Contents lists available at ScienceDirect

Materials Today Communications

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

Material-efficient 2D skeletal structure design using convolutional neural networks

Navod Jayaweera[a], Kavindu Prabasha[a], Chamod Fernando[a], Sumudu Herath[a], Konstantinos Daniel Tsavdaridis[b], D.P.P. Meddage[c,d,*]

[a] Department of Civil Engineering, University of Moratuwa, Bandaranayaka Mawatha, Moratuwa 10400, Sri Lanka
[b] Department of Engineering, School of Science & Technology, City St George's, University of London, Northampton Square, London EC1V 0HB, UK
[c] School of Engineering and Information Technology, The University of New South Wales, Canberra, ACT 2600, Australia
[d] Ceylon Institute for Artificial Intelligence Research (CIAIR), Colombo, Sri Lanka

ARTICLE INFO

Keywords:
Convolutional Neural Networks
Data-Driven Optimization
Design-Aware Optimization
Section Identification
Structural Optimization
Structural Design

ABSTRACT

Structural optimization plays a critical role in achieving efficient material utilization and maximizing structural performance. However, conventional approaches often suffer from high computational costs due to their iterative nature. Incorporating code-compliant design criteria enhances structural integrity and practical applicability but further escalates computational demands. This study presents a convolutional neural network-based framework augmented with a member section classification algorithm, achieving over a sixfold improvement in computational efficiency for generating code-compliant skeletal structures. The proposed workflow generates robust, ready-to-assemble frame designs based on user-defined parameters such as boundary conditions, initial domain geometry, and loading scenarios. Unlike most existing approaches that focus solely on material layout, this study introduces a method capable of jointly determining both structural topology and member dimensions, which is essential for creating assembly-ready designs. Generative convolutional neural models achieve over 91 % accuracy in generating optimal frame images. However, by integrating a strategic geometric feature identification algorithm, geometric features are identified with near-perfect accuracy, effectively overcoming the inherent limitations of generator-induced losses. The effectiveness of the approach is demonstrated through multiple case studies, validating the accuracy of predicted member sections under combined axial, bending, and shear loading, in accordance with established design codes. By incorporating a robust geometric feature extraction mechanism, the framework reliably produces designs that pass critical integrity checks and can be applied across multiple engineering domains.

1. Introduction

Structural optimization aims to find the specific spatial distribution of material to achieve minimum weight (material usage), strain energy (compliance), cost, embodied carbon, or other metric, to resist a given loading while satisfying various constraints, such as a limit stress or displacement. Conversely, the aim may be to maximize capacity for a given material volume. Structural optimization takes various forms, including topology, size, layout, and shape optimization [1–3]. Structural optimization bridges the scientific fields of engineering, mathematics, and computer science [4], with applications ranging from aerospace [5] to civil engineering [1,6] and beyond.

Two main optimization techniques can be identified, which result in optimal skeletal structures consisting of 1D linear members: ground structure-based and continuum-based algorithms [7]. In the ground structure approach, the initial domain is discretized using a dense grid of nodes, which are interconnected to a very high degree with potential members. A linear programming problem is then solved to minimize volume under equilibrium and stress/strain constraints [7–12]. In contrast, continuum-based optimization starts from a continuum domain, pixelated (in 2-dimensions) or voxelated (in 3-dimensions), from which material is removed to obtain the optimal skeletal structure [13–16].

Continuum-based optimization typically leads to an optimal topology, i.e., it indicates the nodal positions and the connectivity between these nodes. Further processing is then needed to automatically extract

* Corresponding author at: School of Engineering and Information Technology, The University of New South Wales, Canberra, ACT 2600, Australia.
E-mail address: p.meddage@unsw.edu.au (D.P.P. Meddage).

https://doi.org/10.1016/j.mtcomm.2025.114400
Received 14 September 2025; Received in revised form 15 November 2025; Accepted 26 November 2025
Available online 28 November 2025

the geometry and conduct further size/shape optimization at the member level. Various workflows to achieve this have been proposed byTomšič and Duhovnik [17], Yin et al. [14], Xia et al. [18], and Tsavdaridis et al. [19].

Previous efforts have been made in literature to integrate optimization with structural integrity checks (beyond the basic requirement of limiting the cross-sectional stress to a material-specific value). For instance, Poulsen et al. [20] proposed a method to address local and global buckling in truss optimization. Similarly, member buckling was considered by Madha et al. [21], Cai et al. [22], and Schwarz et al. [23]. Furthermore, Pederson & Nielsen [24] considered eigenfrequencies, displacements, and stress constraints within their optimization framework. Sarma et al. [15] introduced other design checks, particularly the interaction of axial force and bending at the cross-sectional level. He et al. [25] proposed an optimization approach considering discrete section sizes, accounting for single or multiple load cases and unequal limiting strengths in tension and compression. Zegard & Paulino [26] used similar concepts and extended them to 3D trusses [9]. Pastore et al. [27] used a risk-factor approach in the topology optimization of concrete structural elements to ensure the Von Mises stress criterion is met.

In civil and mechanical engineering, machine learning techniques have been successfully applied across various domains [28], [29], [30]. Specifically, their use in structural optimization is particularly promising, as these methods can offer substantial reductions in computational time relative to traditional iterative approaches. Nguyen et al. [31] presented a Feed Forward Neural Network (FFNN) to predict optimal cross-sections of 2D and 3D trusses for a given applied load and allowable stress. Nguyen et al. [32] and Yucel et al. [33] similarly used FFNN for size optimization of trusses. On the other hand, Nourian et al. [34] and Zheng et al. [35] employed Graph Neural Networks (GNNs) for truss size optimization, representing truss joints and members as graph vertices and edges, respectively. There are several noteworthy implementations to predict optimal truss topologies using computer vision neural networks. Sosnovik & Oseledets [36], Banga et al. [37], utilized Convolutional Neural Networks (CNNs) to predict topology-optimized results by using images of the partially topology-optimized structure as inputs. Further, Zheng et al. [38] used a U-Net CNN architecture to produce images of topology-optimized frames, incorporating volume fractions and forces as inputs. Senhora et al. [39] proposed a generalizable and scalable workflow to generate topology-optimized structures by refining coarse material distributions into finer, optimized outputs using CNNs. Moreover, Herath & Haputhanthri [40] introduced a workflow using a conditional Generative Adversarial Network (cGAN) to predict optimal topology structures, stress distributions, and failure points. In other noteworthy work, Neural Ordinary Differential Equations (NODEs) were applied to predict topology-optimized structures with unseen boundary conditions by Ma et al. [41].

Previous studies have demonstrated the feasibility of computer vision techniques in capturing structural topologies. CNNs are particularly suited, as they specialize in grid data, particularly images. First introduced by Yann et al. [42], CNNs have been extensively applied to image classification [42], [43], [44], image generation [45], and natural language processing [46]. They are designed to learn spatial hierarchies of features through backpropagation by using convolutional blocks adaptively. Transposed convolutional layers can reconstruct images into higher-resolution images, which is termed super-resolution imaging [47, 48]. While overfitting is a potential problem in CNNs, using dropout and batch normalization layers after convolutional or transpose convolutional operations usually offers a solution [49].

The integration of guidelines from design standards into machine learning based optimization usually increases computational costs. It emphasizes the requirement of enhancing the efficiency of machine learning techniques. Although there have been significant advances in topology optimization, limited research has focused on predicting practical and code-compliant skeletal structural topologies.

Previous studies have utilized computer vision techniques, such as CNNs and GANs, for predicting topology-optimized structures; however, their application to skeletal structures remains unexplored. Furthermore, to the best of the authors' knowledge, direct prediction (input-model-output) topology optimization implementations are limited to fixed design domains because of the requirement to maintain fixed input and output dimensions for these models. Additionally, producing practically applicable designs considering code guidelines using computationally efficient machine learning techniques has rarely been addressed in the literature. This study addresses these gaps through the following key contributions:

1.A CNN- and image processing-based technique that predicts standardized optimal cross-sections and the geometric arrangement of members for skeletal structures with a significant computational advantage.

2.An optimal section identification strategy based on image processing that generalizes the framework's applicability even when the design domain varies and provides a method to compensate for errors arising from unavoidable image generator-induced losses.

3.Prediction of practically applicable designs that satisfy code-stipulated guidelines; specifically, the optimization results comply with design rules in EN 1993–1–1: Design of Steel Structures [50].

The rest of the paper is organized as follows: Section 2 presents the methodology. It details the workflow for generative optimal design of code-compliant two-dimensional skeletal structures. This includes dataset generation, as well as how each stage of the workflow works: the low-resolution optimized layout generator (CNN64), the high-resolution spatially generalized frame generator (CNN512), and feature extraction on the optimized frame. Section 3 describes the experiments. These cover dataset generation, model training error estimates, and section identification algorithm calibration. Section 4 discusses the results and analysis. This includes model training and testing, architecture-sensitive quantitative analysis, and computational cost comparison. Section 5 summarizes the conclusions.

2. Methodology

In this study, a workflow is presented for the generative design of code-compliant two-dimensional skeletal structures using convolutional neural networks. Both cantilever and simply supported configurations are considered, and in each case the problem was defined by three design variables: the dimensions of the rectangular design domain (span $\mathscr{S}$ and height $\mathscr{H}$) and the magnitude of the concentrated load $\mathscr{F}$, which are collated in the input vector ($\mathscr{S}, \mathscr{H}, \mathscr{F}$). The output of this surrogate model consists of the geometric details of the optimal skeletal structure. It is noted that in this implementation, further reduction considering $\mathscr{S}/\mathscr{H}$ and $\mathscr{F}/\mathscr{S}$ as inputs is not feasible due to the use of standardized cross-sections. Firstly, code-compliant topology, size, and layout-optimized frames are generated by a specialized algorithm. The resulting frame geometry is rendered to a black-and-white image and then scaled to appropriate dimensions so that it can serve as the target for two separate CNN models in Stage 1 and Stage 2 (see Fig. 1). Further details of the model-training procedure (including loss functions, hyperparameter settings) and data generation (including optimization-related parameters) are provided in Section 3.1. In the evaluation or application stage, the workflow comprises three stages, which are illustrated in Fig. 2. In the succeeding discussion, we use the notation CNN$(\cdot)$ to denote a CNN with an output resolution of $(\cdot) \times (\cdot)$. For instance, CNN64 generates output of resolution 64×64.

