DGL-24 Kaggle Competition: 0-Deg.GraphLearners

Sadegh Emami CID: –
Timothy Wong CID: –
Dilay Ercelik CID: –
Moritz Hauschulz CID: –
Konstantinos Mitsides CID: –

I. Introduction

The popularity of **image super-resolution (SR)** research has grown significantly, especially with the adoption of Deep Learning (DL) techniques [1]. SR methods increase the resolution of data such as **brain connectivity matrices (BCMs)**, i.e., compact representations of the strengths and patterns of the structural connections between brain regions [2]. The large-scale acquisition of high-resolution BCMs from raw neuroimaging data is costly (time, money, compute). **Brain graph SR** could significantly speed up acquisition, via the generation of BCMs from existing ones, thus reducing the cost of research and increasing the availability of brain data for clinical applications (e.g., Alzheimer's detection [3]).

BCMs are naturally represented as graphs (nodes: regions; edges: connections), making the use of **Graph Neural Networks (GNNs)** more intuitive and efficient than traditional DL approaches. Recently, many graph-based frameworks have been proposed for brain graph SR [4]–[8]. Inspired by Graph-UNet [9], we propose a framework to learn the generation of higher-resolution brain graphs inductively: **GraphNET-byNET (GNByN)**.

II. Datasets

A. Low and High Resolution Dataset

The dataset initially comprised low-resolution (LR) and high-resolution (HR) encodings of brain connectivity, represented in vectorized formats as $A_{\text{LR}} \in \mathbb{R}^{1\times 12720}$ and $A_{\text{HR}} \in \mathbb{R}^{1\times 35778}$, respectively [10]. First, we reconstructed a symmetric matrix A from its vector representation x. This reconstruction involved vertically filling the matrices in a symmetric manner, while setting the diagonal elements of the matrices to zero. In the derived adjacency matrices, each element $A_{\text{LR},i,j}$ and $A_{\text{HR},i,j}$ quantifies the strength of connectivity between two brain regions i and j. The dataset comprised 279 samples, of which 167 were allocated for the training set and 112 for the testing set.

B. Node Features Initialization

Instead of random node feature initialization, we employed a Variational Autoencoder (VAE) to initialize the node features. This method encodes each row of the adjacency matrix into a 32-dimensional embedding in latent space, which serves as initial node feature vector. To augment our data, we encoded embeddings from one-hop adjacency matrices A^1 (represented as X_0) and three-hop adjacency matrices A^3 (represented as Y_0).

C. Data Preprocessing - Upsampling

To enhance the generalisability of our model, we upsampled the data by replicating the minority class of graphs five times, where the number five was chosen as it empirically showed the most promising validation results. More specifically, we obtained each graph's 130-dimensional embedding and applied Spectral Clustering to separate the data into a minority-class cluster and a majority-class cluster. Spectral Clustering was chosen for its effectiveness in identifying clusters within data that have complex structures, like graph embeddings.

While upsampling the data slowed down the training and led to slightly higher training losses, it contributed to regularizing the model and achieving significantly lower validation losses. This approach likely prevented the model from memorizing the most frequent graph instances, hence enhancing the model's expressiveness in capturing detailed and complex graph structures.

III. Methods

A. High-level Model Architecture

Shown Figure 3, GNByN it is comprised of the UpChanger and the DownChanger, which are each made up of 7 blocks of GCN layers. Each block inputs and returns two node feature matrices and an adjacency matrix, which is responsible of changing the number of nodes that they represent. For the UpChanger, the number of nodes changes in the order of $160, 175, 191, 206, 222, 237, 253, 268$. The same values are adopted for the DownChanger but reversed.

This is similar to the architecture of Graph-UNet [9], where the dimensionality of the adjacency matrices is projected down, through the DownChanger, and back up, through the UpChanger, via incremental change in dimensionality using GCN layers. However, to project a low-dimensional object to a higher dimension, it is key to learn a projection progressively by intermediate steps. Thus, our model learns a mapping between the LR and HR brain connectivity encoding through a path of intermediate projections of the encoding over incremental change in dimensionality. Intuitively, this anchors the UpChanger and DownChanger, such that during training both of them will converge towards an invertible one-to-one mapping between the LR and HR encodings.

Naturally, to support our design philosophy, GNByN has the characteristic to be deep and stacked with multiple GCN layers. However, as noted by Rusch et al. [11], GNNs generally face the problem of *over-smoothing*, where deeper networks encounter converging node features and would fail to differentiate nodes. Hence, GNByN adopts the Graph-Coupled

Oscillator Network (GraphCON) [12] framework to combat over-smoothing. Since GraphCON requires two node feature matrices to be updated simultaneously at each block for the oscillating-behaviour, we initialised them using A^1 and A^3 mentioned in II-B.

