Neural Networks 154 (2022) 56–67

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Deep reinforcement learning guided graph neural networks for brain network analysis

Xusheng Zhao [a,b], Jia Wu [c], Hao Peng [d,*], Amin Beheshti [c], Jessica J.M. Monaghan [e], David McAlpine [f], Heivet Hernandez-Perez [f], Mark Dras [c], Qiong Dai [a,b,*], Yangyang Li [g], Philip S. Yu [h], Lifang He [i]

[a] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[b] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[c] School of Computing, Macquarie University, Sydney, Australia
[d] Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, China
[e] National Acoustic Laboratories, Sydney, Australia
[f] Department of Linguistics, The Australian Hearing Hub, Macquarie University, Sydney, Australia
[g] National Engineering Laboratory for Risk Perception and Prevention (NEL-RPP), CAEIT, Beijing, China
[h] Department of Computer Science, University of Illinois at Chicago, IL, USA
[i] Computer Science & Engineering, Lehigh University, PA, USA

ARTICLE INFO

Article history:
Received 9 July 2021
Received in revised form 25 May 2022
Accepted 28 June 2022
Available online 3 July 2022

Keywords:
Brain network
Network representation learning
Graph neural network
Deep reinforcement learning

ABSTRACT

Modern neuroimaging techniques enable us to construct human brains as brain networks or connectomes. Capturing brain networks' structural information and hierarchical patterns is essential for understanding brain functions and disease states. Recently, the promising network representation learning capability of graph neural networks (GNNs) has prompted related methods for brain network analysis to be proposed. Specifically, these methods apply feature aggregation and global pooling to convert brain network instances into vector representations encoding brain structure induction for downstream brain network analysis tasks. However, existing GNN-based methods often neglect that brain networks of different subjects may require various aggregation iterations and use GNN with a fixed number of layers to learn all brain networks. Therefore, how to fully release the potential of GNNs to promote brain network analysis is still non-trivial. In our work, a novel brain network representation framework, BN-GNN, is proposed to solve this difficulty, which searches for the optimal GNN architecture for each brain network. Concretely, BN-GNN employs deep reinforcement learning (DRL) to automatically predict the optimal number of feature propagations (reflected in the number of GNN layers) required for a given brain network. Furthermore, BN-GNN improves the upper bound of traditional GNNs' performance in eight brain network disease analysis tasks.

1. Introduction

With the advancement of modern neuroimaging, using neuroimaging data effectively has become a research hotspot in both academia and industry (Alexander, Lee, Lazar, & Field, 2007; Huettel, Song, McCarthy, et al., 2004). Many of these techniques, e.g., diffusion tensor imaging (DTI) (Alexander et al., 2007) and functional magnetic resonance imaging (fMRI) (Huettel et al., 2004), enable us to construct human brains as topological networks (known as "brain networks" or "connectomes") (Urbanski, De Schotten, Rodrigo, et al., 2008; Van Den Heuvel & Pol, 2010). Unlike brain images, which consist of pixels, nodes/vertexes and edges/links are components of brain networks. Specifically, nodes in networks usually indicate regions of interest (ROIs), while edges represent connectivity or correlations between ROI pairs. There are often distinct differences between the brain networks derived from different imaging modalities (Liu et al., 2018). For example, DTI-derived brain networks encode the structural connections among ROIs based on white matter fibers, while fMRI-derived brain networks record the functional activity routes of the regions. Both DTI-derived and fMRI-derived human brain networks are well-researched and applied to the whole-brain analysis.

* Corresponding authors.
E-mail addresses: zhaoxusheng@iie.ac.cn (X. Zhao), jia.wu@mq.edu.au (J. Wu), penghao@buaa.edu.cn (H. Peng), amin.beheshti@mq.edu.au (A. Beheshti), jessica.monaghan@nal.gov.au (J.J.M. Monaghan), david.mcalpine@mq.edu.au (D. McAlpine), heivet.hernandez-perez@mq.edu.au (H. Hernandez-Perez), mark.dras@mq.edu.au (M. Dras), daiqiong@iie.ac.cn (Q. Dai), liyangyang@cetc.com.cn (Y. Li), psyu@uic.edu (P.S. Yu), lih319@lehigh.edu (L. He).

https://doi.org/10.1016/j.neunet.2022.06.035
0893-6080/

The computational analysis of brain networks is becoming more and more popular in healthcare because it can discover meaningful structural information and hierarchical patterns to help understand brain functions and diseases. Here we take brain disease prediction as an example. As one of the most common diseases affecting human health, brain disease has a very high incidence and disability rate, bringing enormous economic and human costs to society (Parisot et al., 2018). Considering the complexity and diversity of the predisposing factors of brain diseases, researchers often analyze the brain network states of subjects to assist in inferring the types of brain diseases and then provide reliable and effective prevention or treatment guidelines. For example, as a geriatric epidemic, Alzheimer's disease (AD) often manifests as memory and visual–spatial skill impairments (Braak & Braak, 1991; Zhang et al., 2018). Although the existing medical methods cannot effectively treat AD patients, it is possible to delay the onset of AD by tracking the subject's brain network changes and performing interventional therapy in the stage of mild cognitive impairment (Huang & Mucke, 2012).

One of the essential techniques in brain network analysis is brain network representation (a.k.a., network embedding) (Cao et al., 2017; Liu et al., 2018), which aims to embed the subjects' brain networks into meaningful low-dimensional representations. These network representations make it easy to separate damaged or special brain networks from normal controls, thereby providing supplementary or supporting information for traditional clinical evaluations and neuropsychological tests. As a popular network/graph representation/embedding framework nowadays, graph neural networks (GNNs) (Hamilton, Ying, & Leskovec, 2017; Kipf & Welling, 2017; Velickovic et al., 2018) applies convolution to the network structure (Peng et al., 2020; Sun et al., 2021), which learns deep features while reasoning about relational induction within the network (LeCun, Bengio, & Hinton, 2015; Liu, Xue et al., 2020; Ma et al., 2021). Therefore, many GNN-based embedding algorithms (Arslan, Ktena, Glocker, & Rueckert, 2018; Ktena et al., 2018; Zhang et al., 2018) for brain networks have emerged recently. For brain analysis tasks that treat a subject's brain network as an instance, such as brain disease prediction, GNN-based methods first use stackable network modules to aggregate information from neighbors at different hops. This way, they capture brain networks' structural information and hierarchical patterns. More specifically, GNN learns node-level feature representations by aggregating neighbor information edgewise, where the number of GNN layers controls the total degree of iterative aggregations. Then they apply global pooling on the node-level feature matrices to obtain network-level representations.

Although GNN-based analysis strategies have been successfully introduced into various brain network analysis tasks, including brain network classification (Arslan et al., 2018) and clustering (Liu et al., 2018), it is still challenging to release the full potential of GNNs on different brain networks. Specifically, existing works usually utilize GNN with a fixed number of layers to learn all brain network instances, ignoring that different brain networks often require distinct optimal aggregation iterations due to structural differences. On the one hand, more aggregation iterations mean considering neighbors at farther hops, which may prompt some brain networks to learn better representations. Unfortunately, increasing the number of aggregations in GNN may also cause over-smoothing problems (Chen et al., 2020; Oono & Suzuki, 2019), which means that all nodes in the same network have indistinguishable or meaningless feature representations. On the other hand, it is infeasible to manually specify the number of iterative aggregations for different brain networks, especially when the instance set is large. A straightforward method to alleviate these problems is to deepen the GNN model with skip-connections (a.k.a., shortcut-connections) (Gao & Ji, 2019; Li, Muller, Thabet, & Ghanem, 2019), which avoid gradient vanishing and a large number of hyperparameter settings. However, this is a sub-optimal strategy because it fails to automate GNN architectures for different brain networks without manual adjustments.

To solve the above problems, a novel GNN-based brain network embedding framework (BN-GNN) designed for brain network analysis is developed. With the recent proposal and application of meta-policy learning (Lai, Zha, Zhou, & Hu, 2020; Zha, Lai, Zhou, & Hu, 2019), we expect a meta-policy that automatically determines the optimal number of feature aggregations (reflected in the number of GNN layers) for a given brain network. Specifically, we heuristically model the optimization and decision iteration of a meta-policy as a Markov decision process (MDP). First of all, we regard the adjacency matrix of a randomly sampled brain network as the initial state and input it into the policy in MDP. Secondly, we guide the construction of the GNN in MDP based on the action (an integer) corresponding to the maximum Q value output by the policy. Here the action value determines the stacking of GNN layers, which controls the total count of feature aggregations on the current brain network. Thirdly, we pool the node features into a network representation. Then we perform network classification to optimize the current GNN and employ a novel strategy to calculate the current immediate reward. Fourthly, we sample the next network instance through a heuristic state transition strategy and record the state–action–reward–state quadruple of this process. In particular, we apply the double deep q-network (DDQN) (Mnih et al., 2015; Van Hasselt, Guez, & Silver, 2016), a classic deep reinforcement learning (DRL) (Arulkumaran, Deisenroth, Brundage, & Bharath, 2017) algorithm, to simulate and optimize the policy. Finally, we utilize the trained policy (i.e., meta-policy) as meta-knowledge to guide the construction and training of another GNN and implement specific brain network analysis tasks.

Overall, the main results are condensed as:

- A novel brain network representation learning framework (i.e., BN-GNN [1]) through GNN and DRL is proposed to assist brain network analysis tasks. In this way, the number of GNN feature aggregations can be altered for different brain networks, thereby releasing the full potential of traditional GNNs in brain network representation learning.
- This is the first study in the field of brain network analysis to introduce DRL into the GNN model. We are also the first to use GNNs with different layers to learn different subjects' brain networks.
- Experiments on eight brain network disease analysis datasets (e.g., BP-DTI and HA-EEG) show that BN-GNN stands out among many advanced algorithms, improving the upper bound of traditional GNNs' performance.

