LIFTGCN: EFFICIENT ENERGY-PRESERVING GRAPH LEARNING VIA JOUKOWSKI SPECTRAL LIFTING FOR FINITE ELEMENT STRESS PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Finite element stress fields often exhibit strong local non-smoothness, where stress concentrations near holes, notches, and loading regions induce sharp spatial gradients and high-frequency graph components. Although graph neural networks naturally operate on irregular finite element meshes, conventional message passing is inherently smoothing and progressively attenuates such high-frequency information. Unitary propagation alleviates this problem by preserving spectral magnitudes, but typically relies on matrix functions and high-order approximations with $O(Ked)$ propagation complexity. We propose LiftGCN, an efficient spectrally stable graph network based on Joukowski spectral lifting. LiftGCN maps the real spectrum of a normalized graph operator onto the unit circle through the Joukowski relation and realizes the resulting spectral transformation as a simple second-order recurrence, avoiding matrix exponentials, eigendecomposition, and high-order polynomial truncation. We show that the linear Joukowski backbone has unit-modulus characteristic roots and admits an energy-preserving structure under a positive-definite metric, preventing exponential attenuation of graph-frequency components with depth. Each layer requires only one sparse neighborhood aggregation, yielding $O(ed)$ propagation complexity, while lightweight local nonlinear residuals provide expressive feature transformations. Experiments on finite element stress prediction demonstrate that LiftGCN achieves competitive overall accuracy while improving reconstruction of stress concentrations and local high-gradient structures with substantially reduced computational cost.

1 INTRODUCTION

Finite element analysis (FEA) is widely used to predict mechanical responses, but repeated high-resolution simulations can be expensive (Bathe et al., 1996; Zienkiewicz et al., 2013; Jayasinghe et al., 2025). This cost motivates learned surrogates for design optimization and structural evaluation (Li et al., 2021; Giacomini & Díez, 2026; Liang et al., 2018). Stress prediction is particularly demanding: a field may be smooth over much of the domain yet contain sharp concentrations near holes, notches, interfaces, and loading regions (Sun & Chen, 2024). These local responses are important for assessing structural failure, so a useful surrogate must recover both the overall field and its steep spatial variations (Zhu et al., 2022; Braun et al., 2020). Figure 1 illustrates this challenge.

Graph neural networks (GNNs) operate directly on irregular finite element meshes, using connectivity to exchange information between nodes (Scarselli et al., 2008; Wu et al., 2020). MeshGraphNets and related models have demonstrated their value for physical simulation and structural prediction (Pfaff et al., 2021; Maurizi et al., 2022). However, repeated neighborhood aggregation tends to smooth node features and attenuate graph-frequency components (Li et al., 2018; Hoang et al., 2021; Roth & Liebig, 2024). In stress fields, these components can encode mechanically meaningful concentrations and gradients. Smoothing them can therefore remove precisely the local details that a surrogate needs to retain.

Several approaches seek less dissipative propagation. Graph-Coupled Oscillator Networks (GraphCON) introduce second-order dynamics, using inertia to reduce the collapse of deep representations (Rusch et al., 2022). Wave-driven GNNs similarly replace diffusion with oscillatory informa-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

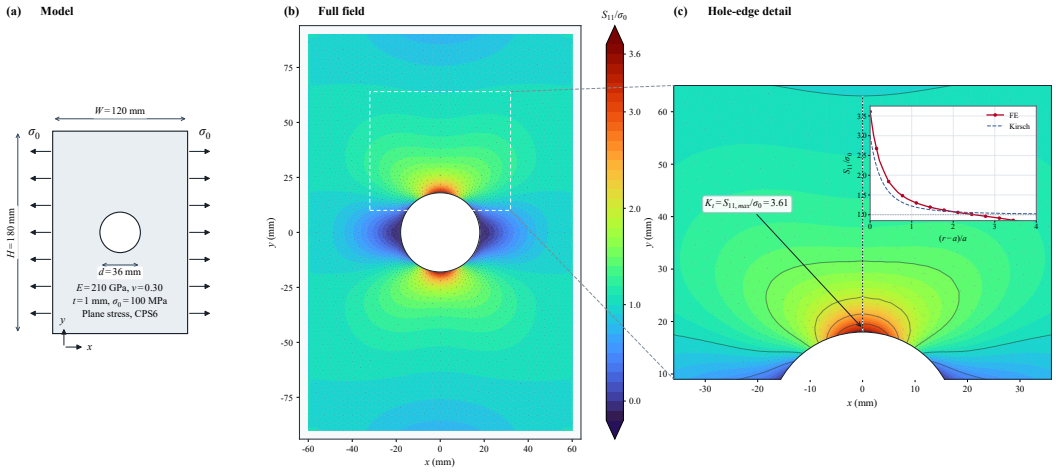


Figure 1: Stress concentration around a circular hole: geometry and loading, the finite element stress field, and local stress decay compared with the Kirsch solution.

tion transport (Wu et al., 2025). These methods show how retaining a previous state can change the behavior of repeated graph aggregation and help preserve information over depth.

A complementary line of work controls propagation in the spectral domain. CayleyNets use rational filters to obtain flexible frequency responses (Levie et al., 2018). Unitary Graph Convolution maps a symmetric graph operator to the complex unit circle, preserving the magnitude of every spectral component (Kiani et al., 2024). This suggests a useful design principle for stress prediction: graph frequencies should evolve through phase while retaining the information needed to reconstruct sharp spatial variations.

The practical difficulty is evaluating the propagation operator. Matrix exponentials on sparse graphs are typically implemented through Taylor, Chebyshev, or related approximations. A K-term approximation requires repeated sparse operations and incurs $O(Ked)$ propagation cost, where e is the number of edges and d is the hidden width (Kiani et al., 2024). Obtaining unit-modulus dynamics with a single neighborhood aggregation would make this principle more accessible to large finite element meshes.

We propose LiftGCN to obtain unit-modulus spectral dynamics with the locality and cost of ordinary message passing. The Joukowski relation (Gutknecht & Trefethen, 1982) converts a real graph eigenvalue into a pair of unit-circle modes. We realize this mapping through a real-valued second-order recurrence whose linear backbone preserves a positive-definite quadratic energy. Lightweight node-wise residuals supply nonlinear feature transformations, and each layer needs only one sparse neighborhood aggregation. This yields $O(ed)$ graph propagation without evaluating a matrix function.

The remainder of this paper is organized as follows. Section 2 introduces finite element stress fields and reviews diffusive and unitary graph propagation. Section 3 presents Joukowski spectral lifting, the LiftGCN architecture, and its spectral and computational properties. Section 4 evaluates LiftGCN through comparative experiments, depth analysis, and ablation studies. Finally, Section 5 concludes the paper.

2 Background

Stress fields contain sharp spatial variations that graph smoothing can suppress. We briefly connect their physical regularity to graph-frequency content, then contrast diffusive and unitary propagation.

2.1 Finite Element Stress Fields and Limited Regularity

For a linear elastic body occupying $\Omega \subset \mathbb{R}^m$, stress is $\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}(\mathbf{u})$, where $\mathbf{u}$ is displacement, $\mathbb{C}$ is the elasticity tensor, and $\boldsymbol{\varepsilon}(\mathbf{u}) = (\nabla\mathbf{u} + \nabla\mathbf{u}^\top)/2$. Equilibrium requires $-\nabla \cdot \boldsymbol{\sigma} = \mathbf{f}$, subject to prescribed displacement and traction conditions. The finite element formulation is based on the weak problem

$$\int_\Omega \boldsymbol{\varepsilon}(\mathbf{v}) : \mathbb{C} : \boldsymbol{\varepsilon}(\mathbf{u})\, d\Omega = \int_\Omega \mathbf{v} \cdot \mathbf{f}\, d\Omega + \int_{\Gamma_N} \mathbf{v} \cdot \bar{\mathbf{t}}\, d\Gamma, \qquad \forall \mathbf{v} \in V_0, \tag{1}$$

where V_0 contains test displacements that vanish on the prescribed-displacement boundary, and $\bar{\mathbf{t}}$ is the traction on Γ_N. Under standard assumptions, the energy solution belongs to $H^1(\Omega)$ (Bathe et al., 1996; Zienkiewicz et al., 2013). Since stress depends on displacement derivatives, its natural regularity is generally $L^2(\Omega)$; global smoothness is not guaranteed.

Holes and smooth fillets can produce large but finite stress gradients, while sharp notches, re-entrant corners, and material interfaces can induce stronger local variations. Near a geometric singularity, the leading stress response may take the form

$$\boldsymbol{\sigma}(r, \theta) \sim r^{\kappa-1}\boldsymbol{\Phi}(\theta), \qquad 0 < \kappa < 1, \tag{2}$$

where r is the distance to the singularity. Such behavior explains why a stress field can be difficult to reconstruct even when the surrounding displacement field appears smooth.

