GCN-SL: GRAPH CONVOLUTIONAL NETWORK WITH STRUCTURE LEARNING FOR DISASSORTATIVE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

In representation learning on the graph-structured data, many popular GNNs may fail to capture long-range dependencies, which leads to their performance degradation. Furthermore, this weakness will be magnified when the concerned graph is disassortative. To solve the above-mentioned issue, we propose a graph convolutional network with structure learning (GCN-SL). The proposed GCN-SL contains two improvements: corresponding to edges and node features, respectively. We build a re-connected adjacency matrix by structure learning from the perspective of edges. Specifically, the re-connected adjacency matrix is built by using a special data preprocessing technique and similarity learning, and can be optimized directly along with GCN-SL parameters. In the aspect of node features, we propose an efficient-spectral-clustering (ESC) and an ESC with anchors (ESC-ANCH) algorithms. The two algorithms can efficiently aggregate feature representations from similar nodes, no matter how far away these similar nodes are from the target node. Both of the two improvements can help GCN-SL capture long-range dependencies, then make GCN-SL is capable of performing representation learning on both disassortative and assortative graphs. Experimental results on a wide range of benchmark datasets illustrate that the proposed GCN-SL outperforms the state-of-the-art GNN counterparts.

1 INTRODUCTION

Graph Neural Networks (GNNs) are powerful for representation learning on graphs with varieties of applications ranging from knowledge graphs to financial networks (Xu et al., 2018; Gilmer et al., 2017; Bronstein et al., 2017; Johnson, 2017; Chen et al., 2018a; Klicpera et al., 2019). In recent years, GNNs have developed many artificial neural networks, e.g., Graph Convolutional Network (GCN) (Kipf & Welling, 2017), Graph Attention Network (GAT) (Velikovi et al., 2017), Graph Isomorphism Network (GIN) (Xu et al., 2019), and GraphSAGE (Hamilton et al., 2017b). For GNNs, each node can iteratively update its feature representation via aggregating the ones of the node itself and its neighbors (Kipf & Welling, 2017; Defferrard et al., 2016; Huang et al., 2018). The neighbors are usually defined as the set of adjacent nodes in a graph, and a diversity of aggregation functions can be adopted to GNNs, e.g., summation, maximum, and mean (Henaff et al., 2015; Pei et al., 2020; Corso et al., 2020).

Convolutional neural networks (CNNs) have developed substantially in recent years and have achieved significant success in a wide range of tasks (Li et al., 2020; Wang et al., 2021b). GCNs are the generalizations of classical CNNs so as to handle graph data such as point could, molecular data, and social networks (Chen et al., 2018b). GCNs are the most attractive GNNs and have been widely applied in a variety of scenarios (Hamilton et al., 2017a; Li et al., 2018). However, one fundamental weakness of GCNs limits the representation ability of GCNs on graph-structured data (Henaff et al., 2020; Zhu et al., 2020). That is GCNs may not capture long-range dependencies in graphs, considering that GCNs update the feature representations of nodes via simply summing the normalized feature representations from all one-hop neighbors (Velikovi et al., 2017; Pei et al., 2020). Furthermore, this weakness will be magnified in disassortative graphs (Henaff et al., 2020; Zhu et al., 2020).

Homophily is a very important principle of many real-world graphs, whereby the linked nodes tend to have similar features and belong to the same class (Zhu et al., 2020). For instance, papers are more likely to cite papers from the same research area in citation networks, and friends tend to have similar age or political beliefs in social networks (Velikovi et al., 2017; Kipf & Welling, 2017). However, there are also settings about "opposites attract" in the real world, leading to graphs with low homophily, i.e., the proximal nodes are usually from different classes and have dissimilar features (Zhang & Zitnik, 2020; Zhu et al., 2019). For example, most people tend to chat with people of the opposite gender in the dating website, and fraudsters are more prefer to contact accomplices than other fraudsters in online gambling networks. The graphs under high homophily are called as assortative graphs, and the graphs under low/medium level of homophily are called as disassortative graphs. Most existing GNNs assume graphs are assortative, including GCNs, therefore they perform poorly on generalizing disassortative graphs even worse than the MLP (Hornik, 1991) that relies only on the node features for classification (Pandit et al., 2007; Zhu et al., 2020).

To solve the above problem, recently some related approaches have been built (Pei et al., 2020; Zhao et al., 2021; Zhu et al., 2020; Wang et al., 2021a), such as Geometric Graph Convolutional Network (Geom-GCN) (Pei et al., 2020), and H_2GCN (Zhu et al., 2020). Although Geom-GCN improves the performance of representation learning of GCNs, the classification performance of Geom-GCN is often unsatisfactory when the concerned datasets are disassortative graphs (Pei et al., 2020). H_2GCN improves the classification performance of GCN, while it is only able to aggregate information from near nodes, resulting in lacking the ability for capturing the features from distant but similar nodes. Nevertheless, it is notable that H_2GCN still has a lot of room for improvement.

In this paper, we propose a novel GNN approach to solve the above-mentioned problem, referred to as the graph convolutional network with structure learning (GCN-SL). Following the spectral clustering (SC) method (Nie et al., 2011), a graph of data points is built according to the distances of the data points in the feature space. Then, the data points are mapped into a new feature space by cutting the graph. In this way, the data points connected closely are usually proximal in the new feature space. Therefore, nodes can aggregate features from similar nodes if SC is employed to process graph-structured data, contributing to that GCN can capture long-range dependencies. It should be noted, the computational complexity of SC is greatly high for large-scale graphs. Hence, we design an efficient-spectral-clustering(ESC) and an ESC with anchors (ESC-ANCH) algorithms to efficiently extract SC features. Then, the extracted SC features combined with the original node features as enhanced features (EF), and EF is utilized to train the proposed GCN-SL model.

Following the research in Zhu et al. (2020), the nodes of the same class always possess similar features, no matter whether the homophily of the concerned graph is high or low. For our approach, it can build a re-connected graph related to the similarities between nodes, and the re-connected graph is optimized jointly with the parameters of GCN-SL. Meanwhile, for dealing with the sparse initial attributes of nodes, a special data preprocessing technique is applied to the original features of nodes. And this data preprocessing technique can also overcome the over-fitting problem that GNNs with structure learning module often face. Afterwards, the original adjacency matrix and the re-connected adjacency matrix are respectively utilized to obtain multiple intermediate representations associated with different rounds of aggregation. Then, we combine several key intermediate representations as the new node embedding and use a learnable weighted vector to highlight the important dimensions of the new node embedding. Finally, we set the result of this calculation as the final node embedding for node classification.

Compared with other GNNs, the contributions of GCN-SL can be summarized as follows. 1) SC is integrated into GNNs for capturing long-range dependencies on graphs, and an ESC and an ESC-ANCH algorithms are proposed to efficiently implement SC on graph-structured data; 2) Our GCN-SL can learn an optimized re-connected adjacency matrix that benefits the downstream prediction task. 3) The special data preprocessing technique can not only help GCN-SL overcome the over-fitting problem that most of GNNs with structure learning module face, but also benefits the generation of the re-connected adjacency matrix; 4) The GCN-SL proposes the improvements for handling disassortative graphs from the aspects of node features and edges, respectively. Meanwhile, GCN-SL combines the two improvements and makes them supplement each other.

