Unsupervised Graph Representation Learning with Cluster-aware Self-training and Refining

YANQIAO ZHU, University of California, Los Angeles, USA
YICHEN XU, École Polytechnique Fédérale de Lausanne, Switzerland
FENG YU, DP Technology, China
QIANG LIU and SHU WU, Center for Research on Intelligent Perception and Computing, Institute of Automation, Chinese Academy of Sciences, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, China

Unsupervised graph representation learning aims to learn low-dimensional node embeddings without supervision while preserving graph topological structures and node attributive features. Previous Graph Neural Networks (GNN) require a large number of labeled nodes, which may not be accessible in real-world applications. To this end, we present a novel unsupervised graph neural network model with Cluster-aware Self-training and Refining (CLEAR). Specifically, in the proposed CLEAR model, we perform clustering on the node embeddings and update the model parameters by predicting the cluster assignments. To avoid degenerate solutions of clustering, we formulate the graph clustering problem as an optimal transport problem and leverage a balanced clustering strategy. Moreover, we observe that graphs often contain inter-class edges, which mislead the GNN model to aggregate noisy information from neighborhood nodes. Therefore, we propose to refine the graph topology by strengthening intra-class edges and reducing node connections between different classes based on cluster labels, which better preserves cluster structures in the embedding space. We conduct comprehensive experiments on two benchmark tasks using real-world datasets. The results demonstrate the superior performance of the proposed model over baseline methods. Notably, our model gains over 7% improvements in terms of accuracy on node clustering over state-of-the-arts.

CCS Concepts: • **Computing methodologies** → **Unsupervised learning**; **Neural networks**; • **Information systems** → *Data mining;*

Additional Key Words and Phrases: Cluster-aware self-training and refining, unsupervised learning, graph representation learning

Y. Zhu and Y. Xu contributed equally to this research.

This work is supported by National Natural Science Foundation of China (Grant No. 62141608, No. U19B2038, and No. 62206291).

Authors' addresses: Y. Zhu, University of California, Los Angeles, 3551 Boelter Hall, 580 Portola Plaza, Los Angeles, CA 90095; email: yzhu@cs.ucla.edu; Y. Xu, EPFL CH-1015 Lausanne, Switzerland; email: yichen.xu@epfl.ch; F. Yu, DP Technology, No. 2, Haidian East 3rd Street, Haidian District, Beijing, China, 100080; email: yufeng@dp.tech; Q. Liu and S. Wu (corresponding author), Center for Research on Intelligent Perception and Computing, Institute of Automation, Chinese Academy of Sciences, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, 95 Zhongguancun East Road, Haidian District, Beijing, China, 100190; emails: {qiang.liu, shu.wu}@nlpr.ia.ac.cn.

2157-6904/2023/08-ART82 $15.00
https://doi.org/10.1145/3608480

ACM Reference format:
Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, and Shu Wu. 2023. Unsupervised Graph Representation Learning with Cluster-aware Self-training and Refining. *ACM Trans. Intell. Syst. Technol.* 14, 5, Article 82 (August 2023), 21 pages.
https://doi.org/10.1145/3608480

1 INTRODUCTION

Graph data is fast becoming a key instrument for understanding complex interactions among real-world objects, for instance, biochemical molecules, protein-protein interactions, purchase networks from e-buy websites, and academic collaboration networks. Recent years have witnessed a surge of graph representation learning methods, which aims to encode nodes and graphs to low-dimensional vector spaces to better serve analysis of graph data. Recently, the **Graph Neural Network (GNN)** model, as a generalized form of convolutional networks in the graph domain, has attracted a lot of attention. Compared with conventional graph embedding methods, GNN shows superior expressive power and has achieved promising performance in many tasks [11, 13, 22, 41, 50].

Despite its great success, one of the predominate problems of most GNNs is that they are established in (semi-)supervised settings and thus require a substantial amount of high-quality labeled data, which in turn, has sparked an effort of unsupervised training for GNNs. During unsupervised training, the main hurdle lies in the absence of label information. Therefore, we have to leverage other supervisory signals captured from intrinsic graph properties to train the model. Previously, classical approaches formulate unsupervised learning as a link prediction problem [17, 31, 32]. They mask a portion of links in the graph and then train the model by enforcing it to predict the masked links. However, they are limited to fine-grained structures in the graphs and have difficulty in leveraging the node attributes in the learning process. In network science, clusters group nodes that share similar functionalities in a graph. The cluster assignments can thus reflect semantic meanings of nodes in graphs and are often informative for downstream tasks. For example, clustering assignments may correspond to item categories in co-purchase networks and authors' research domains in academic co-authorship networks. Therefore, node clusters, as a natural characteristic of graph data, can be used as a good supervision signal to guide the learning of graph embeddings.

