A force neural network framework for structural optimization

Dai D. Mai[a,1], Si T. Do[a,b,2], Seunghye Lee[c,3], Hau T. Mai[d,*]

[a]*Faculty of Mechanical Engineering, Ho Chi Minh City University of Technology and Education, Ho Chi Minh City, Viet Nam*
[b]*Faculty of Electrical-Mechanical, FPT Polytechnic, FPT University, Ho Chi Minh City, Viet Nam*
[c]*Deep Learning Architectural Research Center, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea*
[d]*Faculty of Mechanical Engineering, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Viet Nam*

Abstract

In this paper, an efficient Force Neural Network (FNN) is developed to reformulate the size optimization of truss structures as an operator learning problem. A Deep Neural Network (DNN) is designed to directly map the connectivity information of truss members to the corresponding design variables. Therein, the entire unlabeled training data contains only the connectivity information of members, without any structural responses, weights, or cross-sectional areas. By integrating Force Method (FM), our framework embeds the optimal design problem represented by the objective and constraint functions in the loss function to guide the training process. And it guarantees that the generated solution is consistent with the underlying physical principles. In addition to enhance efficiency in finding the optimum structural weight, Bayesian Optimization (BO) is applied for automatic hyper-parameters tuning instead of the trial and error method. As soon as the training phase ends, the optimal weight of truss structures is found without using any other numerical methods. Several numerical examples are investigated to demonstrate the effectiveness and applicability of the FNN for the optimization of truss structures. The obtained results indicate that it not only be simple to perform but also overcomes the local optimal problem and reduces the computational cost in high-dimensional problems.

Keywords: Force neural network, Deep neural network, Structural optimization, Auto-tuning hyper-parameters, Bayesian optimization

*Corresponding author. *E-mail: maitienhau@iuh.edu.vn, maitienhaunx@gmail.com*
[1]*E-mail: daimd@hcmute.edu.vn*
[2]*E-mail: sidt3@fe.edu.vn, sidt.ncs@hcmute.edu.vn*
[3]*E-mail: seunghye@sejong.ac.kr*

1. Introduction

In the past decades, the optimization of truss structures has received widespread attention from many scholars [1, 2]. Generally, most current work relies on the same fundamental principle, as illustrated in Fig. 1a. Therein, numerical simulations are performed in each iteration of the optimization algorithm to estimate the structural responses. And these optimization algorithms can be grouped into three primary categories: Optimality Criteria (OC), gradient-based, and gradient-free algorithms. Firstly, the OC method employs heuristic updates based on optimality conditions for searching the optimal solution. And it has been successfully applied to handle optimization problems. For instance, Khot and Berke [3] introduced an efficient algorithm based on the OC for the sizing of structures. Besides, Bendsøe et al. [4] developed a displacements based OC method for truss topology design. Saka [5] applied the OC to design the shape of roof trusses. Although using the OC method had many benefits, the obtained results were sensitive to the initial starting point and the chosen parameters. Furthermore, it encountered the challenges in handling multiple constraints and local minima [6]. Next, the second baseline method is a gradient-based algorithm that relies on derivative information to guide the search process. For example, Gu et al. [7] developed a displacement-based optimization method to find the minimum weight of truss structures. A gradient-Hessian matrix-based algorithm was presented by Liu et al. [8] for minimizing the weight of truss structures. Additionally, Schmit and Farshi [9] suggested a succession of linear programs for sizing optimization of structures. To reduce computational costs, Saka and Ulker [10] developed a coupling mechanism based on nonlinear analysis technique and optimality criterion. Despite its remarkable success in the structural optimization, this approach still has limitations related to local optima and the lack of gradient information [11]. To circumvent the above drawbacks, gradient-free algorithms have received much attention for their ability to find near-optimal solutions. Storn and Price [12] firstly introduced a Differential Evolution (DE) algorithm for minimizing possibly nonlinear and non-differentiable continuous space functions. A genetic algorithm based on principles of biological evolution for solving optimization problems was suggested by Holland [13]. More recently, Lieu et al. [14] proposed a firefly algorithm for the optimization of truss structures. In addition, Rao et al. [15] presented teaching-learning-based optimization for solving mechanical design problems, while particle swarm optimization optimizing nonlinear functions was released by Kennedy and Eberhart [16]. Up to now, a variety of metaheuristic algorithms have been successfully developed for optimization [17–24]. However, these algorithms require a large number of structural analyses, become computationally challenging for large-scale problems, and have relatively slow convergence speeds [25].

In recent years, Machine Learning (ML) has been proven to be successful in a range of applications thanks to its ability to tackle complex problems lacking closed-form expressions. And the field of computational mechanics is no exception [26–35]. To the best of our knowledge, the applications of ML to structural optimization problems can be grouped into two main categories. The first one is a data-driven approach where the ML models are trained using pre-existing data to predict structural responses, optimize designs, or approximate solutions without relying on traditional physics-based simulations. Fig. 1b provides a comprehensive overview of the purely data-driven framework. Indeed, this methodology is not a new one and has been introduced since the 1990s. Specifically, Hajela and Berke [36] were among the pioneers in using Neural Networks (NNs) to replace structural analysis steps in the optimization process. And then a nonlinear neural dynamics model for optimization of structures was released by Adeli and Park [37]. Additionally, Ramasamy and Rajasekaran [38] introduced a combination between the genetic algorithm and NN for the design of industrial roofs. Recently, to reduce computational costs, Mai et al. [39] developed an integrated model combining the NN and DE for the design optimization of geometrically nonlinear structures. Besides, Li et al. [40] proposed a non-iterative topology optimizer using ML for heat conduction structure design. The same idea was adopted by White [41] and Chi [42] to replace the finite element analysis for the topology optimization. Although it has achieved certain success in optimization applications [43–45], this strategy also faces several challenges as follows:

(i) The data-driven model is derived from the input-output relationship without relying on precise physical assumptions. Therefore, it requires a larger number of training data to achieve the desired accuracy. Furthermore, computational simulations, such as finite element analysis, are used to collect the available true data. Precisely for this reason, it poses a significant challenge in determining the quality and size of the training data [46].

(ii) In other words, the NN was trained to minimize the distinction between the provided data and predicted results as a loss function. And the physical laws and governing equations of structures were not directly considered in the training process. As a result, the model fails to ensure the physical laws and lacks the generality needed for addressing various optimization problems [47].

(iii) Moreover, an important aspect to highlight here is the choice of hyper-parameters. Many studies have emphasized that the obtained results heavily depend on the selected network architecture [46, 48]. Consequently, it often poses a challenge for tuning hyper-parameters without relying on user experience whilst still ensuring accuracy.

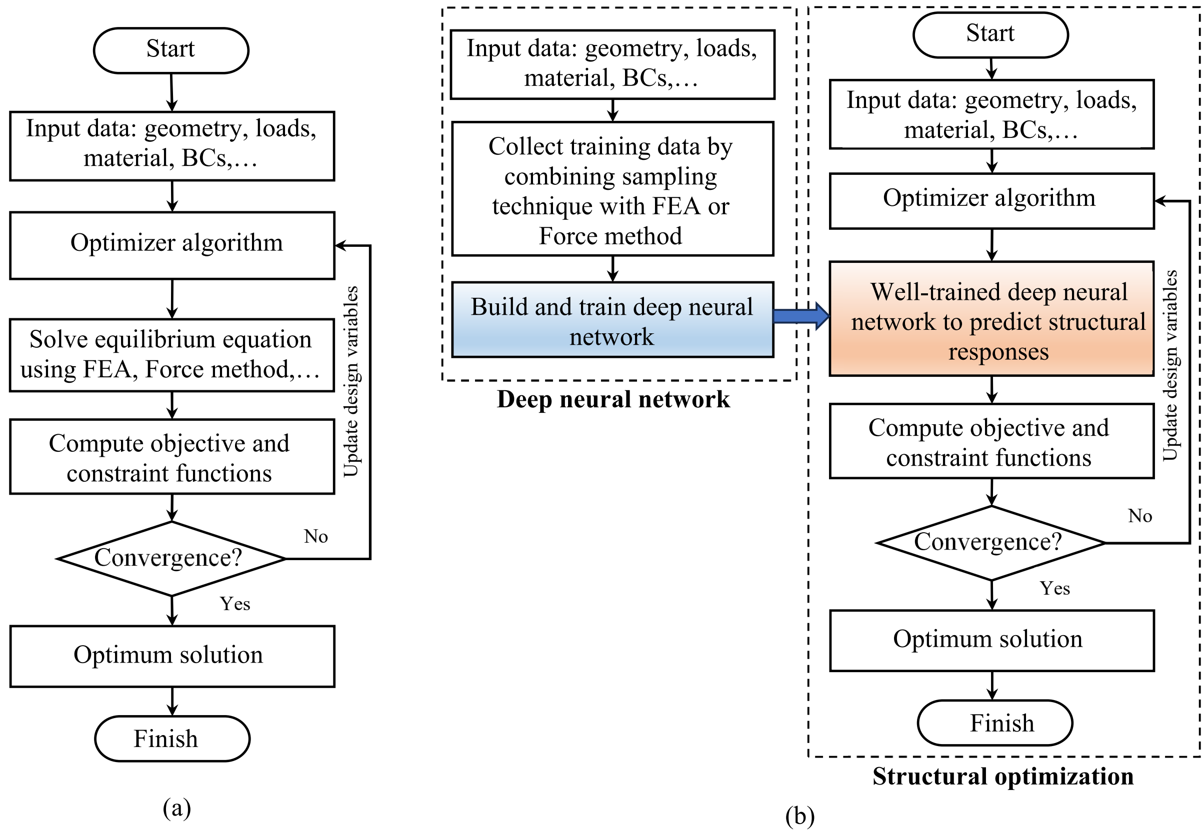
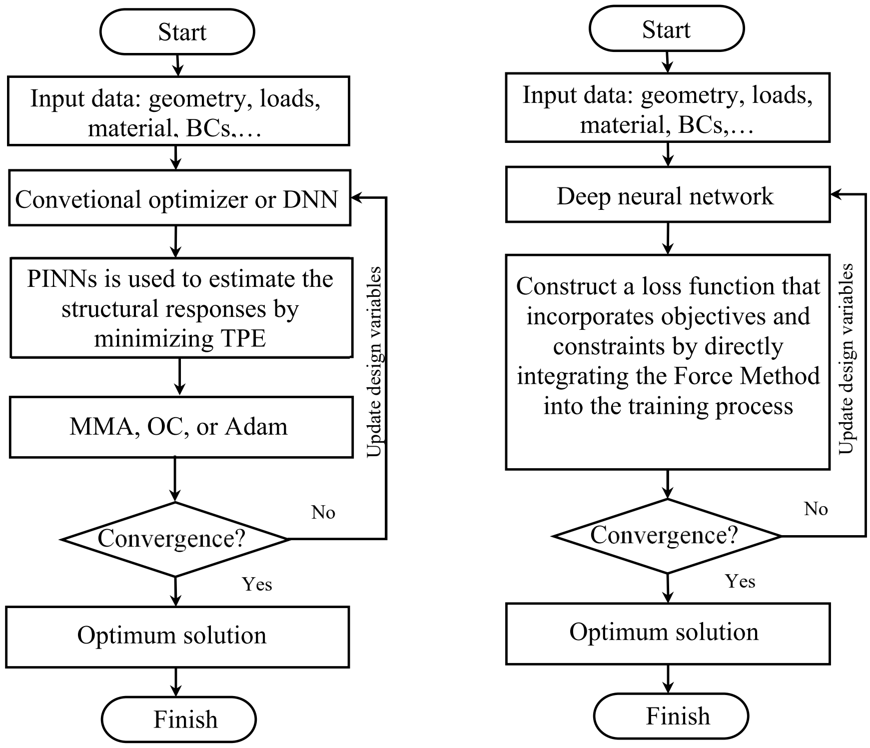


Fig. 1. Process of structural optimization. (a) Conventional approach including optimizer algorithm and structural analysis. (b) Purely data-driven model combines the optimizer algorithm and neural network.

In contrast, the second approach is Physics-Informed Neural Networks (PINNs), where physical laws represented by Partial Differential Equations (PDEs) were embedded in the loss function to guide the learning process. It has been proven to be successful in structural analysis

[49, 50] and solving PDEs [34, 35]. In recent literature, several scholars have successfully applied this approach for the structural optimization. Accordingly, He et al. [51] and Jeong et al. [52] were among the first authors to develop a approach for integrating PINNs-based simulation technique into classical topology optimization. In addition, two PINNs are designed to indicate optimized structures by Jeong et al. [53]. In recent times, Singh [54] introduced a dual PINNs for topology optimization. In the aforementioned studies, the networks are employed to replace structural and sensitivity analyses, as shown in Fig. 2a. Despite their success, this strategy also faces many challenges. First of all, instead of directly solving algebraic equations to estimate the structural responses, the network was trained to solve the energy minimization problem, and this inevitably leads to a large computational cost compared to classical approaches. On the other hand, the training data changes in each iteration of the optimization process. This may result in unstable numerical outcomes due to the changing potential energy landscape, while the hyper-parameters of the network remain fixed. Furthermore, it leads to inefficiencies in optimization performance in terms of both accuracy and computational cost for the dual PINNs. Motivated by this fact, our recent work proposes a Physics-Informed Neural Energy-Force Network (PINEFN) to solve the design optimization of truss structures. In this approach, a single neural network is utilized to minimize the loss function, which is derived from the weight, complementary energy, and constraint equations to determine the optimal solution. Despite its success in estimating the optimal weight, we observed that the training process converged only when the complementary energy of the structure was always positive. In other cases, the model did not converge to the optimal solution. And this can be interpreted as due to the complexity of the loss function when the network was designed to perform optimization of both the complementary energy and weight at the same time. According that core idea, the first term in the loss function was the Euclidean norm of the complementary energy, while the second and third terms related to the violated constraints and weight, respectively. It should be noted that the values of these terms as well as the loss function were always positive in the whole training process. Hence, the loss landscape becomes less smooth and converges to an unfavorable local minimum [55] when the minimum complementary energy of the structure is negative. Nevertheless, in aforementioned works, the choice of hyper-parameters is still a challenging issue

due to the complexity of the loss function.



(a) (b)

Fig. 2. Schematic diagram of structural optimization. (a) PINNs is used to replace FEM to obtain the structural responses. (b) Integrating structural analysis into neural network framework.

Driven by the challenges mentioned above, this study aims to introduce the force neural network framework for the size optimization of truss structures, as illustrated in Fig. 2b. Therein, the DNN is designed to directly estimate the optimal weight design. The trainable parameters, including the weights and biases of the network, are considered as design variables instead of the cross-sectional areas of truss members. The unlabeled training data only contains the connectivity matrix of truss elements. Meanwhile, the unknown cross-sectional areas are derived as output values of the network, which are expressed by the trainable parameters and the connectivity information. Based on the predicted cross-sectional areas, the weight and the corresponding constraint functions found by supporting FM are embed in the loss function of the network to guide the training process. Additionally, the BO framework is applied to automatically tune hyper-parameters of the network. When the training process ends, the optimum design is immediately indicated without using any additional algorithms. Several benchmarks are investigated to evaluate the reliability and efficiency of the proposed model. The obtained results of numerical examples are compared against several well-known recently introduced algorithms.

The main contributions of this study are as follows:

- Force neural network framework equipped with automatic sensitivity analysis capabilities offers simplicity, ease of use, and robustness for solving the optimization of truss structures under multiple constraints.
- Connectivity matrix of truss members is considered as a self-normalized and unlabeled training data without including any structural responses. Hence, it can be easily collected from the geometric information without using any numerical simulations or sampling techniques. In addition, the self-normalized data ensures more stable and efficient parameter updates during training.
- Automatic tuning of hyper-parameters using Bayesian optimization helps to escape the local optima as well as enhances reliability in design optimization.
- Our approach yields high accuracy, converges faster, and saves computational cost in high-dimensional problems compared to conventional optimization algorithms using finite element analysis.

The remainder of this study is organized as follows. In Section 2, a detailed introduction to the force neural network framework is provided. Therein, Section 2.1 presents the training data. While Section 2.2 provides the DNN architecture and loss function, Section 2.3 shows auto-tuning hyper-parameters. In Section 3, several case studies are conducted to demonstrate the accuracy and effectiveness of our model. Next, the efficiency of the proposed approach is discussed in Section 4. Finally, crucial conclusions are summarized in Section 5 of the article.

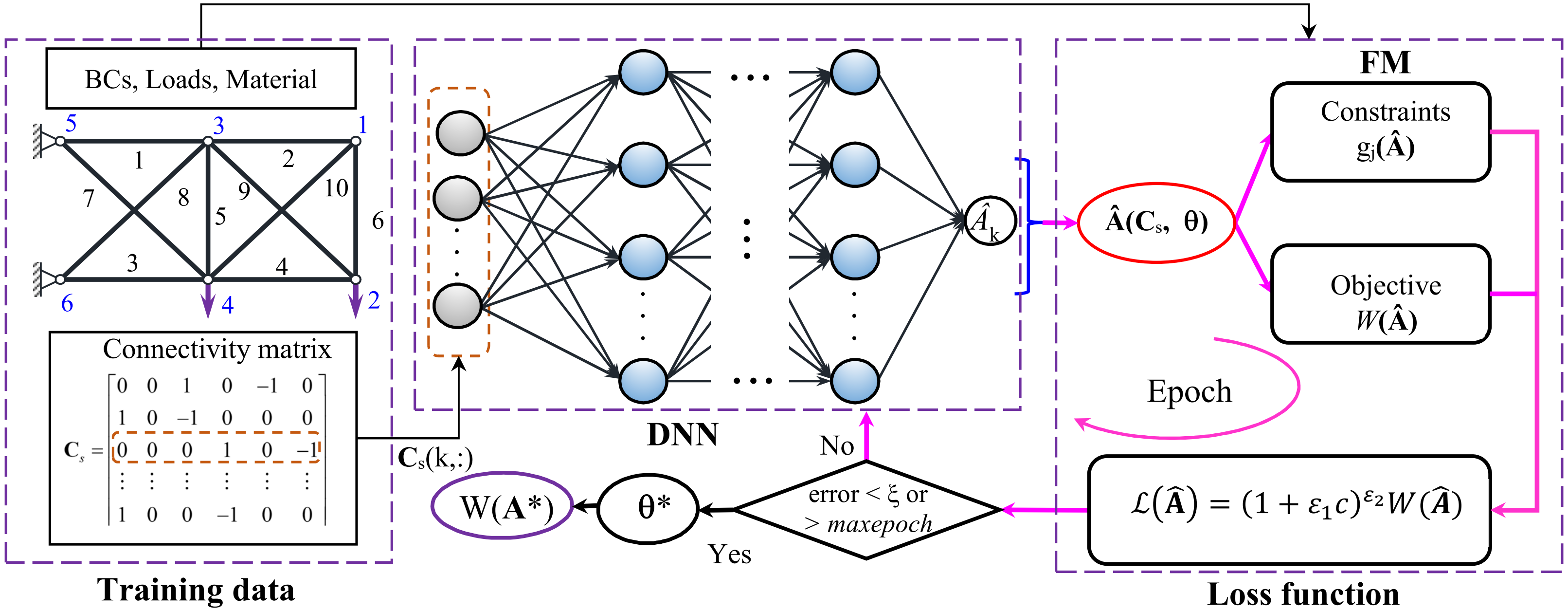


Fig. 3. Force neural network framework for design optimization.

2. Force neural network framework

In this section, the FNN, as shown in Fig. 3, is first introduced to directly perform optimization of truss structures. Therein, the trainable parameters of the network $\boldsymbol{\theta}$ are treated as new design variables instead of the cross-sectional areas. The entire training data contains only the connectivity matrix $\mathbf{C}_s$, which is unlabeled, self-normalized, and without any structural responses. And each row of this matrix $\mathbf{C}_s\,(i,:)$ represents a sample of the training data which is known as an input vector to the neural network. The predicted cross-sectional areas $\hat{\mathbf{A}}$, which are referred to as the output network, are represented as a function of $\mathbf{C}_s$ and $\boldsymbol{\theta}$ through the mapping of the DNN. According to this scheme, the objective and constraints corresponding to the predicted output values $\hat{\mathbf{A}}$, as determined by FM, are embedded into the loss function to guide the network's training process in searching for the optimal structure. To achieve this goal, the network is trained by adjusting the parameters to minimize the loss function of the model. The training phase becomes easy and simple to implement with automatic sensitivity analysis capabilities. Additionally, to enhance the computational efficiency and reliability of the model, the BO is applied to automatically tune hyper-parameters of the network. In general, our framework comprises three fundamental components: training data, DNN architecture, and auto-tuning hyper-parameters. The following subsections provide a detailed description of them.

2.1. Training data

Unlike previous work based on data driven approaches, our unlabeled training data only contains the input data without corresponding output values. More concretely, the connectivity matrix of truss elements $\mathbf{C}_s(\in\ \mathbb{Z}^{el\times n})$ is set up as the entire training data, which does not include the responses of the structure, such as stress, strain, displacement, force members, cross-sectional areas, and so on. Here, el is the number of elements, while n denotes the number of joints. And the value of the kth row and pth column of the connectivity matrix $\mathbf{C}_s$, which shows connecting the nodes i and j $(i<j)$ of the kth member, is defined as follows

$$\mathbf{C}_{s(k,p)} = \begin{cases} 1 & \text{for } p = i, \\ -1 & \text{for } p = j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

And it is evident that the connectivity matrix can be obtained easily from the structure's geometric information without requiring numerical simulations or sampling techniques. It is worth mentioning that from Eq. 1, the self-normalized training data is the connectivity matrix whose entries are -1, 0 or 1. And this brings significant benefits to the efficiency of model training. Firstly, the normalized data helps reduce vanishing or exploding gradient issues and allows for faster convergence during training. Besides, all self-normalized inputs are given the same relevance or scale, ensuring that each feature contributes equally to making predictions. This reduces instability during forward and backward propagations as well as improves the accuracy and generalization capability of the network. Finally, the self-normalized inputs can reduce the network's sensitivity to the hyper-parameters [56, 57]. Furthermore, the cross-sectional areas, which are not included in the training data and are unknown quantities, are designed as the network's output. The important thing that must be highlighted here is that the objective and constraints of the structure are determined based on the predicted values of the network with supporting FM.

