AGNN: Alternating Graph-Regularized Neural Networks to Alleviate Over-Smoothing

Zhaoliang Chen, Zhihao Wu, Zhenghong Lin, Shiping Wang, *Member, IEEE*, Claudia Plant, and Wenzhong Guo

***Abstract*— Graph convolutional network (GCN) with the powerful capacity to explore graph-structural data has gained noticeable success in recent years. Nonetheless, most of the existing GCN-based models suffer from the notorious over-smoothing issue, owing to which shallow networks are extensively adopted. This may be problematic for complex graph datasets because a deeper GCN should be beneficial to propagating information across remote neighbors. Recent works have devoted effort to addressing over-smoothing problems, including establishing residual connection structure or fusing predictions from multi-layer models. Because of the indistinguishable embeddings from deep layers, it is reasonable to generate more reliable predictions before conducting the combination of outputs from various layers. In light of this, we propose an alternating graph-regularized neural network (AGNN) composed of graph convolutional layer (GCL) and graph embedding layer (GEL). GEL is derived from the graph-regularized optimization containing Laplacian embedding term, which can alleviate the over-smoothing problem by periodic projection from the low-order feature space onto the high-order space. With more distinguishable features of distinct layers, an improved Adaboost strategy is utilized to aggregate outputs from each layer, which explores integrated embeddings of multi-hop neighbors. The proposed model is evaluated via a large number of experiments including performance comparison with some multilayer or multi-order graph neural networks, which reveals the superior performance improvement of AGNN compared with the state-of-the-art models.**

***Index Terms*— Graph convolutional network (GCN), graph representation learning, over-smoothing, semi-supervised classification.**

NOMENCLATURE

Notations	*Explanations*
$\mathbf{X}$	Feature matrix of nodes.
$\mathbf{A}$	Adjacency matrix.
$\mathbf{Y}$	Label information.
$\mathbf{H}^{(l)}$	Output of the lth GCL.
$\mathbf{Z}^{(l)}$	Output of the lth GEL.
$\mathbf{D}$	Diagonal degree matrix.
$\mathbf{L}$	Laplacian matrix.
$\mathbf{W}_g^{(l)}$	Weight matrix for the lth GCL.
$\mathbf{W}_{e1}^{(l)}$, $\mathbf{W}_{e2}^{(l)}$	Weight matrices for the lth GEL.
$\mathbf{Prox}_g(\cdot)$	Proximal operator.
$\xi_{(\theta_1,\theta_2)}(\cdot)$	MSReLU function with hyperparameters θ_1 and θ_2.
$c(\cdot)$	Weak classifier.
$\mathbf{S}$	Weighted embedding of multi-order feature fusion.
$\alpha^{(l)}$, $\beta^{(l)}$	Weights of classifiers for the lth GCL and GEL.
π_i	Node weights for AdaBoost.
$e_{\mathbf{H}}^{(l)}$, $e_{\mathbf{Z}}^{(l)}$	Weighted classification error rates.
η_i	Node weight updating rate for AdaBoost.
R	Number of classes.

Manuscript received 27 July 2022; revised 4 December 2022; accepted 22 April 2023. This work was supported in part by the National Natural Science Foundation of China under Grant U21A20472 and Grant 62276065, and in part by the National Key Research and Development Plan of China under Grant 2021YFB3600503. *(Corresponding author: Wenzhong Guo.)*

Zhaoliang Chen, Zhihao Wu, Zhenghong Lin, Shiping Wang, and Wenzhong Guo are with the College of Computer and Data Science and the Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China (e-mail: chenzl23@outlook.com; zhihaowu1999@gmail.com; hongzhenglin970323@gmail.com; shipingwangphd@163.com; guowenzhong@fzu.edu.cn).

Claudia Plant is with the Faculty of Computer Science and the Research Network Data Science, University of Vienna, 1090 Vienna, Austria (e-mail: claudia.plant@univie.ac.at).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3271623.

Digital Object Identifier 10.1109/TNNLS.2023.3271623

I. INTRODUCTION

GRAPH neural network (GNN) has become one of the promising technologies manipulating graph-structural data in recent years, obtaining remarkable achievement in various pattern recognition fields, including node classification or clustering [1], [2], [3], recommender systems [4], [5], [6], and computer vision [7], [8], [9]. As one of the typical GNN-based models, graph convolutional network (GCN) is receiving plentiful attention from a population of researchers [10], [11]. Owing to its powerful ability to extract knowledge from sparse weighted networks, GCN has also been adopted to weight prediction for sparse weighted graphs, such as dynamic graphs [12], [13], [14]. Originated from GCN, graph autoencoder (GAE) was also investigated to conduct weighted link predictions via the reconstruction of the adjacency matrix [15], [16], [17]. GCN propagates node representations across topology networks via convolution operators on non-Euclidean space, which integrates node features and relationships involved in a graph. Nonetheless, recent practice and theoretical analysis have indicated that a two-layer GCN generally performs the best, and a deep GCN often leads to

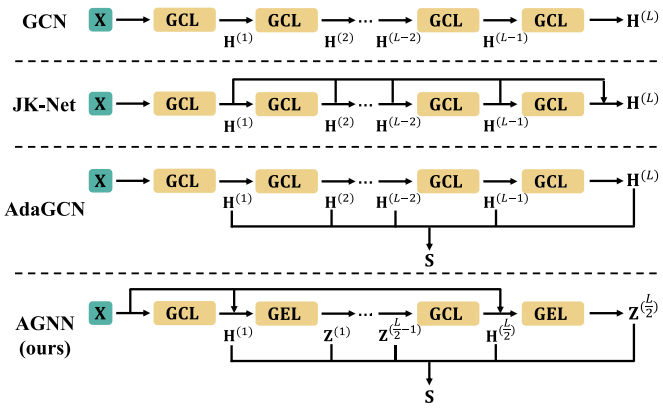


Fig. 1. Architectures of numerous GCN-based methods and our AGNN with graph convolutional layer (GCL) and the proposed graph embedding layer (GEL). GCN [18] is a sequence of GCLs, which encounters severe over-smoothing with deep layers. JK-Net [19] adds connections among layers to carry all low-order information to the last layer. AdaGCN [20] aggregates multi-hop embeddings of all layers. The proposed AGNN simultaneously carries low-order features to deep layers and accumulates node predictions from all layers.

unfavorable performance, which is summarized into the over-smoothing issue.

Over-smoothing is a widely concerned deficiency of GCN, which has been extensively investigated. Recent studies have proved that a graph convolution is exactly a special form of Laplacian smoothing, attributed to which a deeper GCN may result in indistinguishable node features and make the downstream classification tasks challenging [21], [22], [23]. It makes most existing GCN-based models shallow and lack the ability to mine knowledge from high-order neighbors, which is more severe for datasets with high-degree nodes. Considerable works have been devoted to solving this problem. On one hand, some research attempted to consider a similar structure of residual connection leveraged in Euclidean deep convolutional networks [19], [24], [25]. Most of these methods made full use of embeddings from the previous layers or the input matrix to avoid information loss. On the other hand, some studies placed more emphases on effective exploration and combination of hidden representations from different hops of neighbors [20], [23], [26]. A summary of the comparison between the representative algorithms (GCN [10], JK-Net [19], AdaGCN [20]), and the proposed method in this article is shown in Fig. 1. Although some works have succeeded in relieving over-smoothing problems, they were still outperformed by a classical two-layer GCN. In addition, a direct linear combination of embeddings from hidden layers may not work effectively, because the similar and indistinguishable features from deeper layers can annihilate useful information from shallow layers and confound the predictions of classifiers. Accordingly, it is crucial to develop a reliable network where each layer can yield accurate and distinguishable outputs before conducting the prediction fusion.

In pursuit of addressing the aforementioned problems, in this article, we design an alternating graph-regularized neural network (AGNN) that enables the construction of deep layer architecture. AGNN alternately performs forward computation of GCLs and GELs. To get rid of similar and indistinguishable features caused by over-smoothing, GEL is designed to project original node embeddings onto low-dimensional space in deep layers and preserve critical features via sparse outputs. Thus, each proposed GEL aims to learn Laplacian-constrained sparse representations from original features, on the basis of the optimization problem with respect to the Laplacian-based graph regularization and sparsity constraint. We derive the updating rules of this optimization target and transform them into GEL that preserves discriminative node embeddings during network training and alleviates the over-smoothing problem. We analyze the network architecture and draw a conclusion that both GCL and GEL can be approximately regarded as solutions to distinct graph regularization problems. Furthermore, with more accurate predictions yielded by GCL and GEL, an improved Adaboost algorithm is adopted to aggregate node representations from varying hidden layers, so that multi-order information from different depths of networks can be leveraged. In summary, our contributions are as follows.

1) According to a graph-regularized optimization problem and its iterative solutions, we construct a new layer-dubbed GEL, which can alleviate over-smoothing phenomenon via carrying low-order information to deep layers.
2) A graph-regularized neural network with alternating GCLs and GELs is proposed, which adopts both residual connection and embedding aggregation architecture. Its layers can be regarded as approximations of different graph optimization problems, which promote the interpretability of the model.
3) With more accurate embeddings yielded by deep layers, an improved Adaboost algorithm is designed to leverage features from distinct hidden layers, enabling the model to aggregate high-quality node representations from multi-hop neighbor propagation.
4) Substantial experimental results reveal the superiority of the proposed AGNN, which succeeds in coping with over-smoothing issue and outperforms the widely applied two-layer GCN and other multilayer GCN-based methods with deep network structures.

The rest contents of this article are organized as follows. Recent works of GCN and approaches to cope with the over-smoothing issues are discussed in Section II. In Section III, we elaborate on the proposed framework, including detailed analysis and comparison between AGNN and other models. We evaluate AGNN with comprehensive experiments in Section IV, looking into the performance under varying experimental settings. Eventually, we conclude our works in Section V.

