Supplementary Material: Geometry-Aware Abdominal Aortic Aneurysm Digital Twin for Patient-Specific Wall Stress Mapping

Julian Carvajal Rico[1], Victor De Oliveira[2], Satish C. Muluk[3], Mark K. Eskandari[4], Vikram S. Kashyap[5], and Ender A. Finol[1,*]

[1]Department of Mechanical, Aerospace, and Industrial Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[2]Department of Statistics and Data Science, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[3]Department of Thoracic and Cardiovascular Surgery, Allegheny Health Network, Allegheny General Hospital, Pittsburgh, PA 15212, USA
[4]Feinberg School of Medicine, Northwestern University, Chicago, IL 60611, USA
[5]Cardiovascular Health, Corewell Health, Grand Rapids, MI 49503, USA
[*]Corresponding Author: ender.finol@utsa.edu

Appendix

This supplementary material provides additional implementation details to support reproducibility of the reported numerical experiments. It includes the full set of training hyperparameters, hardware configuration, and explicit pseudo-code for each GNN architecture used for AAA wall stress prediction.

A Pseudo-codes for the GNN models

This section provides step-by-step pseudo-code for the three GNN architectures (GGCN, EGNN, and the local GT) used in this study, including data pre-processing, training, and evaluation loops.

A.1 Gated Graph Convolution Network (GGCN)

Algorithm 1 Training residual GGCN for AAA wall stress prediction

Require: Dataset of graphs $\{G_m\}_{m=1}^{M}$ with node features x, positions pos, graph attributes g, and FEA stresses y_{FEA}.

Convert all edge indices to undirected.
For each graph G_m, add stress-based node weights w using local stress quantiles (50/80/95%).
Compute graph-level targets: mean stress and $p99$ stress; build stratified K-fold splits using quantile bins of (mean, $p99$) and ILT class.
for each fold $f = 1, \ldots, K$ **do**
    Split graphs into train and test indices based on stratified folds.
    Compute feature scalers on train graphs: means and stds for x, y_{FEA}, g, and pos.
    Construct train and test datasets:
     For each sampled graph, optionally apply graph-level augmentation to g and selected x, and optional stress-stratified node subsampling (3 bins in stress).
     Normalize x, g, pos with train scalers; normalize y_{FEA} to y.
     Concatenate normalized x, broadcast g, and pos into final node feature h.
    Initialize residual GGCN:
     Input linear layer: $h \to \mathbb{R}^{d_{\text{hid}}}$.
     L GatedGraphConv layers with LayerNorm, ReLU, dropout, and residual connections.
     Output linear layer: $\mathbb{R}^{d_{\text{hid}}} \to \mathbb{R}$ (node-specific stress).
    Initialize Adam optimizer and ReduceLROnPlateau scheduler; set best loss $= \infty$.
    for epoch $= 1, \ldots,$ max_epochs **do**
        Training loop:
        for each batch of graphs **do**
            Forward pass: $\hat{y} = \text{GGCN}(h, \text{edge\_index})$ (normalized stresses).
            Node-specific loss: $L_{\text{node}} = \mathbb{E}[w \cdot (\hat{y} - y)^2]$.
            Mean-stress loss: $L_{\text{mean}} = (\text{mean}(\hat{y}) - \text{mean}(y))^2$.
            Approximate $p99$ of y and $\hat{y}$ using top-k averaging; define $L_{p99} = (p99(\hat{y}) - p99(y))^2$.
            Pinball loss at $q = 0.99$: $L_{\text{pin}} = \mathbb{E}[\max(q(y - \hat{y}), (q-1)(y - \hat{y}))]$.
            Total loss: $L = L_{\text{node}} + \lambda_{\text{mean}} L_{\text{mean}} + \lambda_{p99} L_{p99} + \lambda_{\text{pin}} L_{\text{pin}}$.
            Backpropagate and update model parameters with Adam.

end for
Update learning-rate scheduler using training loss.
Apply early stopping if training loss has not improved for a fixed patience.
end for
Save best-performing model.
Evaluate on train and test loaders:
De-normalize predictions.
Report node-specific MSE, RE, and R^2.
Report graph-level RE for mean stress and $p99$ stress.
end for

A.2 Equivariant Graph Neural Network (EGNN)

Algorithm 2 Training residual EGNN for AAA wall stress prediction

Require: Dataset of graphs $\{G_m\}_{m=1}^{M}$ with node features x, positions pos, graph attributes g, and FEA stresses y_{FEA}.

