Parameterized Consistency Learning-based Deep Polynomial Chaos Neural Network Method for Reliability Analysis in Aerospace Engineering

Xiaohu Zheng, Wen Yao, Yunyang Zhang, Xiaoya Zhang

*Corresponding author. E-mail: wendy0782@126.com

ABSTRACT Polynomial chaos expansion (PCE) is a powerful surrogate model-based reliability analysis method in aerospace engineering. Generally, a PCE model with a higher expansion order is usually required to obtain an accurate surrogate model for some non-linear complex stochastic systems. However, the high-order PCE increases the labeled training data cost for solving the expansion coefficients. To alleviate this problem, this paper proposes a parameterized consistency learning-based deep polynomial chaos neural network (Deep PCNN) method, including the low-order adaptive PCE model (the auxiliary model) and the high-order polynomial chaos neural network (the main model). The expansion coefficients of the high-order main model are parameterized into the learnable weights of the polynomial chaos neural network. The auxiliary model uses a proposed unsupervised consistency loss function to assist in training the main model. The Deep PCNN method can significantly reduce the training data cost in constructing a high-order PCE model without losing surrogate model accuracy by using a small amount of labeled data and many unlabeled data. A numerical example validates the effectiveness of the Deep PCNN method, and the Deep PCNN method is applied to analyze the reliability of two aerospace engineering systems.

KEYWORDS Polynomial chaos expansion, deep neural network, parameterized, consistency learning, reliability analysis, aerospace engineering.

1. Introduction

Aerospace engineering always has uncertain factors like environmental fluctuations and manufacturing deviation [1]. These factors will affect the performance of an aerospace vehicle. Thus, reliability analysis is extremely important and essential for designing an aerospace vehicle [2,3]. Monte Carlo simulation (MCS) is an effective and simple approach for reliability analysis. However, MCS needs a large number of samples to obtain an accurate failure probability. In aerospace engineering, the cost of obtaining large amounts of aerospace vehicle performance data is very expensive and unacceptable. To reduce the number of samples, the first-order and second-order reliability methods [4] construct the local approximation of the system performance function at the most probable point (MPP). However, the MPP is hard to find exactly, which will lead to an inaccurate estimation
of failure probability. Therefore, many surrogate model-based reliability analysis methods have been studied in recent years, such as polynomial chaos expansion (PCE) [5-7], deep neural network (DNN) [8,9], Kriging method [10,11], etc. Among these methods, PCE is a powerful stochastic expansion method for constructing a surrogate model.

PCE can accurately describe the randomness of random variables with any arbitrary distribution based on polynomial chaos theory. The PCE modeling includes two parts, i.e., constructing the orthogonal polynomial basis and solving the expansion coefficients. The constructing method of the orthogonal polynomial basis is very mature. Xiu et al. [5] proposed the generalized polynomial chaos (gPC) to construct the orthogonal polynomial basis of the random variable obeying the common probability distribution like Gaussian, Gamma, Beta, Uniform, etc. For the random variable obeying the arbitrary probability distribution, Oladyshkin et al. [6,7] proposed the arbitrary polynomial chaos expansion (aPC) that uses the raw moments of the random variable to construct the orthogonal polynomial basis. After constructing the orthogonal polynomial basis, the PCE model is built by calculating the sum of the product of the expansion coefficients and the orthogonal polynomial basis. Generally, the expansion coefficients can be solved by the projection method [12] or the regression method [13]. For some non-linear complex stochastic systems, a PCE model with a higher expansion order is usually required to obtain an accurate surrogate model. However, the higher the expansion order, the more expansion coefficients need to be solved, resulting in the increase of labeled training data for solving the expansion coefficients.

Some common approaches like the dimension-reduction model (DRM) [14,15] and the sparse PCE (SPCE) [16-18] are studied for solving the above problem. The DRM approaches, such as the adaptive bivariate DRM [19], the contribution-degree analysis-based DRM [20], decompose the complex stochastic systems into many low-dimensional subsystems and then construct the PCE models. Besides, the common SPCE approaches like the orthogonal matching pursuit SPCE [21], the least angle regression SPCE [22], and the subspace pursuit SPCE [23] detect the important orthogonal polynomial basis automatically and then only solve the expansion coefficients of the important terms. Although the DRM and the SPCE can effectively reduce the number of samples for solving the expansion coefficients, these two methods influence the precision of the surrogate model. Another good technical route is to use some unlabeled data to solve the expansion coefficients with the deep neural network (DNN), solving the cost problem of the labeled training data. Two typical DNN-based PCE methods are the physics-informed neural network aPC (PINN-aPC) [24] and the deep adaptive arbitrary polynomial chaos expansion (Deep aPCE) [25]. Based on some physical knowledge like partial differential equations and boundary conditions, the PINN-aPC [24] can use some unlabeled training data to calculate the partial differential equation loss and the boundary condition loss. Then, the parameters of DNN are learned to solve the expansion coefficients. However, the physical knowledge of some stochastic systems (black-box problem) is difficult or even impossible to acquire, which means the PINN-aPC cannot be applied for the black-box problem. Unlike the PINN-aPC, the Deep aPCE [25] calculates the adaptive aPC property loss by some unlabeled training data to assist in learning the parameters of DNN. Although
the Deep aPCE adopts less labeled training data than the existing PCE method, it still needs more labeled training data to solve the expansion coefficients of higher-order PCE. Thus, how to use fewer data to solve the expansion coefficients of high-order PCE without losing surrogate model accuracy is a problem worth solving.

This paper proposes a parameterized consistency learning-based deep polynomial chaos neural network (Deep PCNN) method for reliability analysis. The Deep PCNN model consists of two models, i.e., the auxiliary model and the main model. Inspired by the Deep aPCE, the auxiliary model is constructed to be a low-order adaptive PCE model by a DNN, where the DNN solves the expansion coefficients of the low-order adaptive PCE model. The main model is a polynomial chaos neural network built based on a high-order PCE model. The polynomial chaos neural network includes an input layer, an orthogonal polynomial neural layer, a hidden layer, and an output layer with one neuron. The expansion coefficients of the high-order PCE model are parameterized to be the weights of the neuron in the output layer. Thus, the parameters of the Deep PCNN model include the parameters of the DNN in the auxiliary model and the parameterized expansion coefficients of the polynomial chaos neural network. The Deep PCNN method adopts three kinds of loss functions to learn the model parameters. By a small amount of labeled training data, two supervised loss functions are calculated to learn the parameters of the Deep PCNN model. Besides, the unsupervised loss function and the unsupervised consistency loss function are computed using many unlabeled training data. For training the DNN in the auxiliary model, the unsupervised loss function is constructed based on the properties of the adaptive aPC [25]. For the same unlabeled data, two predicted outputs of the auxiliary model and the main model should be consistent. Based on this idea, the unsupervised consistency loss function, training the polynomial chaos neural network, is proposed by calculating the mean one-norm of the predicted errors between the main and auxiliary models. Based on the above three kinds of loss functions, the parameterized consistency learning algorithm is proposed to learn the parameters of the Deep PCNN model by a small amount of labeled training data and many unlabeled training data. After training the Deep PCNN model, the MCS can be straightforwardly performed on the main model to analyze the reliability of the aerospace engineering systems.

In summary, the main innovations of the Deep PCNN method are presented in the following: 1) The Deep PCNN method parameterizes the expansion coefficients of the high-order PCE model into the learnable weights of the polynomial chaos neural network; 2) The Deep PCNN method uses the auxiliary model to assist in training the main model by the proposed unsupervised consistency loss function; 3) The Deep PCNN method can significantly reduce the training data cost in constructing a high-order PCE model without losing surrogate model accuracy. The rest of this paper is organized as follows. In section 2, the related theories, including PCE and DNN, are introduced. The Deep PCNN and the parameterized consistency learning algorithm are proposed in section 3. Compared with the existing PCE methods, the effectiveness of the proposed Deep PCNN method is validated by a numerical example in section 4. In section 5, the proposed Deep PCNN method is applied to analyze the reliability of two aerospace engineering systems.
2. Related theory

2.1. Polynomial chaos expansion

**PCE definition.** For a engineering system, suppose that the performance function is \( Y = g(X) \), where \( X = [X_1, X_2, \ldots, X_d]^T \) is the system input random variable (a \( d \)-dimensional vector). A \( p \)-order PCE model [26] approximates this performance function by a weighted linear combination of the multi-dimensional orthogonal basis \( \{ \Phi_1(\xi), \Phi_2(\xi), \ldots, \Phi_M(\xi) \} \), i.e.,

\[
Y \approx y^{(p)}(\xi) = \sum_{i=1}^{M} c_i \Phi_i(\xi),
\]

(1)

where \( \xi = (\xi_1, \xi_2, \ldots, \xi_d)^T \) is the corresponding variable of the input random variable \( X \) in the standard random space, \( c_i \) \( (i = 1, 2, \ldots, M) \) is the expansion coefficient, and \( M = (d + p)!/d!p! \). Generally, the expansion coefficients are solved by the projection method [12] or the regression method [13]. The construction approach of the multi-dimensional orthogonal basis is introduced in the following.

**Constructing multi-dimensional orthogonal basis.** Assume that \( X \) is a random univariable obeying the probability distribution \( f(X) \). If \( f(X) \) is a common probability distribution like Gaussian distribution, Gamma distribution, Beta distribution, etc., the correspondence of the type of orthogonal polynomial [5] to the type of random variable is shown in Table 1. If \( f(X) \) is not the common probability distribution described above, the random variable \( X \) is usually transformed to its closest common distribution by the Rosenblatt or Nataf transformation. After transforming \( X \) into a standard random variable \( \xi \), the univariate orthogonal basis \( \{ \phi^{(0)}(\xi), \phi^{(1)}(\xi), \ldots, \phi^{(p)}(\xi) \} \) is constructed based on orthogonal polynomial.

| Distribution type | Orthogonal polynomial | Support       |
|-------------------|-----------------------|--------------|
| Gaussian          | Hermite               | (\(-\infty, +\infty\)) |
| Gamma             | Laguerre              | \([0, +\infty)\) |
| Beta              | Jacobi                | \([a, b]\) |
| Uniform           | Legendre              | \([a, b]\) |
| Exponential       | Laguerre              | \([0, +\infty)\) |

Table 1 Correspondence of the type of orthogonal polynomial to the type of random variable [5].

