Engineering with Computers
https://doi.org/10.1007/s00366-022-01636-3

ORIGINAL ARTICLE

A novel deep unsupervised learning-based framework for optimization of truss structures

Hau T. Mai[1,2] · Qui X. Lieu[3,4] · Joowon Kang[5] · Jaehong Lee[1]

Received: 5 October 2021 / Accepted: 13 February 2022

Abstract

In this paper, an efficient deep unsupervised learning (DUL)-based framework is proposed to directly perform the design optimization of truss structures under multiple constraints for the first time. Herein, the members' cross-sectional areas are parameterized using a deep neural network (DNN) with the middle spatial coordinates of truss elements as input data. The parameters of the network, including weights and biases, are regarded as decision variables of the structural optimization problem, instead of the member's cross-sectional areas as those of traditional optimization algorithms. A new loss function of the network model is constructed with the aim of minimizing the total structure weight so that all constraints of the optimization problem via unsupervised learning are satisfied. To achieve the optimal parameters, the proposed model is trained to minimize the loss function by a combination of the standard gradient optimizer and backpropagation algorithm. As soon as the learning process ends, the optimum weight of truss structures is indicated without utilizing any other time-consuming metaheuristic algorithms. Several illustrative examples are investigated to demonstrate the efficiency of the proposed framework in requiring much lower computational cost against other conventional methods, yet still providing high-quality optimal solutions.

Keywords Unsupervised learning · Deep neural network · Loss function · Truss optimization

✉ Jaehong Lee
jhlee@sejong.ac.kr

Hau T. Mai
maitienhaunx@gmail.com; maitienhau@iuh.edu.vn

Qui X. Lieu
lieuxuanqui@hcmut.edu.vn

Joowon Kang
kangj@ynu.ac.kr

[1] Deep Learning Architectural Research Center, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea

[2] Faculty of Mechanical Engineering, Industrial University of Ho Chi Minh City, 12 Nguyen Van Bao Street, Ward 4, Go Vap District, Ho Chi Minh City 70000, Vietnam

[3] Faculty of Civil Engineering, Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, Ward 14, District 10, Ho Chi Minh City, Vietnam

[4] Vietnam National University Ho Chi Minh City (VNU-HCM), Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam

[5] School of Architecture, Yeungnam University, 280, Daehak-Ro, Gyeongsan, Gyeongbuk 38541, Republic of Korea

1 Introduction

Size optimization of truss structures has attracted the considerable attention of many scholars during the last decades. As known, this problem often minimizes the structural weight to mandatorily satisfy multiple constraints. As indicated in Refs. [1–3], the constraints of such problems are non-convex and highly nonlinear functions. There have been a variety of algorithms proposed for solving this issue, and they are usually categorized into two types: gradient and gradient-free. The first one relies on the gradient information to search optimum solutions. For instance, an optimality criterion was developed by Khot et al. [4, 5] to minimize the weight of the truss structure. El-Sayed et al. [6] described a combining model using linear and nonlinear goal programming. Besides, the optimality criteria integrated with the nonlinear analysis technique were delivered by Saka and Ulker [7]. Although these methods have achieved certain successes, they often encounter several challenges as indicated in Ref. [8]. In specific, it can fail due to the unavailable gradient information of the objective and constraints functions. Among different approaches, evolutionary and

Published online: 08 April 2022

population-based metaheuristic algorithms have been extensively applied to solve optimization problems for truss structures, such as improved differential evolution (IDE) [2], firefly algorithm (FA) [9], adaptive hybrid evolutionary FA (AHEFA) [10], particle swarm algorithm (PSO) [11], democratic PSO (DPSO) [12], genetic algorithms (GAs) [13], hybrid optimality criterion and genetic algorithm (OC-GA) [14], modified coyote optimization algorithm (MCOA) [15], harmony search (HS) [16], evolutionary ant colony optimization (EACO) [17], improved grey wolf optimizer (IGWO) [18], efficient HS (EHS) [19], and balancing composite motion [20], etc. Despite these algorithms can achieve near-global optimum solutions, they also suffer from several drawbacks, such as high computational cost, slow convergence speed, and many turning parameters [21].

Over the past decade, machine learning (ML) has demonstrated to be effective and promising in many fields to assist decision-making in image recognition [22], self-driving systems [23], medical diagnoses [24], financial services [25], structural analysis [26, 27], to name a few. Indeed, it has displayed salient advantages over conventional methods in handling complex practical problems that do not have a closed-form expression. Recently, DNN which is one of the most widely used ML models has attracted remarkable attention in computational mechanics, for example, structural analysis [28, 29], structural health monitoring [30–33], materials sciences [34–36], and structural optimization is no exception. Since the 1990s, a neural network (NN) combined with other algorithms was developed for optimization of truss structures [37–42]. More recently, Mai et al. [43] proposed an integrated model of DNN and differential evolution (DE) algorithm for optimization of truss structures with geometrically nonlinear behavior. The data-driven ML approach was introduced by Long [44] for solving structural optimization problems. In addition, DNN integrated with the traditional algorithms for tackling multi-objective optimization problems [45, 46]. Besides, it has also been successfully applied to solve topology optimization problems of structures. For instance, Li et al. [47] developed an integrated pre-trained network to replace the iterative optimization for designing heat conduction structures. A surrogate model based on NN was proposed by White et al. [48] to perform optimization of macroscale elastic structures. Also, Deng and Albert [49] used DNN to approximate an implicit function of the level set for topology optimization. To decrease computational cost, Abueidda et al. [50] employed a convolutional neural network (CNN) model to optimize structures with geometric nonlinearity. Despite the remarkable success achieved in this area, it is worth mentioning that most of the above studies use a DNN as a surrogate model, and employ supervised learning to construct the model. The above approaches have several disadvantages, making them inefficient such as: (i) they require a time-consuming effort due to the numerical simulation for collecting data of training process; (ii) it is difficult to estimate the suitable training data size, and (iii) they depend strongly on the quality and quantity of data to build the high-accuracy data-driven predictable model. To circumvent these challenges, physics-informed machine learning frameworks, which integrated the governing physics law into the unsupervised learning process, have obtained increasing attention and successfully applied in many tasks, such as structural analysis [51–53], solving partial differential equations (PDEs) [54, 55], and fluid mechanics [56, 57]. For the structural optimization problem, Chandrasekhar and Suresh [58] firstly proposed a direct topology optimization paradigm using unsupervised learning NN. However, in this context, it remains a bottleneck in the computational effort that a part of the sensitivity analysis is performed by direct differentiation. Furthermore, to the best of our knowledge, a multilayer direct DNN model has still not been yet utilized for optimization of truss structure thus far.

As another alternative effective approach for the truss optimization field, this study aims to firstly propose an efficient DNN-based framework using unsupervised learning for optimization of truss structures under multiple constraints. Our work is different from most existing publications based on ML in a number of ways. In the proposed framework, the training data of the network is a set of the central spatial coordinates of all truss members which can be easily collected from the connectivity information of the structure without any numerical simulations, independent sampling techniques, as well as its small size. In addition, the design variables of truss optimization are the weights and biases of the DNN, instead of the cross-sectional areas of truss members. Therein, the sensitivity of loss function to parameters can be easily estimated by an automatic differentiation framework JAX [59] and backpropagation algorithm. And finally, the FEA is only required to assist in building the loss function while optimizer and backpropagation algorithm allow the network to directly participate in the process of structure optimization design. According to that, the optimum weight of truss structures can be indicated as soon as the training process ends with low computational overhead and without utilizing any other algorithms. The performance and applicability of the proposed approach are demonstrated through several benchmark truss optimization examples under various constraints. The outcomes of the proposed DUL are compared with conventional algorithms to evaluate its efficiency and reliability. The experimental results showed that DUL is able to dramatically reduce the computational cost, yet still yield high-quality optimal solutions.

The rest of the paper is organized as follows. The formulation of the size optimization problem of truss structures with multiple constraints is given in Sect. 2. Afterward, the efficient DNN-based structural optimization paradigm is presented in Sect. 3. Next, several numerical examples are

investigated to demonstrate the efficiency of the proposed method in Sect. 4. Finally, conclusions and challenges are outlined in Sect. 5.

2 Sizing optimization

For the size optimization of truss structures, the goal is to minimize the structural mass while still satisfying all design constraints. In which, the cross-sectional areas of truss members are considered as continuous design variables and predefined in a feasible region. The mathematical formulation of this problem can be represented as follows

$$\begin{aligned} \text{Minimize} \quad & W(\mathbf{A}) = \sum_{i=1}^{m} \rho_i A_i L_i, \\ \text{subjected to} \quad & g_j(\mathbf{A}) = \frac{q_j}{[q]_j} - 1 \le 0, \quad j = 1, 2, \ldots, n, \\ & A_k^{\text{low}} \le A_k \le A_k^{\text{up}}, \qquad k = 1, 2, \ldots, m, \end{aligned} \tag{1}$$

where $\mathbf{A}$ is the design variable vector; A_k is the cross-section area of the kth member; $W(.)$ is the mass of the truss structure; L_i and ρ_i are the length and material density of the ith member; g_j is the jth constraint function; q_j denotes the jth displacement, stress or frequency; $[q_j]$ is the jth allowable displacement, stress or frequency; m and n are the number of members and constraints, respectively; A_k^{low} and A_k^{up} are the lower and upper bound of A_k, respectively.

To solve the above optimization problem, a penalty function is used to convert the constrained optimization problem into an unconstrained optimization one [2, 10, 60]. Thus, Eq. (1) can be rewritten as follows

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{A}) = \left(1 + \varepsilon_1 c\right)^{\varepsilon_2} W(\mathbf{A}), \\ & c = \sum_{j=1}^{n} \max\left(0, g_j(\mathbf{A})\right), \end{aligned} \tag{2}$$

where c is the sum of the violated constraints of the design problem; ε_1 and ε_2 are parameters whose values control the exploration and exploitation rates of the design space.

In this work, the following two types of structural optimization problems are addressed: (1) frequency constraints only, and (2) displacement and stress constraints. For the first case, according to Kaveh [61, 62], ε_1 is set to 1 for a better control on other parameters. Note that the first part of Eq. (2) is a penalty function. And its value increases as the exponent ε_2 increases for infeasible designs. Hence, the coefficient ε_2 is set in a way that the penalty decreases. As mentioned by Kaveh [62], this coefficient directly effects the exploration of the algorithm, and its value varies from 1.5 to 3 of the search process. In this study, the value of ε_2 is set to 1.5 at the first epoch and linearly increased by 0.05 in each epoch until it achieves 3 [10, 12] in solving optimization problem under frequency constraints. For the remaining case, a self-adaptive parameters scheme which is proposed by Sonmez [63] and Hasancebi [64] is utilized for the rest of experiments. Contrary to the first case, ε_2 is set to 1, while the coefficient ε_1 is automatically adjust according to the feedback from previous solution, as expressed below

$$\varepsilon_1^{(t)} = \begin{cases} (1/\kappa)\varepsilon_1^{(t-1)} & \text{if } f^{(t-1)} \text{ is feasible,} \\ \kappa\, \varepsilon_1^{(t-1)} & \text{if } f^{(t-1)} \text{ is infeasible,} \end{cases} \tag{3}$$

in which $\varepsilon_1^{(t)}$ and $\varepsilon_1^{(t-1)}$ denote the penalty coefficients at epochs t and $(t-1)$, respectively. ε_1 is set to 1 at the beginning of the epoch. κ is the learning parameter of $\varepsilon_1^{(t)}$, and it is determined by the following equation [63, 64]

$$\kappa = 1 + \frac{1}{n} > 1.01, \tag{4}$$

where n is the total number of constraints.

3 Deep unsupervised learning-based optimization framework

The unconstrained optimization problem presented in Eq. (2) is usually solved by population-based search algorithms as indicated in Introduction. However, they often require a large number of FEA runs to achieve a good solution. In this section, an alternative approach based on DNN is suggested to directly perform optimization of truss structures with lowest computational cost. The overall flowchart of the proposed framework is illustrated in Fig. 1. Here, the weights and biases of the network $\boldsymbol{\theta}$ are design variables. In the context of this paradigm, the initial weight and bias values are randomly chosen according to a normal distribution on interval $[-1, 1]$ for the network setup. The middle spatial coordinates of truss members are treated as the input data, while the cross-sectional areas are represented through parameters and defined as the outputs of the network $\hat{\mathbf{A}}$ obtained by the feedforward phase. Then, the loss function including the objective and constraints is established based on these outputs, and the responses of structures are attained from FEA. JAX, backpropagation, and optimizer are employed to estimate the sensitivity of the loss function to the parameters. To carry out the training, all of the above operations are going on repetitively until the loss function value reaches the minimum corresponding to the optimum weight of truss structures. In general, the proposed approach consists of the following three major components: (i) training data; (ii) DNN, and (iii) the loss function. They are described in detail in the following three subsections.

