Composite Structures 367 (2025) 119236

Contents lists available at ScienceDirect

Composite Structures

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

# Neural networks based inverse design of inhomogeneous tetrachiral honeycombs for desired deformation

Linzhe Du [a], Jian Sun [a,*], Yanju Liu [b,*], Jinsong Leng [a]

[a] Center for Composite Materials and Structures, Harbin Institute of Technology, Harbin, 150080, China
[b] Department of Astronautical Science and Mechanics, Harbin Institute of Technology, Harbin, 150001, China

ARTICLE INFO

*Keywords:*
Tetrachiral honeycomb
Neural network
Inverse design

ABSTRACT

The challenge of achieving efficient inverse design for honeycomb structures with desired deformations has persisted. To address this, a machine learning framework including two neural networks is introduced, with one used for sensitivity analysis and dataset generation, while the other for inverse design. A tetrachiral honeycomb structure is parametrically modeled using Python scripts and subsequently analyzed with finite element method (FEM) software. A dataset mapping unit cell parameters to honeycomb deformations is fabricated by FEM for training a forward neural network, which has an R-squared value of 0.9680. Based on this trained neural network, four high sensitive parameters were selected for inverse design by sensitive analysis. Then, a dimension-reduced dataset is created to train an inverse neural network with an mean R-squared value of 0.9909. Finally, experimental verifications were performed, which demonstrates an excellent agreement within the design domain. This approach offers promising potential for tailoring honeycomb structures with desired deformation, while also enabling the inverse design of metamaterials with customized properties.

## 1. Introduction

Honeycomb structure is a kind of widely used ultra-light architecture inspired by beehive, such as hexagonal [1], triangular [2], chiral [3,4], star shaped [5] and so on. Especially, honeycomb structure with negative Poisson's ratio(NPR) has attracted widespread attention due to their outstanding performance in mechanics [6], energy absorption [7,8] and acoustic property [9]. Poisson's ratio refers to the ratio of transverse normal strain to axial normal strain of a material under uniaxial tension or compression, while NPR means the material is auxetic. Artificial structures were involved into honeycombs to achieve auxetic behavior, such as chiral [4,10], re-entrant [11,12], disorder [13–15] and their derivative structures [16,17].

The out of plane bending behavior of NPR materials also presents peculiar characteristics. NPR materials exhibit a dome-like bending surface different from the saddle-shaped surfaces of ordinary materials. This unique behavior is particularly suitable for applications in sandwich structures with complex surfaces, such as radar radomes [18] and satellite antennas [19]. Morphing capabilities have become increasingly important [20,21] with the development of these devices. Hence, achieving desired deformations in honeycomb structures will play a crucial role in meeting these demands.

With the development of additive manufacturing technology especially 3D printing, it has become easier to build sophisticated honeycomb structures. For instance, researchers have explored gradient honeycomb structures and obtained fascinating results.[22–24]. However, challenges persist of designing non-uniform honeycomb structures [25]. Artificial intelligence (AI) is noticed as a promising method for designing inhomogeneous honeycomb structures. For example, Shen et al. [26] developed an inverse machine learning framework for optimizing the gradient honeycomb structure under impact loading, achieving a maximum energy absorption 49.5% higher than that of a uniform honeycomb structure. Similarly, Mohammadnejad et al. [27] proposed an inverse neural network approach to achieve a specified stress–strain and Poisson's ratio-strain curves, demonstrating significant time savings without compromising accuracy. Other studies have also explored inverse NNs in mechanical property. Quan et al. [28] proposed a neural network to predict the stress–strain curve from indentation curves, after supplying extra physics information (indentation load and pile-up information), they got good agreements between the results of NNs and experiments. Additionally, Lin et al. [29] employed machine learning to design materials with specific coefficients of thermal expansion and Poisson's ratios, comparing favorably with finite element

* Corresponding authors.
*E-mail addresses:* sunjian@hit.edu.cn (J. Sun), yj_liu@hit.edu.cn (Y. Liu).

https://doi.org/10.1016/j.compstruct.2025.119236
Received 5 December 2024; Received in revised form 22 March 2025; Accepted 28 April 2025
Available online 13 May 2025

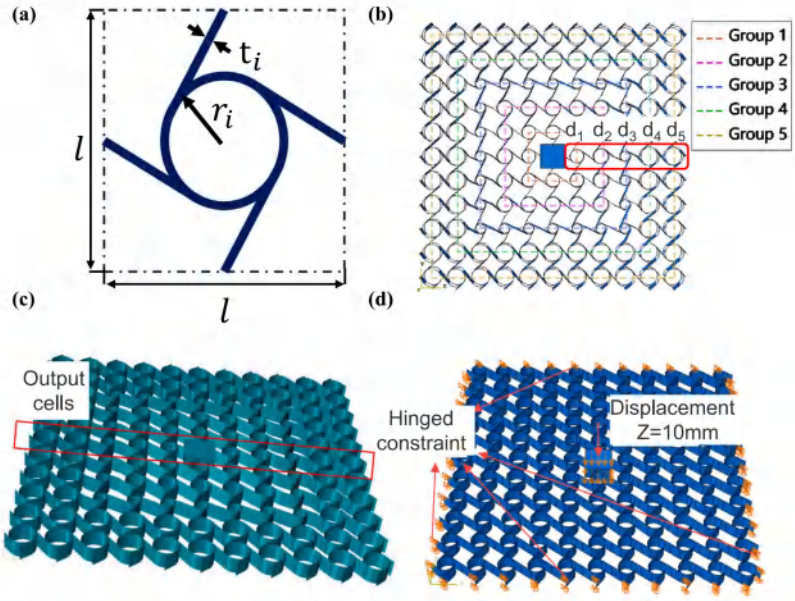


**Fig. 1.** (a) Unit cell of tetrachiral honeycomb. (b) Diagram of the layout of honeycomb board. (c) The mesh results of the honeycombs, the cells enclosed by the red frame are the result output cells. (d) The loading configuration of the honeycomb board.

method (FEM) and experimental results. However, the inverse honeycomb design for a desired out-of-plane deformation using NNs still remains an open problem.

This study explored the utilization of NNs to design a tetrachiral honeycomb board tailored to specific deformation requirements. A dataset was generated using Python scripts within the Abaqus environment, which included size parameters of the honeycomb cells and corresponding deformations under a given load. Then a neural network (forward NNs) was conducted out to predict the deformation of the honeycomb. Using forward NNs, a sensitivity analysis of the size parameters of the honeycomb was conducted out. Parameters with highest sensitivity index were taken as design parameters. Then another neural network (inverse NNs) structure was fabricated to predict the design parameters by deformations. Finally, the accuracy of the inverse NNs was demonstrated through three design cases.

## 2. Dataset

### 2.1. Configuration

The tetrachiral unit cell depicted in Fig. 1(a) is taken into consideration. This unit cell is enclosed within a square of side length $l$. The internal structure of the cell comprises a central circle of radius $r$ and four line segments, which start from the midpoint of square side and be tangent to the middle circle in anticlockwise direction. Both the lines and the circle share the same thickness in one cell, denoted as $t$. So, the unit cell is fully defined and can be described using three parameters: $l$, $r$, and $t$.

Next, consider a honeycomb board composed of the tetrachiral cells. The center of the board is a cube with the same unit size $l$. The tetrachiral unit cells surrounding the cube are divided into five groups according to five concentric squares, as shown in Fig. 1(b). Cells in each group share the same size parameters of $r$ and $t$, and all cells have the same edge length $l$. The innermost group is referred as group 1, and the outermost is group 5. Thus, this board can be defined by three different parameters: $l$, $\boldsymbol{r}$, and $\boldsymbol{t}$. The list $\boldsymbol{r}$ is defined as

$$\boldsymbol{r} = [r_1, r_2, r_3, r_4, r_5] \tag{1}$$

where $r_i$ is the radius of the $i$th group cells. Similarly, $\boldsymbol{t}$ is defined as

$$\boldsymbol{t} = [t_1, t_2, t_3, t_4, t_5] \tag{2}$$

where $t_i$ is the thickness of the $i$th group cells. With these parameters, the relative density of the tetrachiral honeycombs($D$) could be written as

$$D(\boldsymbol{r}, \boldsymbol{t}) = \frac{1}{(2n+1)^2} + \sum_{i=1}^{n} \frac{2\pi r_i t_i + 4t_i\sqrt{\frac{l^2}{4} - {r_i}^2}}{l^2} \frac{8i}{(2n+1)^2} \tag{3}$$

If the outer contours of the board is fixed ($l$ and dimension of cells are constant), $\boldsymbol{r}$ and $\boldsymbol{t}$ could be taken as variable design parameters to alter its mechanical properties so it can adapt to different situations. In this work, the range of $r_i$ is set between 2.0 mm and 4.0 mm and $t_i$ is in range 1.0 mm and 2.0 mm.

