Structures 88 (2026) 111934

Contents lists available at ScienceDirect

Structures

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

Topology and size optimization of trusses using graph neural networks: Towards efficient surrogate modeling

Nisal Ariyasinghe [a,*], Tharindu Wickremasinghe [b], Hansani Weeratunge [c], Chinthaka Mallikarachchi [d,e], Sumudu Herath [d]

[a] School of Engineering, University of Edinburgh, Edinburgh, UK
[b] Purdue University, West Lafayette, Indiana, United States
[c] Sri Lanka Institute of Information Technology, Malabe, Sri Lanka
[d] Department of Civil Engineering, University of Moratuwa, Moratuwa, Sri Lanka
[e] California Institute of Technology, Pasadena, California, United States

ARTICLE INFO

Keywords:
Structural optimization
Truss structures
Graph neural networks
Sustainable structural design
Surrogate models

ABSTRACT

Real-time structural optimization of trusses using machine learning techniques, incorporating both topology and size optimization, is more effective in discrete domains than in continuous ones, with Graph Neural Networks (GNNs) showing strong potential. However, the impact of convolutions in GNNs is not yet fully understood, limiting their full applicability. This paper presents a GNN-based surrogate model for real-time structural optimization and identifies the most suitable convolution type for this task. The study assesses the predictive performance of models trained on a dataset of optimized structures spanning a range of loads, boundary conditions, and design domain sizes. The resulting model effectively predicts both optimal topology and member sizes once the design parameters are provided. Eight graph convolution types are investigated to identify the most suitable method, alongside an evaluation of optimal network architectures. Among the tested approaches, Generalised Graph Convolution achieves the highest accuracy, followed by Topology Adaptive Graph Convolution, producing near-ideal topology predictions for most test cases and maintaining section size prediction errors within ±2% across all test data points. This framework demonstrates strong potential for broader applications.

1. Introduction

Optimization of structures has long been a central focus in structural engineering. At its core, structural optimization involves the formulation of an objective function, subject to a set of governing constraints, where both the objectives and constraints vary depending on the design requirements. In recent years, increasing emphasis on sustainability has driven the need to reduce material consumption and, consequently, the embodied carbon of structural systems. As a result, the minimization of structural self-weight has emerged as one of the most extensively studied objectives in the literature. Alongside this objective, a wide range of constraints has been incorporated into optimization frameworks. These include stress and equilibrium constraints [1], thermal behaviour constraints [2], reliability-based constraints [3,4], buckling constraints [5], and vibrational constraints [6]. The resulting optimized designs have found applications across multiple engineering domains, ranging from civil engineering structures [7], to aerospace systems [8], and automotive engineering applications [9].

Since the mid-20th century, a wide range of optimization methods have been developed to solve these optimization tasks, and these approaches continue to evolve in response to advances in computational techniques and increasing design complexity. These methods can be broadly categorized into density-based, boundary evolution-based, and explicit representation-based topology optimization approaches. In density-based topology optimization, each finite element is assigned a material density variable. Common methods in this category include Solid Isotropic Material with Penalization (SIMP) [10,11], Evolutionary Structural Optimization (ESO) [12], Bi-Directional Evolutionary Structural Optimization (BESO) [13], and Sequential Element Rejection and Admission (SERA) [14]. In boundary evolution-based topology optimization, the boundary of the structure is evolved directly instead of using density variables. Common approaches in

* Corresponding author.
Email address: N.C.A.Geekiyanage@sms.ed.ac.uk (N. Ariyasinghe).

https://doi.org/10.1016/j.istruc.2026.111934
Received 19 January 2026; Received in revised form 31 March 2026; Accepted 21 April 2026
Available online 5 May 2026

this category include the level set method (LSM) [2] and the bubble method [15]. In explicit representation-based topology optimization, the structure is described directly using geometric parameters or structural elements. Examples of such methods include the Moving Morphable Component (MMC) method [16], Moving Morphable Void (MMV) method [17], Particle Swarm Optimization (PSO) [18,19], and Genetic Algorithms (GA) [20,21]. Despite the advancements and diversity of these methods, the demand for reduced computational cost continues to grow in practical applications. The integration of machine learning (ML) into structural optimization has shown considerable promise in addressing this challenge by significantly reducing computational costs. However, which machine learning techniques are most effective and appropriate for structural optimization tasks is still debated.

The most common machine learning (ML) techniques used in structural optimization include Support Vector Machines (SVMs) and various types of Neural Networks (NNs), such as Feed Forward Neural Networks (FFNNs), Physics-Informed Neural Networks (PINNs), Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs), and Graph Neural Networks (GNNs). Among these, some methods are predominantly applied in continuous domains (e.g., image-based representations), such as CNNs and GANs, while others are more suited to discrete domains, such as PINNs and GNNs; however, their applicability is not strictly limited to these domains. In using ML for optimization processes within continuous design domains, Du et al. [22] proposed a generative model-based framework using Least Squares Generative Adversarial Networks (LSGANs) to synthesize new topology-optimized truss designs. Their approach used image augmentation and topology optimization under various loading conditions to create a rich dataset of labelled images, enhancing the generator's ability to replicate realistic structural layouts. Similarly, Seo and Kapania [23] developed a U-Net-based CNN as a surrogate model to predict optimal material densities from the results of the static analysis obtained by SIMP. Their model used finite element simulations in Abaqus/CAE (Computer Aided Engineering), mapping variables like displacement and strain to material distributions. Expanding on CNN-based methods, Zheng et al. [24] compared several deep learning architectures, including U-Net, Res-U-Net, Attention-U-Net, and their proposed Attention-Res-U-Net. Trained using data from the Moving Morphable Component method, the Attention-Res-U-Net outperformed all other models, achieving the highest precision (0.8134), F1 score (0.8001), and Jaccard Index (0.6710). Jayaweera et al. [25] used CNNs to predict truss structures along with integrating a geometric feature identification algorithm, to effectively overcome limitations of generator-induced losses. Hybrid frameworks have also emerged, such as Wang et al. [26], who combined Principal Component Analysis (PCA) with a feed-forward neural network. Their results showed significant accuracy improvements, with a minimum prediction error of 2.663% when using 22% of the sample size as eigenimages. Seo and Min [27] introduced a hybrid model integrating GNNs and Multi-Layer Perceptrons (MLPs) to handle non-Euclidean finite element mesh data. The model encoded spatial and mechanical information through graphs and predicted optimal density distributions. Jeong et al. [28] presented a physics-based approach using PINNs, which learned displacement fields by minimizing the system's potential energy without labelled data. Their method offered a mesh-free, partial differential equation-constrained solution to topology optimization, highlighting the potential of physics-integrated learning frameworks. In a recent study by Li et al. [29], structural optimization was decoupled into two sequential but interacting subproblems: (i) determining the optimal load location (device layout) using a Radial Basis Function Neural Network (RBF-NN) trained on adaptively sampled finite element data, coupled with PSO to efficiently identify the best load positions, and (ii) performing topology optimization under the identified loading conditions. The topology optimization was then carried out using a gradient-based level set method. While machine learning approaches can successfully generate optimized structural topologies in continuous design domains, accurately determining cross-sectional sizes from these outputs remains challenging making results difficult in size optimization. This difficulty arises from the pixel-based representation of cross-sectional properties, which can introduce significant errors, particularly when physical length scales are taken into account.

This limitation is significantly mitigated in machine learning–based optimization within discrete design domains, where topology is defined by element placement, and cross-sectional properties are represented through labels rather than pixel-based representations. This makes such approaches particularly well-suited for truss applications. The topology and size optimization approaches in discrete domains can be broadly categorized into three main branches: (i) methods that incorporate physics-based loss functions, where ML is used as the optimizer and the optimization problem is framed as a learning task; (ii) surrogate-based approaches that approximate finite element analysis (FEA); and (iii) data-driven methods that directly generate optimized designs from trained datasets. In the first two approaches, the repetitive evaluations inherent to the optimization process are not fully eliminated.

In physics-based optimization approaches, Mai et al. [30] incorporated Bayesian Optimization (BO) to automatically tune the hyperparameters of a deep neural network (DNN) for non-linear truss optimization. A Gaussian process surrogate was employed to minimize the DNN's prediction-based loss function, thereby improving model efficiency and performance. Mai et al. [31] further proposed Physics-Informed Direct Generation (PIDG) models, which embed physical principles and objective functions directly into the loss function to guide network training, effectively eliminating the need for external, computationally expensive structural analyses. A related PIDG-based approach, the Advanced Neural Network Algorithm (ANNA), operates as a metaheuristic optimizer using a multi-layered NN structure [32]. Another powerful physics-informed approach is Deep Reinforcement Learning (DRL). Fu et al. [33] demonstrated this in their FrameRL framework for automated steel frame design. This DRL-based method does not require large volumes of pre-processed data; instead, learning is guided by a physics-informed reward function that promotes designs that are both safe and economical. On the other hand, the development of surrogate models for FEA has significantly reduced the number of analyses required to reach an optimal solution, and in some cases can fully replace FEA with a computationally cheaper approximation. Mai et al. [34] introduced a DNN surrogate model integrated with a Differential Evolution (DE) algorithm. The DNN was trained on geometrically non-linear analysis data to predict nodal displacements from input cross-sectional areas. This approach demonstrated high accuracy, with Mean Squared Error (MSE) values of 0.059 and 0.051 for training and validation, respectively, and corresponding Root Mean Squared Error (RMSE) values of approximately 2.4%. Similarly, the Hybrid Intelligent Genetic Algorithm (HIGA) incorporates a DNN trained online during the optimization process to act as a high-efficiency surrogate for evaluating both objective and constraint functions [35]. Pham et al. [36] proposed a k-Nearest Neighbor Comparison (k-NNC) technique that eliminates the training phase by evaluating new design candidates based on their proximity to existing population members. Inferior candidates are rejected early, reducing the number of required structural analyses by 40% to 60%. Nourian et al. [37] proposed a GNN surrogate to predict nodal displacements, requiring FEA in only 10% of optimization iterations while maintaining an MSE of 0.022. Nguyen and Yu [38] employed an AdaBoost classifier within a DE framework to classify the safety of trial solutions. Trial vectors identified as unsafe and less optimal than the current best are discarded without further analysis, thereby improving computational efficiency. Jayaweera et al. [39] applied GNNs for action prediction and combined neural networks with particle swarm optimization for beam–slab layout and reinforcement design.

The third category involves the direct generation of optimized designs from trained datasets, which has the potential to eliminate iterative processes entirely in real-time optimization applications. However, its application to structural optimization remains relatively underexplored, with only a limited number of studies investigating this approach [24,

40,41]. Zhu et al. [40] applied GNNs to automate the generation of machine-defined ground structures; however, their work did not explicitly target real-time structural optimization. Similarly, Zheng et al. [24] employed graph convolution techniques for real-time topology and size optimization. Building on this, Ariyasinghe et al. [41] proposed a GNN-based approach to directly predict optimized topologies and member sizes using Topology Adaptive Graph Convolutions. Despite these developments, a comprehensive understanding of how different graph convolution architectures influence model performance in structural optimization is still lacking. Since convolution operations constitute the core mechanism through which GNNs propagate and aggregate information, their influence on structural optimization outcomes is of critical importance. Most existing studies rely on basic convolutional architectures and do not fully utilize the power of GNNs. This presents a significant opportunity to investigate and identify convolution strategies that are better suited for discrete structural optimization tasks. Building upon prior research, this study aims to explore the role of various graph convolution methods within GNN frameworks, with the objective of enhancing both performance and generalizability in large-scale structural optimization problems.

2. Ground-truth discrete optimization framework

The ground-truth discrete topology and size optimization framework used in this study is based on the approach presented by Zegard and Paulino [1], which aims to minimize the total volume of a truss structure while satisfying stress constraints in its members. Since the present work focuses on exploring the potential of GNNs as surrogates for existing optimization techniques, the purpose here is not to refine the optimization process developed by Zegard and Paulino [1]. Instead, this section provides a brief overview of the underlying optimization procedure (Fig. 1) to facilitate understanding. Readers interested in a detailed explanation are referred to the original work of Zegard and Paulino [1].