3.1 Training data

It should be noted that this work relies on unsupervised learning, where the model only has input data, and no corresponding

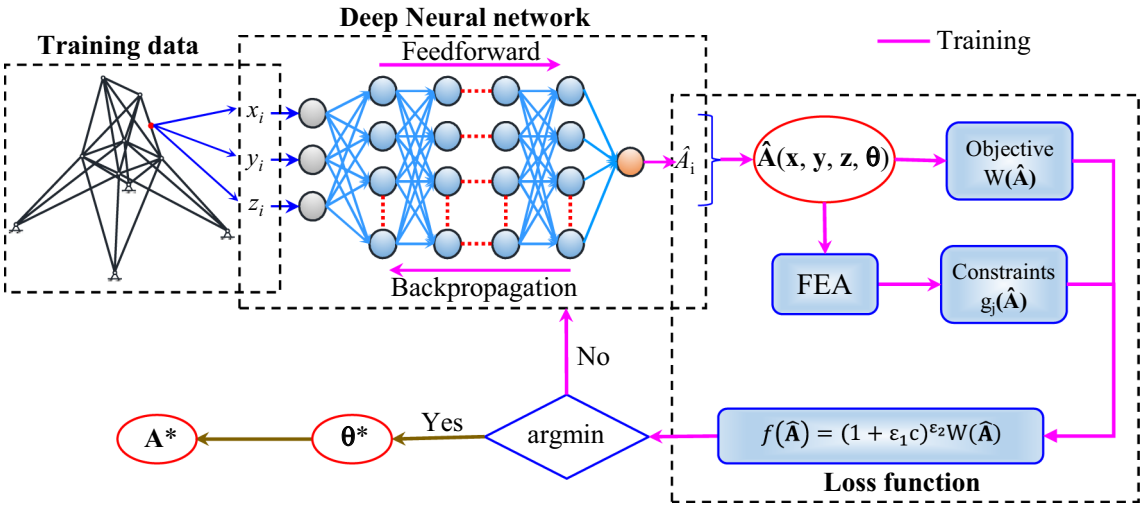

Fig. 1 Flowchart of the DUL-based framework for design optimization

output values are required. Furthermore, the input data only contains the information to describe the problem, such as geometric, material, loads, boundary conditions, and so on. In this work, a set of middle spatial coordinates (x_i, y_i, z_i) of truss elements is selected as training data. Clearly, they are simple and can be easily obtained from known geometry of the structure. Additionally, its size is small ($d \times 2$) and ($d \times 3$) for planar and space truss structures, respectively. Here d denotes the number of truss members, while 2 or 3 implies the number of spatial coordinates for 2- or 3-dimensional truss corresponding to the number of neurons in the input layer. Hence, it helps to reduce the complexity of NN as well as improve the convergence speed.

3.2 Deep neural network

The DNN, which is one of the ML models, is a set of mathematical relationships between the inputs and outputs through a training phase that mimics the way of human brain activities [41, 65]. For illustration, a fully connected DNN with depth L is depicted in Fig. 2. Therein, the first layer is known as the input layer which consists of three neurons corresponding with the spatial coordinates (x, y, z). The middle layers are referred to as hidden layers. Finally, the last layer is called the output layer with one neuron which corresponds to the predicted cross-sectional area ($\hat{A}$). And meanwhile, the number of hidden layers and hidden neurons depend on the complexity of the application.

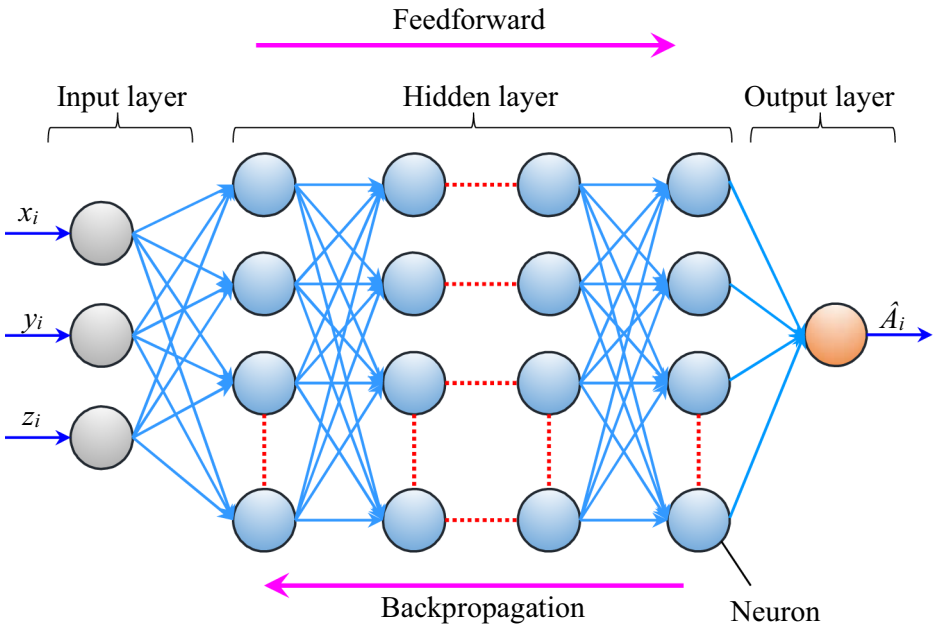

Fig. 2 Architecture of a fully connected DNN

Furthermore, the units of the present layer are connected to all units in the previous layer via weights **W** and biases **b**.

In order to train the network, the feedforward and backpropagation algorithms will be employed to tune the parameters. Specifically, in the feedforward process, a mapping from input to output nodes can be expressed as $\hat{\mathbf{I}}$: $\mathbb{R}^3 \rightarrow \mathbb{R}$. The data is transmitted from the first layer to the last layer by the transformations. Hence, the relation between the input and output of the *k*th hidden layer is expressed as

$$\begin{aligned} \hat{\mathbf{I}}^h &= \mathbf{W}^{h^T}\hat{\mathbf{o}}^{h-1} + \mathbf{b}^h, \\ \hat{\mathbf{o}}^h &= f\left(\hat{\mathbf{I}}^h\right), \end{aligned} \tag{5}$$

where $f(.)$ is the activation function, which allows for learning a complex relationship between input and output. There are several common choices, including ReLU, LeakyReLU, Sigmoid, Softmax, and Tanh. $\hat{\mathbf{o}}^h$ denotes the output of the *h*th hidden layer. And, the output of each layer can be rewritten as follows

$$\begin{aligned} &\text{input layer} \quad : \quad \hat{\mathbf{o}}^0 = \left[x, y, z\right] \in \mathbb{R}^3, \\ &\text{hidden layers} : \quad \hat{\mathbf{o}}^h = f\left(\mathbf{W}^{h^T}\hat{\mathbf{o}}^{h-1} + \mathbf{b}^h\right) \in \mathbb{R}^{m_h}, \\ &\quad \text{for} \;\; 1 \le h \le (L-1), \\ &\text{output layer} \quad : \quad \hat{o}^L = f\left(\mathbf{W}^{L^T}\hat{\mathbf{o}}^{L-1} + \mathbf{b}^L\right) = \hat{A} \;\; \in \mathbb{R}, \end{aligned} \tag{6}$$

where m_h is the number of neurons in the *h*th hidden layer.

In order to achieve the optimal parameters, the training is carried out by minimizing the loss function $\mathcal{L}$. There are many available optimizers to train the DNN model. Adam optimizer is one of the most common ones, so it is utilized to perform the training task. Accordingly, the network parameters at timestep *t* are updated as follows

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon}, \tag{7}$$

where $\boldsymbol{\theta}$ is the parameter vector including weights and biases of the network; η is the learning rate; ε is a constant added to maintain numerical stability; $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ are the bias-corrected first and second raw moment, and they are given by

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}, \tag{8}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t}, \tag{9}$$

in which $\beta_1, \;\; \beta_2 \in [0, 1)$ are the hyper-parameters to control $\mathbf{m}_t$ and $\mathbf{v}_t$ which are two exponential decay rates at timestep *t* as follows

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + \left(1-\beta_1\right).\nabla\mathcal{L}\left(\boldsymbol{\theta}_{t-1}\right), \tag{10}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + \left(1-\beta_2\right).\nabla\mathcal{L}\left(\boldsymbol{\theta}_{t-1}\right), \tag{11}$$

Here, $\nabla\mathcal{L}\left(\boldsymbol{\theta}_{t-1}\right)$ is the first gradient of the loss function with respect to the parameters at timestep $(t-1)$. In this work, Adam optimizer with its default parameters as suggested by Kingma and Ba [66] was used to train the model. For more details, interested readers are suggested to refer to Ref. [66].

3.3 Loss function

It should be noted that the proposed approach is designed based on unsupervised learning. Consequently, the predicted cross-sectional area value $\hat{A}_i(\boldsymbol{\theta})$ is expressed by the parameters of the DNN. The loss function is defined as the penalty function in Eq. (2). It includes the mass and responses of structure obtained by FEA corresponding to the predicted cross-sectional areas $\hat{\mathbf{A}}(\boldsymbol{\theta})$, as expressed below

$$\mathcal{L}(\boldsymbol{\theta}) \;\; = \left(1+\varepsilon_1 c\right)^{\varepsilon_2} W\left(\hat{\mathbf{A}}(\boldsymbol{\theta})\right). \tag{12}$$

It is easily seen that instead of solving the optimization problem in Eq. (1), we now turn to minimize the loss function by training to find the optimal parameters $\boldsymbol{\theta}^*$ of the network.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left(\mathcal{L}(\boldsymbol{\theta}) \right). \tag{13}$$

Once the network is trained, the optimum cross-sectional areas of truss members are found corresponding to the optimal parameters. Note that the derivatives of the loss function with respect to the parameters θ_i obtained by applying the chain rule to Eq. (12) is given by

$$\frac{\partial\mathcal{L}}{\partial\theta_i} \;\; = \sum_{e=1}^{m} \frac{\partial\mathcal{L}}{\partial\hat{A}_e}\frac{\partial\hat{A}_e}{\partial\theta_i}. \tag{14}$$

From Eq. (14), it can be observed that the term $\frac{\partial\hat{A}_e}{\partial\theta_i}$ will be calculated automatically by the backpropagation algorithm, and the other term $\frac{\partial\mathcal{L}}{\partial\hat{A}_e}$ must be provided as follows

$$\frac{\partial\mathcal{L}}{\partial\hat{A}_e} = \varepsilon_2\left(1+\varepsilon_1 c\right)^{\varepsilon_2-1} W\frac{\partial c}{\partial\hat{A}_e} + \left(1+\varepsilon_1 c\right)^{\varepsilon_2}\frac{\partial W}{\partial\hat{A}_e}. \tag{15}$$

Here, the derivative of the sum of the violated constraints $\frac{\partial c}{\partial\hat{A}_e}$ is defined as

$$\frac{\partial c}{\partial\hat{A}_e} = \sum_{j=1}^{n}\frac{\partial}{\partial\hat{A}_e}\max\left(0, g_j\right), \tag{16}$$

with

$$\frac{\partial}{\partial\hat{A}_e}\max\left(0, g_j\right) = \begin{cases} 0 & g_j \le 0, \\ \frac{\partial g_j}{\partial\hat{A}_e} & g_j > 0, \end{cases} \tag{17}$$

in which g_j is the jth constraint function. In this work, three types of constraints are considered including displacement, stress, and frequency. As pointed out by Camarda [67] and Iranmanesh [42], the gradient of these constraints with respect to the area of the members is given by

$$\frac{\partial \mathbf{u}}{\partial A_e} = -\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial A_e}\mathbf{u}, \tag{18}$$

$$\frac{\partial \boldsymbol{\sigma}_e}{\partial \hat{A}_e} = \frac{E}{L_e}\mathbf{T}\frac{\partial \mathbf{u}_e}{\partial \hat{A}_e}, \tag{19}$$

$$\frac{\partial \omega_f}{\partial \hat{A}_e} = \boldsymbol{\phi}_f^T\frac{\partial \mathbf{K}}{\partial \hat{A}_e}\boldsymbol{\phi}_f - \omega_f^2\boldsymbol{\phi}_f^T\frac{\partial \mathbf{M}}{\partial \hat{A}_e}\boldsymbol{\phi}_f, \tag{20}$$

where $\mathbf{u}$ is the displacement vector; $\mathbf{K}$ is the global stiffness matrix; σ_e stands for the stress of the member; $\mathbf{T}$ is the geometric transformation matrix; $\mathbf{u}_e$ denotes the displacement vector of the element; ω_f and $\boldsymbol{\phi}_f$ are the angular frequency and the eigenvalue vector, respectively; M stands for the mass matrix. As can be seen, the method for computing the sensitivity of constraints can be directly determined from Eqs. (16)–(20), but it takes effort to achieve success. To circumvent this, the automatic differentiation tool JAX is utilized in this work for accelerated performance [59]. It has been successfully applied to the topology optimization problem [68]. In contrast, the gradient of the objective function is easily defined as follows

$$\frac{\partial W}{\partial \hat{A}_e} = \rho_e\sum_{k=1}^{ng} L_k, \tag{21}$$

where W is the structural weight; ng denotes the total number of member groups, and ρ_e stands for the density of members.

According to the afore-presented issue, the gradient of the loss function is easily estimated. Hence, the parameters of the network are tuned for each epoch of the training process. The main steps of the proposed approach are summarized in Algorithm 1. It is worth mentioning that this approach is based on the gradient descent algorithm and the optimization problem is a nonconvex one [2, 3]. Therefore, we need to pay attention to the sensitivity of the initial parameters as well as to the possibility of being stuck in local minima and saddle points.

