Hindawi
Complexity
Volume 2020, Article ID 8813738, 11 pages
https://doi.org/10.1155/2020/8813738

Research Article

GNEA: A Graph Neural Network with ELM Aggregator for Brain Network Classification

Xin Bi,[1] Zhixun Liu,[2] Yao He,[2] Xiangguo Zhao,[3] Yongjiao Sun,[3] and Hao Liu[4]

[1]*Key Laboratory of Ministry of Education on Safe Mining of Deep Metal Mines, Northeastern University, Shenyang 110819, China*
[2]*College of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110819, China*
[3]*School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China*
[4]*Tencent, Guangzhou 510000, China*

Correspondence should be addressed to Xin Bi; bixin@mail.neu.edu.cn

Received 28 September 2020; Revised 14 October 2020; Accepted 16 October 2020; Published 29 October 2020

Academic Editor: Jianxin Li

Brain networks provide essential insights into the diagnosis of functional brain disorders, such as Alzheimer's disease (AD). Many machine learning methods have been applied to learn from brain images or networks in Euclidean space. However, it is still challenging to learn complex network structures and the connectivity of brain regions in non-Euclidean space. To address this problem, in this paper, we exploit the study of brain network classification from the perspective of graph learning. We propose an aggregator based on extreme learning machine (ELM) that boosts the aggregation ability and efficiency of graph convolution without iterative tuning. Then, we design a graph neural network named GNEA (Graph Neural Network with ELM Aggregator) for the graph classification task. Extensive experiments are conducted using a real-world AD detection dataset to evaluate and compare the graph learning performances of GNEA and state-of-the-art graph learning methods. The results indicate that GNEA achieves excellent learning performance with the best graph representation ability in brain network classification applications.

1. Introduction

In recent years, researchers have generated functional brain networks from resting-state functional magnetic resonance imaging (rs-fMRI) data [1]. The brain network has given researchers the possibility of analyzing brain regions and the connections among them. However, most brain network analysis methods [2, 3] learn from either manually extracted shallow features or deep features in Euclidean space. There is still an urgent demand for incorporating graph learning methods into brain analysis and disease detection. Exploiting the graph structure and connectivity among brain regions thoroughly can significantly improve the comprehensiveness and depth of brain analyses. Thus, in this paper, we leverage graph learning methods to study the brain network classification problem for the detection of Alzheimer's disease.

The graph neural network (GNN) has become one of the most popular graph representation and learning methods. Sperduti and Starita first applied neural networks to graphs in [4], which motivated the initial outline [5] and detailed description [6] of GNN. But these early networks are based on recurrent neural networks (RNNs) [7], which are computationally expensive. Then, the concept of convolution with graph data is studied. Bruna et al. first developed a graph convolution based on the spectral graph theory in [8], and they followed by applying improvements and extensions, for example, ChebNet [9] and GCN [10]. In general, spectral methods face high computational costs due to eigendecomposition [11]. On the other hand, spatial convolution was first studied in [12], in which the NN4G network performed sum aggregation on neighbor information directly. In follow-up works, GraphSAGE [13] adopted sampling strategy to improve the convolutional efficiency, GAT [14] adopted an attention mechanism to learn edge weights, and CGMM [15] studied probabilistic interpretability based on NN4G.

Although several recent works have applied GNNs to brain-related problems, there are still many remaining open problems. Ktena et al. [16] studied similarity metric learning for brain networks, but the classification performance on brain networks was not explored. Rajchl et al. [17] realized the prediction of spectrum disorders and AD by running node classification on a patient network, in which each node denoted a patient. However, this work focused on disease prediction among patients, and the brain connectivity of each individual patient was not studied. Lee and Huang [18] applied GCN to study the brain connectome, but due to the unknown precise structure of the graph, the graph learning procedure relied on iterative graph generation and was thus very time-consuming.

In this paper, to improve the efficiency of graph learning, we propose a graph learning neural network named GNEA (Graph Neural Network with ELM Aggregator) for the graph classification problem. The graph learning procedure of GNEA is presented in Figure 1. The aggregation performance is boosted by an aggregator based on extreme learning machine (ELM) [19, 20]. The extremely fast training speed and good generalization performance of ELM have been proven in various applications, for example, time-series learning [21–23], text mining [24, 25], biomedical data analysis [26–29], graph classification [29, 30], and game strategy [31]. The ELM aggregator learns a more complex aggregation function than those of other aggregators, which provides an extremely fast learning speed and a powerful aggregation ability.

The contributions of this paper are summarized as follows:

(1) An ELM-based aggregator is proposed, which achieves high aggregation ability and training efficiency.

(2) A graph learning neural network named GNEA is designed, which possesses a powerful learning ability for graph classification tasks.

(3) We apply GNEA to a real-world brain network classification problem to verify its ability to perform graph representation learning and classification.

The remainder of this paper is organized as follows. Section 2 introduces the brain network. Following the framework overview of GNEA in Section 3, Section 4 presents the graph convolution of GNEA, including propagation based on correlation-biased sampling and aggregation based on ELM. Section 5 presents the results of the performance evaluation, comparison, and discussion. Section 6 concludes this paper.

2. Functional Brain Network

A functional brain network is represented as a graph, where each node denotes a brain region and each edge denotes the functional connection between two brain regions [32].

A 4-dimensional fMRI image is a sequence of 3-dimensional brain models, which themselves are sequences of 2-dimensional brain image slices. The 4th dimension is the temporal dimension. The brain regions of each 3-dimensional brain model are mapped using an Atlas. For example, the automated anatomical labeling (AAL) Atlas maps 116 brain regions in the rs-fMRI images. With extracted brain regions, according to the correlation coefficients along the 4th dimension, the connectivity of each pair of brain regions can be estimated. The Pearson correlation coefficient is one of the most popular statistics for measuring the linear correlation between two normally distributed variables [33]. The Pearson correlation coefficient of two brain regions x and y is calculated as

$$\begin{aligned} r_p &= \frac{\mathrm{cov}(x, y)}{\sigma_x \sigma_y} \\ &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \end{aligned} \tag{1}$$

where x and y have the same length n, $\mathrm{cov}(x, y)$ is the covariation of x and y, σ_x and σ_y are the standard deviations of x and y, and $\overline{x}$ and $\overline{y}$ are the mean values of x and y.

