Computer Methods in Applied Mechanics and Engineering 452 (2026) 118753

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Graph neural networks with hybrid local-global attention for effective prediction of mechanical response in structures

Luca Patrignani *, Silvestre T. Pinho

Department of Aeronautics, Imperial College London, Exhibition Rd, South Kensington, SW7 2AZ, London, UK

ARTICLE INFO

Keywords:
Graph neural networks
Finite element analysis
Numerical analysis
Graph transformer
Mechanical behaviour
Laminates

ABSTRACT

Graph Neural Networks (GNNs) are emerging as a transformative approach for predicting mechanical response in structures by naturally encoding unstructured finite element meshes as graphs. While traditional finite element analysis (FEA) provides trusted solutions for mechanics problems, it encounters significant computational and scalability bottlenecks. Existing machine learning approaches using regular grids fail to capture the irregular nature of real-world meshes, whereas standard GNNs suffer from over-smoothing and limited long-range information propagation that compromise accuracy. To overcome these limitations, we present a Graph Transformer methodology that implements an encoder-processor-decoder framework augmented with a frequency-controlled hybrid local-global attention mechanism, which systematically bridges local mesh connectivity with global information awareness across the computational domain. Our approach integrates seamlessly with real-world FEA workflows through an automated FEM-to-GNN pipeline that converts FE simulations to graph representations and generates training datasets directly from commercial solvers, enabling rapid deployment across various engineering applications without manual preprocessing bottlenecks. We validate our approach on open holes Carbon Fibre Reinforced Polymer (CFRP) laminate plates under various loading conditions and geometric configurations, and extend validation to nonlinear woven composites exhibiting plasticity and progressive damage under shear-dominated loadings, employing systematic hyperparameter optimisation with Tree-structured Parzen Estimator (TPE) sampling and mesh convergence studies. The linear case demonstrates excellent predictive accuracy ($R^2 = 0.98$, $RMSE = 0.00028$) with 70× speedup over equivalent-accuracy FEA and $58 - 64\%$ reduction in training and validation loss compared to standard message-passing GNN architectures without attention mechanisms, while the nonlinear case achieves $R^2 = 0.97$ ($RMSE = 0.0013$) with 660× speedup.

1. Introduction

Finite Element Analysis serves as the backbone of modern engineering design across virtually all industrial sectors, from civil infrastructure and biomedical devices to automotive and aerospace. It is a trusted mesh-based approach for addressing problems involving partial differential equations [1], offering adaptive representations and optimising resource use by assigning higher resolution to areas demanding greater accuracy. However, FEA faces scalability and computational issues in large-scale applications, especially when various configurations need separate simulations and remeshing, resulting in high computational costs [2]. This challenge is notable in multiple critical engineering domains, for instance, in damage mechanics (where structures evolve as damage propagates),

* Corresponding author.
E-mail addresses: l.patrignani@imperial.ac.uk (L. Patrignani), silvestre.pinho@imperial.ac.uk (S.T. Pinho).

https://doi.org/10.1016/j.cma.2026.118753
Received 6 October 2025; Received in revised form 9 January 2026; Accepted 10 January 2026
Available online 19 January 2026
0045-7825/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

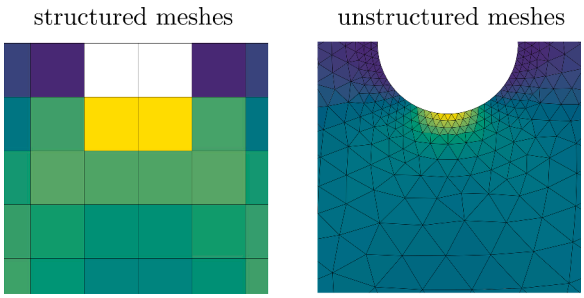


Fig. 1. Unstructured mesh (right) represents better complex structures compared to structured mesh (left).

in topology optimisation (requiring thousands of design evaluations with full structural reanalysis [3]), and in real-time structural monitoring (where instantaneous assessment is essential but computationally prohibitive with traditional methods [4]).

Within this, Machine Learning with convolutional neural networks (CNNs) presents a promising avenue for accelerating simulations. However, traditional CNNs struggle with unstructured domains due to their reliance on regular grid representations [5] (Fig. 1), making them unsuitable for irregular meshes or complex geometries, unable to accommodate adaptive mesh refinement, and requiring interpolation that introduces approximation errors.

In this context, Graph Neural Networks emerge as a natural solution to overcome these limitations by effectively representing and processing data on graphs, learning optimisable transformations on graph attributes, and exploiting the graph topology [6,7]. Demonstrating this approach in practice, Pfaff et al. (2021) [8] developed a framework for learning mesh-based simulations using graph neural networks, showing that message-passing GNNs can learn complex physical behaviour on mesh-based graphs.

Moreover, GNNs substantially improve feature engineering, which involves creating informative input representations. Unlike Convolutional Neural Networks, which require manual construction of features suited to grid-based data, GNNs automatically aggregate and integrate information from neighboring nodes to learn rich, relational, context-aware embeddings without manual intervention [7].

Graph Neural Networks have been applied to predict mechanical responses in structural components and for material modeling. Maurizi et al. (2022) [9] demonstrated that GNN surrogates can predict full-field displacement, stress, and strain —capturing plasticity, wrinkling, and lattice post-buckling— across composites and metamaterials. Perera et al. (2022) [10] proposed a staged graph-neural surrogate that predicts Mode-I/II stress-intensity factors, identifies propagating tips, and advances crack fronts. Gulakala et al. (2024) [11] introduced a generative, graph-based framework that maps FE preprocessor inputs to nodal stress/strain fields. Storm et al. (2024) [12] developed a microstructure-aware GNN that predicts microscopic strains and, via an embedded constitutive model with homogenization, recovers macroscopic stresses for concurrent multiscale analysis. Würth et al. (2024) [13] introduced physics-informed MeshGraphNets that train directly on FEM residuals, solving non-stationary, nonlinear PDEs. Parisi et al. (2024) [14] developed physics-informed graph networks that incorporate conservation laws and constitutive equations to enhance physical consistency, while Singh (2025) [15] proposed residual-constrained GNN architectures enforcing equilibrium and compatibility through auxiliary loss terms. Yacouti et al. (2025) [16] integrated image and graph-based representations—combining CNNs with GNNs—to predict microstructural stress fields in fibre composites.

Addressing boundary condition representation specifically, Fu et al. (2023) [17] introduced boundary-oriented node embeddings encoding constraint types through learned geometric features, while Black & Najafi (2022) [18] developed specialized attention mechanisms prioritizing boundary information propagation.

Despite these advances—comprehensively reviewed in [19]—the core issue is that traditional GNNs rely on local message passing: each node can only receive information from its immediate neighbors. To reach distant nodes, information must propagate through multiple hops [20]. In most engineering domains, this creates a fundamental problem: the mechanical behaviour can be influenced by distant features like boundary conditions and requires a global knowledge, but standard message-passing can only access information within a limited neighborhood edge distance, equal to the number of message-passing layers [20,21]. This limitation manifests in two specific ways. First, as information travels through many layers, it gets compressed and diluted—a phenomenon known as over-squashing [22]. Second, capturing these long-range dependencies requires deep networks, which become expensive to train and prone to over-smoothing that washes out critical local details [23].

Transformers revolutionized natural language processing by introducing attention mechanisms that allow any token in a sequence to directly communicate with any other token, regardless of their positional distance [24]; this offers a promising opportunity.

Building on this transformer paradigm, in this work, our objective is to address these limitations by developing a novel Graph Transformer architecture for applied mechanics and engineering.

This paper is organized as follows: Section 2 presents the fundamental principles of Graph Neural Networks for mesh-based physics. Section 3 details our frequency-controlled hybrid local-global attention model architecture. Section 4 introduces our automated FEA-to-GNN pipeline that seamlessly transforms real-world finite element simulations into graph-structured training data, eliminating manual preprocessing bottlenecks. Section 5 validates the methodology on strain field prediction for Carbon Fibre Reinforced Polymer (CFRP) laminates in open-hole tension specimens under linear elastic behavior. Section 6 extends validation to nonlinear woven fabric

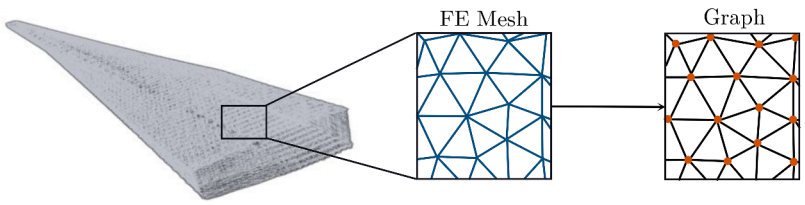


Fig. 2. Mapping of a finite element mesh (blue lines, left) to a graph representation (orange nodes and black edges, right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

composites exhibiting plasticity and progressive damage under shear-dominated loading. Section 7 discusses the results obtained, and Section 8 draws our conclusions.

2. Graph neural networks for mesh-based physics

Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a finite set of nodes (vertices) $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$, that represents entities, and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, that represents pairwise relationships between entities. Each edge $(v, w) \in \mathcal{E}$ represents a direct connection between the nodes v and w. Graphs can be categorized as undirected, where the relationships are symmetric (meaning if there is a connection from v to w, there is also one from w to v), or directed, where the relationships are asymmetric (indicating a one-way connection). Additionally, graphs can be weighted, which quantifies the strength or capacity of the connections between nodes, or unweighted, where connections are treated as binary, indicating only the presence of a direct relationship [25,26].

Real-world systems are especially defined by entities and the relationships between them. As such, graphs provide a highly versatile and powerful framework for representing complex relationships, causality, and structures that are intrinsic to scientific problems.

This framework can be applied to a wide range of fields, spanning from molecular interactions [27] to social networks [28], neural circuits [29], and fluid dynamics [30].