Algorithm 1: Optimization of truss structures using deep unsupervised learning-based framework

Input:
- Structure: geometry, material properties, loads, boundary conditions
- Network: number of hidden layers, hidden neurons, activation function, Adam optimizer

Output: optimal parameters $\boldsymbol{\theta}$, optimum weight of truss structures

Build a DNN with uniformly distributed initial parameters $\boldsymbol{\theta}_0$ from [-1, 1]
Set the default values of Adam optimizer parameters [66]
while $\|\nabla\mathcal{L}(\boldsymbol{\theta})\| > 0.01$ or *Maxepoch* is not reached **do**
  $t = t + 1$
  Predict $\hat{\mathbf{A}}$ using the feedforward phase
  Compute the weight $W\left(\hat{\mathbf{A}}\right)$ of truss structure
  Evaluate the constraint function $g_j\left(\hat{\mathbf{A}}\right)$ by FEA
  Loss function $\mathcal{L}(\boldsymbol{\theta}_{t-1})$ is estimated by Eq. (12)
  $\frac{\partial W}{\partial \hat{A}_e}$ is calculated by Eq. (21)
  $\frac{\partial c}{\partial \hat{A}_e}$ is determined by the automatic differentiation JAX
  $\frac{\partial \hat{A}_e}{\partial \theta_i}$ is calculated automatically by the backpropagation algorithm
  Calculate the sensitivity of loss function $\frac{\partial \mathcal{L}}{\partial \theta_i}$ by Eq. (14)
  Update parameters $\boldsymbol{\theta}_t$ of the network by Eq. (7)

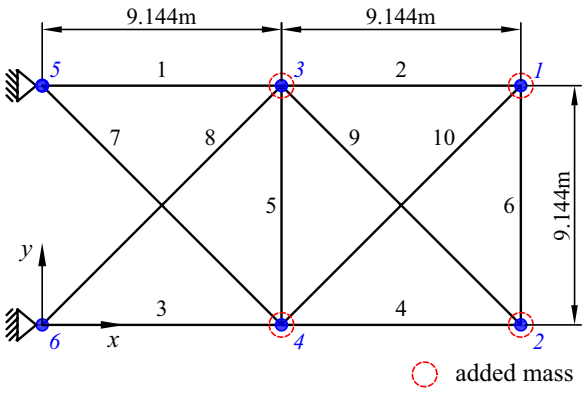

Fig. 3 10-bar planar truss problem

4 Numerical experiments

In this section, several well-known benchmark problems are explored to justify the efficiency of the proposed approach, and obtained results are compared with the DE and other metaheuristic algorithms done by previous works. Depending on the constraints' characteristics, the examples are divided into two categories: the first one is made up of four cases of natural frequency constraints, while the remaining one includes other three problems with displacement and stress constraints. In all investigated examples, two and three neurons are used in the input layer corresponding to 2D and 3D problems, and one neuron with regard to members' cross-sectional area is utilized for the output layer. Besides, the trial-error and grid search (GS) methods [69] are applied to estimate the architecture of the network. In addition, the Adam optimizer (learning rate 0.01, β_1=0.9, β_2=0.999) [66] is adopted for training the model. LeakyReLU and Softmax are activation functions for the hidden and output layer, respectively. Otherwise, the learning process of the network is terminated when either the norm of the gradient value is less than 0.01 or the maximum number of epoch reaches [70]. Note that for the present method, the number of epochs is also the number of FEAs or loss function evaluations.

For DE algorithm, the parameters are set as follows: population size 20, the maximum number of FEAs 3000, mutant factor $F = 0.8$, crossover control parameter $\mathrm{Cr} = 0.9$, and the value of threshold 10^{-6} [2, 10, 60]. The best design is determined by 30 independent runs of each problem, dealing with the stochastic nature of the DE algorithm. To get a fair comparison between the different methods, all the tests are implemented in the library of Tensorflow and using Python software. And all experiments are performed on a desktop computer with Core i5-8500 CPU of @3.0 GHz and 16 GB RAM.

4.1 Frequency constraints

4.1.1 10-bar planar truss

The first numerical example is a 10-bar planar truss structure, as illustrated in Fig. 3. The data for the design, including Young's modulus, density, design variable bounds, and natural frequency constraints, are summarized in Table 1.

Table 1 Material properties, constraints, upper and lower bounds on area variables

Test problems	Young's modulus E (N/m^2)	Material density ρ (kg/m^3)	Natural frequency (Hz)	Cross-sectional area limits (cm^2)
10-bar planar truss	6.98×10^{10}	2770	$7 \le f_1$; $15 \le f_2$; $20 \le f_3$	$0.645 \le A_i \le 50$
72-bar space truss	6.98×10^{10}	2770	$f_1 = 4$; $6 \le f_3$	$0.645 \le A_i \le 20$
120-bar dome truss	2.1×10^{11}	7971.81	$9 \le f_1$; $11 \le f_2$	$1 \le A_i \le 129.3$
200-bar planar truss	2.1×10^{11}	7860	$5 \le f_1$; $10 \le f_2$; $15 \le f_3$	$0.1 \le A_i \le 25$

Table 2 MSE of the optimal cross-sectional areas for the 10-bar planar truss with various optimizers and activation functions

Activation functions	Optimizers							
	Adam	Adamw	Adadelta	Adagrad	Adamax	SGD	ASGD	RMSprop
Softmax	0.0466	0.0610	129.7283	20.4489	6.2696	87.8649	204.8981	7.3892
Softplus	5.7412	5.4541	45.3525	6.0972	5.7074	6.2923	31.0242	22.5772
Tanh	0.0918	0.0529	30.8862	7.7082	0.1728	9.0307	19.7944	7.7507
Sigmoid	6.5302	6.4860	111.7375	6.8888	5.8379	22.6053	99.6836	14.6526
ReLU	0.1864	7.2302	33.8680	6.6109	6.6240	6.2492	19.6532	13.2780
LeakyReLU	0.0247	0.0332	33.3514	6.3805	7.4386	6.3581	20.6535	11.5345

Table 3 Standard errors (%) of weight and frequencies for the 10-bar planar truss with various activation functions using Adam

Error (%)	Activation functions					
	Softmax	Softplus	Tanh	Sigmoid	ReLU	LeakyReLU
Weight	0.3489	1.7766	0.3596	1.5482	0.2001	0.0487
f_1	0.0281	0.0028	0.0432	0.0398	0.0196	0.0014
f_2	2.6639	11.2745	3.1186	11.2354	0.8680	0.1062
f_3	0.1181	0.1950	0.0029	0.0995	0.0949	0.0380

Table 4 Optimization results obtained for the 10-bar planar truss with frequency constraints

Design variables	PSO	FA	DPSO	OC-GA	IDE	AHEFA	This study	
A_i (cm^2)	[11]	[9]	[12]	[14]	[2]	[10]	DE	DUL
A_1	37.712	36.198	35.944	37.284	35.060	35.171	35.288	35.106
A_2	9.959	14.030	15.530	9.445	14.685	14.720	14.577	14.826
A_3	40.265	34.754	35.285	35.051	35.068	35.107	35.087	35.227
A_4	16.788	14.900	15.385	19.262	14.809	14.698	14.782	14.723
A_5	11.576	0.654	0.648	2.783	0.645	0.645	0.645	0.654
A_6	3.955	4.672	4.583	5.450	4.558	4.559	4.561	4.568
A_7	25.308	23.467	23.610	19.041	23.527	23.733	23.510	23.712
A_8	21.613	25.508	23.599	27.939	23.799	23.679	23.972	23.712
A_9	11.576	12.707	13.135	14.950	12.503	12.398	12.462	12.371
A_{10}	11.186	12.351	12.357	10.361	12.459	12.423	12.266	12.371
Best weight (kg)	537.980	531.280	532.390	535.730	524.462	**524.451**	524.463	524.719
Weight error (%)	2.580	1.302	1.514	2.151	0.002	–	0.002	0.051
f_1 (Hz)	7.000	7.000	7.000	7.001	7.000	7.000	7.000	7.000
f_2 (Hz)	17.786	16.164	16.187	17.030	16.185	16.192	16.196	16.213
f_3 (Hz)	20.000	20.003	20.000	20.156	20.000	20.000	20.000	20.008
Number of FEAs	–	5000	3000	12,000	6260	5860	9480	1000
Total times (s)	–	–	–	–	–	–	72.134	29.784

Bold used to emphasize the best minimum weight design

A lumped mass of 454 kg is set at four free nodes. This benchmark problem has been previously analyzed by many researchers using metaheuristic algorithms, such as Gomes [11], Kaveh [12], Ho [2], and Lieu [10], etc. For the first case, a DNN with five hidden layers is chosen to build the model and each hidden layer contains 20 neurons. The model performs training with the maximum number of FEAs equal to 1,000.

With the above architecture, a survey is conducted to investigate the effect of different combinations of optimizers and activation functions on the performance of the network. Herein, mean square error (MSE) of the optimal solution between the DUL and DE is employed as the standard tool to measure. Table 2 summarizes the obtained MSE results. It can be seen that Adam and Adadelta are the best and worst optimizers, respectively. Therein, the combination of Adam and LeakyReLU provides the lowest MSE (0.0247). In addition, Table 3 shows the errors of the objective and constraints compared with the DE when Adam is combined with various activation functions. It is clear that LeakyReLU is the lowest error value and less than 0.2%. Consequently, Adam and LeakyReLU are respectively chosen as optimizer and activation function for the training network in this study.

Table 4 shows a comparison of optimal results gained by this work and other researchers. It can be observed that the optimum weight acquired by DUL (524.719 kg) is very close to DE (524.463 kg), AHEFA (524.451 kg), and IDE (524.462 kg). However, the DUL outperforms other well-known existing algorithms available in the literature. More specifically, it requires the least number of structural analyses with only 1000 analyses, while the metaheuristic algorithms require a larger number of FEA calls, e.g. AHEFA [10] with 5860 analyses, DE with 9480 analyses. This can easily be explained by the fact that the training algorithm is used as a gradient-based optimization of stochastic loss function, so the number of function evaluations will be drastically reduced. All frequency constraints are satisfied with the small relative errors. In addition, the weight error against to the AHEFA is small (0.051%), which is only higher than DE (0.002%) and IDE (0.002%).

The weight convergence histories of the DUL and DE are depicted in Fig. 4. As observed, the proposed DUL model's

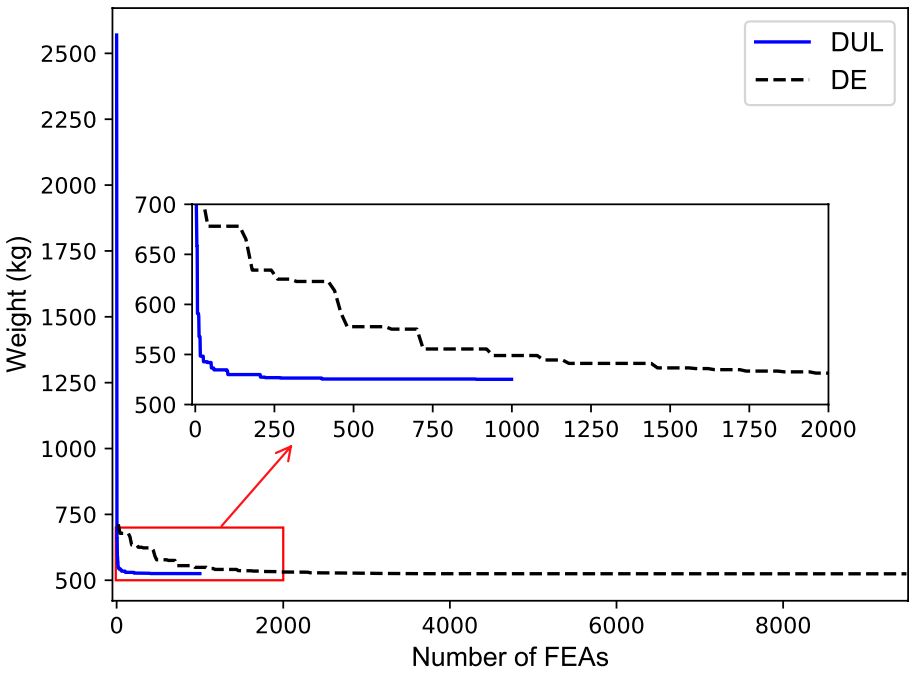

Fig. 4 The weight convergence histories obtained using the DUL and DE for the 10-bar planar truss

learning curve shows that the weight of structure rapidly decreases at the beginning, tends to be stable around 400 analyses, and reaches the optimal solution only through 1000 analyses. On the contrary, the traditional DE converges much slower and requires more than 9 times the number of analyses (9480 analyses) compared to our model. In addition, this work only requires one run time with 29.784 s, while DE approximately takes 72 s to complete 30 independent optimization runs to get as accurate results as possible. Therefore, our approach helps to save more than a half of the computational cost.

4.1.2 72-bar space truss

Next, a 72-bar space truss structure, as shown in Fig. 5, is investigated as the second example for size optimization. Members' cross-sectional areas are classified into 16 categories, which corresponds to the 16 design variables as listed in Table 5. A lumped mass of 2270 kg is mounted at each of the four upper nodes of the structure. The data concerning the design of this example is contained in Table 1. This structure has been formally investigated by various researchers [2, 9–11, 60]. A 6-layer network (3-20-20-20-20-1) is trained with maximum epochs of 3000.