The calculated correlation coefficient between each pair of brain regions indicates the weight of the edge between these two nodes in the brain network. Since a brain network is theoretically a sparse graph with dense local connectivity [34], a weight threshold is set to remove irrelevant edges from the complete graph of the brain network. The edges with lower weights than the threshold are considered inactive connections between brain regions.

An example of rs-fMRI and the corresponding functional brain network is presented in Figure 2. In subfigure Figure 2(a), the original rs-fMRI images are presented in three cuts, namely, frontal, axial, and lateral cuts. Figure 2(b) presents the three cuts of the AAL Atlas, which maps 116 brain regions in the rs-fMRI images. Figure 2(c) is the graph matrix generated from the AAL-mapped brain regions and calculated correlation coefficients. Figure 2(d) presents the final generated brain network according to the graph matrix with a determined weight threshold.

3. Framework Overview of GNEA

To boost the performance of brain network classification by GNNs, we improve the graph learning ability by designing a graph learning model named GNEA, which learns graph embeddings with graph convolution based on the ELM aggregator. A brain network is represented in a graph format, where each node denotes a brain region and each edge denotes a strong connection between two brain regions. The edge weight represents the correlation coefficient. A brain network consists of three matrices, namely, the adjacency matrix $\mathbf{A}$, the node embedding matrix $\mathbf{X}$, and the correlation coefficient matrix $\mathbf{C}$. The training data and targets of the ELM aggregator are generated from the graphs. With the trained aggregator, three layers of graph convolution are applied to update the node embeddings by propagation and aggregation based on the structures of the graphs. With the learned embeddings of nodes and graphs, the fully connected layer and *Softmax* output the

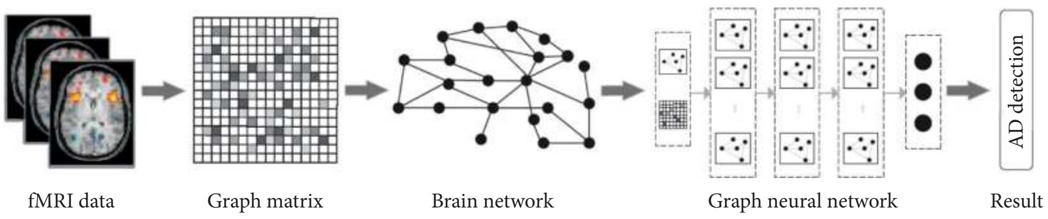


FIGURE 1: Graph learning for brain network classification for AD detection.

classification results of the input graph. The structure of GNEA is presented in Figure 3.

GNEA accepts three graph-formatted matrices as inputs. The adjacency matrix A represents the connectivity of the brain network, where element $\mathbf{A}_{ij} = 1$ indicates a connection between the corresponding i^{th} and j^{th} nodes, while element $\mathbf{A}_{ij} = 0$ indicates a disconnection. The embedding matrix $\mathbf{X}$ concatenates all the node embeddings by rows. The correlation coefficient matrix $\mathbf{C}$ keeps all the weights of the edges. In the case of the brain network, each weight $\mathbf{C}_{ij}$ is the calculated Pearson correlation coefficient between the i^{th} and j^{th} brain regions. Note that we keep both $\mathbf{A}$ and $\mathbf{C}$ instead of a single weighted adjacency matrix, so that all the original correlation coefficients between brain regions can be stored.

Three layers of graph convolution are conducted sequentially to learn both the semantic and structural feature embeddings. For each node v, the graph convolution collects information from the neighbor node set $N(v)$ and disparate sampling node set $S(v)$. Then the collected information is aggregated by the pretrained ELM aggregator.

The convoluted node embeddings of each convolutional layer are activated by the activation function ReLU. The sizes of the node embeddings are reduced by graph pooling. The graph embedding generated from the node embeddings by the readout operation is flattened into a vector and classified by a fully connected layer. Then, the activation function *Softmax* is applied to transform the neural distribution outputs into a final set of class labels for the input graph.

4. Graph Convolution

Graph convolution, as the key module of learning embeddings in GNEA, consists of two major operations, namely, propagation and aggregation. In this section, we propose our sampling-based propagation method and ELM aggregator, following an overview of graph convolution in GNEA.

4.1. Spatial Convolution.

A graph is defined as $G = (V, E)$, where V denotes the node set and $E \subseteq V \times V$ denotes the edge set. In the case of an undirected graph, such as brain network, the condition $(i, j) \subset E$ iff $(j, i) \subset E$ holds. Graph is often represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V \times V|}$.

In a spectral convolutional solution, the normalized graph Laplacian L is defined as

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-(1/2)}\mathbf{A}\mathbf{D}^{-(1/2)}, \tag{2}$$

where $\mathbf{D}$ is the diagonal degree matrix and $\mathbf{I}$ is the identity matrix. The representation $\mathbf{H} \in \mathbb{R}^{N \times C}$ of the $(l+)^{\text{th}}$ layer with C input channels and F filters can be calculated as

$$\mathbf{H}^{(l+1)} = \sigma\left(\widetilde{D}^{-(1/2)}\widetilde{A}\widetilde{D}^{-(1/2)}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{C \times F}$ is a matrix of filter parameters. However, this spectral convolution requires a stable and complete matrix $\mathbf{A}$ and has a computation cost of $O(|\varepsilon|FC)$.

Different from aggregating information from the spectral perspective, the process of spatial convolution involves aggregating a node's information with the information of its spatial neighbors. An increase in the number of layers of spatial convolutions results in the propagation of more information from further neighbor nodes. The spatial convolution used to learn the node representations is described in Algorithm 1.

Given a graph G, the graph convolution algorithm first generates the adjacency matrix $\mathbf{A}$, node embedding matrix $\mathbf{X}$, and correlation coefficient matrix $\mathbf{C}$. The number of sampling operations m and the number of graph convolutional layers K are also accepted as hyperparameters. The input embedding $\mathbf{X}$ is taken as the initialized embedding $\mathbf{H}^0$ (Line 1).

