Reverse Graph Learning for Graph Neural Network

Liang Peng, Rongyao Hu, Fei Kong, Jiangzhang Gan, Yujie Mo, Xiaoshuang Shi, and Xiaofeng Zhu, *Senior Member, IEEE*

***Abstract*— Graph neural networks (GNNs) conduct feature learning by taking into account the local structure preservation of the data to produce discriminative features, but need to address the following issues, i.e., 1) the initial graph containing faulty and missing edges often affect feature learning and 2) most GNN methods suffer from the issue of out-of-example since their training processes do not directly generate a prediction model to predict unseen data points. In this work, we propose a reverse GNN model to learn the graph from the intrinsic space of the original data points as well as to investigate a new out-of-sample extension method. As a result, the proposed method can output a high-quality graph to improve the quality of feature learning, while the new method of out-of-sample extension makes our reverse GNN method available for conducting supervised learning and semi-supervised learning. Experimental results on real-world datasets show that our method outputs competitive classification performance, compared to state-of-the-art methods, in terms of semi-supervised node classification, out-of-sample extension, random edge attack, link prediction, and image retrieval.**

***Index Terms*— Graph learning, graph neural network, out-of-sample extension, robust learning.**

I. Introduction

GRAPH neural networks (GNNs) have been widely applied in a variety of fields such as link prediction, knowledge graphs, and semantic role labeling because they can capture the local geometric structure between data points to extract local, compositional, and discriminative features from non-Euclidean data (e.g., graphs and manifolds) [1]–[3]. Different from convolutional neural networks (CNNs), which consider the original data as the input to work well with Euclidean data such as images and texts [4], the inputs of GNN methods include the original data and an initial graph containing the local structure (e.g., the similarity between the data point and its nearest neighbors [5]–[7]). GNN can thus deal effectively with complex data structures such as knowledge graphs and molecular simulations. For example, Schlichtkrull *et al.* [8] proposed relational graph convolutional networks (GCNs) to impute missing triplets for knowledge graphs and the GNN-based method in [9] showed remarkable performance in predicting various molecular properties.

As aforementioned, one of the critical differences between CNN and GNN is the graph input. In this regard, the underlying information in the graph is critical for GNN models, as the wrong (e.g., incorrect edges) and insufficient (e.g., missing edges) information will be passed to the network construction, misleading the model and limiting its effectiveness [10]–[12]. Previous GNN methods often ignore the quality of the initial graph. As a result, the low-quality graph cannot produce discriminative features [2]. Specifically, previous methods obtain the graph from the original data and keep it invariant during the feature learning process. The original data containing noise and outliers easily results in an incorrect relationship between two data points [13]. Moreover, the initial graph with incorrect information affects the feature learning process and the construction of the classification models [13]–[15].

A huge number of solutions have been proposed to address the above issue by simultaneously performing adaptive graph learning and feature learning in a unified framework [13], [16], [17], i.e., adaptive GNN for short. Specifically, adaptive graph learning can adaptively update the low-quality graph based on the learned features, which are also adjusted by the updated graph iteratively. As a result, the graph is adaptively adjusted to fit the process of feature learning, so that discriminative features can be easily generated [14]. However, it has been shown that such a solution (i.e., [16]) only changes and reduces the weights of the original edges in the graph, but is not available to find the missing relationship [18]. To address this issue, Jiang *et al.* [19] proposed to conduct the graph structure learning while taking into account both the initial graph and the updated graph before GNN model. Chen *et al.* [18] proposed to jointly conduct graph structure learning and feature learning at each layer of the GNN model. However, previous adaptive GNN methods conduct graph learning in the original data space so that it is difficult to find missing edges and accurately adjust edge weights.

Another challenge of previous GNN methods (e.g., GCNs [5]) is that they often suffer from the out-of-sample

Manuscript received July 27, 2021; revised February 2, 2022; accepted March 16, 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102200, in part by the National Natural Science Foundation of China under Grant 61876046, and in part by the Guangxi "Bagui" Teams for Innovation and Research. *(Liang Peng and Rongyao Hu contributed equally to this work.) (Corresponding authors: Xiaoshuang Shi; Xiaofeng Zhu.)*

Liang Peng, Fei Kong, Yujie Mo, and Xiaoshuang Shi are with the Center for Future Media, and the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: xsshi2013@gmail.com).

Rongyao Hu and Jiangzhang Gan are with the Center for Future Media, and the School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the School of Mathematical and Computational Sciences, Massey University at Auckland, Auckland 0745, New Zealand.

Xiaofeng Zhu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518000, China (e-mail: seanzhuxf@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3161030.

Digital Object Identifier 10.1109/TNNLS.2022.3161030

issue [20]. As a transductive learning method, many GNN methods do not generate a specific prediction model in the training process to predict unseen data points. The study of out-of-sample extension has attracted great interest in the field of machine learning and computer vision. For example, the QE method directly projects the unseen data point on the space spanned by itself and its nearest neighbors [20], [21]. And Hamilton *et al.* [22] proposed GraphSAGE, which fuses the features of the neighborhoods as the feature of the unseen data points. However, they do not take into account the quality of the initial graph. As a consequence, the effectiveness of the out-of-sample extension will be affected.

In this article, we propose a novel GNN method with reverse graph learning (rGNN for short), which includes two components to address the above issues:

1) *Reverse Graph Learning:* The representation of the original data usually contains noise and redundancy. Hence, the initial graph constructed in the original data is less reliable and with low quality. To address this issue, reverse graph learning is proposed in this article, where the graph is learned from the intrinsic space of the original data. Moreover, the issue of noise is alleviated. Furthermore, reverse graph learning is adaptively adjusted according to the update of feature learning. As a result, the quality of the graph input into the GNN model can be guaranteed;
2) *Out-of-Sample Extension:* We propose a new out-of-sample extension method by assigning the features of each unseen data point as the features of its nearest nodes. Specifically, our method obtains the initial graph from the intrinsic space of the original data from the reverse graph learning layer, which guarantees to capture real closest nodes.

Compared to previous GNN methods, the contributions of our proposed method are twofold.

1) Our method learns the graph from the intrinsic space of the original dataset, aiming at avoiding the issues, such as the influence of noise and redundancy. Moreover, reverse graph learning can be adaptively adjusted by the process of feature learning, which correctly adjusts the intrinsic space to improve the quality of the graph.
2) Our method is inductive by overcoming the issue of out-of-sample, i.e., generating an explicit mapping function in the training process to predict unseen data points.

As a result, the proposed method can achieve significant performance, compared to the comparison methods, in terms of the learning tasks, such as semi-supervised node classification, out-of-sample extension, random edge attack, link prediction, and image retrieval.

II. Related Works

In this section, we give a brief review of the research topics related to our proposed method, including GNNs, graph learning, and out-of-sample extension on graph datasets.

A. Graph Neural Networks

A large number of GNN methods (e.g., [23], [24]) have been proposed to facilitate deep learning on graph structured data, such as communication networks and citation networks. For example, Bruna *et al.* [25] proposed to generalize the convolution operation from Euclidean data to non-Euclidean data with the Fourier basis of a given graph, aiming at conducting the classification task. Defferrard *et al.* [26] utilized Chebyshev polynomials as the convolution filter to simplify spectral GNNs. GCNs propose an approximation of a localized spectral convolution filter to build node representations [5]. Simple graph convolution (SGC) reduces the graph convolution to a linear model but still achieves competitive performance [27]. Graph attention network (GAT) builds different attention scores on each neighborhood for information aggregation [16]. Diffusion convolutional neural networks (DCNNs) treats graph convolutions as a diffusion process and assigns a certain transition probability to the information transferred from one node to the adjacent node [28].

With the success of GNN methods, they have widely been applied to various kinds of research topics, such as self-supervised learning and graph generation. In the domain of self-supervised learning, GNN has been verified to obtain expressive representations and has recently been extended to graph domain [29], [30]. For example, Velickovic *et al.* [31] employed local and global contrastive learning to contrast the local node representations and the global graph representations. Mavromatis and Karypis [32] designs to maximize the agreement between the local node representations and their corresponding cluster centroids. In the domain of graph generation, GNNs are used to generate novel graphs and are potentially useful for molecule discovery [33], [34]. For example, De Cao and Kipf [35] proposed a generation model for molecular graphs to predict their adjacency matrices with chemical validity and efficiency. Guo *et al.* [36] proposed to utilize a graph generative model (i.e., graph auto-encoder) to model the pairwise distance.

