Graph Neural Networks

Presenters:

Keivan Faghih Niresi
Leandro Von Krannichfeldt

École
polytechnique
fédérale
de Lausanne

Background

EPFL, Switzerland
PhD (2nd year) in Civil and Environmental Engineering, IMOS Lab
- Research topics: Computational Sensing, Graph Representation Learning, Intelligent Infrastructure Systems

National Tsing Hua University, Taiwan
MSc in Communications Engineering
- Research topics: Inverse Problems, Hyperspectral Imaging, Convex Optimization

University of Guilan, Iran
BSc in Electrical Engineering
- Research topics: Signal Processing, Optical Wireless Communication

Introduction

Deep Learning Revolution

Transformed tasks like image classification, video processing, and speech recognition.
Traditional tasks deal with data in Euclidean space.

Challenges with Non-Euclidean Data

Increasing applications with data represented as graphs (complex relationships & interdependencies).
Graph data pose significant challenges for existing machine learning algorithms.

Emergence of Graph Neural Networks (GNNs)

GNNs vs. Network Embedding

Keivan Faghih Niresi

Network Embedding:

- Converts nodes into low-dimensional vectors.
- Focuses on preserving network structure for tasks like classification and clustering.
- Includes deep and non-deep learning methods (e.g., matrix factorization, random walks).

GNNs:

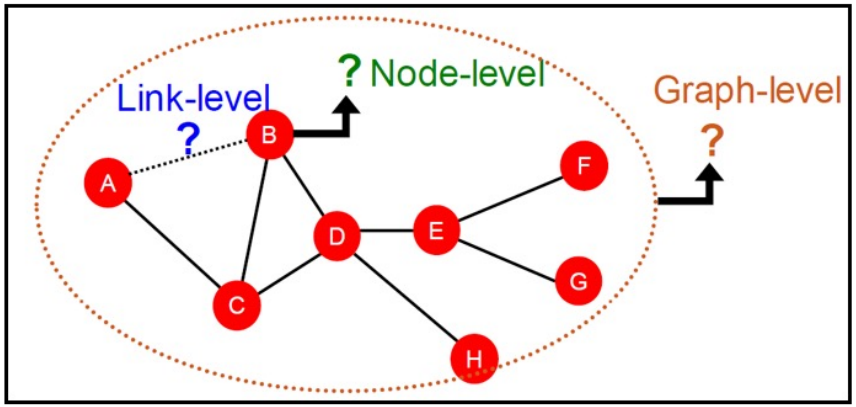
- End-to-end deep learning models for various graph tasks.
- Extract high-level representations.
- Can solve network embedding through a graph autoencoder.

Key Difference:

GNNs are a group of neural network models which are designed for various tasks while network embedding covers various kinds of methods targeting the same task.

Taxonomies

Category	
Recurrent Graph Neural Networks (RecGNNs)	
Convolutional Graph Neural Networks (ConvGNNs)	Spectral methods
	Spatial methods
Graph Autoencoders (GAEs)	Network Embedding
	Graph Generation
Spatial-temporal Graph Neural Networks (STGNNs)	



Can be trained in:

- Supervised manner
- Semi-supervised manner
- Unsupervised manner

Spectral-Based GCNs

- Spectral-based methods have a solid mathematical foundation in graph signal processing.
- They assume graphs to be undirected.

$$\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

Convolution

$$\begin{aligned}\mathbf{x} *_G \mathbf{g} &= \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g})) \\ &= \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g}),\end{aligned}$$

If $\mathbf{g}_\theta = diag(\mathbf{U}^T\mathbf{g})$

Learnable

$$\mathbf{g}_\theta = \mathbf{\Theta}_{i,j}^{(k)}$$

Spectral-based GNNs

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x}.$$

The graph convolutional layer of Spectral CNN is defined as $\mathbf{H}_{:,j}^{(k)} = \sigma(\sum_{i=1}^{f_{k-1}} \mathbf{U}\mathbf{\Theta}_{i,j}^{(k)}\mathbf{U}^T\mathbf{H}_{:,i}^{(k-1)}) \quad (j = 1, 2, \cdots, f_k),$

Limitations

- Any perturbation to a graph results in a change of eigenbasis.
- Learned filters are domain dependent, meaning they cannot be applied to a graph with a different structure.
- Eigen-decomposition requires cubic computational complexity

Keivan Faghih Niresi

ChebNet

- Chebyshev Spectral CNN (ChebNet) approximates the filter by Chebyshev polynomials of the diagonal matrix of eigenvalues

$$\mathbf{g}_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{\mathbf{\Lambda}})$$

$$\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_{max} - \mathbf{I_n} \qquad [-1, 1]$$

$$T_i(\mathbf{x}) = 2\mathbf{x}T_{i-1}(\mathbf{x}) - T_{i-2}(\mathbf{x})$$

$$T_0(\mathbf{x}) = 1 \text{ and } T_1(\mathbf{x}) = \mathbf{x}.$$

Convolution with a defined Chebyshev filter

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\left(\sum_{i=0}^{K} \theta_i T_i(\tilde{\mathbf{\Lambda}})\right)\mathbf{U}^T\mathbf{x},$$

$$T_i(\tilde{\mathbf{L}}) = \mathbf{U}T_i(\tilde{\mathbf{\Lambda}})\mathbf{U}^T \qquad \tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I_n}$$

$$\mathbf{x} *_G \mathbf{g}_\theta = \sum_{i=0}^{K} \theta_i T_i(\tilde{\mathbf{L}})\mathbf{x},$$

CayleyNet

- Filters defined by ChebNet are localized in space, which means filters can extract local features independently of the graph size.

CayleyNet $$\mathbf{x} *_G \mathbf{g}_\theta = c_0\mathbf{x} + 2Re\{\sum_{j=1}^{r} c_j(h\mathbf{L} - i\mathbf{I})^j(h\mathbf{L} + i\mathbf{I})^{-j}\mathbf{x}\},$$

Re(·) returns the real part of a complex number,
c_0 is a real coefficent,
c_j is a complex coefficent,
h is a parameter which controls the spectrum of a Cayley filter

GCN

- Graph Convolutional Network (GCN) introduces a first-order approximation of ChebNet. Assuming $K = 1$ and $\lambda_{max} = 2$

$$\mathbf{x} *_G \mathbf{g}_\theta = \theta_0 \mathbf{x} - \theta_1 \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{x}$$

If $\theta = \theta_0 = -\theta_1$

$$\mathbf{x} *_G \mathbf{g}_\theta = \theta(\mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}})\mathbf{x}.$$

Finally for multichannel input and output

$$\mathbf{H} = \mathbf{X} *_G \mathbf{g}_{\mathbf{\Theta}} = f(\bar{\mathbf{A}}\mathbf{X}\mathbf{\Theta}) \qquad \bar{\mathbf{A}} = \mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

Normalization trick to avoid numerical instability $\mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$

Replace $\bar{\mathbf{A}} = \mathbf{I_n} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ by $\bar{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$ with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}$

GCN can be also interpreted as a spatial-based method

Spatial-based GCNs

Message Passing Neural Network (MPNN) outlines a general framework of spatial-based ConvGNNs.

It treats graph convolutions as a message passing process in which information can be passed from one node to another along edges directly.

MPNN runs K-step message passing iterations to let information propagate further

$$\mathbf{h}_v^{(k)} = U_k(\mathbf{h}_v^{(k-1)}, \sum_{u \in N(v)} M_k(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{x}_{vu}^e)),$$

GraphSAGE

GraphSage adopts sampling to obtain a fixed number of neighbors for each node.

It performs graph convolutions by

$$\mathbf{h}_v^{(k)} = \sigma(\mathbf{W}^{(k)} \cdot f_k(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)}, \forall u \in S_{\mathcal{N}(v)}\})),$$

$f_k(\cdot)$ **is an aggregation function,** should be invariant to the permutations of node orderings such as a mean, sum or max function.

$S_{\mathcal{N}(v)}$ Random sample on node neighbors

Keivan Faghih Niresi

Graph Isomorphism Networks

Graph Isomorphism Network (GIN) finds that previous MPNN-based methods are incapable of distinguishing different graph structures based on the graph embedding they produced.

GIN adjusts the weight of the central node by a learnable parameter.

It performs graph convolutions by

$$\mathbf{h}_v^{(k)} = MLP((1+\epsilon^{(k)})\mathbf{h}_v^{(k-1)} + \sum_{u \in N(v)} \mathbf{h}_u^{(k-1)}),$$

Discriminative and represnetation power of GIN is equal to the power of Weisfelier-Lehman test.

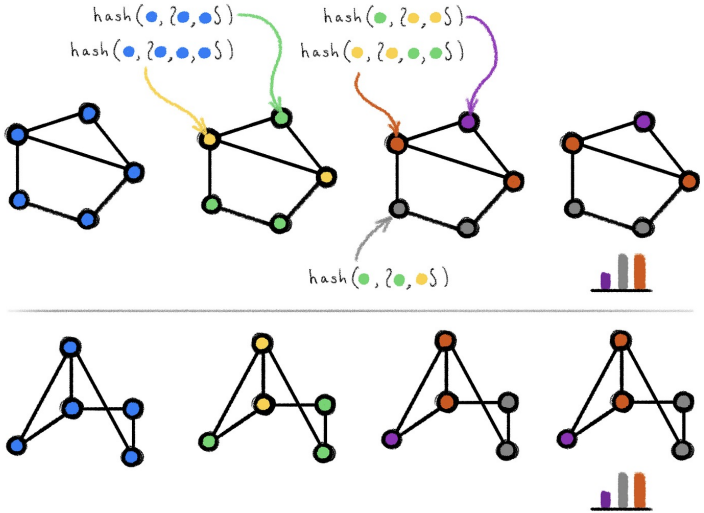
Graph isomorphism problem: Are two graphs topologically identical?

The WL test iteratively (1) aggregates the labels of nodes and their neighborhoods, and (2) hashes the aggregated labels into unique new labels. The algorithm decides that two graphs are non-isomorphic if at some iteration the labels of the nodes between the two graphs differ.

Color Refinement (1-WL test)

Take as input two graphs *G* and *H* and output a certificate that they are different or 'I don't know'.

This means that if the heuristic is able to tell *G* and *H* apart, then they are definitely different, but the other direction does not hold.



Attentional Graph Neural Networks

Let's consider more general form of GCN:

$$\vec{h}_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij} \ \mathbf{W}\vec{h}_j\right)$$

GCN defines α_{ij} explicitly! $\frac{1}{\sqrt{deg(v_i)deg(v_j)}}$ Possible shortcoming!

Instead, we let α_{ij} be computed implictly

a is a learnable

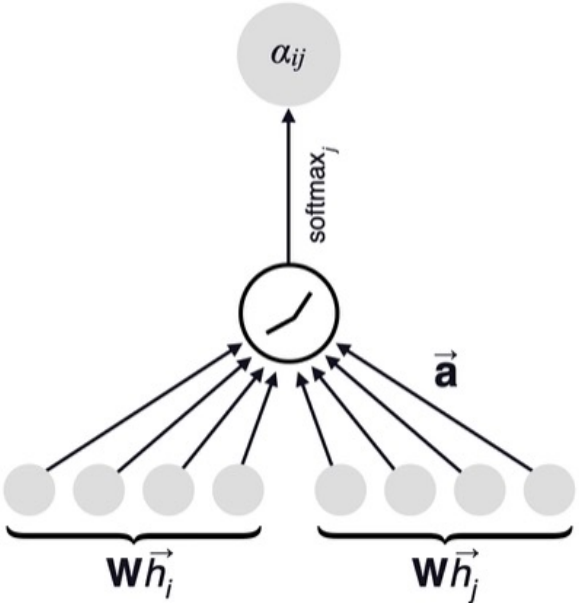
$$a_{ij} = a(\vec{h}_i, \vec{h}_j, \vec{e}_{ij})$$

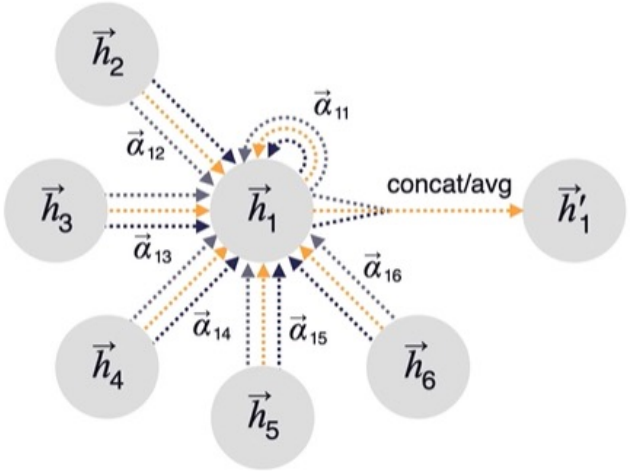
$$\alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{k \in N_i} \exp(a_{ik})}$$

We arrived at Graph Attention Networks (GAT)!