### 2.2. Simulation and data regeneration

First, random $\boldsymbol{r}$ and $\boldsymbol{t}$ were generated and then two matrix sized like the honeycomb board were made using these parameters. the values in these matrixes represent corresponding $r_i$ and $t_i$ of the cell. Next, Python scripts were employed to construct simulation models in Abaqus [30]. The unit cells were constructed separately as shell parts using the generated matrices. Various units were then inserted into the assembly, with a central solid cube among them. The combined structure was merged into a single instance, maintaining unique $r_i$ and $t_i$ values for each unit, effectively sidestepping inter-cell interactions. The materials of all the units were set as elastic with $E = 80$ MPa and $\mu = 0.4$ according to the material used in experiments (stated in Section 2.3). Displacement ('U') of the entire model was set as one of the outputs. S4R elements were used to mesh the honeycombs with an approximate global size of 0.5 mm, while the element type for the center cube is

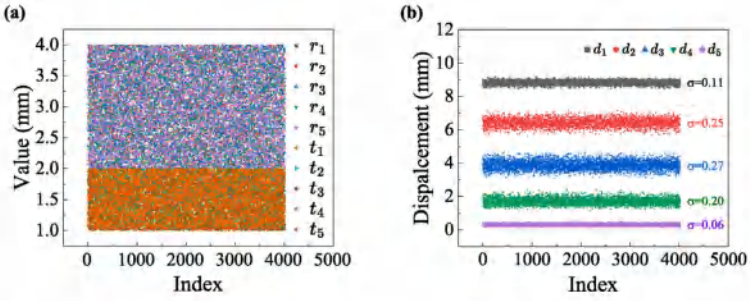


**Fig. 2.** Distribution of data points for forward neural network: (a) input parameters; (b) FEM results.

C3D8R. The numbers of nodes and elements in each cell are around 800 and 720 (determined by specific cell configuration), and for the whole structure are about 97,000 and 90,000. The independence of the mesh is verified by convergence analysis. A displacement load along the $z$-direction of 10 mm was applied on the top of the central cube, and pinned constraints were given on the boundaries. The mesh results and load configuration are shown in Fig. 1(c) and (d).

After the simulation finished, the result file was processed using Python scripts. In consideration of the symmetric characteristics of this board, taking the middle horizontal units' (framed in red in Fig. 1(c)) displacements along $z$-direction as research objects. The mean displacement of the FEM elements in each cell unit was calculated and written in a csv file with design parameters $\boldsymbol{r}$ and $\boldsymbol{t}$. the five mean displacements from inner to outer are defined as

$$\boldsymbol{d} = [d_1, d_2, d_3, d_4, d_5]. \tag{4}$$

To get enough data for neural network training, random $\boldsymbol{r}$ and $\boldsymbol{t}$ were applied in each case and over 4000 simulation cases were adapted in total. The whole data collected by the Python scripts could made available on request. The Fig. 2 shows the distribution of data points. It can be found that the input parameters follow random distribution, and $\boldsymbol{r}$ is located in 2 to 4 while $\boldsymbol{t}$ is between 1 and 2. The different output $d$ fluctuates around different values. The fluctuation ranges of $d_1$ and $d_5$ are relatively small. And the range of $d_2$, $d_3$ and $d_4$ is wider, which means the design space is bigger for these positions.

### 2.3. Simulation data validation

To validate the results obtained from Abaqus, an experiment was conducted. Sample honeycomb boards were made using 3D printing technology with TPU (material: WeNext TPU, manufacturer: WeNext Technology Co., China). As depicted in Fig. 3(b), the sample board measures 140 mm by 150 mm, which includes a 110 mm * 110 mm tetrachiral honeycomb area and a fixing skirt area.

Components were designed and fabricated using a FDM printer (printer: Bambu A1 mini, material: Bambu PLA, manufacturer: Bambu Lab, China) as fixtures and loaders, as shown in Fig. 3(b). The fixtures consisted of a plate and several clamps. The honeycomb board was affixed to the PLA plate using these clamps. A metal frame was also added to prevent displacement of the skirt area. The plate features a central hole through which a key-like loader can pass and be locked by rotation as shown in Fig. 3(c). Once locked in place, the loader can impart a 10 mm displacement in the out-of-plane direction to the central cube of the honeycomb board.

A laser distance sensor (type: HG-C1200, maker: Panasonic Industry Co., Ltd., Japan) was utilized to capture the deformation of the honeycomb board. To enable the sensor to measure displacements at various positions, it was mounted on the extruder of a 3D printer (type: Prusa i3 MK3S+ 3D printer, maker: Prusa Research a. s., Czech Republic), enabling it to move on the X-Z plane paralleling to the board. To create reflective areas on the honeycomb, 3D scanning mark points were affixed to the honeycomb. The sandwich structure was fixed on a breadboard using 3D printed components, which were supported by a bracket. The whole facility was depicted on Fig. 3(a). This experimental setup facilitates precise measurement and validation of the convex deformation of the honeycomb board.

Displacements of units on middle horizontal line and vertical line were measured to compare with simulation results for validation. The results are presented in Fig. 4. Because of symmetry, the displacements calculated by simulation along horizontal line and vertical line are the same. Using Python and scikit-learn package [31] to analyze the errors of experiment and simulation, the RMSE(Root Mean Square Error) of horizontal curve and simulation curve is 0.319 mm and the vertical one is 0.415 mm, The $r^2$ of the two direction are 0.991 and 0.984 respectively. The experimental results exhibit good coherence with the simulation results. Consequently, the simulation results can serve as training data for neural networks.

## 3. Forward neural network

### 3.1. Neural network construction

A neural network is a machine learning algorithm that can model complex and nonlinear relationships within labeled data samples using a large number of hidden parameters. It comprises many simple units called neurons. Each neuron has an input and an output linked by an activation function. When these neurons are organized in series or parallel, it is referred as a neural network. The strength of connections between neurons can be adjusted by modifying the weights associated with the activation function during data fitting. Training a neural network involves optimizing these weights to ensure accurate predictions on unseen data.

Pytorch is a popular tools library to enable rapid research on machine learning models [32]. A neural network can be easily built based on it. In this work, a five-layer fully connected neural network was constructed to predict the deformation of a variable-shape honeycomb board under a specific displacement load. The structure of this neural network is detailed at Table 1 and Fig. 5.

The original input comprises two lists: $\boldsymbol{r}$ and $\boldsymbol{t}$, each containing five elements. To improve fitting performance, the input data using was scaled using scikit-learn's StandardScaler method [31]. The processed data has a mean of zero and a standard deviation of one, conforming to a standard normal distribution. The dataset comprises 4013 simulated samples, with 100 of them reserved for the test set. The rest samples then is split into two parts: 25% for the validation dataset and 75% for the training dataset. Consequently, the length of the training dataset is 2934.

ELU (Exponential Linear Unit, defined as $\max(0, x) + \min(0, \alpha(\exp(x) - 1))$) is a nonlinear activate function for faster and more precise learning in deep neural networks [33], where $\alpha$ is a hyperparameter controlling the saturates for negative inputs. Compared with ReLU(Rectified Linear Unit, defined as $\max(0, x)$) activate function which used

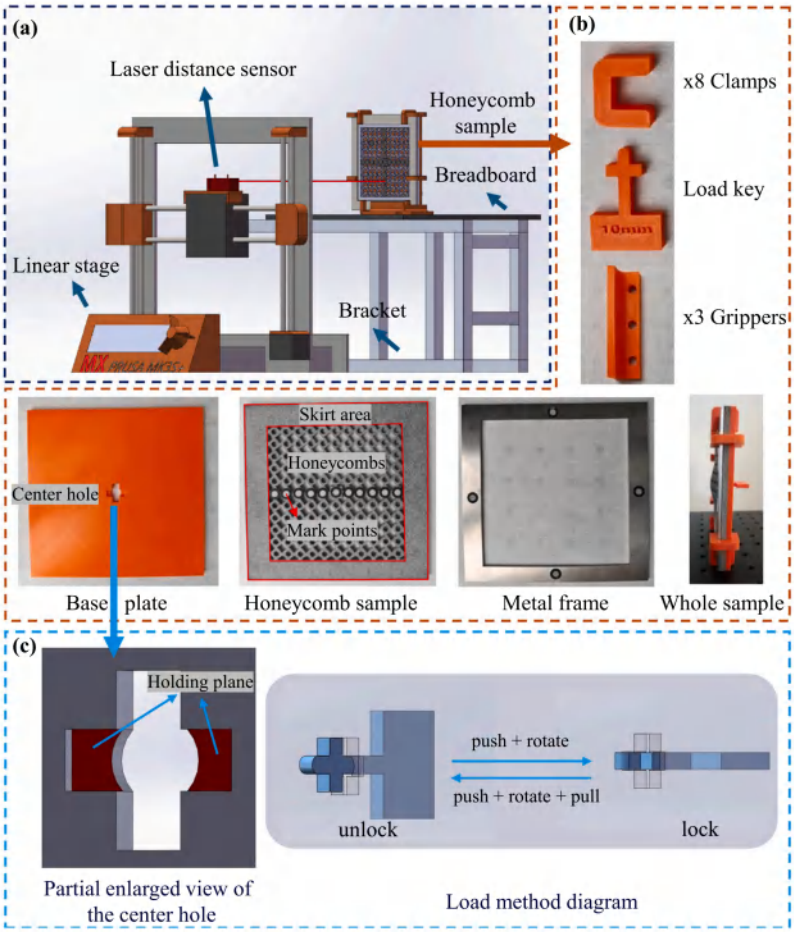


**Fig. 3.** Experiment instruments to measure the deflections of the honeycomb plate. (a) The entire experimental setup. (b) The components of the test sample. (c) Loading method: there is a hole on the center of base plate which the load key can pass through vertically, and the holding plane prevents the horizontal key from being pulled back by honeycombs when the key is rotated.