B. Graph Learning

The key idea of graph learning is to simultaneously learn the underlying graph structure of data points based on graph Laplacian regularize (GLR) [37] and learning tasks [38], [39]. During the training process, the initial graph is iteratively refined via a graph learning model and then fed into the learning models [4], [40]. Previous graph learning methods can be categorized into two subtypes, i.e., traditional graph learning and deep graph learning [3].

In traditional graph learning, on the one hand, the graph measuring the similarity between two data points is learned based on the data distribution. For example, Zhu *et al.* [6] proposed to simultaneously learn the graph and conduct feature selection. On the other hand, the initial graph is updated by embedding the graph learning into the learning tasks. For example, Hu *et al.* [15] and Gan *et al.* [7] proposed to conduct multiple tasks in a unified framework for medical image analysis, such as graph learning, graph fusion, and feature selection.

In deep learning models, GAT conducts graph learning by dynamically updating the weights of edges through the attention mechanism [16]. However, GAT does not discover

new edges. Graph learning convolutional networks (GLCNs) optimizes the learning process by integrating graph learning and graph convolution into a unified network architecture [19]. Following GLCN, joint learning graph representation and node features (JLGCN) deploys Mahalanobis distance to measure similarity between nodes [14]. Deep iterative graph learning (DIAL) searches the underlying graph structure by an iterative method [18] and data augmentation GNN (GAUG) models edge weights by taking the inner product of the embedding of two nodes without additional parameters [41].

Recently, graph learning in deep learning methods are extended to Bayesian and the kernel approach. For example, Elinas *et al.* [42] regarded the initial graph as a sample from a parametric family of random graphs, aiming at conducting graph learning. Pu *et al.* [43] proposed a kernel function-based approach to improve graph-based regularization. More recently, Zhao *et al.* [44] proposes the method of heterogeneous graph structure learning (HGSL) by extending graph learning to the heterogeneous graph structure. Jin *et al.* [45] considers using the graph learning method to defend against different types of graph adversarial attacks. Hu *et al.* [46] employed graph learning for the 3-D point cloud task.

C. Out-of-Sample Issue

In the transductive learning framework, the training process does not output a prediction model for predicting unseen data points. This leads to the issue of out-of-sample. Many GNN methods belong to transductive learning, and thus limiting their applications. Recently, few works have focused on exploring the issue of out-of-sample. For example, GraphSAGE exploits to generate embedding of the unseen data points by sampling and aggregating the features from its local neighborhoods in each layer [22]. SGC first aggregates the features of out-of-sample nodes with the features of their neighbors among in-sample nodes, and then passes the aggregated feature through the multilayers neural network to obtain the final result [27]. In addition, some fine-tuning methods have widely been designed. For example, QE uses the kNN algorithm to find the k nearest neighbors in the original feature space, and then outputs predictions by approximating learned features of neighbors in the embedding space [47]. Similar to QE, Dou *et al.* [1] proposed to learn approximate features by using the features learned by the GNN model. However, these methods ignore the quality of the relationship between out-of-sample nodes and their neighbors among in-sample nodes, and thus the performance is limited.

III. Method

A. Problem Formulation

The graph is designed to model the pairwise relationships between two data points. To conduct feature representation while taking into account pairwise relationships between data points, the GNN model has several hidden layers, and each hidden layer includes feature learning and neighborhood aggregation. Based on [5], the graph convolution operator of the $(l+1)$th GNN layer holds the following equation:

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

where $\mathbf{D} \in \mathbf{R}^{n\times n}$ is a Laplacian matrix of the initial graph $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{W}^{l} \in \mathbb{R}^{d_l \times d_{l+1}}$ is a weight matrix in the lth layer (d_l and d_{l+1} represent dimensions), and $\sigma(.)$ is an activation function. Concretely, the embeddings $\mathbf{Z}$ are obtained from the last layer of the GNN model. Based on this, the final perceptron layer is defined as follows:

$$\mathbf{O} = \text{Softmax}(\text{Linear}(\mathbf{Z})) \tag{2}$$

where Linear$(\cdot)$ is used to regress the probability of each category, and $\mathbf{O} \in \mathbb{R}^{n\times c}$ denotes the output of the GCN model where c denotes the number of classes. Furthermore, the cross-entropy loss function is used to evaluate the classification performance

$$\mathcal{L}_{\text{GNN}} = -\sum_{v_i \in Y_L} (y_i \log(o_i) + (1-y_i)\log(1-o_i)) \tag{3}$$

where Y_L is the set of labeled nodes, y_i is the ground truth, and o_i is the corresponding prediction.

Recently, to overcome various drawbacks (e.g., noisy edges) in the initial graph, GLCN [19] proposed an adaptive GNN method to adaptively update the graph and learn features by

$$\mathcal{L} = \mathcal{L}_{\text{GNN}} + \gamma\, \mathcal{L}_{\text{GL}} \tag{4}$$

where γ is a hyperparameter and $\mathcal{L}_{\text{GL}}$ is obtained from [48]

$$\begin{aligned} \mathcal{L}_{\text{GL}}: \ & \min_{\mathbf{S}} \sum_{i,j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 s_{ij} + \|\mathbf{S}\|_F^2 \\ & \text{s.t.}, \ \sum_{j=1}^{n} s_{ij} = 1, \quad s_{ij} > 0, \ i, j = 1, \ldots, n \end{aligned} \tag{5}$$

where s_{ij} is the latent graph structure learned under the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Compared to other GNN methods, the adaptive GNN adaptively adjusts the graph and thus improves the quality of the graph to some extent [18]. However, it is difficult to guarantee the quality of the graph learned in the original feature space (i.e., $\mathcal{X}$), which contains noise and redundancy. Furthermore, the adaptive GNN method in (5) does not touch the issue of out-of-sample.

B. Reverse Graph Learning

We denote $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n\times D}$ as the node feature matrix on the input space $\mathcal{X}$ where the ith node $\mathbf{v}_i$ has a D-dimensional representation. We also denote the new representation of the ith node $\mathbf{v}_i$ as $\mathbf{z}_i \in \mathbb{R}^{1\times d}$ in the embedding space. We further investigate the reverse graph embedding to preserve the local structure of the data in the intrinsic space by meeting two kinds of consistency, i.e., semantic consistency and structure consistency.

First, semantic consistency is preserving original semantic information (e.g., principal components) in $\mathbf{X}$ while learning the reverse graph embedding in the intrinsic space. Specifically, the reverse graph embedding $\mathbf{ZP}$ should contain mainly semantic information in $\mathbf{X}$. Second, structure consistency preserves the local structure of data points in the input space $\mathcal{X}$. Specifically, the reverse graph embedding $\mathbf{z}_i\mathbf{P}$ and $\mathbf{z}_j\mathbf{P}$ should preserve the similarity between the ith node $\mathbf{v}_i$ and the jth

node $\mathbf{v}_j$. To do this, we assume that there exists an intrinsic space, where the new representation of $\mathbf{X}$ is represented by $\mathbf{ZP}$ via the transformation matrix $\mathbf{P}$. As a result, a function $f_\theta \in \mathcal{F}$ can be found to map the embedding space $\mathcal{Z}$ to the intrinsic space of the data, and (5) can be transferred to the following objective functions:

$$\begin{cases} \mathcal{L}_{\text{RGL}}: \ \min\limits_{\mathbf{S}} \ \sum\limits_{ij}^{n} \|\mathbf{z}_i\mathbf{P} - \mathbf{z}_j\mathbf{P}\|_2^2 s_{ij} + \|\mathbf{S}\|_F^2 & \text{(6a)} \\ \quad \text{s.t.,} \ \sum\limits_{j=1}^{n} s_{ij} = 1, \quad s_{ij} > 0, \ i, j = 1, \ldots, n \\ \mathcal{L}_{\text{RL}}: \ \min\limits_{\mathbf{P}, \mathbf{P}^T\mathbf{P}=\mathbf{I}} \ \|\mathbf{X} - \mathbf{ZP}\|_F^2 & \text{(6b)} \end{cases}$$

where the matrix of $\mathbf{P}^T$ is the transpose of $\mathbf{P}$ and $\mathbf{I}$ is an identity matrix. Equation (6a) achieves structure consistency by guiding the graph learning process, where a large distance $\|\mathbf{z}_i\mathbf{P} - \mathbf{z}_j\mathbf{P}\|_2^2$ between $\mathbf{v}_i$ and $\mathbf{v}_j$ leads to a small value of s_{ij}. Equation (6b) achieves semantic consistency by training the reverse mapping function from $\mathbf{Z}$ to $\mathbf{X}$. Note that $\|\mathbf{X} - \mathbf{ZP}\|$ is similar to the Auto-Encoder method as they learn the new representation $\mathbf{ZP}$ for $\mathbf{X}$. Moreover, the new representation is adjusted by the updated parameters until the value of the objective function becomes small. Using (6b) can lead to new representation $\mathbf{ZP}$ being focused on abstract latent factors (intrinsic properties), rather than details and noise. In this case, we say that it contains less noise than $\mathbf{X}$.

