, providing valuable insights into their respective strengths and limitations.

The novelty of this paper lies in three key aspects. First, this study addresses the mooring line tension prediction problem for floating wind turbines from a dual perspective of ocean engineering and artificial intelligence. It conducts a comprehensive comparative analysis of time-series prediction neural network models designed for short-term dependencies. Second, it formulates the mooring line tension as a functional relationship dependent on the position and velocity of the fairlead. This relationship is derived by applying position and velocity operators to time-series data of mooring tension and 6-DOF platform motions, with DeepONet utilized to execute these operator-based computations. Third, the paper contrasts mature, well-established, and validated neural network models with neural operators, which are particularly suitable for modeling physical fields. These three aspects have not been thoroughly explored in existing literature, making this study an innovative contribution to the field.

2. Methodology

2.1. Numerical modeling of floating offshore wind turbine

A numerical model of the OC4-DeepCWind semi-submersible platform supporting the NREL 5-MW wind turbine was simulated using OrcaFlex and OpenFAST. OrcaFlex is designed for analyzing the dynamic behavior of offshore structures in marine and subsea applications. It models the response of floating and fixed structures to environmental forces, including waves, wind, and currents [24]. The software calculates hydrodynamic loads using the Morison equation for slender structures and potential flow theory for surrounding fluids. Aerodynamic loads on the wind turbine are computed using the blade element momentum (BEM) theory. For mooring systems, OrcaFlex employs the lumped mass method, solving mooring chain tensions through numerical techniques like the Runge-Kutta method and General alpha method. By providing time-domain simulations, OrcaFlex enables the evaluation of motion responses of floating offshore wind turbines (FOWTs), including mooring tensions, displacements, and dynamic stability under varying environmental conditions. In contrast, OpenFAST is an open-source, integrated simulation tool developed by the National Renewable Energy Laboratory (NREL) for modeling the dynamic behavior of wind turbines. It simulates wind turbine responses under various loading conditions, incorporating aeroelasticity, hydrodynamics, and control systems. This research focuses on the OC4-DeepCWind semi-submersible platform, which features three side columns arranged around a central column supporting the wind turbine. The angle between any two adjacent side columns is 120 degrees, as illustrated in Fig. 1. The mooring system of the floating wind turbine consists of three catenary lines, with a platform draft of 20m and a mooring anchor depth of 200m. The mooring tension predicted in this article refers to the fairlead tension of the mooring line aligned with the wind direction, such as Line 1 in Fig. 1, which has the maximum tension and deformation.

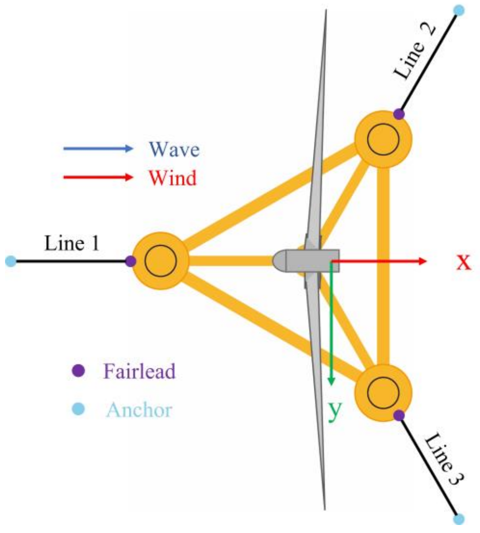


Fig. 1 The wind turbine semi-submersible platform diagram

2.2. Recurrent neural networks

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed for sequential data, where the temporal dependency among data points is crucial. By leveraging internal memory, RNNs can retain information about previous inputs, enabling them to process sequences of arbitrary length. However, standard RNNs face challenges when dealing with long-term dependencies due to the vanishing or exploding gradient problem during backpropagation through time (BPTT) [25].

LSTM networks were introduced as an extension to RNNs to address the issue of long-term dependency. LSTMs use a unique architecture incorporating three gates—input, forget, and output gates—along with a cell state to regulate the flow of information, as shown in Fig. 2. The forget gate allows LSTMs to selectively retain or discard information, while the input and output gates control the addition of new information and its propagation to the next time step. This gating mechanism makes LSTMs highly effective for capturing both short-term and long-term temporal patterns, particularly in tasks such as time-series prediction, language modeling, and speech recognition. It is worth mentioning that while LSTM is designed as an improved version of RNN specifically to address long-term dependencies, it also excels in handling short-term dependencies. This capability is attributed to the gate mechanisms of LSTM, which provide the flexibility to dynamically balance and process both short-term and long-term dependencies simultaneously [26].

The output of LSTM is denoted as y_t, and the output of LSTM cell h_t at the time t can be calculated as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$\widetilde{C_t} = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t} \tag{5}$$

$$h_t = o_t \odot \tanh(C_t) \tag{6}$$

$$y_t = \sigma(W_y h_t + b_y) \tag{7}$$

where σ and $tanh$ are the sigmoid activation function and hyperbolic tangent activation function, respectively; i_t denotes input gate, f_t denotes forget gate, o_t denotes output gate, $\odot$ denotes the product of vector elements; W_*, U_* and b_* are the corresponding weight matrices to be learned, $* \in \{i, f, o, c, y\}$, and $\widetilde{C}_t$ is the temporary state of the input at time t.

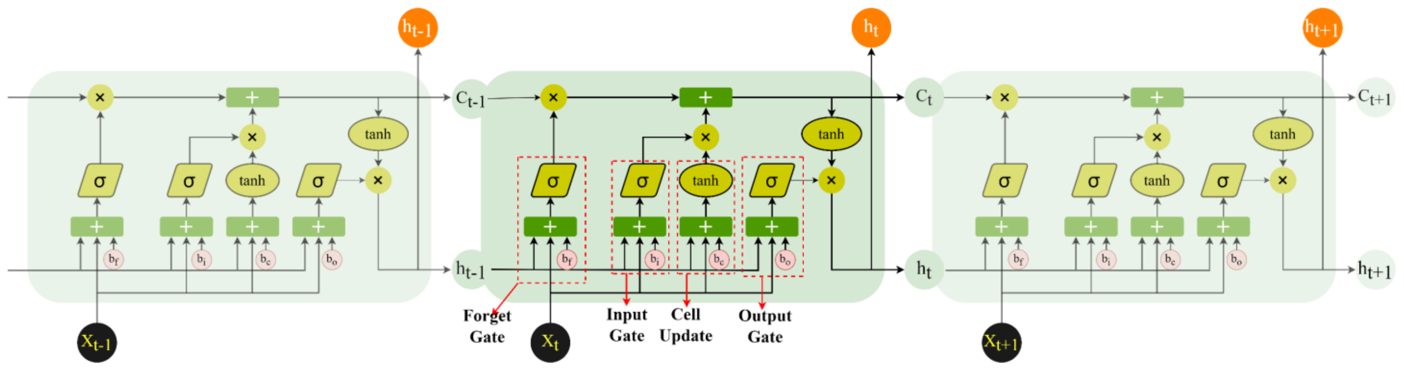


Fig. 2 The neural network structure based on the LSTM model

Bidirectional LSTM (BiLSTM) networks extend LSTMs by processing input sequences in both forward and backward directions, as shown in Fig. 3. This dual processing allows BiLSTMs to capture contextual information from both past and future time steps, making them particularly suitable for tasks where understanding the entire sequence is important, such as natural language processing (NLP) and video analysis. While BiLSTMs increase computational complexity and memory requirements, their ability to leverage bidirectional context often results in improved performance compared to unidirectional LSTMs. The output of the BiLSTM cell h_t at the time t can be calculated as follows:

$$\overrightarrow{h_t} = LSTM_{forward}(x_t, h_{t+1}) \tag{8}$$

$$\overleftarrow{h_t} = LSTM_{backward}(x_t, h_{t+1}) \tag{9}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{10}$$

$$y_t = \sigma(W_y h_t + b_y) \tag{11}$$

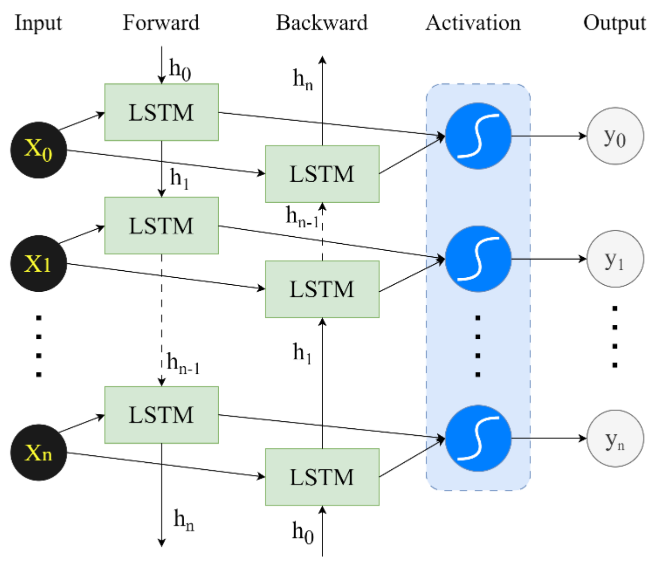


Fig. 3 BiLSTM architecture

The Gated Recurrent Unit (GRU) is another variant of RNNs, designed as a simpler alternative to LSTMs. GRUs merge the forget and input gates into a single update gate, and the cell state is directly combined with the hidden state, reducing the number of parameters, as shown in Fig. 4. Despite its simpler structure, GRUs often achieve comparable performance to LSTMs while being computationally more efficient. GRUs are particularly advantageous in scenarios with limited computational resources or datasets, where model simplicity is critical.

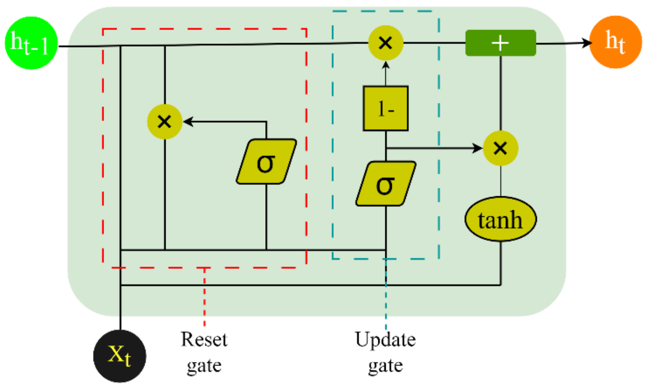


Fig. 4 GRU architecture

2.3. Convolutional neural networks

The primary goal of a Convolutional Neural Network (CNN) is to extract salient features from input data. A typical CNN architecture includes convolutional, pooling, dropout, and fully connected layers. Convolutional and pooling layers act as preprocessing layers, filtering input data and extracting valuable information for subsequent fully connected layers. Convolutional layers perform convolution operations between input data and kernels to generate feature values. These layers require structured matrix inputs, as they were initially designed for image datasets [27]. The convolution kernel, a small matrix of coefficients, slides across the input matrix, computing feature values for each subregion. Multiple kernels produce various feature maps, capturing more meaningful information than raw inputs, enhancing the model's learning and generalization capabilities. Convolutional layers are followed by nonlinear activation functions (e.g., ReLU) and pooling layers. Pooling reduces feature map dimensions, lowering computational complexity and preventing overfitting. Common methods include max pooling (selecting the maximum value) and average pooling (calculating the average value). Pooling preserves key features by summarizing local information, ensuring translation invariance. Although pooling layers lack learnable parameters, they simplify data representation and enable deeper networks. For mooring tension prediction, CNNs extract features from one-dimensional data using PyTorch's "Conv1d" and "MaxPool1d" functions. This process effectively captures patterns in time series data, supporting accurate tension prediction.

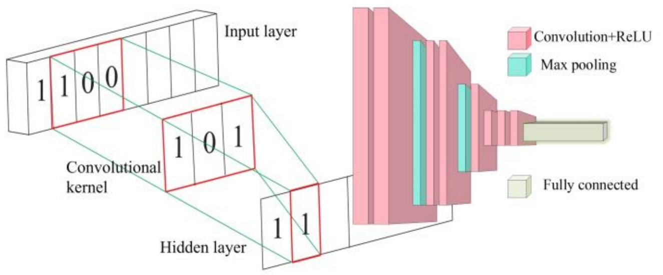


Fig. 5 Convolution layers and CNN architecture

2.4. Transformer and attention mechanism

The Attention Mechanism is a general computational framework that dynamically assigns weights to different parts of an input sequence based on their relevance to a given query. By computing a weighted sum of values, where the weights are determined by the compatibility between the query and a set of keys, it enables models to focus on the most relevant information. Self-Attention, a specialized form of the Attention Mechanism, derives queries, keys, and values from the same input sequence, allowing it to capture dependencies within the sequence by computing relationships between all positions. Building on this, the Transformer, a neural network architecture introduced by Vaswani et al. [28], replaces traditional recurrent or convolutional layers entirely with self-attention mechanisms. This design enables the Transformer to process sequences in parallel, efficiently capturing both local and global dependencies, and has revolutionized tasks in natural language processing and beyond. It is worth noting that the computational complexity of the self-attention mechanism in the original Transformer model is $O(T^2)$, where T represents the sequence length. However, for predicting mooring line tension, focusing on motion data from the distant past is often irrelevant to the current tension. To address this, we optimized the model by introducing a local attention mechanism, reducing the computational complexity to $O(nT)$, where n denotes the local range size of the attention mechanism. It is worth noting that this simplification does not come at the cost of accuracy. In other words, by applying local attention mechanisms to focus on platform motion near the current time step, the model can effectively predict tension while achieving significantly better computational efficiency—a critical advantage for edge-side deployment.

Specifically, the Transformer model with local attention used in this study has a parameter count comparable to traditional LSTMs or GRUs. Depending on operational conditions, the model size ranges between 100,000 and 1,000,000 parameters, with the original PyTorch model file typically occupying around 0.4 MB to 4 MB.

Furthermore, by applying model compression techniques such as quantization, pruning, and knowledge distillation, the model can be further reduced in size. Even low-end edge AI chips with as little as 0.5 TOPS (Tera Operations Per Second) can easily handle the deployment of such optimized models.

This design ensures high performance in real-world edge applications without sacrificing computational feasibility.

