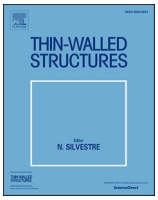
Thin–Walled Structures 199 (2024) 111797

Contents lists available at ScienceDirect

Thin-Walled Structures

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



Full length article

Parallel neural network feature extraction method for predicting buckling load of composite stiffened panels

Tao Zhang [a], Peiyan Wang [a,*], Jianwei Fu [b], Suian Wang [a], Chenchen Lian [a]

[a] School of Mechanics, Civil Engineering and Architecture, Northwestern Polytechnical University, Xi'an, China
[b] AVIC Shenyang Aircraft Design & Research Institute, Shenyang, China

ARTICLE INFO

Keywords:
Composite stiffened panel
Parallel neural network
Compression buckling
Data-driven approach

ABSTRACT

A novel Parallel Neural Network (PNN) feature extraction method is proposed in this paper to predict the buckling load of composite stiffened panels. The PNN effectively processes both stacking sequences and discrete variables by leveraging the parallel operation of the Recurrent Neural Network (RNN) and the Feedforward Neural Network (FNN). This approach addresses limitations of previous models, such as feature loss due to the Classical Laminate Theory (CLT) and struggled with variable length stacking sequences. A Self-attention-based Bidirectional Long Short-Term Memory network (T-Bi-LSTM) is introduced to handle variable length stacking sequences comprehensively. The T-Bi-LSTM, which incorporates self-attention and other mechanisms, improves the network's ability to capture crucial information. The dataset for training and testing is generated using a finite element model verified by the corresponding experiments, where the PNN with T-Bi-LSTM and other contrasting models are trained. The results suggest that the T-Bi-LSTM demonstrates better capability in extracting comprehensive stacking sequences than Bidirectional Long Short-Term Memory network (Bi-LSTM). Furthermore, the proposed PNN feature extraction method exhibits superior fitting ability and generalization performance than the feature extraction method based on CLT.

1. Introduction

Composite stiffened panels, renowned for their exceptional stiffness and strength characteristics, play a crucial role in enhancing buckling resistance while adhering to stringent quality standards [1]. They serve as vital structural components in maritime vessels, aircraft, and aerospace vehicles. Being thin-walled structures, the design of stiffened panels is influenced by both stability and strength considerations. Throughout their operational lifespan, these panels often endure compressive loads, rendering them susceptible to buckling failure [2]. Hence, it is imperative to investigate the buckling load of composite stiffened panels to ensure their structural integrity. However, designing stiffened panels is a labor-intensive and time-consuming process due to the inherent heterogeneity and anisotropy of composite materials. Additionally, the challenges associated with characterizing these materials, accounting for stacking sequences, and managing numerous discrete features in the design process further compound the complexity. Thus, there is a pressing need for an efficient method that strikes a balance between accuracy and computational efficiency to predict the buckling load of stiffened panels and expedite the design process [3].

Historically, designers of stiffened panels resorted to Engineering Calculations (EC) and Finite Element (FE) methods for preliminary buckling analysis. However, these methods have inherent limitations in balancing calculation efficiency and accuracy. Mo et al. proposed an engineering method to predict the initial buckling load of curved panels, proving accurate when the skin thickness remained constant, but exhibiting an error margin of up to 32 % when the skin thickness varied [4]. Similarly, Pevzner et al. extended the "effective width" method to estimate the failure load of composite longitudinal stiffened circular cylindrical plates, but unfortunately, the method generally exhibited inferior accuracy compared to the FE method in most cases [5]. Relatively speaking, the FE method offers higher calculation accuracy. However, the anisotropy of composite materials and complex damage mechanisms pose challenges in modeling, meshing, contact, and boundary setting. Additionally, the calculation time is prolonged due to these complexities, resulting in low calculation efficiency [6–8]. In conclusion, an urgent requirement exists for the development of a precise and efficient methodology to forecast the buckling behavior of composite stiffened panels. Consequently, researchers are focusing their efforts on the field of machine learning to address this important

* Corresponding author.
E-mail address: pywang@nwpu.edu.cn (P. Wang).

https://doi.org/10.1016/j.tws.2024.111797
Received 20 December 2023; Received in revised form 26 February 2024; Accepted 9 March 2024
Available online 11 March 2024

requirement.

In recent years, the application of machine learning to predict the mechanical properties of composite laminates has gained widespread attention due to its exceptional capacity for uncovering implicit relationships [9–11]. However, developing an effective machine learning model requires a substantial number of samples and the cost associated with obtaining these samples through experiments is prohibitively high. Therefore, utilizing the FE method to acquire data emerges as a practical alternative [12]. In previous research, two primary modeling methods for machine learning models considering laminate stacking sequences have been identified. The first method involves using the ply angle of each layer and other discrete features directly as the model's input [13,14]. The second method involves converting the stacking sequence into relevant parameters through CLT before utilizing these parameters and some discrete features as the model's input [15–18]. However, both methods have limitations. Firstly, the first model can only address laminates with a fixed length of stacking sequence due to its reliance on the ply angle of each layer as the input. Moreover, when dealing with long stacking sequences, it may encounter issues such as dimensional disaster and difficulties in model convergence [19]. Secondly, the second model, depending on basic laminate theory, fails to consider out-of-plane load and displacement based on the Kirchhoff hypothesis [20]. This oversight results in the method missing crucial features. Moreover, for specific structures like stiffened panels with varying rib types and loading methods, the model's usability is further complicated by the diverse characteristics that influence their mechanical properties.

In this study, the PNN feature extraction method is employed to address the aforementioned challenges. This method involves the simultaneous operation of two types of neural networks: the RNN and the FNN, which extracting the features from the stacking sequences and other discrete features respectively. As RNNs demonstrated proficiency in processing variable length and long sequence [21,22], the PNN feature extraction method presents a viable solution to the difficulties encountered in processing variable-length stacking sequences and achieving convergence with longer stacking sequences. To enhance the robustness of feature extraction from stacking sequences in this research, the T-Bi-LSTM is introduced on the basis of the Bi-LSTM. The enhanced structure incorporates elements such as self-attention mechanism, residual normalization, adaptive pooling, and other advanced techniques. Consequently, it possesses the capability to capture more comprehensive and intricate features from stacking sequences, thereby contributing to a more accurate analysis. Subsequently, the two kinds of extracted features are combined together, hence the buckling load of the composite stiffened panel can be obtained by processing these fused features using the FNN. The PNN feature extraction method leverages data-driven techniques to accurately predict the buckling load of composite stiffener panels, without relying on any preconceived assumptions. Consequently, the utilization of the PNN feature extraction method eliminates the potential feature loss, caused by feature extraction through CLT. In addition, the data-driven approach automatically extracts pertinent features from various datasets, which aids in the convergence of the network. This significantly reduces the complexities associated with data processing, ultimately making the PNN network more versatile and widely applicable.

2. Parallel neural network feature extraction method

The PNN feature extraction method is divided into three steps and is based on the utilization of two different types of neural network: the FNN and the RNN. To effectively understand the PNN feature extraction method, it is imperative to introduce the basic neural networks that form the backbone of the approach. These include the FNN, the widely used Long Short-Term Memory Neural Network (LSTM) belonging to the RNN family, as well as the self-attention mechanism. Subsequently, a novel RNN known as T-Bi-LSTM is presented, elucidating its structure, mechanism, and its capacity to comprehensively extract stacked sequences. Finally, this section presents the details of the whole process regarding the PNN feature extraction method, including data preparation, network construction, and model training and testing.

2.1. Basic neural network

2.1.1. FNN

Fig. 1 illustrates the architecture of a FNN, consisting of an input layer, hidden layers, and an output layer. Each layer contains a certain number of neurons. In the input layer, each neuron serves as a representation of a distinct feature. In the hidden layers, each neuron is responsible for performing both dimensional transformations and nonlinear operations on these features, while the neurons in the output layer compute the ultimate output result. The neurons in each layer are interconnected through the utilization of weights and biases, and transmitted to the subsequent layer following nonlinear transformations. As an example, the outcome of the first neuron within the initial hidden layer ${h_1}^1$ can be calculated as:

$$ {h_1}^1 = f\left(\sum_{i=1}^{k} w_{i1}x_i + b_1\right) \tag{1} $$

where, x_i represents the i^{th} input feature, w_{i1} is the coefficient between x_i and ${h_1}^1$, k is the feature number of a sample. After summarizing the inputs, the bias b_1 used to tune the activation function is added. Then, the activation function f is employed to change the linear system into nonlinear. The values of other neurons in the hidden layer are also obtained through the same calculation, then step by step, the complex mapping relationship between the output layer and the input layer $y_{pred}^{(i)}$ can be obtained:

$$ y_{pred}^{(i)} = \widehat{f}(x, w, b) \tag{2} $$

where $\widehat{f}$ represents the complex function learned by itself.

2.1.2. Long short-term memory neural network

The RNN tends to forget relevant information as the temporal gap between such information and the point of need increases due to the inherent short-term information dependence [23]. The long short-term memory network (LSTM) was introduced to address this limitation [24], with its structural representation depicted in Fig. 2.

The LSTM introduces a gating mechanism that employs element-wise multiplication and a σ layer for information selection. The element-wise multiplication calculates the correlation between two vectors, while the σ layer produces a value between 0 and 1, determining the extent to which specific information should be allowed to pass through. Within the LSTM architecture, three distinctive gating structures are incorporated: the input gate, the forget gate, and the output gate. The forget gate is used to determine which information is to be retained in the previous cell state. The equations for the σ layer and the forget gate f_t are as follows:

$$ \sigma = \frac{1}{1 + e^{-x}} \tag{3} $$

$$ f_t = \sigma\left(W_{f,x} \times X_t + W_{f,h} \times h_{t-1} + b_f\right) \tag{4} $$

After that, it becomes imperative to delineate the fresh information harbored within the cell state, which comprises two pivotal components. The first facet encompasses the update information i_t, precisely determined by the input gate, while the second facet encapsulates the novel candidate value $\widetilde{C}_t$, artfully generated by the tanh layer, effectively compressing the value to a range between -1 and 1. The equations for the tanh layer, the update information i_t and the novel candidate value $\widetilde{C}_t$ are as follows:

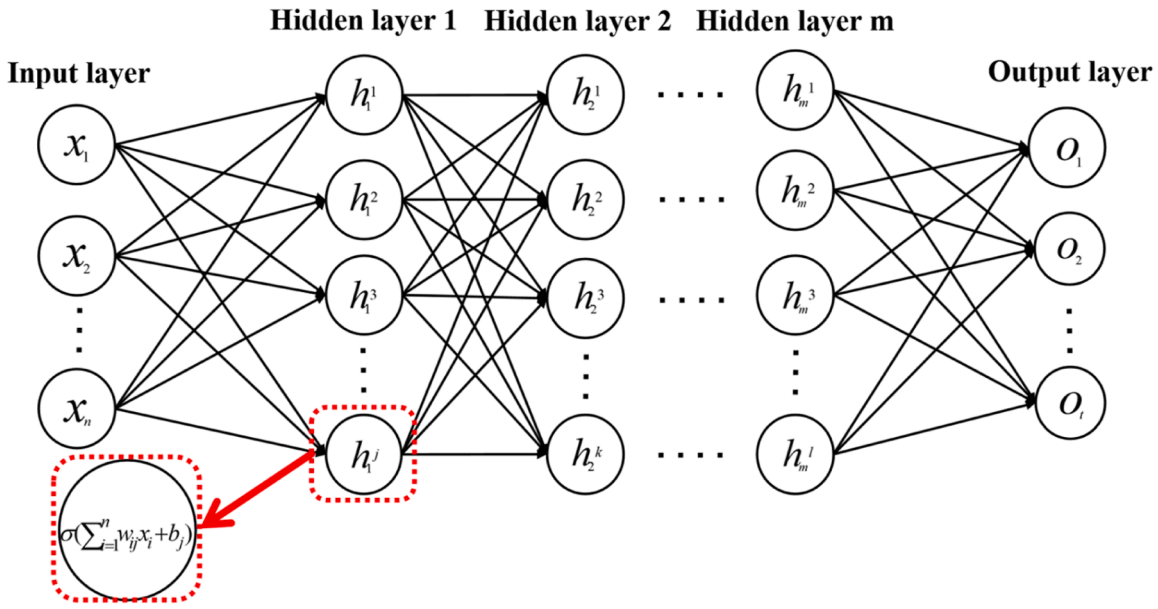


Fig. 1. Network architectures of a FNN.

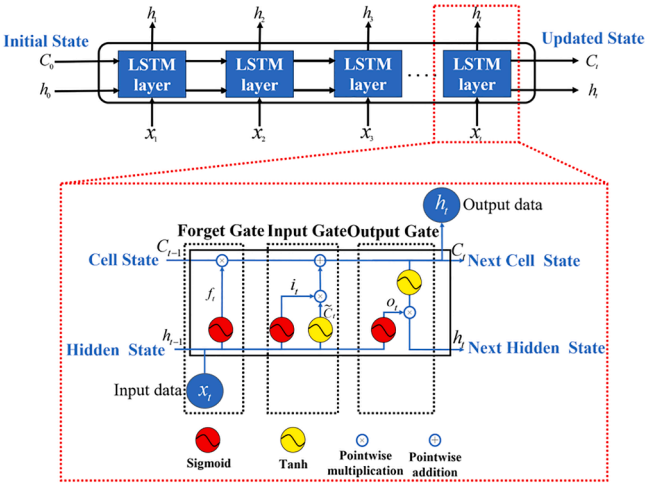


Fig. 2. Network architectures of LSTM.

$$\tanh = \frac{\sinh(x)}{\cosh(x)} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{5}$$

$$i_t = \sigma\left(W_{i,x} \times X_t + W_{i,h} \times h_{t-1} + b_i\right) \tag{6}$$

$$\widetilde{C}_t = \tanh\left(W_{C,x} \times X_t + W_{C,h} \times h_{t-1} + b_C\right) \tag{7}$$

In the process of updating the prior cell state C_{t-1} to the current cell state C_t, a straightforward operation is employed. It involves the multiplication of the previous cell state C_{t-1} by the output of the forgetting gate f_t and the product of the candidate cell state $\widetilde{C}_t$ and the outcome of the input gate i_t. The mathematical representation for C_t is explicitly elucidated as follows:

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t \tag{8}$$

After that, the LSTM produces the filtered cell state o_t. To obtain the current LSTM output h_t, o_t is subjected to multiplication with C_t after undergoing compression through the tanh layer. The precise mathematical expressions for o_t and h_t are detailed as follows:

$$o_t = \sigma\left(W_{o,x} \times X_t + W_{o,h} \times h_{t-1} + b_o\right) \tag{9}$$

$$h_t = o_t \times \tanh(C_t) \tag{10}$$

where $W_{f,x}$,$W_{f,h}$,$W_{i,x}$,$W_{i,h}$,$W_{C,x}$,$W_{C,h}$,$W_{o,x}$ and $W_{o,h}$ are weights, b_f,b_i,b_C and b_o are biases.

The Bi-LSTM neural network represents an enhanced iteration of the LSTM neural network, as detailed in reference [24]. While the LSTM neural network solely processes information in a unidirectional manner, specifically from the past input information, it does not incorporate input data from future observations. In contrast, the Bi-LSTM neural network consists of two LSTM neural networks with opposing transmission directions: one operating from the front to the back and the other from the back to the front. The bidirectional nature enables the Bi-LSTM neural network to amalgamate information from both directions, facilitating a collaborative approach to predicting output values. Consequently, the Bi-LSTM neural network has the capacity to learn from both past and future data. The architectural depiction of the Bi-LSTM neural network, with the LSTM unit serving as a recurring building block, is illustrated in Fig. 3.

2.1.3. Self-attention mechanism

The attention function plays a crucial role in computing the correlation between a query vector and a set of vectors that comprises keys and corresponding values. Weights are assigned to value vectors based on their correlation with the query vector. The ultimate output vector is generated as a linear combination of all the value vectors, as outlined in reference [25]. In this particular study, the scaled dot product attention mechanism is adopted. The computational process closely resembles matrix multiplication, and the potential impact of larger vectors on

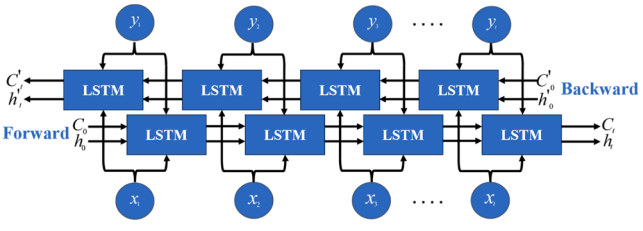


Fig. 3. Network architectures of Bi-LSTM.

attention values are mitigated by scaling down the product. For a more detailed insight into the structure of this mechanism, please refer to Fig. 4.

Initially, the input undergoes multiplication with three weight matrices W^q,$W^k$and W^v to yield the corresponding query matrix Q, key matrix K, and value matrix V. From the query matrix Q, the query vector Q^t with a dimension of d_k is extracted. Subsequently, the dot product is computed between the query vector Q^t and each key vector in the key matrix K. After removing $\sqrt{d_k}$, the weights for the query vector Q^t and each key vector in the key matrix K are determined via the softmax layer, resulting in the formation of a weight vector. After that, the weighted values of the value matrix are generated by applying the weight vector to each point in the value matrix V . The attention vector O^t of the query vector Q^t, the key matrix K, and the value matrix V is then accumulated. In conclusion, the formula for calculating attention between query matrix, key matrix and value matrix, incorporating the softmax layer are as followed:

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{11}$$

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$

2.2. T-Bi-LSTM

Discretizing the stacking sequences is a compelling need due to the inherent complexity of analyzing the stacking sequences of composite stiffened panels. Previous research efforts have attempted to discretize the stacking sequences by incorporating discrete features, such as ply angle and single-layer thickness, in accordance with the CLT. However, the discretization approach is rooted in the Kirchhoff hypothesis, which fails to account for out-of-plane loads and deformations, ultimately resulting in a loss of crucial features.

The data-driven method is a more favorable alternative to mitigate the feature loss stemming from the discretization of stacking sequences. The data-driven method operates without the constraints of pre-existing hypotheses, allowing it to extract features aimed at enhancing neural network convergence. Consequently, the data-driven method exhibits higher accuracy and greater flexibility. In prior studies, researchers employed ANN to process stacking sequences by inputting the ply angle for each layer. However, the ANN model is limited to handling stacking sequences of fixed lengths. Recognizing the capacity of the RNNs to manage variable-length sequences, the RNN is employed to extract the features of stacking sequences. The T-Bi-LSTM is proposed in the paper, which is built upon the Bi-LSTM to achieve a more comprehensive and accurate feature extraction. The T-Bi-LSTM introduces mechanisms such as self-attention, residual normalization, and adaptive pooling, as depicted in Fig. 5. The network is structured into three main components: data preprocessing, residual normalization block, and feature extraction.

In the initial phase of data preprocessing, it is essential to adapt the input data to the tensor matrix format required by the PyTorch [26] deep learning framework. Specifically, variable-length sequences are entailed transforming into matrices through padding. A key consideration in this process involves addressing the numerical values associated with the ply angle, which exhibit no discernible positive or negative correlation with the buckling load of the stiffened panel. Indexing is employed as a corrective measure to mitigate the influence of these numerical values. Subsequently, the indexed sequences are processed with embedding operation. The primary objective here is to convert the discrete index vector into a continuous vector while simultaneously expanding the data's dimensionality. The embedding operation plays a pivotal role in enhancing the data's feature richness, ultimately facilitating the neural network's learning process. It is worth noting that neural networks tend to glean more valuable insights and rules from data that is continuous and densely represented [27].

In the following section, the residual normalization block, a critical component of the network architecture, is introduced. The block comprises two distinct residual normalizations, incorporating both residual connections and normalization techniques. The integration of these elements serves a dual purpose: enhancing the network's ease of optimization, expediting training, and bolstering the network's overall generalization capabilities, as noted in prior studies [28,29]. Upon preprocessing the input data and feeding it into the Bi-LSTM, a pivotal step ensues: the incorporation of the self-attention mechanism. The self-attention mechanism proves invaluable, allowing the network to discern and process vital information efficiently within constrained computational resources. Notably, the mechanism serves as the cornerstone for combating the challenge of information overload, a pervasive issue in complex networks. Historically, attention mechanisms deployed as extensions of the RNNs, aimed at augmenting their performance [30–32]. Building upon the foundation, the residual of the Bi-LSTM and the outcomes of the self-attention mechanism undergo further refinement through residual normalization. Subsequently, the refined results are channeled into a FNN, amplifying the structure's nonlinear fitting capacity significantly. Continuing the iterative process, the residuals derived from the output of the FNN and those from the self-attention mechanism undergo yet another round of residual normalization. The meticulous series of operations culminate in a comprehensive construction of residuals, a process essential for the network's robustness and adaptability.