A comparison of the optimal results attained by the proposed approach and previous works is illustrated in Table 5. It can be observed that the optimal weight (325.161 kg) found by the DUL is close to AHEFA (324.237 kg) with the error less than 0.3%, and smaller than the other studies (PSO [11]: 328.823 kg; FA [9]: 328.334 kg; and HALC-PSO [60]:

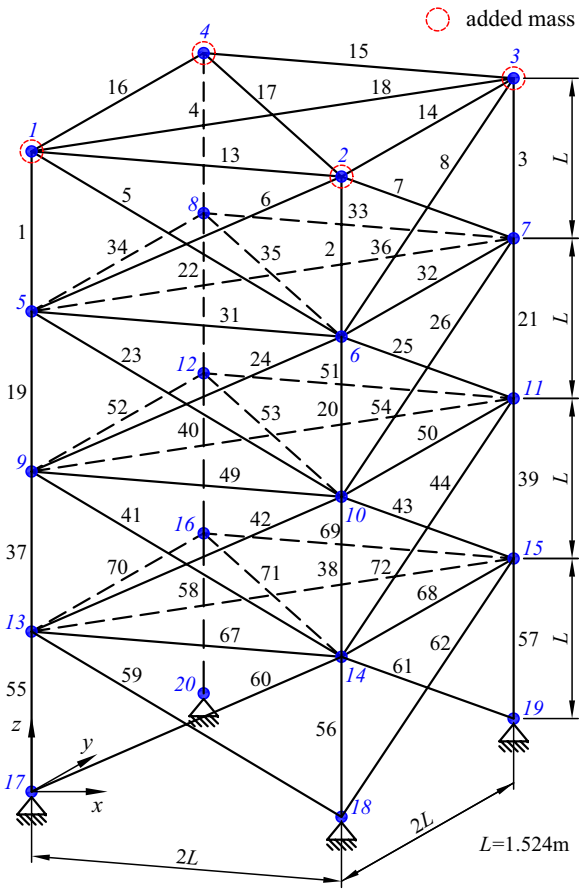

Fig. 5 Schematic of a 72-bar space truss structure

Table 5 Comparison of the obtained results for the 72-bar space truss with frequency constraints

Design variables	PSO	FA	HALC-PSO	IDE	AHEFA	This study	
A_i (cm^2)	[11]	[9]	[60]	[2]	[10]	DE	DUL
A_1–A_4	2.987	3.6803	3.3437	3.5863	3.5612	3.6191	3.8710
A_5–A_{12}	7.849	7.6808	7.8688	7.8278	7.8736	7.8203	7.8864
A_{13}–A_{16}	0.645	0.6450	0.6450	0.6450	0.6450	0.6453	0.6495
A_{17}–A_{18}	0.645	0.6450	0.6450	0.6450	0.6451	0.6452	0.6456
A_{19}–A_{22}	8.765	9.4955	8.1626	8.1052	7.9710	7.9086	8.4929
A_{23}–A_{30}	8.153	8.2870	7.9502	7.8788	7.8928	7.9904	8.0346
A_{31}–A_{34}	0.645	0.6450	0.6452	0.6451	0.6450	0.6451	0.6522
A_{35}–A_{36}	0.645	0.6461	0.6452	0.6450	0.6451	0.6466	0.6482
A_{37}–A_{40}	13.450	11.4510	12.2668	12.5157	12.5404	12.6016	12.9743
A_{41}–A_{48}	8.073	7.8990	8.1845	8.0102	7.9639	7.8204	8.0028
A_{49}–A_{52}	0.645	0.6473	0.6451	0.6450	0.6459	0.6451	0.6462
A_{53}–A_{54}	0.645	0.6450	0.6451	0.6452	0.6462	0.6455	0.6450
A_{55}–A_{58}	16.680	17.4060	17.9632	16.9997	17.1323	17.0752	15.9486
A_{59}–A_{66}	8.159	8.2736	8.1292	8.0362	8.0216	8.1264	7.9057
A_{67}–A_{70}	0.645	0.6450	0.6450	0.6451	0.6450	0.6463	0.6901
A_{71}–A_{72}	0.645	0.6450	0.6450	0.6453	0.6451	0.6461	0.6455
Best weight (kg)	328.823	328.334	327.770	324.244	**324.237**	324.289	325.161
Weight error (%)	1.414	1.264	1.090	0.002	–	0.016	0.285
f_1 (Hz)	4.00	4.00	4.00	4.00	4.00	4.00	4.00
f_3 (Hz)	6.00	6.00	6.00	6.00	6.00	6.00	6.01
Number of FEAs	–	50,000	8000	11,620	8860	11,740	3000
Total times (s)	–	–	–	–	–	236.763	148.489

Bold used to emphasize the best minimum weight design

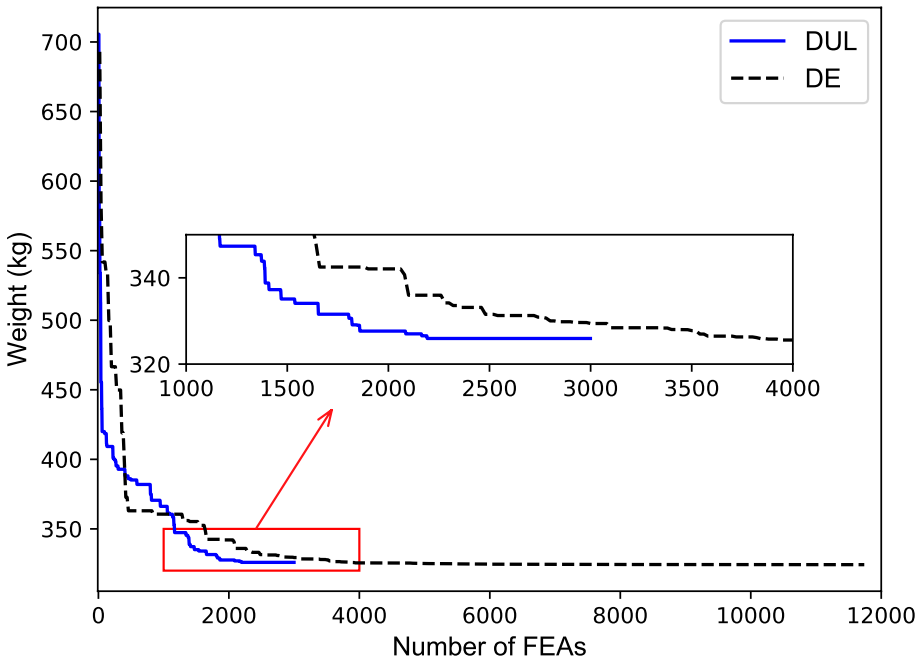

Fig. 6 The weight convergence histories obtained using the DUL and DE for the 72-bar space truss

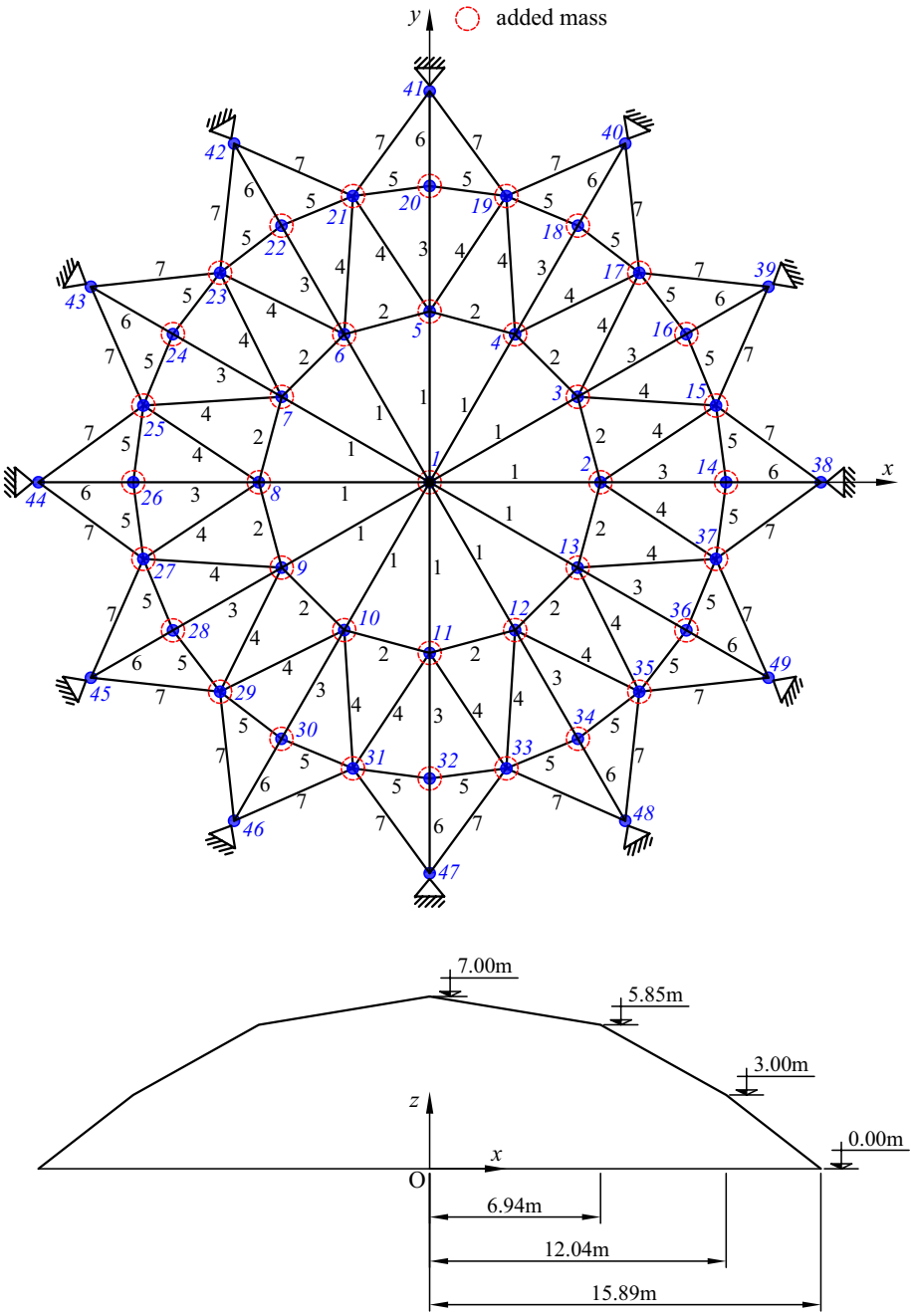

Fig. 7 A 120-bar dome truss structure

327.77 kg). Compared to the computational cost, the DUL required less structural analyses than other optimization techniques, namely 74% DE, 66% AHEFA [10], 74% IDE [2], 62% HALC-PSO [60], and 94% FA [9], but the frequencies still satisfy constraints. Furthermore, it takes 148.489 s for one run to achieve the near-global optimal solution, while DE needs 236.763 s with 30 runs to get the goal. Again, DUL shows its efficiency in significantly reducing the computational effort. Figure 6 shows the weight convergence histories obtained using two different approaches. Clearly, the convergence rate of the DUL is much faster than that of the DE algorithm. The weight obtained by DUL approaches the optimal solution after only 3,000 analyses, while the DE is still far behind (11,740 analyses).

4.1.3 120-bar dome truss

A dome truss with 120 members illustrated in Fig. 7 is examined. Here, $m_1 = 3000$ kg, $m_2 = 500$ kg, and $m_3 = 100$ kg are the lump masses which are attached to node 1, nodes 2–13, and the remaining nodes, respectively. Design variable bounds, material properties, and constraints for this example

Table 6 Optimization results obtained for the 120-bar dome truss with frequency constraints

Design variables	DPSO	OMGSA	HALC-PSO	IDE	AHEFA	This study	
A_i (cm^2)	[12]	[71]	[60]	[2]	[10]	DE	DUL
A_1	19.607	20.263	19.8905	19.4670	19.5094	19.4359	19.5002
A_2	41.290	39.294	40.4045	40.5004	40.3867	40.5901	40.4118
A_3	11.136	9.989	11.2057	10.6136	10.6033	10.5945	10.6020
A_4	21.025	20.563	21.3768	21.1073	21.1168	21.0916	21.1160
A_5	10.060	9.603	9.8669	9.8417	9.8221	9.8641	9.8428
A_6	12.758	11.738	12.7200	11.7735	11.7735	11.8052	11.7916
A_7	15.414	15.877	15.2236	14.8264	14.8405	14.8191	14.8365
Best weight (kg)	8890.480	8724.970	8889.960	8707.289	**8707.255**	8707.404	8708.975
Weight error (%)	2.104	0.203	2.098	0.000	–	0.002	0.020
f_1 (Hz)	9.0001	9.0020	9.0000	9.0000	9.0000	9.0000	9.0008
f_2 (Hz)	11.0007	11.0030	11.0000	11.0000	11.0000	10.9999	11.0004
Number of FEAs	6000	242,700	17,000	4060	3560	5740	2000
Total times (s)	–	–	–	–	–	1366.651	215.346

Bold used to emphasize the best minimum weight design

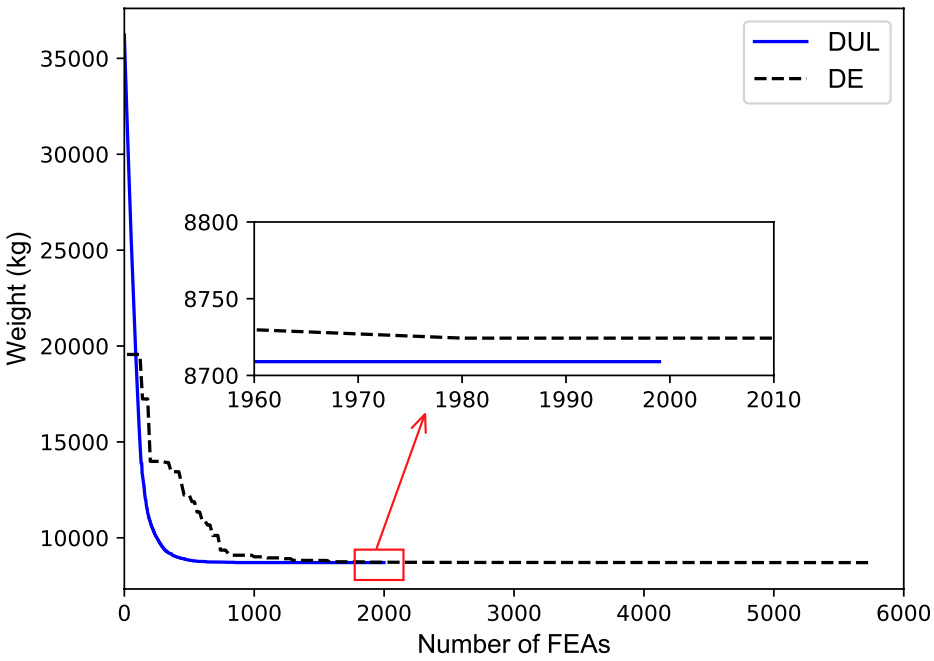

Fig. 8 The weight convergence histories obtained using the DUL and DE for the 120-bar dome truss

are tabulated in Table 1. The DNN model contains 2 hidden layers, 20 neurons in each hidden layer and 2000 epochs is the finest in its performance for this particular application.