B. Graph Convolution Block

To change the dimensionality, a linear mapping from the input dimension to the new dimension is applied to each object inside the block. Yet, to incorporate topological information of the graph, the block uses the adjacency matrix and one of the node feature matrices from its input to create a new node feature matrix through a GATv2Conv layer [13]. The attention layer has two attention heads, such that the new node feature matrix contains more information that can be used for projecting it to a different node dimension. Note that, this enriched information will later be compressed again with a generic convolution layer to resolve the dimensionality mismatch between the new node feature matrix and the ones from the input.

Then, to obtain the new adjacency matrix from the new node feature matrix and old adjacency matrix, we incorporate a forget gate and input gate, which function similarly to how they do in LSTM layers [14]. The rationale is that there should be a long-term dependency over the chain of the adjacency matrices in how they evolve over the different dimensions, and short-term influence based on the node-embeddings and graph convolution. Finally, the new node feature matrices are transformed and outputted based on the GraphCON framework, and then passed into the next block.

C. Invariance and Equivariance Properties

The SR task regards only the adjacency matrix, and node embeddings are only of auxiliary value. With respect to the final auxiliary embeddings, our network is not permutation invariant. In GNByN, if the input A_{lr} is permuted to $P^T A_{lr} P$, then this new node ordering will be preserved by our GATv2Conv layers and dimensionality transformations. Note that invariance in the sense $f_\theta(P^T A_{lr} P) = f_\theta(A_{lr})$ is undesirable, since this would map two potentially *different* input brains to the same output brain. Similarly, note that equivariance in the sense $f_\theta(P^T A_{lr} P) = P^T f_\theta(A_{lr}) P$ is ill-defined, since the dimensions of A_{lr} and $f_\theta(A_{lr})$ differ.

D. Training Strategy

When given a batch of data during training, it is utilised over three stages of the procedure in a sequential manner. Firstly, the LR and HR data are passed into the UpChanger and the DownChanger, respectively, to obtain the two sequences of adjacency matrices over all the steps. Between the two sequences, the adjacency matrices with the same number of nodes are compared to compute the average MSE loss. While keeping the UpChanger frozen, this loss is backpropagated to update the parameters of the DownChanger.

The process repeats in the next stage with the same data handling for matrix sequence generation. Although MSE loss is calculated step-wise over the two sequences, it is a weighted average, where, at epoch t, the final HR projection from the UpChanger has a weight of α_t and the rest of the steps have a equal weight of $1 - \alpha_t$. We defined α_t to be $\alpha_t = (2 - \exp(-t/5))/2$, such that the loss focuses more on the correctness of the final HR projection as training progresses. Based on this loss, the optimiser updates the parameters of the UpChanger while keeping the DownChanger frozen.

Lastly, the LR projection of the HR ground truth is returned by the DownChanger, and this projection is then passed into the UpChanger to obtain the corresponding HR projection. A reconstruction loss is calculated based on the HR projection and the ground truth, and is used to update the parameters in both the UpChanger and DownChanger.

The goal of this training strategy is to: 1) rely on the DownChanger to ensure that the projections of the adjacency matrices at each step converges, 2) generate the correct HR projection using the UpChanger while still keeping its intermediate projections anchored at each step, 3) strengthen the model's generalisability against unseen data when its input data is injected with some noise from the reconstruction.

IV. Results and discussion

To determine the best model, we conducted a 3-fold cross-validation on the training data. Figure 1 presents the final results of this validation process. The results demonstrate a high PCC, signifying a strong positive linear correlation. Also, the JSD is observed to be low, indicating a high similarity between the probability distributions. The MAE has an approximate value of 0.13, which underscores the precision of our predictions. Multiple topological metrics were also explored, including PageRank centrality, betweenness centrality, and eigenvalue centrality. Displayed in the same figure, the corresponding MAE values for these metrics are very low, which attests to the accurate reproduction of the graph's topological characteristics in our model.

During training, the A30 GPU 24GB was used. The average memory usage recorded is 7.7 GB, which falls well within the processing capabilities of the NVIDIA A30 GPU. The memory usage peaked at 11 GB, which likely occurred during a forward pass of the neural network. This peak is considered satisfactory as it does not deviate hugely from the average usage, indicating a balanced memory demand. The training duration was approximately 83 minutes, consistent with the expected time frame for a model of this complexity.

After 3-fold cross-validation, we selected the model with the best results for further training. Utilising the LR training dataset, we allocated 80% for model training and reserved 20% for validation (Figure 2). Then, we employed the LR test data to evaluate our finalized model. Our final outputs are then generated by vectorising all the upper triangular elements per predicted HR matrix. Lastly, on Kaggle our model attained a score of 0.1323 on the public test set (ranking 20th) and 0.1530 on the private test set (ranking 16th).