In addition to the above univariate orthogonal basis constructing method, the aPC [7] use the raw moment of random univariable to construct the univariate orthogonal basis. Firstly, the random univariable \( X \) is centralized and standardized by linear
transformation $\xi = (X - \mu) / \sigma$, where $\mu$ and $\sigma$ are the mean and standard deviation of $X$, respectively. Then, the polynomial $\phi^j(\xi)$ of degree $j$ ($j = 0, 1, \ldots, p$) can be constructed by

$$\phi^j(\xi) = \sum_{n=0}^{j} a_m^{(j)} \xi^n,$$

where $a_m^{(j)}$ is the $m$th ($m = 0, 1, \ldots, j$) coefficient of $\phi^j(\xi)$. According to this reference [7], the coefficient $a_m^{(j)}$ is calculated by the raw moment $\mu^j_{\xi}$ ($\beta = 0, 1, \ldots, 2j - 1$) of random univariable $\xi$, i.e.,

$$a_m^{(j)} = \int \phi^j(\xi) \mu^j_{\xi} \, d\Gamma(\xi).$$

According to the above univariate orthogonal basis constructing method, the multi-dimensional orthogonal basis $\{\phi_1(\xi), \phi_2(\xi), \ldots, \phi_M(\xi)\}$ can be obtained by the univariate orthogonal basis $\{\phi_0^j(\xi_k), \phi_1^j(\xi_k), \ldots, \phi^{(p)}_j(\xi_k)\}$ of random variable $\xi_k$ ($k = 1, 2, \ldots, d$), i.e.,

$$\Phi(\xi) = \prod_{k=1}^{d} \phi_{\xi_k}^{(j)}(\xi_k),$$

where $\xi_k$ is a multivariate index that contains the individual univariate basis combinatoric information [7, 25].

**Properties of PCE.** The PCE’s expansion coefficients $\{c_1, c_2, \ldots, c_M\}$ in Eq. (1) have two properties [7]:

1) **Property I.** The first expansion coefficient $c_1$ is equal to the mean $E(Y)$ of the model output $Y$, i.e.,

$$E(Y) = c_1.$$  

2) **Property II.** The sum of squares of expansion coefficients $\{c_2, c_3, \ldots, c_M\}$ is equal to the variance of the model output $Y$, i.e.,

$$D(Y) = \sum_{i=2}^{M} (c_i)^2.$$
2.2. Deep neural network

DNN [27] includes an input layer (green circle), many hidden layers (yellow circle), and an output layer (red circle). Suppose that DNN has \( L \) hidden layers. As shown in Fig. 1, \( n_\alpha \) (\( \alpha = 1, 2, \cdots, L \)) denotes the number of neurons in the \( \alpha \)th hidden layer. For the input \( x = [x_1, x_2, \ldots, x_r]^T \), the output \( \hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_r] \) of DNN can be expressed as the formula

\[
\hat{y} = \mathcal{F} \left[ \mathcal{F} \left[ \cdots \mathcal{F} \left( x^T W_1 + b_1 \right) W_2 + b_2 \right] \cdots W_L + b_L \right] W_{L+1} + b_{L+1},
\]

(8)

where \( \mathcal{F}(\cdot) \) is the nonlinear activation function, \( W = \{W_1, W_2, \ldots, W_L, W_{L+1}\} \) and \( b = \{b_1, b_2, \ldots, b_L, b_{L+1}\} \) are the weights and the bias, respectively. In Fig. 1, \( \theta_{\alpha}^i \) means represents the weights and bias associated with the third neuron in the second hidden layer, and other symbols have similar meanings. Thus, \( \{\theta_1, \theta_2, \ldots, \theta_{n_1}\} = \{W_1, b_1\} \), \( \{\theta_1^i, \theta_2^i, \ldots, \theta_{n_1}^i\} = \{W_2, b_2\} \), \ldots \), \( \{\theta_{1}^L, \theta_2^L, \ldots, \theta_{n_L}^L\} = \{W_L, b_L\} \), \( \{\theta_1^{L+1}, \theta_2^{L+1}, \ldots, \theta_{n_{L+1}}^{L+1}\} = \{W_{L+1}, b_{L+1}\} \). Denoted that the parameter \( \theta \) of DNN is \( \theta = \{W, b\} \).

![Deep neural network](image)

Fig. 1. Deep neural network.

Given \( N \) training data \( \{(x_i, y_i) | i = 1, 2, \cdots, N\} \), the loss function \( \mathcal{L}(x, y; \theta) \) can be

\[
\mathcal{L}(x, y; \theta) = \frac{1}{N} \sum_{i=1}^{N} \|y_i - \hat{y}_i\|_r^r,
\]

(9)

where \( \| \cdot \|_r \) denotes the \( r \)-norm (\( r = 1 \) or \( r = 2 \)). Thereby, the parameter \( \theta \) is learned by minimizing the loss function \( \mathcal{L}(x, y; \theta) \), i.e.,

\[
\theta^* = \arg \min_{\theta} \mathcal{L}(x, y; \theta).
\]

(10)

3. Parameterized consistency learning-based Deep PCNN method for reliability analysis

3.1. Method framework

This paper proposes the parameterized consistency learning-based Deep PCNN method (Fig. 2) to solve the problem that many labeled data are required to calculate the higher-order PCE model’s expansion coefficients. As shown in Fig. 2, the Deep PCNN
method mainly includes the Deep PCNN model and the loss function $\mathcal{L}(c, \theta)$. For the Deep PCNN model, it consists of the main model $\hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; c)$ and the auxiliary model $\hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta)$. The main model $\hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; c)$ is a $p$-order polynomial chaos neural network, which is constructed by the parameterized expansion coefficients $c = \{c_1, c_2, \ldots, c_M\}$ and the multi-dimensional orthogonal basis $\{\Phi_i(\xi), \Phi_2(\xi), \ldots, \Phi_M(\xi)\}$. Inspired by the Deep aPCE method [25], the $\tilde{p}$-order auxiliary model $\tilde{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta)$ is built by the DNN (calculating the expansion coefficients $\{\tilde{c}_1(\xi; \theta), \tilde{c}_2(\xi; \theta), \ldots, \tilde{c}_{\tilde{p}}(\xi; \theta)\}$) and the multi-dimensional orthogonal basis $\{\tilde{\Phi}_1(\xi), \tilde{\Phi}_2(\xi), \ldots, \tilde{\Phi}_{\tilde{p}}(\xi)\}$. Thus, the parameters of the Deep PCNN model include the parameterized expansion coefficients $c$ and the parameters $\theta$ of DNN, i.e., $(c, \theta)$. For the loss function $\mathcal{L}(c, \theta)$, it is the sum of two loss functions, i.e., $\mathcal{L}_M(c)$ and $\mathcal{L}_A(\theta)$. Given the labeled training data $D_{ld} = \{(\xi_{l}^{ed}, y_{l}^{ed}) | l = 1, 2, \ldots, N_{ed}\}$ and the unlabeled training data $D_{lu} = \{\xi_{l}^{ue} | l = 1, 2, \ldots, N_{ue}\}$, the loss functions $\mathcal{L}_M(c)$ and $\mathcal{L}_A(\theta)$ is calculated based on $\hat{y}^{(p)}$ and $\tilde{y}^{(p)}$. Then, $\mathcal{L}_M(c)$ and $\mathcal{L}_A(\theta)$ are used to train the main model $\hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; c)$ and the auxiliary model $\tilde{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta)$, respectively.

**Fig. 2.** Framework of the parameterized consistency learning-based Deep PCNN method.

In summary, the construction of a surrogate model using the Deep PCNN method includes three parts, i.e.,
Building the Deep PCNN model;

Constructing the loss function \( \mathcal{L}(\mathbf{c}, \theta) \);

Training the Deep PCNN model based on the labeled training data and the unlabeled training data.

The above three parts are introduced in detail in sections 3.2, 3.3, and 3.4, respectively.

3.2. Deep polynomial chaos neural network

As shown in Fig. 2, the main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; \mathbf{c}) \) and the auxiliary model \( \hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta) \) make up the Deep PCNN model. The above two models are constructed as follows.

1) **Main model** \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; \mathbf{c}) \)

As shown in Fig. 3(a), supposed that a DNN with one hidden layer is used to approximate the performance function \( Y = G(\xi) \), where the parameters of DNN are \( \theta^p_{q_1} = \{W_{q_1}^p, b_{q_1}^p\} \) (\( q_2 = 1, 2, \cdots, M \)) and \( \theta^p = \{W_1^p, \cdots, W_M^p, b_2^p\} \). According to Eq.(8), the calculation of the output \( \hat{y} \) is

\[
\begin{align*}
    H_{q_1} &= \mathcal{F} \left( \sum_{q_{n+1}=1}^{q_1} W_{q_{n+1}}^p \xi_{q_{n+1}} + b_{q_{n+1}}^p \right), \\
    H &= (H_1, H_2, \cdots, H_M)^T, \\
    \hat{y} &= \xi^T H + b_2 = \sum_{q_{n+1}=1}^{q_1} W_{q_{n+1}}^p H_{q_{n+1}} + b_2.
\end{align*}
\] (11)

Inspired by the DNN output’s calculation method in Eq.(11), this paper constructs the main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; \mathbf{c}) \) to be a \( p \)-order polynomial chaos neural network as shown in Fig. 3(b), which is modeled based on the \( p \)-order PCE model in Eq.(1).

Compared with the DNN in Fig. 3(a), the \( p \)-order polynomial chaos neural network adds an orthogonal polynomial neural layer, i.e., rounded rectangles in Fig. 3(b). Besides, the values of neurons between the orthogonal polynomial neural layer and the hidden layer of Fig. 3(b) are in a one-to-one correspondence, i.e.,

\[
\begin{align*}
    H_i &= \Phi_i(\xi), \quad (i = 1, 2, \cdots, M), \\
    H &= (H_1, H_2, \cdots, H_M)^T.
\end{align*}
\] (12)

In the \( p \)-order polynomial chaos neural network, the expansion coefficients \( \mathbf{c} = \{c_1, c_2, \cdots, c_M\} \) are parameterized to be the weights of the output layer neuron as shown in Fig. 3(b). Based on Eq.(12), the output \( \hat{y}^{(p)} \) is calculated by

\[
\hat{y}^{(p)} = \xi^T H = \sum_{i=1}^{M} c_i H_i = \sum_{i=1}^{M} c_i \Phi_i(\xi).
\] (13)
Thus, the parameters of the main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; e) \) are the parameterized expansion coefficients \( e = \{c_1, c_2, \ldots, c_M\} \).

(a) The DNN with one hidden layer

(b) The \( p \)-order polynomial chaos neural network

Fig. 3 The comparison between the DNN and the \( p \)-order polynomial chaos neural network.

2) Auxiliary model \( \hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta) \)

Based on two properties of the adaptive aPC, Yao and Zheng et al. [25] proposed the Deep aPCE method which can construct an accurate surrogate model based on a small amount of labeled training data. Inspired by this idea, this paper uses a \( p \)-order adaptive PCE model to build the auxiliary model \( \hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta) \), i.e.,

\[
\hat{y}^{(p)} = \sum_{i=1}^{M} \hat{c}_i(\xi; \theta) \hat{D}_i(\xi). \tag{14}
\]

As shown in Fig. 2, the auxiliary model \( \hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta) \) uses a DNN with \( L \) hidden layers to solve the expansion coefficients \( \{\hat{c}_1(\xi; \theta), \hat{c}_2(\xi; \theta), \ldots, \hat{c}_M(\xi; \theta)\} \). Therefore, the parameters of the auxiliary model \( \hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta) \) are the parameters \( \theta = \{W, b\} \) of DNN.

Finally, the Deep PCNN model can be constructed based on the main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; e) \) and the auxiliary model \( \hat{y}^{(p)} = \mathcal{A}^{(p)}(\xi; \theta) \), where the expansion order of the former is larger than that of the latter, i.e., \( p > \beta \). It is noteworthy that \( \hat{y}^{(p)} \) is the final output of the Deep PCNN model in prediction stage.