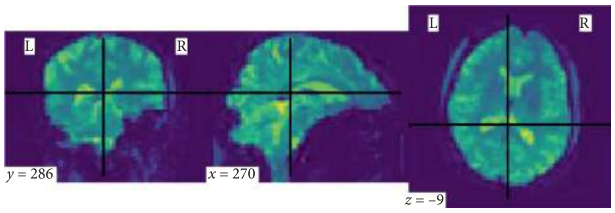
Iteration through the K aggregators (Line 2) can be viewed as propagation through the K layers of spatial convolutions. In each layer, for each node $v \in V$ (Line 3), the function $\phi(v, m, \mathbf{A}, \mathbf{C})$ returns the top-m neighbors of node v according to the adjacency matrix A and the correlation coefficients $\mathbf{C}$ (Line 4).

The aggregator function $f^k_{\text{agg}}(\cdot)$ in the k^{th} spatial convolution layer aggregates the embeddings of all the sampled neighbor embeddings. Then, it generates an aggregated neighbor embedding $\mathbf{h}^k_{N(v)}$ (Line 5). Then, the embedding of node v and the aggregated neighbor embedding are concatenated as $[\mathbf{h}^{k-1}_v, \mathbf{h}^k_{N(v)}]$. Dimension reduction is done by calculating the elementwise average values of the concatenated embeddings (Line 6). This graph convolution algorithm finally returns $\mathbf{H}^K$, which includes the concatenated embeddings in the K^{th} layer of every node (Line 8).

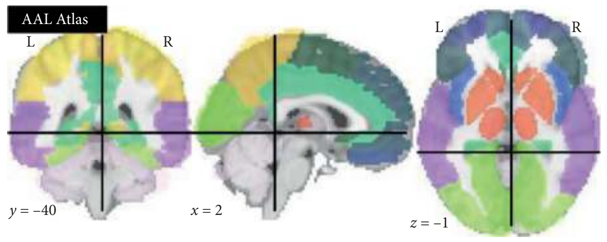
In the following subsections, we present the two major operations in the graph convolution of GNEA, our sampling-based propagation method, and the ELM aggregator.

4.2. Propagation Based on Correlation-Biased Sampling.

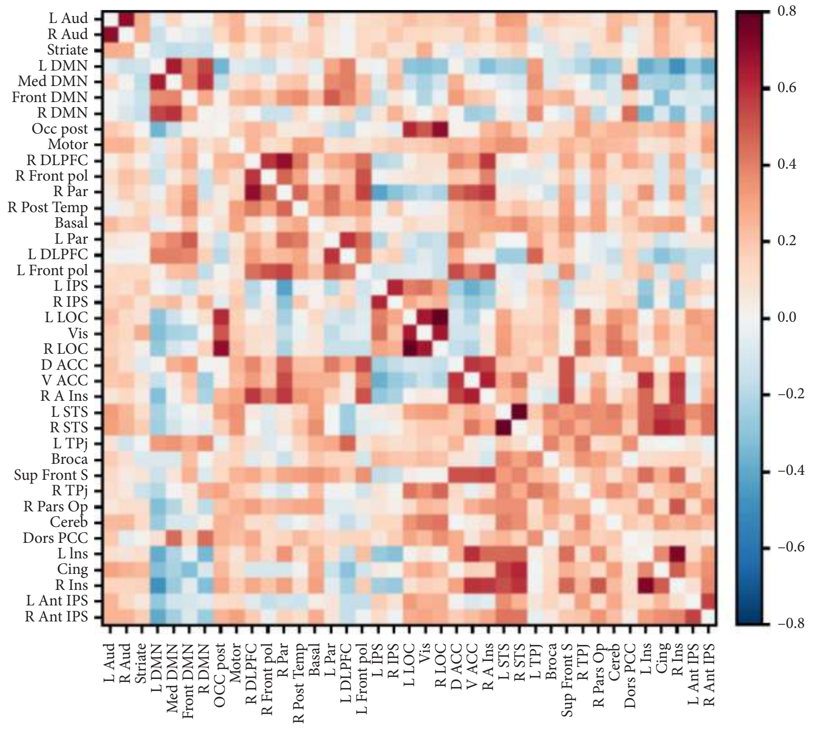
Previous graph learning methods aggregated node embeddings from the entire graph. In our sampling-based propagation method, to maintain the stability of propagation, we



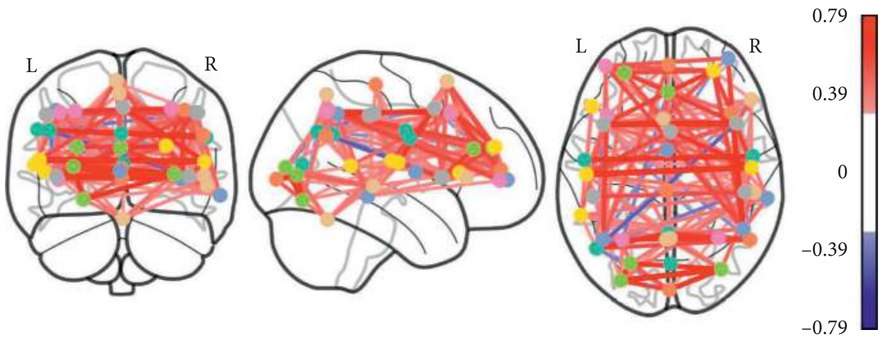
(a)



(b)



(c)



(d)

Figure 2: An example of a functional brain network. (a) Original rs-fMRI images. (b) The AAL Atlas. (c) Generated graph matrix. (d) Generated functional brain network.

