Surrogate Stress Prediction in 3D Finite Element Meshes Using Graph Neural Networks

Alexandre Vallet, Gaspard Lafont, Tomas Jouven
EPFL, Machine Learning Course, Lausanne, Switzerland

***Abstract*—Engineering simulations via the Finite Element Method (FEM) are widely used to evaluate mechanical behaviour in complex 3D geometries. However, stochastic FEM workflows consider uncertainties and require many costly deterministic simulations. Neural Network (NN) surrogate models offer a promising alternative by shifting the computational burden to a one-time training phase. Standard NN architectures struggle with the spatial topology of 3D meshes. Graph Neural Networks (GNNs) provide a natural framework to represent such structures. In this work, we implement and compare three state-of-the-art GNN architectures: U-Net GNN, E(n)-Equivariant GNN, and ClofNet GNN, for stress prediction on 3D FE meshes. We also propose our own GNN architecture variant. Our work builds upon the study and dataset by Ezemba et al. [1](Carnegie Mellon University).**

I. Introduction

A. GNN Fundamentals

GNNs generalize neural networks to graphs $G = (V, E)$. The goal is to learn a function mapping the graph to target outputs (here, node-level stress values) by minimizing a loss function $\mathcal{L}$ via gradient descent. Most GNNs use a **message passing** paradigm. At layer t, a node v updates its features $h_v^{(t)}$ by aggregating messages from neighbors $\mathcal{N}(v)$:

$$h_v^{(t)} = U^{(t)}\Big(h_v^{(t-1)}, \mathcal{A}^{(t)}\big(\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\big)\Big) \tag{1}$$

where $\mathcal{A}^{(t)}$ (AGGREGATE) is permutation-invariant (e.g., sum, mean) and $U^{(t)}$ (UPDATE) is a differentiable function (e.g., MLP). This allows efficient propagation of information across the mesh topology.

II. Dataset Description

We utilize the synthetic dataset provided by Ezemba et al. [1]. It consists of approximately 16,000 unique 3D geometries represented as tetrahedral meshes (2000 nodes each in average), designed to emulate engineering components (Fig. 1). Each geometry is evaluated under multiple point elastic loading cases across three magnitude classes. Linear elastic simulations provide the ground truth von Mises stress field at each node. The input graph representation includes rich node features (3D positions, SDF values, surface normals, boundary flags, and load class indicators) totaling 12 scalar features per node. The target stress values are log-normalized. We use an 85:15 train–evaluation split.

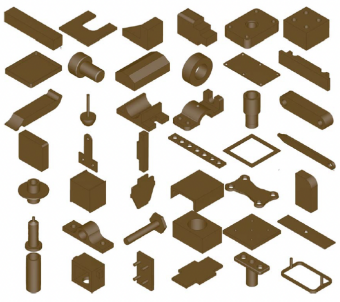

Fig. 1. Sample of 3D geometries in the dataset.

III. Experimental Setup

Implementation relies on the PyTorch (Geometric) library. Due to compatibility constraints regarding GPU acceleration on local macOS environments, all experiments were conducted using Google Colab. The substantial memory footprint of the full 3D mesh dataset exceeded standard runtime limits, necessitating a Colab Pro subscription. This environment provided access to High-RAM GPU instances (e.g., NVIDIA A100/V100), ensuring sufficient memory for data loading and significantly accelerating the training pipeline via CUDA support.

IV. Graph U-Nets: Hierarchical Learning

A. Theory

Standard GNNs often lack hierarchical feature extraction capabilities analogous to pooling in CNNs. Graph U-Nets (gU-Nets)[3] address this by adapting the U-Net encoder-decoder architecture to irregular graphs.