REFERENCES

[1] H. X. Q. L. W. Y. R. C. S. R. E. . Z. C. Chen, H., "Real-world single image super-resolution: A brief review," *Information Fusion*, vol. 79, pp. 124–145, 2022.

[2] R. J. D. B. A. V. H. J. D. . B. S. Y. Brown, J. A., "The ucla multimodal connectivity database: a web-based platform for brain connectivity matrix sharing and analysis," *Frontiers in Neuroinformatics*, vol. 6, no. 28, 2012.

[3] C. J. H. S. C. L. S. P. G. . Z. Y. Shan, X., "Spatial–temporal graph convolutional network for alzheimer classification based on brain functional connectivity imaging of electroencephalogram," *Human Brain Mapping*, vol. 43, no. 17, pp. 5194–5209, 2022.

[4] . R. I. Isallari, M., "Gsr-net: Graph super-resolution network for predicting high-resolution from low-resolution functional brain connectomes," *Machine Learning in Medical Imaging: 11th International Workshop, MLMI 2020*, vol. 12436, pp. 139–149, 2020.

[5] . R. I. Isallari, M., "Brain graph super-resolution using adversarial graph neural network with application to functional brain connectivity," *Medical Image Analysis*, vol. 71, no. 102084, 2021.

[6] N. A. M. M. A. . R. I. Mhiri, I., "Non-isomorphic inter-modality graph alignment and synthesis for holistic brain mapping," *International Conference on Information Processing in Medical Imaging*, pp. 203–215, 2021.

[7] K. A. B. M. M. A. . R. I. Mhiri, I., "Brain graph super-resolution for boosting neurological disorder diagnosis using unsupervised multi-topology connectional brain template learning," *Medical Image Analysis*, vol. 65, no. 101768, 2020.

[8] . R. I. Rajadhyaksha, N., "Diffusion-based graph super-resolution with application to connectomics," *International Workshop on PRedictive Intelligence In MEdicine*, pp. 96–107, 2023.

[9] H. Gao and S. Ji, "Graph u-nets," 2019.

[10] W. Liu, D. Wei, Q. Chen, W. Yang, J. Meng, G. Wu, T. Bi, Q. Zhang, X.-N. Zuo, and J. Qiu, "Longitudinal test-retest neuroimaging data from healthy young adults in southwest china," *Scientific Data*, vol. 4, no. 1, p. 170017, 2017.

[11] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023.

[12] T. K. Rusch, B. P. Chamberlain, J. Rowbottom, S. Mishra, and M. M. Bronstein, "Graph-coupled oscillator networks," 2022.

[13] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," 2022.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

V. APPENDIX

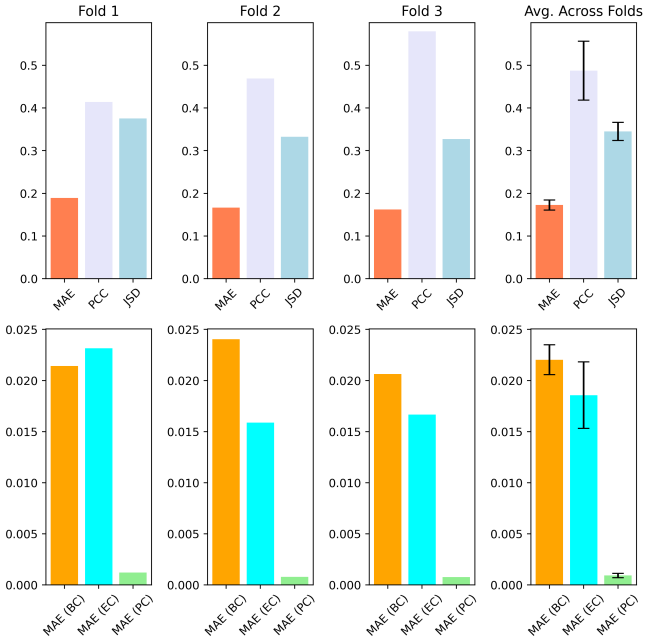

Fig. 1: Results from 3-fcv; MAE, PCC and JSD; MAE for betweenness, eigenvalue and pageRank Centrality.

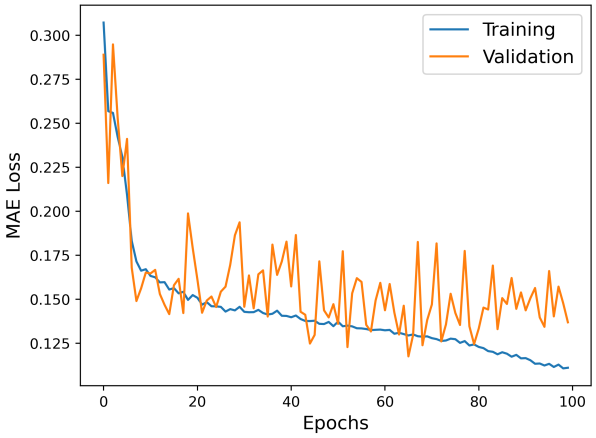

Fig. 2: Training and validation MAE for final model.