**Table 1**
Structure of neural network.

| Layer number | Input size | Activate function | Layer type | Output size | Params |
|---|---|---|---|---|---|
| 1 | 10($\boldsymbol{r}$,$t$) | ELU | Linear-Input | 16 | 176 |
| 2 | 16 | ELU | Linear-Hidden | 32 | 544 |
| 3 | 32 | ELU | Linear-Hidden | 64 | 2112 |
| 4 | 64 | ELU | Linear-Hidden | 16 | 1040 |
| 5 | 16 | – | Linear-Output | 5($\boldsymbol{d}$) | 85 |

**Table 2**
List of customized hyperparameters in forward neural network.

| Hyperparameter | Value |
|---|---|
| Learning rate (lr) | 0.001 |
| Batch size | 64 |
| Log steps | 1000 |
| Eval steps | 100 |
| Epochs | 500 |

more wildly, ELU could be negative, which means it can push activations closer to zero. So it can avoid dying ReLU problem and speed up learning. In this work, activate functions for layer 1–4 were all set as ELU. For excepting continuous output, no activate function was put on the output layer. Labels $\boldsymbol{d} = [d_1, d_2, d_3, d_4, d_5]$ are set as final output.

RMSE(root mean square error) was used to be loss function. $r^2$ (coefficient of determination) is used to judge if the model exhibited a favorable consistency to the dataset. Adam optimizer is used in the networks. It combines the advantages of AdaGrad and RMSProp method [34], and has better performance than GSD optimizer in this networks by practicing. Hyperparameters costumed in this network are listed in table. 2. Other hyperparameters that were not explicitly mentioned are set to their default values.

### 3.2. Train results

Fig. 6 shows the change of loss and $r^2$ curve during training processing. Fig. 6(a) is the train loss and verify loss of the network during training. It shows that both train loss and verify loss decrease rapidly at the initial 1000 steps. Then the two curves tend to be horizontal at a value of 0.03, which is close to zero. This behavior indicates high performance of the neural network. The train loss has a initial value of about 1.1, which would be much bigger if not scaled and then

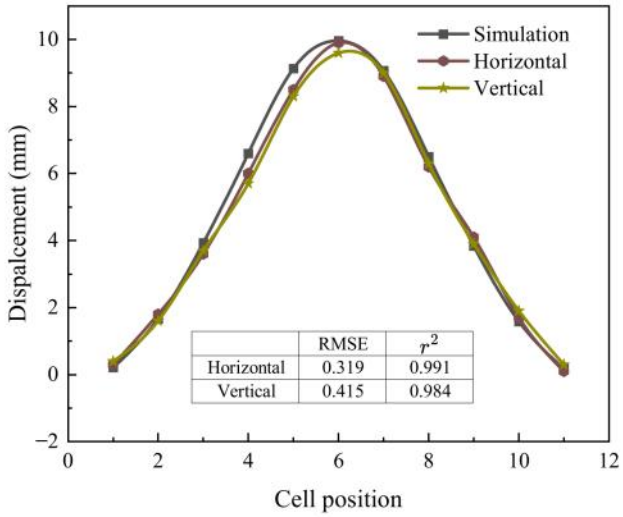


**Fig. 4.** Comparison of the deflections of experiments and simulations.

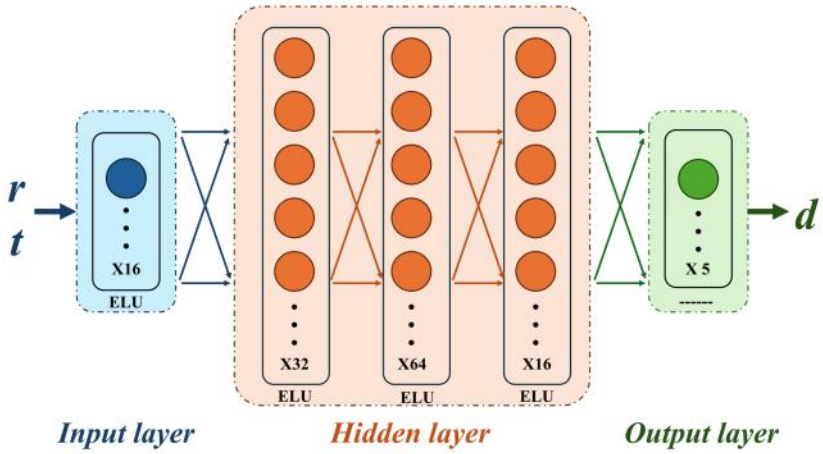


**Fig. 5.** Forward neural network structure.

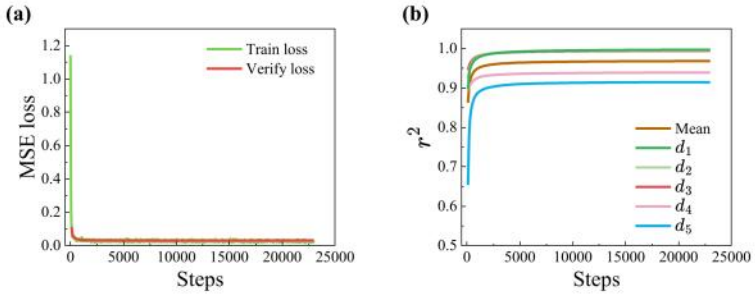


**Fig. 6.** Accuracy curves during train processing: (a) SME loss of train-dataset and verify-dataset; (b) $r^2$ score of the neural network of each output.

slowed down the fitting speed. In the flat phase, the verify loss is more stable than train loss. This is due to the optimizer was adjusting the parameters of the neural network and causing fluctuation. Curves in Fig. 6(b) show the change of $r^2$ during training process. The $r^2$ values of $d_1, d_2, d_3$ have a good beginning and then quickly converge into nearly one. The $r^2$ of $d_4$ has a high initials too and also convergence soon, but the convergence value is not so high. $r^2$ of $d_5$ has a 0.65 as initial value and converged to 0.90, which is the biggest increase among these $r^2$ values. In summary, both loss curves and $r^2$ curves have dramatic changes at the first 1000 steps and then become gentle. The

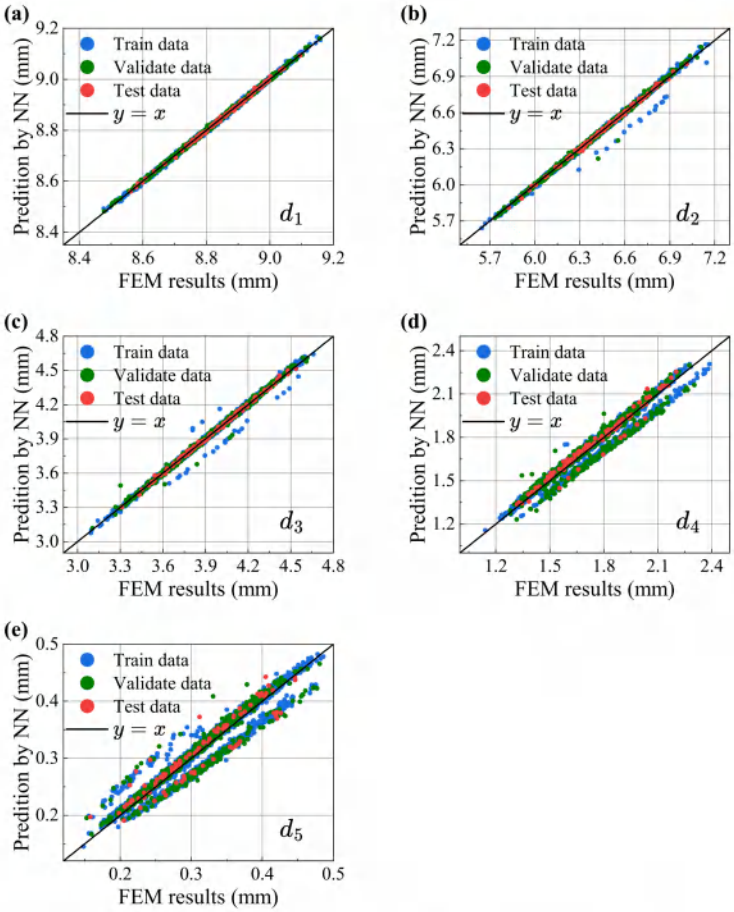


**Fig. 7.** Predictions and FEM results plots. Each subplot one output: (a) for $d_1$, (b) for $d_2$, (c) for $d_3$, (d) for $d_4$ and (e) for $d_5$, respectively.

final verify loss is 0.03078 and the mean $r^2$ is 0.96795, which indicates the favorable fit quality of the neural network. Test dataset created in Section 3.1 is taken to evaluate the fitted neural network. The $R^2$ value is 0.9680 and the loss is 0.0248, both indicating fine-tuned prediction ability.

A scatter plot comparing predictions to true values provides an intuitive method for assessing prediction accuracy. In this plot, the $X$-axis and $Y$-axis represents the true and prediction value respectively. The line (y=x) signifies perfect agreement between predicted and true values, and the closer the points lie to this line, the higher the prediction accuracy. Fig. 7 illustrates the accuracy of NNs across the training dataset, validation dataset, and test dataset. Notably, the accuracy of $d_5$ is the highest, followed by a gradual decrease from $d_5$ to $d_1$. However, there is a discernible gap between the line around which most points cluster and the secondary/third line. This discrepancy arises because all five outputs share a common neural network, and the range of $d_5$ is nearly 20 times greater than that of $d_1$. The same error is more pronounced for $d_1$. Additionally, due to activation functions, the change of the outputs is not continuous, resulting in the gap between the two main points-concentrate lines.