2.1. *Stage 1: Low-resolution optimized layout generator - CNN64*

In Stage 1, the input vector ($\mathscr{S}, \mathscr{H}, \mathscr{F}$) is mapped to a $64\times$ 64 coarse binary layout using an autoencoder-style convolutional neural network (AE-CNN). The 3D input passes through fully connected layers with Batch Normalization and ReLU activations, producing a 256-dimensional latent vector. This is reshaped and decoded via transposed con-

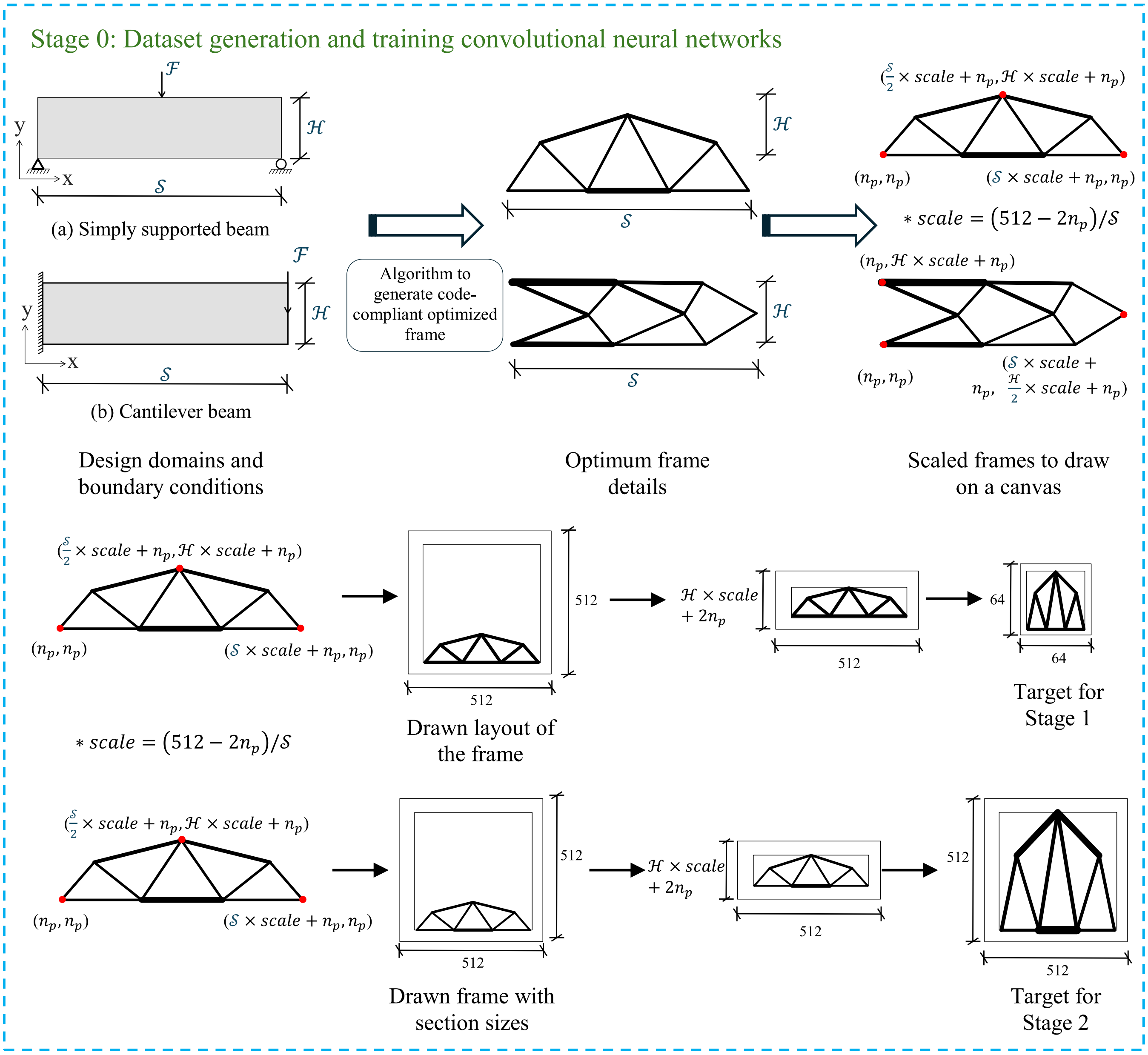

Stage 0: Dataset generation and training convolutional neural networks
(a) Simply supported beam
(b) Cantilever beam
Design domains and boundary conditions
Algorithm to generate code-compliant optimized frame
Optimum frame details
Scaled frames to draw on a canvas
* scale = (512 − 2n_p)/S
Drawn layout of the frame
Target for Stage 1
Drawn frame with section sizes
Target for Stage 2
512
64

Fig. 1. Data generation workflow for Stage 1 and Stage 2 CNN models.

volutional blocks (Deconv + BN + ReLU). A sigmoid activation in the final layer outputs a binarized image indicating material presence per pixel. The full pipeline is shown in Fig. 2 (Stage 1).

2.2. *Stage 2: High-resolution spatially generalized frame generator - CNN512*

Stage 2 refines the Stage 1 output and integrates section-size information to generate a 512× 512 binary image of a code-compliant frame. This encoder–decoder network features two input paths. Path A (section-size branch) processes the design variables $\mathscr{S}, \mathscr{H}, \mathscr{F}$ through fully connected layers with BN and ReLU to form a 256-dimensional latent vector. Simultaneously, the Layout Branch processes the 64 × 64 binary image from Stage 1 via convolutional layers with BN, ReLU, and dropout, yielding another 256-dimensional latent vector. These are concatenated into a 512-dimensional representation, which is decoded through transposed convolutional blocks (Deconv + BN + ReLU). A final sigmoid activation produces the binarized 512 × 512 output, capturing both topology and member sizing. The architecture is illustrated in Fig. 2 (Stage 2 panel).

2.3. *Stage 3: Feature extraction on the optimized frame*

Starting with the Stage 2 results, Stage 3 identifies geometric details (the section properties, node coordinates) of the optimal structure. As illustrated in Fig. 2 (Stage 3 panel), this feature identification algorithm performs three separate tasks: image resizing, node detection, and section identification. These tasks are each explained in more detail below.

2.3.1. *Output image resizing*

Stage 2 always generates spatially generalized images with 512 × 512 resolution, irrespective of the size of the initial design domain. This output needs to be resized to the correct scale before identifying the features of the optimal frame, such as the nodal coordinates and the assigned section sizes. An image with the correct scale is obtained by resizing to an image of width $w = 512$, and height $h =$

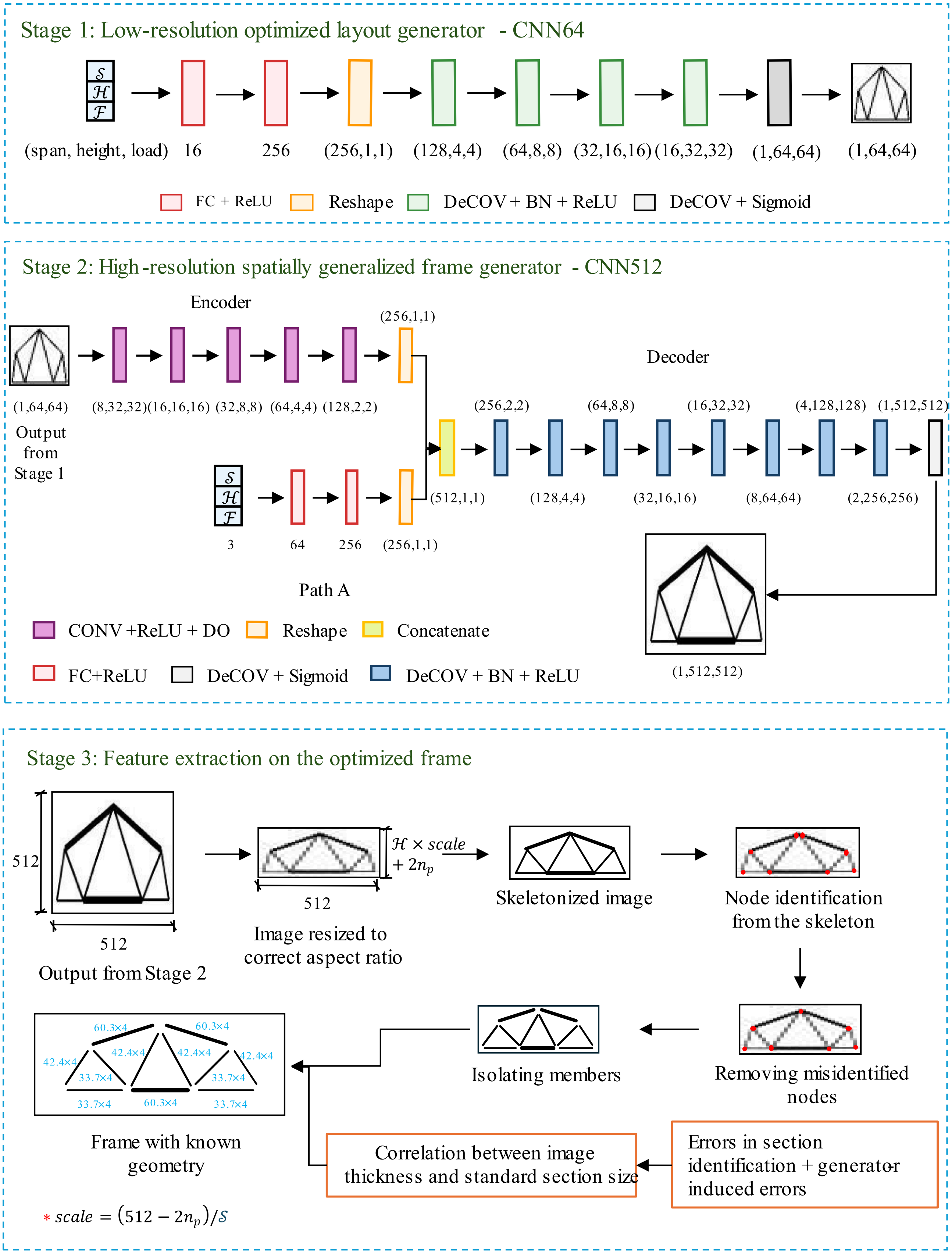


Fig. 2. Three-stage proposed workflow for generative design of code-compliant optimized frames.

$(\mathscr{S} \times \text{scale} + 2n_p)$. The scale $= \mathscr{S}/\mathscr{H}$, and n_p is the number of padding pixels. Padding is introduced during data-set generation to ensure that members located near the boundaries of the design domain (with finite thickness) are not inadvertently truncated when the image is cropped to the prescribed span and height. Specific values of n_p used in numerical implementations are discussed in Section 3.1.4.

2.3.2. Node detection

Features like the cross-section size of members cannot be identified without first isolating members in the resized image. We explore two possible methods to isolate members: (a) detecting nodes as intersection points of straight lines identified by the Hough transformation [51,52] and (b) determining nodes using a skeletal pixel-chain. The Hough transformation identifies a straight boundary between feature pixels and background pixels. However, even after training, the generator exhibits residual prediction errors, manifested as blurred boundaries at member edges and junctions, which reduce line-detection accuracy. Therefore, the second option, in the form of a skeletonization-based method, as expounded in references [11–13], was pursued. Skeletonization was performed using the Zhang-Suen thinning algorithm [53] to reduce member pixels to a 1-pixel-wide chain. Pixels in the skeletal chain are categorized into joint, end, and regular pixels based on their connection with neighboring pixels. However, it has the drawback that when thicker members are connected at a node, there is a possibility of mistakenly identifying multiple nodes instead of just one. To avoid this, nodes within a predefined radius of each other are averaged.

This misidentification of nodes arises from the nature of the Zhang–Suen thinning algorithm. At junctions where two members connect, the overlapping region contains more pixels than the width of the individual members. As a result, the algorithm produces short skeleton branches or pixel clusters instead of converging to a single point (as shown in Fig. 2, Stage 3; Node identification from the skeleton subfigure). Theoretically, this artifact appears when member widths exceed two pixels. In such cases, the maximum distance between pseudo nodes is roughly half the sum of the connecting member widths. Because the exact widths are unknown before member isolation, a predefined radius is used to average node locations. In this study, the radius corresponds to the maximum expected member width of 48 pixels. This approach ensures that all pseudo-nodes separated by less than the maximum possible width are merged effectively. After identifying the node points, the structure can then be decomposed into individual members.

2.3.3. Section identification

Once nodal coordinates and member connectivity are established, the next step is to determine the discrete cross-section size for each member by measuring its pixel width in the resized image. Denote by N_m the total number of members, and for each member $i \in \{1, 2, \ldots, N_m\}$ let D_i be its pixel-width and θ_i its inclination angle with respect to the horizontal axis. A two-pass connected-component labelling algorithm (see [54]) is first applied to the binary image so that all foreground pixels belonging to member i receive the same label. The total number of pixels in this connected component is denoted by A_i. Next, the minimum-area bounding box that encloses all pixels of member i is computed; let B_i and T_i be the width and height, respectively, of this bounding box (see Fig. 2, which illustrates bounding boxes in red for an inclined member (member 4) and a horizontal member (member 5)).