3.3. Loss function for training Deep PCNN

To train the Deep PCNN model, this paper prepares two kinds of training datasets, i.e.,

- The labeled training dataset \( \mathcal{D}_{ld} = \{(\xi^{ld}_l, y^{ld}_l) | l = 1, 2, \ldots, N_{ld}\} \);
- The unlabeled training dataset \( \mathcal{D}_{ul} = \{\xi^{ul}_{l'} | l' = 1, 2, \ldots, N_{ul}\} \).
For the parameterized consistency learning-based Deep PCNN method, the auxiliary model $\tilde{y}^{(p)} = A^{(p)}(\xi; \theta)$ is used to assist in training the main model $\hat{y}^{(p)} = M^{(p)}(\xi; \epsilon)$ based on the proposed loss function $\mathcal{L}(\epsilon, \theta)$, i.e.,

$$\mathcal{L}(\epsilon, \theta) = \mathcal{L}_M(\epsilon) + \mathcal{L}_A(\theta),$$

(15)

where $\mathcal{L}_M(\epsilon)$ and $\mathcal{L}_A(\theta)$ consist of the supervised and unsupervised loss functions as shown in Fig. 2. The calculations of two loss functions $\mathcal{L}_M(\epsilon)$ and $\mathcal{L}_A(\theta)$ are presented as follows.

1) Loss function $\mathcal{L}_A(\theta)$

The corresponding prediction $\tilde{y}^{pd}_i$ of the input $\xi^{pd}_i$ ($i = 1, 2, \ldots, N_{pd}$) is estimated by the auxiliary model, i.e.

$$\tilde{y}^{pd}_i = A^{(p)}(\xi^{pd}_i; \theta).$$

Thus, the supervised loss function $\mathcal{L}_A(\theta)$ is calculated by the mean absolute error between the ground truth values $\{y^{pd}_i | i = 1, 2, \ldots, N_{pd}\}$ and the predictions $\{\tilde{y}^{pd}_i | i = 1, 2, \ldots, N_{pd}\}$, i.e.,

$$\mathcal{L}_A(\theta) = \frac{1}{N_{pd}} \sum_{i=1}^{N_{pd}} \|\tilde{y}^{pd}_i - \tilde{y}^{pd}_i\|.$$  

(16)

For $N_{ue}$ unlabeled inputs $\{\xi^{ue}_i | i = 1, 2, \ldots, N_{ue}\}$, the corresponding predictions $\{\tilde{y}^{ue}_i | i = 1, 2, \ldots, N_{ue}\}$ and the expansion coefficients $\{\tilde{c}_i(\xi^{ue}_i; \theta) | i = 1, 2, \ldots, N_{ue}\}$, where $\tilde{y}^{ue}_i = A^{(p)}(\xi^{ue}_i; \theta)$ and $\tilde{c}_i(\xi^{ue}_i; \theta) = \{\tilde{c}_i(\xi^{ue}_i; \theta) | i = 1, 2, \ldots, \tilde{M}\}$. Thus, the mean $E[A^{(p)}(\xi^{ue}_i; \theta)]$ and the variance $D[A^{(p)}(\xi^{ue}_i; \theta)]$ of the predictions $\{\tilde{y}^{ue}_i | i = 1, 2, \ldots, N_{ue}\}$ are respectively obtained by

$$E[A^{(p)}(\xi^{ue}_i; \theta)] = \frac{1}{N_{ue}} \sum_{i=1}^{N_{ue}} \tilde{y}^{ue}_i,$$

$$D[A^{(p)}(\xi^{ue}_i; \theta)] = \frac{1}{N_{ue}} \sum_{i=1}^{N_{ue}} (\tilde{y}^{ue}_i - E[A^{(p)}(\xi^{ue}_i; \theta)])^2.$$

(17)

Besides, the mean $E[\tilde{c}_i(\xi^{ue}_i; \theta)]$ of the $i$-th ($i = 1, 2, \ldots, \tilde{M}$) expansion coefficient $\tilde{c}_i(\xi^{ue}_i; \theta)$ is

$$E[\tilde{c}_i(\xi^{ue}_i; \theta)] = \frac{1}{N_{ue}} \sum_{i=1}^{N_{ue}} \tilde{c}_i(\xi^{ue}_i; \theta).$$

(18)

According to two properties of expansion coefficients in the reference [25], the unsupervised loss functions $\mathcal{L}_p(\theta)$ and $\mathcal{L}_c(\theta)$ are
In summary, the loss function \( L_A(\theta) \) is calculated based on Eq.(16) and Eq.(19), i.e.,

\[
L_A(\theta) = L_{m}^{(p)}(\theta) + L_{m}^{(s)}(\theta) + L_{m}^{(c)}(\theta).
\]

In the parameterized consistency learning-based Deep PCNN method, the loss function \( L_A(\theta) \) is used to learn the parameters \( \theta = \{W, b\} \) of the auxiliary model \( \mathcal{A}^{(p)}(\xi; \theta) \).

2) Loss function \( L_M(c) \)

For the input \( \xi_{ld}^{(p)} (l = 1, 2, \ldots, N_{ld}) \), the corresponding prediction \( \hat{y}_{ld}^{(p)} \) is predicted by the main model, i.e., \( \hat{y}_{ld}^{(p)} = \mathcal{M}^{(p)}(\xi_{ld}^{(p)}; c) \).

Thus, the supervised loss function \( L_{m}^{(s)}(c) \) is calculated by the mean absolute error between the ground truth values \( \{ y_{ld}^{(p)} | l = 1, 2, \ldots, N_{ld} \} \) and the predictions \( \{ \hat{y}_{ld}^{(p)} | l = 1, 2, \ldots, N_{ld} \} \), i.e.,

\[
L_{m}^{(s)}(c) = \frac{1}{N_{ld}} \sum_{l=1}^{N_{ld}} \| y_{ld}^{(p)} - \hat{y}_{ld}^{(p)} \|_1
= \frac{1}{N_{ld}} \sum_{l=1}^{N_{ld}} \| y_{ld}^{(p)} - \mathcal{M}^{(p)}(\xi_{ld}^{(p)}; c) \|_1.
\]

Besides, the corresponding predictions \( \{ \hat{y}_{ld}^{(c)} | l = 1, 2, \ldots, N_{ld} \} \) of the unlabeled inputs \( \{ \xi_{ld}^{(c)} | l = 1, 2, \ldots, N_{ld} \} \) are obtained by the main model, where \( \hat{y}_{ld}^{(c)} = \mathcal{M}^{(c)}(\xi_{ld}^{(c)}; c) \). In the parameterized consistency learning-based Deep PCNN method, the value \( \hat{y}_{ld}^{(c)} \) predicted by the auxiliary model is regarded as the pseudo label for the input \( \xi_{ld}^{(c)} \). Therefore, the unsupervised consistency loss function \( L_{m}^{(c)}(c) \) is calculated by the mean absolute error between the pseudo labels \( \{ \hat{y}_{ld}^{(c)} | l = 1, 2, \ldots, N_{ld} \} \) and the prediction \( \{ \hat{y}_{ld}^{(c)} | l = 1, 2, \ldots, N_{ld} \} \), i.e.,

\[
L_{m}^{(c)}(c) = \frac{1}{N_{ld}} \sum_{l=1}^{N_{ld}} \| \hat{y}_{ld}^{(c)} - \hat{y}_{ld}^{(c)} \|_1
= \frac{1}{N_{ld}} \sum_{l=1}^{N_{ld}} \| \hat{y}_{ld}^{(c)} - \mathcal{M}^{(c)}(\xi_{ld}^{(c)}; c) \|_1.
\]

In summary, the loss function \( L_M(c) \) is calculated based on Eq.(21) and Eq.(22), i.e.,

\[
L_M(c) = L_{m}^{(s)}(c) + L_{m}^{(c)}(c).
\]
In the parameterized consistency learning-based Deep PCNN method, the loss function \( \mathcal{L}_c (\mathbf{c}) \) is used to learn the parameterized expansion coefficients \( \mathbf{c} = \{c_1, c_2, \ldots, c_M\} \) of the main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)} (\xi; \mathbf{c}) \).

3.4. Parameterized consistency learning algorithm

This section proposes a parameterized consistency learning algorithm to learning the parameters \( (\mathbf{c}, \theta) \) of the Deep PCNN model based on the proposed loss function \( \mathcal{L}(\mathbf{c}, \theta) \). Before training the Deep PCNN model, the parameters \( (\mathbf{c}, \theta) \) need to be initialized firstly. To accelerate the convergence of the main model \( \mathcal{M}^{(p)} (\xi; \mathbf{c}) \), this section proposes many strategies to initialize the parameter \( \mathbf{c} = \{c_1, c_2, \ldots, c_M\} \) as follows:

1) Strategies for initializing the parameter \( c_1 \)

**Strategy I.** According to Eq. (6), the first expansion coefficient \( c_1 \) is always a constant for the PCE model (Eq.(1)) with different orders \( (p = 2, 3, 4, \ldots) \). Besides, the low-order PCE model requires less labeled data to calculate the expansion coefficients than the high-order PCE model. Thus, a \( p_{\text{low}} \)-order PCE model is constructed based on the labeled training dataset \( \mathcal{D}_{\text{ld}} = \{ (\xi_l^{pd}, y_l^{pd}) | l = 1, 2, \ldots, N_{\text{ld}} \} \). Supposed that the first expansion coefficient of the \( p_{\text{low}} \)-order PCE model is denoted as \( c_1^{\text{low}} \). Then, the parameter \( c_1 \) is initialized by the expansion coefficient \( c_1^{\text{low}} \), i.e.,

\[
c_1 = c_1^{\text{low}}.
\]

(24)

**Strategy II.** Based on Eq.(6), the value of the parameter \( c_1 \) is initialized to be the mean of labeled training dataset

\[
\mathcal{D}_{\text{ld}} = \{ (\xi_l^{pd}, y_l^{pd}) | l = 1, 2, \ldots, N_{\text{ld}} \}, \text{i.e.,}
\]

\[
c_1 = \frac{1}{N_{\text{ld}}} \sum_{l=1}^{N_{\text{ld}}} y_l^{pd}.
\]

(25)

In general, **Strategy I** is selected primarily. However, **Strategy II** is adopted to initialize the parameter \( c_1 \) when the \( p_{\text{low}} \)-order PCE model’s accuracy (e.g., R square [25]) is poor.