Keivan Faghih Niresi

GAT and GATv2





GAT (Veličković et al., 2018): $\text{LeakyReLU}\left(\boldsymbol{a}^\top \cdot [\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_j]\right)$

GATv2 (our fixed version): $\boldsymbol{a}^\top \text{LeakyReLU}\left(\boldsymbol{W} \cdot [\boldsymbol{h}_i \| \boldsymbol{h}_j]\right)$

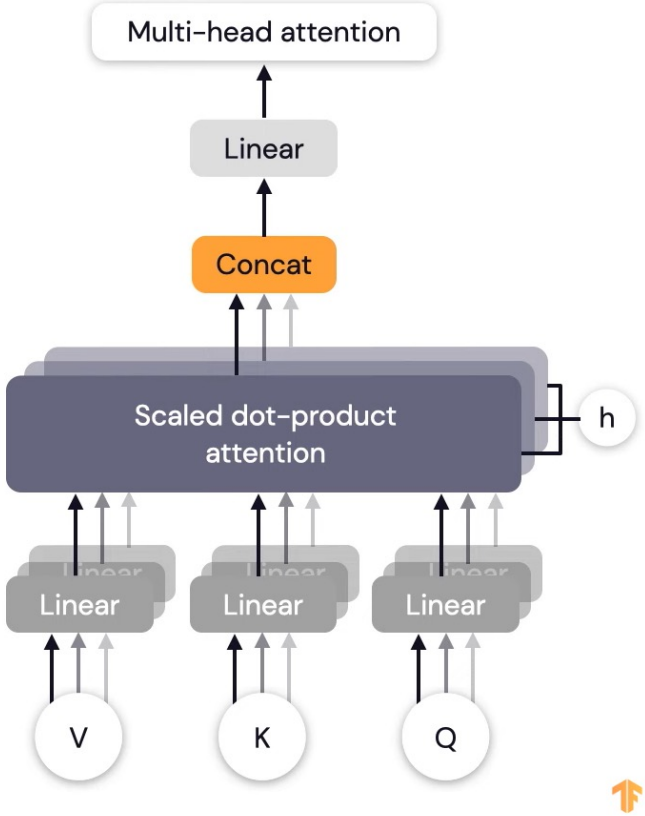
Relations to Transformers

Transformers **are** Graph Neural Networks!

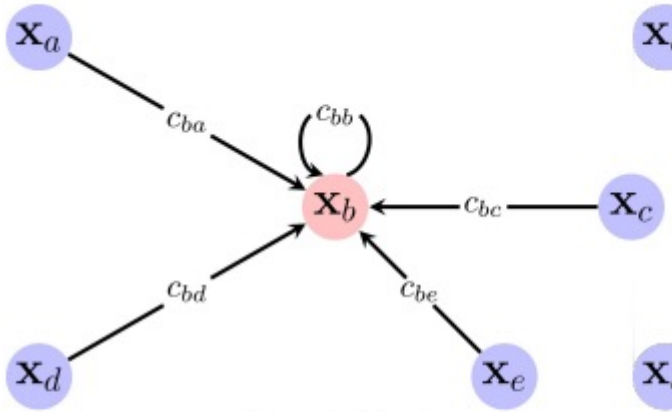
- Fully-connected graph
- Message function == Sender node features
- Aggregation == Attention

The sequential structural information is injected through the **positional embeddings**. Dropping them yields a fully-connected GAT model.

See Joshi (The Gradient; 2020).

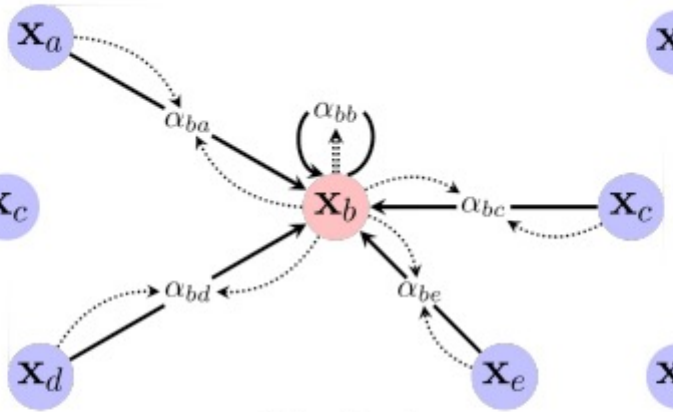


Recap on GNNs



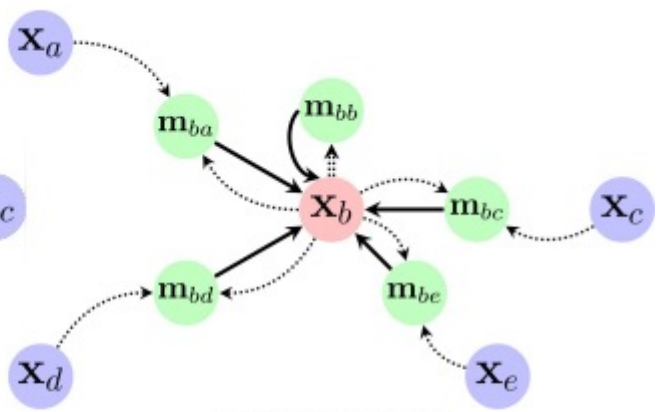
Convolutional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$



Attentional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$



Message-passing

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Spectral vs. Spatial

Spectral Models:

Foundation:
Built on graph signal processing.

Efficiency:
Requires eigenvector computation or processing the entire graph.
Less scalable for large graphs.

Generalization:
Performs poorly on new graphs due to reliance on graph Fourier basis.
Sensitive to graph perturbations.

Graph Types:
Limited to undirected graphs.

Spatial Models:

Operation:
Directly performs convolutions in the graph domain via local information propagation.

Efficiency:
More scalable: Can be computed in batches of nodes.
Avoids full-graph processing.

Generalization:
Generalizes better to new graphs; operates locally on nodes.
Weights can be shared across different graph structures.

Graph Types:
Handles directed, heterogeneous, and signed graphs easily..

Keivan Faghih Niresi

Pooling

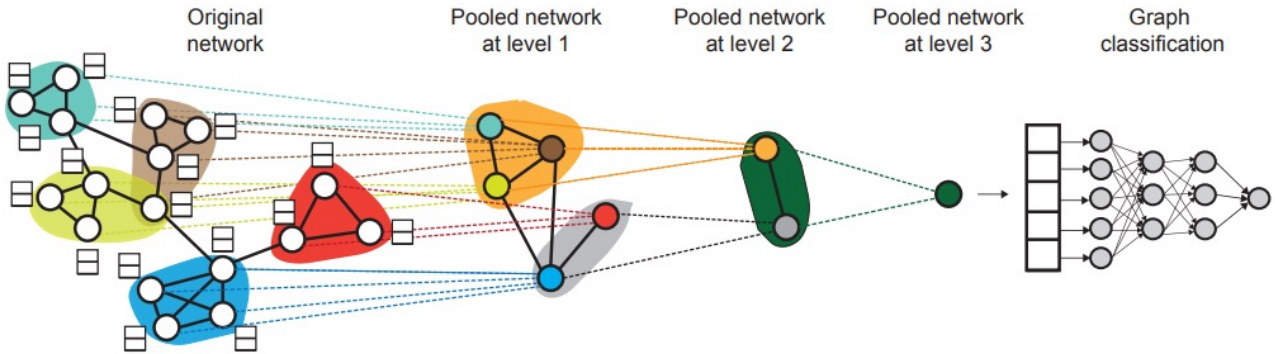
Why Down-sampling in GNNs?

Directly using all node features can be computationally expensive.
A down-sampling strategy is required to reduce parameters and prevent overfitting.

Pooling Operations in GNNs

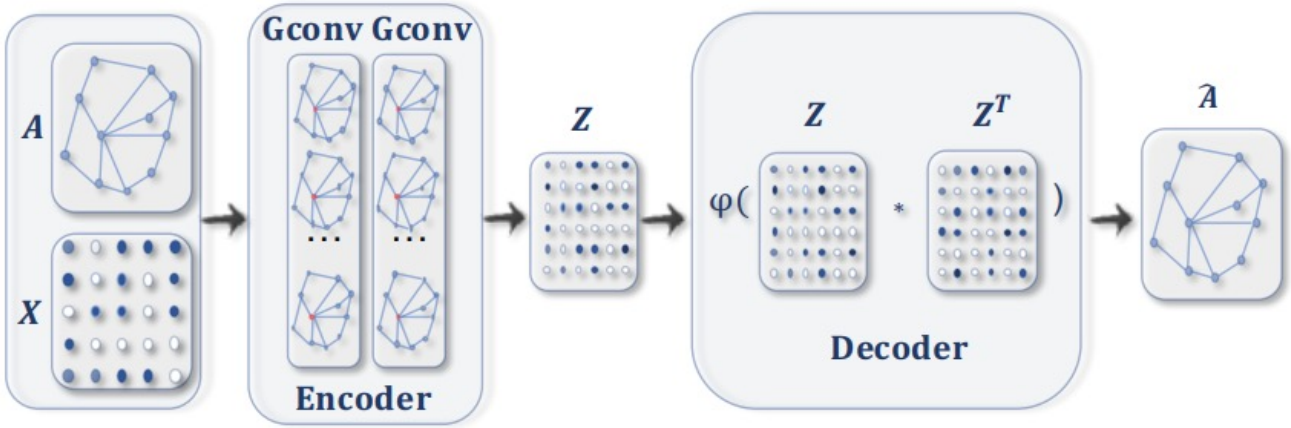
Mean/Max/Sum Pooling

DiffPool



Open Challenge: Balancing effectiveness with computational complexity in pooling strategies.

Graph Autoencoder



$$\mathbf{Z} = enc(\mathbf{X}, \mathbf{A}) = Gconv(f(Gconv(\mathbf{A}, \mathbf{X}; \mathbf{\Theta_1})); \mathbf{\Theta_2}),$$

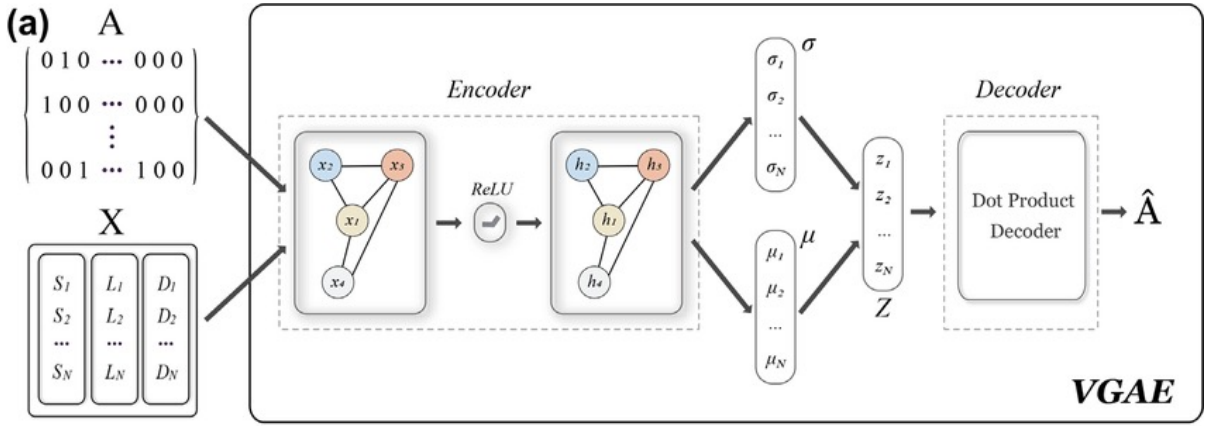
$$\hat{\mathbf{A}}_{v,u} = dec(\mathbf{z}_v, \mathbf{z}_u) = \sigma(\mathbf{z}_v^T \mathbf{z}_u),$$

Loss Function:

Negative cross entropy given the real adjacency matrix and the reconstructed adjacency matrix

Keivan Faghih Niresi

Variational Graph Autoencoder



$$L = E_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}[\log p(\mathbf{A}|\mathbf{Z})] - KL[q(\mathbf{Z}|\mathbf{X},\mathbf{A})||p(\mathbf{Z})],$$

Lower-bound reconstruction error

Theoretical Aspects: Shape of receptive field

Definition: The receptive field of a node refers to the set of neighboring nodes that contribute to its final node representation.

Growth Through Layers: Each additional spatial graph convolutional layer expands the node's receptive field, allowing it to incorporate information from increasingly distant neighbors.
Stacking layers helps nodes gather more global information.

Key Insight:
Micheli (2009) demonstrated that a finite number of graph convolutional layers allows a node's receptive field to cover all nodes in the graph.
Thus, GNNs can extract global graph information by building upon local connections.

Be careful about oversmoothing

Keivan Faghih Niresi

Permutation Invariance or Equivariance

Node-Level Tasks (Equivariance):

A GNN must be equivariant for tasks that predict properties at the node level (e.g., node classification). Equivariance ensures that changing the order of nodes in the input leads to the same change in the output.

