Interpretable Sparsification of Brain Graphs: Better Practices and Effective Designs for Graph Neural Networks

Gaotang Li
University of Michigan, Ann Arbor
gaotang@umich.edu

Marlena Duda
Georgia State University
mduda@gsu.edu

Xiang Zhang
University of North Carolina, Charlotte
xiang.zhang@uncc.edu

Danai Koutra
University of Michigan, Ann Arbor
dkoutra@umich.edu

Yujun Yan
Dartmouth College
yujun.yan@dartmouth.edu

ABSTRACT

Brain graphs, which model the structural and functional relationships between brain regions, are crucial in neuroscientific and clinical applications involving graph classification. However, dense brain graphs pose computational challenges including high runtime and memory usage and limited interpretability. In this paper, we investigate effective designs in Graph Neural Networks (GNNs) to sparsify brain graphs by eliminating noisy edges. While prior works remove noisy edges based on explainability or task-irrelevant properties, their effectiveness in enhancing performance with sparsified graphs is not guaranteed. Moreover, existing approaches often overlook collective edge removal across multiple graphs.

To address these issues, we introduce an iterative framework to analyze different sparsification models. Our findings are as follows: (i) methods prioritizing interpretability may not be suitable for graph sparsification as they can degrade GNNs' performance in graph classification tasks; (ii) simultaneously learning edge selection with GNN training is more beneficial than post-training; (iii) a shared edge selection across graphs outperforms separate selection for each graph; and (iv) task-relevant gradient information aids in edge selection. Based on these insights, we propose a new model, Interpretable Graph Sparsification (`IGS`), which enhances graph classification performance by up to 5.1% with 55.0% fewer edges. The retained edges identified by `IGS` provide neuroscientific interpretations and are supported by well-established literature.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing** → **Bioinformatics**.

KEYWORDS

Graph Neural Networks; Interpretability; Graph Sparsification

ACM Reference Format:
Gaotang Li, Marlena Duda, Xiang Zhang, Danai Koutra, and Yujun Yan. 2023. Interpretable Sparsification of Brain Graphs: Better Practices and Effective Designs for Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599394

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599394

1 INTRODUCTION

Understanding how brain function emerges from the communication between neural elements remains a challenge in modern neuroscience [5]. Over the years, researchers have used brain graphs to encode the correlations of brain activities and uncover interesting connectivity patterns between brain regions. They find that the topological properties of brain graphs are useful in predicting various phenotypes and understanding brain activities [8, 13, 14, 26, 49], which account for the wide usage of brain graphs in neuroscientific research [39, 55, 70]. Adopting the graph representations (often termed "connectomes"), many neuroscientific problems can be cast as graph problems. In this paper, we focus on end-to-end brain graph classification tasks since many brain graph classification tasks have meaningful real-life clinical significance, such as providing a non-invasive neuroimaging biomarker for the identification of certain psychiatric/neurological disorders at an early stage (*e.g.* autism, Alzheimer's disease) [48].

Despite the benefits of modeling brain data as graphs, even well-preprocessed brain graphs pose serious challenges. A functional MRI-based (fMRI) brain graph, which is usually computed as pairwise correlations of fMRI time-series data, is fully connected. The resulting dense graph causes two unavoidable problems. First, it inhibits the use of efficient sparse operations, which leads to large time and memory consumption when the graphs are large [17, 70]. Second, the dense graph suffers from fMRI-related noise, making it extremely hard to train a model that learns useful generalization rules and provides good interpretability [41]. To this end, it is crucial to make brain graphs more sparse and less noisy. The common practice in neuroscience is to remove the "weak" edges, whose weights are below the predefined threshold [52]. However, direct thresholding requires a wide search for the proper threshold [10], and the sparsified graphs may lack useful edges and preserve significant noise. To illustrate it, in Table 1, we show the performance on the original graphs and sparsified graphs obtained using direct thresholding in a classification task. It can be seen that direct thresholding may drop important edges and/or keep unimportant edges, which leads to a decrease in performance.

Table 1: Brain graph classification performance (accuracy) on the original graphs (`Original`) and sparsified graphs (`Direct threholding`). `Direct thresholding` may keep unimportant edges. Details about the data and experimental setup can be found in Section 4.1.

	PicVocab	ReadEng
`Original`	$52.7_{\pm 3.77}$	$55.4_{\pm 3.51}$
`Direct thresholding`	$52.0_{\pm 5.51}$	$54.8_{\pm 3.19}$

Prior work related to graph sparsification generally falls into two categories. The first line of work learns the relative importance of the edges, which can be used to remove unimportant edges in the graph sparsification process. These works usually focus on interpretability explicitly, oftentimes referred to as "explainable graph neural networks (explainable GNNs)" [74]. The core idea embraced by this community is to identify small subgraphs that are most accountable for model predictions. The relevance of the edges to the final predictions is encoded into an edge importance mask, a matrix that reveals the relative importance of the edges and can be used to sparsify the graphs. These works show good interpretability under various measures [51]. However, it remains unclear whether better interpretability indicates better performance. The other line of work tackles unsupervised graph sparsification [42], without employing any label information. Some methods reduce the number of edges by approximating pairwise distances [50], cuts [33], or eigenvalues [58]. These task-irrelevant methods may discard useful task-specific edges for predictions. Fewer works are task-relevant, primarily focusing on node classification [43, 77]. Consequently, these works produce different edge importance masks for each graph. However, in graph classification, individual masks can lead to significantly longer training time and susceptibility to noise. Conversely, a joint mask emerges as the preferred choice, offering robustness against noise and greater interpretability.

This work. To assess the quality of the sparsified graphs obtained from interpretable models in the graph classification task, we propose to evaluate the effectiveness of the sparsification algorithms under an iterative framework. At each iteration, the sparsification algorithms decide which edges to remove and feed the sparsified graphs to the next iteration. We measure the effectiveness of a sparsification algorithm by computing the accuracy of the downstream graph classification task at each iteration. An effective sparsification algorithm should acquire the ability to identify and remove noisy edges, resulting in a performance boost in the graph classification task after several iterations (Section 4.2).

We utilize this iterative framework to evaluate two common practices used in graph sparsification and graph explainability: (1) obtaining the edge importance mask from a trained model and (2) learning an edge importance mask for each graph individually [74]. For instance, `GNNExplainer` [72] learns a separate edge importance mask for each graph **after** the model is trained. Through our empirical analysis, we find that these practices are **not helpful** in graph sparsification, as the sparsified graphs may lead to lower classification accuracy. In contrast, we identify three key strategies that can improve the performance. Specifically, we find that **(S1)** learning a **joint** edge importance mask **(S2) simultaneously with the training** of the model helps improve the performance over the iterations, as it passes task-relevant information through back-propagation. Another strategy to incorporate the task-relevant information is to **(S3)** initialize the mask with the gradient information from the immediate previous iteration. This strategy is inspired by the evidence in the computer vision domain that gradient information may encode data and task-relevant information and may contribute to the explainability of the model [1, 3, 27].

Based on the identified strategies, we propose a new **I**nterpretable model for brain **G**raph **S**parsification, `IGS`. We evaluate our `IGS` model on real-world brain graphs under the iterative framework and find that it can benefit from iterative sparsification. `IGS` achieves up to 5.1% improvement on graph classification tasks with graphs of 55.0% fewer edges than the original compared to strong baselines.

Our main contributions are summarized as follows:

- **General framework.** We propose a general iterative framework to analyze the effectiveness of different graph sparsification models. We find that edge importance masks generated from interpretable models may not be suitable for graph sparsification because they may not improve the performance of graph classification tasks.
- **New insights.** We find that two practices commonly used in graph sparsification and graph explainability are not helpful under the iterative framework. Instead, we find that learning a joint edge importance mask along with the training of the model improves the classification performance during iterative graph sparsification. Furthermore, incorporating gradient information in mask learning also boosts the performance in iterative sparsification.
- **Effective model.** Based on the insights, we propose a new model, `IGS`, which can improve the performance (up to 5.1%) with significantly sparser graphs (up to 55.0% less edges).
- **Interpretability.** Our `IGS` model learns to remove task-irrelevant edges in the iterative process. The edges that are retained by `IGS` have neuroscientific interpretations and are well supported by well-established literature.

2 NOTATION AND PRELIMINARIES

In this section, we introduce key notations, provide a brief background on GNNs, and formally define the problem that we investigate.