II. Related Works

A. Graph Convolutional Network

GCN has been applied to a multitude of applications and attracted attention from a wide range of researchers in recent years. Xu et al. [27] came up with a deep feature aggregation model with a GCN to conduct high spatial resolution scene classification. A GCN-based approach under the autoencoder

framework was proposed to perform unsupervised community detection [28]. To reduce the computational cost of graph convolutions, a low-pass collaborative filter was proposed to utilize GCN with a large graph [29]. Gan et al. [30] designed a multigraph fusion model that combined the local graph and the global graph to produce a high-quality graph for GCN. An aggregation scheme was applied to promote the robustness of GCN against structural attacks [31]. Geometric scattering transformations and residual convolutions were leveraged to enhance the conventional GCN [18]. Xu et al. [32] presented a spatiotemporal multigraph convolutional fusion network, which exploited the graph-structural road network for urban vehicle emission estimation. GCN with a question-aware gating mechanism was presented to aggregate evidences on the path-based graph [33]. A new graph convolution operator was proposed to obtain robust embeddings in the spectral domain [34]. The variant of GCN was derived via a modified Markov diffusion kernel, which explored the global and local contexts of nodes [35]. Weighted link prediction is also a critical application of GCN. For example, a dynamic GCN was proposed with a tensor M-product technique, to cope with adjacency tensor and feature tensor yielded from dynamic graphs [36]. Cui et al. [17] proposed an adaptive graph encoder to strengthen the filtered features for more discriminative node embeddings, which was applied to link prediction tasks. Wang et al. [15] designed a temporal GAE, which encoded the fundamentally asymmetric nature of a directed network from neighborhood aggregation and captured link weights via reshaping the adjacency matrix. However, most of these GCN-based models suffer from shallow network structure owing to the over-smoothing issue.

B. Over-Smoothing Issue

Numerous works have investigated approaches to alleviate the over-smoothing issue. An improved normalization trick applying the "diagonal enhancement" was introduced to help build a deep GCN [37]. Simple graph convolution [38] was proposed to mine high-order embeddings in the graph via utilizing the kth power of the graph convolutional matrix and removing the ReLU function. A multilayer GCN was constructed with AdaBoost to linearly combine embeddings from varying layers [20]. Cui et al. [39] restricted over-smoothing by extracting hierarchical multi-scale node feature representations. PPNP and APPNP [26] were presented to replace the power of the graph convolutional matrix inspired by the personalized PageRank matrix. Residual connections and dilated convolutions in CNN were applied to promote the training of a deep GCN model. Jumping knowledge networks preserved the locality of node embeddings via dense skip connections that merged features from each layer [19]. A deep GCN was proposed with residual connections and identity mappings to relieve the over-smoothing problem [24]. Most of these methods attempted to alleviate over-smoothing via connecting distinct network layers, simplifying multi-order graph convolutions, or conducting multilayer feature fusion. Nonetheless, these existing works did not simultaneously consider cross-layer feature connections and the aggregation of embeddings from varying layers, which benefit a multilayer model to obtain a more precise prediction.

III. Proposed Method

Given a connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes and e edges, we define the corresponding adjacency matrix as $\mathbf{A} \in \mathbb{R}^{n\times n}$. The node features are denoted by the matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, i.e., $\mathbf{x}_i$ is an m-dimensional feature vector of the ith node. The proposed AGNN aims to carry out the semi-supervised classification task with the given set Ω of partially labeled samples and its corresponding ground truth matrix $\mathbf{Y} \in \mathbb{R}^{n\times c}$ encoding one-hot vectors, where c is the number of classes. For the purpose of better readability, we summarize the primarily used mathematical notations in Nomenclature. As described in Fig. 2, AGNN is a sequence of alternating GCL and GEL, and an improved AdaBoost strategy is adopted to merge multilayer features. Both GCL and GEL are constructed from graph-regularized optimization problems, which form a basic network block of AGNN. In particular, GEL periodically projects the original node embeddings onto deep layers to alleviate over-smoothing, which introduces residual connections into AGNN. In Section III-A, we first analyze two distinct graph-regularized optimization problems, on the basis of which AGNN is constructed. After that, an improved AdaBoost is designed to conduct multilayer feature fusion in Section III-B. Finally, we summarized and analyzed the proposed model in Section III-C, including time complexity analysis and comparison to related works.

A. Alternating GCLs and GELs

First, we revisit the definition of a vanilla graph convolution operator. A GCL is formulated as follows:

$$\mathbf{H}^{(l)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}_g^{(l)}\right) \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix that adds self-loop and $[\tilde{\mathbf{D}}]_{\text{ii}} = \sum_j [\tilde{\mathbf{A}}]_{ij}$ denotes the diagonal degree matrix. The optional activation function is denoted as $\sigma(\cdot)$. In fact, the added self-loop $\mathbf{A} + \mathbf{I}$ can be regarded as a simple residual connection to the previous layer. Actually, GCL can be formulated as a graph-regularized optimization problem. Namely, we have the following theorem.

Theorem 1: With a linear transformation matrix $\mathbf{W}_g^{(l)}$ and the node embedding $\mathbf{H}^{(l-1)}$ from the previous layer, the lth GCL defined in (1) is the first-order approximation of the following optimization problem:

$$\mathbf{H}^{(l)} = \arg\min_{\mathbf{E}^{(l)}} \left\|\mathbf{E}^{(l)} - \mathbf{H}^{(l-1)}\mathbf{W}_g^{(l)}\right\|_F^2 + \text{Tr}(\mathbf{E}^{(l)T}\tilde{\mathbf{L}}\mathbf{E}^{(l)}) \tag{2}$$

where $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-(1/2)}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-(1/2)}$.

Proof: The derivative with respect to $\mathbf{E}^{(l)}$ of the optimization problem defined in (2) is

$$\frac{\partial \mathcal{J}}{\partial \mathbf{E}^{(l)}} = 2\left(\mathbf{E}^{(l)} - \mathbf{H}^{(l-1)}\mathbf{W}_g^{(l)}\right) + 2\tilde{\mathbf{L}}\mathbf{E}^{(l)}. \tag{3}$$

Setting the derivative to 0, we have the closed-form solution

$$\mathbf{E}^{(l)} = \left(\mathbf{I} + \tilde{\mathbf{L}}\right)^{-1}\mathbf{H}^{(l-1)}\mathbf{W}_g^{(l)}. \tag{4}$$

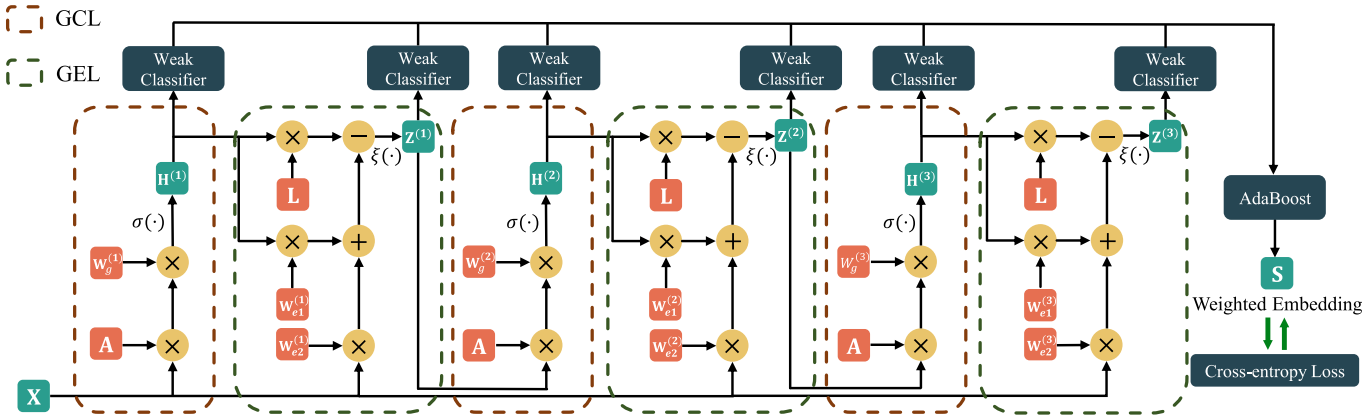


Fig. 2. Framework of a six-layer AGNN, which consists of three GCLs and three GELs. AGNN is a block-wise graph neural network framework constructed with alternating GCL and GEL, where each block contains a GCL and a GEL. For the purpose of exploiting reliable and discriminative multi-hop information, an improved AdaBoost strategy is utilized to aggregate node predictions yielded by weak classifiers in all layers, and the whole framework is evaluated by cross-entropy loss.

Because the term $(\mathbf{I}+\tilde{\mathbf{L}})^{-1}$ can be decomposed into Taylor series, i.e.,

$$\left(\mathbf{I}+\tilde{\mathbf{L}}\right)^{-1}=\mathbf{I}-\tilde{\mathbf{L}}+\tilde{\mathbf{L}}^2+\cdots+(-1)^t\tilde{\mathbf{L}}^t \tag{5}$$

we have the first-order truncated approximation as follows:

$$\left(\mathbf{I}+\tilde{\mathbf{L}}\right)^{-1}\approx\mathbf{I}-\tilde{\mathbf{L}}=\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}. \tag{6}$$

Consequently, we obtain the approximation of $\mathbf{H}^{(l)}$ as follows:

$$\mathbf{H}^{(l)}\approx\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}_g^{(l)} \tag{7}$$

which indicates that GCL is a first-order approximation of problem (2). □

However, as we have analyzed before, deep graph convolutions often suffer from extremely indistinguishable features due to the over-smoothing phenomenon. A solution is enabling the model to carry low-order information by connecting initial node features to each GCL. Thus, we develop a new layer to bring features from the original space to deep layers. To be consistent with GCL, we define a graph-regularized optimization problem to formulate this layer. Instead of directly adding initial embeddings to the end of each GCL, a trainable projection derived from graph regularization optimization is applied, which adaptively learns low-dimensional representations from original node embeddings. Namely, with graph embedding $\mathbf{H}$, we consider the following sparsity-constrained optimization:

$$\mathbf{Z}^{(l)}=\arg\min_{\mathbf{H}}\|\mathbf{X}-\mathbf{H}\mathbf{P}^{(l)}\|_F^2+\mathrm{Tr}\left(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H}\right)+\|\mathbf{H}\|_1 \tag{8}$$