On a mesh graph, let $\mathcal{L} = \mathbf{I} - \widetilde{\mathbf{A}} = \mathbf{Q}\,\mathrm{diag}(\mu_j)\mathbf{Q}^\top$ be the normalized Laplacian. For a nodal stress signal $\mathbf{y}$ with graph Fourier coefficients $\widehat{\mathbf{y}} = \mathbf{Q}^\top\mathbf{y}$,

$$\mathbf{y}^\top\mathcal{L}\mathbf{y} = \sum_j \mu_j |\widehat{y}_j|^2. \tag{3}$$

Higher graph frequencies contribute more strongly to this variation measure. Preserving them helps reconstruct localized stress patterns that a purely smooth representation may miss.

2.2 Diffusive and Unitary Graph Propagation

For an undirected graph, let $\widetilde{\mathbf{A}} = \widehat{\mathbf{D}}^{-1/2}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-1/2}$ denote a symmetric normalized adjacency, where $\widehat{\mathbf{A}}$ may include self-loops. Its eigenvalues λ_j lie in $[-1, 1]$. A standard GCN updates features as $\mathbf{H}_{l+1} = \phi(\widetilde{\mathbf{A}}\mathbf{H}_l\mathbf{W}_l)$. Isolating the linear propagation gives

$$\widetilde{\mathbf{A}}^L = \mathbf{Q}\,\mathrm{diag}(\lambda_j^L)\mathbf{Q}^\top, \qquad |\lambda_j|^L \to 0 \quad \text{for } |\lambda_j| < 1. \tag{4}$$

Repeated aggregation therefore attenuates these modes and favors smoother representations (Li et al., 2018; Hoang et al., 2021). In particular, low Laplacian frequencies correspond to adjacency eigenvalues close to one, which decay slowly. Components with smaller eigenvalue magnitudes disappear more quickly. This unequal attenuation can weaken local stress contrasts as the receptive field grows, linking the physical reconstruction problem to the spectral behavior of propagation.

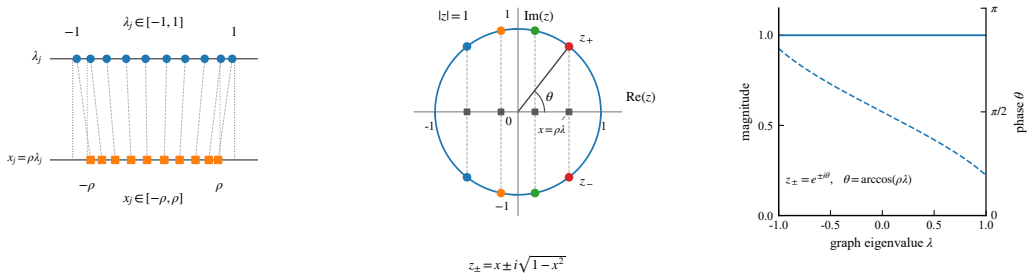
Unitary propagation instead uses

$$\mathbf{U}(t) = e^{\mathrm{i}t\widetilde{\mathbf{A}}} = \mathbf{Q}\,\mathrm{diag}(e^{\mathrm{i}t\lambda_j})\mathbf{Q}^\top, \qquad |e^{\mathrm{i}t\lambda_j}| = 1. \tag{5}$$

Each spectral component changes phase without losing magnitude (Kiani et al., 2024). Consequently, $\mathbf{U}(t)^*\mathbf{U}(t) = \mathbf{I}$, so repeated unitary propagation preserves the Euclidean norm of the feature signal. The attraction is that increasing the propagation distance need not progressively erase frequency content. Approximating the matrix exponential requires multiple graph operations, motivating a simpler realization of unit-modulus dynamics.

3 Joukowski Spectral Lifting and LiftGCN

LiftGCN obtains stable spectral dynamics through state-space lifting. We derive the real-valued recurrence, combine it with local nonlinear transformations, and establish its energy-preserving structure and propagation cost.



(a) Real graph spectrum. (b) Joukowski spectral lifting. (c) Unit-modulus response.

Figure 2: Joukowski spectral lifting maps the scaled real graph spectrum to conjugate unit-circle modes, preserving their magnitude while changing their phase.

3.1 Joukowski Spectral Lifting

The Joukowski map $J(z) = (z + z^{-1})/2$ sends $z = e^{\mathrm{i}\theta}$ to $\cos\theta$ (Gutknecht & Trefethen, 1982). Conversely, for $x \in (-1, 1)$, solving $z^2 - 2xz + 1 = 0$ gives

$$z_\pm = x \pm \mathrm{i}\sqrt{1 - x^2} = e^{\pm \mathrm{i} \arccos x}, \qquad |z_\pm| = 1. \tag{6}$$

Setting $x = \rho_c \lambda_j$ with $|\rho_c| < 1$ lifts each graph eigenvalue to two unit-circle modes, as illustrated in Figure 2. Their phases depend on graph frequency and channel, while their magnitudes remain one.

The same quadratic is the characteristic equation of $h_{j,c}^{(l+1)} = 2\rho_c \lambda_j h_{j,c}^{(l)} - h_{j,c}^{(l-1)}$. Replacing the eigenvalue by the graph operator yields

$$\overline{\mathbf{H}}_{l+1} = 2\widetilde{\mathbf{A}}\mathbf{H}_l\mathbf{R} - \mathbf{H}_{l-1}, \qquad \mathbf{R} = \mathrm{diag}(\rho_1, \ldots, \rho_d). \tag{7}$$

This recurrence implements the lifted dynamics entirely in real arithmetic. The previous hidden state supplies the second degree of freedom required by the conjugate roots; no complex-valued features or explicit spectral decomposition are needed. We learn $\rho_c = \rho_{\max} \tanh(\theta_c)$ with $0 < \rho_{\max} < 1$, sharing each coefficient across depth. Thus, channels can learn distinct spectral phases while remaining in the unit-modulus regime. Sharing the coefficients preserves a stationary linear recurrence, while channel-wise adaptation lets different features use different frequency-dependent phases.

3.2 LiftGCN Architecture

An input encoder maps node features to $\mathbf{H}_0 \in \mathbb{R}^{n \times d}$. The first propagation is $\overline{\mathbf{H}}_1 = \widetilde{\mathbf{A}}\mathbf{H}_0\mathbf{R}$; subsequent steps use Equation 7. Each step then applies a node-wise residual update,

$$\mathbf{H}_{l+1} = \overline{\mathbf{H}}_{l+1} + \alpha_l \,\mathrm{GELU}\big(\mathrm{LN}(\overline{\mathbf{H}}_{l+1})\mathbf{W}_l + \mathbf{b}_l\big)\,, \tag{8}$$

where $\mathbf{W}_l$ and $\mathbf{b}_l$ mix channels, and α_l is a learnable residual scale. A linear readout maps the final two-state representation $[\mathbf{H}_L, \mathbf{H}_{L-1}]$ to nodal stress. Figure 3 summarizes the propagation and residual blocks.

The recurrence controls inter-node information transport, while the residuals learn local nonlinear corrections. Normalization and channel mixing act independently at each node, and the learned residual scale controls the strength of each correction. The two-state readout exposes both states of the recurrence to the predictor. This separation adds expressive feature transformations without introducing extra graph aggregations.

3.3 Spectral and Computational Properties

For each graph mode and channel, the linear recurrence has roots $e^{\pm \mathrm{i} \arccos(\rho_c \lambda_j)}$. Their unit magnitude avoids the exponential decay associated with repeated adjacency multiplication. The recurrence

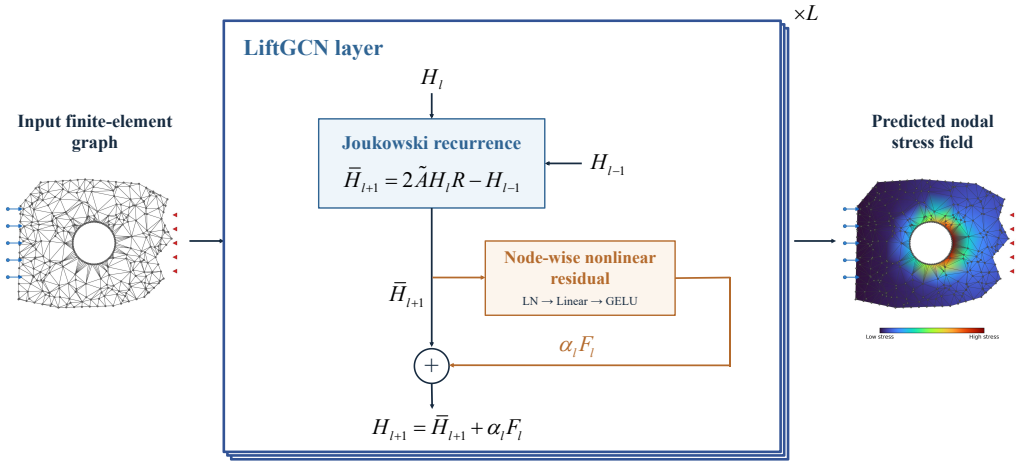


Figure 3: LiftGCN combines second-order Joukowski propagation with node-wise nonlinear residuals. Each layer uses one sparse neighborhood aggregation and the previous hidden state.