2.2. Deep Neural network

One of the machine learning models is the DNN, which is a set mathematical relationship between inputs and outputs developed during a training phase to replicate the way human brain operations work. A fully connected DNN with $(L + 1)$ layers, as depicted in Fig. 4, is constructed to parameterize the cross-sectional areas $\hat{\mathbf{A}}$. It comprises of one input layer with n input neurons and one output layer with one output neuron. Between these two layers, there are $(L - 1)$ hidden layers, and the choice of the number of hidden neurons and hidden layers depends on the complexity of specific problems. In this study, the BO algorithm is applied to automatically optimize them. Note that all units of the current layer are linked to every neuron in the next layer via the training parameters $\boldsymbol{\theta}$, including the weights and biases. And these initial parameter values are randomly generated using the truncated normal distribution in the range [-1, 1]. Accordingly, the predicted cross-sectional area of the ith element $\hat{A}_i$ is expressed as follows

$$
\begin{aligned}
&\text{input layer} \quad : \ \mathbf{h}^0 = \mathbf{C}_s\left(i,:\right) \ \in \mathbb{R}^n, \\
&\text{hidden layers: } \mathbf{h}^l = f_1\left(\mathbf{W}^{l^T}\mathbf{h}^{(l-1)} + \mathbf{b}^l\right) \in \mathbb{R}^{m_l}, \quad \text{for } 1 \leq l < L, \\
&\text{output layer } : \ \mathbf{h}^L = f_2\left(\mathbf{W}^{L^T}\mathbf{h}^{(L-1)} + \mathbf{b}^L\right) = \hat{A}_i \ \in \mathbb{R},
\end{aligned}
\tag{2}
$$

where $\mathbf{h}^l(.)$ is output vector of the lth layer; m_l is the number of units in the lth hidden layer; $\mathbf{W}^{(.)}$ and $\mathbf{b}^{(.)}$ denote the weights and biases, respectively; $f(.)$ is the activation function, which enables the network to learn the complex relationship between the output and input. Several activation functions, such as ReLU, LeakyReLU, Tanh, Sigmoid, Linear, Softmax, and so on, are widely used to solve various problems. Note that this study utilized the Sofmax function for the output layer, whilst the activation function of the hidden layers is identified through BO, which will be explained in detail in the next subsection.

From Eq. 2, it should be noted that the cross-sectional areas $\hat{\mathbf{A}}\left(\mathbf{C}_s, \boldsymbol{\theta}\right)$ are the function of the training parameters and the connectivity matrix. Therefore, the weights and biases of the network are now new design variables of the sizing optimization of truss structures, instead of the cross-sectional areas of truss members as in conventional approaches. In this study, the weight of the structure is minimized subject to the displacement and stress constraints. The optimal design problem can be formulated as follows

$$
\begin{aligned}
\text{Minimize} \quad & W\left(\hat{\mathbf{A}}\left(\mathbf{C}_s, \boldsymbol{\theta}\right)\right) \ = \sum_{i=1}^{el} \rho_i L_i \hat{A}_i\left(\mathbf{C}_s(i,:), \boldsymbol{\theta}\right), \quad i = 1,\ 2,\ ...,\ el\,, \\
\text{subjected to} \quad & \delta_{\min} \leq \delta_j\left(\mathbf{C}_s, \boldsymbol{\theta}\right) \ \leq \ \delta_{\max}\,, \ j = 1,\ 2,\ ...,\ n_d\,, \\
& \sigma_{\min} \leq \sigma_i\left(\mathbf{C}_s, \boldsymbol{\theta}\right) \ \leq \ \sigma_{\max}\,, \ i = 1,\ 2,\ ...,\ el\,, \\
& \sigma_k^b \leq \sigma_k\left(\mathbf{C}_s, \boldsymbol{\theta}\right) \ \leq \ 0\,, \qquad k = 1,\ 2,\ ...,\ n_b, \\
& A_i^{low} \leq \hat{A}_i\left(\mathbf{C}_s(i,:), \boldsymbol{\theta}\right) \leq A_i^{up},
\end{aligned}
\tag{3}
$$

where $W(.)$ is the weight of the whole truss structure; $\hat{\mathbf{A}}$ denotes the predicted cross-sectional area vector; $\hat{A}_i$ is the predicted cross-sectional area of the ith member; ρ_i and L_i are the material density and length of the ith member, respectively; el is the total number of bars in the structure; n_d refers to the number of displacement constraints; n_b denotes the number of compression elements; δ and σ are the nodal deflection and the stress, respectively; σ_k^b is the allowable buckling stress in the kth member when it is in compression; A_i^{low} and A_i^{up} are the lower bound

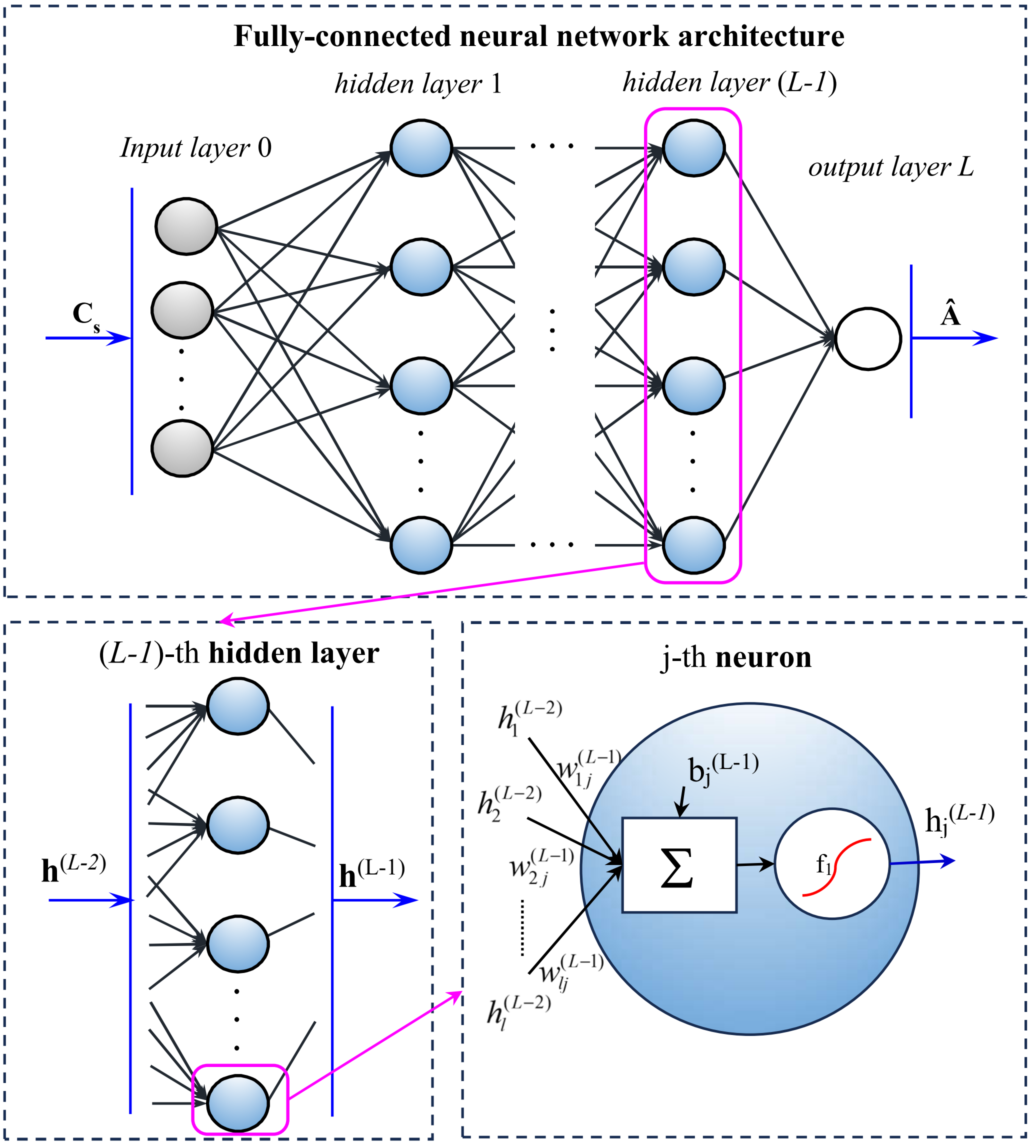


Fig. 4. A fully-connected deep neural network architecture.

and the upper bound of the ith cross-sectional area, respectively.

With respect to the predicted cross-sectional areas of the network, the weight of the truss structure can be easily obtained, while the structural responses, including displacements and stresses, are found by the FM. Accordingly, the objective and constraint values are embed into a penalty function, also known as the loss function of the network, to guide the learning process. Meanwhile, the constrained optimization problem is converted into an unconstrained optimization one. And Eq. 3 is rewritten as follows

$$\text{Minimize} \quad \mathcal{L}(\boldsymbol{\theta}) = \left(1 + \varepsilon_1 c\left(\hat{\mathbf{A}}\left(\mathbf{C}_s, \boldsymbol{\theta}\right)\right)\right)^{\varepsilon_2} W\left(\hat{\mathbf{A}}\left(\mathbf{C}_s, \boldsymbol{\theta}\right)\right), \tag{4}$$

with

$$c\left(\hat{\mathbf{A}}\left(\mathbf{C}_s,\boldsymbol{\theta}\right)\right)=\sum_{j=1}^{n_c}\max\left(0,g_j\left(\hat{\mathbf{A}}\left(\mathbf{C}_s,\boldsymbol{\theta}\right)\right)\right), \tag{5}$$

in which c is the sum of the violated constraints; n_c is the number of constraints in the problem; g_j represents the jth constraint function; ε_1 and ε_2 are parameters which control the exploration and exploitation factors of the design domain. Herein, the parameter ε_2 is set equal to 1, as suggested by Hasancebi [58] and Sonmez [59]. The other parameter ε_1 adjusts itself dynamically according to the feedback from the previous iteration and is defined as follows

$$\varepsilon_1^{(t)}=\begin{cases}(1/\kappa)\,\varepsilon_1^{(t-1)} & \text{if } \mathcal{L}^{(t-1)} \text{ feasible,}\\ \kappa\,\varepsilon_1^{(t-1)} & \text{if } \mathcal{L}^{(t-1)} \text{ infeasible,}\end{cases} \tag{6}$$

where $\varepsilon_1^{(t)}$ represents the penalty coefficient at the tth iteration, with $\varepsilon_1^{(1)}$ initially set at 1. The learning parameter for $\varepsilon_1^{(t)}$, denoted as κ is determined by

$$\kappa=1+\frac{1}{n_c+1}>1.01. \tag{7}$$

It is worthwhile to note that the constraints obtained by FM are consistent with the underlying physical principles and makes the total complementary energy minimum in each iterations. This is a significant difference between the proposed approach and our previously work PINEFN [60]. In addition, note that the output layer uses the Softmax function to limit the output range between 0 and 1. Based on these output network, all predicted cross-sectional areas are renormalized into the design space $[A_i^{low}, A_i^{up}]$. Thus, the constraints related to the limitations of the design variables are removed in Eq. 4. This is meaning that the constraints (g_j) only include the displacements and stresses, which satisfy both equilibrium and compatibility equations. To achieve this goal, the training phase, also known as structural optimization, aims to minimize the loss function in order to determine the network's optimal parameters instead of the cross-sectional areas.

$$\boldsymbol{\theta}^*=\arg\min_{\boldsymbol{\theta}}\left(\,\mathcal{L}\left(\boldsymbol{\theta}\right)\,\right). \tag{8}$$

In order to achieve the goal, Adam optimizer, which is a well-known gradient descent algorithm, is utilized in this study to perform the training task. Therefore, the derivatives of the loss function must be determined to adjust the training parameters. By applying the chain rule to Eq. 4, the sensitivity of the loss function is expressed as follows

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \sum_{j=1}^{el} \left[\varepsilon_2 (1+\varepsilon_1 c)^{\varepsilon_2 - 1} W \frac{\partial c}{\partial \hat{A}_j} + (1+\varepsilon_1 c)^{\varepsilon_2} \frac{\partial W}{\partial \hat{A}_j} \right] \frac{\partial \hat{A}_j}{\partial \theta_i}. \tag{9}$$

From Eq. 9, it can be observed that the second term $\frac{\partial \hat{A}_j}{\partial \theta_i}$ is calculated automatically by the backpropagation algorithm which is integrated into the network. Therein, the remaining term $\frac{\partial W}{\partial \hat{A}_j}$ can be easily determined using the formulation following

$$\frac{\partial W}{\partial \hat{A}_j} = \rho_j L_j, \tag{10}$$

where L_j and ρ_j are the length and material density of the jth truss member, respectively. And the gradient of term $\frac{\partial c}{\partial \hat{A}_j}$ is calculated using Just Another eXtensor (JAX) [61] which is a tool for automatic differentiation developed by Google. It has been successfully applied in computational mechanics fields [62, 63]. Consequently, the sensitivity of the loss function with respect to the training parameters is entirely defined. When the trainable parameters at iteration $(t+1)$ of the training process are adjusted as follows

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\mathbf{m}_{t+1}\sqrt{1-\beta_2^{(t+1)}}}{\left(1-\beta_1^{(t+1)}\right)\left(\sqrt{\mathbf{v}_{t+1}} + \xi\sqrt{1-\beta_2^{(t+1)}}\right)}, \tag{11}$$

in which $\mathbf{m}_{t+1}$ and $\mathbf{v}_{t+1}$ are given by

$$\begin{aligned} \mathbf{m}_{t+1} &= \beta_1 \mathbf{m}_t + (1-\beta_1) . \nabla \mathcal{L}(\boldsymbol{\theta}_t), \\ \mathbf{v}_{t+1} &= \beta_2 \mathbf{v}_t + (1-\beta_2) . \nabla \mathcal{L}(\boldsymbol{\theta}_t), \end{aligned} \tag{12}$$

where β_1 and β_2 are the exponential decay rates which are used to control the first $\mathbf{m}_{t+1}$ and second $\mathbf{v}_{t+1}$ raw moment vectors; η and ξ denote the learning rate and constant added to ensure numerical stability, respectively. In this work, the default settings of the Adam, as suggested by Kingma and Ba [64], were used to train the model. For more information, the readers can refer to [64]. Once the training process is completed, the optimum structural weight corresponding to the optimal parameters of the network is found.

2.3. Auto-tuning hyper-parameters

The parameters associated with the network architecture and training procedure, which are known as hyper-parameters, cannot be directly estimated from the training data but must be set before the learning process. They play a paramount role in enhancing the effectiveness of employing the neural network for real-world applications. Nevertheless, identifying the optimal hyper-parameters encounters difficulties due to the lack of a closed-form expression for the Hyper-parameter Optimization (HPO) problem. And it is described as an expensive black-box problem when searching for extrema. Therefore, conventional algorithms are not suitable for implementing such tuning tasks. More concretely, the gradient-based algorithms are inadequate for solving this problem because the gradient information is not available. Meanwhile, the gradient-free algorithms normally require a large number of training times, which is infeasible for computationally expensive problems. In addition, the grid and random search techniques are usually employed to select the optimal hyper-parameters. However, the grid search trains all possible permutations of hyper-parameters, which can result in training the network for a very long time. Meanwhile, the random search cannot cover the entire parameter space. And a major setback for both techniques is that they are completely unaware of previous evaluations [65]. To overcome this computing challenge, optimization techniques based on surrogate models were suggested for handling expensive optimization problems. Among the different surrogate modeling techniques, the Bayesian optimization algorithm is known as a popular and powerful tool for searching the best combination of hyper-parameters of neural networks [66]. It has demonstrated efficiency and robustness in automatic hyper-parameters tuning of the machine learning models. Thus, the BO is chosen to find the optimal network. Accordingly, the hyper-parameters tuning is posed as an unconstrained optimization problem, which can be expressed as follows

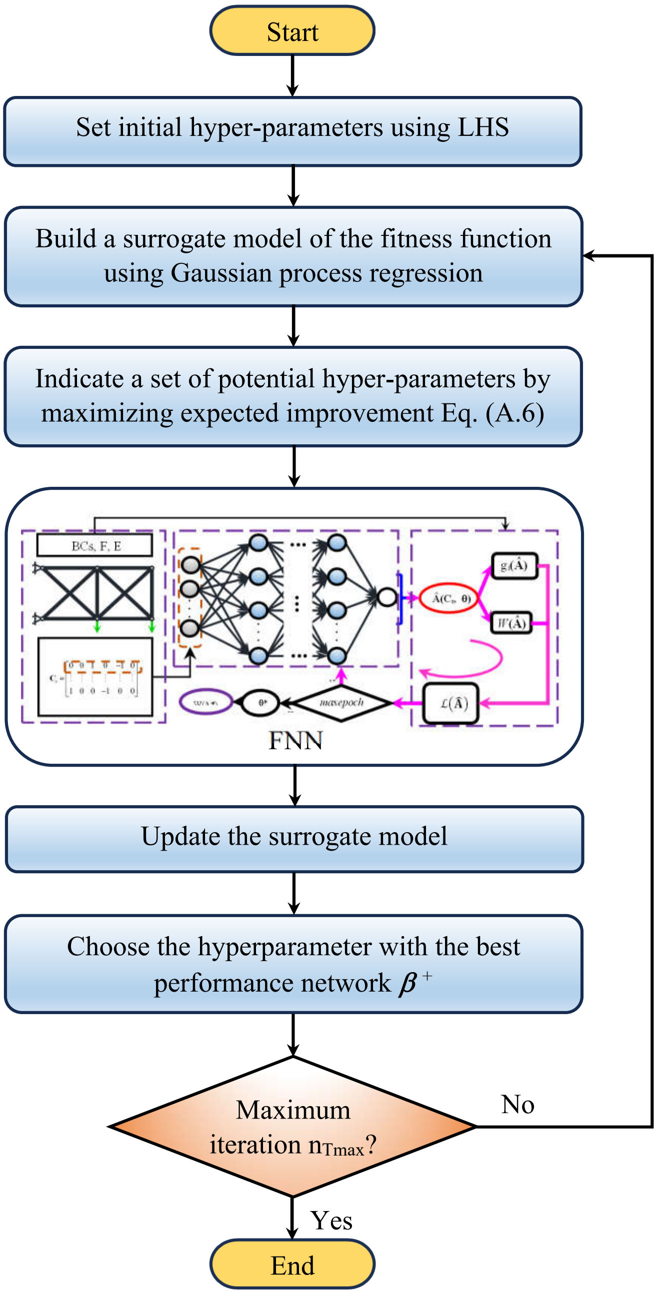


Fig. 5. Schematic of FNN framework into Bayesian optimization for structural optimization.

$$\beta^* = \underset{\beta \in \Omega}{\arg\min} \ \mathcal{L}_{\min}(\beta), \tag{13}$$

where $\mathcal{L}_{\min}(\beta)$ denotes the minimum loss function value found by the network corresponding to the hyper-parameter vector β. And it is regarded as the objective function for tuning hyper-parameters. In order to find the optimal network, a Gaussian Process (GP) as a surrogate model is constructed to approximate this unknown function. And the acquisition function, also known as the infill strategy, is utilized to guide the search of hyper-parameters. In this study, three standard acquisition functions, including Lower Confidence Bound (LCB), Probability of Improvement (PI), and Expected Improvement (EI), are considered to compare and evaluate the performance of BO. Details regarding the derived Bayesian formulation of the Gaussian process model for tuning hyper-parameters are provided in Appendix A.

Fig. 5 illustrates an overall schematic of the suggested framework, which includes two loops. Therein, the inner loop, as shown in Algorithm 1, represents the training phase of the FNN to identify the minimum loss function corresponding to the hyper-parameters. And its basic workflow is summarized as follows:

Step 1: Firstly, the connectivity matrix of truss members, which can be easily collected from the geometric information of structures, is set as the entire training data for the first step.

Step 2: Next, a neural network is built using the hyper-parameters suggested by BO in Algorithm 2. Therein, all trainable parameters $\boldsymbol{\theta}_0$ are initialized using truncated normal distribution in the range [-1, 1], and updated by using Adam optimizer

Step 3: Calculation of the predicted cross-sectional areas $\hat{\mathbf{A}}$ using the feedforward propagation with Eq. (2).

Step 4: Substitution of the values of $\hat{\mathbf{A}}$ into Eq. (3) yields the weight of truss structure, which is known as the objective function of the structural optimization.

Step 5: Force method is employed to estimate the constraints, including the displacements $\boldsymbol{\delta}$ and stresses $\boldsymbol{\sigma}$ corresponding to the predicted cross-sectional areas $\hat{\mathbf{A}}$.

Step 6: Substitution of the values of the weight and constraint values into Eq. (4) achieves the loss function.

Step 7: The gradients of the loss function with respect to the parameters by using Eq. (10), JAX, and backward propagation.

Step 8: The trainable parameters of the network are updated by Eq. (11).

Step 9: The training task ends when either the norm of the residual gradient of two consecutive epochs $\|\nabla\mathcal{L}(\boldsymbol{\theta}_t) - \nabla\mathcal{L}(\boldsymbol{\theta}_{t-1})\|$ must not be greater than 10^{-2} in the last 15 epochs ($n_{wmax} = 15$) or the maximum number of epoch $epoch_{max}$ reaches. If the criterion is not satisfied, then return to step 3; otherwise, stop the training process.

Algorithm 1: Force neural network for structural optimization

---

Input:
- Structure: material properties, geometry, boundary conditions, loads
- NN: hyper-parameters $\boldsymbol{\beta}$, Adam optimizer

Output: optimal parameters $\boldsymbol{\theta}^*$, optimum weight of truss structure W

```
Calculate the connectivity matrix C_s by Eq. 1
Construct a NN with initial parameters θ_0 distributed in the range [-1, 1]
Set the parameters of Adam optimizer as the default settings [64]
while n_f < n_wmax or epoch_max is not reached do
    Predict Â(C_s, θ_t) using the feedforward propagation
    Compute the weight of truss structure W(Â(C_s, θ_t)) by Eq. (3)
    Calculate the displacement δ(C_s, θ_t) and stress σ(C_s, θ_t) by FM
    Loss function L(θ_t) is estimated by Eq. (4)
    ∂W/∂Â_j is calculated by Eq. (10)
    ∂c/∂Â_j is computed by the automatic differentiation JAX
    ∂Â_j/∂θ_t is calculated automatically by the backward propagation
    Update trainable parameters θ_{t+1} of the network by Eq. (11)
    If ||∇L(θ_t) − ∇L(θ_{t−1})|| < 10^{−2} then n_f = n_f + 1
    t=t+1
```

Subsequently, the minimum loss value obtained by the Algorithm 1 is forwarded to the outer loop which allows tuning the hyper-parameters of the network using the BO algorithm. Clearly, the objective of the outer loop is to pinpoint the hyper-parameters that yield the best minimum weight of the truss structure with respect to the best minimum loss function value, as shown in Algorithm 2. The fundamental stages of this algorithm is described as follows:

Step 1: Firstly, Latin Hypercube Sampling (LHS) technique is used to collect a set of initial combination of hyper-parameters $\boldsymbol{\beta}_{1:p}$ from the design domain.