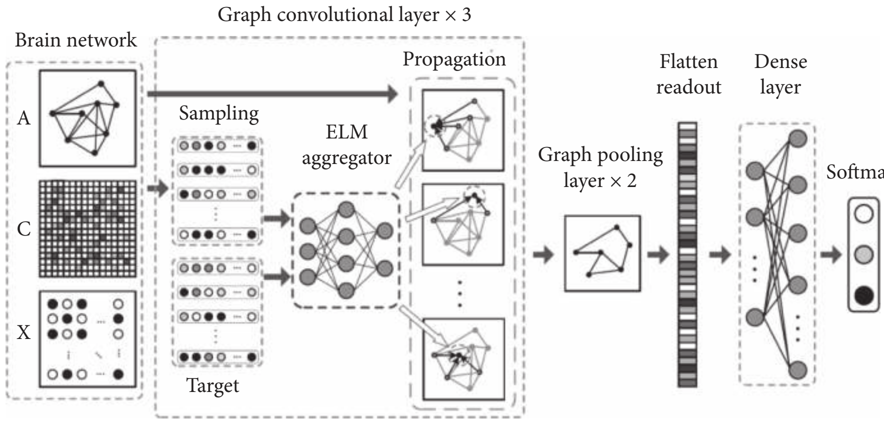


Figure 3: Architecture of GNEA for brain network classification.

```
Input: Adjacency matrix $\mathbf{A}$, node embedding matrix $\mathbf{X}$, correlation coefficient matrix $\mathbf{C}$, sample number $m$, number of graph
  convolutional layers $K$
  Output: Convoluted nodes embedding $\mathbf{H}^K$
(1) $\mathbf{H}^0 \leftarrow \mathbf{X}$
(2) for $k \in 1, \dots, K$ do
(3)     $N'(V) \leftarrow \phi(V, m, A, C)$;
(4)     for $v \in V$ do
(5)         $\mathbf{h}^k_{N'(v)} \leftarrow f^k_{agg}(\{\mathbf{h}^{k-1}_u, \forall u \in N'(v)\})$;
(6)         $\mathbf{h}^k_v \leftarrow \text{AVG}([\mathbf{h}^{k-1}_v, \mathbf{h}^k_{N'(v)}], \text{axis} = 1)$;
(7)     $\mathbf{H}^K = \text{concat}[\mathbf{h}^k_v | \forall v \in V]$;
(8) Return $\mathbf{H}^K$.
```

Algorithm 1: Graph convolution of GNEA.

sample m nodes from the neighbors of each node. Since the sampling is an operation biased by the correlation coefficients, we sample m neighbors with the highest relevance according to the correlation coefficient matrix $\mathbf{C}$. The sampling function $\phi(\cdot)$ (Line 4 in Algorithm 1) is described as

$$\phi(v, m, \mathbf{A}, \mathbf{C}) = \{u_i | u_i \in N_A(v), \quad i = 1, \dots, m, \forall u_j \notin N'(v), C_{ui} > C_{uj}\}, \tag{4}$$

where v is the current node for the embedding update, m is the sample number, $\mathbf{A}$ is the adjacency matrix of the current graph for retrieving neighbor information, and $\mathbf{C}$ is the correlation coefficients matrix as the sampling bias criterion. This sampling function retrieves m nodes $\{u_i, i = 1, \dots, m\}$ from the neighbors $N_A(v)$ of node v, guaranteeing that the correlation coefficients between node v and these m neighbors are the top-m most relevant neighbors of node v.

Note that if the number of neighboring nodes is less than m, the collected node embeddings are zero-padded. Then, the GNEA network tunes the weight matrices W for all the K aggregators and other trainable parameters by using stochastic gradient descent.

4.3. Aggregation Based on ELM

4.3.1. Symmetry Aggregators.

4.3.1. Symmetry Aggregators. Since the neighbor nodes $N(v)$ of node v have no ordering in the non-Euclidean space, the aggregator function of the spatial convolution process should be symmetric to maintain invariance to permutations of the input vectors of representations, for example, the *mean aggregator* and p*ooling aggregator* in GraphSAGE [13].

(1) Mean Aggregator. Taking the elementwise mean of the embedding tensors is nearly equivalent to the convolutional propagation rule used in transductive spectral convolution [13]. The mean aggregator applies the mean operator to the concatenated embeddings of both node v and the propagated neighbor embeddings, that is, $\mathbf{h}^{k-1}_v$ and $\mathbf{h}^{k-1}_{N(v)}$. Thus, the kth mean aggregator function $f^k_{\text{agg-mean}}$ is written as

$$f^k_{\text{agg-mean}} = \sigma\left(W^k \cdot \text{MEAN}\left(\left[\mathbf{h}^{k-1}_v, \mathbf{h}^{k-1}_{N(v)}\right]\right)\right). \tag{5}$$

(2) Pooling Aggregator. The pooling aggregator of a graph convolutional network takes the average or maximum element out of an embedding. It was found in [13] that there is no significant difference between max-pooling and mean-

pooling in practice. Thus, we apply only the max-pooling strategy to realize the pooling aggregator of our rival methods. By performing the max-pooling operation on each of the activated and weighted features, the aggregator is able to capture different aspects of the neighbor nodes. The k^{th} pooling aggregator function $f^{k}_{\text{agg-pool}}$ is described as

$$f^{k}_{\text{agg-pool}} = \text{MAX}\left(\mathbf{h}^{k}_{u \in N_v}\right) = \text{MAX}\left(\sigma\left(\mathbf{W}^{k}\mathbf{h}^{k-1}_{u \in N(v)} + b\right)\right). \quad (6)$$

4.3.2. The ELM Aggregator. The trainable parameters of symmetry aggregators are learned through backpropagation iterations according to the total loss. The aggregator must be tuned along with the other trainable parameters of graph neural networks. However, for a neural network-based aggregator, the weighted mapping of the embeddings can be performed by input mapping within the network.

Thus, to boost both the learning efficiency and performance of the aggregator, we propose an aggregator based on extreme learning machine [19, 20]. The ELM aggregator is capable of powerful aggregation and efficient training because it avoids iteratively tuning the weights with the total loss. In each graph convolutional layer, the ELM aggregator learns an ELM feature mapping from the neighboring embedding space into the aggregated embedding space of the central node.

The ELM aggregator is presented in Figure 4. Given N arbitrary samples $\{x_i, t_i\} \in \mathbb{R}^{n\times m}$, the ELM feature mapping matrix $\mathbf{H}_{\text{ELM}}$ is calculated as

$$\mathbf{H}_{\text{ELM}} = \begin{bmatrix} g(\boldsymbol{\omega}_1, b_1, \mathbf{x}_1) & \cdots & g(\boldsymbol{\omega}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ g(\boldsymbol{\omega}_1, b_1, \mathbf{x}_N) & \cdots & g(\boldsymbol{\omega}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N\times L}, \quad (7)$$

where L is the number of hidden layer nodes, ω_i is the input weight vector from the input nodes to the i^{th} hidden node, and b_i is the bias of i^{th} hidden node. $g(\omega_i, b_i, \mathbf{x})$ is the activation function that generates mapping neurons, and it can be any nonlinear piecewise-continuous functions.