also preserves a quadratic energy in its augmented state. For a single channel with coefficient ρ, write $\mathbf{Z}_l = [\mathbf{h}_l^\top, \mathbf{h}_{l-1}^\top]^\top$ and $\mathbf{Z}_{l+1} = \mathbf{U}_\rho \mathbf{Z}_l$, where

$$\mathbf{U}_\rho = \begin{bmatrix} 2\rho\widetilde{\mathbf{A}} & -\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \qquad \mathbf{G}_\rho = \begin{bmatrix} \mathbf{I} & -\rho\widetilde{\mathbf{A}} \\ -\rho\widetilde{\mathbf{A}} & \mathbf{I} \end{bmatrix}. \tag{9}$$

The eigenvalues of $\mathbf{G}_\rho$ are $1 \pm \rho\lambda_j > 0$, and direct multiplication gives

$$\mathbf{U}_\rho^\top \mathbf{G}_\rho \mathbf{U}_\rho = \mathbf{G}_\rho, \qquad \|\mathbf{Z}_{l+1}\|_{\mathbf{G}_\rho}^2 = \|\mathbf{Z}_l\|_{\mathbf{G}_\rho}^2. \tag{10}$$

The conserved quantity can also be written directly in terms of consecutive hidden states:

$$\|\mathbf{Z}_l\|_{\mathbf{G}_\rho}^2 = \|\mathbf{h}_l\|_2^2 + \|\mathbf{h}_{l-1}\|_2^2 - 2\rho\mathbf{h}_l^\top \widetilde{\mathbf{A}}\mathbf{h}_{l-1}. \tag{11}$$

Energy is therefore preserved in the pair of states, allowing information to move between them as propagation proceeds. This result applies independently to every channel of the linear backbone. It provides a non-dissipative basis for propagation, to which the learned residuals add nonlinear corrections.

Each layer uses one sparse multiplication by $\widetilde{\mathbf{A}}$, giving $O(ed)$ graph propagation cost rather than the $O(Ked)$ cost of a K-step matrix-function approximation. Node-wise channel mixing costs $O(nd^2)$ in both cases. LiftGCN therefore reduces the graph propagation work while retaining one-hop communication and locally stored second-order memory.

This locality is useful on large meshes: each node exchanges only its current representation with immediate neighbors and stores its own previous state. Stable spectral dynamics are obtained by augmenting the state, without expanding the communication neighborhood within a layer. Additional layers can then extend the receptive field through the same sparse connectivity.

4 Experiment

We evaluate LiftGCN in terms of whole-field accuracy, stress-concentration reconstruction, and inference efficiency. All experiments are conducted on a workstation equipped with a single NVIDIA RTX PRO 6000 GPU. All prediction metrics reported in the tables and quantitative figures are averaged over 10 repeated experiments. We first introduce the dataset and evaluation metrics, then compare LiftGCN with representative baselines and analyze its sensitivity to propagation depth, followed by ablation studies on Joukowski propagation, spectral scaling, and local feature transformations. Appendix B extends the comparisons to two additional geometries, with data and training details in Appendices A and C.

4.1 Dataset and Evaluation Metrics

We use *Mines_Paris_Biaxial_Specimen*, comprising 100 finite element simulations of a biaxial specimen with random elastic properties (Kerfriden, 2022). The task is to predict nodal von Mises stress from geometry, material properties, and mesh connectivity. Appendix A.1 specifies the inputs and mesh statistics, and Appendix A.3 describes graph construction. Training settings and checkpoint selection based on test-set loss are detailed in Appendix C.

We measure whole-field accuracy with SNR and R^2. For reference stresses y_i and predictions $\hat{y}_i$, pooling evaluation nodes within each repetition gives

$$\mathrm{SNR} = 10 \log_{10} \frac{\sum_i y_i^2}{\sum_i (\hat{y}_i - y_i)^2}, \qquad R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{12}$$

where $\bar{y}$ is the pooled reference mean. To evaluate concentrations, let S_k contain the top $k\%$ of nodes by reference stress magnitude on each mesh, with $k \in \{1, 5, 10\}$. Peak-region accuracy is

$$\mathrm{PSNR@}k\% = 10 \log_{10} \frac{\max_{i \in S_k} |y_i|^2 + \epsilon}{|S_k|^{-1} \sum_{i \in S_k} (\hat{y}_i - y_i)^2 + \epsilon}. \tag{13}$$

For undirected edges E_k touching S_k, define $g_{ij}(y) = (y_i - y_j)/\max(\|\mathbf{x}_i - \mathbf{x}_j\|_2, \epsilon)$ using node coordinates $\mathbf{x}_i$. High-stress gradient error is

$$\text{HSG-NMSE@}k\% = \frac{\sum_{(i,j) \in E_k} [g_{ij}(\hat{y}) - g_{ij}(y)]^2}{\max(\sum_{(i,j) \in E_k} g_{ij}(y)^2, \epsilon)}. \tag{14}$$

Here ϵ ensures numerical stability. PSNR and HSG-NMSE are averaged over evaluation meshes before averaging across the 10 repetitions. Higher SNR, R^2, and PSNR and lower HSG-NMSE are better. Together, these metrics assess the overall stress field, its peaks, and the sharp variations around them. We also report parameter counts and Latency, defined as the average forward inference time per sample (one complete mesh), measured separately from the repeated training experiments.

4.2 Main Comparisons, Efficiency, and Depth Sensitivity

Table 1 compares LiftGCN with conventional message-passing, mesh-based, dynamical, spectral, and attention-based graph models. At comparable parameter counts, small LiftGCN achieves the lowest latency and improves both whole-field and local accuracy over GCN. It also matches or slightly improves UniGCN's peak-region PSNR with much faster inference. These 10-run mean results show that the lifted recurrence can retain useful local stress information at low propagation cost.

Increasing model capacity makes further use of this efficiency. Medium LiftGCN outperforms MeshGraphNets and UniGCN across the reported prediction metrics while remaining faster. With a latency budget comparable to Hybrid Local–Global Attention GNN and UniGCN, large LiftGCN leads the peak-region PSNR at the two smallest thresholds and all high-stress gradient metrics. Its strongest gains lie in the localized structures that motivate the model, alongside competitive overall accuracy.

Figure 4 makes this advantage visible. LiftGCN reproduces compact high-stress bands near curved boundaries and preserves heterogeneous patterns within the specimen. These local details agree with the peak and gradient improvements observed in the 10-run mean metrics.

The accuracy–latency curves in Figure 5 show a favorable trade-off across model sizes. LiftGCN already reconstructs stress concentrations well at low latency, and additional capacity steadily improves peak and gradient accuracy. The benefit of cheaper propagation is therefore practical: more of the inference budget can support feature learning. Comparisons on the connecting-lug and elbow-bracket datasets in Appendix B further assess this trade-off across geometries.

The two local metrics reveal complementary aspects of this trade-off. Higher peak-region PSNR indicates closer agreement with the stress values in highly loaded regions, while lower HSG-NMSE indicates better recovery of how stress changes around them. LiftGCN improves both as capacity increases. Its advantage therefore extends beyond matching peak values to recovering the surrounding spatial structure, which is central to a faithful stress-field surrogate.

Table 1: Main comparison. Prediction metrics are means over 10 repeated experiments. **Bold** marks the best prediction metric or lowest latency. Hybrid LG denotes Hybrid Local–Global Attention GNN, and Adv-GCN denotes GCN with adversarial training. Parentheses indicate the number of discriminator parameters.

(a) Whole-field accuracy and computational cost

Model	Parameters	Graph propagation complexity	Latency↓ (ms)	SNR↑ (dB)	R^2 ↑
GCN (Kipf & Welling, 2016)	444k	$\mathcal{O}(ed)$	1.719	15.685	0.85744
Adv-GCN (Oommen et al., 2026)	444k (+67k)	$\mathcal{O}(ed)$	1.719	15.605	0.85479
MeshGraphNets (Pfaff et al., 2021)	401k	$\mathcal{O}(ed^2)$	6.173	16.008	0.86764
GCNII (Chen et al., 2020)	439k	$\mathcal{O}(ed)$	1.639	15.095	0.83670
GraphCON (Rusch et al., 2022)	439k	$\mathcal{O}(ed)$	1.703	13.307	0.75349
CayleyNet (Levie et al., 2018)	428k	$\mathcal{O}(rJed)$	28.164	16.126	0.87109
EWGNN (Wu et al., 2025)	462k	$\mathcal{O}(n^2d)$	4.393	15.212	0.84098
GUMP (Qiu & Yao, 2024)	439k	$\mathcal{O}(M_L d)^\dagger$	48.527	13.441	0.76017
Hybrid LG (Patrignani & Pinho, 2026)	438k	$\mathcal{O}(ed^2 + n^2d/M)$	13.365	**17.104**	**0.89607**
A-DGN (Gravina et al., 2023)	403k	$\mathcal{O}(ed)$	2.211	15.671	0.85696
UniGCN (Kiani et al., 2024)	446k	$\mathcal{O}(Ked)$	15.000	16.104	0.87053
LiftGCN (ours), small	443k	$\mathcal{O}(ed)$	**1.528**	15.919	0.86487
LiftGCN (ours), medium	3299k	$\mathcal{O}(ed)$	5.041	16.610	0.88475
LiftGCN (ours), large	16.4M	$\mathcal{O}(ed)$	13.851	16.821	0.89022