The design domain is first discretized into a base mesh, establishing the nodal distribution over which the optimization process is performed. A ground structure is created by connecting the nodes with a set of potential truss members, denoted as N_b, according to a specified connectivity level. Redundant or overlapping members are removed using a collinearity check to prevent duplicate or coincident elements. The remaining members are represented by a connectivity matrix $\mathbf{A}_C \in \mathbb{R}^{N\times N}$, where N is the number of nodes. The matrix defines all possible structural connections between nodes, such that $A_{C,k,k'} = 1$ indicates the presence of a member between nodes k and k', while $A_{C,k,k'} = 0$ indicates no connection. The matrix is symmetric, i.e., $A_{C,k,k'} = A_{C,k',k}$.

Once the connectivity matrix is derived, the optimization problem is formulated with the objective of minimizing the total structural volume V, expressed as a function of the cross-sectional areas $\mathbf{a}$ of the structural members:

$$\min_{\mathbf{a}} V = \mathbf{l}^T\mathbf{a} \tag{1}$$

where $\mathbf{l}$ is the vector of the lengths of the members and $\mathbf{a}$ is the vector of their corresponding cross-sectional areas. The equilibrium condition enforces balance between internal member forces and external nodal loads:

$$\mathbf{B}^T\mathbf{n} = \mathbf{f}. \tag{2}$$

where $\mathbf{B}^T$ is the transposed equilibrium matrix constructed from the directional cosines of the members, $\mathbf{n}$ is the vector of internal axial forces, and $\mathbf{f}$ represents the external nodal forces applied to the structure. These equilibrium constraints ensure that the structure is statically admissible. Stress constraints are imposed to ensure that internal forces in each member remain within the allowable tensile and compressive limits, $\sigma_T > 0$ and $\sigma_C > 0$, respectively:

$$-\sigma_C a_i \le n_i \le \sigma_T a_i \quad \text{for} \quad i = 1, 2, \dots, N_b. \tag{3}$$

To transform these inequality constraints into equalities, slack variables $\mathbf{s}^+$ and $\mathbf{s}^-$ are introduced as follows:

$$n_i + \frac{2\sigma_0}{\sigma_C} s_i^- = \sigma_T a_i \tag{4}$$

$$-n_i + \frac{2\sigma_0}{\sigma_T} s_i^+ = \sigma_C a_i. \tag{5}$$

where σ_0 is the average limit stress $(\sigma_T + \sigma_C)/2$. Using these definitions, the cross-sectional area a_i and the internal force n_i are expressed as:

$$a_i = \frac{s_i^+}{\sigma_T} + \frac{s_i^-}{\sigma_C} \tag{6}$$

$$n_i = s_i^+ - s_i^-. \tag{7}$$

This formulation enables a clear distinction between tension and compression, as only one of s_i^+ or s_i^- is non-zero for each member. By substituting these relationships into the objective and equilibrium equations, the problem is transformed into a linear programming formulation:

$$\min_{\mathbf{s}^+,\mathbf{s}^-} V = \mathbf{l}^T\left(\frac{\mathbf{s}^+}{\sigma_T} + \frac{\mathbf{s}^-}{\sigma_C}\right) \tag{8}$$

$$\text{subject to: } \mathbf{B}^T(\mathbf{s}^+ - \mathbf{s}^-) = \mathbf{f} \tag{9}$$

$$s_i^+, s_i^- \ge 0 \tag{10}$$

The optimization gives the optimum values of the slack variables, which are used to obtain the final area vector $\mathbf{a}$ using the Eq. (6). This results in a topology- and size-optimized structure.

3. Truss optimization with graph neural networks

3.1. Graph convolutions

GNNs are a class of neural networks specifically designed to work with graph-structured data [42]. Unlike traditional neural networks, which are primarily suited for grid-like data, GNNs excel in scenarios where the relationships between entities are naturally represented as graphs [43]. These networks use the inherent structure of graphs, where nodes represent entities and edges depict the relationships between them. The graph neural network learns through data being convolved between the nodes. Therefore, the nature of convolutions plays a significant role in the performance of the framework. The convolution types discussed in this section are later used in this paper to compare their suitability and degree of performance in predicting the optimized truss structures. The PyTorch Geometric (PyG) Python library [44] is used to import these convolution types for the GNN models. The following notations are commonly used in the convolution formulations presented in Table 1. The matrix $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}_N$ denotes the adjacency matrix with added

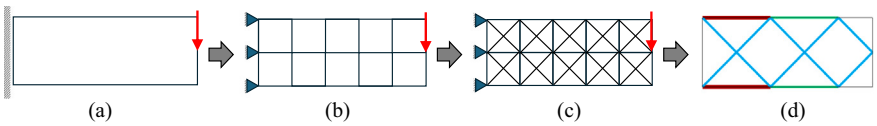


Fig. 1. Base mesh (b) and ground structure generation (level 1 connectivity) (c) of a rectangular cantilever design domain (a) and the final optimized structure (d).

Table 1
Summary of graph convolution operators, key formulations, and representative applications.

Layer	Core Idea/Formulation	Example applications
GCNConv [45]	Performs spectral graph convolution using the renormalized graph Laplacian. The propagation rule for each layer is $\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$ The matrices $\mathbf{H}^{(l)}$ and $\mathbf{H}^{(l+1)}$ represent the input and output node feature matrices at layer l and $l+1$, respectively, $\mathbf{W}^{(l)}$ is the trainable weight matrix. This formulation enables one-hop neighborhood aggregation with symmetric normalization.	[46]: Brain connectivity modeling (MRI) [47]: 3D face analysis [48]: Social media toxicity classification
GATConv [49]	Uses masked self-attention to adaptively weight neighboring node features. For a graph with node features $\mathbf{h}_i \in \mathbb{R}^F$, a shared linear transformation $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is first applied. Attention coefficients between node i and its neighbor $j \in \mathcal{N}_i$ are computed as $\alpha_{ij} = \text{softmax}_j\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j]\right)\right)$, where $\mathbf{a} \in \mathbb{R}^{2F'}$ is a learnable attention vector and softmax_j normalizes over all $j \in \mathcal{N}_i$. The updated node representation is $\mathbf{h}'_i = \Big\|_{k=1}^{K} \sigma\Big(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j\Big)$, where $\mathcal{N}_i$ is the set of neighboring nodes of node i, K is the number of attention heads, $\mathbf{W}^k$ and α_{ij}^k denote head-specific weights and attention coefficients. In the final layer, the outputs of attention heads are averaged instead of concatenated.	[50]: Brain mesh connectivity [51]: Flood segmentation via image-graph attention
TAGConv [52]	Defines convolution in the vertex domain using a polynomial filter over the normalized adjacency matrix. The layer-ℓ polynomial filter mapping input channel c to output channel f is $\mathbf{G}_{c,f}^{(\ell)} = \sum_{k=0}^{K} g_{c,f,k}^{(\ell)} \tilde{\mathbf{A}}^k$, where $g_{c,f,k}^{(\ell)}$ are learnable coefficients and K is the polynomial order (filter size). The output for channel f is $\mathbf{x}_f^{(\ell+1)} = \sigma\Big(\sum_{c=1}^{C_\ell} \mathbf{G}_{c,f}^{(\ell)} \mathbf{x}_c^{(\ell)} + b_f^{(\ell)} \mathbf{1}_N\Big)$, where $\mathbf{x}_c^{(\ell)} \in \mathbb{R}^N$ and $\mathbf{x}_f^{(\ell+1)} \in \mathbb{R}^N$ are the input and output feature vectors for channels c and f, respectively, C_ℓ is the number of input channels, $b_f^{(\ell)}$ is a bias term, $\mathbf{1}_N$ is a vector of ones. The powers $\tilde{\mathbf{A}}^k$ encode k-hop neighborhood information, enabling multi-hop aggregation without requiring spectral decomposition.	[53]: WiFi fingerprint graph modeling [54]: Mild cognitive impairment detection
ClusterConv [55]	Extends the GCN framework by increasing the contribution of self-features during aggregation. The node feature matrix at layer l is $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times F_l}$ and the trainable weight matrix is $\mathbf{W}^{(l)} \in \mathbb{R}^{F_l \times F_{l+1}}$. The feature matrix after message passing is $\mathbf{X}^{(l+1)} = \sigma\left(\left(\hat{\mathbf{A}} + \lambda\,\text{diag}(\hat{\mathbf{A}})\right)\mathbf{X}^{(l)}\mathbf{W}^{(l)}\right)$, where $\hat{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-1}(\mathbf{A}+\mathbf{I})$ is a row-normalized adjacency matrix with self-loops, $\text{diag}(\hat{\mathbf{A}})$ extracts the diagonal elements of $\hat{\mathbf{A}}$ and forms a diagonal matrix, $\lambda \in \mathbb{R}$ is a scaling parameter controlling the contribution of self-features. This modification increases the influence of each node's own features relative to its neighbors during aggregation.	[56]: Large-scale GCN training [57]: Scalable partition-based training
GENConv [58]	Adopts a message-passing framework in which node features are iteratively updated through learnable message construction, aggregation, and update functions. For a graph with nodes v and neighbors $u \in \mathcal{N}(v)$, where $\mathcal{N}(v)$ denotes the set of neighboring nodes of v (including self-loops if present), the message from node u to node v at layer l is defined as $\mathbf{m}_{vu}^{(l)} = \rho^{(l)}\left(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}, \mathbf{h}_{evu}^{(l)}\right)$, where $\mathbf{h}_v^{(l)}$ and $\mathbf{h}_u^{(l)}$ are node feature vectors, $\mathbf{h}_{evu}^{(l)}$ is the edge feature vector associated with edge (v,u), and $\rho^{(l)}(\cdot)$ is a learnable message function (e.g., a neural network). Messages are aggregated using a permutation-invariant function: $\mathbf{m}_v^{(l)} = \zeta^{(l)}\left(\{\,\mathbf{m}_{vu}^{(l)} \mid u \in \mathcal{N}(v)\,\}\right)$, where $\zeta^{(l)}(\cdot)$ is a learnable aggregation operator acting on the multi-set of incoming messages. The node update applies message normalization and a residual connection: $\mathbf{h}_v^{(l+1)} = \text{MLP}\Big(\mathbf{h}_v^{(l)} + s \cdot \frac{\|\mathbf{h}_v^{(l)}\|_2}{\|\mathbf{m}_v^{(l)}\|_2} \cdot \mathbf{m}_v^{(l)}\Big)$, $s \in \mathbb{R}$ is a learnable scaling parameter. In practice, the architecture employs pre-activation residual blocks with normalization and ReLU activations to enable deeper graph networks.	[59]: Mammogram micro-calcification modeling [60]: Molecular property prediction
LEConv [61]	Designed to capture local extrema by emphasizing contrasts between a node and its neighbors. For a graph with nodes i and neighbors $j \in \mathcal{N}(i)$, where $\mathcal{N}(i)$ denotes the set of neighboring nodes of node i (including self-loops if present), the cluster-level formulation is $\boldsymbol{\phi}_i = \sigma\left(\mathbf{x}_i^c \mathbf{W}_1 + \sum_{j\in\mathcal{N}(i)} A_{ij}^c \left(\mathbf{x}_i^c \mathbf{W}_2 - \mathbf{x}_j^c \mathbf{W}_3\right)\right)$, where $\mathbf{x}_i^c \in \mathbb{R}^F$ is the cluster-level feature vector of node i, A_{ij}^c is the (i,j)-th entry of the cluster-level adjacency matrix, and $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{F \times F'}$ are learnable weight matrices. A more general formulation is $\mathbf{f}_i = \sigma\left(\alpha_i \mathbf{x}_i \mathbf{W} + \sum_{j\in\mathcal{N}(i)} \beta_{i,j}\left(\mathbf{x}_i \mathbf{W} - \mathbf{x}_j \mathbf{W}\right)\right)$, where $\mathbf{x}_i \in \mathbb{R}^F$ is the node feature vector, $\mathbf{W} \in \mathbb{R}^{F \times F'}$ is a learnable weight matrix, and $\alpha_i, \beta_{i,j} \in \mathbb{R}$ are learnable or adaptive scalar coefficients controlling the contributions of self-features and neighbor differences, respectively. This formulation highlights local feature differences and is particularly effective for detecting extrema and structural variations in graphs.	[62]: Drug–target affinity prediction [63]: Protein/drug substructure modeling
ARMAConv [64]	Approximates rational spectral filters using a recursive Graph Convolutional Skip (GCS) update. $\bar{\mathbf{X}}^{(t)} \in \mathbb{R}^{N \times F'}$ is the feature matrix at iteration t, with initialization $\bar{\mathbf{X}}^{(0)} = \mathbf{X}$. At iteration $t+1$, node features are updated as $\bar{\mathbf{X}}^{(t+1)} = \sigma\left(\tilde{\mathbf{L}}\bar{\mathbf{X}}^{(t)}\mathbf{W} + \mathbf{X}\mathbf{V}\right)$, where $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is the symmetric normalized graph Laplacian. The matrices $\mathbf{W} \in \mathbb{R}^{F' \times F'}$ and $\mathbf{V} \in \mathbb{R}^{F \times F'}$ are learnable weights. The skip term $\mathbf{X}\mathbf{V}$ preserves original signals and mitigates over-smoothing. The update is applied recursively for T iterations, and expressiveness can be enhanced using K parallel GCS stacks indexed by $k = 1, \ldots, K$, whose outputs are combined (e.g., averaged or concatenated) to produce the final representation.	[65]: Biomedical segmentation [66]: LiDAR tree species profiling [67]: Hyperspectral land-cover classification
EGConv [68]	Enables adaptive, node-specific filtering by combining multiple graph filters weighted by node-wise coefficients. The EGC output is $\mathbf{Y} = \sum_{b=1}^{B} w_b \odot \mathbf{y}_b$, where $\mathbf{y}_b = g_{\theta_b}(\mathbf{L})\mathbf{X} = \mathbf{U}\, g_{\theta_b}(\mathbf{\Lambda})\,\mathbf{U}^\top\mathbf{X}$, where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian. The functions $g_{\theta_b}(\cdot)$ are learnable spectral filters parametrized by θ_b, and $w_b \in \mathbb{R}^N$ are node-wise weighting coefficients, broadcast across feature dimensions. To avoid explicit eigendecomposition, a first-order Chebyshev approximation yields $\mathbf{Y} = \sum_{b=1}^{B} w_b \odot \left(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta}_b\right)$, where $\mathbf{\Theta}_b \in \mathbb{R}^{F \times F'}$ are learnable weight matrices. This formulation enables efficient, node-adaptive filtering without spectral decomposition.	[69]: Nanobody paratope prediction [70]: Molecular dipole prediction