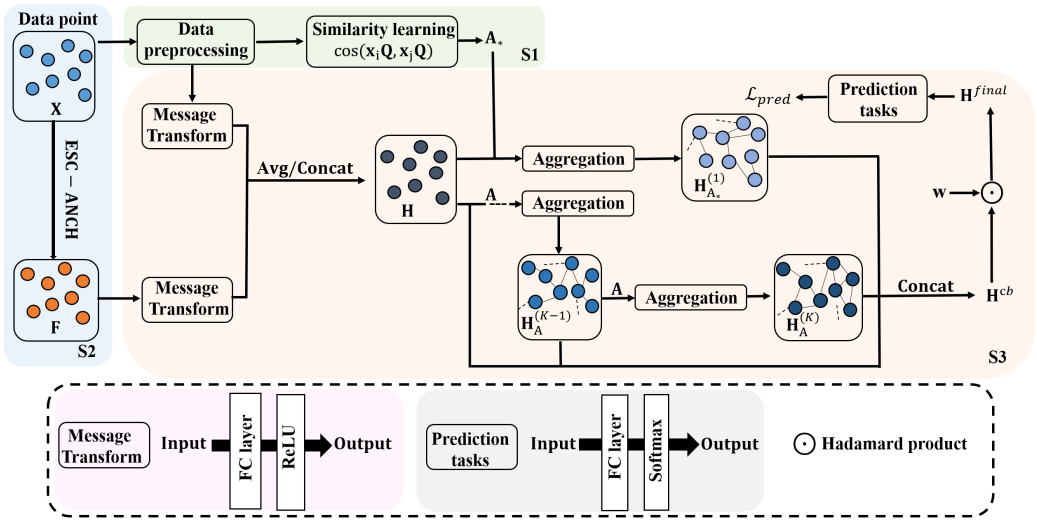


Figure 1: GCN-SL consists of three stages: (S1) *re-connected graph*, (S2) *efficient spectral clustering with anchors*, and (S3) *graph convolutional networks with structure learning*. In (S1), we construct a re-connected adjacency matrix $\mathbf{A}_*$ via similarity learning, and the re-connected graph can be gradually optimized with the training of the GCN-SL model. In (S2), ESC-ANCH generates the SC features $\mathbf{F}$. In (S3), we combine the original features $\mathbf{X}$ and SC features $\mathbf{F}$ as the enhanced features $\mathbf{H}$, and perform feature aggregation on $\mathbf{H}$ by using the $\mathbf{A}_*$ and $\mathbf{A}$, respectively. Furthermore, several key results of aggregation and the $\mathbf{H}$ are combined as the $\mathbf{H}^{cb}$, and then $\mathbf{A}$ and $\mathbf{A}_*$ are used for performing feature aggregation on $\mathbf{H}$, respectively. Finally, a learnable weighted vector $\mathbf{w}$ is used to highlight important dimensions of $\mathbf{H}^{cb}$, which makes GCN-SL adapts to graphs with various levels of homophily.

2 GCN-SL Architecture

We present a novel GNN: GCN-SL for node classification of graph-structured data. Figure 1 shows the pipeline of GCN-SL. The remainder of this section is organized as follows: subsection 2.2 gives the construction details of the re-connected adjacency matrix, subsection 2.3 describes the proposed ESC and ESC-ANCH approaches, subsection 2.4 describes the proposed GCN-SL in detail.

2.1 Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ is the set of nodes or vertices, and $\mathcal{E}$ is the set of edges. $|\mathcal{V}| = n$ is the number of nodes. $v_i \in \mathcal{V}$ is a node and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ is an edge linking v_i and v_j. The neighborhood of a node v_i is defined as $\mathcal{N}_i = \{v_j \in \mathcal{V} | (v_i, v_j) \in \mathcal{E}\}$. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $A_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and $A_{ij} = 0$ if $e_{ij} \notin \mathcal{E}$. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the node feature matrix with $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$ denoting the feature vector of node v_i, and each node is associated with a label y_i. Given a multilayered network and the semantic labels $\mathbf{y}_{lab}$ for a subset of nodes $\mathcal{V}_{lab} \in \mathcal{V}$, where $y \in \mathbf{y}_{lab}$ means one of the C predefined classes. GNNs learn the feature representations of the nodes and graphs by exploiting the graph structure. Then the task of node classification is to predict the label for each node without label $v_i \in \mathcal{V} | v_i \notin \mathcal{V}_{lab}$ according to the feature representation of the corresponding node.

2.2 Re-connected Graph

Most existing GNNs are designed for assortative graphs, where the linked nodes usually possess similar feature representation and belong to the same class. However, there are a large number of disassortative graphs in the real world, where the linked nodes often possess dissimilar features and belong to different classes. To sum up, in practical applications of GNNs, these GNNs designed under the assumption of homophily are greatly inappropriate for the graphs under low/medium homophily.

Fortunately, regardless of the homophily level of graphs, the nodes of the same class always possess high similar features (Chen et al., 2020; Shanthamallu et al., 2020). In this paper, through structure learning, our network can learn a re-connected adjacency matrix that is based on the similarities between nodes and optimized along with the parameters of GCN-SL, which makes the GCN-SL can capture information from more reliable nodes.

A good similarity metric function should be expressively powerful and learnable (Chen et al., 2020). In this work, we set cosine similarity as the metric function to built the similarity matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$. While the sparse initial attributes of nodes in adopted graphs usually result in an invalid similarity matrix $\mathbf{M}$. To overcome the weakness, we apply a special preprocessing to $\mathbf{X}$. Specifically, we randomly choose a feature dimension from $\mathbf{X}$, and then a number greater than 0 is plus to the chosen feature dimension. Since the original features of nodes are greatly sparse, and the values of node features are range between $\{0, 1\}$. To ensure the construction of similarity matrix $\mathbf{M}$, and not affect the similarity of nodes too much, the above positive number is empirically set to 0.5 in all graph-structured data. Meanwhile, Graph structure learning module enables GCN-SL to better fit the downstream task, while it may be easier for the GCN-SL into over-fitting (Chen et al., 2020; Wu et al., 2021). The introduction of additional features not only guarantees the construction of $\mathbf{M}$, but also helps GCN-SL to overcome the over-fitting problem.

The similarity between a pair of nodes can be represented as:

$$M_{ij} = \cos\left(\mathbf{x}_i\mathbf{Q}, \mathbf{x}_j\mathbf{Q}\right) \tag{1}$$

where $\mathbf{Q} \in \mathbb{R}^{d \times p}$ is a learnable weighted matrix. Afterwards, we can obtain the similarity matrix $\mathbf{M}$, the $\mathbf{M}$ is symmetric and the elements in $\mathbf{M}$ range between $[-1, 1]$.

Since a fully connected graph is very computational and might introduce noise, the re-connected adjacency matrix is supposed to be sparse (Chen et al., 2020). Meanwhile, the re-connected adjacency matrix is also expected to be nonnegative and connected. To obtain a non-negative, sparse, and connected adjacency matrix $\mathbf{A}_*$ from $\mathbf{M}$, we use both the KNN and the minimum-threshold methods. Specifically, we first define a positive integer r and a non-negative threshold ϵ. Then, we retain those elements in $\mathbf{M}$ which are greater than ϵ or belong to the set of r largest elements in the corresponding row, and we set the other elements to zero. That is $A_{*ij} = M_{ij}$ if $M_{ij} > \epsilon$ or M_{ij} belong to the set of r largest elements in $M_i \in \mathbb{R}^n$, and $A_{*ij} = 0$ if else. By this way, we obtain the re-connected adjacency matrix $\mathbf{A}_*$. In this paper, the r is uniformly set to 2 for convenience, and the threshold ϵ needs to be searched.

2.3 Efficient Spectral Clustering with Anchors

In graph-structured data, a large number of nodes of the same class possess similar features but are far apart from each other (Henaff et al., 2020). However, GCN simply aggregates the information from first or higher-order neighbors, and the depth of GCN is usually limited (Rong et al., 2020). Obviously, the information from distant but similar nodes is always ignored in GCN. Meanwhile, SC can divide nodes according to the affinities between nodes (Nie et al., 2011). Specifically, the closely connected and similar nodes are more likely to be proximal in new feature space, and vice versa (Nie et al., 2011). Thus, it is very appropriate for combining GCN with SC to extract the features of distant but similar nodes.