2.5. Deep operator network

DeepONet (Deep Operator Network) is a deep neural network architecture designed for learning nonlinear operators, as introduced in [29]. The DeepONet model facilitates mappings between two infinite-dimensional spaces by training on finite input-output pairs. At its core, DeepONet approximates nonlinear operators using neural networks, enabling efficient handling of high-dimensional input and output spaces. The architecture comprises two primary subnetworks: the Branch Net and the Trunk Net. The Branch Net processes information from input functions, which are typically represented in discrete form (e.g., function values at specific points), and encodes this information into a feature vector. The Trunk Net, on the other hand, processes the location information of output points, mapping their coordinates into another feature vector. Together, these subnetworks enable DeepONet to effectively model complex nonlinear relationships in high-dimensional spaces, as shown in Fig. 6. Ultimately, the output of DeepONet is the dot product of the outputs of branch networks and trunk networks:

$$\mathrm{G}(u)(y) = \sum_{i=1}^{p} b_i(u) \cdot t_i(y) \tag{12}$$

$G(u)$ is the output function corresponding to the input function u, where y represents the position of the output point, $b_i(u)$ is the output of the branch network, $t_i(y)$ is the output of the trunk network, and p is the dimension of the feature vector.

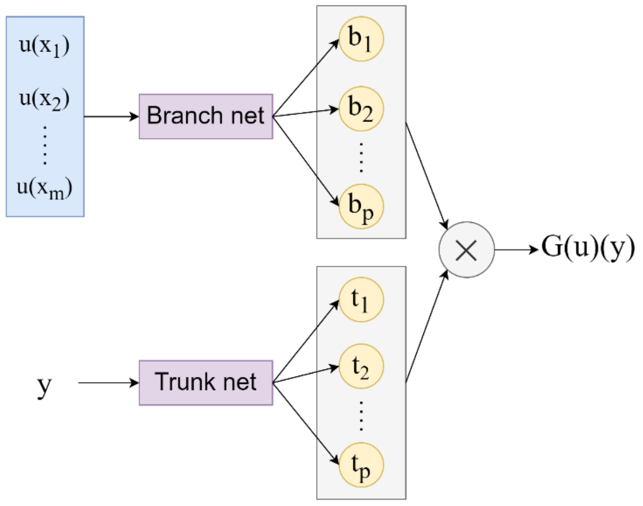


Fig. 6 Architecture of Deep operator network

The functional of mooring line tension can be expressed as a nonlinear operator mapping of fairlead position and velocity:

$$T_{moor}(t) = G_T\big(t; \dot{x}(t), x(t)\big) \tag{13}$$

3. Implemetation process

3.1. Neural network method technical route

Firstly, simulation data was obtained from OpenFAST and OrcaFlex. Secondly, the data underwent pre-processing steps, including normalization, data splitting, and sliding window processing, to prepare it for neural network training. Thirdly, the hyperparameters of the neural network were optimized to enhance model performance. Subsequently, the neural network model was trained and saved. Finally, the trained model was utilized to predict the mooring tension of the FOWT, as shown in Fig. 7.

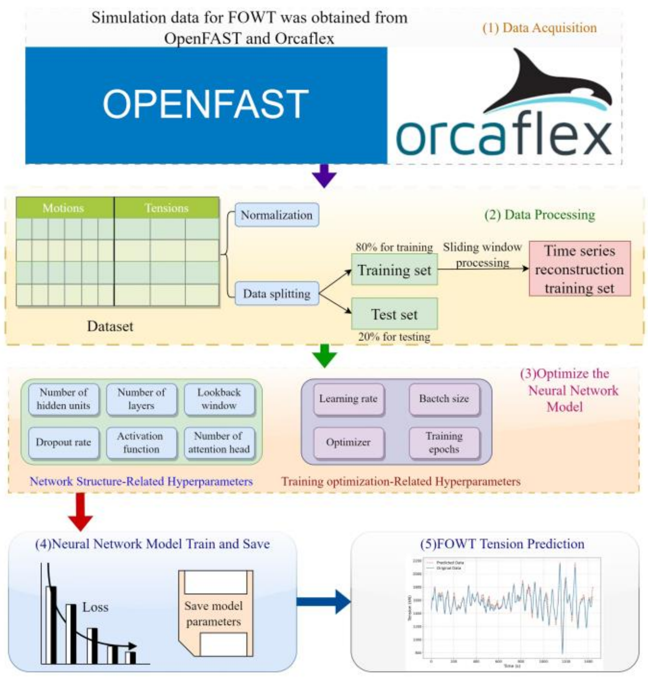


Fig. 7 Neural network method technical route

3.2. Deep operator network method technical route

For the DeepONet model, the first step involves transforming the six degrees of freedom (6-DOF) motion of the platform into the position of the cable hole using the Euler rotation matrix. The Euler rotation matrix R is defined as follows:

$$R = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \tag{14}$$

where ψ, θ, and ϕ represent the yaw, pitch, and roll angles, respectively. Subsequently, the corresponding velocity is calculated through differential computation using the finite difference method:

$$v(t) = \frac{p(t + \Delta t) - p(t)}{\Delta t} \tag{15}$$

where $p(t)$ denotes the position at time t, and Δt is the time step.

In addition, the DeepONet model can be enhanced by integrating an attention mechanism. In this approach, the position and velocity data are processed by two separate branch networks, respectively. The position branch network $\mathcal{N}_p$ and the velocity branch network $\mathcal{N}_v$ are designed as fully connected neural networks. The outputs of these two branches are then combined using a self-attention mechanism $\mathcal{A}$, which computes the weighted sum of the features based on their importance. We refer to this method as $MIONet - ATT$, a multi-input DeepONet framework that incorporates an attention mechanism. The overall architecture can be expressed as:

$$y = \mathcal{A}\left(\mathcal{N}_p(p), \mathcal{N}_v(v)\right) \tag{16}$$

where p and v denote the position and velocity inputs, respectively. The key distinction between MIONet-ATT and DeepONet lies in their feature weighting strategies. MIONet-ATT employs an attention mechanism to dynamically and intelligently weight velocity and displacement vectors, enabling more precise characterization of their varying influences on tension prediction across different temporal stages. This adaptive weighting approach captures the temporal dependencies leading to superior output accuracy in modeling physical systems.

Attention mechanisms have demonstrated transformative potential across multiple AI domains through their ability to learn context-aware feature relevance. In computer vision, the evolution from YOLOv11 [30] to YOLOv12 [31] exemplifies this paradigm shift, where the traditional CNN backbone was replaced by a Transformer architecture enhanced with region attention modules, yielding significant improvements in object detection performance. The natural language processing domain has witnessed similar advancements, with models like Kimi-K2 achieving state-of-the-art results through optimized attention head configurations and innovative sparsity patterns.

The widespread adoption of attention mechanisms stems from their unique advantages, including dynamic feature relevance scoring, interpretable weight distributions, and flexible integration with various neural architectures. These characteristics make attention mechanisms particularly valuable for scientific machine learning tasks that require modeling complex physical relationships. In this work, we extend their utility by integrating attention mechanisms with DeepONet's operator learning framework, creating a hybrid approach that combines the strengths of both methodologies for enhanced physical system modeling. The resulting architecture not only improves prediction accuracy but also provides insights into the relative importance of different physical variables across temporal and spatial domains.

