0429文献汇报：BrainGNN: Interpretable Brain Graph Neural Network for fMRI Analysis

> 文章提出了 `Ra-GConv` 卷积层和 `R-polling` 池化选择层两个关键模块，并根据任务要求做正则化调整，包括几类loss定义

1. Intro

- 常见fmri和图分析流程

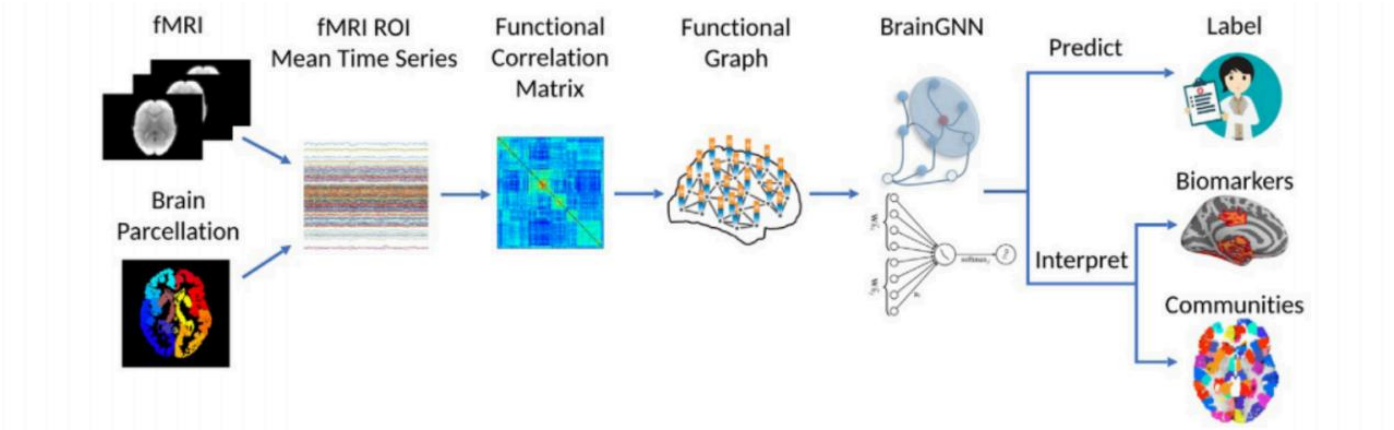


Fig. 1. The overview of the pipeline. fMRI images are parcellated by an atlas and transferred to graphs. Then, the graphs are sent to our proposed BrainGNN, which gives the prediction of specific tasks. Jointly, BrainGNN selects salient brain regions that are informative to the prediction task and clusters brain regions into prediction-related communities.

- 存在问题
  - 已有GNN常对不同节点使用相同嵌入规则，假设节点具有平移不变性

    2019; Gopinath et al., 2019; Nandakumar et al., 2019). Most existing GNNs are built on graphs that do not have a correspondence between the nodes of different instances, such as social networks and protein networks. These methods—including the current GNN methods for fMRI analysis—use the same embedding over different nodes, which implicitly assumes brain graphs are translation invariant and nodes on brain graphs (brain ROIs) are identical. However, nodes in the same brain graph have distinct locations and unique identities. Thus, applying the same embedding over all nodes is problematic. In addition, although recent stud-
  - 缺乏对个体和组水平的协同解释

    over all nodes is problematic. In addition, although recent studies have investigated group-level (Li et al., 2018; Venkataraman et al., 2016; Salman et al., 2019; Yan et al., 2019) and individual-level (Brennan et al., 2019; Mahowald and Fedorenko, 2016; Li et al., 2019) neurological biomarkers, few GNN studies have explored both individual-level and group-level explanations, which are critical in neuroimaging research.
- 框架
  - 在 图卷积层 中提出一种基于聚类的嵌入方法来解决将图节点（脑ROI）视为相同的局限性
  - 提出 图池化 和几个创新的 损失项 保留对组水平和个体水平生物标志物的解释性
  - 框架联合学习 `ROI聚类` 和 `全脑fMRI预测`

> 这项工作的初步版本在 *Pooling Regularized Graph Neural Network (PR-GNN) for fMRI Biomarker Analysis*

2. BrainGNN

notation

Notations used in the paper.