However, it is non-trivial to train the GNN model with node clustering, due to the following two reasons. First, if we simply combine the cross-entropy loss with an off-of-the-shelf clustering algorithm, then the model will easily collapse into a trivial solution, which maps all nodes into a point in the embedding space [2, 6]. To avoid the degenerate solution, we formulate node clustering as an optimal transportation problem and add a constraint to regularize the clusters to be balanced. Second, when leveraging clustering assignments to optimize the model, the quality of produced graph representations highly depends on node clusters. If the clusters align well with the ground-truth label of downstream classification tasks, then they can enforce the model to capture essential semantic information and thus improve representation quality. However, we observe that graphs often contain noisy edges, which connect nodes belonging to different clusters. Such edges may mislead the clustering algorithm and further confine the model from learning useful class information. In a graph with many inter-class edges, when performing graph convolutions through neighborhood aggregation, the learned node embeddings tend to be indistinguishable from different classes [7, 8, 24, 47] and result in misaligned clustering assignments. Therefore, we argue that a key to improving the quality of embeddings is to alleviate the impact of potentially noisy edges and strengthen edges between nodes of the same class, which will help preserve the cluster structures and obtain better-separated node embeddings.

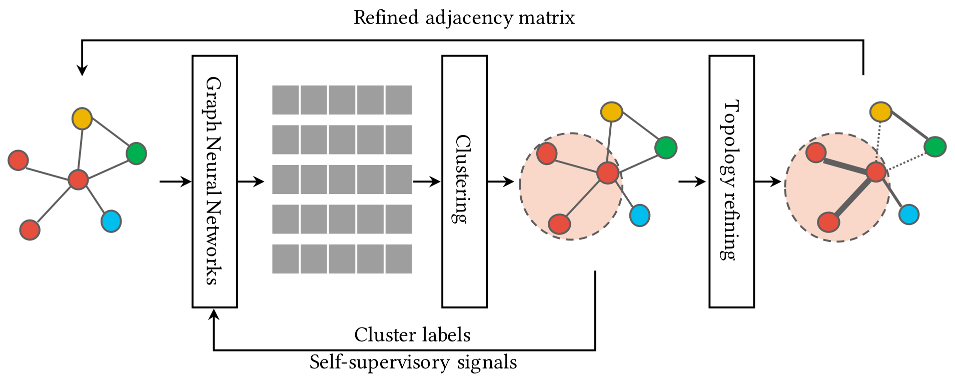


Fig. 1. The pipeline of the proposed CLEAR model. Overall, the CLEAR model alternates between node representation learning and clustering. We first obtain node embeddings using Graph Neural Networks (GNNs). Then, we perform clustering and use the cluster labels as the self-supervision signals. Following that, we leverage a novel cluster-aware topology refining mechanism that reduces inter-cluster edges and strengthens intra-class connections to mitigate the impact of noisy edges.

In this article, we propose a novel **Cluster-aware Self-training and Refining (CLEAR)** model for unsupervised GNNs. As illustrated in Figure 1, our CLEAR model consists of three stages. At the first stage, we perform graph convolutions to obtain node embeddings. Then, the model conducts clustering on the node embeddings and updates the model parameters by predicting the corresponding cluster assignments. To avoid degenerate solutions, we employ a cluster balancing strategy, which formulates the cross-entropy minimization as an optimal transport problem that can be effectively solved in near-linear time with the Greenkhorn algorithm [1]. Furthermore, to alleviate the impact of noisy edges and better preserve cluster structures in the embedding space, we propose a novel graph topology refining scheme based on cluster assignments. The proposed refining scheme strengthens intra-class edges and weakens potentially noisy edges by isolating neighborhood nodes of different clusters.

To summarize, the core contribution of this article is threefold. First, we propose a novel unsupervised GNN model with cluster-aware self-training, which learns embeddings using intrinsic network cluster properties and thus needs no direct supervision from labels. Second, unlike other GNN models that rely on a static graph structure, CLEAR further proposes a topology refining scheme that reduces inter-cluster connections of neighbor nodes to alleviate the impact of noisy edges. Third, extensive experiments conducted on public benchmark datasets demonstrate the superiority over existing baseline methods. It is worth mentioning that the proposed method gains over 7% performance improvement in terms of accuracy on node clustering over state-of-the-arts.

