DGL Project [Spring 2026]: Brain Graph Super-Resolution Challenge

1 Overview

The term project is centered on brain graph super-resolution, leveraging the capabilities of **generative Graph Neural Networks (GNNs)** to predict high-resolution brain connectivity graphs from low-resolution counterparts.

1.1 Problem

Predict high-resolution (HR) brain graph from low-resolution (LR) brain graph using generative Graph Neural Network trained in an **inductive setting**.

Mathematically, the problem of brain graph super-resolution aims to learn a mapping f that maps each LR brain matrix $\mathbf{A}^{LR}$ to the HR graph matrix of the same brain, $\mathbf{A}^{HR}$:

$$f(\mathbf{A}^{LR}) = \hat{\mathbf{A}}^{HR} \approx \mathbf{A}^{HR},$$

where $\hat{\mathbf{A}}^{HR}$ denotes the predicted adjacency matrix by the GNN model f.

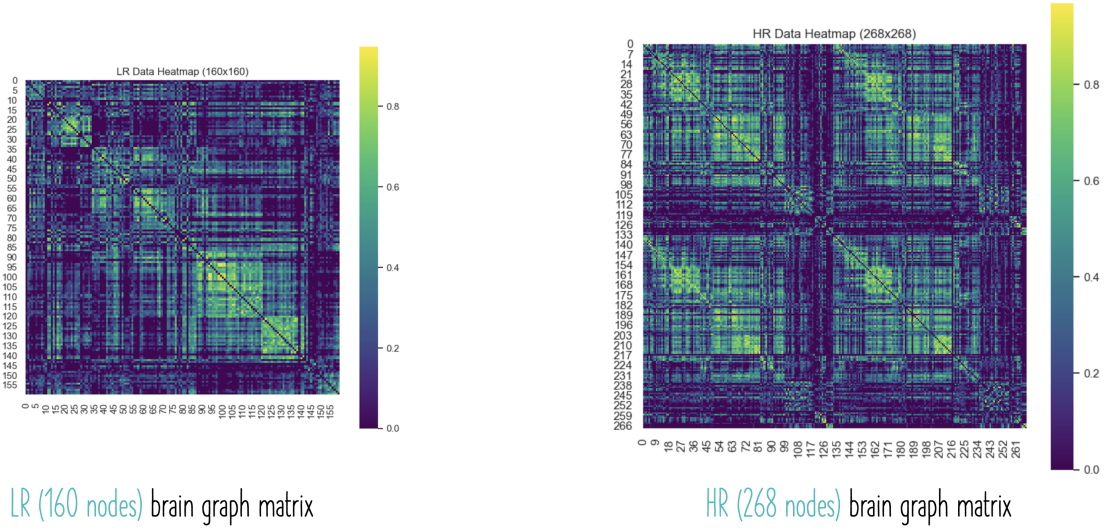


Figure 1: Heatmaps of LR matrix (left) vs. HR matrix (right) - Vertically anti-vectorized

1.2 Dataset Description

The dataset has low-resolution (LR) and high-resolution (HR) encodings of brain connectivity, represented in symmetric weighted connectivity matrices $\mathbf{A}^{LR} \in \mathbb{R}^{160\times160}$ and $\mathbf{A}^{HR} \in \mathbb{R}^{268\times268}$, respectively. Each element $\mathbf{A}^{LR}_{i,j}$ and $\mathbf{A}^{HR}_{i,j}$ quantifies the strength of connectivity (neural correlation) between two brain regions i and j.

The **goal** of this competition is to **design a generative GNN model** that can accurately infer the HR brain connectivity matrix from its LR counterpart. Specifically, the aim is to train a model that generates the high-resolution (HR) connectivity matrix $\mathbf{A}^{HR} \in \mathbb{R}^{268\times268}$, given the LR connectivity matrix $\mathbf{A}^{LR} \in \mathbb{R}^{160\times160}$ of the same brain.