A simple comparison of the time cost between FEM solution and NNs prediction is made in Table 3. The hardware information of the platform is Intel i9-13900K CPU, 64 GB RAM, NVIDIA RTX 4090 GPU, and the software versions are Windows 11 23H2, Python 3.11, CUDA 12.4, Pytorch 2.1.1, Visual Studio Code 1.93.1. The parallelization configuration of Abaqus is *cpus = 24, threads_per_mpi_process = 1*. The time cost of Abaqus is taken as wall clock time from message file of one job, while the time cost of NNs is the mean time in 10,000 predictions of random loops. The advantage of NNs is obvious with a over 50,000 times faster speed than FEM method, which shows promise in structure optimization.

**Table 3**
Comparison of the time cost between FEM solution and NNs prediction.

| | FEM | NNs | FEM/NNs |
|---|---|---|---|
| Time cost (s) | 22 | $4.1 * 10^{-4}$ | 53,568 |

### 3.3. Sensitivity analysis

Sensitivity analysis examines how an independent input affects the output of a mathematical model in comparison to other inputs. It helps identify the key variables that warrant closer attention from a group of variables. Sobol analysis, proposed by Ilya M. Sobol [35], is a global sensitivity method based on variance. As a kind of Monte Carlo probability theory, the Sobol method measures the sensitivity of output to an input variable by quantifying the uncertainty of the input and output into probability distributions and decomposing the output variance into input variables and their combinations. It is a global sensitivity analysis

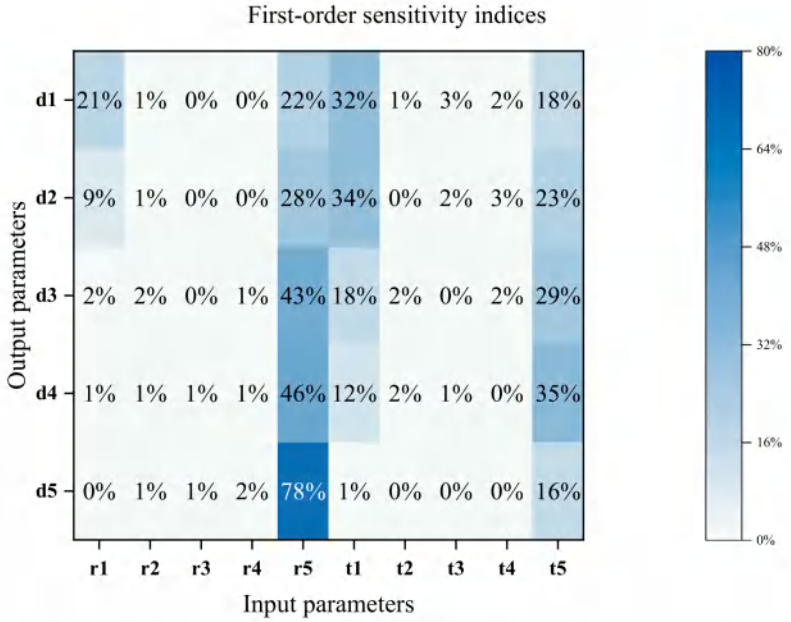


**Fig. 8.** The first-order Sensitivity indices of different independent parameter to dependent parameters respectively.

method that can deal with nonlinear correspondence as well as non-additive systems. Over time, it has become a widely used sensitivity analysis technique, especially after being supplemented and refined by Saltelli et al. [36,37]. The first-order sensitivity index, denoted as $S_i$, expresses the contribution of input $X_i$ to the output variance:

$$S_i = \frac{V(Y|X_i)}{V(Y)} \tag{5}$$

Here, $V(Y|X_i)$ represents the variance of the output $Y$ given input $X_i$, and $V(Y)$ is the total output variance. A value of $S_i$ close to 1 indicates that input $X_i$ significantly influences the output $Y$, while a value close to 0 suggests minimal impact. This approach can help us understand which inputs have the greater impact on the model's output, so the model's performance could be optimized.

The SALib toolkit [38,39] was used to perform sensitivity analysis of the input parameters of the neural network. SALib is a package of Python implementations of sensitivity analysis methods including Sobol. The first-order analyze results are shown as Fig. 8. From the direction of the independent variables, $r_5$ has the strongest impact on the deformation of the honeycomb panel, Specifically, the deformation at the cell corresponding to $r_5$ is the largest, with a sensitivity exceeding 0.7. The deformation effect on other cell elements gradually decreases with the increase of the distance to $r_5$'s cells. The parameter that has the second greater impact on the overall deformation is $t_5$, which has a greater impact on output $d_4$ than its own cell $d_5$. It also has a significant influence on other outputs, which indicating that the deformation of the structure is relatively more dependent on the size of the outermost honeycomb $(r_5, t_5)$ under the load conditions in this study. Parameters $r_1$ and $t_1$ also impact displacement in their own locations, but their effects are less pronounced for cells far from their positions. Other inputs $(r_2, r_3, r_4, t_2, t_3, t_4)$ have relative small effect on the output. From the direction of the dependent variable on Fig. 8, $r_5$ accounts for the major proportion of output $d_3, d_4$ and $d_5$, followed by $t_5$. And $t_1$ also have contribution to $d_3$ and $d_4$. For output $d_1$ and $d_2$, the impact of $t_1$ is more than $r_5$. The $r_1$ and $t_1$ are also important influencing factors.

Fig. 9 provides insights into the influence of each parameter by illustrating the contribution of parameters to the displacements. Notably, the correlation between deformation and radius is generally weak, except for the pair $r_5$ and $d_5$. It reveals a positive correlation between $r_5$ and $d_5$ from the fifth figure in Fig. 9 row 1. The correlation between deformation and thickness is not apparent too for parameters $t_1 to t_4$. But a definite negative correlation for $t_5$ could be seen in Fig. 9 row 2. The information of Fig. 8 and Fig. 9 can correspond to each other, reinforcing the correctness of the sensitivity analysis.

In general, the sensitivity of honeycomb deformation to the size parameters of honeycombs at different locations is different, among which $r_5$ and $t_5$ are the main influencing factors, $t_1$ and $r_1$ have a certain impact on their nearby cells, but have a small impact on their distant cells. Other parameters $(r_2, r_3, r_4, t_2, t_3, t_4)$ have small influences on the deformation of the honeycomb.

## 4. Inverse neural network

By constructing an inverse neural network, the design parameters of the honeycomb panel can be obtained based on the deformation conditions. This approach enables us to solve mechanical problems in reverse and offers a novel solution for the reverse design of mechanical structures.

### 4.1. Inverse neural network construction

Based on the previous sensitivity analysis, the sensitivity indices of the honeycomb panel's size parameters is categorized into three types. First, there are parameters $(r_5, t_5)$ that significantly impact the global output. Second, there are parameters $(r_1, t_1)$ that have a certain influence on the local area and its vicinity. Finally, there are parameters $(r_2, r_3, r_4, t_2, t_3, t_4)$ that have limited impact on honeycomb deformation. For simplicity, this study focuses only on the first two types of influential parameters, while the less impactful parameters are assigned fixed values at the middle of their range:

$$r_2 = r_3 = r_4 = 3.0 \tag{6}$$

$$t_2 = t_3 = t_4 = 1.5 \tag{7}$$

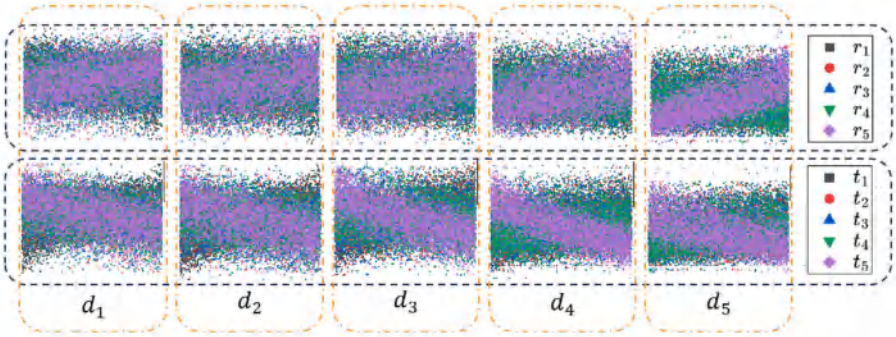


**Fig. 9.** Correlation of the deformation and the cell parameters. The first line is the correlation between deformation and radius; and the second is the correlation between deformation and thickness.

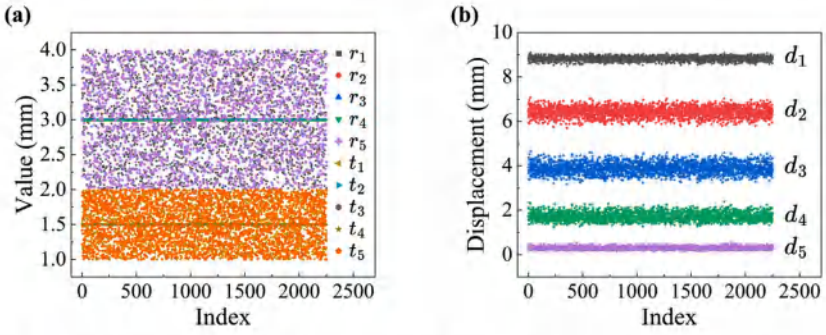


**Fig. 10.** Distribution of data points for inverse neural network: (a) distribution of $\boldsymbol{r}$ and $t$; (b) distribution of $\boldsymbol{d}$.