Notations	Description
C	number of classes
M	number of samples
N	number of ROIs
v_i	node i (ROI i) in the graph
$\mathcal{N}(i)$	neighborhood of v_i
e_{ij}	edge connecting node v_i and v_j
$\tilde{e}_{ij}$	normalized edge score over $j \in \mathcal{N}(i)$
$\mathcal{V}$	nodes set
$\mathcal{E}$	edge set
G	graph, $G = (\mathcal{V}, \mathcal{E})$
E	adjacency matrix, $E = [e_{ij}] \in \mathbb{R}^{N \times N}$
$d^{(l)}$	node feature dimension of the l^{th} layer
$\mathbf{h}_i$	node feature vector associated with v_i, $\mathbf{h}_i \in \mathbb{R}^d$
H	node feature matrix
$\tilde{\mathbf{h}}_i$	embedded node feature vector associated with v_i before pooling, $\tilde{\mathbf{h}}_i \in \mathbb{R}^d$
$\tilde{H}$	embedded node feature matrix before pooling
$\mathbf{s}_m$	node pooling score vector before normalization of subject m
$\tilde{\mathbf{s}}_m$	node pooling score vector after normalization of subject m
$\mathbf{r}_i$	one-hot encoding vector of v_i, $\mathbf{r}_i \in \mathbb{R}^N$, $\mathbf{r}_{i,j} = 0, \forall j \neq i$
k	number of nodes left after pooling
K	number of ROI communities
α_i	learnable membership score vector of v_i to each community, $\alpha_i \in \mathbb{R}^K$
β_u	learnable filter basis, $\boldsymbol{\beta}_u^{(l)} \in \mathbb{R}^{d^{(l+1)} \cdot d^{(l)}}$, $\forall u \in \{1, \ldots, K^{(l)}\}$
$W_i^{(l)}$	graph kernel for node v_i of the l^{th} layer, $W_i^{(l)} \in \mathbb{R}^{d^{(l+1)} \times d^{(l)}}$
λ	parameter associated with loss function

ROI-aware GConv

1. 允许Ra-GConv在**以ROI的几何信息（位置）为条件**的卷积核中学习不同的嵌入权重，权重W可以分解成一组基函数的线性组合，一个基函数代表一个社区

$$\tilde{\mathbf{h}}_i^{(l+1)} = \text{relu}\left(W_i^{(l)} \mathbf{h}_i^{(l)} + \sum_{j \in \mathcal{N}^{(l)}(i)} e_{ij}^{(l)} W_j^{(l)} \mathbf{h}_j^{(l)}\right), \quad (1)$$

2. 将区域信息编码嵌入到核函数，比如节点坐标。已知节点v_i的信息编码r_i，向量化嵌入核函数：

$$vec(W_i^{(l)}) = f_{MLP}^{(l)}(r_i) = \Theta_2^{(l)} relu(\Theta_1^{(l)} r_i) + b^{(l)}$$

其中，$\Theta_2^{(l)}$，$\Theta_1^{(l)}$是MLP的参数

- 使用one-hot编码ROI信息，不使用坐标。对节点v_i，r_i为N维，只在i^{th}为1，其他为0
- assume：

$$\Theta_1^{(l)} = (\alpha_1^{(l)}, ..., \alpha_{N^{(l)}}^{(l)}), \quad N^{(l)}\text{表示}l\text{层}ROI\text{数}$$
$$\alpha_i^{(l)} = (\alpha_{i1}^{(l)}), ..., \alpha_{ik}^{(l)} \in R^{K^{(l)}}, i \in \{1, ..., N^{(l)}\}, K^{(l)}\text{为}l\text{层社区数}$$
$$\Theta_2^{(l)} = (\beta_1^{(l)}, ..., \beta_{K^{(l)}}^{(l)}), \quad \beta_u^{(l)} \in R^{d^{(l+1)} d^{(l)}}, u \in \{1, .., K^{(l)}\}$$