We list the difference between (6) and previous methods in (5) as follows. First, the original data $\mathbf{X}$ is refined by $\mathbf{ZP}$, i.e., $\mathbf{ZP}$ is an estimation of $\mathbf{X}$ as well as represents the structural information in the intrinsic space. In particular, the mapping function $\mathbf{P}$ projects data points in the embedding space to the intrinsic space, aiming at preserving semantic consistency. As a result, the data points in the intrinsic space through $\mathbf{ZP}$ can alleviate the issues in the original data such as noise and redundancy. It is worth noting that the graph matrix $\mathbf{S}$ in (6) is generated by a special graph learning layer, which will be described in the next section. Specifically, the graph matrix $\mathbf{S}$ is trained under the guidance of the $\mathbf{ZP}$ according to (6). Second, (6) is a reverse thought of the Laplacian eigenmap [49], by capturing the local structure in the intrinsic space. On the contrary, the Laplacian eigenmap finds the similarity between two data points in the original space. Although the learned representation $\mathbf{ZP}$ approaches to $\mathbf{X}$, the distances in two different spaces are explicitly altered. In particular, the space $\mathcal{Z}$ spanned by $\mathbf{ZP}$ contains less noise, compared to the input space $\mathcal{X}$, and thus possibly produces a high-quality graph.

Finally, the reverse graph learning in this work is different from that in [50]. More precisely, the reverse graph learning in [50] combines (6a) with (6b) in a formulation to simultaneously optimize the mapping function $\mathbf{P}$, the graph $\mathbf{S}$ and the new representation $\mathbf{Z}$. On the contrary, our method obtains the new representation $\mathbf{Z}$ from the GNN method, and then fixes the mapping function $\mathbf{P}$ in (6a) for learning the reverse graph from the intrinsic space spanned by $\mathbf{ZP}$.

C. Similarity Metric Learning for the New Graph

Similar to previous literature (e.g., [14], [18], [19]), we optimize the graph $\mathbf{S}$ in (6) approximately. Specifically, in the first layer of our proposed GNN model, we aim to optimize a new graph $\mathbf{S}$ representing the pairwise relationship of the new representation between two data points. To this end, we employ the cosine distance function to define the edge weight of the graph $\mathbf{S}$ as follows:

$$s_{ij} = \mathbf{cos}\big(\mathbf{x}_i\mathbf{W}^{(0)}, \mathbf{x}_j\mathbf{W}^{(0)}\big) \tag{7}$$

where s_{ij} is the edge weight between data point $\mathbf{v}_i$ and data point $\mathbf{v}_j$, i.e., $s_{ij} \in [0, 1]$, $\mathbf{W}^{(0)} \in \mathbf{R}^{D\times d'}$ ($d' \leq D$) is the weight matrix. Based on (7), the graph $\mathbf{S}$ will be updated at each epoch. Furthermore, we proceed to extract a sparse graph $\mathbf{S}$ by first keeping the similarity of the top k nearest neighbors for each node and set others as zeros. After that, we normalize $\mathbf{S}$ to satisfy the condition "$\sum_{j=1}^{n} s_{ij} = 1, s_{ij} > 0$" in (6a).

Based on the observation that the initial graph $\mathbf{A}$ may contain important information especially in the manual graph [18], the initial graph $\mathbf{A}$ should be involved for the update of $\mathbf{S}$ to maintain the stability of the training process and to find the missing edges. Hence, we further conduct the reverse graph learning by integrating the initial graph $\mathbf{A}$ and the reverse graph $\mathbf{S}$ as

$$\hat{\mathbf{A}} = (1 - \eta)\mathbf{A} + \eta\mathbf{S} \tag{8}$$

where η is a hyperparameter to make a trade-off between $\mathbf{S}$ and $\mathbf{A}$. The missing edges in $\mathbf{A}$ can be found from the reverse graph $\mathbf{S}$, so $\hat{\mathbf{A}}$ imputed to the GNN is with higher quality than the initial graph $\mathbf{A}$.

D. Objective Function

We list the framework of our proposed method in Fig. 1. Based on Fig. 1, the foreword process of our proposed two-layer rGNN method can be written as follows:

$$\mathbf{Z} = \sigma\left(\hat{\mathbf{A}}\sigma\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)}\right)\mathbf{W}^{(2)}\right) \tag{9}$$

where our method uses $\hat{\mathbf{A}}$ to replace $\mathbf{A}$. The advantages of our strategies are verified in Sections IV-B and IV-D. Finally, the objective function of our proposed rGNN method can be formulated as

$$\mathcal{L} = \mathcal{L}_{\text{GNN}} + \beta\mathcal{L}_{\text{RGL}} + \gamma\,\mathcal{L}_{\text{RL}} \tag{10}$$

where β and γ are two hyperparameters to make a trade-off between losses in (10). Moreover (10) is optimized iteratively in the following order, i.e., $\mathcal{L}_{\text{GNN}} \Rightarrow \mathcal{L}_{\text{RGL}} \Rightarrow \mathcal{L}_{\text{RL}}$. It is noteworthy that the GNN parameters (e.g., $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$) are fixed when optimizing the parameter of the graph learning layer, i.e., $\mathbf{P}$. In addition, the parameter $\mathbf{P}$ is fixed when optimizing the parameter in reverse graph learning (i.e., $\mathbf{W}^{(0)}$) and the GNN parameters (i.e., $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$).

E. Out-of-Sample Extension

Since the training process does not produce a model to directly predict unseen data points, most of previous GNN methods conduct semi-supervised learning where the unseen

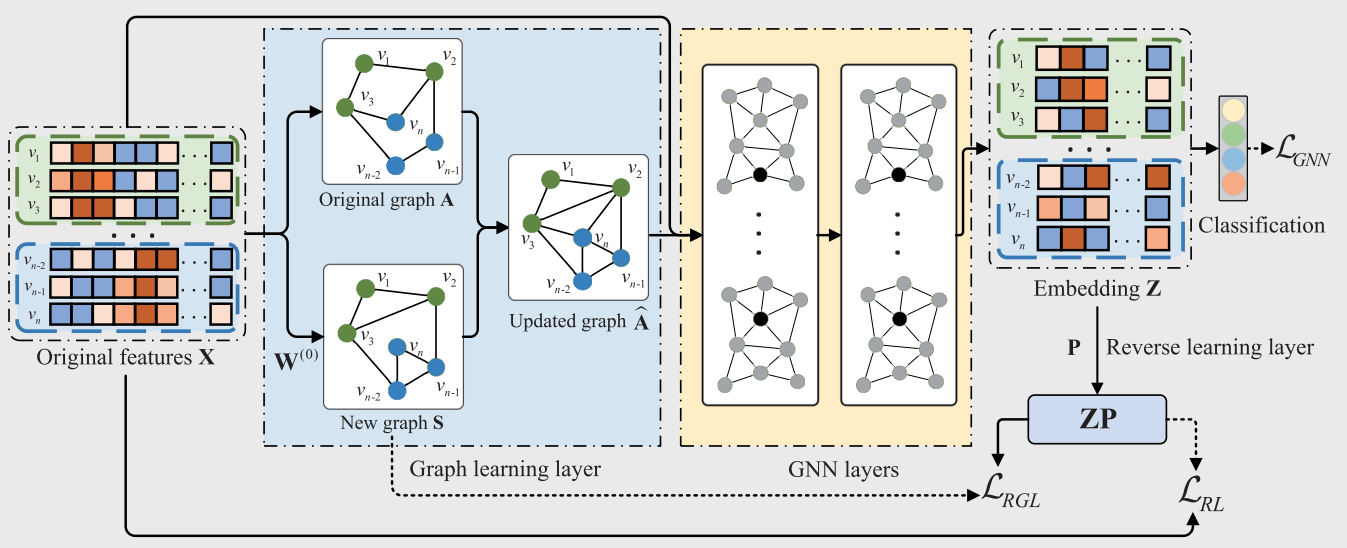

Fig. 1. Flowchart of our proposed method. A new graph $\mathbf{S}$ is generated by the graph learning layer and combined with the original graph $\mathbf{A}$ to form the updated graph $\hat{\mathbf{A}}$. The updated graph and the input features are then fed into the GNN layers for feature learning. The node embedding $\mathbf{Z}$ generated from the GNN layers is further fed into the reverse learning layer to generate the new representation $\mathbf{ZP}$ of the original data, which is used to guide the graph learning layer by (6a). The dashed line means the pathway of the loss.

test sample $\mathbf{x}_q$ can be found in the training dataset. This results in the issue of out-of-sample. To address this issue, query expansion (QE) [51] assigns the feature of the unseen data point as the weighted mean of its k nearest neighbors. Moreover, the weight is the similarity between the unseen data point and its neighbor. GraphSAGE [22] learns a function to generate the representation of the unseen data point by sampling and aggregating the representations from its local neighborhood.