Building upon this foundation, a Python script is used to simulate and generate data. Different from the previous, the vector $\boldsymbol{r}$ now takes the following form:

$$\boldsymbol{r} = [r_1 = random, \quad r_2 = 3.0, \quad r_3 = 3.0, \quad r_4 = 3.0, \quad r_5 = random], \tag{8}$$

Meanwhile, the vector $\boldsymbol{t}$ is

$$\boldsymbol{t} = [t_1 = random, \quad t_2 = 1.5, \quad t_3 = 1.5, \quad t_4 = 1.5, \quad t_5 = random]. \tag{9}$$

A total of 2251 data-points were collected, 100 of which were divided into test sets according to the previous method, and the rest were divided into training sets and validation sets according to the ratio of 3:1, of which the number of training sets was 1613. Similarly, the dataset were scaled using the normalization method in the SK-learn toolkit. The distributions of $\boldsymbol{r}$ and $\boldsymbol{t}$ are shown in Fig. 10(a). Since the values of $r_2$, $r_3$, $r_4$ and $t_2$, $t_3$, $t_4$ are fixed, they form horizontal overlapping lines in Fig. 10(b) illustrates the corresponding displacement distributions. Similar to Fig. 9, the fluctuation ranges of $d_2$, $d_3$ and $d_4$ are larger than those of $d_1$ and $d_5$. Furthermore, Fig. 11 illustrates the correlation between displacement and radius/thickness of cells in the inverse neural network dataset. Fig. 11(a) focuses on $r_1$ and $r_5$, while Fig. 11(b) exhibits $t_1$ and $t_5$. In Fig. 11(a), there is a tendency for displacement to increase as $r_5$ rise, suggesting a positive correlation. However, the correlation between $\boldsymbol{d}$ and $r_1$ is less obvious. In Fig. 11(b), the tends for $t_1$ and $t_5$ are more pronounced, with a positive correlation between $\boldsymbol{d}$ and $t_1$, and a negative one for $t_5$.

The displacements ($\boldsymbol{d}$) are taken as input parameters, while $r_1$, $r_5$ and $t_1$, $t_5$ are used as outputs. Due to the limited dataset size, the previous forward neural network structure was not adopted. Instead, four separate neural networks are constructed, each focusing on predicting one output of $r_1$, $r_5$, $t_1$ and $t_5$. To enhance input dimensionality and accelerate neural network convergence, we mirrored the parameter $\boldsymbol{d}$ and include it as an additional input alongside $\boldsymbol{d}$. Consequently, the neural network input dimension is expanded to a size of 10.

**Table 4**
Inverse neural network structure (taking $r_1$ branch as an example).

| Layer number | Input size | Activate function | Layer type | Output size | Params |
|---|---|---|---|---|---|
| 1 | 10 | ELU | Linear-Input | 64 | 704 |
| 2 | 64 | ELU | Linear-Hidden | 64 | 4160 |
| 3 | 64 | ELU | Linear-Hidden | 64 | 4160 |
| 4 | 64 | ELU | Linear-Hidden | 64 | 4160 |
| 5 | 64 | – | Linear-Output | 1($r_1$) | 65 |

In contrast to the forward network's homogeneous output structure, the inverse network generates four distinct physical parameters characterized by heterogeneous dimensionalities. Each parameter exhibits unique functional roles and differential sensitivity indices. This structural configuration suggests that employing four dedicated neural networks, rather than a unified architecture, can achieve superior performance. The individualized network training protocol not only enhances parameter-specific optimization precision and accelerates convergence rates, but also effectively mitigates cross-parameter interference phenomena commonly observed in multi-output network paradigms. This method is also adapted by other researchers [27,40].

The 4 neural network have same structures. The first layer is the input layer of 64 cells, the activation function is the ELU function. And the second to fourth layers are the hidden layer, the number of cells is 64, 64, and 64 respectively, and the activation function is also the ELU function. The last layer is the output layer, since each sub-network only outputs one result, so the number of neural network cells in the output layer is 1. The configuration of each group of neural networks are summarized in the table 4. The neural network structure is shown in Fig. 12. Table 5 shows some hyperparameters of neural networks, and the remaining values are the default values.

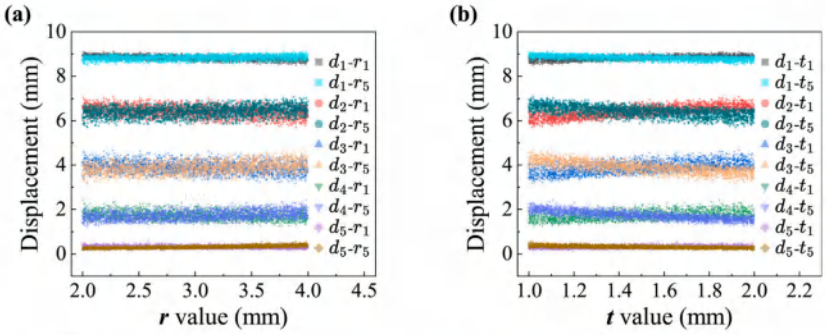


**Fig. 11.** (a) Correlation of $d$ and $r_1$, $r_5$ in dataset for inverse neural network. (b) Correlation of $d$ and $t_1$, $t_5$ in dataset for inverse neural network.

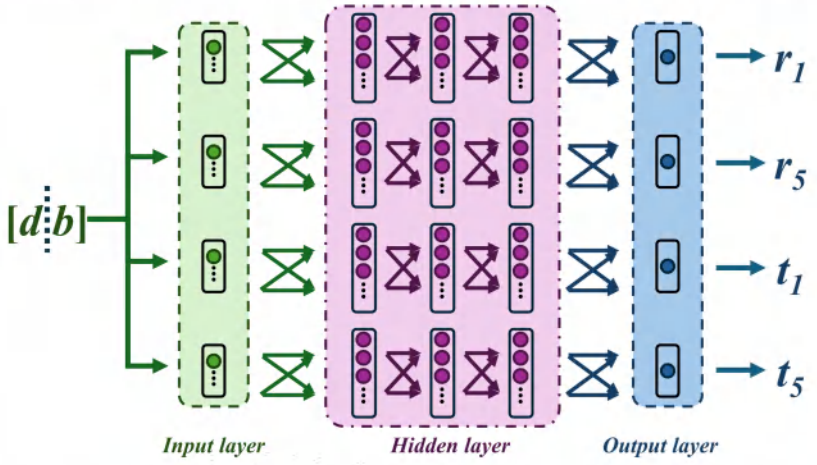


**Fig. 12.** The structure of inverse neural network. $[d|b]$ means the parameter $d$ and its mirror $b$ are combined as input data.

**Table 5**
List of costumed hyperparameters in inverse neural network.

| Hyperparameter | Value |
|---|---|
| Learning rate (lr) | 0.001 |
| Batch size | 64 |
| Log steps | 1000 |
| Eval steps | 100 |
| Epochs | 5000 |

**Table 6**
Evaluation indicators of inverse neural network structure.

| Parameter | Train MSE | Validate MSE | Validate $r^2$ |
|---|---|---|---|
| $r_1$ | 1.01e−4 | 4.69e−5 | 0.99963 |
| $t_1$ | 4.79e−5 | 4.90e−5 | 0.99976 |
| $r_5$ | 3.69e−4 | 0.0849 | 0.91327 |
| $r_5$ early stop | 1.42e−4 | 0.0391 | 0.87183 |
| $t_5$ | 4.62e−4 | 0.02941 | 0.96864 |
| $t_5$ early stop | 0.01086 | 0.01317 | 0.95400 |

### 4.2. Train results

After training the inverse neural network, the results are depicted in Figs. 13 and 14. Fig. 13 illustrates the change of MSE loss for the four outputs ($r_1$, $r_5$, $t_1$, and $t_5$) during training. Initially, all networks experience a decline in MSE loss. And then the curves of $r_1$ and $t_1$ are relatively stable, with values biased toward zero. However, the curves for $r_5$ and $t_5$ exhibit violent fluctuations during convergence, indicating potential overfitting.

To mitigate overfitting risks, the early stopping technique is implemented as a training strategy. This approach dynamically monitors the validation's MSE loss during training. Specifically, when the validation loss exhibits consecutive increases over 60 optimization iterations (as visualized by the dotted line in Figs. 13 and 14), the algorithm automatically terminates the training process. The model parameters corresponding to the lowest observed validation loss are then preserved and output as the final results.

The evaluation results, summarized in Table 6, reveal the following:
For $r_1$ and $t_1$:

- Training MSE and validation MSE are small.
- Validation $r^2$ is close to 1, indicating good network fitting.

For $r_5$ and $t_5$:

- Without the early stop strategy, training MSE can be very low. However, validation MSE remains relatively high.
- After adopting the early stop strategy, validation $r^2$ is reduced, but MSE loss is also reduced, which benefits the neural network model.