For example, in the context of finite element analysis, meshes can be naturally represented as graphs [31], where individual mesh vertices act as nodes and the connectivity between elements forms the edges of the graph, as shown in Fig. 2.

Unlike Convolutional Neural Networks, which function on fixed neighborhood structures, Graph Neural Networks extend the concept of convolution to irregular topologies using message-passing mechanisms [6]. In this approach, each node sends and receives messages to and from its neighbors. As illustrated in Fig. 3, the process consists of three sequential operations: encoding, processing, and decoding.

- The **encoding** transforms raw node and edge features into latent representations through learnable transformations:

$$\mathbf{h}_v^{(0)} = f_{\theta_h}(\mathbf{X}_v) \tag{1}$$

$$\mathbf{e}_{vw}^{(0)} = f_{\theta_e}(\mathbf{E}_{vw}) \tag{2}$$

  where $\mathbf{X}_v \in \mathbb{R}^{d_X}$ represents the raw feature vector for node v, $\mathbf{E}_{vw} \in \mathbb{R}^{d_E}$ denotes the raw edge features between nodes v and w, and $\mathbf{h}_v^{(0)}, \mathbf{e}_{vw}^{(0)} \in \mathbb{R}^{d_h}$ are the resulting embeddings in a shared latent space of dimension d_h. The functions f_{θ_h} and f_{θ_e} are learnable functions (typically multilayer perceptrons [MLP]) with trainable parameters θ_h and θ_e respectively.
- The **processing** propagates information through the graph via iterative message exchange between connected nodes. At each message passing layer ℓ, three sequential steps are performed:

  Step 1: Message computation – Each edge (v, w) computes a message by combining information from both connected nodes and the edge itself:

$$\mathbf{m}_{w\to v}^{(\ell)} = \phi_{\theta_m}(\mathbf{h}_v^{(\ell)}, \mathbf{h}_w^{(\ell)}, \mathbf{e}_{vw}^{(\ell)}) \tag{3}$$

  where $\mathbf{m}_{w\to v}^{(\ell)} \in \mathbb{R}^{d_h}$ is the message sent from node w to node v at layer ℓ, and ϕ_{θ_m} is a learnable function (typically an MLP) with parameters θ_m.

  Step 2: Message aggregation – Each node v collects and combines all incoming messages from its neighbors:

$$\mathbf{m}_v^{(\ell)} = \bigoplus_{w \in \mathcal{N}(v)} \mathbf{m}_{w\to v}^{(\ell)} \tag{4}$$

  where $\mathcal{N}(v) = \{w : (v, w) \in \mathcal{E}\}$ denotes the set of neighboring nodes, $\mathbf{m}_v^{(\ell)} \in \mathbb{R}^{d_h}$ is the aggregated message for node v, and $\bigoplus$ represents a permutation-invariant aggregation function such as summation or mean pooling.

  Step 3: Node state update – Each node updates its representation by combining its current state with the aggregated messages:

$$\mathbf{h}_v^{(\ell+1)} = \psi_{\theta_u}(\mathbf{h}_v^{(\ell)}, \mathbf{m}_v^{(\ell)}) \tag{5}$$

  where ψ_{θ_u} is a learnable update function (typically an MLP) with parameters θ_u that produces the new node representation $\mathbf{h}_v^{(\ell+1)}$ for the next layer. This three-step process repeats for L message-passing layers, enabling information to flow progressively across the graph structure.

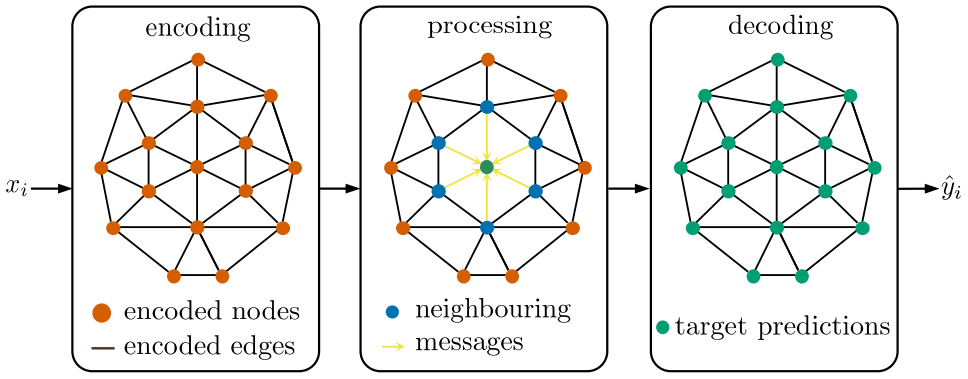


Fig. 3. Graph neural network for mesh-based physics architecture overview showing the three-stage encoder-processor-decoder framework. Input features (x_i) are transformed through learnable transformations, processed via iterative message passing, and decoded to produce target predictions ($\hat{y}_i$).

- The **decoding** maps the refined embeddings to target predictions:

$$\hat{\mathbf{y}}_v = g_{\theta_d}(\mathbf{h}_v^{(L)}) \tag{6}$$

where $\mathbf{h}_v^{(L)} \in \mathbb{R}^{d_h}$ is the final embedding for node v after L message-passing layers, $\hat{\mathbf{y}}_v \in \mathbb{R}^{d_y}$ is the predicted output (with d_y dimensions), and g_{θ_d} is an MLP with parameters θ_d.

The collective set of parameters $\Theta = \{\theta_h, \theta_e, \theta_m, \theta_u, \theta_d\}$ constitutes the trainable weights that are optimised through gradient-based learning algorithms such as stochastic gradient descent or Adam optimisation [32] by minimizing the loss function:

$$\mathcal{L}(\Theta) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \left(\hat{y}_v - y_v^{\text{FEA}}\right)^2 \tag{7}$$

where $\hat{y}_v$ represents the predicted output from the GNN at node v and y_v^{FEA} is the ground truth from finite element analysis, enabling the network to learn complex mappings from input graphs to desired outputs through supervised training on labeled data.

3. Frequency-controlled hybrid local global attention model architecture

This section describes our proposed architecture, which adopts the three-stage encoder-processor-decoder framework from MeshGraphNets [8] and extends it with a frequency-controlled hybrid attention mechanism. In contrast to standard attention-based GNN operators such as GAT [33] and GATv2 [34], which apply attention exclusively at the local edge level over one-hop neighborhoods, our mechanism systematically couples local neighborhood attention with periodic global attention to enable direct long-range interactions across the computational domain. Fig. 4 provides a comprehensive instantiation that will be used in Section 5.

The encoder transforms raw node and edge features into unified latent representations through learnable MLPs:

$$\mathbf{h}_v^{(0)} = \text{MLP}_{\theta_h}(\mathbf{X}_v), \quad \mathbf{e}_{vw}^{(0)} = \text{MLP}_{\theta_e}(\mathbf{E}_{vw}) \tag{8}$$

where each module consists of three linear layers with Parametric Rectified Linear Unit (PReLU) activation functions and dropout regularization, followed by a layer for normalization for stable training dynamics.

The processor implements an advanced message-passing framework that performs three sequential operations repeated for L layers: message computation, message aggregation, and node state updating.

For message computation, the message that node v exchanges with node w is the updated edge embedding, computed as:

$$\mathbf{m}_{w \to v}^{(\ell)} = \mathbf{e}_{wv}^{(\ell+1)} = \text{MLP}_{\theta_m}([\mathbf{h}_w^{(\ell)} \| \mathbf{h}_v^{(\ell)} \| \mathbf{e}_{wv}^{(\ell)}]) + \mathbf{e}_{wv}^{(\ell)} \tag{9}$$

where $\|$ denotes concatenation. This operation combines the current latent representations of the source node ($\mathbf{h}_w^{(\ell)}$), target node ($\mathbf{h}_v^{(\ell)}$), and their connecting edge ($\mathbf{e}_{wv}^{(\ell)}$), processing them through a multi-layer perceptron with a residual connection.

In the node updates, we exploit a **local attention** mechanism that assigns relative importance to neighbouring nodes based on learned compatibility scores. For each node v, attention coefficients are computed through:

$$\alpha_{wv}^{(\ell)} = \text{softmax}_{w \in \mathcal{N}(v)}\left(\text{MLP}_{\theta_\alpha}([\mathbf{h}_w^{(\ell)} \| \mathbf{h}_v^{(\ell)} \| \mathbf{e}_{wv}^{(\ell)}])\right) \tag{10}$$

where the softmax normalization ensures that $\sum_{w \in \mathcal{N}(v)} \alpha_{wv}^{(\ell)} = 1$. Each node v updates its representation through:

$$\mathbf{h}_v^{(\ell+1)} = \text{MLP}_{\theta_u}\left([\mathbf{h}_v^{(\ell)} \| \sum_{w \in \mathcal{N}(v)} \alpha_{wv}^{(\ell)} \mathbf{e}_{wv}^{(\ell+1)}]\right) + \mathbf{h}_v^{(\ell)} \tag{11}$$