1.2.1 Data Pre-processing and Download)

The complex nature of brain connectivity data demands pre-processing techniques to make it ready for advanced analyses. A pivotal step in this process is **vectorization**, where we transform the connectivity matrices into one-dimensional arrays for each resolution type. This involves specifically targeting the off-diagonal upper triangular part of the matrices $\mathbf{A}^{LR}$ and $\mathbf{A}^{HR}$, from which we extract feature vectors $\mathbf{x}^{LR} \in \mathbb{R}^{1\times12720}$ and $\mathbf{x}^{HR} \in \mathbb{R}^{1\times35778}$. These vectors embody the connectivity features at low and high resolutions, respectively, for a single sample. By methodically stacking these vectors for $N = 279$ subjects, we construct the detailed LR data matrix $\mathbf{D}^{LR} \in \mathbb{R}^{N\times12720}$ and HR data matrix $\mathbf{D}^{HR} \in \mathbb{R}^{N\times35778}$.

Conversely, **anti-vectorization** plays an equally critical role by enabling the reversal of this process, converting the one-dimensional arrays back into their original matrix configurations. This reversal is essential for specific types of analyses and visualizations that require the data in its original matrix form. To facilitate this conversion between data formats, vectorization and anti-vectorization methods are provided at the following GitHub repository: https://github.com/basiralab/DGL/blob/main/Project/MatrixVectorizer.py.

The `MatrixVectorizer` class encapsulates functionality for transforming between matrices and their vector representations through **vertical (column-based) vectorization**, catering specifically to symmetric matrices. This class provides two key operations:

- **Vectorization**: Converts a symmetric matrix $\mathbf{A}$ into a vector $\mathbf{x}$ by extracting elements in a vertical sequence, column by column, while optionally excluding or including the diagonal elements.

- **Anti-Vectorization**: Reconstructs a symmetric matrix **A** from its vector representation **x**, filling the matrix vertically and symmetrically, with an option to exclude or include the diagonal elements.

For instance, given a symmetric matrix **A**:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 2 & 4 \\ 1 & 0 & 3 & 5 \\ 2 & 3 & 0 & 6 \\ 4 & 5 & 6 & 0 \end{bmatrix}$$

Applying **vertical vectorization** with `MatrixVectorizer.vectorize(A)`, we extract the upper triangular elements (excluding the diagonal) in a column-wise manner to form a vector **x**:

$$\mathbf{x} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

Conversely, applying **anti-vectorization** with `MatrixVectorizer.anti_vectorize(x, 4)`, where **x** is the vector and the matrix size is 4, we reconstruct the original matrix **A** from **x**, maintaining the vertical symmetry and excluding the diagonal in the reconstruction process.

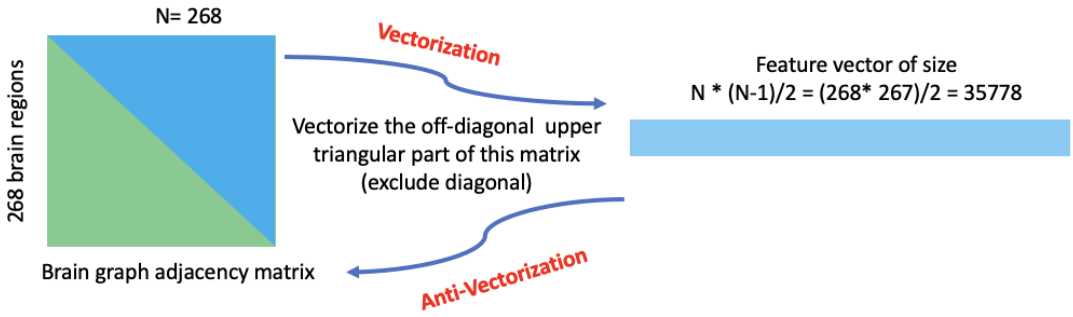


Figure 2: Illustration of the **vertical** vectorization (column by column of the off-diagonal upper-triangular part) and anti-vectorization processes.