Table 6 provides a comparison between this study and other studies available in the literature. It is easily seen that the optimum weight obtained by the DUL (8708.975 kg) is smaller than DPSO (8890.48 kg), OMGSA (8724.97 kg), and HALC-PSO (8889.96 kg). Although the obtained results by the DE (8707.404 kg), IDE [2] (8707.289 kg), and AHEFA [10] (8707.255 kg) are slightly lighter than the present method, the errors between them and AHEFA are less than 0.03%. However, it is clear that DUL outperforms the existing algorithms in terms of computational cost. Our model rapidly finds the optimum weight with only 2000 FEAs, whilst the others demand a larger number of FEAs for the convergence process, i.e. OMGSA with 242,700 analyses, IDE with 4060 analyses. In addition, the computation time of the proposed method (215.346 s) is 6 times less than that of the DE (1366.651 s). Figure 8 depicts the weight convergence histories obtained for this structure using the DUL and DE. Again, the DUL shows its efficiency in significantly improving the convergence rate compared with DE but still ensuring the quality of solution.

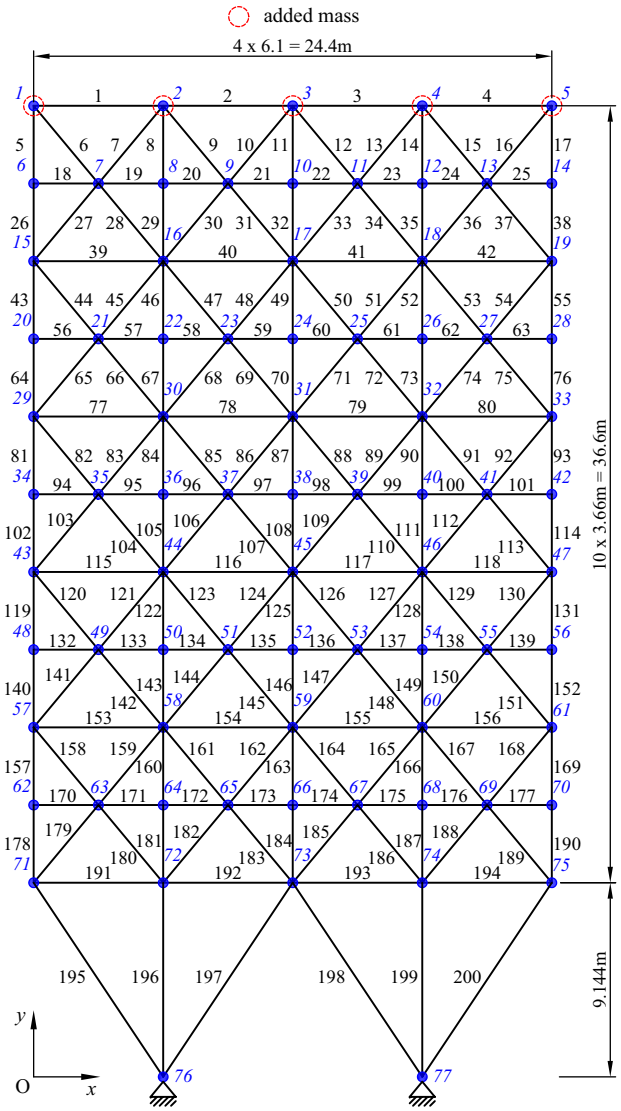

Fig. 9 A 200-bars planar truss structure

4.1.4 200-bar planar truss

The next example deals with the design of a 200-bars planar truss structure as shown in Fig. 9. All members are categorized into 29 groups using symmetry. The grouping information is indicated in Table 7. A lumped mass of 100 kg is added to 5 upper nodes. The data of the optimization is given in Table 1. The DNN uses 5 hidden layers with 20 neurons per layer and a maximum epoch of 2500 for training.

The optimal results, including the design variables, weight as well as frequencies, are tabulated as Table 8. As expected, the obtained results of the proposed model (2161.9 kg) reveals a quite good agreement with the DE (2160.94 kg) and AHEFA [10] (2160.74 kg). Note that although the optimum weight provided by the HALC-PSO [60] (2156.73 kg) and OMGSA [71] (2158.64 kg) are smaller than those by the DUL, the number of structural analyses is larger than those of the proposed approach. More concretely, the DUL only requires 2500 analyses for the convergence performance, while the DE, AHEFA, OMGSA [71], PFJA [72], and HALC-PSO [60] take 20,000, 11,300, 7252, 10,546, and 13,000, respectively. Additionally, DUL (513.965 s) only spends one-tenth of the computation time of DE (6375.934) to obtain the near-global optimal solution with the error less than 0.3%. Along with the results of the above mentioned examples, the proposed approach shows a significant saving of computational efforts when the complexity of problems is increased. The convergence curves in Fig. 10 gives a more detailed performance view for the present work. Clearly, this procedure converges very quickly to the near-optimal solution, while the DE is still quite far from the target value.

4.2 Stress and displacement constraints

4.2.1 25-bar space truss

A 25-bars space truss shown in Fig. 11 is investigated for the design optimization with displacement and stress constraints. The material density and Young's modulus are 0.1 lb/in^3 and 10^4 ksi. It is subjected to concentrated loads P_y = 20 kips and P_z = 5 kips as shown in Fig. 11. Cross-sectional areas of all members are divided into 8 groups corresponding to 8 design variables with the minimum gauge of 0.01 in^2 and the allowable stress of groups are listed in Table. 9. All displacements of nodes are limited in interval $[-0.35, 0.35]$ in. In addition, the network architecture including 3 hidden layers, 20 neurons in each hidden layer, and 1500 epochs is used to build the network.

As the previously presented examples, the optimal results gained by this study in comparison with other studies are summarized in Table 10. It is easily seen that the optimum weight found by the DUL (545.7 kg) agrees well with the DE (545.16 kg), TLBO [75] (545.09 kg), HPSO [74] (545.19 kg), and BB-BC [73] (545.38 kg) without violating constraints, while the lighter designs obtained by the HS [16] (544.38 kg) and HBB-BC [3] (545.16 kg) violated the design constraints. Furthermore, the DUL only requires 1500 structural analyses with the small error 0.113%, and much less than HS [16] (15,000), BB-BC [73] (20,566), HPSO [74] (125,000), HBB-BC [3] (12,500), TLBO [75] (15,318) and DE (8480). In addition, the total times of DUL (49.470 s) is less than a half of DE (104.069 s). Obviously, the current approach requires the lower computational cost compared to the conventional algorithms. Figure 12 illustrates the weight convergence histories using the DUL and DE. One again shows the feature of the algorithm that the convergence speed accelerates in the early stage and quickly stabilizes after only 1200 analyses.

Table 7 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		

Table 8 Comparison of the obtained results for the 200-bar planar truss with frequency constraints

Design variables	HALC-PSO	PFJA	OMGSA	AHEFA	This study	
A_i (cm^2)	[60]	[72]	[71]	[10]	DE	DUL
1	0.3072	0.30785	0.289	0.2993	0.3096	0.3040
2	0.4545	0.47168	0.486	0.4508	0.4485	0.4588
3	0.1000	0.10020	0.100	0.1001	0.1001	0.1016
4	0.1000	0.10010	0.100	0.1000	0.1000	0.1106
5	0.5080	0.54175	0.499	0.5123	0.5114	0.5317
6	0.8276	0.81840	0.804	0.8205	0.8188	0.8247
7	0.1023	0.10096	0.103	0.1011	0.1007	0.1026
8	1.4357	1.42367	1.377	1.4156	1.4241	1.4248
9	0.1007	0.10006	0.100	0.1000	0.1001	0.1026
10	1.5528	1.63199	1.554	1.5742	1.5812	1.5895
11	1.1529	1.13730	1.151	1.1597	1.1662	1.1591
12	0.1522	0.10000	0.131	0.1338	0.1404	0.1472
13	2.9564	2.97378	3.028	2.9672	2.9833	2.9952
14	0.1003	0.10072	0.101	0.1000	0.1005	0.1123
15	3.2242	3.32736	3.261	3.2722	3.2274	3.2739
16	1.5839	1.55580	1.612	1.5762	1.5850	1.5866
17	0.2818	0.23602	0.209	0.2562	0.3016	0.2141
18	5.0696	5.20167	5.020	5.0956	5.0229	5.0919
19	0.1033	0.10000	0.133	0.1001	0.1001	0.1155
20	5.4657	5.47313	5.453	5.4546	5.4589	5.4540
21	2.0975	2.09090	2.113	2.0933	2.0996	2.0911
22	0.6598	0.66730	0.723	0.6737	0.7185	0.6511
23	7.6585	7.53409	7.724	7.6498	7.6974	7.4208
24	0.1444	0.10489	0.182	0.1178	0.1127	0.1880
25	8.0520	7.87075	7.971	8.0682	8.1285	7.7226
26	2.7889	2.81229	2.996	2.8025	2.8448	2.7871
27	10.4770	10.70210	10.206	10.5040	10.3676	10.5569
28	21.3257	21.75078	20.699	21.2935	21.1726	21.5607
29	10.5111	10.44841	11.555	10.7410	10.8075	10.5618
Best weight (kg)	**2156.73**	2171.34	2158.64	2160.74	2160.94	2161.90
Weight error (%)	-	0.6778	0.0886	0.1861	0.1953	0.2400
f_1 (Hz)	5.0000	5.0000	5.0000	5.0000	5.0000	5.0000
f_2 (Hz)	12.2540	12.3074	12.1180	12.1821	12.2668	12.2679
f_3 (Hz)	15.0440	15.0066	15.0290	15.0160	15.0776	15.1197
Number of FEAs	13,000	10,546	7252	11,300	20,000	2500
Total times (s)	–	–	–	–	6375.934	513.965

Bold used to emphasize the best minimum weight design

4.2.2 72-bar space truss

The next example deals with the optimal design of the 72-bars space truss schematized in Fig. 13. The system is subjected to an external loading of 5 kips in the x- and y-axes' positive directions at node 17. Density and Young's modulus of the material are set as 0.1 lb/in^3 and 10^4 ksi. The cross-sectional areas of members as design variables are categorized into 8 groups. All free node displacements are restricted to ±0.25. The stress limitations of all members are ±25 ksi. The minimum design variables are specified as 0.1 in^2. The DNN used for this application consists of 5 hidden layers with 20 hidden neurons in each layer and 1000 epochs.

Table 11 summarizes the results obtained by the DUL and other algorithms reported in recent literatures [3, 16, 73–75]. It is noticed that in this example, the DUL achieves the lightest design overall since the lighter design gained by the HPSO violates the design constraints. Furthermore, the number of structural analyses of the present method is dramatically decreased when compared with previous

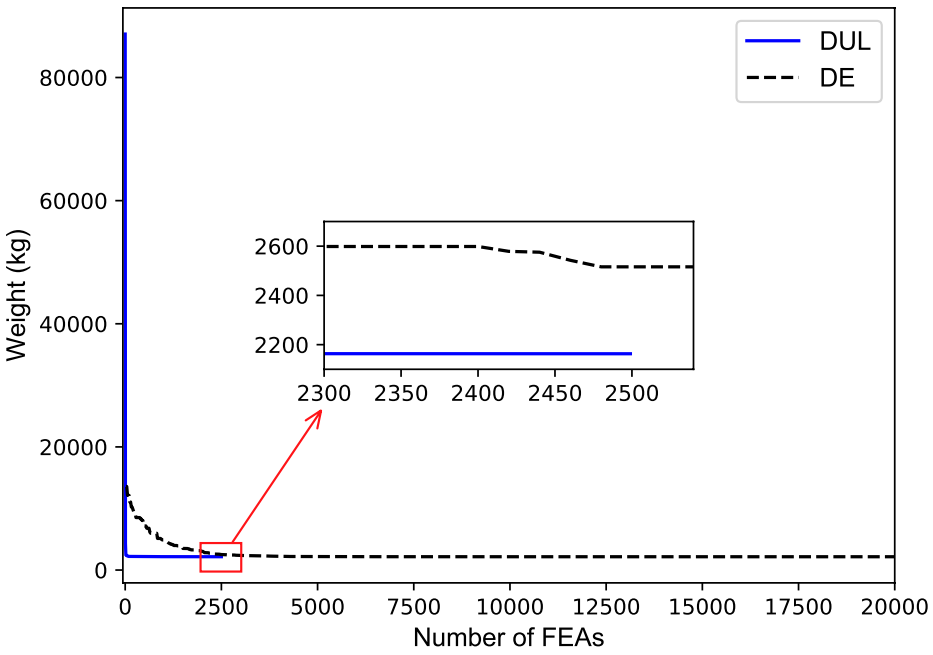

Fig. 10 The weight convergence histories obtained using the DUL and DE for the 200-bar planar truss