However, these methods (e.g., QE and GraphSAGE) are limited to the quality of the nearest neighbors constructed from the high-dimensional original data. To address this issue, we investigate to optimize the neighborhood selection through the graph learning layer. Specifically, after the whole network is trained, we obtain $\mathbf{W}^{(0)}$ in the graph learning layer, $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ in the GNN layer, and $\mathbf{P}$ for the reverse learning. Given an unseen data point $\mathbf{x}_q$, we propose to use $\mathbf{W}^{(0)}$ to project it to the intrinsic low-dimensional space where the distance between two data points is discriminative. After that, the similarity between $\mathbf{x}_q$ and the training dataset $\mathbf{X}_{\text{train}}$ can be calculated by

$$\mathbf{S}_q = \cos(\mathbf{x}_q\mathbf{W}^{(0)}, \mathbf{X}_{\text{train}}\mathbf{W}^{(0)}). \tag{11}$$

We then select the neighborhood by

$$\mathcal{N}_q \leftarrow \text{topk}(\mathbf{S}_q). \tag{12}$$

In each hidden layer, the representation of the unseen data point is the summation of the weighted mean of its k nearest neighbors and its representation in the previous layer, i.e.,

$$\mathbf{H}_q^{(l)} = \sigma\left(\mathbf{W}^{(l)}\left(\sum_{\mathbf{x}_p \in \mathcal{N}_q} \mathbf{s}_{qp}\mathbf{f}_p^{(l)} + \mathbf{H}_q^{(l-1)}\right)\right) \tag{13}$$

where $\mathbf{f}_p^{(l)}$ is the representation of the neighborhood of the unseen data point at the lth layer. Finally, we use the softmax function to predict the label of the unseen data point.

The main idea of our method is to discover neighbors for unseen data points from the intrinsic space of data points, which is adaptively updated by the high-quality graph. In contrast, QE searches the neighbors from the high-dimensional original data. GraphSAGE keeps the neighbors unchanged in the training process while the neighbors are obtained from the original space.

Different from previous graph learning methods (e.g., GLCN and JLGCN), the proposed method is inductive by overcoming the issue of out-of-sample, i.e., generating an explicit mapping function in the training process to predict unseen data points. Moreover, previous graph learning methods learn the latent graph structure from the original feature space in (5), while our method learns the latent graph structure from the new representation in (6).

IV. Experiments

In this section, we compared our method with the comparison methods on real datasets in terms of five different kinds of real applications, including node classification in supervised learning, node classification in supervised learning via out-of-sample extension, random edge attack, link prediction, and image retrieval.

A. Experiment Setting

1) Datasets: We conducted experiments on 11 public datasets, including two citation datasets (e.g., Cora and Citeseer), four image datasets (e.g., Handwritten, Caltech, AWA, and Flower), three text datasets (e.g., 3Source, BBC, and WebKB), and two image retrieval datasets (e.g., Oxford5k and

TABLE I
DESCRIPTION OF THE DATASETS

Data sets	Samples	Classes	Dimensions	Types
Cora	2708	7	1433	Citation
Citeseer	3327	6	3703	Citation
Handwritten	2000	10	76	Image
Caltech	1474	7	1984	Image
AWA	4000	50	2688	Image
BBC	2225	5	1828	Text
WebKB	1051	2	2949	Text
3Source	416	6	352	Text
Flower	800	17	1360	Image
Oxford5k	5062	55 queries	2048	Retrieval
Paris6k	6412	55 queries	2048	Retrieval

Paris6k). We give a detail description about these datasets in Table I.

Cora is a citation network dataset consisting of 2708 data points with 5429 edges, and each data point has a 1433-D representation and belongs to one of seven categories.

Citeseer has 3327 data points, each represented by a 3703-D representation and belonging to one of six classes. The graph contains 4732 edges.

Handwritten[1] comes from the UCI machine learning repository and contains 2000 data points distributed across ten classes. Each data point has six view representations. For each image, we followed [19] to use the first view representation which contains 76-D Fourier coefficients of the character shapes as input feature vector in the experiments.

Caltech is an image dataset which includes 1474 images and seven classes. Six kinds of features were extracted for each image. For each image, we followed [19] to use the first view containing 1984-D HOG features for the experiments.

AWA[2] describes 50 animals by 4000 images. Each image consists of six types of features. The 2688-D color histogram features are used for each image in the experiments.

BBC consists of 2225 news documents with five categories including business, entertainment, politics, sports, and technique. Each document is divided into three sub-documents, each segment is viewed as one view, i.e., 1828, 1832, and 1845 news documents. In the experiments, we followed [19] to use the first view with an 1828-D feature vector for each data point.

WebKB is a dataset of web documents which has 1051 files distributed in two classes (i.e., course and non-course). Each sample has two kinds of feature representations, i.e., 2949-D for full-text features and 334-D for in-link features. We followed [19] to the 2949-D full-text features as the input representations to conduct experiments.

3Source[3] consists of 416 news stories (i.e., samples) within six classes. Each sample is represented by three views, such as 352-D BBC, 302-D Reuters, and 294-D Guardian. We followed [19] to the 352-D BBC text feature in our experiments.

Flower has 17 classes and each class includes 800 flower images. Each image is represented by three views, i.e., 1360-D color features, 1360-D texture features, and 1360-D shape features. The second view with 1360-D texture features was used in the experiments.

Oxford5k contains 5062 images collected from Flickr. It provides a set of 55 queries for 11 landmark, five for each landmark. The 2048-D features are extracted in the experiments using the GeM descriptor [47].

Paris6k consists of 6412 images. We followed the data partitioning and evaluation protocol proposed in guided similarity separation (GSS) [52] to obtain 2048-D embedded features for each image by the GeM descriptors [47].

2) Comparison Methods: In the experiments, the comparison methods included GCN [5], GLCNs [19], joint graph and feature learning GCN (JLGCN) [14], deep iterative and adaptive learning GNN (DIAL) [18], DeepWalk [53], GraphSAGE [22], simplifying graph convolutional networks (SGC) [27], GNN-based query feature expansion (QFE) [1], QE [47], Diffusion [54], exploit graph traversal (EGT) [55], GSS [52], and two variants of GNN, i.e., GNN* and GNN**. We list the details of the comparison methods as follows.

GCN aggregates the features of neighborhoods as well as updates the node features at each layer to obtain a new representation. This is the baseline method for the semi-supervised node classification task.

GLCN integrates both graph learning and graph convolution in unified network architecture and introduces a graph learning loss function that forces graph structure to contain edges on similar node pairs. It is used as a baseline for graph learning.

JLGCN conducts joint learning of the underlying graph structure and node features at each layer of the GCN simultaneously and employs Mahalanobis distance to calculate the distance between two nodes in the latent space.

DIAL aims to iteratively update underlying graph at each graph learning layer and designs a deep iterative adaptive graph learning methods to iteratively update the new graph.

DeepWalk is a graph embedding method which relies on random walk and the skip-gram algorithm to get feature information on the graph data under unsupervised setting.

GraphSAGE defines neighborhood aggregation functions, that each node's own features will aggregate feature information of sampled neighborhoods, from the previous layer.

SGC simplifies the online neighborhood aggregation step to a pre-processing step. The network employs a pre-processing step of the neighborhood aggregation and multiclass logistic regression to simplify the GCN. It can naturally handles the embedding of unseen nodes.

GNN* is the inductive version of the GCN [5]. This variant directly predicts the features of unseen nodes by using the learned weight of each layer in the GCN model without structural information.

GNN** first applies the kNN algorithm on the original data space to select k nearest neighbors for each unseen node, then leverages this structural information on the trained GCN model to compute the features of unseen data points.

QE is a fine-tuning method based on clustering, in which the kNN algorithm is used to compute the k nearest neighbors in the training dataset. QE fine-tunes to formulate a new query feature by averaging the features of its top nearest neighbors.