2) Strategies for initializing the parameters \( \{c_2, c_3, \ldots, c_M\} \)

**Strategy I.** For the \( p \)-order PCE model in Eq. (1), its expansion coefficients can be estimated roughly by the Galerkin projection method [12] or the regression method [13] based on the labeled training dataset \( \mathcal{D}_{\text{ld}} = \{ (\xi_l^{pd}, y_l^{pd}) | l = 1, 2, \ldots, N_{\text{ld}} \} \). Then, the parameters \( \{c_2, c_3, \ldots, c_M\} \) are initialized with the above roughly estimated coefficients.
**Strategy II.** For the labeled training dataset \( \mathcal{D}_{\text{st}} = \{(x_{st}^l, y_{st}^l) | l = 1, 2, \ldots, N_{\text{st}}\} \), the variance \( D(y_{st}^l) \) is

\[
D(y_{st}^l) = \frac{1}{N_{\text{st}}} \sum_{l=1}^{N_{\text{st}}} (y_{st}^l - \bar{y}_{st})^2 - \left( \frac{1}{N_{\text{st}}} \sum_{l=1}^{N_{\text{st}}} y_{st}^l \right)^2.
\]

(26)

For \( i = 2, 3, \ldots, M \), the following inequation can be obtained by Eq. (7), i.e.,

\[
(c_i^2) \leq D(y_{st}^l) \Rightarrow -\sqrt{D(y_{st}^l)} \leq c_i \leq \sqrt{D(y_{st}^l)}.
\]

(27)

Thus, the parameters \( \{c_2, c_3, \ldots, c_M\} \) are initialized by randomly sampling from the uniform distribution with the lower boundary \(-\sqrt{D(y_{st}^l)}\) and the upper boundary \(\sqrt{D(y_{st}^l)}\), i.e.,

\[
c_i \sim U(-\sqrt{D(y_{st}^l)}, \sqrt{D(y_{st}^l)}).
\]

(28)

Generally, **Strategy I** is used firstly. However, if the accuracy (e.g., R square [25]) of the \( p \)-order PCE model is poor, **Strategy II** is chosen to initialize the parameters \( \{c_2, c_3, \ldots, c_M\} \).

By the above strategies, the parameter \( \epsilon \) of the main model \( \mathcal{M}^{(p)}(\xi; e) \) can get the initial values that are relatively close to the final true expansion coefficients. During the training process, the labeled training dataset \( \mathcal{D}_{\text{st}} \) and the unlabeled training dataset \( \mathcal{D}_{\text{ls}} \) are input to the Deep PCNN model to get the corresponding predictions, based on which the loss function \( \mathcal{L}(\epsilon, \theta) \) is calculated by Eq. (15). Then, the gradients \( \partial \mathcal{L}^{(p)}(\epsilon)/\partial \epsilon, \partial \mathcal{L}^{(p)}(\epsilon)/\partial \theta, \partial \mathcal{L}^{(p)}(\theta)/\partial \theta, \partial \mathcal{L}^{(p)}(\epsilon)/\partial \theta \), and \( \partial \mathcal{L}^{(p)}(\theta)/\partial \theta \) are derived by the chain rule for differentiating compositions of functions using automatic differentiation [28]. In the parameterized consistency learning algorithm, the gradient propagation in the Deep PCNN model is shown in Fig. 4. In particular, the loss function \( \mathcal{L}_{\text{aux}}(\epsilon) \) includes the parameters \( \epsilon \) and \( \theta \) according to Eq. (22). Thus, the gradients \( \partial \mathcal{L}_{\text{aux}}(\epsilon)/\partial \epsilon \) and \( \partial \mathcal{L}_{\text{aux}}(\epsilon)/\partial \theta \) will be obtained in the automatic differentiation process. Since the auxiliary model \( \mathcal{A}^{(p)}(\xi; \theta) \) is used to assist the training of the main model \( \mathcal{M}^{(p)}(\xi; e) \), the gradient propagation of \( \partial \mathcal{L}^{(p)}(\epsilon)/\partial \theta \) in the auxiliary model \( \mathcal{A}^{(p)}(\xi; \theta) \) is blocked as shown in Fig. 4. Finally, the gradients \( \partial \mathcal{L}^{(p)}(\theta)/\partial \theta, \partial \mathcal{L}^{(p)}(\theta)/\partial \theta, \) and \( \partial \mathcal{L}^{(p)}(\theta)/\partial \theta \) are used to update the parameter \( \theta \) of the auxiliary model \( \mathcal{A}^{(p)}(\xi; \theta) \), and the gradients \( \partial \mathcal{L}^{(p)}(\epsilon)/\partial \epsilon \) and \( \partial \mathcal{L}^{(p)}(\epsilon)/\partial \epsilon \) are used to update the parameter \( \epsilon \) of the main model \( \mathcal{M}^{(p)}(\xi; e) \).

In this paper, \( e_{\text{p}_{\text{max}}} \) is assumed to be the maximum training epoch. For the \( ep_{\text{th}} \) (\( ep = 1, 2, \ldots, e_{\text{p}_{\text{max}}} \)) epoch, the Adam algorithm [29] is adopted to update the parameters \( \{\epsilon, \theta\} \) of the Deep PCNN model based on five calculated gradients.
\[ \frac{\partial L_{\xi}(c)/\partial c}{\partial L_{\xi}^i(c)/\partial c}, \frac{\partial L_{\xi}^i(\theta)/\partial \theta}{\partial L_{\xi}^i(\theta)/\partial \theta}, \text{ and } \frac{\partial L_{\xi}^i(\theta)/\partial \theta}{\partial L_{\xi}^i(\theta)/\partial \theta}. \] In summary, the pseudo code of the parameterized consistency learning algorithm is shown in the following Algorithm.

**Algorithm:** Parameterized consistency learning algorithm

**Input:**

1. Maximum training epoch \( ep_{\text{max}} \);
2. Labeled training dataset \( \mathcal{D}_{\text{tr}} = \{(x^i, y^i)|i = 1, 2, \cdots, N_{\text{tr}}\} \);
3. Unlabeled training dataset \( \mathcal{D}_{\text{ur}} = \{(x^i', y^i')|i' = 1, 2, \cdots, N_{\text{ur}}\} \).

**Output:**

Trained main model \( \mathcal{M}^{(p)}(\xi; c) \).

1. Choosing suitable strategies to initialize the parameter \( c \) of the main model \( \mathcal{M}^{(p)}(\xi; c) \);
2. for \( ep = 1: ep_{\text{max}} \) do
3. for \( l = 1: N_{\text{tr}} \) do
4. Using the main model to calculate the predicted value \( \hat{y}^l = \mathcal{M}^{(p)}(\xi; x^l) \);
5. Using the auxiliary model to calculate the predicted value \( \hat{y}_{A}^{l} = \mathcal{A}^{(p)}(\xi; \theta) \);
6. end
7. for \( l' = 1: N_{\text{ur}} \) do
8. Using the main model to calculate the predicted value \( \hat{y}^{l'}_{\text{tr}} = \mathcal{M}^{(p)}(\xi; c) \);
9. Using DNN of the auxiliary model to calculate the expansion coefficients \( \hat{e}_{\xi}^{l'}(\xi; \theta) = \{\hat{e}_{\xi}^{l'}(\xi; \theta)|i = 1, 2, \cdots, M\} \);
10. Using the auxiliary model to calculate the predicted value \( \hat{y}^{l'}_{\text{ur}} = \mathcal{A}^{(p)}(\xi; \theta) \)
11. end
12. Calculating the mean \( \bar{E}[\mathcal{A}^{(p)}(\xi; \theta)] = 1/N_{\text{tr}} \sum_{l=1}^{N_{\text{tr}}} \hat{y}^{l} \);
13. Calculating the variance \( \bar{D}[\mathcal{A}^{(p)}(\xi; \theta)] = \frac{1}{N_{\text{tr}}} \sum_{l=1}^{N_{\text{tr}}} (\hat{y}^{l} - \bar{E}[\mathcal{A}^{(p)}(\xi; \theta)])^2 \);
14. for \( l = 1: M \) do
15. Calculating the mean \( \bar{E}[\hat{e}_{\xi}^{l'}(\xi; \theta)] = 1/N_{\text{ur}} \sum_{l'=1}^{N_{\text{ur}}} \hat{e}_{\xi}^{l'}(\xi; \theta) \)
16. end
17. Calculating the loss function \( L(c, \theta) = L_{\text{tr}}(c) + L_{\text{ur}}(\theta) \);
18. Calculating the gradients \( \partial L_{\xi}(c)/\partial c, \partial L_{\xi}^i(c)/\partial c, \partial L_{\xi}^i(\theta)/\partial \theta, \partial L_{\xi}^i(\theta)/\partial \theta \), and \( \partial L_{\xi}^i(\theta)/\partial \theta \);
19. Using the Adam algorithm to update the parameters \( (c, \theta) \) of the Deep PCNN model;
20. end

### 3.5. Deep PCNN-based reliability analysis

According to section 3.4, the Deep PCNN model including can be trained by the proposed parameterized consistency learning algorithm. In particular, the trained main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; c) \) is the final predictive model. Based on the testing data \( \{(x^i, y^i)|i = 1, 2, \cdots, N_{\text{t}}\} \), the accuracy of the trained main model \( \hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; c) \) is evaluated by four error indicators, i.e.,

1) the root mean square error \( \text{RMSE} \)

\[
\text{RMSE} = \sqrt{\frac{1}{N_{\text{t}}} \sum_{i=1}^{N_{\text{t}}} [y_i^i - \mathcal{M}^{(p)}(\xi; c)]^2}, \tag{29}
\]
2) the mean absolute error $\text{MAE}$

$$\text{MAE} = \frac{1}{N_t} \sum_{t=1}^{N_t} |y^t_i - \mathcal{M}^{(p)}(\xi^t_i; c)|,$$  \hspace{1cm} (30)

3) the mean relative error $\text{MRE}$

$$\text{MRE} = \frac{1}{N_t} \sum_{t=1}^{N_t} \left| \frac{y^t_i - \mathcal{M}^{(p)}(\xi^t_i; c)}{y^t_i} \right|,$$  \hspace{1cm} (31)

4) the R square $R^2$

$$R^2 = 1 - \frac{\sum_{t=1}^{N_t} \left[ y^t_i - \mathcal{M}^{(p)}(\xi^t_i; c) \right]^2}{\sum_{t=1}^{N_t} \left[ y^t_i - \bar{y}^t \right]^2},$$  \hspace{1cm} (32)

where the mean $\bar{y}^t$ is calculated by

$$\bar{y}^t = \frac{1}{N_t} \sum_{t=1}^{N_t} \mathcal{M}^{(p)}(\xi^t_i; c).$$  \hspace{1cm} (33)

In this paper, the failure probability $P_f$ is calculated by performing MCS on the trained main model $\hat{y}^{(p)} = \mathcal{M}^{(p)}(\xi; c)$, i.e.,

$$P_f \approx \frac{1}{N_{\text{MCS}}} \sum_{k=1}^{N_{\text{MCS}}} \text{IF}\left[ \mathcal{M}^{(p)}(\xi_{\text{MCS}}; c) \right],$$  \hspace{1cm} (34)

where $N_{\text{MCS}}$ is the number of samples, and $\text{IF}(\cdot)$ is the indicator function. If $\mathcal{M}^{(p)}(\xi; c) < 0$, $\text{IF}\left[ \mathcal{M}^{(p)}(\xi_{\text{MCS}}; c) \right] = 1$, otherwise $\text{IF}\left[ \mathcal{M}^{(p)}(\xi_{\text{MCS}}; c) \right] = 0$. 

**Fig. 4** Schematic diagram of gradient propagation in the Deep PCNN model.
4. Numerical example

As shown in Fig. 5, this paper chooses the cantilever tube [30][31] to validate the effectiveness of the proposed parameterized consistency learning-based Deep PCNN method.