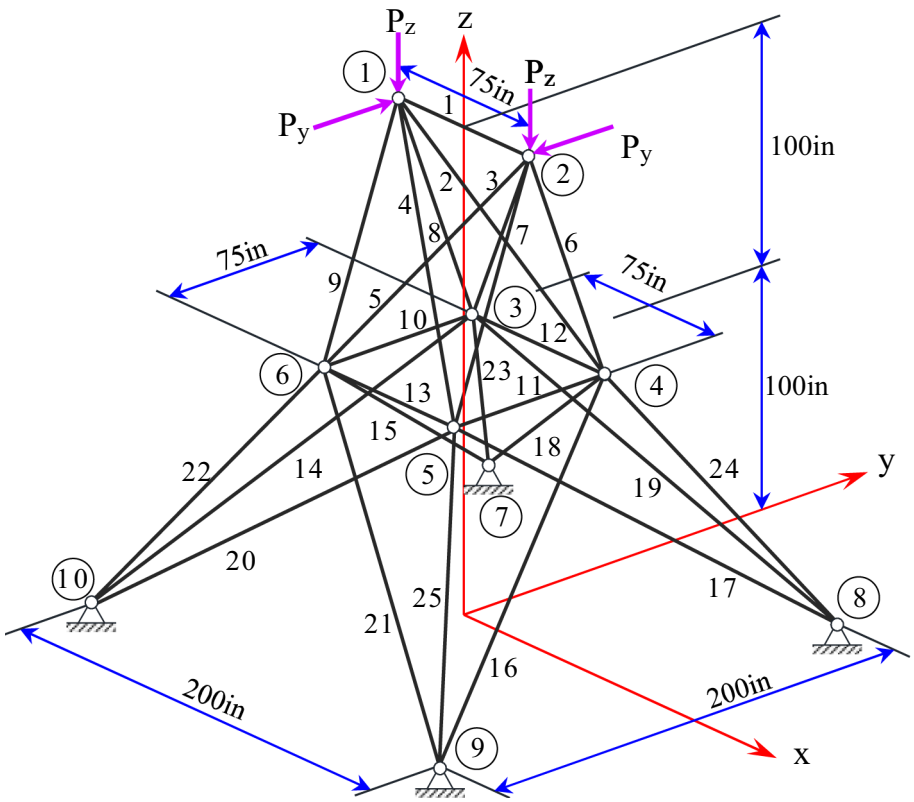

Fig. 11 A 25-bars space truss structure

Table 9 Allowable stress values for 25-bar space truss

Design variables A_i (in^2)	Allowable compressive stress (ksi)	Allowable tension stress (ksi)
A_1	35.092	40.0
A_2-A_5	11.59	40.0
A_6-A_9	17.305	40.0
A_{10}-A_{11}	35.092	40.0
A_{12}-A_{13}	35.092	40.0
A_{14}-A_{17}	6.759	40.0
A_{18}-A_{21}	6.959	40.0
A_{22}-A_{25}	11.082	40.0

studies. More specifically, it takes only 1000 analyses for the convergence performance, while the DE, TLBO [75], HBB-BC [3], HPSO [74], BB-BC [73], and HS [16] take 8580, 19,709, 13,200, 125,000, 19,621, and 20,000, respectively. Moreover, the computational time of DUL (63.715 s) is still much smaller than that of the DE (488.135 s). Finally, Fig. 14 represents the weight convergence histories obtained using two different approaches. As can be seen on the plot, the convergence rate of the learning process is much faster than the traditional DE. Therefore, the present approach has proven again to be the most efficient optimizer.

4.2.3 200-bar planar truss

The last structural optimization problem done herein is to optimize a 200-bar planar truss. The geometry and finite element representation are shown in Fig. 15. As shown in Table. 7, all cross-section areas of the structure are classified into 29 groups corresponding to design variables. The Young's modulus is 30,000 ksi and the density is 0.283 lb/in^3 for all elements. The allowable stress for all members is set to 10 ksi both in tension and compression. The minimum design variables are 0.1 in^2. In this example, the displacement constraints are not considered. The structure is designed for three independent loading conditions as following: (1) 1 kip applying in the positive direction of the *x*-axis at nodes 1, 6, 15, 20, 29, 34, 43, 48,57, 62 and 71; (2) 10 kip acting in the negative direction of the y-axis at 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) including conditions (1) and (2) acting together. In this case, the network configuration with 5 hidden layers, 20 neurons per layer, and 4,000 epochs are utilized to perform the optimization process.

Table 12 provides a comparison of optimal solutions found by the DUL and several previous studies. It is easily seen that the DUL performs better than the HS [16], HBB-BC [3], CMLPSA [76], and DE in terms of the optimum weight, violated constraints, and needed analyses. The optimum weight obtained by the proposed method is close to the result of the EHS [19], but it requires a smaller number of structural analyses.

It can be observed that the MCOA [15] and TLBO [75] are the first and second-best among the eight algorithms. Note, however, that the selection of control parameter values for the metaheuristic algorithms plays an important role in performance, robustness, and efficiency of them [75, 77–82]. Although, in this case, it was not considered in

Table 10 Optimization results obtained for the 25-bar space truss with displacement and stress constraints

Design variables	HS	BB-BC	HPSO	HBB-BC	TLBO	This study	
A_i (in^2)	[16]	[73]	[74]	[3]	[75]	DE	DUL
A_1	0.047	0.010	0.010	2.662	0.0100	0.0100	0.01306
A_2-A_5	2.022	2.092	1.970	1.993	2.0712	1.9834	1.95147
A_6-A_9	2.950	2.964	3.016	3.056	2.9570	2.9984	2.96618
A_{10}-A_{11}	0.010	0.010	0.010	0.010	0.0100	0.0100	0.01248
A_{12}-A_{13}	0.014	0.010	0.010	0.010	0.0100	0.0100	0.01282
A_{14}-A_{17}	0.688	0.689	0.694	0.665	0.6891	0.6864	0.69678
A_{18}-A_{21}	1.657	1.601	1.681	1.642	1.6209	1.6776	1.72595
A_{22}-A_{25}	2.663	2.686	2.643	2.679	2.6768	2.6576	2.64302
Best weight (lb)	544.38	545.38	545.19	545.16	**545.09**	545.16	545.70
Weight error (%)	0.130	0.053	0.018	0.013	–	0.014	0.113
Constraint tolerance (%)	0.206	None	None	2.06	None	None	None
Number of FEAs	15,000	20,566	125,000	12,500	15,318	8480	1500
Total times (s)	–	–	–	–	–	104.069	49.470

Bold used to emphasize the best minimum weight design

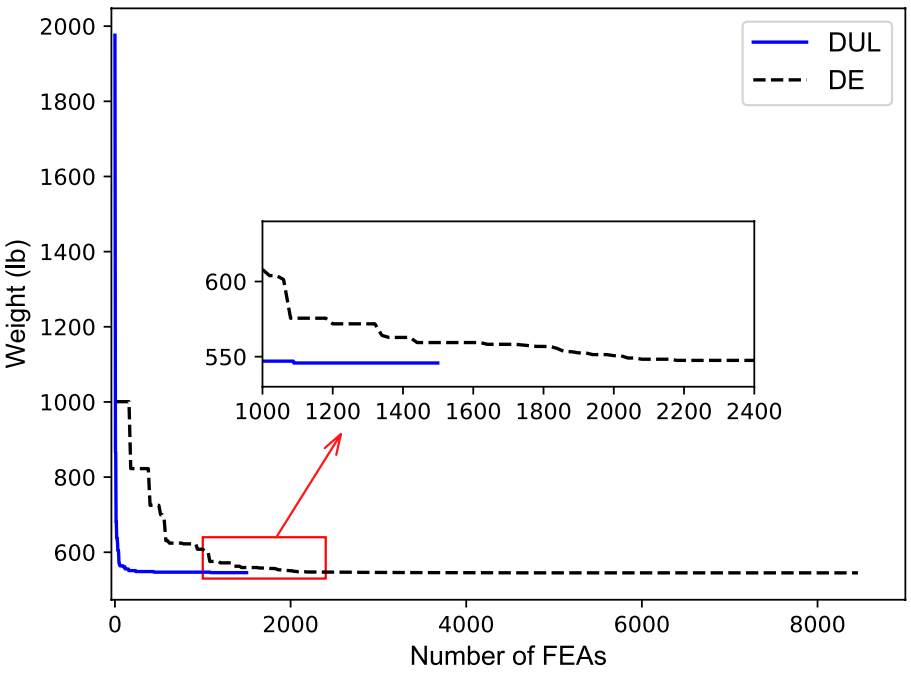

Fig. 12 The weight convergence histories obtained using the DUL and DE for the 25-bar planar truss

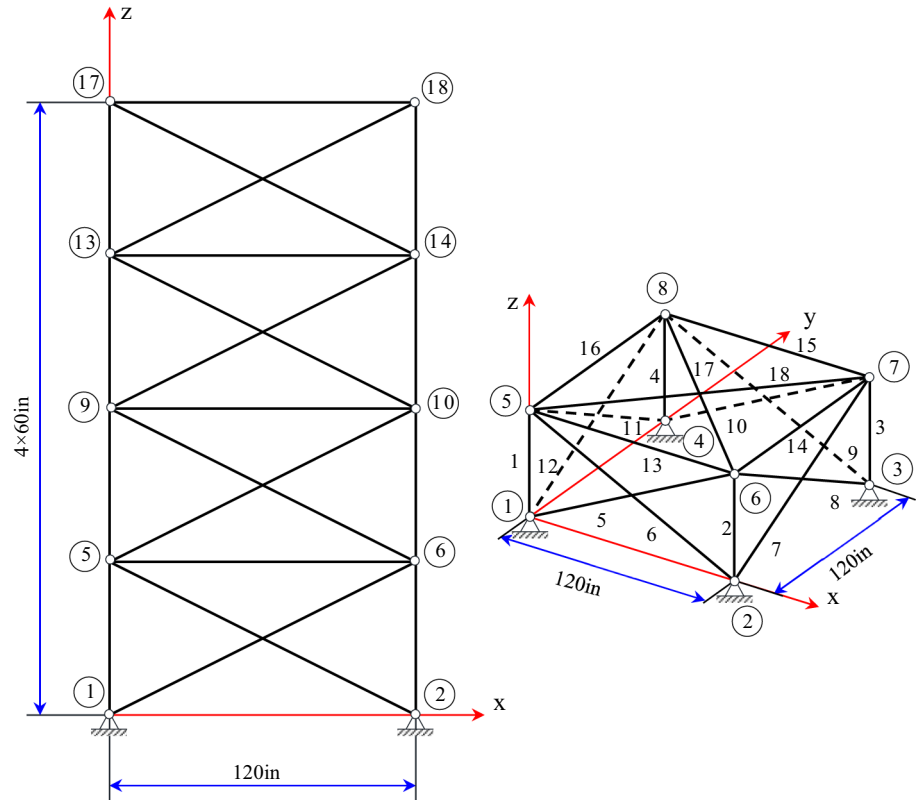

Fig. 13 72-bar space truss structure

Table 11 Optimization results obtained for the 72-bar space truss with displacement and stress constraints

Design variables	HS	BB-BC	HPSO	HBB-BC	TLBO	This study	
A_i (in^2)	[16]	[73]	[74]	[3]	[75]	DE	DUL
A_1-A_4	1.790	1.8577	1.857	1.9042	1.9064	1.8356	1.8606
A_5-A_{12}	0.521	0.5059	0.505	0.5162	0.5061	0.5359	0.5008
A_{13}-A_{16}	0.100	0.1000	0.100	0.1000	0.1000	0.1002	0.1015
A_{17}-A_{18}	0.100	0.1000	0.100	0.1000	0.1000	0.1001	0.1011
A_{19}-A_{22}	1.229	1.2476	1.255	1.2582	1.2617	1.2991	1.2635
A_{23}-A_{30}	0.522	0.5269	0.503	0.5035	0.5111	0.4959	0.5060
A_{31}-A_{34}	0.100	0.1000	0.100	0.1000	0.1000	0.1001	0.1013
A_{35}-A_{36}	0.100	0.1012	0.100	0.1000	0.1000	0.1007	0.1009
A_{37}-A_{40}	0.517	0.5209	0.496	0.5178	0.5317	0.4759	0.4971
A_{41}-A_{48}	0.504	0.5172	0.506	0.5214	0.5159	0.5140	0.5077
A_{49}-A_{52}	0.100	0.1004	0.100	0.1000	0.1000	0.1000	0.1017
A_{53}-A_{54}	0.101	0.1005	0.100	0.1007	0.1000	0.1043	0.1032
A_{55}-A_{58}	0.156	0.1565	0.100	0.1566	0.1562	0.1002	0.1002
A_{59}-A_{66}	0.547	0.5507	0.524	0.5421	0.5493	0.4932	0.5185
A_{67}-A_{70}	0.442	0.3922	0.400	0.4132	0.4097	0.3840	0.4013
A_{71}-A_{72}	0.590	0.5922	0.534	0.5756	0.5698	0.5658	0.5374
Best weight (lb)	379.27	379.85	369.65	379.66	379.63	370.30	**370.04**
Weight error (%)	2.421	2.578	0.176	2.527	2.519	0.070	–
Constraint tolerance (%)	0.218	None	39.075	None	None	None	None
Number of FEAs	20,000	19,621	125,000	13,200	19,709	8580	1000
Total times (s)	–	–	–	–	–	488.135	63.715