The geometric relations in Eqs. (1a) and (1b) were derived by modeling each pixelated member as a thin rectangle with uniform thickness. D_i, inclined at an angle θ_i. The total pixel count A_i approximates the area $D_i \times l_i$, where l_i is the centerline length. Using bounding box dimensions B_i (width) and T_i (height), the equations match the projected geometry. The width B_i equals the horizontal projection of the length plus the vertical projection of the thickness: $l_{i,h} = \frac{A_i\cos(\theta_i)}{D_i}$, so $B_i = l_{i,h} + D_i\sin\theta_i$. Similarly, the height T_i equals the vertical projection of the length plus the horizontal projection of the thickness: $l_{i,v} = \frac{A_i\sin(\theta_i)}{D_i}$, so $T_i = l_{i,v} + D_i\cos\theta_i$. Hence, the following geometric relations hold:

$$B_i = \frac{A_i\cos(\theta_i)}{D_i} + D_i\sin(\theta_i) \tag{1a}$$

$$T_i = \frac{A_i\sin(\theta_i)}{D_i} + D_i\cos(\theta_i) \tag{1b}$$

Once member pixel widths have been determined, they are mapped to standard section sizes using the lookup values in Table 3. This mapping table is constructed by first establishing the nominal correspondence between pixel width and section dimensions and then adjusting for the uncertainties introduced by Stage 3 measurement error and residual generator inaccuracies. Further details regarding the error quantification and calibration process are provided in Sections 3.2.2 and 3.3.3.

2.4. Implementation details

The entire workflow is developed in a Python environment [55]. The CNN models in Stages 1 and 2 were implemented using PyTorch 2.8.0 [56] with CUDA 12.6 [57] acceleration for GPU training. The feature extraction algorithm (Stage 3) was also coded in Python [58], leveraging scikit-image (v 0.25.0) [58] for skeletonization and OpenCV (v 4.10.0.84) [59] for two-pass connected-component labeling and bounding box computation.

3. Experiments

3.1. Data generation

The optimization algorithm developed by Sarma et al. [15], implemented in-house using written Python code, was used to generate training data for the model. The data generation workflow consists of six steps: (a) conducting a topology optimization, (b) extracting the medial (skeleton) axis from the topology-optimized frame, (c) converting the skeleton chain to Euler-Bernoulli frame members using a graph model, (d) removing redundant and short members by merging and pruning, (e) iterative size and layout optimization equipped with finer merging and pruning, (f) structural analysis, integrity check based on EN 1993–1–1 [60] and assignment of standard section sizes for cost optimization (See Fig. 3). This study assigns standard Circular Hollow Sections (CHS) to the optimal frame as per EN 10210–2:2019: Hot finished steel structural hollow sections [61]. For application 1 (cantilever problem), 1708 data points were generated, whereas for application 2 (simply supported problem), 1783 data points were generated. Herein, 1) problem type (cantilever or simply supported), 2) height ($\mathscr{H}$), 3) span ($\mathscr{S}$) of the design domain, and 4) concentrated load value ($\mathscr{F}$) were considered as input features. Succeeding discussions, furnish the optimization environment (Section 3.1.1) and range of design variables (Section 3.1.2) used to train, validate, and calibrate the proposed workflow.

3.1.1. Optimization-related parameters

In this study, the properties of structural steel S275 (elastic modulus: 210 GPa, Poisson's ratio: 0.3, and yield strength: 275 MPa) are utilized. However, the proposed workflow can also be applied to other homogeneous isotropic materials, such as aluminum and other strength classes of steel. The topology optimization stage employs the density-based SIMP method, with the volume fraction of 0.3 and a penalization factor of 3. Moreover, as the mesh size 0.01 m is maintained for topology optimization, while the filter radius is adaptively adjusted to address mesh dependency across different design domain sizes. An in-house developed solver, based on the optimality criteria algorithm, is used to perform the optimization. Subsequently, the topology preserved skeleton is extracted using a modified Zhang-Suen thinning algorithm, as detailed in the literature [15]. Following skeletonization, redundant and short members are removed based on the merging ratio, which is

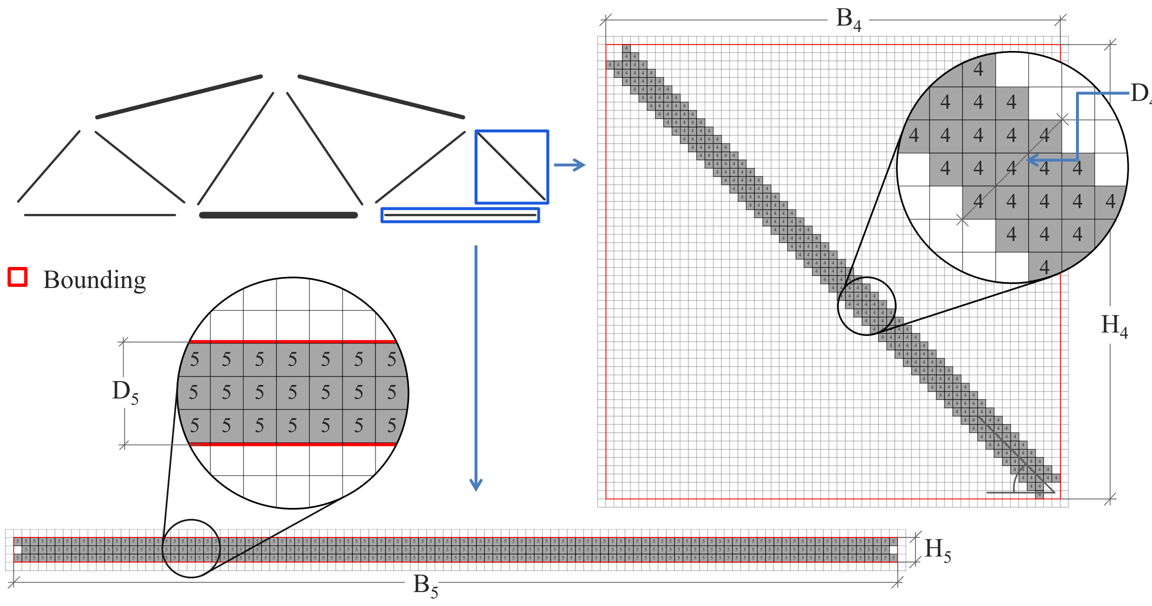


Fig. 3. Bounding boxes are drawn for each isolated element, and the width (B) and height (T) are determined to evaluate the member thickness using Equation (1).

defined as the ratio of a member's length to the maximum length of any member connected to either of its nodes. Although the merging ratio is application-specific, values between 0.1 and 0.2 are adopted in this study based on sensitivity analyses. Sequential size and layout optimizations are then carried out using the sequential least squares programming algorithm. Finally, the structural integrity of the optimized design is evaluated, considering critical internal forces, maximum displacement, and stability conditions in accordance with EN 1993–1–1 [60]. This process ultimately yields a design-informed skeletal structure.

3.1.2. *Design of experiments*

This paper aims to develop a novel technique for code-compliant generative modeling. Hence, a more theoretically oriented design of experiments is conducted to demonstrate the applicability of the developed workflow across various settings. To this end, it illustrates the ranges of the selected problem parameters for each problem type. For instance, in the cantilever configuration, the domain height ($\mathscr{H}$) takes values $\mathscr{H} \in [0.5, 2]$ m, whereas domain span $\mathscr{S} \in [1, 4]$ m, and concentrated vertical load $\mathscr{F} \in [5, 200]$ kN is applied at the middle of the right-hand-side edge (see Fig. 4 (top)). For the cantilever problem, 1708 data points were generated, while for the simply supported problem, 1783 data points were generated. The number of data points reflects the size of the variable domain space and follows the data point counts used in previous topology optimization studies for direction prediction (input–model–output) with computer vision models. For instance, Herath and Haputhanthri [62] used 670 data points for the cantilever boundary condition, and Behzadi and Bi [63] used 1000 data points per boundary condition configuration.

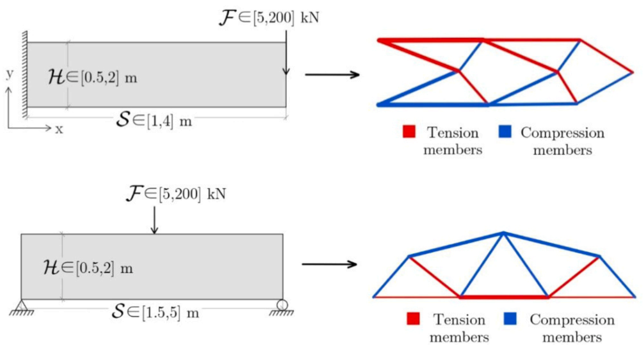

$\mathcal{F}\in[5,200]$ kN
$\mathcal{H}\in[0.5,2]$ m
$\mathcal{S}\in[1,4]$ m
y
x
Tension members
Compression members
$\mathcal{F}\in[5,200]$ kN
$\mathcal{H}\in[0.5,2]$ m
$\mathcal{S}\in[1.5,5]$ m
Tension members
Compression members

Fig. 5. Range of design variables for the (top) cantilever problem dataset (bottom) simply supported problem dataset.

3.1.3. *Data normalization*

As a preprocessing step, the generated data is normalized to enhance CNN training stability and convergence. This scaling ensures all inputs lie within a uniform [0,1] range [64], [65]. The minimum-maximum scaling technique is adopted for this study. This normalization takes the form:

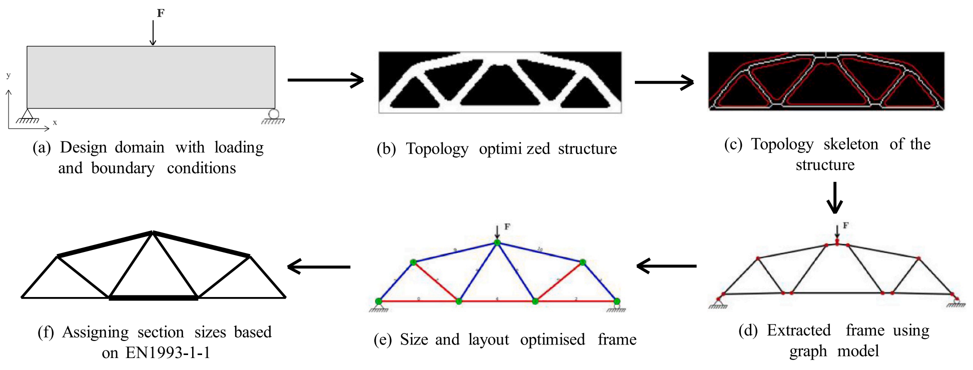

F
y
x
(a) Design domain with loading and boundary conditions
(b) Topology optimi zed structure
(c) Topology skeleton of the structure
(d) Extracted frame using graph model
(e) Size and layout optimised frame
(f) Assigning section sizes based on EN1993-1-1

Fig. 4. Workflow for design-informed structural optimisation of 2D skeletal frames.

$$\bar{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

where $\bar{x}_i$, x_i, $x_{\min}$, and $x_{\max}$ are the normalized input, the original input, the minimum, and the maximum entries of the inputs x_i, respectively.

3.1.4. Spatial generalization

Maintaining a consistent output size for computer vision networks is essential, particularly for generating target images of 64×64 and 512×512 dimensions for Stages 1 and 2 in the proposed workflow (see Fig. 2). To cater to this requirement, we introduce spatial generalization. In this process, each nodal coordinate is mapped according to:

$$z_{sg} = z \times \text{scale} + n_p \tag{3}$$

where z_{sg} and z denote the spatially generalized and physical x, y coordinates, respectively. Employing a scale $= (512 - 2n_p)/\mathscr{S}$ with a pixel-padding $n_p = 25$ is suitable to generate images with the above-mentioned dimensions. With the acquired spatially generalized coordinates, the frame elements are mapped onto a 512×512 canvas. For Stage 1, the elements are drawn with uniform thickness. For Stage 2, equivalent line thicknesses are selected, resulting from structural optimization. To eliminate the void region of the drawn images, both images are cropped. The image is cropped such that the region is enclosed by the coordinates (0,0), (512,0), (512,$\mathscr{H}\times$ scale$+2n_p$), and (0,$\mathscr{H}\times$ scale$+2n_p$). Then, the Stage 1 image is resized to 64×64, and the Stage 2 image is resized to 512×512 resolution (see Fig. 1).