The remainder of the article is briefly described as: Sections 2 and 3 describe related work and preliminary knowledge, respectively. Section 4 details the implementation of BN-GNN with graph convolutional network (GCN) (Kipf & Welling, 2017) and DDQN (Van Hasselt et al., 2016). Section 5 gives the experimental results and corresponding analysis. Section 6 summarizes our work.

2. Related work

2.1. Graph neural networks

GNNs are currently the preferred strategy for processing topological graph data, which follows the principle that neighbor

[1] https://github.com/RingBDStack/BNGNN

information affects the feature embedding of central nodes edge-wise. Spectral- and spatial-based are two major labels of existing GNN models. A representative product of GNN models is GCN (Kipf & Welling, 2017), which is inspired by the traditional convolutional operations on images in the Euclidean space. To widen and improve aggregation performance bounds and explainability, the graph attention network (GAT) (Velickovic et al., 2018) refines the relative importance of neighbors from the perspective of target nodes. To improve the aggregation efficiency of large-size graphs, GraphSAGE (Hamilton et al., 2017) extracts a consistent and predefined number of local neighbors for each target one. In addition, many GNN variants based on the above methods or frameworks have been proposed, and they have made outstanding contributions, such as in clinical medicine (Zhang et al., 2018).

2.2. Reinforcement learning guided graph neural networks

Recently, with the advances of RL, many works combine RL with GNNs to further raise the performance boundary of GNNs. For example, Dou et al. (2020) proposed a GNN algorithm, i.e., CARE-GNN to improve the capability to recognize fraudsters in fraud inspection tasks. CARE-GNN first sorts the neighbors based on their credibility and then uses RL to guide the traditional GNN to filter out the most valuable neighbors for each node to avoid fraudsters from interfering with normal users. Peng, Zhang et al. (2021) proposed a novel recursive and reinforced graph neural network framework learn more discriminative and efficient node representation from multi-relational graph data. Peng et al. (2022) proposed a GNN that considers different relations to achieve network representation, which is based on multi-agent RL for relation importance assignment. Gao, Yang, Zhang, Zhou, and Hu (2020) proposed a graph NAS algorithm, namely GraphNAS. GraphNAS first utilizes a recurrent network to create variable-length strings representing the architectures of GNNs and then applies RL to update the recurrent network for maximizing quality of model building. Nishi, Otaki, Hayakawa, and Yoshimura (2018) proposed a GCN-based algorithm for traffic signal control called NFQI, which applies a model-free RL approach to learn responsive traffic control in order to deal with temporary traffic demand changes when environmental knowledge is insufficient.

Since deep RL (DRL) combines the perception capability of deep learning with the decision capability of RL, it is a new research hotspot in artificial intelligence. For example, Yan, Ge, Wu, Li, and Li (2020) proposed a virtual network embedding algorithm (i.e., V3C+GCN) that combines DRL with a GCN-based module. Lai et al. (2020) proposed a meta-policy framework (i.e., Policy-GNN), which adaptively learns an aggregation strategy to use DRL to perform various aggregation iterations on different nodes. Though the above methods directly or indirectly use RL or DRL to improve GNNs, there is still no work using DRL to guide GNNs to assist brain network analysis, which often requires different models for different brain networks.

2.3. GNN-based brain network representation learning

Unlike traditional shallow methods for brain network representation learning, such as tensor decomposition (Cao et al., 2017; Liu et al., 2018), some works use GNNs to capture deep feature representations of brain networks for downstream brain analysis tasks. Concretely, Li et al. (2020) proposed a PR-GNN that includes regularized pooling layers, which calculates node pooling scores to infer which brain regions are obligatory parts of certain brain disorders. Bi et al. (2020) proposed an aggregator that applies extreme learning machines (ELMs), which avoids tuning iterations and widens the feature passing performance boundaries. In addition, they provided a GNEA model based on the aforementioned aggregator to enable brain graph analysis. Hi-GCN method proposed by Jiang, Cao, Xu, Yang, and Zaiane (2020) can perform hierarchical embedding of brain networks. In order to improve the accuracy of brain disease analysis, Hi-GCN considers graph structure induction and also introduces patient group-level structural information. Ma et al. (2019) developed an graph learning algorithm for brain network analysis, namely HS-GCN, which utilizes two GCNs to build a siamese model and learns brain network representations by means of supervised metrics. Zhong, Wang, and Miao (2020) proposed a regularized GNN (i.e., RGNN) for emotion recognition based on electroencephalogram. Xing et al. (2021) developed a GCN-based algorithm (i.e., DS-GCNs) that can condense meaningful representations from functional connections readily available in neuroanalytical tasks. DS-GCNs calculate a dynamic functional connectivity matrix with a sliding window and implement a long and short-term memory layer based on graph convolution to process dynamic graphs.

Corresponding to unimodal brain connectome studies, GNNs are also quite popular in multi-modality brain analysis scenarios. For example, Zhang et al. (2018) proposed a GCN model (i.e., MVGCN) for combining different view information in brain analysis tasks, helping to distinguish Parkinson's disease cases from healthy controls. Gurbuz and Rekik (2021) proposed a multi-view normalization network based on GNN (i.e., MGN-Net), which normalizes and combines a set of multi-view brain networks into one.

Although these GNN-based single- or multi-modality methods have made significant breakthroughs in many brain network analysis tasks, they have failed to implement customized aggregation for different subjects' brain networks in experiments such as brain disease prediction.

3. Preliminaries

First of all, we formulate the brain network analysis. Next, we introduce the network representation learning method when the GNN layer is predefined, MDP, and DRL. The key notations/symbols are given in Table 1.

3.1. Problem formulation

Generally, a brain connectome can be abstracted as a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ indicates the node set, E contains weighted edges that represent topological relations among nodes. Let $\mathbf{W}$ denote G's initial weighted matrix, so $\mathbf{W}(i, j)$ means the edge correlation between v_i and v_j (which may tend to zero for no or weak connection). Let $D = \{G_1, \ldots, G_m\}$ be an ensemble of brain networks based on m brain subjects. We assume that these network instances have different structures but the same nodes, where a specific region division strategy determines the number of nodes. Given the kth brain network $G_k = (V_k, E_k)$, we abstract it as a weighted matrix $\mathbf{W}_k \in \mathbb{R}^{n \times n}$.

We focus on the problem of brain network representation learning based on DRL-introduced GNNs, which is used for both classification and clustering. Specifically, we focus on the classification task since it is often the research basis for brain network analysis. Given dataset D, we assume that the corresponding network labels $\mathbf{Y}$ are known. For convenience, we denote the training, validation, and test set of D as D_{train}, D_{val}, and D_{test}, respectively, where $D = D_{train} \cup D_{val} \cup D_{test}$. Based on the brain networks in $D_{train} \cup D_{val}$, we first continuously optimize a policy π. Next, we employ the trained policy (i.e., meta-policy) to guide the construction of GNN and utilize the customized GNN to learn the

Table 1
Glossary of notations.

Notation	Definition
D; D_{train}; D_{val}; D_{test}	The brain network dataset; The training set of D; The validation set of D; The test set of D
G; V; E	The brain network; Nodes in G; Edges in G
S; A	The state space in MDP; The action space in MDP
$\mathbf{W}$; $\mathbf{A}$; $\widehat{\mathbf{A}}$	The initial weighted matrix of G; The adjacency matrix of G; The normalized form of $\widetilde{\mathbf{A}}$
$\widetilde{\mathbf{A}}$; $\widetilde{\mathbf{D}}$	The self-loop form of $\mathbf{A}$; The degree matrix of $\widetilde{\mathbf{A}}$
$\mathbf{E}$; $\mathbf{F}$	The network-level feature matrix of D; The node-level feature matrix of G
$\mathbf{T}$	The feature transformation operator
$\mathbf{C}$; $\widehat{\mathbf{C}}$	The importance coefficient matrix of G; The normalized form of $\mathbf{C}$
m; n	Total count of brain networks of D; Total node count of V
d	The vector representation dimension of $\mathbf{E}$ or $\mathbf{F}$
l	The total number of layers of GNN layers or iterative aggregations
i; j; k	These notations represent index variables
s; a; r	The state in MDP; The action in MDP; The reword in MDP
t; b	The final timestep value of MDP; Set size containing all actions
w	The window size of the history records in $REW(\cdot)$
$\oplus$	The feature combination operation, such as summation and concatenation
π	The policy function in MDP or the meta-policy
γ; ϵ	The discount coefficient of $\mathcal{R}$; The epsilon probability of exploration of π
$\sigma(\cdot)$	The activation function, such as *Tanh* and *ReLU*
$AGG(\cdot)$	The feature aggregation function of GNN, such as convolution and attention
$REW(\cdot)$	The immediate reward function in MDP
$PER(\cdot)$	The classification performance metric, such as accuracy.
$\mathcal{R}$	The discounted cumulative return in MDP
$\mathcal{Q}_{eval}$; $\mathcal{Q}_{target}$	The evaluation DNN in DDQN; The target DNN in DDQN
$\mathcal{L}_{GNN}$; $\mathcal{L}_{policy}$	The training loss of GNN; The training loss of meta-policy

node representations of each brain network that meet the number of feature aggregations. Then, by applying the global pooling at the last layer of GNN, we convert the node-level feature tensor into a low-dimensional network-level representation matrix $\mathbf{E}$, allowing brain network instances with different labels to be easily separated. Last, we feed $\mathbf{E}$ into the full-connected layer to perform brain network classification.