The core components are Graph Convolutional Network (GCN) layers and novel graph pooling (gPool) and unpooling (gUnpool) operations. The basic GCN layer updates node features H using the normalized adjacency matrix $\hat{A}$ and trainable weights W:

$$H^{(\ell+1)} = \sigma\Big(\hat{A}H^{(\ell)}W^{(\ell)}\Big) \tag{2}$$

B. Training and Results

1) Results: The training progression is illustrated in Figure 2, which shows the convergence of the loss over 50 epochs. For a detailed overview of the training configuration and hyperparameters, please refer to **Section A of the Appendix**.

Fig. 2. Learning curve of the U-Graph Net architecture over 50 epochs.

The final quantitative results for the validation set are summarized in Table I.

2) Discussion: This highlights a significant performance disparity across the validation set. The model achieves strong predictive accuracy for the top quantile ($R^2_{90} = 0.74$), suggesting effective learning on simpler geometries. However, performance degrades for the median ($R^2_{50} = 0.41$) and fails for the lowest quantile ($R^2_{10} = -0.08$). This negative score implies that for the most complex 10% of graphs, the model struggles to generalize, performing worse than a simple mean baseline.

This high variance suggests that while the U-Graph Net is capable of learning the underlying physics, it struggles to generalize to the most complex samples in the validation set.

3) Visualization of Stress Distributions: To qualitatively assess the U-Graph Net, we analyzed two representative geometries, the geometry that generated the best predictions measures for our model and a benchmark geometry that will be used to compare the implemented models. These visualisations highlight a clear trade-off between global field detection and local resolution.

The model excels on simple, beam-like structures (Figure 3), where it accurately captures smooth gradients ($R^2 > 0.7$).

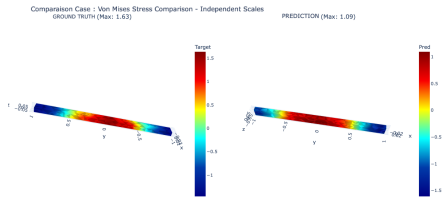

Fig. 3. Best Case (Part 3026): Accurate prediction on a simple geometry.

a) The Benchmark Case: Localized Forces on Plates: Figure 4 represents the "average" performance on a thin rectangular plate. While the geometry is simple, the stress pattern is complex, featuring distinct "islands" of high stress due to localized forces.

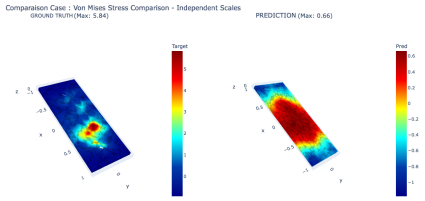

Fig. 4. Benchmark case: Distinct stress peaks are blurred into a diffuse cloud. This case serves as a key benchmark for model comparison.

Here, the model correctly identifies the *region* of stress but fails to separate the distinct peaks, merging them into a diffuse, low-magnitude cloud. This "blurring" effect confirms that the pooling layers reduce spatial resolution, preventing the capture of high-frequency variations. **Because this case isolates the issue of spatial resolution without geometric complexity, we will use it as the primary reference to compare subsequent models.**

V. EGNN: Incorporating Physical Symmetries

A. Motivations

In the context of SFEM simulations, the physical laws governing stress distribution exhibit specific symmetries. A fundamental property of 3D mechanical systems is that the underlying physics does not depend on the arbitrary choice of the coordinate system. For example, rotating a mechanical part and its applied loads results in a corresponding rotation of the internal stress field, without altering the magnitude or relative distribution of that stress.

Standard GNNs struggle to naturally account for these geometric transformations in Euclidean space, often requiring intensive data augmentation to implicitly learn these symmetries.

To address this, we implement the E(n)-Equivariant Graph Neural Network (EGNN) [4] architecture. By enforcing these symmetries (rotations, translations, reflections) as an inductive bias within the network structure, the EGNN is restricted to learning physically relevant functions. This intrinsic geometric awareness leads to significantly higher data efficiency compared to standard GNNs.

For a detailed description of the EGNN architecture, its mechanism of coordinate updating, and the specific training hyperparameters, please consult the **Section B of the Appendix**.