Convert all edge indices to undirected.
For each graph G_m, add stress-based node weights w using local stress quantiles (50/80/95%).
Compute graph-level targets: mean stress and $p99$ stress; build stratified K-fold splits using quantile bins of (mean, $p99$) and ILT class.
for each fold $f = 1, \ldots, K$ **do**
Split graphs into train and test indices based on stratified folds.
Compute feature scalers on train graphs: means and stds for x, y_{FEA}, g, and pos.
Construct train and test datasets:
Normalize x, g, pos with train scalers; normalize y_{FEA} to y.
Concatenate normalized x, broadcast g, and pos into final node feature h.
Initialize residual EGNN:
Input linear layer: $h \to \mathbb{R}^{d_{\text{hid}}}$.
For each layer and edge $i \leftarrow j$:

$$\Delta x_{ij} = x_i - x_j,\ d_{ij}^2 = \|\Delta x_{ij}\|^2.$$
$$m_{ij} = \phi_e(h_i, h_j, d_{ij}^2).$$
$$\text{Aggregate } m_i = \textstyle\sum_j m_{ij}.$$
$$h_i^{\text{new}} = \phi_h(h_i, m_i).$$
$$c_{ij} = \phi_x(m_{ij}),\ \Delta x_i = \textstyle\sum_j c_{ij} \Delta x_{ij}.$$
$$x_i^{\text{new}} = x_i + \Delta x_i.$$

Apply LayerNorm, ReLU, dropout, and residual skips.
Output linear layer maps hidden features to node-specific normalized stress.
for epoch $= 1, \ldots,$ max_epochs **do**
for each batch of graphs **do**
Forward pass: $\hat{y} = \text{EGNN}(h, \text{edge\_index}, pos)$.
Compute L_{node}, L_{mean}, L_{p99}, and L_{pin} as in Algorithm 1.
Backpropagate and update parameters with Adam.
end for
Update scheduler and apply early stopping.
end for
Save best-performing model and evaluate de-normalized node- and graph-level errors.
end for

A.3 Graph Transformer (GT)

Algorithm 3 Training local Graph Transformer for AAA wall stress prediction

Require: Dataset of graphs $\{G_m\}_{m=1}^{M}$ with node features x, positions pos, graph attributes g, and FEA stresses y_{FEA}.

Convert all edge indices to undirected.
Add stress-based node weights w using local stress quantiles.
Build stratified K-fold splits using graph-level stress summaries and ILT class.
for each fold $f = 1, \ldots, K$ **do**
Split graphs into train and test indices.
Compute train-only scalers for x, y_{FEA}, g, and pos.

Construct train and test datasets with normalization, augmentation, and stress-stratified node subsampling.
Concatenate normalized node features, broadcast graph attributes, and coordinates into h.
Initialize local Graph Transformer:
    Input linear layer: $h \to \mathbb{R}^{d_{\text{hid}}}$.
    L local multi-head self-attention layers restricted to graph connectivity.
    LayerNorm, ReLU, dropout, and residual connections after each attention block.
    Output linear layer maps hidden features to node-specific normalized stress.
for epoch $= 1, \ldots,$ max_epochs **do**
    for each batch of graphs **do**
        Forward pass: $\hat{y} = \text{GT}(h, \text{edge\_index})$.
        Compute weighted node-specific MSE.
        Compute mean-stress penalty, $p99$ penalty, and $q = 0.99$ pinball loss.
        Combine loss terms and update parameters with Adam.
    end for
    Update ReduceLROnPlateau scheduler and apply early stopping.
end for
Save best-performing model and evaluate de-normalized node- and graph-level errors.
end for

B Training Hyperparameters and Hardware Configuration

Table B1. Training hyperparameters and hardware configuration used for the GNN wall stress prediction experiments.

Model	Hidden dim.	Layers	Dropout	Batch size	Learning rate
GGCN	128	7	0.25	2	1×10^{-3}
EGNN	128	7	0.25	2	1×10^{-3}
GT	128	6	0.25	2	1×10^{-3}

Optimizer	Weight decay	Loss	q	λ_{mean}	λ_{p99}
Adam	1×10^{-5}	Tail-aware composite	0.99	1.0	1.0

Random seed	Folds	Max epochs	Scheduler	Early stopping patience	Augmentation	Subsampling
11	5	1500	ReduceLROnPlateau	50 epochs	Yes (strength 0.08)	Yes (ratio 0.5)

Hardware	GPU	CPU	RAM
ARC UTSA	Tesla V100S-PCIE-32GB (CUDA 12.3)	$2 \times$ 20-core CPU	384 GB