Fig. 5 Cantilever tube [30].

The performance function $g(x,z)$ is defined to be

$$g(x,z) = S_y - \sqrt{\sigma_x^2 + 3\tau_{xz}^2},$$

where $S_y$ is the yield strength, the normal stress $\sigma_x$ is

$$\sigma_x = \frac{P + F_1 \sin(\theta_1) + F_2 \sin(\theta_2) + Md}{A} \frac{2I}{},$$

and the torsional stress $\tau_{xz}$ is

$$\tau_{xz} = \frac{Td}{4I}.$$

In Eq. (36),

$$A = \frac{\pi}{4} \left[ d^2 - (d - 2t)^2 \right],$$

$$I = \frac{\pi}{64} \left[ d^4 - (d - 2t)^4 \right],$$

and the bending moment $M$ is calculated by

$$M = F_1 L_1 \cos(\theta_1) + F_2 L_2 \cos(\theta_2)$$

The random variables are shown in Table 2. In this paper, the parameters $\theta_1$ and $\theta_2$ are set to be 5 deg and 10 deg, respectively. If $g(x,z) \leq 0$, the cantilever tube fails.
Table 2 Nine random variables for the cantilever tube numerical example.

| Random variable | Mean (Lower boundary) | Standard deviation (Upper boundary) | Distribution |
|-----------------|-----------------------|-------------------------------------|--------------|
| $t$             | 5 mm                  | 0.1 mm                              | Normal       |
| $d$             | 42 mm                 | 0.5 mm                              | Normal       |
| $L_1$           | 119.75 mm             | 120.25 mm                           | Uniform      |
| $L_2$           | 59.75 mm              | 60.25 mm                            | Uniform      |
| $F_1$           | 3.0 kN                | 0.3 kN                              | Normal       |
| $F_2$           | 3.0 kN                | 0.3 kN                              | Normal       |
| $P$             | 12.0 kN               | 1.2 kN                              | Gumbel       |
| $T$             | 90.0 Nm               | 9.0 Nm                              | Normal       |
| $S_y$           | 220 MPa               | 22.0 MPa                            | Normal       |

**Constructing PCNN model**  In this numerical example, a 2-order adaptive PCE model is used to construct the auxiliary model, and the adaptive expansion coefficients are solved by a DNN with 55 outputs and five hidden layers, where the neuron numbers of five hidden layers are 32, 64, 128, 64, and 64, respectively. Besides, the activation function is the $ReLU(x)$. The main model is a 4-order, 5-order, or 6-order polynomial chaos neural network. For three polynomial chaos neural networks, the numbers of hidden layer neurons (parameterized expansion coefficients) three polynomial chaos neural networks are 715, 2002, and 5005, respectively. The dataset $D_{re}$ consists of $2 \times 10^5$ unlabeled training data, and the number $N_{pd}$ of labeled training data is 90, 200, and 400, respectively. The maximum training epoch $ep_{max}$ is set to be 20000.

**Results analysis**  Based on the UQLab-V2.0-104 [32], the cantilever tube example is also solved by the Ordinary Least-Squares PCE (OLS-PCE) method, the Orthogonal Matching Pursuit Sparse PCE (OMP-SPCE) method [21], and the Least Angle Regression Sparse PCE (LAR-SPCE) method [22]. The first four statistical moments and their relative errors are shown in Table 3, and the bold results indicate the best results. The results by the MCS method ($1 \times 10^6$ runs) are assumed to be the real results.

For $p = 4$, although four methods can correctly calculate the mean of the performance function value, the relative errors of three existing PCE methods on the standard deviation, the skewness, and the kurtosis are larger than the proposed Deep PCNN. Among three existing PCE methods, the relative error on the standard deviation of the OLS-PCE ($N_{pd} = 390$) is minimal (0.01587%). However, it is still larger than the proposed Deep PCNN (0.01178%). Besides, the minimal relative errors on the skewness and the kurtosis is the OMP-SPCE ($N_{pd} = 590$), i.e., 8.2132% on the skewness and 0.004423% on the kurtosis, while the relative error on the skewness and the kurtosis of the proposed Deep PCNN ($N_{pd} = 90$) are only 2.7507% and 0.003104%, respectively.
respectively. For \( p = 5, 6 \), the relative errors on the standard deviation, the skewness, and the kurtosis of the proposed Deep PCNN are still minimum in all methods. It is noteworthy that the proposed Deep PCNN uses the fewest labeled training data.

Table 3 The first four statistical moments and their relative errors in the cantilever tube numerical example.

| \( P \) | Method     | \( N_{sp} \) | Mean      | S.D.      | Skewness (R.E.) | Kurtosis (R.E.) |
|--------|------------|--------------|-----------|-----------|----------------|-----------------|
|       | MCS        | \( 1\times10^5 \) | 85.78     | 23.9463   | -0.005410      | 2.9976          |
| 4     | OLS-PCE    | 390          | 85.78     | 23.9425   | (0.01587%)     | 2.9962 (0.049763%) |
|       | OMP-SPCE   | 590          | 85.78     | 23.9526   | (0.02630%)     | 2.9975 (0.004423%) |
|       | LAR-SPCE   | 690          | 85.78     | 23.9519   | (0.02356%)     | 2.9971 (0.018136%) |
|       | Deep PCNN  | 90           | 85.78     | 23.9434   | (0.01178%)     | 2.9976 (0.003104%) |
| 5     | OLS-PCE    | 900          | 85.78     | 23.9510   | (0.01983%)     | 2.9978 (0.006856%) |
|       | OMP-SPCE   | 2500         | 85.78     | 23.9418   | (0.01857%)     | 2.9975 (0.004170%) |
|       | LAR-SPCE   | 3400         | 85.78     | 23.9344   | (0.04944%)     | 2.9971 (0.018618%) |
|       | Deep PCNN  | 200          | 85.78     | 23.9429   | (0.01421%)     | 2.9976 (0.000527%) |
| 6     | OLS-PCE    | 2060         | 85.78     | 23.9500   | (0.01546%)     | 2.9985 (0.029554%) |
|       | OMP-SPCE   | 3600         | 85.78     | 23.9751   | (0.02090%)     | 2.9975 (0.004362%) |
|       | LAR-SPCE   | 6000         | 85.78     | 23.9406   | (0.02349%)     | 2.9971 (0.018373%) |
|       | Deep PCNN  | 400          | 85.78     | 23.9441   | (0.00921%)     | 2.9976 (0.000493%) |

S.D. = Standard deviation \( R.E. = \) Relative error

To further validate the accuracy of the proposed Deep PCNN, four error indicators, the failure probability \( P_f \) and its relative estimation error rate \( \varepsilon_{P_f} \) are also calculated as shown in Table 4. For \( p = 4 \), the minimum \( RMSE, MAE, \) and \( MRE \) in three existing PCE methods are 0.1588, 0.0756, and \( 0.4045\times10^{-5} \), respectively. However, the \( RMSE, MAE, \) and \( MRE \) of the proposed Deep PCNN are only 0.0438, 0.0287, and \( 0.4045\times10^{-5} \), respectively. Apparently, the \( RMSE, MAE, \) and \( MRE \) of the proposed Deep PCNN are smaller than three existing PCE methods. The maximum \( R^2 \) of three existing PCE methods is 0.99995602, while the proposed Deep PCNN is 0.99999665. Besides, the estimated failure probability \( P_f \) of the proposed Deep PCNN is \( 1.840\times10^{-4} \), which is the closest value to the MCS result \( 1.850\times10^{-4} \). Compared with the failure probability estimated by MCS, the relative estimation error rates of the failure probability \( P_f \) for four methods are 5.4054\% (LAR-SPCE), 4.8649\% (OLS-PCE), 2.7027\% (OMP-SPCE), 0.5405\% (Deep PCNN), respectively. Thus, the proposed Deep PCNN can provide the most accurate estimation of the failure probability \( P_f \) in all methods. For \( p = 5, 6 \), the \( RMSE, MAE, \) and \( MRE \) of the proposed Deep PCNN are still minimum in all methods, and the proposed Deep PCNN still has the largest \( R^2 \). In particular, the 6-order Deep PCNN can estimate the failure probability \( P_f \) without any relative estimation error.
Table 4 Four error indicators and the estimated failure probability in the cantilever tube numerical example.

| $p$ | Method      | $N_{\text{gd}}$ | RMSE    | MAE    | MRE ($\times 10^{-3}$) | $R^2$     | $P_f$ ($\times 10^{-4}$) | $\sigma_p$ |
|-----|-------------|-----------------|---------|--------|-------------------------|-----------|--------------------------|------------|
| -   | MCS         | $1 \times 10^4$ | -       | -      | -                       | -         | 1.850                    | -          |
| 4   | OLS-PCE     | 390             | 0.1676  | 0.0756 | 1.0159                  | 0.99995103| 1.760                    | 4.8649%    |
|     | OMP-SPCE    | 590             | 0.1588  | 0.1248 | 1.5977                  | 0.99995602| 1.800                    | 2.7027%    |
|     | LAR-SPCE    | 690             | 0.3729  | 0.2782 | 3.6541                  | 0.99975751| 1.750                    | 5.4054%    |
|     | Deep PCNN   | 90              | 0.0438  | 0.0287 | 0.4045                  | 0.99999665| 1.840                    | 0.5405%    |
| 5   | OLS-PCE     | 900             | 0.3234  | 0.0817 | 1.1331                  | 0.99981755| 1.800                    | 2.7027%    |
|     | OMP-SPCE    | 2500            | 0.1584  | 0.1242 | 1.6020                  | 0.99995623| 1.790                    | 3.2432%    |
|     | LAR-SPCE    | 3400            | 0.3696  | 0.2769 | 3.6325                  | 0.99976172| 1.740                    | 5.9459%    |
|     | Deep PCNN   | 200             | 0.0139  | 0.0089 | 0.1382                  | 0.99999666| 1.840                    | 0.5405%    |
| 6   | OLS-PCE     | 2060            | 0.2788  | 0.0498 | 0.6819                  | 0.99986445| 1.810                    | 2.1622%    |
|     | OMP-SPCE    | 3600            | 0.1581  | 0.1242 | 1.5907                  | 0.99995639| 1.800                    | 2.7027%    |
|     | LAR-SPCE    | 6000            | 0.3696  | 0.2780 | 3.6419                  | 0.99976182| 1.740                    | 5.9459%    |
|     | Deep PCNN   | 400             | 0.0249  | 0.0067 | 0.1057                  | 0.99999892| 1.850                    | 0.0000%    |

The absolute error boxplots of four methods are shown in Fig. 6. According to Fig. 6, the medians and the interquartile ranges (IQRs) of the LAR-SPCE and the OMP-SPCE are always larger than the OLS-PCE and the Deep PCNN. It means that the overall accuracies of the latter two methods are higher than the former two methods. Compared with the OLS-PCE and the Deep PCNN, the former two methods perform the orthogonal basis sparse operation. Thus, the orthogonal basis sparse operation of the LAR-SPCE and the OMP-SPCE affect the accuracy of the surrogate model. Besides, the medians and the IQRs of the OLS-PCE and the Deep PCNN decrease gradually as the order increases, but the medians and the IQRs of the proposed Deep PCNN are always far less than the OLS-PCE. What’s more, the proposed Deep PCNN always adopts less labeled training data than the OLS-PCE. Especially, the OLS-PCE ($N_{\text{gd}} = 2060$) uses more than five times as much labeled training data as the proposed Deep PCNN ($N_{\text{gd}} = 400$) for $p = 6$. Therefore, the proposed Deep PCNN can construct a more accurate surrogate model than the OLS-PCE with less labeled training data.