The ELM aims to minimize the training error and the norm of the output weights. Thus, the output weight matrix $\boldsymbol{\beta}$ can be calculated as

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}_{\text{ELM}}\mathbf{T}, \quad (8)$$

where $\mathbf{H}^{\dagger}_{ELM}$ is the Moore-Penrose inverse of $\mathbf{H}_{\text{ELM}}$ and $\mathbf{T}$ is the training target. The training procedure of the ELM is presented in Algorithm 2.

During the feedforward phase, the aggregation result of the ELM aggregator is calculated as

$$f^{k}_{\text{ELM}} = \sigma\left(\boldsymbol{\omega}^{k}\mathbf{h}^{k-1}_{v} + b\right)\boldsymbol{\beta}^{k}. \quad (9)$$

4.3.3. Supervised Learning of the ELM Aggregator. The ELM aggregator is trained with supervised learning. The target $\mathbf{T}$ should be specified before the training procedure. However, only the target labels of the downstream task, that is, the labels for graph classification, are acknowledged. Thus, we obtain $\mathbf{T}$ according to the total loss of GNEA, which is calculated using categorical cross-entropy loss.

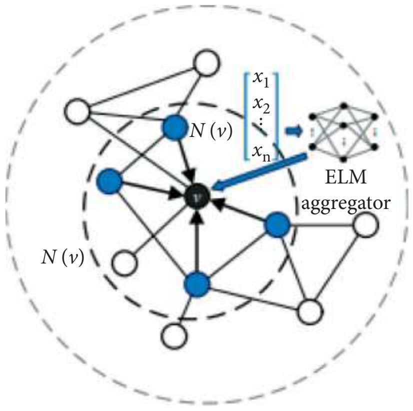


Figure 4: Aggregation using ELM aggregator on node$v$.

```
Input: Number of hidden nodes $L$
    Output: Output weight matrix $\boldsymbol{\beta}$
(1) for $i = 1$ to $s$ do
(2)     Initiate input weight $\omega_i$ randomly;
(3)     Initiate bias $b_i$ randomly;
(4) Calculate ELM feature mapping matrix $\mathbf{H}_{\text{ELM}}$;
(5) Calculate output weight matrix $\boldsymbol{\beta}$;
```

Algorithm 2: ELM.

The ELM target matrix $\mathbf{T}$ is first initialized randomly. Each row in $\mathbf{T}$ denotes the target embedding of a node. Then the corresponding targets of the sample graph are updated according to the total loss. The partial derivative of the total loss with respect to $\mathbf{T}$ is obtained for the update, and it is calculated as

$$\mathbf{T}^{(k)} = \mathbf{T}^{(k-1)} - \lambda\frac{\partial J(\theta, y, \hat{y})}{\partial \mathbf{T}}, \quad (10)$$

where $\mathbf{T}^{(k)}$ is the updated target of the k^{th} layer, λ is the update rate, and $J(\theta, y, \hat{y})$ is the total loss of the downstream task, which is calculated as

$$J(\theta, y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N} y_{ij}\log\left(\hat{y}_{ij}\right), \quad (11)$$

where y is the target label, $\hat{y}$ is the output label, M is the number of classes, and N is the number of samples.

5. Experiments

In this section, we first introduce our generated brain network dataset and experimental setup. Then, the performance of GNEA is evaluated and compared with those of state-of-the-art graph neural networks.

5.1. Dataset. The brain networks in our dataset are generated using resting-state fMRI data from the Alzheimer's Disease

Neuroimaging Initiative (ADNI) by the LONI Image and Data Archive (IDA). Three patient types are included in our dataset, which are patients with Alzheimer's disease (AD), patients with mild cognitive impairment (MCI), and patients for normal control (NC). We select 118 samples of each patient type from all four project phases of the ANDI, which results in a total of 354 samples in our dataset. The samples of all the different patient types have similar distributions of ages and genders, and these distributions are presented in Figure 5.

All rs-fMRI images are processed using the toolboxes Nilearn[2] and DPABI (Data Processing and Analysis for Brain Imaging) [35]. The AAL Atlas [36] is applied to map the brain into 116 brain regions. Then, the brain networks are generated, each of which is in the form of a 116×116 square matrix.

The dataset is split into training data and testing data at a ratio of 4 : 1. Performance evaluation and comparison are obtained by using the testing samples.

5.2. Experimental Setup. We use the AUC (area under curve) with a 95% CI (confidence interval) to evaluate the classification performance. The 95% CI is calculated as

$$\left(\overline{x} \pm t_{n-1}, 1 - \frac{\alpha}{2} \frac{s_d}{\sqrt{n}}\right), \tag{12}$$

where n is the number of cross-validation folds, $\overline{x}$ is the mean result of the n-fold cross-validation process,$\alpha = 1 - 0.95 = 0.05$, t_{n-1} is the t-test with an n-1 degree of freedom, and s_{d} is the standard deviation, which is calculated as

$$s_d = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2}{n-1}}. \tag{13}$$

We evaluate the testing performance on three binary classification problems, namely, NC-AD, NC-MCI, and AD-MCI. Furthermore, we also evaluate the 3-class classification problem, in which the macroaverage strategy is applied in the AUC calculation.

All the experiments are conducted on a PC with a 3.7 GHz Intel Core CPU, an NVIDIA GeForce RTX 2080 Ti graphics card, 32 GB of 2400 MHz DDR4 RAM, and a 500 GB solid-state disk drive. The proposed method is realized using MATLAB R2018a and Python 3.6. The deep learning frameworks are TensorFlowGPU 1.8 and Keras 2.2.

5.3. Results. We first evaluate the performance of GNEA with varied hyperparameter settings, namely, the graph convolutional layers and classification layers. Then, we compare the overall performance of GNEA with those of several state-of-the-art methods.

5.3.1. Evaluation of Graph Convolutional Layers. The number of graph convolutional layers can be viewed as the distance of information propagation. In other words, k layers of graph convolution collect information from k-hop neighbors. The neighborhood sample size determines $|N'(v)|$, which is the number of nodes sampled from the neighbors $N(v)$ of node v. Thus, we evaluate the AUC in Figure 6(a) and the runtime in Figure 6(b) with varied layer numbers and sample sizes.

