GroupINN: Grouping-based Interpretable Neural Network for Classification of Limited, Noisy Brain Data

Yujun Yan
University of Michigan
yujunyan@umich.edu

Jiong Zhu
University of Michigan
jiongzhu@umich.edu

Marlena Duda
University of Michigan
marlenad@umich.edu

Eric Solarz
University of Michigan
esolarz@umich.edu

Chandra Sripada
University of Michigan
sripada@umich.edu

Danai Koutra
University of Michigan
dkoutra@umich.edu

ABSTRACT

Mapping the human brain, or understanding how certain brain regions relate to specific aspects of cognition, has been and remains an active area of neuroscience research. Functional magnetic resonance imaging (fMRI) data—in the form of images, time series or graphs—are central in this research, but pose many challenges in phenotype prediction tasks (e.g., noisy, small training samples). Standardly employed handcrafted models and newly proposed neural network methods pose limitations in the expressive power and interpretability, respectively, in this context.

In this work focusing on fMRI-derived brain graphs, a modality that partially handles some challenges of fMRI data, we propose a grouping-based interpretable neural network model, GroupINN, that effectively classifies cognitive performance with 85% fewer model parameters than baseline deep models, while also identifying the most predictive brain subnetworks within several task-specific contexts. Our method incorporates the idea of node grouping into the design of the neural network. That way, unlike other methods that employ clustering as a preprocessing step to reorder nodes, GroupINN learns the node grouping and extracts graph features jointly. Experiments on task-based fMRI datasets show that our method is $2.6-69\times$ faster than other deep models, while achieving comparable or better accuracy and providing interpretability.

ACM Reference Format:
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 *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330921

1 INTRODUCTION

For decades, a main goal in the field of neuroscience has been to understand how specific aspects of cognition and intelligence are functionally encoded in the brain [9, 13, 24, 38]. Functional magnetic resonance imaging (fMRI), a non-invasive measure of neural activation, has been paramount in advancing our understanding of the functional organization of the brain [21]. Large strides have been made specifically in brain graphs or connectomes, which are obtained by computing pairwise correlations between fMRI time series of different regions of interest (ROIs) [27], as illustrated in Fig. 1. In this work, we propose a neural network-based framework for mapping regional and cross-regional functional activation patterns to cognitive phenotypes, formalized as a classification problem.

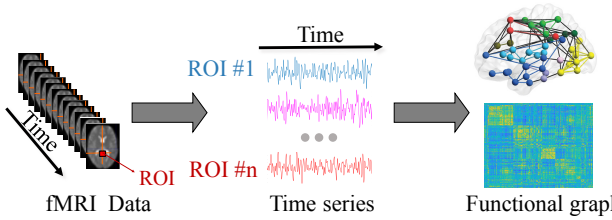


Figure 1: From fMRI images[1] to functional graphs [5]: first, fMRI images are processed via correction, denoising, smoothing and parcellation to obtain spatially-averaged time series per region of interest (ROI). Then, time series are further processed by computing Pearson's correlation and Fisher's r to z transformation to obtain functional graphs.

Our work is motivated by the special considerations needed for analyzing fMRI data and the methodological gaps in addressing these considerations to date. First, fMRI data are inherently high dimensional and can contain on the order of 10^6 activation values per subject. Second, fMRI data have a low signal-to-noise ratio caused by changing levels of non-neural noise derived from cardiac and respiratory processes or scanner instability. Third, most available datasets contain a relatively low number of samples when compared to the dimensionality of the data [22]. Classically in the neuroscience literature, linear models such as PCA, ICA and other matrix factorization methods are adopted for dimensionality reduction and denoising [1, 12, 31], but given the highly nonlinear nature of functional interactions [41], these models may not adequately capture the complex relationships between cortical regions. More recently, neural networks have been proposed for prediction tasks using fMRI data [28, 41], which can appropriately model nonlinearities but require many training samples and long training times for numerous parameters. Moreover, these neural network models suffer from the "black box" stigma, as it is difficult to interpret how the underlying brain activation patterns contribute to the final model prediction, and are therefore of limited use to the functional

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330921

[1] Retrieved from https://slideplayer.com/slide/9047889/

neuroscience community. To address the limitations of these prior works, we focus on the following problem:

Problem 1. ***Given*** *a set of subjects, each with corresponding fMRI data and a label associated with a certain phenotype, we seek to* ***devise*** *an* ***efficient***, ***interpretable***, *and* ***parsimonious*** *neural network model that can predict each phenotype with high accuracy.*

In this work, we rely on fMRI-based functional graphs (Fig. 1) to address some of the above-mentioned challenges. Instead of regressing phenotypes, which are often noisy [9], we cast Problem 1 as a classification problem, which is commonly done in the literature [41] [40]. Further considering the limitations of prior works, we propose GroupINN, a new neural network-based architecture which operates on these brain graphs and is efficient, uses fewer parameters, and provides interpretability. To reduce the number of parameters used in the model, we adopt the idea of multi-graph clustering (where the goal is to find a common clustering across multiple graphs) to summarize the original graph into a supergraph with each cluster as a supernode [23]. Unlike other work that performs grouping as a preprocessing step, we achieve the summarization by designing a *novel node grouping layer* to reduce dimensionality as part of our **end-to-end** neural network model. To extract features from the learned supergraph, we design a new variant of a graph convolutional layer that achieves higher accuracy and can be viewed as an extension of random walk with restart. The main contributions of our work are:

- **Novel, Fast & Parsimonious Model.** We propose a new end-to-end neural network-based formulation that can learn efficiently and effectively from little and noisy data by incorporating the idea of multi-graph clustering into the architecture design of the neural network.
- **Interpretability.** Beyond parameter reduction, the node grouping layer of GroupINN can explain relationships between brain subnetworks and cognitive functions. Some of our findings are supported in the literature and some may serve as starting points for further investigation in neuroscience [5–7].
- **Experiments on Real Data.** Extensive experiments on real data and comparisons to state-of-the-art approaches show that GroupINN needs only 0.01% parameters compared to CNN, 11% compared to Diffpool [44] and 15% compared to the GCN [14], while it is up to 69×, 3× and 2.6× faster, respectively. At the same time, it achieves better or comparable accuracy across a variety of tasks.

The code is available at: https://github.com/GemsLab/GroupINN.

The rest of the paper is organized as follows: Sec. 2 introduces the related work; Sec. 3 describes GroupINN, our proposed approach; in Sec. 4, we present our empirical analysis, in which we discuss the performance (runtime, classification accuracy), parsimony, and interpretability of our method; we summarize our work and point out future directions in Sec. 5.

2 RELATED WORK

Our work is related to brain graph analysis and graph classification. Table 1 presents a qualitative comparison of our approach to state-of-the-art techniques that are used for classification of functional brain graphs, or general graphs.

Brain Graph Analysis. Brain graphs are simple models of the real underlying connectome [32]. Recent studies have shown that both structural and functional brain graphs demonstrate common topological properties, such as modularity, small-worldness, and heterogeneous degree distributions [4, 17]. Modularity indicates a high level of neighborhood clustering [4, 25], which justifies our design for graph coarsening. The small-world organization of the brain suggests efficient local information processing, together with several long-distance connections responsible for global communication [3, 34, 38]. This further motivates our design choice to have a graph convolutional layer after coarsening; coarsening captures localized features, while the convolutional layer captures global communication.

Graph Classification. We discuss two approaches for classification: neural network- and kernel-based. In recent years, deep learning methods [29, 43] have often been used to tackle graph-based problems. Those methods aim to generalize the traditional convolutional neural networks (CNN) used in image classification. For example, [10] defines the neighborhoods in the graph spectral domain and uses Chebyshev polynomials as a basis to speed up the convolution computation. A variant [14] of this approach solves the vanishing/exploding gradient problem via a localized first-order approximation of the spectral graph convolutions. [45] proposes another variant for general graph classification, which is shown to have a close relationship to Weisfeiler-Lehman kernels. [44] introduces a framework that jointly learns the pooling and node embeddings. However, these deep models are hard to interpret and usually contain a large number of parameters which result in slow training and overfitting. More traditional approaches for graph classification include graph similarity approaches based on different network properties [16, 18] and graph kernels combined with kernelized classifiers (e.g. SVM). Kernel methods usually compute the similarity between two networks based on substructures, such as walks [39], shortest paths [2], graphlets [30], or other subgraphs [15]. [19] proposes a valid assignment kernel that achieves state-of-the-art graph classification accuracy. However, most graph kernels cannot be applied to signed and weighted graphs that brain graphs belong to, so they are inappropriate for brain graph classification.