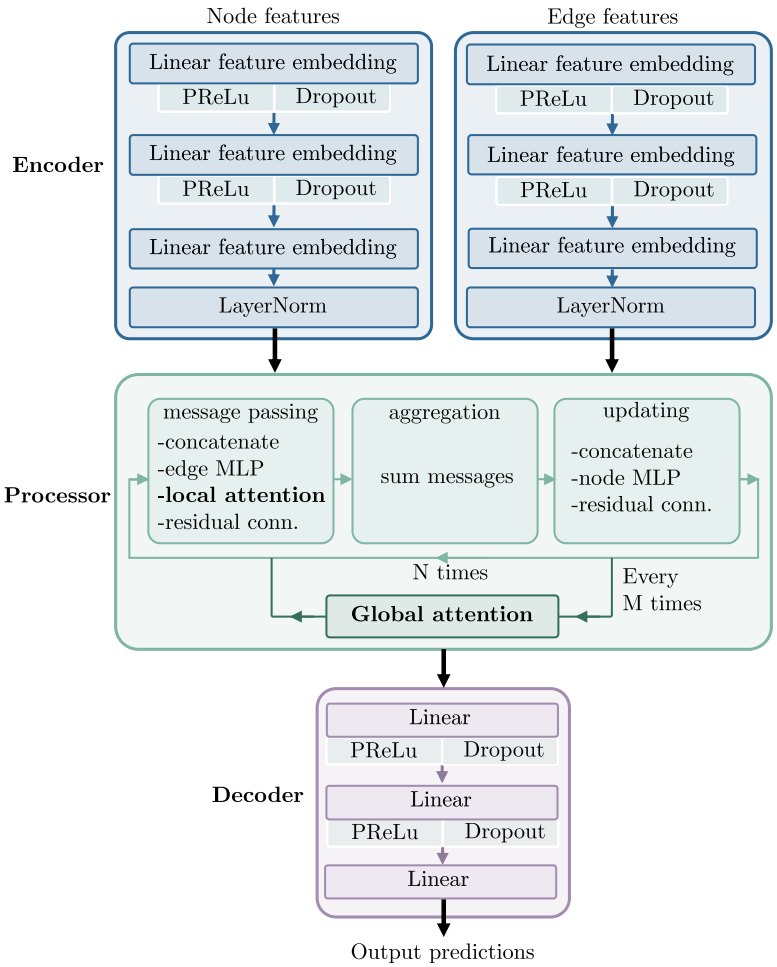


Fig. 4. Illustration of a possible instantiation of the proposed frequency-controlled hybrid local-global attention model. This particular instantiation is used in Section 5. The encoder uses parallel MLPs with PReLU activations and layer normalization to transform node and edge features. The processor employs iterative message-passing with local attention and global attention every M layers. The decoder generates predictions through an MLP. A supplementary video animation illustrating the complete inference pipeline is available online (Video S1).

processing its current state alongside the attention-weighted aggregation of neighboring messages through a multi-layer perceptron with residual connections.

Global attention is applied with a tailored frequency every M message-passing layers to enable direct long-range interactions. Following the transformer architecture [24], the attention mechanism employs the Query-Key-Value paradigm where queries represent "what information is being sought," keys represent "what information is available," and values contain "the actual information to be aggregated." This framework enables each node to selectively attend to all other nodes based on learned compatibility between queries and keys [35]. The mechanism operates as follows:

$$\mathbf{Q}^{(\ell)} = \mathbf{H}^{(\ell)}\mathbf{W}_Q, \quad \mathbf{K}^{(\ell)} = \mathbf{H}^{(\ell)}\mathbf{W}_K, \quad \mathbf{V}^{(\ell)} = \mathbf{H}^{(\ell)}\mathbf{W}_V \tag{12}$$

$$\mathbf{A}^{(\ell)} = \text{softmax}\left(\frac{\mathbf{Q}^{(\ell)}(\mathbf{K}^{(\ell)})^T}{\sqrt{d_h}}\right)\mathbf{V}^{(\ell)} \tag{13}$$

$$\tilde{\mathbf{h}}_v^{(\ell)} = \mathbf{h}_v^{(\ell)} + \alpha_{\text{global}} \cdot \mathbf{A}_v^{(\ell)} \tag{14}$$

where $\mathbf{H}^{(\ell)} = [\mathbf{h}_{v_1}^{(\ell)}, \mathbf{h}_{v_2}^{(\ell)}, \ldots, \mathbf{h}_{v_N}^{(\ell)}]^T \in \mathbb{R}^{N\times d_h}$ contains all node embeddings at layer ℓ, $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_h\times d_h}$ are learnable projection matrices for queries, keys, and values respectively, and α_{global} controls the mixing coefficient between local and global information.

The decoder maps the enriched final node embeddings to strain predictions through an MLP with PReLU activations and dropout, producing target predictions.

$$\hat{\varepsilon}_{xx,v} = \text{MLP}_{\theta_d}(\mathbf{h}_v^{(L)}) \quad \text{for } v \in \mathcal{V} \tag{15}$$

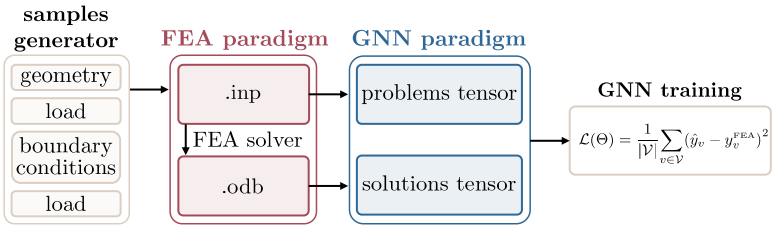


Fig. 5. Automated FEA-to-GNN pipeline that, given an engineering problem, creates a group of problems and solutions tensors that constitute the training dataset to train the Graph Neural Networks.

4. Automated FEA-to-GNN pipeline for surrogate model generation

Traditional surrogate modeling suffers from a critical preprocessing bottleneck: the extensive manual effort required to transform finite element data into machine learning-compatible formats [36]. While GNNs naturally represent mesh structures as graphs, reducing feature engineering complexity [7], a systematic framework for bridging typical FEA software with GNN architectures remains essential.

We developed an automated pipeline that transforms FEA simulations into GNN training data, systematically mapping the traditional *input-solver-output* workflow to supervised learning's *problem-solution* paradigm. Abaqus [37] was selected for this implementation due to its comprehensive material models, user-defined subroutine capabilities, interpretable input file format, and integrated Python scripting environment.

As shown in Fig. 5, the pipeline operates through integrated stages that seamlessly bridge the 'FEA paradigm' and 'GNN paradigm'. The 'samples generator' creates diverse training configurations through automated Python scripting within Abaqus CAE, systematically varying the desired design parameters, examples may include geometric parameters (hole size, plate dimensions), loading conditions (displacement magnitude, boundary locations), and material properties to span the design space and generate a large number of input files (.inp).

The 'FEA solver' executes these configurations automatically, producing output databases (.odb) that contain the complete mechanical response field. The pipeline then performs an input file parsing, extracting model and mesh features to construct the 'problems tensor', while processing the output database to extract target predictions, creating the 'solutions tensor'. More details on this operation are given in Appendix A. Finally, the 'GNN training' utilizes these paired problem-solution tensors to train the Graph Neural Network.

5. Implementation and validation of graph transformers for CFRP strain field prediction on a linear elastic laminate

5.1. Problem setup and samples generation

We apply our method presented in Section 3 for predicting longitudinal strain fields (ε_{xx}) in CFRP open-hole tension plates under uniaxial loading. As shown in Fig. 6, plate dimensions are randomly sampled with both length (ℓ) and height (h) ranging uniformly from 100 to 200 mm. Hole geometry includes two components: hole radius (r) varying uniformly between 10 and 20 mm, and position (x_c and y_c) defined by factors ranging from 0.3 to 0.7 of the plate dimensions from each edge. Applied displacement (u) magnitude varies from 1 to 2 mm. This parametric combination generates 500 distinct samples. The layup configuration maintains a symmetric $\left[0°/45°/90°/-45°\right]_s$ stacking sequence with uniform 0.25 mm ply thickness. Boundary conditions are standardized with full encastre (fixed) constraints applied to the left edge and uniaxial displacement loading on the right edge. Mesh generation employs triangular shell elements with adaptive sizing targeting approximately 1000 elements per model.

5.2. Feature engineering

Node features must incorporate both geometric and physical information relevant to strain field prediction. For this reason, each node v in the graph is characterized by the following six-dimensional feature vector:

$$\mathbf{X}_v = [x_v, y_v, \mathbb{I}_{\text{fixed}}, \mathbb{I}_{\text{hole}}, \mathbb{I}_{\text{displaced}}, u_{\text{applied}}]^T \tag{16}$$

where the spatial coordinates (x_v, y_v) provide geometric context, while the binary indicators encode boundary condition information: $\mathbb{I}_{\text{fixed}} \in \{0,1\}$ identifies nodes with encastre constraints, $\mathbb{I}_{\text{hole}} \in \{0,1\}$ marks nodes on the circular hole boundary, and $\mathbb{I}_{\text{displaced}} \in \{0,1\}$ identifies nodes subject to prescribed displacement loading. If $\mathbb{I}_{\text{displaced}}$ is equal to one, then the scalar u_{applied} represents the magnitude of applied displacement on the given node.

For each edge connecting nodes v and w, the feature vector is:

$$\mathbf{E}_{vw} = [x_v - x_w, y_v - y_w, \|\mathbf{r}_{vw}\|_2]^T \tag{17}$$

where $(x_v - x_w, y_v - y_w)$ represent the relative displacement vector components providing directional information, and $\|\mathbf{r}_{vw}\|_2$ is the Euclidean distance encoding edge length.

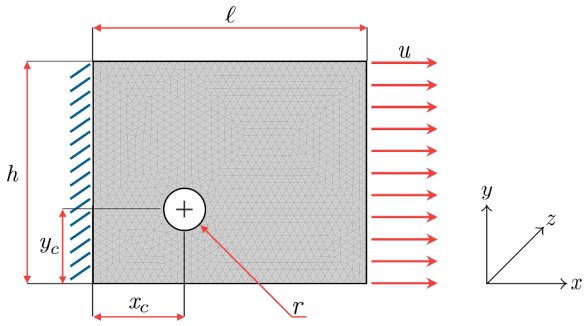


Fig. 6. CFRP open-hole tension plate configuration showing the variable features in red: length (ℓ), height (h), hole radius (r) and position (x_c and y_c), and loading conditions (u), and the fixed variable features in blue: fixed boundaries on the left side of the plate. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Optimal Hyperparameter Configuration obtained by tuning.