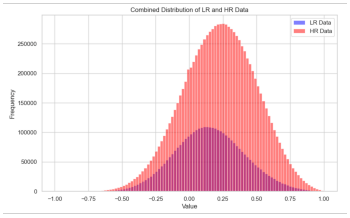
Participants are presented with the option to engage in only **inductive** GNN training paradigms. As a side note, if you intend to train your model on the whole brain graph matrix (not the vectorized version), you need to first anti-vectorize the provided data back into independent adjacency matrices at the outset. Submissions must include data in its vectorized form.

Another crucial step in the data pre-processing involves cleansing the data to ensure its suitability for analysis. This includes two main operations:

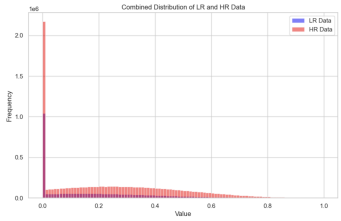
1. Replacing all negative values with 0.

2. Replacing any 'NaN' values with 0.

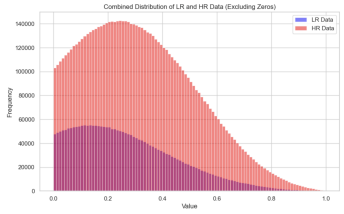
Here are the histograms illustrating the data distributions before and after these pre-processing steps:



(a) Original data distribution (no pre-processing)



(b) Pre-processed data distribution (including zero values)



(c) Pre-processed data distribution (excluding zero values)

Figure 3: Comparison of data distributions before and after pre-processing steps. The postprocessing involves (1) replacing all negative values with 0, and (2) replacing any 'NaN' values with 0. The first pre-processing histogram includes zero values, while the second excludes them for visualization.

1.2.2 The provided dataset

The dataset is available on the Kaggle competition page. You can either work on Kaggle directly or download the dataset and work in your environment (Colab, DoC Clusters, your own PC, etc). The whole dataset comprises **pre-processed** feature matrices of vectorized brain connectivity matrices, encapsulated within NumPy arrays. Both the LR and HR datasets are structured as NumPy arrays. The data shapes are as follows:

- **Low-Resolution (LR) Data:** Shaped as $(279, 12720)$, indicating 279 samples, each with 12,720 features.
- **High-Resolution (HR) Data:** Shaped as $(279, 35778)$, also with 279 samples but with 35,778 features each.

This dataset, spanning 279 examples, is split into training, public test, and private test sets, with proportions of 0.6, 0.2, and 0.2, respectively. This allocation results in:

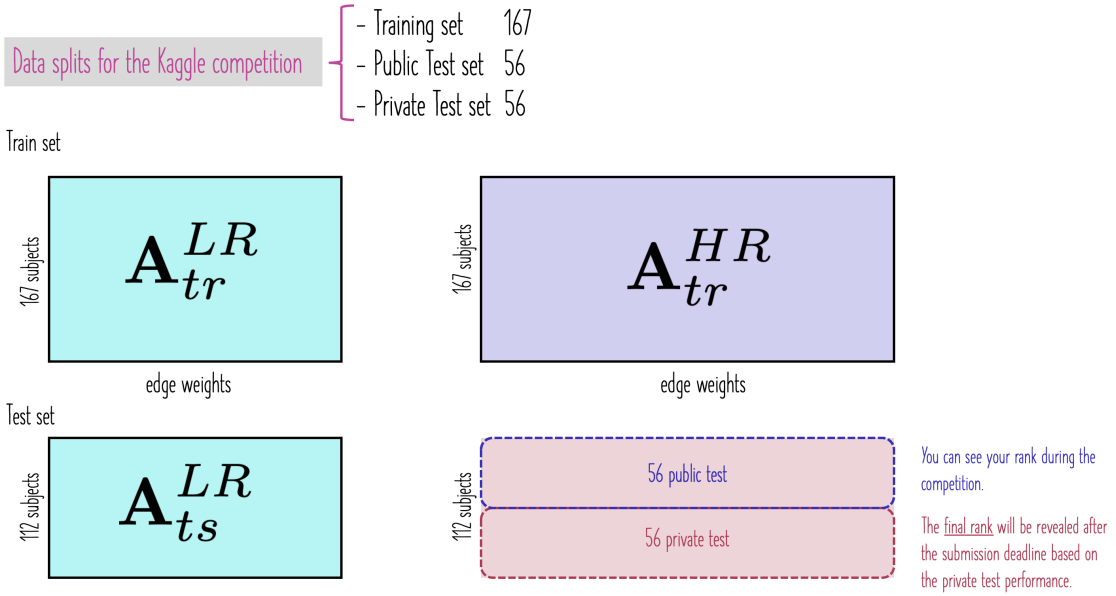


Figure 4: Kaggle competition data split. The matrices highlighted in solid bold lines will be shared with the participants. We aim to generate the dashed HR test matrix for all public and private test samples. tr: train data. ts: test data.

- **Training Set:** 167 examples
- **Public Test Set:** 56 examples
- **Private Test Set:** 56 examples

The participants will have access to (**Fig**4): **1)** the 167 training set comprising the LR and HR vectorized feature matrices, and **2)** the 112 input LR test set.

Data ethics and sharing. The competition dataset is derived from the SLIM functional MRI dataset [Liu et al., 2017]. This data is not to be used for other purposes of the competition and not to be made available/shared or posted without consulting the module leader.

1.3 Scientia Submission Policy:

- Submit your PDF report and the code in a zip file through Scientia on time.
- Unnecessary uploads (files, pictures, etc.) will be penalized!
- Only include items in your zip file that you are asked to.

1.4 Project Evaluation Criteria

Below are the details of the point distribution and the bonus points for competition rankings.

1.4.1 Point Distribution

The project evaluation encompasses three components, each weighted by points to signify its importance. Kaggle Competition Participation and Submission is awarded 5 points. The Report contributes 75 points to the evaluation. The technical execution, particularly through Code with 3-fold Cross-Validation, is the most significant component, valued at 20 points. The total score for the project is calculated based on the following components:

Component	**Points**
Kaggle Competition Participation	5
Report	75
Code (3-fold Cross-Validation)	20

Table 1: Point distribution for project evaluation

1.4.2 Bonus Points for Competition Ranking

Achievements in the Kaggle competition are rewarded with bonus points as follows:

Rank	**Bonus Points**
1st Place	5
2nd Place	4
3rd Place	3

Table 2: Bonus points based on Kaggle competition ranking

Remark: It's important to note that if multiple teams attain the same ranking, each team will be awarded the same amount of bonus points (e.g. teams finishing in **1st Place** each is awarded 5 bonus points).

2 Kaggle Competition (5 points)

2.1 Description

We have created a private class competition on Kaggle for the term project. To access the competition, please follow this link:
https://www.kaggle.com/t/14b045d8cc3646ed966894712d20b762.

2.2 Competition Rules

- Every student has to create a Kaggle account.
- Form a team of 3 to 4 students (The "team" tab on the competition).
- Individual submissions are not allowed. In such a case, send us an email so that we assign you to a group.
- Submission format is explained on the competition webpage.
- You are allowed to use only Python programming language for the implementation.
- You are free to use machine learning, deep learning, and graph processing libraries such as scikit-learn, PyTorch, Tensorflow, PyTorch Geometric (PyG), etc.
- The direct usage of a pre-trained deep learning model (such as fetching and inferring a model from Hugging Face) is not allowed. You need to develop, code, train, and test **your own model**.
- Academic dishonesty including cheating, plagiarism, and direct copying is unacceptable. Note that your codes and reports will be checked using plagiarism tools!

2.3 Evaluation Details

2.3.1 Evaluation Datasets

The competition will utilize two sets for evaluation (each with 56 examples):

- **Public Test Set**: Released at the start of the competition for development and validation. Scores on this set will be visible on the public leaderboard.