The organization of the remaining of the article is summarized below. We first review prior arts in relevant domains in Section 2. Then, in Section 3, we introduce our proposed CLEAR model in detail. After that, we present empirical studies in Section 4. Finally, we conclude the article and point out future research directions in Section 5.

2 RELATED WORK

2.1 Unsupervised Representation Learning on Visual Data

To alleviate the dependency on abundant manual annotations, unsupervised learning techniques that train the model with predefined pretext tasks is attracting increasing interests. The pretext tasks constructed from the raw labels can produce general embeddings for various downstream machine learning tasks of interest. Many strategies for pretext learning tasks, such as image in-painting [29], jigsaw puzzles [27, 28], grayscale image colorizing [23, 51], and geometric transformation recognition [16], have been proposed recently. However, the methods using the classification objective, which minimizes the cross-entropy loss, still obtain the best performance [2, 6]. Along this line, many methods focus on how to obtain proper labels for the classification task. For example, DeepCluster [6] is proposed to iteratively cluster images using *k*Means; the cluster assignments are then fed as supervision to train the convolutional network. NAT [3] proposes to fix a set of randomly initialized target vectors and train the network by aligning the embeddings to targets. Recently, Asano et al. [2] combine representation learning and clustering and propose a novel self-labeling scheme to balance the size of clusters, which outperforms existing methods.

2.2 Unsupervised Representation Learning on Graphs

Representation learning on graphs is far more complex than image data due to the non-Euclidean property and the lack of spatial locality. Classical methods learn unsupervised graph representation based on random walks [4, 17, 31, 33], which sample random walk sequences from the graph and learn node embeddings using sequential models. Apart from random-walk-based methods, another line of development focuses on matrix factorization techniques [37, 44], which produce node embeddings by explicitly factorizing the proximity matrix. Notably, Qiu et al. [32] manage to theoretically unify random walks and matrix factorization techniques into a cohesive framework. Recently, to accelerate computational tasks on large-scale graphs, NRL-MF [25] proposes a network representation lightening framework based on matrix factorization. This approach employs both hashing and quantization approaches and has delivered remarkable performance in node classification and recommendation tasks. However, traditional network embedding methods may suffer from insufficient representation ability, because they generate embedding for each node independently and no parameters are shared between nodes [18].

GNNs, however, encode both structure and node features into dense embeddings via message passing in the local neighborhood and achieve strong expressive power [22, 40, 45, 46]. Nevertheless, most GNNs are established in the (semi-)supervised setting, and requires substantial label annotations to be trained. In reality, it is expensive to obtain high quality label annotations. To mitigate the problem of label scarcity, there is a growing body of literature focusing on unsupervised training of GNNs. One line of research work proposes to employ GNNs as autoencoders [21, 43], which formulates unsupervised learning as a link prediction problem and leverages the raw graph structures as supervision. The representative GAE method [21] optimizes node embeddings by reconstructing the original graph topology from learned representations. To take node attributes into consideration, Gao and Huang [14] propose to use two graph autoencoders to preserve the proximity of both graph topology and node attributes, respectively. As another promising research direction, a number of research work focuses on training unsupervised GNNs based on the InfoMax principle [26]. Pioneering work DGI [41] utilizes a contrastive objective that discriminate node embeddings from the original graph and a corrupted graph, whose training objective is proved to be a lower bound of the **Mutual Information (MI)** between the input graph and the learned representations. Follow-up work proposes the **graphical mutual information (GMI)** [30] that takes mutual information between edges into consideration. Following this work, Zhu et al. [53] propose to generate graph views with stochastic graph augmentation functions and directly maximize the agreement between node embeddings across views.

Despite their success, Tschannen et al. [38] point out that the embedding equality is not strongly correlated with the MI bound and the stricter bound can even bring worse performance. In other words, the success behind the above contrastive learning models may be attributed to the design of augmentation and contrastive architectures. Similarly, our proposed CLEAR approach also involves information maximization. However, different from maximizing the mutual information between graph representation, our approach directly optimize the information between representation and clustering assignments using the cross-entropy objective.