Finally, the focus shifts to feature extraction. The pool layer is employed to extract features, as it offers a more natural approach that

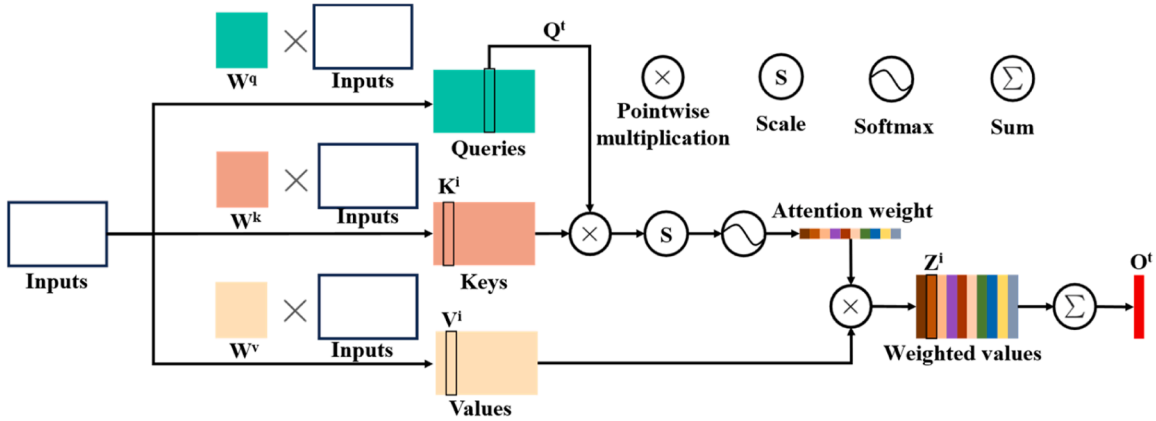


Fig. 4. Network architectures of scaled dot-product self-attention.

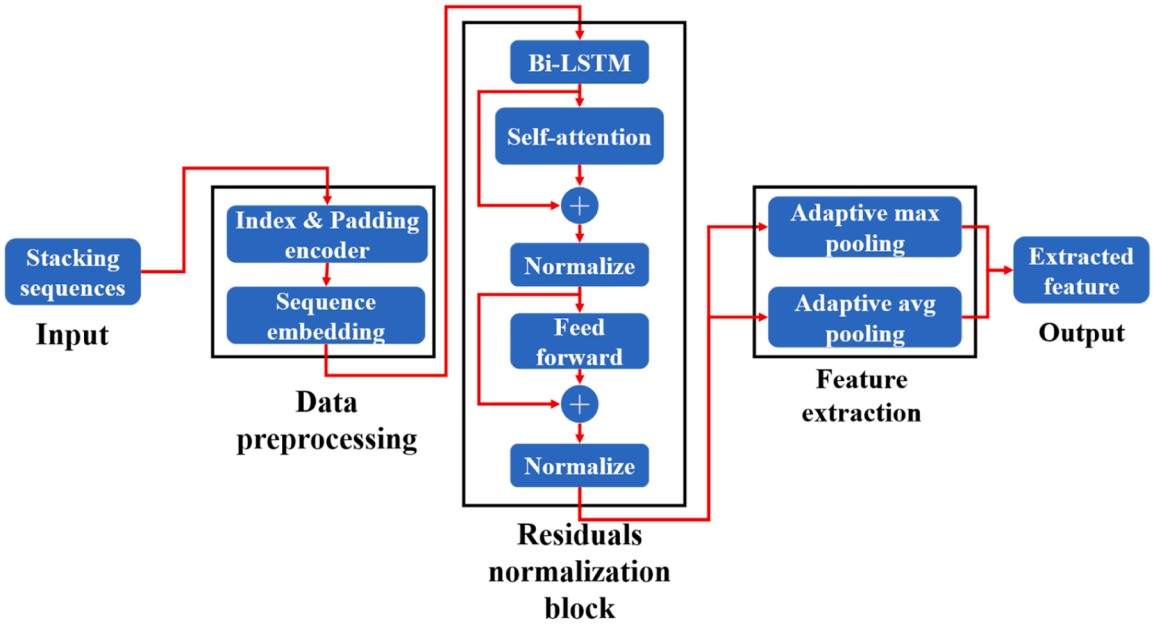


Fig. 5. Network architectures of T-Bi-LSTM network.

does not require parameter training. Notably, utilizing the pooling layer mitigates the risk of network overfitting, a concern prevalent in sophisticated neural networks [33]. Moreover, the feature extraction process in this study is approached from two distinct angles: maximum and average pooling. The dual-perspective approach enriches the extracted features, imbuing them with both depth and breadth. By diversifying the extraction methods, the resulting feature set becomes more comprehensive, enhancing the network's ability to capture nuanced patterns and correlations within the data.

2.3. Methodology of PNN feature extraction method

The design of composite stiffened panels requires careful consideration of two distinct data types: the stacking sequences and a multitude of discrete variables. The inherently dissimilar data types pose a challenge for simultaneous processing through a single network. Drawing inspiration from the GoogLeNet [34], a novel method is introduced: the development of the PNN structure dedicated to extracting features from these two disparate data. To elaborate, the method involves the deployment of the T-Bi-LSTM in parallel with the FNN. The dual-network setup facilitates the extraction of features from the stacking sequences using the T-Bi-LSTM, while concurrently extracting features from the discrete variables using the FNN. Fig. 6 illustrates the stepwise process of the PNN feature extraction method, which can be segmented into three distinct phases: data preparation, network construction, and model training and testing.

Step 1: Data preparation.

The effective training of neural networks necessitates access to a substantial volume of dependable data. Given the financial constraints and logistical implications of experimental endeavors, it's impractical to rely solely on experimental data. To address this challenge, this study employs a two-pronged strategy. Initially, the ABAQUS FE model is subjected to validation through experimental testing. Subsequent to this validation, a dataset is generated via Python scripting. The strategy guarantees both the reliability of the data and affords control over its quantity.

Step 2: Network construction.

The variables that generate the dataset are divided into stacking sequences and discrete variables. The features of the stacking sequence are extracted by T-Bi-LSTM network, and the features of the discrete

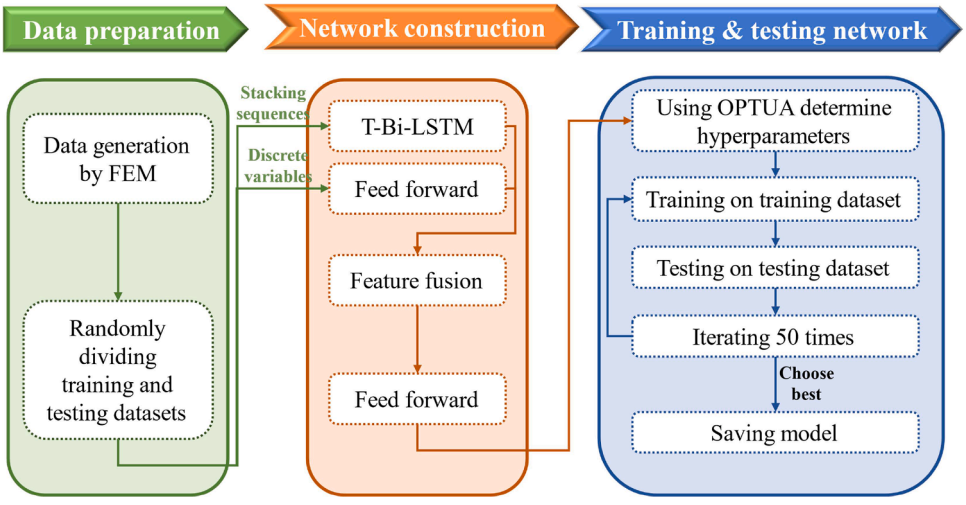


Fig. 6. A flow chart of PNN feature extraction method.

variables are extracted by the FNN. After that, the extracted features are fused. The feature fusion method of vector splicing is used, which is a common feature fusion method. The vectors are connected in the feature dimension, which not only achieves the function of feature fusion, but also expands the feature dimension. Finally, the fused features are put into the FNN to complete the construction of the network.

Step 3: Model training and testing.

In a machine learning model, two distinct categories of parameters exist. The first encompasses model parameters, which undergo learning from the available data. The second category, referred to as hyperparameters, necessitates pre-configuration before the training process commences. Due to the expansive search space that hyperparameters occupy and their profound impact on the model's outcomes, the principal challenge and focal point in model training resides in pinpointing a suitable set of hyperparameters. The optuna parameter adjustment framework for hyperparameter selection in this paper. Optuna empowers users to dynamically construct the search space and execute an efficient parameter exploration, rendering it a robust tool for hyperparameter optimization [35]. The process of network training and testing unfolds as follows: initial determination of hyperparameters through optuna, followed by model training on the designated dataset and subsequent evaluation on a separate testing dataset. Based on the model's performance on both sets, optuna iteratively refines the hyperparameters using the chosen algorithm, subsequently repeating the training and testing phases. To strike a balance between computational time and algorithm convergence, it is recommended to prescribe a total of 50 updates in this paper, culminating in the selection and preservation of the best-performing model. In summation, the model's training and testing regimen is concluded.

The aforementioned delineates the comprehensive procedure of the PNN feature extraction method. It is imperative to highlight that in cases where the model exhibits poor convergence, an initial scrutiny should be directed towards the reliability and comprehensiveness of the dataset. The accuracy of simulation verification and the adequacy of the data represent the pivotal determinants influencing the model's convergence. Furthermore, in the realm of hyperparameters selection, if the presented results fail to converge, a prudent approach would involve incrementing the number of updates until convergence is achieved.