(1). The Number of Graph Convolutional Layers. Regarding our dataset, although more graph convolutions lead to a longer runtime, the network with three graph convolutional layers has the highest AUC performance. We believe that more than three layers of graph convolution may lead to the oversmoothing of the graph embedding, which decreases the discrimination among the brain networks of different class labels.

(2). The Neighborhood Sample Size. More sample nodes lead to higher matrix calculation cost. The incrementation is nonlinear due to the matrix operations in the information propagation and aggregation process. Although the runtime continues to increase nonlinearly, the growth in the AUC performance stagnates when the neighborhood sample size increases to 20. Since the sampling is biased by the correlation coefficients, when the number of neighbors is larger than a set threshold, extra information from the neighbors $N(v)$ of node v provides a little contribution to the embedding h_v of node v.

(3). The ELM Aggregator. We evaluate the AUCs and running times obtained with varied hidden numbers of the ELM aggregator, along with varied numbers of convolution layers. It can be found from the evaluation results presented in Figure 7 that the runtime continues to increase as the number of hidden nodes of the ELM aggregator increases. However, the AUC performance begins to drop when the number of ELM hidden nodes is larger than 400. Given a fixed dimension of input space, extra hidden nodes do not result in a more powerful learning ability.

5.3.2. Evaluation of Classification Layers. The fully connected layers in GNEA provide it with the ability to classify the representations of brain networks. We evaluate the AUCs in Figure 8(a) and runtimes in Figure 8(b) with varied numbers of fully connected layers and fully connected nodes in each layer. Both larger numbers of nodes and layers lead to longer runtimes. Since the major computational cost lies in the graph convolution operations, the differences of runtimes between various settings of the fully connected layers are low. Regarding the AUC performance, one fully connected layer is able to achieve a strong classification performance due to the excellent quality of the graph representation generated by GNEA. Specifically, the AUC performance reaches its maximum when the number of nodes increases to approximately 100 and begins to drop afterward. A larger number of either layers or nodes may lead to overfitting and poor testing performance.

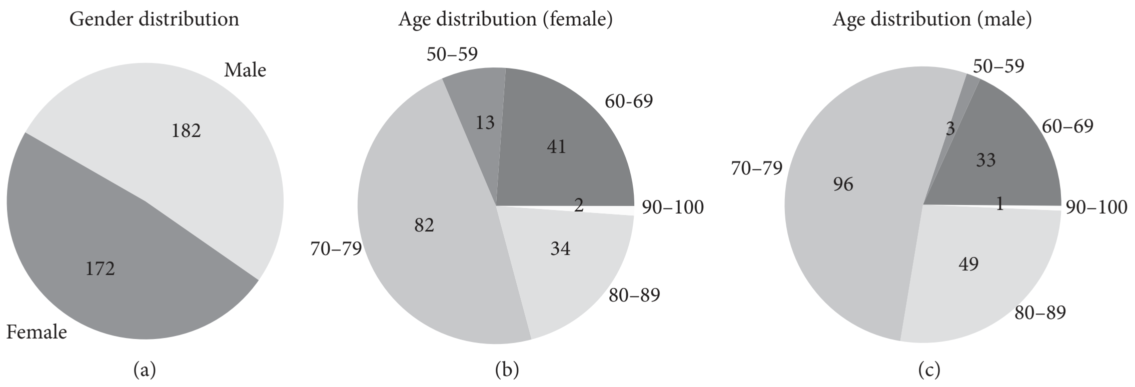


Figure 5: Distribution of patient ages and genders. (a) Gender distribution. (b) Female age distribution. (c) Male age distribution.

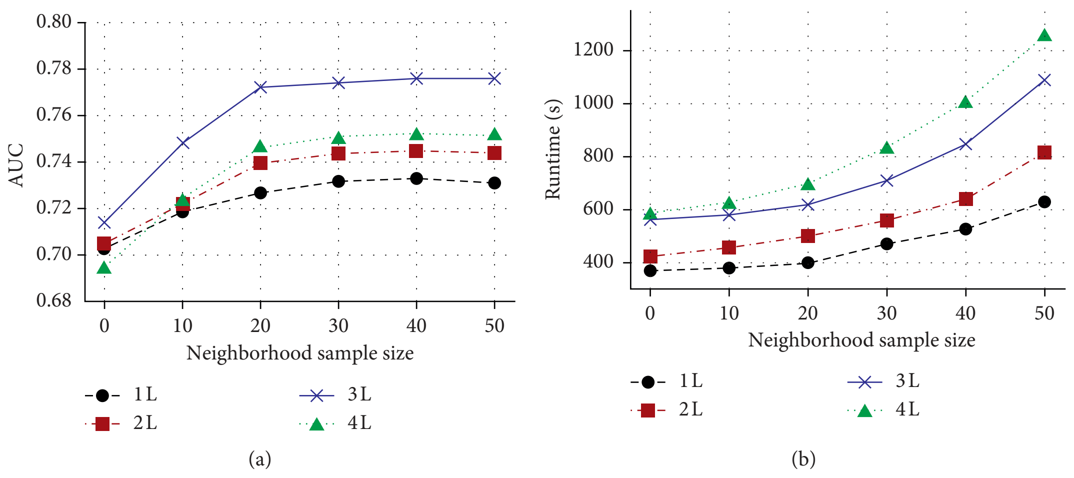


Figure 6: Performance evaluation of graph convolutional layers. (a) AUC. (b) Runtime.