Condition:

$$f(\mathbf{Q}\mathbf{A}\mathbf{Q}^T, \mathbf{Q}\mathbf{X}) = \mathbf{Q}f(\mathbf{A}, \mathbf{X})$$

Q: Permutation matrix

Graph-Level Tasks (Invariance):

A GNN must be invariant for tasks that predict properties of the entire graph (e.g., graph classification). Invariance ensures that reordering nodes doesn't affect the overall graph representation.
Condition:

$$f(\mathbf{Q}\mathbf{A}\mathbf{Q}^T, \mathbf{Q}\mathbf{X}) = f(\mathbf{A}, \mathbf{X})$$

Popular Benchmarks

Category	Data set	Source	# Graphs	# Nodes(Avg.)	# Edges (Avg.)	#Features	# Classes	Citation
Citation Networks	Cora	[117]	1	2708	5429	1433	7	[22], [23], [25], [41], [43], [44], [45] [49], [50], [51], [53], [56], [61], [62]
	Citeseer	[117]	1	3327	4732	3703	6	[22], [41], [43], [45], [50], [51], [53] [56], [61], [62]
	Pubmed	[117]	1	19717	44338	500	3	[18], [22], [25], [41], [43], [44], [45] [49], [51], [53], [55], [56], [61], [62] [70], [95]
	DBLP (v11)	[118]	1	4107340	36624464	-	-	[64], [70], [99]
Bio-chemical Graphs	PPI	[119]	24	56944	818716	50	121	[18], [42], [43], [48], [45], [50], [55] [56], [58], [64]
	NCI-1	[120]	4110	29.87	32.30	37	2	[25], [26], [46], [52], [57], [96], [98]
	MUTAG	[121]	188	17.93	19.79	7	2	[25], [26], [46], [52], [57], [96]
	D&D	[122]	1178	284.31	715.65	82	2	[26], [46], [52], [54], [96], [98]
	PROTEIN	[123]	1113	39.06	72.81	4	2	[26], [46], [52], [54], [57]
	PTC	[124]	344	25.5	-	19	2	[25], [26], [46], [52], [57]
	QM9	[125]	133885	-	-	-	-	[27], [69]
	Alchemy	[126]	119487	-	-	-	-	-
Social Networks	Reddit	[42]	1	232965	11606919	602	41	[42], [48], [49], [50], [51], [56]
	BlogCatalog	[127]	1	10312	333983	-	39	[18], [55], [60], [64]
Others	MNIST	[128]	70000	784	-	1	10	[19], [23], [21], [44], [96]
	METR-LA	[129]	1	207	1515	2	-	[48], [72], [76]
	Nell	[130]	1	65755	266144	61278	210	[22], [41], [50]

Recurrent Graph Neural Networks and STGNNs

Keivan Faghih Niresi

Example: GraphConvLSTM

$$
\begin{aligned}
i &= \sigma(W_{xi} *_{\mathcal{G}} x_t + W_{hi} *_{\mathcal{G}} h_{t-1} + w_{ci} \odot c_{t-1} + b_i), \\
f &= \sigma(W_{xf} *_{\mathcal{G}} x_t + W_{hf} *_{\mathcal{G}} h_{t-1} + w_{cf} \odot c_{t-1} + b_f), \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc} *_{\mathcal{G}} x_t + W_{hc} *_{\mathcal{G}} h_{t-1} + b_c), \\
o &= \sigma(W_{xo} *_{\mathcal{G}} x_t + W_{ho} *_{\mathcal{G}} h_{t-1} + w_{co} \odot c_t + b_o), \\
h_t &= o \odot \tanh(c_t).
\end{aligned}
$$

Spatial-Temporal Graph Neural Networks

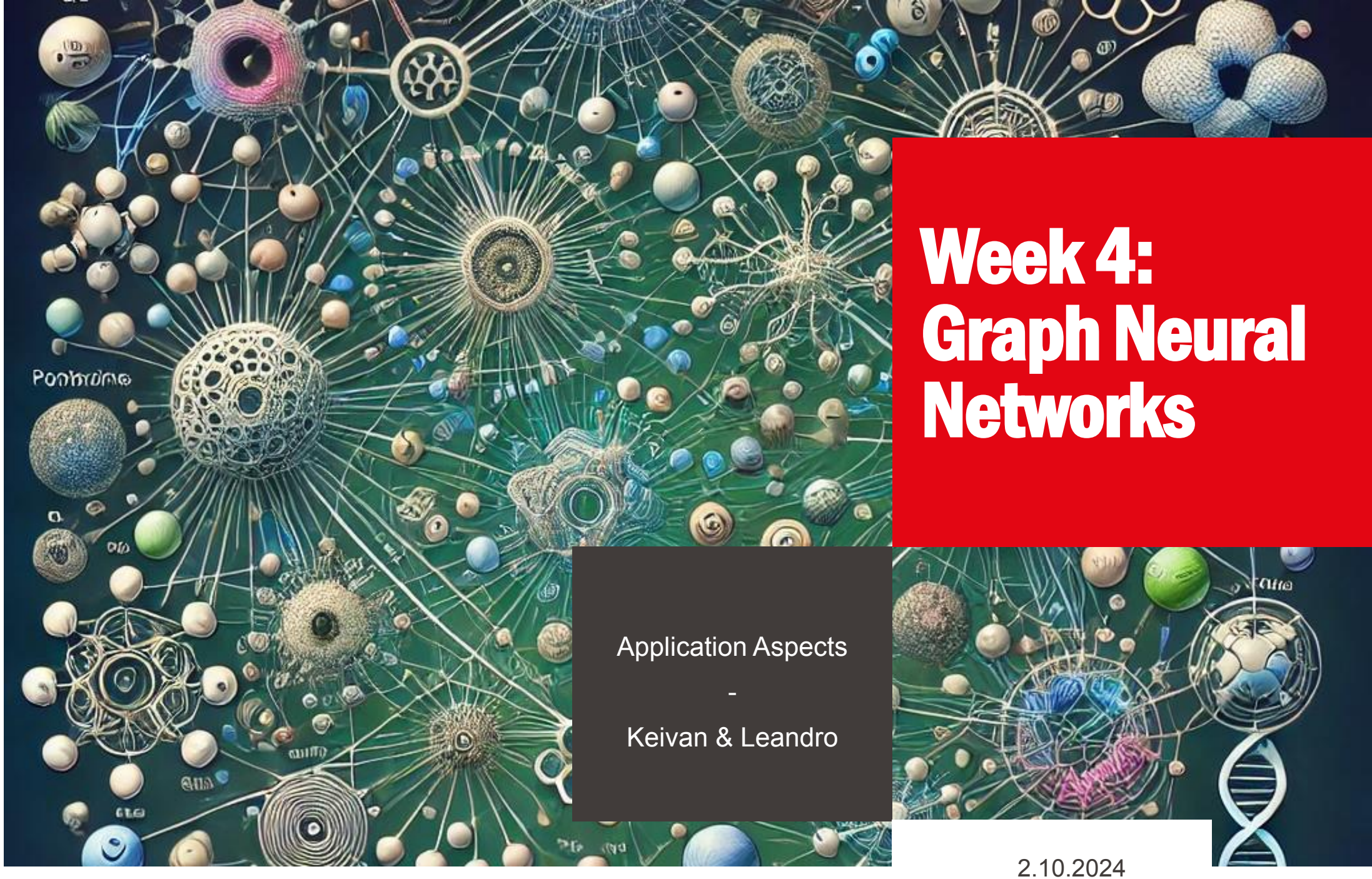
Time-then-graph (temporal module + spatial module)
Graph-then-time (spatial module + temporal module)

Future Direction

1. Model Depth
Deep learning success relies on deep architectures.
GNNs face performance drops with increasing layers.
Challenge: Too many layers cause node representations to converge to a single point.

2. Heterogeneity
Current GNNs assume homogeneous graphs.

3. Scalability Trade-off
Scalability requires sacrificing graph completeness.
Challenge: Sampling can miss influential nodes, and clustering may lose key structural patterns.



Week 4: Graph Neural Networks

Application Aspects

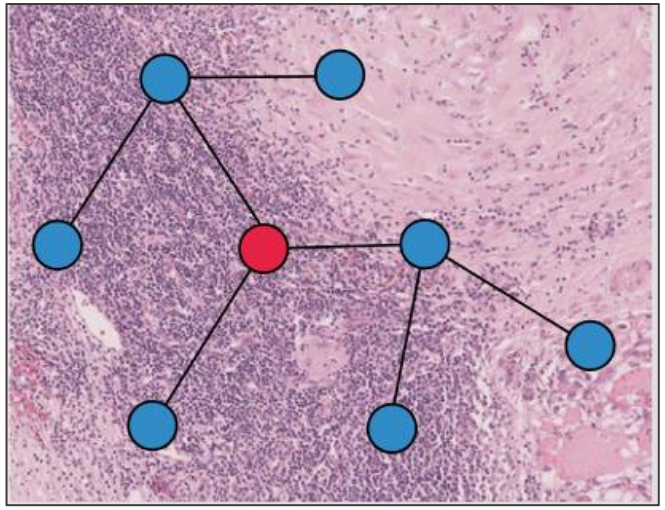
-

Keivan & Leandro

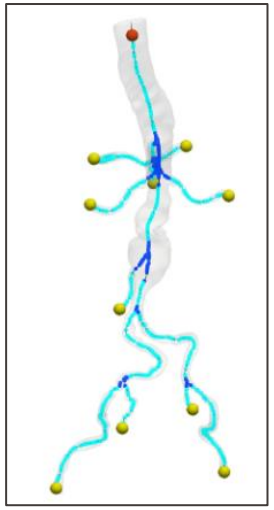
2.10.2024

Overview

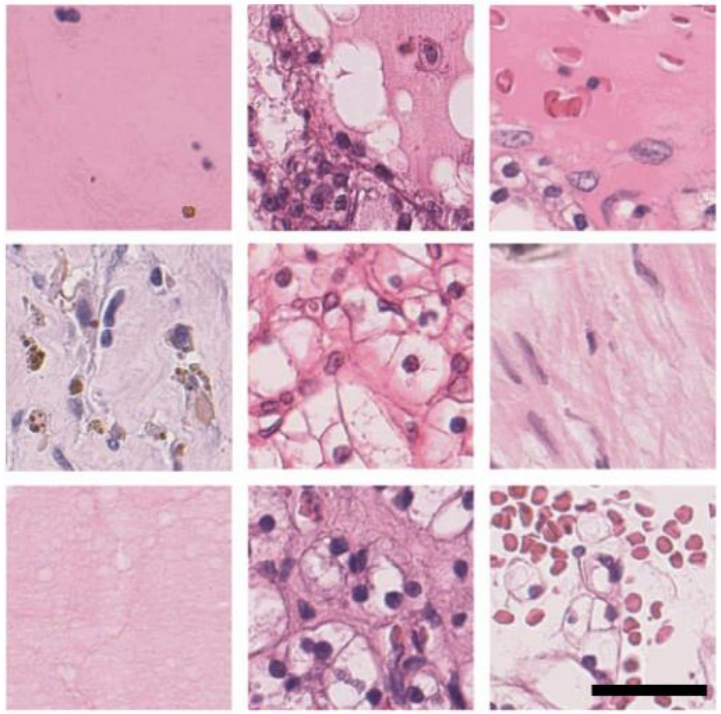
- **Tumor prognostics** from slide images with **Graph Attention Network**

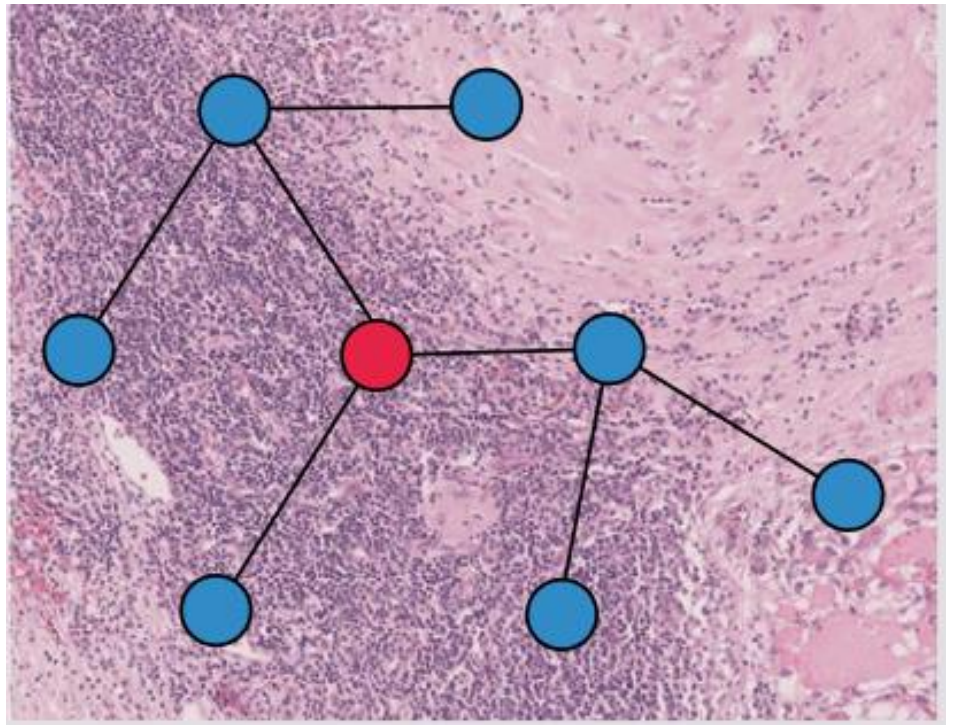


- Cardiovascular **blood flow** simulation with **physics-inspired Graph Neural Networks**