The curve in Fig. 14 illustrates the evolution of the $r^2$ during the training of the inverse neural network. As observed from the figure, after an initial training phase, the $r^2$ curves for the four outputs exhibit rapid improvement, followed by a stable period where they remain

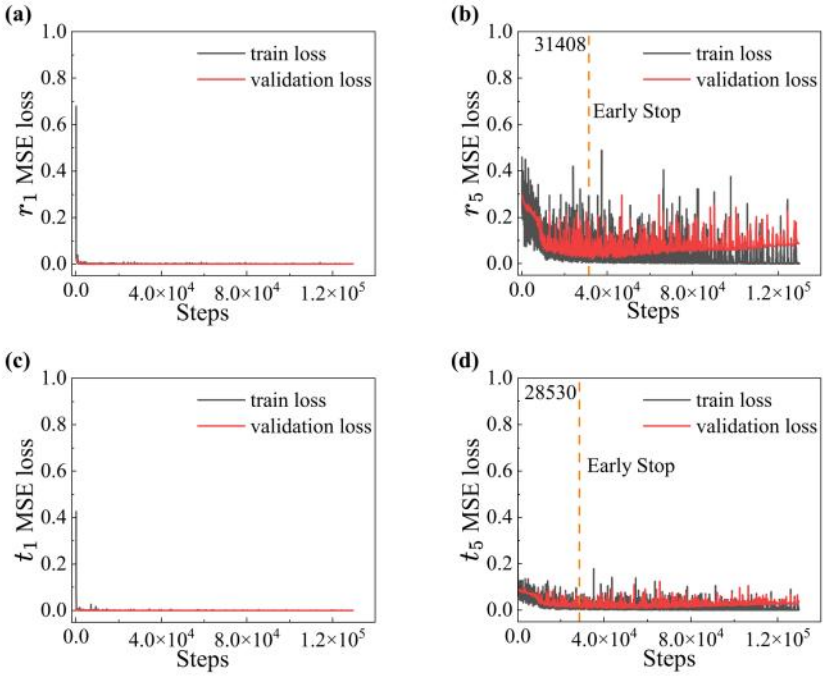


**Fig. 13.** MSE loss of inverse neural network: (a) illustrates the results for $r_1$, (b) for $t_1$, (c) for $r_5$ and (d) for $t_5$.

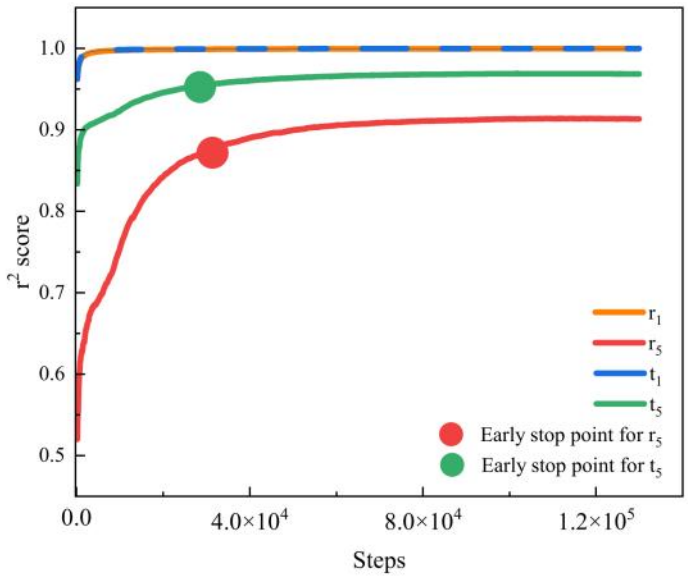


**Fig. 14.** $r^2$ score of the train dataset of inverse neural network.

essentially unchanged. Both $r_1$ and $t_1$ curves show a high degree of coincidence, with values exceeding 0.999. The $t_5$ curve converges around 0.95, while the $r_5$ curve undergoes significant changes—from approximately 0.5 to around 0.9. Due to the early stop strategy, the $r_5$ and $t_5$ values at 40k steps, which do not differ significantly from those at the 120k steps.

Loading the test dataset, evaluating the neural network, the $r^2$ and MSE loss of the model on the test set are shown in the Table 7:

Similar to the results from the validation dataset, both $t_1$ and $r_1$ exhibit the best performance, achieving an $r^2$ value exceeding 0.99 and a loss within 0.0001. The accuracy of $t_5$ ranks second, also reaching an $r^2$ of 0.99 with a loss of 0.01. However, the accuracy of $r_5$ is the lowest among the four neural networks, with an $r^2$ of 0.9758 and an MSE loss of 0.025. These findings align with the results of the previous sensitivity analysis, where honeycomb deformation showed a strong correlation with the $r_5$ variable.

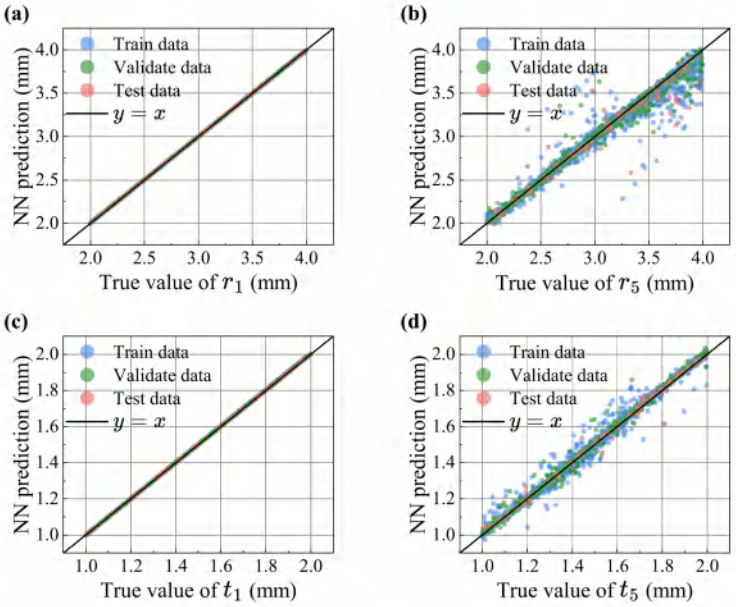


**Fig. 15.** Predictions and true values plots, with each subplot dedicated to a specific parameter: (a) illustrates the results for $r_1$, (b) for $r_5$, (c) for $t_1$, and (d) for $t_5$ respectively.

**Table 7**
$r^2$ score and MSE loss of inverse neural network in test dataset.

| | $r_1$ | $r_5$ | $t_1$ | $t_5$ | Mean |
|---|---|---|---|---|---|
| $r^2$ score | 0.9979 | 0.9758 | 0.9999 | 0.9900 | 0.9909 |
| MSE loss | 3.94e−5 | 0.0250 | 3.59e−5 | 0.00773 | 0.0082 |

Fig. 15 demonstrates the accuracy of inverse neural network by plotting prediction and true(label) points. Specifically, Fig. 15(a) and Fig. 15(c) show a strong fitting ability for $r_1$ and $t_1$, while Fig. 15(b) and Fig. 15(d) exhibit a less perfect fit, consistent with the findings in Tables 6 and 7. In comparison to Fig. 7, Fig. 15 lacks a significant second cluster line. This absence can be attributed to each of the four outputs having its own dedicated neural network, preventing parameter distortion caused by interactions with other outputs.

## 5. Design cases

To validate the efficacy of the inverse neural network in structure design, three practical cases are presented. Initially, the feasible input domains for the design are determined. Utilizing the dataset obtained from simulations, the range of honeycomb deformation is defined, as illustrated in Fig. 16(a). Notably, the pink region represents the feasible area of honeycomb deformation within the scope of this study. The first case involves scatter points situated within the pink region. The second case targets a Gaussian curve that does not entirely fall within the feasible area. While the third case is a sine curve almost entirely within the pink region.

First, the input parameters are calculated for the three distinct cases. For the scatter points design case, five points are selected within the feasible domain, as depicted in Fig. 16(a). For the Gaussian curve design case, the deflection curve of the honeycombs is assumed to follow a Gaussian function expressed as:

$$\frac{y}{a} = e^{-\frac{x^2}{b}+c} \tag{10}$$

Based on symmetry, $c$ is set to zero. Given that the function passes through the point (0, 10), $a$ is determined to be 10. Since the function value cannot be zero, a small value of 0.25 is assigned to the boundary cell as the displacement of this point. Substituting this value, $b$ is calculated to be 677.72. The curve of this function is shown in Fig. 16(a), illustrating that the curve does not completely locate within the feasible region, indicating the presence of certain errors. Similarly, for the sine curve case, the sine curve is defined as:

$$y = a + b\sin(cx) \tag{11}$$

The coefficients of the function are adjusted to ensure that the curve falls within the feasible domain, with the values of $a$, $b$ and $c$ calculated to be 4.8366, 4.9149 and 0.05277 respectively. As depicted in Fig. 16(a), the curve almost entirely locates within the feasible region, indicating a competent fitting results between the destination deflections and real deformations.

Next, the coordinate values of the cell centers are substituted to derive the corresponding displacements, which are then fed into the inverse neural network to predict the size parameters. To ensure the validity of the predictions, the output results are trimmed at the boundary of the feasible domain. Subsequently, these predicted size parameters are input into the forward neural network to obtain the NNs' prediction. Additionally, FEM simulations are carried out to compare the results of the inverse design. Finally, deformation experiments are performed on honeycomb pieces with the same output size as predicted by the inverse NNs. The results are illustrated in Fig. 16(b-d) and the comparisons of these results are presented in Table 8.