3. PNN based prediction of buckling load

3.1. Data preparation

3.1.1. FE analysis

The test specimen comprises a skin, four stiffeners, and two transverse ribs, featuring a longitudinal length of 780 mm and a transverse length of 541 mm. Notably, 50mm-wide clamping boundaries are situated at both the upper and lower ends of the test piece. For an actual representation of the test specimen, refer to Fig. 7(a), while Fig. 7(b) provides insights into its dimensions and structure. The skin and stiffeners are crafted from AC531/CCF800H carbon fiber single-layer prepreg composite, boasting a single-layer thickness of 0.14 mm. Detailed stacking sequences can be found in Table 1, and comprehensive material parameters are presented in Table 2. The stiffeners are uniformly spaced

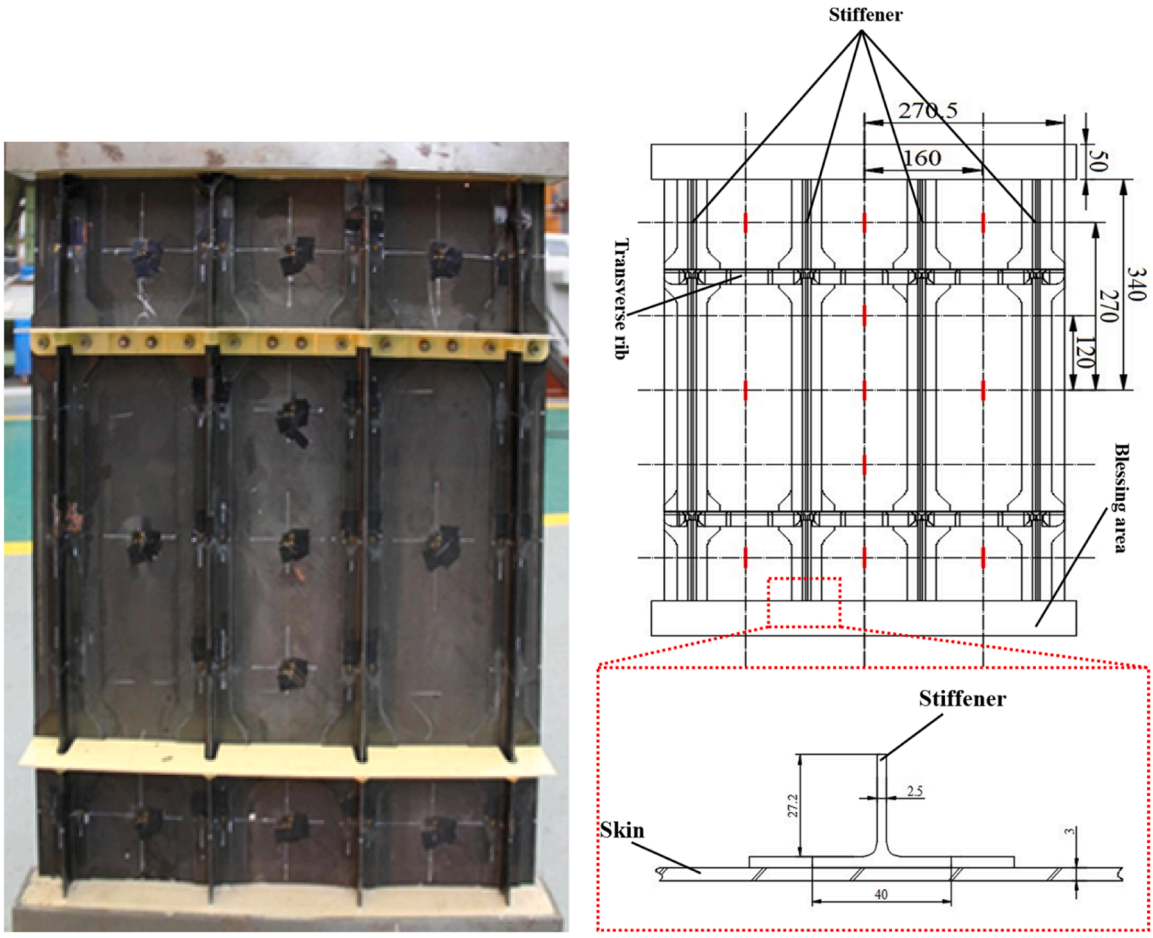


Fig. 7. (a) Photo of composite hat-stiffened panel and (b) geometric dimensions.

Tabel 1
Lay-up information of specimen.

Material	Lay-up for skin	Lay-up for stiffener
AC631/ CCF800H	[45/-45/0/-45/0/45/0/90/-45/ 0/45/90]s	[45/0/-45/0/90/0/45/0/-45/0]s

Tabel 2
Performance parameters of composite materials.

Parameters	Value	Parameter	Value
E_{11} /GPa	155	E_{22}/GPa	8
X_t /MPa	2550	Y_t/MPa	60
X_c /MPa	1440	Y_c/MPa	200
G_{12} /GPa	4	G_{13}/GPa	4
G_{23} /GPa	4	S_{12}/MPa	95
v_{12}	0.3		

at intervals of 160 mm and are securely bonded to the skin. Meanwhile, the transverse rib is fabricated from aluminum alloy, exhibiting elastic modulus of 71 GPa and Poisson's ratio of 0.33. The primary purpose of the transverse rib is to replicate the lateral elastic brace between the panel and the rib.

The resistance strain gauge plays a crucial role in assessing the strain state across the surface of the test specimen. To pinpoint the precise placement of these strain gauges, please refer to Fig. 8. Given that the central region of the skin constitutes the primary focus for analysis, the attention is directed toward the strain data within this specific area. The primary analysis zone is denoted by the red box in the illustration. Within this central analysis area, a total of 5 sets of strain gauges are thoughtfully affixed, both on the front and rear sides. Each strain gauge within the analysis area bears a unique label to facilitate data analysis. It's important to note that the labels enclosed in brackets on the diagram represent those affixed to the rear side of the respective strain gauge, while labels outside the brackets pertain to the front side of the strain gauge. For instance, the label "1 (6)" signifies that the front strain gauge at this position bears the label "1," and the rear strain gauge is labeled as "6."

In this study, ABAQUS is employed for the numerical simulation of the specimen, and the constraints and boundary conditions are depicted in Fig. 9. Due to the presence of 50 mm wide clamping areas at the upper and lower ends of the test piece, the stiffness of these regions within the model has been enhanced through motion coupling. The clamped regions have been coupled with two reference points, namely the loading point and the constrained point, respectively. The coupling is essential to streamline the application of loads and constraints. The loading point is restricted to motion solely along the longitudinal translational degree

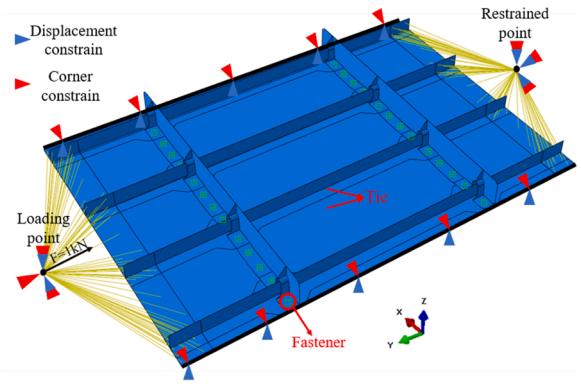


Fig. 9. Boundary constraints.

of freedom, where a longitudinal loading force of 1 kN is applied. Meanwhile, the constrained point is subject to complete constraint. To ensure that the stiffened panel experiences pure compression during testing, both sides of the panel are firmly clamped using a fixture. Consequently, in the simulation model, the degree of freedom outside the plane is constrained on both sides of the stiffened panel. The tie contact is employed to bind stiffeners to the skin and the fastener is utilized between the transverse rib and the plate to replicate the screw connection in the test, where the screw diameter is set at 8 mm.

ABAQUS is also employed to mesh the composite stiffened panel utilizing the S4R element type. To assess the mesh convergence of our simulation model, the mesh seed size is varied to 5 mm, 6 mm, 7 mm, 8 mm, and 10 mm, resulting in total element counts of 33,619, 24,798, 183,200, 14,974, and 12,811, respectively. The computations are conducted on a device equipped with the 12th Gen Intel Core i7-12,700 CPU. The corresponding calculation times are 273 s, 211 s, 160 s, 140 s, and 118 s, respectively. Fig. 10 illustrates the grid convergence verification diagram, displaying the local grid divisions of the model under the corresponding grid seed sizes, along with the respective calculation times and buckling loads. The trend where the buckling load converges with increasing calculation time is revealed by the analysis. Considering both the computational efficiency and the degree of convergence in the results, a 5 mm grid seed size is selected for grid division in this study.

The load-strain curve of the area highlighted in the red box in Fig. 8 is illustrated in Fig. 11. An initial linear increase in both strain and load is depicted. However, once the compression load reaches 391 kN, there is a noticeable 5 % difference between the positive and negative strain values at the same observation point, accompanied by a clear bifurcation in the curve. The strain bifurcation phenomenon signifies the onset

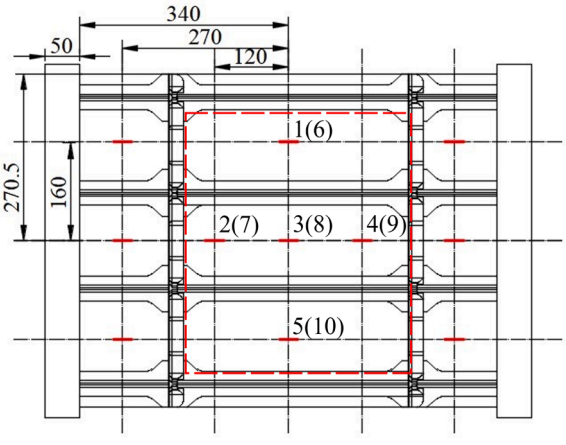


Fig. 8. Schematic diagram of measurement point location.

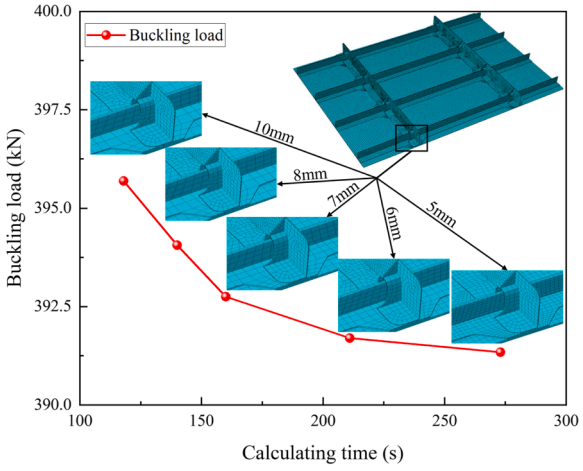


Fig. 10. Meshing convergence analysis.

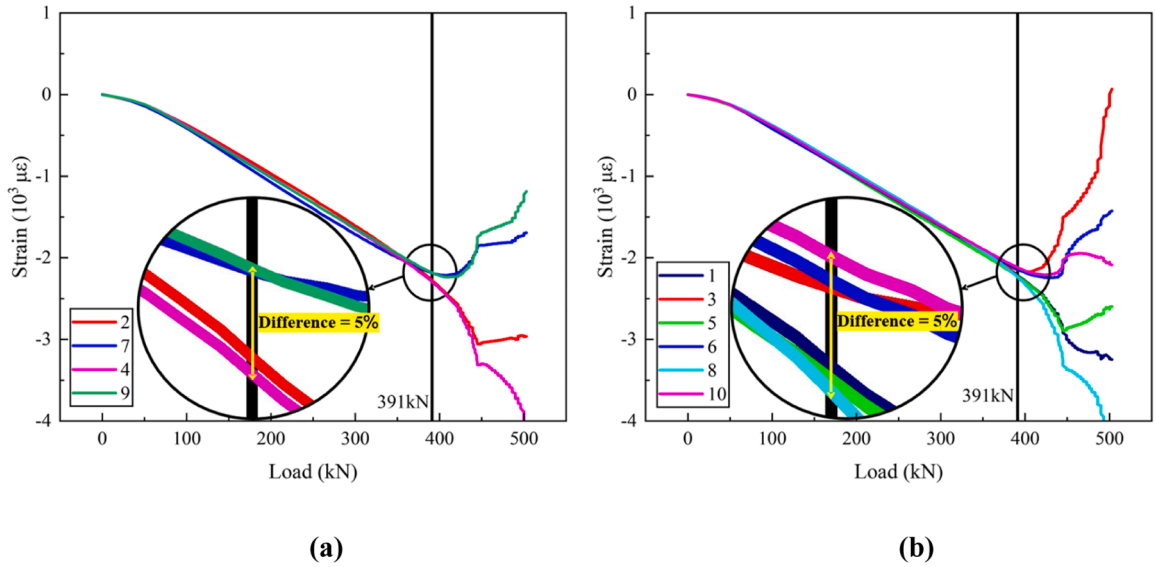


Fig. 11. Strain–load curves(a)left and right, (b)middle side.

of local buckling at this specific location, with one side experiencing compression and the other undergoing tension. The load corresponding to this bifurcation point in strain is termed the buckling load. The linear eigenvalue analysis is employed to determine the buckling eigenvalue, with the subspace iteration method utilized for calculation. The first-order buckling load measures 391.34 kN, demonstrating an error margin of less than 1 %.