self-loops, where $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ is the identity matrix for a graph with N nodes. The corresponding degree matrix $\tilde{\mathbf{D}}$ has diagonal entries defined as $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, where i and j denote node indices. Similarly, $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents the adjacency matrix of the graph, and $\mathbf{D}$ is its degree matrix with entries $D_{ii} = \sum_j A_{ij}$. The normalized adjacency matrix is defined as $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. For a graph with N nodes, the input node feature matrix is denoted by $\mathbf{X} \in \mathbb{R}^{N \times F}$ and the output by $\mathbf{Y} \in \mathbb{R}^{N \times F'}$, where F and F' represent the number of input and output features per node, respectively. Additional notation is introduced to aid interpretation of the convolution formulations: $\odot$ denotes the element-wise (Hadamard) product, $\|\cdot\|_2$ denotes the Euclidean norm, and $\text{MLP}(\cdot)$ represents a multi-layer perceptron. The eigendecomposition of the graph Laplacian is given by $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is the matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. The function $\sigma(\cdot)$ denotes a nonlinear activation function (such as ReLU (Rectified Linear Unit) or LeakyReLU).

3.2. Graph representation of truss structures for GNN

When optimization is carried out for an unseen input, the GNN-based optimization follows the framework shown in Fig. 2, using the graph

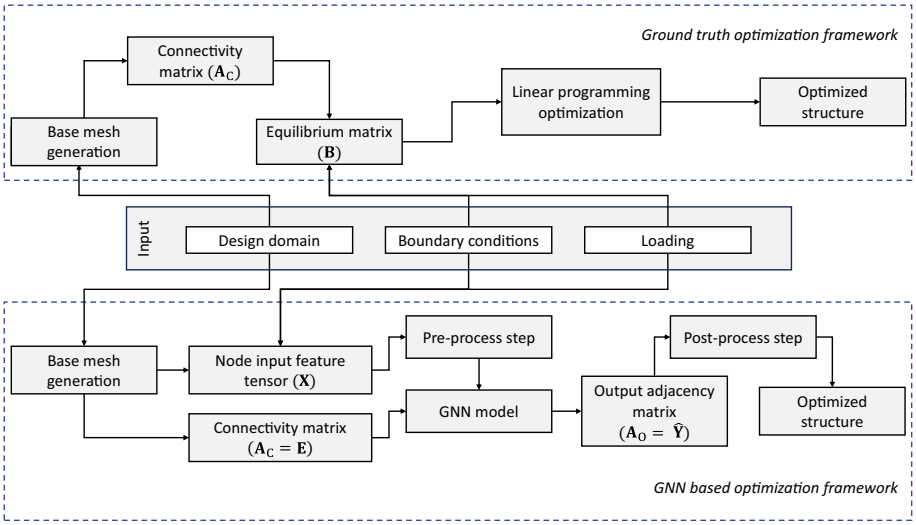


Fig. 2. Ground-truth discrete optimization framework and proposed GNN based surrogate model for optimization.

Table 2
Summary of notations used for graph representation in the GNN-based structural optimization framework.

Notation	Description
$G = (\mathcal{N}, \mathcal{E})$	Graph representation of the truss structure
$\mathcal{N} = \{\text{node}_1, \ldots, \text{node}_N\}$	Set of nodes in the graph
$\mathcal{E}$	Set of edges in the graph between nodes
N	Total number of nodes in the graph
$\mathbf{x}_k$	Feature vector of node k
(x_k, y_k)	Spatial coordinates of node k
$(L_{x,k}, L_{y,k})$	External loads applied on node k
$(S_{x,k}, S_{y,k})$	Global restraints of node k
$\mathbf{X} \in \mathbb{R}^{N\times 6}$	Node feature matrix of graph
$\mathbf{A}_C \in \mathbb{R}^{N\times N}$	Connectivity matrix of graph
$\mathbf{A}_O \in \mathbb{R}^{N\times N}$	Optimized output adjacency matrix

representations defined in this section (use Table 2 for notation definitions). Each truss structure is represented as a graph $G = (\mathcal{N}, \mathcal{E})$, where node-level information is encoded through feature vectors and assembled into the matrix $\mathbf{X}$. The nodal feature vector of node_k is a 6-dimensional vector: $\mathbf{x}_k = [x_k, y_k, L_{x,k}, L_{y,k}, S_{x,k}, S_{y,k}]$, based on the nodal locations, load and boundary condition inputs in the (x, y) coordinate system in the 2D design domain.

In addition to the nodal features, it is essential to provide the GNN with $\mathcal{E}$, which enables information propagation between nodes and forms the basis of graph convolutions. For undirected graphs, edges facilitate bidirectional message passing between nodes, and therefore a symmetric adjacency matrix is employed. The connectivity matrix $\mathbf{A}_C$, generated within the ground truth optimization framework [1], provides an appropriate representation of these edges, where $A_{C,k,k'} = \begin{cases} 1 & \text{if node } k \text{ is connected to node } k', \\ 0 & \text{otherwise.} \end{cases}$

The output of the structural optimization is given by an output adjacency matrix $\mathbf{A}_O$. Each entry $A_{O,k,k'}$ gives the cross-sectional area a of the member between nodes k and k' in the optimized structure, with $A_{O,k,k'} = \begin{cases} a > 0 & \text{if a member exists between nodes } k \text{ and } k', \\ 0 & \text{otherwise.} \end{cases}$

The matrix $\mathbf{A}_O$ is symmetric and sparse, containing non-zero values only where members exist in the final optimized layout. It jointly encodes the structural topology (member presence) and size (cross-sectional area) in a single tensor.

The complete methodology is illustrated in Fig. 2. The construction of the connectivity matrix $\mathbf{A}_C$ for the GNN-based optimization follows the procedure outlined in Zegard and Paulino [1] which is briefly described in Section 2.

3.3. Network architecture

The GNN architecture used in this study follows a consistent design across all convolution operators. Each model processes node features with six input channels, as described in Section 3.2. The architecture employs three hidden layers, selected through iterative testing. Although increasing network depth can enhance the ability of a model to learn hierarchical representations, existing studies indicate that the benefit of additional layers decreases after a certain point, while training time and optimization challenges increase significantly [71]. In our experiments, adding more than three layers did not yield a noticeable improvement in the final loss, which suggests that further depth would increase computational cost without enhancing predictive accuracy. Therefore, three hidden layers offered the most effective balance between model complexity, training efficiency, and performance. Residual connections help reduce the problem of vanishing gradients and support more stable convergence in deeper networks [72]. Within this architecture, a residual link between the first and third hidden layers proved most effective based on trial evaluations, since it improved gradient propagation without introducing unnecessary additional parameters. ReLU activation functions are applied selectively to the hidden layers in order to capture non-linear relationships that cannot be represented by linear transformations alone [73]. The final layer outputs an N-dimensional output vector for each node, which is used to build the output adjacency matrix that represents the predicted topology and size optimized structural layout, as outlined in Section 3.2. The forward pass of the model is summarized in Algorithm 1.

3.4. Data generation

The dataset used to train and evaluate the GNN was generated using a procedural pipeline derived from the optimization framework introduced in Section 2 and the graph representation described in Section 3.2. The process explores a range of truss configurations by varying domain size, boundary conditions, and loading, while maintaining a fixed 55-node orthogonal grid (10 divisions along the length and 4 along the height).

The primary objective of this study is to demonstrate the capability of Graph Neural Networks (GNNs) as surrogate models for structural optimization. To this end, several case studies were considered under two types of boundary conditions: cantilever (C) and simply supported (SS).

Algorithm 1 Forward Pass of Graph Convolutions.

Input:
  Node features: x
  Edge index: $edge\_index$
Initialize:
  conv_1: Conv layer from $I\_channels$ to $H\_channels_1$
  conv_2: Conv layer from $H\_channels_1$ to $H\_channels_2$
  conv_3: Conv layer from $H\_channels_2$ to $H\_channels_3$
  conv_4: Conv layer from $H\_channels_3$ to $O\_channels$
procedure FORWARD(x, $edge\_index$)
  $x_1 \leftarrow \text{ReLU}(\text{conv}_1(x, edge\_index))$
  $x_2 \leftarrow \text{conv}_2(x_1, edge\_index)$
  $x_3 \leftarrow \text{ReLU}(\text{conv}_3(x_2, edge\_index)) + x_1$
  $x_4 \leftarrow \text{conv}_4(x_3, edge\_index)$
  return x_4
end procedure

Each boundary condition was further combined with two categories of external loading, namely point loads (PL) and uniformly distributed loads (UDL). The selected load ranges and configurations are representative of typical roof truss applications. However, the proposed framework is not limited to these conditions and can be readily extended to accommodate heavier loads or larger design domains by augmenting the dataset used for model training. To reduce computational effort, the simply supported configuration was modeled as a half-domain, with a pinned support at one bottom corner and roller supports along the opposite vertical edge. The design domain length, L_x, was varied between 1 m and 20 m, while the height, L_y, ranged from 0.32 m to 9.6 m. The height-to-length ratios L_y/L_x, were constrained such that they remained within the range 0.32 to 0.48, ensuring geometrically reasonable aspect ratios for the design domains. Point loads were sampled between 1 kN and 10 kN, whereas UDLs ranged from 0.05 kN/m to 10 kN/m. For the cantilever case, PLs were applied at the free edge, and UDLs were distributed along the top, bottom, and free edges of the domain. In the simply supported configuration, PLs were applied at the roller-supported edge, while UDLs were distributed along the top or bottom edges, either over the full length, half, or quarter of the span. The geometric parameters and load magnitudes were sampled using the Latin Hypercube Sampling (LHS) technique to ensure uniformly distributed but stratified coverage of the design space [74]. For each sampled configuration, the structural optimization was carried out as detailed in Section 2.