- **Private Test Set**: Used for final evaluation after the competition ends. The results on this set will determine the winners and will not be revealed until the competition concludes.

The evaluation strategy is designed to ensure a comprehensive assessment of the models' generalization capabilities through a split between public and private test sets.

- **During the competition:**
  - Kaggle will utilize the public test data to compute and rank participant submissions.
  - These rankings will be displayed on the public leaderboard, accessible to all competitors.
- **Upon the completion of the competition:**
  - The final evaluation of model performance will be conducted using the private test data set.
  - **Important:** It is the private test set scores *only* that determine the final ranking once the competition concludes.
  - The rankings on the private leaderboard, based on this assessment, will remain confidential until the competition is officially concluded.

2.3.2 Scoring and Metrics

The primary metric for evaluation will be the accuracy of the predicted HR matrices against the ground truth, measured through appropriate error metrics such as Mean Absolute Error (MAE).

2.4 Submission Process

To evaluate the performance of your model on the test data, you are required to submit your predictions to Kaggle in the specified format.

Due to Kaggle's submission constraints, your predicted high-resolution (HR) output, inherently an array sized 112×35778, must be vectorized –a process often referred to as "serialization" or "melting".

2.4.1 Creating a submission file:

1. **Step I: Serialization:**

   - **Objective:** Convert the 112×35778 prediction matrix into a 1D array.
   - **Method:** Use the `numpy.flatten()` method for an efficient transformation from a multi-dimensional array into a one-dimensional array.

     ```
     meltedDF = df.to_numpy().flatten()
     ```

   - **Alternative:** For those using pandas, the `pandas.melt()` function is available but not recommended due to the simplicity of the NumPy method.

2. **Step II: Create a CSV file:**

   - **Header Row:** Ensure a header row is present with the titles "ID" and "Predicted" for correct file processing.
   - **File Structure:** Assemble the submission into a CSV file with two columns: "ID" and "Predicted", and a total of 4,007,136 entries.
   - **ID Column:** Create an "ID" column, numbering from 1 to 4,007,136, to uniquely identify each entry in the flattened array.
   - **Predicted Column:** Include your model's real-valued predictions corresponding to each ID, ensuring the values follow the same sequence as the flattened array.

2.4.2 Compliance

- Ensure your submission does not contain any additional columns, excess rows, non-integer values in the "ID" column, or non-real values in the "Predicted" column.
- Deviations from these instructions, including discrepancies in column naming or data formatting, will likely cause submission errors.

2.4.3 Resources

- For detailed instructions on the `pandas.melt()` function and other data manipulation tools, refer to the official pandas documentation and NumPy documentation.

Important submission note: Please ensure that your code and report **only** feature the **final** algorithm/model you have developed and decided to be your best option. While working on the project, you will experiment with various architectures, algorithms and models, but your deliverables must only showcase your final algorithm/model. Avoid including trial runs or experiments with other models in your submissions.

3 Deliverables Details

You will submit a **zip folder** containing the following files:

1. The final report PDF using the provided LaTeX template. For more details check Section 3.2.
2. Source code folder. Do not include trained model weights and dataset. Include a requirements.txt file for us to install your dependencies. Your code should run smoothly on any OS.
3. The three HR prediction CSV files named as `predictions_fold_{fold_num}.csv`.

All details are provided below.

3.1 Code (20 points)

Submit the code version of your model that runs a **3-fold cross-validation (CV)** with an anchor random seed for the reproducibility of the final results. Your 3F-CV will run on the provided 167 **training LR** and **training HR** data matrices, referred to as $\mathbf{A}_{tr}^{LR}$ and $\mathbf{A}_{tr}^{HR}$, respectively, in **Fig** 4. Your 3-fold CV code should generate the following outputs:

1. **(4 points)** The predicted high-resolution (HR) samples, which should be saved in a `predictions_fold_{fold_num}.csv` file. The format of these 3 csv files should be identical to the one used in the Kaggle competition.
2. **(1 point)** The **bar plots** of the **3-fold CV** evaluation measures below compare the predicted and ground-truth HR samples (see Figure 5). The evaluation code is provided here. If the model performance varies dramatically across evaluation measures, resulting in obscured bars for some measures, consider presenting the measures in separate plots for better visualization.