Table 1: Qualitative comparison to related work.

	Fast	Parsimonious	Interpretable
CNN [41], GraphCNN [10]	✗	✗	✗
GCN [14], DGCNN [45]	✓	✗	✗
Diffpool [44]	✓	✗	inadequate
GroupINN	✓	✓	✓

3 PROPOSED ARCHITECTURE

In this section, we first analyze the challenges of fMRI data. Then we give an overview of our proposed architecture and the design details. We give the main symbols that are used throughout the paper, and their definitions in Table 2.

3.1 Challenges

Handling fMRI data poses a series of challenges, which we aim to address with the design of our approach, GroupINN.

3.1.1 Noisy fMRI time series. The typical pipeline for handling fMRI data involves transforming the imaging data into multi-dimensional time series. For one subject, each time series represents the average activation in one region of interest (ROI) over time. These time series exhibit patterns that differ vastly across subjects [37], and there is often no correspondence between timepoints in the series obtained by different fMRI sessions. To alleviate these issues, instead of operating on the fMRI time series, we represent each subject's data as a brain graph (Fig. 1), where physical ROIs are represented as nodes and functional connections (i.e., time series correlations) between them are represented as edges [4].

3.1.2 Noisy fMRI-based brain graphs. The brain graphs or correlation matrices obtained from different sessions for the same subject vary significantly (even in the same day), exhibiting less than 0.7 correlation. To reduce the variability, we propose a neural network-based approach that performs classification based on coarsened brain graphs instead of the (noisier) original graphs.

3.1.3 Small samples of high-dimensional data. Existing datasets consist of a few hundred subjects, each of whom is represented by a brain graph (correlation matrix) with entries in the order of 10^4 or even 10^6 (assuming a few hundred or thousand ROIs / nodes [11]). Under these conditions, neural network models face the challenge of overfitting. To overcome this issue, we turn to dimensionality reduction techniques. But instead of using unsupervised techniques (e.g., PCA, autoencoders) which are decoupled from the end goal (e.g., classification), our proposed method supervises the dimensionality reduction process, and provides an end-to-end model.

3.1.4 Need for interpretability. In neuroscience, ability to explain the results of a model is important for driving scientific discoveries, such as the relation between brain activation and cognition. Towards that end, we bring the idea of graph clustering and random walk on graphs into the design of the neural network: our design allows us to inspect which connections between ROIs are indicative to a specific phenotype, which may help answer important questions in neuroscience.

3.2 Architecture Overview

Our neural network architecture is illustrated in Fig. 2a. It is formed by three different types of layers: node grouping layer, graph convolutional layer and fully connected layer.

A **node grouping layer** is used for dimension reduction. In this layer, the original graph $\mathcal{G}$ will be summarized into a supergraph $\mathcal{G}^s$ where each supernode is a group of nodes in $\mathcal{G}$ [23]. The output $\mathbf{W}^s$ of this layer can be viewed as the weighted adjacency matrix of the supergraph.

Graph convolutional layers are used to probe the graph structure. There are many variations [10, 14, 45] of how a graph convolutional layer is designed. In this work, we design a new variation so that it not only has better performance, but also can be explained from the perspective of random walk with restart. Finally, the **fully connected layer** is used for label prediction.

Different from prior graph classification works, we use two branches in our architecture. This is because the correlation graph is essentially a weighted signed network. We use one branch to process the graph with only positive edges ($\mathcal{G}_p$) and another branch to process the negative edges ($\mathcal{G}_n$). In brain graphs, positive and negative edges have different functional meanings. Without separating them, the positive and negative edges will offset each other during the group aggregation and graph convolution, resulting in a drop in prediction accuracy (we elaborate more in § 4.6).

Table 2: Major symbols and their definitions.

Symbol	Definition
$\mathcal{G}$=($\mathcal{V}$, $\mathcal{E}$)	graph $\mathcal{G}$ with node set $\mathcal{V}$, edge set $\mathcal{E}$
$\lvert\mathcal{V}\rvert = n$	number of nodes in $\mathcal{G}$
$\mathbf{W}$	weighted adjacency matrix of $\mathcal{G}$, with elements w_{ij}
$\mathcal{G}^s$=($\mathcal{V}^s$, $\mathcal{E}^s$)	supergraph with supernode set $\mathcal{V}^s$, superedge set $\mathcal{E}^s$
SN_i	supernode i in supergraph $\mathcal{G}^s$
$\mathbf{W}^s$	weighted adjacency matrix of $\mathcal{G}^s$, with elements w^s_{ij}
s_i	importance score of node i learned in the node grouping layer
k	number of groups
l_i	label of ith subject
$+/-$	superscripts that denote the positive/negative branch
$\mathbf{\Lambda}$	characteristic matrix of graph $\mathcal{G}$
$\mathbf{P}$	the common factor in matrix factorization
$\mathbf{F}$	nonnegative matrix with the node importance scores s_i per group
$\mathbf{Q}$	weights that need to be learned in neural networks
$\mathbf{Y}_i$	output of i^{th} layer
$\mathbf{L}$	losses
$c(\cdot)$	$c : \mathcal{V} \rightarrow \mathbb{N}^+$, function mapping the node to its group
$\mathcal{R}_i$	functional subnetwork (e.g., dorsal attention) with $\lvert\mathcal{R}_i\rvert$ nodes
$S_{\mathcal{R}}$	importance score of functional subnetwork $\mathcal{R}$
$S^c_{\mathcal{R}_i,\mathcal{R}_j}$	cross-subnetwork importance score between $\mathcal{R}_i$, $\mathcal{R}_j$

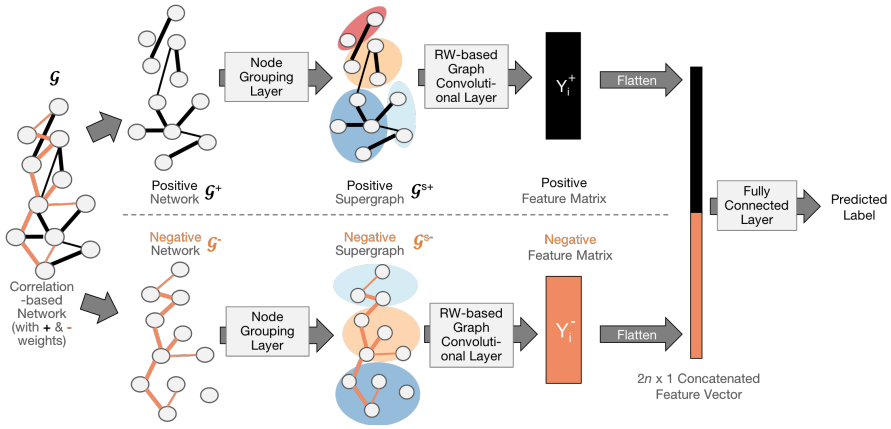
3.3 GroupINN Architecture

3.3.1 Node Grouping Layer. **Intuition.** As mentioned in Sec. 3.1.3 a layer for dimensionality reduction is needed. Recent findings have shown that some ROIs are most related to cognition [5, 7], suggesting that some edges are more indicative of predicting cognitive performance. Therefore, the node grouping layer is designed to "hide" the non-indicative ('noisy') edges by grouping them into a cluster (supernode [23]), thus highlighting the indicative edges. Figure 2b shows an example of how grouping is done conceptually. We note that our node grouping is *different* from traditional clustering since it does *not* require that similar nodes are grouped together. Instead, two nodes are assigned to different groups if their connection is identified as important.