3.3. Model evaluation criteria

Models' performance was evaluated using three common metrics: mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE). These metrics quantify the difference between predicted and actual values, with lower values indicating better forecasting accuracy. Additionally, the coefficient of determination (R^2) was considered, with values closer to 1 indicating higher prediction quality. The detailed mathematical expressions for the above evaluation indicators are shown below [24].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i-1}^{n}(y_i - \bar{y}_i)^2} \tag{17}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{18}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{19}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{20}$$

where the y_i and $\hat{y}_i$ symbolize the measured and predicted values, respectively, and $\bar{y}$ represents the mean of all the true values.

Table 1
Environmental conditions and load cases

No.	V_w (m/s)	H_s (m)	T_p (s)	V_c (m/s)	θ_{wave} (°)
Case 1	11.4	7.5	14.8	0	0
Case 2	25.0	13.1	14.9	1.20	30
Case 3	25.0	12.6	13.7	1.27	0

In our experimental setup, we employed a systematic grid search approach to optimize the hyperparameters across all models. Given the diverse range of architectures involved in this study and potential variations in computational environments, we provide both a concise description of our hyperparameter optimization strategy and a comprehensive reference table detailing the parameter ranges (Table 2).

The model architectures in this work typically contain between 100,000 to 500,000 trainable parameters. To illustrate, our baseline LSTM implementation consists of three stacked LSTM layers with 192 units each, combined with fully-connected input and output layers, resulting in approximately 450,000 parameters. This parameter scale has proven sufficient for capturing the complex temporal dependencies in our regression tasks while maintaining computational efficiency.

For training configuration, we established 100 epochs as the upper limit while implementing an early stopping criterion that terminates training when the validation loss fails to improve for 10 consecutive epochs. In practice, most models converged within 40-60 epochs, demonstrating stable learning dynamics. This balanced approach ensures thorough optimization without unnecessary computational overhead.

The complete hyperparameter search space, including learning rates, batch sizes, and regularization coefficients, is systematically documented in the supplementary materials to facilitate reproducibility and future benchmarking efforts.

Table 2
Hyperparameters and model configurations

Hyperparameters	Reference range
Number of layers	2-4 layers
Number of units (hidden_size)	96-256 units
Initial learning rate	4×10^{-3}
Batch size	32
Activation function	ReLU
Optimizer	Adam (β_1=0.9, β_2=0.999)

4. Case study

This paper refers to the extreme sea conditions in the South China Sea and sets four cases to explore the predictive performance of different models. The wind conditions are characterized by turbulence, adhering to a Kaimal wind spectrum and blowing in a positive x-axis direction. The waves follow a Jonswap spectrum, while the flow is maintained as constant. Specific parameters are listed in Table. 1, where the abbreviation "*w*" stands for "wind", and "*c*" stands for "current".

4.1. Performance evaluation of neural/operator network models

We will divide the models into three groups: Group A (GRU, LTSM, BiLSTM), Group B (LSTM, CNN-LSTM-ATT, LA-Transformer (Local Attention Transformer)), and Group C (LSTM, DeepONet, MIONet-ATT), using the LSTM model as the benchmark model to evaluate these models' performance.

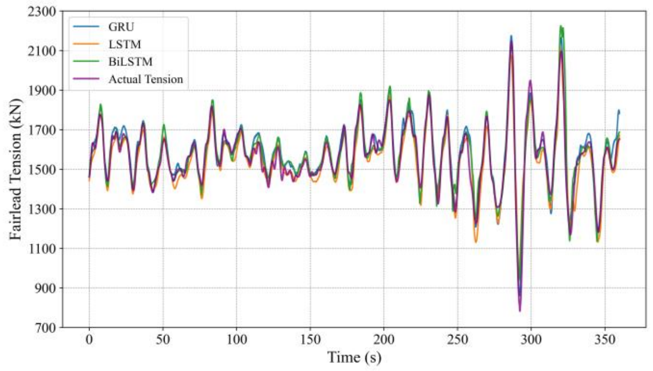


Fig. 8 Comparison of the performance metrics of different models (Case 1, Group 1)

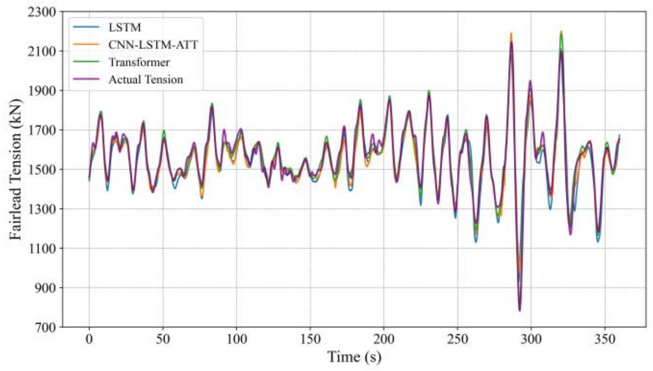


Fig. 9 Comparison of the performance metrics of different models (Case 1, Group 2)

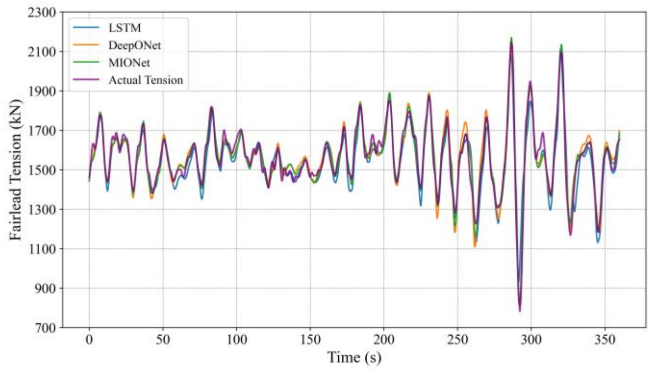


Fig. 10 Comparison of the performance metrics of different models (Case 1, Group 3)

Table 3
Comparison of Performance Metrics for Different Models: Case 1

Model	RMSE (kN)	MAE (kN)	MAPE	R^2
GRU	41.80	30.62	2.02%	0.953
LSTM	52.60	37.90	2.51%	0.913
BiLSTM	52.14	34.52	2.38%	0.919
CNN-LSTM-ATT	42.25	31.24	2.07%	0.951
LA-Transformer	28.52	21.71	1.42%	0.969
DeepONet	37.75	29.27	1.93%	0.945
MIONet-ATT	31.65	23.39	1.53%	0.961

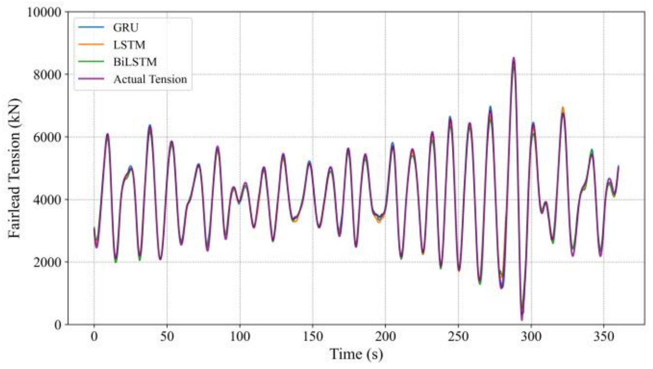


Fig. 11 Comparison of the performance metrics of different models (Case 2, Group 1)