which explores Laplacian-constrained representations from the original feature space after the lth GCL. In pursuit of obtaining more distinguishable compressed node embeddings, we adopt $\|\mathbf{H}\|_1$ to consider sparse representations. The sparsity constraint enables GEL to yield more discriminative node representations that only include important features, and alleviates the similar features of different nodes in deep layers, which is beneficial to solve the over-smoothing issue. Consequently, it should have the same dimension as the previous GCL, and we can project it onto the original feature space with an overcomplete dictionary matrix $\mathbf{P}^{(l)}\in\mathbb{R}^{d_l\times m}$, where $d_l<m$ is the number of hidden units at the lth GCL. In addition, we adopt the Laplacian embedding criterion $\mathrm{Tr}(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H})$ to make nodes close when they are connected, where the Laplacian matrix $\tilde{\mathbf{L}}$ is precomputed. To obtain more representative low-dimensional features, $\|\mathbf{H}\|_1$ promoting the sparsity of outputs is added to extract robust projected embeddings during training. Letting $f(\mathbf{H})=\mathrm{Tr}(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H})+\|\mathbf{X}-\mathbf{H}\mathbf{P}^{(l)}\|_F^2$ and $g(\mathbf{H})=\|\mathbf{H}\|_1$, we can derive the updating rules of problem (8) at $\mathbf{H}^{(l)}$ via proximal gradient descent method. Namely

$$\begin{aligned}\mathbf{Z}^{(l)}&=\arg\min_{\mathbf{H}}f(\mathbf{H}^{(l)})+\langle\nabla f(\mathbf{H}^{(l)}),\mathbf{H}-\mathbf{H}^{(l)}\rangle\\&\quad+\frac{\tau}{2}\left\|\mathbf{H}-\mathbf{H}^{(l)}\right\|_F^2+\left\|\mathbf{H}^{(l)}\right\|_1\\&=\arg\min_{\mathbf{H}}\frac{\tau}{2}\|\mathbf{H}-\mathbf{Y}\|_F^2+\left\|\mathbf{H}^{(l)}\right\|_1\end{aligned} \tag{9}$$

where $\mathbf{Y}=\mathbf{H}^{(l)}-(1/\tau)\nabla f(\mathbf{H}^{(l)})$, and τ is the Lipschitz constant. Given the proximal operator $\mathbf{Prox}_g(\cdot)$, problem (9) can be solved by the proximal mapping with respect to ℓ_1 norm. Because we have the derivatives

$$\nabla f(\mathbf{H}^{(l)})=2\tilde{\mathbf{L}}\mathbf{H}^{(l)}+2\left(\mathbf{H}^{(l)}\mathbf{P}^{(l)}-\mathbf{X}\right)\mathbf{P}^{(l)T} \tag{10}$$

the proximal mapping can be derived from

$$\begin{aligned}\mathbf{Z}^{(l)}&=\mathbf{Prox}_g\left(\mathbf{H}^{(l)}-\frac{1}{\tau}\nabla f(\mathbf{H}^{(l)})\right)\\&=\mathbf{Prox}_g\left(\mathbf{H}^{(l)}-\frac{1}{\tau}\left(2\tilde{\mathbf{L}}\mathbf{H}^{(l)}+2\left(\mathbf{H}^{(l)}\mathbf{P}^{(l)}-\mathbf{X}\right)\mathbf{P}^{(l)T}\right)\right)\\&=\mathbf{Prox}_g\left(\mathbf{H}^{(l)}\left(\mathbf{I}-\frac{2}{\tau}\mathbf{P}^{(l)}\mathbf{P}^{(l)T}\right)-\frac{2}{\tau}\tilde{\mathbf{L}}\mathbf{H}^{(l)}+\frac{2}{\tau}\mathbf{X}\mathbf{P}^{(l)T}\right).\end{aligned} \tag{11}$$

Transforming terms $\mathbf{I}-(2/\tau)\mathbf{P}^{(l)}\mathbf{P}^{(l)T}$ and $(2/\tau)\mathbf{P}^{(l)T}$ into trainable weight matrices $\mathbf{W}_{e1}^{(l)}\in\mathbb{R}^{d_l\times d_l}$ and $\mathbf{W}_{e2}^{(l)}\in\mathbb{R}^{m\times d_l}$, respectively, we have the following proximal projection:

$$\mathbf{Z}^{(l)}=\mathbf{Prox}_g\left(\mathbf{H}^{(l)}\mathbf{W}_{e1}^{(l)}+\mathbf{X}\mathbf{W}_{e2}^{(l)}-\lambda\tilde{\mathbf{L}}\mathbf{H}^{(l)}\right) \tag{12}$$

where $\lambda=(2/\tau)$ is a hyperparameter. Because $\mathbf{Prox}_g(\cdot)$ can be regarded as an activation function, (12) is similar to the definition of a neural network layer with two trainable weight

matrices. In particular, the proximal operator for ℓ_1 constraint promoting the sparsity is

$$\mathbf{Prox}_g\left(\mathbf{Z}_{ij}^{(l)}\right) = \text{sign}\left(\mathbf{Z}_{ij}^{(l)}\right)\left(\left|\mathbf{Z}_{ij}^{(l)}\right| - \theta\right)_+ \tag{13}$$

which is the soft thresholding (ST) function and θ is the hyperparameter to guarantee the sparsity of the output [40]. It can be realized by a parameterized ReLU-based activation function, i.e.,

$$\xi_\theta(z) \ = \text{ReLU}(z-\theta) - \text{ReLU}(-z-\theta). \tag{14}$$

Due to the definition of the ST function, $|\xi_\theta(z)|$ is actually smaller than $|z|$ when $z > \theta$ and $z < -\theta$. This may be problematic due to the gap between original features and outputs of $\xi_\theta(z)$ when θ is relatively large. For the sake of relieving the influence of this problem, in this article, we adopt a multistage proximal projection for the sparsity constraint, as shown below

$$\xi_{(\theta_1,\theta_2)}(z) = \begin{cases} z, & \theta_2 \le z \\ \left(\dfrac{2\theta_2-\theta_1}{\theta_2}\right)(z-\theta_1), & \theta_1 \le z < \theta_2 \\ 0, & -\theta_1 \le z < \theta_1 \\ \left(\dfrac{2\theta_2-\theta_1}{\theta_2}\right)(z+\theta_1), & -\theta_2 \le z < -\theta_1 \\ z, & z < -\theta_2 \end{cases} \tag{15}$$

where $\theta_2 \ge \theta_1 > 0$. As a matter of fact, it also can be implemented by the combination of ReLU functions. Consequently, we define a new ReLU-based activation function as follows:

$$\begin{aligned}\xi_{(\theta_1,\theta_2)}\left(\mathbf{Z}^{(l)}\right) = w_1\big(&\text{ReLU}\left(\mathbf{Z}^{(l)}-\theta_1\right) - \text{ReLU}\left(-\mathbf{Z}^{(l)}-\theta_1\right)\big) \\ &- w_2\big(\text{ReLU}\left(\mathbf{Z}^{(l)}-\theta_2\right) - \text{ReLU}\left(-\mathbf{Z}^{(l)}-\theta_2\right)\big)\end{aligned} \tag{16}$$

where w_1 and w_2 are computed according to the parameter settings of θ_1 and θ_2, that is

$$\begin{aligned} w_1 &= \frac{2\theta_2-\theta_1}{\theta_2} \\ w_2 &= w_1 - 1 = \frac{\theta_2-\theta_1}{\theta_2}. \end{aligned} \tag{17}$$

Consequently, we have $2 \ge w_1 \ge 1 \ge w_2 \ge 0$. Eq. (16) is termed as a Multistage ReLU (MSReLU) function. The comparison of MSReLU and other activation functions is shown in Fig. 3. It can be observed that with suitable θ_1 and θ_2, it has less gap between $|\xi_{\theta_1,\theta_2}(z)|$ obtained by MSReLU and $|z|$ due to the increasing slope when $\theta_1 < z < \theta_2$ and $-\theta_2 < z < -\theta_1$, which is beneficial to obtaining more accurate features. When $z > \theta_2$ and $z < -\theta_2$, the slope is the same as ReLU and ST to maintain the feature distribution of outputs.

Associated with GCL, we can formulate a basic block of the alternating forward computation (contains two layers) as follows:

$$\mathbf{H}^{(l)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}_g^{(l)}\right) \tag{18}$$

$$\mathbf{Z}^{(l)} = \xi_{(\theta_1,\theta_2)}\left(\mathbf{H}^{(l)}\mathbf{W}_{e1}^{(l)} + \mathbf{X}\mathbf{W}_{e2}^{(l)} - \lambda\tilde{\mathbf{L}}\mathbf{H}^{(l)}\right) \tag{19}$$

where $\mathbf{H}^{(l-1)} = \mathbf{Z}^{(l-1)}$ for $l = 2, \ldots, t$ and $\mathbf{H}^{(0)} = \mathbf{X}$. We term the forward computation defined in (19) as GEL. The definition of GEL shows that it refines graph representations from the previous GCL and considers one-hop embeddings of neighbors via $\tilde{\mathbf{L}}\mathbf{H}^{(l)}$. Here we adopt $\mathbf{H}^{(l)}$ generated by GCL as the input of GEL, because GCL also implicitly optimizes the graph Laplacian regularization term. Actually, both GCL and GEL are one-step approximations of Laplacian-based graph regularization problems. GEL also leverages the information of original features via an input-injected computation defined by $\mathbf{X}\mathbf{W}_{e2}^{(l)}$ to preserve sparse and discriminative representations of nodes at the hidden and the last layers, thereby alleviating the over-smoothing problem. On the basis of (18) and (19), we can construct a deep block-wise graph neural network with $2t$ layers that consists of GCL and GEL alternately.

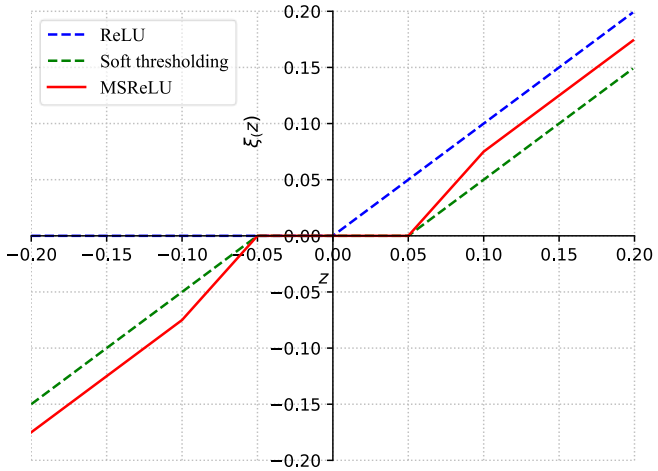


Fig. 3. Comparison of different activation functions (ReLU, ST and MSReLU) for sparse proximal projection, where hyperparameters are fixed as $\theta = 0.05$ for ST, and $\theta_1 = 0.05$, $\theta_2 = 0.10$ for MSReLU.

