DGL-25 : Brain connectivity super-resolution

Denis Fouchard
Hippolyte Verninas
Hugo Pham
Sami Ribardière
Yassine Benbihi

I. Methodology & Novelty (40 points)

A. Problem description & motivation (5 points)

This project focuses on using generative Graph Neural Networks (GNNs) to predict high-resolution (HR) brain connectivity graphs from their low-resolution (LR) counterparts. This task is motivated by the need to enhance the resolution of brain connectivity data, which is crucial for advancing our understanding of brain function. The problem involves learning a mapping function f that can accurately transform a low-resolution brain matrix A^{LR} into a high-resolution brain matrix A^{HR}. This task is essential because high-resolution brain graphs provide more detailed and precise information about neural connections, which can significantly improve the analysis and diagnosis of neurological conditions. By developing a model that can infer high-resolution connectivity from low-resolution data, we aim to overcome the limitations of current imaging techniques and pave the way for more accurate and insightful brain research.

B. State-of-the art methods (3 points)

See Table I

TABLE I
Overview of GNN-based SOTA Models

Model Name	Brief Description
GraphUNet [1]	Inspired by usual U-Net networks, GCN with an encoder-decoder structure and pooling, enabling hierarchical feature extraction and improved node classification.
GSR-Net [2]	Built on top of the GraphUNet architecture. Designed to work on fully connected brain graphs (connectomes). Adds a Graph Super Resolution (GSR) Layer inspired by imaging methods to upscale graphs.
AGSR-Net [3]	Built on top of the GSR-Net architecture. This adds a adversarial regularization to ensure the upscaled graphs match a prior ground-truth HR distribution.
GraphTransformer [4]	Incorporates self-attention and positional encoding into GNNs, enabling global message passing for improved representation learning in graph-based tasks.
GIN [5]	Maximizes the expressive power of GNNs by using injective multi-layer perceptrons (MLPs) as aggregation functions

C. Main figure (4 points)

See Figure 1

D. Brief overview of the proposed GNN (5 points)

Node features initialization We initialize the node feature matrix of size $n \times d$ with a mix of various topological features (betweeneness centrality, closeness, clustering), and adjacency Laplacian eigenvectors (first $d-3$ components).
UNet Architecture Our model is built over the Graph-UNet [1] architecture, with an up-sampling layer on top. We replaced the standard graph convolution layers by graph attention layers [4], [6], because they provide adaptive feature weighting which improves representation learning for heterogeneous connectivity patterns in those graphs. It starts by learning node embeddings through successive down-convolutions and pooling. In each pooling layer, the number of nodes is scaled down by a hyperparameter factor $0 < k < 1$, while the node feature space dimension increases by an amount $\frac{1}{k}$. This allows for a transfer of representation between the adjacency matrix and the node features. After these down convolutions, just like in regular UNet networks, the learned embedding is then passed into a bottleneck convolution. Then, we successively reconstruct our input through a series of up-convolutions and unpooling, to match the dimensions of outputs with those of the down-pooling layers. Finally, we pass our reconstructed inputs through a "Graph Upsampler". This upsampler is composed of linear layer upsampling our nodes embeddings from 160 to 268. It then reconstructs the associated adjacency matrix and optionally refines it through two convolution layers. Transformer Convolution layers are used to capture long-range dependencies in brain connectivity, which traditional convolution-based models struggle with. Moreover, we provide for long-range feature consistency by reconstructing the adjacency matrix from the feature matrix $\tilde{A} = \sigma(XX^T)$ at each up-pooling level, and computing the loss with the corresponding down-pooled ground truth adjacency (see Figure).

E. Innovative components (10 points)

Our model introduces two key innovations compared to existing SOTA for brain graph super-resolution.

Unlike classical Graph UNet models that only compute loss at the final HR adjacency matrix, we introduce intermediate losses at each down-sampling stage. It helps the model preserve hierarchical graph information throughout the encoding process. By reinforcing meaningful representations at lower resolutions, the model achieves smoother feature transitions and better reconstruction fidelity in the final HR output.