3.2. Learning network representations with layer-fixed GNN

GNNs learn node-level feature representations through the network structure. Given an instance $G = (V, E)$, we receive its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and initial region features $\mathbf{F}^{(0)} \in \mathbb{R}^{n\times d^{(0)}}$, and then express the feature aggregation process of $v_i \in V$ in a layer-fixed GNN as follows (Dou et al., 2020):

$$\mathbf{F}^{(l)}(i) = \sigma\left(\mathbf{F}^{(l-1)}(i) \oplus AGG^{(l)}\left(\left\{\mathbf{F}^{(l-1)}(j) : \mathbf{A}(i,j) > 0\right\}\right)\right), \tag{1}$$

where $\mathbf{F}^{(l-1)} \in \mathbb{R}^{n\times d^{(l-1)}}$ and $\mathbf{F}^{(l)} \in \mathbb{R}^{n\times d^{(l)}}$ indicate the input and output features of the model. $AGG^{(l)}$ indicates the aggregation module, their superscript (l) indicates that the features or modules belong to the l-th layer. $\oplus$ is an operation used to fuse the features of v_i and its neighbors. σ means the activation function like *Tanh*. It is worth noting that $\mathbf{A}$ should be reliable as it remains unchanged at all layers. Taking graph convolutional network (GCN) (Kipf & Welling, 2017) with two aggregations as an example, it implements Eq. (1) through convolution:

$$\begin{aligned}&\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}\\ &\mathbf{F}^{(2)} = ReLU(\widehat{\mathbf{A}}ReLU(\widehat{\mathbf{A}}\mathbf{F}^{(0)}\mathbf{T}^{(1)})\mathbf{T}^{(2)})\end{aligned}, \tag{2}$$

where $\widehat{\mathbf{A}} \in \mathbb{R}^{n\times n}$ is the symmetrical normalized form of $\widetilde{\mathbf{A}}$, $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with the identity matrix $\mathbf{I} \in \mathbb{R}^{n\times n}$ added, and $\widetilde{\mathbf{D}} \in \mathbb{R}^{n\times n}$ is the degree matrix of $\widetilde{\mathbf{A}}$. At the first layer, since $\widehat{\mathbf{A}}$ encodes each node's direct (1-hop) neighbor information, $\widehat{\mathbf{A}}\mathbf{F}^{(0)}$ essentially implements the first convolution aggregation through summation. At the second layer, the continuous multiplication of the adjacency matrix (i.e., $\widehat{\mathbf{A}}ReLU(\widehat{\mathbf{A}})$) makes the neighbor's neighbor (2-hop) information included in the second aggregation. Therefore, when GNN models stack and aggregate more, the receptive field of GNN becomes wider, so more neighbors participate in aggregation. In addition, $\mathbf{T}^{(1)} \in \mathbb{R}^{d^{(0)}\times d^{(1)}}$ and $\mathbf{T}^{(2)} \in \mathbb{R}^{d^{(1)}\times d^{(2)}}$ are the learnable matrices for feature transformation of the first and second layer, respectively. Unlike GCN, which equally distributes the importance of all neighbors, GAT (Velickovic et al., 2018) counts neighborhood importance weights through attention when summing their features. Taking the first layer of a single-head GAT as an example, the node feature aggregation is as follows:

$$\begin{aligned}&\mathbf{C}(i,j) = (\mathbf{F}^{(0)}(i)\mathbf{T}^{(1)} \oplus \mathbf{F}^{(0)}(j)\mathbf{T}^{(1)})q^T\\ &\widehat{\mathbf{C}} = softmax(\mathbf{C}(i,j)) = \frac{\exp(ReLU(\mathbf{C}(i,j)))}{\sum_{v_k\in V(i)}\exp(ReLU(\mathbf{C}(i,k)))},\\ &\mathbf{F}^{(1)}(i) = ReLU(\sum_{v_j\in V(i)}\widehat{\mathbf{C}}\mathbf{F}^{(0)}(j)\mathbf{T}^{(1)})\end{aligned} \tag{3}$$

where $\mathbf{F}^{(0)}(i) \in \mathbb{R}^{1\times d^{(0)}}$ represents the initial feature representation of node v_i, $\mathbf{T}^{(1)} \in \mathbb{R}^{d^{(0)}\times d^{(1)}}$ is the feature transformation matrix whose parameters are shared, $\oplus$ is the concatenation operation, and $q \in \mathbb{R}^{1\times 2d^{(1)}}$ is the attention feature vector. $\mathbf{C}(i,j)$ is the importance coefficient (a real number) of node v_j to node v_i, $\widehat{\mathbf{C}}$ is the normalized form of $\mathbf{C}$, and $\mathbf{F}^{(1)}(i) \in \mathbb{R}^{1\times d^{(1)}}$ is the transformed representation vector of the target v_i. Here $V(i) = \left\{v_j : \mathbf{A}(i,j) > 0\right\}$ indicates the neighbor set of node v_i. Similarly, GAT also controls the number of feature aggregations by changing the number of layers. After the final aggregation is completed at the last layer l, GNN-based methods perform global pooling on all nodes to obtain the final network representation. The process of global average pooling is described below:

$$\mathbf{E}(i) = \frac{1}{n}\sum_{v_j\in V}\mathbf{F}_i^{(l)}(j), \tag{4}$$

where $\mathbf{E}(i) \in \mathbb{R}^{1\times d^{(l)}}$ and $\mathbf{F}_i^{(l)}(j)$ are the network-level feature vector and node-level feature matrix of the ith network instance G_i, respectively. Then the cross entropy loss of this part is as follows:

$$\mathcal{L}_{GNN} = -\sum_{G_i\in D_{train}}\log(\mathbf{E}(i)\mathbf{T}^{(l+1)})\mathbf{Y}(i)^T, \tag{5}$$

where $\mathbf{T}^{(l+1)}$ is the fully connected layer applied as a classifier, $\mathbf{Y}(i)$ indicates the i-th brain instance's class label.

In brain network disease analysis, most instances can be abstracted as weighted matrices describing the connections among

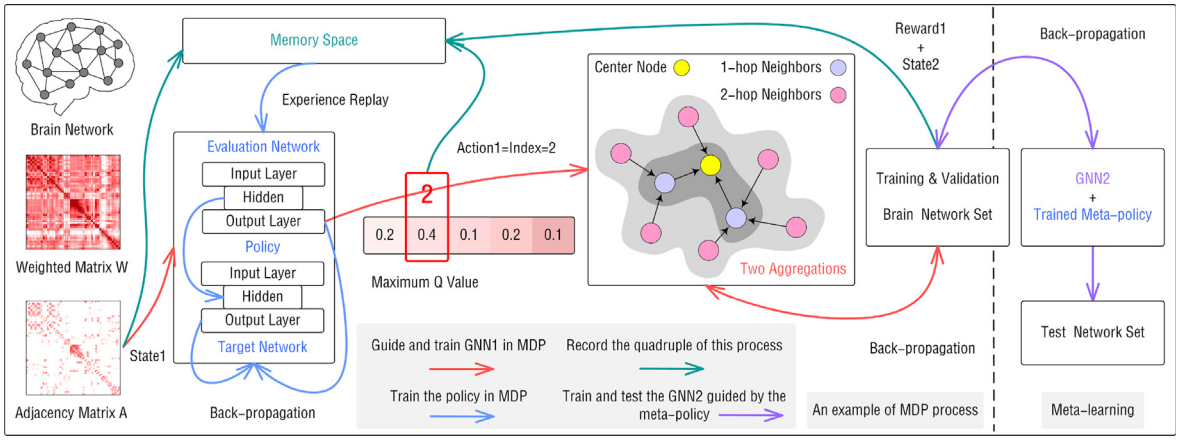


Fig. 1. Illustration of our proposed framework BN-GNN for brain network representation learning. The left side of the dotted line illustrates an example of the MDP process. First of all, we treat the adjacency matrix derived from the brain network building module as the current state and input it into the policy. Then we follow the guiding signals (Q values) output by the policy to decide the current action. Here we treat the index of the selected action in the action space as the action value to guide the number of aggregations of the current brain network in GNN. After feature aggregation, we employ global pooling to harvest network instance-level representations and train GNN models in MDP. Subsequently, the current reward is calculated by comparing the performance changes on the verification set. Finally, the transition strategy is applied to obtain the state of the next timestep. To train the policy in MDP, we record the state–action–reward–state quadruple of this process to the memory space and follow the DDQN method to calculate the loss of the policy. On the right side of the dotted line, we apply the trained meta-policy to guide the training of a new GNN and perform brain network analysis tasks.

brain regions, but they often do not have initial region features. Besides, there is little research on constructing informative node features and edges for GNN-based brain network learning. A common strategy is to use the initial weighted matrix $\mathbf{W}$ associated with each brain network G as its initial node features (i.e., $\mathbf{F}^{(0)} = \mathbf{W}$) and define a group-level adjacency matrix $\mathbf{A}$ for GNN. For example, Zhang et al. (2018) defined $\mathbf{A}$ as a coarse-grained network processed by k-nearest neighbor (KNN), and Zhang and Huang (2019) constructed the Laplacian operator through the small-world model to infer $\mathbf{A}$ in the process of representation learning. However, using the same adjacency matrix for different brain networks may blur the differences between different networks. Different from the previous work (Zhang et al., 2018; Zhang & Huang, 2019), our goal is to generate a separate adjacency matrix for each brain network and implement network representation learning for different brain networks based on GNNs with different layers.

3.3. Markov decision process

MDP is a natural description of sequential decision problems, used to simulate the random actions and rewards that the agent can achieve in an environment with Markov properties. Here we denote the MDP as a quintuple $(S, A, \pi, REW, \mathcal{R})$, where S and A mean the state and action set/space, π is the policy that outputs the action conditional probability distribution of the input state, $REW : S \times A \to \mathbb{R}$ is the immediate reward function, and $\mathcal{R}$ is the accumulation of rewards over time (a.k.a, return). The decision process in each timestep $i \in [1, t]$ is as follows: the agent first perceives the current state $s_i \in S$ and then implement an action as directed by π. Then, the environment (which is affected by the action a_i) feeds back to the agent the next state s_{i+1} as well as a reward $r_i = REW(s_i, a_i)$. In the standard MDP, our goal is to train π to maximize the accumulation of discounted returns. The total return can be expressed in summed form as:

$$\mathcal{R} = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{t-1} r_t = \sum_{i=1}^{t} \gamma^{i-1} r_i, \tag{6}$$

where $\gamma \in (0, 1)$ is the discount coefficient used to constrain future rewards with low reliability. The optimization steps for policy π are detailed in the next section.