[1]https://archive.ics.uci.edu/ml/datasets/Multiple+Features
[2]http://attributes.kyb.tuebingen.mpg.de/
[3]http://mlg.ucd.ie/datasets/3sources.html

TABLE II
CLASSIFICATION ACCURACY OF ALL METHODS ON NINE DATASETS FOR SEMI-SUPERVISED NODE CLASSIFICATION

Methods	Cora	Citeseer	Handwritten	Caltech	AWA	BBC	WebKB	3sources	Flower
GCN	81.06$\pm$0.5	71.22$\pm$0.6	93.38$\pm$0.8	80.48$\pm$1.2	56.29$\pm$0.9	72.56$\pm$0.8	72.28$\pm$1.1	79.61$\pm$0.6	45.51$\pm$1.2
GLCN	81.80$\pm$0.7	70.56$\pm$0.9	94.22$\pm$0.9	79.36$\pm$1.4	55.81$\pm$0.7	73.36$\pm$1.0	75.76$\pm$0.9	79.22$\pm$0.6	46.65$\pm$1.0
DIAL	82.41$\pm$0.4	71.68$\pm$0.7	93.75$\pm$1.2	76.79$\pm$1.2	56.57$\pm$0.7	71.39$\pm$1.2	70.33$\pm$1.0	78.18$\pm$0.4	45.49$\pm$1.0
JLGCN	83.66$\pm$0.5	72.94$\pm$0.6	94.60$\pm$0.8	82.74$\pm$0.8	58.96$\pm$0.5	76.12$\pm$0.6	75.65$\pm$0.8	75.46$\pm$0.8	50.61$\pm$0.8
Proposed	**83.89**$\pm$0.7	**73.35**$\pm$0.8	**95.89**$\pm$0.9	**83.92**$\pm$0.6	**61.60**$\pm$0.5	**80.81**$\pm$0.5	**77.39**$\pm$0.7	**81.43**$\pm$0.3	**50.79**$\pm$0.7

QFE uses a global and local GNN-based query feature expansion for image retrieval, in which the local structure is constructed by kNN algorithm and the global structure is preserved by bipartite graph.

Diffusion is a generic framework in the scope of image retrieval. We explored the method in [54], where a new diffusion-embedding layer is introduced for feature diffusion.

EGT is a graph-based approach for image retrieval by traversing the kNN graph. It can effective retrieval with previously unseen images by integrating the new query in the graph and can be retrieved for other queries.

GSS uses a GNN model to replace the diffusion process for end-to-end training on image retrieval task. GSS deploys a unsupervised model for image retrieval based on the GNN.

3) Setting-Up: All experiments were conducted on a server with four Tesla V100 (32-GB memory each). We implemented our method on the PyTorch framework and obtained the codes of all comparison methods from the authors or online. In all experiments, we repeated the experiments ten times with random seeds for all methods and datasets to report the average results and the corresponding standard deviation (std).

In our method, we set the maximal number of epochs as 1000 for the training process with the Adam optimizer [56], and set the initial learning rate and the weight decay to 0.05 and 0.0005, respectively. We also applied a dropout ratio of 0.2 in front of each layer for all datasets. In all experiments, we implement the proposed method with the PyTorch framework and employ a graph convolutional neural network (GCN) with two hidden layers as the backbone network. Additionally, we re-implement all the comparison methods based on the public codes provided by the authors and use the same backbone networks as ours method. The hyperparameter β in (10) is set to 10 and γ is set to 1, for balancing the magnitudes of three terms, such as $\mathcal{L}_{\mathrm{RGL}}$, $\mathcal{L}_{\mathrm{RL}}$, and $\mathcal{L}_{\mathrm{GL}}$. In (7) and (12), the parameter k was set from 5 to 25 unless otherwise stated. The hyperparameter η in (8) was set from 0.2 to 0.8 unless otherwise stated. For a fair comparison, we obtained the author-verified codes for all comparison methods and searched the best hyperparameters via a grid search to achieve their the best performance.

B. Result Analysis of Semi-Supervised Node Classification

Semi-supervised node classification is the most widely used task to evaluate GNN methods, which provides only a small number of node labels to predict the full node labels. In this section, we evaluated our proposed method on nine datasets including Cora, Citeseer, Handwritten, Caltech, AWA, BBC, WebKB, 3sources, and Flower, compared to baseline method GCN [5] and graph learning-based methods including GLCN [19], DIAL [18], and JLGCN [14], in terms of node classification accuracy.

For experimental settings of semi-supervised node classification, we followed [27] to set the label ratio of Cora and Citeseer datasets, respectively, as 5.2% and 3.6%. We also followed [18] to set label ratio of other datasets used as 10%.

Table II indicates that our proposed rGNN method consistently outperformed all comparison methods on all datasets, followed by JLGCN, DIAL, GLCN, and GCN. For example, our method improved by on average 2.99% and 6.51%, respectively, compared to the best comparison method (i.e., JLGCN) and the baseline method (i.e., GCN) on all datasets. This demonstrates the advantages of our proposed method over the comparison methods, i.e., the reverse graph learning adjusts the quality of the graph by mapping the high-dimensional original data to its intrinsic space where the issues such as noise, redundancy, and curse of dimensionality are alleviated. Second, the methods (i.e., GLCN, JLGCN, DIAL, and our method) worked pretty well and outperformed GCN in our experiments. For example, our method achieved the maximal improvement (i.e., 11.36%) and the minimal improvement (i.e., 2.28%), compared to GCN, on the dataset BBC and 3sources. This observation verifies the advantages of the adaptive GNN methods over GCN because the graph structure learning layers were proposed to improve the original graph structure.

C. Result Analysis of Out-of-Sample Extension

In this section, we compared our method with the methods on out-of-sample extension, including DeepWalk, GraphSAGE, SGC, QFE, GNN*, and GNN**, on nine datasets such as Cora, Citeseer, Handwritten, Caltech, AWA, BBC, WebKB, 3sources, and Flower.

We randomly split the in-sample nodes and out-of-sample nodes as 80% nodes and 20% nodes for each datasets. We then trained the embedding GNN model for each method on the in-sample nodes to obtain the embeddings of in-sample nodes, followed by evaluating the node classification accuracy of out-of-sample nodes for all methods.

In Table III, our rGNN method achieved the best classification performance, followed by GNN**, GraphSAGE, GNN*, QFE, Deepwalk, and SGC. For instance, our method improved by on average 5.18%, 8.05%, 8.99%, 10.91%, 12.28%, and 21.19% respectively, compared to GNN**, GraphSAGE, GNN*, QFE, Deepwalk, and SGC, on all datasets. This demonstrates that simultaneously considering the quality of the graph and the dynamic graph learning in our method is

TABLE III
CLASSIFICATION ACCURACY OF ALL METHODS ON NINE DATASETS FOR OUT-OF-SAMPLE EXTENSION

	Cora	Citeseer	Handwritten	Caltech	AWA	BBC	WebKB	3sources	Flower
Deepwalk	85.23±0.2	61.81±0.1	96.94±0.1	83.45±0.3	61.88±0.2	78.91±0.2	79.29±0.2	83.47±0.4	53.12±0.6
GraphSAGE	84.68±0.4	72.97±0.3	96.72±0.1	**83.69**±0.2	65.21±0.3	79.50±0.3	76.27±0.7	85.52±0.5	58.52±0.7
SGC	73.43±0.3	68.32±0.5	88.95±0.2	81.14±0.4	62.63±0.4	74.74±0.4	61.83±1.2	70.22±0.4	50.59±0.5
GCN*	78.30±0.5	76.70±0.3	92.50±0.2	82.09±0.4	64.83±0.5	84.89±0.3	82.79±0.4	79.48±0.3	56.54±0.7
GCN**	88.71±0.2	81.29±0.4	97.05±0.1	81.61±0.3	64.92±0.5	85.61±0.3	80.93±0.6	89.47±0.1	56.03±0.6
QFE	79.85±0.2	70.12±0.3	97.00±0.1	80.27±0.5	62.11±0.5	84.31±0.1	80.72±0.7	87.37±0.2	49.93±0.9
Proposed	**89.26**±0.2	**81.47**±0.4	**97.90**±0.1	83.50±0.2	**70.30**±0.5	**92.95**±0.1	**89.77**±0.3	**92.63**±0.2	**62.36**±0.6

feasible. Second, our method is robust and stable, compared to all the comparison methods on all datasets. For example, GraphSAGE is much better on Cora dataset, but is less effective on the set (i.e., Citeseer), compared to GNN*. The possible reason is that the neighborhoods selected from the original data influence the quality of the graph as well as the robustness of the GNN method. On the contrary, our method searches the neighborhoods from the intrinsic space of the original data by the reverse graph learning layer to reduce the issues such as the influence of noise and redundancy, and thus achieving the best performance on all datasets.