B. AGNN With Improved Adaboost

To further leverage underlying features at each layer and obtain results contributed by different hops of neighborhood relationships, we adopt a variant of Adaboost to compute the final predictions of the model. For the purpose of obtaining graph representations with the same dimension, we adopt a weak classifier

$$c\left(\mathbf{H}^{(l)}\right) = \text{Softmax}\left(\sigma\left(\mathbf{H}^{(l)}\mathbf{W}_c + b\right)\right) \tag{20}$$

for each layer of GCL, where $\mathbf{W}_c \in \mathbb{R}^{d_l \times d_L}$. The weak classifier $c(\mathbf{Z}^{(l)})$ for GEL is homologous. We assign corresponding weights $\alpha^{(l)}$ and $\beta^{(l)}$ for each GCL and GEL. Formulaically, the final weighted result of various classifiers is

$$\mathbf{S} = \sum_{l=1}^{t}\left(\alpha^{(l)}c\left(\mathbf{H}^{(l)}\right) + \beta^{(l)}c\left(\mathbf{Z}^{(l)}\right)\right) \tag{21}$$

where $\alpha^{(l)}$ indicates the weight of classifier with respect to $\mathbf{H}^{(l)}$ and $\beta^{(l)}$ indicates the weight of classifier with respect to $\mathbf{Z}^{(l)}$. We measure the performance of each weak classifier on labeled nodes to calculate classifier weights, which ensures that classifiers with higher accuracy on the training set are assigned to larger weights. First, the weighted error rates of two types of classifiers are computed by the following

equations:

$$e_{\mathbf{H}}^{(l)} = \sum_{i\in\Omega} \pi_i \mathbb{I}\big(c(\mathbf{H}^{(l)})_i \neq y_i\big) \Big/ \sum_{i\in\Omega} \pi_i \tag{22}$$

$$e_{\mathbf{Z}}^{(l)} = \sum_{i\in\Omega} \pi_i \mathbb{I}\big(c(\mathbf{Z}^{(l)})_i \neq y_i\big) \Big/ \sum_{i\in\Omega} \pi_i \tag{23}$$

where Ω is the set of samples having supervision information and π_i is the weight of a labeled node. The sample weights are initialized by $\pi_i = (1/|\Omega|)$. Therefore, classifier weights $\alpha^{(l)}$ and $\beta^{(l)}$ are computed by the following equations:

$$\alpha^{(l)} = \frac{1}{2}\log\frac{1-e_{\mathbf{H}}^{(l)}}{e_{\mathbf{H}}^{(l)}} + \log(R-1) \tag{24}$$

$$\beta^{(l)} = \frac{1}{2}\log\frac{1-e_{\mathbf{Z}}^{(l)}}{e_{\mathbf{Z}}^{(l)}} + \log(R-1) \tag{25}$$

where R is the number of classes. We apply the softmax normalization to all classifier weights, i.e.,

$$[\boldsymbol{\alpha}, \boldsymbol{\beta}] \leftarrow \text{Softmax}([\boldsymbol{\alpha}, \boldsymbol{\beta}]) \tag{26}$$

where $\boldsymbol{\alpha} = [\alpha^{(1)}, \ldots, \alpha^{(l)}]$ and $\boldsymbol{\beta} = [\beta^{(1)}, \ldots, \beta^{(l)}]$. For the purpose of increasing weights on incorrect classified nodes, we update π_i by the following equations:

$$\pi_i \leftarrow (1+\eta_i)\pi_i \mathbb{I}(c_i \neq y_i) \tag{27}$$

$$\pi_i \leftarrow \max(1-\eta_i, \rho)\pi_i \mathbb{I}(c_i = y_i) \tag{28}$$

where c_i is the predicting result of the former classifier and y_i is the ground truth. η_i is an updating rate that changes the sample weight automatically according to predictions of the weak classifier. The threshold $0 < \rho < 1$ is adopted to avoid nodes with weights of zeros. In particular, the updating rate η_i applied in this article is defined by the following equation:

$$\eta_i = \exp\left(\log\left(\frac{p_{i,r}}{\max\left(\sum_{j=1,j\neq r}^{R} p_{i,j}, \epsilon\right)}\right)\right) \tag{29}$$

where $p_{i,r}$ is the probability of the ith sample belonging to the rth class and is obtained from the rth entry of $[c(\mathbf{H}^{(l)})]_i$ or $c[(\mathbf{Z}^{(\mathbf{l})})]_i$. Namely, $p_{i,r} = [c(\mathbf{H}^{(l)})]_{i,r}$ or $p_{i,r} = [c(\mathbf{Z}^{(l)})]_{i,r}$. Here ϵ is a tiny value avoiding the divide-by-zero error. A higher η_i indicates that the importance of the ith sample should be larger if it is incorrectly classified, and should be smaller otherwise. For a correctly predicted node, the weight of it would decrease remarkably if $p_{i,r}$ is higher. This indicates that the model should pay less attention to correct predictions with high confidence. As for a misclassified node, the weight of it would grow up considerably with a higher $p_{i,r}$, attributed to the reason that the node prediction result is much against the ground truth.

With the weighted node embedding obtained by (21), the objective of the proposed AGNN is the cross-entropy loss function, i.e.,

$$\mathcal{L} = -\sum_{i\in\Omega}\sum_{j=1}^{c} \mathbf{Y}_{ij} \ln \mathbf{S}_{ij} \tag{30}$$

which only works on nodes in the training set Ω to perform the semi-supervised classification task.

Algorithm 1 Alternating Graph-Regularized Neural Network

Input: Adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, feature matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, hyperparameters λ, ρ, θ_1, and θ_2;
Output: Graph embedding $\mathbf{S} \in \mathbb{R}^{n\times c}$.

```
while not convergent do
  for l = 1 → t do
    Compute the output H^(l) of the lth GCL via Eq. (18);
    Compute the output Z^(l) of the lth GEL via Eq. (19);
  end for
  Initialize weights {π_i}_{i∈Ω} by π_i = (1/|Ω|);
  for l = 1 → t do
    Update sample weights {π_i}_{i∈Ω} for the lth GCL via Eqs. (27), (28) and (29);
    Calculate the classifier weight α^(l) for the lth GCL via Eqs. (22) and (24);
    Update sample weights {π_i}_{i∈Ω} for the lth GEL via Eqs. (27), (28) and (29);
    Calculate the classifier weight β^(l) for the lth GEL via Eqs. (23) and (25);
  end for
  Obtain weighted embeddings S via Eqs. (26) and (21);
  Update all trainable parameters via back propagation;
end while
return Weighted graph embedding S.
```

C. Model Analysis

Algorithm 1 depicts the procedure of AGNN. In general, the procedure of AGNN is divided into two parts: forward computation of multiple network layers and calculation on weighted graph embedding $\mathbf{S}$ via the variant of Adaboost. Given weight matrix $\mathbf{W}_g^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$, the computational complexity for the lth GCL is linear to the number of edges $|\mathcal{E}|$. Namely, it is $\mathcal{O}(|\mathcal{E}|d_{l-1}d_l)$. As to the lth GEL, the computational complexity is $\mathcal{O}(|\mathcal{E}|d_l + nmd_l)$. Consequently, the forward computation of a basic block with a GCL and a GEL is approximately $\mathcal{O}(|\mathcal{E}|d_{l-1}d_l + nmd_l)$. Owing to $d_l \ll \min(n, m)$, GEL does not significantly increase the computational cost of the networks.

In light of previous analysis, both GCL and GEL are approximations of optimization problems with respect to graph regularization, attributed to which they can be considered as two distinct layers. Hence, AGNN can be approximately regarded as an alternating optimization procedure of problems (2) and (8). The difference between the two layers is that the former optimization performs graph convolutions, and the latter optimization is a sparse graph-regularized projection from the original feature space. In a nutshell, the proposed AGNN is a block-wise graph neural network that simultaneously considers cross-layer connection and aggregation of multi-hop information, which is beneficial to obtaining reliable high-order neighborhood embeddings before conducting information fusion. The primary differences to existing models are summarized as follows.

1) Different from methods that directly combine node embeddings from outputs of varied layers (e.g., AdaGCN [20]), AGNN gets rid of inaccurate predictions

of deep layers via periodic projection from original feature space to latent embeddings.

2) Instead of widely used additive connections from previous layers, AGNN establishes an optimization-inspired GEL module which is derived from the Laplacian-based graph regularization problem. This makes the initial features propagate to each GCL dexterously with less information loss.

IV. Experimental Analysis

In this section, comprehensive experiments are conducted including evaluation against several state-of-the-art models and ablation studies. All experiments are run on a platform with AMD R9-5900X CPU, NVIDIA GeForce RTX 3060 12 GB GPU, and 32 GB RAM.

A. Experimental Setup

For the following experiments, we compare the proposed AGNN with numerous methods. Apart from classical baselines (MLP and Chebyshev [41]), other state-of-the-art methods can be divided into two categories: vanilla GNN-based models (GraphSAGE [42], GAT [43], and ScatteringGCN [18]), and multilayer or high-order information-based GCN methods (GCN [10], APPNP [26], JK-Net [19], SGC [38], ClusterGCN [37], GCNII [24], SSGC [35], and AdaGCN [20]). In particular, APPNP, SGC, and SSGC propagate node information via the proposed high-order filters, where numbers of order can be regarded as numbers of layers for other multilayer approaches. The compared models are demonstrated in detail as follows.