3.4. Solving MDP with deep reinforcement learning

In many scenarios, the state space S is huge or inexhaustible. In this case, it is sub-optimal or infeasible to train the policy π by maintaining and updating a state–action table. Deep reinforcement learning (DRL) (Arulkumaran et al., 2017) is an effective solution because it can use neural networks to simulate and approximate the actual relationship between any state and all possible actions. Here, we focus on a classic DRL algorithm called double deep q-learning (DDQN) (Mnih et al., 2015; Van Hasselt et al., 2016), which uses two deep neural networks (DNNs) to simulate the policy π. Concretely, in each timestep i, DDQN first inputs the current state s_i into the evaluation DNN to get the predicted values of all actions and regard the action corresponding to the maximum Q value as the current action, which should conform to the following formula:

$$a_i = \begin{cases} random\ action, & \text{w.p. } \epsilon \\ \text{argmax}_{a_i}(Q_{eval}(s_i, a_i)), & \text{w.p. } 1 - \epsilon, \end{cases} \tag{7}$$

where ϵ-greedy can make DDQN more portable and avoid the dilemma of exploration and utilization. The maximum Q value $\max_{a_i}(Q(s_i, a_i))$ is essentially the expected maximum discounted return in the current state s_i, and the corresponding Bellman equation can be expressed as follows:

$$\begin{aligned} \max_{a_i}(Q_{eval}(s_i, a_i)) = \max(\mathcal{R}_i) &= \max(r_i + \gamma(r_{i+1} + \gamma(r_{i+2} + \cdots))) \\ &= r_i + \gamma \max(\mathcal{R}_{i+1}). \end{aligned} \tag{8}$$

After determining the current action a_i, DDQN uses the reward function (designed according to the specific environment) to calculate the current actual reward (i.e., $r_i = REW(s_i, a_i)$) and then performs state transition to obtain the next state s_{i+1}. Moreover, there is a memory space in DDQN that records each MDP process (also known as "experience" and recorded as a state–action–reward–state quadruple $\langle s_i, a_i, r_i, s_{i+1} \rangle$). To optimize DNNs with experience replay, DDQN first records the current experience and then randomly extracts a memory block from the memory space. For example, through the experience $\langle s_i, a_i, r_i, s_{i+1} \rangle$, the loss of DNNs (i.e., policy) can be calculated as follows:

$$\begin{aligned} \mathcal{L}_{policy} &= (\max(\mathcal{Q}_{target}(s_i, a_i)) - \max(\mathcal{Q}_{eval}(s_i, a_i)))^2 \\ &= (r_i + \gamma \max_{a_{i+1}}(\mathcal{Q}_{target}(s_{i+1}, a_{i+1})) - \max_{a_i}(\mathcal{Q}_{eval}(s_i, a_i)))^2, \end{aligned} \tag{9}$$

where state s_i is input to the evaluation DNN to obtain the predicted maximum Q value $\max_{a_i}(\mathcal{Q}_{eval}(s_i, a_i))$ in the ith timestep, the next state s_{i+1} is input to the target DNN to calculate the maximum Q value $\max_{a_{i+1}}(\mathcal{Q}_{target}(s_{i+1}, a_{i+1}))$ in the next timestep, and r_i is the actual reword. Based on the Bellman equation, DDQN takes the $\max(\mathcal{Q}_{target}(s_i, a_i))$ output from the target DNN as the actual maximum Q value in timestep i and trains the evaluation DNN through the back-propagation algorithm. It is worth noting that DDQN does not update the target network through loss but copies the parameters of the evaluation DNN to the target DNN. In this way, DDQN effectively alleviates the over-estimation problem often found in DRL. Since DDQN uses two DNNs to simulate the policy π, the above is also the training process of π.

4. Methodology

Fig. 1 illustrates the brain network representation learning framework BN-GNN, which consists of three modules: network building module, meta-policy module and GNN module. The network building module provides the state space for the meta-policy module. The meta one utilizes feedbacks (i.e., rewards) of the GNN module to search for the optimal meta-policy π continuously, and the GNN module performs brain network representation learning according to the guidance (i.e., actions) of the meta-policy. Next, we introduce the technical details of each module.

4.1. Network building module

The network building module generates adjacency matrices from the brain networks' initial weighted matrices, providing state space for the meta-policy module. In GNN, feature aggregation that relies on adjacency matrix $\mathbf{A}$ is essential. Therefore, $\mathbf{A}$ should be appropriately designed to reflect the neighborhood correlations because it directly affects the node feature representation learning. Inspired by the previous work in Zhang et al. (2018), we utilize KNN to construct reliable adjacency matrices to advance learning brain network representation for GNNs. Specifically, taking a brain instance $G = (V, E)$, we utilize its weighted matrix $\mathbf{W} = \mathbf{F}^{(0)}$ and KNN to obtain reliable neighbors $V(i)$ of any node $v_i \in V$. If $v_i \in V(j)$ or $v_j \in V(i)$, then $\mathbf{A}(i, j) = 1$ and $\mathbf{A}(j, i) = 1$, otherwise both are zero. After that, we calculate new edge confidences to refine the reliable matrix $\mathbf{A}$:

$$\mathbf{A}(i,j) = \begin{cases} \exp(-\left\|\mathbf{F}^{(0)}(i) - \mathbf{F}^{(0)}(j)\right\|), & \mathbf{A}(i,j) = 1 \\ 0, & \mathbf{A}(i,j) = 0, \end{cases} \tag{10}$$

where $\mathbf{F}^{(0)}(i)$ is $\mathbf{F}^{(0)}$'s i-th row and v_i's initial feature embedding. The module is also used to construct the subject network in the state transition strategy, which will be introduced in the following subsection.

4.2. Meta-policy module

The meta-policy module trains a policy that can be viewed as meta-knowledge to determine the number of aggregations of brain network features in GNN. As mentioned in Section 3.3, the learning of the policy π is abstracted as an MDP that contains five essential components, i.e., $(S, A, \pi, REW, \mathcal{R})$. Here, we give the relevant definitions in the context of brain network embedding in timestep i.

- State space (S): The state $s_i \in S$ represents the adjacency matrix of the brain instance.
- Action space (A): The action $a_i \in A$ determines the number of iterations for feature aggregation that the brain network requires, which is reflected in the number of GNN layers. Since the GNN layer count is a positive integer, we define the index of each action in the action space as the corresponding action value.
- Policy (π): The policy in timestep i outputs action a_i according to the input state s_i. Here we apply double deep q-network (DDQN) presented in Section 3.4 to simulate and train the policy and call the trained policy a meta-policy.
- Reward function (REW): The reward function outputs the reward r_i in timestep i. Since we expect to improve network representation performance through policy-guided aggregations, we intuitively define the current immediate reward r_i as the difference (a decimal) between the current validation classification performance and the performance of the previous timestep.
- Return ($\mathcal{R}$): The return $\mathcal{R}_i$ in timestep i indicates the discounted accumulation of all rewards in the interval $[i, t]$. Based on the DDQN, we approximate the Q values output by the DNNs in the DDQN to the rewards over different actions. Since DDQN always chooses the action that maximizes return, it aligns with the goal of standard MDP.

According to these definitions, the process of the meta-policy module in each timestep i includes five stages: (1) Sample a brain network and take its adjacency matrix as the current state s_i. (2) Determine the number of layers of the GNN that processes the current brain network according to the action a_i corresponding to the maximum Q value output by the policy π. (3) Calculate the current reward r_i based on performance changes (technical details will be introduced in the following subsection). (4) Obtain the next state s_{i+1} with a new heuristic strategy for state transition. Concretely, we abstract the brain network of each subject as a coarse node and construct a subject network according to the network building module, where the initial node features are obtained by vectorizing the weighted matrices. Then we realize state transition through node sampling. For example, given the current state s_i and action a_i, we randomly sample a a_i-hop neighbor of the coarse node corresponding to state s_i in the subject's network, where brain instance's adjacency matrix corresponding to the sampled neighbor is the next state s_{i+1}. In this way, the state transition obeys Markov, i.e., the next state s_{i+1} is only affected by the current one s_i without considering previous states. (5) Record the process of this timestep and train the policy π according to Eq. (9) and the back-propagation algorithm.

4.3. GNN module

The GNN module contains two GNNs with a pooling layer to learn brain network representations. The first GNN (called GNN1) is used in the MDP to train the policy π. As defined in Section 4.2, each action is a positive integer in the interval $[1, b]$, where b is the total count of all possible actions. Since the action a_i specifies the number of feature aggregations and GNN achieves different aggregations by controlling the number of layers, GNN1 needs to stack j neural networks when $a_i = j$ (j is the index of a_i in A). Considering that the actions in different processes are usually different, reconstructing GNN1 in each timestep is very time- and space-consuming. To alleviate this problem, we use a parameter sharing mechanism to construct a b-layer GNN1. For example, given the current action $a_i = j$, we only use the first j layers of GNN1 to learn the current brain network G_i. The aggregation process realized by GCN (Kipf & Welling, 2017) is as follows:

$$\mathbf{F}^{(j)} = ReLU(\widehat{\mathbf{A}} \cdots ReLU(\widehat{\mathbf{A}}\mathbf{F}^{(0)}\mathbf{T}^{(1)}) \cdots \mathbf{T}^{(j)}). \tag{11}$$

After obtaining the final node feature matrix $\mathbf{F}^{(j)}$, we apply the pooling of Eq. (4) to obtain the network representation. Then we use the back-propagation algorithm of Eq. (5) to train GNN1. Since the current timestep only involves the first b layers of GNN1, only the parameters of the first b layers are updated. Compared with constructing a GNN for each network separately

Algorithm 1 BN-GNN: GNN-based brain network representation learning framework

Input: Brain network dataset D, number of timesteps t, number of all possible actions b, discount coefficient γ, epsilon probability ϵ, window size of the history records w.