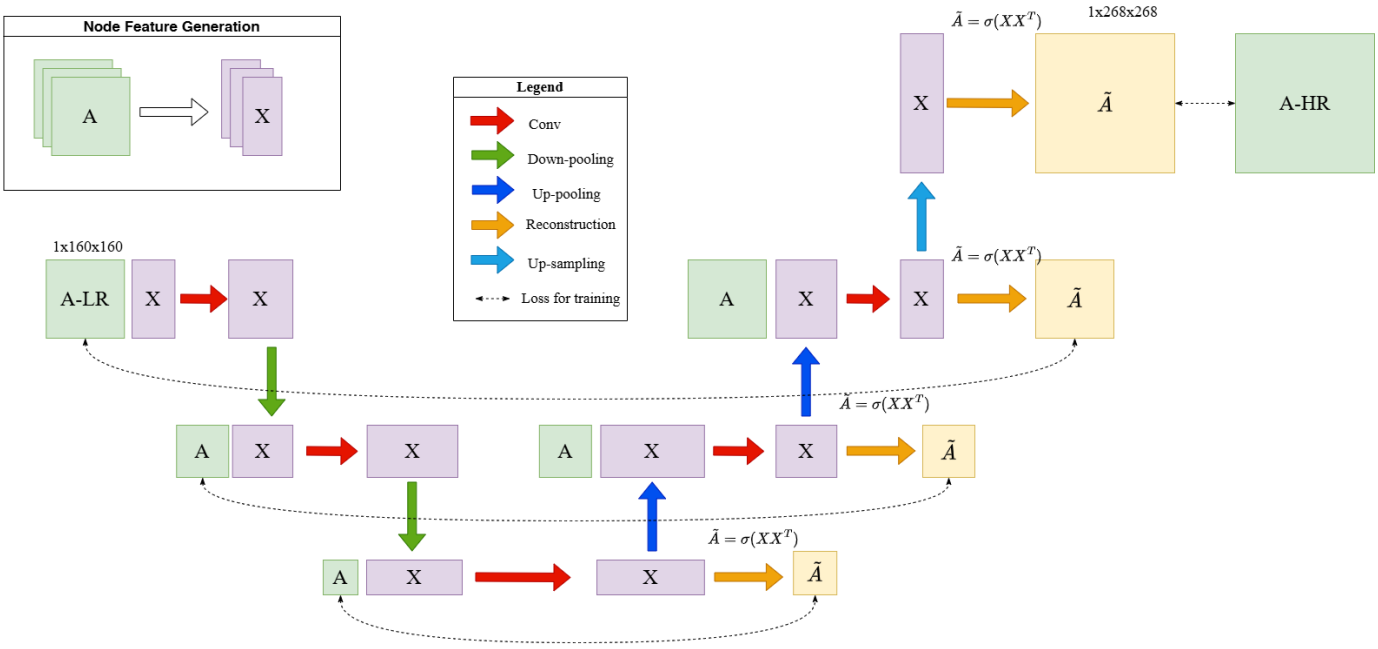

Fig. 1. Graph U-Net like auto-encoding architecture

Instead of only using raw node features or random initialization, we incorporate the eigenvectors of the graph Laplacian as initial node embeddings, (along with various topological features such as betweenness centrality, closeness, and clustering). Since the brain graph we are working with exhibits modular structures with well-defined clusters (and clusters within clusters), Laplacian eigenvectors naturally encode these hierarchical and spectral properties. This initialization helps preserve long-range dependencies, which is essential for reconstructing HR connectivity patterns from the LR matrix. Therefore, we use global graph structure from the start, enabling the model to better capture connectivity patterns.

See Table II.

TABLE II
INNOVATIVE COMPONENTS OF THE PROPOSED GNN FRAMEWORK

Novel Contribution	**Rationale**
Intermediate Losses at Each Down-Pooling Level	Instead of only computing loss at the final output, we introduce intermediate losses at each down-sampling stage. This enables the lower-resolution embeddings to preserve meaningful hierarchical information, which can lead to better reconstruction of the HR adjacency matrix.
Use of Laplacian Eigenvectors for Node Feature Initialization	We initialize node features using the eigenvectors of the Laplacian matrix. Laplacian eigenvectors capture the global connectivity patterns of the graph, encoding spectral properties that reflect how information propagates across the network. This allows our model to encode global graph structure from the beginning, which improves connectivity preservation.

F. Mathematical properties of the proposed GNN (13 points)

Permutation invariance (5 points)

a) (1 point) A function f is said to be permutation invariant if for any permutation h, $f \circ h = f$

b) (4 points) Our GNN architecture takes as input a graph of 160 nodes and outputs a new graph of 268 nodes. Permutation invariance is not satisfied since the GNN gives a new embedding for the nodes.

Permutation equivariance (5 points)

a) (1 point) A function f is said to be permutation equivariant if for any permutation h, $f \circ h = h \circ f$

b) (4 points) We will break down the components of our model. Firstly the convolutions, which are permutation equivariant. The reconstruction, for P a permutation matrix, $\sigma((PX)(PX)^T) = \sigma(PXX^TP^T) = P\tilde{A}P^T$, is permutation equivariant. A downpooling-uppooling layer is permutation equivariant as well, since the top ranked nodes will be selected and used to unpool. Finally, the reconstruction layer is made of a linear layer and convolutions, so it is permutation equivariant as well.

Expressiveness (3 points)

a) (1 point) Expressiveness refers to how well a GNN can distinguish different graphs or different parts of a graph based on their structure and node features.

b) (2 points) To maximize expressiveness, we used transformer convolutions, and a U-net like architecture to capture key features of the graph.