B. Training and Results

The training process shows stable convergence characteristics. The training loss decreases rapidly during the initial epochs as the model learns coarse features of the stress distribution, before gradually leveling off. Crucially, the validation loss follows a closely similar trajectory without significant divergence from the training loss. This indicates that the model is generalizing effectively to unseen 3D geometries and is not overfitting too much.

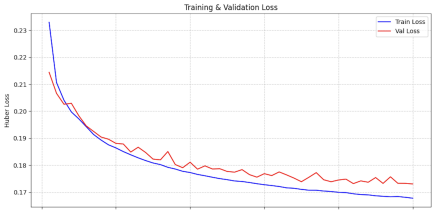

Fig. 5. Learning curve of the EGNN architecture over 50 epochs.

You can find the final quantitative results of the training in Table I. The EGNN outperforms the baseline U-Net across all global metrics, achieving a lower RMSE of 0.743 and improving the median R^2 score to 0.49. Crucially, it demonstrates significantly higher robustness on complex geometries, as evidenced by the 10th percentile R^2 recovering to 0.00, indicating that the physical symmetry constraints successfully prevent having catastrophic predictions as observed in the previous model.

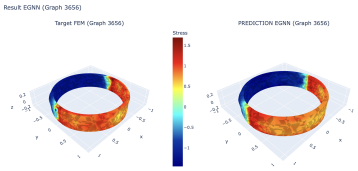

Fig. 6. Best prediction with R^2 = 0.95 for EGNN over 50 epochs

For geometries with well-distributed stress gradients, such as the cylindrical ring (Fig. 6), the EGNN demonstrates it's best performance (R^2 = 0.95), accurately capturing both the magnitude and spatial propagation of the high-stress zones.

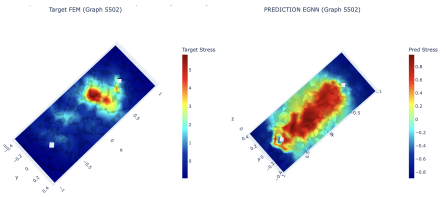

Fig. 7. Benchmark Case: Prediction for the benchmark geometry R^2 = 0.40

Overall, the EGNN outperforms the baseline U-Net by achieving lower global error and higher median correlation, demonstrating that enforcing physical symmetries significantly improves robustness on complex geometries where standard models previously failed. However, its reliance on isotropic filtering still limits its ability to fully resolve sharp, localized stress gradients in certain configurations.

VI. ClofNet: Handling Anisotropy with Local Frames

A. Motivations

While EGNNs successfully enforce global equivariance, their reliance on radial distances acts as an isotropic filter. In mechanical contexts, this creates a "direction degeneration" problem: the network struggles to distinguish between different orientations of interactions relative to the bond, making it difficult to capture complex anisotropic fields such as shear stress or torsion.

To address this limitation, we implement the ClofNet (Complete Local Frames GNN) architecture [5]. Unlike standard equivariant models, ClofNet explicitly constructs a unique, complete orthonormal basis. Geometric input vectors are then projected onto these local frames to obtain rotation invariant scalars. This mechanism allows ClofNet to explicitly encode complex directional information, effectively "seeing" the orientation of stress fields while guaranteeing equivariance.

For a detailed description of the ClofNet architecture, please consult the **Section C of the Appendix**.

B. Training and Results

The model shows rapid initial learning, but unlike the EGNN, a divergence between training and validation losses appears in later epochs, indicating mild overfitting. This likely arises from the high expressivity of complete local frames, which capture fine-grained training details that do not fully generalize, in contrast to the more constrained isotropic filtering of the EGNN.

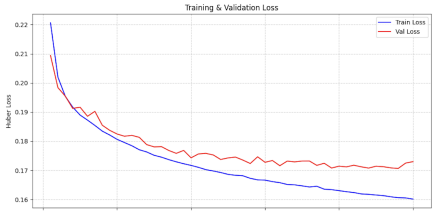

Fig. 8. Learning curve of the ClofNet architecture over 50 epochs.