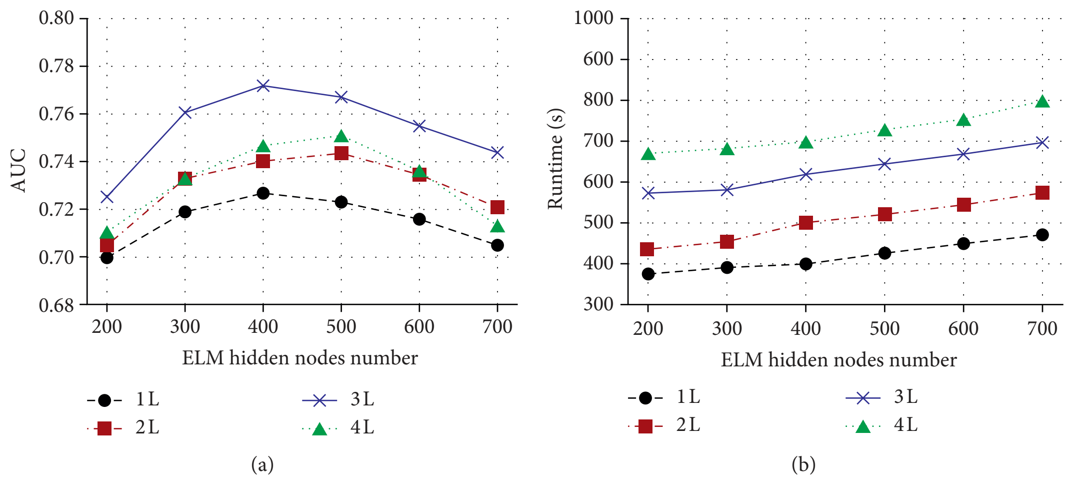
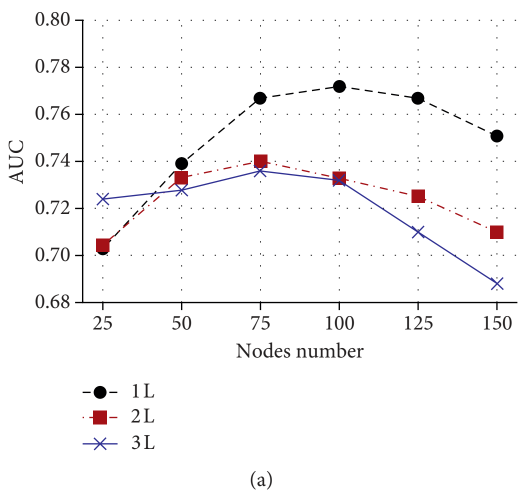


Figure 7: Performance evaluation of the ELM aggregator. (a) AUC. (b) Runtime.

5.3.3. Performance Comparison. Comparisons of the graph classification performances achieved by GNEA and state-of-the-art methods, namely, a convolutional neural network (CNN) [37], a long-short term memory (LSTM) [38] network, a graph convolutional network (GCN) [10], and GraphSAGE [13] with three aggregators, which are denoted as GS-mean, GS-pool, and GS-LSTM, respectively, are given in Table 1.

Among these rival methods, the CNN and LSTM are deep neural networks in Euclidean space. The GCN, GraphSAGE, and GNEA form the group of graph



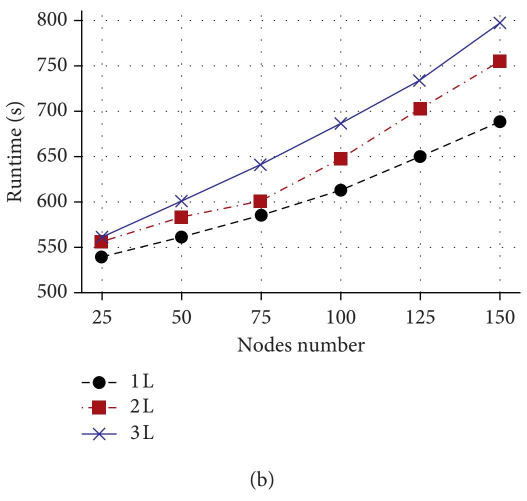


Figure 8: Performance evaluation of fully connected layers. (a) AUC. (b) Runtime.

Table 1: Performance comparison of AUCs with a 95% CI on different problems.

Methods	3-class	NC-AD	NC-MCI	MCI-AC
CNN	0.688 ± 0.042	0.743 ± 0.008	0.596 ± 0.078	0.579 ± 0.085
LSTM	0.671 ± 0.047	0.726 ± 0.013	0.601 ± 0.083	0.565 ± 0.098
GCN	0.717 ± 0.033	0.806 ± 0.005	0.674 ± 0.075	0.605 ± 0.081
GS-mean	0.725 ± 0.022	0.802 ± 0.003	0.688 ± 0.088	0.618 ± 0.091
GS-pool	0.720 ± 0.043	**0.813** ± 0.002	0.690 ± 0.097	0.610 ± 0.068
GS-LSTM	0.729 ± 0.012	**0.811** ± 0.004	**0.702** ± 0.099	0.613 ± 0.035
GNEA	**0.732** ± 0.025	**0.813** ± 0.003	**0.703** ± 0.078	**0.637** ± 0.045

convolutional networks, of which GCN applies spectral convolution, while GraphSAGE and our GNEA perform spatial convolution.

To thoroughly measure the performance of AD detection, we decompose the original 3-class classification problem into three binary classification problems. The NC-AD task is to distinguish between normal controls and Alzheimer's disease patients. The NC-MCI task is to distinguish between normal controls and patients with mild cognitive impairment. The MCI-AD task is to distinguish between patients with mild cognitive impairment and patients with Alzheimer's disease.

The comparison results presented in Table 1 indicate that (1) the graph learning neural networks achieve higher performance than the deep neural networks in Euclidean space, including the CNN and LSTM; (2) within the group of graph neural networks, spatial convolution outperforms spectral convolution; (3) for the general three-class problem, our proposed GNEA has the best AUC performance; (4) compared with their performances in the other binary classification tasks, all the methods have higher AUC scores in the NC-AD task, since the distinction between MCI and other classes is trivial and explicit; (5) for the NC-AD problem, both GraphSAGE and GNEA exhibit satisfactory learning ability due to the relatively explicit distinction between the brain networks of NCs and AD patients; (6) for the MCI-AD problem with the most implicit distinction, GNEA earns a dominant position due to the powerful representation ability of the ELM aggregator.

6. Conclusion

To address the graph learning problem for brain network classification, we propose a graph convolution aggregator based on extreme learning machine. The ELM aggregator exhibits an efficient and powerful aggregation ability. Then, we design a graph neural network named GNEA, which achieves high performance in graph embedding and graph classification. The results of extensive experiments on a real-world Alzheimer's disease detection task indicate that our proposed GNEA outperforms the state-of-the-art rival methods in the application of brain network classification.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant no. 2016YFC1401902 and the National Natural Science Foundation of China under Grants 61702086, 61672145, 61572121, and 61972077.