3.2. Calibration of the section identification algorithm

As mentioned earlier, identifying discrete member sizes is formulated as a probabilistic classification task. Tolerance thresholds for standard sections are established based on inaccuracies arising from the computer-vision workflow. Errors may manifest through two primary mechanisms: (a) errors introduced by the feature extraction algorithm (see 3.2.1) and (b) inaccuracies arising from generator residual error (see 3.2.2).

3.2.1. Errors introduced by the feature identification algorithm

The feature identification workflow introduces errors from several sources. These include misinterpretation of nodal positions and reliance on Equation 1, which is optimized for rectangular elements. However, isolated members often exhibit non-rectangular shapes due to circular punching for separation. Other key sources are spatial generalization and resizing back to the original scale (changing resolution). We treated these errors as a single aggregated unit. This approach captures the overall impact on section size identification. We comprehensively studied the errors associated with straight lines through feature identification, where straight lines were generated on images by varying the thickness, length, and inclination of the lines, as well as the width and height of the images. Then the thickness was identified using the feature's identification algorithm for each straight line, and the difference between the thickness and the identified thickness was evaluated. The width and height of the image are analogous parameters to $\mathscr{S}$ and $\mathscr{H}$. In this study, all variables are sparse in the domain where they vary in the outline design problems (Section 3.1.2). Correlations between the error in section size and other parameters are shown in Table 1.

Table 1
Correlation between independent variables and error in D_i in the section identification algorithm calibration study.

Parameter	Correlation between the error in D_i
Length of the straight line	-0.424
Actual thickness (D'_i)	0.325
Inclination (θ_i)	0.502
Width of the image	0.078
Height of the image	-0.170

The correlations between the width, height, and actual thickness D'_i with the error of the D_i appear to be negligible as per a mutual heat-map analysis. Similarly, the correlations of the inclination–error and actual thickness–error are also found to be insignificant, see Table 8. Consequently, a range for errors is identified instead of developing a direct relationship between independent parameters and errors. To this end, it was found that the (min, max) error associated with the identified section falls within the range of (-1.5,1.5) pixels.

3.2.2. Errors introduced by the generator residual error

Residual errors persist in any trained machine learning model. These residual errors in the trained models of Stages 1 and 2 lead to prediction inaccuracies of Stage 3, which eventually impact the accuracy of the identified section sizes. The total accurate count of feature pixels in a generated image (A') is derived by adding falsely identified feature pixels (FN) and subtracting falsely identified background pixels (FP) from the identified feature pixels (A). This error (e) takes the form as in Equation 10. Considering a unit length of a member, Eq. 4a is modified into Eq. 4b to calculate the corrected member thickness (D'_i) using the determined diameter (D_i) from Equation 1.

$$\mathrm{e} = \frac{A - A'}{A'} = \frac{FN - FP}{A'} \tag{4a}$$

$$D'_i = \frac{D_i}{1 + e} \tag{4b}$$

The error is shown due to the residual error in the generator as a dataset-averaged deterministic range. Hence, the ratio e is computed for each instance by comparing the target and predicted images across both configurations, using the entire dataset. Table 2 shows the 95 % confidence intervals $= \left(\mu_e - \frac{1.96\sigma_e}{\sqrt{n}}, \mu_e + \frac{1.96\sigma_e}{\sqrt{n}}\right)$, where μ_e, σ_e, and n are the mean, standard deviation for the calculated error per feature pixel (e), and the total number of elements in the dataset, respectively. The comparatively lower 95 % confidence interval for the simply supported dataset is due to the lower complexity stemming from the symmetry of the optimal designs in the dataset.

3.2.3. Probabilistic calibration

As standard sections, we selected CHS according to EN 10210–2 [61], where each section is defined uniquely by its standard diameter ($\overline{D}$) and thickness ($\overline{t}$). Standard sections are drawn in the canvas in equivalent member diameters (D_j^{eq}) in pixels shown in Table 2 for generated images. At this juncture, using a 95 % confidence interval, we selected equivalent section sizes in steps of 6 pixels to prevent overlapping the error margin, considering both errors in feature identification (±1.5 px) and generator residual error ($\pm\max\left(|\mu_e - 1.96\frac{\sigma_e}{\sqrt{n}}|, |\mu_e + 1.96\frac{\sigma_e}{\sqrt{n}}|\right) \times \max(D^{eq})$). After the feature identification algorithm identifies section sizes, a standard CHS is assigned by mapping according to Table 3. For example, if an identified section diameter (D_i) falls within (21.25,26.75) px, a 60.3 mm× mm CHS section is proposed as per Table 3.

3.3. Model evaluations and error estimates

To comprehensively evaluate the fidelity of the generated binary

Table 2
Confidence intervals at a 95 % level for 1+e, accounting for errors arising from trained models.

Dataset	95 % confidence interval for $1 + e$
Cantilever	(0.982,0.988)
simply supported	(0.986,0.991)

Table 3
Mapping of CHS diameters from [61] to the equivalent D_j^{eq} in the CNN-predicted image.

Standard section $\overline{D}_j$ (mm) $\times$ $\overline{t}_j$ (mm) [61]	Equivalent diameter in the resized image D_j^{eq} (px)	$D_j \in D_j^{eq} \pm 2.75$ (px)
21.3×3.2	6	(3.25,8.75)
33.7×4	12	(9.25,14.75)
42.4×4	18	(15.25,20.75)
60.3×4	24	(21.25,26.75)
76.1×5	30	(27.25,32.75)
88.9×5	36	(33.25,38.75)
114.3×5	42	(39.25,44.75)
139.7×6.3	48	(45.25,50.75)

images, we adopted four established image segmentation metrics. Herein, we define mean Binary Cross Entropy (mean BCE), Pixel Accuracy (ACC), True Positive Rate (TPR) (or recall), and True Negative Rate (TNR) (or selectivity) as evaluation metrics to test generated images from Stage 2. Mean BCE is preferred for binary image outputs because it magnifies prediction errors for binary values [58]. ACC metric presents the proportion between correctly identified and total pixels [59,60]. Accurate identification of member sizes is crucial in this work. Hence, the TPR metric is chosen for its ability to gauge the proportion of correctly identified feature pixels and total feature pixels. Similarly, TNR evaluates the proportion between correctly identified and total background pixels [59]. Further, the Mean Squared Error (MSE) and BCE are employed as loss functions for training, as detailed later in Section 4.1. The mean BCE is given in Eq. 4, and the MSE is given in Table 4, where N and M are the total number of images and the total number of pixels in the image, respectively. x_{true}^{ij} and x_{pred}^{ij} are the target value and the predicted value of the j-th pixel in the i-th image, respectively. Table 4 lists the equations for ACC, TPR, and TNR, where the used notations are explained in Table 4.

$$\text{Mean BCE} = -\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[x_{\text{true}}^{ij}\log\left(x_{\text{pred}}^{ij}\right) + \left(1 - x_{\text{true}}^{ij}\right)\log\left(1 - x_{\text{pred}}^{ij}\right)\right] \tag{4}$$

4. Results and discussion

4.1. Model training and testing

We trained CNN models (Stage 1: CNN64 and Stage 2: CNN512) for two design-informed optimization problems: cantilever and simply supported. As Fig. 6 illustrates, both models were trained separately for both datasets. Mean Squared Error (MSE) and Binary Cross-Entropy (BCE) loss functions were used to evaluate the model training. Results relevant to training and validation are based on models trained using the BCE loss function (see Eq. 4). Following the recommendations of Goodfellow et al. [49], the Adaptive Moment Estimation (Adam) optimizer was utilized for training models with decaying rates of 0.9 and 0.99. Both models were trained for 500 epochs. The computational tasks for training were executed on a computational unit powered by an Intel (R) Core(TM) i5–8250U CPU running at 1.60 GHz. Fig. 6 illustrates the converging learning curves of CNN64 and CNN512 models for both cantilever and simply supported datasets.

Table 4
Evaluation metrics.

Evaluation criterion	Equation
Mean Binary Cross Entropy (mean BCE)	Eq. 4
Pixel Accuracy (ACC)	$\frac{TN + TP}{TN + TP + FP + FN}$
True Positive Rate (TPR)	$\frac{TP}{TP + FN}$
True Negative Rate (TNR)	$\frac{TN}{TN + FP}$
Mean Squared Error (MSE)	$\frac{1}{NM}\sum_{i=1}^{N}\sum_{i=1}^{M}\left(x_{\text{pred}}^{ij} - x_{\text{true}}^{ij}\right)^2$

The optimal hyperparameters, particularly the learning rate and mini-batch size, were identified through an iterative grid search method. This method evaluated combinations of hyperparameters using the metrics in Table 4, and values with the highest performance are identified. The considered sets for the grid search and the obtained optimal values for the hyperparameters are detailed in Table 5. Both cantilever and simply supported datasets showed optimal performance with a mini-batch size of 40 and a learning rate of 0.001 when training the CNN64 model. In contrast, the CNN512 model required a mini-batch size of 80 for both datasets. However, the learning rate for the cantilever dataset was 0.005, while the simply supported dataset required a learning rate of 0.01. The consistency in mini-batch size suggests that both datasets benefit from the same level of gradient estimation accuracy and computational efficiency during training. However, the different learning rates suggest inherent differences in the characteristics of the datasets. Specifically, the cantilever data set contains complex patterns, thus necessitating a lower learning rate to achieve stable learning, whereas simply supported cases have an axis of symmetry. In this spectrum of design variables, the member cross sections in the cantilever configuration are more sensitive during the training phase.

4.2. Architecture-sensitive quantitative analysis

In this study, models are trained on two distinct datasets. Due to each dataset's unique features, finding an optimum computer vision model that works with both datasets and has fewer learnable parameters is nearly impossible. Hence, we present the best-performing model in the methodology, considering both datasets. This section compares three other nearly optimal models against the proposed model (CNN64 +CNN512) with the cantilever and simply supported datasets. Binary Cross Entropy (BCE), pixel accuracy (ACC), True Positive Rate (TPR), and True Negative Rate (TNR) metrics are used to validate these three deep networks, which are discussed below.

CNNC512: The CNNC512 is an auto-encoder convolutional neural network model designed to generate 512×512 binary images, taking $(\mathscr{S}, \mathscr{H}, \mathscr{F})$ as inputs. It features a (512,) latent space similar to the proposed model and generates spatially generalized, design-informed, optimized frame images without a primary generator like CNN64. Its architecture is similar to the proposed model's, utilizing fully connected layers and transposed convolutional layers to upsample the three input parameters into the binary image.

FFNN64 +CNN512: The FFNN64 +CNN512 models are similar to the CNN64 +CNN512 proposed combination. Instead of a CNN, an FFNN is utilized in this model to generate the layout of an optimized frame. The FFNN64 model comprises fully connected layers, followed by 1D BN, ReLU activation, and DO layers. The final layers include a reshape layer and a sigmoid layer, converting the output into a 64×64 image.

CNN64 +CNN256: The CNN64 +CNN256 model utilizes the same architecture as the proposed model, taking $(\mathscr{S}, \mathscr{H}, \mathscr{F})$ as inputs. However, produce design-informed frame images at the 256×256 resolution. To create datasets for the model, we downsample 512×512 images to 256×256. One convolutional block is fewer in the CNN256 compared to the CNN512 model to decrease the resolution by half. The justification for considering this model is to investigate the compromise between accuracy versus computational time taken for model calibration.

The succeeding Sections 4.3 and 4.4 discuss the applicability and comparative performance of the proposed method using cantilever and simply supported problems, respectively.