1: Generate adjacency matrices of brain networks and the subject network via Eq. (10), $\mathbf{A}(i,j) = \exp(-\left\|\mathbf{F}^{(0)}(i) - \mathbf{F}^{(0)}(j)\right\|)$ or 0.
2: Initialize two DNNs in DDQN and two GNNs with b layers.
3: Randomly sample a brain network from D_{train} and get the starting state according to the adjacency matrix.
4: **for** $i = 1, 2, \ldots, t$ **do**
5: Select the action via Eq. (7), $a_i = argmax_{a_i}(Q_{eval}(s_i, a_i))$ or a random action.
6: Train the action value guided GNN1 in MDP via Eq. (5), $\mathcal{L}_{GNN} = -\sum_{G_j \in D_{train}} \log(\mathbf{E}(i)\mathbf{T}^{(l+1)})\mathbf{Y}(i)^T$.
7: Calculate the reward via Eq. (12), $r_i = REW(s_i, a_i) = PER(s_i, a_i) - \frac{1}{w}\sum_{i-w}^{i-1} PER(s_i, a_i)$.
8: Sample the next state and record this process.
9: Train the policy in MDP via Eq. (9), $\mathcal{L}_{policy} = (\max(\mathcal{Q}_{target}(s_i, a_i)) - \max(\mathcal{Q}_{eval}(s_i, a_i)))^2$.
10: **end for**
11: Use the trained policy, meta-policy as meta-knowledge to guide the training and testing of GNN2.
12: Obtain network representations of the test set D_{test} and perform brain analysis tasks.

in each timestep, the parameter sharing mechanism significantly improves the training efficiency.

To calculate the current reward r_i, we measure the classification performance of GNN1 on the validation set D_{val}. The immediate reward in MDP is obtained as follows:

$$r_i = REW(s_i, a_i) = PER(s_i, a_i) - \frac{1}{w}\sum_{i-w}^{i-1} PER(s_i, a_i), \tag{12}$$

where PER represents the performance metric of the classification result on the validation data (here we apply accuracy). w indicates the number of historical records used to determine benchmark performance $\frac{1}{w}\sum_{i-w}^{i-1} PER(s_i, a_i)$. Compared with only considering the performance of the previous timestep $(i-1)$, the benchmark based on multiple historical performances improves the reliability of r_i.

Since the training of GNN1 and policy in MDP are usually not completed in the same timestep, it is inconvenient and inappropriate to use GNN1 to perform brain analysis tasks on the test set D_{test}. Therefore, after MDP, we apply the trained meta-policy to guide the training and testing of a new GNN (called GNN2), where GNN2 and GNN1 have the same aggregation type and parameter sharing mechanism. The detailed steps of BN-GNN is presented in Algorithm 1.

5. Experiments

Eight real-world brain network datasets are used to evaluate our proposed BN-GNN. Our first step is to give the brain analysis dataset information (Section 5.1), the comparison baselines, and the experimental settings (Section 5.2). We then conduct sufficient experiments on the brain network classification task to address multiple research questions (RQs) about BN-GNN's effectiveness:

- RQ1. Is BN-GNN better than other advanced brain network representation algorithms? (Section 5.3)
- RQ2. Can the three modules included in BN-GNN improve brain network representations learning? (Section 5.4)
- RQ3. How do important hyperparameters in BN-GNN affect model representation performance? (Section 5.5)

5.1. Datasets

The brain analysis tasks consist of eight brain network datasets covering different neurological disorders. Table 2 shows the internal information of all the data. More specifically, they are collected and processed as detailed below:

Human Immunodeficiency Virus Infection (HIV-DTI & HIV-fMRI): Ragin et al. (2012) collected this raw data from two modalities, i.e., DTI and fMRI. The subject data contains 70 instances, where half the patients and half are healthy, with similar profiles in age, gender, education level, etc. Following Ma et al. (2017), we employ DPARSF (Yan & Zang, 2010) to preprocess the fMRI data and then perform Gaussian smoothing on the images. To eliminate noise and drift, we also apply linear trending and bandpass filtering techniques. Further, we divide any instance into 116 ROIs by AAL atlas (Tzourio-Mazoyer et al., 2002) and discard 26 of them. We then harvest the network initial weighting matrices for all subjects' brain instances. For DTI, we first utilize the FSL (Smith et al., 2004) with techniques such as noise filtering, image correction, etc. to process DTI data. Then we obtain the corresponding weighted matrices of brain networks with 90 regions.

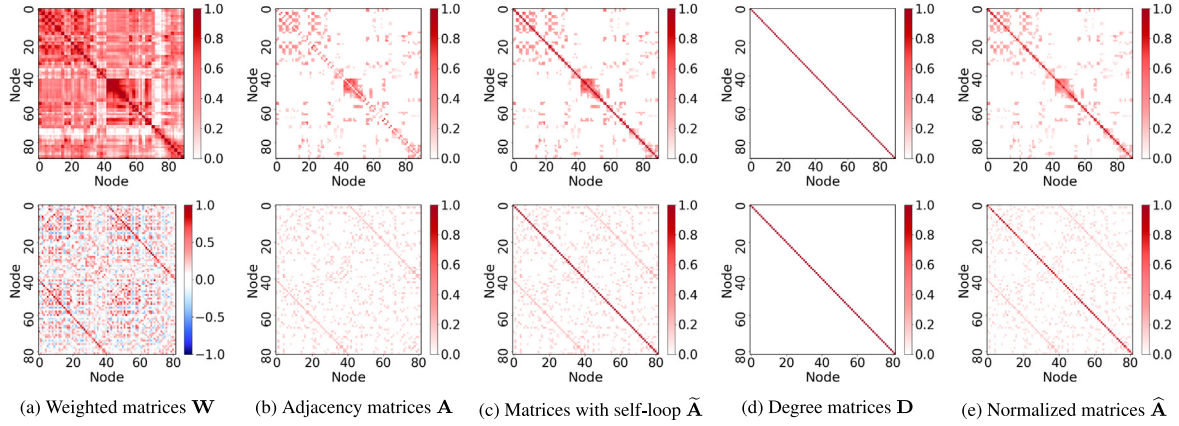
Bipolar Disorder (BP-DTI & BP-fMRI): The dataset, which also contains fMRI and DTI modalities, includes 45 healthy subjects and 52 bipolar patients with similar profiles (Cao et al., 2015). For fMRI, we employ the CONN (Whitfield-Gabrieli & Nieto-Castanon, 2012) to get the initial brain networks. Concretely, We first realign and co-register the original EPI images and then perform normalization and smoothing. After that, confounding influences caused by motion artifacts, cerebrospinal fluid, etc. will disappear. In the end, each initial connectome is calculated from the marked gray matter area. For the DTI data, we follow the data processing strategy in Ma et al. (2017) to generate brain networks whose regions are the same as those of the fMRI network.

Attention Deficit Hyperactivity Disorder (ADHD-fMRI) & Hyperactive Impulsive Disorder (HI-fMRI) & Gender (GD-fMRI): The initial data was constructed from the whole brain fMRI atlas (Craddock, James, Holtzheimer III, Hu, & Mayberg, 2012). Following the work in Pan, Wu, Zhu, Long, and Zhang (2016), we use the functional segmentation result CC200 from Craddock et al. (2012), which divides each instance into two hundred ROIs. In order to explore the relationship between ROIs, we record the average value of each ROI in a specific voxel time course. Similarly, we obtain the correlation between the two ROIs according to the Pearson correlation between the two time courses, and generate three reliable brain network instance sets with the threshold specified in Pan et al. (2016). More processing is stated in Craddock et al. (2012) and Pan et al. (2016).

Hearing Activity (HA-EEG): The raw electroencephalogram (EEG) data was recorded from 61 healthy adults using 62 electrodes (Hernandez-Perez et al., 2021). The participants were either actively listening to individual words over headphones (active condition) or watching a silent video and ignoring the speech (passive condition). To transform the dataset into a usable version, we perform source analysis using the fieldtrip toolkit (Oostenveld, Fries, Maris, & Schoffelen, 2011) with a cortical-sheet based source model and a boundary element head model. Specifically, we calculate the coherence of all sources and segment the sources based on the 68 regions of the Desikan–Killiany cortical atlas. Furthermore, we utilize the imaginary part of the coherence spectrum as the connectivity metric to reduce the effect of electric field spread (Nolte et al., 2004).

Table 2
Statistics of brain network datasets.

Dataset	BP-DTI	BP-fMRI	HIV-DTI	HIV-fMRI	ADHD-fMRI	HI-fMRI	GD-fMRI	HA-EEG
Network instances	70	70	97	97	83	79	85	61
Healthy/Male/Active	35	35	45	45	46	44	36	21
Patient/Female/Passive	35	35	52	52	37	35	49	40
Nodes × Nodes	82 × 82	82 × 82	90 × 90	90 × 90	200 × 200	200 × 200	200 × 200	68 × 68



(a) Weighted matrices $\mathbf{W}$ (b) Adjacency matrices $\mathbf{A}$ (c) Matrices with self-loop $\widetilde{\mathbf{A}}$ (d) Degree matrices $\mathbf{D}$ (e) Normalized matrices $\widehat{\mathbf{A}}$

Fig. 2. Examples of brain networks of two subjects in fMRI modality. The subject above is from HIV, while below one is from BP. From left to right are the initial weighted matrices **W**, the adjacency matrices **A** generated by the network building module (Eq. (10)), and the other matrices involved in the GCN aggregation process (Eq. (2)).

5.2. Baselines and settings

To evaluate BN-GNN, we compare it with multiple excellent baselines whose information is as follows:

DeepWalk & Node2Vec (Grover & Leskovec, 2016; Perozzi, Al-Rfou, & Skiena, 2014): The main idea of Deepwalk is to perform random walks in the network, then generate a large number of node sequences, further input these node sequences as samples into word2vec (Mikolov, Chen, Corrado, & Dean, 2013), and finally obtain meaningful node representation vectors. Compared with DeepWalk, Node2Vec balances the homophily and structural equivalence of the network through biased random walks. Both of them are commonly used baselines in network representation learning.