Step 2: Based on the above set of hyper-parameters, FNN is trained to estimate the corresponding minimum loss function values $\boldsymbol{\mathcal{L}}_{\min_{(1:p)}}$ by Algorithm 1.

Step 3: Next, a set of initial observations $\mathcal{D} = \left\{\boldsymbol{\beta}_{1:p}, \boldsymbol{\mathcal{L}}_{\min_{(1:p)}}\right\}$ containing the hyper-parameters and the corresponding minimum loss function values is collected.

Step 4: The surrogate model based on the Gaussian process model is built on $\mathcal{D}$.

Step 5: A next potential hyper-parameter configuration $\boldsymbol{\beta}_{n+1}$ is found by maximizing the acquisition function Eq. (A.6).

Step 6: FNN with respect to the new sample point $\boldsymbol{\beta}_{n+1}$ is trained to evaluate the minimum loss function $\mathcal{L}_{\min_{n+1}}$ by Algorithm 1.

Step 7: The new data point $\left(\boldsymbol{\beta}_{n+1}, \mathcal{L}_{\min_{n+1}}\right)$ is appended to the existing data $\mathcal{D}$.

Step 8: Check the stopping criterion ($n \leq n_{Tmax}$). If the the stopping criterion is not satisfied, go to step 4, else the solution with the best weight W^*_{min} corresponding the optimal hyper-parameters $\boldsymbol{\theta}^*$

3. Numerical examples

In the following section, several numerical examples are investigated to verify and evaluate the capability of the proposed framework for sizing optimization of truss structures. For this purpose, the obtained results will be compared with the conventional algorithms, such as DE, Particle Swarm Optimizer (PSO), PSO with passive congregation (PSOPC), Heuristic PSO (HPSO), Harmony Search, Teaching-Learning-Based Optimization, Big Bang–Big Crunch, and recently published results using the machine learning models like Deep Unsupervised Learning (DUL), and PINEFN. To enhance the reliability and computational efficiency of the network, the BO algorithm is utilized for the automatic hyper-parameters tuning. In order to get the best possible network, the initial number of hyper-parameter sets (p) used to build the initial surrogate model is set to 10, while the total number of hyper-parameter combinations (n_{Tmax}) evaluated throughout the entire BO process is set to 30 in all examples. In this work, two types of hyper-parameters are chosen to fine-tune the model: one was related to the network structure, including the number of hidden layers, neurons, and activation function, and the other was associated with adjusting the learning rates. The allowed ranges of values for each hyper-parameter are listed in Table 1. Note that the number of hidden neurons and activation functions are the same for all hidden layers. In addition, SoftMax and Adam are adopted as the activation function for the output layer and optimizer during the performance process. Furthermore, the training process of the FNN concludes when either the maximum allowed number of epochs is reached, or the norm of the gradient value is less than 0.01 in the last 15 epochs $(n_{wmax} = 15)$ [62, 67]. To evaluate the influence of uncertain quantities, HPO is executed through thirty independent runs with different initial points.

Algorithm 2: Automatic tuning of DNN hyper-parameters using Bayesian optimization

Input:

-p : initial number of hyper-parameter sets

-n_{Tmax}: maximum number of training times

Output: optimal hyper-parameters $\boldsymbol{\beta}^*$, best weight of truss structure W^*_{min}

LHS is used to collect hyper-parameters $\boldsymbol{\beta}_{1:p}$ from the design domain

Training the network corresponding to $\boldsymbol{\beta}_{1:p}$ to estimate the minimum loss function values $\boldsymbol{\mathcal{L}}_{\min_{(1:p)}}$

Collect a set of initial observations $\mathcal{D} = \left\{\boldsymbol{\beta}_{1:p}, \boldsymbol{\mathcal{L}}_{\min_{(1:p)}}\right\}$

Current best combination of hyper-parameters $\boldsymbol{\beta}^+ = \underset{\boldsymbol{\beta}_i \in \mathcal{D}}{\arg\min}\ \mathcal{L}_{min}\left(\boldsymbol{\beta}_i\right)$

Set $n = p$

while $n \leq n_{Tmax}$ **do**

- GP's parameters are found by maximizing likelihood function
- Build the GP model on $\mathcal{D}_n$
- Find $\boldsymbol{\beta}_{n+1}$ by maximizing Eq. (A.6)
- Training FNN with the hyper-parameters $\boldsymbol{\beta}_{n+1}$ to evaluate $\mathcal{L}_{\min_{n+1}}$ by Algorithm 1
- Append $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \left\{\left(\boldsymbol{\beta}_{n+1}, \mathcal{L}_{\min_{n+1}}\right)\right\}$
- Estimate $\boldsymbol{\beta}^*$
- Update $\boldsymbol{\beta}^+ = \boldsymbol{\beta}^*$
- $n = n + 1$

Meanwhile, the parameters of the DUL, PINEFN, and DE algorithms are set similar to Hau et al. [60, 68]. Due to the stochastic nature of the metaheuristic algorithm, the best result is determined through 30 independent runs to ensure the reliable solution of the DE. To get an unbiased comparison of the different models, all numerical examples were executed on a personal computer utilizing the Pytorch library in the Python language. Furthermore, all computations were performed on a desktop PC equipped with an Intel Core i5-8500 CPU running at @ 3.0 GHz, 16 GB of RAM, and Windows 10.

Table 1
Configuration space for the hyper-parameters of the network.

Hyper-parameter	Search space	Type
No. of hidden layers	[1, 4]	Integer
No. of hidden neurons	[20, 60]	Integer
Activation function	[ReLU, Sigmoid, Softmax, Tanh, LeakyReLU]	Categorical
Learning rate	[0.001, 0.1]	Real
Step size	[2, 10]	Integer
Gamma	[0.05, 0.8]	Real

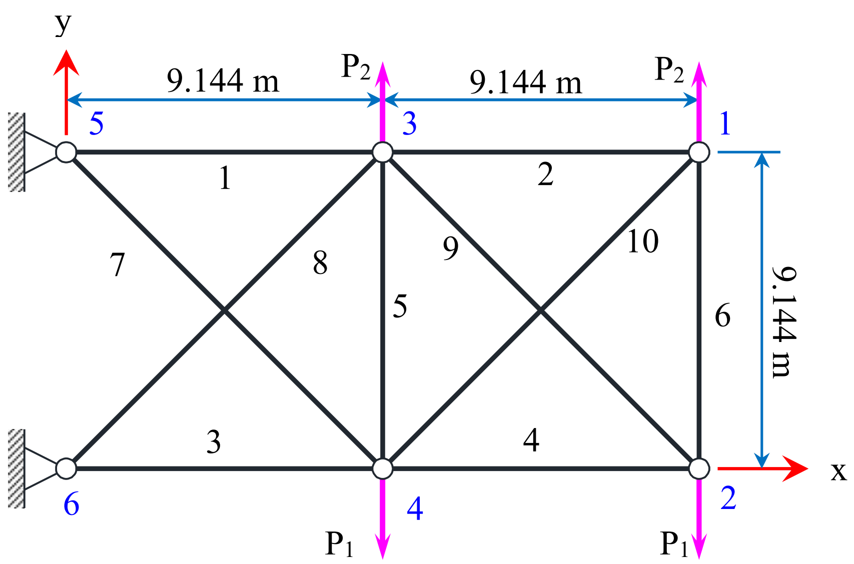


Fig. 6. A 10-bar planar truss structure.

3.1. 10-bar truss

A ten-bar planar truss structure, subjected to two loading conditions as shown in Fig. 6, is examined as the first design optimization problem. The loading conditions are as follows: (1) the first condition with P_1 = 444.822 kN and P_2 = 0 kN; (2) the other condition with P_1 = 667.233 kN and P_2 = 222.411 kN. The cross-sectional areas of truss members, which are considered as continuous design variables, have their minimum values specified at 0.645 cm^2.

All members are made of a material with an elastic Young's modulus of 68947.573 MPa, mass density of 27679.905 kg/m^3, and allowable stresses of 172.369 MPa in tension and compression. In addition, the displacements of free nodes are restricted to ± 5.08 cm in all directions. In both loading cases, the network performs training with the maximum epoch size of 1000 as a stopping criterion. Furthermore, all infill strategies within the BO framework utilize the same set of 10 initial hyper-parameter configurations for a fair comparison.

Table 2
Statistics of the optimal weight (kg) with different acquisition functions for the 10-bar planar truss (Case 1).

Metric	Acquisition functions		
	PI	LCB	EI
Min	2295.855	2295.831	2295.655
Max	2296.907	2296.716	2295.917
Mean	2296.280	2296.353	2295.749
Std	0.118	0.102	0.027
95% CIU	2296.303	2296.373	2295.754
95% CIL	2296.257	2296.333	2295.744

Table 3
Optimum hyper-parameters of the network obtained using the BO with different acquisition functions for the 10-bar planar truss (Case 1).

Acquisition function	Hyper-parameter					
	No. of hidden layers	No. of hidden neurons	Learning rate	Activation function	Step size	Gamma
PI	3	60	0.100	ReLU	2	0.050
LCB	4	25	0.010	ReLU	8	0.500
EI	3	60	0.022	ReLU	5	0.158

For the first loading case, a comparison of the statistics of the optimal weight, including minimum (Min), maximum (Max), mean, standard deviation (Std), 95% confidence interval upper (95% CIU), and lower (95% CIL) bounds found by the network using various infill strategies, are summarized in Table 2. Additionally, Table 3 presents the optimal hyper-parameters of the network corresponding to the best weight. Firstly, it is easily seen that the best optimal weight obtained by various acquisition functions are in good agreement. Although there were not significant differences between the minimum weights, the EI infill strategy identified the lightest design overall (W$_{min}$ = 2295.655 kg; Std = 0.027 kg; 95% CI [2295.754, 2295.744] kg), followed by the LCB (W$_{min}$ = 2295.831 kg; Std = 0.102 kg; 95% CI [2296.373, 2296.333]

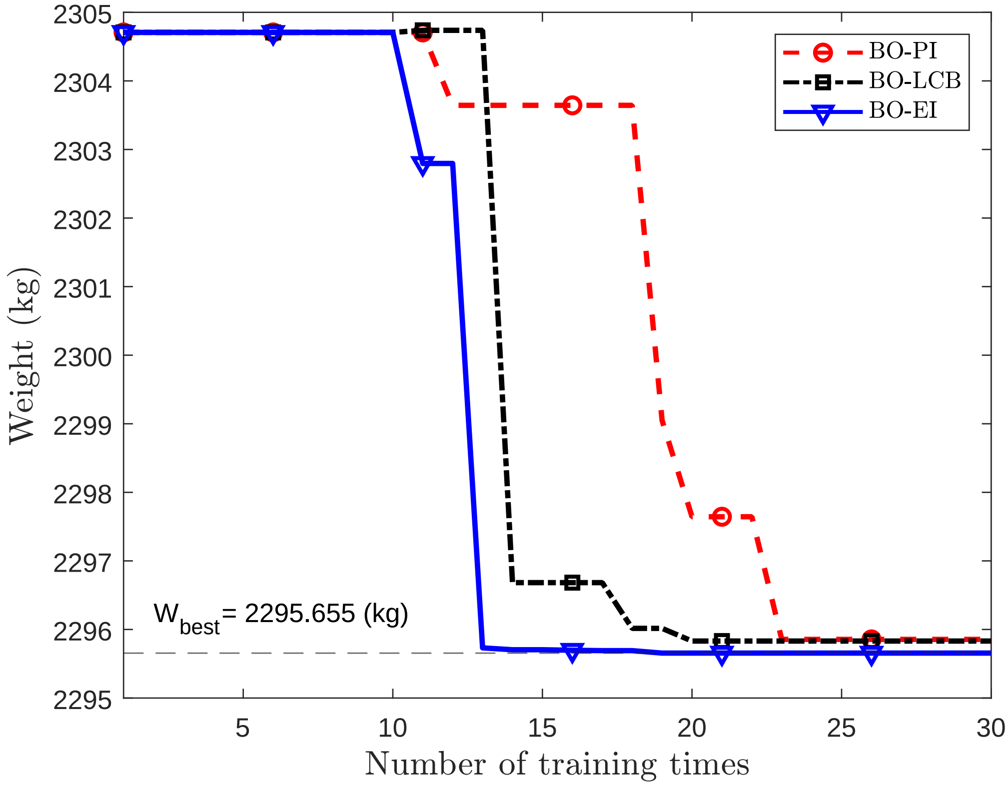


Fig. 7. Convergence histories of the HPO using BO for the 10-bar planar truss (Case 1).

kg), and then the PI (W_{min} = 2295.855 kg; Std = 0.118 kg; 95% CI [2296.303, 2296.257] kg). In addition, the mean value (2295.749 kg) is very close to the 95% CIU and 95% CIL with the smallest Std, and this indicates the high reliability of the EI infill strategy in identifying the optimal hyper-parameters of the network. From the data in Table 3, it is observed that although both EI and PI indicate a similar architecture network, there are different parameters associated with learning rates. And this shows the significant role of adjusting the learning rate for fitting the neural network. Besides, the ReLU activation function, identified by all infill strategies, possesses salient advantages, such as computationally efficiency, fast convergence, parameter-free, and helping to prevent gradient saturation [69, 70]. Finally, the convergence histories of the optimal hyper-parameters tuning process using different infill strategies are depicted in Fig. 7 for the first loading case. Note that all convergence curves coincide during the first 10 iterations because they use the same ten initial hyper-parameter sets generated by the LHS to ensure a fair comparison between the infill strategies. It is obvious that the EI demonstrates its efficiency and faster convergence in tuning the hyper-parameters with the best minimum weight design compared to the others. As a result, it is selected as the infill strategy for BO in this work.

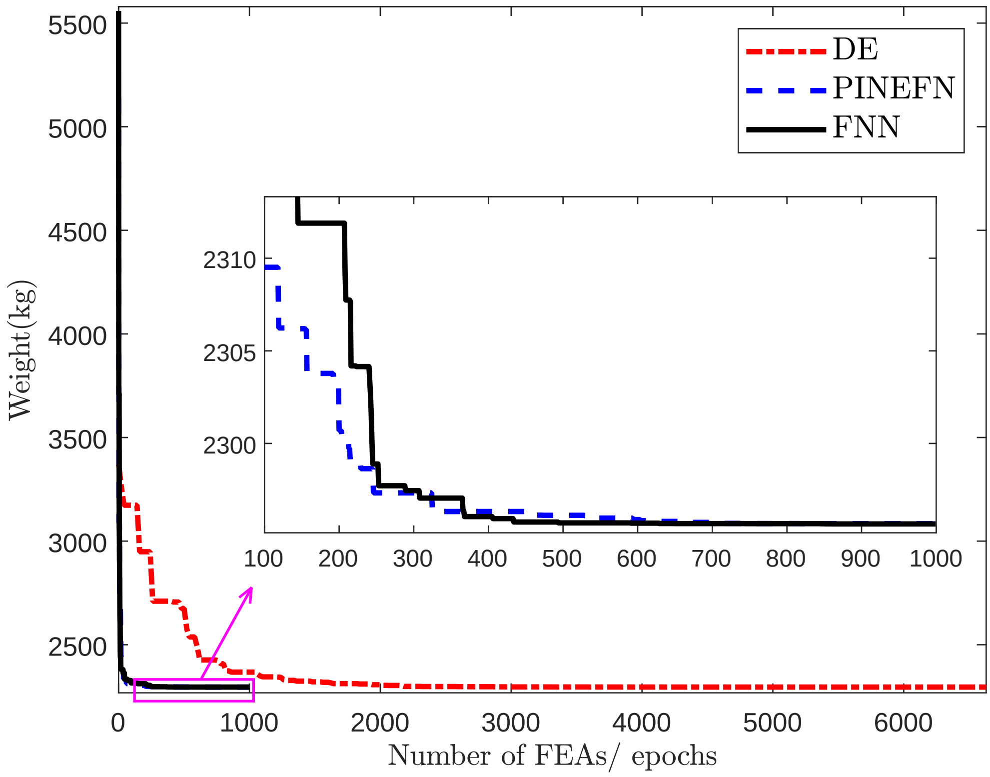


Fig. 8. Weight convergence histories of the 10-bar truss obtained using the FNN and other algorithms for the first load case.

Tables 4 - 5 summarized a comparison of the optimal solutions obtained by our framework with the optimal network and other studies for the first loading case. It is easily seen that the optimum weight obtained through FNN (2295.655 kg) agrees well with the PSOPC [71] (2295.631 kg), HPSO [71] (2295.595 kg), PINEFN [60] (2295.658 kg), and DE (2295.580 kg) without violating constraints. Although a lighter design found by Lee [72] (2294.216 kg), it violates the design constraints with maximum constraint violation error (CVE) 0.091%. As seen from Table 5, it is evident that none of the constraints are violated. In addition, the statistical results obtained by the proposed model show quite good agreement with the DE. However, in terms of reliability, the present model outperforms the DE algorithm regarding the statistics of the minimum weight while still maintaining accuracy. The weight convergence histories of three algorithms are depicted in Fig. 8. As observed, FNN and PINEFN have similar convergence rates that rapidly decrease in the first 200 epochs. However, our model tends to be stable and achieves the optimal weight around 500 epochs, while the PINEFN needs 700 epochs to approach the optimal solution. In contrast, the DE algorithm exhibits slow convergence and requires a significant number of Finite Element Analysis (FEA) evaluations (6680) to get the optimal weight. This can be explained by the fact that the DE is the gradient-free algorithm, so it demands a large number of function evaluations for optimization. Meanwhile, FNN and PINEFN models are designed based on the neural network, which serves as an optimizer for solving optimization problems. Thus, the weight optimization process for both these approaches relied on the gradient descent method and automated sensitivity analyses. And that's why it significantly reduces the number of function evaluations, as well as their convergence rates are much faster than the DE.

For the second load case, the optimal results obtained by FNN in comparison with other studies, including the hyper-parameters, design variable, weight, statistics, and convergence histories, are reported in Tables 6, 7, 8, and Figs. 9-10. Accordingly, the proposed framework found the minimum weight (2121.507 kg) with respect to the best combination of hyper-parameters (3, 40, 0.076, ReLU, 10, 0.303) obtained after 23 training iterations. It can be easily seen that the minimum weight obtained by Rizzi [73] (2121.415 kg) and FNN (2121.507 kg) are ranked as the first and second best among all compared algorithms without violating constraints, as shown in Table 7. Although the weight obtained by Lee [72] (2117.737 kg) represents the lightest designs, the constraints are violated with the CVE of 0.195 %. As the first load case, the data show that the FNN can find the optimum design more efficiently and reliably than the DE. More concretely, the deviation (0.168 kg) between the maximum (2121.675 kg) and minimum (2121.507 kg) optimal values of the structural weight found by FNN is very small, whilst it is 4.221 kg for the DE algorithm. Additionally, it is clear that the Std of the optimal objective function value obtained by the present model (0.021 kg) is relatively small compared to that of the DE (0.148 kg). For the DE algorithm, the stress constraint at member 5 is a little bit violated. Meanwhile, the structural responses found by the FNN satisfy all the constraints. A comparison of the structural weight convergence histories is depicted in Fig. 10. Clearly, the FNN converges more rapidly than the PINEFN and DE. It reaches the optimal mass of structure after only 800 epochs. On the contrary, the PINEFN and DE require 900 epochs and more than 80 times the number of analyses (8000), respectively.

Table 4
Comparison of the obtained results for the 10-bar planar truss (Case 01).

A_i (cm^2)	Li [71]			Schmit	Rizzi	Lee	Mai [60]	This study	
	PSO	PSOPC	HPSO	[9]	[73]	[72]	PINEFN	DE	FNN
A_1	215.929	197.219	198.090	215.690	198.264	194.516	196.993	196.774	196.993
A_2	0.710	0.645	0.645	0.645	0.645	0.658	0.652	0.645	0.652
A_3	149.529	148.219	149.464	156.516	154.413	146.516	149.729	149.774	149.742
A_4	99.839	97.729	97.955	92.000	95.052	98.516	98.258	98.052	98.219
A_5	23.542	0.645	0.645	0.645	0.645	0.658	0.645	0.645	0.645
A_6	0.748	3.529	3.555	0.645	0.645	3.510	3.555	3.523	3.561
A_7	53.729	48.342	48.129	53.793	55.107	48.652	48.116	48.161	48.103
A_8	150.580	136.510	135.342	133.806	135.187	139.097	135.826	136.206	135.658
A_9	148.477	139.071	138.761	127.032	140.877	138.387	138.677	138.522	138.871
A_{10}	1.226	0.645	0.645	0.645	0.645	0.645	0.645	0.645	0.645
W_{best} (kg)	2508.139	2295.631	2295.595	2308.332	2302.734	2294.216	2295.658	2295.580	2295.655
CVE_{max} (%)	None	None	None	21.136	None	0.091	None	None	None

Table 5
Statistics of the constraints and weight for the 10-bar planar truss (Case 01).

Metric	DE				FNN			
	v_1 (cm)	v_2 (cm)	σ_5 (MPa)	W (kg)	v_1 (cm)	v_2 (cm)	σ_5 (MPa)	W (kg)
Min	-5.080	-5.060	172.305	2295.580	-5.080	-5.057	172.286	2295.655
Max	-5.080	-5.057	172.369	2296.411	-5.080	-5.057	172.369	2295.917
Mean	-5.080	-5.057	172.357	2295.768	-5.080	-5.057	172.339	2295.749
Std	0.000	0.000	0.003	0.036	0.000	0.000	0.010	0.027
95% CIU	-5.080	-5.057	172.356	2295.754	-5.080	-5.057	172.341	2295.754
95% CIL	-5.080	-5.057	172.358	2295.780	-5.080	-5.057	172.337	2295.743

Table 6
Optimum hyper-parameters obtained by using the BO for different problems.

Test problems	Hyper-parameters					
	No. of hidden layers	No. of hidden neurons	Learning rate	Activation function	Step size	Gamma
10-bar truss (case2)	3	40	0.076	ReLU	10	0.303
17-bar planar truss	3	57	0.053	ReLU	8	0.535
25-bar space truss	1	26	0.048	ReLU	10	0.499
72-bar truss (case 1)	3	33	0.084	LeakyReLU	4	0.304
72-bar truss (case 2)	4	54	0.031	LeakyReLU	8	0.631
120-bar dome truss	2	45	0.001	LeakyReLU	8	0.339
200-bar planar truss	1	30	0.064	LeakyReLU	8	0.307

3.2. 17-bar truss

Next, a 17-bar plane truss structure illustrated in Fig. 11 is examined as the second numerical example for size optimization. The structure is subjected to a vertical load of 444.822 kN in the negative y-direction at node 9. All cross-sectional areas of elements are considered as design variables. The Young's modulus and material density are 206842.718 MPa and 7418.214 kg/m^3 for all members. The displacements of free nodes are limited to ± 5.08 cm, and allowable stresses of members are set to 344.738 MPa in both compression and tension. And a maximum of 1000 epochs is used as a stopping criterion for the training process.