Bold used to emphasize the best minimum weight design

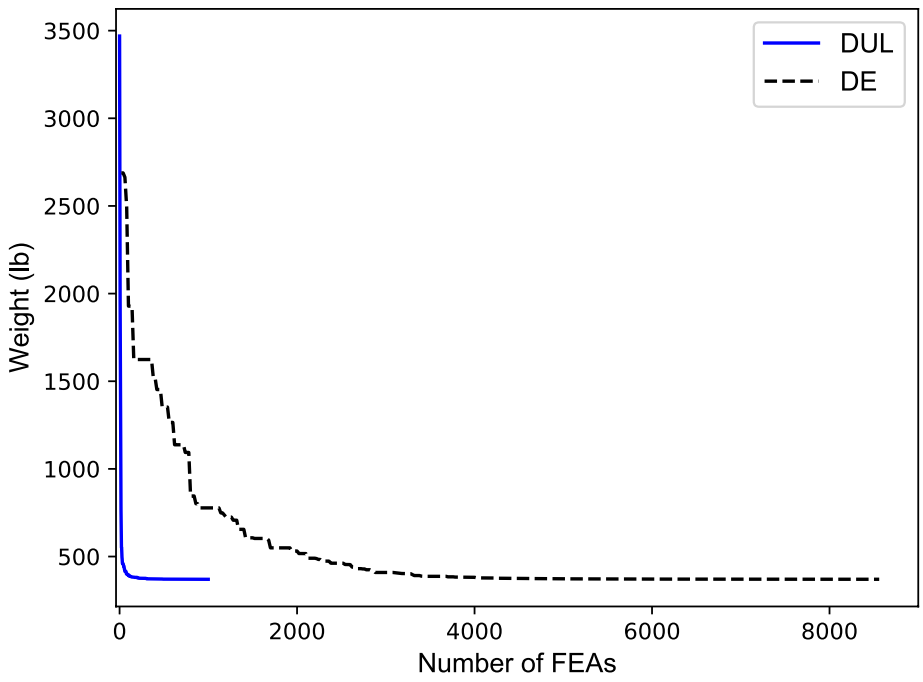

Fig. 14 The weight convergence histories obtained using the DUL and DE for the 72-bars space truss

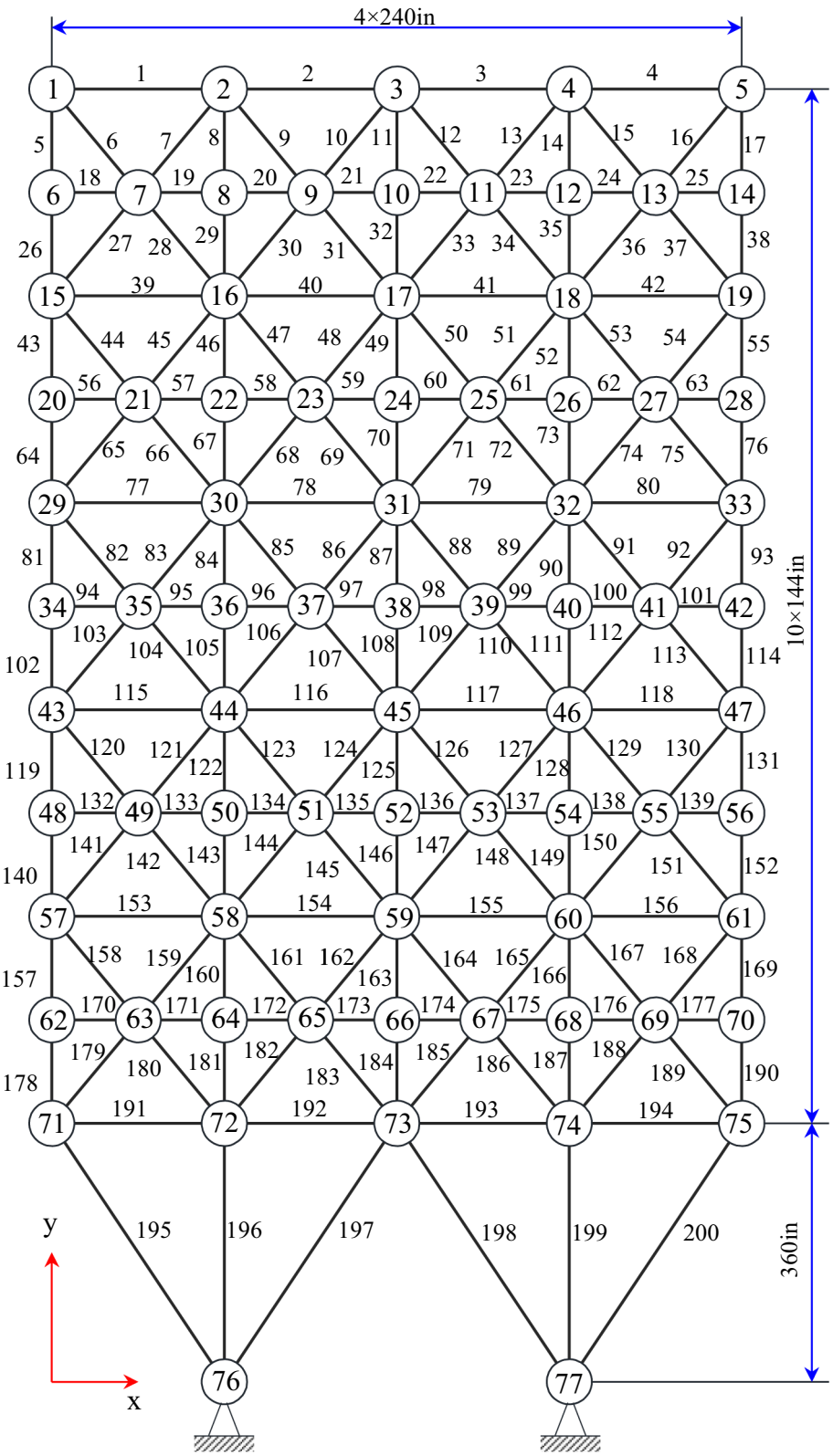

Fig. 15 A 200-bar planar truss structure

Table 12 Optimization results obtained for the 200-bar planar truss with stress constraints

Design variables	HS	HBB-BC	CMLPSA	EHS	TLBO	MCOA	This study	
A_i (in^2)	[16]	[3]	[76]	[19]	[75]	[15]	DE	DUL
1	0.1253	0.1033	0.1468	0.150	0.146	0.1390	0.1206	0.1183
2	1.0157	0.9184	0.9400	0.946	0.941	0.9355	0.9345	0.9907
3	0.1069	0.1202	0.1000	0.101	0.100	0.1000	0.1168	0.1142
4	0.1096	0.1009	0.1000	0.100	0.101	0.1000	0.1000	0.2323
5	1.9369	1.8664	1.9400	1.945	1.941	1.9355	1.9292	1.9579
6	0.2686	0.2826	0.2962	0.296	0.296	0.2909	0.2870	0.2906
7	0.1042	0.1000	0.1000	0.100	0.100	0.1000	0.1102	0.1624
8	2.9731	2.9683	3.1042	3.161	3.121	3.0816	3.0780	3.1520
9	0.1309	0.1000	0.1000	0.102	0.100	0.1000	0.2074	0.1380
10	4.1831	3.9456	4.1042	4.199	4.173	4.0816	4.0783	4.1834
11	0.3967	0.3742	0.4034	0.401	0.401	0.3967	0.4329	0.3843
12	0.4416	0.4501	0.1912	0.181	0.181	0.2959	0.1546	0.2115
13	5.1873	4.9603	5.4284	5.431	5.423	5.3854	5.3500	5.4466
14	0.1912	1.0738	0.1000	0.100	0.100	0.1000	0.1027	0.1272
15	6.2410	5.9785	6.4284	6.428	6.422	6.3853	6.3502	6.4490
16	0.6994	0.7863	0.5734	0.571	0.571	0.6332	0.5636	0.5317
17	0.1158	0.7374	0.1327	0.156	0.156	0.1842	0.5160	0.2149
18	7.7643	7.3809	7.9717	7.961	7.958	8.0396	7.9508	8.0113
19	0.1000	0.6674	0.1000	0.100	0.100	0.1000	0.1017	0.1434
20	8.8279	8.3000	8.9717	8.959	8.958	9.0395	8.9503	8.9967
21	0.6986	1.1967	0.7049	0.722	0.720	0.7460	0.8932	0.7054
22	1.5563	1.0000	0.4196	0.491	0.478	0.1306	0.1525	0.2450
23	10.9806	10.8262	10.8636	10.909	10.897	10.9114	11.0423	10.8275
24	0.1317	0.1000	0.1000	0.101	0.100	0.1000	0.1000	0.1265
25	12.1492	11.6976	11.8606	11.985	11.897	11.9114	12.0423	11.8557
26	1.6373	1.3880	1.0339	1.084	1.080	0.8627	0.9196	0.8580
27	5.0032	4.9523	6.6818	6.464	6.462	6.9169	6.7136	6.8944
28	9.3545	8.8000	10.8113	10.802	10.799	10.9674	10.7305	11.1690
29	15.0919	14.6645	13.8404	13.936	13.922	13.6742	13.8833	13.6032
Best weight (lb)	25447.10	25156.50	25445.63	25542.50	25488.15	**25,450.18**	25564.99	25547.90
Weight error (%)	0.012	1.154	0.018	0.363	0.149	–	0.451	0.384
Constraint tolerance (%)	3.69	9.97	0.071	None	None	None	None	None
Number of FEAs	48,000	9875	9650	22,851	28,059	27,720	47,100	4000
Total times (s)	–	–	–	–	–	–	3553.642	1305.273

Bold used to emphasize the best minimum weight design

studies [15, 75]. However, this is demonstrated in this study by the DE algorithm. Despite the same parameters, there is a good agreement between the results obtained by the DE and other studies for the first six examples, but it is more difficult to resolve the last example. Herein, the optimum weight obtained by the DE (25564.99 lb) is much heavier than the MCOA (25,450.18 lb).

From Table 12, it is easily be seen that the optimum weight gained by the DUL is larger than the MCOA [15] and TLBO [75] with the error less than 0.4%. One explanation for these differences is that the backpropagation algorithm based on the gradient descent method deals with the initial value of the weights, biases, and learning rate [83]. Furthermore, the total computational cost of DUL (1305.273 s) reduces by more than a half of DE (3553.642 s). A comparison of the convergence rate between the DUL and DE is shown in Fig. 16. Obviously, the DUL always converges much more quickly than the DE. Again, this shows the efficiency of our method.

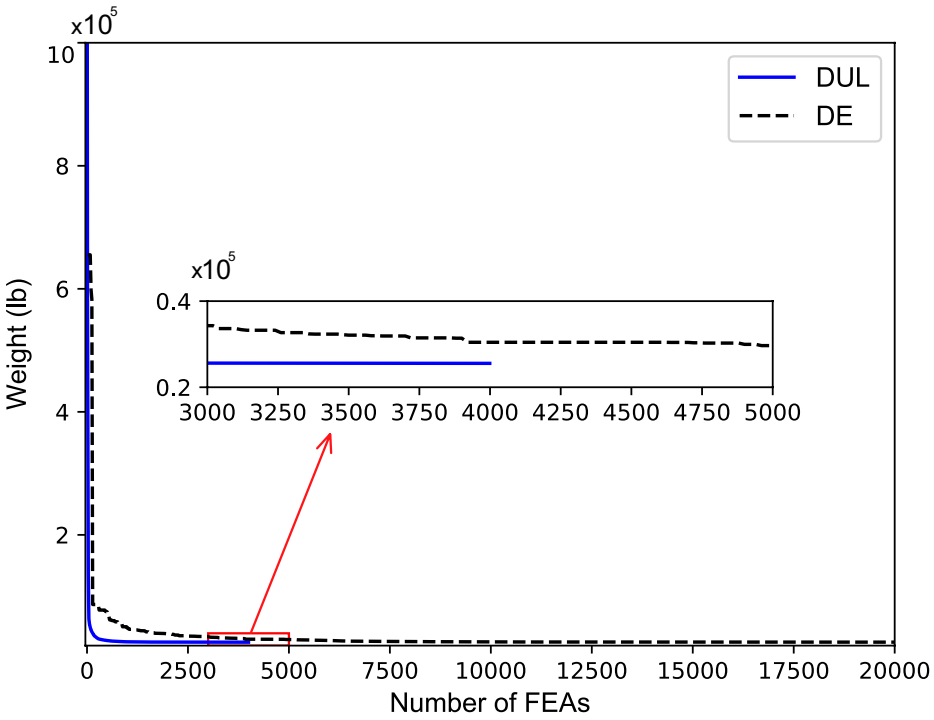

Fig. 16 The weight convergence histories obtained using the DUL and DE for the 200-bars planar truss

5 Conclusions

In this article, an efficient DUL-based methodology is first introduced to directly perform optimization of truss structures under multiple constraints. The members' cross-sectional areas are parameterized by the parameters of the DNN. According to that core idea, the weights and biases are considered as the design variables of the structural optimization problem, instead of the cross-sectional areas. The loss function is constructed relying on the output values of the DNN and the structural responses using FEA. The optimum weight of the structure is found as soon as the training phase ends without any other algorithms. The efficiency of the proposed approach is demonstrated through several numerical examples for size optimization of truss structures. The obtained results have indicated that the optimum weight obtained by this work is a good agreement with six of the seven tests. The DUL dramatically saves computational cost in almost all problems in comparison with other algorithms. In addition, its convergence speed is very fast at the beginning of iterations. This approach promises to extend its applications to more complex optimization problems such as discrete variables, multi-objective, and so on.

However, the implementation of this study may encounter some challenges which have still left unsolved. Firstly, the optimal architecture and hyperparameters of the DNN need to be indicated for each research area. In this study, GS and trial-error-tuning approaches were used to choose the best-fitted model, but it depends on prior knowledge and expert experience too much. To circumvent this limitation, one of the promising directions is recommended as genetic algorithms, Bayesian optimization techniques, etc. Secondly, the examined optimization problems are non-convex ones, whilst Adam is a gradient-based optimization algorithm. Hence, optimized solutions may be trapped in local minimum due to initial parameters. Accordingly, other robust optimizers such as gradient descent with momentum, Nesterov accelerated gradient, etc. [84] are promising to circumvent these shortcomings.