(b) Stress-concentration reconstruction

Model	PSNR (dB)↑			HSG-NMSE↓		
	@1%	@5%	@10%	@1%	@5%	@10%
GCN (Kipf & Welling, 2016)	18.886	21.779	23.406	0.145440	0.117880	0.101360
Adv-GCN (Oommen et al., 2026)	19.094	21.947	23.417	0.142850	0.117070	0.104030
MeshGraphNets (Pfaff et al., 2021)	19.737	22.282	23.772	0.089125	0.082616	0.076469
GCNII (Chen et al., 2020)	18.662	21.225	22.710	0.196790	0.163430	0.148320
GraphCON (Rusch et al., 2022)	17.062	19.514	20.868	0.415490	0.403890	0.397550
CayleyNet (Levie et al., 2018)	20.783	22.281	23.589	0.083929	0.098303	0.098552
EWGNN (Wu et al., 2025)	18.746	21.312	22.943	0.150180	0.132090	0.117000
GUMP (Qiu & Yao, 2024)	17.478	19.888	21.232	0.350990	0.379950	0.388420
Hybrid LG (Patrignani & Pinho, 2026)	21.501	23.813	**25.138**	0.072304	0.071222	0.067561
A-DGN (Gravina et al., 2023)	19.086	21.778	23.338	0.095384	0.090815	0.083688
UniGCN (Kiani et al., 2024)	20.096	22.526	23.974	0.073533	0.072321	0.067939
LiftGCN (ours), small	20.327	22.615	23.975	0.087084	0.081528	0.075953
LiftGCN (ours), medium	21.884	23.744	24.871	0.058066	0.061305	0.060404
LiftGCN (ours), large	**22.201**	**24.035**	25.086	**0.053120**	**0.057062**	**0.057472**

Propagation complexity is reported per layer and excludes purely node-wise feature transformations. Here n, e, and d denote the numbers of nodes, edges, and hidden features, respectively. K is the Taylor truncation order in UniGCN; r and J are the Cayley filter order and number of Jacobi iterations in CayleyNet; and M is the global-attention interval in Hybrid LG, whose cost is shown as the amortized per-layer complexity. For GUMP, $M_L = \sum_{v \in V} \deg(v)^2$ is the number of admissible transitions in the directed line graph. †GUMP additionally requires unitary operator construction, whose K_G-step blockwise Newton–Schulz projection costs $\mathcal{O}\big(K_G \sum_{v \in V} \deg(v)^3\big)$.

Figure 6 shows that LiftGCN benefits consistently from greater depth across the tested range. Whole-field and peak accuracy improve while local gradient error decreases. GCN largely saturates, and the unitary and attention baselines perform best at shallower or intermediate depths. These 10-run averages suggest that LiftGCN can incorporate wider spatial context while retaining the local variations needed for stress reconstruction.

This joint improvement matters because increasing the receptive field is useful only if the additional context remains informative. The depth curves show that LiftGCN can improve the broad stress distribution without sacrificing the sharp features measured by the local metrics. This behavior is

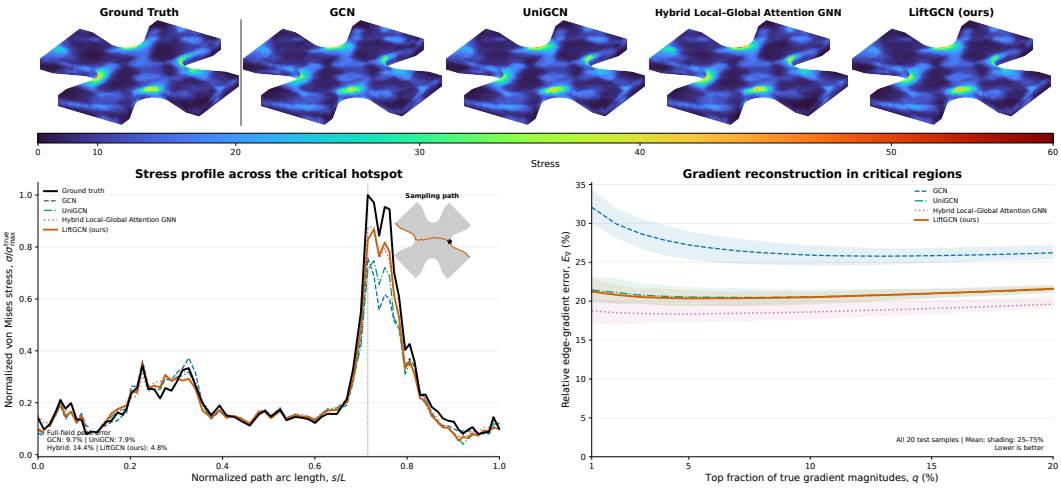
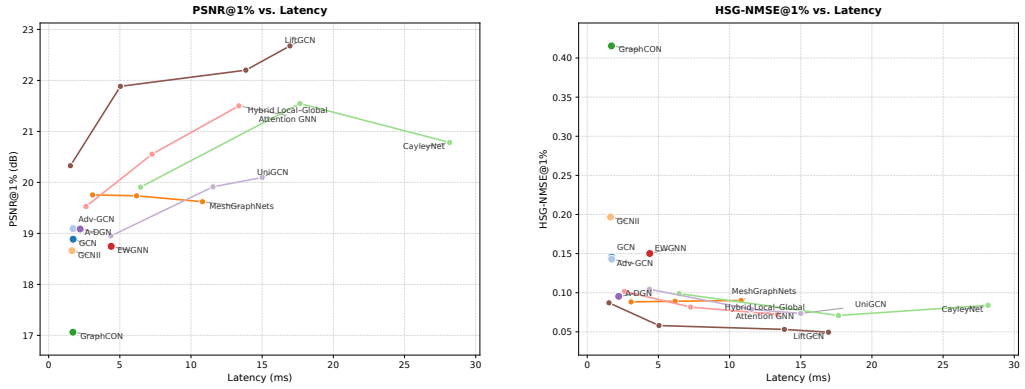


Figure 4: Stress-field reconstruction and hotspot analysis: representative stress fields, a hotspot stress profile, and gradient errors across the test set. LiftGCN preserves localized concentrations and spatial variations.



(a) Peak-stress accuracy versus latency.

(b) High-stress gradient error versus latency.

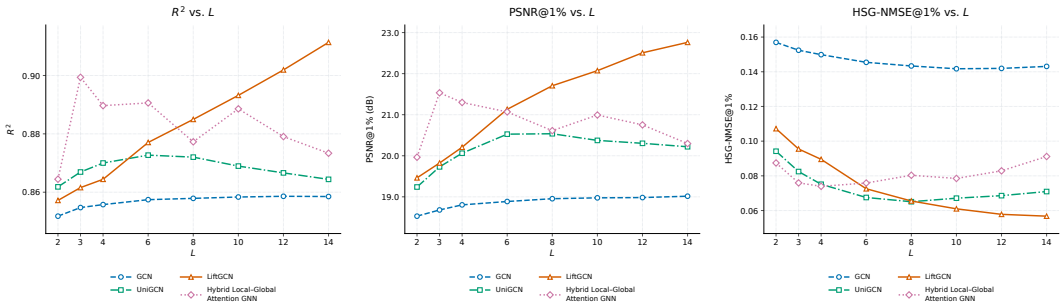
Figure 5: Accuracy–latency trade-offs. Prediction metrics are averaged over 10 repeated experiments. LiftGCN (ours) achieves strong peak-stress and local-gradient reconstruction across inference budgets.

consistent with the role of the lifted backbone: it supports repeated information transport while the residual blocks progressively adapt the features to the prediction task.

4.3 Ablation Studies

We test four variants, retaining the input encoder, depth, and two-state readout throughout. *w/o Joukowski* replaces the lifted recurrence with first-order aggregation, while *Fixed-ρ* fixes the spectral coefficients and retains the local residual blocks. *w/o nonlinear* removes only GELU from each propagation-layer residual block, retaining LayerNorm, channel mixing, and the residual connection. *Pure Joukowski* removes these entire blocks, leaving the hidden states to follow the second-order recurrence. Both variants retain learnable channel-wise spectral coefficients and the nonlinear input encoder. Table 2 reports the means over 10 repetitions.

Removing Joukowski propagation reduces accuracy across all metrics, especially in local gradient reconstruction, even with the local transformations retained. The lifted recurrence therefore plays an



(a) Whole-field R^2. (b) PSNR@1%. (c) HSG-NMSE@1%.

Figure 6: Depth sensitivity. Every point is a mean over 10 repeated experiments. LiftGCN (ours) improves both whole-field and local reconstruction as propagation depth increases.

Table 2: Ablation results averaged over 10 repeated experiments. Bold marks the best mean. Accurate stress reconstruction benefits from both Joukowski propagation and local nonlinear transformations; learnable spectral scaling provides further refinement.