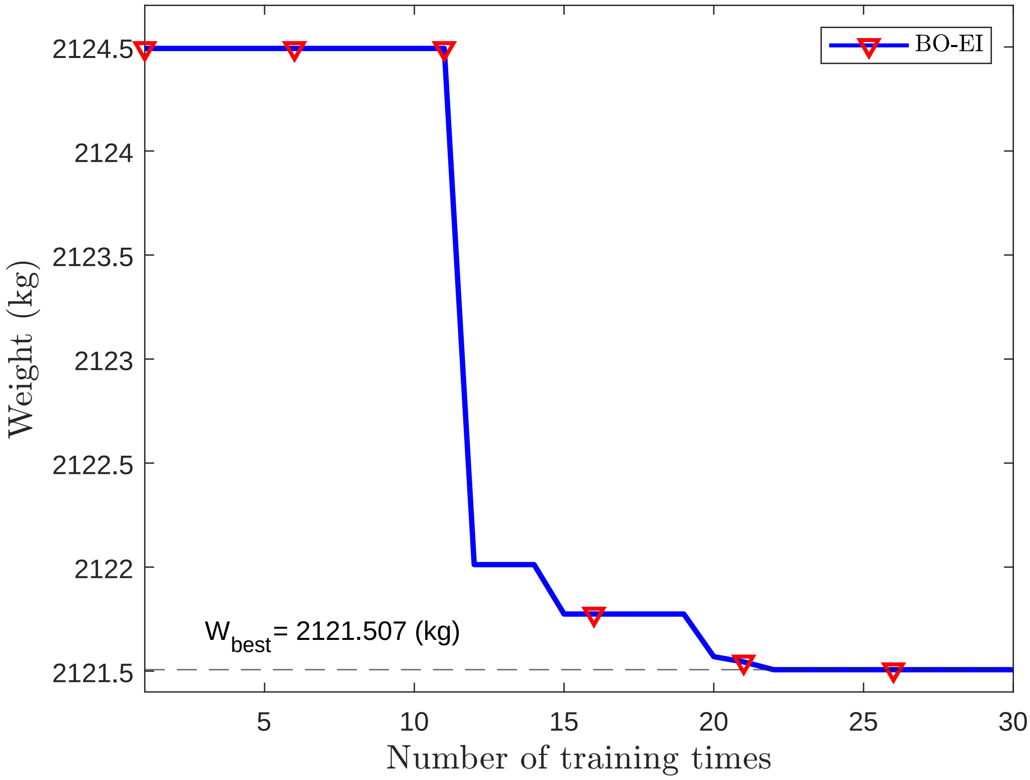


Fig. 9. Convergence history of the HPO using BO for the 10-bar truss structure for the second load case.

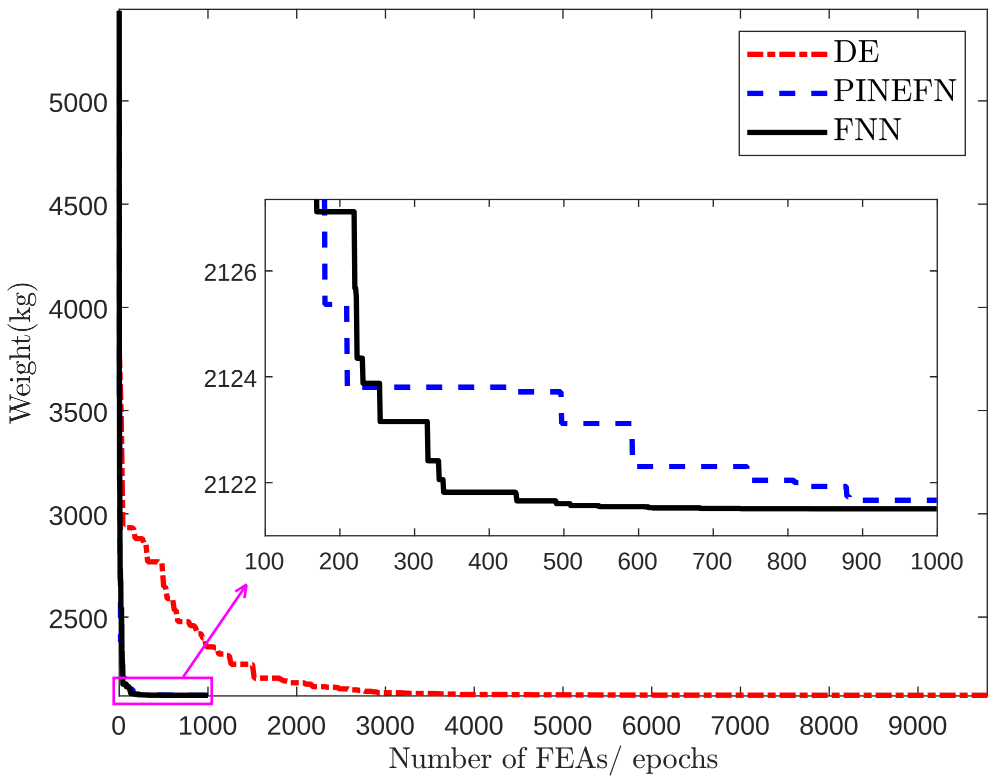


Fig. 10. Weight convergence histories of the 10-bar planar truss using the FNN and other works for the second load case.

Table 7
Comparison of the obtained results for the 10-bar planar truss (Case 02).

A_i (cm^2)			Li [71]	Schmit	Rizzi	Lee	Mai [60]		This study
	PSO	PSOPC	HPSO	[9]	[73]	[72]	PINEFN	DE	FNN
A_1	147.968	153.180	150.664	156.710	151.826	150.000	153.103	151.193	152.690
A_2	0.729	0.652	0.645	0.645	0.645	0.658	0.652	0.645	0.645
A_3	163.580	163.142	164.529	150.619	163.168	166.000	162.826	163.297	162.826
A_4	92.729	92.987	91.935	88.090	92.735	93.613	91.729	93.181	92.819
A_5	0.645	0.645	0.645	0.645	0.645	0.645	0.645	0.645	0.645
A_6	12.839	12.703	12.723	12.710	12.708	12.755	12.742	12.710	12.710
A_7	79.652	79.755	79.761	81.742	79.929	78.774	79.974	79.897	79.877
A_8	83.374	81.897	83.187	80.929	82.742	81.355	82.342	82.890	82.226
A_9	133.406	131.116	131.329	141.748	131.148	131.355	131.568	131.077	131.303
A_{10}	0.645	0.665	0.652	0.645	0.645	0.645	0.645	0.645	0.645
W_{best} (kg)	2122.572	2121.769	2121.583	2128.183	2121.415	2117.737	2121.565	2121.435	2121.507
CVE_{max} (%)	None	None	None	None	None	0.195	None	0.000	None

Table 8
Statistics of the constraints and weight for the 10-bar planar truss (Case 02).

Metric	DE					FNN				
	v_2 (cm)	v_4 (cm)	σ_5 (MPa)	σ_6 (MPa)	W (kg)	v_2 (cm)	v_4 (cm)	σ_5 (MPa)	σ_6 (MPa)	W (kg)
Min	-5.080	-3.973	172.361	172.330	2121.435	-5.080	-3.962	172.362	172.355	2121.507
Max	-5.080	-3.909	172.369	172.369	2125.655	-5.080	-3.955	172.369	172.369	2121.675
Mean	-5.080	-3.947	172.367	172.360	2121.885	-5.080	-3.957	172.368	172.367	2121.587
Std	0.000	0.003	0.000	0.002	0.148	0.000	0.000	0.001	0.001	0.021
95% CIU	-5.080	-3.950	172.367	172.360	2121.832	-5.080	-3.957	172.368	172.367	2121.591
95% CIL	-5.080	-3.947	172.367	172.361	2121.938	-5.080	-3.960	172.368	172.367	2121.583

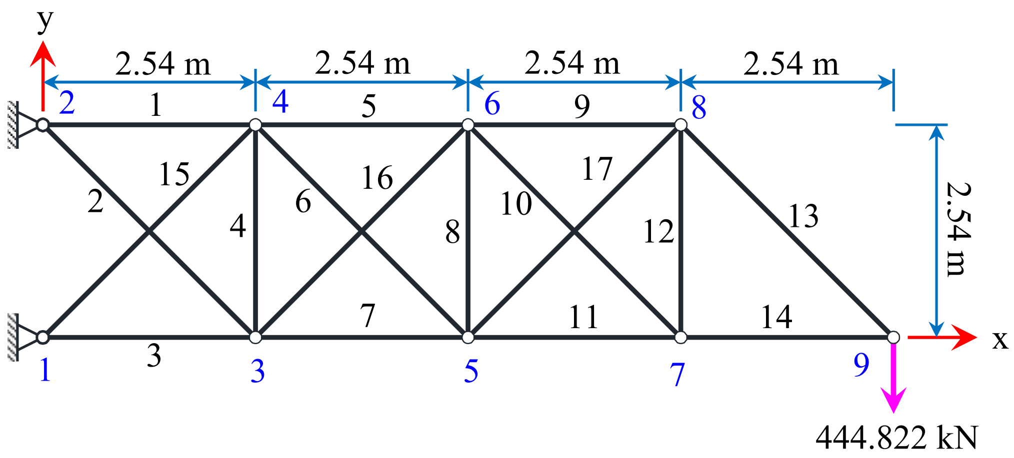


Fig. 11. A 17-bar planar truss structure.

As the previous example illustrates, the optimal hyper-parameters, as shown in Table 6 and Fig. 12, were found after 30 training times by the BO using the EI infill strategy. Additionally, Tables 9 and 10 report the optimal network's results, including the design variables, weights, constraints, and statistics. With a weight of 1171.128 kg, the FNN is the second-lightest design, surpassed only by the optimal weight obtained by Khot [3] (1171.126 kg). However, it is smaller than the other studies (PSO [71]:1235.753 kg; PSOPC [71]: 1171.561 kg; HPSO [71]: 1171.148 kg; PINEFN [60]: 1171.162 kg; and DE 1171.133 kg). Although the smallest weight found by Lee [72] is 1170.636 kg, it violates the design constraints (0.044%). Furthermore, the 95% CI values obtained by FNN are quite close to the minimum, maximum, and mean optimal weights, with a very small deviation (0.004 kg), whilst the Std of the DE (0.034 kg) is 8 times greater than that of our approach. From the data in Table 10 and Fig. 13, it is easily seen that the FNN performs better than the DE in terms of the reliability as well as the number of structural analyses. Our model rapidly indicates the optimum weight with only 1000 analyses, whereas the DE takes 9960. As can be seen in the plot, the convergence speed of the proposed framework with the optimal network is improved and faster than that of the PINEFN. Therefore, this once again demonstrates the effectiveness of automatic hyper-parameter tuning of the FNN.

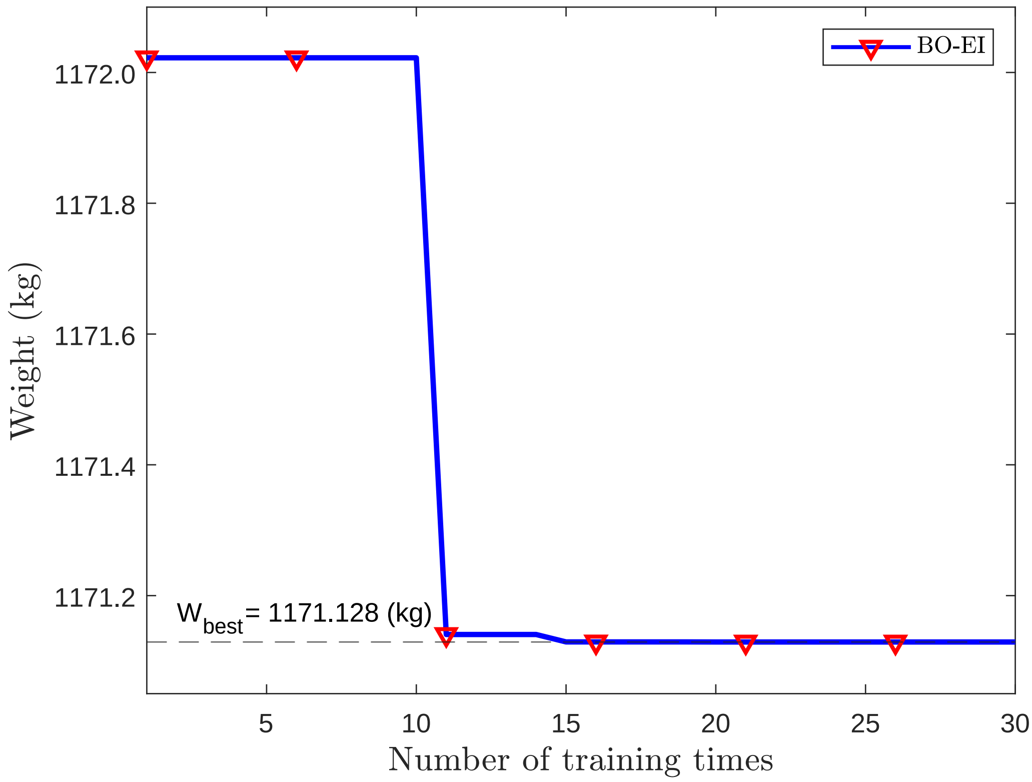


Fig. 12. Convergence history of the HPO using BO for the 17-bar planar truss.

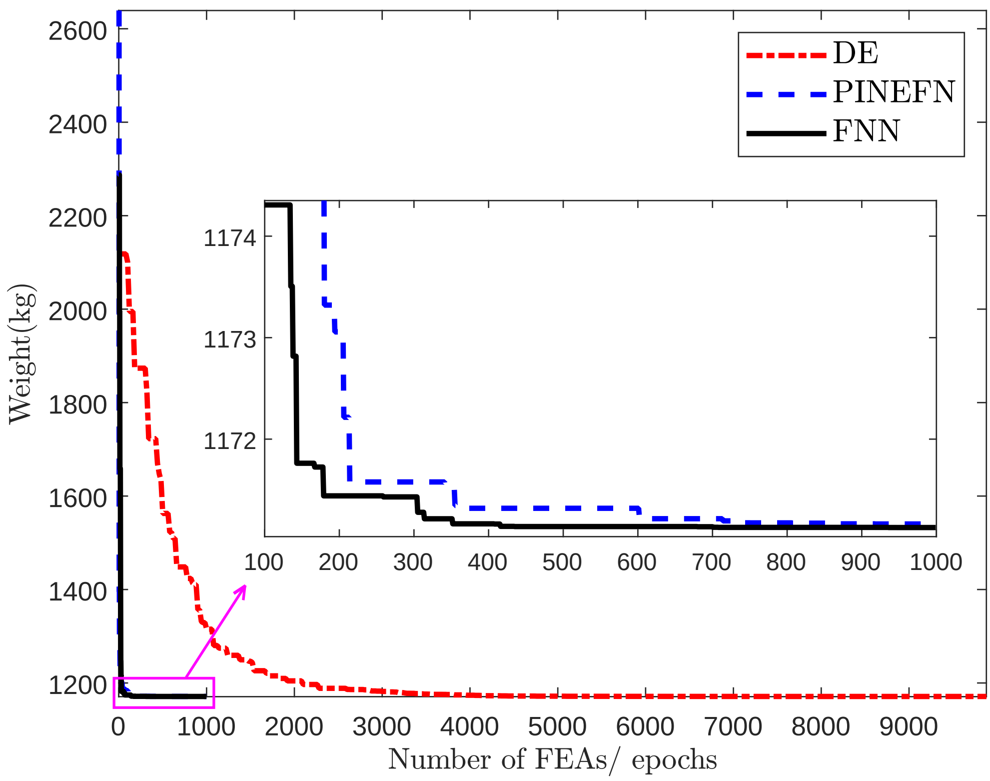


Fig. 13. Weight convergence histories of the 17-bar planar truss using the FNN and other works.

Table 9
Comparison of the obtained results for the 17-bar planar truss.

A_i (cm^2)	Lee			Li [71]	Khot	Adeli	Mai [60]		This study
	[72]	PSO	PSOPC	HPSO	[3]	[74]	PINEFN	DE	FNN
A_1	102.071	101.716	103.103	102.555	102.774	103.413	102.619	103.026	102.761
A_2	0.697	14.600	0.645	0.665	0.645	0.690	0.729	0.645	0.652
A_3	77.393	89.381	78.335	78.013	77.871	78.600	77.858	77.903	77.877
A_4	0.645	0.684	0.645	0.645	0.645	0.710	0.645	0.645	0.645
A_5	52.581	73.264	52.245	52.019	52.045	54.303	52.090	51.955	52.019
A_6	35.529	25.258	35.910	36.071	35.884	36.871	35.852	35.864	35.890
A_7	76.316	52.071	75.690	76.871	76.987	73.103	76.884	77.148	77.000
A_8	0.645	0.645	0.645	0.645	0.645	0.677	0.645	0.645	0.645
A_9	51.187	37.742	51.497	51.387	51.258	47.103	51.284	51.252	51.232
A_{10}	0.645	14.800	0.729	0.645	0.645	0.742	0.665	0.645	0.645
A_{11}	26.406	40.729	26.284	26.297	26.161	26.103	26.200	26.058	26.174
A_{12}	0.645	21.774	0.852	0.645	0.645	0.652	0.645	0.645	0.645
A_{13}	36.516	35.058	36.561	36.581	36.497	36.200	36.548	36.426	36.503
A_{14}	26.200	25.277	25.748	25.794	25.806	26.103	25.839	25.806	25.819
A_{15}	36.490	22.800	35.839	35.794	35.858	33.239	35.806	35.794	35.858
A_{16}	0.645	14.929	0.652	0.665	0.645	0.690	0.671	0.652	0.645
A_{17}	36.013	22.852	35.839	35.723	35.994	34.103	35.994	35.961	35.987
W_{best} (kg)	1170.636	1235.753	1171.561	1171.148	1171.126	1176.809	1171.162	1171.133	1171.128
CVE_{max} (%)	0.044	None	None	None	None	1.693	None	None	None

Table 10
Statistics of the constraints and weight for the 17-bar planar truss.

Metric	DE		FNN	
	v_9 (cm)	W (kg)	v_9 (cm)	W (kg)
Min	-5.080	1171.133	-5.080	1171.128
Max	-5.080	1171.971	-5.080	1171.165
Mean	-5.080	1171.282	-5.080	1171.134
Std	0.000	0.034	0.000	0.004
95% CIU	-5.080	1171.270	-5.080	1171.135
95% CIL	-5.080	1171.294	-5.080	1171.133

3.3. 25-bar space truss

The next example deals with the design of a 25-bar space truss structure, as shown in Fig. 14. All truss members are made of a material with a density of 2767.990 kg/m^3 and a Young's modulus 68947.573 MPa. For this structure, two loading cases, as presented in Table 11, are considered. In addition, the cross-sectional areas of members are classified into 8 groups according to the design variables and their corresponding allowable stresses, as listed in Table 12. Besides, the displacements of free nodes are constrained within the interval [- 0.889, 0.889] cm. To find the minimum mass of the structure, the network performs training with the maximum number of analyses equal to 1000.

Table 11
Loading conditions for the 25-bar space truss(kN).

Node	Case 1			Case 2		
	F_x	F_y	F_z	F_x	F_y	F_z
1	0	88.964	-22.241	4.448	44.482	-2.224
2	0	-88.964	-22.241	0	44.482	-2.224
3	0	0	0	2.224	0	0
6	0	0	0	2.224	0	0

Table 12
Allowable stresses for the structural elements of the 25-bar space truss.

A_i	Compressive stress (MPa)	Tension stress (MPa)
A_1	241.951	275.790
A_2-A_5	79.910	275.790
A_6-A_9	119.314	275.790
A_{10}-A_{11}	241.951	275.790
A_{12}-A_{13}	241.951	275.790
A_{14}-A_{17}	46.602	275.790
A_{18}-A_{21}	47.981	275.790
A_{22}-A_{25}	76.408	275.790

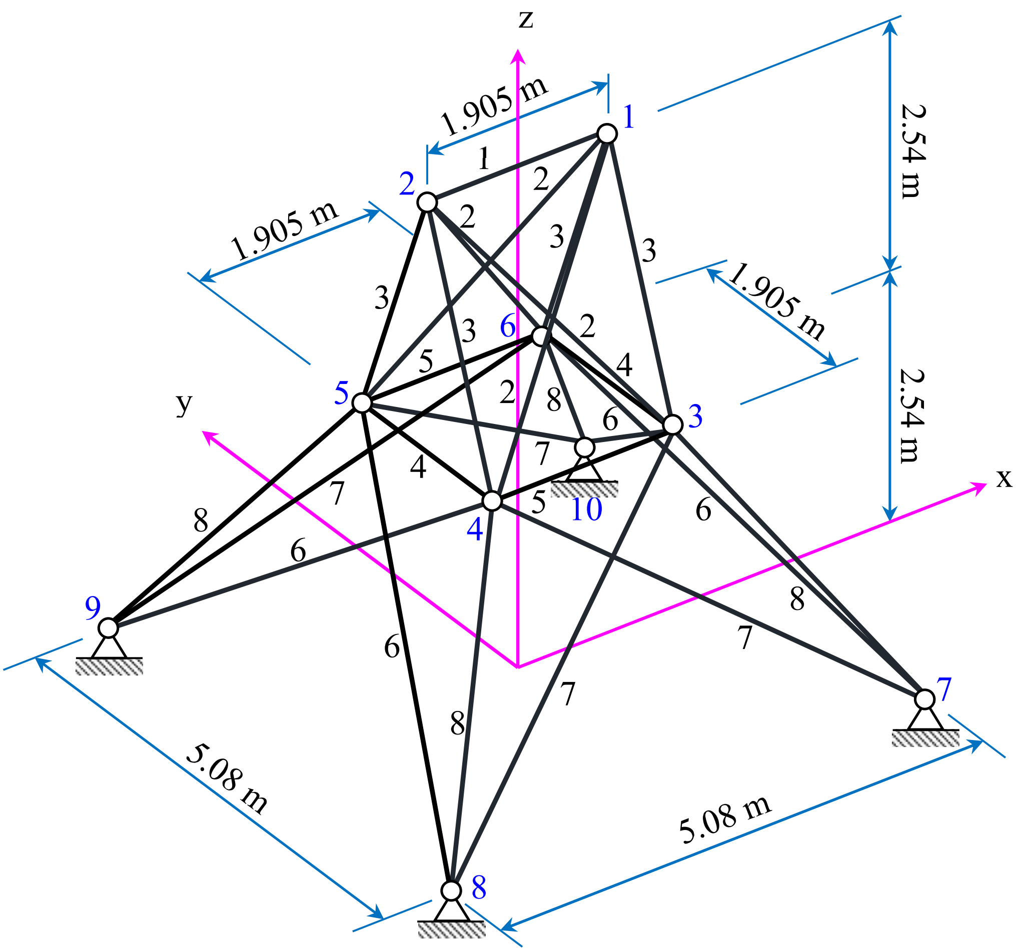


Fig. 14. A 25-bar space truss structure.