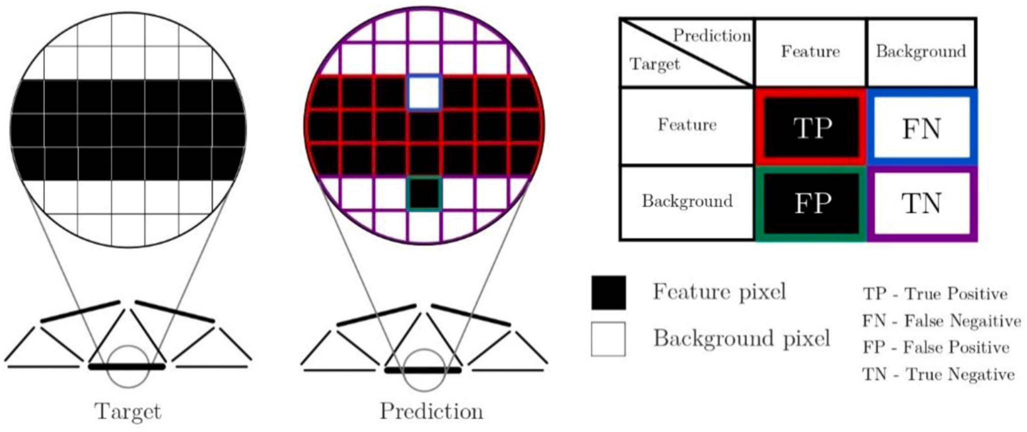


Fig. 6. Color coded confusion matrix for section identification.

Table 5
Optimal hyperparameters selected through a trial-and-error method to train the models.

Model	Hyperparameter	Considered set	Optimal value
CNN64 with the cantilever data set	Mini batch size	{16,32,64,128}	40
	Learning rate	{0.0005, 0.001, 0.005,0.01}	0.001
CNN512 with the cantilever data set	Mini batch size	{16,32,64,128}	80
	Learning rate	{0.001, 0.005,0.01,0.02}	0.005
CNN64 with the simply supported data set	Mini batch size	{16,32,64,128}	40
	Learning rate	{0.0005, 0.001, 0.005,0.01}	0.001
CNN512 with the simply supported data set	Mini batch size	{16,32,64,128}	80
	Learning rate	{0.001, 0.005,0.01.0.02}	0.01

4.3. Model evaluations for the cantilever dataset

4.3.1. Optimum layout and generalized frame generator (Stages 1 and 2)

Fig. 7 illustrates a set of images predicted by each model (rows (a) to (d)) for model-unseen input variables. The same figure depicts the ground truth in row (e), which is generated from the validated framework of Sarma et al. [15]. Table 6 presents the comparison of model performances against predefined error metrics. The BCE metric for the CNNC512 model is significantly higher than that of the other three models, indicating its poorer ability to learn patterns in the cantilever dataset. Additionally, as the target image output size increases, the ACC, TPR, and TNR metrics improve. The FFNN64 +CNN256 model exhibits the lowest ACC compared to other models with 512×512 resolution target images. This lower ACC in the FFNN64 +CNN256 model is illustrated by the poorly predicted images shown in Fig. 7 row (c) compared to other rows. By adding just one convolutional block, which results in a mere 0.037 % increase in learnable parameters, the CNN64 +CNN512 model shows significant improvements of 22.69 % in BCE, 0.77 % in ACC, 0.62 % in TPR, and 0.64 % in TNR metrics compared to the CNN64 +CNN256 model. This slight increase in learnable parameters enhances the model's capacity and feature extraction ability, allowing it to learn more complex patterns.

Member isolation in the feature identification algorithm ensures that the discontinuities in predicted members do not affect the accuracy of the identified section. However, the considerable disappearance of members causes issues in the feature extraction process. Hence, the CNNC512 and CNN64 +CNN256 models are not very suitable for predicting design-informed optimized frames for the cantilever configuration (rows (a) and (c) in Fig. 7). Moreover, the FFNN64 +CNN512 model and the CNN64 +CNN512 model demonstrate comparatively superior performance across all four metrics. This is illustrated by the very similar predictions shown in row (b) and row (d) in Fig. 7. However, the FFNN64 +CNN512 model has 87.56 % more learnable parameters compared to the CNN64 +CNN512 model. Between the best-performing FFNN64 +CNN512 and CNN64 +CNN512 models, the CNN64 +CNN512 model is selected as the optimum model for its overall superior performance with fewer learnable parameters. Furthermore, the generator error (e) is calculated for both the CNN64 +CNN512 model and the FFNN64 +CNN512 model using the same method explained in Section 3.2.2. Table 7 presents $\max\left(|\mu_e - 1.96\frac{\sigma_e}{\sqrt{n}}|, |\mu_e + 1.96\frac{\sigma_e}{\sqrt{n}}|\right)$ for both models. The FFNN64 +CNN512 model has a 42.02 % higher $\max\left(|\mu_e - 1.96\frac{\sigma_e}{\sqrt{n}}|, |\mu_e + 1.96\frac{\sigma_e}{\sqrt{n}}|\right)$ value compared to the CNN64 +CNN512 model, indicating a significantly larger generator error. Therefore, we claim that the CNN64 +CNN512 model is the overall best-performing model, as it has a lower generator error and fewer trainable parameters.

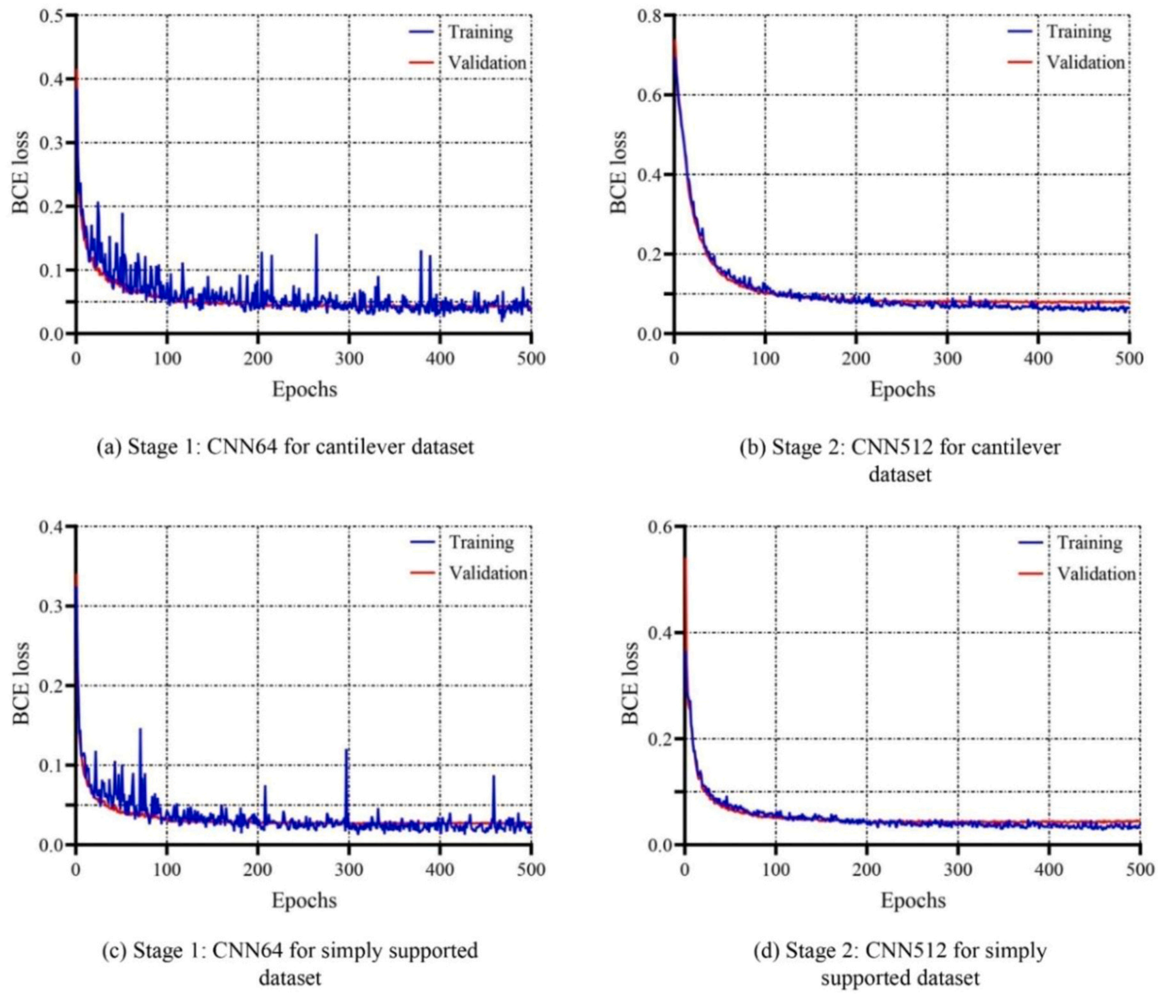
4.3.2. Feature extraction on the optimized frame (Stage 3)

Herein, the accuracy of the proposed probabilistic section identification algorithm for the cantilever dataset with $\mathscr{S} = 2.4$ m, $\mathscr{H} = 0.8$ m, and $\mathscr{F} = 80$ kN is discussed. The same is summarized in Table 8 with members defined as per Fig. 8. As given in Equation 6, we selected the tolerance margin for equivalent section sizes to prevent section misidentification through statistical tools. The results in Table (compare the true section sizes in column 2 versus the predicted section sizes in column 6) demonstrate the higher accuracy of the section identification of the predicted optimal structure.

4.4. Model evaluations for the simply supported dataset

4.4.1. Optimum layout and generalized frame generator (Stages 1 and 2)

The higher ACC, TPR, and TNR metrics for each model on the simply supported dataset are attributed to the problem's lower complexity, resulting from symmetry and less variation in section sizes compared to the cantilever dataset. However, like the cantilever dataset, the CNNC512 model demonstrates poor learning ability, as evidenced by its significantly higher mean BCE value than the other three models. Additionally, the CNNC512 model exhibits the poorest performance across the other three metrics. The blurry image shown in Fig. 9 provides evidence for poor prediction from the CNNC512 model. Enhancing the CNN64 +CNN256 model to the CNN64 +CNN512 model results in improved performance across all four metrics. As shown in Table 9 mean BCE, ACC, TPR, and TNR are improved by 41.68 %, 0.53 %, 1.04 %, and 0.42 %, respectively. Despite having 87.56 % more learnable



(a) Stage 1: CNN64 for cantilever dataset

(b) Stage 2: CNN512 for cantilever dataset

(c) Stage 1: CNN64 for simply supported dataset

(d) Stage 2: CNN512 for simply supported dataset

Fig. 7. Training and validation BCE loss curves for models in the proposed workflow.

Table 6
Evaluation metrics for selected models trained with the cantilever dataset. All models are trained with optimal hyperparameters prior to this evaluation.

Model	Trainable parameters	Mean BCE	ACC	TPR	TNR
CNNC512	2815,733	7.934	0.917	0.807	0.948
FFNN64 +CNN512	7909,389	0.333	0.917	0.807	0.947
CNN64 +CNN256	4215,350	0.401	0.911	0.802	0.942
CNN64 +CNN512	4216,910	0.310	0.918	0.807	0.948

Table 7
Comparison of generator error (e) between CNN64 +CNN512 and FFNN64 +CNN512 models for the cantilever dataset.

Model	$\max\left(\lvert\mu_e - 1.96\frac{\sigma_e}{\sqrt{n}}\rvert, \lvert\mu_e + 1.96\frac{\sigma_e}{\sqrt{n}}\rvert\right)$
FFNN64 +CNN512	2.535×10^{-2}
CNN64 +CNN512	1.785×10^{-2}

Table 8
Accuracy of the section identification for cantilever problem with $\mathscr{S} = 2.4$ m, $\mathscr{H} = 0.8$ m, $\mathscr{F} = 80$ kN. (Member indices are shown in Fig. 8).