Notations. We consider a set of graphs $\mathcal{G}$. Each graph $G_i(\mathcal{V}, \mathcal{E}_i) \in \mathcal{G}$ in this set has n nodes, and the corresponding node set and edge set are denoted as $\mathcal{V}$ and $\mathcal{E}_i$, respectively. The graphs share the same set of nodes. The set of neighboring nodes of node v is denoted as $\mathcal{N}_v$. We focus on the setting where the input graphs are weighted, and we represent the weighted adjacency matrix of each input graph G_i as $\mathbf{A}_i \in \mathbb{R}^{n \times n}$. The node features in G_i are represented by a matrix $\mathbf{X}_i \in \mathbb{R}^{n \times d}$, where its j-th row $\mathbf{X}_i[j,:]$ represents the features of the j-th node, and d refers to the dimensionality of the node features. For conciseness, we use $\mathbf{X}_i^{(l)}$ to represent the node representations/output at the l-th layer of a GNN. Given our emphasis on graph classification problems, we denote the number of classes as k, the set of labels as $\mathcal{Y}$, and associate each graph G_i with a corresponding label $y_i \in \mathcal{Y}$.

We also leverage gradient information [56] in this work: $\nabla f_j(G_i)$ denotes the gradients of the output in class j with respect to the

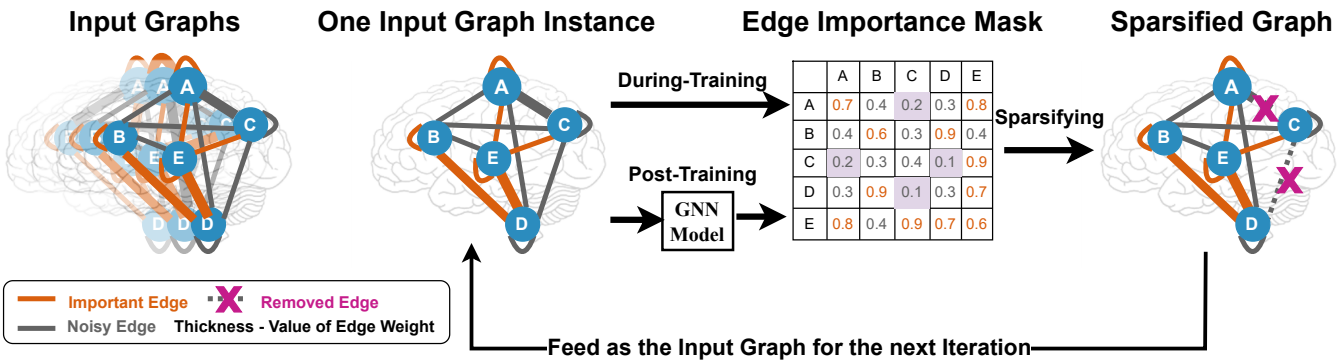

Figure 1: General iterative framework of sparsification. This framework progressively eliminates noisy edges from input brain graphs by learning an edge importance mask for each/all graph(s). The edge importance mask(s) can be generated from a well-trained GNN model or trained simultaneously with a GNN model. Important edges are depicted in orange, while noisy edges are shown in grey. Dashed lines with purple crosses represent the removed edges in the sparsified graphs.

input graph G_i. These gradients are obtained through backpropagation and are referred to as the gradient map.

Supervised Graph Classifcation. Given a set of graphs $\{G_1, G_2, \cdots, G_t\}$ and their labels $\{y_1, y_2, \cdots, y_t\}$ for training, we aim to learn a function $f : \mathcal{G} \rightarrow \mathcal{Y}$, such that the loss $\mathbb{E}(\mathcal{L}(y_i, \hat{y}_i))$ is minimized, where $\mathbb{E}$ denotes expectation, $\mathcal{L}$ denotes a loss function, and $\hat{y}_i = f(G_i)$ denotes the predicted label of G_i.

GNNs for Graph Classification. An L-layer GNN model [35, 62, 67, 69, 70] often follows the message-passing framework, which consists of three components [21]: (1) neighborhood propagation and aggregation: $\mathbf{m}_v^{(l)} = \texttt{AGGREGATE}(\mathbf{X}_i^{(l)}[u,:], u \in \mathcal{N}_v)$; (2) combination: $\mathbf{X}_i^{(l+1)}[v,:] = \texttt{COMBINE}(\mathbf{X}_i^{(l)}[v,:], \mathbf{m}_v^{(l)})$, where AGGREGATE and COMBINE are learnable functions; (3) global pooling. $\mathbf{x}^{G_i} = \texttt{Pooling}(\mathbf{X}_i^{(L)})$, where the Pooling function operates on all node representations, including options like Global_mean, Global_max or other complex pooling functions [37, 73]. The loss is given by $L = \frac{1}{N^G} \sum_{G_i \in \mathcal{G}_{\text{train}}} \texttt{CrossEntropy}(\texttt{Softmax}(\mathbf{x}^{G_i}), y_i)$, where $\mathcal{G}_{\text{train}}$ represents the set of training graphs and $N^G = |\mathcal{G}_{\text{train}}|$. Though our framework does not rely on specific GNNs, we illustrate the effectiveness of our framework using the GCN model proposed in [35].

The performance of GNN models heavily depends on the quality of the input graphs. Messages propagated through noisy edges can significantly affect the quality of the learned representations [70]. Inspired by this observation, we focus on the following problem:

Problem: Interpretable, Task-relevant Graph Sparsification.

Given a set of input graphs $\mathcal{G} = \{G_1, G_2, \cdots, G_t\}$ and the corresponding labels $\mathcal{Y} = \{y_1, y_2, \cdots, y_t\}$, we seek to learn a set of graph-specific edge importance masks $\{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_t\} \in \{0,1\}^{n \times n}$, **OR** a joint edge importance mask $\mathcal{M} \in \{0,1\}^{n \times n}$ shared by all graphs, which can be used to remove the noisy edges and retain the most task-relevant ones. This should lead to enhanced classification performance on sparsified graphs. Edge masks that effectively identify task-relevant edges are considered to be interpretable.

3 PROPOSED METHOD: IGS

In this section, we introduce our proposed iterative framework for evaluating various sparsification methods. Furthermore, we introduce IGS, a novel and interpretable graph sparsification approach that incorporates three key strategies: (S1) joint mask learning, (S2) simultaneous learning with the GNN model, and (S3) utilization of gradient information. We provide detailed explanations of these strategies in the following subsections.

3.1 Iterative Framework

Figure 1 illustrates the general iterative framework. At a high level, given a sparsification method, our framework iteratively removes unimportant edges based on the edge importance masks generated by the method at each iteration. In detail, the method can generate either a separate edge importance mask $\mathcal{M}_i$ for each input graph G_i or a joint edge importance mask $\mathcal{M}$ shared by all input graphs $\mathcal{G} = \{G_1, G_2, \cdots\}$. These edge importance masks indicate the relevance of edges to the task's labels. In our setting, we also allow training the masks simultaneously with the model. Ideal edge masks are binary, where zeros represent unimportant edges to be removed. In reality, many models (*e.g.* GNNs [72, 76]) learn soft edge importance masks with values between [0,1]. In each iteration, our framework removes either the edges with zero values in the masks (if binary) or a fixed percentage p of edges with the lowest importance scores in the masks. We present the framework of iterative sparsification in Algorithm 1, where $\mathcal{G}^i$ denotes the set of sparsified graphs at iteration i, and G_j^i denotes the j-th graph in the set $\mathcal{G}^i$.

Though existing works [28, 51] have proposed different ways to define the "importance" of an edge and thus they generate different sparse graphs, *we believe that a direct and effective way to evaluate these methods is to track the performance of these sparsified graphs* under this iterative framework. The trend of the performance reveals the relevance of the remaining edges to the predicted labels.