Following Nie et al. (2011), the object of performing SC is to generate the cluster assignment matrix $\mathbf{F}$. $\mathbf{F}$ can only be calculated by eigenvalue decomposition on the normalized similar matrix $\hat{\mathbf{S}} = \mathbf{D}^{-\mathbf{1/2}}\mathbf{S}\mathbf{D}^{-\mathbf{1/2}}$, which takes $O(n^2c)$ time complexity, where $\hat{\mathbf{S}} \in \mathbb{R}^{\mathbf{n} \times \mathbf{n}}$, n and c are the the numbers of nodes and clusters, respectively. For some large-scale graphs, the computational complexity is an unbearable burden.

In order to overcome this problem, we propose the efficient-spectral-clustering (ESC) method to efficiently perform SC. In the proposed ESC, instead of the S is calculated by equation 16, we employ inner product to construct affinity matrix $\mathbf{S} = \mathbf{X}\mathbf{X}^T$, and $\mathbf{S} \in \mathbb{R}^{n \times n}$. Thus, the normalized similar matrix $\hat{\mathbf{S}}$ in the ESC method can be represented as:

$$\begin{aligned} \hat{\mathbf{S}} &= \mathbf{D}_s^{-1/2}\mathbf{X}\mathbf{X}^T\mathbf{D}_s^{-1/2} \\ &= (\mathbf{D}_s^{-1/2}\mathbf{X})(\mathbf{D}_s^{-1/2}\mathbf{X})^T. \end{aligned} \tag{2}$$

Then, we define $\mathbf{P} = \mathbf{D}_s^{-1/2}\mathbf{X}$, thus $\hat{\mathbf{S}} = \mathbf{P}\mathbf{P}^{\mathbf{T}}$, where $\mathbf{P} \in \mathbb{R}^{n\times d}$. The singular value decomposition (SVD) of $\mathbf{P}$ can be represent as follows:

$$\mathbf{P} = \mathbf{U}\Sigma\mathbf{V}^T \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathbb{R}^{n\times d}$ and $\mathbf{V} \in \mathbb{R}^{d\times d}$ are left singular vector matrix, singular value matrix, and right singular vector matrix, respectively. In addition, $\mathbf{U}$ and $\mathbf{V}$ meet $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. Thus, $\hat{\mathbf{S}}$ can be represented as:

$$\begin{aligned}\hat{\mathbf{S}} &= \mathbf{U}\Sigma\mathbf{V}^T\left(\mathbf{U}\Sigma\mathbf{V}^T\right)^T \\ &= \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma^T\mathbf{U}^T \\ &= \mathbf{U}\Sigma\Sigma^T\mathbf{U}^T\end{aligned} \tag{4}$$

where $\Sigma\Sigma^T \in \mathbb{R}^{n\times n}$ is a diagonal matrix and the diagonal elements correspond to the eigenvalues of $\hat{\mathbf{S}}$. Meanwhile, the eigenvalues of $\hat{\mathbf{S}}$ are equal to the eigenvalues squared of $\mathbf{P}$. Therefore, the left singular vector matrix of $\mathbf{P}$ can be used to build the eigenvectors of $\hat{\mathbf{S}}$.

Hence, we can easily obtain the cluster assignment matrix $\mathbf{F}$ by performing a SVD on $\mathbf{P}$. Specifically, the $\mathbf{F}$ is constructed by the column vectors from $\mathbf{U}$, and the column vectors selected are corresponding to the c largest eigenvalues in Σ. The computational complexity of SVD performed on $\mathbf{P} \in \mathbb{R}^{n\times d}$ ($O = ndc$) is much lower, compared with directly performing eigenvalue decomposition on $\hat{\mathbf{S}} \in \mathbb{R}^{\mathbf{n}\times\mathbf{n}}$ ($O = nnc$).

Since the dimension d of nodes' original features are usually high in many graph-structured data, the efficiency of the ESC method still needs to be boosted. Thus, we propose the ESC-ANCH. Specifically, we first randomly select m nodes from the set of nodes $\mathcal{X}$ as the set of anchor nodes $\mathbf{X}'$, where $m << n$ and $m < d$. Then, we calculate the node-anchor similar matrix $\mathbf{R} \in \mathbb{R}^{n\times m}$, and the chosen similarity metric function is cosine. Here, $\mathbf{R}$ is employed as the new feature representations of nodes. Thus, in ESC-ANCH, $\mathbf{P}$ is redefined as $\mathbf{P} = \mathbf{D}_s^{-1/2}\mathbf{R}$, where $\mathbf{D}_s$ is the degree matrix of $\mathbf{S} = \mathbf{R}\mathbf{R}^T$ and $\mathbf{P} \in \mathbb{R}^{n\times m}$. Thus, the ESC-ANCH only takes $O(nmc)$ time complexity to perform the singular value decomposition of $\mathbf{P}$, which performs more efficiently than SC and ESC. In this paper, we default to use ESC-ANCH to obtain SC features.

2.4 Graph Convolutional Networks with Structure Learning

We first utilize the original features $\mathbf{X}$ and the SC features $\mathbf{F}$ to construct the EF. The first layer of the proposed GCN-SL is represented as:

$$\mathbf{H} = \text{ReLU}\left(\frac{\mathbf{X}\mathbf{W}_X^0 + \mathbf{F}\mathbf{W}_F^0}{2}\right) \tag{5}$$

where $\mathbf{W}_X^0 \in \mathbb{R}^{d\times q}$ and $\mathbf{W}_F^0 \in \mathbb{R}^{c\times q}$ are trainable weight matrix of the first layer. Meanwhile, we can also use the concatenation to combine $\mathbf{X}$ and $\mathbf{F}$.

$$\mathbf{H} = \text{ReLU}\left(\mathbf{X}\mathbf{W}_X^0 \| \mathbf{F}\mathbf{W}_F^0\right) \tag{6}$$

where $\|$ represents the concatenation of features. Here, we call the EF as EF_{av} if average methd is adopted, and call EF as EF_{cc} if concatenation method is adopted. After the first layer is constructed, we use $\mathbf{A}_*$ generated from subsection 2.2 to aggregate features of nodes to obtain the intermediate representations:

$$\mathbf{H}_{A_*}^{(1)} = \hat{\mathbf{A}}_*\mathbf{H} \tag{7}$$

where $\hat{\mathbf{A}}_*$ is the row normalized $\mathbf{A}_*$. Similarly, we use the original adjacency matrix $\mathbf{A}$ to obtain the intermediate representations $\mathbf{H}_A^{(k)}$ of nodes.

$$\mathbf{H}_A^{(k)} = \hat{\mathbf{A}}\mathbf{H}_A^{(k-1)} \tag{8}$$

where $k = 1, 2, \ldots, K$, K represents the times of feature aggregation, $\mathbf{H}_A^{(0)} = \mathbf{H}$, and $\hat{\mathbf{A}}$ is the normalized $\mathbf{A}$ with self-loop. After K rounds of feature aggregation, we combine several most key intermediate representations as a new node embedding, the formula can be written as:

$$\mathbf{H}^{cb} = \text{COMBINE}(\mathbf{H}, \mathbf{H}_A^{(K-1)}, \mathbf{H}_A^{(K)}, \mathbf{H}_{A_*}^{(1)}) \tag{9}$$

Table 1: Summary of the datasets utilized in our experiments.