The BO with the EI strategy indicated the optimal hyper-parameters of the network after only 20 training iterations, as shown in Table 6 and Fig. 15. Accordingly, the optimal architecture of the network found by our approach (10-26-1) is smaller than those of DUL (6-20-20-20-1) and PINEFN (6-30-30-30-2). Therefore, the smaller network trains faster due to requiring fewer weights and biases. To evaluate the performance of the proposed method, a comparison of the optimal results found by the FNN and the other algorithms is reported in Tables 13 and 14. As expected, it can be observed that the optimal weight identified by the FNN (247.321 kg) agrees well with DUL [60] (247.529 kg) and Camp [75] (247.380 kg) without violating constraints. Note that Lee [72] found the smallest weight (246.927 kg), but it violates the constraints with an error of 0.206%. Although the results gained by Li [71] (247.294 kg), Degertekin [76] (247.249 kg), PINEFN [60] (247.308 kg), and DE (247.282 kg) are slightly lighter than the FNN, the errors between them and Degertekin [76] are less than 0.02%. According to the obtained statistical results, the present framework demonstrates the stability of optimized weight with the small standard deviation of 0.008 kg. A visual representation of the convergence histories between the FNN, DE, DUL, and PINEFN is illustrated in Fig. 16. As the above examples, the proposed approach converges faster than the DE and DUL. It only requires 1000 structural analyses, while the DUL and DE demand 1500 and 7520, respectively.

Table 13
Optimization results obtained for the 25-bar space truss.

A_i (cm^2)	Lee	Li	Kaveh	Degertekin	Mai [60]		Camp	This study	
	[72]	[71]	[77]	[76]	DUL	PINEFN	[75]	DE	FNN
A_1	0.303	0.065	17.174	0.065	0.084	0.084	0.065	0.065	0.077
A_2-A_5	13.045	12.710	12.858	13.361	12.587	12.781	13.497	12.832	12.865
A_6-A_9	19.032	19.458	19.716	19.077	19.135	19.381	19.123	19.290	19.252
A_{10}-A_{11}	0.065	0.065	0.065	0.065	0.077	0.071	0.065	0.065	0.090
A_{12}-A_{13}	0.090	0.065	0.065	0.065	0.084	0.071	0.065	0.065	0.071
A_{14}-A_{17}	4.439	4.477	4.290	4.445	4.497	4.432	4.445	4.413	4.419
A_{18}-A_{21}	10.690	10.845	10.594	10.458	11.135	10.819	10.329	10.819	10.813
A_{22}-A_{25}	17.181	17.052	17.284	17.271	17.052	17.135	17.329	17.181	17.187
W_{best} (kg)	246.927	247.294	247.280	247.249	247.529	247.308	247.380	247.282	247.321
CVE_{max} (%)	0.206	None	2.06	None	None	None	None	None	None

Table 14
Statistics of the constraints and weight for the 25-bar space truss.

Metric	DE			FNN		
	v_1 (cm)	v_2 (cm)	W (kg)	v_1 (cm)	v_2 (cm)	W (kg)
Min	0.889	-0.889	247.282	0.889	-0.889	247.321
Max	0.889	-0.889	247.455	0.889	-0.889	247.409
Mean	0.889	-0.889	247.292	0.889	-0.889	247.358
Std	0.000	0.000	0.006	0.000	0.000	0.008
95% CIU	0.889	-0.889	247.290	0.889	-0.889	247.359
95% CIL	0.889	-0.889	247.294	0.889	-0.889	247.357

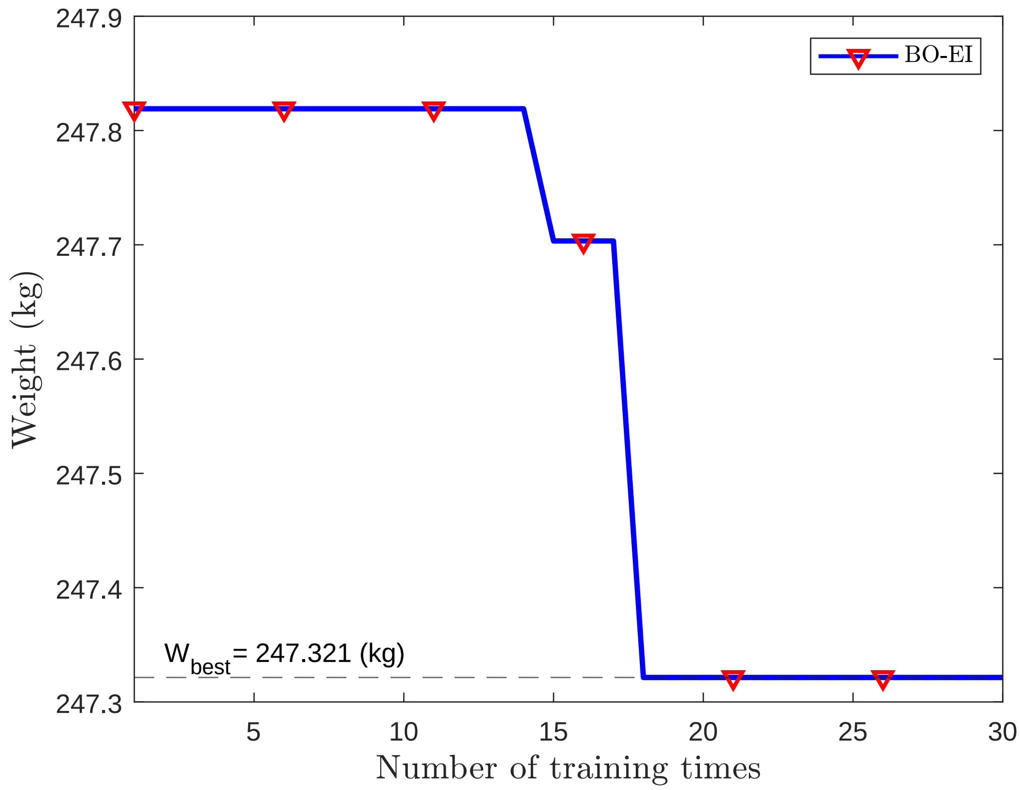


Fig. 15. Convergence history of the HPO using BO for the 25-bar space truss.

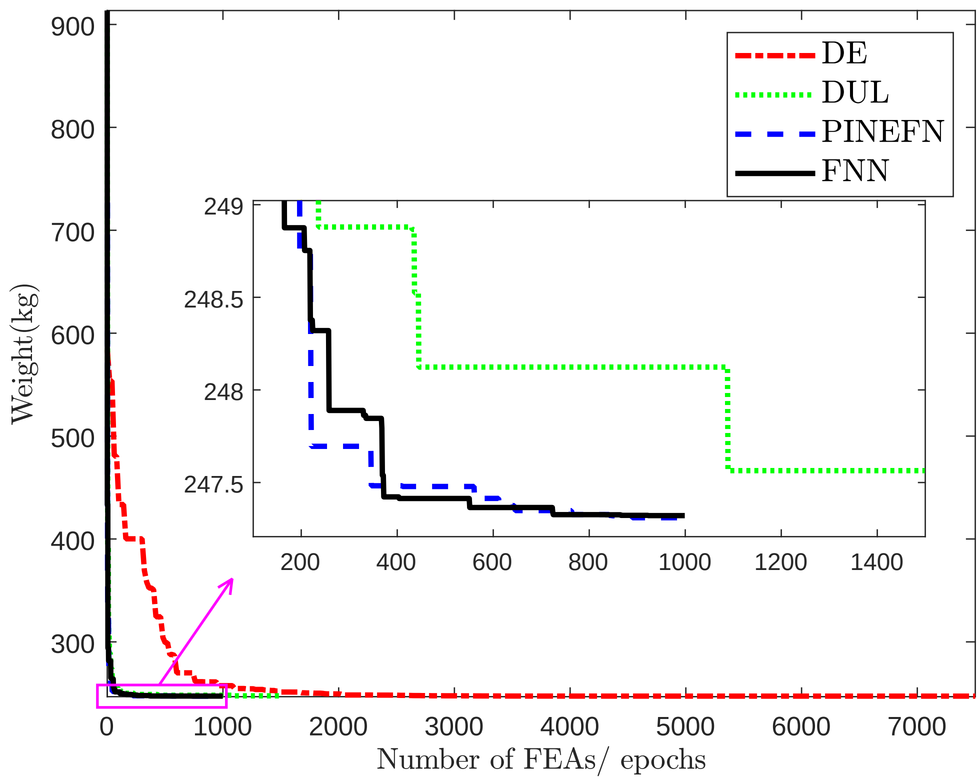


Fig. 16. Weight convergence histories of the 25-bar space truss using the FNN and other works.

3.4. 72-bar space truss

A 72-bars space truss shown in Fig. 17 is considered for the next numerical example. All cross-sectional areas of the truss members are divided into 16 groups corresponding to design variables, as listed in Table 16. The bars are made of the same material with Young's modulus of 68947.572 MPa, density of 2767.990 kg/m^3, and allowable stress of ± 172.369 MPa. Besides, the displacements of joints are restricted to ± 0.635 cm. This structure is subjected to two loading conditions, as tabulated in Table 15. Therefore, the lower bounds of the design variables are set at 0.645 cm^2 and 0.065 cm^2 for the first and second cases, respectively. To achieve the goal, the maximum epoch is set to 1000 for this particular application.

Table 15
Loading conditions for the 72-bar space truss (kN).

Node	Case 1			Case 2		
	F_x	F_y	F_z	F_x	F_y	F_z
17	22.241	22.241	-22.241	0	0	-22.241
18	0	0	0	0	0	-22.241
19	0	0	0	0	0	-22.241
20	0	0	0	0	0	-22.241

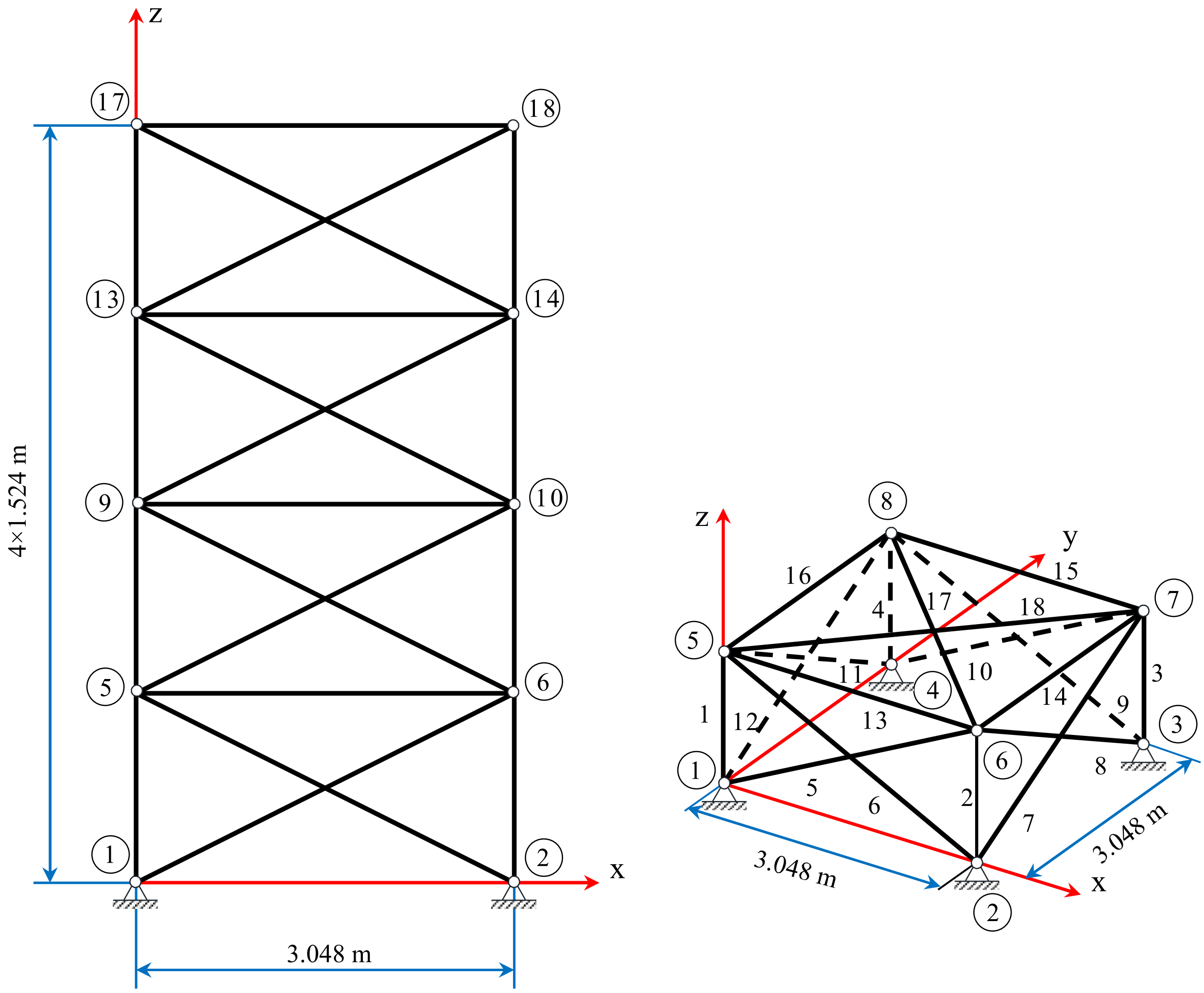


Fig. 17. A 72-bar space truss structure.

Figs. 18, 20, and Table 6 present the convergence curves and the network's best hyper-parameters for different loading cases. From these graphs, the infill strategy EI with 10 initial samples found two best-fitted hyper-parameter sets, which are found after 27 and 22 samples for the first and second loading cases, respectively. A comparison between the FNN with the optimal network and other algorithms for the structural optimization is reported in Tables 16, 17, and 18. From the data in these tables, the results reveal that: 1) the optimum weights found by the proposed approach are ranked as the second-best and best designs without violating constraints for the first and second loading cases, respectively; 2) the standard deviations obtained by the FNN are small (0.005 kg and 0.026 kg); 3) clearly, the confidence upper bounds of the optimal weights are quite close to the confidence lower bounds. Based on these results, our model has once again demonstrated its effectiveness in automatic hyper-parameter tuning, as well as its capability to yield a high reliability solution for the optimization of truss structure. Additionally, Figs. 19 and 21 depict the convergence histories of the FNN, DE, DUL, and PINEFN. As can be seen on these plots, the proposed FNN model's learning curves always converge more quickly than the DUL and conventional DE algorithms, and similar to the learning process of the PINEFN. Both our model and PINEFN indicate the optimal solution with only 1000 analyses, while the DUL (2500) and DE (15000) are still far behind. Therefore, the present approach shows more efficiency than the conventional algorithms.

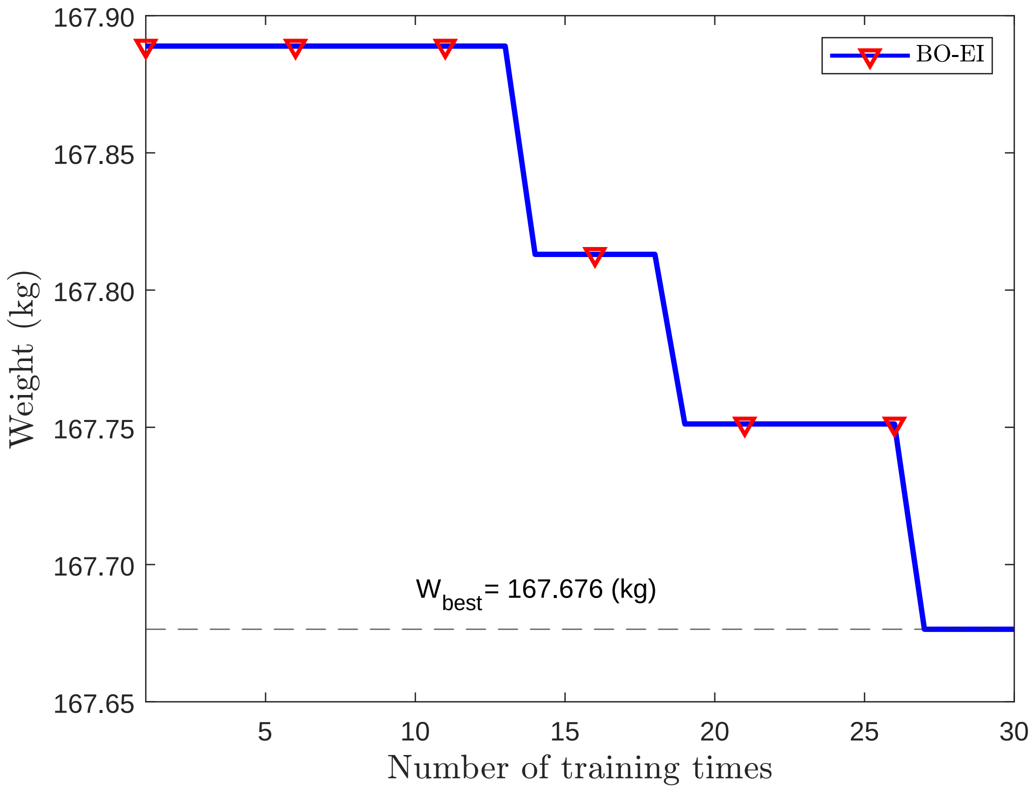


Fig. 18. Convergence history of the HPO using BO for the 72-bar space truss (Case 1).

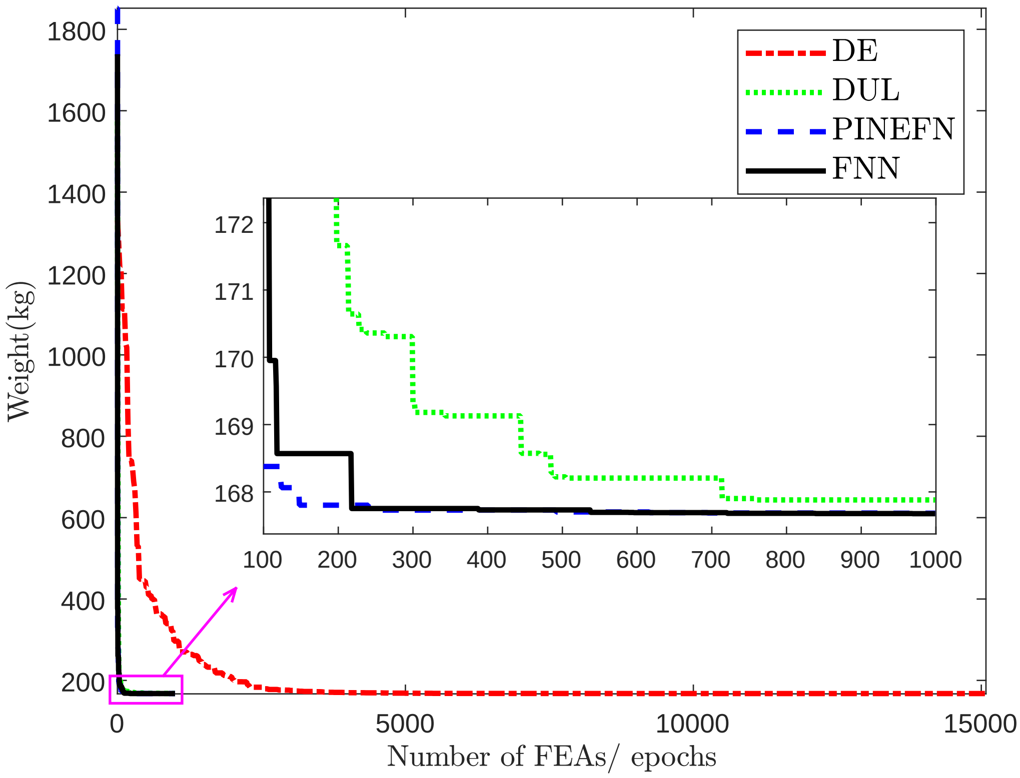


Fig. 19. Weight convergence histories of the 72-bar space truss using the FNN and other works (Case 1).

Table 16
Optimization results obtained for the 72-bar space truss with displacement and stress constraints (Case 01).

A_i (cm^2)	Kaveh [77]	Camp [75]	Degertekin [76]	Bekdaş [78]	Mai [60] DUL	Mai [60] PINEFN	Ehsan [79]	This study DE	This study FNN
A_1-A_4	12.284	11.987	12.297	12.103	12.006	11.981	12.452	11.974	11.865
A_5-A_{12}	3.329	3.265	3.265	3.329	3.232	3.258	3.284	3.245	3.265
A_{13}-A_{16}	0.645	0.645	0.645	0.645	0.658	0.645	0.645	0.645	0.645
A_{17}-A_{18}	0.645	0.645	0.645	0.645	0.652	0.645	0.645	0.645	0.645
A_{19}-A_{22}	8.116	8.052	8.142	8.381	8.155	8.090	8.045	8.116	8.110
A_{23}-A_{30}	3.252	3.400	3.297	3.387	3.265	3.252	3.310	3.239	3.252
A_{31}-A_{34}	0.645	0.645	0.645	0.645	0.652	0.645	0.645	0.645	0.645
A_{35}-A_{36}	0.645	0.652	0.645	0.645	0.652	0.645	0.645	0.645	0.645
A_{37}-A_{40}	3.342	3.361	3.432	3.206	3.206	3.194	3.419	3.219	3.194
A_{41}-A_{48}	3.361	3.335	3.329	3.284	3.277	3.277	3.335	3.277	3.284
A_{49}-A_{52}	0.645	0.645	0.645	0.645	0.658	0.645	0.645	0.645	0.645
A_{53}-A_{54}	0.652	0.652	0.645	0.645	0.665	0.645	0.645	0.645	0.645
A_{55}-A_{58}	1.013	1.013	1.006	1.019	0.645	0.645	1.006	0.645	0.645
A_{59}-A_{66}	3.497	3.555	3.542	3.439	3.348	3.368	3.510	3.374	3.374
A_{67}-A_{70}	2.665	2.529	2.645	2.639	2.587	2.568	2.652	2.574	2.574
A_{71}-A_{72}	3.716	3.819	3.677	3.697	3.465	3.458	3.626	3.458	3.465
$W_{bes}t$ (kg)	172.211	172.297	172.1973	171.95	167.8487	167.675	172.208	167.669	167.676
CVE_{max} (%)	None	None	None	None	None	None	None	None	None

Table 17
Optimization results obtained for the 72-bar space truss with displacement and stress constraints (Case 02).