![Fig. 6 The absolute error boxplots of four methods under different orders in the cantilever tube numerical example.](image)

19
By the result analyses for Table 3, Table 4, and Fig. 6, the proposed Deep PCNN can use less labeled training data to build a more accurate surrogate model without any complex orthogonal basis sparse operation than the existing PCE methods. Thus, the proposed Deep PCNN can solve the existing PCE methods’ shortcoming, i.e., needing to increase the expansion order to improve the accuracy of the surrogate model but causing more labeled data to solve the expansion coefficients.

5. Engineering Applications

This section uses two engineering cases to validate the effectiveness of the proposed parameterized consistency learning-based Deep PCNN method. Two applications use the 2-order ($\tilde{p} = 2$) adaptive PCE model to construct the auxiliary model $\bar{y}^{(2)} = A^{(2)}(\xi; \theta)$. Besides, the values of random input variables are sampled by the Latin Hypercube Sampling. The relevant codes by Python are available on this website\(^1\). The OLS-PCE method, the OMP-SPCE method [21], the LAR-SPCE method [22], and the Subspace Pursuit Sparse PCE (SP-SPCE) [23] are achieved by the UQLab-V2.0-104 [32].

5.1. Application 1: Reliability analysis for microsatellite conceptual design

5.1.1. Engineering background of application 1

"Tiantuo-1” satellite is a microsatellite that integrates basic functional components on a single circuit board, as shown in Fig. 7. Combined with the development experience of the "Tiantuo-1” satellite, the microsatellite generally uses many new technologies and commercial components to reduce its volume and mass, which greatly increases the uncertainty of the system. Besides, the design and production cycle of microsatellites is short, and the ground test procedure is simplified. Thus, it is essential to analyze the system reliability in the microsatellite conceptual design stage.

Since the mass is directly and positively related to satellite cost, the microsatellite mass is one of the design indexes to consider in the conceptual design stage. Generally, the microsatellite mass is influenced by the following seven parameters [33], i.e., the

\(^1\) https://github.com/Xiaohu-Zheng/Deep-Polynomial-Chaos-Neural-Network-Method
sun-synchronous circle orbit height $h_o$, the charge-coupled device (CCD) camera focal length $f_c$, the cross profile width (vertical flight direction) $b_{sat}$, the satellite height $l_{sat}$, the satellite wall thickness $t_{sat}$. The Application 1 studies a conceptual microsatellite with the following design scheme, i.e., $h_o = 600\, \text{km}$, $f_c = 280\, \text{mm}$, $b_{sat} = 800\, \text{mm}$, $l_{sat} = 700\, \text{mm}$, and $t_{sat} = 5\, \text{mm}$.

For this conceptual microsatellite, if the microsatellite mass is more than 183 kg, the design scheme does not satisfy the design requirements. Due to many factors, such as the earth’s oblateness and the influence of the low-orbit atmosphere, there is a certain drift in the orbital height. On account of the effect of machining accuracy, there is also a slight random deviation in the focal length of the CCD camera. Therefore, the parameters $h_o$ and $f_c$ are modeled to be the normal random variables [34]. Besides, the parameters $b_{sat}$, $l_{sat}$, and $t_{sat}$ are the normal random variables [34] by reason of the machining errors in the structure. In addition to the above five parameters, the data handling (DH) subsystem quality estimation coefficient $c_{DH}$, and the telemetry, tracking and command subsystem (TTC) quality estimation coefficient $c_{TTC}$ also affect the microsatellite mass. Because the designers have insufficient information about the DH and TTC subsystems, the values of the coefficients $c_{DH}$ and $c_{TTC}$ cannot be obtained exactly. The above seven parameters’ statistical properties are shown in Table 5.

**Table 5** Seven random variables that effect the conceptual microsatellite mass.

| Random variable | Mean (Lower boundary) | Standard deviation (Upper boundary) | Distribution |
|-----------------|-----------------------|-------------------------------------|--------------|
| $h_o$           | 600 km                | 6 km                                | Normal       |
| $f_c$           | 280 mm                | 1 mm                                | Normal       |
| $b_{sat}$       | 800 mm                | 10 mm                               | Normal       |
| $l_{sat}$       | 700 mm                | 10 mm                               | Normal       |
| $t_{sat}$       | 5 mm                  | 0.1 mm                              | Normal       |
| $c_{DH}$        | 0.04                  | 0.05                                | Uniform      |
| $c_{TTC}$       | 0.05                  | 0.06                                | Uniform      |

5.1.2. Constructing Deep PCNN model and preparing training data for application 1

In the application 1, the adaptive expansion coefficients of the auxiliary model are solved by a DNN with 36 outputs and five hidden layers, where the neuron numbers of five hidden layers are 64, 128, 128, 128, and 128, respectively. The DNN adopts the activation function $\text{ReLU}(x)$. Besides, the main model is a 5-order, 6-order, 7-order, or 8-order polynomial chaos neural network. For four polynomial chaos neural networks, the numbers of hidden layer neurons (parameterized expansion coefficients) three polynomial chaos neural networks are 792, 1716, 3432, and 6435, respectively. The dataset $\mathcal{D}_{\alpha}$ consists of $2 \times 10^5$ unlabeled training data, and the number $N_{rd}$ of labeled training data is 50, 70, 90, and 150, respectively. The maximum training epoch $ep_{max}$ is set to be 20000.
5.1.3 Result analysis and discussion of application 1

By the trained Deep PCNN model, the first four statistical moments and their relative errors are shown in Table 6, and the bold results indicate the best results. Besides, this engineering problem is also solved by the OLS-PCE, OMP-SPCE, and LAR-SPCE. The results by the MCS method ($5 \times 10^5$ runs) are assumed to be the real results.

| $p$ | Method       | $N_{\text{tr}}$ | Mean | S.D. | Skewness (R.E.) | Kurtosis (R.E.) |
|-----|--------------|------------------|------|------|-----------------|-----------------|
|    | MCS          | $5 \times 10^5$  | 176.24 | 2.52 | 0.0464          | 3.0034          |
| 5   | OLS-PCE      | 1000             | 176.24 | 2.52 | 0.0498 (7.1637%)| 3.0092 (0.1933%)|
|     | OMP-SPCE     | 500              | 176.24 | 2.52 | 0.0435 (6.6356%)| 3.0052 (0.0588%)|
|     | LAR-SPCE     | 200              | 176.24 | 2.52 | 0.0385 (17.1328%)| 3.0063 (0.0968%)|
|     | Deep PCNN    | 50               | 176.24 | 2.52 | **0.0458 (1.3374%)** | 3.0025 (0.0297%) |
| 6   | OLS-PCE      | 3500             | 176.24 | 2.52 | 0.0503 (8.0721%)| 3.0076 (0.1387%)|
|     | OMP-SPCE     | 1200             | 176.24 | 2.52 | 0.0387 (16.7120%)| 2.9996 (0.1268%)|
|     | LAR-SPCE     | 340              | 176.24 | 2.52 | 0.0407 (12.3542%)| 3.0020 (0.0473%)|
|     | Deep PCNN    | 70               | 176.24 | 2.52 | **0.0499 (7.380%)** | 3.0029 (0.0173%) |
| 7   | OLS-PCE      | 6000             | 176.24 | 2.52 | 0.0423 (8.880%) | 3.0229 (0.6507%)|
|     | OMP-SPCE     | 2800             | 176.24 | 2.52 | 0.0391 (15.7727%)| 3.0010 (0.0804%)|
|     | LAR-SPCE     | 1100             | 176.24 | 2.52 | 0.0388 (16.3974%)| 3.0014 (0.0665%)|
|     | Deep PCNN    | 90               | 176.24 | 2.52 | **0.0497 (7.1240%)** | 3.0031 (0.0093%) |
| 8   | OLS-PCE      | 15000            | 176.24 | 2.52 | 0.0448 (3.6559%) | 3.0024 (0.0317%)|
|     | OMP-SPCE     | 6000             | 176.24 | 2.52 | 0.0425 (8.4810%) | 3.0013 (0.0681%)|
|     | LAR-SPCE     | 2000             | 176.24 | 2.52 | 0.0447 (3.6974%) | 3.0024 (0.0343%)|
|     | Deep PCNN    | 150              | 176.24 | 2.52 | **0.0450 (3.0250%)** | 3.0026 (0.0269%) |

For $p = 5$, although four methods can correctly calculate the mean and the standard deviation of the performance function value, the relative errors of three existing PCE methods on the skewness and the kurtosis are larger than the proposed Deep PCNN. Among three existing PCE methods, the OMP-SPCE ($N_{\text{tr}} = 500$) has the minimal relative errors on the skewness (6.6356%) and the kurtosis (0.0588%), while the relative error on the skewness and the kurtosis of the proposed Deep PCNN ($N_{\text{tr}} = 90$) are only 1.3374% and 0.0297%, respectively. Besides, the LAR-SPCE uses the minimum labeled training data in three existing PCE methods, i.e., $N_{\text{tr}} = 200$. However, the proposed Deep PCNN only uses 90 labeled training data. What’s more, the relative errors on the skewness (17.1328%) and the kurtosis (0.0968%) of the LAR-SPCE are far larger than the proposed Deep PCNN. For $p = 6, 7, 8$, the relative errors on the skewness and the kurtosis of the proposed Deep PCNN are always minimum in all methods. It is noteworthy that the proposed Deep PCNN always uses the fewest labeled training data.
To further validate the accuracy of the proposed Deep PCNN, four error indicators, the failure probability $P_f$ and its relative estimation error rate $\varepsilon_{p_f}$ are also calculated as shown in Table 7. For $p = 5$, the minimum $RMSE, MAE,$ and $MRE$ in three existing PCE methods are 0.0227, 0.0169, and $0.9567 \times 10^{-3}$, respectively. However, the $RMSE, MAE,$ and $MRE$ of the proposed Deep PCNN are only 0.0101, 0.0079, and $0.4511 \times 10^{-3}$, respectively. Apparently, the $RMSE, MAE,$ and $MRE$ of the proposed Deep PCNN are smaller than three existing PCE methods. The maximum $R^2$ of three existing PCE methods is 0.999919, while the proposed Deep PCNN is 0.999984. Compared with the failure probability estimated by MCS, the relative estimation error rates of the failure probability $P_f$ for four methods are 3.1579% (LAR-SPCE), 0.5742% (OLS-PCE), 0.3349% (Deep PCNN), 0.0957% (OMP-SPCE). Although the $\varepsilon_{p_f}$ of the proposed Deep PCNN is slightly more than the OMP-SPCE, the OMP-SPCE uses ten times as much labeled training data as the proposed Deep PCNN. Besides, the OMP-SPCE needs the complex orthogonal basis sparse operation while the proposed Deep PCNN does not. Thus, the proposed Deep PCNN can use fewer labeled training data to obtain the failure probability $P_f$ close to the results of MCS.