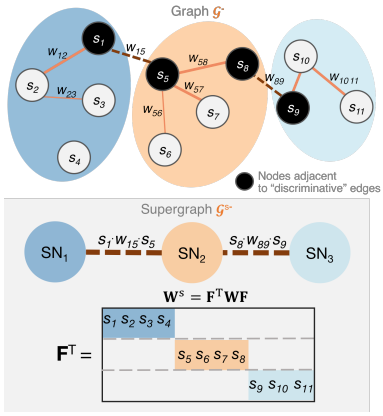
Design. Given a weighted graph $\mathcal{G}$—in which edge (i, j) is associated with a weight w_{ij}—, the nodes linked to the non-indicative edges are grouped together and form a supernode. Groups do not overlap. In Fig. 2b, there are three groups that form a supergraph with three supernodes. For node i in graph $\mathcal{G}$, an importance value $s_i \in R^+$ is assigned indicating how important node i is for the prediction task. In the supergraph, the weight of the superedge is computed as a weighted sum of cross-group edges. Formally, the weight of the superedge $w^s_{SN_1,SN_2}$ between supernode SN_1 and supernode SN_2 is defined as:

$$w^s_{SN_1,SN_2} = \sum_{t\in SN_1, k\in SN_2} s_t \cdot w_{t,k} \cdot s_k \tag{1}$$

To incorporate the grouping idea into the design of a neural network, we need to represent it in a matrix form. Suppose we are given an adjacency matrix $\mathbf{W}$ (size $n \times n$) for graph $\mathcal{G}$ and a nonnegative matrix $\mathbf{F}$ (size $n \times k$), where n is the number of



(a) Overview: the functional graph (correlation-based) is first split into positive and negative networks, each coarsened by node grouping layer, convoluted by random-walk-based graph convolutional layer, flattened, concatenated and finally sent to the fully connected layer.



(b) Example: In the node grouping layer indicative edges are learned to be placed across groups.

Figure 2: GroupINN Architecture

nodes and k is the number of desired groups. $\mathbf{F}_{ij}$ satisfies: $\mathbf{F}_{ij} = s_i$, iff $i \in SN_j$; $\mathbf{F}_{ij} = 0$, otherwise. Following this definition, $\mathbf{F}_{ij}$ can also be interpreted as the membership of the node i to the supernode SN_j. This is a more general definition, even suitable for overlapping clusters, and it requires $\mathbf{F}$ to be nonnegative. Note that there can be some nodes which *do not* belong to any groups. Then $\mathbf{F}^{\mathrm{T}}\mathbf{W}\mathbf{F}$ represents the weighted adjacency matrix of the supergraph after grouping the nodes. If $\mathbf{W}^s$ represents the output of the node grouping layer, then we have: $\mathbf{W}^s = \mathbf{F}^{\mathrm{T}}\mathbf{W}\mathbf{F}$. In Fig. 2b, we show the matrix representation for the node grouping. Notice here, we *do not* manually design the matrix $\mathbf{F}$; instead, the neural network learns $\mathbf{F}$ via back-propagation.

Connections to other work. As we show next, our formulation is related to the unsupervised multi-graph clustering problem [35].

THEOREM 3.1 (RELATION TO MULTI-GRAPH CLUSTERING). *Let $\mathbf{W}^{(m)}$ be the adjacency matrix of each input graph, $\mathbf{\Lambda}^{(m)}$ be the corresponding characteristic matrix (not necessarily diagonal) and $\mathbf{P}$ be the common factor to be learned. The grouping layer, defined by $\mathbf{F}^T\mathbf{W}^{(m)}\mathbf{F}$ with l_2 regularization on $\mathbf{F}$ and supervision on the characteristic matrices $\mathbf{\Lambda}^{(m)}$, learns the same clustering matrix as the multi-graph clustering problem [35] that minimizes the following objective:*

$$\mathcal{G} = \frac{1}{2}\sum_{m=1}^{M}||\mathbf{W}^{(m)} - \mathbf{P}\mathbf{\Lambda}^{(m)}\mathbf{P}^T||_F^2 + \frac{\alpha}{2}\left(\sum_{m=1}^{M}||\mathbf{\Lambda}^{(m)}||_F^2 + ||\mathbf{P}||_F^2\right) \quad (2)$$

where $\mathbf{P}$ is orthogonal (i.e., $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ and $\mathbf{I}$ is the identity matrix).

PROOF. Given that $\mathbf{P}$ is orthogonal, $Z = ||\mathbf{W}^{(m)} - \mathbf{P}\mathbf{\Lambda}^{(m)}\mathbf{P}^{\mathrm{T}}||_F^2 = ||\mathbf{P}^{\mathrm{T}}\mathbf{W}^{(m)}\mathbf{P} - \mathbf{\Lambda}^{(m)}||_F^2$ holds. For $\mathbf{F} = \mathbf{P}$, the factor $\mathbf{P}^{\mathrm{T}}\mathbf{W}^{(m)}\mathbf{P}$ is the output of the grouping layer and term Z can be viewed as the l_2 loss when supervised on $\Lambda^{(m)}$. The term $||\mathbf{P}||_F^2$ is the l_2 regularization on $\mathbf{P}$. In this case, the expressions of the total loss of the neural network and the multi-graph clustering are the same. □

The formulation of [35] is unsupervised so $\mathbf{P}$ will not change with different prediction goals. Matrix $\mathbf{P}$ can be viewed as the matrix learned by the grouping layer with supervision on characteristic matrices. In the context of other prediction goals, the grouping layer can be easily incorporated and supervised on those goals.

3.3.2 Graph Convolutional Layer. Graph convolutional layers are used to capture the structure of the supergraph. Moreover, they do not require pre-ordering of the nodes and they can perform nonlinear transformations in the neighborhood of spectral domain. Many variations of graph convolutional layers have been proposed to cater to different needs [10, 14, 45].

Intuition. As it is presented in the literature [20, 42], random walk is a useful tool to sample graph structures. The scores obtained from random walk with restart (RWR) can reveal a graph's structure by quantifying the similarities of other nodes to the selected nodes. Given the teleport vector $\mathbf{q}$ (a set of seed nodes), the RWR scores are given by $(1-c)(\mathbf{I} - c\widetilde{\mathbf{W}})^{-1}\mathbf{q}$, where $c < 1$ is a constant and $\widetilde{\mathbf{W}}$ is the column normalized adjacency matrix. Since the largest eigenvalue of $c\widetilde{\mathbf{W}}$ is smaller than one, Taylor expansion can be applied here.

$$(1-c)(\mathbf{I} - c\widetilde{\mathbf{W}})^{-1}\mathbf{q} = (1-c)(\mathbf{I} + c\widetilde{\mathbf{W}} + c^2\widetilde{\mathbf{W}}^2 + \ldots)\mathbf{q} \quad (3)$$

If the series is truncated to a finite sum of m terms, it represents the influence of the seed nodes spread over m-hop neighbors. To better capture the graph structure, different seed sets can be chosen to characterize the graph structure with different hops of neighborhoods. For the i-hop neighborhood, if m seed sets are selected to probe the graph structure, and the corresponding matrix is $\widetilde{\mathbf{Q}}_i$ (each column of $\widetilde{\mathbf{Q}}_i$ represents a seed set), the sketch matrix $\mathbf{M}$ of the graph $\mathcal{G}$ can be written as:

$$\mathbf{M} = (1-c)(\widetilde{\mathbf{Q}}_0 + c\widetilde{\mathbf{W}}\widetilde{\mathbf{Q}}_1 + c^2\widetilde{\mathbf{W}}^2\widetilde{\mathbf{Q}}_2 + \ldots) \quad (4)$$

Design. This expression can be modified to design a graph convolutional layer, which shares some similarities to the literature [10, 14, 45]. Ignoring the constants and using the reduced adjacency matrix $\mathbf{W}^s$ to replace $\widetilde{\mathbf{W}}$, the output $\mathbf{Y}_i$ of layer i can be computed as a nonlinear function of the previous layer output $\mathbf{Y}_{i-1}$:

$$\mathbf{Y}_i = \sigma(c\mathbf{W}^s\mathbf{Y}_{i-1}\mathbf{Q}_i + \mathbf{I}) \quad (5)$$

where σ is a nonlinear function. In this work, σ represents ReLU. If σ is the identity function, then the output after stacking i layers

is: $\mathbf{Y}_i = \mathbf{I} + c\mathbf{W}^s\mathbf{Q}_i + c^2(\mathbf{W}^s)^2\mathbf{Q}_i\mathbf{Q}_{i-1} + \ldots$, which is analogous to Eq. (4) truncated to i terms (where we omit the constants and replace $\mathbf{Q}_i\mathbf{Q}_{i-1}\ldots\mathbf{Q}_{i-j+1}$ with $\widetilde{\mathbf{Q}}_j$).