A_i (cm^2)	Adeli [80]	Adeli [37]	Sarma [81]	Lee [72]	Li [71]	Mai [60]	This study	
							DE	FNN
A_1-A_4	13.071	17.774	11.174	12.665	12.303	11.839	12.348	12.239
A_5-A_{12}	3.439	3.290	3.368	3.103	3.381	3.355	3.323	3.323
A_{13}-A_{16}	0.065	0.065	0.065	0.065	0.065	0.065	0.065	0.065
A_{17}-A_{18}	0.065	0.065	0.084	0.071	0.065	0.065	0.065	0.065
A_{19}-A_{22}	7.465	8.839	8.677	7.955	8.310	8.477	8.445	8.394
A_{23}-A_{30}	3.671	3.271	3.555	3.265	3.374	3.342	3.348	3.355
A_{31}-A_{34}	0.065	0.065	0.065	0.071	0.065	0.065	0.071	0.071
A_{35}-A_{36}	0.065	0.065	0.084	0.077	0.065	0.071	0.123	0.071
A_{37}-A_{40}	3.316	3.103	3.174	3.471	3.510	3.361	3.387	3.361
A_{41}-A_{48}	3.090	3.277	3.516	3.439	3.406	3.348	3.335	3.329
A_{49}-A_{52}	0.065	0.065	0.426	0.065	0.123	0.065	0.065	0.071
A_{53}-A_{54}	0.065	0.413	0.084	1.077	0.129	0.652	0.684	0.658
A_{55}-A_{58}	1.019	1.387	1.148	1.039	1.135	1.084	1.142	1.084
A_{59}-A_{66}	3.548	3.342	3.381	3.497	3.452	3.471	3.426	3.439
A_{67}-A_{70}	2.226	2.703	2.555	3.084	2.748	2.923	2.935	2.897
A_{71}-A_{72}	3.213	3.252	3.839	3.555	3.948	3.768	3.774	3.729
W_{best} (kg)	3172.052	170.778	165.289	165.257	165.498	165.130	165.708	165.099
CVE_{max} (%)	-	-	-	-	-	None	None	None

Table 18
Statistics of the weight for the 72-bar space truss (kg).

Algorithm	Metric					
	Min	Max	Mean	Std	95% CIU	95% CIL
			Case 1			
DE	167.669	167.713	167.678	0.001	167.678	167.679
FDNN	167.676	167.720	167.695	0.005	167.696	167.694
			Case 2			
DE	165.708	165.892	165.747	0.007	165.744	165.749
FDNN	165.099	165.305	165.180	0.026	165.185	165.174

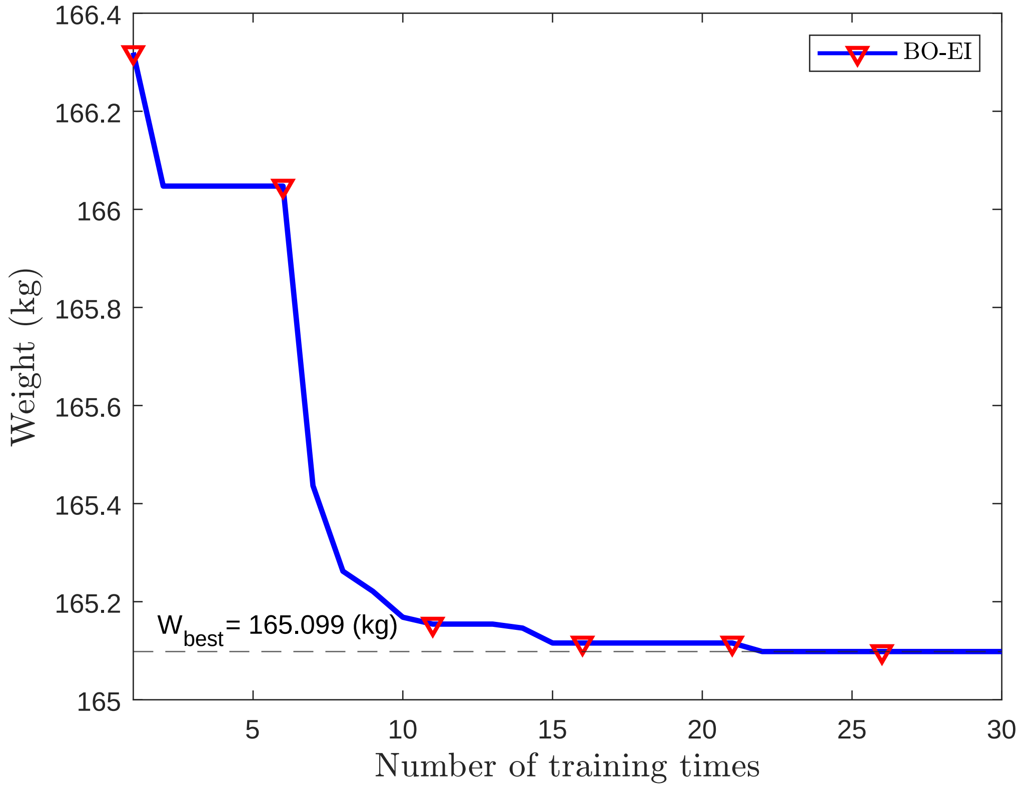


Fig. 20. Convergence history of the HPO using BO for the 72-bar space truss (Case 2).

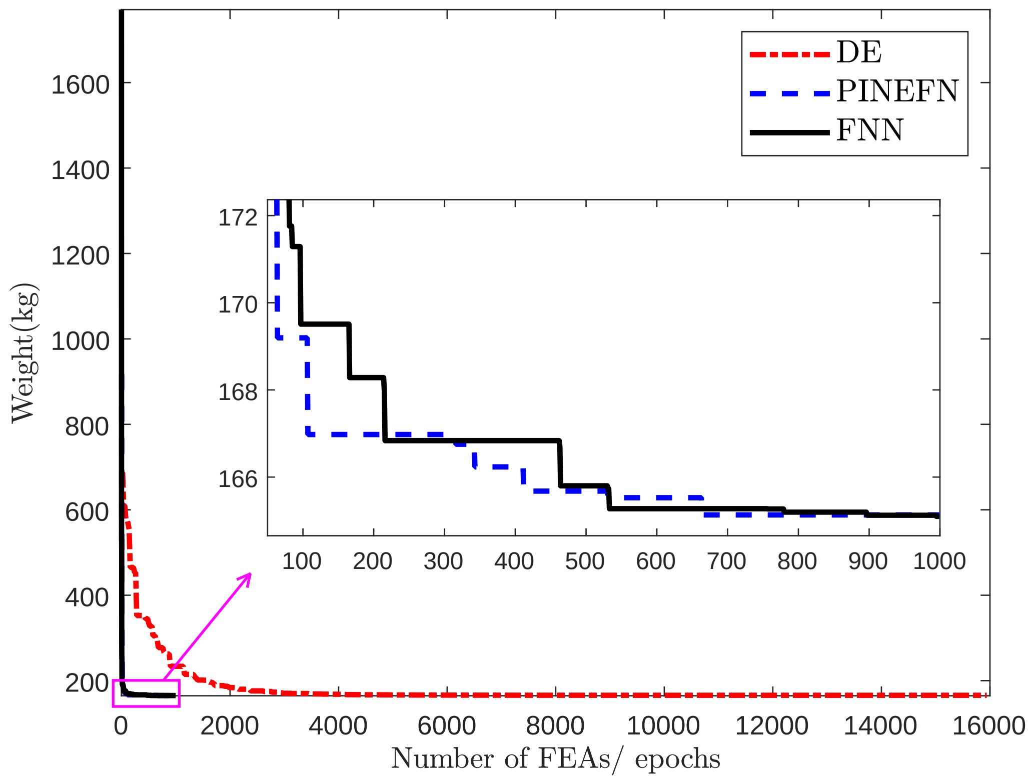


Fig. 21. Weight convergence histories of the 72-bar space truss using the FNN and other works (Case 2).

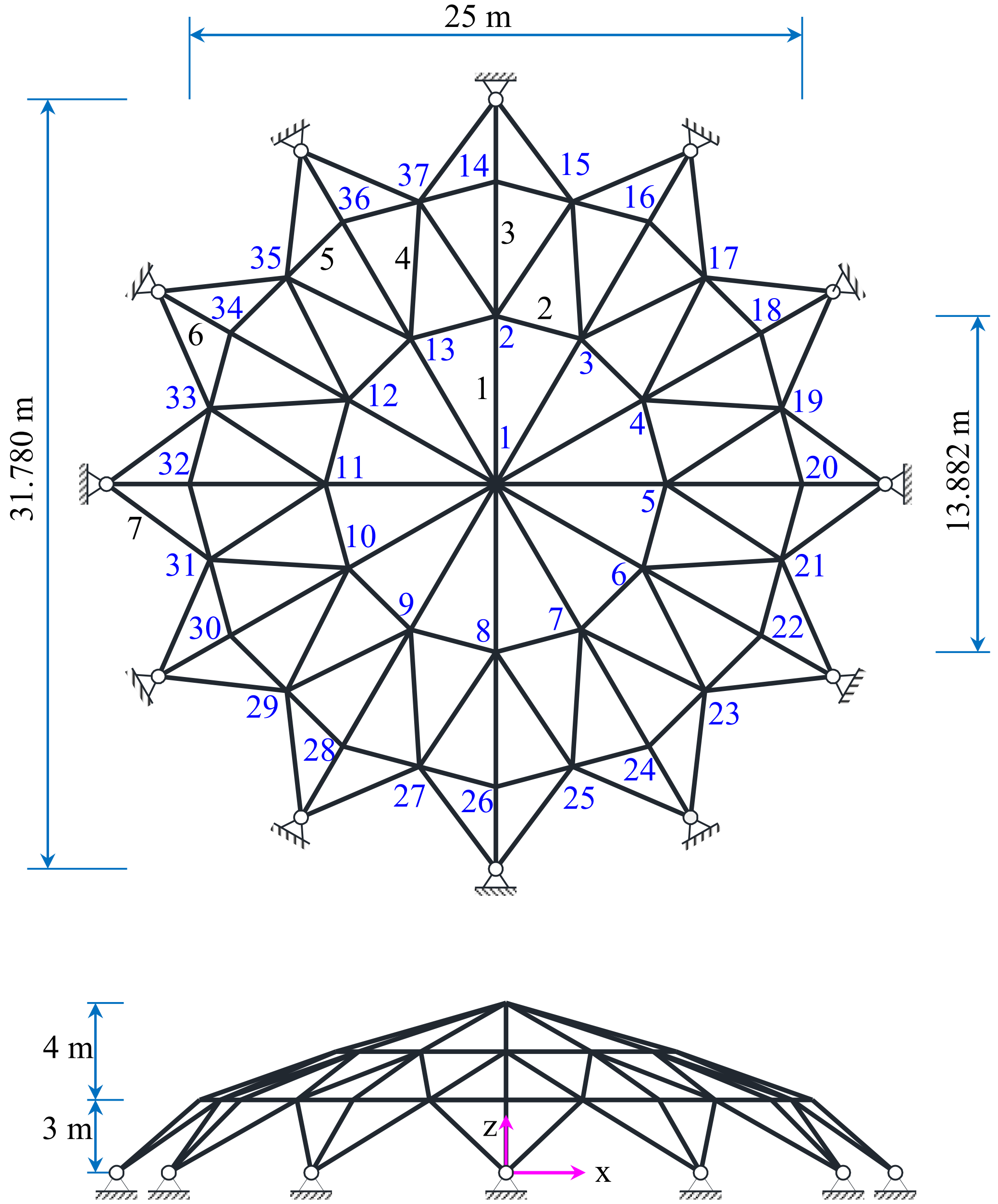


Fig. 22. 120-bar dome truss.

3.5. 120-bar dome truss

A 120-bar dome truss, as shown in Fig. 22, is investigated as the fifth example. As depicted in this plot, the design variables, representing the cross-sectional areas of the members, are categorized into seven groups. All members are made of steel with a yield stress (σ_y) of 399.896 MPa, an elasticity modulus (E) of 209945.360 MPa, and a density of 7971.813 kg/m^3. The minimal values for the design variables are 5 cm^2. According to the AISC ASD (1989) [82], the permissible tensile (σ_i^t) and compressive (σ_i^c) stresses are calculated as follows:

$$\begin{cases} \sigma_i^t = 0.6\sigma_y & \text{for } \sigma_i \geq 0, \\ \sigma_i^c & \text{for } \sigma_i < 0, \end{cases} \tag{14}$$

with

$$\sigma_i^c = \begin{cases} \left[\left(1 - \frac{\lambda_i^2}{2C_C^2}\right)\sigma_y\right] / \left(\frac{5}{3} + \frac{3\lambda_i}{8C_C} - \frac{\lambda_i^3}{8C_C^3}\right) & \text{for } \lambda_i < C_C, \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_C, \end{cases} \tag{15}$$

where L_i denotes the truss member length; C_C is the slenderness factor that separates the elastic and inelastic buckling regions ($C_C = \sqrt{2\pi^2 E/\sigma_y}$); λ_i represents the slenderness ratio ($\lambda_i = kL_i/r_i$); k is the effective length factor; r_i is the radius of gyration ($r_i = aA_i^b$); a and b denote constants, which are set to 0.4993 and 0.6777 for bars. In this example, where only vertical loads act on the structure in the negative direction of the z-axis, they are composed of 60.007 kN at node 1, 29.999 kN at nodes 2-13, and 10 kN at nodes 14-37. Besides the stress constraints, all vertical displacements of free joint are restricted to 0.5 cm. Similar to the previous examples, the total number of epochs is 1000 for the training process.

The solution and iteration history for addressing the hyper-parameters optimization problem using the infill sampling criteria EI of the BO are shown in Table 6 and Fig. 23. A comparison between the obtained results corresponding to the optimal network and other algorithms is summarized in Tables 19 and 20. As expected, the BO requires only 30 training iterations to identify the optimal combination of hyper-parameters (2, 45, 0.001, LeakyReLU, 8, 0.3387), resulting in the minimum weight (14741.589 kg). It is worth mentioning that the FNN achieves the lightest design overall. It is interesting here that our model outperforms the state-of-the-art approach PINEFN (14744.442 kg) by Mai et al. [60] in terms of the quality of solution. Furthermore, the best optimum weight is very close to worst (14741.612 kg) and mean (14741.601 kg) weights with the small Std values (0.003 kg). On the other hand, the 95% CI upper (14741.601 kg) and lower (14741.600 kg) bounds are not significantly different as well as close to the best weight. From the obtained statistical results of the objective and constraints, the FNN provides higher reliability than the DE algorithm. Fig. 24 displays the learning curves of the present method, PINEFN, and DE for the structural weight. Once again shows that our approach converges the fastest compared to the other methods.

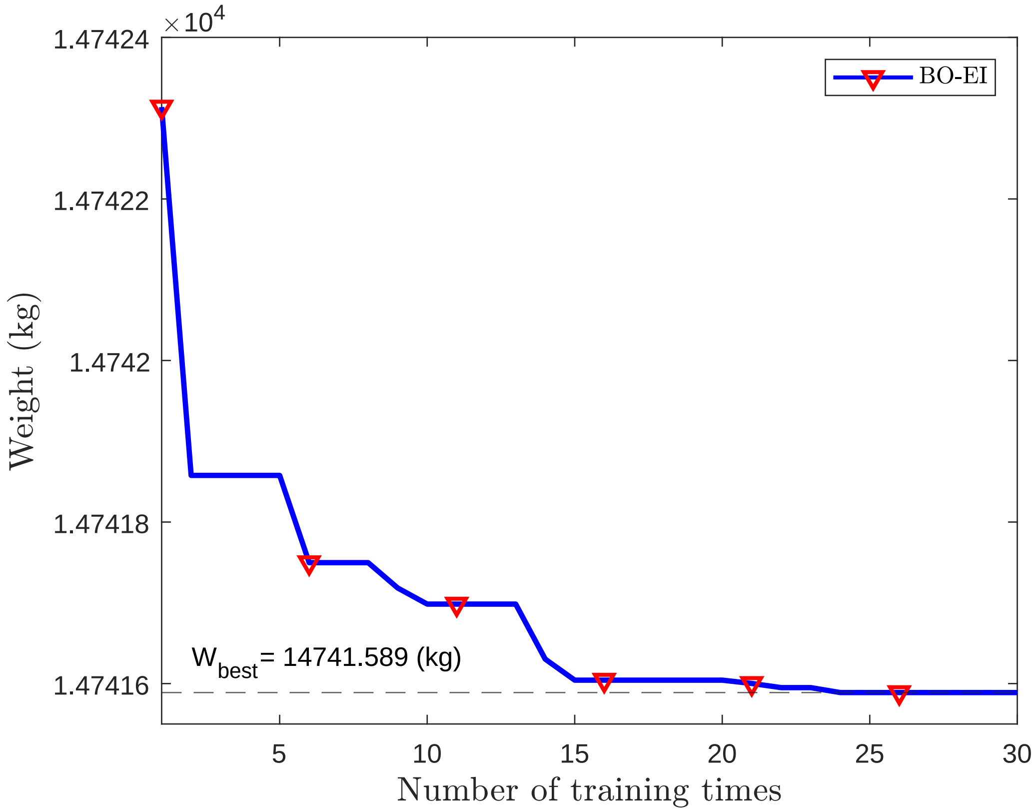


Fig. 23. Convergence history of the HPO using BO for the 120-bar dome truss.

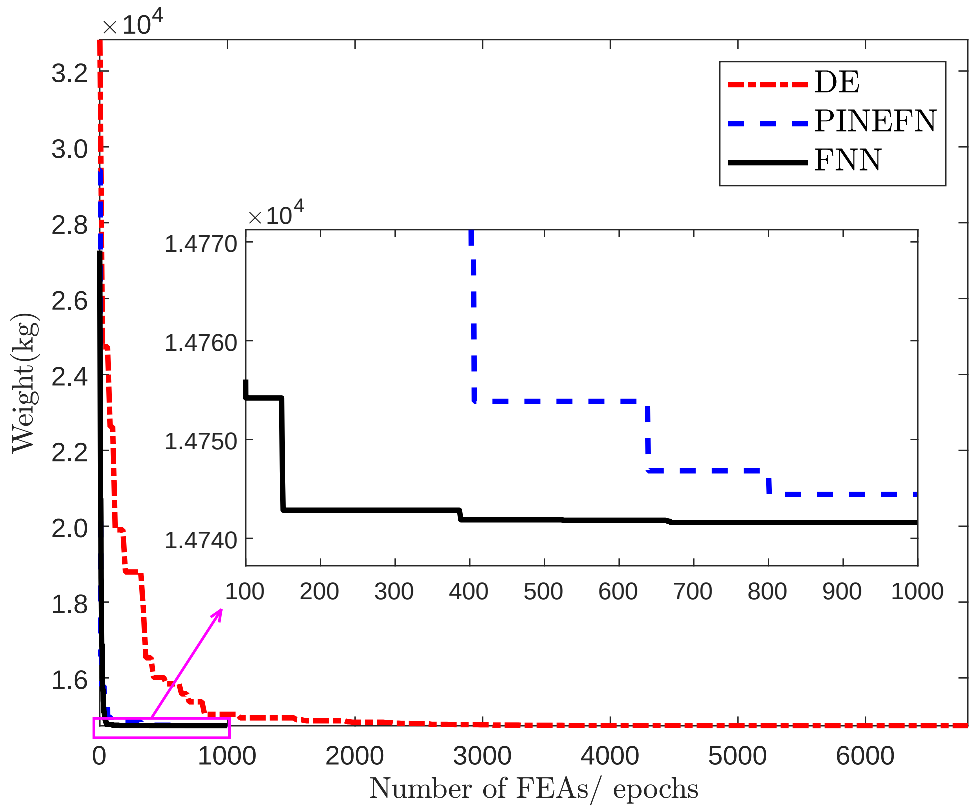


Fig. 24. Weight convergence histories of the 120-bar dome truss using the FNN and other works.

Table 19
Comparison of the obtained results for the 120-bar dome truss.

A_i (cm^2)	Kaveh [83]	Kaveh [84]	Kaveh [85]	Talatahari [86]	Kaveh [87]	Kaveh [77]	Mai [60]	This study	
								DE	FNN
A_1	19.968	19.529	19.510	19.510	19.510	19.510	12.368	12.342	12.342
A_2	92.935	94.232	94.948	95.355	95.361	95.374	96.310	96.052	96.058
A_3	32.387	32.542	32.774	32.626	32.594	32.587	37.116	37.097	37.084
A_4	21.626	20.252	20.239	20.232	20.239	20.239	16.561	16.600	16.548
A_5	55.684	55.116	54.845	54.729	54.839	54.826	64.716	64.826	64.858
A_6	22.142	21.723	21.303	21.355	21.219	21.239	23.090	22.968	23.077
A_7	16.123	16.110	16.110	16.116	16.110	16.110	12.748	12.768	12.768
W_{best} (kg)	15081.447	15082.808	15082.137	15082.500	15081.969	15081.833	14744.442	14741.630	14741.589
CVE_{max} (%)	-	-	-	-	-	-	None	None	None

Table 20
Statistics of the constraints and weight for the 120-bar dome truss.

Metric	DE					FNN				
	v_{20} (cm)	w_3 (cm)	σ_{100} (MPa)	σ_{65} (MPa)	W (kg)	u_{14} (cm)	w_3 (cm)	σ_{99} (MPa)	σ_{62} (MPa)	W (kg)
Min	0.091	-0.500	-19.843	12.534	14741.631	0.094	-0.500	-19.509	12.643	14741.589
Max	0.097	-0.500	-18.732	12.786	14746.243	0.094	-0.500	-19.475	12.677	14741.612
Mean	0.094	-0.500	-19.417	12.671	14742.681	0.094	-0.500	-19.494	12.657	14741.601
Std	0.000	0.000	0.041	0.011	0.233	0.000	0.000	0.003	0.003	0.003
95% CIU	0.094	-0.500	-19.432	12.667	14742.597	0.094	-0.500	-19.493	12.658	14741.601
95% CIL	0.094	-0.500	-19.403	12.675	14742.764	0.094	-0.500	-19.494	12.657	14741.600

Table 21
Design variables of the 200-bar planar truss.