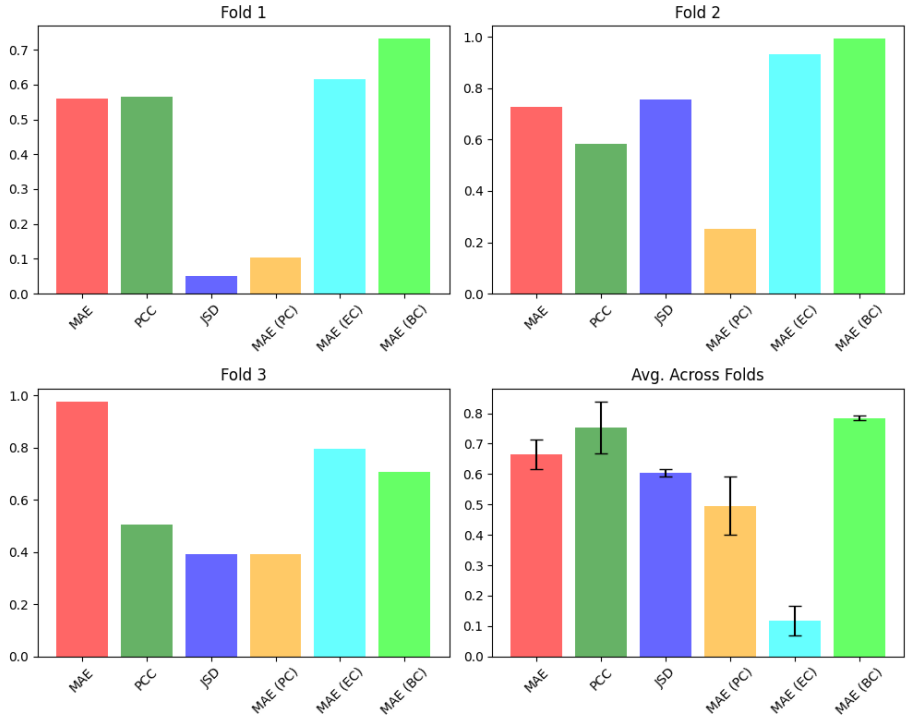


Figure 5: A randomly generated example for the bar plots for you to take as a reference. The error lines in the bottom-right plot represent the standard deviation across the folds, which is important for your report.

- MAE
- Pearson Correlation Coefficient (PCC)
- Jensen-Shannon Distance (JSD)[1]
- Average MAE of PageRank Centrality (PC)
- Average MAE of Eigenvector Centrality (EC)
- Average MAE of Betweenness Centrality (BC)

For the evaluation measures, please refer to https://github.com/basiralab/DGL/blob/main/Project/evaluation_measures.py and review the code carefully. For example, to compute MAE, we first vectorize each HR matrix and then flatten all the predictions into a 1-D array, then compute MAE.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.jensenshannon.html

3. **(15 points)** The model needs to be trained **inductively**, NOT transductively. Ensure your code is streamlined to:

   (a) Run smoothly. Omit the trained weights and dataset to keep the file size manageable. Also, include a `requirements.txt` file that lists all dependencies to ensure reproducible execution on any operating system.

   (b) Explore two additional geometric and/or topological measures. Display the updated plots in your notebook including the old and new measures (8 in total) in each subfigure (see Figure 5).

   (c) Generate bar plots and CSV files for predictions in a single run of the notebook.

   (d) Receive all necessary inputs as function parameters, including train and test data, along with model parameters.

   (e) Produce the required output, specifically the submission file for test predictions.