Hyperparameter	Value
Layers	6
Attention Frequency	3
Batch Size	4
Hidden Dimensions	48
Dropout	0.253
Optimizer	Adam
LR Decay Step	46
LR Decay Rate	0.668
Weight Decay	1.07×10^{-5}
Learning Rate	8.24×10^{-4}

Target variables represent the longitudinal strain field extracted from finite element analysis results:

$$y_v = \varepsilon_{xx,v} \tag{18}$$

Feature normalization is applied using training set statistics to ensure stable gradient propagation and improved convergence:

$$\tilde{\mathbf{X}}_v = \frac{\mathbf{X}_v - \boldsymbol{\mu}_x}{\boldsymbol{\sigma}_x}, \quad \tilde{\mathbf{E}}_{vw} = \frac{\mathbf{E}_{vw} - \boldsymbol{\mu}_e}{\boldsymbol{\sigma}_e} \tag{19}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent the mean and standard deviation computed over the dataset.

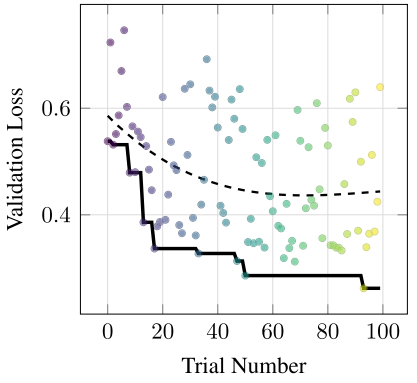
5.3. Hyperparameter optimisation

Systematic hyperparameter tuning was performed using Optuna [38] with Tree-structured Parzen Estimator sampling. The search space included the Number of message-passing Layers, Attention Frequency, Batch Size, Hidden Dimension, Dropout Rate, Optimizer, LR Decay Step, Learning Rate Decay Rate, Learning Rate, and Weight Decay. A total of 100 trials were conducted to explore the hyperparameter landscape.

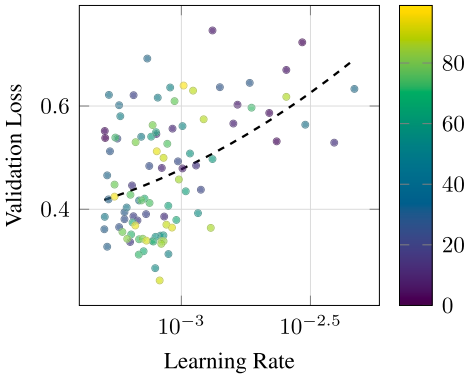
As shown in Fig. 7, the optimisation analysis revealed that the convergence history shows rapid initial improvement within 20 trials, achieving the best validation loss of 0.2621 at trial 93 (Fig. 7a), whereas the learning rate sensitivity analysis reveals optimal performance around 8×10^{-4}, with clear performance degradation at higher learning rates (Fig. 7b). Parameter importance analysis identifies learning rate and batch size as the most influential factors, contributing over 50% of performance variance (Fig. 7c). The validation loss distribution demonstrates a bimodal pattern, with well-configured trials clustering below 0.4 and suboptimal configurations above 0.5 (Fig. 7d). Table 1 presents the final optimal hyperparameter configuration identified through the systematic optimisation process.

5.4. Results

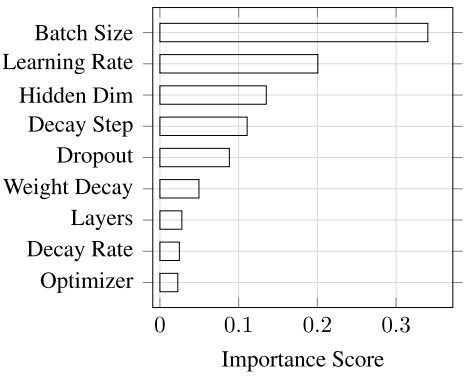
Fig. 8 shows a representative prediction, where our model effectively predicted the longitudinal strain patterns around hole boundaries as evidenced by the close correspondence between actual (Fig. 8a) and predicted (Fig. 8b) strain fields, with minimal errors (Fig. 8c). Quantitative analysis reveals excellent model accuracy: the correlation plot in Fig. 9 demonstrates a coefficient of determination $R^2 = 0.9857$ with root mean square error ($RMSE$) of 0.00028. The regression line closely follows the ideal diagonal, indicating minimal systematic bias and excellent predictive capability across the full strain range.



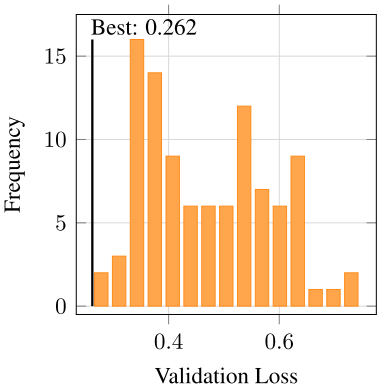
(a) Hyperparameter search consistently discovers better configurations, with gains concentrated in the early trials.



(b) A narrow learning-rate band yields the strongest generalization; moving away from it degrades performance.

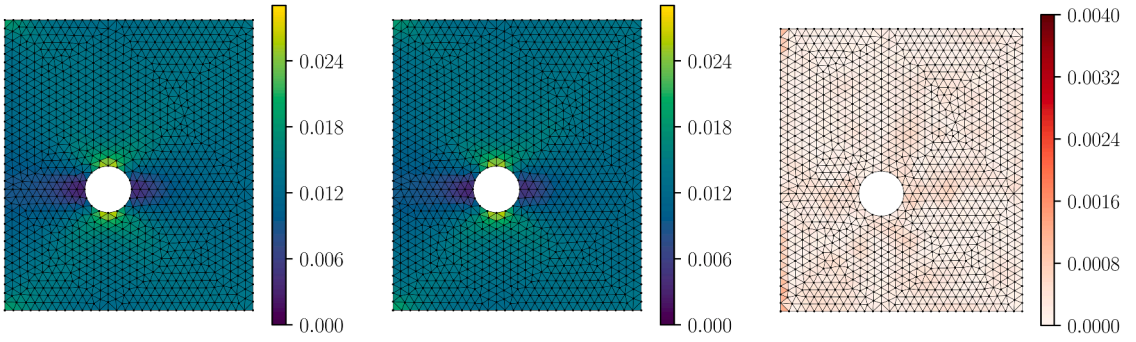


(c) Performance is driven primarily by batch size and learning rate.



(d) Validation losses vary greatly, underscoring the importance of tuning.

Fig. 7. Validation performance is governed by a narrow operating regime, largely determined by learning rate and batch size. Systematic tuning including the attention frequency leads to large, reproducible gains and a favorable accuracy-cost trade-off.



(a) Actual Strain $\varepsilon_{xx}^{\text{FEA}}$ (b) Predicted Strain ε_{xx} (c) Nominal error $\varepsilon_{xx}^{pred} - \varepsilon_{xx}^{act}$

Fig. 8. Strain field prediction for CFRP open-hole tension specimen showing minimal deviations in prediction error.

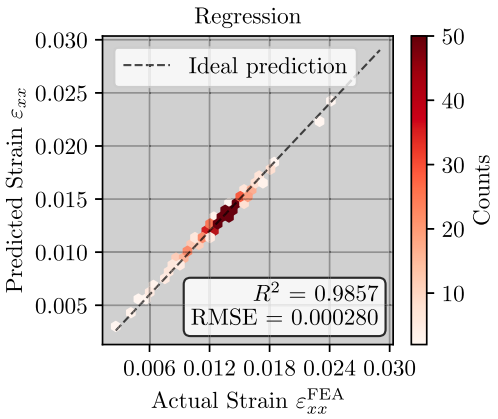
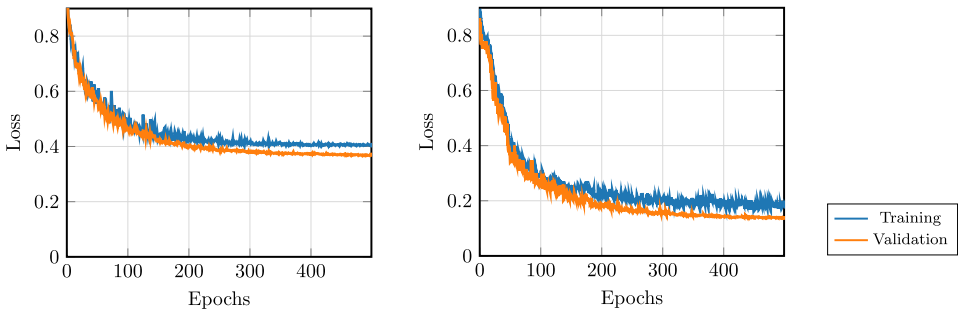


Fig. 9. Predicted vs. actual strain correlation plot demonstrating excellent model accuracy and minimal deviation from the ideal diagonal line.



(a) Standard Model **(b)** Local-Global Attention Model

Fig. 10. Training and validation loss comparison between standard and attention models clearly demonstrates the benefit of attention in enabling continued learning and improved convergence.

To highlight the benefit of the proposed frequency-controlled hybrid local-global attention mechanism, we conducted an ablation study comparing the full model against a baseline variant without attention, as shown in Fig. 10. The baseline was independently optimized through Bayesian hyperparameter search with expanded capacity, namely message-passing layers in $[6, 24]$ (versus 6 for the attention model) and hidden dimensions in $[32, 96]$ (versus 48 for the attention model), selecting 18 layers and 48 hidden dimensions. The standard model's training progression shows both training and validation losses plateau at relatively high values (training loss ≈ 0.37, validation loss ≈ 0.40) after initial convergence around epoch 200. This plateau indicates the model's limited capacity to capture complex strain field patterns due to oversmoothing and information propagation bottlenecks.