3.2 Strategies

3.2.1 Trained Mask (S1+S2). We aim to learn a joint edge importance mask $\mathcal{M} \in \{0,1\}^{n \times n}$ along with the training of a GNN model,

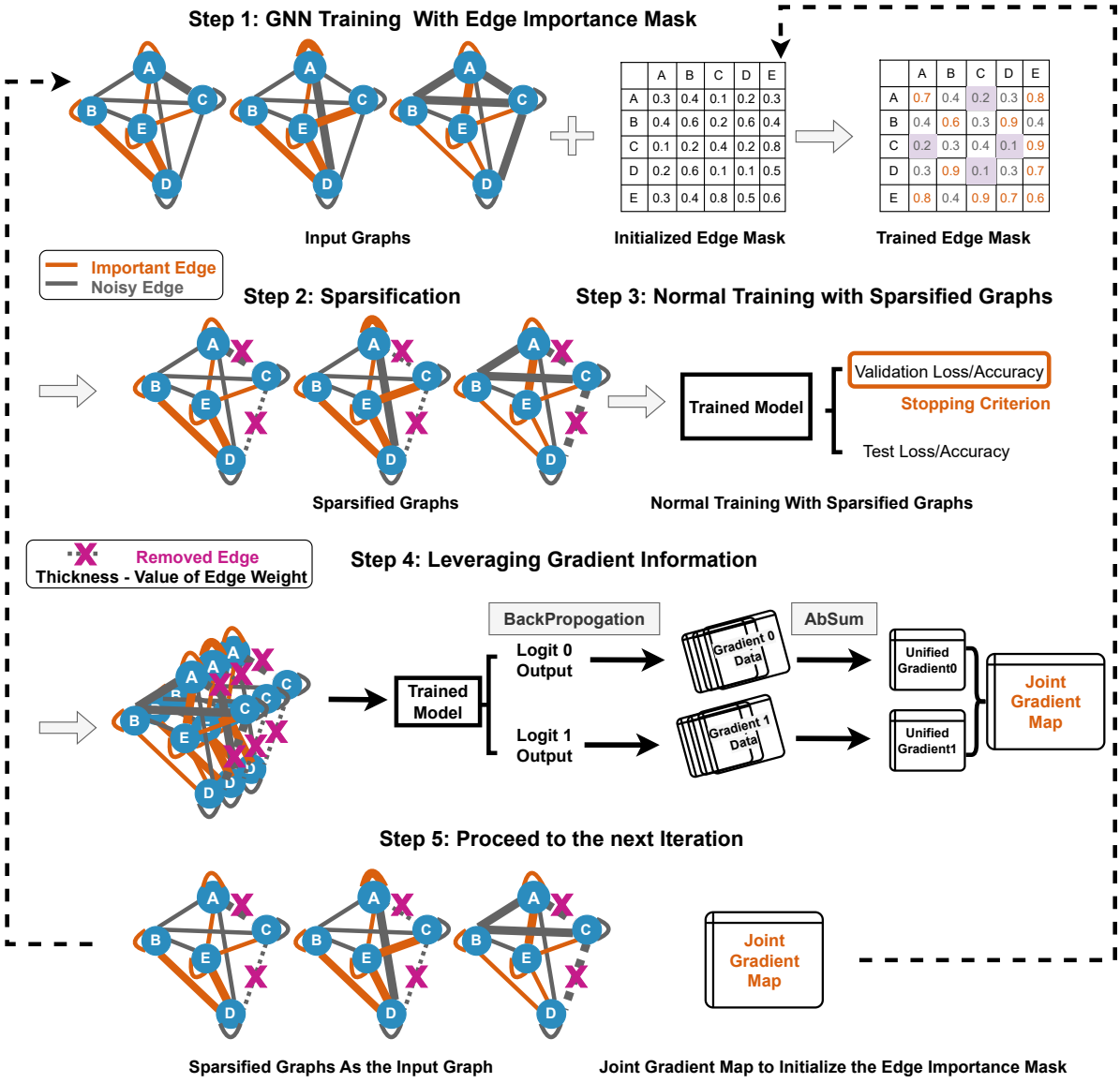

Figure 2: Training process of IGS. At iteration i, IGS takes a set of input graphs and initializes its joint edge importance mask using the joint gradient map from the previous iteration. It trains the GNN model and the edge importance mask together, followed by sparsifying all input graphs using the obtained mask. Normal training is then conducted on the sparsified graphs. The gradient information is later extracted by computing a joint gradient map. Finally, IGS feeds the sparsified graphs to the next iteration and uses the joint gradient map to initialize the subsequent joint edge importance mask. IGS is model-agnostic and can be seamlessly integrated with existing GNN models.

as shown in Figure 2. Each entry in $\mathcal{M}$ represents if the corresponding edge in the original input graph should be kept (value 1) or not (value 0). Directly learning the discrete edge mask is hard as it cannot generate gradients to propagate back. Thus, at each iteration, we learn a soft version of $\mathcal{M}$, where each entry is within $[0, 1]$ and reflects the relative importance of each edge. Considering the symmetric nature of the adjacency matrix for undirected brain graphs, we require the learned edge importance mask to be symmetric. We design the soft edge importance mask as $\sigma(\Phi^T + \Phi)$, where Φ is a matrix to be learned and σ is the `Sigmoid` function. A good initialization of Φ can boost the performance and accelerate the training speed. Thus, we initialize this matrix with the gradient map (Section 3.2.2) from the previous iteration (Step 5 in Figure 2). Furthermore, following [72], we regularize the training of Φ by requiring $\sigma(\Phi^T + \Phi)$ to be sparse. Thus we apply a l_1 regularization on $\sigma(\Phi^T + \Phi)$. In summary, we have the following training objective:

$$\min \mathcal{L}(f(\mathbf{A} \odot \sigma(\Phi^T + \Phi), \mathbf{X}), \mathcal{Y}) + \lambda \sum_{ij} \sigma(\Phi^T + \Phi)_{ij} \quad (1)$$

Algorithm 1 Iterative Sparsification Framework

INPUT: Sparsification Method S, Input Graph Set $\mathcal{G}^1$, Graph Labels $\mathcal{Y}$, Training Set Index $\mathbb{1}_{\text{Train}}$, Validation Set Index $\mathbb{1}_{\text{Val}}$, Number of Iterations N, a GNN model

for i = 1, …, N **do**
  if S. MaskTime ( ) == *PostTrain* **then**
    GNN_Trained ← Train (GNN, $\mathcal{G}^i[\mathbb{1}_{\text{Train}}], \mathcal{Y}[\mathbb{1}_{\text{Train}}]$)
    if S. MaskType ( ) == *Individual* **then**
      // *Individual Edge Importance Mask:*
      $\mathcal{M}_j = S$. MaskTrain (GNN_Trained, G_j^i, y_i)
      $G_j^{i+1} \leftarrow \mathcal{M}_j \odot G_j^i, \ \forall j$
    else
      // *Joint Edge Importance Mask:*
      $\mathcal{M} = S$. MaskTrain (GNN_Trained, $\mathcal{G}^i$, $\mathcal{Y}$, $\mathbb{1}_{\text{Train}}$)
      $G_j^{i+1} \leftarrow \mathcal{M} \odot G_j^i, \ \forall j$
  else
    // *Joint Edge Importance Mask:*
    $\mathcal{M} = S$. MaskTrain (GNN, $\mathcal{G}^i$, $\mathcal{Y}$, $\mathbb{1}_{\text{Train}}$)
    $G_j^{i+1} \leftarrow \mathcal{M} \odot G_j^i, \ \forall j$
  Validation loss L^i = Train&Val (GNN,
  $\mathcal{G}^{i+1}$, $\mathcal{Y}$, $\mathbb{1}_{\text{Train}}$, $\mathbb{1}_{\text{Val}}$)

OUTPUT: $\mathcal{G}^i$ with smallest L^i

where $\odot$ denotes the Hadamard product; $\mathcal{L}$ is the `Cross-Entropy` loss; λ is the regularization coefficient. We optimize the joint mask across all training samples in a batch-training fashion to achieve our objective of learning a shared mask. Subsequently, we convert this soft mask into an indicator matrix by assigning zero values to the lowest p percentage of elements:

$$\mathcal{M}[i,j] = \begin{cases} 0 & \text{if } \sigma(\Phi^T + \Phi)_{ij} \text{ in lowest } p\% \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

The indicator matrix $\mathcal{M}$ can then be used to sparsify the input graph through an element-wise multiplication, *e.g.* $G_i' = \mathcal{M} \odot G_i$.

3.2.2 Joint Gradient Information (S3). Inspired from the evidence in the computer vision domain that gradient information may encode data and task-relevant information and may contribute to the explainability of the model [1, 3, 27], we utilize the gradient information, i.e., gradient maps to initialize and guide the learning of the edge importance mask.

Step 4 in Figure 2 illustrates the general idea of generating a joint gradient map by combining gradient information from each training graph. Each training graph G_i has k gradient maps $\nabla f_j(G_i), j = 1, 2, \cdots, k$, each corresponding to the output in class j (Section 2). Instead of using the "saliency maps" [56], which consider only the gradient maps from the predicted class, we leverage all the gradient maps as they provide meaningful knowledge. For $G_1, \ldots, G_n \in \mathcal{G}_{\text{train}}$, we compute the unified mask of class j as the sum of the absolute values of each gradient map, represented as

$$\bigcup f_j = \sum_{i=1}^{t} |\nabla f_j(G_i)| \tag{3}$$

By summing the unified masks of all classes, we generate the joint edge gradient map denoted as $\mathbf{T} = \sum_{j=1}^{k} \bigcup f_j$.