3.3.3 Constraints and Losses. We add several loss terms to regularize the learning process. Some are added due to the requirement for interpretability, while others are used to prevent overfitting. All of them improve the performance, as we show empirically in § 4.6.

For the node grouping layer, the matrix $\mathbf{F}$ is nonnegative because it represents the membership of a node to a group. However, in the neural network, the trainable matrices can have negative values. Thus, for the trainable real-valued matrix $\widetilde{\mathbf{F}}$, we represent as: $\mathbf{F} = \text{Relu}(\widetilde{\mathbf{F}})$, so the output of the node grouping layer becomes: $\mathbf{W}^s = \text{Relu}(\widetilde{\mathbf{F}}^{\mathrm{T}})\,\mathbf{W}\,\text{Relu}(\widetilde{\mathbf{F}})$ in the dimensionality reduction layer. This will create a problem because for negative values initialized in the matrix $\widetilde{\mathbf{F}}$, the gradients are 0 and they will not be updated later. To avoid this problem, we penalize the sum of negative values in matrix $\widetilde{\mathbf{F}}$, which can be expressed as: $\mathbf{L}_{\text{neg\_reduce}} = \text{sum}(\text{Relu}(-\widetilde{\mathbf{F}}))$. To prevent overlapping of groups, orthogonal constraints are added (on the nonnegative matrices) by penalizing the off-diagonal elements of $\mathbf{F}^{\mathrm{T}}\mathbf{F}$: $\mathbf{L}_{\text{otho}} = ||\mathbf{F}^{\mathrm{T}}\mathbf{F} - \text{diag}(\text{diag\_part}(\mathbf{F}^{\mathrm{T}}\mathbf{F}))||_{\mathrm{F}}$. Here, diag_part$(\cdot)$ extracts diagonal elements from a matrix and diag$(\cdot)$ builds a matrix with the given diagonal elements. Furthermore, to balance the group sizes (which helps with interpretability), we introduce the balance loss: $\mathbf{L}_{\text{balance}} = \text{Var}(\text{diag\_part}(\mathbf{F}^{\mathrm{T}}\mathbf{F}))$, where Var$(\cdot)$ means variance.

Similarly, since $\mathbf{Q}_i$ represents the selection of seed sets, positive values are more encouraged than the negative ones. A similar loss to penalize negative values is applied here: $\mathbf{L}_{\text{neg\_RWR}} = \text{Sum}(\text{Relu}(-\mathbf{Q}_i))$. At last, for the last dense layer with softmax as activation, $\mathbf{L}_2$ loss is used to reduce overfitting.

3.3.4 Putting everything together. All in all, the architecture (Fig. 2a) consists of three kinds of layers and two branches. One branch processes the positive graphs and the other processes the negative ones. The input graph is the correlation matrix $\mathbf{W}$. The first layer is a dimensionality reduction layer and the output is a matrix $\mathbf{W}^s$ representing the supergraph. For the positive branch (using "+" as superscript), we have $\mathbf{W}^{s+} = \text{Relu}(\mathbf{F}^{+\mathrm{T}})\mathbf{W}^{+}\text{Relu}(\mathbf{F}^{+})$. We have a similar expression for negative branch (using "−" as superscript). Following the dimensionality reduction layer, three graph convolutional layers are used

$$\mathbf{Y}_0^+ = \mathbf{I} \quad \text{and} \quad \mathbf{Y}_i^+ = \sigma(c\mathbf{W}^{s+}\mathbf{Y}_{i-1}^+\mathbf{Q}_i^+ + \mathbf{I}), \quad i = 1, 2, 3 \tag{6}$$

$$\mathbf{Y}_0^- = \mathbf{I} \quad \text{and} \quad \mathbf{Y}_i^- = \sigma(c\mathbf{W}^{s-}\mathbf{Y}_{i-1}^-\mathbf{Q}_i^- + \mathbf{I}), \quad i = 1, 2, 3 \tag{7}$$

At last, $\mathbf{Y}_3^+$ and $\mathbf{Y}_3^-$ are concatenated, flattened and sent to the fully connected layer (with softmax activation). The total loss is expressed as follows:

$$\mathbf{L}_{\text{total}} = \mathbf{L}_{\text{cross\_entropy}} + \sum_{i=+,-}\left(\lambda_{\text{otho}}^{(i)}\mathbf{L}_{\text{otho}}^{(i)} + \lambda_{\text{balance}}^{(i)}\mathbf{L}_{\text{balance}}^{(i)}\right) + \lambda_{\text{neg}}\left(\mathbf{L}_{\text{neg\_RWR}} + \mathbf{L}_{\text{neg\_reduce}}\right) + \lambda_{\text{dense}}\mathbf{L}_2 \tag{8}$$

4 EXPERIMENTS

Through our empirical analysis we aim to answer five key questions:

- **Q1** How well does GroupINN perform in terms of accuracy and training time when compared to neural network-based models in brain graph classification tasks?
- **Q2** How does GroupINN compare to the accuracy of non-neural-network-based models in brain graph classification tasks?
- **Q3** Is GroupINN parsimonious? How many fewer parameters does it require compared with other neural network-based methods?
- **Q4** Are the results of GroupINN interpretable? How can it be used to gain insights into the data?
- **Q5** How does positive and negative network splitting help in terms of accuracy? How do different regularization terms affect classification results?

First, we describe the data we used in our experiments, the baseline methods, the metrics used for evaluation, and the experimental setup. Then, we present our experimental results to answer our five key questions.

4.1 Experimental Setup

4.1.1 Data. In our experiments, we use four labeled datasets from Human Connectome Project 1200 release (`HCPt`) [36] to evaluate our proposed framework and compare it to baseline approaches. A total of 966 subjects had brain activity measured in fMRI session while performing specific tasks designed to probe different aspects of cognition. Voxel-level time series were spatially averaged according to the parcellation in [26], resulting in 264 distinct ROIs with a time series for each. The length of each time series depends on the task being performed, ranging from 176 to 405 time points. The four task-based datasets used in our experiments are: `Emotion`, `Gambling`, `Social`, and `Working Memory`. Per subject we also have a score called General Executive Factor (GenExec), a measure of general intellectual ability that we use as the label to predict (see Section 4.1.3).

Following the steps in Fig. 1 and Sec. 1, we transform the fMRI time series data into functional graphs. Since each subject has two trials per task, we follow a similar approach as in [33] to generate a single connectome by averaging the correlation matrices generated from the time series of each trial.

4.1.2 Baselines. Since Problem 1 is drawn from neuroscience, fMRI data can be in the form of time series or functional graphs, and we provide a neural network-based solution, we compare GroupINN to representative methods from each research area:

(1) **Flattened Correlation Matrix (FCM).** In this approach, the full correlation matrix is computed for all time series and its flattened upper triangular matrix is taken as a feature vector. SVM with radial basis function kernel is applied for classification.
(2) **Flattened Partial Correlation Matrix (FPCM).** This approach is the same as above, but rather than full correlation we compute *partial* correlation to regress out covariates.
(3) **PCA.** PCA with 100 components [31] is performed on the stacked flattened correlation matrices for all subjects. SVM is then used for classification.
(4) **Autoregression (AR).** In this approach, a k-order auto-regression (AR) model is used to fit the time series of each ROI of every subject. The corresponding $k + 1$ parameters of each ROI are stacked to form a parameter matrix, and the flattened parameter matrix is adopted as the feature vector of the subject. Here we select $k = 5$ to ensure that the correlation values can fall into the 99% confidence interval. The feature vectors for each subject

are subjected to dimensionality reduction using PCA with 100 components before they are sent into the classifier.

(5) **Weisfeiler-Lehman Optimal Assignment Kernel (WL_OA).** To apply WL_OA kernel [19] for classification, we first obtain unweighted functional brain networks by thresholding the correlations at 0.6 (based on the performance of validation set).
(6) **CNN.** The architecture of our Convolutional Neural Network models includes convolution layers (C), pooling layers (P) and a fully connected layer (F); the activation function is ReLU. The configuration of each layer follows the setting in [41]. We implement two CNN models as baselines: **(6a) CNN_1** (1 convolution layer, 87122K parameters): C-P-F-Softmax; and **(6b) CNN_2** (2 convolution layers, 21844K parameters): C-P-C-P-F-Softmax.
(7) **GCN**. We concatenate the node features obtained from the GCN [14] model and fed to a fully connected layer.
(8) **Diffpool.** For this hierarchical graph embedding approach [44], we set the parameters based on the authors' guidelines.