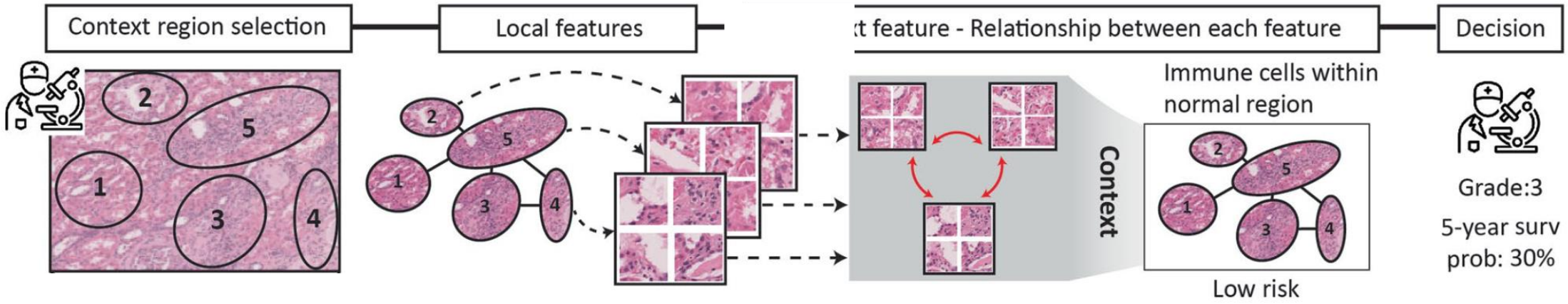


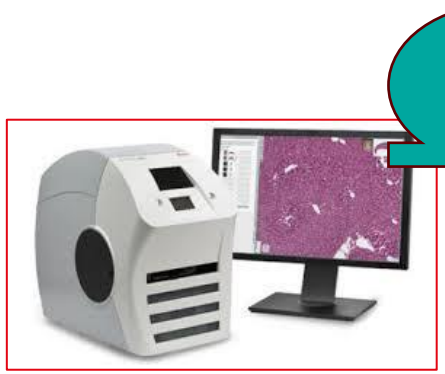
Derivation of Prognostic Contextual Histopathological Features from Whole-slide Images of Tumours via Graph Deep Learning





Introduction: Diagnostics Pipeline





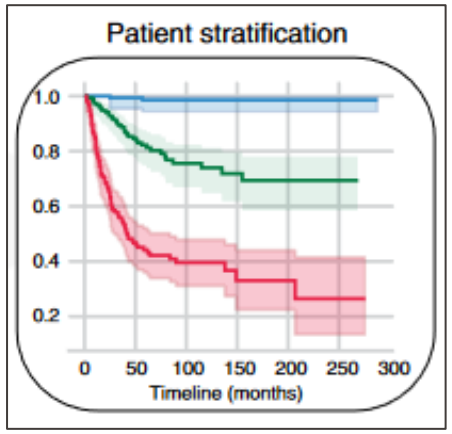
Rich tissue Infos

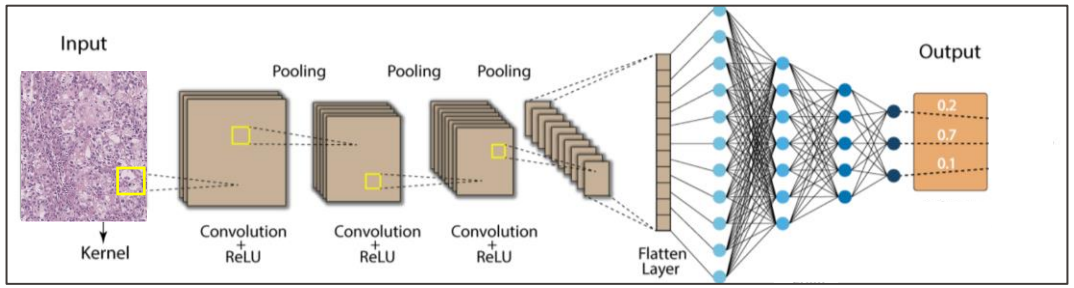
Whole slide imaging

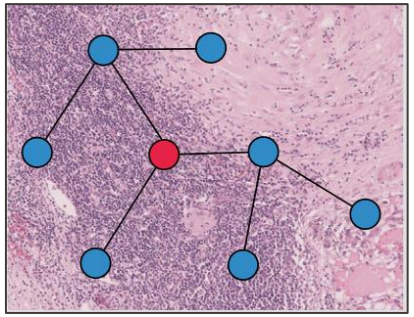
Automation of pipeline

Literature Gap

- Deep Learning advent
  - Cell type classification
  - Extraction of risk-related features
- Convolutional Neural Networks
  - Local focus on features
  - No context!
- Graph Neural Networks
  - Computational issues
  - No positional information
  - Interpretability

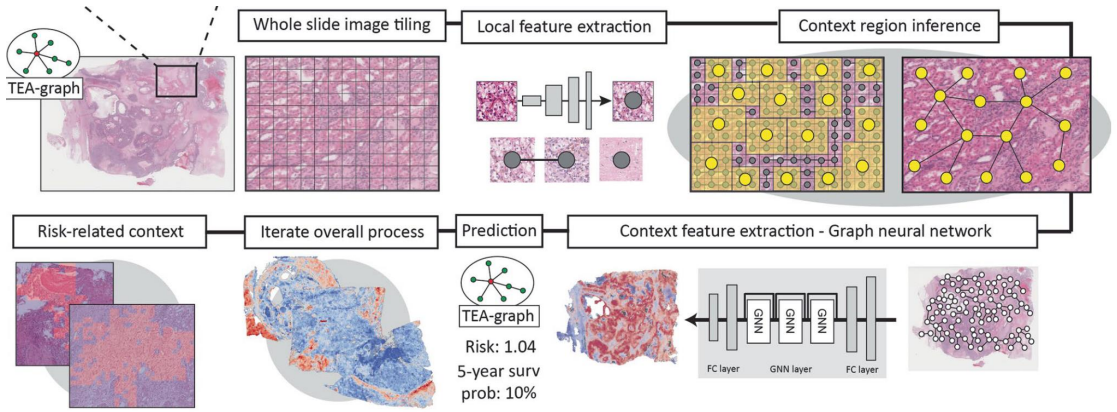




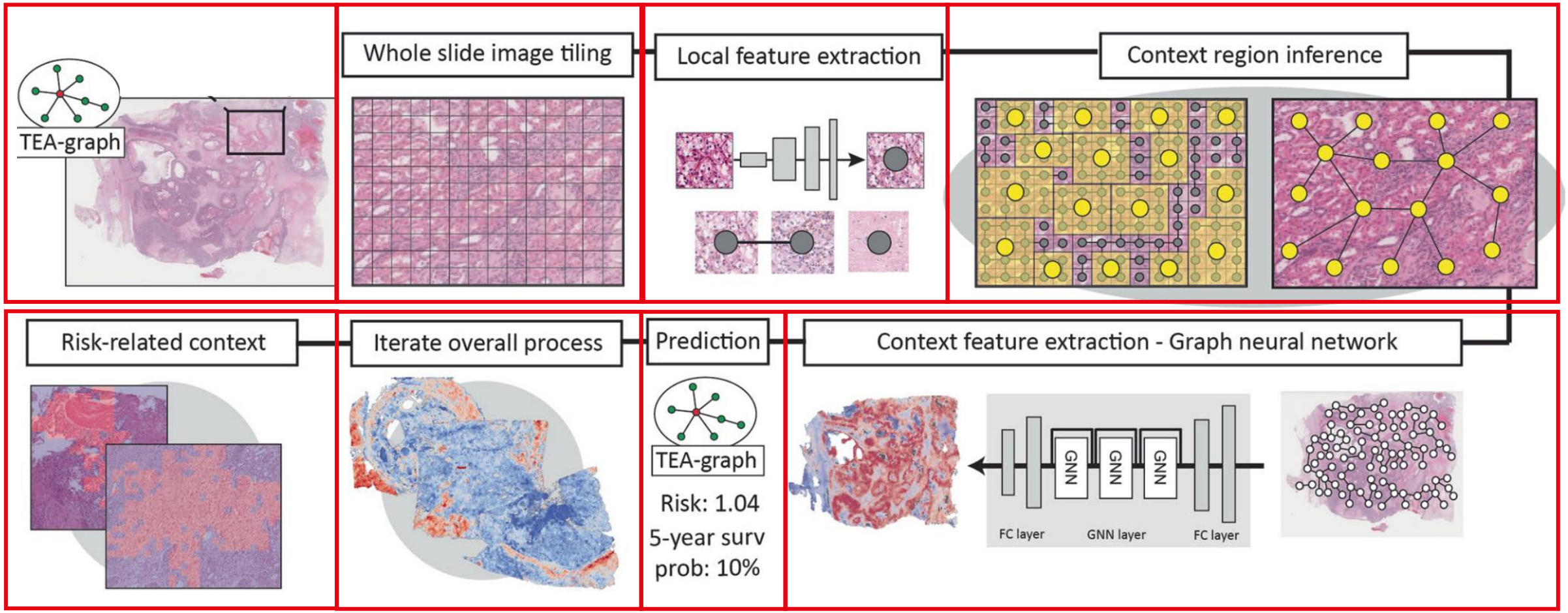


Framework

- Tumor Environment Associated Graph (TEA)
  - Graph Attention Network trained semi-supervised
  - Superpatch graph for compression
  - Interpretability with attention score and Integrated Gradient

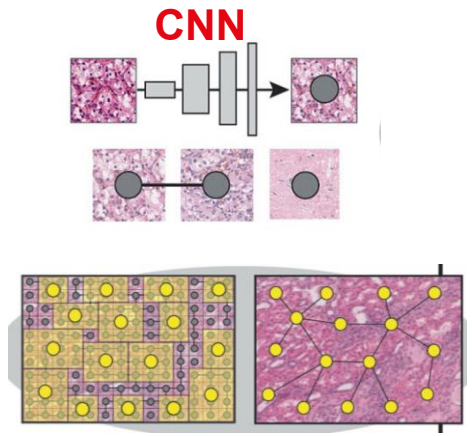


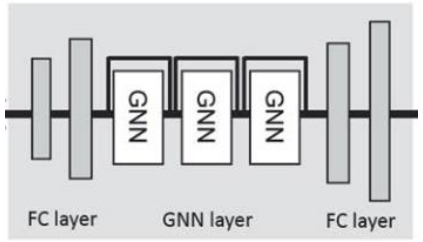
Framework

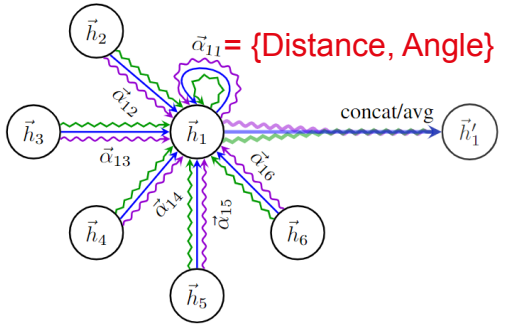

TEA-graph
Whole slide image tiling
Local feature extraction
Context region inference
Risk-related context
Iterate overall process
Prediction
TEA-graph
Risk: 1.04
5-year surv
prob: 10%
Context feature extraction - Graph neural network
GNN
GNN
GNN
FC layer
GNN layer
FC layer

Method

- Preprocessing
  - Otsu filter for background
  - Feature extraction with fine-tuned ImageNet
  - Superpatch graph construction
- Model
  - Forward Connected Layers
  - Graph attention network
  - Position as edge features
- Postprocessing
  - Integrated gradients