Model	SNR (dB)↑	R^2 ↑	PSNR@1% (dB)↑	HSG-NMSE@1%↓
LiftGCN (ours)	**16.329**	**0.87705**	**21.131**	**0.072548**
w/o Joukowski	14.227	0.80057	18.027	0.275960
Fixed-ρ (0.9)	16.327	0.87698	21.098	0.073043
w/o nonlinear	15.342	0.84570	19.387	0.120570
Pure Joukowski	12.897	0.72913	12.050	0.407700

important role in preserving stress structure. Fixed spectral scaling retains most of the full model's performance, with learnable coefficients providing a small additional improvement.

Removing only GELU also weakens whole-field and local reconstruction, showing that nonlinear activation improves how propagated features are used. GELU has no trainable parameters, so this comparison preserves model size and isolates the activation within the local residual blocks. Removing the entire local residual module causes the largest degradation, particularly in peak-stress and gradient accuracy. The much stronger results with channel mixing and residual updates retained show the value of feature transformations between propagation steps. Together, the 10-run means support the combined design: Joukowski propagation preserves information, while local nonlinear transformations turn it into accurate stress predictions.

5 Conclusion

LiftGCN realizes Joukowski spectral lifting through a simple real-valued second-order recurrence. Its linear backbone preserves a quadratic energy, and each layer requires only one sparse neighborhood aggregation. Experiments averaged over 10 repetitions show that this design combines efficient inference with accurate reconstruction of stress concentrations and local gradients. The model uses additional capacity and depth effectively, while ablations demonstrate the complementary benefits of lifted propagation and local nonlinear transformations. These findings support spectral lifting as an efficient basis for graph-based stress prediction.

More broadly, the method connects information-preserving dynamics with the sparse communication structure of finite element meshes. It offers a practical route to retaining local physical detail as graph models aggregate wider context. Extending this approach to broader geometries, material models, and physical prediction tasks is a natural direction for future work.

AI USE STATEMENT

Generative AI tools, including ChatGPT, were used to assist with the refinement of theoretical formulations and mathematical presentation, experimental design, software implementation, and interpretation of experimental results. They were also used to assist with literature search and summarization, drafting and editing parts of the manuscript, improving readability, and preparing research code and scientific figures. All AI-assisted mathematical derivations, claims, references, code, and written content were independently reviewed, verified, and revised by the authors. The generated and modified code was tested by the authors, and all reported experimental results were obtained from experiments conducted and evaluated by the authors rather than generated by AI. The authors take full responsibility for the methodology, results, claims, and final content of this work.

REPRODUCIBILITY STATEMENT

Section 3 specifies the LiftGCN recurrence, architecture, and assumptions for the energy-preservation result. Appendix A documents dataset sizes, simulation parameters, stress extraction, and graph construction. Appendix B provides the additional benchmark results, and Appendix C records normalization, optimization, repeated splits, checkpoint selection, and inference timing. The public Mines Paris data are identified by their dataset citation (Kerfriden, 2022).

REFERENCES

Klaus-Jürgen Bathe et al. *Finite element procedures*, volume 2. Prentice hall New Jersey, 1996.

Moritz Braun, André Mischa Müller, Aleksandar-Saša Milaković, Wolfgang Fricke, and Sören Ehlers. Requirements for stress gradient-based fatigue assessment of notched structures according to theory of critical distance. *Fatigue & Fracture of Engineering Materials & Structures*, 43 (7):1541–1554, 2020.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020.

Dassault Systèmes SIMULIA. *Flap mechanism*, 2025a. Abaqus 2025 documentation: Example Problems Guide.

Dassault Systèmes SIMULIA. *Connecting lug*, 2025b. Abaqus 2025 documentation: Getting Started with Abaqus.

Matteo Giacomini and Pedro Díez. Surrogates for physics-based and data-driven modelling of parametric systems: Review and new perspectives. *Archives of Computational Methods in Engineering*, pp. 1–33, 2026.

Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-symmetric dgn: a stable architecture for deep graph networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=J3Y7cgZOOS.

Martin H Gutknecht and Lloyd N Trefethen. Real polynomial chebyshev approximation by the carathéodory–fejér method. *SIAM Journal on Numerical Analysis*, 19(2):358–371, 1982.

NT Hoang, Takanori Maehara, and Tsuyoshi Murata. Revisiting graph neural networks: Graph filtering perspective. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 8376–8383. IEEE, 2021.

SC Jayasinghe, M Mahmoodian, A Alavi, A Sidiq, F Shahrivar, Z Sun, J Thangarajah, and S Setunge. A review on the applications of artificial neural network techniques for accelerating finite element analysis in the civil engineering domain. *Computers & Structures*, 310:107698, 2025.

Pierre Kerfriden. Surrogate-modelling & machine learning dataset: finite element stress analysis of biaxial specimen with random elastic properties - 100 samples. Zenodo, version V0.9, 2022. URL https://doi.org/10.5281/zenodo.6827035.

Bobak T Kiani, Lukas Fesser, and Melanie Weber. Unitary convolutions for learning on graphs and groups. *Advances in Neural Information Processing Systems*, 37:136922–136961, 2024.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Xiaoke Li, Qingyu Yang, Yang Wang, Xinyu Han, Yang Cao, Lei Fan, and Jun Ma. Development of surrogate models in reliability-based design optimization: A review. *Mathematical Biosciences and Engineering*, 18(5):6386–6409, 2021.

Liang Liang, Minliang Liu, Caitlin Martin, and Wei Sun. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. *Journal of The Royal Society Interface*, 15(138):20170844, 2018.

Marco Maurizi, Chao Gao, and Filippo Berto. Predicting stress, strain and deformation fields in materials and structures with graph neural networks. *Scientific reports*, 12(1):21834, 2022.

Vivek Oommen, Siavash Khodakarami, Aniruddha Bora, Zhicheng Wang, and George Em Karniadakis. Learning turbulent flows with generative models for super resolution and sparse flow reconstruction. *Nature Communications*, 17(1):3707, 2026.

Luca Patrignani and Silvestre T Pinho. Graph neural networks with hybrid local-global attention for effective prediction of mechanical response in structures. *Computer Methods in Applied Mechanics and Engineering*, 452:118753, 2026.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Haiquan Qiu and Quanming Yao. Graph unitary message passing, 3 2024. URL `https://arxiv.org/abs/2403.11199`.

Andreas Roth and Thomas Liebig. Rank collapse causes over-smoothing and over-correlation in graph neural networks. In *Learning on Graphs Conference*, pp. 35–1. PMLR, 2024.

T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International conference on machine learning*, pp. 18888–18909. PMLR, 2022.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Chao Sun and Zhen Chen. End-to-end deep learning method to reconstruct full-field stress distribution for ship hull structure with stress concentrations. *Ocean Engineering*, 313:119431, 2024.

Peihan Wu, Hongda Qi, Sirong Huang, Dongdong An, Jie Lian, and Qin Zhao. Wave-driven graph neural networks with energy dynamics for over-smoothing mitigation. In *IJCAI*, pp. 6579–6587, 2025.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Shun-Peng Zhu, Wen-Long Ye, José A.F.O. Correia, Abílio M.P. Jesus, and Qingyuan Wang. Stress gradient effect in metal fatigue: Review and solutions. *Theoretical and Applied Fracture Mechanics*, 121:103513, 2022. ISSN 0167-8442.

Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, and Jian Z Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2013.

A Datasets and Graph Representation

This appendix specifies the finite element data used in Section 4.1, including the quantities actually supplied to the networks. Table 3 summarizes all complete samples in the processed datasets. Counts are measured from the coordinate and element-connectivity files; graph edges follow the constructions in Appendix A.3. An edge is counted once as an undirected pair, before adding self-loops or storing both message-passing directions.

Table 3: Dataset statistics. Variable mesh sizes are reported as minimum–maximum, with arithmetic means on separate rows. The feature dimension includes the degree feature.

Property	Mines Paris	Connecting lug	Elbow bracket
Samples	100	225	288
Nodes per mesh	10,502	9,344–11,762	10,282–17,944
Mean nodes	10,502	10,683.72	13,878.89
Elements per mesh	43,399	5,778–7,515	6,088–10,936
Mean elements	43,399	6,735.65	8,354.37
Undirected graph edges	60,544	15,918–20,132	17,396–30,500
Mean graph edges	60,544	18,251.52	23,544.44
Element nodes	4	10 (C3D10M)	10 (C3D10M)
Input/output channels	$5/1$	$4/1$	$4/1$
Shared mesh across samples	Yes	No	No
Training/evaluation samples	$80/20$	$180/45$	$230/58$

A.1 Mines Paris Biaxial Specimen

We use the 100 random-material realizations of the public Mines Paris biaxial-specimen dataset, version V0.9 (Kerfriden, 2022). The processed samples have identifiers 1–100; the homogeneous reference realization indexed by 0 in the original archive is not included. All 100 processed coordinate files are identical, as are all 100 connectivity files. The common three-dimensional mesh has 10,502 nodes and 43,399 four-node tetrahedra. Its coordinate bounds are $[-1.5, 1.5] \times [-1.5, 1.5] \times [0, 0.2]$ in the source coordinate units. The original VTK files and the processed CSV files agree on the mesh size. Each realization changes the spatial Young's-modulus field on this fixed specimen. The nodal modulus values across the processed realizations span approximately $[2.9708, 129.9183]$ in the source material units.