ClofNet demonstrates performance highly comparable to the EGNN, yielding a slightly improved median accuracy $R^2_{50} = 0.50$ and the highest top-tier precision $R^2_{90} = 0.81$ among all models. This confirms that the complete local frames provide a marginal advantage in capturing fine geometric details for well-conditioned meshes. However, the performance on the most complex 10% of cases remains challenging $R^2_{10} = -0.05$, lagging slightly behind the EGNN. Overall, these results indicate that explicitly encoding anisotropy through local frames achieves a similar level of robustness and generalization as the rotation equivariant approach of the EGNN.

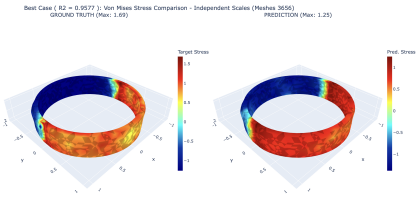

Fig. 9. Best Case: ClofNet prediction on a simple geometry ($R^2 \approx 0.96$).

The model achieves peak accuracy on rotationally symmetric geometries such as the cylindrical ring (Fig. 9), where its geometry-aware design effectively resolves stress relative to the surface rather than global axes, enabling high-fidelity capture of curved stress gradients.

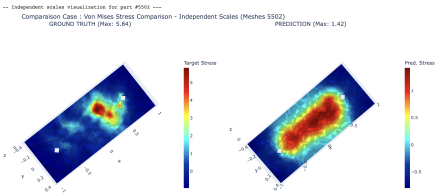

Fig. 10. Benchmark Case (Part 5502): Unlike the U-Net and EGNN, ClofNet successfully separates the distinct stress peaks.

A second qualitative improvement is observed on the benchmark plate (Fig. 10), where stress regions are more sharply localized compared to diffuse reconstructions. However, peak magnitudes remain underestimated and spatial boundaries are not fully recovered, indicating that while local frames reduce smoothing and improve definition, they are not yet sufficient to capture the full intensity and sharp geometry of the ground truth field.

VII. Proposed Model

In our proposed model, we implement a **Graph Transformer** architecture. This approach generalizes the convolutional methods typically found in architectures like Graph U-Nets.

Our motivation stems from the physical nature of the problem: a force applied to one side of a geometry can induce stress on the opposite side. While standard graph convolutions are often limited to local neighborhoods, the Transformer architecture empowers the model to capture these long-range dependencies by mapping the stress value at each node within the global context of all other nodes in the graph.

Regarding the implementation, we utilized the `TransformerConv` and `LayerNorm` modules available in the PyTorch Geometric library. We developed a custom model class that alternates between Transformer convolution layers and Layer Normalization. Additionally, the architecture is designed to increase the hidden channel dimensions throughout the layers, which required the definition of a custom forward pass.

To fully leverage the capacity of our architecture, specifically the attention mechanism which relies on spatial relationships, we engineered a rich set of edge features. To encode the geometric structure, we construct a 4-dimensional feature vector $\mathbf{e}_{ij}$ for each edge connecting nodes i and j with positions $\mathbf{p} \in \mathbb{R}^3$. This vector concatenates the relative position and the Euclidean distance:

$$\mathbf{e}_{ij} = [(\mathbf{p}_j - \mathbf{p}_i),\ \|\mathbf{p}_j - \mathbf{p}_i\|_2] \in \mathbb{R}^4 \tag{3}$$

A. Training and Results

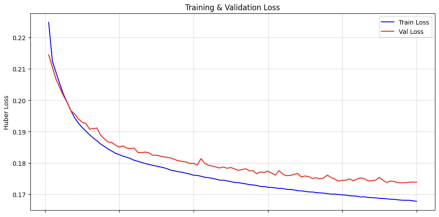

Fig. 11. Learning curve of the Transformers architecture over 100 epochs.