In contrast, our local-global attention model achieves training and validation losses of 0.13 and 0.17 respectively—representing a 58% and 64% reduction compared to the baseline. Despite triple the layer count, the baseline reaches RMSE $= 0.00145$ versus 0.000272 for the attention model (5.3× higher error) while requiring 16.53 ms inference versus 2.5 ms (6.6× slower). This demonstrates that attention provides fundamental architectural advantages beyond what deeper message-passing networks can achieve through hyperparameter tuning alone.

5.5. Mesh convergence study

Following McGreivy et al. [39], we conduct a mesh convergence study to establish a rigorous performance baseline. This benchmarking framework specifies two principles: (1) compare methods at equal accuracy or equal runtime, and (2) compare against efficient numerical implementations rather than suboptimal baselines.

To satisfy the first principle, we generate FEA models at varying mesh densities and compute RMSE against the finest reference solution. Fig. 11 shows this convergence analysis identifies the coarsest mesh achieving accuracy comparable to our GNN (RMSE ≈ 0.00028). The equivalent-accuracy FEA mesh requires 0.25 s per solve, while GNN inference completes in 3.5 ms—a 70× speedup at matched accuracy.

For the second principle, our FEA baseline employs efficient solution strategies: direct sparse solvers for linear equilibrium and 3-node triangular shell elements (S3 in Abaqus) suitable for thin composite plates. The complete computational investment comprises 150 s for dataset generation (500 samples on a single Intel Xeon w9-3495X CPU core) and 25 min for model training (NVIDIA RTX A2000 12GB GPU).

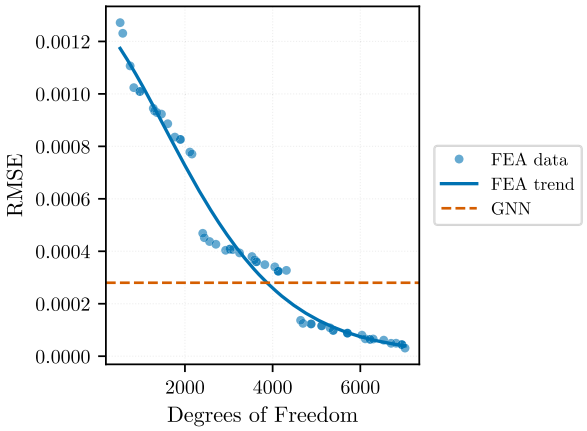


Fig. 11. Mesh convergence analysis comparing FEA solution accuracy against mesh refinement and GNN prediction accuracy. The equivalent mesh that achieves an RMSE comparable to the GNN is identified.

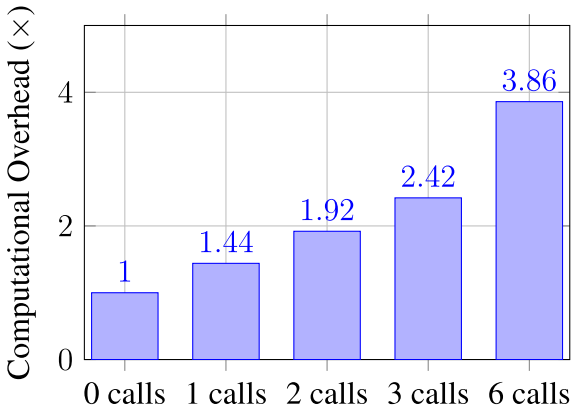


Fig. 12. Computational overhead scales linearly with attention frequency, validating theoretical complexity predictions.

Table 2
Operation count breakdown for varying attention frequencies.

Global Attention Calls	Message Passing (M)	Global Attention (M)	Total Operations (M)
0 calls	41.5	0.0	41.5
1 call	41.5	48.0	89.5
2 calls	41.5	96.0	137.5
3 calls	41.5	144.0	185.5
6 calls	41.5	288.0	329.5

5.6. Computational complexity analysis

The frequency-controlled attention mechanism introduces a trade-off between expressivity and computational cost. Each global attention layer scales as $O(N^2 \cdot d_h)$, compared to $O(E \cdot d_h^2)$ for message-passing, where N is the number of nodes and E the number of edges. Crucially, the frequency parameter M controls how often attention is invoked: with L total layers and attention applied every M layers, the number of attention calls is $K = \lfloor L/M \rfloor$, yielding total complexity $O(L \cdot E \cdot d_h^2 + K \cdot N^2 \cdot d_h)$.

Fig. 12 shows empirical training times on the meshes used in this study, confirming that overhead scales linearly with attention frequency—from 1× (no attention) to 3.9× (attention every layer). At the optimal configuration (M=3, i.e., K=2 attention calls), the overhead is 1.92×, which delivers substantial accuracy gains at acceptable cost for the mesh sizes considered here.

Table 2 provides a breakdown of operation counts, showing that attention layers contribute up to 70% of total operations at higher frequencies. The frequency parameter M thus provides direct control over the accuracy–cost trade-off: reducing attention frequency maintains the architectural benefits while controlling computational overhead. For the moderate-sized meshes in this study, the computational overhead remains practical. For larger meshes, Section 7 discusses how the frequency-controlled design is compatible with efficient attention implementations that preserve the same trade-off at scale.

Table 3
Material properties for woven fabric composite [41].

Property	Symbol	Value
Elastic	E_1, E_2	27.5 GPa
	ν_{12}	0.11
	G_{12}	2.9 GPa
Strength	$X_1^{\pm}, X_2^{\pm}$	604, 291 MPa
	S	75 MPa
Damage	$G_{If}^{\pm}$	20438 J/m^2
	$G_{IIf}^{\pm}$	4625.7 J/m^2
	$\alpha_{12}, d_{12}^{\max}$	0.32, 0.21
Plasticity	$\tilde{\sigma}_{y0}$	75 MPa
	C, p	800, 0.552
Failure	$d_{\max}, \varepsilon_{\max}^{pl}$	0.4, 0.18

6. Implementation and validation of graph transformers for strain field prediction on a nonlinear woven-fabric composite

6.1. Problem formulation and material model

To demonstrate generalization beyond linear elasticity, we validate the framework on woven-fabric composites exhibiting shear-dominated plasticity and progressive damage. The constitutive model [40] employs orthotropic damaged elasticity with independent damage variables d_1 and d_2 for fiber fracture along the warp and weft directions, and d_{12} for matrix microcracking under shear. The total strain decomposes as $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^{\mathrm{el}} + \boldsymbol{\varepsilon}^{\mathrm{pl}}$, where plasticity is confined to the shear component ($\varepsilon_{12}^{\mathrm{pl}}$). Denoting the in-plane stress vector as $\boldsymbol{\sigma} = [\sigma_{11}, \sigma_{22}, \tau_{12}]^T$, the elastic strain–stress relation is

$$\boldsymbol{\varepsilon}^{\mathrm{el}} = \begin{bmatrix} \frac{1}{(1-d_1)E_1} & -\frac{\nu_{12}}{E_1} & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{(1-d_2)E_2} & 0 \\ 0 & 0 & \frac{1}{2(1-d_{12})G_{12}} \end{bmatrix} \boldsymbol{\sigma},$$

where damage degrades only the diagonal stiffness terms. For fiber-dominated modes ($\alpha = 1, 2$), damage initiates when the stress reaches the tensile or compressive strength: $\sigma_\alpha \geq X_\alpha^+$ (tension) or $|\sigma_\alpha| \geq X_\alpha^-$ (compression). Once initiated, damage evolves according to the history variable

$$r_\alpha(t) = \max_{0 \leq \tau \leq t} \phi_\alpha(\tau), \qquad \phi_\alpha(\tau) = \frac{\tilde{\sigma}_\alpha(\tau)}{X_\alpha},$$

where $\tilde{\sigma}_\alpha = \sigma_\alpha/(1 - d_\alpha)$ is the effective stress, and the evolution law

$$d_\alpha = 1 - \frac{1}{r_\alpha} \exp\big(-A_\alpha(r_\alpha - 1)\big), \qquad \dot{d}_\alpha \geq 0.$$

The parameter A_α is computed from the fracture energy G_f^α, the elastic energy density at initiation $g_0^\alpha = X_\alpha^2/(2E_\alpha)$, and the characteristic length L_c. Yielding initiates when the shear stress reaches the initial shear yield strength $\tilde{\sigma}_{y0}$, and follows the hardening law

$$|\tau_{12}| = \tilde{\sigma}_{y0} + C\big(\bar{\varepsilon}_{12}^{\mathrm{pl}}\big)^p,$$

where C and p are hardening parameters (Table 3). Shear damage initiates when the effective shear stress reaches the shear strength S, i.e. when $\phi_{12} = \tilde{\sigma}_{12}/S = 1$, and evolves according to $r_{12}(t) = \max_{0 \leq \tau \leq t} \phi_{12}(\tau)$ via

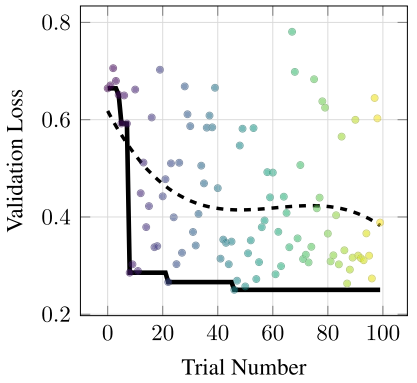
$$d_{12} = \min(\alpha_{12} \ln r_{12},\ d_{12}^{\max}).$$

To maximize shear nonlinearity, we employ a $[45/-45]_s$ symmetric layup with four 0.25 mm plies. Under uniaxial tension applied at the laminate level, this off-axis configuration induces shear-dominated loading within each ply's local coordinate system, activating both plasticity and progressive damage mechanisms. The material properties used for this case study [41] are summarized in Table 3.