References

[1] F. Shi, N. Dey, A. S. Ashour, D. Sifaki-Pistolla, and R. S. Sherratt, "Meta-KANSEI modeling with valence-arousal fMRI dataset of brain," *Cognitive Computation*, vol. 11, no. 2, pp. 227–240, 2019.

[2] M. Termenon, M. Graña, A. Savio et al., "Brain MRI morphological patterns extraction tool based on extreme learning machine and majority vote classification," *Neurocomputing*, vol. 174, pp. 344–351, 2016.

[3] J. Liu, M. Li, W. Lan, F.-X. Wu, Y. Pan, and J. Wang, "Classification of alzheimer's disease using whole brain Hierarchical network," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 2, pp. 624–632, 2018.

[4] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.

[5] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 729–734, Montreal, Canada, August 2005.

[6] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[7] J. J. Hopfield, "Neural networks and physical systems with emergent collective computationa," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[8] J. Bruna, W. Zaremba, A. Szlam et al., "Spectral networks and locally connected networks on graphs," in *Proceedings of the International Conference on Learning Representations*, Bejing, China, November 2014.

[9] F. Monti, D. Boscaini, J. Masci et al., "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5425–5434, Honolulu, HI, USA, July 2017.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, Toulon, France, April 2017.

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[12] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[13] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

[14] P. Velickovic, G. Cucurull, A. Casanova et al., "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, 2018.

[15] D. Bacciu, F. Errica, and A. Micheli, "Contextual graph Markov model: A deep and generative approach to graph processing," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 294–303, Stockholm, Sweden, July 2018.

[16] S. I. Ktena, S. Parisot, E. Ferrante et al., "Metric learning with spectral graph convolutions on brain connectivity networks," *NeuroImage*, vol. 169, pp. 431–442, 2018.

[17] S. Rajchl, S. I. Ktena, E. Ferrante et al., "Disease prediction using graph convolutional networks: Application to Autism spectrum disorder and Alzheimer's disease," *Medical Image Analysis*, vol. 48, pp. 117–130, 2018.

[18] Y. Lee and H. Huang, "New graph-blind convolutional network for brain connectome data analysis," in *Proceedings of the International Conference on Information Processing in Medical Imaging*, pp. 669–681, Hong Kong, China, July 2019.

[19] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, Budapest, Hungary, July 2004.

[20] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[21] X. Bi, C. Zhang, X. Zhao, D. Li, Y. Sun, and Y. Ma, "CODES: Efficient incremental semi-supervised classification over drifting and evolving social streams," *IEEE Access*, vol. 8, pp. 14024–14035, 2020.

[22] J. P. Nóbrega and A. L. I. Oliveira, "A sequential learning method with Kalman filter and extreme learning machine for regression and time series forecasting," *Neurocomputing*, vol. 337, pp. 235–250, 2019.

[23] Y. Rizk and M. Awad, "On extreme learning machines in sequential and time series prediction: A non-iterative and approximate training algorithm for recurrent neural networks," *Neurocomputing*, vol. 325, pp. 1–19, 2019.

[24] X. Bi, X. Zhao, G. Wang, Z. Zhang, and S. Chen, "Distributed learning over massive XML documents in ELM feature space," *Mathematical Problems in Engineering*, vol. 2015, pp. 1–13, Article ID 923097, 2015.

[25] X. Zhao, X. Bi, G. Wang, Z. Zhang, and H. Yang, "Uncertain XML documents classification using extreme learning machine," *Neurocomputing*, vol. 174, pp. 375–382, 2016.

[26] G. Zhang, Y. Zhao, and D. Wang, "A protein secondary structure prediction framework based on the extreme learning machine," *Neurocomputing*, vol. 72, no. 1, pp. 262–268, 2008.

[27] X. Bi, H. Ma, J. Li, Y. Ma, and D. Chen, "A positive and unlabeled learning framework based on extreme learning machine for drug-drug interactions discovery," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2018.

[28] L. Duan, M. Bao, S. Cui, Y. Qiao, and J. Miao, "Motor imagery EEG classification based on kernel Hierarchical extreme learning machine," *Cognitive Computation*, vol. 9, no. 6, pp. 758–765, 2017.

[29] X. Qiao, X. Zhao, H. Huang, D. Chen, and Y. Ma, "Functional brain network classification for Alzheimer's disease detection with deep features and extreme learning machine," *Cognitive Computation*, vol. 12, no. 3, pp. 513–527, 2020.

[30] Y. Sun, B. Li, Y. Yuan, X. Bi, X. Zhao, and G. Wang, "Big graph classification frameworks based on extreme learning machine," *Neurocomputing*, vol. 330, pp. 317–327, 2019.

[31] X. Zhao, Z. Ma, B. Li, Z. Zhang, and H. Liu, "ELM-based convolutional neural networks making move prediction in Go," *Soft Computing*, vol. 22, no. 11, pp. 3591–3601, 2018.

[32] X. Liu, Y. Zeng, T. Zhang, and B. Xu, "Parallel brain simulator: A Multi-scale and parallel brain-inspired neural network modeling and simulation platform," *Cognitive Computation*, vol. 8, no. 5, pp. 967–981, 2016.

[33] A. J. Bishara and J. B. Hittner, "Testing the significance of a correlation with nonnormal data: Comparison of Pearson, Spearman, transformation, and resampling approaches," *Psychological Methods*, vol. 17, no. 3, pp. 399–417, 2012.

[34] D. S. Bassett and E. Bullmore, "Small-world brain networks," *The Neuroscientist*, vol. 12, no. 6, pp. 512–523, 2006.

[35] C.-G. Yan, X.-D. Wang, X.-N. Zuo, and Y.-F. Zang, "DPABI: Data processing & analysis for (Resting-State) brain imaging," *Neuroinformatics*, vol. 14, no. 3, pp. 339–351, 2016.

[36] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou et al., "Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain," *Neuroimage*, vol. 15, no. 1, p. 273, 2002.

[37] Y. Crivello and Y. Bengio, "Convolutional networks for images, speech, and time series,," in *The Handbook of Brain Theory and Neural Networks*, pp. 255–258, MIT Press, Cambridge, MA, USA, 1998.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.