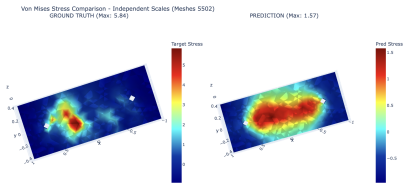

Fig. 12. Benchmark case: Distinct stress peaks are blurred into a diffuse cloud. This case serves as a key benchmark for model comparison.

Even with this approach, we still have this "over smoothing" effect, this show the limitations of the data set provided and the limitation of surrogate model for stress prediction.

Despite the integration of these geometric edge features, we observe a persistent *over-smoothing* effect in the predicted stress fields. This phenomenon highlights two critical bottlenecks: first, the limitations of the provided dataset, which may lack the granularity required to represent extreme gradients; and second, the inherent difficulty for surrogate models to capture high spatial frequency features, such as stress singularities, leading to a regression towards the mean behavior.

VIII. Comparative Model Performance Analysis

TABLE I
Comparison of Final Performance Metrics between Models on the Validation Set

Metric	**U-Graph Net**	**EGNN**	**ClofNet**	**Transformers**
Validation Loss	0.1955	0.1730	0.1706	0.1737
RMSE	0.794	0.743	0.749	0.748
R^2 (10th percentile)	-0.08	0.00	-0.05	-0.05
R^2 (50th percentile)	0.41	0.49	0.50	0.49
R^2 (90th percentile)	0.74	0.79	0.81	0.80

IX. Results and Discussion

This study explored the potential of Graph Neural Networks (GNNs) as surrogate models for predicting von Mises stress in 3D Finite Element meshes. By implementing and comparing U-Net, EGNN, ClofNet, and a Graph Transformer, we demonstrated that deep learning can effectively approximate physical simulations, provided that the architecture respects the underlying geometric laws.

A. The Primacy of Geometric Equivariance

Our results clearly establish the superiority of equivariant architectures over standard topological models. Both EGNN and ClofNet significantly outperformed the U-Net baseline. This confirms that enforcing $E(n)$ or $SE(3)$ equivariance acts as a crucial inductive bias, allowing the models to generalize physical behaviors (such as rotation invariance) without requiring extensive data augmentation. The standard U-Net, lacking this geometric awareness, failed to capture the stress distribution in complex geometries, particularly in the lower performance quantiles ($R^2_{10} \approx -0.08$).

B. Anisotropy and Dataset Limitations

While ClofNet is theoretically more expressive than EGNN due to its use of complete local frames allowing it to capture anisotropic features like shear or torsion our experiments revealed only a marginal performance gain over the isotropic EGNN. This suggests a limitation in the provided dataset. It is likely that the loading cases in the dataset are dominated by stress fields where radial interactions suffice, or that the mesh resolution is too coarse to fully leverage the directional sensitivity of ClofNet. As noted in our qualitative analysis, both models suffer from an "over-smoothing" effect, failing to resolve sharp stress singularities.

C. Computational Constraints and Transformer Efficiency

The Graph Transformer architecture yielded competitive results while offering superior computational efficiency in terms of execution time. However, our study was constrained by hardware limitations (GPU memory on Google Colab), which capped the size of the hidden channels and the depth of our networks. It is highly probable that with high-performance computing resources (e.g., clusters of A100s), scaling up the Transformer or ClofNet models would reduce the bias and improve the capture of high-frequency stress variations.

D. The Convergence Ceiling: Overfitting or Data Limitation?

A persistent observation across all implemented models is the convergence ceiling of the top-tier accuracy at approximately $R^2_{90} \approx 0.80$. The inability of distinct architectures to break this barrier strongly points to a deficiency in the dataset's richness rather than a model-specific failure. Furthermore, the divergence observed between training and validation losses in the later epochs suggests that the models begin to overfit when pushed beyond this limit. They attempt to memorize the noise in the training set rather than learning the generalized physics of the most complex geometries (the bottom 10% of cases), where performance remains poor.