2.3 Graph Clustering with Graph Neural Networks

In the deep learning era, a number of methods solve the graph clustering problem using GNNs. Zhang et al. [52] propose to address the graph clustering problem via a plain neighborhood aggregation scheme; their proposed method simply aggregates information from neighborhood nodes without parameters to learn from data. Wang et al. [42] propose to use a deep-clustering-based method on node embeddings for graph clustering, where a cluster hardening loss [39] is introduced to emphasize the clusters with high confidence. Moreover, the same deep clustering scheme has been applied to semi-supervised learning with few labels as well [36]. In this work, the clustering algorithm incrementally generates labels for unlabeled data from labeled nodes belonging to the same cluster.

Among the aforementioned methods, none tries to directly solve the unsupervised graph representation problem. On the contrary, our work aims to learn discriminative graph representations without label supervision. By utilizing cluster information as supervision signal, which reveals an intrinsic property of the graph, our approach produces high quality embeddings that benefit a variety of downstream tasks.

3 THE PROPOSED METHOD: CLEAR

In this section, we first formulate the problem of unsupervised graph representation learning. Then, we describe our proposed CLEAR method in detail with discussions on the time complexity and connections to information maximization.

3.1 Problem Formulation and Notations

Consider an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. We denote $\boldsymbol{X} \in \mathbb{R}^{N\times M}$ and $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ as the node feature matrix and the adjacency matrix, respectively, where $\boldsymbol{A}_{ij} = 1$ iff $(v_i, v_j) \in \mathcal{E}$ and $\boldsymbol{A}_{ij} = 0$ otherwise. The goal of unsupervised graph representation learning is to learn a low-dimensional representation $\boldsymbol{h}_i \in \mathbb{R}^D$ for each node $v_i \in \mathcal{V}$ with no access to ground-truth labels, where D is the dimension of node representations and $D \ll M$. We summarize all notations used throughout this article in Table 1 for better readability.

3.2 Graph Representation Learning by Cluster-aware Self-training

Typically, GNN models are trained using the classification objective in a supervised manner. In our unsupervised model where no ground-truth labels are given, we generate pseudo-labels to provide self-supervision by iteratively performing clustering on the embeddings. To be specific, in this unsupervised training phase, CLEAR alternates between optimizing parameters of the GNN model by predicting cluster labels and adjusting the cluster assignments of nodes.

Representation learning via graph convolutional networks. We use **Graph Convolutional Networks (GCNs)** [22] as the base model to learn node embeddings. GCN is a multilayer feedforward network in the graph domain that generates node embeddings by aggregating and transforming information from neighboring nodes. We define $\boldsymbol{H}^{(t)}$ as the output of the tth layer. The propagation rule of each layer can be defined as

$$\boldsymbol{H}^{(t)} = \sigma(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(t-1)}\boldsymbol{W}^{(t)}), \tag{1}$$

where $\widetilde{\boldsymbol{A}}$ is the normalized adjacency matrix of $\boldsymbol{A}$ with self-loops added, $\widetilde{\boldsymbol{D}}$ is the degree matrix for $\widetilde{\boldsymbol{A}}$ with entries $\widetilde{\boldsymbol{D}}_{ii} = \sum_{j=1}^{N} \tilde{\boldsymbol{A}}_{ij}$, σ denotes the activation function, e.g., $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$, and

Table 1. Notations Used Throughout this Article

Notation	Description
$\mathcal{G}$	input graph
$\mathcal{V}$	set of vertices
$\mathcal{E}$	set of edges
v_i	node with index i
$\boldsymbol{A}$	adjacency matrix of graph $\mathcal{G}$
$\widetilde{\boldsymbol{A}}$	adjacency matrix with self-loops added
$\widetilde{\boldsymbol{D}}$	degree matrix of $\widetilde{\boldsymbol{A}}$
$\boldsymbol{X}$	feature matrix
$\boldsymbol{x}_i$	feature of node v_i
$\boldsymbol{H}$	output embedding matrix
$\boldsymbol{h}_i$	embedding of node v_i
$\boldsymbol{W}^{(t)}$	trainable weight matrix in the tth layer GCN
$\boldsymbol{H}^{(t)}$	output embedding of the tth layer
y_i	cluster label of v_i
$\boldsymbol{C}$	cluster-assignment matrix
$\boldsymbol{c}_i$	cluster assignment of v_i
$\boldsymbol{\mu}_{y_i}$	embedding of the centroid that belongs to y_i
$\phi_p(\cdot)$	graph purity function
τ_a	threshold for adding edge in topology refining
τ_r	threshold for removing edge in topology refining

$\boldsymbol{W}^{(t)}$ is the trainable weight parameter of the tth layer. The input to GCN is the feature matrix, i.e., $\boldsymbol{H}^{(0)} = \boldsymbol{X}$. We employ an L-layer GCN to produce node embeddings for nodes, i.e., $\boldsymbol{H} = \boldsymbol{H}^{(L)}$.