   (f) Include **explanatory comments** for clarity and understanding.

Note 1: Be creative and give a cool name to your model. Then, fill out the README template located at https://github.com/basiralab/DGL/blob/main/Project/README.md .

Note 2: For reproducibility, insert the configurations at the beginning of your notebook/code on the following GitHub repository: https://github.com/basiralab/DGL/blob/main/Project/reproducibility.py.

Note 3: For 3-fold CV, use the `sklearn.model_selection.KFold` function from scikit-learn. Set `shuffle` to True and `random_state` to the previously defined `random_state` variable.

Note 4: As mentioned earlier, the data has been pre-processed such that the data range is $[0, 1]$, i.e., there are no negative values. So, it would make sense for you to **post-process** the outputs of your model such that they do not include any negative values too.

3.2 Report (75 points)

We included the report instructions right after the Contact Information section. Read them carefully.

For submission, compile a report PDF utilizing the provided IEEE Conference Paper template in LaTeX at (https://www.overleaf.com/read/dhmzkhvmwskb#d179c7). Do not change the template or structure.

Be creative and give a cool name to your model such as GraphSAGE, FastGCN etc.

Computational Resources

To train and evaluate your models effectively, leverage the computational resources available through platforms like Kaggle, Google Colab, DoC cluster and even your own personal machines, if applicable.

Timeline

- Competition Launch: February 17, 2026
- Kaggle Submission Deadline: March 6, 2026
- Kaggle Winners Announcement: March 8, 2026
- Scientia zipped file submission: March 9, 2026

Contact Information

For further inquiries, participants can post their questions on EdStem or email (for urgent questions only) the following GTA as detailed below and based on their specialized domain.

- Furkan Pala, f.pala23@imperial.ac.uk

References

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*, 4(1):1–9, 2017.

DGL-26 Kaggle Competition: Your Team Name, Final Rank

Your Name 1, Student CID 1, Username 1

Your Name 2, Student CID 2, Username 2

Your Name 3, Student CID 3, Username 3

Your Name 4, Student CID 4, Username 4

This report is worth 75 points of the total mark of the DGL projet.

I. Methodology & Novelty (40 points)

A. Problem description & motivation (5 points)

Explain in your own words what the graph super-resolution problem is. Discuss its importance, potential applications, and the challenges that make it a significant research topic.

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

Identify three GNN-based state-of-the-art (SOTA) papers on brain graph super-resolution and briefly describe the backbone of each model in Table I. Remove the GraphSAGE example.

TABLE I
Overview of GNN-based SOTA Models

Model Name	Brief Description
GraphSAGE [1]	Refines GCN by employing neighborhood sampling and aggregation techniques to efficiently learn node embeddings in large graphs.
Name [?]	Describe.

Later, you will select one of these models for benchmarking using 3-fold cross-validation against your model.

C. Main figure (4 points)

Incorporate a key figure to illustrate the essential steps of your proposed solution and the learning pipeline. The training and testing processes of your model should be clearly represented in the figure. A well-designed figure should be self-explanatory, clear, and easy to follow. You may draw inspiration from other GNN papers.

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

Provide a concise overview of your GNN architecture and its key components. Walk us through your main figure and explain the reasoning behind the inclusion of each component/block.

E. Innovative components (10 points)

Specify two novel contributions you propose in comparison to existing GNN-based SOTA models. Justify the rationale behind each in Table II. Clearly articulate how your proposed GNN model differs from current SOTA methods.

TABLE II
Innovative Components of the Proposed GNN Framework

Novel Contribution	Rationale
Contribution 1	Justify the rationale behind this contribution.
Contribution 2	Justify the rationale behind this contribution.

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

This section examines the mathematical properties of your proposed framework. Below are the key properties to address, each represented in color-coded boxes for clarity. Remove the instructions and provide your answers inside the boxes. Retain items (in Latex) a) and b) for clearer structuring.