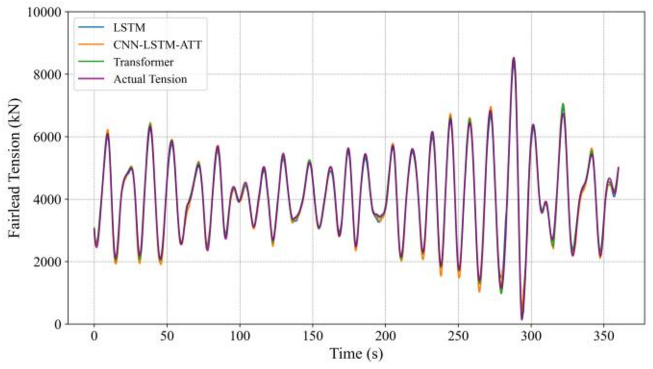


Fig. 12 Comparison of the performance metrics of different models (Case 2, Group 2)

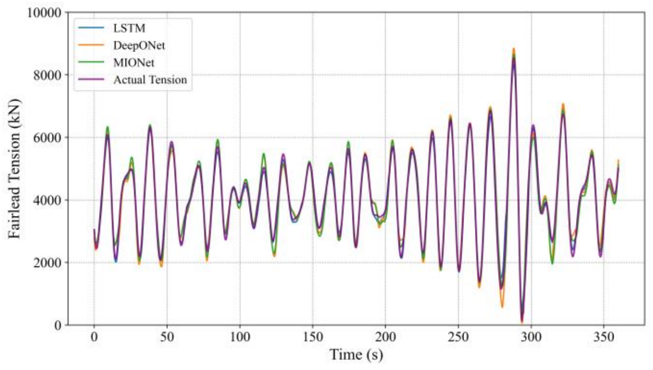


Fig. 13 Comparison of the performance metrics of different models (Case 2, Group 3)

Table 4
Comparison of Performance Metrics for Different Models: Case 2

Model	RMSE (kN)	MAE (kN)	MAPE	R^2
GRU	95.39	70.22	2.30%	0.995
LSTM	114.87	91.32	3.18%	0.992
BiLSTM	117.70	88.36	3.08%	0.992
CNN-LSTM-ATT	128.87	97.62	3.40%	0.990
LA-Transformer	80.29	59.56	1.83%	0.996
DeepONet	231.23	177.96	5.67%	0.968
MIONet-ATT	213.34	163.90	5.02%	0.972

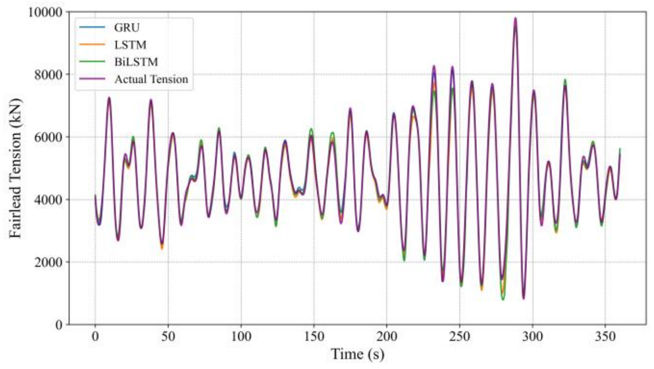


Fig. 14 Comparison of the performance metrics of different models (Case 3, Group 1)

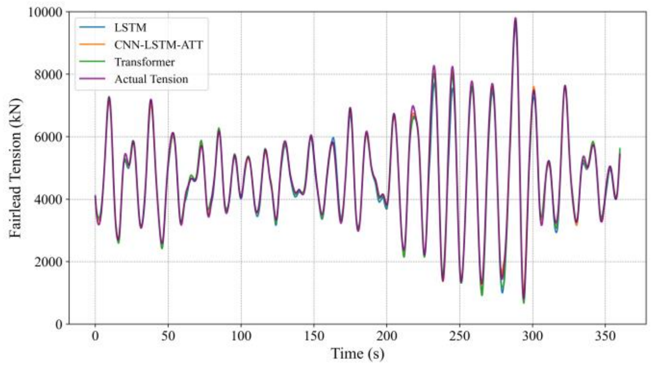


Fig. 15 Comparison of the performance metrics of different models(Case 3, Group 2)

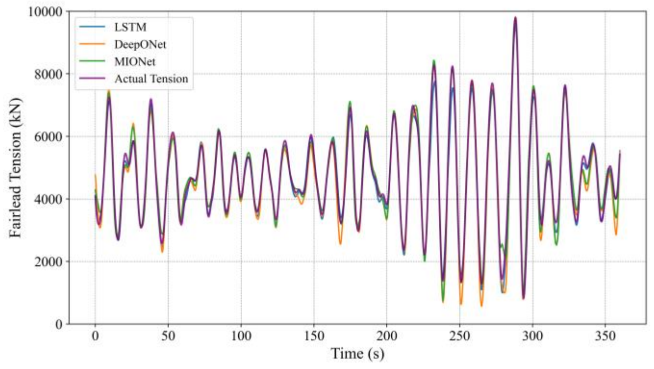


Fig. 16 Comparison of the performance metrics of different models(Case 3, Group 3)

Table 5
Comparison of Performance Metrics for Different Models: Case 3

Model	RMSE (kN)	MAE (kN)	MAPE	R^2
GRU	123.76	96.09	2.39%	0.993
LSTM	180.31	126.75	2.92%	0.985
BiLSTM	189.94	132.39	3.05%	0.983
CNN-LSTM-ATT	92.67	70.47	1.75%	0.996
LA-Transformer	128.94	88.89	2.33%	0.992
DeepONet	312.01	229.54	6.18%	0.955
MIONet-ATT	264.67	198.74	5.21%	0.968

In order to ensure a fair comparison between different neural network architectures, we adjust the number of layers and the number of neurons in each layer of the models. This adjustment aims to keep the parameter counts of the models on the same order of magnitude. For example, the number of layers in the LSTM model is set to be twice that of the BiLSTM model. Neural networks demonstrate strong predictive performance, with the Transformer model achieving the highest accuracy. This validates our approach to predicting mooring line tension based on short-term dependency time series data. However, the neural operator exhibits slightly reduced accuracy under once-in-a-century sea conditions, particularly at peak and trough regions, as shown in Fig. 17. These regions coincide with significant velocity changes, where errors in velocity—derived from position differences—are more pronounced. Additionally, the input velocity data itself contains inherent uncertainties, which may contribute to these inaccuracies.