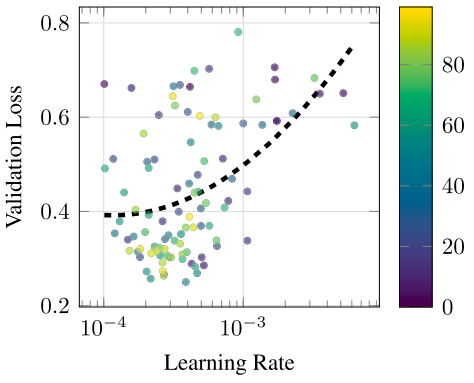
The geometric configuration mirrors Section 5; we generated 500 samples using the automated pipeline described in Section 4 with identical feature engineering.

6.2. Hyperparameter optimization

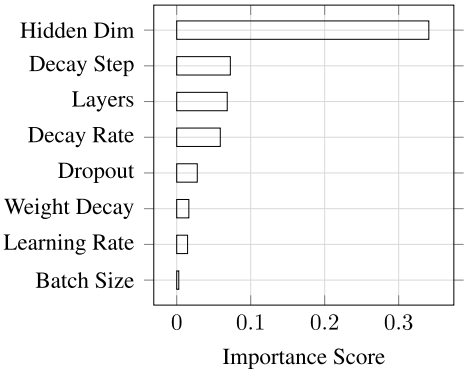
We performed systematic hyperparameter optimization following the approach established in Section 5.3, employing Optuna with Tree-structured Parzen Estimator sampling over the same search space. Fig. 13 shows the optimization achieved the best validation loss of 0.250 at trial 46. Table 4 presents the optimal configuration.



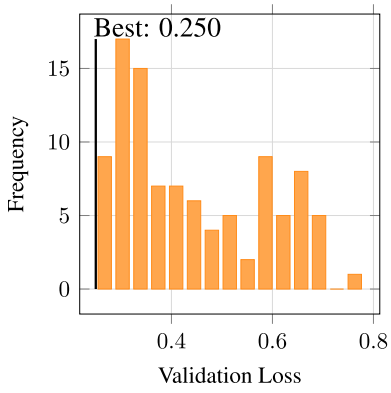
(a) Optimization converges rapidly, reaching the best configuration by trial 46.



(b) Optimal performance clusters around 2–5×10^{-4}; higher rates degrade generalization.



(c) Performance is driven primarily by hidden dimensions.



(d) Validation losses span a wide range; well-tuned trials cluster below 0.35.

Fig. 13. Validation performance is governed by a narrow operating regime, largely determined by hidden dimensions. Systematic tuning including the attention frequency leads to large, reproducible gains and a favorable accuracy-cost trade-off.

Table 4
Optimal Hyperparameter Configuration obtained by tuning.

Hyperparameter	Value
Layers	5
Attention Frequency	4
Batch Size	4
Hidden Dimensions	128
Dropout	0.218
Optimizer	Adam
LR Decay Step	55
LR Decay Rate	0.701
Weight Decay	9.73×10^{-5}
Learning Rate	3.88×10^{-4}

6.3. Results

Fig. 14 shows a representative prediction demonstrating close correspondence between actual (Fig. 14a) and predicted (Fig. 14b) strain fields, with minimal errors (Fig. 14c). Quantitative analysis reveals $R^2 = 0.9707$ with $RMSE = 0.0013$ (Fig. 15). The regression line closely follows the ideal diagonal, indicating minimal systematic bias across the full strain range.

Moreover, Fig. 16 demonstrates the model's capability to predict different strain localization patterns across diverse scenarios.

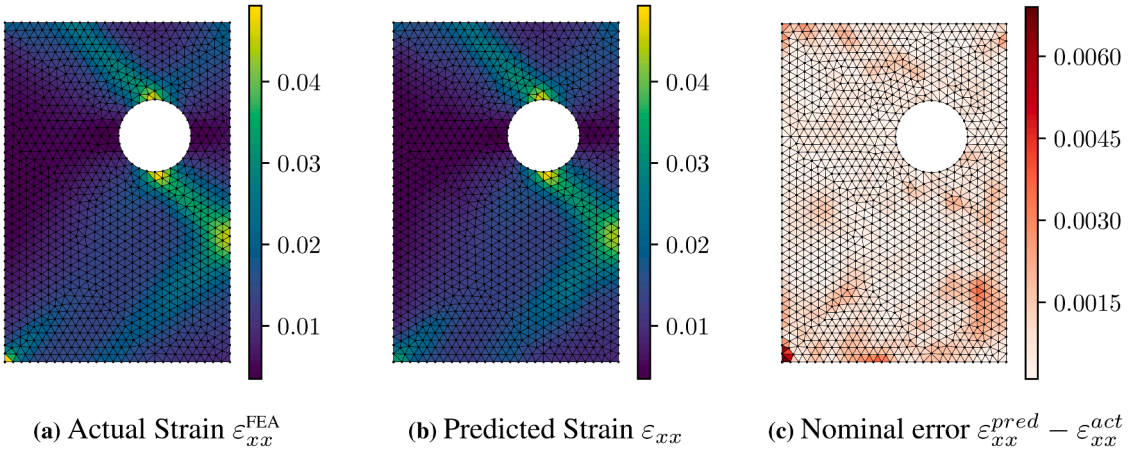


Fig. 14. Strain field prediction for a nonlinear composite woven showing minimal deviations in prediction error.

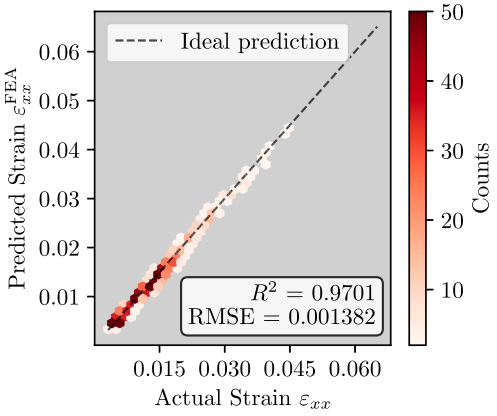


Fig. 15. Predicted vs. actual strain correlation plot demonstrating excellent model accuracy and minimal deviation from the ideal diagonal line.

6.4. Mesh convergence study

Following the same benchmarking methodology (McGreivy et al. [39]), Fig. 17 shows RMSE versus degrees of freedom against the finest reference solution. The equivalent-accuracy FEA mesh (RMSE ≈ 0.0013) requires 9 s per solve, while GNN inference completes in 5.03 ms—a 660× speedup at matched accuracy.

Our FEA baseline employs efficient nonlinear solution strategies: Newton-Raphson iteration with line search for equilibrium convergence, automatic time stepping based on convergence rates, and S3 triangular shell elements providing thin-shell kinematics with plasticity and continuum damage mechanics in Abaqus/Standard. The complete computational investment comprises 75 min for dataset generation (500 samples on a single Intel Xeon w9-3495X CPU core) and 55 min for model training (NVIDIA RTX A2000 12GB GPU).

7. Discussion

The model attains excellent predictive accuracy (Fig. 15) while preserving high fidelity and significantly outperforms standard message-passing GNNs (Fig. 10).

Beyond accuracy, the computational analysis (Section 5.6) shows that applying global attention every M layers with low mixing weights yields a favorable accuracy–cost trade-off (Fig. 12), consistent with attention dominating operations at higher frequencies (Table 2). Moreover, for scaling to larger structures, as we use standard dot-product attention, the architecture allows for:

- integrating kernel-optimized backends such as FlashAttention-2 [42], which reduces memory access while maintaining full attention computation (achieving $2-5\times$ speedups), and
- approximate attention patterns (e.g., sparse/blockwise [43,44], adaptive spans [45], Longformer [46], Linformer [47], Reformer [48], Ring Attention [49]) that strategically limit candidate selection while preserving essential long-range dependencies based on a learned compatibility.

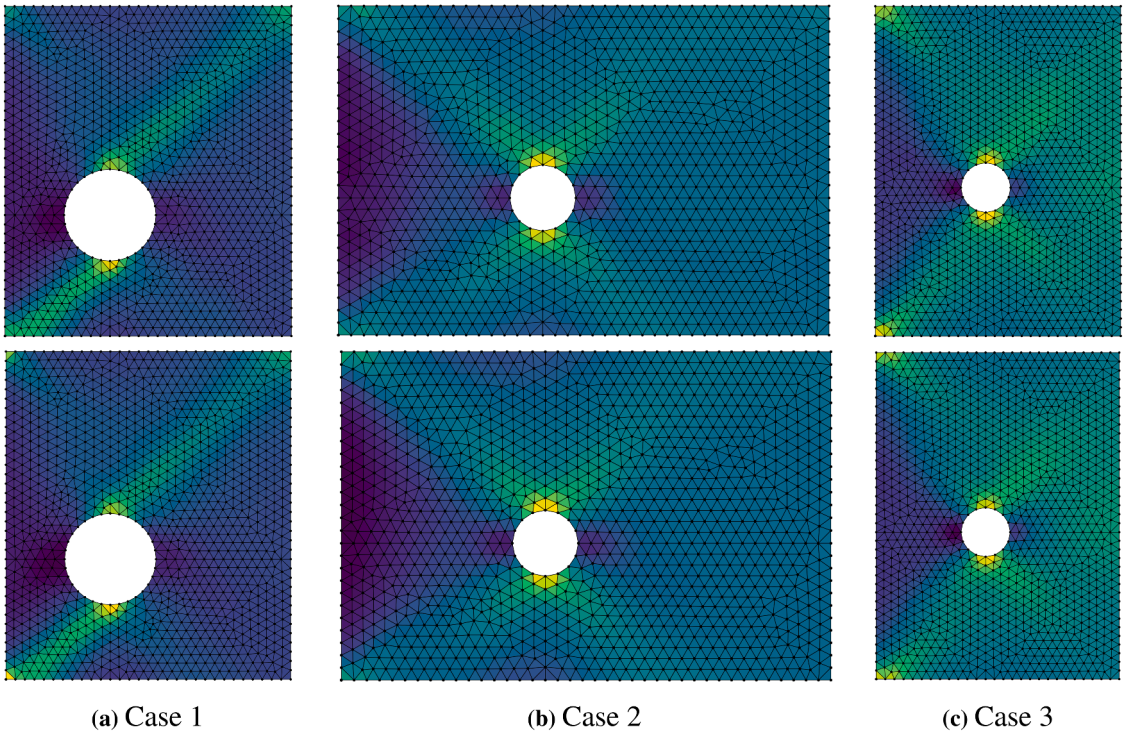


Fig. 16. Predicted (top row) versus actual (bottom row) strain fields across three test samples. The model is able to predict different localization patterns in different scenarios. Colour scales omitted for clarity; colours indicate relative strain patterns only.