Let $\mathbf{P} \in \mathbb{R}^{n\times 3}$ contain nodal coordinates, $\mathcal{T}$ the tetrahedral connectivity, and $\mathbf{E}^{(s)} \in \mathbb{R}^{n\times 1}$ the nodal Young's modulus of sample s. The physical prediction task is

$$(\mathbf{P}, \mathcal{T}, \mathbf{E}^{(s)}) \longmapsto \mathbf{y}^{(s)} = (\sigma^{(s)}_{\mathrm{vm},i})_{i=1}^{n} \in \mathbb{R}^{n\times 1}. \tag{15}$$

The implemented feature matrix is $\mathbf{X}^{(s)} = [\mathcal{Z}_P(\mathbf{P}), \mathcal{Z}_E(\mathbf{E}^{(s)}), \mathbf{q}] \in \mathbb{R}^{n\times 5}$, where $\mathcal{Z}$ denotes training-set standardization and $\mathbf{q}$ is the standardized log-degree feature defined below. The network predicts one scalar per node. Although the original archive contains stress-tensor components, the experiments supervise only the scalar von Mises field; displacements and stress components are not input features. Boundary conditions and load magnitudes are not separate model inputs in this fixed experimental setting.

The loader matches coordinate, modulus, connectivity, and stress files by sample identifier, and aligns modulus and stress rows to coordinate rows by node identifier. For this common mesh, the topology can be constructed once and reused. The material field, rather than connectivity, supplies the sample-dependent input.

A.2 ABAQUS DATASETS: GEOMETRY, LOADING, AND SIMULATION

We construct two additional datasets with Abaqus/CAE 2025 and the Abaqus/Standard static solver. Their parametric scripts adapt the official connecting-lug example and the modelling workflow of the official flap-mechanism example (Dassault Systèmes SIMULIA, 2025b;a). The latter is reduced to a single perforated elbow bracket, rather than the full articulated mechanism. Both use homogeneous isotropic linear elasticity with Young's modulus 200 GPa and Poisson's ratio 0.30, and modified ten-node tetrahedral elements (C3D10M). Geometries are remeshed for every parameter combination. Coordinates, force, and stress use metres, newtons, and pascals in the simulation and exported data; Table 4 expresses lengths in millimetres for readability.

Table 4: Actual parameter grids, verified against the per-sample parameter records. $a : \Delta : b$ denotes values from a to b with increment Δ. Each dataset uses the Cartesian product of its listed levels.

Dataset / parameter	Values	Number of levels
Connecting lug: 225 combinations		
Shank length	100 mm	1
Outer radius	$28.6 : 0.2 : 29.4$ mm	5
Pin-hole radius	12.5 mm	1
Mounting-hole radius	$4.8, 5.0, 5.2$ mm	3
Thickness	$16.6 : 0.2 : 19.4$ mm	15
Global mesh seed	6 mm	1
Total force	20,000 N in the $-y$ direction	1
Elbow bracket: 288 combinations		
Horizontal arm length	$160, 180, 200$ mm	3
Vertical arm length	$180, 200, 220$ mm	3
Centreline bend radius	$60, 70$ mm	2
Arm width	$50, 60$ mm	2
Hole radius	$9, 10, 11, 12$ mm	4
Thickness	$18, 20$ mm	2
Global mesh seed	8 mm	1
Total force	2,000 N	1
Force angle from $+x$ toward $+y$	-45°	1

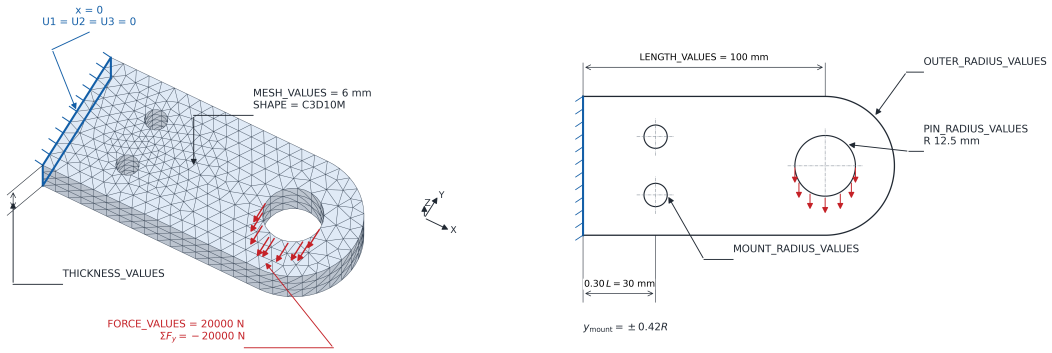
Connecting lug. The lug consists of a straight shank, a rounded end with a pin hole, and two smaller mounting holes. The shank face at $x = 0$ is fully fixed. The prescribed resultant acts on the lower half of the pin-hole surface: if this region contains N_{load} mesh nodes, each receives $\mathbf{f}_i = (0, -20000/N_{\text{load}}, 0)$ N. Thus the implemented loading is a set of concentrated nodal forces with the stated total resultant. This adapts the pressure-loaded official example using the prescribed force resultant. Varying the outer radius, mounting-hole radius, and thickness yields $5 \times 3 \times 15 = 225$ samples. Figure 7 shows the geometry and representative finite element stress fields.

Elbow bracket. Two straight perforated arms are joined by a curved 90° bend. The face at $x = 0$ is fully fixed, and the force is distributed equally among nodes on the free end of the vertical arm. For $\theta = -45^\circ$, the nodal force is $\mathbf{f}_i = 2000(\cos\theta, \sin\theta, 0)/N_{\text{load}}$ N. The six varied geometric parameters yield $3 \times 3 \times 2 \times 2 \times 4 \times 2 = 288$ combinations. Figure 8 illustrates the geometry, loading, and spatially localized stresses around the bend, holes, and constrained region.

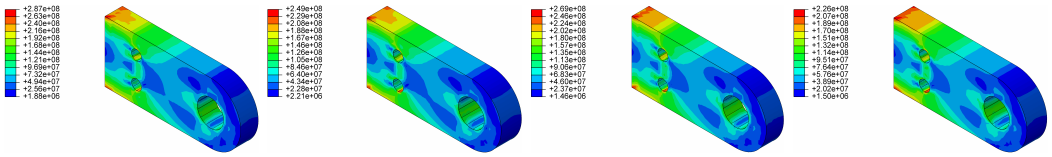
Stress extraction and learning task. The postprocessing scripts extract stress tensors at the `ELEMENT_NODAL` positions of the selected ODB frame (the final frame by default). For a node shared by several elements, tensor components are first averaged arithmetically over the element-nodal contributions. Von Mises stress is then computed from this averaged tensor:

$$\sigma_{\text{vm}} = \sqrt{\tfrac{1}{2}[(\bar{\sigma}_{xx} - \bar{\sigma}_{yy})^2 + (\bar{\sigma}_{yy} - \bar{\sigma}_{zz})^2 + (\bar{\sigma}_{zz} - \bar{\sigma}_{xx})^2] + 3(\bar{\sigma}_{xy}^2 + \bar{\sigma}_{yz}^2 + \bar{\sigma}_{zx}^2)}. \tag{16}$$

This operation differs from averaging element-wise von Mises scalars. The exported stress table contains six tensor components and the scalar invariant, but training uses only the latter.

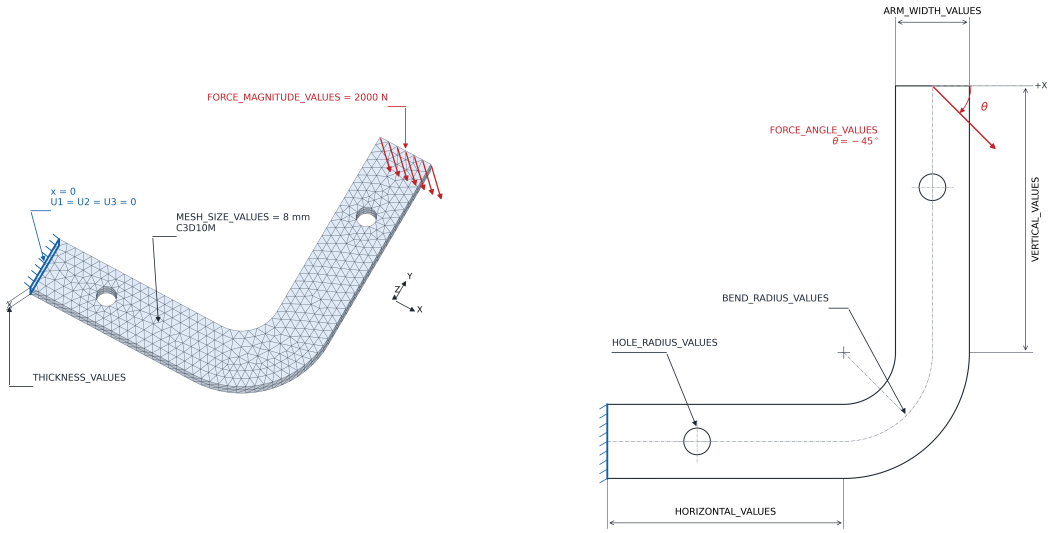


(a) Mesh, fixed boundary, nodal loading, and geometric parameters.