1) *MLP* is a classical baseline for classification, which is a multilayer perceptron architecture with a softmax function as the classifier.
2) *Chebyshev* is a GCN-like baseline that adopts Chebyshev filters to perform graph convolutions with the given node features and the topology network.
3) *GCN* conducts a variant of convolution on the graph, which is exactly the first-order approximation of the Chebyshev polynomial.
4) *GraphSAGE* constructs a graph neural network that explores node embeddings through sampling and accumulating features from local neighbors of a node.
5) *GAT* is a graph neural network adopting an attention mechanism to explore node attributes across the graph, which enables the implicit assignment of weights to distinct nodes in a neighborhood.
6) *JK-Net* dexterously exploits various neighborhood ranges of nodes via a jumping knowledge structure that considers residual connections.
7) *SGC* proposes a faster variant of GCN via successively removing nonlinearities and collapsing weight matrices between consecutive layers.
8) *APPNP* leverages personalized PageRank to improve the performance of GCN-like models, which derives an improved propagation scheme.
9) *ClusterGCN* is a GCN-based framework that samples a group of nodes by a graph clustering algorithm, which alleviates the over-smoothing problem via a diagonal enhancement architecture.
10) *GCNII* is a variant of GCN with residual connection and identity mapping, which effectively alleviates the over-smoothing phenomenon.
11) *ScatteringGCN* builds an augmented GCN with geometric scattering transforms and residual convolutions to alleviate the over-smoothing issue.
12) *SSGC* develops a variant of GCN by adopting a modified Markov diffusion kernel, which explores the global and local contexts of nodes.
13) *AdaGCN* integrates learned knowledge from distinct layers of GCN in an Adaboost way, which updates layer weights iteratively.

In this article, eight different graph-structural datasets are adopted to evaluate the performance of numerous methods, as listed below.

1) *Citeseer*[1] is a benchmark dataset for literature citation networks, where nodes represent papers and edges represent citations between them.
2) *CoraFull*[2] is the larger version of Cora dataset, which is another well-known citation network. Herein, each node denotes paper and edge stands for citation. All nodes are classified according to their topics.
3) *Chameleon*[3] contains node relationships of a large number of articles on a topic of the English Wikipedia website, where edges represent the mutual links among articles.
4) *BlogCatalog*[4] includes a large number of bloggers and their social relationships from the website. Node features are extracted from the keywords of user information and all bloggers are divided into six distinct types.
5) *ACM*[5] is a paper network where each node denotes a paper. Different from citation networks, edges connect papers that share the same authors.
6) *Flickr*[6] is a social network that records relationships among users from an image and video hosting website. All users are grouped into nine categories on the basis of their personal interests.
7) *UAI*[7] is a dataset for the test of GCN on community detection, which is a webpage citation network. Nodes representing webpages are collected from multiple universities and each edge denotes the citation.
8) *Actor*[8] is a subgraph of the film–director–actor–writer network, which only includes the connections of various actors. Each edge represents the co-occurrence of two actors on the same Wikipedia page.

A statistical summary of these datasets is demonstrated in Table I. For fair comparison and avoiding undesired influence raised by data distribution, we shuffle all datasets and

[1] https://linqs.soe.ucsc.edu/data
[2] https://github.com/shchur/gnn-benchmark#datasets
[3] https://github.com/benedekrozemberczki/MUSAE/
[4] https://networkrepository.com/soc-BlogCatalog.php
[5] https://github.com/Jhy1993/HAN
[6] https://github.com/xhuang31/LANE
[7] https://github.com/zhumeiqiBUPT/AM-GCN
[8] https://github.com/CUAI/Non-Homophily-Large-Scale

TABLE I
BRIEF STATISTICS OF ALL GRAPH DATASETS AND DATA SPLIT MODES

Datasets	# Nodes	# Edges	# Features	# Classes	# Train	# Valid	# Test	Data types
Citeseer	3,327	4,732	3,703	6	120	500	1,000	Citation network
CoraFull	19,793	63,421	8,710	70	1,400	500	1,000	Citation network
Chameleon	2,277	18,050	2,325	5	100	500	1,000	Link network
BlogCatalog	5,196	171,743	8,189	6	120	500	1,000	Social network
ACM	3,025	13,128	1,870	3	60	500	1,000	Paper network
Flickr	7,575	239,738	12,047	9	180	500	1,000	Social network
UAI	3,067	28,311	4,973	19	380	500	1,000	Citation network
Actor	7,600	15,009	932	5	100	500	1,000	Social network

TABLE II
PERFORMANCE (ACCURACY) COMPARISON WITH 20 LABELED SAMPLES PER CLASS AS SUPERVISION SIGNALS, WHERE THE HIGHEST ACCURACY IS HIGHLIGHTED IN RED. THE LAST COLUMN SHOWS THE AVERAGE RANKS OF THE PERFORMANCE OF DIFFERENT METHODS. FOR MULTILAYER OR MULTI-ORDER INFORMATION-BASED MODELS, THE OPTIMAL LAYER NUMBERS OR ORDERS ARE RECORDED IN BRACKETS

Methods / Datasets	Citeseer	CoraFull	Chameleon	BlogCatalog	ACM	Flickr	UAI	Actor	Avg Ranks
MLP	0.366	0.051	0.286	0.646	0.812	0.431	0.188	0.224	13.3
Chebyshev [41]	0.693	0.534	0.217	0.357	0.829	0.304	0.215	0.182	13.4
GraphSAGE [42]	0.620	0.521	0.437	0.525	0.886	0.286	0.483	0.191	13.0
GAT [43]	0.683	0.571	0.460	0.681	0.889	0.429	0.597	0.246	7.38
ScatteringGCN [18]	0.679	0.519	0.410	0.690	0.890	0.419	0.364	0.214	11.4
GCN [10]	0.697 (2)	0.567 (2)	0.447 (2)	0.711 (2)	0.875 (2)	0.414 (2)	0.498 (2)	0.240 (4)	9.25
JK-Net [19]	0.703 (4)	0.568 (2)	0.475 (20)	0.747 (16)	0.892 (8)	0.547 (2)	0.494 (18)	0.224 (18)	6.38
APPNP [26]	0.698 (4)	0.576 (4)	0.404 (2)	0.813 (8)	0.885 (4)	0.521 (2)	0.560 (2)	0.212 (4)	7.75
SGC [38]	0.697 (10)	0.583 (2)	0.445 (2)	0.716 (2)	0.887 (2)	0.410 (2)	0.571 (2)	0.247 (4)	7.50
ClusterGCN [37]	0.681 (2)	0.576 (2)	0.449 (2)	0.731 (2)	0.893 (2)	0.483 (2)	0.525 (2)	0.239 (8)	6.75
GCNII [24]	**0.710 (12)**	0.576 (18)	0.449 (16)	0.845 (12)	0.901 (12)	0.545 (12)	0.619 (14)	0.238 (10)	3.50
SSGC [35]	0.702 (20)	0.575 (4)	0.446 (2)	0.760 (2)	0.889 (2)	0.478 (2)	0.523 (10)	0.248 (2)	6.50
AdaGCN [20]	0.663 (2)	0.587 (10)	0.479 (4)	0.800 (2)	0.894 (2)	0.552 (2)	0.588 (2)	0.230 (2)	4.88
AGNN w/o Adaboost	0.689 (2)	0.564 (2)	0.440 (2)	0.766 (10)	0.888 (2)	0.503 (20)	0.574 (10)	0.254 (16)	7.13
AGNN	0.703 (4)	**0.589 (6)**	**0.492 (14)**	**0.849 (10)**	**0.903 (8)**	**0.584 (4)**	**0.647 (6)**	**0.256 (6)**	**1.13**

randomly select 20 labeled samples per class for training, 500 samples for validation, and 1000 samples for testing.

To provide a fair test bed for all compared methods, we list some hyperparameters in experiments. Learning rates of these methods are fixed as 0.01 or 0.005, which are preferred to be smaller when more network layers are utilized. For all GNN-based methods, we fix the number of hidden units at each layer as 128 or 16. Other method-specific hyperparameters are fixed as their settings in original papers.

As for the proposed AGNN, we also apply the same hidden layers as compared methods. The learning rates are also selected from 0.01 and 0.005. In general, a deeper AGNN requires a smaller learning rate, and we adopt learning rate adaptation via decreasing it when there is no loss drop for a period of training epochs. The Adam optimizer is adopted and the weight decay is fixed as 5×10^{-4}. The activation function $\sigma(\cdot)$ is $\tanh(\cdot)$ for weak classifiers while $\mathrm{ReLU}(\cdot)$ for GCL. As for the thresholds in the MSReLU function of GEL, we fix them as $\theta_1 = 0.02$ and $\theta_2 = 0.04$. For the Adaboost strategy, the tiny value in (29) is fixed as 10^{-4}.

B. Experimental Results

1) Performance Comparison: First of all, we compare the performance of the proposed AGNN with all selected approaches. Table II exhibits the semi-supervised classification accuracy on eight datasets. In pursuit of conducting the ablation study and validating the effectiveness of the designed network structure, we further examine the performance of AGNN without the Adaboost framework (dubbed AGNN w/o AdaBoost), which does not aggregate embeddings of all network layers but directly outputs predictions of the final GEL. Because multilayer or multi-order information-based models aim to improve GCN via mining information from deep layers, we record the highest accuracy of these models and the corresponding numbers of layers. The optimal numbers of layers or orders of neighbors are shown in brackets. We run all experiments ten times and record the average accuracy.

To validate the statistical significance of the experimental results, we follow [12] and adopt Friedman test. The average ranks of all compared models are recorded in the last column of Table II, on the basis of which we obtain the Friedman testing score $F_F = 10.57$. With 15 compared models and eight test datasets, the critical value is 1.794 for $\alpha = 0.05$, which indicates that F_F is higher than the critical value. Thus, we can reject the null hypothesis, which points out that the performance of all compared methods is significantly different with a confidence level at 95%.

From experimental results, we draw the following conclusions. First, the experimental results indicate that the proposed AGNN attains encouraging performance and outperforms the other methods by a considerable margin on most datasets. Second, it can be observed that AGNN obtains the optimal classification accuracy with more layers. In most cases, AGNN achieves the best performance with more than 6 layers.