3.2.3 Algorithm. We incorporate these three strategies into `IGS` and outline our method in Algorithm 2:

Algorithm 2 Interpretable Graph Sparsification: `IGS`

INPUT: Input Graph Dataset $\mathcal{G}^1$, Training Set Index $\mathbb{1}_{\text{Train}}$, Validation Set Index $\mathbb{1}_{\text{Val}}$, Removing Percentage p, Number of Iterations N, GNN model, Regularization Coeffient λ

for i = 1, …, N **do**
  // **Step 1: GNN Training with Edge Importance Mask**
  if i == 1 **then**
    Initialize Φ using Xavier normal initiation.
  else
    Initialize Φ using the previous joint gradient map $\mathbf{T}^{(i)}$
  $\sigma(\Phi^T + \Phi) \leftarrow$ Train (GNN, $\mathcal{G}^i$, $\mathcal{Y}$, $\mathbb{1}_{\text{train}}$, λ). (Equation (1))
  Obtain *joint Edge Importance Mask* $\mathcal{M}$ following Equation (2)
  // **Step 2: Sparsification**
  $G_j^{i+1} \leftarrow \mathcal{M} \odot G_j^i, \ \forall j$
  // **Step 3: Normal Training with Sparsified Graphs**
  Validation loss L^i, GNN_Trained = Train&Val (GNN,
  $\mathcal{G}^{i+1}$, $\mathcal{Y}$, $\mathbb{1}_{\text{Train}}$, $\mathbb{1}_{\text{Val}}$)
  // **Step 4: Leveraging Gradient Information**
  $\mathbf{T}^{(i+1)} \leftarrow$ JointGradient(GNN_Trained, $\mathcal{G}^{i+1}$, $\mathbb{1}_{\text{Train}}$)

OUTPUT: $\mathcal{G}^i$ with smallest L^i

4 EMPIRICAL ANALYSIS

In this section, we aim to answer the following research questions using our iterative framework: (Q1) Is learning a joint edge importance mask better than learning a separate mask for each graph? (Q2) Does simultaneous training of the edge importance mask with the model yield better performance than training the mask separately from the trained model? (Q3) Does the gradient information help with graph sparsification? (Q4) Is our method `IGS` interpretable?

4.1 Setup

4.1.1 Dataset. We use the WU-Minn Human Connectome Project (HCP) 1200 Subjects Data Release as our benchmark dataset to evaluate our method and baselines [61]. The pre-processed brain graphs can be obtained from ConnectomeDB [45]. These brain graphs are derived from the resting-state functional magnetic resonance imaging (rs-fMRI) of 812 subjects, where no explicit task is being performed. Predictions using rs-fMRI are generally harder than task-based fMRI [44]. The obtained brain graphs are fully connected, and the edge weights are computed from the correlation of the rs-fMRI time series between each pair of brain regions [57]. The parcellation of the brain is completed using Group-ICA with 100 components [9, 20, 22–24, 54], which results in 100 brain regions comprising the nodes of our brain graphs. Additionally, a set of cognitive assessments were performed on each subject, which we utilized as cognitive labels in our prediction tasks. Specifically,

we utilize the scores from the following cognitive domains as our labels, which incorporate age adjustment [45]:

- PicVocab (Picture Vocabulary) assesses language/vocabulary comprehension. The respondent is presented with an audio recording of a word and four photographic images on the computer screen and is asked to select the picture that most closely matches the word's meaning.
- ReadEng (Oral Reading Recognition) assesses language/reading decoding. The participant is asked to read and pronounce letters and words as accurately as possible. The test administrator scores them as right or wrong.
- PicSeq (Picture Sequence Memory) assesses the Open of episodic memory. It involves recalling an increasingly lengthy series of illustrated objects and activities presented in a particular order on the computer screen.
- ListSort (List Sorting) assesses working memory and requires the participant to sequence different visually- and orally-presented stimuli.
- CardSort (Dimensional Change Card Sort) assesses the cognitive flexibility. Participants are asked to match a series of bivalent test pictures (e.g., yellow balls and blue trucks) to the target pictures, according to color or shape. Scoring is based on a combination of accuracy and reaction time.
- Flanker (Flanker Task) measures a participant's attention and inhibitory control. The test requires the participant to focus on a given stimulus while inhibiting attention to stimuli flanking it. Scoring is based on a combination of accuracy and reaction time.

More details can be found in ConnectomeDB [45]. These scores are continuous. In order to use them for graph classification, we assign the subjects achieving scores in the top third to the first class and the ones in the bottom third to the second class.

4.1.2 Baselines. We outline the baselines used in our experiments.

Grad-Indi [7]. This method obtains the edge importance mask for each individual graph from a trained GNN model. In contrast to the gradient information (Strategy S3) proposed in Section 3.2.2, a gradient map of each sample is generated for the predicted class C_i: $\mathbf{T}_i = \nabla f_{C_i}(G_i) \odot \nabla f_{C_i}(G_i)$ [7]. Later, the edge importance mask $\mathcal{M}_i$ for G_i is generated based on Equation (2).

Grad-Joint. We adapt Grad-Indi [7] to incorporate our proposed strategies (S1+S3) and learn an edge importance mask *shared by all graphs* from a *trained* GNN model. Specifically, we leverage the method described in Section 3.2.2 that generates the joint gradient map to obtain the joint importance mask.

Grad-Trained. We further modify Grad-Indi [7] to train the joint edge mask concurrently with the GNN training (S2). We also use the joint gradient map (Section 3.2.2) to initialize the edge importance mask (Strategies S1+S2+S3). The main differences of `Grad-Trained` from `IGS` are that: (1) it does not require symmetry of the edge mask; (2) it does not require edge mask sparsity (without l_1 regularization).

GNNExplainer-Indi [72]. This method trains an edge important mask for each individual graph after the GNN model is trained. We follow the code provided by [40].

GNNExplainer-Joint. Adapted from [72], this model trains a joint edge important mask for all graphs (Strategy S1).

GNNExplainer-Trained. Adapted from [72], this method simultaneously trains a joint edge important mask and the GNN model (Strategies S1+S2). Compared with `IGS`, this method does not use gradient information.

BrainNNExplainer [18]. This method (also known as IBGNN) trains a joint edge important mask for all graphs after the GNN is trained. It is slightly different from `GNNExplainer-Joint` in terms of objective functions. We follow the original setup in [18].

BrainGNN [38]. This method does not explicitly perform the graph sparsification task, but uses node pooling to identify important subgraphs. It learns to preserve important nodes and all the connections between them. We follow the original setup in [38].

4.1.3 Training Setup. To fairly evaluate different methods under the iterative framework, we adopt the same GNN architecture [34], hyper-parameter settings, and training framework. We set the number of convolutional layers to four, the dimension of the hidden layers to 256, the dropout rate to 0.5, the batch size to 16, the optimizer to Adam, the learning rate to 0.001, and the regularization coefficient λ to 0.0001. Note that though we use the GNN from [34], `IGS` is model-agnostic, and we provide the results of other backbone GNNs in Table 4. For each prediction task, we shuffle the data and take four different data splits. The train/val/test split is 0.7/0.15/0.15. To reduce the influence of imbalances, we manually ensure each split has equal labels. In each iteration, we adopt early stopping [53] and set the patience to 100 epochs. We stop training if we cannot observe a decrease in validation loss in the latest 100 epochs. We fix the removing ratio $p\%$ to be 5% per iteration. In the iterative sparsification, we run a total of 55 iterations and use the validation loss of the sparsified graphs as the criterion to select the best iteration (Step 3 in Figure 2). We present the average and standard deviation of test accuracies over four splits, using the model obtained from the best iteration. **The code is available at https://github.com/motivationss/IGS.git.**

4.2 (Q1-Q3) Graph Classification under the Iterative Framework

In Table 2, we present the results of `IGS` with the eight baselines mentioned in section 4.1.2. The first row represents the prediction task we study; the second row represents the performance averaged across four different splits using the original graph; and the rest of the rows denote the performance of other baselines. Notably, for better comparison across different baselines, the last column shows the average rank of each method. Below we present our observations from Table 2:

First, learning a joint mask contributes to a better performance than learning a mask for each graph separately. We can start by comparing the performance between `GNNExplainer-Joint` and `GNNExplainer-Indi` as well as `Grad-Joint` and `Grad-Indi`. The performance disparity between the methods in each pair is notable and consistent across all prediction tasks. Notably, `Grad-Joint` (rank: 4.33) outperforms `Grad-Indi` (rank: 7.67) by a considerable margin, while `GNNExplainer-Joint` (rank: 4.67) ranks significantly

Table 2: Results of test accuracies of different approaches evaluated on six prediction tasks (PicVocab, ReadEng, PicSeq, ListSort, CardSort, and Flanker) across four data splits generated by different random seeds. We report the mean and standard deviation for each of them. The first row denotes the performance using the original graph trained by GCN [34]; the last column denotes the average rank of each method. The best result is marked in bold.