GCN & GAT (Kipf & Welling, 2017; Velickovic et al., 2018): Graph convolutional network (GCN) performs convolution aggregations in the graph Fourier domain, while graph attention network (GAT) performs aggregations in combination with the attention mechanism. Both of them are outstanding GNNs

GCN+skip & GAT+skip: Following Li et al. (2019), we construct GCN+skip and GAT+skip by adding residual skip-connections to GCN and GAT, respectively.

GraphSAGE & FastGCN (Chen, Ma, & Xiao, 2018; Hamilton et al., 2017): They are two improved GNN algorithms with different sampling strategies. For the sake of computational efficiency, GraphSAGE only samples a predefined number of neighbor nodes as objects to aggregate. Unlike GraphSAGE, which samples neighbor nodes, FastGCN samples all nodes, constructs a new topology based on the initial structure and encodes global information.

PR-GNN & GNEA & Hi-GCN (Bi et al., 2020; Jiang et al., 2020; Li et al., 2020): Three GNN-based baselines for brain network analysis, all of which contain methods for optimizing the initial brain network generated by neuroimaging technology. PR-GNN utilizes the regularized pooling layers to filter nodes in the network and uses GAT for feature aggregation. GNEA determines a constant number of neighbors for all nodes through the correlation coefficient in each brain network. Hi-GCN uses the eigenvector-based pooling layers EigenPooling to generate multiple coarse-grained sub-graphs from the initial network and then aggregates network information hierarchically and generates network representations.

SDBN (Wang et al., 2017): Instead of involving GNNs, it introduces convolutional neural networks (CNNs) (Krizhevsky, Sutskever, & Hinton, 2012) to perform connectome embedding for subjects' brain instances.

For settings, we complete BN-GNN with GCN and GAT, respectively, namely **BN-GCN and BN-GAT**. Moreover, we set the total number of timesteps t to 1000, the total number of all possible actions b to 3, the window size w of *REW* to 20, and the discount coefficient γ to 0.95. For the epsilon probability ϵ, we set it to decrease linearly in the first 20 timesteps, with a starting probability of 1.0 and an ending probability of 0.05. For all GNN-based methods, we use *ReLU* with a slope of 0.2 as the activation operator of feature aggregations and use dropout with a rate of 0.3 between every two adjacency neural networks. For a fair comparison, we use *Adam* optimizers with learning rates of 0.0005 and 0.005 to update the policy and GNN2. We set the network representation dimension of all methods to 128 and employ the strategies mentioned in the corresponding papers to adjust the parameters of baselines and show results with the best settings. Besides, we use the same data split ($|D_{trian}| : |D_{val}| : |D_{test}| = 8 : 1 : 1$) to repeat each experiment 10 times, where each experiment records the test result with the highest verification value within 100 epochs. All experiments are performed on the same server with two 20-core CPUs (126G) and an NVIDIA Tesla P100 GPU (16G).

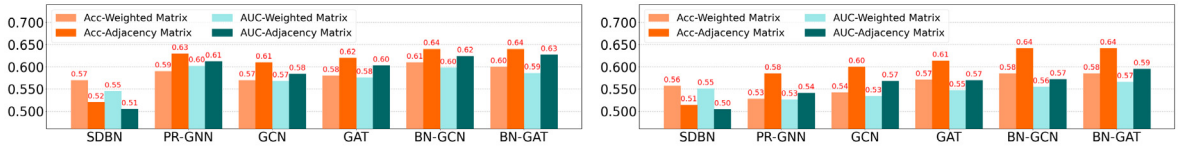
5.3. Model comparison (RQ1)

To compare the performance of all methods, we perform disease or gender prediction (i.e., brain network classification) tasks on eight real-world datasets. Moreover, average accuracy and AUC are utilized as measurement metrics. Considering that some baselines are challenging to deal with the initial weighted matrices of the brain networks that are almost complete graphs, we perform representation learning for all methods on adjacency matrices generated by the network building module. Taking

Table 3
Performance comparison of multiple algorithms on brain network classification tasks.

Method	Layers	BP-DTI	BP-fMRI	HIV-DTI	HIV-fMRI	ADHD-fMRI	HI-fMRI	GD-fMRI	HA-EEG
DeepWalk	–	0.520 ± 0.097	0.530 ± 0.134	0.514 ± 0.159	0.485 ± 0.130	0.512 ± 0.141	0.462 ± 0.148	0.550 ± 0.127	0.566 ± 0.270
Node2Vec	–	0.530 ± 0.110	0.550 ± 0.111	0.514 ± 0.145	0.500 ± 0.172	0.525 ± 0.075	0.475 ± 0.122	0.562 ± 0.170	0.583 ± 0.200
PR-GNN	2	0.590 ± 0.186	*0.630 ± 0.110*	0.557 ± 0.174	0.585 ± 0.100	*0.625 ± 0.167*	*0.600 ± 0.165*	0.600 ± 0.145	0.650 ± 0.157
GNEA	3	0.560 ± 0.149	0.600 ± 0.134	0.557 ± 0.118	0.585 ± 0.196	0.550 ± 0.127	0.562 ± 0.160	*0.612 ± 0.087*	0.633 ± 0.221
HI-GCN	3	0.540 ± 0.162	0.600 ± 0.109	0.528 ± 0.192	0.571 ± 0.127	0.562 ± 0.128	0.562 ± 0.160	0.587 ± 0.148	0.616 ± 0.183
GraphSAGE	2	*0.610 ± 0.192*	0.610 ± 0.113	0.571 ± 0.202	0.600 ± 0.124	0.575 ± 0.127	0.575 ± 0.100	0.600 ± 0.165	*0.716 ± 0.076*
FastGCN	2	0.590 ± 0.113	0.620 ± 0.140	*0.585 ± 0.134*	*0.628 ± 0.145*	0.612 ± 0.180	0.600 ± 0.145	0.600 ± 0.175	0.700 ± 0.194
BN-GNN	1∼3	**0.630 ± 0.167**	**0.640 ± 0.120**	**0.614 ± 0.111**	**0.642 ± 0.146**	**0.637 ± 0.130**	**0.612 ± 0.205**	**0.637 ± 0.141**	**0.733 ± 0.200**
Gain	–	**2.0** ↑	**1.0** ↑	**2.9** ↑	**1.4** ↑	**1.2** ↑	**1.2** ↑	**2.5** ↑	**1.7** ↑
GCN	1	0.560 ± 0.101	0.600 ± 0.148	0.542 ± 0.178	0.557 ± 0.174	0.562 ± 0.150	*0.587 ± 0.080*	*0.600 ± 0.122*	0.650 ± 0.189
GCN	2	*0.590 ± 0.130*	0.610 ± 0.122	0.571 ± 0.180	*0.600 ± 0.166*	*0.600 ± 0.165*	0.587 ± 0.125	0.600 ± 0.108	0.666 ± 0.129
GCN	3	0.540 ± 0.180	0.600 ± 0.184	0.542 ± 0.189	0.585 ± 0.118	0.525 ± 0.122	0.562 ± 0.187	0.575 ± 0.150	0.616 ± 0.106
GCN+skip	3	0.590 ± 0.164	*0.620 ± 0.116*	*0.585 ± 0.134*	0.528 ± 0.111	0.562 ± 0.170	0.575 ± 0.160	0.600 ± 0.183	*0.700 ± 0.163*
BN-GCN	1∼3	**0.610 ± 0.170**	**0.640 ± 0.120**	**0.614 ± 0.111**	**0.642 ± 0.172**	**0.637 ± 0.130**	**0.600 ± 0.165**	**0.625 ± 0.125**	**0.716 ± 0.197**
Gain	–	**2.0** ↑	**2.0** ↑	**2.9** ↑	**4.2** ↑	**3.7** ↑	**1.3** ↑	**2.5** ↑	**1.6** ↑
GAT	1	0.570 ± 0.141	*0.620 ± 0.172*	0.542 ± 0.261	0.571 ± 0.169	0.562 ± 0.128	0.575 ± 0.203	0.612 ± 0.189	0.650 ± 0.203
GAT	2	0.590 ± 0.130	0.610 ± 0.157	0.585 ± 0.149	*0.614 ± 0.111*	*0.600 ± 0.215*	*0.587 ± 0.137*	0.612 ± 0.152	0.683 ± 0.174
GAT	3	0.550 ± 0.128	0.610 ± 0.144	0.571 ± 0.202	0.585 ± 0.149	0.550 ± 0.127	0.575 ± 0.160	0.612 ± 0.171	0.666 ± 0.235
GAT+skip	3	*0.600 ± 0.184*	0.610 ± 0.083	*0.600 ± 0.153*	0.557 ± 0.162	0.575 ± 0.169	0.587 ± 0.185	*0.625 ± 0.111*	*0.700 ± 0.124*
BN-GAT	1∼3	**0.630 ± 0.167**	**0.640 ± 0.128**	**0.614 ± 0.181**	**0.642 ± 0.146**	**0.612 ± 0.180**	**0.612 ± 0.205**	**0.637 ± 0.141**	**0.733 ± 0.200**
Gain	–	**3.0** ↑	**2.0** ↑	**1.4** ↑	**2.8** ↑	**1.2** ↑	**2.5** ↑	**1.2** ↑	**3.3** ↑

The first part compares the experimental results of multiple methods, while the second and third parts refine the representation performance of GCNs and GATs with different layers. The bold and italicized values in each part represent the best and second-best results of all methods, respectively. ↑ indicates the improvement (%) of our BN-GNN compared to the best baseline of each part.



(a) BP-fMRI (b) HIV-fMRI

Fig. 3. Visualization of ablation experiments to verify the performance of the network building module on the two datasets.