Self-training with cluster assignments. In CLEAR, cluster labels are used as pseudo-supervision for model training. A cluster in the graph is a group of nodes that are closely correlated to each other in terms of both topology structures and features. A variety of clustering methods have been developed, and in our article, we choose kMeans due to its simplicity. Assume that the cluster labels of $v_i \in \mathcal{V}$ is denoted by $y_i \in \{1, 2, \ldots, K\}$, drawn from a space of K possible clusters. We denote the cluster assignments by $\boldsymbol{C} \in \{0, 1\}^{N \times K}$, where each row represents the cluster assignments of one node using one-hot encoding. Conventional kMeans aims to learn the centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ and cluster assignments $y_1, \ldots, y_N$ by optimizing

$$\min_{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{h}_i - \boldsymbol{\mu}_{y_i}\|. \tag{2}$$

Here, we slightly abuse the notation $\boldsymbol{\mu}_{y_i}$ being the centroid that has the label y_i.

To train the model without human annotations, we ask the model to predict cluster labels. To this end, we employ a **MultiLayer Perception (MLP)** network as the classifier. The MLP takes node embeddings $\boldsymbol{H}$ as input and predicts correct labels on top of these embeddings. For a typical classification problem with deterministic labels, we solve the following optimization problem:

$$\min -\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y} \log p(\boldsymbol{y} \mid v_i), \tag{3}$$

where $p(\boldsymbol{y} \mid v_i) = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{h}_i))$ is the prediction for node v_i.

Cluster reassignment with equipartitioning. We note that it is nontrivial to directly adopt kMeans for clustering-based self-supervised representation learning. This can be seen by a fact that when we optimize the embeddings $\boldsymbol{H}$ along with cluster assignments $\boldsymbol{y}$, a trivial solution can be obtained by mapping all nodes to the same point in the embedding space and treating them as a single cluster [6].

To address this problem, we propose to adjust the cluster assignments with a novel equipartition strategy. Formally, we first treat the cluster assignments $\boldsymbol{C}$ as probability distributions $c(\boldsymbol{y} \mid v_i)$. We further restrict each cluster to be *equally partitioned* [2]. With that requirement, Equation (3) can be rewritten as follows [2]:

$$\begin{aligned} \min \; & -\frac{1}{N}\sum_{i=1}^{N}\sum_{y=1}^{K} c(y \mid v_i)\log p(y \mid v_i), \\ \text{subject to } \; & \forall y : c(y \mid v_i) \in \{0,1\}, \\ & \sum_{i=1}^{N} c(y \mid v_i) = \frac{N}{K}, \end{aligned} \tag{4}$$

where $c(y \mid v_i) = \boldsymbol{c}_{iy}$ is the cluster assignment for node v_i. The first requirement guarantees that each node belongs to exactly one cluster and the second ensures that all N nodes are equally split into K clusters. Then, optimizing with respect to $c(\boldsymbol{y} \mid v_i)$ for all nodes is equivalent to reassigning cluster labels that satisfy the equipartition requirement.

Please kindly note that the equipartition requirement should be regarded as a *regularization* that aims to avoid the downgraded trivial solution, rather than a constraint that requires the input data to be in equally sized clusters. Moreover, considering we are agnostic to the number of classes in an unsupervised setting, we set the number of classes to be relatively larger to the real numbers (to which we refer as the *overclustering* strategy) following previous work [2, 6]. The overclustering strategy decomposes each data cluster into smaller sub-clusters and thereby allows these smaller sub-clusters to be in a similar size.

Due to the combinatorial nature of Equation (4), we resort to optimal transportation to solve it efficiently [2]. Specifically, we relax the cluster assignments $\boldsymbol{C}$ to be an element of the transportation polytope [12], given by

$$U(\boldsymbol{r},\boldsymbol{c}) := \left\{ \boldsymbol{C} \in \mathbb{R}_{+}^{N\times K} \,\middle|\, \boldsymbol{C}^{\top}\boldsymbol{1} = \boldsymbol{r}, \boldsymbol{C}\boldsymbol{1} = \boldsymbol{c} \right\}, \tag{5}$$

where $\boldsymbol{r} = \boldsymbol{1} \in \mathbb{R}^{N}$ and $\boldsymbol{c} = \frac{N}{K}\boldsymbol{1} \in \mathbb{R}^{K}$, which corresponds to our equipartition regularization.