Permutation invariance (5 points)

a) Provide a brief mathematical definition of permutation invariance. (1 point)
b) Analyze whether your model, including its components and operations, satisfies this property. Provide justification. (4 points)

Permutation equivariance (5 points)

a) Provide a brief mathematical definition of permutation equivariance. (1 point)
b) Analyze whether your model, including its components and operations, satisfies this property. Provide justification. (4 points)

Expressiveness (3 points)

a) Provide a lay definition of expressiveness. (1 point)
b) Analyze whether your model, including its components and operations, satisfies this property. Provide justification. (2 points)

II. Experimental Setup & Evaluation (27 points)

A. Results (9 points)

a) Introduce two additional geometric/topological measures, as shown in Table III, to further evaluate your model. Describe what each measure captures and explain why you selected it over existing ones. (2 points)
b) Present the findings from your 3-fold cross-validation on the initial dataset of 167 samples. Include a figure displaying performance plots across all 8 evaluation measures. Discuss how well your model generalizes and performs across these measures. (4 points)
c) Specify the total training time for your 3F-CV model and report its RAM usage. (2 points)
d) For the Kaggle competition's test_LR, describe the process of retraining your model on the full train_LR dataset before testing it on test_LR to predict test_HR, following competition guidelines. Finally, report your Kaggle score and ranking. (1 point)

TABLE III
Additional Topological/Geometric Measures

Measure Name	Brief Description & Rationale
Measure 1	Describe and justify.
Measure 2	Describe and justify.

B. Comparison Against Other Methods (6 points)

Evaluate your model's performance against (1) a naive solution using Simple Graph Convolution (https://github.com/basiralab/DGL/tree/main/Tutorials/Tutorial-2)[2], and (2) a selected state-of-the-art (SOTA) method, using the same 3-fold cross-validation train/test split. Address the following:

a) Did your model outperform both comparison methods? Provide quantitative results using the 8 evaluation measures and highlight key performance improvements, if any.
b) Compare these results with your expectations. Were there any surprising findings? Discuss possible reasons behind performance differences.

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

Analyze the scalability of your model concerning larger datasets and increased computational demands. Justify your response.

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

Assess the reproducibility of your model's performance. How consistent are the results when the training and testing data distributions are varied? Provide an explanation for why your model behaves in this manner.

III. Discussion & Reflections (8 points)

a) Reflect critically on your work. What were the major strengths of your approach, and where did you identify weaknesses? List two strengths and two weaknesses. Provide concrete examples from your work to support your reflections in Table IV (4 points).

TABLE IV
Strengths & Weaknesses of the Proposed GNN

Strength 1	Provide an example.
Strength 2	Provide an example.
Weakness 1	Provide an example.
Weakness 2	Provide an example.

b) Suggest two key improvements or alternative strategies for future work (4 points). You may consider:
- Modifications to the architecture or training process.
- Ideas for addressing limitations encountered.
- Broader implications for the field of graph super-resolution.

IV. Further guidelines

- Be Specific: Vague or ambiguous answers will be penalized. Ensure your responses are clear and concise, and directly address the questions.
- Latex Response Template: Use the shared Latex Response Template to generate your report.
- Avoid Plagiarism: Any form of plagiarism including the use of LLMs to generate your answers such as GPT will result in a significant penalty. Always provide proper citations if external sources are referenced.

- Page Limit: The report should not exceed 5 pages using the provided double-column template (excluding references), although an additional page may be allowed solely for a principal figure illustrating the learning pipeline.
- References: List all references cited in your report. Ensure each citation within the report is properly attributed to its corresponding reference. You do not need to cite material from the course lecture slides or lab notebooks.
- Language and Presentation: Answers must be written in professional and grammatically correct language. Poorly presented work may be penalized.

Note: Guidelines are designed to encourage clarity, precision, and originality in your work. Non-compliance may affect your overall grade.

References

[1] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in neural information processing systems, vol. 30, 2017.

[2] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in International conference on machine learning. Pmlr, 2019, pp. 6861–6871.