	PicVocab	ReadEng	PicSeq	ListSort	CardSort	Flanker	Average Rank
GCN (Original Graphs)	52.7±3.77	55.4±3.51	51.9±2.18	52.1±2.55	56.6±6.50	48.91±5.83	-
Grad-Indi	53.4±1.65	53.7±9.48	49.3±3.71	48.7±6.94	46.9±4.65	50.7±2.76	7.67
Grad-Joint	57.8±3.34	58.2±3.08	50.1±6.17	48.9±5.10	52.4±5.02	51.5±3.94	4.33
Grad-Trained	55.5±5.29	60.0±1.36	49.5±4.12	50.2±2.20	56.3±7.66	51.6±4.03	3.83
GNNExplainer-Indi	49.7±3.86	55.3±4.06	48.9±3.29	44.8±3.76	52.1±3.86	47.3±1.58	8.33
GNNExplainer-Joint	56.4±7.94	55.8±7.33	52.0±2.84	50.1±3.01	53.5±8.32	50.3±5.81	4.67
GNNExplainer-Trained	56.8±3.10	59.2±2.96	51.4±3.51	51.2±2.01	56.0±4.71	50.9±2.01	3.17
BrainNNExplainer	57.0±3.77	55.7±5.76	50.3±1.47	49.8±4.47	52.4±3.63	50.9±3.95	4.83
BrainGNN	53.0±3.25	47.5±3.00	50.7±3.13	50.9±3.13	50.1±1.12	49.0±6.22	6.67
IGS	**57.8**±3.10	**60.1**±2.78	**53.0**±4.66	**51.8**±2.12	**57.0**±5.49	**52.1**±1.97	1.00

higher than GNNExplainer-Indi (rank: 8.33). Using a joint mask instead of individual masks can provide up to 6.7% boost in accuracy, validating our intuition in section 3.2.2 that a joint mask is more robust to sample-wise noise.

Second, training the mask and the GNN model simultaneously yields better results than obtaining the mask from the trained model. We can see this by comparing the performance between the Trained and the Joint variants of Grad and GNNExplainer. Changing from post-training to joint-training can provide up to 3.4% performance improvements, as demonstrated in the ReadEng task by the two variants of GNNExplainer. Even though in some tasks the post-training approach may outperform the trained approach (*e.g.* Grad-Joint and Grad-Trained in the PicVocab task), the trained approach has a higher average rank than the post-training approach (*e.g.* 3.83 vs. 4.33 for Grad and 3.17 vs. 4.67 for GNNExplainer). In addition, the better performance of IGS over BrainNNExplainer also demonstrates the effectiveness of obtaining the edge mask during training rather than after training.

Third, incorporating gradient information helps improve classification performance. We can see this by first comparing the performance of Grad-Joint and Grad-Trained against the original graphs. The use of gradient information can provide up to 5.1% higher accuracy, though the improvement depends on the task. Furthermore, since the main difference between GNNExplainer-Trained and IGS lies in the use of gradient information, the consistent superior performance of IGS strengthens this conclusion.

Fourth, we compare the performance of the baselines against the performance of the original graphs (second row). Grad-Indi [7] and GNNExplainer-Indi [72] are implementations that faithfully follow their original formulation or are provided directly by the authors. These two approaches fail to achieve any performance improvements through iterative sparsification, with the exception of Grad-Indi in the task of PicVocab and ReadEng. This raises the question of whether these existing instance-level approaches can identify the most meaningful edges in noisy graphs. These methods may be vulnerable to severe sample-wise noise. On the contrary, with our suggested modifications, the joint and trained versions can remove the noise and provide up to 5.1% performance boost compared to the base GCN method applied to the original graphs. However, the improvement is dataset-dependent. For instance, GNNExplainer-Trained provides decent performance boosts in PicVocab, ReadEng, and Flanker, but degrades in PicSeq, ListSort, and CardSort.

Finally, our proposed approach, IGS, achieves the **best** performance across all prediction tasks, demonstrated by its highest rank among all methods. Compared with the performance on the original graphs, IGS can provide consistent performance boost across all prediction tasks, with the exception of ListSort, which is a challenging task that no baseline surpasses the original performance. Furthermore, using the sparsified graph identified by IGS generally results in less variance in accuracy and leads to better stability when compared to the original graphs, with the exception on the PicSeq task. In addition, the superior performance of IGS over BrainGNN demonstrates the effectiveness of using edge importance masks as opposed to node pooling.

Graph Sparsity. In Table 3, we present the final average sparsity of the graphs obtained by IGS over four data splits. We observe that with significantly fewer edges retained, IGS can still achieve up to 5.1% performance boost.

Table 3: Final sparsity of the sparsified brain graphs identified by IGS averaged over different splits. The initial sparsity is 50% by thresholding. IGS can remove more than half of the edges while achieving up to 5.1% performance boost.

	PicVocab	ReadEng	PicSeq	ListSort	CardSort	Flanker
Sparsity(%)	22.5	35.5	35.5	30.0	25.0	25.0

4.3 (Q4) Interpretability of IGS

We now evaluate the interpretability of the edge masks derived for each of our prediction tasks.

Setup. We assign anatomical labels to each of the 100 components comprising the nodes of our brain networks by computing the

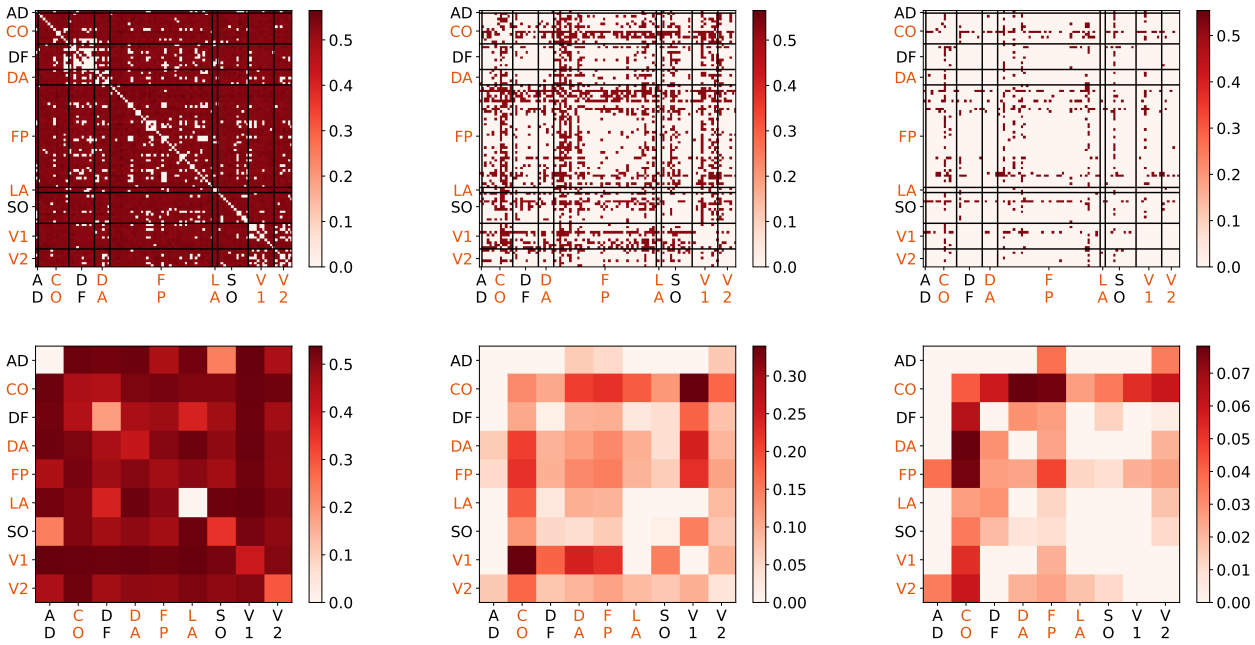

Figure 3: Weighted brain network edge masks at both node (top row) and subnetwork level (bottom row - computed as the average of corresponding edges) for PicVocab task. Early, middle, and final phases of training are depicted from left to right, and high-importance subnetworks are highlighted in red. We find that IGS gradually removes noisy edges and its final edge importance mask can provide high-quality interpretations. Highlighted (Orange) label names represent the regions that are meaningful in this task. Brain network labels and abbreviations: Auditory (AD), Cingulo-Opercular (CO), Dorsal Attention (DA), Default (DF), Frontoparietal (FP), Language (LA), Somatomotor (SO), Visual 1 (V1), Visual 2 (V2).

largest overlap between regions identified in the Cole-Anticevic parcellation [30]. We then obtained the edge masks from the best-performing iteration of each prediction task and assessed the highest-weighted edges in each mask.