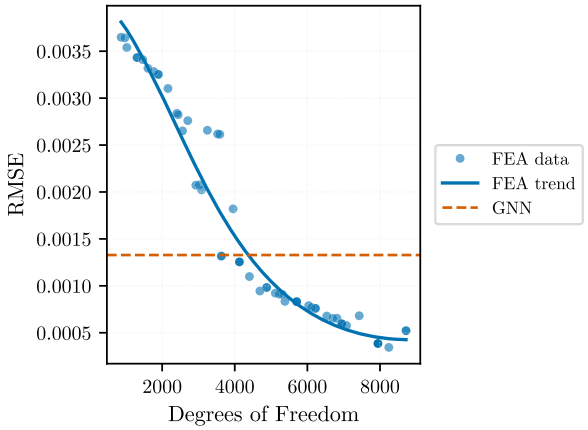


Fig. 17. Mesh convergence analysis comparing FEA solution accuracy against mesh refinement and GNN prediction accuracy. The equivalent mesh that achieves an RMSE comparable to the GNN is identified.

The potential to use drop-in kernel backends and compatible architectural variants supports the conclusion that the proposed frequency-controlled scheme can effectively scale to larger structures.

Moreover, the field-based training paradigm (Fig. 5) reveals a key architectural advantage: because the model learns from displacement/strain fields as supervision targets, any experimentally measured field data can be directly used for training without preprocessing or mesh alignment constraints. This design choice—validated through our FEA training—positions the methodology as naturally compatible with experimental measurements, providing a direct pathway to experimental training and validation.

In this work we deliberately focus the validation on single-component strain prediction (ε_{xx}) under fixed boundary conditions on 2D plate geometries, using quasi-static implicit solutions for the linear case and explicit dynamic solutions for the nonlinear case. This design choice allows us to isolate and assess the core contribution of the proposed architecture. Given the demonstrated effectiveness across diverse material behaviors—from linear elasticity to nonlinear plasticity and progressive damage—the framework is well-positioned for extensions to multi-component fields, 3D domains, variable boundary conditions, physics-informed constraints, and

auto-regressive formulations for path-dependent loading. Alternative approaches based on neural operators [50] offer complementary strategies for cross-condition generalization through continuous function-space representations, though they require coordinate transformations that our graph-based method avoids through direct mesh compatibility. Pursuing these directions would require dedicated developments, such as, but not limited to, expanded decoders, conditional encodings, equilibrium-residual constraints, or recurrent formulations.

8. Conclusions

GNNs are a powerful tool that allows using ML on unstructured meshes represented as graphs; however, this mechanism fails in deep meshes as the information propagated is diluted and the overall predictions can become oversmoothed, leading to poor performance. To overcome these limitations, we propose a novel Graph Transformer methodology that augments message passing GNN with a frequency-controlled hybrid local-global attention mechanism, to bridge local mesh connectivity with global information awareness across the computational domain. Our key conclusions are:

- the frequency-controlled hybrid local-global attention mechanism addresses information propagation bottlenecks in Graph Neural Networks for predicting mechanical response in materials, demonstrated by a $58-64\%$ reduction in training and validation loss compared to independently optimized standard message-passing GNNs without attention;
- GNNs can successfully predict strain fields in complex materials (laminated, orthotropic) while conforming to complex shapes, achieving excellent predictive accuracy for both linear elastic laminates ($R^2 = 0.98$, $RMSE = 0.00028$, $70\times$ speedup) and nonlinear woven composites exhibiting plasticity and progressive damage ($R^2 = 0.97$, $RMSE = 0.0013$, $660\times$ speedup), at equivalent accuracy compared to finite element analysis, while maintaining high fidelity in capturing strain concentrations around geometric discontinuities;
- the frequency-controlled hybrid attention offers a tunable accuracy-cost trade-off; in combination with efficient attention implementations, it enables in principle, practical scaling to larger structures;
- the automated FEM-to-GNN pipeline eliminates preprocessing bottlenecks and enables seamless integration with existing commercial finite element workflows, removing key adoption barriers for engineering disciplines; and
- the architecture is directly compatible with experimentally measured field data on arbitrary meshes, with demonstrated success on 2D geometries motivating future extensions such as multi-component fields, 3D domains, variable boundary conditions, and physics-informed constraints.

We consider the methodology proposed herein to be a promising strategy for ML applications in mechanics and engineering, and a catalyst for democratizing accelerated mechanical analysis, iterative design, and ultimately accelerating innovation across engineering disciplines.

CRediT authorship contribution statement

Luca Patrignani: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Visualization, Writing – original draft, Writing – review & editing; **Silvestre T. Pinho:** Conceptualization, Investigation, Methodology, Project administration, Supervision, Validation, Visualization, Writing – review & editing.

Data Availability

The complete implementation, including the Graph Neural Network architecture with hybrid attention mechanisms, automated FEA-to-GNN pipeline with Abaqus integration, trained model weights for both linear and nonlinear cases, representative training datasets (500 samples each), and comprehensive documentation, is publicly available at https://github.com/EngLPat/local-global-graph-transformer and on Zenodo at https://doi.org/10.5281/zenodo.18260333.

Declaration of competing interest

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

Acknowledgments

This study was supported by funding from the Department of Aeronautics, Imperial College London. The authors acknowledge funding from EPSRC project EP/W022508/1.

Appendix A. Details of FEA-to-GNN pipeline

This section provides a detailed implementation of the automated pipeline's core operations: input file parsing for graph construction, FEA solver execution, and output processing for target tensor generation.

The input file parsing stage transforms Abaqus .inp files into graph-structured data through systematic keyword-based information extraction, for instance:

- From `*NODE` keywords, we extract vertex coordinates (x_v, y_v, z_v) that define the spatial positioning of node v in the computational domain.
- From `*ELEMENT` keywords, we extract connectivity patterns that establish relationships between nodes (e.g., nodes v and w) to form graph edges (v, w). For example, a quadrilateral element connecting nodes $\{1, 2, 3, 4\}$ is mapped to generate edges $\{(1, 2), (2, 3), (3, 4), (4, 1)\}$.
- From `*BOUNDARY` keywords, we extract constraint and loading specifications that encode boundary conditions (e.g., node v fixed in x-direction with prescribed displacement).

The FEA solver execution stage processes all generated input configurations through Abaqus/Standard, producing output databases (.odb files) containing, for instance, displacements, strains, and stresses across the structural domain.
The output processing stage maps target fields from output dataset files to the solutions tensor through Python scripting in Abaqus with systematic database queries.

This integrated pipeline generates three fundamental tensor structures that constitute the machine learning dataset:

- **Node tensor:** $\mathcal{T}_{\mathcal{N}} \in \mathbb{R}^{S\times N_{\max}\times d_X}$ encodes nodal information across all S samples, where $N_{\max}$ represents the maximum number of nodes across all samples and d_X denotes the node feature dimensionality. For sample s:

$$\mathcal{T}_{\mathcal{N}}[s,:,:] = \begin{bmatrix} X^{(s)}_{1,1} & X^{(s)}_{1,2} & \cdots & X^{(s)}_{1,d_X} \\ X^{(s)}_{2,1} & X^{(s)}_{2,2} & \cdots & X^{(s)}_{2,d_X} \\ \vdots & \vdots & \ddots & \vdots \\ X^{(s)}_{N_{\max},1} & X^{(s)}_{N_{\max},2} & \cdots & X^{(s)}_{N_{\max},d_X} \end{bmatrix} \tag{A.1}$$

  where $X^{(s)}_{v,i}$ represents spatial coordinates, geometric indicators, and boundary condition specifications for node v in sample s. Configurations with fewer than $N_{\max}$ nodes employ zero-padding for tensor uniformity.
- **Edge tensor:** $\mathcal{T}_{\mathcal{E}} \in \mathbb{R}^{S\times M_{\max}\times d_E}$ represents connectivity and geometric relationships across all samples, where $M_{\max}$ denotes the maximum number of edges and d_E specifies edge feature dimensionality. For sample s:

$$\mathcal{T}_{\mathcal{E}}[s,:,:] = \begin{bmatrix} i^{(s)}_1 & j^{(s)}_1 & \|\mathbf{r}_1\|^{(s)} & r^{(s)}_{x,1} & r^{(s)}_{y,1} & \cdots \\ i^{(s)}_2 & j^{(s)}_2 & \|\mathbf{r}_2\|^{(s)} & r^{(s)}_{x,2} & r^{(s)}_{y,2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ i^{(s)}_{M_{\max}} & j^{(s)}_{M_{\max}} & \|\mathbf{r}_{M_{\max}}\|^{(s)} & r^{(s)}_{x,M_{\max}} & r^{(s)}_{y,M_{\max}} & \cdots \end{bmatrix} \tag{A.2}$$

  where indices i and j specify the nodes forming edge k, $\|\mathbf{r}_k\|^{(s)}$ encodes edge length, and $(r^{(s)}_{x,k}, r^{(s)}_{y,k})$ represent directional components. Samples with fewer edges utilize zero-padding for tensor uniformity.
- **Target tensor:** $\mathcal{T}_{\mathcal{Y}} \in \mathbb{R}^{S\times N_{\max}\times d_y}$ contains mechanical response fields extracted from FEA output databases, where d_y represents the dimensionality of output variables. For sample s:

$$\mathcal{T}_{\mathcal{Y}}[s,:,:] = \begin{bmatrix} \hat{y}^{(s)}_{1,1} & \cdots & \hat{y}^{(s)}_{1,d_y} \\ \hat{y}^{(s)}_{2,1} & \cdots & \hat{y}^{(s)}_{2,d_y} \\ \vdots & \ddots & \vdots \\ \hat{y}^{(s)}_{N_{\max},1} & \cdots & \hat{y}^{(s)}_{N_{\max},d_y} \end{bmatrix} \tag{A.3}$$

  representing ground truth field values.