Finally, the solution to Equation (4) is equivalent to solving the following problem (up to a constant shift $-\log N$):

$$\min_{\boldsymbol{C} \in U(\boldsymbol{r},\boldsymbol{c})} \langle \boldsymbol{C}, \boldsymbol{P} \rangle, \tag{6}$$

where $\boldsymbol{P} \in \mathbb{R}^{N\times K}$ with entries $\boldsymbol{p}_{iy} = -\log(p(y \mid v_i))$ is the cost matrix and $\langle \cdot, \cdot \rangle$ is the Frobenius dot-product between two matrices.

Note that although we relax $\boldsymbol{C}$ to be continuous, the solution to Equation (6) is guaranteed to be integral, which can be obtained in near-linear time using the Sinkhorn-Knopp matrix scaling algorithm [12]. Specifically, we can solve the problem by approximating the Sinkhorn projection of $e^{\mu \boldsymbol{P}}$ using the scaling algorithm, where μ is a hyper-parameter. In CLEAR, we employ a greedy version of the Sinkhorn algorithm, Greenkhorn [1], to approximate the solution, which is proven to outperform the original version significantly in practice. The cluster reassignment algorithm is given in Algorithm 1. For the details of the Greenkhorn algorithm, we refer the readers of interest to Appendix A.

ALGORITHM 1: CLEAR cluster reassignment with equiparitioning

```
1 $\boldsymbol{P} \leftarrow -\log(\text{softmax}(\text{MLP}(\boldsymbol{H})))$
2 $\boldsymbol{M} \leftarrow e^{-\eta \boldsymbol{P}}$
3 $\boldsymbol{r} \leftarrow \mathbf{1}$
4 $\boldsymbol{c} \leftarrow \frac{N}{K}\mathbf{1}$
5 $\boldsymbol{M} \leftarrow \texttt{Greenkhorn}(\boldsymbol{M}, \boldsymbol{r}, \boldsymbol{c})$
6 $\boldsymbol{C} \leftarrow \boldsymbol{M}$
```

3.3 Topology Refining

After obtaining cluster assignments, we further refine the graph topology by strengthening intra-class edges and reducing inter-class connections. Specifically, given cluster assignments $\boldsymbol{C}$, for each edge (v_i, v_j), we remove it if the probability that v_i and v_j fall into the same cluster is less than a threshold τ_r, i.e., $\boldsymbol{c}_i^\top \boldsymbol{c}_j < \tau_r$. Additionally, for each node pair (v_i, v_j), if the probability that v_i and v_j belong to the same cluster is greater than another threshold τ_a, we add the edge (v_i, v_j) to the graph.

Note that in each iteration, we refine the graph topology based on the original graph instead of the previously refined graph, since informative edges might be accidentally removed at the early stage of the training procedure. Additionally, when adding edges, we consider (v_i, v_j) as candidates only when they connect nodes belonging to the same cluster, i.e., $\text{argmax}_k\, c_{ik} = \text{argmax}_l\, c_{jl}$ to reduce excessive computation.

The topology refining procedure is designed to increase the *purity* of the whole graph $\phi_{\mathcal{G}}$, which is defined as the probability of an edge in $\mathcal{G}$ connecting nodes from the same cluster. Formally, we define graph purity as

$$\phi_p(\mathcal{G}) = \frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} P(y_i = y_j) = \frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} \boldsymbol{c}_i^\top \boldsymbol{c}_j. \tag{7}$$

We can see that topology refining can increase graph purity when the threshold is less than or equal to the current purity, i.e., $\tau \leq \phi_p(\mathcal{G})$. In practice, τ is chosen dynamically with $\tau = \frac{1}{2}\phi_p(\mathcal{G})$.

Our motivation is that learning embeddings for a graph with higher purity will better preserve cluster structures. Considering that embeddings of neighboring nodes are smoothed in graph convolutions, embeddings of nodes belonging to different clusters will become similar due to inter-cluster edges, resulting in indistinguishable node embeddings. We further illustrate this idea through visualization. On the Karate club dataset [49], we conduct topology refining to increase graph purity and add noise edges to decrease graph purity. The learned embeddings are shown in Figure 2. We can see that the modified graph (Figure 2(b)) with higher purity produces embeddings with well-separated clusters, while the clusters are indistinguishable in the graph (Figure 2(c)) with lower purity.