Each optimized structure was encoded into a graph-based representation comprising node features, connectivity information, and optimized cross-sectional areas, as described in Section 3.2. The dataset $\mathcal{D}$ of sample inputs and their corresponding outputs is generated. The complete dataset, consisting of n_t graphs (data points), is constructed as

$$\mathcal{D} = \left\{ (\mathbf{X}^{(i)}, \mathbf{A}_{\mathrm{C}}^{(i)}, \mathbf{A}_{\mathrm{O}}^{(i)}) \;\middle|\; i = 1, \ldots, n_t \right\},$$

where each tuple represents a graph sample with its corresponding input features, connectivity, and optimized output.

3.5. Model training

This study investigates the performance of 12 GNN models to evaluate how different convolution types, data sample sizes, and architectural complexities influence the predictive capabilities of the model. Each model employs one of eight convolutional variants introduced in Section 3.1, with the same convolution type applied across all layers of a given model. The models are constructed using a shared forward-pass configuration described in Algorithm 1, where the four key convolutional layers ($conv_1$ through $conv_4$) implement the selected convolution type. To evaluate the impact of data volume on model performance, three datasets with varying sizes are used: 6000 samples (D1), 12,000 samples (D2), and 24,000 samples (D3). Alongside this, three levels of architectural complexity are examined. These configurations differ only in the number of hidden channels per layer while maintaining the same overall structure. Architecture A1 uses 36 hidden channels in the first and third convolutional layers ($H_{\text{channels},1}$ and $H_{\text{channels},3}$), and 81 in the second layer ($H_{\text{channels},2}$). A2 expands the layer widths to 64, 121, and 64 channels respectively, and A3 further increases them to 100, 225, and 100. This experimental setup facilitates a structured comparison, where each trained model is uniquely defined by its convolution type, dataset size, and architectural configuration.

Hyperparameters of the convolutions play a major role in the predictive capability of the model [75]. In the study a trial and error hyperparameter tuning was conducted and all relevant hyperparameters selected are specified as shown in Table 3.

The dataset

$$\mathcal{D} = \left\{ (\mathbf{X}^{(i)}, \mathbf{A}_{\mathrm{C}}^{(i)}, \mathbf{A}_{\mathrm{O}}^{(i)}) \;\middle|\; i = 1, \ldots, n_t \right\} \rightarrow \left\{ (\mathbf{X}^{(i)}, \mathbf{E}^{(i)}, \mathbf{Y}^{(i)}) \;\middle|\; i = 1, \ldots, n_t \right\}$$

is split into a training set $\mathcal{D}_{\text{train}}$ and a validation set $\mathcal{D}_{\text{test}}$ using an 80:20 ratio. The total number of graphs $n_t = n_{tr} + n_{te}$, where n_{tr} is the number of training graphs and n_{te} is the number of validation graphs. These datasets are structured as pairs of input node features and target adjacency matrices. Specifically, the training set is given by $\mathcal{D}_{\text{train}} = \{(\mathbf{X}_{tr}^{(i)}, \mathbf{E}_{tr}^{(i)}, \mathbf{Y}_{tr}^{(i)}) \mid i = 1, \ldots, n_{tr}\}$, and the validation set is given by $\mathcal{D}_{\text{test}} = \{(\mathbf{X}_{te}^{(i)}, \mathbf{E}_{te}^{(i)}, \mathbf{Y}_{te}^{(i)}) \mid i = 1, \ldots, n_{te}\}$. The GNN model $f_\theta(\cdot)$ is trained to predict adjacency matrices representing optimized truss configurations. The model predictions are given by $\hat{\mathbf{Y}}_*^{(i)} = f_\theta\left(\mathbf{X}_*^{(i)}, \mathbf{E}_*^{(i)}\right)$, where the subscript $* \in \{tr, te\}$ denotes either the training or validation (evaluation) phase.

To ensure physical consistency, symmetry is enforced and self-loops are removed using a zero-diagonal mask $\mathbf{M}_{\text{zero diag}}$. These operations are combined into a single post-processing step:

$$\hat{\mathbf{Y}}_*^{(i)} \leftarrow \left(\frac{\hat{\mathbf{Y}}_*^{(i)} + \hat{\mathbf{Y}}_*^{(i)\top}}{2} \right) \odot \mathbf{M}_{\text{zero diag}}.$$

After a series of trial-and-error experiments, the best-performing combination of preprocessing steps is summarized here. Standard normalization was applied to each node feature in the set $\left\{ \mathbf{X}_{tr}^{(i)} \;\middle|\; i = 1, \ldots, n_{tr} \right\}$, which led to an improvement in model accuracy. This behaviour is consistent with established practice in machine learning, since node-level features often span different numerical scales and unscaled features can disproportionately influence the learning process and normalization is widely recognised as an effective strategy for mitigating this issue [76]. The normalization parameters for $\mathbf{X}_{tr}$ were saved and were used to normalize the $\mathbf{X}_{te}$.

The same normalization procedure was initially applied to the output adjacency matrices $\left\{ \mathbf{Y}_{tr}^{(i)} \;\middle|\; i = 1, \ldots, n_{tr} \right\}$, but this resulted in a reduction in performance. In these matrices, non-zero entries indicate existing structural members, while zero entries indicate the absence of connections. The prediction task is therefore sensitive, as the model must determine both the binary topology and the cross-sectional areas of existing members. Normalization compressed small but non-zero values too far towards zero, making it difficult for the GNN to distinguish between elements with smaller cross section areas and absent ones. A two-stage classification and regression strategy was investigated but did not yield satisfactory results. To preserve the topological information while scaling non-zero cross-sectional areas, a selective min–max scaling approach was adopted. All positive entries of $\mathbf{Y}$ were linearly scaled to a predefined range $[\min, \max]$, while all zero entries were left unchanged. This allowed the model to clearly identify the eliminated members in the optimized topology. Through iterative tuning, the range 500 to 2000 provided the best overall performance. The scaling procedure is

Table 3
Hyperparameters for the convolutions imported from PyG.

Model	Hyperparameters $\boldsymbol{\Theta}$
GENConv	aggr = Softmax, t = 1.0, learn_t = False, p = 1.0, learn_p = False, msg_norm = False, learn_msg_scale = False, norm = batch, num_layers = 2, expansion = 2, eps = 1×10^{-7}, bias = True, edge_dim = None
TAGConv	K = 3, bias = True, normalize = True
ARMAConv	num_stacks = 1, num_layers = 1, act = Relu, shared_weights = False, dropout = 0, bias = True
GATConv	concat = False, neg_slope = 0.2, dropout = 0, add_self_loops = True, edge_dim = None, fill_value = mean, bias = True, **Per-Conv parameters:** Conv1 - heads = 2, Conv2 - heads = 4, Conv3 - heads = 5, Conv4 - heads = 4
GCNConv	improved = False, cached = False, add_self_loops = None, normalize = True, bias = True
CLUSTERConv	diag_lambda = 0, add_self_loops = True, bias = True
EGConv	aggregators = symnorm, num_bases = 4, cached = False, add_self_loops = True, bias = True, **Per-Conv parameters:** Conv1 - num_heads = 6, Conv2 - num_heads = 9, Conv3 - num_heads = 6, Conv4 - num_heads = 5
LEConv	bias = True

presented in Eq. (11).

$$y_{i,k,k'} = \begin{cases} 500 + \dfrac{y^{(i)}_{k,k'} - y_{\min}}{y_{\max} - y_{\min}} \cdot (2000 - 500) & \text{if } y^{(i)}_{k,k'} > 0, \\ 0 & \text{if } y^{(i)}_{k,k'} = 0, \end{cases} \tag{11}$$

Here, $y^{(i)}_{k,k'}$ denotes the (k,k') element of the matrix $\mathbf{Y}^{(i)}$, for $i = 1,\dots,n_t$, while $y_{\min}$ and $y_{\max}$ are the minimum and maximum values over all matrix elements of $\left\{\mathbf{Y}^{(i)}_{tr} \mid i = 1,\dots,n_{tr}\right\}$. During evaluation in Section 4.2, the inverse transformation is applied.

The Asymmetric Mean Squared Error (AMSE), defined in Eq. (12), was selected as the error function [77]. Asymmetric loss functions have been applied in areas such as engine useful life prediction [78], image processing [79], and bridge pier scour prediction [80]; however, asymmetric loss functions have not yet been widely applied to structural optimization. This asymmetry penalizes under-predictions more heavily than over-predictions, based on the intuition that, from a safety perspective, it is preferable to overestimate the cross-section rather than underestimate it. A weight factor of $w = 3$ is used to scale the penalty for underestimating the cross-sectional area of truss members after an iterative tuning process.

The training dataset was partitioned into batches of size $B = 64$, ensuring that memory usage remained within GPU limits. To emulate training with a larger effective batch size, gradients were accumulated over $A = 4$ consecutive batches before performing a parameter update. This corresponds to an effective batch size of $AB = 256$, resulting in more stable gradient estimates and improved training consistency without exceeding memory constraints. During both training and validation, the dataset $\mathcal{D}_*$ (where $*\in \{tr, te\}$) is processed in batches indexed by $j = 1,\dots,\left\lceil \frac{n_*}{B} \right\rceil$. The index set corresponding to the j-th batch is defined as $\mathcal{I}_j = \left\{(j-1)B+1,\dots,\min(jB, n_*)\right\}$, which ensures that all samples are covered, including the final batch that may contain fewer than B samples. In this formulation, $\hat{y}^{(i)}_{*,k,k'}$ represent the (k,k') element of the matrix of the predicted output adjacency matrix $\hat{\mathbf{Y}}^{(i)}_*$, for $i \in \mathcal{I}_j$ and $y^{(i)}_{*,k,k'}$ is its corresponding ground truth output adjacency matrix $\mathbf{Y}^{(i)}_*$. The $*$ is tr in the loss calculation during training and te in the validation stage of an epoch. The batch loss is defined as $\mathcal{L}_{\text{batch}}\left(\{\hat{\mathbf{Y}}^{(i)}_*\}_{i\in\mathcal{I}_j}, \{\mathbf{Y}^{(i)}_*\}_{i\in\mathcal{I}_j}\right)$, and computed using AMSE as

$$\mathcal{L}_{\text{batch}}(\hat{\mathbf{Y}}_*, \mathbf{Y}_*) = \frac{1}{N^2} \sum_{i\in\mathcal{I}_j} \sum_{k=1}^{N} \sum_{k'=1}^{N} \begin{cases} w\left(\hat{y}^{(i)}_{*,k,k'} - y^{(i)}_{*,k,k'}\right)^2, & \text{if } \hat{y}^{(i)}_{*,k,k'} < y^{(i)}_{*,k,k'}, \\ \left(\hat{y}^{(i)}_{*,k,k'} - y^{(i)}_{*,k,k'}\right)^2, & \text{otherwise.} \end{cases} \tag{12}$$

Model training was performed using the Adam optimizer [81]. The learning rate η is a critical hyperparameter that influences training stability, prevents premature convergence, and facilitates effective exploration of the loss landscape [82]. The learning rates for the models were determined separately through an iterative tuning process to ensure they were sufficiently small to maintain stability while enabling efficient convergence. As a large number of epochs was required to achieve low error values, the use of a constant learning rate was found to be ineffective. Therefore, a piecewise constant learning rate schedule [83] was adopted, allowing controlled reductions in the learning rate at predefined stages of training. The specific learning rate schedules used are provided in Table 4.

With the learning rate η defined, gradients of the batch loss are accumulated over multiple batches to emulate a larger effective batch size. Let $\nabla_\theta \mathcal{L}_{\text{acc}}$ denote the accumulated gradient. For each batch, $\nabla_\theta \mathcal{L}_{\text{acc}} \leftarrow \nabla_\theta \mathcal{L}_{\text{acc}} + \nabla_\theta \mathcal{L}_{\text{batch}}$.

When the prescribed number of accumulation steps A is reached, or when all batches in the epoch have been processed, the model parameters are updated. Let n_b denote the total number of samples contributing to the accumulated gradients. The parameter update is then given by $\theta \leftarrow \theta - \eta \frac{\nabla_\theta \mathcal{L}_{\text{acc}}}{n_b}$.