（1）式重写为：$vec(W_i^{(l)}) = \sum_{k=1}^{K^{(l)}} (\alpha_{ik}^{(l)}) + \beta_u^{(l)} + b^{(l)}$，$(\alpha_{ik}^{(l)})^+$视为坐标或$ROI_i$对社区$u$的非负分配得分，$\beta_u^{(l)}$视为基

coding to represent the ROI's location information, instead of using coordinates, because the nodes in the brain are aligned well. Specifically, for node v_i, its ROI representation $\mathbf{r}_i$ is a N-dimensional vector with 1 in the i^{th} entry and 0 for the other entries. Assume that $\Theta_1^{(l)} = [\boldsymbol{\alpha}_1^{(l)}, \ldots, \boldsymbol{\alpha}_{N^{(l)}}^{(l)}]$, where $N^{(l)}$ is the number of ROIs in the l^{th} layer, $\boldsymbol{\alpha}_i^{(l)} = [\alpha_{i1}^{(l)}, \ldots, \alpha_{iK^{(l)}}^{(l)}]^\top \in \mathbb{R}^{K^{(l)}}$, $\forall i \in \{1, \ldots, N^{(l)}\}$, where $K^{(l)}$ can be seen as the number of clustered communities for the $N^{(l)}$ ROIs. Assume $\Theta_2^{(l)} = [\boldsymbol{\beta}_1^{(l)}, \ldots, \boldsymbol{\beta}_{K^{(l)}}^{(l)}]$ with $\boldsymbol{\beta}_u^{(l)} \in \mathbb{R}^{d^{(l+1)} \cdot d^{(l)}}$, $\forall u \in \{1, \ldots, K^{(l)}\}$. Then Eq. (2) can be rewritten as

$$\mathrm{vec}(W_i^{(l)}) = \sum_{u=1}^{K^{(l)}} (\alpha_{iu}^{(l)})^+ \boldsymbol{\beta}_u^{(l)} + \boldsymbol{b}^{(l)}. \tag{3}$$

We can view $\{\boldsymbol{\beta}_u^{(l)} : j = 1, \ldots, K^{(l)}\}$ as a basis and $(\alpha_{iu}^{(l)})^+$ as the coordinates. From another perspective, $(\alpha_{iu}^{(l)})^+$ can be seen as the non-negative assignment score of ROI i to community u. If we train different embedding kernels for different ROIs for the l^{th} layer, the total parameters to be learned will be $N^{(l)} d^{(l)} d^{(l+1)}$. Usually we have $K^{(l)} \ll N^{(l)}$. By Eq. (3), we can reduce the number of learnable parameters to $K^{(l)} d^{(l)} d^{(l+1)} + N^{(l)} K^{(l)}$ parameters, while still assigning a separate embedding kernel for each ROI. The ROIs in the same community will be embedded by the similar kernel so that nodes in different communities are embedded in different ways.

As the graph convolution operations in Gong and Cheng (2019), the node features will be multiplied by the edge weights, so that neighbors connected with stronger edges have a larger influence.