3.4 Model Training

To train the proposed CLEAR model, we first initialize the parameters of GCN by training it with a reconstruction loss [21], which forces the node embeddings to preserve pairwise similarity. Then, we initialize the cluster assignments by employing kMeans on the node embeddings.

After initialization, the node embeddings will be improved in further steps using cluster-aware self-training. Specifically, we iteratively update model weights in three stages, namely, graph representation learning, cluster reassigning, and topology refining. At first, when we perform graph representation learning by solving Equation (3), we fix the cluster assignments $\boldsymbol{C}$. Then, with

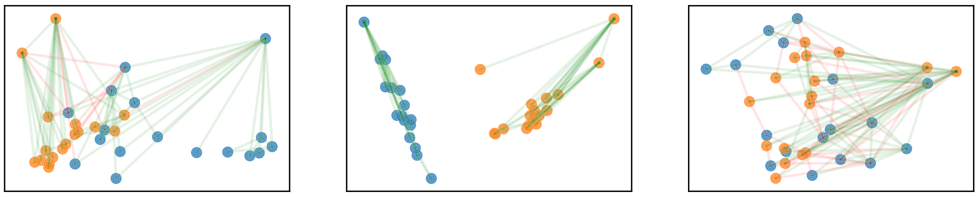


Fig. 2. Visualization of node embeddings with different graph purity on the Karate club dataset. Node colors indicate classes. Green lines indicate inter-class edges while red lines indicating intra-class edges.

node representations $\boldsymbol{H}$ fixed, we adjust the cluster assignments by solving Equation (6) using the Greenkhorn algorithm. Following Asano et al. [2], to stabilize training, we distribute cluster reassignment and topology refining steps throughout the whole training process. We denote $\mathcal{S}$ as the set of epochs where cluster assignments will be adjusted. $\mathcal{S}$ can be chosen freely as long as cluster reassignment is performed at proper intervals. In our implementation, we set updating epoch $s_i \in \mathcal{S}$ as $s_i = (E-W)\frac{i}{U+1} + W,\ i = 1, 2, \ldots, U$, where U is the total number of reassignment steps throughout training, W is the number of warm-up epochs where cluster reassignment will not be performed, and E is the total number of training epochs. The whole training algorithm is summarized in Algorithm 2.

ALGORITHM 2: CLEAR training algorithm

```
1  $\mathcal{E}_0 \leftarrow \mathcal{E}$;
2  Initialize the weights of GCN by reconstruction objectives;
3  Generate initial embedding $\boldsymbol{H}$ using GCN with Equation (1);
4  Initialize initial cluster assignments using $k$Means;
5  for $epoch \leftarrow 1, 2, \ldots$ do
6      Update weights of GCN and MLP by solving Problem (3);
7      if $epoch \in \mathcal{S}$ then
8          Adjust cluster assignments by solving Problem (6) using the Greenkhorn algorithm;
           // Perform topology refining
9          $\tau_r \leftarrow \frac{1}{2}\phi_p(\mathcal{G})$;
10         $\mathcal{E} \leftarrow \{(v_i, v_j) \mid (v_i, v_j) \in \mathcal{E}_0, \mathbf{c}_i^\top \mathbf{c}_j \geq \tau_r\}$;      // Remove inter-class edges
11         for $y \leftarrow 1, 2, \ldots, K$ do
12             $\mathcal{V}_y \leftarrow \{v_i \in \mathcal{V} \mid \mathrm{argmax}_l\ c_{il} = y\}$;
13             $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v_i, v_j) \in \mathcal{V}_y \times \mathcal{V}_y, \mathbf{c}_i^\top \mathbf{c}_j > \tau_a\}$;      // Add intra-class edges
```

3.5 Discussions

3.5.1 Time Complexity Analysis. The time complexity of updating cluster assignments using the Greenkhorn algorithm is $O(NK)$. In the topology refining procedure, we compute the correlation between each connected node pair and delete edges with low correlation, which results in the time complexity of $O(|\mathcal{E}|(K+1))$. Note that in the real world, graphs are usually sparse, i.e., $|\mathcal{E}| \ll N^2$. Therefore, the overall time complexity of each cluster updating iteration is $O(NK + (K+1)|\mathcal{E}|)$.