After the update, the accumulated gradients are reset to zero before continuing the training process.

Table 4
Learning rate (LR) schedules for different GNN models.

Model (s)	Learning rate (η) schedule
GENConv_A1_D2 GENConv_A2_D2 GENConv_A3_D2 GENConv_A2_D1 GENConv_A2_D3	Epochs 0–499: 0.01 Epochs 500–999: 0.005 Epochs 1000–1299: 0.001 Epochs 1300–1699: 0.0005 Epochs 1700–1999: 0.0001
GENConv_A1_D2	Epochs 0–499: 0.01 Epochs 500–999: 0.005 Epochs 1000–1299: 0.001 Epochs 1300–1699: 0.0001 Epochs 1700–1999: 0.00005
TAGConv_A2_D2 EGConv_A2_D2 CLUSTERConv_A2_D2 ARMAConv_A2_D2 LEConv_A2_D2 GCNConv_A2_D2	Epochs 0–1999: 0.01 Epochs 2000–3999: 0.001
GATConv_A2_D2	Epochs 0–1999: 0.001 Epochs 2000–3999: 0.0001

All GNN computations were performed on CUDA-enabled GPUs, which provide significant acceleration for matrix operations and graph convolutions through parallel processing [84].

4. Results and discussion

4.1. Model training and validation

The loss values presented in Figs. 3–5 correspond to the AMSE defined in Eq. (12), evaluated during training for both the training and validation datasets. All models exhibit comparable training and validation losses throughout the training process, indicating that they do not suffer from significant overfitting or underfitting.

4.2. Model performance evaluation

Visualizing the loss curves across training epochs enables a comparative evaluation of different model configurations, including variations in convolution type, architectural complexity, and dataset size. Among the convolution types considered (Fig. 3), GENConv demonstrates the strongest performance, achieving loss values on the order of tens. In contrast, the remaining convolution types exhibit substantially higher errors, often in the range of thousands or greater. Based on the final validation loss values, the models can be ranked in increasing order of performance as follows: GCNConv, GATConv, LEConv, ARMAConv, CLUSTERConv, EGConv, TAGConv, and GENConv as the best-performing architecture. These results indicate that GENConv has a superior ability to capture the underlying structural optimization patterns present in the dataset.

Focusing on the influence of architectural complexity and dataset size on the performance of the GENConv model, Fig. 4 presents the training and validation loss curves for the A2 architecture trained on datasets of varying sizes. The results indicate that increasing the dataset size improves model generalization. However, beyond a certain threshold, the improvement becomes less significant, suggesting a saturation effect. The impact of architectural complexity is illustrated in Fig. 5, which compares three architectures (A1, A2, and A3) trained on a consistent dataset of 12,000 samples. As expected, increasing model complexity leads to lower training and validation losses, demonstrating the benefits of a higher number of hidden channels.

Algorithm 2 Training and validation of the GNN.

Input: preprocessed training and test datasets $\mathcal{D}_{tr}, \mathcal{D}_{te}$
Hyperparameters: Θ, B, A, e
Initialize model parameters θ, optimizer, η scheduler
for epoch = 1 to e **do**
  Set training mode; initialize $\mathcal{L}_{tr} = 0$ and set $p = 0$
  for each training batch $\mathcal{B}_j \subset \mathcal{D}_{tr}$ **do**
    Predict batch outputs: $\hat{\mathbf{Y}}_{tr} \leftarrow f_\theta(\mathbf{X}_{tr}, \mathbf{E}_{tr})$
    Symmetrize and mask: $\hat{\mathbf{Y}}_{tr} \leftarrow \frac{1}{2}\left(\hat{\mathbf{Y}}_{tr} + \hat{\mathbf{Y}}_{tr}^{\top}\right) \odot \mathbf{M}_{\text{zero diag}}$
    Compute $\mathcal{L}_{\text{batch}}$ and accumulate gradients
    $p \leftarrow p + 1$
    if $p = A$ or last batch **then**
      Update θ using accumulated gradients; clear gradients
      set $p \leftarrow 0$
    end if
    $\mathcal{L}_{tr} \leftarrow \mathcal{L}_{tr} + \mathcal{L}_{\text{batch}}$
  end for
  Normalize and store $\mathcal{L}_{tr}$
  Set evaluation mode; initialize $\mathcal{L}_{val} = 0$
  for each validation batch $\mathcal{B}_j \subset \mathcal{D}_{te}$ **do**
    Predict, symmetrize, and mask batch outputs
    Compute validation batch loss $\mathcal{L}_{\text{batch}}$
    $\mathcal{L}_{val} \leftarrow \mathcal{L}_{val} + \mathcal{L}_{\text{batch}}$
  end for
  Normalize and store $\mathcal{L}_{val}$
  Step scheduler
end for

Although AMSE plots provide a comparative assessment of model performance, additional evaluation metrics are required to comprehensively assess overall predictive accuracy as well as the effectiveness of the model in topology and size optimization separately. Accordingly, a set of evaluation metrics is selected to capture both the overall prediction quality and the distinct performance in topology and size prediction.

To assess overall performance, we use the Normalized Root Mean Squared Error (NRMSE) and the coefficient of determination (R^2). These metrics compare the predicted structural optimization outputs with ground truth results and are widely used in related studies. For a sample with N_{obs} observations, ground truth outputs $Y_{\text{obs},i}$, and corresponding predictions $Y_{\text{pred},i}$, the NRMSE and R^2 are computed as:

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{N_{\text{obs}}}\sum_{i=1}^{N_{\text{obs}}}\left(Y_{\text{obs},i} - Y_{\text{pred},i}\right)^2}}{\overline{Y}_{\text{obs}}} \tag{13}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N_{\text{obs}}}\left(Y_{\text{obs},i} - Y_{\text{pred},i}\right)^2}{\sum_{i=1}^{N_{\text{obs}}}\left(Y_{\text{obs},i} - \overline{Y}_{\text{obs}}\right)^2} \tag{14}$$

For topology optimization, we employ classification metrics: recall, precision, F1 score, and Jaccard similarity index. These evaluate the overlap between predicted and actual structural members:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

$$\text{Jaccard Similarity} = \frac{TP}{TP + FP + FN} \tag{18}$$

Here, TP (true positives), FP (false positives), and FN (false negatives) refer to structural members predicted in the topology relative to the ground truth. All these metrics range from $(-\infty, 1)$, with 1 indicating a perfect prediction.

To assess size optimization accuracy (i.e., the predicted cross-sectional areas) we use the mean absolute percentage error (MAPE) and symmetric mean absolute percentage error (SMAPE). MAPE only considers deviations in ground truth members, while SMAPE accounts for both ground truth and predicted members:

$$\text{MAPE} = \frac{1}{|E_{\text{gt}}|}\sum_{i=1}^{|E_{\text{gt}}|}\left|\frac{A_i - P_i}{A_i}\right| \times 100 \tag{19}$$

$$\text{SMAPE} = \frac{1}{|\mathbf{E}_{\text{gt}} \cup \mathbf{E}_{\text{pred}}|}\sum_{i=1}^{|\mathbf{E}_{\text{gt}} \cup \mathbf{E}_{\text{pred}}|}\frac{|P_i - A_i|}{|P_i| + |A_i|} \times 100 \tag{20}$$

Here, A_i and P_i are the actual and predicted cross-sectional areas for edge i, and $\mathbf{E}_{\text{gt}}$, $\mathbf{E}_{\text{pred}}$ denote the sets of edges in the ground truth and predicted structures, respectively.

The test data points are used to quantify the performance of the model. The evaluation metrics introduced are applied to the test data after the post-processing stage. Since min-max scaling was used on the outputs, a threshold value of 300 was selected, based on a minimum scaling reference of 500, as described in the preprocessing steps in Section 3.4. This thresholding sets all predicted cross-sectional area values below 300 to zero, while the remaining values are left unchanged. Following this step, the outputs are scaled back to their original range which is the inverse of what was done in Eq. (11). Table 5 presents the mean values of the evaluation metrics across all models and all test

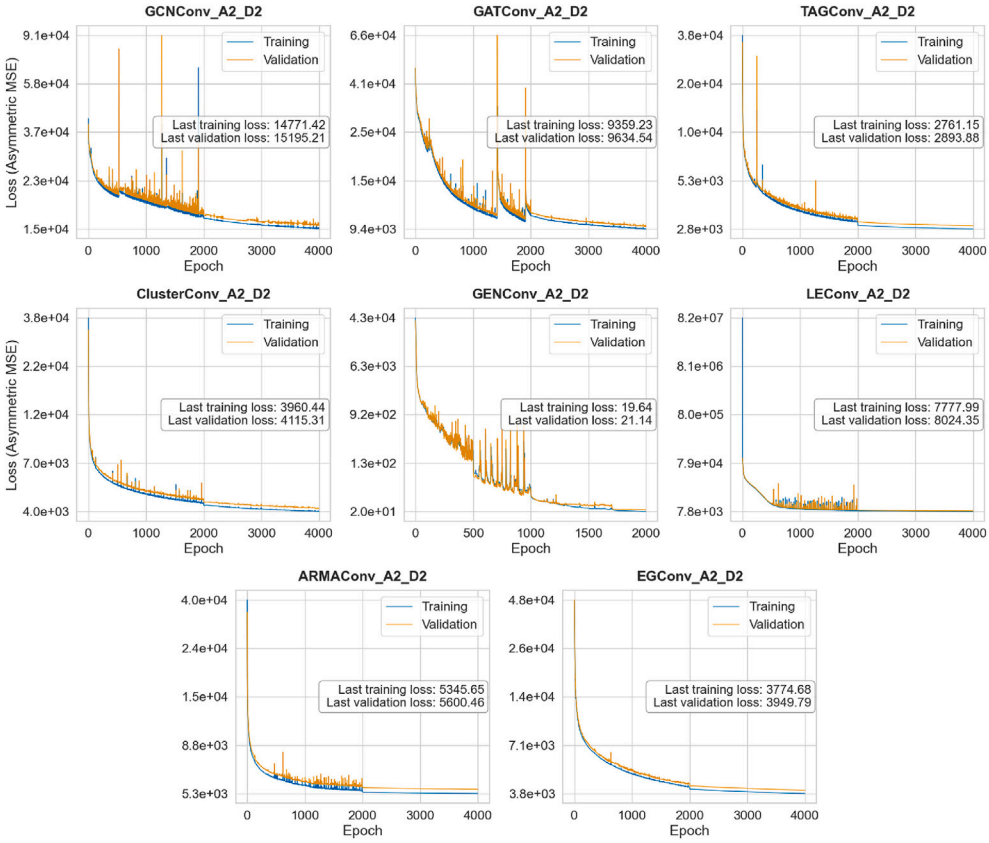


Fig. 3. Training and validation loss curves during the training process of different convolution methods.

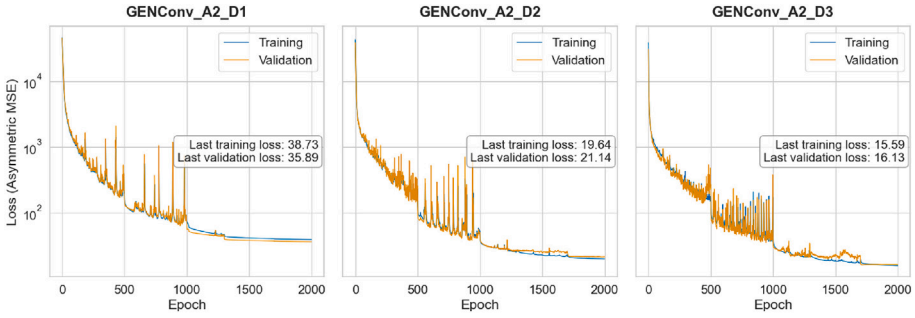


Fig. 4. Training and validation loss curves during the training process of the A2 architecture models.