Node pooling layer:pruning the original graph G to G_s

> 基于节点特征在$W^{(l)} \in R^{d^{(l)}}$投影，只保留重要的ROI

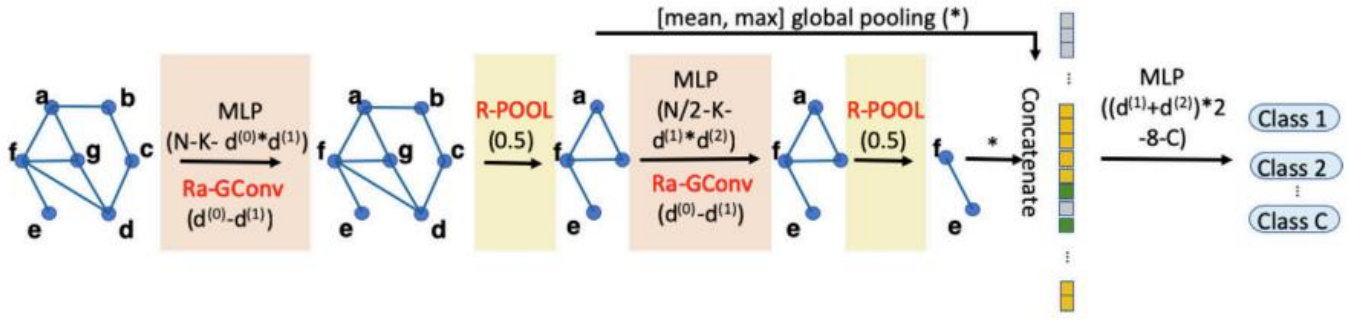
reduction step. Therefore the node (ROI) pooling layer (R-pool) is designed to keep the most indicative ROIs while removing *noisy* nodes, thereby reducing the dimensionality of the entire graph. *Design* To make sure that down-sampling layers behave idiomatically with respect to different graph sizes and structures, we adopt the approach in Cangea et al. (2018) and Gao and Ji (2019) for reducing graph nodes. The choice of which nodes to drop is determined based on projecting the node features onto a learnable vector $\mathbf{w}^{(l)} \in \mathbb{R}^{d^{(l)}}$. The nodes receiving lower scores will experience less feature retention. We denote $\tilde{H}^{(l+1)} = [\tilde{\mathbf{h}}_1^{(l+1)}, \ldots, \tilde{\mathbf{h}}_{N^{(l)}}^{(l+1)}]^\top$, where $N^{(l)}$ is the number of nodes at the l^{th} layer. Fully written out, the operation of this pooling layer (computing a pooled graph, $(\mathcal{V}^{(l+1)}, \mathcal{E}^{(l+1)})$, from an input graph, $(\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$), is expressed as follows:

$$\begin{aligned}
\mathbf{s}^{(l)} &= \tilde{H}^{(l+1)} \mathbf{w}^{(l)} / \|\mathbf{w}^{(l)}\|_2 \\
\tilde{\mathbf{s}}^{(l)} &= (\mathbf{s}^{(l)} - \mu(\mathbf{s}^{(l)}))/\sigma(\mathbf{s}^{(l)}) \\
\mathbf{i} &= \mathrm{top}k(\tilde{\mathbf{s}}^{(l)}, k) \\
H^{(l+1)} &= (\tilde{H}^{(l+1)} \odot \mathrm{sigmoid}(\tilde{\mathbf{s}}^{(l)}))_{\mathbf{i},:} \\
E^{(l+1)} &= E^{(l)}_{\mathbf{i},\mathbf{i}}.
\end{aligned} \tag{4}$$

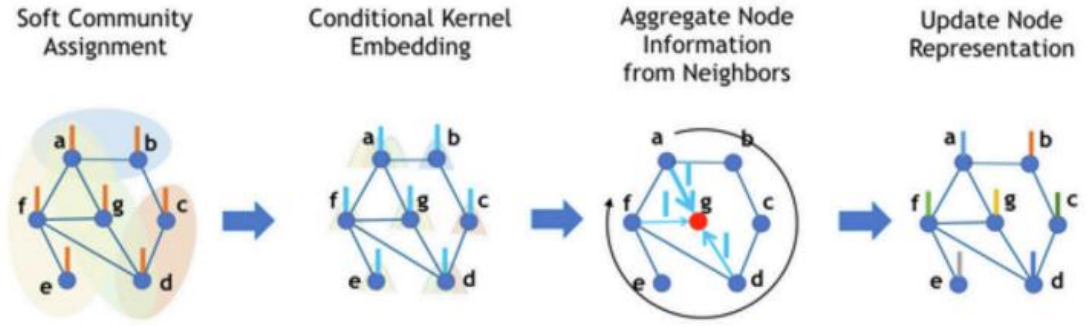
Here $\|\cdot\|$ is the L_2 norm, μ and σ take the input vector and output the mean and standard deviation of its elements. The notation top$k$ finds the indices corresponding to the largest k elements in score vector $\tilde{\mathbf{s}}$. $\odot$ is (broadcasted) element-wise multiplication, and $(\cdot)_{\mathbf{i},\mathbf{j}}$ is an indexing operation which takes elements at row indices specified by $\mathbf{i}$ and column indices specified by $\mathbf{j}$ (colon denotes all indices). The pooling operation retains sparsity by requiring only a projection, a point-wise multiplication and a slicing into the original features and adjacency matrix. Different