Recently more NN-based methods that apply to brain data have been proposed [41] [40]. However, the code of these models is not publicly available, and they require significant engineering which is hard to reproduce fairly based solely on the papers.

4.1.3 Settings. The same 90% training/validation and 10% testing split is used for all the experiments. The siblings in the HCP data are assigned either to the training or the testing set (but not both).

The goal is to classify subjects with high or low executive function using the GenExec score. GenExec is based on overall accuracy for three tasks: n-back working memory task, relational processing task, and Penn Progressive Matrices task. We cast the problem as a binary classification task by assigning the top quartile of subjects to the 'positive' class, and the bottom quartile to the 'negative' class.

For neural network based methods, we further split a subset (10%) from the training data as the validation set to select proper hyper-parameters for evaluation. We train the model for 300 epochs. After each epoch, we evaluate the model using the validation set, and keep the model with the highest F1 score so far, while also satisfying the restrictions that the difference between true positive rate and true negative rate is smaller than 5%, requiring the classifier to be unbiased. For the same method, we run three independent training sessions, and report the average accuracy of the three trained models on the test set and the total CPU time spent in the training process. To ensure the comparability of the performance of different methods, the same training, validation and testing split is used in all experiments. For baseline experiments using SVM, five-fold cross validation is used with grid search to determine best hyperparameters.

4.1.4 Evaluation Metrics. To evaluate the prediction performance, following [41], our evaluation metrics are classification accuracy and runtime (total CPU time, including both user and system CPU time). In addition, we use F1 score of the validation set to select our models. To show that GroupINN is parsimonious, model sizes are given for comparisons. To evaluate the interpretability of GroupINN, we define the importance scores of inner- and across- subnetworks to rank the most relevant subnetworks to GenExec scores. We further compare the relevant regions found by different baselines, and verify the quality based on the support from the literature.

4.2 Q1: Performance Comparisons with Neural network-based Models

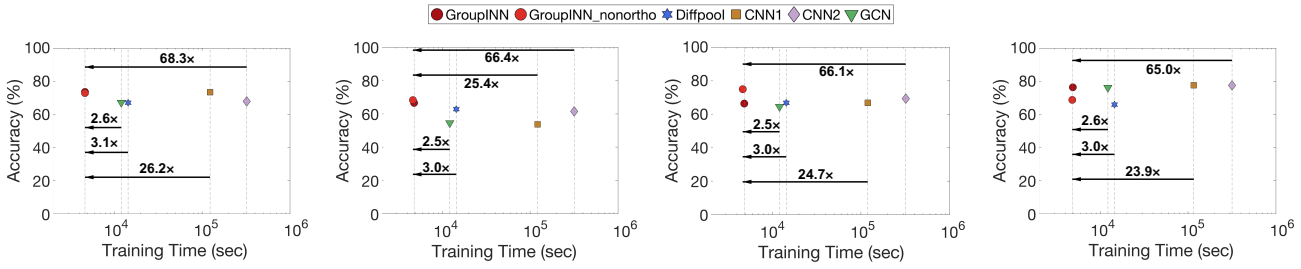
In this experiment, we compare with different neural network-based methods via accuracy and training time. Two variants of our model are used: one is GroupINN, the other is GroupINN without the orthogonality constraint. As we illustrated in Section 3.3.3, the orthogonality constraint controls how much overlap is allowed between the groups found. Figure 3 shows the comparisons of prediction accuracy vs training time over four different datasets: `Emotion`, `Gambling`, `Social` and `Working Memory`. Speedup of GroupINN over the baseline methods is also annotated in the plots. Ideally, the best method lies in the top left corner of the plots.

In general, our methods (including two variants) achieve the best accuracy in three out of four datasets, namely, `Emotion`, `Gambling` and `Social`. In the dataset `Working Memory`, the difference of accuracy between GroupINN and the best method CNN_1 is merely around 1%. Notice that not all tasks are cognitive intensive and closely related to GenExec score, which makes it harder to predict in less relevant tasks. The most related task is `Working Memory`, which is used to calculate the GenExec score. In this task, every method (except for Diffpool) performs equally well. However, in less relevant tasks, the advantages of our methods are revealed.

More importantly, our methods *always* take the least training time among all the baseline methods. We achieve around 2.5× speedup against GCN method, 3× speedup against Diffpool, 25× speedup against one-layer CNN and 66× speedup against two-layer CNN. This is due to our model having considerably fewer parameters (§ 4.4).

4.3 Q2: Performance Comparison with Non-neural-network-based Models

Figure 4 shows the comparison of prediction accuracy between our methods and non-neural-network-based methods. In all the datasets, either GroupINN or GroupINN without orthogonality performs the best. Methods that try to utilize the temporal patterns do not work well in general. AR models the temporal relationship of the time series and the prediction accuracy is 15%-20% lower than ours. In our experiments, we also tried other time-series-based methods, such as LSTM, 1D CNN, but none of them produce satisfactory results (less than 60% accuracy) that are close to graph-based methods. For the graph based methods, FPCM consistently performs the worst, and even in the most related task, `Working Memory`, its prediction accuracy is as low as 40%. This might result from the noisy temporal patterns which produce false partial correlations. On the other hand, a similar method, FCM, performs much better than FPCM, but still worse than ours: 1%, 7%, 3%, 2% lower accuracy in `Emotion`, `Gambling`, `Social` and `Working Memory` tasks, respectively. The state-of-the-art graph kernel, WL_OA, cannot apply on signed and weighted graphs. Thus, for this approach we first threshold the data to obtain unweighted graphs. The incurred information loss from this process results in WL_OA achieving consistently lower performance than GroupINN, PCA and FCM. PCA gives comparable results in `Social` and `Working Memory` tasks, but in the `Emotion` and `Gambling` tasks (where the related patterns are not obvious) our methods outperform PCA by 5%. In Section 4.5,



(a) Emotion (b) Gambling (c) Social (d) Working Memory

Figure 3: Training time (x axis) vs. accuracy (y axis) for GroupINN and other neural network-based models, with the speedup of GroupINN annotated over the arrows. The most ideal points lie on the top left corner, corresponding to low runtime and high accuracy. Our proposed GroupINN models are up to 69× faster at training than all the baseline methods, while achieving same or higher accuracy in a variety of prediction tasks.

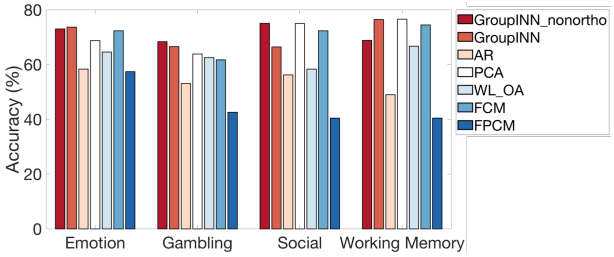


Figure 4: Accuracy compared with non-neural-network-based models. Our models have better or comparable accuracy across all data.

Table 3: Number of parameters used in neural network based methods. GroupINN can use 15% or much less model parameters to achieve comparable or better performance of the baseline methods.

Methods	# parameters	Normalized wrt GroupINN
CNN-1	87,121,602	30, 125.04×
CNN-2	21,844,152	7, 553.30×
GCN	19,874	6.87×
Diffpool	26,678	9.22×
GroupINN	2,892	1×

we will see that even in the less related tasks, our model can still pick out the task positive functional subnetworks.

All in all, if the data is directly related to the phenotype predicted (`Working Memory`), the prediction accuracy of graph-based methods, like PCA and FCM, is equal or lower than ours by a small margin (around 1-2%). However, if the data is more complex and not directly related to the phenotype (`Emotion` and `Gambling`), our methods outperform them by more than 5%. This is understandable, because neural networks are known for modeling complex relations.