TABLE III

ACCURACY COMPARISON (GCN, SGC, CLUSTERGCN, SSGC, ADAGCN, AND AGNN) WITH VARIOUS NUMBERS OF LAYERS ON CHAMELEON, CORAFULL, FLICKR AND UAI DATASETS. THE OPTIMAL NUMBERS OF LAYERS FOR EACH METHOD ARE HIGHLIGHTED WITH "*," AND THE BEST PERFORMANCE OF DIFFERENT METHODS WITH THE SAME NUMBER OF LAYERS IS HIGHLIGHTED IN RED

Datasets	Models	2-layer	4-layer	6-layer	8-layer	10-layer	12-layer	14-layer	16-layer	18-layer	20-layer
CoraFull	GCN [10]	**0.567***	0.495	0.451	0.443	0.408	0.376	0.332	0.204	0.119	0.019
	JK-Net [19]	**0.568***	0.534	0.531	0.493	0.553	0.506	0.456	0.523	0.527	0.530
	APPNP [26]	0.569	**0.576***	0.560	0.561	0.550	0.556	0.552	0.549	0.547	0.544
	SGC [38]	**0.583***	**0.576**	0.562	0.551	0.534	0.512	0.495	0.474	0.441	0.418
	ClusterGCN [37]	**0.576***	0.518	0.494	0.475	0.442	0.391	0.337	0.264	0.257	0.214
	GCNII [24]	0.539	0.536	0.558	0.565	0.568	0.571	0.568	0.574	**0.576***	0.565
	SSGC [35]	0.572	**0.575***	0.572	0.561	0.562	0.561	0.541	0.564	0.562	0.537
	AdaGCN [20]	0.552	0.553	0.571	0.573	**0.587***	**0.579**	**0.586**	0.575	0.564	0.535
	AGNN w/o Adaboost	**0.564***	0.544	0.532	0.554	0.544	0.554	0.523	0.536	0.545	0.541
	AGNN	0.570	0.574	**0.589***	**0.583**	0.580	0.568	0.565	**0.577**	**0.584**	**0.574**
BlogCatalog	GCN [10]	**0.697***	0.548	0.231	0.125	0.142	0.154	0.187	0.164	0.159	0.171
	JK-Net [19]	0.725	0.711	0.693	0.711	0.670	0.724	0.696	**0.747***	0.668	0.698
	APPNP [26]	0.791	0.810	0.811	**0.813***	0.809	0.806	0.809	0.811	0.805	0.804
	SGC [38]	**0.716***	0.616	0.490	0.394	0.313	0.238	0.232	0.225	0.220	0.237
	ClusterGCN [37]	**0.731***	0.542	0.395	0.256	0.171	0.192	0.182	0.176	0.172	0.171
	GCNII [24]	**0.816**	0.813	0.799	0.802	0.843	**0.845***	0.810	**0.838**	0.801	0.796
	SSGC [35]	**0.760***	0.744	0.744	0.736	0.683	0.728	0.726	0.723	0.722	0.661
	AdaGCN [20]	**0.800***	0.723	0.678	0.682	0.681	0.684	0.678	0.688	0.684	0.688
	AGNN w/o Adaboost	0.762	0.745	0.746	0.741	**0.766***	0.736	0.737	0.751	0.754	0.748
	AGNN	0.720	**0.824**	**0.824**	0.805	**0.849***	**0.815**	**0.820**	0.814	**0.808**	**0.814**
Flickr	GCN [10]	**0.414***	0.127	0.161	0.091	0.100	0.092	0.089	0.091	0.094	0.095
	JK-Net [19]	**0.547***	0.421	0.392	0.418	0.422	0.409	0.439	0.445	0.343	0.345
	APPNP [26]	**0.521***	0.502	0.461	0.485	0.465	0.475	0.468	0.474	0.487	0.464
	SGC [38]	**0.410***	0.337	0.220	0.197	0.154	0.179	0.167	0.160	0.155	0.154
	ClusterGCN [37]	**0.483***	0.398	0.314	0.322	0.217	0.198	0.184	0.143	0.112	0.103
	GCNII [24]	0.489	0.499	0.514	0.511	0.530	**0.545***	0.523	0.538	0.524	0.524
	SSGC [35]	**0.478***	0.433	0.388	0.356	0.340	0.328	0.320	0.315	0.309	0.304
	AdaGCN [20]	**0.552***	0.546	**0.539**	**0.539**	0.542	**0.545**	**0.545**	**0.545**	0.544	0.546
	AGNN w/o Adaboost	0.481	0.494	0.488	0.490	0.495	0.494	0.499	0.495	0.493	**0.503***
	AGNN	**0.560**	**0.584***	0.529	0.521	**0.543**	0.511	0.522	0.535	**0.545**	**0.557**
UAI	GCN [10]	**0.498***	0.301	0.195	0.202	0.175	0.186	0.192	0.123	0.109	0.080
	JK-Net [19]	0.474	0.467	0.466	0.492	0.467	0.485	0.490	0.484	**0.494***	0.476
	APPNP [26]	**0.560***	0.507	0.484	0.531	0.520	0.510	0.493	0.540	0.526	0.486
	SGC [38]	**0.571***	0.481	0.258	0.141	0.136	0.157	0.126	0.080	0.072	0.036
	ClusterGCN [37]	**0.522***	0.516	0.452	0.358	0.244	0.265	0.224	0.198	0.167	0.165
	GCNII [24]	**0.598**	0.602	0.611	0.608	0.617	**0.622***	0.619	0.618	0.619	0.615
	SSGC [35]	0.508	0.500	0.509	0.522	**0.523***	0.518	0.519	0.523	0.519	0.521
	AdaGCN [20]	**0.588***	0.583	0.581	0.582	0.582	0.580	0.579	0.576	0.581	0.582
	AGNN w/o Adaboost	0.567	0.561	0.551	0.565	**0.574***	0.562	0.560	0.562	0.545	0.563
	AGNN	0.572	**0.630**	**0.647***	**0.640**	**0.641**	**0.638**	**0.623**	**0.623**	**0.622**	**0.621**

Although other compared methods sometimes also achieve better performance with more layers, two or four layers are still the best choice for most datasets. Last but not least, AGNN w/o Adaboost obtains competitive classification results and sometimes gets higher accuracy with over ten layers (Flickr, UAI, and Actor datasets). This phenomenon indicates that AGNN w/o Adaboost guarantees the discrimination of node embeddings and the reliability of deep layers. From the ablation study, we find that AGNN performs satisfactorily compared with AGNN w/o Adaboost, which indicates the effectiveness of the proposed improved Adaboost. In addition, the experimental results point out that the optimal layer numbers of AGNN are not always higher than that of AGNN w/o Adaboost. This may be owing to the fact that the aggregation process enables the model to obtain competitive accuracy with fewer layers. Besides, deep-layer models do not always mine more information on some datasets, which depends on the topological structure of datasets. However, it is significant that AGNN obtains higher accuracy with more layers on some datasets, especially on Chameleon and ACM. In a nutshell, these observations reveal that the performance leading of AGNN is significant with larger numbers of layers.

2) Performance With Deep Layers: Because the proposed AGNN aims to tackle the over-smoothing issue and extract more distinctive characteristics with deep layers, we further conduct comparing experiments on some multilayer or multi-order information-based GCN methods to explore accuracy trends with varying numbers of layers.

Table III demonstrates the classification accuracy of selected methods with 20 labeled nodes for each class, from which we have the following observation. As most existing works have analyzed, GCN encounters a dramatic accuracy plunge with over two GCLs on all tested datasets. In contrast, the performance decline of other compared methods is not as severe as GCN, and some of them even gain marginal performance improvement as the number of layers rises. Nevertheless, several compared approaches still attain the highest accuracy with a two-layer architecture, and sometimes performance

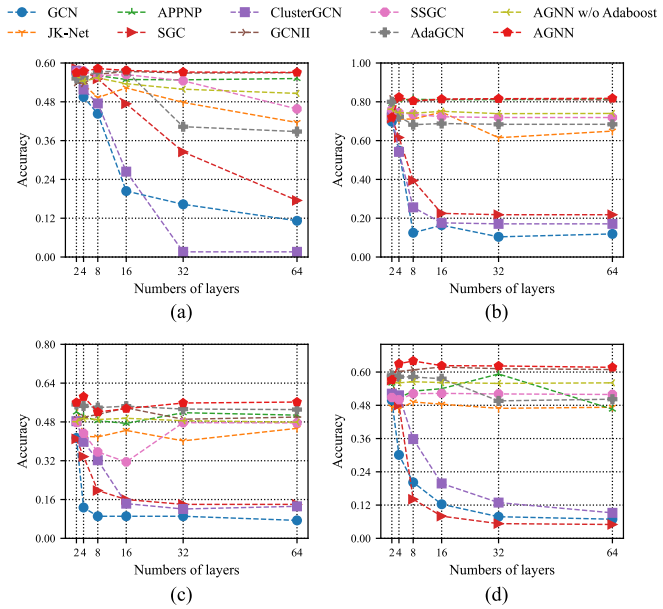


Fig. 4. Performance of baselines and the proposed AGNN with 2/4/8/16/32/64 layers. (a) CoraFull. (b) BlogCatalog. (c) Flickr. (d) UAI.

may dwindle as the numbers of layers are larger. In general, AdaGCN which also integrates multi-hop node embeddings behaves favorably on most datasets. Nonetheless, as we have discussed, it still suffers from indistinguishable node features from deep layers on some datasets (e.g., BlogCatalog), owing to which deep AdaGCN leads to unsatisfactory performance. We can discover that AGNN achieves competitive performance with fewer layers, and often outperforms other models with more stacked layers. Above all, the proposed model maintains accuracy at a high level with more layers, and a suitable multilayer AGNN is helpful to exploring representative high-order node features. As for AGNN w/o Adaboost, although it does not always outperform other models, it succeeds in lessening the negative influence of over-smoothing compared with other models and performs satisfactorily on all tested datasets. We also visualize the performance trends of compared methods in Fig. 4 with more layers (32 and 64 layers), which intuitively shows the ability of compared models to overcome over-smoothing. AGNN generally performs the best with deeper layers. We find that AGNN also gains marginal improvement or keeps stable with 32/64 layers, which indicates that it gets rid of over-smoothing. Generally, AGNN with no more than 20 layers can achieve the optimal accuracy, as recorded in Table II. In conclusion, these experimental results point out that the proposed AGNN has a powerful ability to mine underlying node embeddings with a deep network architecture.

3) Weak Classifier Weight Distribution: In this section, we explore the assigned weights of weak classifiers in the proposed method with varying numbers of layers, as shown in Fig. 5. The weight assignments demonstrate that shallow layers account for a significant portion of final predicted results, indicating that classification problems of most nodes can be effectively solved by extracting representations of one or two hops of neighbors. In general, the top four layers (top two blocks) of AGNN play the most critical role in the final prediction, and the rest layers complement the prediction with more high-order information. Fig. 5 reveals that AGNN achieves the best performance with eight layers on both two selected datasets, indicating that multilayer models are essential for improving accuracy via exploring remote neighbors. Although Fig. 5 shows that AGNN with more than eight layers is not the optimal selection, the improved Adaboost can maintain the classification accuracy of extremely deep networks by assigning tiny weights to deep layers, if most nodes have been correctly classified through shallow layers. In a word, a multilayer architecture often benefits the embedding learning, and AGNN attempts to leverage high-order information at the best.