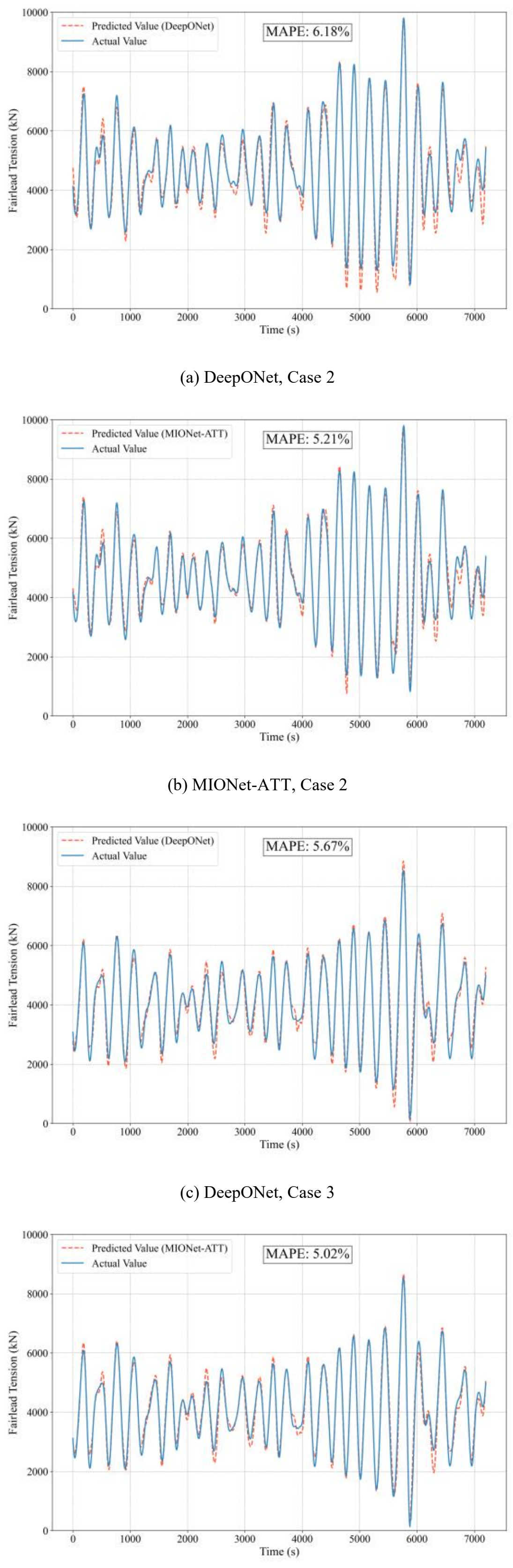


Fig. 17 Performance evaluation of DeepONet and MIONet-ATT in Case 2 and Case 3

4.2. Generalization performance comparison of operator networks and neural networks

The concept of generalization refers to a model's ability to perform well on unseen data, a critical metric for real-world applicability. In this study, we compare the generalization performance of operator networks (e.g., DeepONet, MIONet) with traditional neural networks (e.g., LSTM, Transformer). Operator networks learn mappings between function spaces, making them effective for modeling physical fields [32, 33], while traditional networks excel at capturing discrete temporal dependencies.

To empirically evaluate generalization, we design Case 4—a loading condition intermediate between Cases 2 and 3 (Table 6.). Models are trained on 50% of time-series segments from Cases 2-3 and 20% from Case 4, then tested on unseen segments of Case 4. This validates their ability to extrapolate under novel sea conditions.

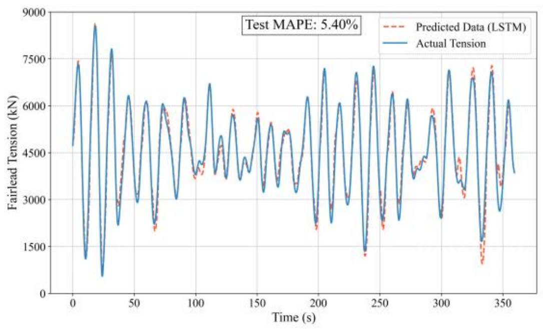
The fundamental distinction between generalization testing and accuracy testing lies in the relationship between the training and testing datasets. In accuracy testing, where both training and testing data are drawn from different temporal segments of the same sea state, we evaluate a model's ability to make predictions within a specific, known environmental condition. In contrast, generalization testing employs training and testing sets from different sea states altogether, assessing whether a model trained on one set of conditions can be effectively applied to similar or even substantially different environmental scenarios. This capability for cross-condition prediction is particularly valuable for practical deployment, as it enables the use of fewer models across varying operational conditions, thereby conserving computational resources and simplifying implementation.

Notably, inputs differ between architectures: platform 6-DOF motions for neural networks versus fairlead position/velocity for neural operators.

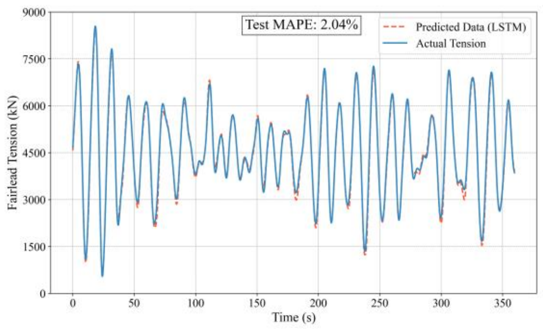
Table 6
Environmental condition and load case for generalization validation

No.	V_w (m/s)	H_s (m)	T_p (s)	V_c (m/s)	θ_{wave} (deg)
Case 4	25.0	12.8	14.3	1.24	0

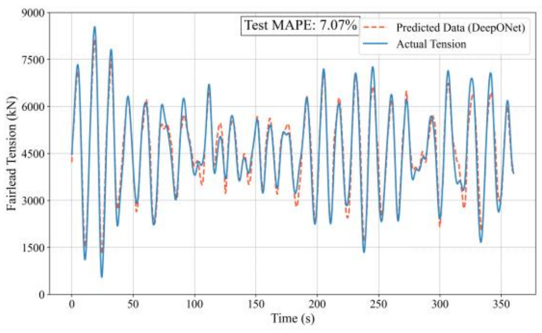
The generalization test results are as follows:



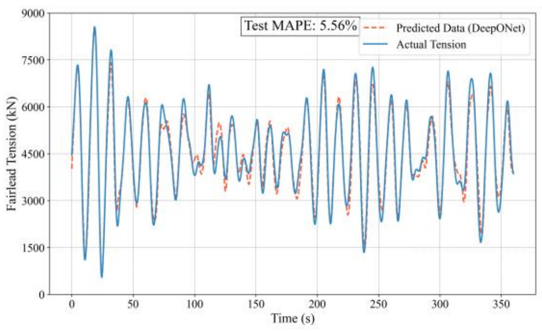
(a) LSTM



(b) LA-Transformer



(c) DeepONet



(d) MIONet-ATT

Fig. 18 Generalization validation of neural network models and neural operator models

Table 7
Comparison of generalization performance for different models: Case 4

Model	RMSE (kN)	MAE (kN)	MAPE	R^2
LSTM	265.88	193.64	5.40%	0.965
LA-Transformer	104.67	76.26	2.04%	0.995
DeepONet	328.45	257.55	7.07%	0.946
MIONet-ATT	279.98	199.38	5.56%	0.962