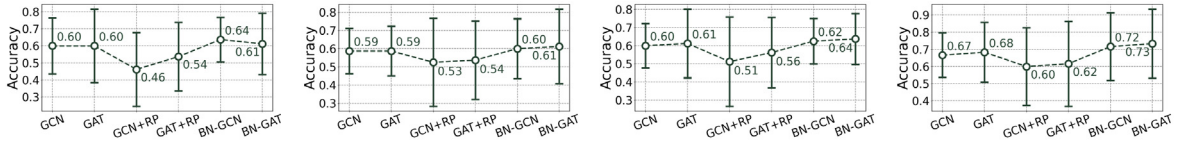
GCN as an example, Fig. 2 visualizes the transformation process of the adjacency matrices of two subjects from the HIV-fMRI and BP-fMRI. According to the values displayed in Table 3, five conclusions are reached:

(1) BN-GNN always obtains the highest average accuracy value on all datasets, proving that its brain network representation performance is superior to the baselines. Specifically, the classification accuracy of BN-GNN on eight datasets is on average about 2.0% higher than that of the sub-optimal algorithm. (2) All GNN-based methods outperform traditional network representation methods (i.e., DeepWalk and Node2Vec). This phenomenon is expected because the GNN architecture can better capture the local structural features in brain networks, which in turn yield more informative regional representations. In addition, in brain network classification tasks, end-to-end learning strategies in brain network classification tasks are often superior to unsupervised representation learning methods. (3) GAT-based methods are generally better than GCN-based methods. Compared with the latter, when layers are stacked over two, the performance of the former is usually not greatly degraded. This is because the attention mechanism included in GAT alleviates the over-smoothing problem on some datasets. (4) GNNs combined with skip-connections (i.e., GCN+skip and GAT+skip) do not always enable deeper neural networks to perform better. Compared with the best GCN and GAT models (in the last two parts of Table 3), BN-GCN and BN-GAT have an average accuracy improvement of 2.5% and 2.1% on eight classification tasks, respectively. Although these observations reveal the limitations of skip-connections, they also confirm the hypothesis of this work that different brain networks require different aggregation iterations. In other words, since the brain networks of real subjects are usually different, customizing different GNN architectures for different subjects is essential to improve network representation performance and provide therapeutic intervention. (5) Though GraphSAGE and FastGCN improve the efficiency or structure information mining capability of the original GCN, their performance is still inferior to our BN-GNN. This phenomenon indicates that searching suitable feature aggregation strategies for network instances in brain network analysis may be more important than exploring sampling or structural reconstruction strategies.

5.4. Ablation study (RQ2)

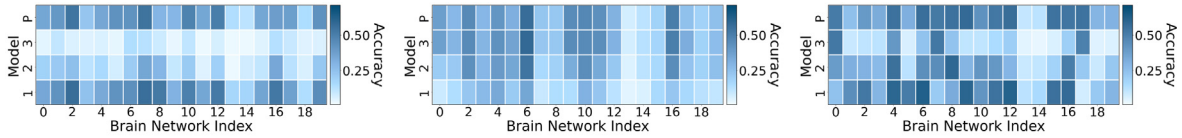
The classification results and analysis in Section 5.3 confirm the superiority of GNN-based methods in processing brain network data. Furthermore, we implement ablation studies to detect the independent influence of the network building and meta-policy modules contained in our proposed BN-GNN on the above classification tasks. Specifically, for the network building module, we compare the classification results based on the initial weighted matrices and processed adjacency matrices on BP-fMRI and HIV-fMRI, respectively. For the meta-policy module, we compare the performance of BN-GNN and random-policy-based GNN on four datasets. Furthermore, we show examples of practical use of our idea on various input types to see better how it works.

Fig. 3 visualizes the accuracy and AUC scores of ablation studies for network building, from which we can obtain three key observations: (1) Utilizing the adjacency matrices generated by the network building module to replace the initial weighted matrices greatly improves the classification performance of the GNN-based methods under the two metrics. This phenomenon indicates that our proposed network building module is beneficial to promote the application of GNN in brain network analysis studies. (2) The performance of SDBN on the initial matrices is



(a) ADHD-fMRI (b) HI-fMRI (c) GD-fMRI (d) HA-EEG

Fig. 4. Visualization of ablation experiments to verify the performance of the meta-policy module on four datasets, where GCN+RP and GAT+RP apply a random-policy instead of the meta-policy to guide the aggregation process of GCN and GAT.



(a) Input weighted matrices $\mathbf{W}$ (b) Input degree matrices $\mathbf{D}$ (c) Input normalized matrices $\widehat{\mathbf{A}}$

Fig. 5. Visualization of classification results for 20 brain networks randomly sampled from BP-DTI, where different figures use different types of inputs/adjacency matrices. From bottom to top, the *y*-axis represents the GCN model with *y* layers and the GCN model guided by our meta-policy. The *x*-axis represents the brain network index. The colors from light to dark mean the average accuracy from low to high.

better than that on the adjacency matrices. On the one hand, SDBN reconstructs the brain networks and reinforce the spatial structure induction of the initial weighted matrices, thereby enabling the CNN to capture the highly non-linear features. On the other hand, the sparse adjacency matrices generated by the network building module may not be suitable for CNN-based methods. Notably, the optimal results of SDBN are always inferior to that of BN-GNN, indicating that it is meaningful to learn topological brain networks based on GNN. (3) Even though GAT-based methods (including PR-GNN and GAT) use the attention technique to learn weights for different neighbors, they are still difficult to deal with densely connected initial brain networks. Thus, generating adjacency matrices (as shown in Fig. 2) for GNN can improve brain network representation learning.

We replace the meta-policy module in BN-GNN with a random-policy (randomly chooses an action for a given instance) to construct the baselines of ablation experiments, namely GCN+RP and GAT+RP. Fig. 4 illustrates the results of ablation experiments for the meta-policy module, from which two conclusions can be drawn: (1) The performance of GCN+RP and GAT+RP are worse than BN-GNN and original GNNs on ADHD-fMRI, HI-fMRI, GD-fMRI, and HA-EEG. (2) Our BN-GNN is better than the original GNNs. These phenomena once again imply that the introduction of our meta-policy can effectively improve the classification performance of brain networks.

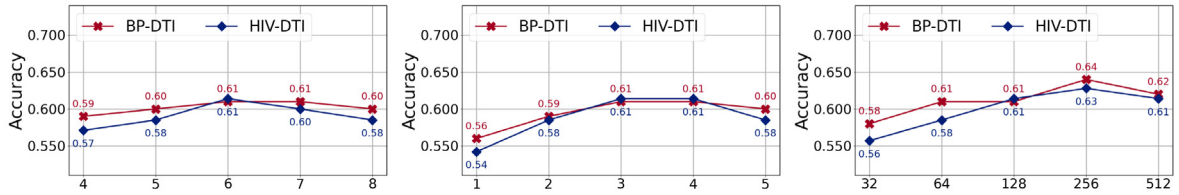
To explore the practical application of our idea on different input types, we illustrate the classification performance of layer-fixed GCNs and GCN guided by meta-policy on BP-DTI in Fig. 5. First, we observe Fig. 5(a) and find that single-layer GCNs generally perform best when using initial weighted matrices as inputs. Besides, GCN performance degrades drastically when its model stack increases. This may be because the initial brain networks usually have a high connection density (as shown in Fig. 2(a)), which leads to the over-smoothing problem of multi-layer GCNs during the learning process. Second, Fig. 5(b) shows the classification results with the degree matrices as inputs. Since the degree matrices do not encode neighbor information (as shown in Fig. 2(b)), these GCNs degenerate into fully-connected neural networks without feature aggregation. Therefore, the brain network classification performance of models with different numbers of layers varies relatively smoothly. Finally, after processing the initial data based on our network building module, the optimal number of GCN layers required for normalized matrices with reduced density becomes very different, as shown in Fig. 2(c). In addition, reliable brain networks improve the overall performance upper bound compared to the brain classification of the first two types of inputs. This again implies the importance of building reliable brain networks and customizing the optimal number of aggregations. In particular, our meta-policy is often able to find the optimal number of model layers corresponding to brain instances for three input types. Therefore, the GCN with our meta-policy (i.e., BN-GCN) generally performs the best (shown in the first row of subfigures in Fig. 2), even if the adjacency matrices have extremely large or small connection densities. In future research, the network building module can continue to serve brain network analysis. And the meta-policy may be extended to other scenarios with differentiated requirements for the number of model layers rather than being limited to the application of GNNs.

5.5. Hyperparameter analysis (RQ3)

We also explore the floating perturbations of key hyperparameters in three modules of BN-GNN, namely the neighbor pre-defined amount in the network building module, the action set's size in the meta-policy module, and the dimension of the network representation in the GNN module. Fig. 6(a) shows that increasing the number of neighbors (determined by k of KNN) when building the adjacency matrix does not always yield better network representations. The possible reason is that each region in the brain network only has meaningful connections with a limited number of neighbors. From Fig. 6(b), we observe that the performance of BN-GNN is often the best when the number of aggregations is 3. When the action space is further expanded, BN-GNN still maintains a relatively stable performance. These two phenomena verify that the best representation of most brain networks can be obtained within 3 aggregations, and BN-GNN is robust to the fluctuation of action set size (i.e., all possible actions, which is also the maximum aggregation iterations that may occur). The results in Fig. 6(c) show that unless the dimension is too small, the performance of BN-GNN will not fluctuate excessively by updating the embedding dimension.

6. Conclusions and future work

In this work, a GNN-based brain network representation framework, namely BN-GNN is proposed. In particular, BN-GNN combines DRL and GNN for the first time to achieve customized aggregation for different networks, effectively improving traditional GNNs in brain network representation learning. Experimental



(a) Number of Neighbors (b) Number of All Possible Actions (c) Dimension of Representations

Fig. 6. Hyperparameter sensitivity analysis of BN-GNN on the two datasets.

results imply that BN-GNN consistently outperforms state-of-the-art baselines on eight brain network disease analysis tasks. In future work, we will improve BN-GNN from both technical and practical aspects. Technically, we discuss the idea of automatically searching hyperparameters for our model BN-GNN as a future trend. Practically, we state the explainability of our model, which is an emerging area in brain network analysis. Specifically, even if we observe that changes in the number of neighbors have little effect on the performance of BN-GNN, setting this hyperparameter manually is not an optimal solution. Therefore, multi-agent reinforcement learning can be introduced into BN-GNN to automatically search for the optimal brain network structure and model layer number. Besides, for applications of brain disease analysis, the explainability of the classification results is often as important as the accuracy. In other words, while successfully predicting a damaged brain network, it is also necessary to understand which regions within the network are responsible for the damage. Therefore, to improve the explainability, better pooling or interpretation techniques like attention-based pooling and grad-cam, can be introduced to BN-GNN.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors of this paper were supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC, China through grants U20B2053 and 62073012, S&T Program of Hebei, China through grant 20310101D, Beijing Natural Science Foundation, China through grants 4202037 and 4222030. Philip S. Yu was supported by the NSF under grants III-1763325, III-1909323, and SaTC-1930941. We also thank CAAI-Huawei MindSpore Open Fund and Huawei MindSpore platform for providing the computing infrastructure.