Dataset	Cora	Citeseer	Pubmed	Squirrel	Chameleon	Cornell	Texas	Wisconsin
Hom.ratio h	0.81	0.74	0.8	0.22	0.23	0.3	0.11	0.21
# Nodes	2708	3327	19717	5201	2277	183	183	251
# Edges	5429	4732	44338	198493	31421	295	309	499
# Features	1433	3703	500	2089	2325	1703	1703	1703
# Classes	7	6	3	5	5	5	5	5

where COMBINE is set as concatenation so as to make full use of these intermediate representations. For assortative graphs, $\mathbf{H}_A^{(K-1)}$ and $\mathbf{H}_A^{(K)}$ are sufficient for representing embeddings of nodes. This can be proved by GCN (Kipf & Welling, 2017) and GAT (Velikovi et al., 2017). In addition, $\mathbf{H}_{A_*}^{(1)}$ can be treated as the supplement to $\mathbf{H}_A^{(K-1)}$ and $\mathbf{H}_A^{(K)}$. For disassortative graphs, $\mathbf{H}$ and $\mathbf{H}_{A_*}^{(1)}$ can also perform well on learning feature representation. Afterwards, we generate a learnable weight vector $\mathbf{w}$ that has the same dimension as $\mathbf{H}^{cb}$. Then we take the Hadamard product between $\mathbf{w}$ and $\mathbf{H}^{cb}$ to obtain the final feature representations of nodes,

$$\mathbf{H}^{final} = \text{ReLU}(\mathbf{w} \odot \mathbf{H}^{cb}). \tag{10}$$

The purpose of equation 10 is highlight the important dimensions of $\mathbf{H}^{cb}$. Then, nodes are classified by the following way:

$$\mathbf{Z} = softmax\left(\mathbf{H}^{final}\mathbf{W}^1\right) \tag{11}$$

where $\mathbf{W}^1$ is trainable weighted matrix of the last layer. We then calculate the cross-entropy error over all labeled nodes:

$$\mathcal{L}_{\text{pred}} = -\sum_{i \in \mathcal{Y}_{lab}} \sum_{j=1}^{C} Y_{ij} \ln Z_{ij} \tag{12}$$

where $\mathcal{Y}_{lab}$ is the set of node indices that have labels.

In this paper, the proposed GCN-SL with EF_{cc} is called as GCN-SL_{cc}, and similarly, the GCN-SL with EF_{av} is named as GCN-SL_{av}. A pseudocode of the proposed GCN-SL is given in Algorithm 1.

3 Experimental Results

In the experimental section, to validate the merit of GCN-SL, we compare GCN-SL with some state-of-the-art GNNs on transductive node classification tasks. Meanwhile, the experiments are performed on several benchmark datasets with different homophily. Before presenting the detailed evaluation, we first describe the graph data sets considered in this paper and briefly discuss the employed baseline techniques.

3.1 Datasets

In simulations, we adopt eight open graph datasets to validate the proposed GCN-SL, including three citation networks, three WebKB networks, and two Wikipedia networks.

The citation networks are standard citation network benchmark datasets, involving Cora, Citeseer, and Pubmed datasets (Sen et al., 2008; Namata et al., 2012; Yang et al., 2019). In these datasets, nodes correspond to papers and edges correspond to citations. Node features represent the bag-of-words representation of the paper, and the label of each node is the academic topics of the paper.

The sub-networks of the WebKB networks cover Cornell, Texas, and Wisconsin datasets. They are collected from various universities' computer science departments (Pei et al., 2020). In these datasets, nodes correspond to webpages, and edges represent hyperlinks between webpages. Node features denote the bag-of-words representation of webpages. These webpages can be divided into 5 classes.

Wikipedia networks are page-page networks about specific topics in Wikipedia, e.g., Chameleon and Squirrel data. Nodes correspond to pages, and edges correspond to the mutual links between pages. Node features represent some informative nouns in Wikipedia pages. These nodes are classified into four categories according to the amount of their average traffic.

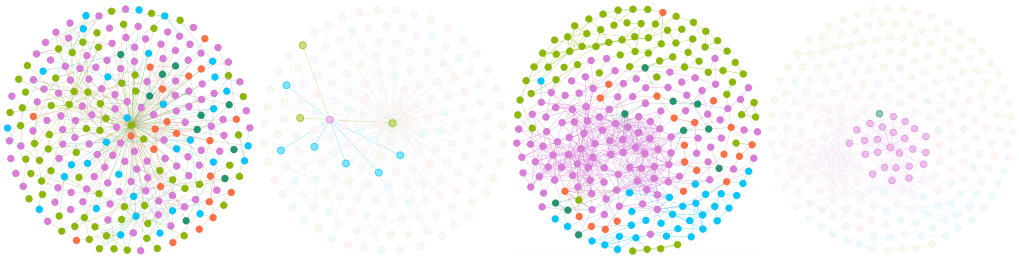
In all datasets, we randomly split nodes per class into 48%, 32%, and 20% for training, validation, and testing. The experimental results are the mean and standard deviation of ten runs. Testing is performed when validation losses achieve the minimum on each run. An overview summary of the characteristics of all datasets is shown in Table 1.

The level of homophily of graphs is one of the most important characteristics of graphs, so it is significant for us to analyze and employ graphs. For describing the homophily level of a graph, we utilize the edge homophily ratio $h = \frac{|\{(u,v):(u,v)\in\mathcal{E}\wedge y_u=y_v\}|}{|\mathcal{E}|}$, where $\mathcal{E}$ donate the set of edges, y_v and y_u represent the label of node v and u, respectively . The h is the fraction of edges in a graph which linked nodes that have the same class label (i.e., intra-class edges). This definition is proposed in Zhu et al. (2020). Obviously, graphs have strong homophily when h is high ($h \to 1$), and graphs have weak homophily when h is low ($h \to 0$). The h of each graph is listed in Table 1. From the homophily ratios of all adopted graphs, we can observe that all the citation networks are graphs under high homophily (i.e., assortative graphs), and all the WebKB networks and Wikipedia networks are graphs under low homophily (i.e., disassortative graphs).

3.2 Experimental Setup

For comparison, we use six state-of-the-art node classification algorithms including MLP (Hornik, 1991), GCN (Kipf & Welling, 2017), GAT (Velikovi et al., 2017), MixHop (AbuElHaija et al., 2019), Geom-GCN (Pei et al., 2020), and H_2GCN (Zhu et al., 2020). For H_2GCN, we consider two variants: H_2GCN-1 and H_2GCN-2 according to the furthest distance of neighbor nodes. In the above six models, all hyper-parameters are set according to Zhu et al. (2020). For the proposed GCN-SL, we perform a hyper-parameter search on the validation set. We utilize RELU as the activation function. All models are trained to minimize cross-entropy on the training nodes. Adam optimizer is employed for all models (Kingma & Ba, 2015).

3.3 Effectiveness Verification for Re-connected Adjacency Matrix



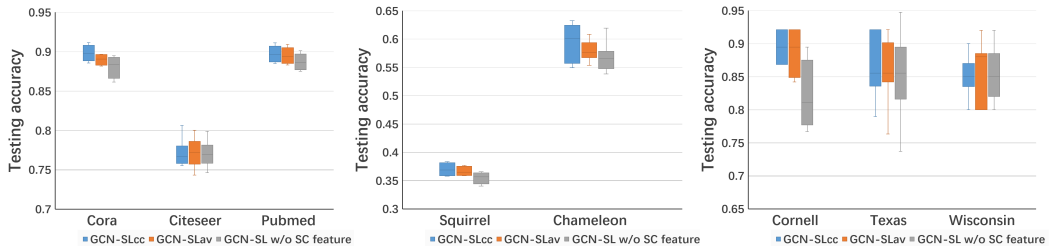
(a) Original graph (b) Neighbors in original graph (c) Re-connected graph (d) Neighbors in re-connected graph

Figure 2: Visualization of the graph structures. The graph-structured data is a real-world graph, namely the Wisconsin network, where color indicates the labels of nodes. The original graph is shown in (a) and the re-connected graph learned by GCN-SL$_{cc}$ is shown in (c). We fade the remaining nodes and edges to emphasize the neighbors of selected node in (b) and (d).

Table 2: Node classification accuracies (%) of GCN-SL$_{cc}$ and GCN-SL$_{av}$.