From the generalization test results, the LA-Transformer demonstrates superior performance, while the LSTM closely follows the MIONet-ATT, and the DeepONet lags slightly behind. Although the accuracy of neural operators is not outstanding, they still hold promising application prospects. As previously mentioned, in neural operators, the velocity vector is derived by differentiating the position vectors. During the reciprocating motion of floating platforms, the velocity undergoes significant fluctuations, which can adversely affect the accuracy of neural operators. Under other sea conditions, however, the accuracy is expected to improve. Additionally, the data sampling interval in this study is 0.05 seconds, simulating a 20Hz GPS or sensor signal. If the sampling frequency is increased, the differential velocity obtained would be more accurate. Finally, the MIONet-ATT model exhibits a notable improvement in accuracy compared to the DeepONet model, and there remains ample room for further exploration of optimization methods for neural operators. At the end of this section, we provide a proof that extreme sea conditions amplify the approximation error of first-order differences. The truncation error of first-order difference is approximately $O(\Delta t \cdot a)$, where a is the acceleration and Δt is the time step. The truncation error of $O(\Delta t \cdot a)$ in first-order difference schemes originates from Taylor series expansion analysis. The formal derivation proceeds as follows:

a. Taylor Expansion of Displacement

Consider the displacement function $s(t)$. Its Taylor expansion around time t is:

$$s(t + \Delta t) = s(t) + v(t)\Delta t + \frac{1}{2}(\Delta t)^2 + O((\Delta t)^3) \tag{21}$$

where, $v(t)$ is the instantaneous velocity, $a(t)$ is the instantaneous acceleration, $O((\Delta t)^3)$ is the higher-order infinitesimals.

b. First-Order Difference Definition

The first-order difference approximation of velocity is:

$$v_{diff}(t) \approx \frac{s(t+\Delta t) - s(t)}{\Delta t} \quad (22)$$

c. Substitution of Taylor Expansion

$$v_{diff}(t) = v(t) + \frac{a(t)}{2}\Delta t + O((\Delta t)^2) \quad (23)$$

d. Truncation Error Expression

The error between true velocity *v(t)* and approximated velocity $v_{diff}(t)$ is

$$Error = v_{diff}(t) - v(t) = \frac{a(t)}{2}\Delta t + O((\Delta t)^2) \quad (24)$$

Neglecting higher-order terms $O((\Delta t)^2)$, the dominant error term is:

$$Error \approx \frac{a(t)}{2}\Delta t \quad (25)$$

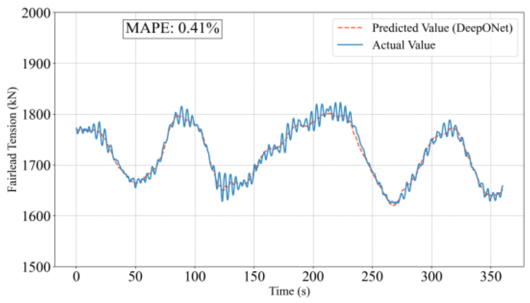
The truncation error $O(\Delta t \cdot a)$ of the first-order difference is essentially a linear approximation defect of Taylor expansion, which is significantly enlarged when the acceleration is large or the time step is coarse. The acceleration surge under extreme sea conditions is the fundamental reason for the aggravation of the error.

We have added three non extreme sea conditions, in which the operator network has higher accuracy. The sea conditions are as follows:

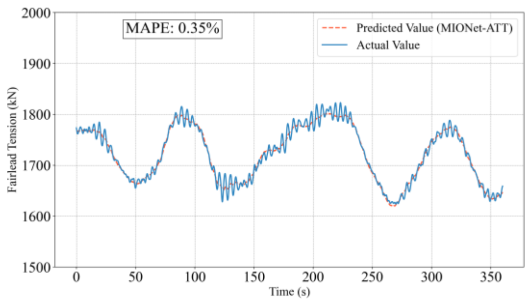
Table 8
Environmental conditions in moderate sea states

No.	V_w (m/s)	H_s (m)	T_p (s)	V_c (m/s)	θ_{wave} (°)
Case 5	11.4	2	3.5	0.3	0
Case 6	11.4	5	4.5	0	0
Case 7	11.4	6	4.7	0.1	0

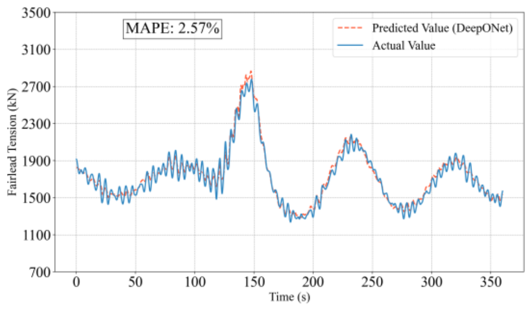
The operator networks demonstrates superior prediction accuracy in moderate sea states, as shown in Fig. 19.



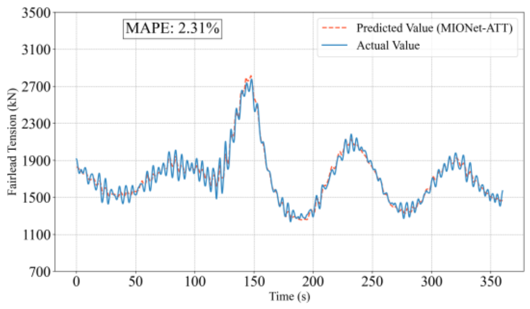
(a) DeepONet, Case 5



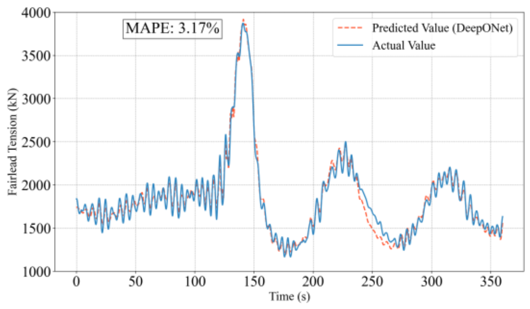
(b) MIONet-ATT, Case 5



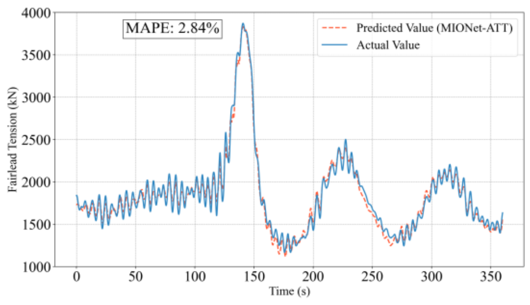
(c) DeepONet, Case 6



(d) MIONet-ATT, Case 6



(e) DeepONet, Case 7



(f) MIONet-ATT, Case 7

Fig. 19 Prediction accuracy/performance of operator networks in moderate sea states

5. Conclusions

In this study, we developed several neural networks and operator networks to predict the dynamic mooring tension behavior of a semi-submersible floating wind turbine. The training and testing datasets are obtained from coupled numerical simulations in Orcaflex and OpenFAST. The neural network method utilizes the six-degree-of-freedom (6-DoF) time series of the platform as input and selects a neural network model suitable for short-term dependency time series prediction to forecast the mooring line tension of floating wind turbines. On the other hand, the operator network method first transforms the 6-DoF motion of the platform into the position of the fairlead through Euler rotation. The velocity is then obtained by differentiating the position, and the mooring tension is assumed to depend solely on the position and velocity of the fairlead, without considering changes in hydrodynamic forces. Consequently, time, position, and velocity operators are employed to represent variations in mooring cable tension. The main conclusions of this study are as follows:

- In neural network models, the selected architectures demonstrate strong performance in predicting mooring line tension, with the Transformer model under a local attention mechanism (LA-Transformer) achieving the highest accuracy.
- In operator networks, the DeepONet combined with an attention mechanism (MIONet-ATT) shows significant improvement in prediction accuracy compared to the traditional DeepONet, along with better generalization capabilities.
- The accuracy of the operator network is closely related to the precision of the velocity in the input data. The drastic velocity changes during extreme sea conditions pose a challenge to the accuracy of operator networks. Nevertheless, the MIONet-ATT model still achieves reasonably high accuracy, indicating that operator networks hold great potential in the field of mooring line tension prediction.
- In summary, while relatively mature neural network models undoubtedly excel in purely data-driven applications, emerging operator networks are beginning to show promise, particularly due to their potential for integrating physical principles, which is highly encouraging.