The buckling mode of the test can be determined by analyzing the deflection of the load-strain curve exhibited by the specimen after buckling. For instance, when the curve labeled as '1′ deflects in a downward direction and the curve marked as '6′ deflects upwards, it can be inferred that the upper surface is experiencing compressive forces while the lower surface is subjected to tension. Consequently, the buckling mode in this instance is concave. By examining the post-buckling behavior of the load-strain curve, the buckling mode diagram is constructed, as depicted in Fig. 12(a). Additionally, a simulated buckling mode diagram, presented in Fig. 12(b), is derived. Upon comparing Fig. 12(a) with Fig. 12(b), it becomes evident that the simulated buckling mode aligns with the observed behavior, thus validating the accuracy of our simulation model. In conclusion, the simulation model is confirmed to be accurate.

3.1.2. *Data generation*

Variations in the design parameters are introduced to explore the correlation between buckling load and the stacking sequence of both the skin and stiffener, as well as the thickness of a single layer in this study. The ply angles are randomly selected from among 0°, 45°, -45°, and 90°, ensuring a symmetrical distribution in the stacking sequences, to adhere to engineering best practices. Given the symmetric distribution of the stacking sequences, only half of the input is necessary. The length of the skin stacking sequence varies randomly between 8 and 16, while the stiffener stacking sequence length fluctuates between 8 and 12. The single layer thickness is randomized within the range of 0.125 to 0.175, with three significant digits maintained. A comprehensive dataset comprising 1500 data sets is generated by employing the previously validated finite element model and through the secondary development

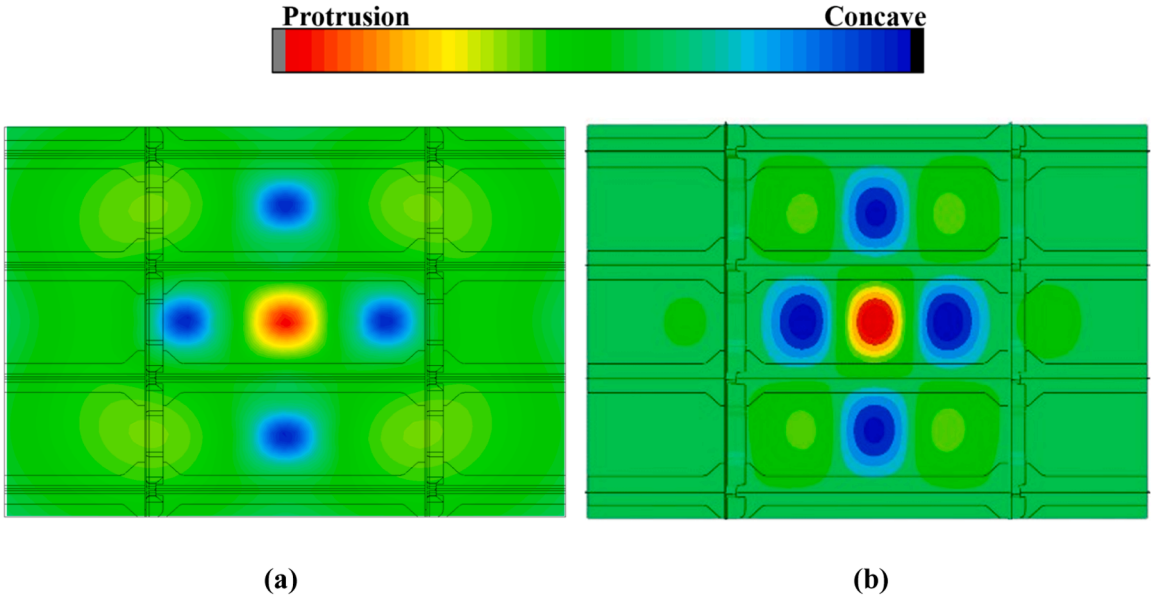


Fig. 12. Buckling mode of (a) test (b) finite element analysis.

of ABAQUS Python. Among the dataset, 80 % are allocated for the training dataset, while the remaining 20 % are reserved for the testing dataset.

3.2. Network construction and generalization analysis

Fig. 13 illustrates the architecture of the parallel neural network employed in this study. The T-Bi-LSTM is specifically employed to extract the stacking sequences of both the skin and stiffener. Subsequently, the FNN is utilized to extract the feature related to single-layer thickness. After that, the extracted feature vector undergoes processing within the FNN, ultimately concluding the network operations.

The specific modifications are employed to the FNN structure within the feature processing component to enhance network generalization and mitigate overfitting. Two key adjustments are made: Firstly, the addition of a dropout layer after the activation layer, and secondly, the incorporation of a batch normalization layer following the activation layer. Regrettably, the endeavors result in network non-convergence. The phenomenon is proceeded to analyze in the subsequent sections.

The dropout layer plays a crucial role in the training of deep learning networks by temporarily excluding neural network units based on a specified probability. The mechanism compels individual neural units to collaborate with randomly selected counterparts, enhancing overall performance and diminishing the interdependence among neuron nodes, thus boosting network generalization [36]. Despite the merits, the incorporation of the dropout layer in the network under study hindered convergence. The underlying issue stems from the fact that each unit within the network encapsulates vital information during the feature processing stage. Consequently, discarding any unit results in the loss of significant and effective information, impeding the network's ability to converge successfully.

The purpose of the batch normalization layer is to expedite the model's convergence rate, enhance its stability, and promote generalization. The purpose is achieved by maintaining consistent input distributions across all layers within the neural network for a given batch of data [37]. However, the challenge arises when attempting to apply the batch normalization layer in scenarios where convergence proves elusive. The root of the problem lies in the nature of the inputs during the regression phase. The inputs consist of variable-length stacking sequences and single-layer thickness features, extracted through the PNN. Given the variable-length nature of the stacking sequences, the input probability distribution within the same batch becomes disparate. Herein lies the issue: the batch normalization layer forcefully homogenizes these input distributions, thereby disrupting the inherent relationship between features. The disruption ultimately culminates in a failure to achieve convergence. Notably, researchers delved into this phenomenon, shedding light on the limitations of the batch normalization layer in certain sequence processing contexts [38].

The specific measures are implemented to enhance the network's generalization capabilities, as outlined below:

(1) Selection of activation function.

The SELU activation function is opted for due to its inherent self-normalization characteristics. The SELU activation function effectively addresses challenges related to the disappearance and explosion of network gradients. By doing so, faster convergence of the network is facilitated, and the generalization ability of the model is improved [39]. The formula for the SELU activation function is elucidated as follows.

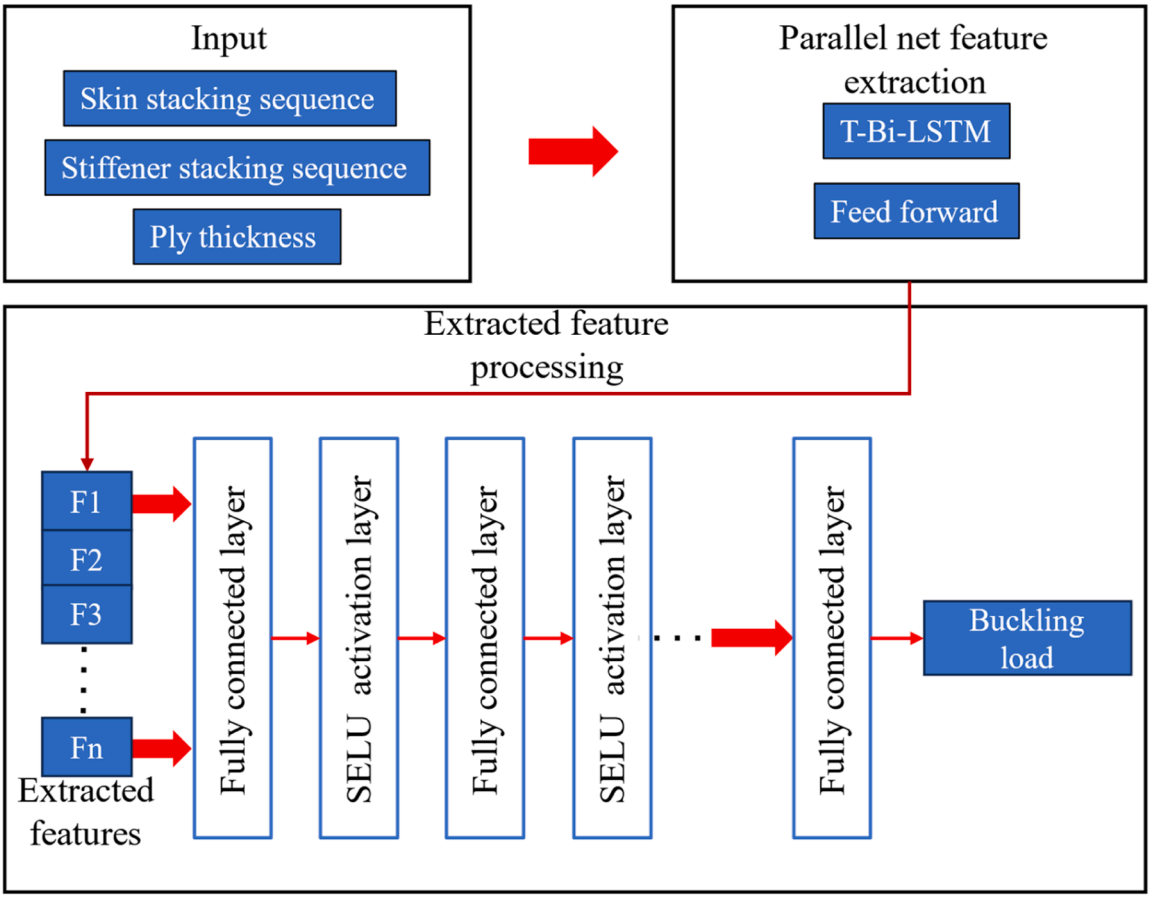


Fig. 13. Network architectures of a PNN.

$$SELU(x) = \lambda_1 \begin{cases} x & if\ x > 0 \\ \alpha e^x - \alpha & if\ x \le 0 \end{cases} \tag{13}$$

where λ_1 and α are parameter that the network can learn.