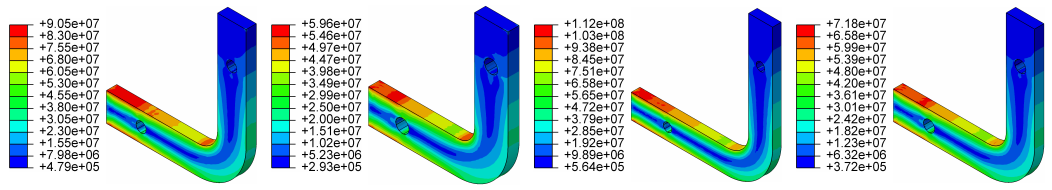


(b) Representative simulated von Mises stress fields for different geometries.

Figure 7: Connecting-lug dataset. The contour panels illustrate geometric variation under a fixed total load; each panel retains its own stress colour scale in Pa.



(a) Mesh, boundary conditions, force direction, and geometric parameters.



(b) Representative simulated von Mises stress fields for different geometries.

Figure 8: Elbow-bracket dataset. Each contour panel uses its own stress colour scale in Pa, so colours should be interpreted together with the corresponding legend.

For either custom dataset, sample s defines a geometry-dependent mesh $(\mathbf{P}^{(s)}, \mathcal{T}^{(s)})$ and target $\mathbf{y}^{(s)} \in \mathbb{R}^{n_s \times 1}$. The physical and implemented mappings are

$$(\mathbf{P}^{(s)}, \mathcal{T}^{(s)}) \longmapsto \mathbf{y}^{(s)}, \qquad f_\Theta(\mathbf{X}^{(s)}, \widetilde{\mathbf{A}}^{(s)}) = \widehat{\mathbf{y}}^{(s)}_{\mathrm{norm}}, \qquad \mathbf{X}^{(s)} = [\mathcal{Z}_P(\mathbf{P}^{(s)}), \mathbf{q}^{(s)}] \in \mathbb{R}^{n_s \times 4}. \tag{17}$$

Material constants, total load, load direction, and boundary-condition rules are fixed within each dataset, so they are not supplied as additional channels. In particular, the number of loaded nodes varies with the mesh, and the simulation divides the fixed total force accordingly. These experiments assess stress prediction across geometric configurations under the prescribed physical setting, using finite element solutions as reference fields. Model comparisons on both custom datasets are reported in Appendix B.

A.3 From Finite Element Meshes to Graphs

Each mesh node becomes a graph vertex, including the midside nodes of quadratic elements. Node labels are mapped to contiguous row indices, and target rows are aligned by label. Connectivity is used to define an undirected graph $G = (V, \mathcal{E})$; it is not treated as a dense input feature.

Connectivity rules. For Mines Paris, all distinct node pairs in each four-node tetrahedron are connected, giving its six edges. The custom-data loader uses the Abaqus C3D10M ordering: nodes 1–4 are corner nodes and 5–10 lie on edges $(1,2), (2,3), (3,1), (1,4), (2,4), (3,4)$, respectively. Each quadratic edge is split at its midside node. The twelve local segments are

$$\begin{aligned}\mathcal{E}_{\mathrm{tet}} = \{&(1,5), (5,2), (2,6), (6,3), (3,7), (7,1),\\ &(1,8), (8,4), (2,9), (9,4), (3,10), (10,4)\}.\end{aligned} \tag{18}$$

Edges shared by elements are deduplicated globally. Each undirected edge is stored in both directions for message passing. The generic fallback in the custom-data loader connects all valid pairs for other element sizes, but every custom-data element used here has ten nodes and follows the twelve-segment rule. Thus these quadratic tetrahedra are not converted to ten-node cliques.

Features and graph operators. Let d_i be the number of distinct neighbours before adding self-loops and $\ell_i = \log(1 + d_i)$. The last input channel is

$$q_i = \frac{\ell_i - \bar{\ell}}{s_\ell + 10^{-6}}, \qquad \bar{\ell} = \frac{1}{n}\sum_i \ell_i, \qquad s_\ell^2 = \frac{1}{n}\sum_i (\ell_i - \bar{\ell})^2. \tag{19}$$

Coordinates and, for Mines Paris, Young's modulus are standardized componentwise using all nodes in the training samples only. The target is similarly standardized for optimization, with $\widehat{y}_i = \sigma_y \widehat{y}_{\mathrm{norm},i} + \mu_y$ before evaluating physical stress metrics. Any training-set standard deviation below 10^{-12} is replaced by one. The degree normalization is computed within each graph and does not use target values.

For LiftGCN, the binary symmetric adjacency $\mathbf{A}$ gives $\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{\mathbf{A}} = \widehat{\mathbf{D}}^{-1/2}\widehat{\mathbf{A}}\widehat{\mathbf{D}}^{-1/2}$, as in Section 2. The base graph has no separately measured edge attributes. MeshGraphNets derives a four-dimensional edge feature from the difference of standardized endpoint coordinates and its Euclidean norm. Hybrid LG instead forms coordinate differences and lengths in the original coordinates, then standardizes these edge features using training-set statistics. Local-gradient evaluation uses the undirected edges without self-loops and the original, unstandardized coordinates, consistently with Eq. (14).

B Model Comparisons on the Custom Datasets

Tables 5 and 6 extend Section 4.2 to the connecting lug and elbow bracket. They report the same whole-field and local metrics as the main comparison, including all three hotspot thresholds. Prediction scores are means over 10 repetitions. The checkpoint-selection protocol is specified in Appendix C.

Table 5: Connecting-lug comparison. Prediction metrics are means over 10 repetitions. Bold marks the best score in each column, including ties. Parameters in parentheses belong to the Adv-GCN discriminator.

(a) Whole-field accuracy and computational cost

Model	Parameters	Latency (ms)↓	SNR (dB)↑	R^2 ↑
GCN	1512k	**0.691**	25.021	0.98754
Adv-GCN	1512k (+244k)	**0.691**	25.014	0.98754
MeshGraphNets	1549k	2.383	26.594	0.99132
GCNII	1503k	1.412	16.707	0.91574
GraphCON	1503k	1.894	24.101	0.98463
CayleyNet	1551k	33.694	28.413	0.99431
EWGNN	1525k	3.855	18.764	0.94696
GUMP	1504k	27.285	22.386	0.97719
Hybrid LG	1547k	7.848	28.913	0.99473
A-DGN	1526k	2.592	24.733	0.98667
UniGCN	1519k	21.510	26.422	0.99096
LiftGCN, small	1514k	0.900	25.762	0.98953
LiftGCN, medium	3023k	2.610	28.071	0.99384
LiftGCN, large	11571k	8.870	**29.558**	**0.99563**

(b) Stress-concentration reconstruction

Model	PSNR (dB)↑			HSG-NMSE↓		
	@1%	@5%	@10%	@1%	@5%	@10%
GCN	27.962	29.510	30.518	0.036797	0.033067	0.031227
Adv-GCN	27.934	29.487	30.509	0.035820	0.032702	0.031144
MeshGraphNets	**34.056**	34.029	33.867	0.020698	0.028265	0.029088
GCNII	19.496	21.695	22.719	0.498130	0.419740	0.356840
GraphCON	26.991	28.602	29.493	0.058009	0.052488	0.048874
CayleyNet	31.141	33.428	34.543	**0.019631**	**0.021604**	**0.021072**
EWGNN	24.741	25.422	26.082	0.186240	0.232120	0.228620
GUMP	26.447	27.979	28.774	0.059343	0.064130	0.064358
Hybrid LG	31.979	33.791	34.642	0.028996	0.028105	0.027341
A-DGN	28.886	29.853	30.668	0.032105	0.034136	0.033857
UniGCN	30.185	31.387	32.107	0.024199	0.024338	0.024768
LiftGCN, small	29.517	30.625	31.367	0.039788	0.040724	0.040821
LiftGCN, medium	31.341	33.081	33.939	0.031984	0.037151	0.035538
LiftGCN, large	33.366	**35.357**	**36.035**	0.025049	0.023864	0.023407

On the connecting lug, small LiftGCN improves SNR and peak-region PSNR over GCN at a latency of 0.900 ms, although its HSG-NMSE is higher. Large LiftGCN attains the best SNR (29.558 dB), R^2 (0.99563), and PSNR at the 5% and 10% thresholds. It takes 8.870 ms, compared with 21.510 ms for UniGCN and 33.694 ms for CayleyNet. However, MeshGraphNets has the best PSNR@1%, CayleyNet leads all three gradient metrics, and GCN/Adv-GCN have the lowest latency. These results demonstrate a useful accuracy–latency trade-off across model sizes.

Table 6: Elbow-bracket comparison. Prediction metrics are means over 10 repetitions. Bold marks the best score in each column, including ties. Parameters in parentheses belong to the Adv-GCN discriminator.