D. Result Analysis of Random Edge Attack

Random edge attack task is to simulate the situation when the graph structure receives an attack, e.g., edge missing. To examine the performance of graph learning from incomplete graph data, we conducted experiments to evaluate the behaviors of our proposed method under different ratios of the random edge attack. Specifically, if the model can learn a clean and valuable graph structure, the impact of missing edges should be mitigated from incomplete graph data. In this section, we evaluated our proposed method on nine datasets including Cora, Citeseer, Handwritten, Caltech, AWA, BBC, WebKB, 3sources, and Flower, compared to baseline method GCN and graph learning-based methods including GLCN, DIAL, and JLGCN, in terms of different ratios of random edge attack. The evaluation metric is node classification accuracy.

For experimental settings of random edge attack, we constructed the initial graph with missing edges by randomly sampling. Specifically, we removed a ratio of the edges randomly from 0% to 100% with a step size of 10% in the original graph. That is, the higher the missing ratio is, the less information the original graph provides. In addition, other settings such as the partitioning of the dataset are consistent with the semi-supervised node classification task.

Fig. 2 lists the classification accuracy of all methods under different ratios of edges attack. First, our method achieved the best performance, followed by JLGCN, DIAL, GLCN, and GCN. Specifically, our method improved around 1.48% and 17.76%, compared to the best comparison method (i.e., JLGCN) and the worst comparison method (i.e., GCN), in terms of classification accuracy at 100% ratios of the random edge attack, on datasets Cora and Citeseer. The reason is that our method can discovers potential graph structures without initial graph information, thus reducing the impact of random edge attack. Second, graph learning methods (such as JLGCN, DIAL, and GLCN) can achieve better performance under the incomplete graph structure, compared to the GCN method. It illustrates that graph learning methods can against edge missing attack by exploring the sound graph structure. Third, our method consistently outperformed other graph learning methods. The reason is that our reverse graph learning method learns the graph structure from the intrinsic space. Lastly, the higher ratio the edge attack, the better the performance improvement of our method is, compared to GCN. For example, the classification performance of our method varies a little with the increasing of the random delete ratios on datasets Cora and Citeseer.

TABLE IV
MAP OF ALL METHODS ON DATASETS OXFORD5K AND PARIS6K FOR IMAGE RETRIEVAL

Method	Oxford5k		Paris6k	
	Medium	Hard	Medium	Hard
QE	67.20±0.4	40.80±0.3	80.70±0.5	61.80±0.3
Diffusion	72.06±0.2	47.63±0.3	91.27±0.4	83.14±0.2
EGT	70.04±0.2	47.65±0.1	87.00±0.3	74.15±0.1
GSS	77.80±0.1	57.50±0.1	92.40±0.3	83.50±0.1
Proposed	**78.70**±0.2	**58.49**±0.1	**92.58**±0.3	**83.62**±0.1

E. Result Analysis of Image Retrieval

Image retrieval task aims to find similar images to a query image among an image dataset. In this section, we evaluated our proposed method on datasets Oxford5k and Paris6k, compared to two fine-tuning-based methods (i.e., QE and Diffusion) and two GNN-based methods (i.e., EGT and GSS). The evaluation metric is mean average precision (mAP).

For image retrieval task, the evaluation were divided into Easy, Medium, and Hard tasks based on the complexity of the retrieval task. In our experiments, we followed [52] to focus on the more challenging Medium and Hard tasks. Moreover, for each image, we extracted a 2048-D feature vector by a pre-trained GeM descriptor [47].

We listed the mAP results of all methods in Table IV. Our method achieved the best performance, compared to the state-of-the-art methods, on two datasets for the image retrieval. For example, our method improved by on average 0.8% and 27.62%, respectively, compared to the best comparison method (i.e., GSS) and the baseline method (i.e., QE). This shows the reasonability of our method for the learning the high-quality graph in the GNN model.

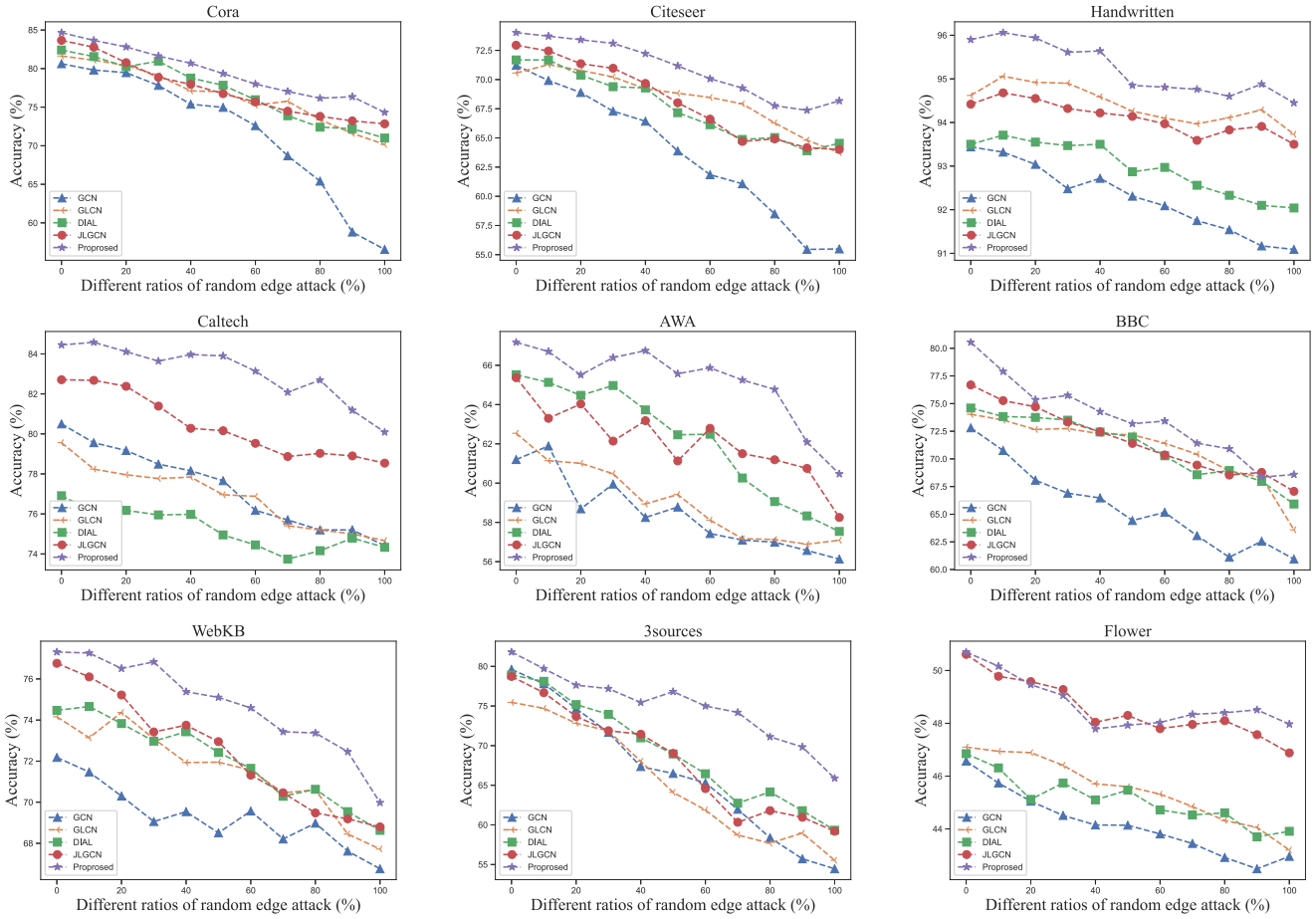

Fig. 2. Results of all models under different ratios of random edge attack.

F. Result Analysis of Link Prediction

Link prediction is a task to estimate the probability of edges between nodes in a graph. In this section, we evaluated our proposed method on two datasets Cora and Citeseer because they provided the initial graph for link prediction. The comparison methods had GCN and graph learning-based methods (i.e., GLCN, DIAL, and JLGCN), in terms of Area under the ROC Curve (AUC).

For link prediction task, we randomly masked a portion of the edges and random sampled the same number of negative edges in the initial graph, and then inferred unobserved real links. we progressively performed the missing ratios from 0.1 to 0.3 on two datasets Cora and Citeseer.

Fig. 3 reports the AUC of all methods on Cora and Citeseer datasets. First, all graph-based methods outperformed the GCN in terms of different occlusion rates on all two datasets. For example, our method improved on average by 0.82% and 12.32%, respectively, compared to DIAL and GNN, on all two datasets. The reason is that the negative effects caused by the missing edges and the noise information in original graph in our method can be improved by the reverse graph learning so that more useful information can be mined for each node to boost prediction performance.