II. Experimental Setup & Evaluation (27 points)

A. Results (9 points)

a) (2 points) See Table III. The KL divergence is especially effective for the graphs we have, which are highly connected, with various weights. Unlike the other topological metrics, the core-periphery structure analyses the hierarchical structure of the graph.

b) (4 points) See Figure 2. To assess the generalization of our model, we performed 3-fold cross-validation on the training set of 167 samples, evaluating its performance across 8 different metrics.

The Mean Absolute Error remains stable across folds, indicating consistent reconstruction performance. The Pearson Correlation Coefficient (PCC) varies slightly, but remains high, demonstrating that the predicted high-resolution graphs preserve key structural relationships. The Jensen-Shannon (JS) Distance shows slightly higher variability across folds, suggesting some sensitivity in capturing probabilistic graph differences.

This metric quantifies the similarity between the edge weight distributions of the predicted and ground-truth HR graphs. There is a high divergence, which suggests that the model fails the capture the global distribution, and maybe focuses on certain subgraphs or nodes.

Evaluating graph topology, we measure betweenness centrality, eigenvector centrality, PageRank centrality, and core-periphery structure. The model preserves centrality measures well, indicating that key nodes retain their structural roles. The core-periphery structure shows some variation, suggesting that while the model reconstructs global connectivity patterns effectively, it may struggle with highly modular brain regions.

Generalization & Performance Overall, our model generalizes well across folds, as indicated by the low variance in MAE, PCC, and topological metrics. The primary source of variability lies in JS Distance and core-periphery structure, which could be due to the heterogeneity in brain graphs across subjects. Future improvements, such as incorporating anatomical priors, could enhance robustness in these aspects.

c) (2 points) The 3-fold validation took around 20 minutes on a Tesla V100, using around 500mb of RAM.

d) (1 point) After we experimented with our model and made sure the results were consistent with the 3-Fold cross validation, we used the whole `train_LR` dataset to train a model with our architecture. Then, we predicted the high-resolution images of the `test_LR` dataset before vectorizing the output to submit them to Kaggle where we achieved a score of approximately 0.1625 (ranked 24th).

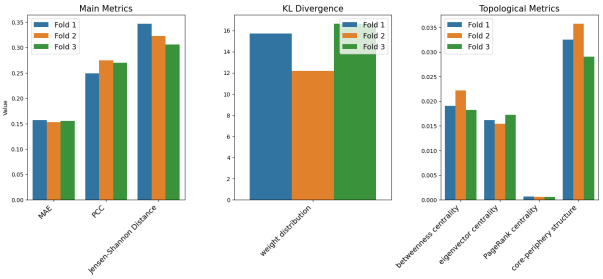

Fig. 2. Our model 3-Fold Validation Results

TABLE III
Additional Topological/Geometric Measures

Measure Name	Brief Description & Rationale
KL divergence of weight distribution	Measures the difference between the edge weight distributions of predicted HR graphs and ground truth HR graphs. This captures whether the model correctly preserves the statistical properties of brain connectivity.
Core-periphery structure	Evaluates how well the model maintains the hierarchical modularity of brain graphs, where core nodes have high connectivity and peripheral nodes have lower connectivity.

B. Comparison Against Other Methods (6 points)

On Figure 3 and Figure 4, we plotted the results of the 3-fold metrics on the validation set. Overall, the GSR performs better than the MLP. Our model accuracy is also very close to the GSR, however the KL-divergence of the weights distribution is much higher for our model. Our model is also more unstable, with some topological measures having much more variance. The KL-divergence of our model is quite similar to the KL-divergence of the MLP. The core-periphery structure of our model is much higher, which suggests that our model may miss some important information on certain nodes. We can suppose that our model may overfit on the training data, and may not capture the importance of some significant nodes.

C. Scalability of Your Proposed GNN Model (7 points)

Our model is designed to be scalable, using Graph UNet with Graph Attention and Transformer layers. We use hierarchical pooling/unpooling operations, which helps reduce the computational cost while keeping important structural information.

Additionnally, our model has a relatively low number of parameters compared to large-scale transformer-based architectures. Indeed, it has X parameters.