Results. Since our IGS model performed best in the language-related prediction tasks, ReadEng and PicVocab, we focus our interpretability analysis on this domain. There is ample evidence in the neuroscience literature that supports the existence of an intrinsic language network that is perceptible during resting state [11, 36, 59]; thus, it is unsurprising that our rs-fMRI based brain networks are predictive of language task performance. It has also been well established for over a century that the language centers (including Broca's area, Wernicke's area, the angular gyrus, etc.) are characteristically left-lateralized in the brain [12, 65]. In both ReadEng and PicVocab, the majority of the highest weighted edges retained in the masks involved brain regions localized to the left hemisphere, falling in line with the expectations for a language task.

PicVocab. Figures 3 and 4 depict the progression of the edge masks at both the node and subnetwork level over the training iterations towards optimal edge mask in both the ReadEng and PicVocab tasks. Evaluating the edge masks at the subnetwork level offers valuable insights into which functional connections are most important for the prediction of each task. The PicVocab edge mask homed in on functional connections involving the Cingulo-Opercular (CO) network, specifically between CO and the Dorsal Attention (DA), Visual1 (V1), Visual2 (V2) and Frontoparietal (FP) networks. The CO network has been shown to be implicated in word recognition [60], and its synchrony with other brain networks identified here may represent the stream of neural processing related to the PicVocab task, in which subjects respond to an auditory stimulus of a word and are prompted to choose the image that best represents the word. Connectivity between the Auditory (AD) and V2 networks is also evident in the PicVocab edge mask, suggesting the upstream integration of auditory and visual stimuli involved in the PicVocab task are also predictive of task performance.

ReadEng. The IGS model also found edge mask connections between the V1 network and the CO, Language (LA) and DA networks, as well as CO-LA and CO-AD connections, to be most predictive of ReadEng performance. This task involves the subject reading aloud words presented on a screen. From our results, it follows that the ability of Vis1 to integrate with networks responsible for language processing (LA and CO) and attention (DA), as well as the capacity for functional synchrony between the language-related networks (CO-LA), would be predictive of overall ReadEng performance. The importance of the additional CO-AD connectivity identified by our model also suggests that the ability of the CO language network to integrate with auditory centers may be involved in the neural processes responsible for the proper pronunciation of the words given by visual cues.

Key take-aways. Overall, in addition to the IGS model's superior classification performance, our results suggest that the iterative pruning of the IGS edge masks during training does indeed retain important and neurologically meaningful edges while removing noisy connections. While it has been shown in the literature that resting-state connectivity can be used to predict task performance [6, 32, 46], the ability of the IGS model to sparsify the resting state brain graph to clearly task-relevant edges for prediction of task performance further underscores the interpretability of the resultant edge masks.

5 RELATED WORK

5.1 Graph Explainability

Our work is related to explainable GNNs given that we identify important edges/subgraphs that account for the model predictions. Some explainable GNNs are "perturbation-based", where the goal is to investigate the relation between output and input variations. GNNExplainer [72] learns a soft mask for the nodes and edges, which explains the predictions of a well-trained GNN model. SubgraphX [75] explains its predictions by efficiently exploring different subgraphs with a Monte Carlo tree search. Another approach for explainable GNNs is surrogate-based; the methods in this category generally construct a simple and interpretable surrogate model to approximate the output of the original model in certain neighborhoods [74]. For instance, GraphLime [29] considers the N-hop neighboring nodes of the target node and then trains a nonlinear surrogate model to fit the local neighborhood predictions; RelEx [76] first uses a GNN to fit the BFS-generated datasets and then generates soft masks to explain the predictions; PGM-Explainer [64] generates local datasets based on the influence of randomly perturbing the node features, shrinks the size of the datasets via the Grow-Shrink algorithm, and employs a Bayesian network to fit the datasets. In general, most of these methods focus on the node classification task and make explanations for a single graph, which is not applicable to our setting. Others only apply to simple graphs, which cannot handle signed and weighted brain graphs [29, 75]. Additionally, most methods generate explanations after a GNN is trained. Though some methods achieve decent results in explainability-related metrics (*e.g.* fidelity scores [51]), it remains unclear whether their explanations can necessarily remove noise and retain the "important" part of the original graph, which improves the classification accuracy.

5.2 Graph Sparsification

Compared to the explainable GNN methods, graph sparsification methods explicitly aim to sparsify graphs. Most of the existing methods are unsupervised [77]. Conventional methods reduce the size of the graph through approximating pairwise distances [50], preserving various kinds of graph cuts [33], node degree distributions [19, 63], and using some graph-spectrum based approachse [2, 15, 16]. These methods aim at preserving the structural information of the original input graph without using the label information, and they assume that the input graph is unweighted. Relatively fewer supervised works have been proposed. For example, NeuralSparse [77] builds a parametrized network to learn a k-neighbor subgraph by limiting each node to have at most k edges. On top of NeuralSparse, PTDNet [43] removes the k-neighbor assumption, and instead, it employs a low-rank constraint on the learned subgraph to discourage edges connecting multiple communities. Graph Condensation [31] proposes to parameterize the condensed graph structure as a function of condensed node features and optimizes a gradient-matching training objective. Despite the new insights offered by these methods, most of them focus exclusively on node classification, and their training objectives are built on top of that. A work that shares similarity to our proposed method, IGS, is BrainNNExplainer [18] (also known as IBGNN). It is inspired by GNNExplainer [72] and obtains the joint edge mask in a post-training fashion. On the other hand, our proposed method, IGS, trains a joint edge mask along with the backbone model and incorporates gradient information in an iterative manner. Another line of work leverages node pooling to identify important subgraphs, and learns to preserve important nodes and all the connections between them. One representative work is BrainGNN [38]. However, the connections between preserved nodes are not necessarily all informative, and some may contain noise.

5.3 Saliency Maps

Saliency maps are first proposed to explain the deep convolutional neural network models in image classification tasks [56]. Specifically, the method proposes to use the gradients backpropagated from the predicted class as the explanations. Recently, [7] introduces the concept of saliency maps to graph neural networks, employing squared gradients to explain the underlying model. Additionally, [4] suggests using graph saliency to identify regions of interest (ROIs). In general, the gradients backpropagated from the output logits can serve as the importance indicators for model predictions. In this work, inspired by the line of saliency-related works, we leverage the gradient information to guide our model.

6 CONCLUSIONS

In this paper, we studied neural-network-based graph sparsification for brain graphs. By introducing an iterative sparsification framework, we identified several effective strategies for GNNs to filter out noisy edges and improve the graph classification performance. We combined these strategies into a new interpretable graph classification model, IGS, which improves the graph classification performance by up to 5.1% with 55% fewer edges than the original graphs. The retained edges identified by IGS provide neuroscientific interpretations and are supported by well-established literature.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive feedback. This material is based upon work supported by the National Science Foundation under IIS 2212143, CAREER Grant No. IIS 1845491, a Precision Health Investigator Award at the University of Michigan, and AWS Cloud Credits for Research. Data were provided [in part] by the Human Connectome Project, WU-Minn Consortium (PIs: D. Van Essen and K. Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the

authors and do not necessarily reflect the views of the National Science Foundation or other funding parties.

REFERENCES

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. *Advances in neural information processing systems* 31 (2018).

[2] Bijaya Adhikari, Yao Zhang, Sorour E Amiri, Aditya Bharadwaj, and B Aditya Prakash. 2017. Propagation-based temporal network summarization. *IEEE Transactions on Knowledge and Data Engineering* 30, 4 (2017), 729–742.

[3] Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. 2020. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 275–285.

[4] Salim Arslan, Sofia Ira Ktena, Ben Glocker, and Daniel Rueckert. 2018. Graph saliency maps through spectral convolutional networks: Application to sex classification with brain connectivity. In *Graphs in Biomedical Image Analysis and Integrating Medical Imaging and Non-Imaging Modalities: Second International Workshop, GRAIL 2018 and First International Workshop, Beyond MIC 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 2*. Springer, 3–13.

[5] Andrea Avena-Koenigsberger, Bratislav Misic, and Olaf Sporns. 2018. Communication dynamics in complex brain networks. *Nature reviews neuroscience* 19, 1 (2018), 17–33.