Dataset	$\mathbf{A}$	$\mathbf{A}_*$	Cora	Citeseer	Pubmed	Squirrel	Chamele.	Cornell	Texas	Wiscons.
GCN-SL$_{cc}$	–	–	74.7±1.9	72.1±1.8	87.1±1.4	29.4±2.9	49.4±2.9	82.6±5.5	82.3±3.9	86.5±2.1
	✓	–	**89.4±0.4**	**77.6±1.5**	88.5±1.1	37.1±3.2	57.5±2.8	76.6±8.6	81.2±6.2	81.8±5.3
	✓	✓	88.8±1.3	77.3±1.5	89.7±1.2	**43.5±3.1**	**61.4±3.2**	**86.3±5.1**	**87.6±6.2**	**88.6±4.8**
GCN-SL$_{av}$	–	–	75.2±2.7	71.7±2.7	86.9±0.6	30.2±1.4	48.7±3.1	82.4±5.9	82.1±4.8	86.2±2.2
	✓	–	88.7±0.6	77±1.5	**89.9±0.8**	35.6±3.0	55.9±3.7	76.8±5.9	81.9±3.2	84±6.0
	✓	✓	89.1±1.1	77±2.1	89.5±0.8	43.4±2.2	60.4±1.2	84.2±6.3	87.4±5.9	87.1±4.2

To examine the graph structure changes brought by GCN-SL intuitively, we visualize the original graph and the re-connected graph of the Wisconsin network by using Gephi tool in Figure 2. Meanwhile, we elect one specific node, and highlight the changes of its neighborhood in Figure 2(b) and



(a) (b) (c)

Figure 3: Ablation study about SC features on the proposed GCN-SL, w/o denotes without.

2(d). As shown in Figure 2, the original graph is relatively chaotic, and there exist lots of Inter-class connections. After applying our proposed graph structure learning method, the structure of the re-connected graph is much crisp, and the fraction of intra-class edges can reach 90%. This was a huge step forward. Thus the graph structure learning module can significantly improve the graph structure.

Afterwards, we explore the impact of the $\mathbf{A}_*$ on the accuracy of the proposed GCN-SL by using the ablation study. Table 2 gives the accuracies of GCN-SL in all adopted graphs, and the best results are bolded. We can see that the original adjacency matrix $\mathbf{A}$ is very important for GCN-SL$_{av}$ and GCN-SL$_{cc}$ in citation networks. However, the impact of $\mathbf{A}$ is very limited and even bad in the WebKB networks. This is because citation networks are assortative graphs, while WebKB networks are disassortative graphs. It is difficult for GCN-SL to aggregate useful information by only using $\mathbf{A}$ in graphs with low homophily. By contrast, the introduction of $\mathbf{A}_*$ is greatly helpful for the performance of GCN-SL in WebKB networks, and it does not hurt the performance of GCN-SL in citation networks. Meanwhile, the adoption of $\mathbf{A}_*$ is also helpful for GCN-SL in Wikipedia networks. This is owing to the re-connected graphs are constructed by similarity learning and optimized as the model is optimized. Thus, the re-connected graphs are more reliable than the original graph in disassortative benchmarks.

3.4 Effect of SC Features on Accuracy

In this experiment, we explore the impact of SC features on the node classification accuracy of GCN-SL. The SC features are extracted by the proposed ESC-ANCH method. In this work, we still use the ablation study. The classification accuracies of GCN-SL$_{cc}$, GCN-SL$_{av}$, and GCN-SL without SC features are shown in Figure 3, where all graph datasets are adopted. As can be seen, GCN-SL$_{cc}$ and GCN-SL$_{av}$ get better performance than GCN-SL without SC features. This owning to the SC features not only reflect the ego-embedding of nodes but also the features of similar nodes.

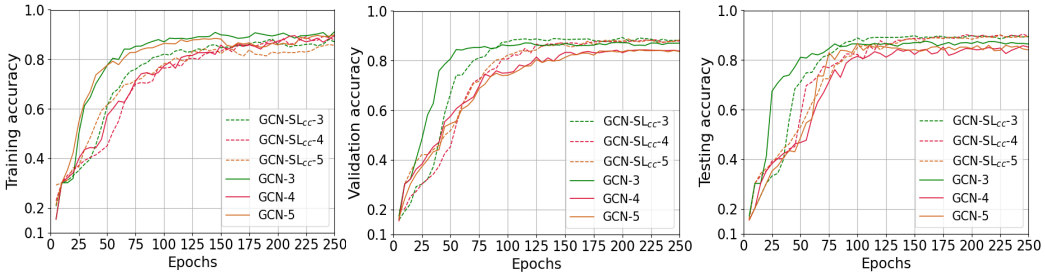
As a further insight, we focus on the running times of the original SC, the proposed ESC and ESC-ANCH in all adopted graph datasets. Table 3 records the results of the running times, and the proposed methods and the best results are bolded. From Table 3, we can see that ESC, and ESC-ANCH are much more efficient than SC. Meanwhile, ESC-ANCH is faster than ESC due to the introduction of anchor nodes. Specifically, ESC-ANCH only takes 1.2 s in Cora dataset, which is 47 times faster than original SC method. In Squirrel networks, ESC-ANCH takes 1.1 s, which is 90 times faster than original SC method. ESC-ANCH takes about half as long as original SC in WebKB networks. Having said all of the above, ESC-ANCH is a very efficient SC method.

Table 3: Running Time (Seconds) of SC, ESC and ESC-ANCH, on all graph datasets (OM Error).

Method	Cora	Citeseer	Pubmed	Squirrel	Chameleon	Cornell	Texas	Wisconsin
SC	58.1	63.2	OM	98.5	29.3	1.7	1.7	1.8
ESC	1.7	4.6	2.0	2.5	2.2	1.8	1.8	1.9
ESC-ANCH	**1.2**	**0.98**	**1.4**	**1.1**	**0.96**	**0.96**	**0.83**	**0.89**

3.5 Comparison Among Different GNNs

In Figure 4, we implement three multi-layer GCNs and three GCN-SL$_{cc}$-K on Cora, where K is the number of rounds in the aggregation stage. In this experiment, K is set as 3, 4, and 5, respectively.



(a) (b) (c)

Figure 4: Performance of Multi-layer GCNs and the proposed GCN-SL$_{cc}$ on Cora. GCN-SL$_{cc}$-3, 4, 5 represent the GCN-SL$_{cc}$ with different rounds of aggregation K respectively (i.e., K equal to 3, 4, and 5).

Table 4: Classification accuracies (%) of the MLP, GCN, GAT, Geom-GCN*, MixHop, H_2GCN and the proposed GCN-SL.

Method	Cora	Citeseer	Pubmed	Squirrel	Chameleon	Cornell	Texas	Wisconsin
MLP	74.8±2.2	72.4±2.2	86.7±0.4	29.7±1.8	46.4±2.5	81.1±6.4	81.9±4.8	85.3±3.6
GCN	87.3±1.3	76.7±1.6	87.4±0.7	36.9±1.3	59.0±4.7	57.0±4.7	59.5±5.2	59.8±7
GAT	82.7±1.8	75.5±1.7	84.7±0.4	30.6±2.1	54.7±2.0	58.9±3.3	58.4±4.5	55.3±8.7
Geom-GCN*	85.3	**78.0**	**90.1**	38.1	60.9	60.8	67.6	64.1
MixHop	87.6±0.9	76.3±1.3	85.3±0.6	**43.8±1.5**	60.5±2.5	73.5±6.3	77.8±7.7	75.9±4.9
H_2GCN-1	86.9±1.4	77.1±1.6	89.4±0.3	36.4±1.9	57.1±1.6	82.2±4.8	84.9±6.8	86.7±4.7
H_2GCN-2	87.8±1.4	76.9±1.7	89.6±0.3	37.9±2.0	59.4±2.0	82.2±6.0	82.2±5.3	86.3±4.2
GCN-SL$_{cc}$	**89.4±1.3**	77.6±1.5	89.7±1.2	43.5±3.1	**61.4±3.2**	**86.3±5.1**	**87.6±6.2**	**88.6±4.8**
GCN-SL$_{av}$	89.1±1.1	77±2.1	89.9±0.8	43.4±2.2	60.4±1.2	84.2±6.3	87.4±5.9	87.1±4.2

Figure 4 implies that the performance of multi-layer GCNs gets worse as the depth increases. The main reason can be owed to over-fitting and over-smoothing. Meanwhile, GCN-SL$_{cc}$ performs well and do not suffer from over-fitting and over-smoothing as the depth increases. Obviously, the GCN-SL$_{cc}$-5 aggregates the same features from original neighbors as 5-layer GCN. This shows that the reason for the performance degradation of multi-layer GCN is not over-smoothing but over-fitting as the depth increases. In addition, this also shows that GCN-SL can be immune to over-fitting no matter how many node features are aggregated.