Design variables	Member group	Design variables	Member group
A_1	1, 2, 3, 4	A_{16}	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
A_2	5, 8, 11, 14, 17	A_{17}	115, 116, 117, 118
A_3	19, 20, 21, 23, 24	A_{18}	119, 122, 125, 128, 131
A_4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	A_{19}	133, 134, 135, 136, 137, 138
A_5	26, 29, 32, 35, 38	A_{20}	140, 143, 146, 149, 152
A_6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	A_{21}	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
A_7	39, 40, 41, 42	A_{22}	153, 154, 155, 156
A_8	43, 46, 49, 52, 55	A_{23}	157, 160, 163, 166, 169
A_9	57, 58, 59, 60, 61, 62	A_{24}	171, 172, 173, 174, 175, 176
A_{10}	64, 67, 70, 73, 76	A_{25}	178, 181, 184, 187, 190
A_{11}	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	A_{26}	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
A_{12}	77, 78, 79, 80	A_{27}	191, 192, 193, 194
A_{13}	81, 84, 87, 90, 93	A_{28}	195, 197, 198, 200
A_{14}	95, 96, 97, 98, 99, 100	A_{29}	196, 199
A_{15}	102, 105, 108, 111, 114		

3.6. 200-bar planar truss

To demonstrate the efficiency and reliability of the FNN, the last numerical example considered is a plane truss with 200 members. They are made of material with density and elastic modulus of 7833.413 kg/m^3 and 206842.719 MPa, respectively. The geometric information, finite element representation, and boundary conditions are depicted in Fig. 25. The continuous design variables, with the lower bound to be 0.645 cm^2, are cross-sectional areas categorized into 29 groups, as shown in Table 21. The stress of members is limited in interval [-68.948; 68.948] MPa. The system is subjected to three loading cases: 1) a horizontal load of 4.448 kN in the positive direction of the x-axis is applied to nodes 1, 6, 15, 20, 29, 34, 43, 48,57, 62 and 71; 2) a vertical load of 44.482 kN in the negative direction of the y-axis is imposed on nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20,22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60, 61, 62, 64, 66, 68, 70, 71, 72, 73, 74 and 75; 3) both cases (1) and (2) acting together. In this problem, the maximum number of epochs allowed for the training process was 5000.

Table 22
Optimization results obtained for the 200-bar planar truss.

A_i (cm^2)	Lee	Kaveh	Lamberti	Degertekin	Mai [60]		Pierezan	This study	
	[72]	[77]	[88]	[89]	DUL	PINEFN	[90]	DE	FNN
A_1	0.806	0.665	0.948	0.942	0.761	0.665	0.897	0.813	0.645
A_2	6.555	5.923	6.065	6.071	6.394	6.110	6.039	6.013	6.084
A_3	0.690	0.774	0.645	0.645	0.735	0.748	0.645	0.690	0.665
A_4	0.710	0.652	0.645	0.652	1.497	0.716	0.645	0.645	0.665
A_5	12.497	12.039	12.516	12.523	12.632	12.581	12.490	12.458	12.535
A_6	1.735	1.826	1.910	1.910	1.877	1.923	1.877	1.845	1.852
A_7	0.671	0.645	0.645	0.645	1.045	0.794	0.645	0.658	0.652
A_8	19.181	19.148	20.026	20.135	20.335	20.226	19.884	19.806	20.206
A_9	0.845	0.645	0.645	0.645	0.890	0.671	0.645	1.587	0.658
A_{10}	26.987	25.458	26.477	26.923	26.987	26.652	26.335	26.258	26.658
A_{11}	2.561	2.413	2.600	2.587	2.477	2.716	2.561	2.890	2.639
A_{12}	2.852	2.903	1.232	1.168	1.368	0.684	1.910	1.174	0.658
A_{13}	33.464	32.000	35.019	34.987	35.142	35.219	34.742	34.781	35.110
A_{14}	1.232	6.929	0.645	0.645	0.819	0.684	0.645	0.748	0.690
A_{15}	40.264	38.574	41.471	41.432	41.606	41.658	41.194	41.232	41.568
A_{16}	4.510	5.071	3.697	3.684	3.432	3.555	4.084	3.839	3.523
A_{17}	0.748	4.755	0.858	1.006	1.387	0.794	1.187	1.329	0.897
A_{18}	50.090	47.619	51.432	51.342	51.684	51.593	51.871	51.406	51.445
A_{19}	0.645	4.303	0.645	0.645	0.923	0.897	0.645	1.116	0.652
A_{20}	56.955	53.548	57.884	57.793	58.045	57.968	58.323	57.858	57.897
A_{21}	4.510	7.723	4.548	4.645	4.548	4.697	4.813	5.090	4.594
A_{22}	10.039	6.452	2.710	3.084	1.581	1.542	0.845	1.329	1.490
A_{23}	70.845	69.845	70.090	70.303	69.852	70.052	70.393	70.368	69.819
A_{24}	0.852	0.645	0.645	0.645	0.819	1.084	0.645	0.742	0.645
A_{25}	78.381	75.471	76.523	76.755	76.490	76.490	76.845	76.819	76.271
A_{26}	10.561	8.955	6.671	6.968	5.535	6.323	5.568	6.006	5.929
A_{27}	32.277	31.948	43.110	41.690	44.477	43.077	44.626	44.897	44.232
A_{28}	60.355	56.774	69.748	69.671	72.058	70.400	70.755	70.090	70.987
A_{29}	97.368	94.613	89.290	89.819	87.761	88.819	88.219	88.806	88.084
W_{best} (kg)	11542.610	11410.796	11541.944	11561.230	11588.334	11537.798	11544.007	11595.009	11500.491
CVE_{max} (%)	3.69	9.97	0.071	None	None	None	None	None	None

Table 6 and Fig. 26 illustrate the optimal hyper-parameters found after 30 training times. Additionally, the optimum results with respect to the optimal network, which include the weight, design variables, constraints, and statistics, are reported in Tables 22 and 23. It is interesting that in this structure, the optimum weights obtained by the other studies (DE: 11595.009 kg; Pierezan [90]: 11544.007 kg; Degertekin [89]: 11561.230 kg; DUL [60]: 11588.334 kg; and PINEFN [60]: 11537.798 kg) are much larger than the proposed approach (11500.491 kg). Clearly, our framework saves over 30 kg compared to the second-best approach PINEFN. From the data in Table 23, it can easily be seen that the range of confidence interval (11501.959 kg to 11502.104 kg) changes for narrow and close to the worst (11503.045 kg), mean (11502.031 kg), and best (11500.491 kg) weights with the small Std (0.333 kg). More importantly, all constraint values found by FNN are very close and satisfy the allowable stresses. Meanwhile, the objective and constraints found by DE have a large standard deviation and the maximum values are still quite far from our result. And clearly, this work gives the best result in terms of both the constraints and optimum weight. Although the DUL, PINEFN, and FNN models based on the gradient descent method all utilize the neural network as the backbone, our paradigm outperforms the other two algorithms. This can easily be explained by the fact that the gradient-based approaches are very sensitive to the choice of the starting point. And its position is influenced heavily by the hyper-parameters of the network. Therein, the DUL and PINEFN models fix all hyper-parameters of the network during the whole training process. Hence, both algorithms may become trapped in local optima without tuning hyper-parameters. Meanwhile, our framework has completely overcome this drawback by using BO for automatic tuning of hyper-parameters of the network. And this process is the automatic selection of the starting point, when changing the hyper-parameters lead to change the position of starting point. Therefore, the FNN is capable of effectively handling design problems that contain local minima, as well as improving accuracy. The loss convergence histories of the DE, PINEFN, and FNN are depicted in Fig. 27. Clearly, the learning curve of the proposed method converges much more rapidly than those of the other algorithms. It achieves the optimal weight around 4000 analyses, while the DE and PINEFN demand 35000 and 4500, respectively. Consequently, this once again demonstrates the efficiency of the automatic hyper-parameter tuning of the DNN for solving structural optimization problems.

Table 23
Statistics of the constraints and weight for the 200-bar planar truss.

Metric	DE			FNN		
	σ_{112} (MPa)	σ_{168} (MPa)	W (kg)	σ_{200} (MPa)	σ_{121} (MPa)	W (kg)
Min	-68.948	1.481	11595.010	-68.948	68.915	11500.491
Max	-7.472	60.840	12051.035	-68.946	68.948	11503.045
Mean	-17.827	5.218	11718.071	-68.947	68.939	11502.031
Std	1.898	1.956	19.851	0.000	0.005	0.333
95% CIU	-18.506	4.518	11710.968	-68.947	68.940	11502.104
95% CIL	-17.148	5.917	11725.175	-68.947	68.938	11501.959

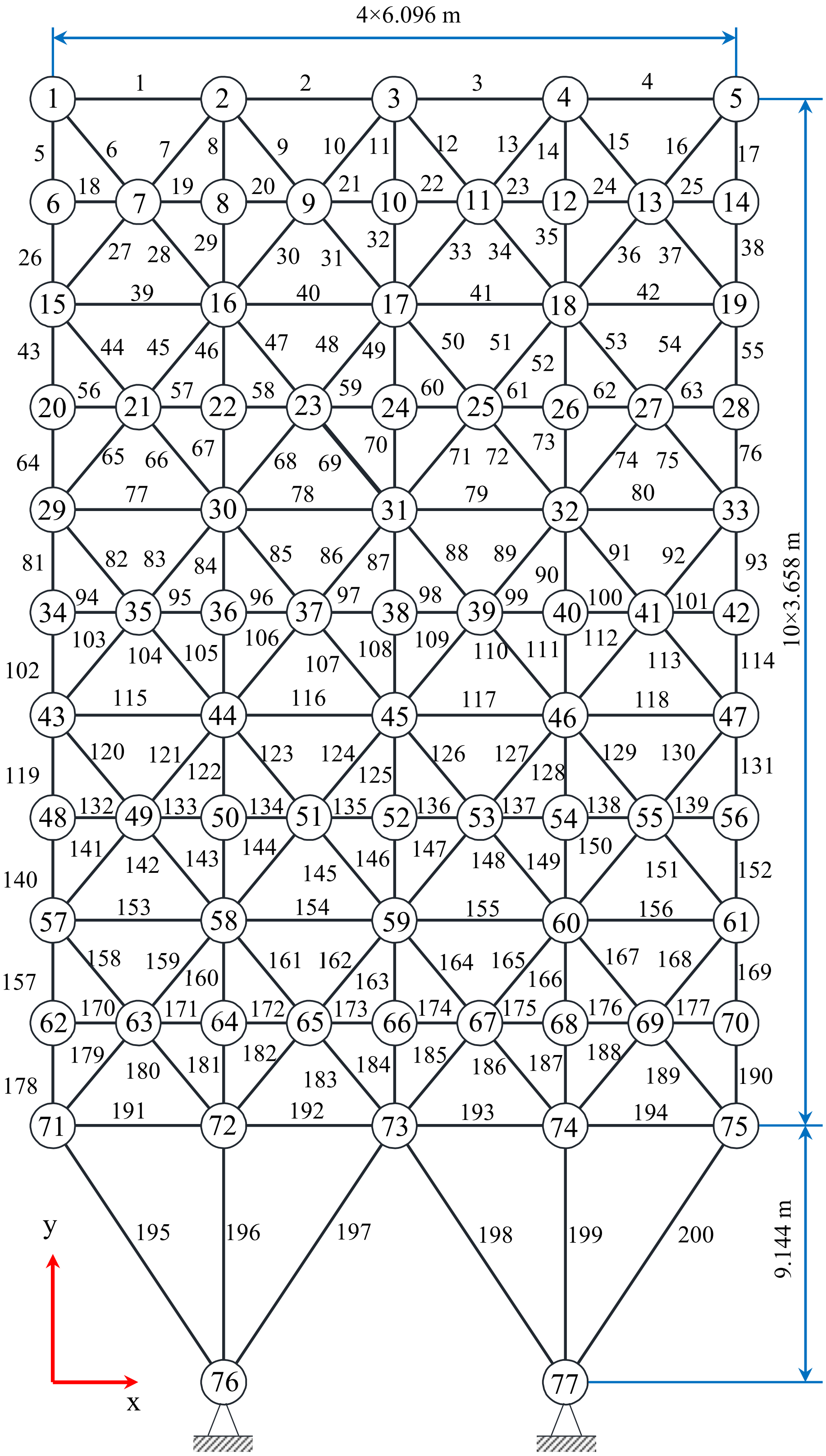


Fig. 25. A 200-bar planar truss structure.

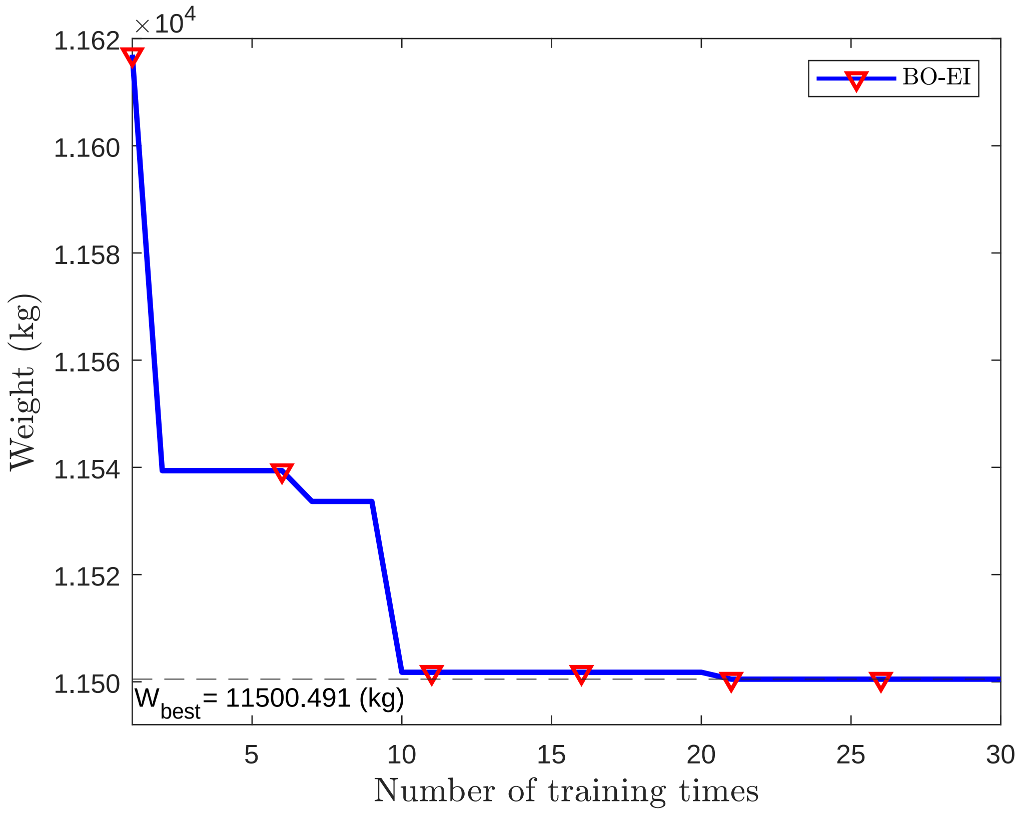


Fig. 26. Convergence history of the HPO using BO for the 200-bar planar truss.

4. Discussion

From Table 24, it is easily noticeable that the FNN requires significantly fewer structural analyses per run compared to the DE across all problems. More concretely, it takes only 1000 and 5000 analyses in numerical examples 1-5 and 6, respectively. In contrast, the DE converges much more slowly, requiring 6 to 28 times the average number of structural analyses (Avg) compared to our model. In addition, compared to the neural network-based approaches, this study not only achieves better optimal weights but also converges faster than the DUL and PINEFN, especially in large-scale problems. Clearly, the hyper-parameters tuning process provides a good starting point which helps to improve performance. Furthermore, this is partially due to the self-normalized training data, which also increase the convergence rate for the training process. In terms of the computational time, the efficiency of the DE method performs better than the FNN, but only for optimization problems with few design variables (ie less than 10), such as those given in numerical examples 1, 3, and 5. When the number of design vari-

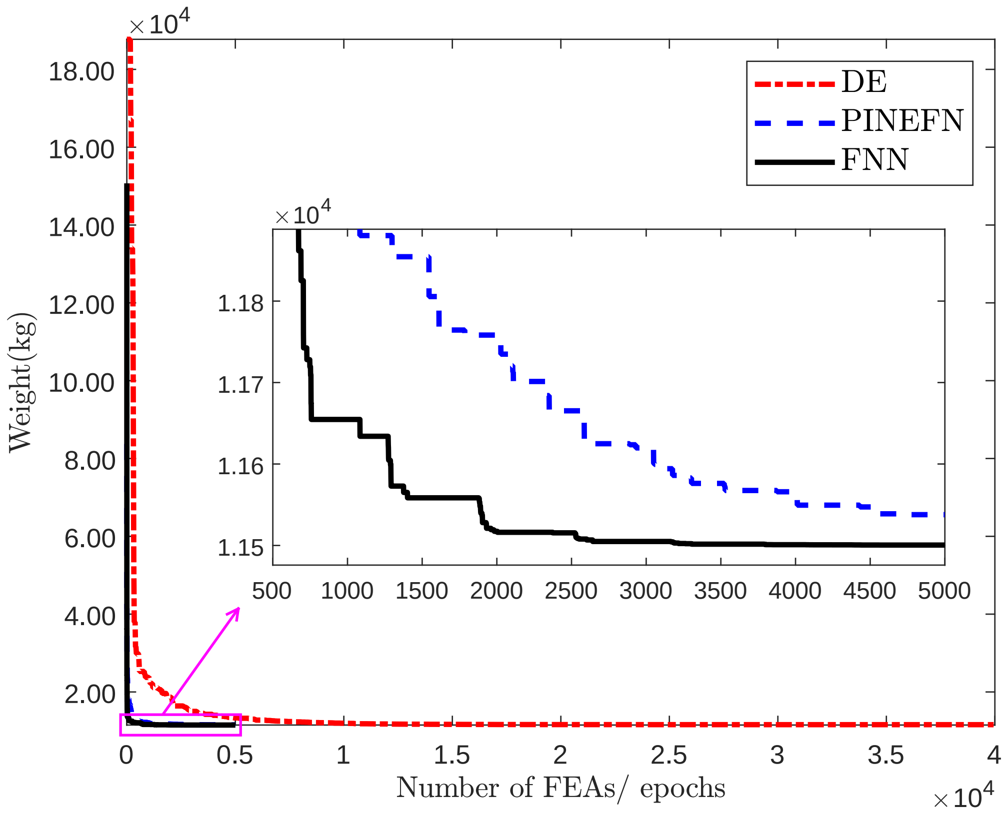


Fig. 27. Weight convergence histories of the 200-bar planar truss using the FNN and other works.

ables and the complexity of structures increase, the computational cost of the DE is higher than the FNN for the other structures. Specifically, the total times of the FNN (357.681 s) is less than the DE (410.076 s) for the 17-bar planar truss. For the 72-bar space truss, it takes only 545.927 s and 1116.490 s to gain optimal solutions, while the DE requires 932.225 s and 2144.720 s for the first and second cases, respectively. Meanwhile, its computational cost (4049.605 s) is reduced by more than three times compared to the DE (15857.088 s) for the 200-bar planar truss with the largest design variable (29). This can be easily explained by the fact that the DE is a population-based optimization algorithm. Hence, it requires a larger number of function evaluations to effectively explore the search space for high-dimensional problems [91]. And this is clearly shown in the examples 2, 4, and 6. Contrary to the DE algorithm, our framework relies on the gradient based optimization strategy using automatic differentiation tools to evaluate the sensitivity. Thus this demonstrated the cost-effectiveness of the FNN for the large-scale and complex structures compared with the conventional algorithms.

Table 24
Efficiency of the different algorithms.

Example	No. of elements	No. of dofs	No. of design variables	DE Total times (s)	DE Avg	DE W_{best} (kg)	FNN Total times (s)	FNN Avg	FNN W_{best} (kg)
10-bar planar truss (Case 1)	10	8	10	206.145	9410	2295.580	239.261	1000	2295.655
10-bar planar truss (Case 2)				214.245	11214	2121.435	256.230	1000	2121.507
17-bar planar truss	17	14	17	410.076	28563	1171.133	357.681	1000	1171.129
25-bar space truss	25	18	8	294.685	7495	247.282	354.144	1000	247.321
72-bar space truss (Case 1)	72	48	16	932.225	15689	167.669	545.927	1000	167.676
72-bar space truss (Case 2)				2144.720	22691	165.708	1116.490	1000	165.099
120-bar dome truss	120	111	7	781.164	6793	14741.630	854.994	1000	14741.589
200-bar planar truss	200	150	29	15857.088	49325	11595.009	4049.605	5000	11500.491

In terms of the accuracy, it is easily observed that the optimum weights found by the FNN are a good agreement with the DE, DUL, and PINEFN for the examples 1-5 with very small deviation. For the 200-bar planar truss, our approach (11500.491 kg) achieves the lightest weight structure saving from 37 to 95 kg, compared to the alternative methods (DE: 11595.009 kg; DUL: 11588.334 kg; PINEFN: 11537.007 kg) while satisfying all constraints. It means that these existing methods converge towards a local optimum instead of the global optimum. Clearly, their performance is influenced by the choice of algorithm parameters. More concretely, the DE, also known as the gradient-free algorithm, is sensitive to control parameters, such as population size, mutation factor, and crossover rate, when the dimensionality of the problem increases. And its ability to balance exploration and exploitation to locate the optimal solution is closely related to parameter tuning. This is one major drawback that leads to inefficiency and more time-consuming optimization processes [91]. Meanwhile, the DUL and PINEFN based on the gradient of the neural network may become trapped in a local minimum due to the position of the initial starting point, which depends on the hyper-parameters. In these two approaches, user experience was applied to select them. Hence, their accuracy is strongly dependent on prior knowledge and experience. For our framework, its most outstanding characteristic is that BO allows for the automatic tuning of the network's hyper-parameters. Therefore, FNN is capable of effectively handling large-scale problems, improving in terms of speed of convergence as well as escaping local minima. In addition, it yields a simple and easily applied model due to automatically calculating sensitivity.

5. Conclusions

In this article, an efficient FNN-based framework was successfully developed for the design optimization of truss structures. In order to achieve this goal, a deep neural network with automatic hyper-parameters tuning using BO was constructed to guide the learning process by minimizing the loss function, which was designed based on the weight and constraint functions of the structure with supporting FM. And the optimum weight of the structure was found immediately at the training end corresponding to the minimum loss. The simplicity, effectiveness, and reliability of the proposed approach were demonstrated numerical examples for the size optimization of truss structures. The obtained results revealed that our paradigm outperforms previously released works in solution quality, convergence speed, and computational efficiency for the large-scale problem. One outstanding characteristic is that the connectivity matrix was considered as the self-normalized and unlabeled training data without using any structural analyses as well as sampling techniques. Hence, its learning possibility only relies upon the connectivity information, which are known as the input data. In addition, a potentially more interesting point is that it could automatically tune hyper-parameters to avoid being trapped in a local optimum. On the other hand, the sensitivity calculation became easy and simple to determine by automatic differentiation tools. Owing to these aforementioned excellent properties, FNN shows great potential as an alternative approach for addressing complex issues in structural optimization.