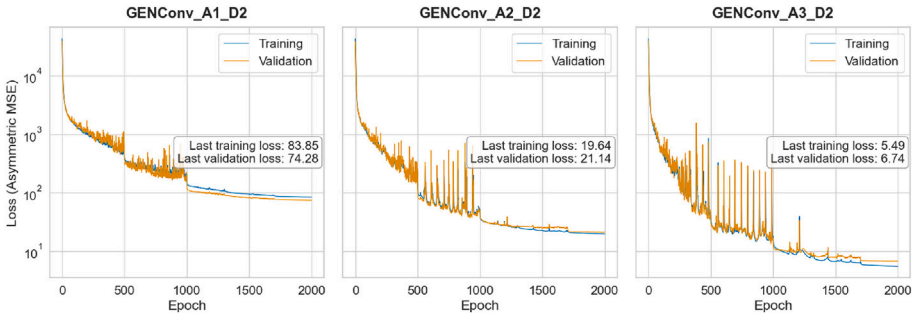


Fig. 5. Training and validation loss curves during the training process of the models using the 12,000 data points sample.

Table 5
Mean of the evaluation metrics across all test data points.

	NRMSE	R^2	Precision	Recall	F1 Score	J. Similarity	MAPE	SMAPE
GCNConv_A2_D2	0.039	0.599	0.6774	0.7338	0.7044	0.5437	43.81	106.23
GATConv_A2_D2	0.034	0.695	0.7389	0.8075	0.7717	0.6283	34.73	87.61
TAGConv_A2_D2	0.017	0.926	0.9148	0.9645	0.9390	0.8849	12.81	32.16
ClusterConv_A2_D2	0.021	0.882	0.8754	0.9427	0.9078	0.8312	16.80	44.29
LEConv_A2_D2	0.031	0.752	0.7859	0.8598	0.8212	0.6966	29.93	74.68
ARMAConv_A2_D2	0.022	0.870	0.9683	0.8573	0.9094	0.8339	21.08	39.80
EGConv_A2_D2	0.021	0.881	0.8811	0.9371	0.9082	0.8319	17.65	44.35
GENConv_A2_D1	0.002	0.9994	0.99999	0.99993	0.99996	0.9999	1.80	1.79
GENConv_A2_D2	0.002	0.9993	0.99980	0.99985	0.99983	0.9997	1.70	1.74
GENConv_A2_D3	0.002	0.9994	0.99984	0.99989	0.99986	0.9997	1.59	1.62
GENConv_A1_D2	0.003	0.9976	0.99919	0.99963	0.99941	0.9988	3.38	3.53
GENConv_A3_D2	0.001	0.9998	0.99996	0.99994	0.99995	0.9999	0.79	0.80

data points. These results provide a clearer understanding of the overall model performance, particularly in its ability to perform both topology optimization and size optimization. This level of insight was not fully apparent from the training and validation curves shown in Section 4.1.

Among the architectures, the models using GENConv convolutions significantly outperform all other models. GENConv_A3_D2 achieves the best results across nearly all metrics. It reports the lowest mean NRMSE (0.001), the highest mean R^2 (0.9998), and F1 Score mean (0.99995), along with the lowest mean MAPE (0.79%) and mean SMAPE (0.80). These results demonstrate improved performance compared to the optimization approach of Zheng et al. [24], where the reported R^2 values for the predicted outputs are approximately 0.8. Similarly, the method in Ariyasinghe et al. [41] reports mean MAPE and SMAPE values of around 5% and 10%, respectively. In terms of topology prediction, a high Jaccard similarity of 0.9999 is achieved, compared to 0.962 reported in Ariyasinghe et al. [41] though a larger dataset of 24,000 samples was considered. All variants of models using GENConv score above 0.9997 in Jaccard Similarity and exhibit less than 2% error in MAPE and SMAPE. The minor variations among the GENConv configurations suggest that the architecture is highly stable and performs consistently well across the settings of the problem at hand. Even the lowest-performing GENConv variant, GENConv_A1_D2, surpasses all non-GENConv models. It achieves an F1 Score of 0.99941 and SMAPE of just 3.53.

Among the non-GENConv architectures, TAGConv_A2_D2 emerges as the most successful. It reports an NRMSE of 0.017, an R^2 of 0.926, and an F1 Score of 0.9390. It also achieves the best Jaccard Similarity (0.8849) and the lowest MAPE (12.81%) and SMAPE (32.16) among this group. Compared to GCNConv_A2_D2, TAGConv reduces NRMSE by more than 56%, improves R^2 by 54.5%, and increases the F1 Score by 33.3%. ClusterConv_A2_D2 and EGConv_A2_D2 also perform well, with R^2 scores above 0.88 and F1 Scores above 0.90. ARMAConv_A2_D2 is particularly strong in Precision (0.9683), although it has a relatively lower Recall (0.8573), suggesting a more conservative approach to positive predictions. ClusterConv and EGConv provide a better balance between these metrics. LEConv_A2_D2 shows moderate performance, with an F1 Score of 0.8212 and a SMAPE of 74.68, but falls short of the top-performing models. GCNConv_A2_D2 records the weakest overall performance, with the highest NRMSE (0.039), lowest R^2 (0.599), and the highest error rates in both MAPE (43.81%) and SMAPE (106.23).

When compared to the next best performing model after GENConv variants, which is the TAGConv_A2_D2, GENConv_A3_D2 reduces NRMSE by 94.1% and increases R^2 by approximately 0.37%. Similarly, it improves the F1 Score from 0.9390 to 0.99995, representing a relative increase of 6.5%. It also reduces MAPE by 93.8% and SMAPE by 97.5%.

An intuitive explanation for the superior performance of GENConv is that optimized truss prediction is governed by global load transfer and equilibrium rather than purely local geometric relationships. In the present problem, the input graph encodes nodal coordinates, external loads, and support conditions, while the target is a sparse weighted adjacency matrix representing both member existence and cross-sectional area. Therefore, the model must identify which candidate members participate in the dominant load paths and which should be suppressed.

Compared with simpler operators, GENConv provides a more expressive message-passing mechanism through learnable message construction, flexible aggregation, and residual/normalization-based feature updates. This makes it better suited to capturing nonlinear interactions between geometry, support conditions, and loading, while preserving sharp distinctions between active and inactive members. The fact that TAGConv, which also incorporates multi-hop neighborhood information, is the strongest non-GENConv baseline further supports the view that successful prediction of optimized truss configurations requires aggregation beyond immediate neighbors. In this context, GENConv appears particularly effective because it combines broader structural context with adaptive aggregation, leading to more accurate identification of load paths and member sizes.

4.3. Best model case-by-case predictions

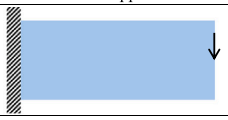
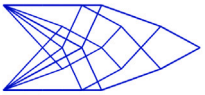
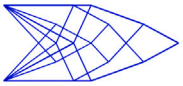
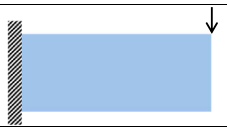
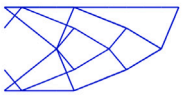
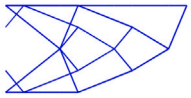
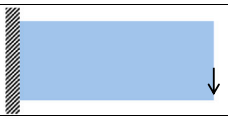
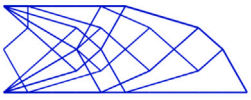
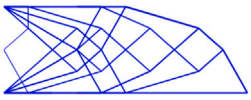
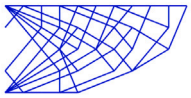
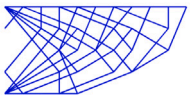
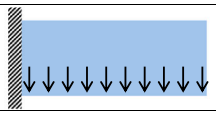
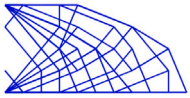
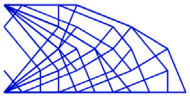
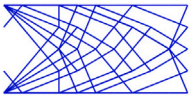
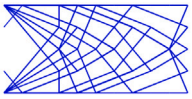
To demonstrate the ability of the model to predict optimized structures, the test results for the GENConv_A3_D2 configuration have been selected. The dataset included 12 different scenarios that varied by combinations of loading conditions, support configurations, design domain sizes, and load magnitudes, as described in Section 3.4.

Based on the mean values of various evaluation metrics (see Table 6), the model showed near-perfect prediction performance across nearly all scenarios. Precision, recall, F1 score, and Jaccard similarity values were consistently equal to 1.0000 in 10 out of the 12 cases, indicating complete agreement between predicted and reference topologies. The corresponding NRMSE values remained extremely low, and R^2 values approached 1.0000, reflecting excellent accuracy. Slight deviations were observed in cases 8 and 12, where a few test samples resulted in values marginally below 1.0000 for recall, F1 score, and Jaccard similarity. In

Table 6
Mean of the evaluation metrics across cases for all test data points.

Case	NRMSE	R^2	Precision	Recall	F1 Score	J. Similarity	MAPE	SMAPE
1	0.0008	0.9999	1.0000	1.0000	1.0000	1.0000	0.87	0.86
2	0.0009	0.9999	1.0000	1.0000	1.0000	1.0000	0.73	0.74
3	0.0009	0.9999	1.0000	1.0000	1.0000	1.0000	0.75	0.75
4	0.0011	0.9999	1.0000	1.0000	1.0000	1.0000	0.65	0.65
5	0.0011	0.9999	1.0000	1.0000	1.0000	1.0000	0.64	0.64
6	0.0011	0.9999	1.0000	1.0000	1.0000	1.0000	0.81	0.81
7	0.0011	0.9999	1.0000	1.0000	1.0000	1.0000	0.88	0.88
8	0.0022	0.9995	1.0000	0.9996	0.9998	0.9996	0.86	0.90
9	0.0007	0.9999	1.0000	1.0000	1.0000	1.0000	0.74	0.73
10	0.0007	0.9999	1.0000	1.0000	1.0000	1.0000	0.79	0.79
11	0.0009	0.9999	1.0000	1.0000	1.0000	1.0000	0.89	0.88
12	0.0015	0.9995	0.9995	0.9996	0.9996	0.9992	0.92	1.04

Table 7
Cantilever trusses: Highest SMAPE (worst) predictions.

Case	Load and support cond.	Ground truth	Predicted
1			
	NRMSE: 0.0040 R^2: 0.9992 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.12% SMAPE: 2.10%		
2			
	NRMSE: 0.0054 R^2: 0.9983 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.81% SMAPE: 2.84%		
3			
	NRMSE: 0.0097 R^2: 0.9952 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.76% SMAPE: 3.12%		
4			
	NRMSE: 0.0062 R^2: 0.9990 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.22% SMAPE: 2.26%		
5			
	NRMSE: 0.0023 R^2: 0.9998 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 1.20% SMAPE: 1.20%		
6			
	NRMSE: 0.0080 R^2: 0.9982 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.72% SMAPE: 2.76%		

case 8, while precision remained perfect, the recall value dropped to 0.9996, leading to a minor decrease in F1 score and Jaccard similarity. Case 12 showed a similar trend, with a precision of 0.9995 and a Jaccard similarity of 0.9992.

Table 7 presents the test samples from the cantilever cases that recorded the highest SMAPE values. Although these samples represent the worst-performing predictions within this specific category, the evaluation metrics indicate that the model has still maintained a high level of accuracy. Across all six samples, the values for precision, recall, F1 score, and Jaccard similarity are all equal to 1.0. The R^2 values remain very close to 1.0, ranging from 0.9952 to 0.9998, suggesting excellent agreement between predicted and ground truth.

NRMSE values are also low, with the highest being 0.0097, which reflects only a minor deviation in the overall predictions. Although SMAPE values for these samples range from 1.20% to 3.12%, the errors are minimal and well within acceptable limits for practical applications. The highest SMAPE value of 3.12% appears in a case with an R^2 of 0.9952 and perfect classification metrics, which implies that even in less ideal conditions, the predicted structures are highly accurate.

Table 8 displays the test samples with the highest SMAPE values for the simply supported cases. These represent the least accurate predictions within this category. Among the six samples listed, four exhibit perfect values for precision, recall, F1 score, and Jaccard similarity. Their R^2 values are all above 0.998, and their SMAPE values remain low, ranging from 1.77% to 2.36%. These results suggest that in most cases, the predicted topologies closely match the ground truth structures.