(a) Whole-field accuracy and computational cost

Model	Parameters	Latency (ms)↓	SNR (dB)↑	R^2 ↑
GCN	1512k	**0.570**	17.354	0.95714
Adv-GCN	1512k (+244k)	**0.570**	17.175	0.95526
MeshGraphNets	1549k	3.228	18.245	0.96493
GCNII	1503k	2.383	8.932	0.70091
GraphCON	1503k	2.663	16.160	0.94357
CayleyNet	1551k	41.259	23.418	0.98935
EWGNN	1525k	6.188	12.277	0.78350
GUMP	1504k	25.150	11.739	0.84394
Hybrid LG	1547k	13.318	18.945	0.96905
A-DGN	1526k	4.754	15.492	0.93417
UniGCN	1519k	27.688	18.997	0.97063
LiftGCN, small	1514k	2.361	18.246	0.96510
LiftGCN, medium	3023k	4.861	21.216	0.98239
LiftGCN, large	11571k	12.714	**24.019**	**0.99074**

(b) Stress-concentration reconstruction

Model	PSNR (dB)↑			HSG-NMSE↓		
	@1%	@5%	@10%	@1%	@5%	@10%
GCN	24.735	24.915	25.266	0.124680	0.113800	0.127510
Adv-GCN	24.469	24.693	25.081	0.133390	0.120480	0.135120
MeshGraphNets	26.303	26.262	26.449	0.088613	0.089554	0.100940
GCNII	14.184	14.322	14.651	1.483500	1.731300	1.745500
GraphCON	23.362	23.781	24.088	0.255970	0.231220	0.248400
CayleyNet	28.899	29.158	29.631	0.054606	0.053158	0.059374
EWGNN	17.384	18.293	19.027	0.696530	0.627560	0.636530
GUMP	18.903	18.391	18.012	0.591310	0.604710	0.655690
Hybrid LG	24.311	25.441	26.196	0.161900	0.155140	0.158990
A-DGN	25.053	24.760	24.346	0.170570	0.192130	0.257930
UniGCN	26.998	26.656	27.008	0.053734	0.059160	0.066304
LiftGCN, small	26.543	26.205	26.586	0.090990	0.089279	0.108020
LiftGCN, medium	29.036	28.743	28.998	0.046467	0.053178	0.062416
LiftGCN, large	**29.904**	**29.985**	**30.354**	**0.038466**	**0.044894**	**0.054048**

On the elbow bracket, large LiftGCN leads all reported prediction metrics: SNR reaches 24.019 dB, R^2 reaches 0.99074, and HSG-NMSE@1% falls to 0.038466. Relative to CayleyNet, it improves SNR by 0.601 dB and reduces HSG-NMSE@1% by approximately 29.6%, with $3.25\times$ lower forward latency (12.714 versus 41.259 ms). Medium LiftGCN also improves every prediction metric over UniGCN at 4.861 versus 27.688 ms. GCN remains the fastest model. Across both datasets, greater LiftGCN capacity improves every reported prediction metric, with parameter counts and latency documenting the computational cost of scaling.

C Experimental Settings

C.1 Shared Training and Evaluation Protocol

Table 7 summarizes the common training protocol in the experiment scripts, with the observed model-specific exceptions explicitly noted. Each repetition shuffles sample identifiers with a NumPy generator seeded by $42 + r$, for $r = 0, \ldots, 9$, and assigns the first $\lfloor 0.8N \rfloor$ samples to training. The remaining samples form the evaluation partition. The split is at the simulation level, not at the node level. Models using the same dataset and repetition seed receive the same partition. Normalization statistics are recomputed from the training partition for each repetition.

Table 7: Training settings shared across the three datasets, as implemented in the supplied scripts. Model-specific configurations are listed in Tables 8 and 9.

Setting	Value / procedure
Repetitions and seeds	10 runs; seeds 42–51 for Python, NumPy, and PyTorch
Training/evaluation ratio	80%/20%; counts are given in Table 3
Batch and sample order	One complete mesh per optimizer update; shuffle training meshes each epoch using seed plus epoch index
Training duration	50 epochs, without early termination
Predictor optimizer	AdamW, learning rate 2×10^{-3}, weight decay 10^{-5}
Learning-rate schedule	Cosine annealing, $T_{\max} = 50$, minimum learning rate 0
Gradient clipping	Global parameter-gradient norm bounded by 1.0
Regression objective	Mean squared error over nodes in standardized stress units
Dropout	0.10
Adv-GCN exception	Predictor additionally uses an adversarial loss; discriminator optimized separately with Adam
Checkpoint choice	Lowest evaluation-partition MSE over 50 epochs; first minimum retained on a tie
Final scores	Restore the chosen predictor checkpoint, undo target standardization, evaluate, then average scores over 10 runs
Inference timing	CUDA events; 20 warm-up and 100 timed forward calls per profiled graph shape, outside training

Checkpoint selection. After each epoch, we compute the test-set MSE in standardized stress units, averaging the node-mean MSE over test meshes. For each repetition, we retain the checkpoint with the lowest test-set MSE across 50 epochs and restore it to compute the reported metrics. This procedure is used for all three datasets.

Inference timing. Latency is the average forward inference time per sample on the NVIDIA RTX PRO 6000. The benchmark places graph tensors on the GPU before timing, switches the predictor to evaluation mode, and measures forward calls with synchronized CUDA events. CSV loading, graph construction, host-to-device transfer, normalization, and metric calculation are excluded. The LiftGCN scripts cache timing measurements by the tuple (node count, input dimension, adjacency nonzeros) within each partition, profiling one representative of each shape. Each profiled forward call processes one complete mesh, and the timings are averaged across the profiled shapes and calls. Adv-GCN uses only its predictor at inference; its discriminator parameters are reported separately.

C.2 Model-Specific Configurations

Tables 8 and 9 specify the model configurations used for Mines Paris and the two custom datasets, respectively. Parameter names follow the implementation. Connecting lug and elbow bracket use the same model configurations. Shared optimization settings are given in Table 7.

Table 8: Model configurations for Mines Paris.

Model	Model-specific hyperparameters
GCN	`hidden_dim=270`, `layers=6`
Adv-GCN	`hidden_dim=270`, `layers=6`, `adv_beta=0.1`, `disc_hidden_dim=128`, `disc_layers=4`
MeshGraphNets	`mgn_latent_size=64`, `mgn_message_passing_steps=10`
GCNII	`hidden_dim=270`, `layers=6`, `gcnii_alpha=0.1`, `gcnii_lambda=0.50`
GraphCON	`hidden_dim=270`, `layers=6`, `graphcon_dt=1.0`, `graphcon_alpha=1.0`, `graphcon_gamma=1.0`
CayleyNet	`hidden_dim=110`, `layers=6`, `cayley_order=3`, `cayley_jacobi_iters=5`, `cayley_h_init=1.0`
EWGNN	`hidden_dim=195`, `layers=4`, `ewgnn_alpha=0.5`, `ewgnn_eta=1.0`
GUMP	`hidden_dim=240`, `layers=4`, `base_layers=1`, `gump_attn_dim=32`, `gump_ns_iters=10`
Hybrid LG	`hidden_dim=90`, `layers=6`, `attention_frequency=3`
A-DGN	`hidden_dim=400`, `layers=6`, `adgn_epsilon=0.1`, `adgn_gamma=0.1`
UniGCN	`hidden_dim=235`, `layers=4`, `taylor_order=10`, `uniconv_time=1.0`
LiftGCN, small	`hidden_dim=330`, `layers=4`
LiftGCN, medium	`hidden_dim=640`, `layers=8`
LiftGCN, large	`hidden_dim=1280`, `layers=10`

Table 9: Shared model configurations for connecting lug and elbow bracket.

Model	Model-specific hyperparameters
GCN	`hidden_dim=500`, `layers=6`
Adv-GCN	`hidden_dim=500`, `layers=6`, `adv_beta=0.1`, `disc_hidden_dim=200`, `disc_layers=6`
MeshGraphNets	`mgn_latent_size=160`, `mgn_message_passing_steps=6`
GCNII	`hidden_dim=500`, `layers=6`, `gcnii_alpha=0.1`, `gcnii_lambda=0.50`
GraphCON	`hidden_dim=500`, `layers=6`, `graphcon_dt=1.0`, `graphcon_alpha=1.0`, `graphcon_gamma=1.0`
CayleyNet	`hidden_dim=210`, `layers=6`, `cayley_order=3`, `cayley_jacobi_iters=5`, `cayley_h_init=1.0`
EWGNN	`hidden_dim=290`, `layers=6`, `ewgnn_alpha=0.5`, `ewgnn_eta=1.0`
GUMP	`hidden_dim=380`, `layers=6`, `base_layers=2`, `gump_attn_dim=32`, `gump_ns_iters=10`
Hybrid LG	`hidden_dim=170`, `layers=6`, `attention_frequency=3`
A-DGN	`hidden_dim=780`, `layers=6`, `adgn_epsilon=0.1`, `adgn_gamma=0.1`
UniGCN	`hidden_dim=355`, `layers=6`, `taylor_order=10`, `uniconv_time=1.0`
LiftGCN, small	`hidden_dim=500`, `layers=6`
LiftGCN, medium	`hidden_dim=500`, `layers=12`
LiftGCN, large	`hidden_dim=800`, `layers=18`