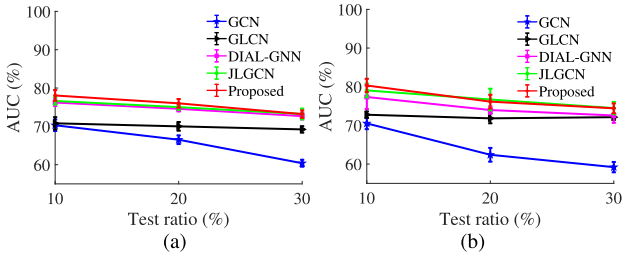

Fig. 3. AUC of all methods on datasets: (a) Cora and (b) Citeseer.

G. Ablation Study

1) Effectiveness of Reverse Graph Learning: The key difference between our rGNN method and previous GNN methods [1], [22], [27] is the reverse graph learning. We considered the effectiveness of the proposed reverse graph learning for unseen data points by using our proposed method as the training process and the QE method in [1] to solve the issue of out-of-sample to generate a new method named Proposed-QE. We listed the classification accuracy of our method and Proposed-QE in Fig. 4(a). The accuracy of our method improved by on average 8.05%, compared to Proposed-QE, on all datasets. This indicates that the reverse graph learning in our method is reasonable. In addition, Proposed-QE improved

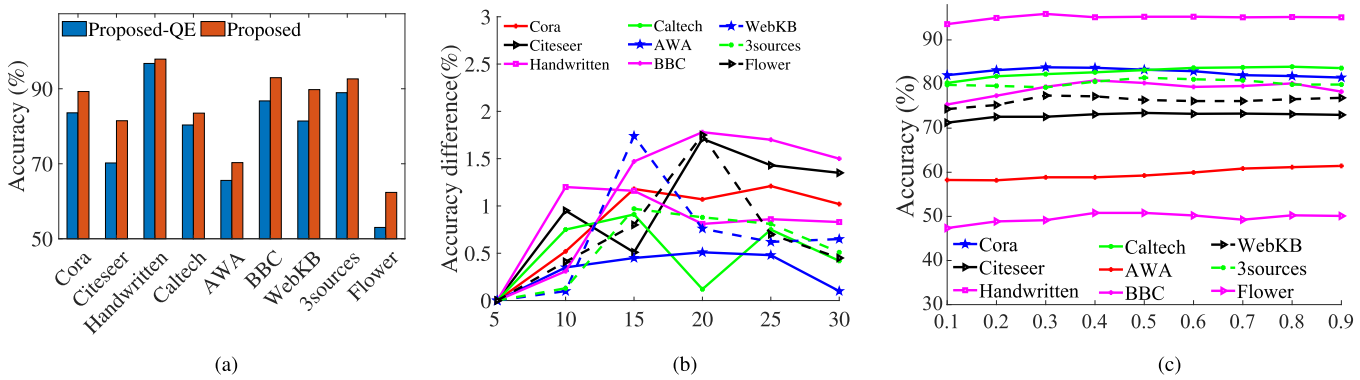

Fig. 4. Experimental results of ablation study on all datasets. (a) Effectiveness of our reverse graph learning and Proposed-QE on out-of-sample extension. (b) Sensitivity analysis of different values of k in Section III-C. (c) Sensitivity analysis of η in (8).

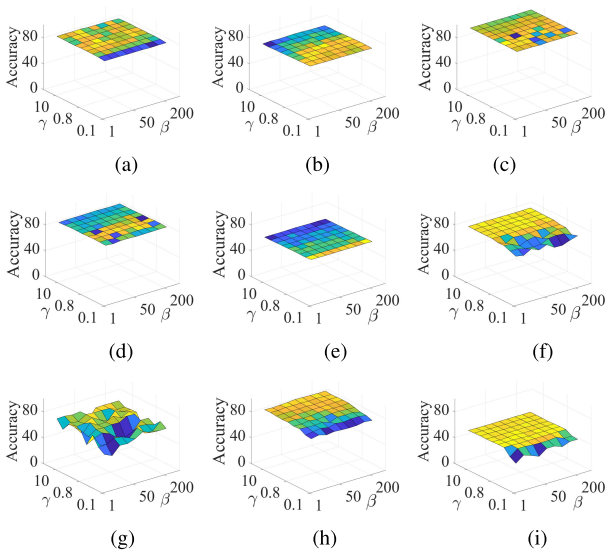

Fig. 5. Classification accuracy of our method with different parameter settings (i.e., β and γ). (a) Cora. (b) Citeseer. (c) Handwritten. (d) Caltech. (e) AWA. (f) BBC. (g) WebKB. (h) 3sources. (i) Flower.

by on average 2.18%, compared to QFE in Table III, on all datasets, in terms of classification accuracy. This verifies the advantages of the $\hat{\mathbf{A}}$ output from our proposed graph learning layer is better at selecting neighbors than the original $\mathbf{A}$ again.

2) Parameter Sensitivity: The value of k in Section III-C is the number of nearest neighbors. We set the range for the value of k to {5, 10, 15, 20, 25, 30}. As shown in Fig. 4(b), the classification accuracy of our method increased with increasing values of k, i.e., from $k = 5$ to $k = 20$. However, the classification performance of our method decreased with the increase of the k value, i.e., from $k = 20$ to $k = 30$. The reason is that small k value cannot fully describe the neighborhoods of the data. A large value of k represents more information about the neighbors, and theoretically the more the better. But a large value of k will increase the chance of wrong neighborhoods, making the relationship between data points less discriminative, and too many neighbors can be cause over-smooth issue in the graph. In this observation, we choose a moderate value of k. This demonstrates that: 1) the GNN method and its variants are sensitive to the number of neighborhoods and the accuracy of neighborhoods and 2) the optimal value of k can be easily found in our method (i.e., around $k = 15$) on all datasets.

The parameter η in (8) conducts the trade-off between the initial graph $\mathbf{A}$ and the new graph $\mathbf{S}$. The larger the value of η, the more important the new graph $\mathbf{S}$ is. To this end, we varied the values of η from 0.1 and 0.9, as shown in Fig. 4(c). As a result, first, the classification performance steadily increased with the value of η increased (i.e., from 0.1 to 0.4). This indicates that the new graph plays an essential role in the training process and should be considered for the GNN model. Second, increasing the value of η beyond a threshold can hurt the performance, i.e., larger than 0.5, implying that the initial graph maintains the necessary information and helps to maintain the stability of the training, as shown in [18].

We reported the classification accuracy of our method with different parameter settings (i.e., β and γ) in Fig. 5. Specifically, β and γ control the balance among three terms, i.e., $\mathcal{L}_{\text{GNN}}$, $\mathcal{L}_{\text{RGL}}$, and $\mathcal{L}_{\text{RL}}$. As a result, even the parameters β is set from 1 to 200 and γ is set from 0.01 to 10, the variation of the classification accuracy of our method is relatively stable (i.e., less than 5% for the change ratio), so the other parameters (i.e., β and γ) are insensitive in our experiments. On the contrary, all these results are better than GNN. This indicates that two regularization terms (i.e., $\mathcal{L}_{\text{RGL}}$ and $\mathcal{L}_{\text{RL}}$) are necessary in our proposed method.

V. Conclusion

This article proposed a new GNN method that improves the quality of the graph used in GNN methods and addresses the out-of-sample problem by jointly performing reverse graph learning and feature learning. Experimental results on real datasets showed that our method outperformed the state-of-the-art method on a number of datasets and various applications. Moreover, the experimental results also demonstrated the effectiveness of the proposed reverse graph learning and out-of-sample extension method. In our future work, we will designed a simple framework to achieve effective and efficient graph learning as reverse graph learning requires one more step.

REFERENCES

[1] Z. Dou, H. Cui, L. Zhang, and B. Wang, "Learning global and local consistent representations for unsupervised image retrieval via deep graph diffusion networks," 2020, *arXiv:2001.01284*.

[2] Y. Zhu, J. Ma, C. Yuan, and X. Zhu, "Interpretable learning based dynamic graph convolutional networks for Alzheimer's disease analysis," *Inf. Fusion*, vol. 77, pp. 53–61, Jan. 2022.

[3] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li, "Learning discriminative binary codes for large-scale cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2494–2507, May 2017.

[4] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210–3221, Jul. 2018.

[5] N. T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017.

[6] X. Zhu, S. Zhang, Y. Zhu, P. Zhu, and Y. Gao, "Unsupervised spectral feature selection with dynamic hyper-graph learning," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 18, 2020, doi: 10.1109/TKDE.2020.3017250.