Two specific samples, however, stand out with relatively higher SMAPE values. Case 8 shows a SMAPE of 14.86% and a corresponding MAPE of 8.52%, indicating a more noticeable deviation between prediction and ground truth. Although the precision is still 1.0, the recall drops to 0.938, which leads to an F1 score of 0.968 and a Jaccard similarity of 0.938. The R^2 value for this case is 0.9368, which, while still high, is lower than the others. Case 12 also shows a similar trend, with a SMAPE of 14.39% and slightly reduced classification metrics. The precision in this case is 0.949, and the Jaccard similarity is 0.933. Despite these two outliers, the remaining samples show excellent agreement with the ground truth. Since these represent the worst-performing cases, the remaining predictions in the simply supported category demonstrate even closer alignment with the ground truth structures.

4.4. Computational cost savings

Table 9 presents a comparison of the computational efficiency of the proposed GNN models relative to the ground truth optimization framework developed by Zegard and Paulino [1]. The average runtime of

Table 8
Simply supported trusses: Highest SMAPE (worst) predictions.

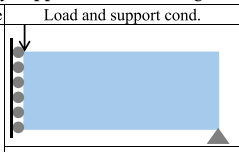
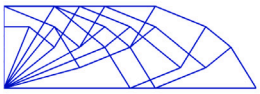
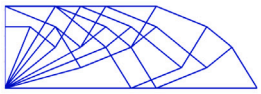
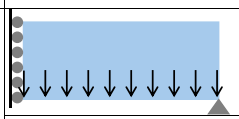
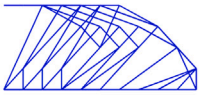
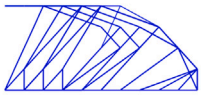
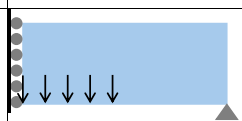
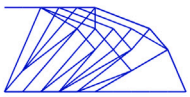
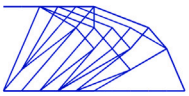
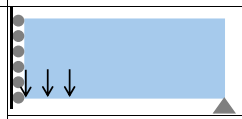
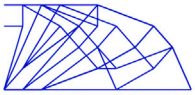
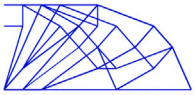
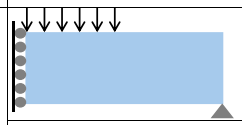
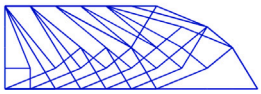
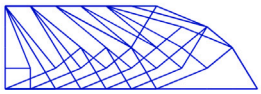
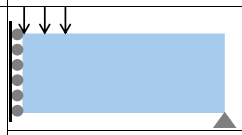
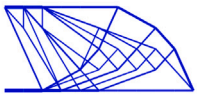
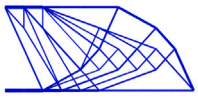
Case	Load and support cond.	Ground truth	Predicted
7			
	NRMSE: 0.0045 R^2: 0.9993 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 1.78% SMAPE: 1.77%		
8			
	NRMSE: 0.0456 R^2: 0.9368 Precision: 1.0 Recall: 0.938 F1 score: 0.968 J.Similarity: 0.938 MAPE: 8.52% SMAPE: 14.86%		
9			
	NRMSE: 0.0049 R^2: 0.9991 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 1.91% SMAPE: 1.93%		
10			
	NRMSE: 0.0072 R^2: 0.9980 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.27% SMAPE: 2.36%		
11			
	NRMSE: 0.0069 R^2: 0.9984 Precision: 1.0 Recall: 1.0 F1 score: 1.0 J.Similarity: 1.0 MAPE: 2.16% SMAPE: 2.23%		
12			
	NRMSE: 0.0149 R^2: 0.9762 Precision: 0.949 Recall: 0.982 F1 score: 0.967 J.Similarity: 0.933 MAPE: 2.87% SMAPE: 14.39%		

Table 9
Comparison of inference time and speed-up of GNN models relative to the ground truth optimization framework.

Model	Avg Time (ms)	Speedup	Model	Avg Time (ms)	Speedup
GCNConv_A2_D2	1.11	44.4	ARMAConv_A2_D2	1.02	48.3
GATConv_A2_D2	2.75	17.9	EGConv_A2_D2	0.75	65.7
TAGConv_A2_D2	2.16	22.8	GCNConv_A1_D2	1.63	30.2
ClusterConv_A2_D2	1.03	47.8	GCNConv_A2_D2	1.87	26.4
LEConv_A2_D2	1.28	38.5	GCNConv_A3_D2	2.35	21.0

the ground truth framework was measured as 49.28 ms on a system equipped with an AMD Ryzen 7 5800H processor (8 cores, 16 threads, base clock 3.2 GHz) and 16 GB of RAM. The GNN models were executed on the same system using GPU-accelerated computations with an NVIDIA GeForce GTX 1650 featuring 4 GB of dedicated VRAM, enabled via CUDA-supported drivers. The speedup factor is defined as the ratio of the runtime of the ground truth framework to that of the GNN model. As shown in Table 9, all GNN models achieve substantial reductions in computational time. Among them, the EGConv_A2_D2 model achieves the highest speed-up of 65.7. The GCNConv_A3_D2 model, which demonstrated the best predictive performance, achieves a speed-up factor of 21.0.

5. Conclusions

This study introduced a Graph Neural Network (GNN)-based surrogate approach for real-time structural optimization of truss systems, addressing both topology and size optimizations. A comprehensive set of evaluation metrics was used to assess performance, including regression-based, classification-based, and percentage error-based indicators. Among various convolution types tested, Generalised Graph Convolution (GENConv) consistently delivered the highest accuracy across all metrics. The GENConv_A3_D2 configuration achieved the lowest NRMSE and percentage errors, and near-perfect scores in precision, recall, F1 score, and Jaccard similarity, confirming its effectiveness in

predicting both the optimal member layout and their cross-sectional areas. This was followed by the Topology Adaptive Graph Convolution (TAGConv). This shows that prediction capability in the optimization tasks is enhanced by using a more expressive message-passing mechanism through learnable message construction, flexible aggregation, and residual/normalization-based feature updates which GENConv and TAGConv utilize in the convolutions.

Further analysis showed that increasing architectural complexity and dataset size leads to improvements in training and validation performance, although returns begin to diminish beyond a certain point. Evaluations across twelve different loading and support scenarios demonstrated consistent accuracy, with only minor deviations observed in a few cases. Even the worst predictions maintained close agreement with the ground truth, and most test cases achieved perfect classification scores. The results highlight the potential of this GNN-based approach to replace conventional optimization routines, offering a fast, scalable, and accurate alternative. Though the predictions provide ideal topological and low error size predictions for most cases, there could be some predictions where the recall is not ideal, and this would violate equilibrium, which is an issue that persists in ML surrogates where predictions are not always accurate. However, the better the results in model evaluations, the lower the probability that this occurs. In this context, a post-processing filtering step could be added similar to steps used in Refs. [85,86], which would lead to a fallback to the physics-based ground truth optimization.

Regarding future adaptations of this framework, different types of optimization problems with additional constraints, such as buckling [5,87], reliability [88], and natural frequency constraints [89], are outlined. In addition, the current framework could be extended to 3D systems, where the main difference would lie in the graph representation of the truss structure (Section 3.2). In such cases, the feature vector would be adapted to the new coordinate system, resulting in $\mathbf{X} \in \mathbb{R}^{N\times 9}$, along with additional nodes to ensure sufficient coverage of the design domain.

CRediT authorship contribution statement

Nisal Ariyasinghe: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Tharindu Wickremasinghe:** Writing – original draft, Visualization, Methodology, Investigation, Formal analysis. **Hansani Weeratunge:** Writing – review & editing, Supervision, Project administration, Methodology, Conceptualization. **Chinthaka Mallikarachchi:** Writing – review & editing, Supervision, Software, Resources, Project administration, Funding acquisition. **Sumudu Herath:** Writing – review & editing, Supervision, Software, Project administration, Methodology, Formal analysis, Conceptualization.

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.

References

[1] Zegard T, Paulino GH. Grand — ground structure based topology optimization for arbitrary 2D domains using matlab. Struct Multidiscip Optim Nov 2014;50(5):861–82.

[2] Li Z, Wang L, Chai Y, Zhang L, Liu Y. Topology design of multi-scale thermo-elastic structures considering performance reduction and disturbance based on interval models. Appl Math Model 2025;146:116167.

[3] Li Z, Wang L, Gu K. Efficient reliability-based concurrent topology optimization method under pid-driven sequential decoupling framework. Thin-walled Struct 2024;203:112117.

[4] Zhao X, Wang L. Double-scale time-dependent reliable topology optimization based on the first-passage failure and interval process theories. Comput Methods Appl Mech Eng 2025;443:118088.

[5] Bugarin F, Gogu C, Mitjana F, Cafieri S, Castanie F. Optimization of structures under buckling constraints using frame elements. Eng Optim 2019;51(1):140–59.

[6] Li Q, Wu Q, Liu J, He J, Liu S. Topology optimization of vibrating structures with frequency band constraints. Struct Multidiscip Optim 2021;63(3):1203–18.

[7] Saeed N, Jankowski R, Manguri A, Hassan H. Topology, size, and shape optimization in civil engineering structures: a review. Comput Model Eng Sci 2025;142(2):933–71.

[8] Peng L, Shen X, Dong S, Fang Y. Topology optimization methods and its applications in aerospace: a review. Struct Multidiscip Optim 2025;68(5):105.

[9] Su H, An D, Ma L, He Y. A survey of multi-scale optimization methods in fiber-reinforced polymer composites design for automobile applications. Proc Inst Mech Eng D J Automob Eng 2026;240(2–3):805–21.

[10] Islam MS, Nayem AZ, Hoque KN. Finite element-based optimization procedure for an irregular domain with unstructured mesh. Heliyon Feb 2024;10(4):e25994. https://doi.org/10.1016/j.heliyon.2024.e25994

[11] Gebremedhen HS, Woldemicahel DE, Hashim FM. Three-dimensional stress-based topology optimization using SIMP method. Int J Simul Multidisci Des Optim 2019;10:A1.

[12] Cui N, Huang S, Ding X. An improved strategy for genetic evolutionary structural optimization. Adv Civ Eng 2020;2020(1):5924198.

[13] Xu T, Lin X, Xie YM. Bi-directional evolutionary structural optimization with buckling constraints. Struct Multidiscip Optim Mar 2023;66(4):67.

[14] Ansola Loyola R, Querin OM, Garaigordobil Jiménez A, Alonso Gordoa C. A sequential element rejection and admission (sera) topology optimization code written in matlab. Struct Multidiscip Optim 2018;58(3):1297–310.

[15] Cai S, Zhang W. An adaptive bubble method for structural shape and topology optimization. Comput Methods Appl Mech Eng 2020;360:112778.

[16] Zhang W, Li D, Yuan J, Song J, Guo X. A new three-dimensional topology optimization method based on moving morphable components (MMCS). Comput Mech 2017;59(4):647–65.

[17] Zhang W, Li D, Kang P, Guo X, Youn S-K. Explicit topology optimization using iga-based moving morphable void (MMV) approach. Comput Methods Appl Mech Eng 2020;360:112685.

[18] Chen Z, Cao H, Qian X, Zhu H. Enhanced particle swarm optimization for size and shape optimization of truss structures. Eng Optim 2017;49(11):1939–56.

[19] Gholizadeh S, Seyedpoor SM, Talebian SR. An efficient structural optimisation algorithm using a hybrid version of particle swarm optimisation with simultaneous perturbation stochastic approximation. Civ Eng Environ Syst 2010;27(4):295–313.

[20] Chen S-Y, Shui X-F, Li D-F, Huang H. Improved genetic algorithm with two-level approximation for truss optimization by using discrete shape variables. Math Probl Eng 2015;2015(1):521482.

[21] Ingrid Delyová, Peter Frankovský, Jozef Bocko, Peter Trebuňa, Jozef Živčák, Barbara Schürger, Sára Janigová. Sizing and topology optimization of trusses using genetic algorithm. Materials, 14(4), 2021.

[22] Du W-F, Wang Y-Q, Wang H, Zhao Y-N. Intelligent generation method for innovative structures of the main truss in a steel bridge. Soft Comput 2023;27(9):5587–601.

[23] Seo J, Kapania RK. Development of deep convolutional neural network for structural topology optimization. AIAA J 2023;61(3):1366–79.