Member	Section by [15] $\overline{D}$ (mm) × $\overline{t}$ (mm)	Equivalent thickness (px)	D_i (px)	Error (px)	Proposed section $\overline{D}$ (mm) × $\overline{t}$ (mm)
M_c^1	76.1 × 5	30	28.24	1.76	76.1 × 5
M_c^2	33.7 × 4	12	13.34	-1.34	33.7 × 4
M_c^3	33.7 × 4	12	13.17	-1.17	33.7 × 4
M_c^4	76.1 × 5	30	27.36	2.64	76.1 × 5
M_c^5	42.4 × 4	18	20.44	-2.44	42.4 × 4
M_c^6	42.4 × 4	18	15.91	2.09	42.4 × 4
M_c^7	33.7 × 4	12	12.71	-0.71	33.7 × 4
M_c^8	42.4 × 4	18	19.49	-1.49	42.4 × 4
M_c^9	33.7 × 4	12	11.55	0.45	33.7 × 4
M_c^{10}	33.7 × 4	12	10.66	1.34	33.7 × 4
M_c^{11}	33.7 × 4	12	14.38	-2.38	33.7 × 4
M_c^{12}	33.7 × 4	12	14.73	-2.73	33.7× 4
M_c^{13}	33.7 × 4	12	10.14	1.86	33.7 × 4
M_c^{14}	33.7 × 4	12	9.93	2.07	33.7 × 4

parameters, the FFNN64 +CNN512 model shows inferior performance in comparison to the proposed CNN64 +CNN512 model. Herein, performance reductions of 54.14 % in BCE, 0.95 % in ACC, 2.63 % in TPR, and 0.52 % in TNR metrics are observed. This is caused by the FFNN's poorer learning capability with grid-type data when compared to CNNs [32]. For ACC, TPR, and TNR metrics, the CNN64 +CNN256 model's performance is greater than the FFNN64 +CNN512 model. Hence, the best two CNN64 +CNN256 and CNN64 +CNN512 models are considered for further generator loss analysis.

Table 10 shows the generator loss relevant parameter $\max\left(\lvert\mu_e - 1.96\frac{\sigma_e}{\sqrt{n}}\rvert, \lvert\mu_e + 1.96\frac{\sigma_e}{\sqrt{n}}\rvert\right)$ for CNN64 +CNN256 and

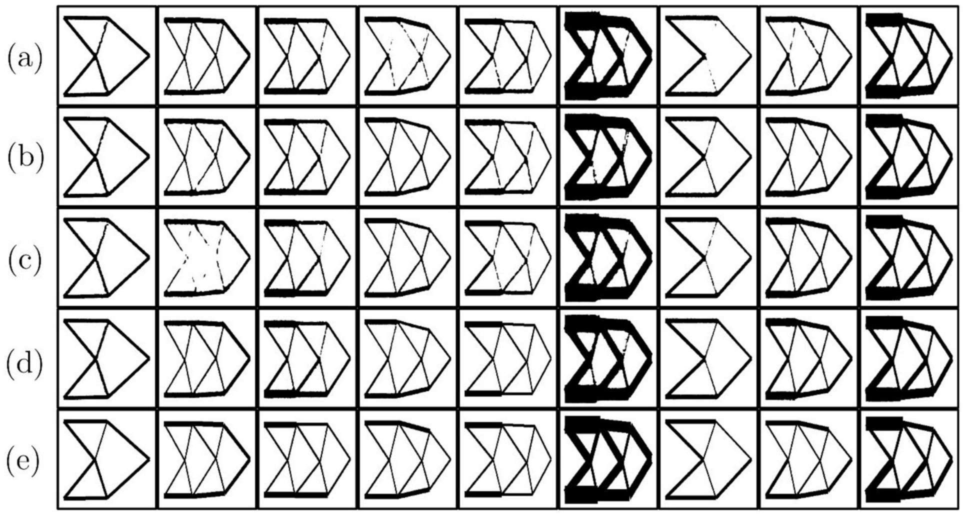


Fig. 8. Visual comparison of predictions on unseen data: (a) CNNC512, (b) FFNN64 +CNN512, (c) CNN64 +CNN256, (d) CNN64 +CNN512, (e) ground truth for the cantilever dataset.

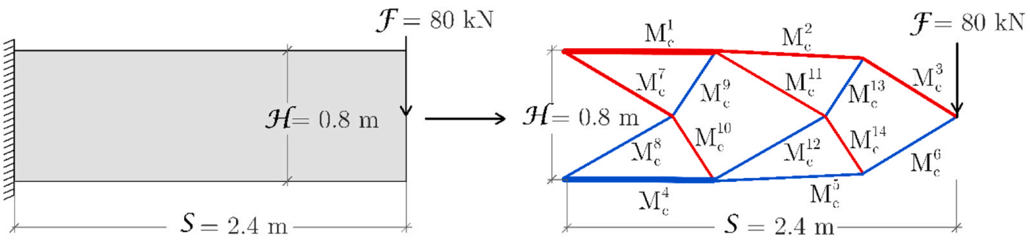


Fig. 9. Topology-size-layout optimal structure and its member identifiers for the cantilever problem with $\mathscr{S}$ = 2.4 m, $\mathscr{H}$ = 0.8 m, and $\mathscr{F}$ = 80 kN.

Table 9
Evaluation metrics for selected models trained with the simply supported dataset. All models are trained with optimal hyperparameters prior to this evaluation.

Model	Trainable parameters	Mean BCE	ACC	TPR	TNR
CNNC512	2815,733	6.747	0.932	0.841	0.959
FFNN64 +CNN512	7909,389	0.632	0.937	0.851	0.962
CNN64 +CNN256	4215,350	0.703	0.941	0.865	0.963
CNN64 +CNN512	4216,910	0.410	0.946	0.874	0.967

Table 10
Comparison of generator error (e) between CNN64 +CNN256 and CNN64 +CNN512 models for the simply supported dataset.

Model	$\max\left(\lvert\mu_e - 1.96\frac{\sigma_e}{\sqrt{n}}\rvert, \lvert\mu_e + 1.96\frac{\sigma_e}{\sqrt{n}}\rvert\right)$
CNN64 +CNN256	2.005×10^{-2}
CNN64 +CNN512	1.352×10^{-2}

CNN64 +CNN512 models. The CNN64 +CNN256 model has a higher generator loss (48.30 %) compared to the proposed model. Hence, the CNN64 +CNN512 model is evident to be the best-performing model.

4.4.2. *Feature extraction on the optimized frame (Stage 3)*

The accuracy of the proposed probabilistic section identification algorithm, applied to the simply supported problem with parameters $\mathscr{S}$ = 3 m, $\mathscr{H}$ = 1 m, and $\mathscr{F}$ = 130 kN (see Fig. 10), is quantified in Table 11. Similar to the observations made in conventional methods [15], it is evident that the predicted section sizes are in complete agreement with the section sizes in the test dataset.

4.5. *Computational cost comparison*

We compare the computational cost of the proposed design-informed optimization algorithm with the existing approach presented by [15], applied to various design domains and concentrated loads. The comparison was conducted using an Intel(R) Core(TM) i5–8250U CPU @ 1.60 GHz computational unit. In the proposed workflow, the time required to generate high-resolution images of spatially generalized optimized frames in Stages 1 and 2 is negligible, as shown in Table 12. For Stage 3, the computation time varies, with a mean of 16.31 s and a standard deviation of 6.51 s, depending on the specific problem parameters. In contrast, the computational cost in the existing framework proposed by Sarma et al. [15] escalates significantly as the number of elements and the design domain dimensions increase. To achieve optimized results using [15], the algorithm often requires multiple iterative executions, with adjustments to critical parameters such as the merging ratio. Herein, Table 12 assumes that the optimal results of Sarma et al. [15] are achieved on the first attempt, hence the best timing. In comparison, the proposed method demonstrates that the result generation is independent of the design domain dimensions. Table 12 clearly illustrates that the proposed framework offers significant time savings compared to the existing method.

4.6. *Limitations and research prospects*

Although this work is novel in the computational structural optimization domain, a few limitations and future areas for research are outlined here. In this study, we investigated the applicability of the workflow specifically for simply supported and cantilever boundary

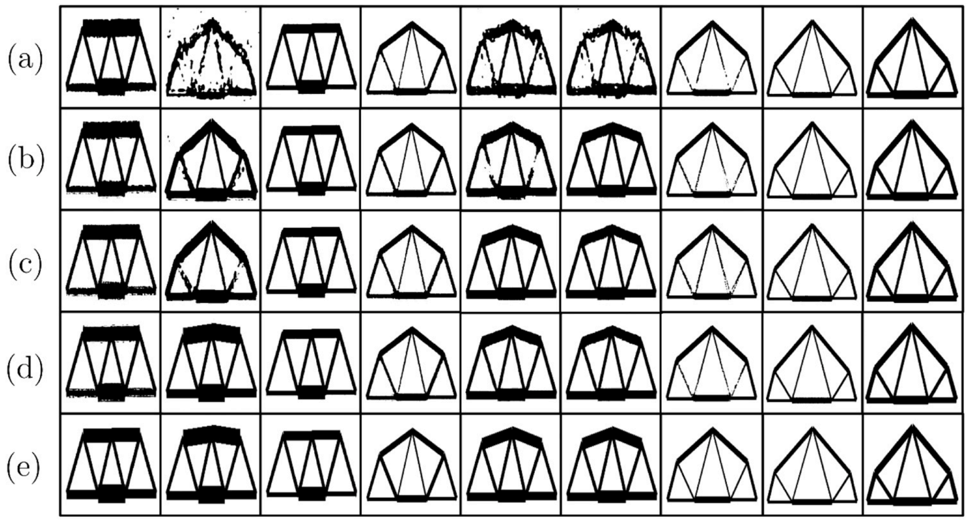


Fig. 10. Visual comparison of predictions on unseen data: (a) CNNC512, (b) FFNN64 +CNN512, (c) CNN64 +CNN256, (d) CNN64 +CNN512, (e) ground truth for the simply supported dataset.

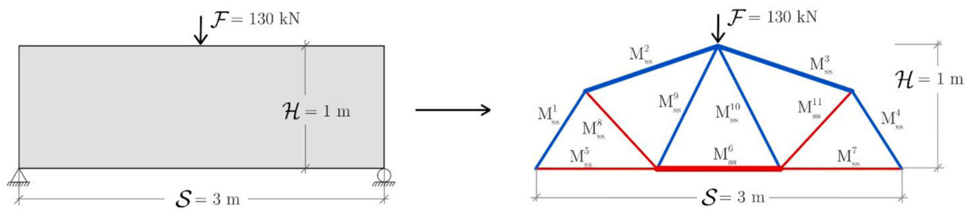


Fig. 11. Topology-size-layout optimal structure and its member identifiers for the simply supported problem with S= 3 m, H= 1 m, F= 130 kN.

Table 11
Accuracy of the section identification for simply supported problem $\mathscr{S}$ = 3 m, $\mathscr{H}$ = 1 m, $\mathscr{F}$ = 130 kN. (Member indices are shown in Fig. 10).

Member	Section by [15] $\bar{D}$ (mm) × $\bar{t}$ (mm)	Equivalent thickness (px)	D_i (px)	Error (px)	Proposed section $\bar{D}$ (mm) × $\bar{t}$ (mm)
M^1_{ss}	42.4 × 4	18	16.31	1.69	42.4 × 4
M^2_{ss}	60.3 × 4	24	23.66	0.34	60.3 × 4
M^3_{ss}	60.3 × 4	24	23.58	0.42	60.3 × 4
M^4_{ss}	42.4 × 4	18	15.29	2.71	42.4 × 4
M^5_{ss}	33.7 × 4	12	12.98	-0.98	33.7 × 4
M^6_{ss}	42.4 × 4	18	18.76	-0.76	42.4 × 4
M^7_{ss}	33.7 × 4	12	11.85	0.15	33.7 × 4
M^8_{ss}	33.7 × 4	12	11.43	0.57	33.7 × 4
M^9_{ss}	33.7 × 4	12	11.64	0.36	33.7 × 4
M^{10}_{ss}	33.7 × 4	12	11.1	0.90	33.7 × 4
M^{11}_{ss}	33.7 × 4	12	11.57	0.43	33.7 × 4

Table 12
Computational cost comparison between existing and proposed methods.