TABLE IV

IMPACT OF IDENTITY FUNCTION (IF), ST, RELU, AND MSRELU IN GEL, WHERE $\theta = 0.02$ (ST), $\theta_1 = 0.02$, AND $\theta_2 = 0.04$ (MSRELU). LAYER NUMBERS ARE FIXED AS 4

Methods / Datasets	BlogCatalog	Flickr	UAI	Chameleon
AGNN + IF	0.805	0.568	0.615	0.450
AGNN + ST	0.813	0.561	0.612	0.437
AGNN + ReLU	0.815	0.411	0.594	0.458
AGNN + MSReLU	**0.824**	**0.584**	**0.630**	**0.480**

4) Model Analysis: In this section, we further analyze the proposed model. First, the impact of hyperparameters used in AGNN is discussed. The accuracy changes with respect to λ and ρ on all datasets are demonstrated in Fig. 6, from which we find that the performance of AGNN fluctuates marginally and a suitable choice of two parameters is crucial on most datasets. Overall, AGNN is robust to varied hyperparameters on Citeseer and ACM datasets. Although the optimal selections of hyperparameters differ on other datasets, small values of λ and ρ often lead to undesired performance, especially on CoraFull, Chameleon, UAI, and BlogCatalog datasets. In our previous experiments, we select the optimal combination of these two hyperparameters to obtain better experimental results.

Furthermore, we validate the effectiveness of the designed activation function MSReLU in GEL, as exhibited in Table IV. All parameter settings except those in compared activation functions are the same. We also evaluate the performance of AGNN with identify function and ReLU function. It is noted that ReLU only preserves nonzero entries in the matrix. Experimental results indicate that MSReLU function succeeds in promoting classification accuracy compared with taking other functions as activation functions, attributed to the ability of making sparse outputs closer to original features. Sometimes, AGNN with ReLU encounters severe performance decline (e.g., Flickr and UAI datasets). This is because it ignores negative entries in the feature matrix, which often results in the information loss. In a word, these observations suggest that a suitable MSReLU function benefits the learning of more accurate and robust node embeddings.

5) Convergence Analysis: Convergence curves of the proposed AGNN on BlogCatalog, Flickr, Actor, and Chameleon datasets are demonstrated in Fig. 7. These curves indicate that loss values of AGNN drop as the number of iterations

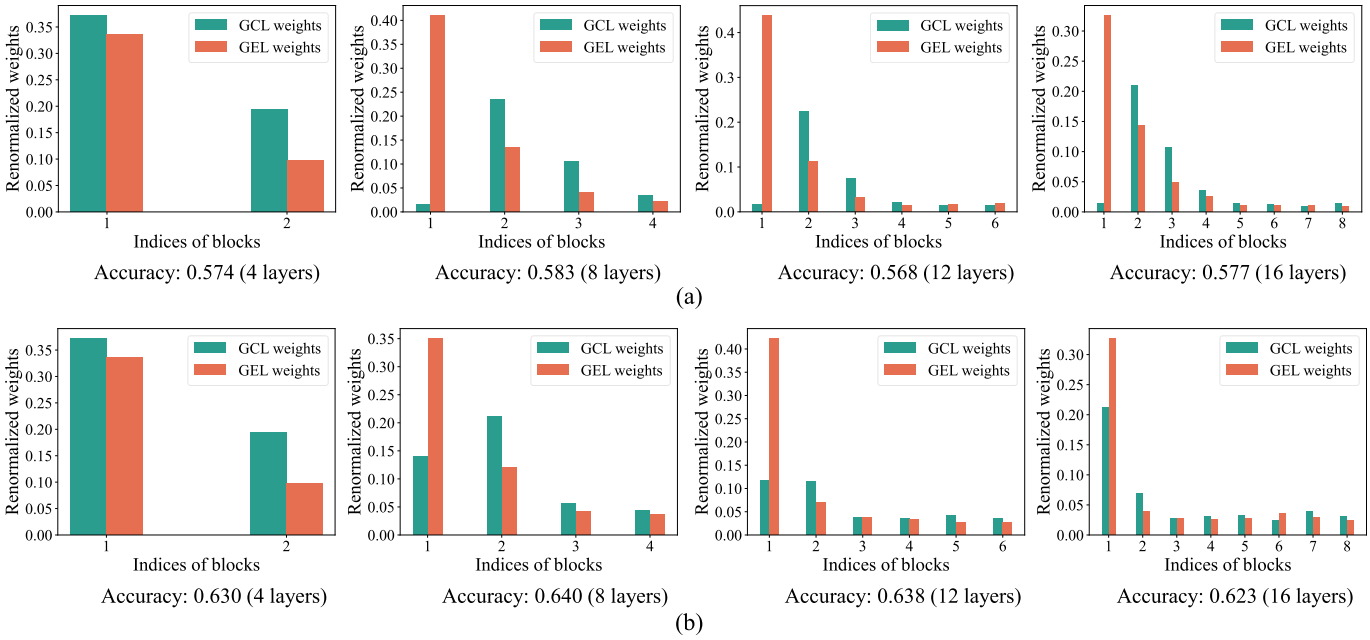


Fig. 5. Weight distribution of AGNN with 4/8/12/16 layers for each weak classifier on CoraFull and UAI datasets. (a) CoraFull. (b) UAI.

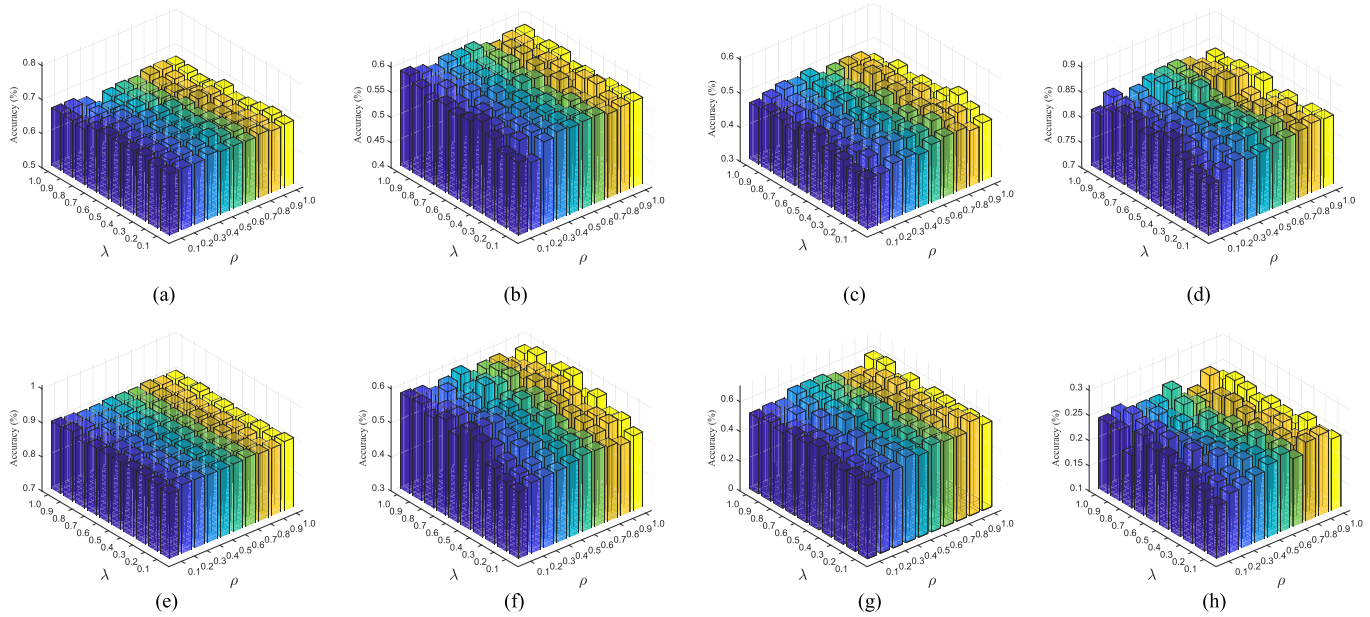


Fig. 6. Parameter sensitivity of AGNN with respect to λ and ρ on various datasets. (a) Citeseer. (b) CoraFull. (c) Chameleon. (d) BlogCatalog. (e) ACM. (f) Flickr. (g) UAI. (h) Actor.

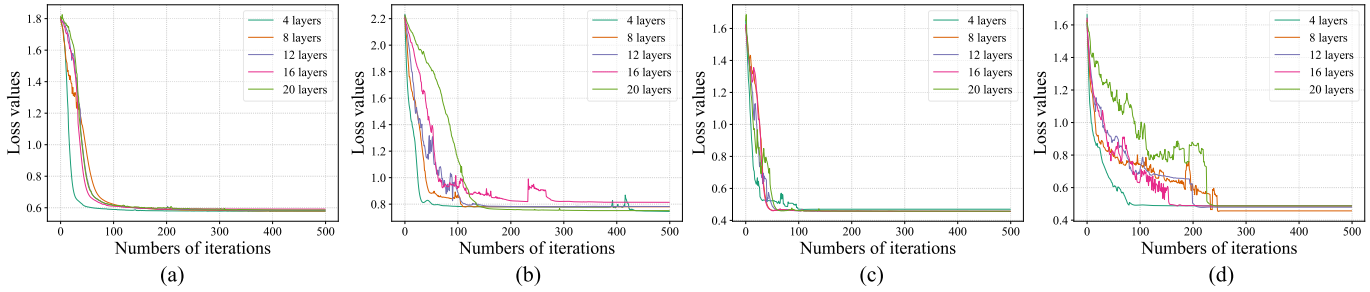


Fig. 7. Training convergence curves of AGNN with varying numbers of layers ranging in $\{4, 8, \ldots, 20\}$ on BlogCatalog, Flickr, Actor, and Chameleon datasets. (a) BlogCatalog. (b) Flickr. (c) Actor. (d) Chameleon.

increases and are finally convergent. Although loss values may sometimes fluctuate, the overall trends of curves are suggestive of their convergence. The fluctuation during training is caused by the Adaboost strategy that reassigns sample weights at each iteration. Nevertheless, loss values are stable and converge eventually. The figure also shows that AGNN with shallow

layers generally converges more quickly than that with deep layers, due to the larger solution space caused by more trainable parameters. Overall, AGNN with all numbers of layers leads to similar convergent points. However, AGNN with deeper layers generally reaches lower values of cross-entropy, indicating the ability of exploring multi-hop embeddings. It is noteworthy that AGNN with deep layers does not always correspond to better convergence, attributed to the various data distributions of different datasets.