Form Fig. 16(b-d), it can be observed significantly that all cases exhibits agreement of the four deflections. Among them, the scatter case and sine case show a particularly high level of coincidence, making it difficult to distinguish the different result curves in the figure. The Gaussian case, however, shows less coincidence between the target curve and the other three curves. This is because in this specific case, the inputs dose not fully lie within feasible domain. Consequently, the inverse neural network's output exceeds the permissible bounds for cellular size, leading to truncation at the extremities of the feasible range. This results in a relatively low alignment between the actual deformation of the cellular structure and the desired target value. Regardless of the target curve, the remaining curves still demonstrate good consistency, which indicates the high accuracy of NNs. From

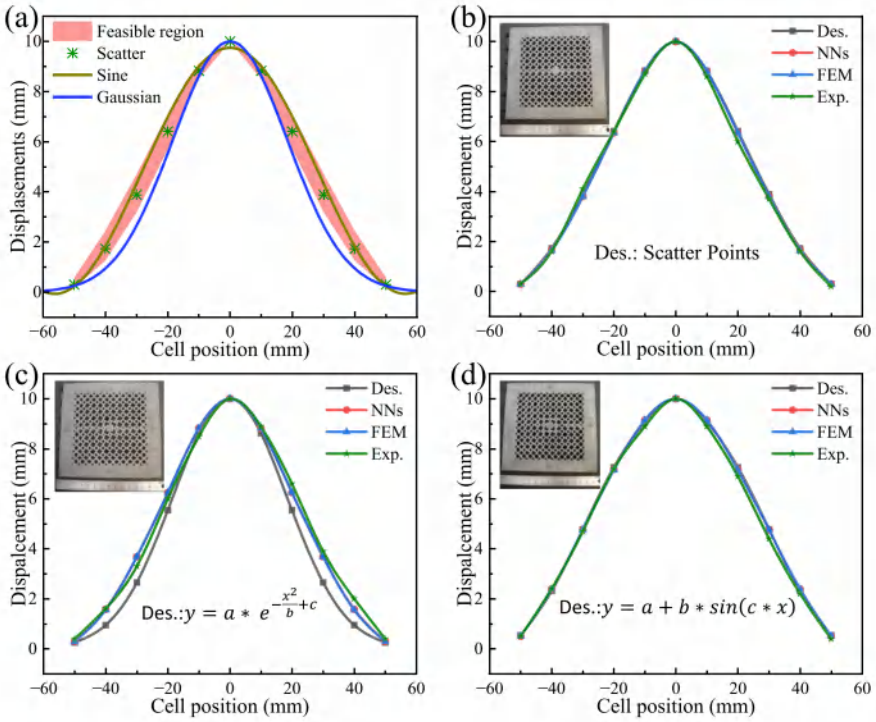


**Fig. 16.** The feasible domains of the design parameters and the locations of each design case.

Table 8, a similar conclusion can be drawn compared to that in Fig. 16. The $r^2$ between all curves in the three design cases are all exceeds 0.9. Notably, in the scatter case and sine case, the $r^2$ values surpass 0.99. Additionally, the Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) are relatively small in these two cases. In the Gaussian design case, due to the overshoot of the feasible domain, the errors between the target and the other three deflections are significantly larger. However, the remaining three curves still exhibit good consistency. Among the three cases, the consistency between NNs and FEM is the highest, indicating the forward NNs have better accuracy. The high overlap among these values indicates that the inverse neural network performs exceptionally well in this case and demonstrates robust inverse-design capabilities. As evidenced by the aforementioned three inverse design cases, the forward neural network exhibits exceptional predictive capabilities, demonstrating a high degree of consistency with FEM simulation outcomes. Furthermore, within the feasible design space, the inverse neural network proves adept at accurately conducting inverse reasoning for honeycomb size. This competency has been confirmed through experimental validation.

The analysis of three design cases demonstrates that the inverse design network can successfully generate honeycomb configurations when the target curve lies within the feasible domain. However, prediction accuracy degrades significantly when target curves exceed this domain. This limitation primarily stems from the constrained feasible region caused by:

- Narrow parameter ranges of honeycomb cells, dictated by dataset acquisition costs,
- Practical manufacturability requirements for cellular structures.

To overcome these constraints, the following directions are proposed for future research:

**Table 8**
The honeycomb deformation comparison of the destinations, NNs predictions, simulations and experiments.

| | | Des. Vs. NNs | Des. Vs. FEM | Des. Vs. Exp. | NNs Vs. FEM | NNs Vs. Exp. | FEM Vs. Exp. |
|---|---|---|---|---|---|---|---|
| Scatter | RMSE | 0.056 | 0.064 | 0.180 | 0.012 | 0.162 | 0.161 |
| | MAPE | 1.92% | 2.74% | 7.86% | 0.87% | 6.85% | 6.44% |
| | $r^2$ | 0.9997 | 0.9996 | 0.9971 | 0.9999 | 0.9977 | 0.9977 |
| Gaussian | RMSE | 0.660 | 0.653 | 0.737 | 0.012 | 0.248 | 0.251 |
| | MAPE | 30.22% | 29.32% | 21.14% | 0.60% | 8.74% | 9.07% |
| | $r^2$ | 0.9549 | 0.9558 | 0.9510 | 0.9999 | 0.9945 | 0.9943 |
| Sine | RMSE | 0.061 | 0.064 | 0.187 | 0.016 | 0.183 | 0.177 |
| | MAPE | 2.29% | 1.67% | 6.19 | 0.79% | 4.86% | 5.43% |
| | $r^2$ | 0.9996 | 0.9996 | 0.9968 | 0.9999 | 0.9969 | 0.9971 |

1. Expand the parameter ranges through active learning strategy [41,42] to progressively enrich the training dataset.
2. Implement Physics-Informed Neural Networks (PINNs) to integrate mechanical constraints into the neural network [43].
3. Enrich cellular diversity by incorporating multi-scale and multi-topology honeycomb configurations.

## 6. Conclusion

In this study, the usage of NNs for designing tetrachiral honeycomb boards based on specific out-of-plane deformation requirements is investigated. Different sizes of honeycombs are modeled and simulated using Python scripts and FEM software. A dataset mapping size parameters and corresponding deformations is generated from FEM results. A neural network is then constructed and trained to predict the deformation of honeycombs. A sensitivity analysis is performed on this

NNs to identify the influence of size parameters. Additionally, another NNs is developed, taking deformation as input and outputting size parameters filtered by sensitive analysis. The accuracy of the inverse NNs was demonstrated through three design cases. The forward neural network demonstrated promising results, with a low MSE loss of 0.0308 and an average $r^2$ of 0.9680. Sensitivity analysis highlighted critical honeycomb size parameters ($r_5, t_5, t_1$ and $r_1$). These parameters play a pivotal role in the reverse design process. The inverse neural network, trained utilizing an early stopping strategy and dependent network for each output, yielded impressive performance with an average test MSE loss of 0.0082 and a remarkable average $r^2$ score of 0.9909. Finally, three inverse design cases are employed to validate the performance of the inverse neural network. Notably, in the scatter case and the sine curve case, both of which reside entirely within the design domain, the neural network's predicted honeycomb deformation exhibits a remarkable alignment with the target value. Nevertheless, for Gaussian curve cases that extend beyond the design domain, there is a decrease in the network's consistency. Future research endeavors could concentrate on broadening the scope of the design domain.

This proposed method provides a generalized framework for inverse design of out-of-plane deformation in different honeycomb structures, achieving cross-topology applicability through modular dataset substitution. Furthermore, it serves as a reference for similar approaches in other metamaterial structural inverse designs (e.g. Origami structure [44] and lattice structure [45]).

## CRediT authorship contribution statement

**Linzhe Du:** Writing – original draft. **Jian Sun:** Writing – review & editing, Methodology, Formal analysis, Conceptualization. **Yanju Liu:** Project administration. **Jinsong Leng:** Investigation.

## 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.

## Acknowledgments

This work was supported by the National Key R & D Program of China (2022YFB3805700).

## Data availability

Data will be made available on request.