Table 4 gives classification results per benchmark, the proposed GNNs and best results are bolded. From the table 4, we can observe that all GNN models can achieve a satisfactory result in citation networks. However, some GNN models perform even worse than MLP in WebKB networks and Wikipedia networks, such as GCN, GAT, Geom-GCN, and MixHop. The main reason for this phenomenon is that these GNN models aggregate useless information from wrong neighborhoods, and these GNN models do not separate ego-embedding and useless neighbor-embedding. By contrast, H_2GCN and GCN-SL can achieve good classification results no matter which graph is adopted. Meanwhile, the results of GCN-SL$_{cc}$ and GCN-SL$_{av}$ are relatively better than H_2GCN. This is mainly due to the introduction of the re-connected graph in GCN-SL$_{cc}$ and GCN-SL$_{av}$. In addition, the introduction of SC features can improve the ego-embedding of nodes by clustering similar nodes together.

4 Conclusion

In this paper, we propose an effective GNN approach, referred to as GCN-SL. Compared with other GNNs, our research includes three main contributions: 1) From the aspect of node features, SC is integrated into GNNs for capturing long-range dependencies on graphs, and we propose an ESC-ANCH algorithm for dealing with large-scale graph-structured data; 2) From the aspect of edges, the proposed GCN-SL can learn a high quality re-connected adjacency matrix by using similarity learning and special data preprocessing technique; 3) GCN-SL is appropriate for all levels of homophily by combining multiple key node intermediate representations. Experimental results have illustrated the proposed GCN-SL is superior to other existing counterparts.

REFERENCES

Sami AbuElHaija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, pp. 21–29. PMLR, 2019.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Bo Chen, Le Sun, and Xianpei Han. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018a.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *6th International Conference on Learning Representations, ICLR*, 2018b.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems*, 2020.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. Advances in Neural Information Processing Systems, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. pp. 3844–3852. Advances in Neural Information Processing Systems, 2016.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1263–1272. JMLR.org, 2017.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. pp. 52–74. Advances in Neural Information Processing Systems, 2017a.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. pp. 1024–1034. Advances in Neural Information Processing Systems, 2017b.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data, 2015. URL `http://arxiv.org/abs/1506.05163`.

Mikael Henaff, Joan Bruna, and Yann LeCun. Non-local graph neural networks, 2020. URL `https://arxiv.org/abs/2005.14612`.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4 (2):251–257, 1991.

Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. pp. 4563–4572. Advances in Neural Information Processing Systems, 2018.

Daniel D. Johnson. Learning graphical state transitions. pp. 1–19. International Conference on Learning Representations, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. pp. 1–14. International Conference on Learning Representations, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Gnnemann. Predict then propagate: Graph neural networks meet personalized pagerank. International Conference on Learning Representations, 2019.

Chengcheng Li, Zi Wang, and Hairong Qi. An efficient pipeline for pruning convolutional neural networks. In *19th IEEE International Conference on Machine Learning and Applications, ICMLA*, pp. 907–912. IEEE, 2020.

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. pp. 3546–3553. AAAI Press, 2018.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, 2012.

Feiping Nie, Zinan Zeng, Ivor W. Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Trans. Neural Networks*, 22(11):1796–1808, 2011.

Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th International Conference on World Wide Web, WWW*, pp. 201–210. ACM, 2007.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: geometric graph convolutional networks. International Conference on Learning Representations, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR*, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

Uday Shankar Shanthamallu, Jayaraman J. Thiagarajan, Huan Song, and Andreas Spanias. Gramme: Semisupervised learning using multilayered graph attention models. *IEEE Trans. Neural Networks Learn. Syst.*, 31(10):3977–3988, 2020.

Petar Velikovi, G. Cucurull, A. Casanova, A. Romero, P Li, and Y. Bengio. Graph attention networks. pp. 1–12. International Conference on Learning Representations, 2017.

Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. Graph structure estimation neural networks. pp. 342–353. ACM / IW3C2, 2021a.

Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 14913–14922. Computer Vision Foundation / IEEE, 2021b. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Convolutional_Neural_Network_Pruning_With_Structural_Redundancy_Reduction_CVPR_2021_paper.html.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32 (1):4–24, 2021.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462. PMLR, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks. pp. 1–17. International Conference on Learning Representations, 2019.

Liang Yang, Zhiyang Chen, Junhua Gu, and Yuanfang Guo. Dual self-paced graph convolutional network: Towards reducing attribute distortions induced by topology. In Sarit Kraus (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 4062–4069. ijcai.org, 2019.

Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *9th IEEE International Conference on Computer Vision ICCV*, pp. 313–319. IEEE Computer Society, 2003.

Xiang Zhang and Marinka Zitnik. Gnnguard: defending graph neural networks against adversarial attacks. Advances in Neural Information Processing Systems, 2020.

Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. Heterogeneous graph structure learning for graph neural networks. pp. 4697–4705. AAAI Press, 2021.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. pp. 1399–1407. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, 2019.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: current limitations and effective designs. Advances in Neural Information Processing Systems, 2020.

A Appendix

A.1 The Define of Notations

Unless particularly specified, the notations employed in this paper are illustrated in Table 5.

Table 5: Commonly Used Notations

Notations	Descriptions
$\odot$	Hadamard product.
$\|\|$	Concatenation.
$\mathcal{G}$	A graph.
$\mathcal{V}$	The set of edges in graph.
v_i	A node $v_i \in \mathcal{V}$.
$\mathcal{N}_i$	The neighbors of a node v_i.
$\mathbf{A}$	The graph adjacency matrix.
$\mathbf{D}$	The degree matrix of $\mathbf{A}$, $D_{ii} = \sum_j A_{i,j}$.
$\hat{\mathbf{A}}$	The normalized adjacency matrix with self-loop.
$\mathbf{A}_*$	The re-connected adjacency matrix.
$\hat{\mathbf{A}}_*$	The row normalized re-connected adjacency matrix.
$\mathbf{S}$	The similar matrix of a graph.
$\mathbf{D}_s$	The degree matrix of $\mathbf{S}$, $D_{Sii} = \sum_j S_{i,j}$.
$\mathbf{L}_s$	The Laplacian matrix of $\mathbf{S}$, $\mathbf{L}_S = \mathbf{D}_S - \mathbf{S}$.
$\hat{\mathbf{S}}$	The normalized similar matrix.
n	The number of nodes, $n = \|\mathcal{V}\|$.
m	The number of anchor nodes.
d	The dimension of a node feature vector.
c	The dimension of the SC feature of a node.
C	The number of node labels.
$\mathbf{X} \in \mathbb{R}^{n \times d}$	The node feature matrix.
$\mathbf{x}_i \in \mathbb{R}^d$	The feature vector of the node v_i.
$\mathbf{R} \in \mathbb{R}^{n \times m}$	The node-anchor similar matrix.
$\mathbf{P} \in \mathbb{R}^{n \times m}$	The processed node-anchor similar matrix.
$\mathbf{U}, \Sigma, \mathbf{V}$	left singular vector matrix, singular value matrix, and right singular vector matrix of $\mathbf{P}$, respectively .
$\mathbf{F} \in \mathbb{R}^{n \times c}$	The node SC feature matrix.
$\mathbf{f}_i \in \mathbb{R}^c$	The SC feature vector of the node v_i.
$\mathbf{H}$	The enhanced node hidden feature matrix.
$\mathbf{H}^{cb}$	The combined node hidden feature matrix.
$\mathbf{H}^{final}$	The final node hidden feature matrix.
$\mathbf{Z} \in \mathbb{R}^{n \times r}$	The probability distribution of nodes.
$\mathbf{Q}, \mathbf{W}_X^0, \mathbf{W}_F^0, \mathbf{w}, \mathbf{W}^1$	Learnable model parameters