V. Conclusion

In this article, we proposed an AGNN to improve the performance of GCN in terms of semi-supervised node classification tasks, which coped with the over-smoothing issue that occurred in most GCN-based models. We first reviewed the concept of GCN and validated that it was an approximation of a graph-regularized optimization problem. Next, we elaborated on the proposed GEL, which was derived from another graph-regularized optimization objective formulating the transformation from the original feature space to the intermediate graph embedding space at each layer. Therefore, GEL allowed the model to carry low-order information from the input to deep layers. Theoretically, the proposed AGNN alternately propagated node information on the basis of two graph-constrained problems. Furthermore, an improved Adaboost strategy was leveraged to integrate hidden graph representations from all layers. Due to more reliable and distinguishable node embeddings learned from GCL and GEL, this strategy could obtain more accurate predictions. Extensive experiments validated that the proposed method succeeded in promoting the performance of GCN with deeper layers. In the future, we will devote ourselves into further investigation of multilayer GCN with techniques such as attention mechanism and residual networks.

References

[1] Z. Zhang et al., "SHNE: Semantics and homophily preserving network embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 12, 2021, doi: 10.1109/TNNLS.2021.3116936.

[2] H. Zhong et al., "Graph contrastive clustering," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9204–9213.

[3] M.-S. Chen, C.-D. Wang, and J.-H. Lai, "Low-rank tensor based proximity learning for multi-view clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 5076–5090, May 2023.

[4] Y.-H. Chen, L. Huang, C.-D. Wang, and J.-H. Lai, "Hybrid-order gated graph neural network for session-based recommendation," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1458–1467, Mar. 2022.

[5] Q. Wang, Y. Wei, J. Yin, J. Wu, X. Song, and L. Nie, "DualGNN: Dual graph neural network for multimedia recommendation," *IEEE Trans. Multimedia*, vol. 25, pp. 1074–1084, 2023, doi: 10.1109/TMM.2021.3138298.

[6] Z.-H. Deng, C.-D. Wang, L. Huang, J.-H. Lai, and S. Y. Philip, "G^3SR: Global graph guided session-based recommendation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 24, 2022, doi: 10.1109/TNNLS.2022.3159592.

[7] Y. Yang, H. Li, X. Li, Q. Zhao, J. Wu, and Z. Lin, "SOGNet: Scene overlap graph network for panoptic segmentation," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 12637–12644.

[8] X. Xu, T. Wang, Y. Yang, A. Hanjalic, and H. T. Shen, "Radial graph convolutional network for visual question generation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1654–1667, Apr. 2021.

[9] Y. Xu, C. Han, J. Qin, X. Xu, G. Han, and S. He, "Transductive zero-shot action recognition via visually connected graph convolutional networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3761–3769, Aug. 2021.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

[11] Z. Chen, L. Fu, J. Yao, W. Guo, C. Plant, and S. Wang, "Learnable graph convolutional network and feature fusion for multi-view learning," *Inf. Fusion*, vol. 95, pp. 109–119, Jul. 2023.

[12] M. Shang, Y. Yuan, X. Luo, and M. Zhou, "An α–β-divergence-generalized recommender for highly accurate predictions of missing user preferences," *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 8006–8018, Aug. 2022.

[13] X. Luo, H. Wu, Z. Wang, J. Wang, and D. Meng, "A novel approach to large-scale dynamically weighted directed network representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 1–17, Dec. 2021, doi: 10.1109/TPAMI.2021.3132503.

[14] Y. Yuan, X. Luo, M. Shang, and Z. Wang, "A Kalman-filter-incorporated latent factor analysis model for temporally dynamic sparse data," *IEEE Trans. Cybern.*, early access, Jul. 25, 2022, doi: 10.1109/TCYB.2022.3185117.

[15] Q. Wang, H. Jiang, M. Qiu, Y. Liu, and D. Ye, "TGAE: Temporal graph autoencoder for travel forecasting," *IEEE Trans. Intell. Transp. Syst.*, early access, Sep. 13, 2022, doi: 10.1109/TITS.2022.3202089.

[16] B. Chen, B. Wu, A. Zareian, H. Zhang, and S.-F. Chang, "General partial label learning via dual bipartite graph autoencoder," in *Proc. 34th AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 10502–10509.

[17] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 976–985.

[18] Y. Min, F. Wenkel, and G. Wolf, "Scattering GCN: Overcoming oversmoothness in graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 14498–14508.

[19] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 5449–5458.

[20] K. Sun, Z. Zhu, and Z. Lin, "AdaGCN: Adaboosting graph convolutional networks into deep models," in *Proc. 9th Int. Conf. Learn. Represent.*, 2021, pp. 1–15.

[21] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.

[22] M. Eliasof, E. Haber, and E. Treister, "PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 3836–3849.

[23] H. Chen et al., "AMA-GCN: Adaptive multi-layer aggregation graph convolutional network for disease prediction," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 2235–2241.

[24] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, Jul. 2020, pp. 1725–1735.

[25] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9266–9275.

[26] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–15.

[27] K. Xu, H. Huang, P. Deng, and Y. Li, "Deep feature aggregation framework driven by graph convolutional network for scene classification in remote sensing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5751–5765, Oct. 2022, doi: 10.1109/TNNLS.2021.3071369.

[28] D. He et al., "Community-centric graph convolutional network for unsupervised community detection," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3515–3521.

[29] W. Yu and Z. Qin, "Graph convolutional network for recommendation with low-pass collaborative filters," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 10936–10945.

[30] J. Gan et al., "Multigraph fusion for dynamic graph convolutional network," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 16, 2022, doi: 10.1109/TNNLS.2022.3172588.

[31] L. Chen, J. Li, Q. Peng, Y. Liu, Z. Zheng, and C. Yang, "Understanding structural vulnerability in graph convolutional networks," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 2249–2255.

[32] Z. Xu, Y. Kang, Y. Cao, and Z. Li, "Spatiotemporal graph convolution multifusion network for urban vehicle emission prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3342–3354, Aug. 2021.

[33] Z. Tang, Y. Shen, X. Ma, W. Xu, J. Yu, and W. Lu, "Multi-hop reading comprehension across documents with path-based graph convolutional network," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3905–3911.

[34] M. Jin, H. Chang, W. Zhu, and S. Sojoudi, "Power up! Robust graph convolutional network via graph powering," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 8004–8012.

[35] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *Proc. 9th Int. Conf. Learn. Represent.*, 2021, pp. 1–15.

[36] O. A. Malik, S. Ubaru, L. Horesh, M. E. Kilmer, and H. Avron, "Dynamic graph convolutional networks using the tensor M-product," in *Proc. SIAM Int. Conf. Data Mining*, 2021, pp. 729–737.

[37] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 257–266.

[38] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 6861–6871.

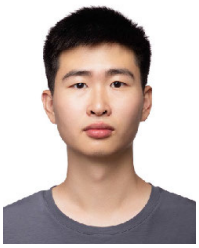
[39] L. Cui, L. Bai, X. Bai, Y. Wang, and E. R. Hancock, "Learning aligned vertex convolutional networks for graph classification," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 10, 2023, doi: 10.1109/TNNLS.2021.3129649.

[40] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[41] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, vol. 29, 2016, pp. 3837–3845.

[42] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1024–1034.

[43] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018, pp. 1–12.



Zhaoliang Chen received the B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2019, where he is currently pursuing the Ph.D. degree with the College of Computer and Data Science.

From 2022 to 2023, he is a Visitor at Faculty of Computer Science, University of Vienna, Vienna, Austria. His current research interests include machine learning, deep learning, graph neural networks, and recommender systems.

Zhihao Wu received the B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2021, where he is currently pursuing the M.S. degree with the College of Computer and Data Science.

He is also currently visiting the Shenzhen Research Institute of Big Data and Chinese University of Hong Kong (Shenzhen), Shenzhen, China. His current research interests include machine learning, multi-view learning, and graph neural networks.

Zhenghong Lin received the B.S. degree from the College of Computer and Data Science, Fujian Agriculture and Forestry University, Fuzhou, China, in 2019. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou.

His current research interests include active learning, machine learning, deep learning, graph neural networks, computer vision, and recommender systems.

Shiping Wang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2014.

He worked as a Research Fellow at Nanyang Technological University, Singapore, from August 2015 to August 2016. He is currently a Professor with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His current research interests include machine learning, computer vision, and granular computing.

Claudia Plant received the Ph.D. degree from the University of Vienna, Vienna, Austria, in 2007.

Currently, she is a Full Professor of data mining at the University of Vienna. She has been published in top-level database and data mining conferences such as KDD and ICDM, as well as in application-related top level journals. Her research focuses on database-related data mining, especially clustering, parameter-free data mining, and integrative mining of heterogeneous data.

Dr. Plant received four Best Paper Awards together with her group, among them the ICDM 2014 Best Paper Award, and the KDD 2021 Best Student Paper Award.

Wenzhong Guo received the B.S. and M.S. degrees in computer science and technology and the Ph.D. degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively.

He completed the Post-Doctoral Fellow with the Institute of Computer Science, National University of Defense and Technology, Changsha, China, in 2013, and was a Senior Visiting Scholar with the Faculty of Engineering, Information and System, University of Tsukuba, Tsukuba, Japan, in 2013. He is currently a Professor with the College of Computer and Data Science, Fuzhou University, and leads the Network Computing and Intelligent Information Processing Laboratory, which is a Key Laboratory of Fujian, China. He is also the Deputy Director at the Laboratory of Spatial Data Mining and Information Sharing, which is a Key Laboratory of Ministry of Education, Beijing, China. He has authored or coauthored more than 130 technical articles in scientific journals and conference proceedings. His current research interests include VLSI physical design, wireless sensor networks, big data, and image processing.

Dr. Guo is also a member of ACM and a Senior Member of the China Computer Federation (CCF).