E. Future Outlook: Towards Hybrid Solvers

Ultimately, this work highlights that while GNNs are promising, they are not yet a complete replacement for FEM in high-fidelity engineering scenarios, particularly for complex singularities. The "regression to the mean" behavior observed in difficult cases limits their utility as standalone validators. Consequently, the most promising direction for future research lies in hybrid workflows. A GNN could serve as a rapid, near-real-time preconditioner or estimator to guide a traditional FEM solver, thereby combining the inference speed of machine learning with the rigorous precision of numerical methods.

X. Ethical and Societal Implications

The development of deep learning surrogates for physical simulations presents distinct ethical challenges concerning safety-critical applications and dual-use technologies.

A. Reliability in Safety-Critical Systems

The primary ethical risk identified in this study stems from the "over-smoothing" phenomenon observed in GNN predictions. As noted in the results, all architectures struggled to fully capture extreme stress singularities. In engineering practice, these peak stress values are the most critical factors for predicting structural failure. There is a risk of *automation bias*, where engineers might blindly trust the rapid inference of the GNN over rigorous, albeit slower, numerical verification. If applied to safety-critical infrastructure (e.g., aerospace components, civil structures) without adequate safeguards, the underestimation of local stress concentrations could lead to catastrophic failures. Therefore, these models must be strictly framed as preconditioners or exploratory tools, never as replacements for certification-grade FEM solvers.

B. Dual-Use and Defense Implications

Furthermore, accelerating the design cycle of mechanical components creates a dual-use dilemma. While this technology aims to foster innovation in sustainable sectors, such as optimizing lightweight components for electric vehicles or renewable energy infrastructure, it inevitably lowers the computational barrier for defense applications. The ability to rapidly iterate on stress analysis without costly supercomputing resources can accelerate the development of kinetic

weapons and lethal autonomous systems. As future engineers, we must acknowledge that democratizing high-fidelity physics simulation contributes to the efficiency of military hardware optimization, raising concerns regarding the proliferation of advanced weaponry.

REFERENCES

[1] J. Ezemba, C. McComb, and C. Tucker, "Neural Network Surrogate Modeling for Stochastic Finite Element Method Using Three-Dimensional Graph Representations: A Comparative Study," *ASME IDETC/CIE*, 2023.

[2] P. Reiser et al., "Graph neural networks for materials science and chemistry," *Communications Materials*, vol. 3, no. 1, 2022.

[3] H. Gao and S. Ji, "Graph U-Nets," in *International Conference on Machine Learning (ICML)*, 2019.

[4] V. G. Satorras, E. Hoogeboom, and M. Welling, "E(n) Equivariant Graph Neural Networks," in *ICML*, 2021.

[5] W. Du et al., "SE(3) Equivariant Graph Neural Networks with Complete Local Frames," in *ICML*, 2022.

APPENDIX

A. U-Net Graph: Theory and Experimental Implementation Details

To implement the U-Graph Net architecture, we utilized a framework based on PyTorch Geometric. We integrated the pre-defined class structure into our pipeline to facilitate graph pooling and unpooling operations.

1) Hyperparameters and Training Configuration: To strike a balance between computational efficiency and model performance, we selected the following hyperparameters:

- **Architecture Depth:** 3 layers (comprising GCN and gPool/gUnpool blocks).
- **Hidden Channels:** 12.
- **Pooling Ratio:** 0.8.

The training configuration was defined as follows:

- **Epochs:** 50.
- **Learning Rate:** 1×10^{-4}.
- **Batch Size:** 64 graphs.

2) Loss Function and Hardware: We employed the **Huber Loss** ($\delta = 0.5$) function for optimization. This loss function is particularly advantageous for tailored distributions, such as stress distribution, as it is robust to outliers and behaves like a combination of Mean Squared Error (MSE) and Mean Absolute Error (MAE).

The model was trained on an NVIDIA A100 GPU (via Google Colab Pro), with a total execution time of approximately 2 hours.