The framework also incorporates the capability to produce structured mesh datasets suitable for CNN algorithms such as U-Net [51]. This allows for an objective evaluation of performance metrics between methodologies using ML on structured (CNNs) and unstructured (GNN) meshes.

The automated architecture significantly reduces the time required for extensive manual preprocessing from several weeks to a matter of hours, thereby facilitating swift investigations into mechanics challenges utilizing consistent GNN models.

The modular design facilitates extension to other FEA packages, maximizing practical impact across real-world workflows.

Supplementary material

Video S1: Animation of the GNN inference pipeline for strain field prediction. The video illustrates: (1) finite element mesh input, (2) encoding of node features into latent space representations, (3) iterative message passing with local attention, (4) periodic global attention, and (5) decoding to produce the predicted longitudinal strain field (ε_{xx}). Supplementary material associated with this article can be found in the online version at 10.1016/j.cma.2026.118753

References

[1] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The finite element method: its basis and fundamentals, Elsevier, 2005.
[2] S.T. Pinho, M. Matos, R.O.S.S. Costa, A. Ibbotson, M. Ostergaard, Enabling multiscale analysis of very large composite structures, in: 8th ECCOMAS Thematic Conference on the Mechanical Response of Composites, Gothenburg, Sweden, 2021.

[3] M.P. Bendsøe, O. Sigmund, Topology optimization: theory, methods, and applications, Springer Science & Business Media, 2003.
[4] K. Worden, G. Manson, Structural health monitoring: a machine learning perspective, Phil. Trans. Royal Soc. A: Mathem., Phys. Eng. Sci. 365 (1851) (2007) 515–537.
[5] M. Rezasefat, J.D. Hogan, A finite element-convolutional neural network model (FE-CNN) for stress field analysis around arbitrary inclusions, Mach. Learn.: Sci. Technol. 4 (4) (2023) 045052.
[6] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Networks 20 (1) (2008) 61–80.
[7] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[8] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.
[9] M. Maurizi, C. Gao, F. Berto, Predicting stress, strain and deformation fields in materials and structures with graph neural networks, Sci. Rep. 12 (1) (2022) 21834.
[10] R. Perera, D. Guzzetti, V. Agrawal, Graph neural networks for simulating crack coalescence and propagation in brittle materials, Comput. Methods Appl. Mech. Eng. 395 (2022) 115021.
[11] R. Gulakala, V. Bhaskaran, M. Stoffel, A generative learning and graph-based framework for computing field variables in finite element simulations, Comput. Methods Appl. Mech. Eng. 428 (2024) 117111.
[12] J. Storm, I.B. Rocha, F.P. van der Meer, A microstructure-based graph neural network for accelerating multiscale simulations, Comput. Methods Appl. Mech. Eng. 427 (2024) 117001.
[13] T. Würth, N. Freymuth, C. Zimmerling, G. Neumann, L. Kärger, Physics-informed meshgraphnets (PI-MGNs): neural finite element solvers for non-stationary and nonlinear simulations on arbitrary meshes, Comput. Methods Appl. Mech. Eng. 429 (2024) 117102.
[14] F. Parisi, S. Ruggieri, R. Lovreglio, M.P. Fanti, G. Uva, On the use of mechanics-informed models to structural engineering systems: application of graph neural networks for structural analysis, in: Structures, 59, Elsevier, 2024, p. 105712.
[15] N.K. Singh, A comprehensive evaluation of graph neural networks and physics informed learning for surrogate modelling of finite element analysis, arXiv preprint arXiv:2510.15750 (2025).
[16] M. Yacouti, M. Shakiba, Integrated convolutional and graph neural networks for predicting mechanical fields in composite microstructures, Compos. Part A: Appl. Sci. Manufac. 190 (2025) 108618.
[17] X. Fu, F. Zhou, D. Peddireddy, Z. Kang, M.B.-G. Jun, V. Aggarwal, An finite element analysis surrogate model with boundary oriented graph embedding approach for rapid design, J. Comput. Des. Eng. 10 (3) (2023) 1026–1046.
[18] N. Black, A.R. Najafi, Learning finite element convergence with the multi-fidelity graph neural network, Comput. Methods Appl. Mech. Eng. 397 (2022) 115120.
[19] Y. Zhao, H. Li, H. Zhou, H.R. Attar, T. Pfaff, N. Li, A review of graph neural network applications in mechanics-related domains, Artif. Intell. Rev. 57 (11) (2024) 315.
[20] F. Hendriks, V. Menkovski, M. Doškář, M.G.D. Geers, O. Rokoš, Similarity equivariant graph neural networks for homogenization of metamaterials, Comput. Methods Appl. Mech. Eng. 439 (2025) 117867.
[21] R. Perera, V. Agrawal, Multiscale graph neural networks with adaptive mesh refinement for accelerating mesh-based simulations, Comput. Methods Appl. Mech. Eng. 429 (2024) 117152.
[22] U. Alon, E. Yahav, On the bottleneck of graph neural networks and its practical implications, Int. Conf. Learn. Represent. (2021).
[23] K. Oono, T. Suzuki, Graph neural networks exponentially lose expressive power for node classification, Int. Conf. Learn. Represent. (2020).
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[25] R. Diestel, Graph theory, Springer-Verlag GmbH Germany, 2017.
[26] D.B. West, et al., Introduction to graph theory, 2, Prentice hall Upper Saddle River, 2001.
[27] A.-L. Barabasi, Z.N. Oltvai, Network biology: understanding the cell's functional organization, Nat. Rev. Genet. 5 (2) (2004) 101–113.
[28] M. Newman, Networks, Oxford university press, 2018.
[29] E. Bullmore, O. Sporns, Complex brain networks: graph theoretical analysis of structural and functional systems, Nat. Rev. Neurosci. 10 (3) (2009) 186–198.
[30] K. Taira, A.G. Nair, Network-Based analysis of fluid flows: progress and outlook, Prog. Aerosp. Sci. 131 (2022) 100823.
[31] C. Jiang, N.-Z. Chen, Graph neural networks (GNNs) based accelerated numerical simulation, Eng. Appl. Artif. Intell. 123 (2023) 106370.
[32] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).
[34] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, arXiv preprint arXiv:2105.14491 (2021).
[35] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).
[36] Z. Miao, A. Vasileiou, H. Yin, L. Margetts, Surrogate model development for digital experiments in welding, J. Visual. Experiments (JoVE) (217) (2025) e67576.
[37] M. Smith, ABAQUS/Standard User's Manual, Version 6.9, Dassault Systèmes Simulia Corp, United States, United States, 2009.
[38] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: a next-generation hyperparameter optimization framework, in: Proceedings of the 25Th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2623–2631.
[39] N. McGreivy, A. Hakim, Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations, Nat. Mach. Intell. 6 (10) (2024) 1256–1269.
[40] VUMAT for Fabric Reinforced Composites, Technical Report, Dassault Systèmes, 2008. ABAQUS/Explicit user material documentation.
[41] S. Sridharan, Performance evaluation of two progressive damage models for composite laminates under high velocity projectile impact (2017).
[42] T. Dao, Flashattention-2: faster attention with better parallelism and work partitioning, arXiv preprint arXiv:2307.08691 (2023).
[43] R. Child, S. Gray, A. Radford, I. Sutskever, Generating long sequences with sparse transformers, arXiv preprint arXiv:1904.10509 (2019).
[44] J. Qiu, H. Ma, O. Levy, S.W.-t. Yih, S. Wang, J. Tang, Blockwise self-attention for long document understanding, arXiv preprint arXiv:1911.02972 (2019).
[45] S. Sukhbaatar, E. Grave, P. Bojanowski, A. Joulin, Adaptive attention span in transformers, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (2019) 331–335.
[46] I. Beltagy, M.E. Peters, A. Cohan, Longformer: the long-document transformer, arXiv preprint arXiv:2004.05150 (2020).
[47] S. Wang, B.Z. Li, M. Khabsa, H. Fang, H. Ma, Linformer: self-attention with linear complexity, arXiv preprint arXiv:2006.04768 (2020).
[48] N. Kitaev, Ł. Kaiser, A. Levskaya, Reformer: the efficient transformer, arXiv preprint arXiv:2001.04451 (2020).
[49] H. Liu, M. Zaharia, P. Abbeel, Ring attention with blockwise transformers for near-infinite context, arXiv preprint arXiv:2310.01889 (2023).
[50] Z. Jin, B. Zheng, C. Kim, G.X. Gu, Leveraging graph neural networks and neural operator techniques for high-fidelity mesh-based physics simulations, APL Mach. Learn. 1 (4) (2023).
[51] L. Liang, M. Liu, C. Martin, W. Sun, A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis, J. Royal Soc. Interf. 15 (138) (2018) 20170844.