Table 7 Four error indicators and the estimated failure probability in the Application 1.

| $p$ | Method   | $N_{pt}$ | $RMSE$ | $MAE$ | $MRE \times 10^4$ | $R^2$ | $P_f \times 10^{-3}$ | $\varepsilon_{p_f}$  |
|-----|----------|----------|--------|-------|-------------------|-------|---------------------|----------------------|
| 5   | MCS      | $5 \times 10^5$ | -      | -     | -                 | -     | 4.1800              | -                    |
|     | OLS-PCE  | 1000     | 0.0477 | 0.0228 | 1.2929           | 0.999641 | 4.2040              | 0.5742%              |
|     | OMP-SPCE | 500      | 0.0245 | 0.0180 | 1.0225           | 0.999905 | **4.1840**          | **0.0957%**           |
|     | LAR-SPCE | 200      | 0.0227 | 0.0169 | 0.9567           | 0.999919 | 4.0480              | 3.1579%              |
|     | Deep PCNN| 50       | **0.0101** | **0.0079** | **0.4511** | **0.999984** | 4.1660              | 0.3349%              |
| 6   | OLS-PCE  | 3500     | 0.0334 | 0.0142 | 0.8072           | 0.999824 | **4.2480**          | **1.6268%**           |
|     | OMP-SPCE | 1200     | 0.0232 | 0.0180 | 1.0196           | 0.999915 | 4.0960              | 2.0096%              |
|     | LAR-SPCE | 340      | 0.0208 | 0.0153 | 0.8668           | 0.999932 | 4.0880              | 2.2010%              |
|     | Deep PCNN| 70       | **0.0156** | **0.0117** | **0.6646** | **0.999916** | 4.2580              | 1.8660%              |
| 7   | OLS-PCE  | 6000     | 0.0710 | 0.0212 | 1.2055           | 0.999206 | 4.2180              | 0.9091%              |
|     | OMP-SPCE | 2800     | 0.0318 | 0.0248 | 1.4053           | 0.999841 | 4.1000              | 1.9139%              |
|     | LAR-SPCE | 1100     | 0.0193 | 0.0148 | 0.8392           | 0.999941 | 4.0860              | 2.2488%              |
|     | Deep PCNN| 90       | **0.0121** | **0.0093** | **0.5267** | **0.999977** | **4.1960**          | **0.3828%**           |
| 8   | OLS-PCE  | 15000    | 0.0150 | 0.0674 | 0.8535           | 0.999286 | 4.1280              | 1.2440%              |
|     | OMP-SPCE | 6000     | 0.0303 | 0.0236 | 1.3379           | 0.999855 | 4.1100              | 1.6746%              |
|     | LAR-SPCE | 2000     | **0.0109** | **0.0086** | **0.4907** | **0.999981** | **4.1660**          | **0.3349%**           |
|     | Deep PCNN| 150      | **0.0114** | **0.0083** | **0.4720** | 0.999979 | 4.1640              | 0.3828%              |

For $p = 6,7$, the $RMSE, MAE,$ and $MRE$ of the proposed Deep PCNN are still minimum in all methods, and the proposed Deep PCNN still has the largest $R^2$. For $p = 8$, the $RMSE, MAE, MRE, R^2$, and $\varepsilon_{p_f}$ of the proposed Deep PCNN ($N_{pt} = 150$) and the LAR-SPCE ($N_{pt} = 2000$) are very close. However, the LAR-SPCE adopts more than thirteen times as much labeled training data as the proposed Deep PCNN.
data as the proposed Deep PCNN. Based on the kernel density estimation (KDE), the estimated probability density functions (PDFs) of the satellite mass by the proposed \( p \)-order (\( p = 5, 6, 7, 8 \)) Deep PCNN and the MCS are shown in Fig. 8. The red curves (solid line) are the results of the proposed Deep PCNN, and the green curves (dash line) are the results of the MCS. Refer to Fig. 8, the PDF curves of the satellite mass by the proposed \( p \)-order (\( p = 5, 6, 7, 8 \)) Deep PCNN are very closer to the results of the MCS.

![PDF curves of the satellite mass by the proposed \( p \)-order Deep PCNN and the MCS](image)

**Fig. 8** The estimated probability density functions of the satellite mass by the proposed \( p \)-order (\( p = 5, 6, 7, 8 \)) Deep PCNN and the MCS.

In summary, the proposed Deep PCNN can use less labeled training data to build a more accurate surrogate model without any complex orthogonal basis sparse operation than the existing PCE methods for the Application 1. According to the estimated failure probability in Table 7, the conceptual design scheme of the studied microsatellite has high reliability.

### 5.2. Application 2: First-order frequency reliability analysis of “Tiantuo-3” satellite load-bearing structure

#### 5.2.1. Engineering background of application 2

As shown in Fig. 9 (a), “Tiantuo-3” satellite is a highly integrated microsatellite with low mass. For its load-bearing structure, including frame structure, support rod, and separating device in Fig. 9 (b), the strength design is important for developing the “Tiantuo-3” satellite. If the “Tiantuo-3” satellite and the launch vehicle resonate during the launch process, the former’s load-bearing structure will be broken, causing the satellite to fail. Thus, the first-order frequency of the load-bearing structure is usually more than 81.7925 Hz.

![“Tiantuo-3” satellite and its load-bearing structure](image)

**Fig. 9** “Tiantuo-3” satellite and its load-bearing structure.
For the “Tiantuo-3” satellite’s load-bearing structure, it is mainly determined by six physical parameters, i.e., the aluminum alloy density \( \rho_{AE} \), the spring steel density \( \rho_{SS} \), the titanium alloy density \( \rho_{TA} \), the aluminum elastic modulus \( E_{AE} \), the spring steel elastic modulus \( E_{SS} \), and the titanium alloy elastic modulus \( E_{TA} \). Due to uncertainty factors existing in the ambient temperature, material molding conditions, etc., the above six physical parameters are modeled as six random variables [25], as shown in Table 8.

Table 8 Six random variables involved in the “Tiantuo-3” satellite’s load-bearing structure

| Random variable | Mean          | Standard deviation      | Distribution |
|-----------------|---------------|-------------------------|--------------|
| \( \rho_{AE} \) | 2.69 g/cm\(^3\) | 8.97 \times 10^{-3} g/cm\(^3\) | Normal       |
| \( \rho_{SS} \) | 7.85 g/cm\(^3\) | 2.617 \times 10^{-2} g/cm\(^3\) | Normal       |
| \( \rho_{TA} \) | 4.43 g/cm\(^3\) | 1.477 \times 10^{-2} g/cm\(^3\) | Normal       |
| \( E_{AE} \)   | 6.89 \times 10^4 MPa | 2.29667 \times 10^4 MPa | Normal       |
| \( E_{SS} \)   | 2.00 \times 10^5 MPa | 2.00 \times 10^5 MPa | Normal       |
| \( E_{TA} \)   | 1.138 \times 10^5 MPa | 1.89667 \times 10^5 MPa | Normal       |

5.2.2. Constructing Deep PCNN model and preparing training data for application 2

In the application, the adaptive expansion coefficients of the auxiliary model are solved by a DNN with 28 outputs and five hidden layers, where the neuron numbers of five hidden layers are 32, 64, 128, 64, and 64, respectively. The DNN adopts the activation function \( \text{ReLU}(x) \). Besides, the main model is a 6-order, or 8-order polynomial chaos neural network. For two polynomial chaos neural networks, the numbers of hidden layer neurons (parameterized expansion coefficients) three polynomial chaos neural networks are 924 and 3003, respectively. The dataset \( \mathcal{D}_e \) consists of \( 4 \times 10^5 \) unlabeled training data, and the number \( N_{ld} \) of labeled training data is 60 and 70, respectively. The maximum training epoch \( ep_{max} \) is set to be 20000.

5.2.3. Result analysis and discussion of Application 2

The first four statistical moments and their relative errors are calculated by the trained Deep PCNN model, as shown in Table 9, and the bold results indicate the best results. Besides, this engineering problem is also solved by the SP-SPCE, OMP-SPCE, and LAR-SPCE. Due to the limited experimental data collected, this paper takes the results of 600 MCSs as the real results.

For \( p = 6 \), although four methods can correctly calculate the mean and the standard deviation of the first-order frequency, the relative errors of three existing PCE methods on the skewness and the kurtosis are larger than the proposed Deep PCNN. Among three existing PCE methods, the minimal relative errors on the skewness and the kurtosis are 1.0148\% and 0.0169\%, respectively. In contrast, the relative error on the skewness and the kurtosis of the proposed Deep PCNN (\( N_{ld} = 60 \)) are only 0.8673\% and 0.0042\%, respectively. Besides, the SP-SPCE uses the minimum labeled training data in three existing PCE methods, i.e.,...
\( N_{ad} = 100 \). However, the proposed Deep PCNN only uses 60 labeled training data. What’s more, the relative errors on the skewness (1.0798\%) and the kurtosis (0.0581\%) of the SP-SPCE are larger than the proposed Deep PCNN. For \( p = 8 \), the relative errors on the skewness and the kurtosis of the proposed Deep PCNN are also minimum in all methods. It is noteworthy that the proposed Deep PCNN still uses the fewest labeled training data.

**Table 9** The first four statistical moments and their relative errors in the Application 2.

| \( p \) | Method      | \( N_{ad} \) | Mean  | Standard deviation | Skewness (R.E.) | Kurtosis (R.E.) |
|--------|-------------|--------------|-------|--------------------|-----------------|-----------------|
| 6      | LAR-SPCE    | 140          | 82.836| 0.453              | -0.07239 (0.6078\%) | 2.6516 (0.0042\%) |
|        | OMP-SPCE    | 130          | 82.836| 0.453              | -0.073282 (1.0148\%) | 2.6522 (0.0184\%) |
|        | SP-SPCE     | 100          | 82.836| 0.453              | -0.073329 (1.0798\%) | 2.6533 (0.0581\%) |
|        | Deep PCNN   | 60           | 82.836| 0.453              | **-0.07319 (0.8673\%)** | **2.6516 (0.0042\%)** |
| 8      | LAR-SPCE    | 180          | 82.836| 0.453              | -0.07219 (0.4908\%) | 2.6515 (0.0929\%) |
|        | OMP-SPCE    | 170          | 82.836| 0.453              | -0.07216 (0.5366\%) | 2.6523 (0.0309\%) |
|        | SP-SPCE     | 140          | 82.836| 0.453              | -0.07162 (1.2782\%) | 2.6524 (0.0231\%) |
|        | Deep PCNN   | 70           | 82.836| 0.453              | **-0.07229 (0.3773\%)** | **2.6517 (0.0030\%)** |

To further validate the accuracy of the proposed Deep PCNN, four error indicators, the failure probability \( P_f \) and its relative estimation error rate \( \hat{\varepsilon}_f \) are also calculated as shown in Table 10. For \( p = 6 \), the minimum \( \text{RMSE}, \text{MAE}, \) and \( \text{MRE} \) in three existing PCE methods are \( 6.0706 \times 10^{-4}, 4.0844 \times 10^{-4}, \text{ and } 0.0493 \times 10^{-4} \), respectively. However, the \( \text{RMSE}, \text{MAE}, \) and \( \text{MRE} \) of the proposed Deep PCNN are only \( 5.4234 \times 10^{-4}, 3.7483 \times 10^{-4}, \text{ and } 0.0452 \times 10^{-4} \), respectively. Thus, the \( \text{RMSE}, \text{MAE}, \) and \( \text{MRE} \) of the proposed Deep PCNN are smaller than three existing PCE methods. The maximum \( R^2 = 0.9999986 \) of three existing PCE methods is still less than the \( R^2 = 0.9999986 \) of proposed Deep PCNN. Compared with the failure probability estimated by MCS, three existing PCE methods have a 20\% relative estimation error rate, while the proposed Deep PCNN can estimate the failure probability accurately. The labeled training data used by the proposed Deep PCNN is also less than three existing PCE methods. Besides, three existing PCE methods need the complex orthogonal basis sparse operation while the proposed Deep PCNN does not. For \( p = 8 \), the \( \text{RMSE}, \text{MAE}, \) and \( \text{MRE} \) of the proposed Deep PCNN are always minimum in all methods, and the proposed Deep PCNN still has the largest \( R^2 \). All methods can estimate the failure probability accurately.