B. EGNN: Theory and Experimental Implementation Details

1) Theory: Standard GNNs are designed to be permutation equivariant, meaning they process graph topology regardless of node ordering. However, they do not naturally account for geometric transformations in Euclidean space. To handle 3D data with standard GNNs, the network must implicitly "learn" these symmetries through extensive data augmentation (e.g., training on many rotated versions of the same mesh), which is computationally inefficient and requires larger datasets.

To address this, we implement the E(n)-Equivariant Graph Neural Network (EGNN) [4] architecture. This architecture is explicitly designed to enforce equivariance to rotations, translations, and reflections as an inductive bias within the network structure.

The EGNN achieves this via a two-part message passing mechanism that couples the node features ($\mathbf{h}$) and the coordinates ($\mathbf{x}$):

1) **Invariant Message Creation ($\mathbf{m}_{ij}$):** The edge message is created using only invariant features, specifically the relative squared distance $||\mathbf{x}_i^l - \mathbf{x}_j^l||^2$. This ensures that the information content remains unaffected by rotations or translations:
$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, ||\mathbf{x}_i^l - \mathbf{x}_j^l||^2, a_{ij}\right) \tag{4}$$
2) **Equivariant Coordinate Update ($\mathbf{x}_i^{l+1}$):** The model then uses the calculated message $\mathbf{m}_{ij}$ to determine how the coordinates should be displaced. The key is that the displacement is a

weighted sum of the relative difference vectors, $(\mathbf{x}_i^l - \mathbf{x}_j^l)$, which transforms identically to the rotation/translation applied to the input:

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \sum_{j \neq i} (\mathbf{x}_i^l - \mathbf{x}_j^l) \phi_x(\mathbf{m}_{ij}) \tag{5}$$

where ϕ_e and ϕ_x are Multi-Layer Perceptrons. By implementing the full EGNN framework, including this dynamic coordinate update, the model not only benefits from the geometric features but also ensures that the intermediate positions $\mathbf{x}^{l+1}$ and final node features $\mathbf{h}^{l+1}$ respect the underlying physical symmetries. This inductive bias leads to significantly higher data efficiency compared to standard GNNs.

2) Implementation Details: These are the hyperparameters that we used for the implementaiton of the EGNN. We maximised all the features to use the maximum capacity of the GPU (Nvidia A100) .

- **Architecture Depth:** 4 layers.
- **Hidden Channels:** 32 features per node.
- **Dataset Features:** 9.
- **Number of neighbors:** 16.
- **Epochs:** 50.
- **Learning Rate:** 5×10^{-4}
- **Weight Decay:** 1×10^{-16}.
- **Gradient Clipping:** 1.0.
- **Batch Size:** 4 graphs.

The implemented EGNN model consists of three sequential stages. First, an initial **Embedding Layer** (linear projection) maps the raw input node features (9 dimensions) into a higher-dimensional latent space ($d = 32$). Second, the core processing is performed by a stack of 4 Equivariant Graph Convolutional Layers (EGCLs). As shown in the forward pass, these layers are coupled: each layer takes the node features $\mathbf{h}$ and positions $\mathbf{pos}$ from the previous step and updates both, effectively propagating geometric information through the depth of the network. The neighbor graph is dynamically recomputed at each layer using the k-nearest neighbors algorithm ($k = 16$) on the updated coordinates. Finally, a **Stress Head** consisting of a Multi-Layer Perceptron **(Linear $\rightarrow$ ReLU $\rightarrow$ Linear)** maps the refined node embeddings to the final scalar stress prediction.

C. ClofNet: Theory and Experimental Implementation Details

1) Theory: While EGNNs ensure SE(3) equivariance using radial distances, they fundamentally rely on isotropic kernels which can lead to "direction degeneration"—a loss of angular information critical for modeling anisotropic stress tensors (e.g., torsion or shear). To overcome this, we implement the "ClofNet" (Complete Local Frames Network) architecture [5].