	Test case ($\mathscr{S}$(m), $\mathscr{H}$(m), $\mathscr{F}$ (kN))	Time taken by existing algorithm [15] (s)	Time taken by Stage 1 and Stage 2
Simply supported	(4.00,2.00,10)	1089.1	0.034
	(3.50,1.50,75)	298.4	0.056
	(4.00,1.75,140)	521.6	0.034
	(3.00,1.25,100)	222.8	0.028
	(2.50,1.00,135)	140.7	0.046
	(2.00,0.80,120)	94.8	0.038
Cantilever	(5.00,2.00,5)	724.0	0.038
	(3.00,0.70,95)	219.2	0.039
	(5.00,1.50,200)	711.5	0.042
	(4.20,1.50,105)	408.2	0.041
	(4.50,1.50,120)	423.1	0.034
	(3.50,1.80,105)	405.1	0.041

conditions, each with a single concentrated load configuration in a 2D continuum domain. The scalability of the proposed workflow to other problems, such as multiple loadings, different boundary conditions, and 3D continua, remains to be studied. The methodology is inherently extensible to account for such variations but requires additional data set generation, model training, model architecture changes, and calibration of the section size identification algorithm. Another limitation is the sole use of CHS sections in the predicted frames, which should be extended to other standard sections.

5. Conclusions

This study investigates the application of computer vision techniques to skeletal-structure optimization, an area that remains unexplored. It addresses these gaps by introducing a CNN- and image-processing-based technique that predicts standardized optimal cross-sections and the geometric arrangement of members for skeletal structures, offering significant computational advantages. Furthermore, direct prediction topology-optimization implementations are limited to fixed design domains. This limitation arises from the need to maintain fixed input and output dimensions for these models. In this study, we propose a strategy to treat the design domain as a variable. Additionally, the proposed workflow yields practically applicable designs that consider code guidelines and standard section sizes, which have rarely been addressed in the literature.

The proposed three-stage workflow combines two convolutional neural networks: CNN64 (Stage 1) and CNN512 (Stage 2), with a probabilistic geometric feature extraction algorithm (Stage 3) to predict structurally viable skeletal configurations based on just three intuitive input parameters: domain span, domain height, and applied load.

The CNN-based models achieve high predictive accuracy, generating optimized structural images with pixel accuracies of 91.8 % and 94.6 % for cantilever and simply supported configurations, respectively (Tables 6 and 9). Acknowledging the sensitivity of CNN performance to dataset characteristics, comparative evaluations against alternative architectures (CNNC512, FFNN64 +CNN512, and CNN64 +CNN256) confirm the superior accuracy, efficiency, and robustness of the proposed CNN64 +CNN512 model.

Compared to traditional optimization approaches, the framework achieves over a sixfold reduction in computational time (Table 12) while producing ready-to-assemble designs that satisfy Eurocode 3 standards. Notably, the study introduces a spatial generalization strategy that enables CNNs to handle variable-sized input domains, resolving a major challenge in vision-based structural ML applications. To further mitigate the inherent prediction errors in machine learning models, a systematic calibration mechanism is incorporated, allowing near-perfect recovery of member dimensions and ensuring code compliance. Collectively, these advancements establish a scalable and practical foundation for applying deep learning in structural optimization, advancing the field toward fully autonomous, design-aware engineering solutions.

Replication of results

The authors believe that sufficient information has been provided to allow readers to replicate the presented results. All models and data could be shared upon a reasonable request to the corresponding author.

CRediT authorship contribution statement

Navod Jayaweera: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Meddage D. P. P:** Writing – review & editing. **Konstantinos Daniel Tsavdaridis:** Writing – review & editing, Supervision. **Kavindu Prabasha:** Writing – original draft, Visualization, Software, Investigation, Data curation. **Sumudu Herath:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Conceptualization. **Chamod Fernando:** Writing – original draft, Visualization, Software, Investigation.

Declaration of Competing Interest

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

Acknowledgments

The authors would like to acknowledge the Senate Research Council (SRC) grant SRC/LT/2025/15 provided by the University of Moratuwa in completing this research work successfully.

Annexure

A. Design provisions from Eurocode 3 for code-compliant member sizing

This section outlines the key design provisions from Eurocode 3: Design of steel structures – Part 1–1: General rules and rules for buildings [50] applied in this study for verifying member resistances under pure or combined actions, including axial tension, axial compression, bending, and shear. In the study, cross-section classes 1–3 are considered for strength checks; class 4 sections are excluded from compression and bending design in this study. Welded joints are provided. All partial safety factors follow the UK National Annex [66], with γ_{M0} and γ_{M1} taken as 1.0 unless stated otherwise.

A.1. Cross-section and member resistances

A.1.1 Tension Members

The design tensile resistance is the capacity that the member can safely carry in tension. For steel sections, EN 1993–1–1 prescribes that this resistance is the smaller of (i) the plastic resistance of the gross section and (ii) the ultimate resistance of the net section, where holes or fasteners could reduce strength. In this study, because connections are welded and do not introduce net-section reductions at holes, the design tensile resistance reduces to the plastic resistance of the gross cross-section. Denote:

$$N_{t,Rd} = N_{pl,Rd} = Af_y \Big/ \gamma_{M0} = Af_y \tag{A1}$$

where A is the gross area and f_y is the yield strength. The corresponding area requirement under a design axial tensile force $N_{t,Ed}$ is:

$$A \geq N_{t,Ed} \Big/ f_y \tag{A2}$$

A.1.2 Compression members (cross-section resistance)

For non-slender (Class 1–3) cross-sections, the design plastic resistance in compression is:

$$N_{c,Rd} = Af_y \Big/ \gamma_{M0} = Af_y \tag{A3}$$

and the minimum gross area to carry a design compressive force $N_{c,Ed}$ satisfies:

$$A \geq N_{c,Ed} \Big/ f_y \tag{A4}$$

A.1.3 Flexural buckling of members

Buckling resistance is evaluated with the reduction factor (χ) per the EN 1993–1–1 column curves. For members in Circular Hollow Sections (CHS), which provide high radii of gyration for a given mass, the design axial buckling resistance is:

$$N_{b,Rd} = \frac{\chi A f_y}{\gamma_{m1}} \tag{A5}$$

with a reduction factor:

$$\chi = \frac{1}{\varnothing + \sqrt{\varnothing^2 - \bar{\lambda}^2}} \leq 1.0 \tag{A6}$$

and

$$\varnothing = 0.5\left[1 + \alpha(\bar{\lambda} - 0.2) + \bar{\lambda}^2\right] \tag{A7}$$

The imperfection factor α can be obtained from EN 3 Table 6.1. Non-dimensional member slenderness λ ‾ is given by,

$$\bar{\lambda} = \sqrt{\frac{f_y}{\sigma_F}} \tag{A8}$$

The elastic flexural buckling stress is:

$$\sigma_F = \frac{\pi^2 EI}{A(KL)^2} \tag{A9}$$

with E the elastic modulus, I the second moment of area, L the system length, and K the effective length factor. For the fixed-ended members in the rigid frames considered, K = 0.7.

For thin-walled CHS, local shell buckling and global flexural buckling provide additional geometric constraints. The elastic local buckling stress is:

$$\sigma_L = \frac{2E}{\sqrt{3(1-\nu^2)}}\frac{t}{d} \approx 1.21E\frac{t}{d} \tag{A10}$$

The average axial stress under force $N_{c,Ed}$ is:

$$\sigma_s = \frac{N_{c,Ed}}{\pi dt} \tag{A11}$$

Equating σ_s, σ_L, and σ_F for CHS using $I/A \approx d^2/8$ form practical lower bounds for the wall thickness $t$and diameter d as follows:

$$t \geq 0.513 \quad \left(\frac{N_{c,Ed}}{E}\right)^{0.5} \tag{A12}$$

$$d \geq 0.627\left(\frac{N_{c,Ed}L^4}{E}\right)^{1/6} \tag{A13}$$

To limit excessive flexibility and avoid overly slender members in service, the following upper bound is used as per Ye et al. [67]

$$\lambda = \frac{L}{i} \leq 150 \tag{A14}$$

where $i = \sqrt{I/A}$ is the radius of gyration.

A.1.3 Bending resistance

Bending resistance depends on cross-section class. For Class 1 and 2 sections:

$$M_{c,Rd} = M_{pl,Rd} = \frac{W_{pl}f_y}{\gamma_{M0}} \tag{A15}$$

For Class 3 sections:

$$M_{c,Rd} = M_{el,Rd} = \frac{W_{el,\min}f_y}{\gamma_{M0}} \tag{A16}$$

where $W_{pl}$is the plastic section modulus and $W_{el,\min}$ is the minimum elastic section modulus about the relevant axis. Since the considered CHS in the study has the same flexural stiffness in any axis, they are susceptible to lateral torsion buckling; hence, the check was omitted.

A. 1.4 Shear resistance

In the absence of significant torsion and for standard shapes, the plastic shear resistance:

$$V_{c,Rd} = V_{pl,Rd} = \frac{A_v f_y}{\sqrt{3} \quad \gamma_{M0}} \tag{A17}$$

with $A_v$the shear area; for CHS, $A_v = 2A/\pi$. Applied shear load should be limited to the shear resistance given by A17 for pure shear situations.

A. 1.5 Bending shear interaction

Where the design shear force exceeds half the plastic shear resistance, EN 1993–1–1, 6.2.8, requires a reduction in the effective yield strength for bending. The reduced yield strength is:

$$f_{y,new} = (1-\rho)f_y \text{where} \rho = \left(\frac{2V_{Ed}}{V_{Pl,Rd}} - 1\right)^2 \tag{A18}$$

and the moment resistance is then recalculated using $f_{y,\text{new}}$. Here V_{Ed} is the design shear force and $V_{pl,Rd}$ is given by Equation (A.17)

A.1.6 Combined Bending and Axial Action

In frame members subjected to both bending and axial forces, it is essential to assess interaction. In 2D studies, bending moments are uniaxial, and for CHS with adequate restraint, lateral-torsion effects are not dominant. The interaction checks involve comparing the design axial force and bending moment to their respective resistances, ensuring that their combination remains within the interaction envelope defined by Eurocode 3 [50].

Cross-section check

$$\frac{N_{Ed}}{N_{Rd}} + \frac{M_{Ed}}{M_{c,Rd}} \leq 1 \tag{A19}$$

Member stability

$$\frac{N_{Ed}}{N_{b,Rd}} + k_{yy}\frac{M_{Ed}}{M_{b,Rd}} \leq 1 \tag{A20}$$

$$\frac{N_{Ed}}{N_{b,Rd}} + k_{zy}\frac{M_{Ed}}{M_{b,Rd}} \leq 1 \tag{A21}$$

where N_{Ed} is the design axial force, M_{Ed} is the design bending moment, N_{Rd} is axial resistance, $M_{c,Rd}$ is bending resistance, $N_{b,Rd}$ is the buckling resistance and $M_{b,Rd}$ is the design buckling resistance moment. Interaction factors k_{yy} and k_{zy} can be found in Eurocode 3 Annex A and B [50].

A.2 Check for deflection

Deflection is checked as a serviceability criterion. The maximum deflection due to serviceability load at the point with the highest deflection is considered. UK National Annex [66], recommended values for cantilever span/180 and simply supported span/200 are considered in this study.

Data availability

Data will be made available on request.

References

[1] L. Mei, Q. Wang, Structural optimization in civil engineering: a literature review, Buildings 11 (2) (2021) 66.

[2] P.W. Christensen, A. Klarbring, *An Introduction to Structural Optimization*, vol. 153, in: Solid Mechanics and Its Applications, 153, Springer, Dordrecht, 2009, https://doi.org/10.1007/978-1-4020-8666-3.

[3] R. Jankowski, A. Manguri, H. Hassan, N. Saeed, Topology, Size, and Shape Optimization in Civil Engineering Structures: A Review, Comput. Model. Eng. Sci. 142 (2) (2025) 933–971, https://doi.org/10.32604/cmes.2025.059249.

[4] S.E. Manios, N.D. Lagaros, E. Nassiopoulos, Nested topology optimization methodology for designing two-wheel chassis, Front Built Environ. 5 (2019) 34.

[5] J.-H. Zhu, W.-H. Zhang, L. Xia, Topology optimization in aircraft and aerospace structures design, Arch. Comput. Methods Eng. 23 (2016) 595–622.

[6] J. Kingman, K.D. Tsavdaridis, and V.V. Toropov, Applications of topology optimization in structural engineering, in *Civil Engineering for Sustainability and Resilience International Conference (CESARE)*, 2014.