A.2 RELATED WORK

A.2.1 GCN

Following the work in Kipf & Welling (2017), GCN updates node features via adopting an isotropic averaging operation over the feature representations of one-hop neighbors. Let $\mathbf{h}_j^\ell$ be the feature representation of node v_j in the l-th GCN layer, and we have

$$\mathbf{h}_i^{\ell+1} = \mathrm{ReLU}\left(\sum_{j\in\mathcal{N}_i} \frac{1}{\sqrt{\mathrm{deg}_i \mathrm{deg}_j}} \mathbf{h}_j^\ell \mathbf{W}^\ell\right) \tag{13}$$

where $\mathcal{N}_i$ represents the set of one-hop neighbors of nodes v_i, $\mathbf{W}^\ell$ is a learnable weight matrix, and ReLU is employed as the activation function. Note that deg_i and deg_j are the in-degrees of nodes v_i and v_j, respectively. Furthermore, the forward model of a 2-layer GCN can be represented as:

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = softmax\left(\hat{\mathbf{A}}\mathrm{ReLU}\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^0\right)\mathbf{W}^1\right), \tag{14}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the feature matrix of nodes and is also the input of the first GCN layer, and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix. n and d represent the number and feature dimension of nodes, respectively. The adjacency matrix with self-loop is $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is an identity matrix. Here, $\tilde{\mathbf{A}}$ can be normalized by $\hat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}$, and the normalized adjacency matrix $\hat{\mathbf{A}}$ is employed for aggregating the feature representation of neighbors, where $\widetilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. Each element $\hat{A}_{i,j}$ is defined as:

$$\hat{A}_{i,j} = \begin{cases} \frac{1}{\sqrt{\mathrm{deg}_i \mathrm{deg}_j}} & \text{nodes } i,j \text{ are one} - \text{hop neighbors} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

A.2.2 SPECTRAL CLUSTERING

Spectral clustering (SC) is an advanced algorithm evolved from graph theory, has attracted much attention (Nie et al., 2011). Compared with most traditional clustering methods, the implementation of SC is much simpler (Nie et al., 2011). It is remarkable that, for SC, a weighted graph is utilized to partition the dataset. Assume that $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ represents a dataset. The task of clustering is to segment $\mathcal{X}$ into c clusters. The cluster assignment matrix is denoted as $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n]^T \in \mathbb{R}^{n\times c}$, where $\mathbf{f}_i \in \mathbb{R}^{c\times 1}(1 \le i \le n)$ is the cluster assignment vector for the pattern $\mathbf{x}_i$. From another perspective, $\mathbf{f}_i$ can be considered as the feature representation of $\mathbf{x}_i$ in the c-dimensional feature space. SC contains different types, while our work focuses on the SC with k-way normalized cut due to considering the overall performance of the algorithm, where the related concepts are explained in Yu & Shi (2003).

Let $\mathcal{G} = \{\mathcal{X}, \mathbf{S}\}$ be an undirected weighted graph, where $\mathcal{X}$ denotes the set of nodes, and $\mathbf{S} \in \mathbb{R}^{n\times n}$ denotes affinity matrix, and n is the number of nodes in $\mathcal{G}$. Note that $\mathbf{S}$ is a symmetric matrix, and each element $S_{i,j}$ represents the affinity of a pair of nodes in $\mathcal{G}$. The most common way to built $\mathbf{S}$ is full connection method. Following the description in Nie et al. (2011), $S_{i,j}$ can be represented as:

$$S_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \quad i,j = 1,2,\ldots,n \tag{16}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the features of node v_i and node v_j in $\mathcal{X}$, respectively, and σ can control the degree of similarity between nodes. Then, the Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L}_s = \mathbf{D}_s - \mathbf{S}$, where degree matrix $\mathbf{D}_s$ is a diagonal matrix, and the diagonal element is denoted as $D_{sii} = \sum_j S_{ij}, \forall i$. The objective function of SC with normalized cut is

$$\min_{\mathbf{F}^T\mathbf{F}=\mathbf{I}} tr\left(\mathbf{F}^T\mathbf{D}_s^{-1/2}\mathbf{L}_s\mathbf{D}_s^{-1/2}\mathbf{F}\right) \tag{17}$$

where $\mathbf{F} \in \mathbb{R}^{n\times c}$ is the clustering indicator matrix, and the objective function can be rewritten as:

$$\max_{\mathbf{F}^T\mathbf{F}=\mathbf{I}} tr\left(\mathbf{F}^T\mathbf{D}_s^{-1/2}\mathbf{S}\mathbf{D}_s^{-1/2}\mathbf{F}\right). \tag{18}$$

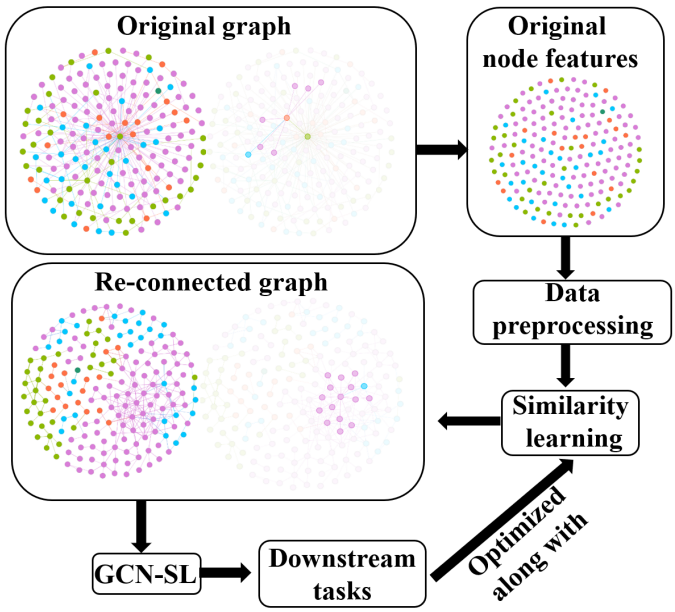


Figure 5: Flowchart of the graph structure learning. The graph-structured data is a real-world graph, namely the Texas network, where color indicates the labels of nodes. Through data preprocessing and similarity learning, we obtain a re-connected graph. The re-connected graph is optimized together with the parameters of GCN-SL. Finally, we can obtain a re-connected graph with high homophily.

The optimal solution $\mathbf{F}$ of objective function is constructed by the eigenvectors corresponding to the c largest eigenvalues of $\hat{\mathbf{S}} = \mathbf{D_s^{-1/2}SD_s^{-1/2}}$. In general, $\mathbf{F}$ can be not only considered as the result of clustering of nodes, but also regarded as the new feature matrix of nodes, in which node has c feature elements (Nie et al., 2011).

A.3 The proposed GSN-SL

A.3.1 The Construction of Re-connected Adjacency Matrix

We describe the construction of re-connected adjacency matrix in Figure 5. Through structure learning, the proposed GCN-SL can learn a re-connected adjacency matrix. The new adjacency matrix is based on the similarities between nodes and can be optimized along with the parameters of GCN-SL. which makes the GCN-SL can capture information from more reliable nodes.

A.3.2 Pseudocode of GCN-SL

A pseudocode of the proposed GCN-SL is given in Algorithm 1. In addition to the original adjacency matrix and node features, the re-connected adjacency matrix and SC feature are also employed to generate the embedding of node. Consequently, GCN-SL can capture long-range dependencies better than GCN.