[6] Antonello Baldassarre, Christopher M Lewis, Giorgia Committeri, Abraham Z Snyder, Gian Luca Romani, and Maurizio Corbetta. 2012. Individual variability in functional connectivity predicts performance of a perceptual task. *Proceedings of the National Academy of Sciences* 109, 9 (2012), 3516–3521.

[7] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).

[8] Danielle Smith Bassett and ED Bullmore. 2006. Small-world brain networks. *The neuroscientist* 12, 6 (2006), 512–523.

[9] Christian F Beckmann and Stephen M Smith. 2004. Probabilistic independent component analysis for functional magnetic resonance imaging. *IEEE transactions on medical imaging* 23, 2 (2004), 137–152.

[10] Cécile Bordier, Carlo Nicolini, and Angelo Bifone. 2017. Graph analysis and modularity of brain functional connectivity networks: searching for the optimal threshold. *Frontiers in neuroscience* 11 (2017), 441.

[11] Paulo Branco, Daniela Seixas, and Sao L Castro. 2020. Mapping language with resting-state functional magnetic resonance imaging: A study on the functional profile of the language network. *Human Brain Mapping* 41, 2 (2020), 545–560.

[12] Paul Broca. 1861. Remarques sur le siège de la faculté du langage articulé, suivies d'une observation d'aphémie (perte de la parole). *Bulletin et Memoires de la Societe anatomique de Paris* 6 (1861), 330–357.

[13] Ed Bullmore and Olaf Sporns. 2009. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience* 10, 3 (2009), 186–198.

[14] Edward T Bullmore and Danielle S Bassett. 2011. Brain graphs: graphical models of the human brain connectome. *Annual review of clinical psychology* 7 (2011), 113–140.

[15] Daniele Calandriello, Alessandro Lazaric, Ioannis Koutis, and Michal Valko. 2018. Improved large-scale graph learning through ridge spectral sparsification. In *International Conference on Machine Learning*. PMLR, 688–697.

[16] Alireza Chakeri, Hamidreza Farhidzadeh, and Lawrence O Hall. 2016. Spectral sparsification in spectral clustering. In *2016 23rd international conference on pattern recognition (icpr)*. IEEE, 2301–2306.

[17] Moo K Chung. 2018. Statistical challenges of big brain network data. *Statistics & probability letters* 136 (2018), 78–82.

[18] Hejie Cui, Wei Dai, Yanqiao Zhu, Xiaoxiao Li, Lifang He, and Carl Yang. 2021. Brainnnexplainer: An interpretable graph neural network framework for brain network based disease analysis. *arXiv preprint arXiv:2107.05097* (2021).

[19] Talya Eden, Shweta Jain, Ali Pinar, Dana Ron, and C Seshadhri. 2018. Provable and practical approximations for the degree distribution using sublinear graph samples. In *Proceedings of the 2018 World Wide Web Conference*. 449–458.

[20] Bruce Fischl. 2012. FreeSurfer. *Neuroimage* 62, 2 (2012), 774–781.

[21] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *ICML* (2017).

[22] Matthew F Glasser, Timothy S Coalson, Emma C Robinson, Carl D Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F Beckmann, Mark Jenkinson, et al. 2016. A multi-modal parcellation of human cerebral cortex. *Nature* 536, 7615 (2016), 171–178.

[23] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. 2013. The minimal preprocessing pipelines for the Human Connectome Project. *Neuroimage* 80 (2013), 105–124.

[24] Ludovica Griffanti, Gholamreza Salimi-Khorshidi, Christian F Beckmann, Edward J Auerbach, Gwenaëlle Douaud, Claire E Sexton, Enikő Zsoldos, Klaus P Ebmeier, Nicola Filippini, Clare E Mackay, et al. 2014. ICA-based artefact removal and accelerated fMRI acquisition for improved resting state network imaging. *Neuroimage* 95 (2014), 232–247.

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[26] Christopher J Honey, Olaf Sporns, Leila Cammoun, Xavier Gigandet, Jean-Philippe Thiran, Reto Meuli, and Patric Hagmann. 2009. Predicting human resting-state functional connectivity from structural connectivity. *Proceedings of the National Academy of Sciences* 106, 6 (2009), 2035–2040.

[27] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. 2015. Online tracking by learning discriminative saliency map with convolutional neural network. In *International conference on machine learning*. PMLR, 597–606.

[28] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems* 32 (2019).

[29] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. 2022. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[30] Jie Lisa Ji, Marjolein Spronk, Kaustubh Kulkarni, Grega Repovš, Alan Anticevic, and Michael W Cole. 2019. Mapping the human brain's cortical-subcortical functional network organization. *Neuroimage* 185 (2019), 35–57.

[31] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580* (2021).

[32] O Parker Jones, NL Voets, JE Adcock, R Stacey, and S Jbabdi. 2017. Resting connectivity predicts task activation in pre-surgical populations. *NeuroImage: Clinical* 13 (2017), 378–385.

[33] David R Karger. 1994. Random sampling in cut, flow, and network design problems. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 648–657.

[34] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[35] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[36] Julian Klingbeil, Max Wawrzyniak, Anika Stockert, and Dorothee Saur. 2019. Resting-state functional connectivity: An emerging method for the study of language networks in post-stroke aphasia. *Brain and cognition* 131 (2019), 22–33.

[37] Boris Knyazev, Graham W Taylor, and Mohamed Amer. 2019. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems* 32 (2019).

[38] Xiaoxiao Li, Yuan Zhou, Nicha Dvornek, Muhan Zhang, Siyuan Gao, Juntang Zhuang, Dustin Scheinost, Lawrence H Staib, Pamela Ventola, and James S Duncan. 2021. Braingnn: Interpretable brain graph neural network for fmri analysis. *Medical Image Analysis* 74 (2021), 102233.

[39] Martin A Lindquist. 2008. The statistical analysis of fMRI data. *Statistical science* 23, 4 (2008), 439–464.

[40] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, et al. 2021. DIG: A turnkey library for diving into graph deep learning research. *The Journal of Machine Learning Research* 22, 1 (2021), 10873–10881.

[41] Thomas T Liu. 2016. Noise contributions to the fMRI signal: An overview. *NeuroImage* 143 (2016), 141–151.

[42] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph summarization methods and applications: A survey. *ACM computing surveys (CSUR)* 51, 3 (2018), 1–34.

[43] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*. 779–787.

[44] Han Lv, Zhenchang Wang, Elizabeth Tong, Leanne M Williams, Greg Zaharchuk, Michael Zeineh, Andrea N Goldstein-Piekarski, Tali M Ball, Chengde Liao, and Max Wintermark. 2018. Resting-state functional MRI: everything that nonexperts have always wanted to know. *American Journal of Neuroradiology* 39, 8 (2018), 1390–1399.

[45] Daniel S Marcus, John Harwell, Timothy Olsen, Michael Hodge, Matthew F Glasser, Fred Prior, Mark Jenkinson, Timothy Laumann, Sandra W Curtiss, and David C Van Essen. 2011. Informatics and data mining tools and strategies for the human connectome project. *Frontiers in neuroinformatics* 5 (2011), 4.

[46] Maarten Mennes, Clare Kelly, Xi-Nian Zuo, Adriana Di Martino, Bharat B Biswal, F Xavier Castellanos, and Michael P Milham. 2010. Inter-individual differences in resting-state functional connectivity predict task-induced BOLD activity. *Neuroimage* 50, 4 (2010), 1690–1701.

[47] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4602–4609.