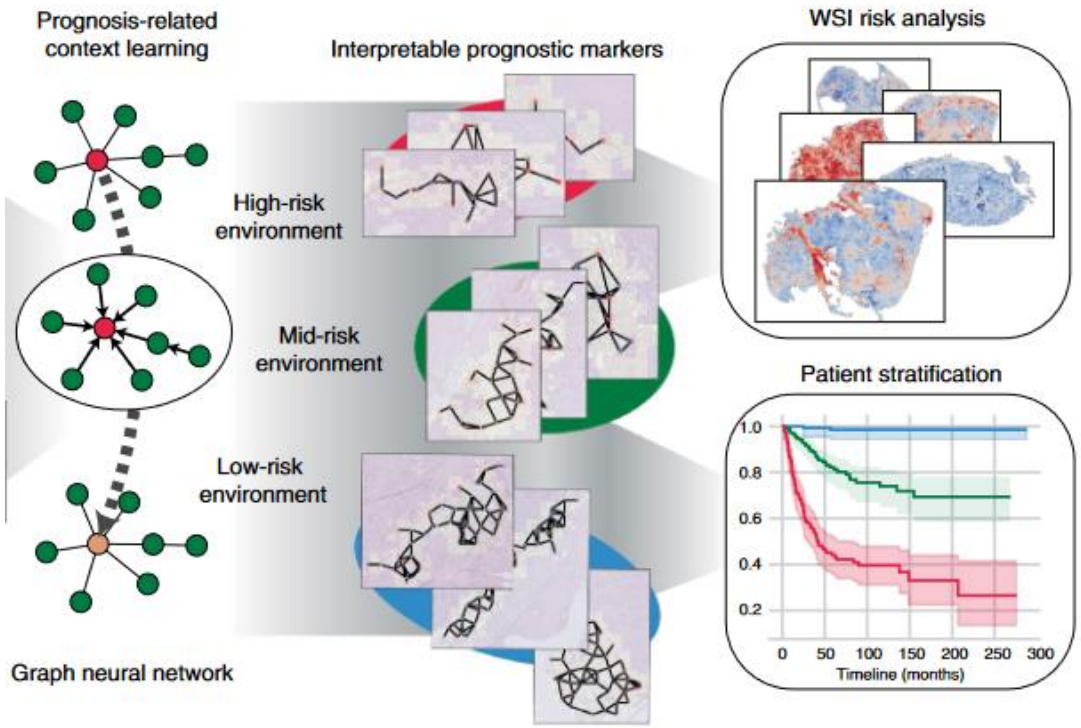




$$\mathrm{IG}\left(V^i\right) = \sum_{k=1}^{l}\left(V_k^i - \tilde{V}_k^i\right) \times \sum_{p=1}^{m} \frac{\partial GNN\left[\tilde{V} + \frac{p}{m} \times \left(V - \tilde{V}\right)\right]}{\partial V_k^i} \times \frac{1}{m}$$

Application

- Goals
  - Risk of events
  - Context features

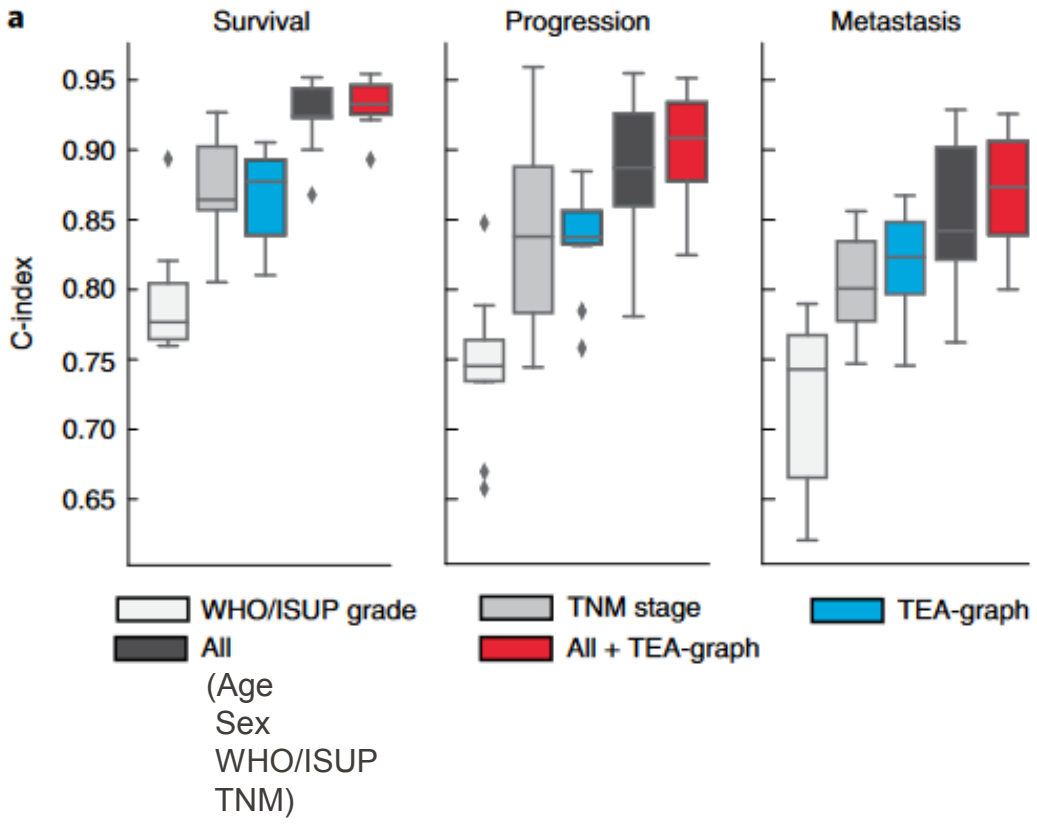


- Data
  - Lung, breast, kidney, uterine cancer
  - 831 patients
  - 70%/10%/20% for train/validation/test sets

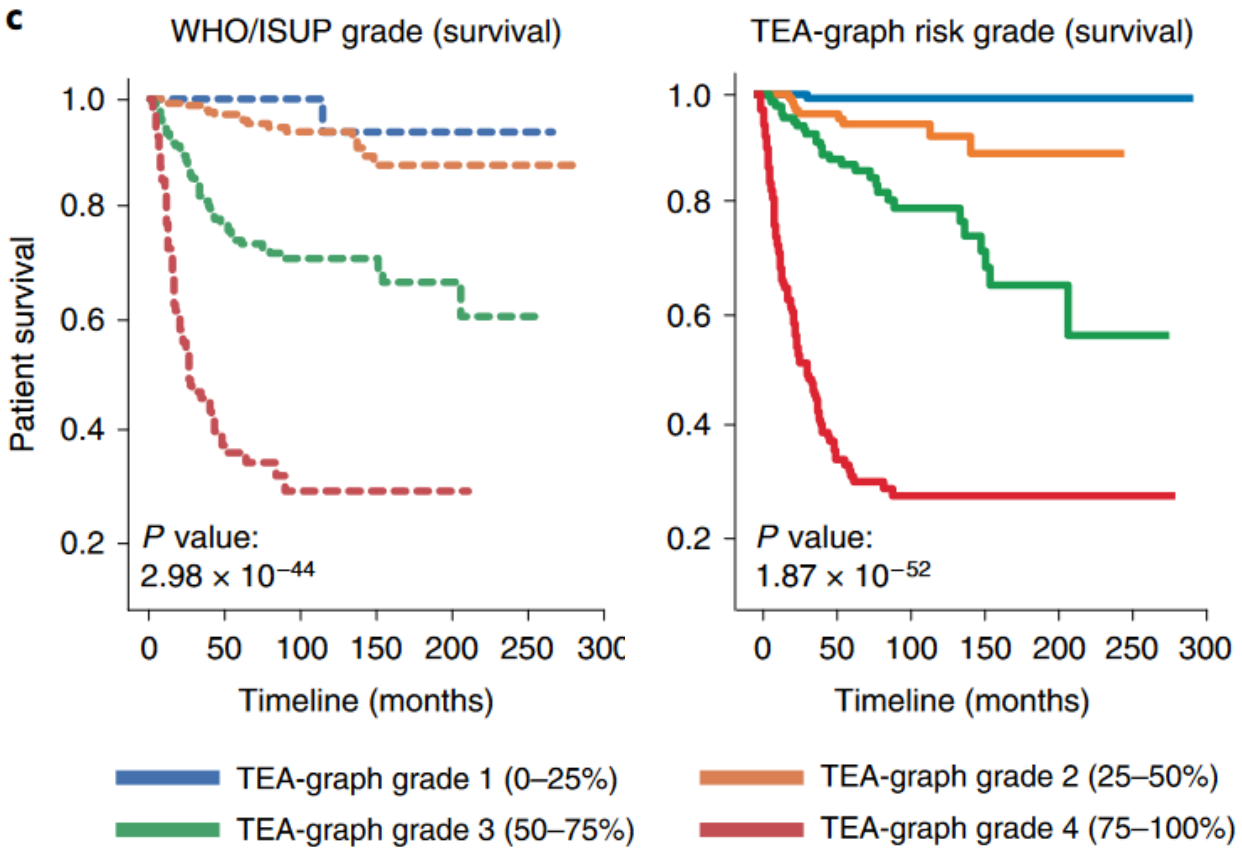
Results

- Predicting prognosis
- Correlation to known pathologic prognostic markers
- Hetergeneous contextual features

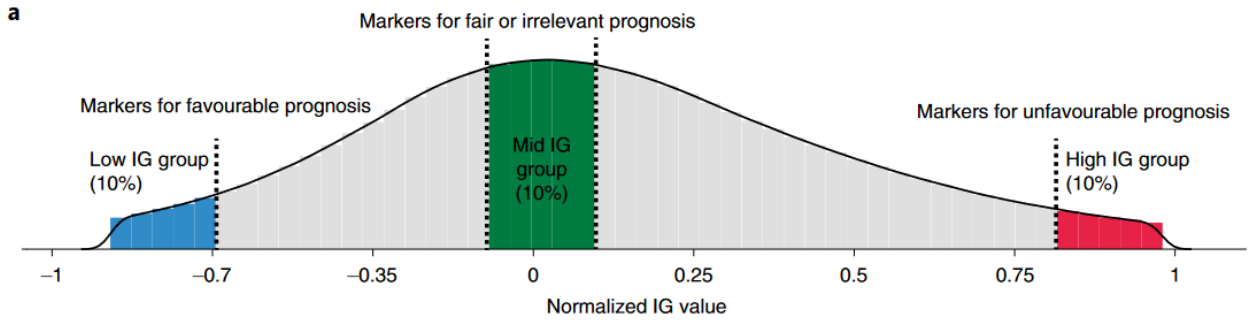
Results: Predicting Prognosis

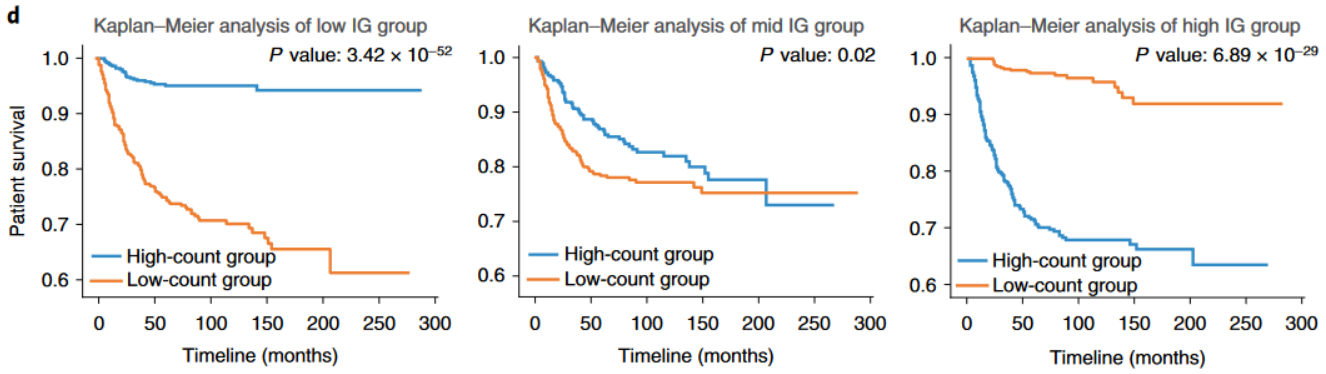


(Age
Sex
WHO/ISUP
TNM)