References

Alexander, A. L., Lee, J. E., Lazar, M., & Field, A. S. (2007). Diffusion tensor imaging of the brain. *Neurotherapeutics*, *4*(3), 316–329.

Arslan, S., Ktena, S. I., Glocker, B., & Rueckert, D. (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* (pp. 3–13). Springer.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38.

Bi, X., Liu, Z., He, Y., Zhao, X., Sun, Y., & Liu, H. (2020). GNEA: A graph neural network with ELM aggregator for brain network classification. *Complexity*, *2020*.

Braak, H., & Braak, E. (1991). Neuropathological stageing of Alzheimer-related changes. *Acta Neuropathologica*, *82*(4), 239–259.

Cao, B., He, L., Wei, X., Xing, M., Yu, P. S., Klumpp, H., et al. (2017). t-bne: Tensor-based brain network embedding. In *International conference on data mining* (pp. 189–197). SIAM.

Cao, B., Zhan, L., Kong, X., Yu, P. S., Vizueta, N., Altshuler, L. L., et al. (2015). Identification of discriminative subgraph patterns in fMRI brain networks in bipolar affective disorder. In *International conference on brain informatics and health* (pp. 105–114). Springer.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 34*, (pp. 3438–3445).

Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International conference on learning representations*.

Craddock, R. C., James, G. A., Holtzheimer III, P. E., Hu, X. P., & Mayberg, H. S. (2012). A whole brain fMRI atlas generated via spatially constrained spectral clustering. *Human Brain Mapping*, *33*(8), 1914–1928.

Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., & Yu, P. S. (2020). Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the ACM international conference on information knowledge management* (pp. 315–324).

Gao, H., & Ji, S. (2019). Graph u-nets. In *International conference on machine learning* (pp. 2083–2092). PMLR.

Gao, Y., Yang, H., Zhang, P., Zhou, C., & Hu, Y. (2020). Graph neural architecture search. In *International joint conference on artificial intelligence*: *Vol. 20*, (pp. 1403–1409).

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery data mining* (pp. 855–864).

Gurbuz, M. B., & Rekik, I. (2021). MGN-Net: A multi-view graph normalizer for integrating heterogeneous biological network populations. *Medical Image Analysis*, *71*, Article 102059.

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NeurIPS* (pp. 1025–1035).

Hernandez-Perez, H., Mikiel-Hunter, J., McAlpine, D., Dhar, S., Boothalingam, S., Monaghan, J. J. M., et al. (2021). Understanding degraded speech leads to perceptual gating of a brainstem reflex in human listeners. *PLOS Biology*, *19*, 1–37.

Huang, Y., & Mucke, L. (2012). Alzheimer mechanisms and therapeutic strategies. *Cell*, *148*(6), 1204–1222.

Huettel, S. A., Song, A. W., McCarthy, G., et al. (2004). *Functional magnetic resonance imaging*, *Vol. 1*. MA: Sinauer Associates Sunderland.

Jiang, H., Cao, P., Xu, M., Yang, J., & Zaiane, O. (2020). Hi-GCN: A hierarchical graph convolution network for graph embedding learning of brain network and brain disorders prediction. *Computers in Biology and Medicine*, *127*, Article 104096.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105.

Ktena, S. I., Parisot, S., Ferrante, E., Rajchl, M., Lee, M., Glocker, B., et al. (2018). Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, *169*, 431–442.

Lai, K. -H., Zha, D., Zhou, K., & Hu, X. (2020). Policy-GNN: Aggregation optimization for graph neural networks. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery data mining* (pp. 461–471).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Li, G., Muller, M., Thabet, A., & Ghanem, B. (2019). Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9267–9276).

Li, X., Zhou, Y., Dvornek, N. C., Zhang, M., Zhuang, J., Ventola, P., et al. (2020). Pooling regularized graph neural network for fmri biomarker analysis. In *International conference on medical image computing and computer-assisted intervention* (pp. 625–635). Springer.

Liu, Y., He, L., Cao, B., Yu, P. S., Ragin, A. B., & Leow, A. D. (2018). Multi-view multi-graph embedding for brain network clustering analysis. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 32*, (pp. 117–124). AAAI Press.

Liu, F., Xue, S., Wu, J., Zhou, C., Hu, W., Paris, C., et al. (2020). Deep learning for community detection: Progress, challenges and opportunities. In *International joint conference on artificial intelligence* (pp. 4981–4987).

Ma, G., Ahmed, N. K., Willke, T. L., Sengupta, D., Cole, M. W., Turk-Browne, N. B., et al. (2019). Deep graph similarity learning for brain data analysis. In *Proceedings of the ACM international conference on information and knowledge management* (pp. 2743–2751).

Ma, G., He, L., Lu, C. -T., Shao, W., Yu, P. S., Leow, A. D., et al. (2017). Multi-view clustering with graph embedding for connectome analysis. In *Proceedings of the ACM international conference on information knowledge management* (pp. 127–136).

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q. Z., et al. (2021). A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *International conference on learning representations*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Nishi, T., Otaki, K., Hayakawa, K., & Yoshimura, T. (2018). Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *International conference on intelligent transportation systems* (pp. 877–883). IEEE.

Nolte, G., Bai, O., Wheaton, L., Mari, Z., Vorbach, S., & Hallett, M. (2004). Identifying true brain interaction from EEG data using the imaginary part of coherency. *Clinical Neurophysiology*, *115*(10), 2292–2307.

Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. In *International conference on learning representations*.

Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J. -M. (2011). FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. *Computational Intelligence and Neuroscience*, *2011*.

Pan, S., Wu, J., Zhu, X., Long, G., & Zhang, C. (2016). Task sensitive feature exploration and learning for multitask graph classification. *IEEE Transactions on Cybernetics*, *47*(3), 744–758.

Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., et al. (2018). Disease prediction using graph convolutional networks: Application to autism spectrum disorder and Alzheimer's disease. *Medical Image Analysis*, *48*, 117–130.

Peng, H., Li, J., Gong, Q., Ning, Y., Wang, S., & He, L. (2020). Motif-matching based subgraph-level attentional convolutional network for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 34*, (pp. 5387–5394).

Peng, H., Zhang, R., Dou, Y., Yang, R., Zhang, J., & Yu, P. S. (2021). Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Transactions on Information Systems*, 1–46.

Peng, H., Zhang, R., Li, S., Cao, Y., Pan, S., & Yu, P. (2022). Reinforced, incremental and cross-lingual event detection from social messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery data mining* (pp. 701–710).

Ragin, A. B., Du, H., Ochs, R., Wu, Y., Sammet, C. L., Shoukry, A., et al. (2012). Structural brain alterations can be detected early in HIV infection. *Neurology*, *79*(24), 2328–2334.

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E., Johansen-Berg, H., et al. (2004). Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage*, *23*, S208–S219.

Sun, Q., Li, J., Peng, H., Wu, J., Ning, Y., Yu, P. S., et al. (2021). SUGAR: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *Proceedings of the web conference* (pp. 2081–2091).

Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., et al. (2002). Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *Neuroimage*, *15*(1), 273–289.

Urbanski, M., De Schotten, M. T., Rodrigo, S., et al. (2008). Brain networks of spatial awareness: Evidence from diffusion tensor imaging tractography. *Journal of Neurology Neurosurgery and Psychiatry*, *79*(5), 598–601.

Van Den Heuvel, M. P., & Pol, H. E. H. (2010). Exploring the brain network: A review on resting-state fMRI functional connectivity. *European Neuropsychopharmacology*, *20*(8), 519–534.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 30*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations*.

Wang, S., He, L., Cao, B., Lu, C. -T., Yu, P. S., & Ragin, A. B. (2017). Structural deep brain network mining. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery data mining* (pp. 475–484).

Whitfield-Gabrieli, S., & Nieto-Castanon, A. (2012). Conn: A functional connectivity toolbox for correlated and anticorrelated brain networks. *Brain Connectivity*, *2*(3), 125–141.

Xing, X., Li, Q., Yuan, M., Wei, H., Xue, Z., Wang, T., et al. (2021). DS-GCNs: Connectome classification using dynamic spectral graph convolution networks with assistant task training. *Cerebral Cortex*, *31*(2), 1259–1269.

Yan, Z., Ge, J., Wu, Y., Li, L., & Li, T. (2020). Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. *IEEE Journal on Selected Areas in Communications*, *38*(6), 1040–1057.

Yan, C., & Zang, Y. (2010). DPARSF: A MATLAB toolbox for "pipeline" data analysis of resting-state fMRI. *Frontiers in Systems Neuroscience*, *4*, 13.

Zha, D., Lai, K. -H., Zhou, K., & Hu, X. (2019). Experience Replay Optimization. In *International joint conference on artificial intelligence*.

Zhang, X., He, L., Chen, K., Luo, Y., Zhou, J., & Wang, F. (2018). Multi-view graph convolutional network and its applications on neuroimage analysis for Parkinson's disease. In *AMIA annual symposium proceedings*: *Vol. 2018*, (pp. 1147–1156). American Medical Informatics Association.

Zhang, Y., & Huang, H. (2019). New graph-blind convolutional network for brain connectome data analysis. In *International conference on information processing in medical imaging* (pp. 669–681). Springer.

Zhong, P., Wang, D., & Miao, C. (2020). EEG-based emotion recognition using regularized graph neural networks. *IEEE Transactions on Affective Computing*, 1.