(2) Selection of loss function.

The Mean Squared Error (MSE) is designated as the primary loss function. The auxiliary L_2 regularization term is integrated into the loss function to enhance the model's generalization capacity and counteract overfitting. The supplementary term proves highly effective in constraining the network's scale and thus mitigating the risks of overfitting. The comprehensive formulation of the loss function, now incorporating this beneficial L_2 regularization term, is illustrated as follows.

$$Loss(y, \widehat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 + \frac{\lambda_2}{2}\| w \|_2 \tag{14}$$

where y_i signifies the actual or true value, while $\widehat{y}_i$ represents the predicted value. $\| w \|_2$ corresponds to the L_2 norm of the parameter vector w and $\lambda_2$denotes the regularization coefficient.

(3) Selection of metrics.

Given that the MSE metric is highly influenced by outliers and the inclusion of a regularization term is intricately tied to the model's complexity, a multifaceted standpoint should be sought to assess the model's performance from. The Mean Absolute Percentage Error (MAPE) is introduced as an evaluation metric to gauge the model's predictive accuracy to determine whether overfitting is occurred. The MAPE, an acronym for the average percentage error, is a metric that exhibits sensitivity to absolute errors. The computation formula is explicitly depicted as follows.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\widehat{y}_i - y_i}{y_i}\right| \tag{15}$$

(4) Selection of optimizer.

The choice of optimizer falls upon the Adam optimizer. The Adam optimizer possesses the unique capability to dynamically adjust the learning rate based on historical gradient information. The adaptive feature allows for the utilization of a larger learning rate during the initial phases of training, facilitating rapid convergence. Moreover, a smaller learning rate is gracefully transitioned in the later stages of training, enhancing the ability to accurately locate the minimum of the loss function. Furthermore, the Adam optimizer incorporates parameter regularization during updates, thereby contributing to the enhancement of network generalization [40].

(5) Selection of early stopping mechanism.

The early stopping mechanism is employed which involves halting the model training process when it is determined that further training is unlikely to enhance generalization performance. The preventative measure guards against overfitting, a consequence of excessive model training. Extensive research attests to the efficacy of early stopping in bolstering network generalization [41]. The testing loss is adopted as the criterion for early stopping due to the testing loss on the testing dataset is the most straightforward metric for gauging model generalization. Considering both training duration and model convergence, if after 200 iterations the testing loss fails to exhibit further reduction, it is concluded that the model's generalization performance can not be substantially improved. At this juncture, the early stopping mechanism is triggered, and the model with the lowest testing loss is preserved.

3.3. Network training and testing

The hyperparameters are determined by simulated annealing based on the optuna parameter adjustment framework. The hyperparameters encompass crucial elements, including the embedding dimension of the stacking sequence following the sequence embedding operation, the dimensions of the T-Bi-LSTM outputs responsible for extracting the skin stacking sequence and the stiffener stacking sequence, and the FNN output for extracting discrete feature thickness. These are respectively referred to as the embedding size, skin hidden size, stiffener hidden size, and thickness hidden size. A total of 50 parameter adjustments are executed to achieve optimal settings, as illustrated in Fig. 14, which provides a visual representation of the parameter adjustment process. Notably, as depicted in Fig. 14, following the execution of 50 parameter adjustments, a discernible trend towards convergence in the parameter adjustments becomes evident.

The model's performance is assessed through a dual lens, examining both loss and MAPE. Given the primary emphasis on bolstering the model's generalization prowess during training, the evaluation metrics of choice are the testing loss and testing MAPE, serving as pivotal indicators of model quality. Lower values in testing loss and testing MAPE correspond to superior model performance. Concurrently, the scrutiny extends to the training loss and training MAPE metrics. Disparities between the loss and MAPE values observed on the training dataset versus the testing dataset may indicate potential issues of overfitting or underfitting. Employing the comprehensive evaluation approach, Table 3 presents the top ten models.

4. Results and discussion

To comprehensively evaluate the processing impacts of two distinct feature extraction methods, namely, the CLT feature extraction method and the PNN feature extraction method, on both stacking sequences and discrete features, as well as to assess the feature extraction capabilities of two neural network architectures, T-Bi-LSTM and Bi-LSTM, with regard to stacking sequences, four neural network models are trained in the present study. Among all four models, two are developed based on the pioneering work of Mallela and Sun et al. Since both of these models employ the CLT method for feature extraction, referring to them as CLT (Mallela) and CLT (Sun), respectively. The remaining two models utilize the PNN method for feature extraction. Specifically, one employs the Bi-LSTM network to extract layer sequences, and the other leverages the T-Bi-LSTM network for the same purpose. These two models are denoted as PNN (Bi-LSTM) and PNN (T-Bi-LSTM), respectively. Fig. 15 illustrates the Loss and MAPE curves for these four models as the number of

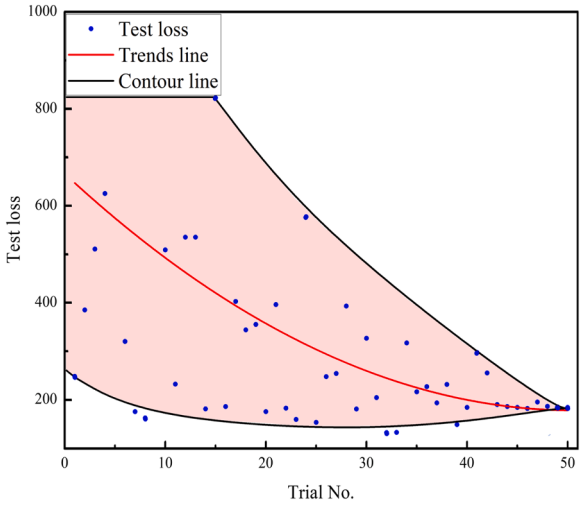


Fig. 14. Parameter adjustment process.

Tabel 3
Result of parameter adjustment.

Embedding size	Skin hidden size	Stiffener hidden size	Thickness hidden size	Train loss	Train MAPE	Test loss	Test MAPE
2048	2048	2048	4096	117.94	0.032	131.65	0.034
2048	2048	2048	2048	176.72	0.038	132.01	0.04
2048	2048	1024	1024	179.74	0.037	148.03	0.039
2048	4096	2048	2048	106.12	0.031	152.59	0.038
2048	1024	2048	2048	26.38	0.017	158.64	0.032
2048	4096	1024	2048	133.32	0.031	161.75	0.040
2048	4096	2048	1024	110.28	0.033	174.62	0.035
4096	1024	1024	2048	279.30	0.049	174.65	0.039
2048	512	2048	1024	126.71	0.033	180.11	0.039
4096	512	2048	2048	123.62	0.030	180.26	0.040

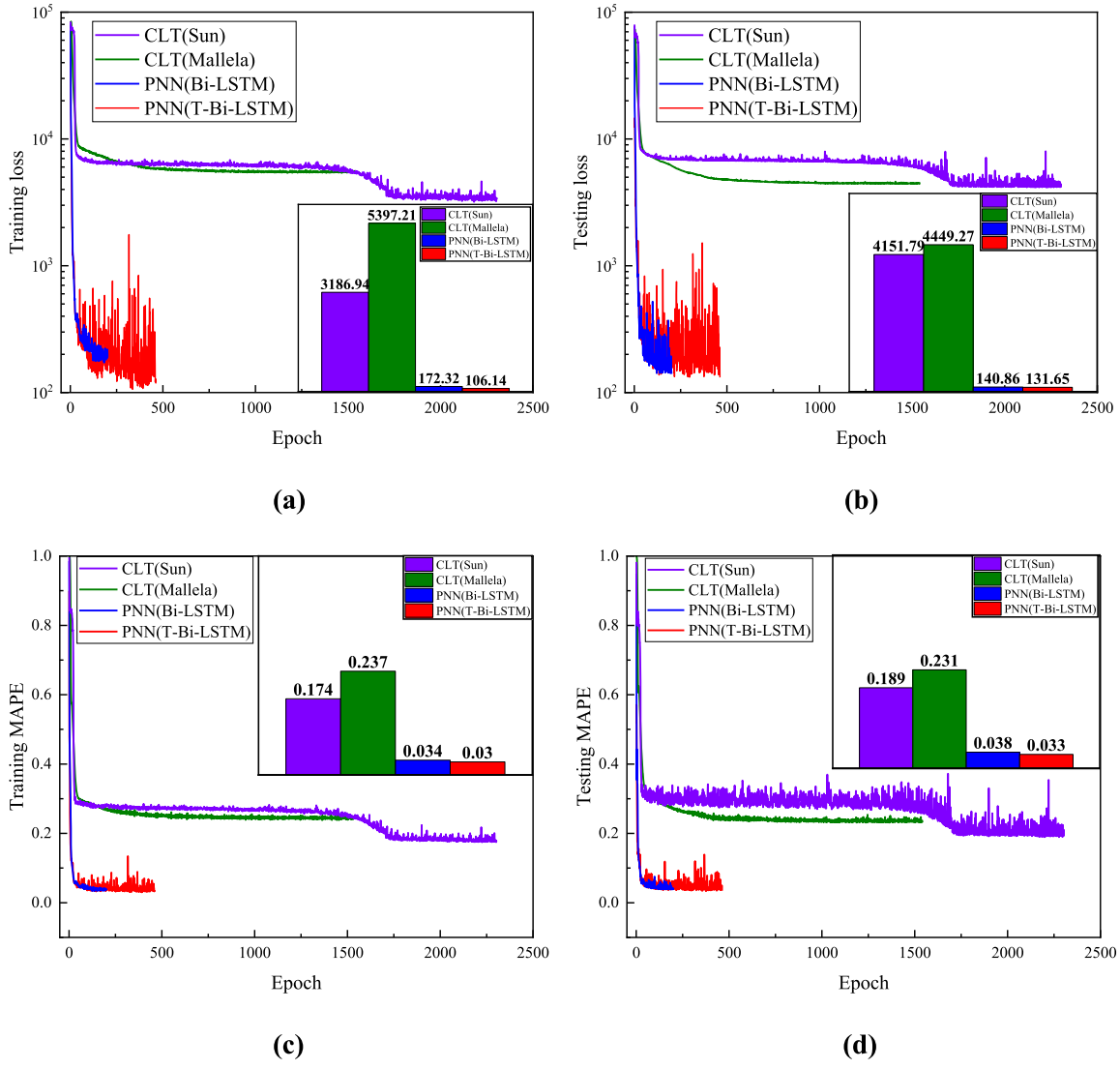


Fig. 15. Comparison of (a) training loss(b) testing loss(c) training MAPE(d) testing MAPE for predicting buckling loads using four different models.

iterations increases during training. The accompanying bar graph in the figure provides the corresponding metric values achieved when the networks reach convergence.