Acknowledgements This research was supported by a grant (NRF-2021R1A4A2002855) from NRF (National Research Foundation of Korea) funded by MEST (Ministry of Education and Science Technology) of Korean government.

Author Contributions Hau T. Mai: conceptualization, methodology, software, formal analysis, investigation, writing - original draft, writing—review and editing, visualization. Qui X. Lieu: methodology, writing—original draft, writing—review and editing. Joowon Kang: data curation, validation, resources. Jaehong Lee: conceptualization, methodology, supervision, funding acquisition.

Declarations

Conflict of 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.

References

1. Kaveh A, Mahjoubi S (2019) Hypotrochoid spiral optimization approach for sizing and layout optimization of truss structures with multiple frequency constraints. Eng Comput 35:1443–1462
2. Ho-Huu V, Vo-Duy T, Luu-Van T, Le-Anh L, Nguyen-Thoi T (2016) Optimal design of truss structures with frequency constraints using improved differential evolution algorithm based on an adaptive mutation scheme. Autom Constr 68:81–94
3. Kaveh A, Talatahari S (2009) Size optimization of space trusses using big bang-big crunch algorithm. Comput Struct 87:1129–1140
4. Khot N (1983) Nonlinear analysis of optimized structure with constraints on systemstability. AIAA J 21:1181–1186
5. Khot N, Kamat M (1985) Minimum weight design of truss structures with geometric nonlinear behavior. AIAA J 23:139–144
6. El-Sayed ME, Ridgely BJ, Sandgren E (1989) Nonlinear structural optimization using goal programming. Comput Struct 32:69–73
7. Saka M, Ulker M (1992) Optimum design of geometrically nonlinear space trusses. Comput Struct 42:289–299
8. Shin M-K, Park K-J, Park G-J (2007) Optimization of structures with nonlinear behavior using equivalent loads. Comput Methods Appl Mech Eng 196:1154–1167
9. Miguel LFF, Miguel LFF (2012) Shape and size optimization of truss structures considering dynamic constraints through modern metaheuristic algorithms. Expert Syst Appl 39:9458–9467
10. Lieu QX, Do DT, Lee J (2018) An adaptive hybrid evolutionary firefly algorithm for shape and size optimization of truss structures with frequency constraints. Comput Struct 195:99–112
11. Gomes HM (2011) Truss optimization with dynamic constraints using a particle swarm algorithm. Expert Syst Appl 38:957–968
12. Kaveh A, Zolghadr A (2014) Democratic pso for truss layout and size optimization with frequency constraints. Comput Struct 130:10–21
13. Toğan V, Daloğlu AT (2006) Optimization of 3d trusses with adaptive approach in genetic algorithms. Eng Struct 28:1019–1027
14. Zuo W, Bai J, Li B (2014) A hybrid oc-ga approach for fast and global truss optimization with frequency constraints. Appl Soft Comput 14:528–535
15. Pierezan J, dos Santos Coelho L, Mariani VC, de Vasconcelos Segundo EH, Prayogo D (2021) Chaotic coyote algorithm applied to truss optimization problems. Comput Struct 242:106353
16. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
17. Buntara G, Takahiro H, Aylie H, Alisjahbana S, As'ad S (2017) Evolutionary aco algorithms f or truss optimization problems. Proc Eng 171:1100–1107
18. Kaveh A, Zakian P (2018) Improved gwo algorithm for optimal design of truss structures. Eng Comput 34:685–707
19. Degertekin S (2012) Improved harmony search algorithms for sizing optimization of truss structures. Comput Struct 92:229–241
20. Le-Duc T, Nguyen Q-H, Nguyen-Xuan H (2020) Balancing composite motion optimization. Inf Sci 520:250–270
21. Fan H-Y, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Glob Optim 27:105–129
22. Koo B, Jung R, Yu Y (2021) Automatic classification of wall and door bim element subtypes using 3d geometric deep neural networks. Adv Eng Inform 47:101200
23. Thorat Z, Mahadik S, Mane S, Mohite S, Udugade A (2019) Self driving car using raspberry-pi and machine learning. IRJET 06:969–972
24. De Bruijne M (2016) Machine learning approaches in medical image analysis: from detection to diagnosis. Med Image Anal 33:94–97
25. Na H, Kim S (2021) Predicting stock prices based on informed traders' activities using deep neural networks. Econ Lett 204:109917
26. Jokar M, Semperlotti F (2021) Finite element network analysis: a machine learning based computational framework for the simulation of physical systems. Comput Struct 247:106484
27. Lee S, Kim H, Lieu QX, Lee J (2020) Cnn-based image recognition for topology optimization. Knowl Based Syst 198:105887
28. Papadopoulos V, Soimiris G, Giovanis D, Papadrakakis M (2018) A neural network-based surrogate model for carbon nanotubes with geometric nonlinearities. Comput Methods Appl Mech Eng 328:411–430
29. Zhuang X, Guo H, Alajlan N, Zhu H, Rabczuk T (2021) Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning. Eur J Mech A Solids 87:104225
30. Truong TT, Dinh-Cong D, Lee J, Nguyen-Thoi T (2020) An effective deep feedforward neural networks (dfnn) method for damage identification of truss structures using noisy incomplete modal data. J Build Eng 30:101244
31. Zhao J, Nguyen H, Nguyen-Thoi T, Asteris PG, Zhou J (2021) Improved Levenberg–Marquardt backpropagation neural network by particle swarm and whale optimization algorithms to predict the deflection of rc beams. Eng Comput 1–23. https://doi.org/10.1007/s00366-020-01267-6
32. Truong TT, Lee J, Nguyen-Thoi T (2021) Joint damage detection of structures with noisy data by an effective deep learning framework using autoencoder-convolutional gated recurrent unit. Ocean Eng 243:110142
33. Le HQ, Truong TT, Dinh-Cong D, Nguyen-Thoi T (2021) A deep feed-forward neural network for damage detection in functionally graded carbon nanotube-reinforced composite plates using modal kinetic energy. Front Struct Civ Eng 15:1453–1479
34. Truong TT, Lee S, Lee J (2020) An artificial neural network-differential evolution approach for optimization of bidirectional functionally graded beams. Compos Struct 233:111517
35. Shariati M, Mafipour MS, Mehrabi P, Shariati A, Toghroli A, Trung NT, Salih MN (2021) A novel approach to predict shear strength of tilted angle connectors using artificial intelligence techniques. Eng Comput 37:2089–2109
36. Lee S, Vo TP, Thai H-T, Lee J, Patel V (2021) Strength prediction of concrete-filled steel tubular columns using categorical gradient boosting algorithm. Eng Struct 238:112109
37. Hajela P, Berke L (1991) Neurobiological computational models in structural analysis and design. Comput Struct 41:657–667
38. Hajela P, Berke L (1991) Neural network based decomposition in optimal structural synthesis. Comput Syst Eng 2:473–481
39. Adeli H, Park HS (1995) Optimization of space structures by neural dynamics. Neural Netw 8:769–781
40. Kang H-T, Yoon CJ (1994) Neural network approaches to aid simple truss design problems. Comput Aid Civ Infrastruct Eng 9:211–218
41. Ramasamy J, Rajasekaran S (1996) Artificial neural network and genetic algorithm for the design optimization of industrial roofs-a comparison. Comput Struct 58:747–755
42. Iranmanesh A, Kaveh A (1999) Structural optimization by gradient-based neural networks. Int J Numer Methods Eng 46:297–311
43. Mai HT, Kang J, Lee J (2021) A machine learning-based surrogate model for optimization of truss structures with geometrically nonlinear behavior. Finite Elem Anal Des 196:106461
44. Nguyen LC, Nguyen-Xuan H (2020) Deep learning for computational structural optimization. ISA Trans 103:177–191
45. Ly DK, Truong TT, Nguyen-Thoi T (2021) Multi-objective optimization of laminated functionally graded carbon nanotube reinforced composite plates using deep feedforward neural

networks-nsgaii algorithm. Int J Comput Methods. https://doi.org/10.1142/S0219876221500651
46. Truong TT, Lee J, Nguyen-Thoi T (2021) Multi-objective optimization of multi-directional functionally graded beams using an effective deep feedforward neural network-smpso algorithm. Struct Multidiscip Optim 63:2889–2918
47. Li B, Huang C, Li X, Zheng S, Hong J (2019) Non-iterative structural topology optimization using deep learning. Comput Aided Des 115:172–180
48. White DA, Arrighi WJ, Kudo J, Watts SE (2019) Multiscale topology optimization using neural network surrogate models. Comput Methods Appl Mech Eng 346:1118–1135
49. Deng H, To AC (2021) A parametric level set method for topology optimization based on deep neural network. J Mech Des 143:091702
50. Abueidda DW, Koric S, Sobh NA (2020) Topology optimization of 2d structures with nonlinearities using deep learning. Comput Struct 237:106283
51. Nguyen-Thanh VM, Zhuang X, Rabczuk T (2020) A deep energy method for finite deformation hyperelasticity. Eur J Mech A/Solids 80:103874
52. Li W, Bazant MZ, Zhu J (2021) A physics-guided neural network framework for elastic plates: comparison of governing equations-based and energy-based approaches. Comput Methods Appl Mech Eng 383:113933
53. Guo H, Zhuang X, Rabczuk T (2021) A deep collocation method for the bending analysis of kirchhoff plate. arXiv:2102.02617
54. Panghal S, Kumar M (2020) Optimization free neural network approach for solving ordinary and partial differential equations. Eng Comput 37:2989–3002
55. Wang S, Wang H, Perdikaris P (2021) On the eigenvector bias of fourier feature networks: from regression to solving multi-scale pdes with physics-informed neural networks. Comput Methods Appl Mech Eng 384:113938
56. Jin X, Cai S, Li H, Karniadakis GE (2021) Nsfnets (navier-stokes flow nets): physics-informed neural networks for the incompressible navier-stokes equations. J Comput Phys 426:109951
57. Zhu Q, Liu Z, Yan J (2021) Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks. Comput Mech 67:619–635
58. Chandrasekhar A, Suresh K (2021) Tounn: topology optimization using neural networks. Struct Multidiscip Optim 63:1135–1149
59. Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, Wanderman-Milne S (2020) Jax: composable transformations of python+ numpy programs. 4:16. http://github.com/google/jax
60. Kaveh A, Ghazaan MI (2015) Hybridized optimization algorithms for design of trusses with multiple natural frequency constraints. Adv Eng Softw 79:137–147
61. Kaveh A, Zolghadr A (2012) Truss optimization with natural frequency constraints using a hybridized css-bbbc algorithm with trap recognition capability. Comput Struct 102:14–27
62. Kaveh A, Azar BF, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23:167–181
63. Sonmez M (2011) Artificial bee colony algorithm for optimization of truss structures. Appl Soft Comput 11:2406–2418
64. Hasançebi O (2008) Adaptive evolution strategies in structural optimization: enhancing their computational performance with applications to large-scale structures. Comput Struct 86:119–132
65. Cao X, Sugiyama Y, Mitsui Y (1998) Application of artificial neural networks to load identification. Comput Struct 69:63–78
66. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
67. Camarda CJ, Adelman HM (1984) Static and dynamic structural-sensitivity derivative calculations in the finite-element-based engineering analysis language (eal) system. No. NASA-TM-85743
68. Chandrasekhar A, Sridhara S, Suresh K (2021) Auto: a framework for automatic differentiation in topology optimization. arXiv:2104.01965
69. Lee S, Ha J, Zokhirova M, Moon H, Lee J (2018) Background information of deep learning for structural engineering. Arch Comput Methods Eng 25:121–129
70. Paul M (2018) Applied machine learning. https://cmci.colorado.edu/classes/INFO-4604/resources.html. Accessed 19 Apr 2021
71. Khatibinia M, Naseralavi SS (2014) Truss optimization on shape and sizing with frequency constraints based on orthogonal multi-gravitational search algorithm. J Sound Vib 333:6349–6369
72. Degertekin S, Bayar GY, Lamberti L (2021) Parameter free jaya algorithm for truss sizing-layout optimization under natural frequency constraints. Comput Struct 245:106461
73. Camp CV (2007) Design of space trusses using big bang-big crunch optimization. J Struct Eng 133:999–1008
74. Li L, Huang Z, Liu F, Wu Q (2007) A heuristic particle swarm optimizer for optimization of pin connected structures. Comput Struct 85:340–349
75. Degertekin S, Hayalioglu M (2013) Sizing truss structures using teaching-learning-based optimization. Comput Struct 119:177–188
76. Lamberti L (2008) An efficient simulated annealing algorithm for design optimization of truss structures. Comput Struct 86:1936–1953
77. Mohamed AW, Sabry HZ (2012) Constrained optimization based on modified differential evolution algorithm. Inf Sci 194:171–208
78. Mallipeddi R, Suganthan PN, Pan Q-K, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11:1679–1696
79. Pan Q-K, Suganthan PN, Wang L, Gao L, Mallipeddi R (2011) A differential evolution algorithm with self-adapting strategy and control parameters. Comput Oper Res 38:394–408
80. Rao RV, Kalyankar V, Waghmare G (2014) Parameters optimization of selected casting processes using teaching-learning-based optimization algorithm. Appl Math Model 38:5592–5608
81. Wang D, Tan D, Liu L (2018) Particle swarm optimization algorithm: an overview. Soft Comput 22:387–408
82. Ho-Huu V, Nguyen-Thoi T, Vo-Duy T, Nguyen-Trang T (2016) An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. Comput Struct 165:59–75
83. Yam JY, Chow TW (2000) A weight initialization method for improving training speed in feedforward neural network. Neurocomputing 30:219–232
84. Ng A (2019) Machine learning yearning: technical strategy for AI engineers in the era of deep learning [online]. https://www.deeplearning.ai/machine-learning-yearning/