A.4 Experiment

A.4.1 Device Information

We implement the proposed GCN-SL with deep learning library PyTorch. All experiments are conducted on a Linux server with a GPU (NVIDIA GeForce RTX 2080 Ti). The Python and PyTorch versions are 3.8.10 and 1.9.0, respectively.

Algorithm 1 GCN-SL

Input: $\mathbf{X}$: feature matrix of nodes, $\mathbf{Y}_{lab}$: label matrix of nodes with labels, $\mathbf{A}$: original adjacency matrix, $\mathbf{Q}, \mathbf{W}_X^0, \mathbf{W}_F^0, \mathbf{W}^1$: trainable weighted matrixs, $\mathbf{w}$: trainable weighted vector.
Output: $\mathbf{Z}$: probability distribution of nodes.
1. $\mathbf{F} \leftarrow \mathbf{X}$ in 2.3.
2. $\mathbf{X} \leftarrow \{\mathbf{X}, Data\ preprocessing\}$.
3. $\mathbf{A}_* \leftarrow \{\mathbf{X}, \mathbf{Q}\}$ in 2.2.
4. $\mathbf{H} \leftarrow \left\{\mathbf{X}, \mathbf{F}, \mathbf{W}_X^0, \mathbf{W}_F^0\right\}$ in the way of average in Eq. 5, or in the way of concatenation in Eq. 6.
5. $\mathbf{H}_{A_*}^{(1)} \leftarrow \{\mathbf{H}, \mathbf{A}_*\}$ in Eq. 7.
repeat (initialize k to 1)
6. Update $\mathbf{H}_A^{(k)} \leftarrow \left\{\mathbf{H}_A^{(k-1)}, \mathbf{A}\right\}$, $\mathbf{H}_A^{(0)} = \mathbf{H}$ in Eq. 8, $k = k + 1$.
until $k = K$
7. $\mathbf{H}^{cb} \leftarrow \left\{\mathbf{H}, \mathbf{H}_A^{(K-1)}, \mathbf{H}_A^{(K)}, \mathbf{H}_{A_*}^{(1)}\right\}$ in Eq. 9.
8. $\mathbf{H}^{final} \leftarrow \left\{\mathbf{H}^{cb}, \mathbf{w}\right\}$ in Eq. 10
9. $\mathbf{Z} \leftarrow \left\{\mathbf{H}^{final}, \mathbf{W}^1\right\}$ in Eq. 11.
10. $\mathcal{L}_{pred} \leftarrow$ LOSS $(\mathbf{Z}_{lab}, \mathbf{Y}_{lab})$ in Eq. 12.
11. Back-propagate $\mathcal{L}_{pred}$ to update model weights.

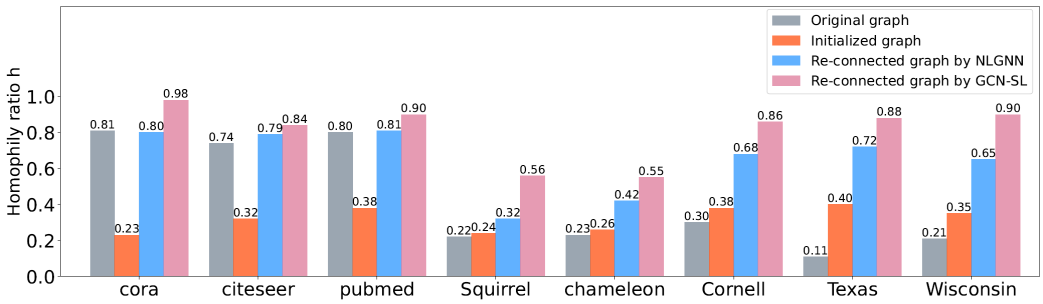


Figure 6: Comparisons of homophily ratio h on networks, including the original graphs, initialized graphs and the re-connected graphs.

A.4.2 The Homophily Ratio h of Re-connected Adjacency Matrixes

In this experiment, we use the homophily ratios h to examine the correctness of the re-connected adjacency matrix $\mathbf{A}_*$. High h means high correctness, and low h means low correctness. Figure 6 shows the h of the original graph, initialized graph, and the re-connected graphs in all graphs. The initialized graph is generated by GCN-SL before training, and the re-connected graphs are learned by NLGNN and GCN-SL, respectively. We can observe that the h of the re-connected graphs is much higher than the original graph and initialized graphs, especially for WebKB networks and Wikipedia networks. This is mainly because the re-connected graphs are not only related to the similarity between nodes but also can be optimized along with the parameters of the GNN model. Meanwhile, we can also see that the h of the re-connected graphs learned by GCN-SL are always higher than the NLGNN. This is because the similarity metric function chosen by NLGNN is the inner product, which is not powerful enough to represent the similarities between nodes. Meanwhile, the special data preprocessing technique utilized in GCN-SL can relieve the over-fitting problem, and the technique is also greatly useful for structure learning. In addition, compared with NLGNN, the ability of representation learning of the proposed GCN-SL is more capable, and thus GCN-SL can offer more help to the learning of the re-connected graph.

A.4.3 The Description of Hyperparameters

The description per each hyper-parameter that need to be searched and the corresponding range of values tried are provided in Table 6. Table 7 summarizes the values of all hyper-parameters, and the

node classification accuracies of GCN-SL on different datasets. The random seed is set as 42 for all experiments.

Table 6: Hyper-parameter descriptions and range of values tried

Hyper-para.	Descriptions	Range of values tried
lr	Learning rate.	0.01, 0.02, ..., 0.05
wd	Weight decay.	5e-3, 5e-4, ..., 5e-6
d	Dropout rate.	0.1, 0.2, ..., 0.9
ϵ	Non-negative threshold in similarity learning.	0.8, 0.85, ..., 0.95
m	Number of anchor nodes.	100, 200, ..., 700
c	Dimension of SC features.	10, 15, ..., 80
q	Output dimension of learnable weighted matrix $\mathbf{W}_X^0$ and $\mathbf{W}_F^0$.	16, 32, 48
K	The number of rounds in the original neighborhood aggregation stage.	1, 2, ..., 5
p	Output dimension of learnable weighted matrix $\mathbf{Q}$.	16, 32, 48

Table 7: The hyper-parameters of best accuracy for GCN-SL on all datasets.

Dataset	Accuracy		Hyper-parameters
	GCN-SL$_{cc}$	GCN-SL$_{av}$	
Cora	89.4±1.3	89.1±1.1	lr:0.04, wd:5e-4, d:0.5, ϵ:0.9, m:700, c:75, q:32, K:4, p:16, $seed$ = 42
Citeseer	77.6±1.5	77.0±2.1	lr:0.02, wd:5e-4, d:0.5, ϵ:0.95, m:500, c:30, q:32, K:3, p:16, $seed$ = 42
Pubmed	89.7±1.2	89.9±0.8	lr:0.02, wd:5e-4, d:0.5, ϵ:0.95, m:500, c:25, q:32, K:2, p:16, $seed$ = 42
Squirrel	43.5±3.1	43.4±2.2	lr:0.03, wd:5e-4, d:0.6, ϵ:0.85, m:400, c:15, q:32, K:2, p:16, $seed$ = 42
Chameleon	61.4±3.2	60.4±1.2	lr:0.02, wd:5e-4, d:0.6, ϵ:0.95, m:200, c:15, q:32, K:3, p:16, $seed$ = 42
Cornell	86.3±5.1	84.2±6.3	lr:0.01, wd:5e-4, d:0.3, ϵ:0.85, m:100, c:15, q:48, K:1, p:16, $seed$ = 42
Texas	87.6±6.2	87.4±5.9	lr:0.01, wd:5e-4, d:0.3, ϵ:0.85, m:100, c:35, q:32, K:1, p:16, $seed$ = 42
Wisconsin	88.6±4.8	87.1±4.2	lr:0.01, wd:5e-4, d:0.3, ϵ:0.85, m:100, c:20, q:32, K:1, p:16, $seed$ = 42