[24] Zheng S, Qiu L, Lan F. TSO-GCN: a graph convolutional network approach for real-time and generalizable truss structural optimization. Appl Soft Comput 2023;134:110015.

[25] Jayaweera N, Prabasha K, Fernando C, Herath S, Tsavdaridis KD, Meddage DPP. Material-efficient 2D skeletal structure design using convolutional neural networks. Mater Today Commun 2026;50:114400.

[26] Wang Y, Shi F, Chen B. Real-time structure generation based on data-driven using machine learning. Processes 2023;11(3).

[27] Seo M, Min S. Graph neural networks and implicit neural representation for near-optimal topology prediction over irregular design domains. Eng Appl Artif Intell 2023;123:106284.

[28] Jeong H, Bai J, Batuwatta-Gamage CP, Rathnayaka C, Zhou Y, Gu Y. A physics-informed neural network-based topology optimization (pinnto) framework for structural optimization. Eng Struct 2023;278:115484.

[29] Li Z, Wang L, Li F. Model-and-data-driven robust concurrent optimization for structural topology and device layout integrating load location and non-design domain effects. Aerosp Sci Technol 2026;170:111566.

[30] Mai HT, Lee S, Kim D, Lee J, Kang J, Lee J. Optimum design of nonlinear structures via deep neural network-based parameterization framework. Eur J Mech A-Solid 2023;98:104869.

[31] Mai HT, Mai DD, Kang J, Lee J, Lee J. Physics-informed neural energy-force network: a unified solver-free numerical simulation for structural optimization. Eng Comput 2024;40(1):147–70.

[32] Khodadadi N, Talatahari S, Gandomi AH. Anna: advanced neural network algorithm for optimisation of structures. Proc Inst Civ Eng Struct Build 2024;177(6):529–51.

[33] Fu B, Gao Y, Wang W. A physics-informed deep reinforcement learning framework for autonomous steel frame structure design. Comput-aided Civ Infrastruct Eng 2024;39(20):3125–44.

[34] Mai HT, Kang J, Lee J. A machine learning-based surrogate model for optimization of truss structures with geometrically nonlinear behavior. Finite Elem Anal Des 2021;196:103572.

[35] Liu J, Xia Y. A hybrid intelligent genetic algorithm for truss optimization based on deep neutral network. Swarm Evol Comput 2022;73:101120.

[36] Pham H-A, Dang V-H, Vu T-C, Nguyen B-D. An efficient K-NN-based rao optimization method for optimal discrete sizing of truss structures. Appl Soft Comput 2024;154:111373.

[37] Nourian N, El-Badry M, Jamshidi M. Design optimization of truss structures using a graph neural network-based surrogate model. Algorithms 2023;16(8).

[38] Nguyen T-H, Vu A-T. An efficient differential evolution for truss sizing optimization using adaboost classifier. Comput Model Eng Sci 2022;134(1):429–58.
[39] Jayaweera N, Abeyrathna A, Herath S, Ekanayake IU, Meddage DPP. Code-compliant optimal design of reinforced concrete slab beam systems for low to mid-rise buildings using heterogeneous graph neural networks with attention layers. J Build Eng 2026;117:114599.
[40] Zhu S, Ohsaki M, Hayashi K, Guo X. Machine-specified ground structures for topology optimization of binary trusses using graph embedding policy network. Adv Eng Softw 2021;159:103032.
[41] Ariyasinghe N, Weeratunga H, Mallikarachchi C, Herath S. Graph neural network based surrogate model for design informed structural optimization. In: Proceedings of the 15th international conference on sustainable built environment. Singapore: Springer Nature Singapore; 2025. p. 197–211.
[42] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2009;20(1):61–80.
[43] Wu L, Cui P, Pei J, Zhao L, Song L. Graph neural networks. Singapore: Springer Nature Singapore; 2022. p. 27–37.
[44] Fey M, Lenssen JE. Fast graph representation learning with PyTorch Geometric, arXiv:1903.02428, 2019.
[45] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks, 2017.
[46] Ding T, Xiang D, Schubert KE, Dong L. GKAN: explainable diagnosis of alzheimer's disease using graph neural network with kolmogorov-arnold networks, 2025.
[47] Agnelli F, Ghezzi O, Blandano G, Burger J, Facchi G, Schmid L. Enhancing 3D face analysis using graph convolutional networks with kernel-attentive filters. In: Proceedings of the 40th ACM/SIGAPP symposium on applied computing, SAC '25. New York, NY, USA: Association for Computing Machinery; 2025. p. 638–44.
[48] Rashid A, Kumar S, Saxena R. TOXNET: a multi-layer network approach to toxicity analysis using graph neural networks. In: Patel K, Santosh KC, Gomes de Oliveira G, Patel A, Ghosh A, editors. Soft computing and its engineering applications. Cham: Springer Nature Switzerland; 2025. p. 295–307.
[49] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks, 2018.
[50] Guo J, Azcona E, Lopez-Tapia S, Katsaggelos A. Stage detection of mild cognitive impairment: region-dependent graph representation learning on brain morphable meshes. In: Oguz I, Noble J, Li X, Styner M, Baumgartner C, Rusu M, Heinmann T, Kontos D, Landman B, Dawant B, editors. Medical imaging with deep learning, volume 227 of proceedings of machine learning research. PMLR; 10–12 Jul 2024. p. 888–904.
[51] Danish MU, Buwaneswaran M, Fonseka T, Grolinger K. Graph attention convolutional u-net: a semantic segmentation model for identifying flooded areas, 2025.
[52] Du J, Zhang S, Wu G, Moura JMF, Kar S. Topology adaptive graph convolutional networks, 2018.
[53] Gu F, Guo F, Yu F, Long X, Chen C, Liu K, Hu X, Shang J, Guo S. Accurate and efficient floor localization with scalable spiking graph neural networks. Satellite Navigation 2024;5(1):6.
[54] Liang S, Chen T, Ma J, Ren S, Lu X, Du W. Identification of mild cognitive impairment using multimodal 3D imaging data and graph convolutional networks. Phys Med Biol Nov 2024;69(23):235002.
[55] Chiang W-L, Liu X, Si S, Li Y, Bengio S, Hsieh C-J. Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, KDD '19. ACM; Jul 2019.
[56] Wu S, Xiong Y, Liang H, Weng C. Clustering with entropy-based recombination for training GCNS on large graphs. In: 2023 IEEE international conference on data mining workshops (ICDMW); 2023. p. 1170–7.
[57] Gurukar S, Venkatakrishnan SB, Ravindran B, Parthasarathy S. Policyclustergcn: identifying efficient clusters for training graph convolutional networks. In: Proceedings of the 2023 IEEE/ACM international conference on advances in social networks analysis and mining, ASONAM '23. New York, NY, USA: Association for Computing Machinery; 2024. p. 245–52.
[58] Li G, Xiong C, Thabet A, Ghanem B. DeeperGCN: all you need to train deeper GCNS, 2020.
[59] Du H, Yao MM-S, Chen L, Chan WP, Feng M. Multi-task graph convolutional neural network for calcification morphology and distribution analysis in mammograms, 2021.
[60] Feng J, Wang Z, Li Y, Ding B, Wei Z, Xu H. MGMAE: molecular representation learning by reconstructing heterogeneous graphs with a high mask ratio. In: Proceedings of the 31st ACM international conference on information & knowledge management, CIKM '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 509–19.
[61] Ranjan E, Sanyal S, Talukdar PP. Asap: adaptive structure aware pooling for learning hierarchical graph representations, 2020.
[62] Langore T, Hsu T-C, Hsieh Y-H, Lin C. LE-DTA: local extrema convolution for drug target affinity prediction. In: ICASSP 2023 - 2023 IEEE international conference on acoustics, speech and signal processing (ICASSP); 2023. p. 1–5.
[63] Tsui L-I, Hsu T-C, Lin C. NG-DTA: drug-target affinity prediction with n-gram molecular graphs. In: 2023 45th annual international conference of the IEEE engineering in medicine & Biology society (EMBC); 2023. p. 1–4.
[64] Bianchi FM, Grattarola D, Livi L, Alippi C. Graph neural networks with convolutional arma filters. IEEE Trans Pattern Anal Mach Intell 2021:1.
[65] Reddy KSG, Saran B, Adityaja AM, Shigwan SJ, Kumar N, Mukherjee S. Unsegarmanet: unsupervised image segmentation using graph neural networks with convolutional arma filters, 2024.
[66] Chattoraj J, Yang F, Lim CW, Gobeawan L, Liu X, Raghavan VSG. Knowledge-driven transfer learning for tree species recognition. In: 2022 17th international conference on control, automation, Robotics and vision (ICARCV); 2022. p. 149–54.
[67] Ding Y, Zhao X, Zhang Z, Cai W, Yang N, Zhan Y. Semi-supervised locality preserving dense graph neural network with arma filters and context-aware learning for hyperspectral image classification. IEEE Trans Geosci Remote Sens 2022;60:1–12.
[68] Tailor SA, Opolka FL, Liò P, Lane ND. Do we need anisotropic graph neural networks? 2022.
[69] Meng X, Li S, Li R, Huang B, Wang X. Nanotope: a graph neural network-based model for nanobody paratope prediction. Biorxiv 2023.
[70] Nguyen DP, Le PT. Leveraging graph neural networks for enhanced prediction of molecular solubility via transfer learning. Journal of Technical Education Science Jun 2024;19(3):57–64.
[71] Uchizawa K, Abe H. Trade-offs between energy and depth of neural networks. Neural Comput Jul 2024;36(8):1541–67.
[72] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR); 2016. p. 770–8.
[73] Agarap AF. Deep learning using rectified linear units (RELU), 2019.
[74] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 1979;21(2):239–45.
[75] Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? 2019.
[76] Chatterjee A, Vallières M, Dohan A, Levesque IR, Ueno Y, Saif S, Reinhold C, Seuntjens J. Creating robust predictive radiomic models for data from independent institutions using normalization. IEEE Trans Radiat Plasma Med Sci 2019;3(2):210–5.
[77] Zhou X, Liu X, Jiang J, Gao X, Ji X. Asymmetric loss functions for learning with noisy labels, 2021.
[78] Rengasamy D, Rothwell B, Figueredo GP. Asymmetric loss functions for deep learning early predictions of remaining useful life in aerospace gas turbine engines. In: 2020 international joint conference on neural networks (IJCNN); 2020. p. 1–7.
[79] Zhu L, Han Y, Xi X, Zhang Z, Liu M, Li L, Tan S, Yan B. Asymmetric loss based on image properties for deep learning-based image restoration. Computers, Materials and Continua 2023;77(3):3367–86.
[80] Toth E. Asymmetric error functions for reducing the underestimation of local scour around bridge piers: application to neural network models. J Hydraul Eng 2015;141(7):04015011.
[81] Kingma DP, Ba J. Adam: a method for stochastic optimization, 2017.
[82] Konar J, Khandelwal P, Tripathi R. Comparison of various learning rate scheduling techniques on convolutional neural network. In: 2020 IEEE international students' conference on electrical, electronics and computer science (SCEECS). IEEE; 2020. p. 1–5.
[83] Smith LN, Topin N. Super-convergence: very fast training of neural networks using large learning rates, 2018.
[84] Kirk D. Nvidia CUDA software and GPU parallel computing architecture. In: Proceedings of the 6th international symposium on memory management, ISMM '07. New York, NY, USA: Association for Computing Machinery; 2007. p. 103–4.
[85] Sarma LS, Mallikarachchi C, Herath S. Design-informed generative modelling of skeletal structures using structural optimization. Computers & Structures 2024;302:107474.
[86] Giacomini M, Huerta A. A surrogate model for topology optimisation of elastic structures via parametric autoencoders. Comput Methods Appl Mech Eng 2026;448:118503.
[87] Varma TV, Sarkar S, Mondal G. Buckling restrained sizing and shape optimization of truss structures. J Struct Eng 2020;146(5):04020048.
[88] Rodrigues da Silva LA, Torii AJ, Beck AT. System-reliability-based sizing and shape optimization of trusses considering millions of failure sequences. Struct Saf 2024;108:102448.
[89] Kaveh A, Ilchi Ghazaan M. Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints. Acta Mech 2017;228(1):307–22.