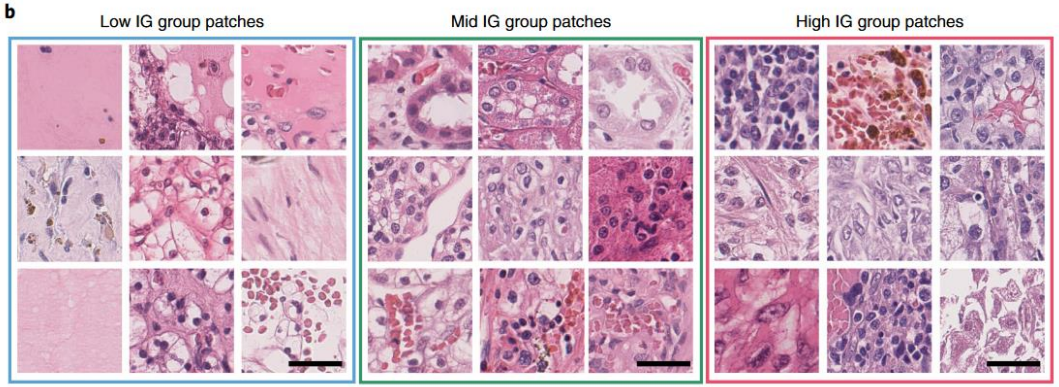


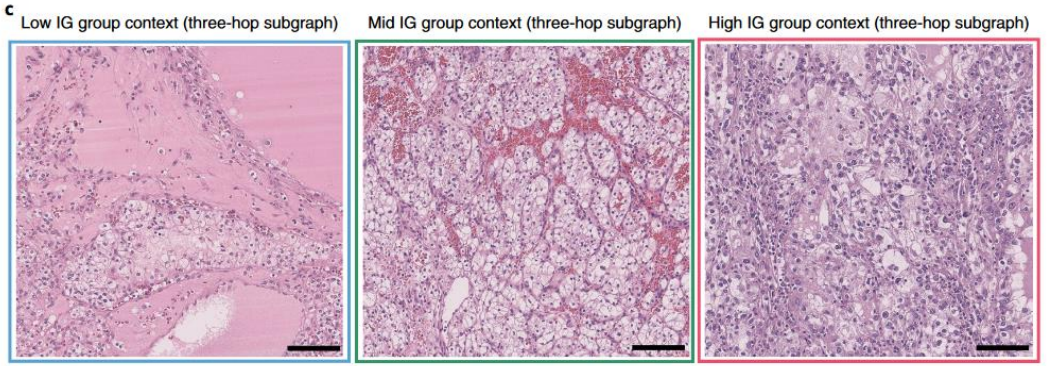
Results: Correlation to Known Pathologic Markers





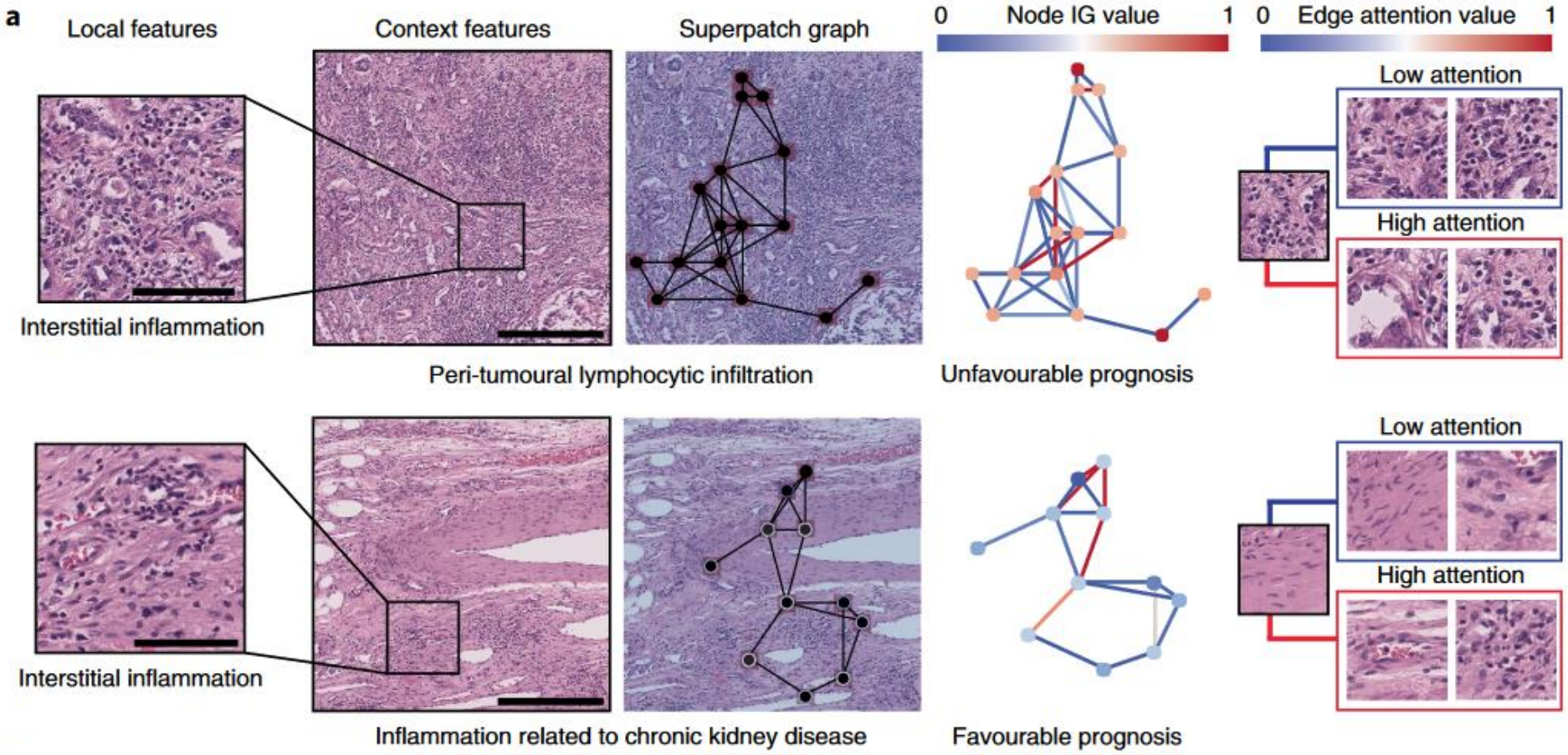
Small tumor cells Normal regions Aggressive tumor cells





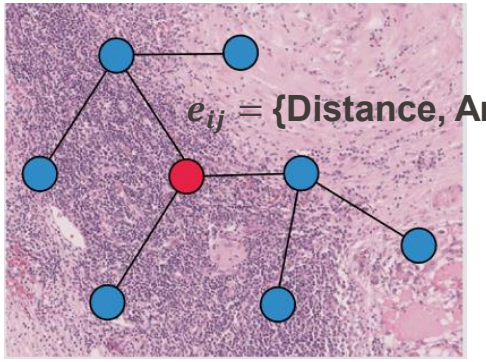
Tissue changes Capillary growth Cell deaths

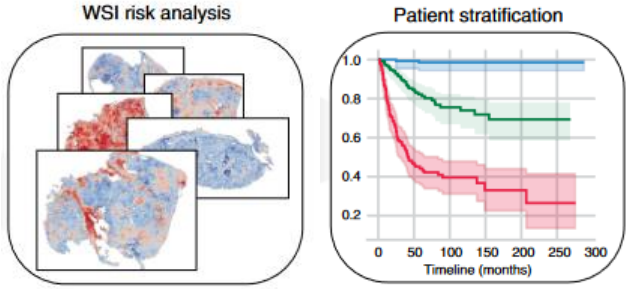
Results: Heterogeneous Contextual Features

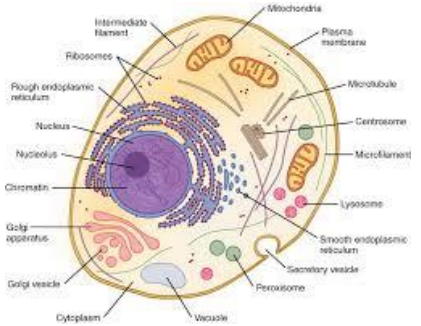


Summary

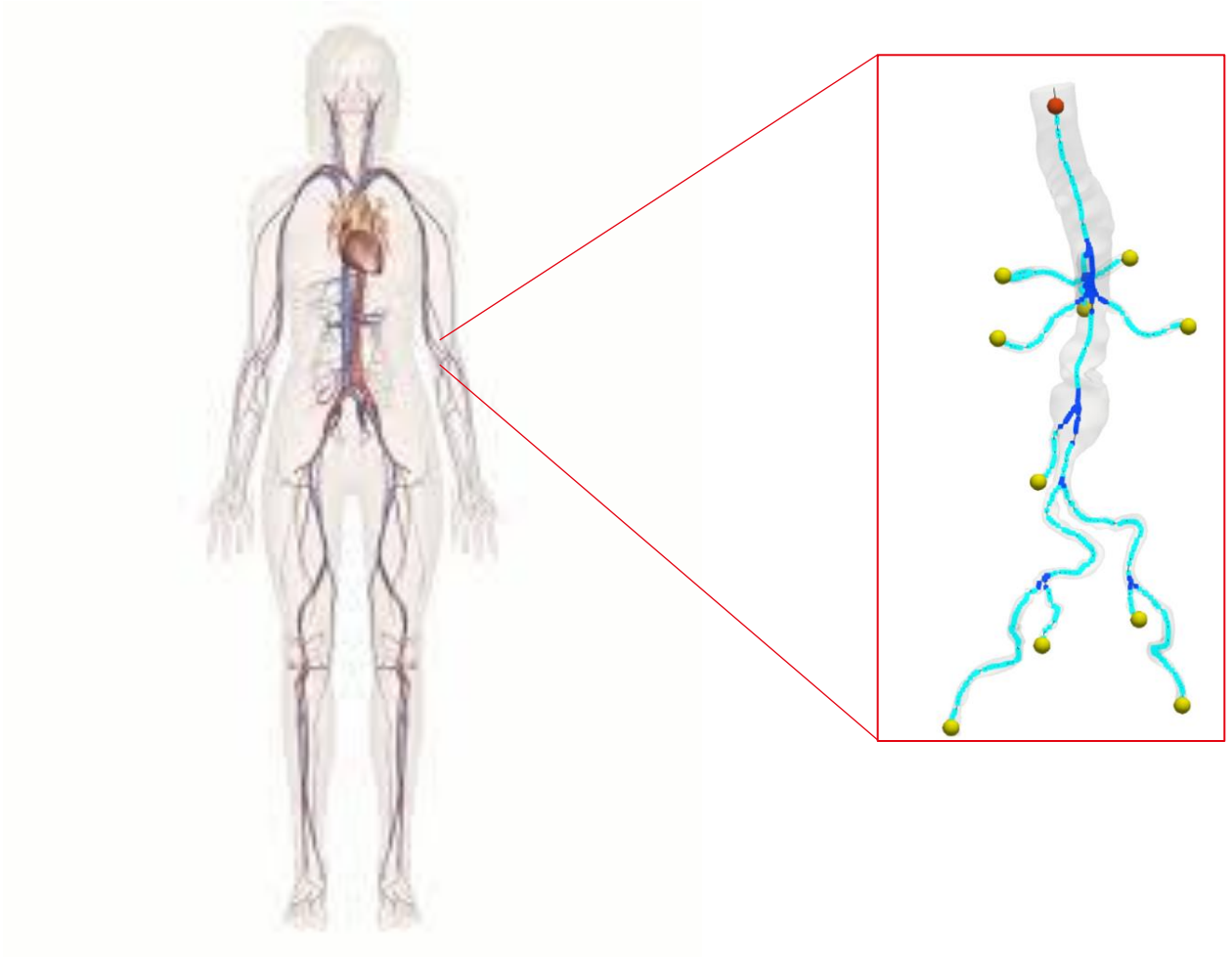
- Graph Attention Network
  - Position edge features
  - Integrated gradients
- Application
  - Risk prediction
  - Context features
- Improvements
  - Cellular-level features
  - Attention model





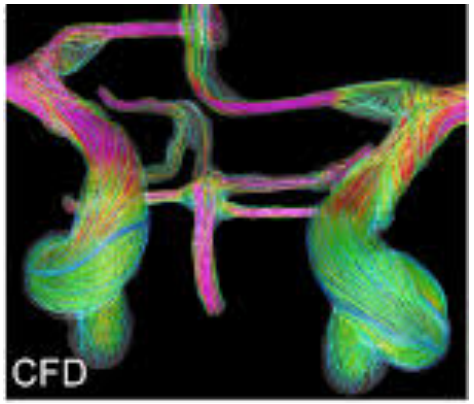


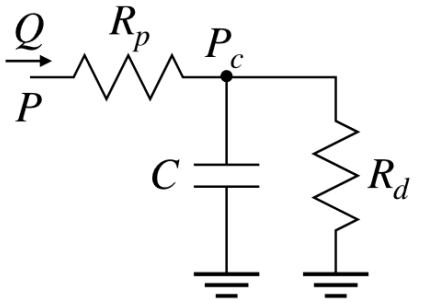
Learning Reduced-Order Models for Cardiovascular Simulations with Graph Neural Networks

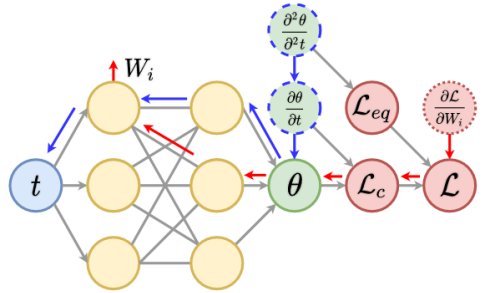


Introduction

- Computational Fluid Dynamics
  - Non-invasive method
  - High computational cost!
- Reduced order models
  - Reduced precision - static pressures!
  - Generalization - specific models!
- Physics-informed Neural Networks
  - Don’t support unseen geometries!