Readout layer:summarize $\{h_i^{(l)}\}$ into $\{z^{(l)}\}$ for graph classification

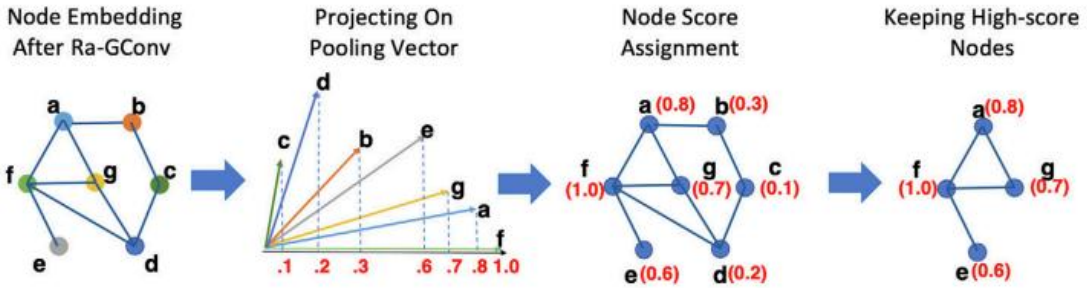
$$z^{(l)} = meanH^{(l)}concatmaxH^{(l)}$$
$$z = z^{(1)}concatz^{(2)}...concatz^{(l)}$$
$$z--MLP-->predictions$$



(a) BrainGNN Architecture



(b) Operations in Ra-GConv layer



(c) Operations in R-pool Layer

loss function

1. 分类交叉熵 L_{ce}

The classification loss is the cross entropy loss:

$$L_{ce} = -\frac{1}{M}\sum_{m=1}^{M}\sum_{c=1}^{C} y_{m,c}\log(\hat{y}_{m,c}), \quad (6)$$

where M is the number of instances, C is the number of classes, y_{mc} is the ground truth label and $\hat{y}_{mc}$ is the model output.

2. 单位损失，应对可识别性问题 L_{unit}

Now we describe the loss terms designed to regulate the learning process and control the interpretability. *Unit loss* As we mentioned in Section 2.3.2, we project the node representation to a learnable vector $\mathbf{w}^{(l)} \in \mathbb{R}^{d^{(l)}}$. The learnable vector $\mathbf{w}^{(l)}$ can be arbitrarily scaled while the pooling scores $\mathbf{s}^{(l)} = \tilde{H}^{(l+1)}(a\mathbf{w}^{(l)})/\|a\mathbf{w}^{(l)}\|$ remain the same with non-zero scalar $a \in \mathbb{R}$. This suggests an identifiability issue, i.e. multiple parameters generate the same distribution of the observed data. To remove this issue, we add a constraint that $\mathbf{w}^{(l)}$ is a unit vector. To avoid the problem of identifiability, we propose unit loss:

$$L_{unit}^{(l)} = (\|\mathbf{w}^{(l)}\|_2 - 1)^2. \tag{7}$$

3. 组一致性损失 Group-level consistency loss L_{GLC}

- 只在第一层pooling后计算GLC
- 保留组一致性和个体水平解释的灵活性
- 视作被试的池化得分相似性

Group-level consistency loss We propose group-level consistency (GLC) loss to force BrainGNN to select similar ROIs in a R-pool layer for different input instances. This is because for some applications, users may want to find the common patterns/biomarkers for a certain neuro-prediction task. Note that $\tilde{\mathbf{s}}^{(l)}$ in Eq. (4) is computed from the input $H^{(l)}$ and they change as the layer goes deeper for different instances. Therefore, for different inputs $H^{(l)}$, the selected entries of $\tilde{\mathbf{s}}^{(l)}$ may not correspond to the same set of nodes in the original graph, so it is not meaningful to enforce similarity of these entries. Thus, we only use the GLC loss regularization for $\tilde{\mathbf{s}}^{(l)}$ vectors after the first pooling layer.

Now, we mathematically describe the novel GLC loss. In each training batch, suppose there are M instances, which can be partitioned into C subsets based on the class labels, $\mathcal{I}_c = \{m : m = 1, \ldots, M, y_{m,c} = 1\}$, for $c = 1, \ldots, C$. And $y_{m,c} = 1$ indicates the m^{th} instance belongs to class c. We form the scoring matrix for the instances belonging to class c as $S_c^{(1)} = [\tilde{\mathbf{s}}_m^{(1)} : m \in \mathcal{I}_c]^\top \in \mathbb{R}^{M_c \times N}$, where $M_c = |\mathcal{I}_c|$. The GLC loss can be expressed as:

$$L_{GLC} = \sum_{c=1}^{C} \sum_{m,n \in \mathcal{I}_c} \| \tilde{\mathbf{s}}_m^{(1)} - \tilde{\mathbf{s}}_n^{(1)} \|^2 = 2 \sum_{c=1}^{C} \mathrm{Tr}((S_c^{(1)})^\top L_c S_c^{(1)}), \tag{8}$$

where $L_c = D_c - W_c$ is a symmetric positive semidefinite matrix, W_c is a $M_c \times M_c$ matrix with values of 1, D_c is a $M_c \times M_c$ diagonal matrix with M_c as diagonal elements (Von Luxburg, 2007), m and n are the indices for instances. Thus, Eq. (8) can be viewed as calculating pairwise pooling score similarities of the instances.

4. TopK pooling loss L_{TPK}

- 所选节点得分显著高于未选节点得分

TopK pooling loss The original TPK pooling(Gao and Ji, 2019) used in our R-pool layer does not have regulations on the pooling scores. Thus, the brain ROIs' importance rankings may be very different for different input instances. This can be problematic if the objective is to find the important ROIs shared within a group. Therefore, we propose TopK pooling (TPK) loss to encourage reasonable node selection in R-pool layers. In other words, we hope the top k selected indicative ROIs should have significantly different scores than those of the unselected nodes. Ideally, the scores for the selected nodes should be close to 1 and the scores for the unselected nodes should be close to 0. To achieve this, we rank sigmoid($\tilde{\mathbf{s}}_m^{(l)}$) for the mth instance in a descending order, denote it as $\hat{\mathbf{s}}_m^{(l)} = [\hat{s}_{m,1}^{(l)}, \ldots, \hat{s}_{m,N^{(l)}}^{(l)}]$, and apply a constraint to all the M training instances to make the values of $\hat{\mathbf{s}}_m^{(l)}$ more dispersed. In practice, we define TPK loss using binary cross-entropy as:

$$L_{TPK}^{(l)} = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N^{(l)}}\Big(\sum_{i=1}^{k}\log(\hat{s}_{m,i}^{(l)})) + \sum_{i=1}^{N^{(l)}-k}\log(1-\hat{s}_{m,i+k}^{(l)})\Big), \quad (9)$$

We show the kernel density estimate plots of normalized node pooling scores (indication of the importance of the nodes) changing over the training epoch in Fig. 3 when $k = \frac{1}{2}N^{(l)}$. It is clear to see that the pooling scores are more dispersed over time, Hence the top 50% selected nodes have significantly higher importance scores than the unselected ones. In the experiments below, we fur-