[7] C.C. Swan, J.S. Arora, F.Y. Kocer, Continuum and ground structure topology methods for concept design of structures, Struct. Eng. 21st Century (1999) 590–593.

[8] M. Gilbert, A. Tyas, Layout optimization of large-scale pin-jointed frames, Eng. Comput. (Swans.) 20 (8) (2003) 1044–1064.

[9] T. Zegard, G.H. Paulino, GRAND3—Ground structure based topology optimization for arbitrary 3D domains using MATLAB, Struct. Multidiscip. Optim. 52 (2015) 1161–1184.

[10] M.P. Bendsoe, O. Sigmund, Topology Optimization: Theory, Methods, and Applications, Springer Science & Business Media, 2013.

[11] O. da Silva Smith, Generation of ground structures for 2D and 3D design domains, Eng. Comput. (Swans.) 15 (4) (1998) 462–500.

[12] F. He, R. Feng, Q. Cai, Topology optimization of truss structures considering local buckling stability, Comput. Struct. 294 (Apr. 2024) 107273, https://doi.org/10.1016/j.compstruc.2024.107273.

[13] Y. Lai, Q. Cai, Y. Li, J. Chen, Y.M. Xie, A new evolutionary topology optimization method for truss structures towards practical design applications, Eng. Struct. 324 (Feb. 2025) 119326, https://doi.org/10.1016/j.engstruct.2024.119326.

[14] G. Yin, X. Xiao, F. Cirak, Topologically robust CAD model generation for structural optimisation, Comput. Methods Appl. Mech. Eng. 369 (2020) 113102.

[15] L.S. Sarma, C. Mallikarachchi, S. Herath, Design-informed generative modelling of skeletal structures using structural optimization, Comput. Struct. 302 (2024) 107474.

[16] S.O. Degertekin, L. Lamberti, I.B. Ugur, Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm, Appl. Soft Comput. 79 (2019) 363–390.

[17] P. Tomšič, J. Duhovnik, Simultaneous topology and size optimization of 2D and 3D trusses using evolutionary structural optimization with regard to commonly used topologies, Adv. Mech. Eng. 6 (2014) 864807.

[18] Y. Xia, M. Langelaar, M.A.N. Hendriks, Optimization-based three-dimensional strut-and-tie model generation for reinforced concrete, Comput. Aided Civ. Infrastruct. Eng. 36 (5) (2021) 526–543.

[19] K.D. Tsavdaridis, Applications of Topology Optimization in Structural Engineering: High - Rise Buildings and Steel Components, Jordan J. Civ. Eng. 9 (3) (Jul. 2015) 335–357, https://doi.org/10.14525/jjce.9.3.3076.

[20] P.N. Poulsen, J.F. Olesen, M. Baandrup, Truss optimization applying finite element limit analysis including global and local stability, Struct. Multidiscip. Optim. 62 (1) (2020) 41–54.

[21] H. Madah, O. Amir, Truss optimization with buckling considerations using geometrically nonlinear beam modeling, Comput. Struct. 192 (2017) 233–247.

[22] Q. Cai, R. Feng, Z. Zhang, Topology optimization of truss structure considering nodal stability and local buckling stability, Structures (2022) 64–73.

[23] J. Schwarz, T. Chen, K. Shea, T. Stanković, Efficient size and shape optimization of truss structures subject to stress and local buckling constraints using sequential linear programming, Struct. Multidiscip. Optim. 58 (2018) 171–184.

[24] N.L. Pedersen, A.K. Nielsen, Optimization of practical trusses with constraints on eigenfrequencies, displacements, stresses, and buckling, Struct. Multidiscip. Optim. 25 (2003) 436–445.

[25] L. He, M. Gilbert, X. Song, A Python script for adaptive layout optimization of trusses, Struct. Multidiscip. Optim. 60 (2019) 835–847.

[26] T. Zegard, G.H. Paulino, GRAND—Ground structure based topology optimization for arbitrary 2D domains using MATLAB, Struct. Multidiscip. Optim. 50 (2014) 861–882.

[27] T. Pastore, V. Mercuri, C. Menna, D. Asprone, P. Festa, F. Auricchio, Topology optimization of stress-constrained structural elements using risk-factor approach, Comput. Struct. 224 (2019) 106104.

[28] Z. Sun, et al., Investigation of electrical resistivity for fiber-reinforced coral aggregate concrete, Constr. Build. Mater. 414 (Feb. 2024) 135011, https://doi.org/10.1016/j.conbuildmat.2024.135011.

[29] Z. Sun, Y. Li, Y. Yang, L. Su, S. Xie, Splitting tensile strength of basalt fiber reinforced coral aggregate concrete: Optimized XGBoost models and experimental validation, Constr. Build. Mater. 416 (Feb. 2024) 135133, https://doi.org/10.1016/j.conbuildmat.2024.135133.

[30] Z. Sun, et al., Electrical resistivity prediction model for basalt fibre reinforced concrete: hybrid machine learning model and experimental validation, Mater. Struct. 58 (3) (Apr. 2025) 89, https://doi.org/10.1617/s11527-025-02607-y.

[31] L.C. Nguyen, H. Nguyen-Xuan, Deep learning for computational structural optimization, ISA Trans. 103 (2020) 177–191.

[32] T.-H. Nguyen, A.-T. Vu, Prediction of Optimal Cross-Sectional Areas of Truss Structures Using Artificial Neural Networks, CIGOS 2021 Emerg. Technol. Appl. Green. Infrastruct. Proc. 6th Int. Conf. Geotech. Civ. Eng. Struct. (2022) 1897–1905.

[33] M. Yücel, G. Bekdaş, S.M. Nigdeli, Prediction of optimum 3-bar truss model parameters with an ANN model, Proc. 6th Int. Conf. Harmon. Search Soft Comput. Appl. ICHSA 2020 Istanb. (2021) 317–324.

[34] N. Nourian, M. El-Badry, M. Jamshidi, Design optimization of truss structures using a graph neural network-based surrogate model, Algorithms 16 (8) (2023) 380.

[35] S. Zheng, L. Qiu, F. Lan, TSO-GCN: A Graph Convolutional Network approach for real-time and generalizable truss structural optimization, Appl. Soft Comput. 134 (2023) 110015.

[36] I. Sosnovik, I. Oseledets, Neural networks for topology optimization, Russ. J. Numer. Anal. Math. Model. 34 (4) (2019) 215–223.

[37] S. Banga, H. Gehani, S. Bhilare, S. Patel, and L. Kara, 3D topology optimization using convolutional neural networks, *arXiv preprint arXiv:1808.07440*, 2018.

[38] S. Zheng, Z. He, H. Liu, Generating three-dimensional structural topologies via a U-Net convolutional neural network, ThinWalled Struct. 159 (2021) 107263.

[39] F.V. Senhora, H. Chi, Y. Zhang, L. Mirabella, T.L.E. Tang, G.H. Paulino, Machine learning for topology optimization: Physics-based learning through an independent training strategy, Comput. Methods Appl. Mech. Eng. 398 (2022) 115116.

[40] S. Herath, U. Haputhanthri, Topologically optimal design and failure prediction using conditional generative adversarial networks, Int J. Numer. Methods Eng. 122 (23) (2021) 6867–6887.

[41] Q. Ma, E.C. De Meter, S. Basu, TO-NODE: Topology optimization with neural ordinary differential equation, Int J. Numer. Methods Eng. 125 (7) (2024) e7428.

[42] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[43] S. Bunrit, N. Kerdprasop, K. Kerdprasop, Evaluating on the transfer learning of CNN architectures to a construction material image classification task, Int J. Mach. Learn Comput. 9 (2) (2019) 201–207.

[44] M. Flah, A.R. Suleiman, M.L. Nehdi, Classification and quantification of cracks in concrete structures using deep learning image-based techniques, Cem. Concr. Compos 114 (2020) 103781.

[45] D.W. Abueidda, S. Koric, N.A. Sobh, Topology optimization of 2D structures with nonlinearities using deep learning, Comput. Struct. 237 (2020) 106283.

[46] R. Yamashita, M. Nishio, R.K.G. Do, K. Togashi, Convolutional neural networks: an overview and application in radiology, Insights Imaging 9 (2018) 611–629.

[47] C. Tian, Y. Xu, W. Zuo, B. Zhang, L. Fei, C.-W. Lin, Coarse-to-fine CNN for image super-resolution, IEEE Trans. Multimed. 23 (2020) 1489–1502.

[48] H. Huang, R. He, Z. Sun, T. Tan, Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution, Proc. IEEE Int. Conf. Comput. Vis. (2017) 1689–1697.

[49] I. Goodfellow, Deep learning, MIT press, 2016.

[50] British Standards Institution, Eurocode 3: Design of steel structures - Part 1-1: General rules and rules for buildings, British Standards Institution, London, 2005.

[51] C. Galamhos, J. Matas, J. Kittler, Progressive probabilistic Hough transform for line detection, Proc. 1999 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (Cat. No PR00149) (1999) 554–560.

[52] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.

[53] T.Y. Zhang, C.Y. Suen, A fast parallel algorithm for thinning digital patterns, Commun. ACM 27 (3) (Mar. 1984) 236–239, https://doi.org/10.1145/357994.358023.

[54] R.C. Gonzalez, Digital image processing, Pearson education india, 2009.

[55] Python Software Foundation, Python 3.10, Python Software Foundation, 2021, 3.10.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, PyTorch: An Imperative Style, High-Performance Deep Learning Library, Proc. 33rd Int. Conf. Neural Inf. Process. Syst. (2019).

[57] NVIDIA Corporation, CUDA Toolkit, NVIDIA Corporation, 2025, 12.6.

[58] S. van der Walt, et al., scikit-image: image processing in Python, PeerJ 2 (Jun. 2014) e453, https://doi.org/10.7717/peerj.453.

[59] A. Zelinsky, Learning OpenCV—Computer Vision with the OpenCV Library (Bradski, G.R. et al.; 2008)[On the Shelf], pp. 100–100, IEEE Robot Autom. Mag. 16 (3) (Sep. 2009), https://doi.org/10.1109/MRA.2009.933612.

[60] C.E.N. EN, 1-1, Eurocode 3: Design of steel structures, Gen. rules rules Build. (1993).

[61] EN 10210–2: Hot finished structural hollow sections of non-alloy and fine grain steels - Part 2: Tolerances, dimensions and sectional properties, 2019.

[62] S. Herath, U. Haputhanthri, Topologically optimal design and failure prediction using conditional generative adversarial networks, Int J. Numer. Methods Eng. 122 (23) (Dec. 2021) 6867–6887, https://doi.org/10.1002/nme.6814.

[63] M.M. Behzadi, H.T. Ilieş, Real-Time Topology Optimization in 3D via Deep Transfer Learning, Comput. Aided Des. 135 (Jun. 2021) 103014, https://doi.org/10.1016/j.cad.2021.103014.

[64] Y. Li, Z. Sun, S. Mangalathu, Y. Li, H. Yang, W. He, Seismic damage states prediction of in-service bridges using feature-enhanced swin transformer without reliance on damage indicators, Eng. Appl. Artif. Intell. 159 (Nov. 2025) 111651, https://doi.org/10.1016/j.engappai.2025.111651.

[65] Y. Li, Z. Sun, Y. Li, H. Yang, X. Liu, W. He, A vision transformer-based method for predicting seismic damage states of RC piers: Database development and efficient assessment, Reliab Eng. Syst. Saf. 263 (Nov. 2025) 111287, https://doi.org/10.1016/j.ress.2025.111287.

[66] British Standards Institution (BSI), UK National Annex to Eurocode 3: Design of steel structures - Part 1-1: General rules and rules for buildings, British Standards Institution, London, 2008.

[67] J. Ye, P. Kyvelou, F. Gilardi, H. Lu, M. Gilbert, L. Gardner, An End-to-End Framework for the Additive Manufacture of Optimized Tubular Structures, IEEE Access 9 (2021) 165476–165489, https://doi.org/10.1109/ACCESS.2021.3132797.