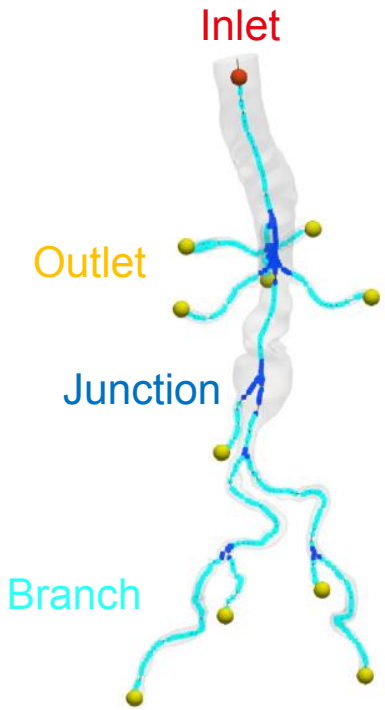


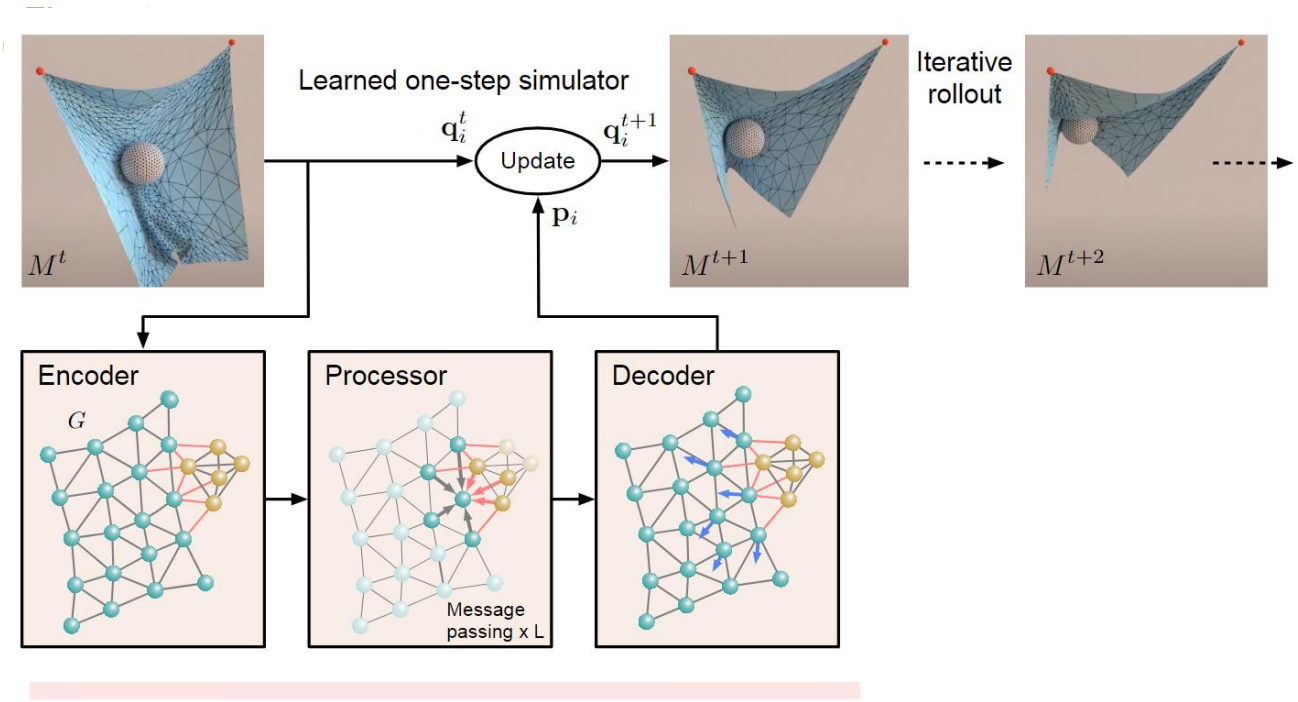




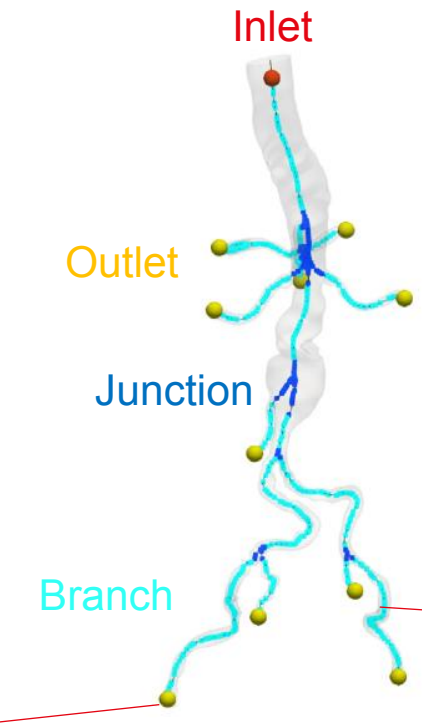
Framework

- Modified MeshGraphNet



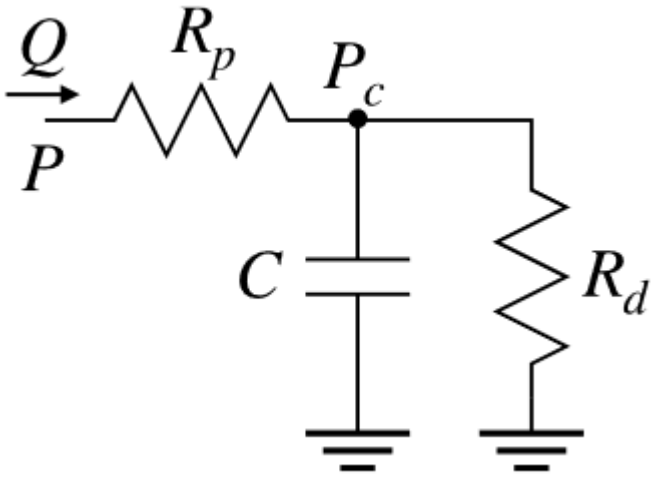


Method



Each node

$$\frac{dP_c}{dt} = \frac{1}{C}\left(Q(t) - \frac{P_c}{R_d}\right)$$

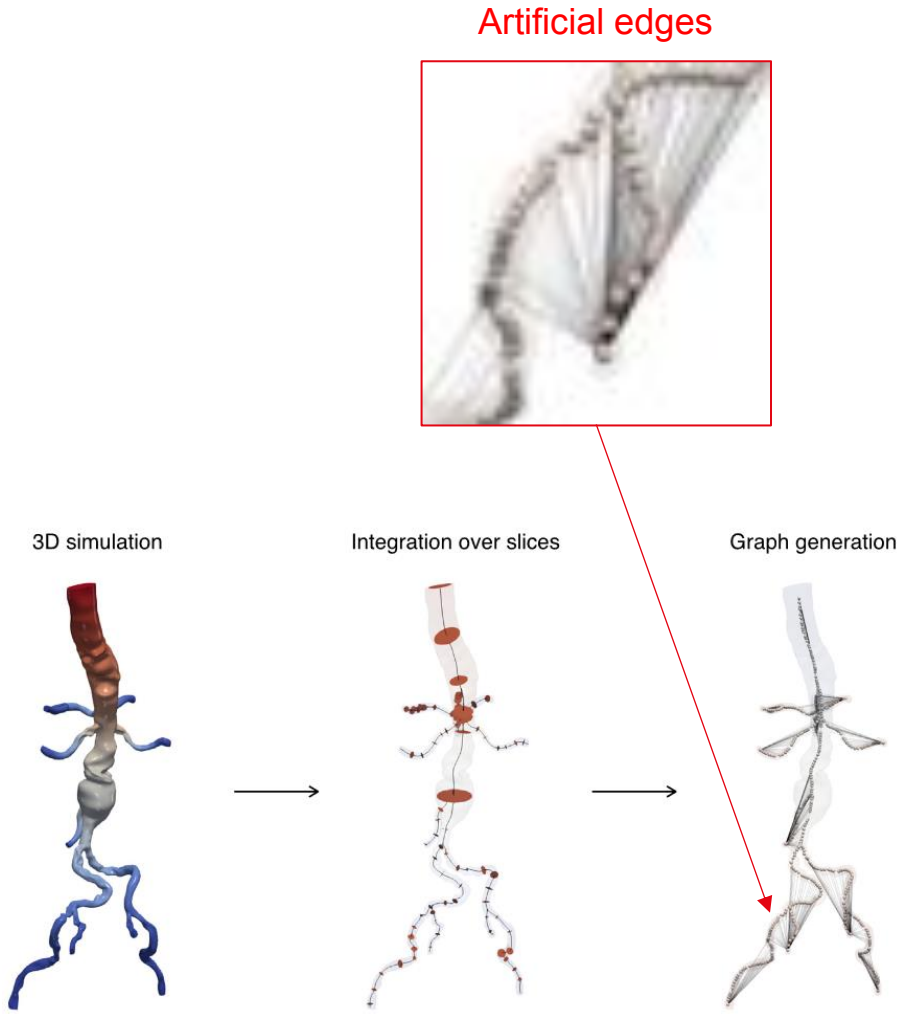


Pressure
Flow rate
Area
Type
Boundary

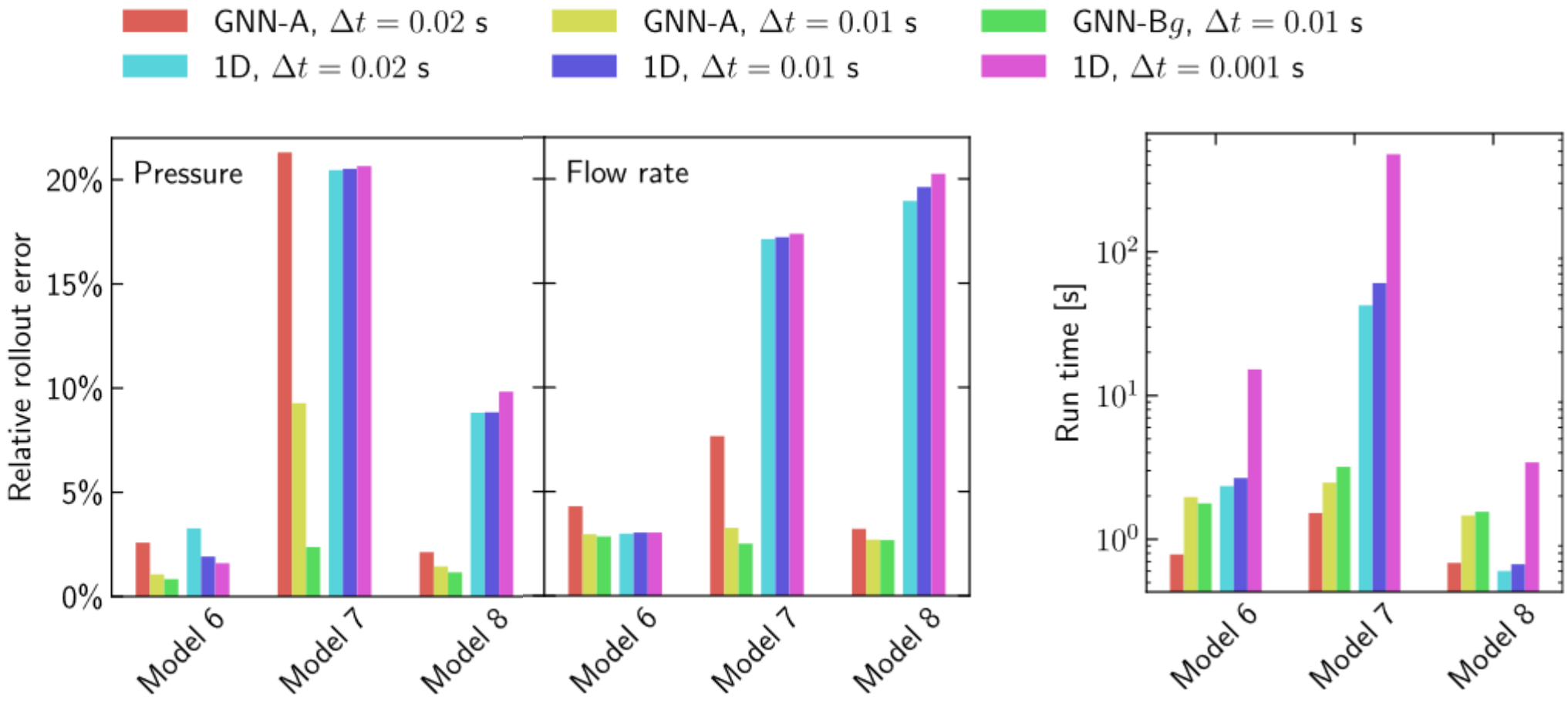
Nodes' positions
Distance
Type

Application

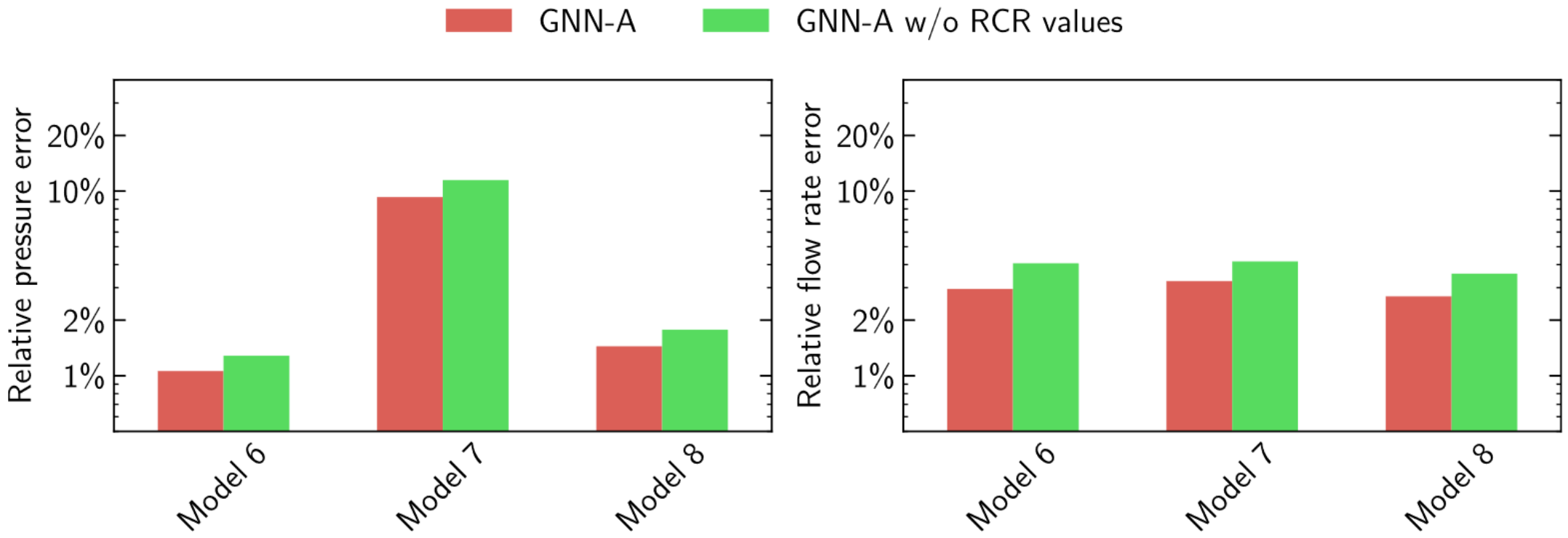
- Training specifics
  - Mean squared error on 5 step roll-out
  - Gaussian noise for robustness
  - Cross-validation
- Data
  - 5 healthy & 3 diseased patient models