4.4 Q3: Parsimony of GroupINN

So far we have seen that our models have better or comparable accuracy to the baseline methods. In addition to that, compared to neural network methods, GroupINN achieves significantly faster training time, only 1.33 hours total CPU time on average for all datasets. The speedup comes from the small number of model parameters. Table 3 shows the parameters used in each neural network-based model (first column) and we also show the ratio compared to ours. We can see that even the fastest baseline GCN uses about 7× more parameters than our methods. This explains why other methods need at least 2.5× more training time than GroupINN. In most task based cases, GCN and Diffpool do not perform as well as our GroupINN. CNN sometimes is comparable, but uses many more parameters than ours (up to 30,125× more than GroupINN), so it takes more than 65 times longer to train than our methods. These observations indicate that GroupINN effectively captures the main characteristics of the data.

4.5 Q4: Interpretability of GroupINN

Our method is interpretable and can reveal the subnetworks that are most informative to the prediction goals. In this section, we use 14 expertly-defined functional subnetworks and analyze them along with the learned matrices **F**. We show how our method can provide insights on the relation.

We use the learned matrices **F** obtained from each task. Though the data is acquired when people are performing different tasks, the prediction goal is the same: to predict the GenExec levels.

As illustrated in Fig. 2b, the **cross** edges are weighted and summed up to form a superedge. For the edge with weight $w_{i,j}$, it is multiplied by a factor $s_i s_j$. In this sense, $s_i s_j$ can be viewed as an amplification factor of the edge (i, j). Given a subnetwork, we compute the average amplification factor of an edge within it. Mathematically, for a given subnetwork (a node set $\mathcal{R}$) and a group mapping function (maps the node to the groups found in **F**), $c : \mathbb{N} \to \mathbb{N}$, we assign an importance score S to each subnetwork as:

$$S_{\mathcal{R}} = \frac{2}{|\mathcal{R}|^2} \sum_{i,j \in \mathcal{R} \text{ and } c(i) \neq c(j)} s_i s_j \tag{9}$$

We then rank the subnetworks based on S score. Also, to explore the important cross-subnetwork connections, we define a score, S^c, between any functional subnetworks $\mathcal{R}_1$ and $\mathcal{R}_2$:

$$S^c_{\mathcal{R}_1,\mathcal{R}_2} = \frac{1}{|\mathcal{R}_1||\mathcal{R}_2|} \sum_{(i \in \mathcal{R}_1, j \in \mathcal{R}_2 \text{ or } i \in \mathcal{R}_2, j \in \mathcal{R}_1), c(i) \neq c(j)} s_i s_j \tag{10}$$

We rank the combination of subnetworks based on the S^c score.

We list the three most important subnetworks (with highest S score) and cross-subnetwork combinations (with the highest S^c score) in Table 4. We provide their visualizations in supplementary material C. The results are based on **F** in the positive branch because positive links are more related. For comparison, we also list the important regions found by two baseline methods: For PCA, $S_{\mathcal{R}}$ is defined as the average node weight of region $\mathcal{R}$ in the first principal component. For Diffpool, we compute $S_{\mathcal{R}}$ via Eq. (9) for each subject separately because the learned clustering differs per graph. We find

Table 4: Tasks and the corresponding most important subnetworks and cross-subnetwork combinations (ordered by importance). GroupINN is interpretable as it can find meaningful task-positive subnetworks (shown in black font, left part of the table). On the other hand, PCA and Diffpool fail to discover them and rank as important other subnetworks (in gray font). Some of our findings about the cross-subnetwork interactions (right part of the table) are supported by the literature and may be worth further investigation.
Acronyms of brain subnetworks.* **AN: auditory; CBLN: cerebellar; CON: cingulo-opercular; DAN: dorsal attention; FPN: frontoparietal; MRN: memory retrieval; SN: salience ; VAN: ventral attention ; VN: vision; SM.M: sensory/somatomotor mouth; SM.H: sensory/somatomotor hand.

Tasks	Within subnetworks									Across subnetworks		
	GroupINN			PCA			Diffpool			GroupINN		
Working Memory	MRN	FPN	SN	SM.M	SM.H	AN	SM.M	MRN	CBLN	(MRN, FPN)	(MRN, VN)	(MRN, SN)
Gambling	VAN	VN	DAN	SM.H	AN	SM.M	SM.M	FPN	MRN	(MRN, FPN)	(MRN, VN)	(SM.M, VN)
Emotion	SN	CON	VAN	SM.M	SM.H	AN	DMN	MRN	SM.M	(MRN, SN)	(VAN, SN)	(VAN, FPN)
Social	FPN	SN	VAN	SM.M	CBLN	AN	DAN	SM.M	FPN	(MRN, VN)	(VAN, VN)	(SM.M, SN)

the most important regions by ranking their average scores across all the subjects.

Focusing on the within-network connections found by GroupINN, salience (SN) and ventral attention (VAN) are seen in three out of four tasks, fronto-parietal task control (FPN) is seen in two out of four, cingulo-opercular task control (CON) and dorsal attention (DAN) are also present. SN, VAN, FPN, CON, and DAN comprise major elements of the so-called task-positive network that are active whenever a person performs a cognitively demanding task [7, 13]. It is not surprising that these networks are predictive of GenExec, which represents ability to perform cognitively demanding tasks. GroupINN successfully discovers all of them. On the other hand, PCA fails to discover any of the task-positive networks. Instead, PCA 'picks' noisy but strong signals that are related to motion, vision and hearing. Similarly, Diffpool only finds two related regions. The results suggest that GroupINN is interpretable and can pinpoint the regions that relate to cognition during various tasks more accurately than baseline approaches.

We further investigate the cross connections found by our method. The cross connections between (memory retrieval ↔ FPN) and (memory retrieval ↔ salience) are found to be highly predictive in two of the four tasks. Many studies of task-specific functional activation have also shown strong integration of memory subnetworks with other subnetworks in cognitively demanding tasks [5, 6, 8], such as those represented by the GenExec measure. Our findings suggest that the above-listed task-specific network integrations may warrant further study in the context of executive function.

4.6 Q5: Impact of network splitting and regularization terms

In this section, we explore the importance of the various design choices for GroupINN. We first compare our full method with a variant (`wo_+-_split`) in which the positive and negative networks are not split, but are handled as a unified network. Figure 5 shows the comparison. We observe that in three out of four cases, splitting the network into positive and negative networks has higher accuracy than not splitting. Especially in the `Gambling` dataset, splitting achieves 12% higher accuracy.

Then, to show the impact of each regularization term, we perform experiments to compare the prediction accuracy when different regularization terms are not added. The different variants are: not adding orthogonality loss $\mathbf{L}_{\text{otho}}$ (`wo_orthogonality`), not adding group balance loss $\mathbf{L}_{\text{balance}}$ (`wo_balance`), not adding the nonnegativity loss which consists of $\mathbf{L}_{\text{neg\_reduce}}$ and $\mathbf{L}_{\text{neg\_RWR}}$

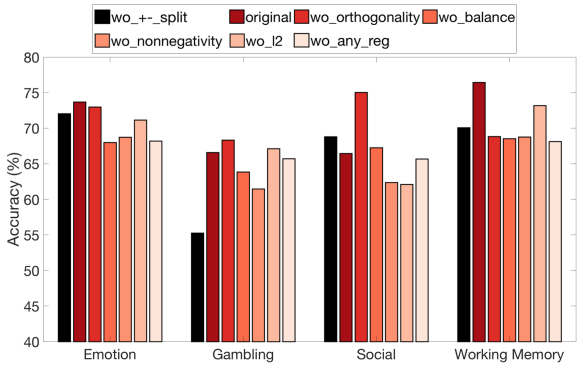


Figure 5: Impact of adding each regularization term. Splitting the network into positive and negative ones, adding variance, nonnegativity constraints and l2 regularization are effective in improving accuracy.

(`wo_nonnegativity`), not adding $\mathbf{L}_2$ loss (`wo_l2`) and not adding any regularization (`wo_any_reg`). From Fig. 5, we can see that adding balance and nonnegativity constraints are always helpful as GroupINN wins all the cases over the variant (`wo_balance`) or (`wo_nonnegativity`). This result is in line with our assumption about grouping, since membership score should not be negative and unbalanced group assignment should be avoided. Adding $\mathbf{L}_2$ regularization also helps, as the accuracy is improved in the `Social` and `Working Memory` datasets, but it does not improve much in the `Emotion` and `Gambling` datasets. The influence of adding orthogonality constraint is more complex: In the `Emotion` dataset, adding orthogonality constraint has a slight accuracy decrease (0.7%), while in the `Gambling` dataset a small accuracy boost (1.7%) is observed; In the `Working Memory` dataset there is a 7% accuracy drop, while in the `Social` dataset, there is a 8% accuracy increase. We posit that this complex behavior may be related to how brain regions interact with each other in different tasks—the more overlapping the data, the more ROIs are coordinating between different regions.