ClofNet distinguishes itself by explicitly constructing a unique, SE(3)-equivariant orthonormal basis (a "local frame") for every edge in the graph. This allows the network to project geometric input vectors into invariant scalars without loss of information. The architecture operates in two key phases:

1) **Complete Local Frame Construction ($\mathcal{F}_{ij}$):** For every pair of connected nodes i and j, a local frame $\mathcal{F}_{ij} = (\mathbf{a}_{ij}, \mathbf{b}_{ij}, \mathbf{c}_{ij})$ is derived from the node positions $\mathbf{x}$. Unlike spherical harmonics, this is achieved efficiently via cross-products:

$$\mathbf{a}_{ij} = \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \quad \text{(Radial direction)} \tag{6}$$

$$\mathbf{b}_{ij} = \frac{\mathbf{x}_i \times \mathbf{x}_j}{\|\mathbf{x}_i \times \mathbf{x}_j\|} \quad \text{(Cross product)} \tag{7}$$

$$\mathbf{c}_{ij} = \mathbf{a}_{ij} \times \mathbf{b}_{ij} \quad \text{(Vertical direction)} \tag{8}$$

These vectors form a rotation-equivariant basis. Any geometric vector projected onto this frame yields rotation-invariant scalars (e.g., $\mathbf{v} \cdot \mathbf{a}_{ij}$), which are then fed into the message passing MLP.

2) **Equivariant Coordinate Update ($\mathbf{x}_i^{l+1}$):** Unlike EGNN, which updates coordinates solely along the radial direction $(\mathbf{x}_i - \mathbf{x}_j)$, ClofNet learns to update positions along all three axes of the local frame. The coordinate update is formulated as a weighted sum of the frame vectors:

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \sum_{j \in \mathcal{N}(i)} (\phi_a \mathbf{a}_{ij} + \phi_b \mathbf{b}_{ij} + \phi_c \mathbf{c}_{ij}) \tag{9}$$

where ϕ_a, ϕ_b, ϕ_c are scalar coefficients predicted by the edge MLP based on invariant features. This mechanism allows the network to model complex, non-radial geometric relaxations in the latent space, providing superior expressiveness for structural mechanics.

2) Implementation Details: These are the specific hyperparameters used for the ClofNet implementation. We adapted the configuration to maximize the expressiveness of the local frames while maintaining training stability on the static dataset.

- **Architecture Depth:** 4 layers (Clof_GCL blocks).
- **Hidden Channels:** 64 features per node.
- **Dataset Features:** 9 scalars per node.
- **Geometric Features:** 9 scalars per edge (1 radial distance + 8 frame projections).
- **Coordinate Update Weight:** 1.0 (Enabled) with `TANH=False`, allowing unconstrained virtual relaxation of the mesh.
- **Epochs:** 50.
- **Learning Rate:** 1×10^{-3} .
- **Weight Decay:** 1×10^{-12}.
- **Gradient Clipping:** 1.0.
- **Batch Size:** 64 graphs.

3) Loss Function and Hardware: We employed the **Huber Loss** ($\delta = 0.5$) for optimization, consistent with the other models. The model was trained on an NVIDIA L4 GPU (via Google Colab), with a total execution time of approximately 1 hour and 40 minutes.

D. Transformers: Experimental Implementation Details

1) Hyperparameters and Training Configuration: To strike a balance between computational efficiency and model performance, we selected the following hyperparameters:

- **Number of layers:** 5 layers.
- **Hidden Channels:** 30.
- **Number of attention heads:** 3.
- **Concatenation:** True.

The training configuration was defined as follows:

- **Epochs:** 100.
- **Learning Rate:** 1×10^{-4}.
- **Batch Size:** 32 graphs.

2) Loss Function and Hardware: We employed the **Huber Loss** ($\delta = 0.5$) function for optimization. This loss function is particularly advantageous for tailored distributions, such as stress distribution, as it is robust to outliers and behaves like a combination of Mean Squared Error (MSE) and Mean Absolute Error (MAE).

The model was trained on an NVIDIA L4 GPU (via Google Colab Pro), with a total execution time of approximately 2 hours.