## References

[1] Qi C, Jiang F, Yang S. Advanced honeycomb designs for improving mechanical properties: A review. Compos Part B: Eng 2021;227:109393. http://dx.doi.org/10.1016/j.compositesb.2021.109393.
[2] Han J, Chen H, Xu X, Li Z, Chen Q, Gu H, et al. Mechanical characterization of a novel gradient thinning triangular honeycomb. Thin-Walled Struct 2023;188:110862. http://dx.doi.org/10.1016/j.tws.2023.110862.
[3] Bian Z, Gong Y, Sun Z, Zhao L, Zhang J, Hu N. Design and energy absorption characteristics of a novel honeycomb with embedded chiral structures. Compos Struct 2024;333:117944. http://dx.doi.org/10.1016/j.compstruct.2024.117944.
[4] Mousanezhad D, Haghpanah B, Ghosh R, Hamouda AM, Nayeb-Hashemi H, Vaziri A. Elastic properties of chiral, anti-chiral, and hierarchical honeycombs: A simple energy-based approach. Theor Appl Mech Lett 2016;6(2):81–96. http://dx.doi.org/10.1016/j.taml.2016.02.004.
[5] Li X, Li Z, Guo Z, Mo Z, Li J. A novel star-shaped honeycomb with enhanced energy absorption. Compos Struct 2023;309:116716. http://dx.doi.org/10.1016/j.compstruct.2023.116716.
[6] Mousanezhad D, Haghpanah B, Ghosh R, Hamouda AM, Nayeb-Hashemi H, Vaziri A. Elastic properties of chiral, anti-chiral, and hierarchical honeycombs: A simple energy-based approach. Theor Appl Mech Lett 2016;6(2):81–96. http://dx.doi.org/10.1016/j.taml.2016.02.004.
[7] Qi C, Jiang F, Yu C, Yang S. In-plane crushing response of tetra-chiral honeycombs. Int J Impact Eng 2019;130:247–65. http://dx.doi.org/10.1016/j.ijimpeng.2019.04.019.
[8] Niu X, Xu F, Zou Z, Fang T, Zhang S, Xie Q. In-plane dynamic crashing behavior and energy absorption of novel bionic honeycomb structures. Compos Struct 2022;299:116064. http://dx.doi.org/10.1016/j.compstruct.2022.116064.
[9] Shi P, Chen Y, Wei J, Xie T, Feng J, Sareh P. Design and low-velocity impact behavior of an origami-bellow foldcore honeycomb acoustic metastructure. Thin-Walled Struct 2024;197:111607. http://dx.doi.org/10.1016/j.tws.2024.111607.
[10] Prall D, Lakes RS. Properties of a chiral honeycomb with a Poisson's ratio of — 1. Int J Mech Sci 1997;39(3):305–14. http://dx.doi.org/10.1016/S0020-7403(96)00025-2.
[11] Ma N, Han S, Han Q, Li C. Design and compressive behaviors of the gradient re-entrant origami honeycomb metamaterials. Thin-Walled Structures 2024;198:111652. http://dx.doi.org/10.1016/j.tws.2024.111652.
[12] Wang L, Liu H-T. Parameter optimization of bidirectional re-entrant auxetic honeycomb metamaterial based on genetic algorithm. Compos Struct 2021;267:113915. http://dx.doi.org/10.1016/j.compstruct.2021.113915.
[13] Shen X, Fang C, Jin Z, Tong H, Tang S, Shen H, et al. Achieving adjustable elasticity with non-affine to affine transition. Nat Mater 2021;20(12):1635–42. http://dx.doi.org/10.1038/s41563-021-01046-8.
[14] Zhu C, Fang C, Jin Z, Li B, Shen X, Xu L. A cyclical route linking fundamental mechanism and AI algorithm: An example from tuning Poisson's ratio in amorphous networks. Appl Phys Rev 2024;11(3):031401. http://dx.doi.org/10.1063/5.0199530.
[15] Jin Z, Fang C, Shen X, Xu L. Designing amorphous networks with adjustable Poisson ratio from a simple triangular lattice. Phys Rev Appl 2022;18(5):054052. http://dx.doi.org/10.1103/PhysRevApplied.18.054052.
[16] Kolken HMA, Zadpoor AA. Auxetic mechanical metamaterials. RSC Adv 2017;7(9):5111–29. http://dx.doi.org/10.1039/C6RA27333E.
[17] Wu W, Hu W, Qian G, Liao H, Xu X, Berto F. Mechanical design and multifunctional applications of chiral mechanical metamaterials: A review. Mater Des 2019;180:107950. http://dx.doi.org/10.1016/j.matdes.2019.107950.
[18] Hou Y, Tai Y, Lira C, Scarpa F, Yates J, Gu B. The bending and failure of sandwich structures with auxetic gradient cellular cores. Compos Part A: Appl Sci Manuf 2013;49:119–31. http://dx.doi.org/10.1016/j.compositesa.2013.02.007.
[19] Kumar S, Vyavahare S, Teraiya S, Kootikuppala J, Bogala H. A state of the art review of additively manufactured auxetic structures. In: Dave HK, Dixit US, Nedelcu D, editors. Recent advances in manufacturing processes and systems. Singapore: Springer Nature Singapore; 2022, p. 69–84.
[20] Jeong H, Park E, Lim S. Frequency memorizing shape morphing microstrip monopole antenna using hybrid programmable 3-dimensional printing. Addit Manuf 2022;58:102988. http://dx.doi.org/10.1016/j.addma.2022.102988.
[21] Guo X, Lin S, Dai F. A metal hybrid bistable composite tube for multifunctional and reconfigurable antenna. Compos Sci Technol 2023;233:109887. http://dx.doi.org/10.1016/j.compscitech.2022.109887.
[22] Li J, Wei Y, Wu H, Shen X, Yuan M. Experimental crushing behavior and energy absorption of angular gradient honeycomb structures under quasi-static and dynamic compression. Def Technol 2024. http://dx.doi.org/10.1016/j.dt.2024.02.002, S2214914724000205.
[23] Hu Y, Li Y, Zhang Y, Ding S, Wang R, Xia R. Design methodology for functional gradient star-shaped honeycomb with enhanced impact resistance and energy absorption. Mater Today Commun 2024;38:108020. http://dx.doi.org/10.1016/j.mtcomm.2023.108020.
[24] Ma N, Han S, Han Q, Li C. Design and compressive behaviors of the gradient re-entrant origami honeycomb metamaterials. Thin-Walled Struct 2024;198:111652. http://dx.doi.org/10.1016/j.tws.2024.111652.
[25] Jeong D, Li Y, Kim S, Choi Y, Lee C, Kim J. Mathematical modeling and computer simulation of the three-dimensional pattern formation of honeycombs. Sci Rep 2019;9(1):20364. http://dx.doi.org/10.1038/s41598-019-56942-6.
[26] Shen X, Yan K, Zhu D, Hu Q, Wu H, Qi S, et al. Inverse machine learning framework for optimizing gradient honeycomb structure under impact loading. Eng Struct 2024;309:118079. http://dx.doi.org/10.1016/j.engstruct.2024.118079.
[27] Mohammadnejad M, Montazeri A, Bahmanpour E, Mahnama M. Artificial neural networks for inverse design of a semi-auxetic metamaterial. Thin-Walled Struct 2024;200:111927. http://dx.doi.org/10.1016/j.tws.2024.111927.
[28] Jiao Q, Chen Y, Kim J-h, Han C-F, Chang C-H, Vlassak JJ. A machine learning perspective on the inverse indentation problem: Uniqueness, surrogate modeling, and learning elasto-plastic properties from pile-up. J Mech Phys Solids 2024;185:105557. http://dx.doi.org/10.1016/j.jmps.2024.105557.
[29] Lin B, Lu F, Zhang C, Wei T, Li W, Zhu Y. Machine learning-accelerated inverse design of programmable bi-functional metamaterials. Compos Struct 2024;346:118445. http://dx.doi.org/10.1016/j.compstruct.2024.118445.
[30] Abaqus 6.14, Vélizy-Villacoublay, France.
[31] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[32] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, et al. Automatic differentiation in PyTorch. In: 31st conference on neural information processing systems (NIPS 2017). Long Beach, CA, USA.; 2017.

[33] Clevert D-A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). 2016, arXiv:1511.07289.
[34] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2017, arXiv:1412.6980.
[35] Sobol' I. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. Math Comput Simulation 2001;55(1–3):271–80. http://dx.doi.org/10.1016/S0378-4754(00)00270-6.
[36] Saltelli A. Making best use of model evaluations to compute sensitivity indices. Comput Phys Comm 2002;145(2):280–97. http://dx.doi.org/10.1016/S0010-4655(02)00280-1.
[37] Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput Phys Comm 2010;181(2):259–70. http://dx.doi.org/10.1016/j.cpc.2009.09.018.
[38] Herman J, Usher W. SALib: An open-source python library for sensitivity analysis. J Open Source Softw 2017;2(9). http://dx.doi.org/10.21105/joss.00097.
[39] Iwanaga T, Usher W, Herman J. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. Socio- Environ Syst Model 2022;4:18155. http://dx.doi.org/10.18174/sesmo.18155.
[40] Jiao Q, Chen Y, Kim J-h, Han C-F, Chang C-H, Vlassak JJ. A machine learning perspective on the inverse indentation problem: Uniqueness, surrogate modeling, and learning elasto-plastic properties from pile-up. J Mech Phys Solids 2024;185:105557. http://dx.doi.org/10.1016/j.jmps.2024.105557.
[41] Chen C-T, Gu GX. Generative deep neural networks for inverse materials design using backpropagation and active learning. Adv Sci 2020;7(5):1902607. http://dx.doi.org/10.1002/advs.201902607.
[42] Wang Z, Xian W, Baccouche MR, Lanzerath H, Li Y, Xu H. Design of phononic bandgap metamaterials based on Gaussian mixture beta variational autoencoder and iterative model updating. J Mech Des 2022;144(4):041705. http://dx.doi.org/10.1115/1.4053814.
[43] Hu H, Qi L, Chao X. Physics-informed neural networks (PINN) for computational solid mechanics: Numerical frameworks and applications. Thin-Walled Struct 2024;205:112495. http://dx.doi.org/10.1016/j.tws.2024.112495.
[44] Yasuda H, Yamaguchi K, Miyazawa Y, Wiebe R, Raney JR, Yang J. Data-driven prediction and analysis of chaotic origami dynamics. Commun Phys 2020;3(1):168. http://dx.doi.org/10.1038/s42005-020-00431-0.
[45] Yasuda H, Yamaguchi K, Miyazawa Y, Wiebe R, Raney JR, Yang J. Data-driven prediction and analysis of chaotic origami dynamics. Commun Phys 2020;3(1):168. http://dx.doi.org/10.1038/s42005-020-00431-0.