[7] J. Gan, Z. Peng, X. Zhu, R. Hu, J. Ma, and G. Wu, "Brain functional connectivity analysis based on multi-graph fusion," *Med. Image Anal.*, vol. 71, Jul. 2021, Art. no. 102057.

[8] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. ESWC*, 2018, pp. 593–607.

[9] Z. Hao *et al.*, "ASGN: An active semi-supervised graph neural network for molecular property prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 731–752.

[10] P. Berger, G. Hannak, and G. Matz, "Efficient graph learning from noisy and incomplete data," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 105–119, 2020.

[11] J. Wei, Y. Yang, X. Xu, X. Zhu, and H. T. Shen, "Universal weighting metric learning for cross-modal retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jun. 14, 2021, doi: 10.1109/TPAMI.2021.3088863.

[12] S. Hao, T. Chen, Y. Wang, Y. Guo, and M. Wang, "Adaptive multi-task dual-structured learning with its application on Alzheimer's disease study," *ACM Trans. Internet Technol.*, vol. 21, no. 2, pp. 1–16, May 2021.

[13] C. Yuan, Z. Zhong, C. Lei, X. Zhu, and R. Hu, "Adaptive reverse graph learning for robust subspace learning," *Inf. Process. Manage.*, vol. 58, no. 6, Nov. 2021, Art. no. 102733, doi: 10.1016/j.ipm.2021.102733.

[14] J. Tang, W. Hu, X. Gao, and Z. Guo, "Joint learning of graph representation and node features in graph convolutional neural networks," 2019, *arXiv:1909.04931*.

[15] R. Hu *et al.*, "Multi-band brain network analysis for functional neuroimaging biomarker identification," *IEEE Trans. Med. Imag.*, vol. 40, no. 12, pp. 3843–3855, Dec. 2021.

[16] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018.

[17] R. Hu, X. Zhu, Y. Zhu, and J. Gan, "Robust SVM with adaptive graph learning," *World Wide Web*, vol. 23, pp. 1945–1968, Dec. 2020.

[18] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *Proc. NIPS*, vol. 33, 2020, pp. 19314–19326.

[19] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proc. CVPR*, Jun. 2019, pp. 11313–11320.

[20] H. Strange and R. Zwiggelaar, "A generalised solution to the out-of-sample extension problem in manifold learning," in *Proc. AAAI Conf. Artif. Intell.*, 2011, pp. 471–476.

[21] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *Proc. ICCV*, Oct. 2007, pp. 1–8.

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1024–1034.

[23] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012, doi: 10.1016/j.aiopen.2021.01.001.

[24] V. Prakash Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," 2020, *arXiv:2003.00982*.

[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. ICLR*, 2014.

[26] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, vol. 2016, pp. 3844–3852.

[27] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. ICML*, 2019, pp. 6861–6871.

[28] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *NIPS*, pp. 1993–2001, 2016.

[29] W. Jin *et al.*, "Self-supervised learning on graphs: Deep insights and new direction," 2020, *arXiv:2006.10141*.

[30] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," 2021, *arXiv:2102.10757*.

[31] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. ICLR*, 2019.

[32] C. Mavromatis and G. Karypis, "Graph infoclust: Maximizing coarse-grain mutual information in graphs," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2021, pp. 541–553.

[33] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," 2020, *arXiv:2007.06686*.

[34] F. Faez, Y. Ommi, M. S. Baghshah, and H. R. Rabiee, "Deep graph generators: A survey," *IEEE Access*, vol. 9, pp. 106675–106702, 2021.

[35] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," in *Proc. ICML*, 2018, pp. 1–11.

[36] X. Guo, Y. Du, S. Tadepalli, L. Zhao, and A. Shehu, "Generating tertiary protein structures via interpretable graph variational autoencoders," *Bioinf. Adv.*, vol. 1, no. 1, Jun. 2021, doi: 10.1093/bioadv/vbab036.

[37] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[38] X. Zhu *et al.*, "Joint prediction and time estimation of COVID-19 developing severe symptoms using chest CT scan," *Med. Image Anal.*, vol. 67, Jan. 2021, Art. no. 101824.

[39] H. T. Shen, Y. Zhu, W. Zheng, and X. Zhu, "Half-quadratic minimization for unsupervised feature selection on incomplete data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3122–3135, Jul. 2021.

[40] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3034–3044, Dec. 2018.

[41] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11015–11023.

[42] P. Elinas, E. V. Bonilla, and L. Tiao, "Variational inference for graph convolutional networks in the absence of graph data and adversarial settings," in *Proc. NIPS*, 2021, pp. 18648–18660.

[43] X. Pu, S. L. Chau, X. Dong, and D. Sejdinovic, "Kernel-based graph learning from smooth signals: A functional viewpoint," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 7, pp. 192–207, 2021.

[44] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, and Y. Ye, "Heterogeneous graph structure learning for graph neural networks," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 1–9.

[45] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 66–74.

[46] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3D point cloud denoising," *IEEE Trans. Signal Process.*, vol. 68, pp. 2841–2856, 2020.

[47] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with, no., human annotation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1655–1668, Jun. 2018.

[48] F. Nie, X. Wang, M. I. Jordan, and H. Huang, "The constrained Laplacian rank algorithm for graph-based clustering," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1969–1976.

[49] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. NIPS*, 2002, pp. 585–591.

[50] Q. Mao, L. Wang, S. Goodison, and Y. Sun, "Dimensionality reduction via graph structure learning," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 765–774.

[51] R. Padaki, Z. Dai, and J. Callan, "Rethinking query expansion for BERT reranking," in *Proc. Eur. Conf. Inf. Retr.* Springer, 2020, pp. 297–304.

[52] C. Liu *et al.*, "Guided similarity separation for image retrieval," in *Proc. NIPS*, 2019, pp. 1556–1566.

[53] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.

[54] F. Yang, R. Hinami, Y. Matsui, S. Ly, and S. Satoh, "Efficient image retrieval via decoupling diffusion into online and offline processing," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9087–9094.

[55] C. Chang, G. Yu, C. Liu, and M. Volkovs, "Explore-exploit graph traversal for image retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9423–9431.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

Liang Peng received the B.S. degree in software from the University of Electronic Science and Technology of China, Chengdu, China, in 2018, and the master's degree from the Intelligent Information Technologies and Applications Laboratory, University of Electronic Science and Technology of China, in 2019, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.

He worked with NVIDIA, Shanghai, China, in 2017. His current research interests include graph representation learning and medical image analysis.

Rongyao Hu received the M.S. degree in computer science from Guangxi Normal University, Guilin, China. He is currently pursuing the Ph.D. degree with Massey University at Auckland, Auckland, New Zealand.

He has published 25 articles in English, including one ESI Hotspot article and four ESI Highly Cited articles. He has served as a reviewer for several top journals and international conferences, such as IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), *Information Fusion*, IEEE TRANSACTIONS ON CYBERNETICS, the Association for the Advancement of Artificial Intelligence (AAAI) from 2020 to 2022, International Joint Conferences on Artificial Intelligence Organization (IJCAI) from 2021 to 2022, and ACM International Conference on Multimedia (ACM MM) from 2021 to 2022. His research interests include feature extraction, graph representation learning, and image detection.

Fei Kong received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2020, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering.

His research interests include graph representation learning, its related applications and adversarial examples.

Jiangzhang Gan received the M.S. degree in computer science and technology from Guangxi Normal University, Guilin, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Natural and Computational Science, Massey University at Auckland, Auckland, New Zealand.

His current research interests include medical image analysis, feature selection, graph representation learning, and sparse learning.

Yujie Mo received the B.S. degree in computer science and technology from Northeastern University, Shenyang, China, in 2020. He is currently pursuing the M.S. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include graph representation learning and related applications.

Xiaoshuang Shi received the B.S. degree in automation from Northwestern Polytechnical University, Xi'an, China, in 2009, the M.S. degree in automation from Tsinghua University, Beijing, China, in 2013, and the Ph.D. degree from the J. Crayton Pruitt Family Department of Biomedical Engineering, University of Florida, Gainesville, FL, USA, in 2019.

From September 2013 to April 2015, he was a Research Assistant with the Shenzhen Key Laboratory of Broadband Network & Multimedia, Graduate School at Shenzhen, Tsinghua University, Shenzhen, China. His current research interests include large-scale image retrieval, deep learning, and medical image analysis.

Xiaofeng Zhu (Senior Member, IEEE) received the Ph.D. degree in computer science from The University of Queensland, Brisbane, QLD, Australia, in 2013. He is currently with the University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include big data, artificial intelligence, and medical image analysis.