Despite its advantages, the FNN may face the computing challenges that have yet to be resolved. Firstly, this model only can solve the sizing optimization of truss structures. Therefore, future studies can extend to the size and shape optimization, topology optimization, and reliability-based design optimization problems. Next, it cannot handle scenarios of unforeseen conditions, such as loads, material properties, constraints, and so on, while still accurately predicting the optimal weight without conducting the optimization again. To overcome this challenge, the integration of FNN and transfer learning offers a promising approach for generalizing to new conditions, enabling the prediction of optimal weight without requiring model re-training for future developments. In addition, this work only considers the design optimization of structures with linear behavior. However, in reality, all structures exhibit nonlinear responses in some way, so these need to be considered in the optimization process. And this is one of the future directions to fully understand real structural performance. And finally, a multi-fidelity model that integrates the FNN with high-fidelity data obtained from experiments is a promising approach for addressing optimization challenges in real-life scenarios.

Author contributions

Dai D. Mai: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - original draft, Writing review & editing, Visualization. **Si T. Do**: Data curation, Validation. **Seunghye Lee**: Data curation, Validation, Funding acquisition, Writing review & editing. **Hau T. Mai**: Conceptualization, Methodology, Investigation, Writing - original draft, Writing review & editing, Supervision.

Declaration of competing interest

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

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2023R1A2C2003310)

Appendix A. Gaussian process

LHS is employed to generate p initial combinations of hyper-parameters in the design space. And then the corresponding minimum loss function values $(\mathcal{L}_{\min})$ are found by training the networks with respect to each combination of hyper-parameters $(\boldsymbol{\beta})$. Based on the obtained training sample set $\left\{\boldsymbol{\beta}_{1:p},\ \boldsymbol{\mathcal{L}}_{\min(1:p)}\right\}$, a GP-based surrogate model is constructed to approximate the objective function for tuning hyper-parameters. The samples follow a multivariate Gaussian distribution $\boldsymbol{\mathcal{L}}_{\min(1:p)} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where the kernel matrix $\mathbf{K}$ is expressed as follows

$$\mathbf{K} = \begin{bmatrix} k(\boldsymbol{\beta}_1, \boldsymbol{\beta}_1) & \cdots & k(\boldsymbol{\beta}_1, \boldsymbol{\beta}_p) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{\beta}_p, \boldsymbol{\beta}_1) & \cdots & k(\boldsymbol{\beta}_p, \boldsymbol{\beta}_p) \end{bmatrix}, \tag{A.1}$$

in which k is the Matérn kernel function. Let $\mathcal{L}_{\min(p+1)}$ denote the minimum loss function value achieved by the network with respect to the next combination of hyper-parameters $\boldsymbol{\beta}_{p+1}$. When $\boldsymbol{\mathcal{L}}_{\min(1:p)}$ and $\mathcal{L}_{\min(p+1)}$ are jointly Gaussian, and they are given by

$$\begin{pmatrix} \boldsymbol{\mathcal{L}}_{\min(1:p)} \\ \mathcal{L}_{\min(p+1)} \end{pmatrix} = \mathcal{N}\left(\mathbf{0},\ \begin{bmatrix} \mathbf{K} & \mathbf{k}_1 \\ \mathbf{k}_1^T & k\left(\boldsymbol{\beta}_{p+1},\ \boldsymbol{\beta}_{p+1}\right) \end{bmatrix} \right), \tag{A.2}$$

with

$$\mathbf{k}_1 = \begin{bmatrix} k\left(\boldsymbol{\beta}_1,\ \boldsymbol{\beta}_{p+1}\right) & k\left(\boldsymbol{\beta}_2,\ \boldsymbol{\beta}_{p+1}\right) & \cdots & k\left(\boldsymbol{\beta}_p,\ \boldsymbol{\beta}_{p+1}\right) \end{bmatrix}^T. \tag{A.3}$$

Based on Bayes' rule, the posterior probability distribution $\mathcal{L}_{\min(p+1)}$ at a next sample $\boldsymbol{\beta}_{p+1}$ can be expressed as

$$\mathbf{P}\left(\mathcal{L}_{\min(p+1)}|\boldsymbol{\beta}_{p+1},\ \boldsymbol{\beta}_{1:p},\ \boldsymbol{\mathcal{L}}_{\min(1:p)}\right) \sim \mathcal{N}\left(\mu\left(\boldsymbol{\beta}_{p+1}\right), \sigma^2\left(\boldsymbol{\beta}_{p+1}\right)\right), \tag{A.4}$$

where $\sigma^2(.)$ and $\mu(.)$ are the covariance function and posterior mean, respectively. They are given by

$$\begin{aligned} \sigma^2\left(\boldsymbol{\beta}_{p+1}\right) &= k\left(\boldsymbol{\beta}_{p+1},\ \boldsymbol{\beta}_{p+1}\right) - \mathbf{k}_1^T[\mathbf{K}+\sigma^2\mathbf{I}]^{-1}\mathbf{k}_1, \\ \mu\left(\boldsymbol{\beta}_{p+1}\right) &= \mathbf{k}_1^T[\mathbf{K}+\sigma^2\mathbf{I}]^{-1}\boldsymbol{\mathcal{L}}_{\min(1:p)}. \end{aligned} \tag{A.5}$$

At each iteration of BO, the acquisition function, also known as the infill strategy, is used to guide the selection of the next set of the hyper-parameters to evaluate. It plays a central role in the tuning process. There are three most widely used acquisition functions, namely LCB, PI, EI. And their mathematical expressions are reformulated as follows

$$LCB\left(\boldsymbol{\beta}\right) = \mu\left(\boldsymbol{\beta}\right) - \lambda\sigma\left(\boldsymbol{\beta}\right), \tag{A.6}$$

$$PI\left(\boldsymbol{\beta}\right) = \frac{\mu\left(\boldsymbol{\beta}\right) - W_{\min}\left(\boldsymbol{\beta}^+\right)}{\sigma\left(\boldsymbol{\beta}\right)}, \tag{A.7}$$

$$EI\left(\boldsymbol{\beta}\right) = \left(\mu\left(\boldsymbol{\beta}\right) - \mathcal{L}_{\min}\left(\boldsymbol{\beta}^+\right)\right)\Phi\left(\xi\right) + \sigma\left(\boldsymbol{\beta}\right)\phi\left(\xi\right), \tag{A.8}$$

where

$$\xi = \frac{\mu\left(\boldsymbol{\beta}\right) - \mathcal{L}_{\min}\left(\boldsymbol{\beta}^+\right)}{\sigma\left(\boldsymbol{\beta}\right)}, \tag{A.9}$$

in which $\lambda > 0$ is a trade-off parameter between exploration and exploitation; $\boldsymbol{\beta}^+$ represents the current best hyper-parameters obtained from the surrogate model $\mathcal{L}_{\min}$; $\Phi(.)$ and $\phi(.)$ are the standard normal cumulative distribution and probability density functions, respectively.

References

[1] A. Manguri, N. Saeed, F. Kazemi, M. Szczepanski, R. Jankowski, Optimum number of actuators to minimize the cross-sectional area of prestressable cable and truss structures, in: Structures, volume 47, Elsevier, 2023, pp. 2501–2514.

[2] A. Manguri, N. Saeed, F. Kazemi, N. Asgarkhani, M. Szczepanski, R. Jankowski, Computational bar size optimization of single layer dome structures considering axial stress and shape disturbance, in: International Conference on Mathematical Modeling in Physical Sciences, Springer, 2023, pp. 173–185.

[3] N. Khot, L. Berke, Structural optimization using optimality criteria methods (1984).

[4] M. P. Bendsøe, A. Ben-Tal, Truss topology optimization by a displacements based optimality criterion approach, in: Optimization of large structural systems, Springer, 1993, pp. 139–155.

[5] M. Saka, Optimum geometry design of roof trusses by optimality criteria method, Computers & structures 38 (1991) 83–92.

[6] G. Rozvany, M. Zhou, A note on truss design for stress and displacement constraints by optimality criteria methods, Structural optimization 3 (1991) 45–50.

[7] W. Gu, Z. Gürdal, S. Missoum, Elastoplastic truss design using a displacement based optimization, Computer methods in applied mechanics and engineering 191 (2002) 2907–2924.

[8] Q. Liu, J. Paavola, J. Zhang, Shape and cross-section optimization of plane trusses subjected to earthquake excitation using gradient and hessian matrix calculations, Mechanics of Advanced Materials and Structures 23 (2016) 156–169.

[9] L. Schmit Jr, B. Farshi, Some approximation concepts for structural synthesis, AIAA journal 12 (1974) 692–699.

[10] M. Saka, M. Ulker, Optimum design of geometrically nonlinear space trusses, Computers & Structures 42 (1992) 289–299.

[11] M.-K. Shin, K.-J. Park, G.-J. Park, Optimization of structures with nonlinear behavior using equivalent loads, Computer Methods in Applied Mechanics and Engineering 196 (2007) 1154–1167.

[12] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (1997) 341–359.

[13] J. Holand, Adaptation in natural and artificial systems. university of michigan press, Ann Arbor 1 (1975) 1.

[14] Q. X. Lieu, D. T. Do, J. Lee, An adaptive hybrid evolutionary firefly algorithm for shape and size optimization of truss structures with frequency constraints, Computers & Structures 195 (2018) 99–112.

[15] R. V. Rao, V. J. Savsani, D. P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-aided design 43 (2011) 303–315.

[16] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, ieee, 1995, pp. 1942–1948.

[17] S. Kumar, G. G. Tejani, P. Mehta, S. M. Sait, A. R. Yildiz, S. Mirjalili, Optimization of truss structures using multi-objective cheetah optimizer, Mechanics Based Design of Structures and Machines (2024) 1–22.

[18] N. Panagant, N. Pholdee, S. Bureerat, A. R. Yildiz, S. Mirjalili, A comparative study of recent multi-objective metaheuristics for solving constrained truss optimisation problems, Archives of Computational Methods in Engineering (2021) 1–17.

[19] S. Debnath, S. Debbarma, S. Nama, A. K. Saha, R. Dhar, A. R. Yildiz, A. H. Gandomi, Centroid opposition-based backtracking search algorithm for global optimization and engineering problems, Advances in Engineering Software 198 (2024) 103784.

[20] S. Kumar, B. S. Yildiz, P. Mehta, N. Panagant, S. M. Sait, S. Mirjalili, A. R. Yildiz,

Chaotic marine predators algorithm for global optimization of real-world engineering problems, Knowledge-Based Systems 261 (2023) 110192.

[21] T. Kunakote, N. Sabangban, S. Kumar, G. G. Tejani, N. Panagant, N. Pholdee, S. Bureerat, A. R. Yildiz, Comparative performance of twelve metaheuristics for wind farm layout optimisation, Archives of Computational Methods in Engineering (2022) 1–14.

[22] P. Mehta, B. S. Yildiz, S. M. Sait, A. R. Yildiz, Hunger games search algorithm for global optimization of engineering design problems, Materials Testing 64 (2022) 524–532.

[23] H. Abderazek, A. R. Yildiz, S. M. Sait, Mechanical engineering design optimisation using novel adaptive differential evolution algorithm, International Journal of vehicle design 80 (2019) 285–329.

[24] B. S. Yıldız, S. Kumar, N. Panagant, P. Mehta, S. M. Sait, A. R. Yildiz, N. Pholdee, S. Bureerat, S. Mirjalili, A novel hybrid arithmetic optimization algorithm for solving constrained optimization problems, Knowledge-Based Systems 271 (2023) 110554.

[25] H.-Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, Journal of global optimization 27 (2003) 105–129.

[26] H. T. Mai, S. Lee, J. Kang, J. Lee, A damage-informed neural network framework for structural damage identification, Computers & Structures 292 (2024) 107232.

[27] N. Asgarkhani, F. Kazemi, A. Jakubczyk-Gałczyńska, B. Mohebi, R. Jankowski, Seismic response and performance prediction of steel buckling-restrained braced frames using machine-learning methods, Engineering Applications of Artificial Intelligence 128 (2024) 107388.

[28] F. Kazemi, N. Asgarkhani, R. Jankowski, Optimization-based stacked machine-learning method for seismic probability and risk assessment of reinforced concrete shear walls, Expert Systems with Applications 255 (2024) 124897.

[29] F. Kazemi, N. Asgarkhani, R. Jankowski, Machine learning-based seismic fragility and

seismic vulnerability assessment of reinforced concrete structures, Soil Dynamics and Earthquake Engineering 166 (2023) 107761.

[30] Z. Meng, Q. Qian, M. Xu, B. Yu, A. R. Yıldız, S. Mirjalili, Pinn-form: a new physics-informed neural network for reliability analysis with partial differential equation, Computer Methods in Applied Mechanics and Engineering 414 (2023) 116172.

[31] H. T. Mai, Q. X. Lieu, J. Kang, J. Lee, A robust unsupervised neural network framework for geometrically nonlinear analysis of inelastic truss structures, Applied Mathematical Modelling 107 (2022) 332–352.

[32] H. T. Mai, T. T. Truong, J. Kang, D. D. Mai, J. Lee, A robust physics-informed neural network approach for predicting structural instability, Finite Elements in Analysis and Design 216 (2023) 103893.

[33] D. D. Mai, T. D. Bao, T.-D. Lam, H. T. Mai, Physics-informed neural network for nonlinear analysis of cable net structures, Advances in Engineering Software 196 (2024) 103717.

[34] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561 (2017).

[35] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational physics 378 (2019) 686–707.

[36] L. Berke, P. Hajela, Applications of artificial neural nets in structural mechanics, Springer, 1992.

[37] H. Adeli, H. S. Park, Optimization of space structures by neural dynamics, Neural networks 8 (1995) 769–781.

[38] J. Ramasamy, S. Rajasekaran, Artificial neural network and genetic algorithm for the

design optimizaton of industrial roofs—a comparison, Computers & Structures 58 (1996) 747–755.

[39] H. T. Mai, J. Kang, J. Lee, A machine learning-based surrogate model for optimization of truss structures with geometrically nonlinear behavior, Finite Elements in Analysis and Design 196 (2021) 103572.

[40] B. Li, C. Huang, X. Li, S. Zheng, J. Hong, Non-iterative structural topology optimization using deep learning, Computer-Aided Design 115 (2019) 172–180.

[41] D. A. White, W. J. Arrighi, J. Kudo, S. E. Watts, Multiscale topology optimization using neural network surrogate models, Computer Methods in Applied Mechanics and Engineering 346 (2019) 1118–1135.

[42] H. Chi, Y. Zhang, T. L. E. Tang, L. Mirabella, L. Dalloro, L. Song, G. H. Paulino, Universal machine learning for topology optimization, Computer Methods in Applied Mechanics and Engineering 375 (2021) 112739.

[43] S. M. Sait, P. Mehta, N. Pholdee, B. S. Yıldız, A. R. Yıldız, Artificial neural network infused quasi oppositional learning partial reinforcement algorithm for structural design optimization of vehicle suspension components, Materials Testing (2024).

[44] S. M. Sait, P. Mehta, A. R. Yıldız, B. S. Yıldız, Optimal design of structural engineering components using artificial neural network-assisted crayfish algorithm, Materials Testing (2024).

[45] M. U. Erdaş, M. Kopar, B. S. Yildiz, A. R. Yildiz, Optimum design of a seat bracket using artificial neural networks and dandelion optimization algorithm, Materials Testing 65 (2023) 1767–1775.

[46] L. Linden, D. K. Klein, K. A. Kalina, J. Brummund, O. Weeger, M. Kästner, Neural networks meet hyperelasticity: A guide to enforcing physics, Journal of the Mechanics and Physics of Solids 179 (2023) 105363.

[47] W. Li, M. Z. Bazant, J. Zhu, A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches, Computer Methods in Applied Mechanics and Engineering 383 (2021) 113933.

[48] D. Chakraborty, Artificial neural network based delamination prediction in laminated composites, Materials & design 26 (2005) 1–7.

[49] J.-H. Bastek, D. M. Kochmann, Physics-informed neural networks for shell structures, European Journal of Mechanics-A/Solids 97 (2023) 104849.

[50] D. T. Trinh, K. A. Luong, J. Lee, An analysis of functionally graded thin-walled beams using physics-informed neural networks, Engineering Structures 301 (2024) 117290.

[51] J. He, C. Chadha, S. Kushwaha, S. Koric, D. Abueidda, I. Jasiuk, Deep energy method in topology optimization applications, Acta Mechanica 234 (2023) 1365–1379.

[52] H. Jeong, J. Bai, C. P. Batuwatta-Gamage, C. Rathnayaka, Y. Zhou, Y. Gu, A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization, Engineering Structures 278 (2023) 115484.

[53] H. Jeong, C. Batuwatta-Gamage, J. Bai, Y. M. Xie, C. Rathnayaka, Y. Zhou, Y. Gu, A complete physics-informed neural network-based framework for structural topology optimization, Computer Methods in Applied Mechanics and Engineering 417 (2023) 116401.

[54] A. Singh, S. Chakraborty, R. Chowdhury, A dual physics-informed neural network for topology optimization, arXiv preprint arXiv:2410.14342 (2024).

[55] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Advances in neural information processing systems 34 (2021) 26548–26560.

[56] V. Bolon-Canedo, B. Remeseiro, Feature selection in image analysis: a survey, Artificial Intelligence Review 53 (2020) 2905–2931.

[57] H. Kabir, N. Garg, Machine learning enabled orthogonal camera goniometry for accurate and robust contact angle measurements, Scientific reports 13 (2023) 1497.

[58] O. Hasançebi, Adaptive evolution strategies in structural optimization: Enhancing their computational performance with applications to large-scale structures, Computers & structures 86 (2008) 119–132.

[59] M. Sonmez, Artificial bee colony algorithm for optimization of truss structures, Applied Soft Computing 11 (2011) 2406–2418.

[60] H. T. Mai, D. D. Mai, J. Kang, J. Lee, J. Lee, Physics-informed neural energy-force network: a unified solver-free numerical simulation for structural optimization, Engineering with Computers (2023) 1–24.

[61] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, et al., Jax: composable transformations of python+ numpy programs (2018).

[62] A. Chandrasekhar, S. Sridhara, K. Suresh, Auto: a framework for automatic differentiation in topology optimization, Structural and Multidisciplinary Optimization 64 (2021) 4355–4365.

[63] A. Chandrasekhar, K. Suresh, Tounn: Topology optimization using neural networks, Structural and Multidisciplinary Optimization 63 (2021) 1135–1149.

[64] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[65] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization., Journal of machine learning research 13 (2012).

[66] F. Hutter, L. Kotthoff, J. Vanschoren, Automated machine learning: methods, systems, challenges, Springer Nature, 2019.

[67] M. Paul, Applied machine learning, `https://cmci.colorado.edu/classes/INFO-4604/resources.html`, 2018. Accessed: 2021–04-19.

[68] H. T. Mai, Q. X. Lieu, J. Kang, J. Lee, A novel deep unsupervised learning-based framework for optimization of truss structures, Engineering with Computers 39 (2023) 2585–2608.

[69] H. T. Mai, S. Lee, D. Kim, J. Lee, J. Kang, J. Lee, Optimum design of nonlinear structures via deep neural network-based parameterization framework, European Journal of Mechanics-A/Solids 98 (2023) 104869.

[70] M. Korenciak, Go game move prediction using convolutional neural network (2018).

[71] L. Li, Z. Huang, F. Liu, Q. Wu, A heuristic particle swarm optimizer for optimization of pin connected structures, Computers & Structures 85 (2007) 340–349.

[72] K. S. Lee, Z. W. Geem, A new structural optimization method based on the harmony search algorithm, Computers & structures 82 (2004) 781–798.

[73] P. Rizzi, Optimization of multi-constrained structures based on optimality criteria?, in: 17th structures, structural dynamics, and materials conference, 1976, p. 1547.

[74] H. Adeli, S. Kumar, Distributed genetic algorithm for structural optimization, Journal of Aerospace engineering 8 (1995) 156–163.

[75] C. V. Camp, Design of space trusses using big bang–big crunch optimization, Journal of Structural Engineering 133 (2007) 999–1008.

[76] S. Degertekin, M. Hayalioglu, Sizing truss structures using teaching-learning-based optimization, Computers & Structures 119 (2013) 177–188.

[77] A. Kaveh, S. Talatahari, Size optimization of space trusses using big bang–big crunch algorithm, Computers & structures 87 (2009) 1129–1140.

[78] G. Bekdaş, S. M. Nigdeli, X.-S. Yang, Sizing optimization of truss structures using flower pollination algorithm, Applied Soft Computing 37 (2015) 322–331.

[79] E. Pouriyanezhad, H. Rahami, S. Mirhosseini, Truss optimization using eigenvectors of the covariance matrix, Engineering with Computers 37 (2021) 2207–2224.

[80] H. Adeli, O. Kamal, Efficient optimization of space trusses, Computers & structures 24 (1986) 501–511.

[81] K. C. Sarma, H. Adeli, Fuzzy genetic algorithm for optimization of steel structures, Journal of Structural Engineering 126 (2000) 596–604.

[82] A. M. Committee, A. I. of Steel Construction, Manual of Steel Construction: Allowable Stress Design, American Institute of Steel Construction, 1989.

[83] A. Kaveh, S. Talatahari, Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, Computers & Structures 87 (2009) 267–283.

[84] A. Kaveh, S. Talatahari, Optimal design of skeletal structures via the charged system search algorithm, Structural and Multidisciplinary Optimization 41 (2010) 893–911.

[85] A. Kaveh, T. Bakhshpoori, Optimum design of space trusses using cuckoo search algorithm with levy flights (2013).

[86] S. Talatahari, M. Kheirollahi, C. Farahmandpour, A. H. Gandomi, A multi-stage particle swarm for optimum design of truss structures, Neural Computing and Applications 23 (2013) 1297–1309.

[87] A. Kaveh, T. Bakhshpoori, E. Afshari, An efficient hybrid particle swarm and swallow swarm optimization algorithm, Computers & Structures 143 (2014) 40–59.

[88] L. Lamberti, An efficient simulated annealing algorithm for design optimization of truss structures, Computers & Structures 86 (2008) 1936–1953.

[89] S. Degertekin, Improved harmony search algorithms for sizing optimization of truss structures, Computers & Structures 92 (2012) 229–241.

[90] J. Pierezan, L. dos Santos Coelho, V. C. Mariani, E. H. de Vasconcelos Segundo, D. Prayogo, Chaotic coyote algorithm applied to truss optimization problems, Computers & Structures 242 (2021) 106353.

[91] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied soft computing 11 (2011) 1679–1696.