5 CONCLUSION

In this work, we introduce a novel neural network-based method, GroupINN, for mining fMRI data. FMRI data is characterized by large variations, both between subjects and within a single subject, thus exhibiting a significant level of noise. Furthermore, only a limited amount of data is accessible (e.g., few subjects) relative to the dimensionality of the data. These challenges require a model

with few parameters, as well as the capability of capturing non-linearities. Prior works using linear models cannot capture complex relationships between brain regions, while traditional neural network-based methods often require numerous parameters and lack interpretability. By introducing the idea of node grouping into the design of the neural network and designing a random-walk-based variant of graph convolutional layer, GroupINN requires up to ×69 times *less* training time than the baselines, achieves 85 – 99% reduction in parameters, and shows consistently better or comparable prediction accuracy. Moreover, our model can provide insights into brain subnetworks that are relevant to the prediction goal, providing interpretable results that are useful for neuroscientists. For future work, combining temporal features into the current architecture is an interesting direction.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, an Adobe Digital Experience research faculty award, and an Amazon faculty award. 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 author(s) and do not necessarily reflect the views of the National Science Foundation or other funding parties.

REFERENCES

[1] Ariana Anderson, Pamela K Douglas, Wesley T Kerr, Virginia S Haynes, Alan L Yuille, Jianwen Xie, Ying Nian Wu, Jesse A Brown, and Mark S Cohen. Non-negative matrix factorization of multimodal mri, fmri and phenotypic data reveals differential changes in default mode subnetworks in adhd. *NeuroImage*, 102:207–219, 2014.

[2] Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *ICDM*, pages 74–81. IEEE, 2005.

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

[4] Edward T Bullmore and Danielle S Bassett. Brain graphs: graphical models of the human brain connectome. *Annu Rev Clin Psychol*, 7:113–140, 2011.

[5] Jessica R Cohen and Mark D'Esposito. The segregation and integration of distinct brain networks and their relationship to cognition. *J Neurosci*, 36(48):12083–12094, 2016.

[6] Michael W Cole, Danielle S Bassett, Jonathan D Power, Todd S Braver, and Steven E Petersen. Intrinsic and task-evoked network architectures of the human brain. *Neuron*, 83(1):238–251, 2014.

[7] Michael W Cole and Walter Schneider. The cognitive control network: integrated cortical regions with dissociable functions. *Neuroimage*, 37(1):343–360, 2007.

[8] Elizabeth N Davison, Kimberly J Schlesinger, Danielle S Bassett, Mary-Ellen Lynall, Michael B Miller, Scott T Grafton, and Jean M Carlson. Brain network adaptability across task states. *PLoS comput biol*, 11(1):e1004029, 2015.

[9] Ian J Deary, Lars Penke, and Wendy Johnson. The neuroscience of human intelligence differences. *Nat rev neurosci*, 11(3):201, 2010.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

[11] Bruce Fischl, David H Salat, Evelina Busa, Marilyn Albert, Megan Dieterich, Christian Haselgrove, Andre Van Der Kouwe, Ron Killiany, David Kennedy, Shuna Klaveness, et al. Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. *Neuron*, 33(3):341–355, 2002.

[12] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[13] Rex E Jung and Richard J Haier. The parieto-frontal integration theory (p-fit) of intelligence: converging neuroimaging evidence. *Behavioral and Brain Sciences*, 30(2):135–154, 2007.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[15] Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. In *NeurIPS*, 2016.

[16] Danai Koutra and Christos Faloutsos. *Individual and Collective Graph Mining: Principles, Algorithms, and Applications*. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2017.

[17] Danai Koutra, Yu Gong, Sephira Ryman, Rex Jung, Joshua Vogelstein, and Christos Faloutsos. Are all brains wired equally. In *OHBM*, 2013.

[18] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *SDM*, pages 162–170. SIAM, 2013.

[19] Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. In *NeurIPS*, 2016.

[20] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636. ACM, 2006.

[21] Martin A Lindquist et al. The statistical analysis of fmri data. *Statistical science*, 23(4):439–464, 2008.

[22] Huan Liu, Mianzhi Zhang, Xintao Hu, Yudan Ren, Shu Zhang, Junwei Han, Lei Guo, and Tianming Liu. Fmri data classification based on hybrid temporal and spatial sparse representation. In *ISBI*, pages 957–960. IEEE, 2017.

[23] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *CSUR*, 51(3):62, 2018.

[24] Aljoscha C Neubauer, Andreas Fink, and Dietmar G Schrausser. Intelligence and neural efficiency: The influence of task content and sex on the brain–iq relationship. *Intelligence*, 30(6):515–536, 2002.

[25] Mark EJ Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.

[26] 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. Functional network organization of the human brain. *Neuron*, 72(4):665–678, 2011.

[27] Tara Safavi, Chandra Sripada, and Danai Koutra. Scalable hashing-based network discovery. In *ICDM*, pages 405–414. IEEE, 2017.

[28] Saman Sarraf and Ghassem Tofighi. Deep learning-based pipeline to recognize alzheimer's disease using fmri data. In *FTC*, pages 816–820. IEEE, 2016.

[29] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018.

[30] N. Shervashidze, S. V. N. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *AISTATS*, volume 5, pages 488–495. JMLR: W&CP, 2009.

[31] Stephen M Smith, Aapo Hyvärinen, Gaël Varoquaux, Karla L Miller, and Christian F Beckmann. Group-pca for very large fmri datasets. *Neuroimage*, 101:738–749, 2014.

[32] Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS computational biology*, 1(4):e42, 2005.

[33] Chandra Sripada, Mike Angstadt, Saige Rutherford, Daniel Kessler, Yura Kim, Mike Yee, and Liza Levina. Fundamental units of inter-individual variation in resting state connectomes. *bioRxiv*, page 326082, 2018.

[34] Cornelis J Stam and Jaap C Reijneveld. Graph theoretical analysis of complex networks in the brain. *Nonlinear biomedical physics*, 1(1):3, 2007.

[35] Wei Tang, Zhengdong Lu, and Inderjit S Dhillon. Clustering with multiple graphs. In *ICDM*, pages 1016–1021. IEEE, 2009.

[36] Kamil Ugurbil and David Van Essen. Hcp young adult. https://www.humanconnectome.org/study/hcp-young-adult, 2017.

[37] Heather L Urry, Carien M van Reekum, Tom Johnstone, and Richard J Davidson. Individual differences in some (but not all) medial prefrontal regions reflect cognitive demand while regulating unpleasant emotion. *Neuroimage*, 47(3):852–863, 2009.

[38] Martijn P van den Heuvel, Cornelis J Stam, René S Kahn, and Hilleke E Hulshoff Pol. Efficiency of functional brain networks and intellectual performance. *J Neurosci*, 29(23):7619–7624, 2009.

[39] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Imre Kondor, and Karsten M. Borgwardt. Graph kernels. *JMLR*, 11:1201–1242, 2010.

[40] Qi Wang, Mengying Sun, Liang Zhan, Paul Thompson, Shuiwang Ji, and Jiayu Zhou. Multi-modality disease modeling via collective deep matrix factorization. In *KDD*, pages 1155–1164. ACM, 2017.

[41] Shen Wang, Lifang He, Bokai Cao, Chun-Ta Lu, Philip S. Yu, and Ann B. Ragin. Structural deep brain network mining. In *KDD*, pages 475–484. ACM, 2018.

[42] Yujun Yan, Mark Heimann, Di Jin, and Danai Koutra. Fast flow-based random walk with restart in a multi-query setting. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 342–350. SIAM, 2018.

[43] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[44] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, pages 4800–4810, 2018.

[45] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

SUPPLEMENTARY MATERIAL ON REPRODUCIBILITY

A ENVIRONMENTS USED FOR THE EXPERIMENTS

The experimental platform is implemented in Python 3; all neural network models, including GroupINN, CNN and GCN, are implemented using TensorFlow 1.8; some test scripts are written in bash. Anaconda (version 5.1 or higher), TensorFlow (version 1.8 or higher), GNU time and other standard Unix-style terminal tools are utilized to run the experiments.