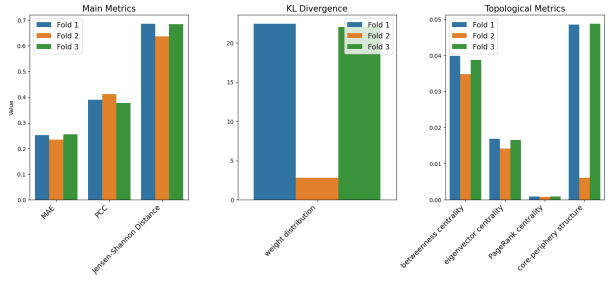

Fig. 3. MLP 3-Fold Validation Results

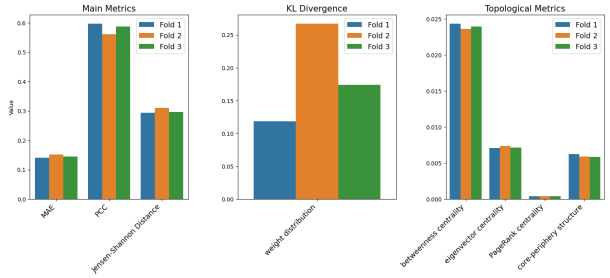

Fig. 4. GSR 3-Fold Validation Results

Concerning computational efficiency, we benchmarked training on an NVIDIA V100 GPU, where each epoch takes approximately 3 seconds. This indicates that even when scaling to larger brain graph datasets, our model is still computationally feasible. Additionally, we used mini-batching and efficient sparse matrix operations to make it adaptable for larger graphs.

Consequently, our model achieves a good balance between expressiveness and efficiency, which makes it adapted for larger datasets.

D. Reproducibility of Your Proposed GNN Model (5 points)

Our model ensures reproducibility by fixing random seeds across PyTorch, NumPy, and Python's random library, and by enforcing deterministic operations in PyTorch. We employed 3-fold cross-validation to reduce variability across runs.

While minor fluctuations may happen with graph distribution shifts, our architectural choices help mitigate instability. We used graph Transformers to enhance robust message passing, while adjacency eigenvector initialization provides a strong structural prior. Additionally, intermediate losses at each pooling level guide training and prevent overfitting.

Our complete implementation, with dataset preprocessing and training scripts, is publicly available for full reproducibility: https://github.com/denisfouchard/DGL-Group-Project.

III. Discussion & Reflections (8 points)

a) (4 points)

Our model has several strengths. First, it is a low-parameter model, making it computationally efficient while still achieving strong performance. The use of Graph U-Net and Graph Transformer layers allows effective multi-scale representation learning without too much parameter growth, which enables fast training. Another strength is our model's simplicity and transferability. It only uses standard GNN operations such as pooling and attention layers, which makes it adaptable to other graph-based tasks.

However, our model has weaknesses. One limitation is the lack of domain-specific adaptation. While our model learns from the data, it does not use fMRI-specific priors like anatomical parcellations, which could improve performance for brain graph super-resolution. Additionally, our pooling mechanism is brain-structure agnostic: it uses learned scores rather than leveraging the hierarchical organization of brain networks. This can limit its ability to capture meaningful clusters that are essential for understanding the brain.

TABLE IV
Strengths & Weaknesses of the Proposed GNN

Low parameter model	Our architecture uses a low number of layers and parameters, which enables fast training and reduces overfitting.
Simple, transferable model	By using standard GNN components such as Graph Transformers and pooling layers, the model can be easily adapted to various types of graph data beyond brain graphs.
Lacks domain-specific adaptation (fMRI connectivity)	The model does not incorporate neuroscientific priors such as anatomical parcellations that could improve performance specifically on fMRI-derived brain connectivity data.
Brain-structure agnostic (clustering)	Our pooling strategy is only based on learned scores and does not use known hierarchical structures of the brain, which may limit its ability to capture meaningful clusters.

b) (4 points)

One limitation of our current model is its lack of domain-specific adaptation. Future work could incorporate anatomical priors from fMRI studies, such as modular parcellations or hierarchical brain regions, to guide the model when learning. For instance, a biologically inspired loss function could penalize unrealistic connections that do not align with known brain structures. This would help the model generate more interpretable high-resolution graphs.

To improve generalization and reduce overfitting, contrastive learning could be introduced during training. By encouraging the model to learn invariant representations across different subjects while distinguishing between distinct brain networks, we could enhance its ability to generalize to unseen test samples. This could be achieved through self-supervised contrastive objectives, where positive samples are augmented versions of the same subject's graph, while negatives are graphs from different individuals.

References

[1] H. Gao and S. Ji, "Graph u-nets," in *International Conference on Machine Learning*, 2019, pp. 2083–2092.

[2] M. Isallari and I. Rekik, "Graph super-resolution network for predicting high-resolution connectomes from low-resolution connectomes," in *International Workshop on PRedictive Intelligence In MEdicine*. Springer, 2020.
[3] ——, "Brain graph super-resolution using adversarial graph neural network with application to functional brain connectivity," 2021. [Online]. Available: https://arxiv.org/abs/2105.00425
[4] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *CoRR*, vol. abs/1911.06455, 2019. [Online]. Available: http://arxiv.org/abs/1911.06455
[5] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *CoRR*, vol. abs/1810.00826, 2018. [Online]. Available: http://arxiv.org/abs/1810.00826
[6] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.