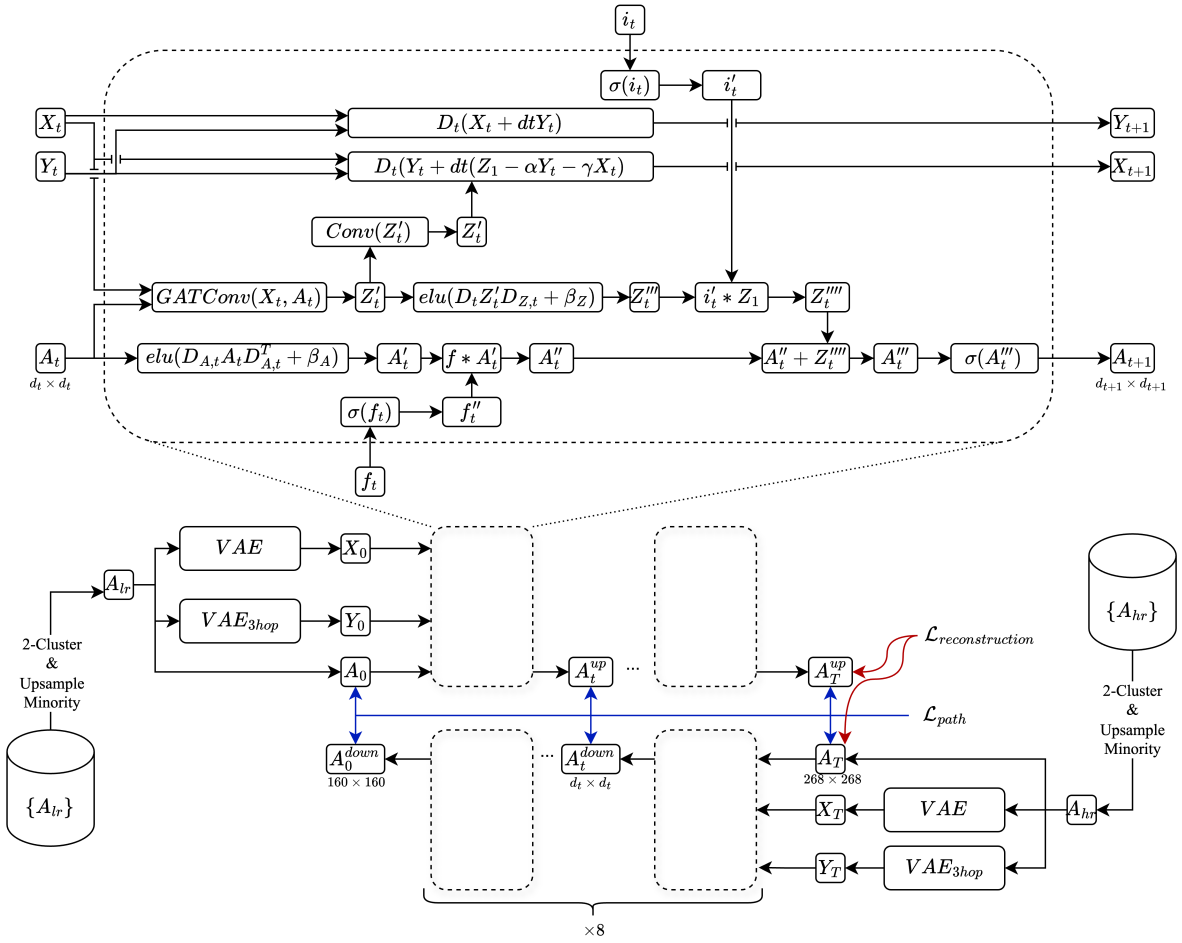

Fig. 3: High-level architecture and step-level computational graph. Batchnorm, layernorm and dropout layers are omitted. Code repository can be consulted for details. Learnable parameters (and dimensions) excluding VAEs and standard layers: $D_{A,t}$ $(d_{t+1} \times d_t)$, $D_{A,t}(d_{t+1} \times d_t)$, $D_{Z,t}(channels \times d_{t+1})$, β_A $(d_{t+1} \times 1)$, β_Z $(d_{t+1} \times 1)$, i_t $(d_{t+1} \times 1)$, f_t $(d_{t+1} \times 1)$