[48] Angela M Muller and Martin Meyer. 2014. Language in the brain at rest: new insights from resting state data and graph theoretical analysis. *Frontiers in human neuroscience* 8 (2014), 228.
[49] Chang-hyun Park, Soo Yong Kim, Yun-Hee Kim, and Kyungsik Kim. 2008. Comparison of the small-world topology between anatomical and functional connectivity in the human brain. *Physica A: statistical mechanics and its applications* 387, 23 (2008), 5958–5962.
[50] David Peleg and Alejandro A Schäffer. 1989. Graph spanners. *Journal of graph theory* 13, 1 (1989), 99–116.
[51] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10772–10781.
[52] Jonathan D Power, Alexander L Cohen, Steven M Nelson, Gagan S Wig, Kelly Anne Barnes, Jessica A Church, Alecia C Vogel, Timothy O Laumann, Fran M Miezin, Bradley L Schlaggar, et al. 2011. Functional network organization of the human brain. *Neuron* 72, 4 (2011), 665–678.
[53] Lutz Prechelt. 2012. Early stopping—but when? *Neural networks: tricks of the trade: second edition* (2012), 53–67.
[54] Emma C Robinson, Saad Jbabdi, Matthew F Glasser, Jesper Andersson, Gregory C Burgess, Michael P Harms, Stephen M Smith, David C Van Essen, and Mark Jenkinson. 2014. MSM: a new flexible framework for multimodal surface matching. *Neuroimage* 100 (2014), 414–426.
[55] Tara Safavi, Chandra Sripada, and Danai Koutra. 2017. Scalable hashing-based network discovery. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 405–414.
[56] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).
[57] Stephen M Smith, Thomas E Nichols, Diego Vidaurre, Anderson M Winkler, Timothy EJ Behrens, Matthew F Glasser, Kamil Ugurbil, Deanna M Barch, David C Van Essen, and Karla L Miller. 2015. A positive-negative mode of population covariation links brain connectivity, demographics and behavior. *Nature neuroscience* 18, 11 (2015), 1565–1567.
[58] Daniel A Spielman and Shang-Hua Teng. 2011. Spectral sparsification of graphs. *SIAM J. Comput.* 40, 4 (2011), 981–1025.
[59] Dardo Tomasi and Nora D Volkow. 2012. Resting functional connectivity of language networks: characterization and reproducibility. *Molecular psychiatry* 17, 8 (2012), 841–854.
[60] Kenneth I Vaden, Stefanie E Kuchinsky, Stephanie L Cute, Jayne B Ahlstrom, Judy R Dubno, and Mark A Eckert. 2013. The cingulo-opercular network provides word-recognition benefit. *Journal of Neuroscience* 33, 48 (2013), 18979–18986.
[61] David C Van Essen, Kamil Ugurbil, Edward Auerbach, Deanna Barch, Timothy EJ Behrens, Richard Bucholz, Acer Chang, Liyong Chen, Maurizio Corbetta, Sandra W Curtiss, et al. 2012. The Human Connectome Project: a data acquisition perspective. *Neuroimage* 62, 4 (2012), 2222–2231.
[62] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations (ICLR)* (2018). https://openreview.net/forum?id=rJXMpikCZ
[63] Elli Voudigari, Nikos Salamanos, Theodore Papageorgiou, and Emmanuel J Yannakoudakis. 2016. Rank degree: An efficient algorithm for graph sampling. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 120–129.
[64] Minh Vu and My T Thai. 2020. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems* 33 (2020), 12225–12235.
[65] Carl Wernicke. 1874. *Der aphasische Symptomencomplex: eine psychologische Studie auf anatomischer Basis*. Cohn & Weigert.
[66] Shaokai Wu and Fengyu Yang. 2023. Boosting Detection in Crowd Analysis via Underutilized Output Features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15609–15618.
[67] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? *International Conference on Learning Representations* (2018).
[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[69] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2021. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. *arXiv preprint arXiv:2102.06462* (2021).
[70] Yujun Yan, Jiong Zhu, Marlena Duda, Eric Solarz, Chandra Sripada, and Danai Koutra. 2019. Groupinn: Grouping-based interpretable neural network for classification of limited, noisy brain data. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 772–782.
[71] Fengyu Yang and Chenyan Ma. 2022. Sparse and Complete Latent Organization for Geospatial Semantic Segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 1799–1808.
[72] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019).
[73] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems* 31 (2018).
[74] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
[75] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*. PMLR, 12241–12252.
[76] Yue Zhang, David Defazio, and Arti Ramesh. 2021. Relex: A model-agnostic relational model explainer. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 1042–1049.
[77] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*. PMLR, 11458–11468.

A IGS WITH OTHER BACKBONE GNNS

In Table 4, we present the results of IGS evaluated on different GNN backbones (noted by "- IGS") and compare it against the original performance (noted by "Original Graphs"). Specifically, we consider three additional GNN models: GraphSAGE [25], GraphConv [47], and GIN [68]. The experimental and hyperparameter settings follow those in Section 4.1. Compared with the performance of the original graphs, the sparsified graphs obtained from IGS consistently contribute to performance gains across all GNN backbones and prediction tasks. It provides an average of 4.72% increase in the test accuracies for GraphSage, an average of 1.92% increase in the test accuracies for GraphConv, and an average of 1.45% increase in the test accuracies for GIN. This demonstrates that the improvements achieved by IGS are model-agnostic.

B ADDITIONAL STUDIES ON INTERPRETABILITY

In Figure 4, we provide the interpretability analysis for the ReadEng task, following the same setting as Figure 3. The "ReadEng" task involves the subjects reading aloud words presented on a screen. As can be seen in Figure 4, the IGS model effectively identifies the significance of interactions between the visual (Vis1) network and the Cingulo-Opercular (CO), Language (LA), and Dorsal Attention (DA) networks for this prediction task. Furthermore, it elucidates that the functional synchrony between the language-related networks (CO-LA, CO-AD) is accountable for this task.

Table 4: Performance of IGS with different GNN backbones, following the same setup in Section 4.1. The performance improvements achieved by IGS are model-agnostic.

	PicVocab	ReadEng	PicSeq	ListSort	CardSort	Flanker
GraphSage (Original Graphs)	56.2±5.47	49.6±2.37	48.1±4.92	50.6±0.63	50.3±1.75	49.0±2.04
GraphSage - IGS	60.9±4.27	56.4±2.27	55.1±3.52	52.6±1.66	54.4±11.8	52.7±2.40
GraphConv (Original Graphs)	53.2±6.70	54.9±5.06	48.2±1.19	49.4±0.63	50.6±2.93	49.4±2.54
GraphConv - IGS	57.1±8.21	55.9±3.41	52.3±1.93	50.7±2.19	50.8±7.70	50.4±9.37
GIN (Original Graphs)	55.8±5.42	56.4±6.94	49.9±3.53	52.6±2.84	55.0±3.00	48.5±3.83
GIN - IGS	59.3±5.83	56.7±5.54	51.0±3.28	54.1±5.70	55.0±6.47	50.8±4.80

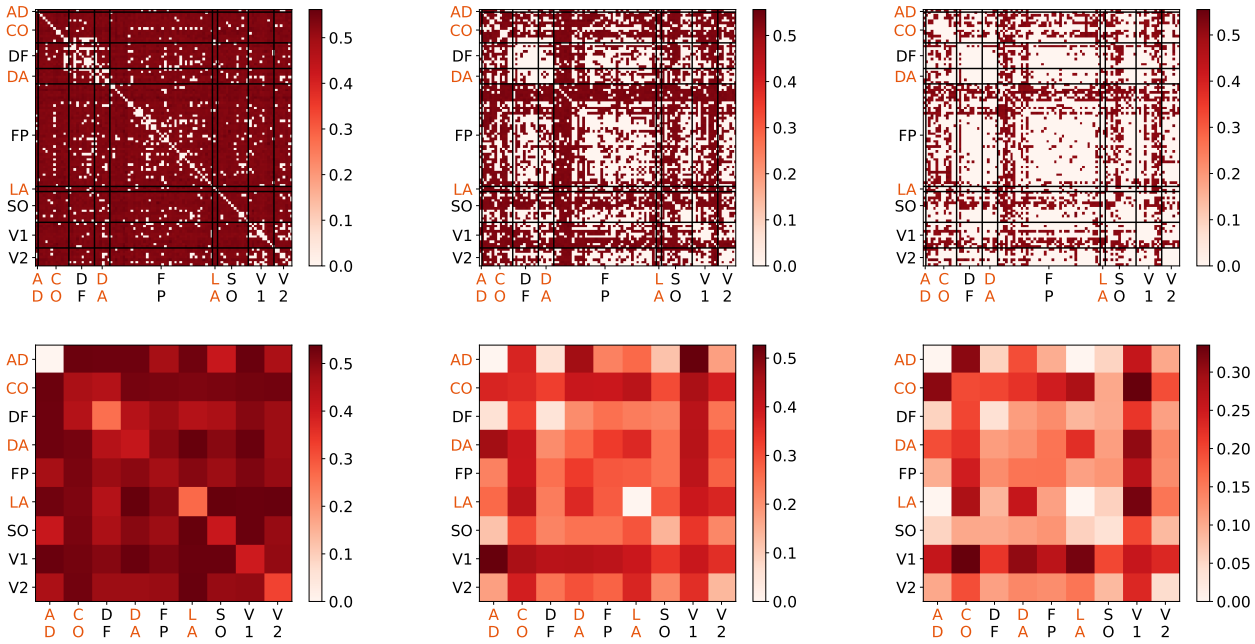

Figure 4: Weighted brain network edge masks at both node (top row) and subnetwork level (bottom row) for the ReadEng task, following the same setup in Section 4.3.