Fig. 15 reveals that the performance of the CLT network on the dataset in this article is markedly inferior to that of PNN. Network performance on dataset can be assessed from two perspectives: convergence speed and degree of convergence. According to the convergence rate analysis, when CLT (Sun) converges, the number of iterations is 2304; for CLT (Mallela), it's 1543; for PNN (T-Bi-LSTM), it's 462; and for PNN (Bi-LSTM), it's 201. The number of iterations for the CLT network is at least three times that of the PNN network. Moreover, the graph clearly shows that the slope of the loss and MAPE curves of the PNN network is significantly greater than that of the CLT network as the number of model iterations increases. The degree of convergence of the model can be assessed through its loss and MAPE values on the dataset during convergence. It is also evident from the curve that when the CLT network tends to converge, the loss and MAPE curves are much higher than those of the PNN network. The bar chart provides a more intuitive

observation, indicating that the loss and MAPE of the CLT network on the dataset are much greater than those of PNN. In summary, the CLT feature extraction method exhibits markedly inferior performance in processing layer sequences and discrete features compared to the PNN feature extraction method. The reasons for this phenomenon are as follows:

(1) The CLT (Mallela) network takes as its input the features extracted by Mallela et al. through CLT, which specifically pertain to the shear buckling load of T-shaped stiffened panels [17]. On the other hand, the CLT (Sun) network utilizes features extracted by Sun et al. via CLT, primarily focused on the axial compressive buckling load of hat-stiffened panels [19]. However, the dataset employed in this article is intended for predicting the axial compressive buckling load of T-shaped stiffened panels. Consequently, the inputs of these two networks are not entirely suitable for predicting the axial compressive load of T-shaped reinforced panels, resulting in unsatisfactory performance on the dataset presented in this article. In relative terms, PNN significantly enhances network flexibility and streamlines the feature processing procedure through the data-driven method.

(2) The dataset generation approach for CLT (Mallela) and CLT (Sun) involves defining the feature range [15,17]. This dataset generation approach is effective in substantially reducing the incidence of outliers and simplifying dataset complexity. In contrast, the dataset employed in this article directly modifies parameters related to stiffened panels and subsequently transforms them into features, drawing from Mallela and Sun papers. This method does not impose constraints on feature ranges, resulting in a more intricate dataset for this article. However, the models employed by CLT (Mallela) and CLT (Sun) tend to be overly simplistic. The research consistently demonstrated that such simplistic models are susceptible to underfitting when dealing with complex datasets [42]. In comparison, PNN exhibits greater complexity and boasts enhanced non-linear fitting capabilities, thus enabling it to achieve commendable results even in the face of complex datasets.

Fig. 15 demonstrates that the PNN (Bi-LSTM) exhibits a lower level of fluctuation in the training set curve compared to the PNN (T-Bi-LSTM). However, given the consistent mapping relationship between input and output data in both the training and testing datasets, a proficiently trained model should manifest akin trends in the loss and MAPE curves across these two datasets. By analyzing the loss and MAPE curves of the PNN (T-Bi-LSTM) and the PNN (Bi-LSTM) individually on both the training and testing datasets, it is observed that the PNN (T-Bi-LSTM) exhibits consistent curve fluctuations across the testing and training datasets. Conversely, the PNN (Bi-LSTM) demonstrates notably lower curve volatility on the training dataset in comparison to the testing dataset. This phenomenon strongly suggests that the knowledge garnered by the PNN (T-Bi-LSTM) during its training phase effectively translates to the testing dataset, whereas the insights acquired by the PNN (Bi-LSTM) during its training do not seamlessly transfer.

The phenomenon of the PNN (Bi-LSTM) requiring fewer epochs to converge than the PNN (T-Bi-LSTM) is evident in Fig. 15. It is observed that both the PNN (T-Bi-LSTM) and the PNN (Bi-LSTM) exhibit identical descent speeds at the initial phase of the curve, and convergence is defined in this study as the testing loss failing to decrease after 200 training iterations. As a result, the PNN (Bi-LSTM) achieves convergence faster than PNN (T-Bi-LSTM), suggesting that the PNN (Bi-LSTM) struggles to further reduce loss and MAPE when trapped in a local optimum. The pronounced volatility, extended iterations, and superior performance demonstrated by the PNN (T-Bi-LSTM) curve collectively suggest its enhanced capability to escape local optima.

Collectively, these observations underscore the limited learning capacity and inferior model performance of the PNN (Bi-LSTM) when compared to the PNN (T-Bi-LSTM). Furthermore, the bar graph in Fig. 15 serves as an intuitive visual confirmation, highlighting the substantial divergence between the PNN (Bi-LSTM) testing loss and training loss at model convergence, thereby reinforcing this conclusion. The reinforced learning capacity and model performance of the PNN (T-Bi-LSTM) model can be attributed to the integration of mechanisms such as self-attention, residual normalization, and adaptive pooling. While processing the stacking sequence, the self-attention mechanism augments the model's focus on crucial features, thus advancing its efficacy. The residual normalization block configuration amplifies the model's non-linear fitting ability, and the incorporation of adaptive pooling amplifies the breadth of features extracted by the model.

Fig. 16 provides a more intuitive comparison of the generalization performance between the PNN (T-Bi-LSTM) and the PNN (Bi-LSTM). In Fig. 16(a), it is evident that the fitting quality of the PNN (T-Bi-LSTM) surpasses that of the PNN (Bi-LSTM). In the visualization, the red dots represent the predicted values of the PNN (T-Bi-LSTM), the blue dots represent those of the PNN (Bi-LSTM), and the black line represents a straight line where the true values match the predicted values. Notably, both red and blue dots are distributed on either side of the black line, but the red dots cluster closer to the black line. The R^2 statistic, a widely employed regression evaluation metric, is calculated to gauge the fitting performance. A higher R^2 value signifies a superior fitting effect, with 1 being the highest attainable score. The R^2 value for the PNN (T-Bi-LSTM) stands at 0.9909, marking a 0.76 % improvement over the PNN (Bi-LSTM). Fig. 16(b) delves into the fitting stability, illustrating that the PNN (T-Bi-LSTM) exhibits greater stability compared to the PNN (Bi-LSTM). In the representation, the red line represents the predicted values curve of the PNN (T-Bi-LSTM), the blue line represents that of the PNN (Bi-LSTM), and the black line signifies the curve of true values. Evidently, the red curve displays significantly lower volatility in contrast to the blue curve. The MSE is computed to quantify this stability. The PNN (T-Bi-LSTM) showcases a 25 % lower MSE compared to the PNN (Bi-LSTM), strongly emphasizing the superior stability of the PNN (T-Bi-LSTM). In conclusion, it is evident that T-Bi-LSTM outperforms Bi-LSTM in extracting features from stacking sequences, as indicated by its superior fitting degree and fitting stability.

5. Conclusion

A PNN feature extraction method is proposed in this paper to predict the buckling load of composite stiffened panels with different stacking sequences and discrete variables. This method can effectively process variable length and long stacking sequences by paralleling the RNN and the FNN, and powerfully reduce feature loss, simplify the process of feature processing and improve the application scope of the network through data-driven mechanism. The dataset for training and testing is prepared using FE model verified by experiment, where the PNN(T-Bi-LSTM)and other three contrasting models, the CLT(sun), the CLT(Mallela) and the PNN(Bi-LSTM), are trained and compared . The fundamental conclusions are obtained as follows:

(1) The PNN exhibits significantly superior convergence speed and accuracy compared to CLT networks. The result identifies that the CLT network suffers from feature loss and complexity in its processing, while the PNN adeptly circumvents these limitations through data-driven approach. Leveraging its robust learning capabilities, the PNN demonstrates remarkable proficiency in predicting the buckling load of stiffened panels, even in situations with intricate features and complex scenarios.

(2) Despite both the PNN (T-Bi-LSTM) and the PNN (Bi-LSTM) employing the PNN feature extraction method, the former demonstrates superior generalization performance on the testing dataset. The R^2 value for the PNN (T-Bi-LSTM) stands at 0.9909, showcasing a substantial 0.76 % enhancement over the PNN (Bi-LSTM). Furthermore, the MSE value for the PNN (T-Bi-LSTM) is 85.26, signifying a remarkable 25 % reduction in comparison to the PNN (Bi-LSTM). These results highlight

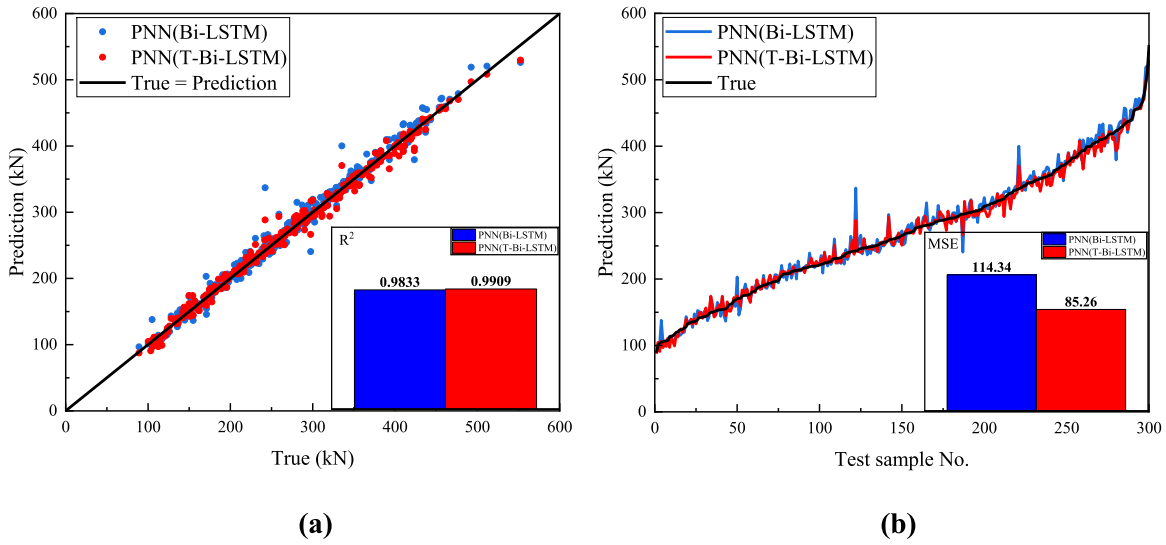


Fig. 16. Comparison of fitting (a) degree (b) stability between the PNN(Bi-LSTM) and the PNN(T-Bi-LSTM).

the T-Bi-LSTM achieves the superiority over the Bi-LSTM by leveraging the self-attention mechanism, residual connection, AVG/max pooling, and other mechanisms to obtain richer and more comprehensive features from the stacking sequences.