Acknowledgement

The present work is Supported by the Open Fund of State Key Laboratory of Coastal and Offshore Engineering with grant No. LP2521, Guangdong Basic and Applied Basic Research Foundation (No. 2022B1515250002), and the Hong Kong SAR Government on the project Second-order direct analysis for the design of steel members with irregular cross-sections (PolyU 15203121/22E).

References

[1] Hall M, Goupee A. Validation of a lumped-mass mooring line model with DeepCwind semisubmersible model test data. Ocean Engineering. 2015;104:590-603.
[2] Chen L, Basu B, Nielsen SR. A coupled finite difference mooring dynamics model for floating offshore wind turbine analysis. Ocean Engineering. 2018;162:304-15.
[3] Zhang Y, Shi W, Li D, Li X, Duan Y. Development of a numerical mooring line model for a floating wind turbine based on the vector form intrinsic finite element method. Ocean Engineering. 2022;253:111354.
[4] Sidarta DE, Kyoung J, O'Sullivan J, Lambrakos KF. Prediction of offshore platform mooring line tensions using artificial neural network. International Conference on Offshore Mechanics and Arctic Engineering: American Society of Mechanical Engineers; 2017. p. V001T01A79.
[5] Qiao D, Li P, Ma G, Qi X, Yan J, Ning D, et al. Realtime prediction of dynamic mooring lines responses with LSTM neural network model. Ocean Engineering. 2021;219:108368.
[6] Wang Z, Qiao D, Yan J, Tang G, Li B, Ning D. A new approach to predict dynamic mooring tension using LSTM neural network based on responses of floating structure. Ocean engineering. 2022;249:110905.
[7] Zheng Z, Hu Z, Xie X, Huang W. Local scour around the monopile: A microscopic perspective using CFD-DEM. Ocean Engineering. 2024;299:117318.
[8] Ma H, Zhang S, Li B, Huang W. Local scour around the monopile based on the CFD-DEM method: Experimental and numerical study. Computers and Geotechnics. 2024;168:106117.
[9] Li B, Huang W, Chen X. Tendon fatigue analysis of tension leg platform using a new time-domain quasi-dynamic method. Marine Structures. 2025;99:103701.
[10] Huang W, Li B, Chen X. A new quasi-dynamic method for the prediction of tendon tension of TLP platform. Ocean Engineering. 2023;270:113590.
[11] Ning J-H, Liu S-W, Wan J-H, Huang W. Line-element formulation for upheaval buckling analysis of buried subsea pipelines due to thermal expansion. Adv Steel Constr. 2021;17:210-20.
[12] Ning J, Lin T, Liu S, Bai R, Huang W. Three-dimensional pipeline element formulation for global buckling analysis of submarine pipelines with sleeper. Marine Structures. 2023;90:103428.
[13] Wang C, Liu J, Li B, Huang W. The absolute nodal coordinate formulation in the analysis of offshore floating operations, Part II: Code validation and case study. Ocean Engineering. 2023;281:114650.
[14] Li B, Wang C. The absolute nodal coordinate formulation in the analysis of offshore floating operations Part I: Theory and modeling. Ocean Engineering. 2023;281:114645.
[15] Huang W, Tang X. Prediction of the dynamic response on the spread catenary mooring system for FPSO by long short-term memory method. Ocean Engineering. 2025;340:122113.
[16] Tang X, Huang W, Li X, Ma G. Prediction of mooring tension of floating offshore wind turbines by CNN-LSTM-ATT and Chebyshev polynomials. Ocean Engineering. 2025;331:121327.
[17] Payenda MA, Wang S, Jiang Z, Prinz A. Prediction of mooring dynamics for a semi-submersible floating wind turbine with recurrent neural network models. Ocean Engineering. 2024;313:119490.
[18] Das S, Saha P. A review of some advanced sensors used for health diagnosis of civil engineering structures. Measurement. 2018;129:68-90.
[19] Wan A, Chang Q, Khalil A-B, He J. Short-term power load forecasting for combined heat and power using CNN-LSTM enhanced by attention mechanism. Energy. 2023;282:128274.
[20] Li Y, Xiao L, Wei H, Kou Y, Yang L, Li D. A time–frequency physics-informed model for real-time motion prediction of semi-submersibles. Ocean Engineering. 2024;299:117379.
[21] Cao Q, Goswami S, Tripura T, Chakraborty S, Karniadakis GE. Deep neural operators can predict the real-time response of floating offshore structures under irregular waves. Computers & Structures. 2024;291:107228.
[22] Zhao J, Zhao Y, Zou L. A fully adaptive time–frequency coupling model using self-attention mechanism based on deep operator network for very short-term forecasting of ship motion. Physics of Fluids. 2024;36.
[23] Zhang Q, Zhang H, Zhao X, Ding J, Xu D. Multiple-input operator network prediction method for nonlinear wave energy converter. Ocean Engineering. 2025;317:120106.
[24] Thomsen JB, Bergua R, Jonkman J, Robertson A, Mendoza N, Brown C, et al. Modeling the TetraSpar floating offshore wind turbine foundation as a flexible structure in OrcaFlex and OpenFAST. Energies. 2021;14:7866.
[25] Salehinejad H, Sankar S, Barfett J, Colak E, Valaee S. Recent advances in recurrent neural networks. arXiv preprint arXiv:180101078. 2017.
[26] Graves A. Long short-term memory. Supervised sequence labelling with recurrent neural networks. 2012:37-45.
[27] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012;25.
[28] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems. 2017;30.
[29] Lu L, Jin P, Karniadakis GE. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:191003193. 2019.
[30] Khanam R, YOLOv11 MH. an overview of the key architectural enhancements. Published online October 23, 2024. 2024. doi: 10.48550. arXiv preprint arXiv241017725.
[31] Tian Y, Ye Q, Doermann D. Yolov12: Attention-centric real-time object detectors. arXiv preprint arXiv:250212524. 2025.
[32] Chen Y, Yuan L, Qin L, Zhang N, Li L, Wu K, et al. A forecasting model with hybrid bidirectional long short-term memory for mooring line responses of semi-submersible offshore platforms. Applied Ocean Research. 2024;150:104145.
[33] Lee S, Shin Y. On the training and generalization of deep operator networks. SIAM Journal on Scientific Computing. 2024;46:C273-C96.