  - 3D-simuluation and conversion to 1D
  - 90%/10% - train/test split



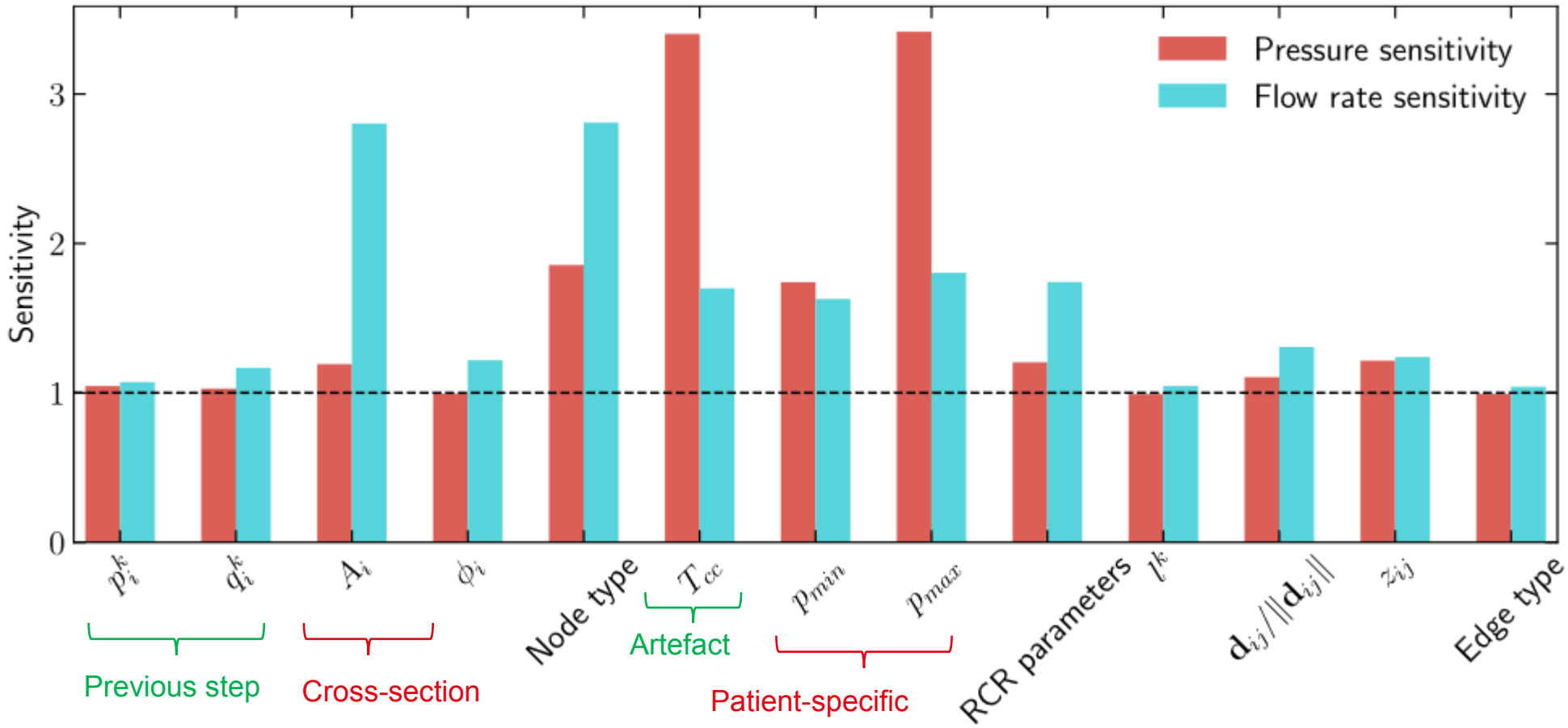
Results: Accuracy



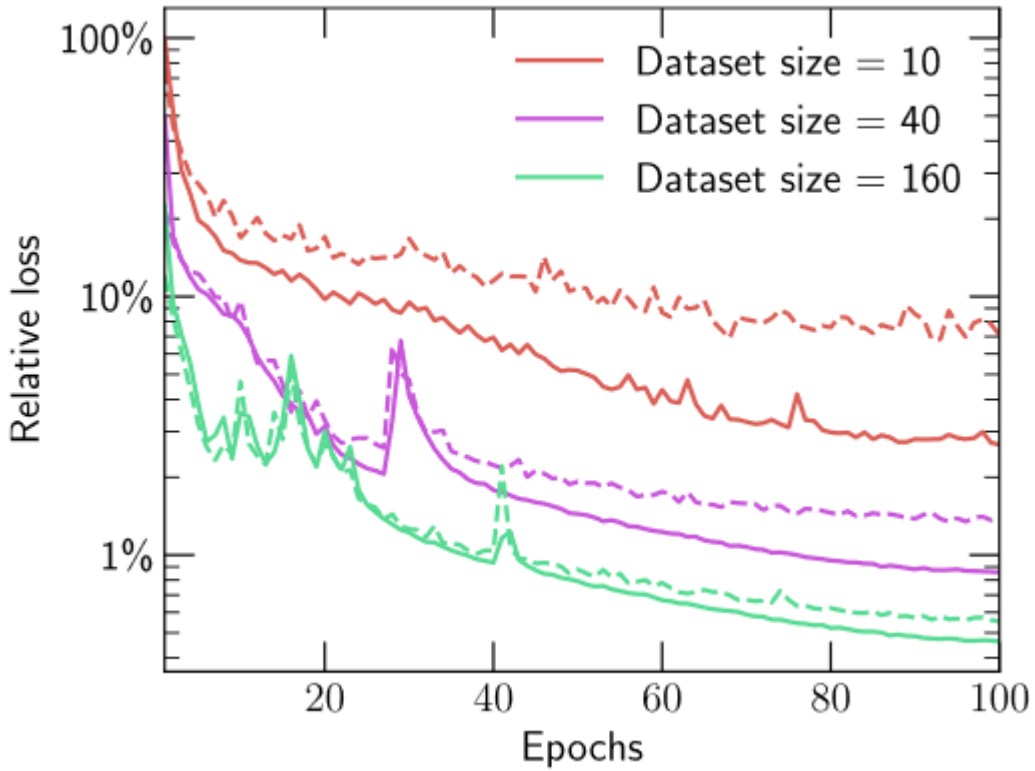
Results: RC-Ablation

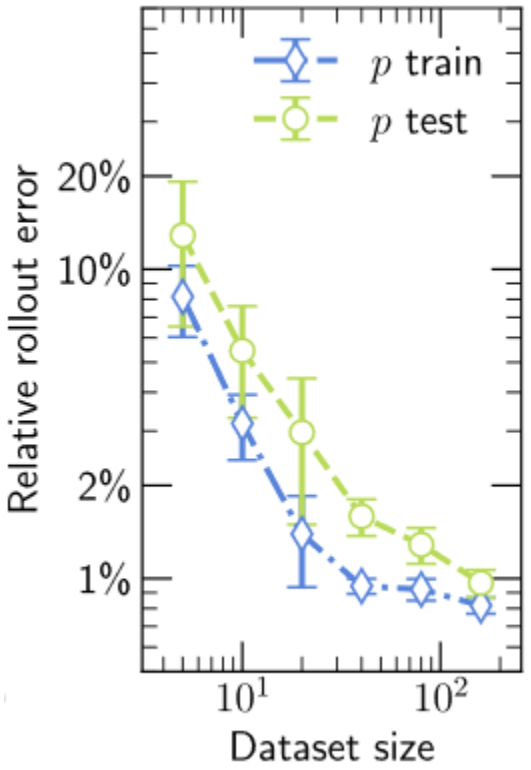


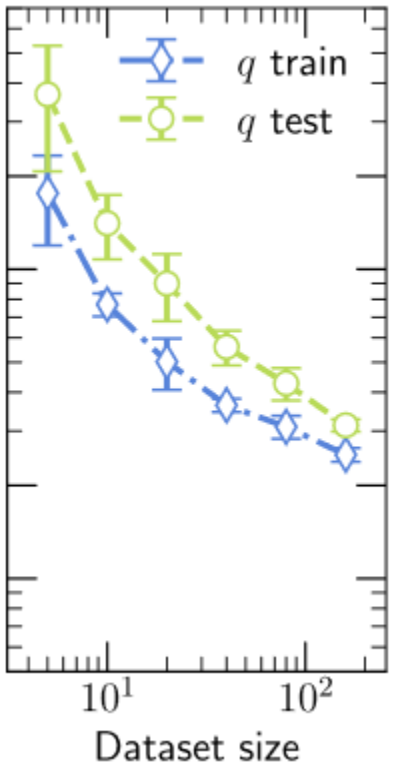
Results: Sensitivity Analysis



Results: Data Dependency

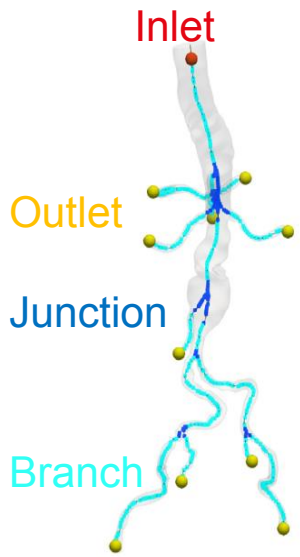


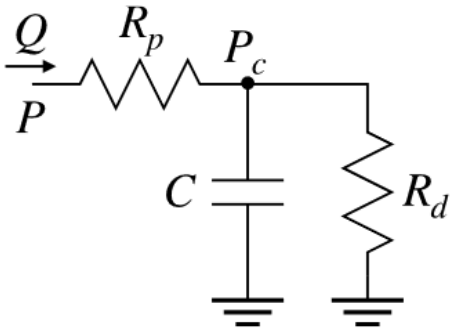




Summary

- Modified MeshGraphNet
  - Physics-informed
  - Geometry generalization
  - Sensitivity analysis
- Application
  - Blood flow simulation
- Improvements
  - Robustness for geometry generalization
  - Model size investigation





École
polytechnique
fédérale
de Lausanne