We use two machine in the experimental process. One machine is equipped with GPU to accelerate training and evaluation process for neural network-based methods, especially for CNN models, and it is used in all experiments except for the experiment of measuring the training time. The GPU should have at least 4 GB graphical memory to ensure that it can host all CNN parameters in the memory, and TensorFlow 1.8 with GPU support should be installed.

In experiment described in Section 4.2 about measuring and comparing the training time for different neural network-based methods, we test each methods on a machine without GPU. The reasons for this choice are two folds: on one hand, we would like to illustrate that GroupINN and GCN do not necessarily require GPU to accelerate, since it only uses 1.33 hours in average total CPU time for all datasets; on the other hand, when GPU acceleration is not available, all computations will be performed by CPU, which allows us to compare the computing resources needed for each method by comparing their total CPU time directly. The hardware configuration of the machine is shown in Table 5.

Table 5: Hardware configuration for the machine used in training time measurement.

HW Category	Specifications
CPU	2 × Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz
Memory	1024 GB
GPU	N/A

B DATA PREPARATION

As we have mentioned in Section 4.1, we use datasets from Human Connectome Project 1200 release (HCPt) [36] to evaluate our proposed framework and compare it to baseline approaches. We sort the subjects by the strength of General Executive Factor (GenExec) in descending order, and cast the problem as a binary classification task by assigning the top quartile of subjects to the 'positive' class, and the bottom quartile to the 'negative' class. The top and bottom quartile of subjects are the samples which are used in our experiments.

Within the top and the bottom quartile, we follow the training and testing split recommended in this dataset, since the recommended split has been chosen to avoid siblings between the training and testing set. However, since the number of 'positive' and 'negative' samples are not balanced in the training set following the recommended split, a stratified selection is made when selecting 10% validation set from the training data. Thus, the validation set will have the same ratio of positive to negative samples as the training set.

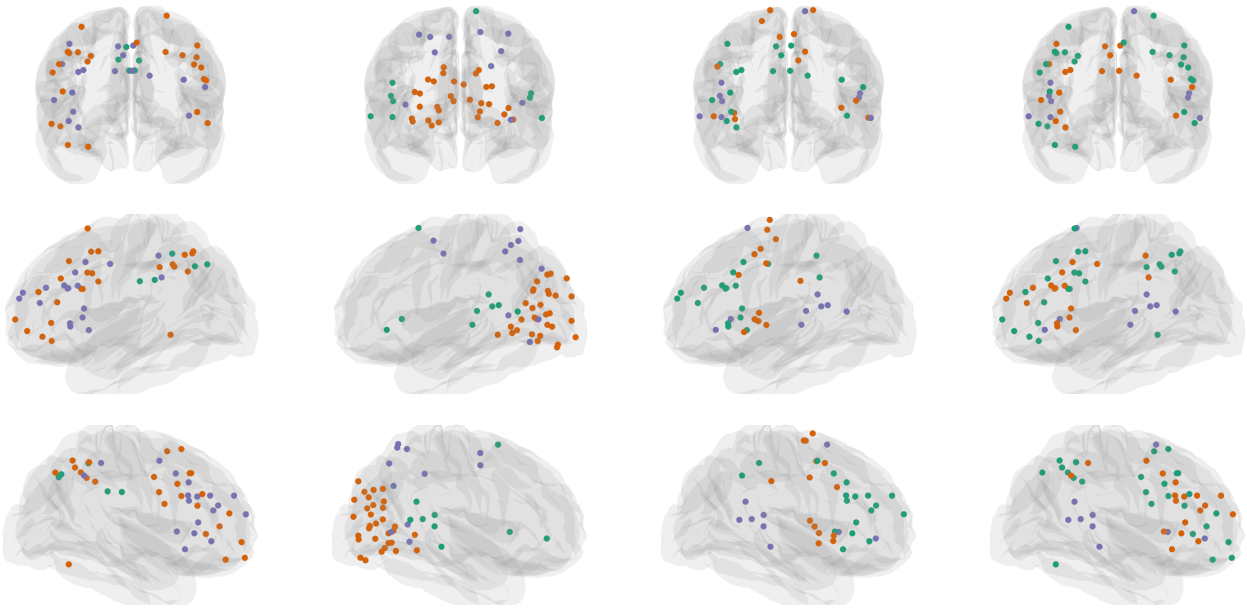
When generating each sample batch used for training, we randomize the permutation and balance the number of positive and negative samples of each batch to increase the convergence speed and robustness of the training process:

(1) Each epoch begins with a sample pool consisting all samples from the training set.
(2) Balancing the number of positive samples and negative samples in the sample pool: we randomly replicate some samples in the smaller category to make the size of the smaller category and the larger category equal in the pool.
(3) To form a batch of n samples with equal number of positive and negative samples, we randomly take $n/2$ positive samples and $n/2$ negative samples respectively out of the pool.
(4) Repeat the previous step, until the sample pool has less than n samples. In this case, take all remaining samples in the pool as the final batch of this epoch.

In our experiments we used batch size $n = 16$.

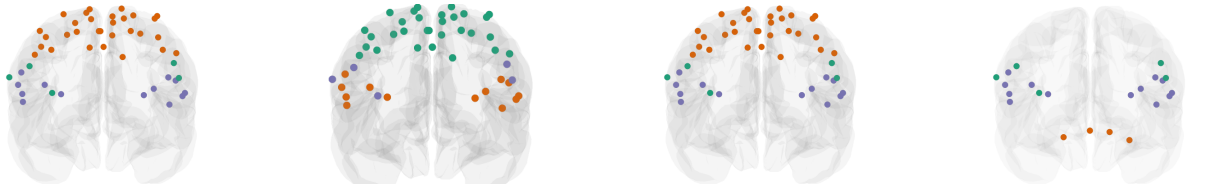
C VISUALIZATION OF IDENTIFIED SUBNETWORKS

In Section 4.5 we discussed the interpretability of GroupINN and contrasted it to baseline methods, such as PCA and Diffpool. Here we visualize the results of Table 4. Specifically, Fig. 6, 7, 8 show the top 3 important regions identified by GroupINN, PCA, and Diffpool, respectively. Green represents the 1st important region, orange represents the 2nd most important region, purple represents the 3rd most important region.



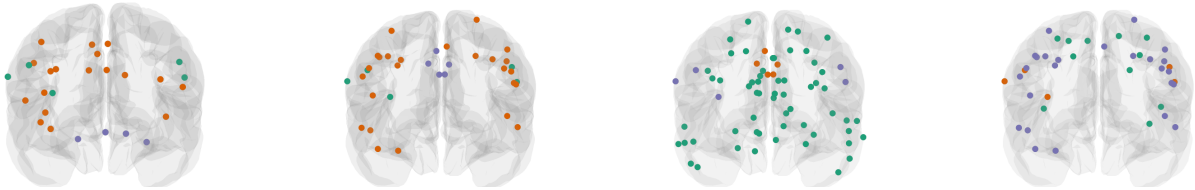
(a) `Working Memory`: MRN, FPN, SN (b) `Gambling`: VAN, VN, DAN (c) `Emotion`: SN, CON, VAN (d) `Social`: FPN, SN, VAN

Figure 6: Front, left and right views of the most important functional subnetworks identified by GroupINN during different tasks. Green color for the top-1 identified region; orange for the top-2 region; and purple for the top-3 region.



(a) `Working Memory`: SM.M, SM.H, AN (b) `Gambling`: SM.H, AN, SM.M (c) `Emotion`: SM.M, SM.H, AN (d) `Social`: SM.M, CBLN, AN

Figure 7: Front views of regions identified by PCA during different tasks. Green color for the top-1 identified region; orange for the top-2 region; and purple for the top-3 region.



(a) `Working Mem.`: SM.M, MRN, CBLN (b) `Gambling`: SM.M, FPN, MRN (c) `Emotion`: DMN, MRN, SM.M (d) `Social`: DAN, SM.M, FPN

Figure 8: Front views of regions identified by Diffpool during different tasks. Green color for the top-1 identified region; orange for the top-2 region; and purple for the top-3 region.