In summary, the result analyses for Table 9 and Table 10 show that the proposed Deep PCNN can use less labeled training data to build a more accurate surrogate model without any complex orthogonal basis sparse operation than the existing PCE methods. To more accurately estimate the failure probability, \( 1 \times 10^6 \) MCSs are performed on the trained 8-order Deep PCNN model. The estimated failure probability is 0.008405. Therefore, the “Tiantuo-3” satellite load-bearing structure has high first-order frequency reliability.
Table 10 Four error indicators and the estimated failure probability in the Application 2.

| $p$ | Method     | $N_{ol}$ | $RMSE \times 10^{-4}$ | $MAE \times 10^{-4}$ | $MRE \times 10^{-4}$ | $R^2$       | $P_f$     | $\epsilon_{r_f}$ |
|-----|------------|----------|------------------------|-----------------------|----------------------|------------|-----------|-------------------|
| 6   | LAR-SPCE   | 140      | 7.3182                 | 0.0493                | 0.9999974            | 0.0067     | 0.99999    | 20.00%           |
|     | OMP-SPCE   | 130      | 6.0706                 | 0.0521                | 0.9999982            | 0.0067     | 0.99999    | 20.00%           |
|     | SP-SPCE    | 100      | 8.1117                 | 0.0554                | 0.9999968            | 0.0067     | 0.99999    | 20.00%           |
|     | Deep PCNN  | 60       | 5.4234                 | 0.0452                | 0.9999986            | 0.0083     | 0.99999    | 0.00%            |
| 8   | LAR-SPCE   | 180      | 8.3160                 | 0.0518                | 0.9999966            | 0.0083     | 0.99999    | 0.00%            |
|     | OMP-SPCE   | 170      | 6.7627                 | 0.0540                | 0.9999978            | 0.0083     | 0.99999    | 0.00%            |
|     | SP-SPCE    | 140      | 7.8291                 | 0.0535                | 0.9999970            | 0.0083     | 0.99999    | 0.00%            |
|     | Deep PCNN  | 70       | 5.9560                 | 0.0503                | 0.9999983            | 0.0083     | 0.99999    | 0.00%            |

6. Conclusions

This paper proposes a parameterized consistency learning-based deep polynomial chaos neural network (Deep PCNN) method for analyzing stochastic system reliability. The Deep PCNN model consists of two models, i.e., the auxiliary and the main models. Inspired by the Deep aPCE, the auxiliary model is constructed to be a low-order adaptive PCE model by a DNN, where the DNN solves the expansion coefficients of the low-order adaptive PCE model. The main model is a polynomial chaos neural network built by a high-order PCE model. In particular, the expansion coefficients of the high-order PCE model are parameterized into the learnable weights of the polynomial chaos neural network. The auxiliary model uses a proposed unsupervised consistency loss function to assist in training the main model. Based on a small amount of labeled training data and many unlabeled training data, the parameterized consistency learning algorithm is proposed to learn the parameters of the Deep PCNN model. After training the Deep PCNN model, the MCS can be straightforwardly performed on the main model to analyze the system reliability. This paper adopts a numerical example to validate the effectiveness of the Deep PCNN method, and the results show that the Deep PCNN method can significantly reduce the training data cost in constructing a high-order PCE model without losing surrogate model accuracy. Finally, the Deep PCNN method is applied to analyze the reliability of two aerospace engineering systems. Next, the authors will study how to reduce the amount of labeled training data in building the PCE models of the high dimensional stochastic systems in future research.

Acknowledgments

This work was supported by the Postgraduate Scientific Research Innovation Project of Hunan Province (No.CX20200006) and the National Natural Science Foundation of China (Nos.11725211).
References

[1] Yao W, Chen XQ, Luo WC, Tooren MV, Guo J. Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles. Progress in Aerospace Sciences 2011;47:450-79.

[2] Zheng XH, Yao W, Xu YC, Chen XQ. Improved compression inference algorithm for reliability analysis of complex multistate satellite system based on multilevel Bayesian network. Reliability Engineering & System Safety 2019;189:123-42.

[3] Zheng XH, Yao W, Xu YC, Chen XQ. Algorithms for Bayesian network modeling and reliability inference of complex multistate systems: Part I – Independent systems. Reliability Engineering & System Safety 2020;202:107011.

[4] Yao W, Tang GJ, Wang N, Chen XQ. An improved reliability analysis approach based on combined FORM and Beta-spherical importance sampling in critical region. Structural and Multidisciplinary Optimization 2019;60:35-58.

[5] Xiu DB, Karniadakis GE. Modeling uncertainty in flow simulations via generalized polynomial chaos. Journal of Computational Physics 2003;187(1):137-67.

[6] Oladyshkin S, Nowak W. Incomplete statistical information limits the utility of high-order polynomial chaos expansions. Reliability Engineering & System Safety 2018;169:137-48.

[7] Oladyshkin S, Nowak W. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. Reliability Engineering and System Safety 2012;106:179-90.

[8] Bao YQ, Xiang ZL, Li H. Adaptive subset searching-based deep neural network method for structural reliability analysis. Reliability Engineering & System Safety 2021;213:107778.

[9] Liu D, Wang SP, Cui XY. An artificial neural network supported Wiener process based reliability estimation method considering individual difference and measurement error. Reliability Engineering & System Safety 2022;218:108162.

[10] Wang J, Sun ZL, Cao RN. An efficient and robust Kriging-based method for system reliability analysis. Reliability Engineering & System Safety 2021;216:107953.

[11] Zhou T, Peng YB. Reliability analysis using adaptive Polynomial-Chaos Kriging and probability density evolution method. Reliability Engineering & System Safety 2022;220:108283.

[12] Ghanem R, Spanos PD. A stochastic Galerkin expansion for nonlinear random vibration analysis. Probabilistic Engineering Mathematics 1993;8(3-4):255-64.
[13] Berveiller M, Sudret B, Lemaire M. Stochastic finite element: a nonintrusive approach by regression. European Journal of Computational Mechanics 2006;15(1-3):81-92.

[14] Chen ZY, Zhou P, Liu Y, Ji PF. A novel approach to uncertainty analysis using methods of hybrid dimension reduction and improved maximum entropy. Structural and Multidisciplinary Optimization. 2019;60:1841-66.

[15] Liu RY, Fan WL, Wang YL, Ang AHS, Li ZL. Adaptive estimation for statistical moments of response based on the exact dimension reduction method in terms of vector. Mechanical Systems and Signal Processing. 2019;126:609-25.

[16] Schöbi R, Sudret B. Global sensitivity analysis in the context of imprecise probabilities (p-boxes) using sparse polynomial chaos expansions. Reliability Engineering & System Safety. 2019;187:129-41.

[17] Nagel JB, Rieckermann J, Sudret B. Principal component analysis and sparse polynomial chaos expansions for global sensitivity analysis and model calibration: Application to urban drainage simulation. Reliability Engineering & System Safety 2020;195.

[18] Zhou YC, Lu ZZ, Yun WY. Active sparse polynomial chaos expansion for system reliability analysis. Reliability Engineering & System Safety 2020;202.

[19] Fan WL, Wei JH, Ang AH, Li ZL. Adaptive estimation of statistical moments of the responses of random systems. Probabilistic Engineering Mechanics 2016; 43:50–67.

[20] Zhang Y, Xu J. Efficient reliability analysis with a CDA-based dimension-reduction model and polynomial chaos expansion. Computer Methods in Applied Mechanics and Engineering 2021;373.

[21] Pati YC, Rezaiifar R, Krishnaprasad PS. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. Proceedings of 27th Asilomar Conference on Signals, Systems and Computers 1993. p. 40-4.

[22] Blatman G, Sudret B. Adaptive sparse polynomial chaos expansion based on least angle regression. Journal of Computational Physics 2011;230:2345-67.

[23] Diaz P, Doostan A, Hampton J. Sparse polynomial chaos expansions via compressed sensing and D-optimal design. Computer Methods in Applied Mechanics and Engineering 2018;336:640-66.

[24] Zhang DK, Lu L, Guo L, Karniadakis GE. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. Journal of Computational Physics 2019;397:108850.
[25] Yao W, Zheng XH, Zhang J, Wang N, Tang GJ. Deep Adaptive Arbitrary Polynomial Chaos Expansion: A Mini-data-driven Semi-supervised Method for Uncertainty Quantification. Preprint.

[26] Meecham WC, Siegel A. Wiener-Hermite expansion in model turbulence at large Reynolds numbers. Physics of Fluids 1964;7(8):1178-90.

[27] Goodfellow I, Bengio Y, Courville A. Deep Learning. The MIT Press; 2016.

[28] Baydin AG, Pearlmutter BA, Radul AA. Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research 2018;18:1-43.

[29] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. Proceedings of the International Conference on Learning Representations; 2015 May 7-9; San Diego, USA. Ithaca: arXiv.org; 2015.

[30] Du XP. Unified Uncertainty Analysis by the First Order Reliability Method. Journal of Mechanical Design 2008;130.

[31] Yao W, Chen XQ, Huang YY, Tooren MV. An enhanced unified uncertainty analysis approach based on first order reliability method with single-level optimization. Reliability Engineering & System Safety 2013;116:28-37.

[32] Marelli, S, Lüthen, N, Sudret, B. UQLab user manual - Polynomial Chaos Expansions, Report UQLab-V2.0-104. Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich; 2022.

[33] Wertz JR, Larson WJ. Space Mission Analysis and Design. 3rd ed. California: Microcosm Press; 1999.

[34] Chen XQ, Yao W, Ouyang Q. Theory and application of uncertainty-based multidisciplinary design optimization for flight vehicles.1st ed. Beijing: Science Press; 2021. Chinese.