The above statement indicates that both the PNN feature extraction method and the T-Bi-LSTM stacking sequence extraction neural network can be effectively employed in the structural optimization and mechanical performance prediction of composite stiffened panels.

CRediT authorship contribution statement

Tao Zhang: Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Peiyan Wang:** Writing – review & editing, Visualization, Validation, Supervision. **Jianwei Fu:** Formal analysis, Data curation. **Suian Wang:** Writing – review & editing, Visualization, Validation, Supervision, Conceptualization. **Chenchen Lian:** Validation, Supervision, Project administration.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This investigation was supported by the Natural Science Foundation of Shaanxi Province, grant number [S2021-JC-YB-0590]. The authors would also like to deliver their sincere thanks to the editors and anonymous reviewers.

References

[1] M.A. Bessa, S. Pellegrino, Design of ultra-thin shell structures in the stochastic post-buckling range using Bayesian machine learning and optimization, Int. J. Solids Struct. 139-140 (2018) 174–188, https://doi.org/10.1016/j.ijsolstr.2018.01.035.
[2] U.K. Mallela, A. Upadhyay, Buckling of laminated composite stiffened panels subjected to in-plane shear: a parametric study, Thin-Walled Struct. 44 (2006) 354–361, https://doi.org/10.1016/j.tws.2006.03.008.
[3] M.-K. Kazi, F. Eljack, E. Mahdi, Data-driven modeling to predict the load vs. displacement curves of targeted composite materials for industry 4.0 and smart manufacturing, Compos. Struct. 258 (2021) 113207, https://doi.org/10.1016/j.compstruct.2020.113207.
[4] Y. Mo, D. Ge, J. Zhou, Experiment and analysis of hat-stringer-stiffened composite curved panels under axial compression, Compos. Struct. 123 (2015) 150–160, https://doi.org/10.1016/j.compstruct.2014.11.074.
[5] P. Pevzner, H. Abramovich, T. Weller, Calculation of the collapse load of an axially compressed laminated composite stringer-stiffened curved panel–An engineering approach, Compos. Struct. 83 (2008) 341–353, https://doi.org/10.1016/j.compstruct.2007.05.001.
[6] R. Bai, Z. Lei, X. Wei, W. Tao, C. Yan, Numerical and experimental study of dynamic buckling behavior of a J-stiffened composite panel under in-plane shear, Compos. Struct. 166 (2017) 96–103, https://doi.org/10.1016/j.compstruct.2017.01.022.
[7] D. Liu, R. Bai, R. Wang, Z. Lei, C. Yan, Experimental study on compressive buckling behavior of J-stiffened composite panels, Opt. Lasers Eng. 120 (2019) 31–39, https://doi.org/10.1016/j.optlaseng.2019.02.014.
[8] R. Bai, S. Bao, Z. Lei, C. Liu, Y. Chen, D. Liu, C. Yan, Experimental study on compressive behavior of I-stiffened CFRP panel using fringe projection profilometry, Ocean Eng. 160 (2018) 382–388, https://doi.org/10.1016/j.oceaneng.2018.04.085.
[9] S. Hasebe, R. Higuchi, T. Yokozeki, S. Takeda, Internal low-velocity impact damage prediction in CFRP laminates using surface profiles and machine learning, Compos. Part B Eng. 237 (2022) 109844, https://doi.org/10.1016/j.compositesb.2022.109844.
[10] D. Cristiani, F. Falcetelli, N. Yue, C. Sbarufatti, R. Di Sante, D. Zarouchas, M. Giglio, Strain-based delamination prediction in fatigue loaded CFRP coupon specimens by deep learning and static loading data, Compos. Part B Eng. 241 (2022) 110020, https://doi.org/10.1016/j.compositesb.2022.110020.
[11] A. Bhaduri, A. Gupta, L. Graham-Brady, Stress field prediction in fiber-reinforced composite materials using a deep learning approach, Compos. Part B Eng. 238 (2022) 109879, https://doi.org/10.1016/j.compositesb.2022.109879.
[12] C. Zhang, Y. Li, B. Jiang, R. Wang, Y. Liu, L. Jia, Mechanical properties prediction of composite laminate with FEA and machine learning coupled method, Compos. Struct. 299 (2022) 116086, https://doi.org/10.1016/j.compstruct.2022.116086.
[13] A.R. Reddy, B.S. Reddy, K.V.K. Reddy, Application of design of experiments and artificial neural networks for stacking sequence optimizations of laminated composite plates, Int. J. Eng. Sci. Technol. 3 (2011) 295–310, https://doi.org/10.4314/ijest.v3i6.24.
[14] J.A. Artero-Guerrero, J. Pernas-Sánchez, J. Martín-Montal, D. Varas, J. López-Puente, The influence of laminate stacking sequence on ballistic limit using a combined Experimental/FEM/Artificial Neural Networks (ANN) methodology, Compos. Struct. 183 (2018) 299–308, https://doi.org/10.1016/j.compstruct.2017.03.068.
[15] Z. Sun, Z. Lei, R. Bai, H. Jiang, J. Zou, Y. Ma, C. Yan, Prediction of compression buckling load and buckling mode of hat-stiffened panels using artificial neural network, Eng. Struct. 242 (2021) 112275, https://doi.org/10.1016/j.engstruct.2021.112275.
[16] Z. Sun, Z. Lei, J. Zou, R. Bai, H. Jiang, C. Yan, Prediction of failure behavior of composite hat-stiffened panels under in-plane shear using artificial neural network, Compos. Struct. 272 (2021) 114238, https://doi.org/10.1016/j.compstruct.2021.114238.
[17] U.K. Mallela, A. Upadhyay, Buckling load prediction of laminated composite stiffened panels subjected to in-plane shear using artificial neural networks, Thin-Walled Struct. 102 (2016) 158–164, https://doi.org/10.1016/j.tws.2016.01.025.

[18] S. Kumar, R. Kumar, S. Mandal, A.K. Rahul, The prediction of buckling load of laminated composite hat-stiffened panels under compressive loading by using of neural networks, TOCIEJ 12 (2018) 468–480, https://doi.org/10.2174/1874149501812010468.
[19] X. Liu, J. Qin, K. Zhao, C.A. Featherston, D. Kennedy, Y. Jing, G. Yang, Design optimization of laminated composite structures using artificial neural network and genetic algorithm, Compos. Struct. 305 (2023) 116500, https://doi.org/10.1016/j.compstruct.2022.116500.
[20] T.R.C. Chuaqui, A.T. Rhead, R. Butler, C. Scarth, A data-driven Bayesian optimisation framework for the design and stacking sequence selection of increased notched strength laminates, Compos. Part B Eng. 226 (2021) 109347, https://doi.org/10.1016/j.compositesb.2021.109347.
[21] H. Liu, Z. Zhang, H. Jia, Q. Li, Y. Liu, J. Leng, A novel method to predict the stiffness evolution of in-service wind turbine blades based on deep learning models, Compos. Struct. 252 (2020) 112702, https://doi.org/10.1016/j.compstruct.2020.112702.
[22] Q. Chen, R. Jia, S. Pang, Deep long short-term memory neural network for accelerated elastoplastic analysis of heterogeneous materials: an integrated data-driven surrogate approach, Compos. Struct. 264 (2021) 113688, https://doi.org/10.1016/j.compstruct.2021.113688.
[23] Gradient flow in recurrent nets: the difficulty of learning LongTerm dependencies, A Field Guide to Dynamical Recurrent Networks, IEEE, 2009, https://doi.org/10.1109/9780470544037.ch14.
[24] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, arXiv.Org. (2017). https://arxiv.org/abs/1706.03762v7 (accessed October 13, 2023).
[26] https://pytorch.org/.
[27] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv.Org. (2013). https://arxiv.org/abs/1301.3781v3 (accessed October 13, 2023).
[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.
[29] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer Normalization, arXiv.Org. (2016). https://arxiv.org/abs/1607.06450v1 (accessed October 13, 2023).
[30] Z. Niu, G. Zhong, H. Yu, A review on the attention mechanism of deep learning, Neurocomputing 452 (2021) 48–62, https://doi.org/10.1016/j.neucom.2021.03.091.
[31] G. Brauwers, F. Frasincar, A general survey on attention mechanisms in deep learning, IEEE Trans. Knowl. Data Eng. 35 (2023) 3279–3298, https://doi.org/10.1109/TKDE.2021.3126456.
[32] S. Chaudhari, V. Mithal, G. Polatkan, R. Ramanath, An Attentive survey of attention models, arXiv.Org. (2019). https://arxiv.org/abs/1904.02874v3 (accessed October 13, 2023).
[33] M. Lin, Q. Chen, S. Yan, Network in network, arXiv.Org. (2013). https://arxiv.org/abs/1312.4400v3 (accessed October 13, 2023).
[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, arXiv.Org. (2014). https://arxiv.org/abs/1409.4842v1 (accessed October 13, 2023).
[35] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: a next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, NY, USA, Association for Computing Machinery, 2019, pp. 2623–2631, https://doi.org/10.1145/3292500.3330701.
[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[37] Batch normalization: accelerating deep network training by reducing internal covariate shift, (n.d.). https://arxiv.org/abs/1502.03167v3 (accessed November 22, 2023).
[38] S. Shen, Z. Yao, A. Gholami, M.W. Mahoney, K. Keutzer, PowerNorm: rethinking batch normalization in transformers, arXiv.Org. (2020). https://arxiv.org/abs/2003.07845v2 (accessed November 22, 2023).
[39] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, arXiv.Org. (2017). https://arxiv.org/abs/1706.02515v5 (accessed October 13, 2023).
[40] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, CoRR. (2014). https://www.semanticscholar.org/paper/Adam%3A-A-Method-for-Stochastic-Optimization-Kingma-Ba/a6cb366736791bcccc5c8639de5a8f9636bf87e8 (accessed October 13, 2023).
[41] L. Prechelt, Early stopping-but when? In G, Lecture Notes in Computer Science. 1524 (1998).
[42] J. Huang, L. Qu, R. Jia, B. Zhao, O2U-Net: a simple noisy label detection approach for deep neural networks, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 3325–3333, https://doi.org/10.1109/ICCV.2019.00342.