3.5.2 Comparison with Graph Structure Learning. We note that the proposed topology refining scheme is conceptually similar to graph structure learning [15, 20], which proposes to simultaneously refine the given graph topology and learn the node representations to mitigate the noise in graph structures. In particular, both these two models propose to quantify the connection strengths of every N^2 pairs of nodes using cosine similarity. Though they further employ kNN thresholding on the similarity graph to create a sparse graph, which mitigates the computational burden for GNNs, they have the potential problem of inflexibility in setting a fixed k value (e.g., $k = 20$ for all datasets, as done in Jin et al. [20]). Our CLEAR model, on the contrary, preserves the sparsity, an important graph prior of real-world graphs, by thresholding the purity of edges. The threshold values controlling the number of modified edges could be dynamically adjusted according to the cluster assignments as the training progresses. Moreover, most existing graph structure learning models target at supervised settings [9, 55], while our approach considers unsupervised learning, which is conceptually harder.

3.5.3 Connection with the Information Maximization Principle. Finally, we interpret the proposed CLEAR approach from information theoretic perspectives. Different from contrastive learning methods [41, 53] that maximize the mutual information between input graph and learned representation, our approach maximizes the lower bound of mutual information between the learnt node embeddings and the cluster labels [2]. As shown in experiments, the cluster labels produced by CLEAR have strong a correlation to the ground-truth classes of downstream node classification tasks, which explains why CLEAR can produce high quality node embeddings that improve the performance of downstream tasks.

4 EXPERIMENTS

In this section, we present the results and analysis of empirical evaluation of our proposed method. These experiments are conducted to answer the following four research questions:

- **RQ1**: How does the proposed method compare with existing baselines in traditional graph mining tasks?
- **RQ2**: How does the cluster-aware topology refining mechanism help improve the quality of node embeddings? How does adding intra-class edges and removing inter-class edges independently contribute to improve the quality of node embeddings?
- **RQ3**: Does the soft topology refining scheme outperform the proposed hard refining scheme?
- **RQ4**: How do key hyper-parameters affect model performance?

To answer RQ1, in the experiments, we extensively compare the proposed CLEAR for two graph mining tasks, node classification and node clustering. Then, we conduct detailed ablation studies on the cluster-aware topology refining procedure to answer RQ2. Following the ablation study of the cluster-aware topology refining module, we further compare the proposed hard refining scheme with its "soft" variant to answer RQ3. After that, to answer RQ4, we perform parameter sensitivity analysis on several key hyper-parameters of the model. Finally, we provide visualization of node embeddings to give qualitative results of our proposed methods.

4.1 Experimental Setup

Datasets. For a comprehensive comparison with state-of-the-art methods, we evaluate our model using four widely used datasets: among them, three are citation networks Cora, Citeseer, and Pubmed, for predicting article subject categories [34, 48], and the other co-authorship network **Coauthor-CS (CS)** [35] is for predicting the research fields of the authors. In the three citation datasets, graphs are constructed from computer science papers of various subjects. Specifically, nodes correspond to articles and undirected edges correspond to citation links. Each node has

Table 2. Statistics of Datasets Used Throughout the Experiments

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
CS	18,333	81,894	6,805	15

a sparse 0/1 bag-of-words feature and a corresponding class label. In the co-authorship dataset, nodes represent authors and they are connected if the two authors have co-authored a paper. Each node has a bag-of-words feature representing keywords for each author's papers. The statistics of these datasets is summarized in Table 2.

Experimental configurations. We train the model using the Adam optimizer with a learning rate of 0.01. Initially, we train the GCN model with the reconstruction loss for 500 epochs on Cora and PubMed, 250 epochs on CiteSeer, and 200 epochs on CS. Following that, in the self-supervised learning phase, we train the whole model for 15, 60, 50, and 800 epochs on Cora, Citeseer, Pubmed, and CS, respectively. On all the datasets, the optimal transportation solver is run for a fixed number epochs E_{ot} and with the same hyper-parameter μ. Prior to training, we initialize the weight of the encoders by training it with reconstruction loss. Technically, we optimize the loss by negative sampling. It is also possible to initialize the parameters in GCN with a variant of reconstruction loss proposed in VGAE [21]. In our experiments, we initialize our model with the standard reconstruction loss on Cora, Pubmed, and CS, while on Citeseer, we use the variational reconstruction loss.

Hyper-parameter settings. We set the dimension of the node embeddings to 64, the weight decay to 0.0008 in all datasets. For the number of clusters, to avoid trivial solution [6], we set the number of clusters to be around twice the number of ground-truth classes. Specifically, we set the number of clusters to 10, 11, 5, and 20 on Cora, Citeseer, Pubmed, and CS, respectively. For the set of epochs where we perform cluster reassignment, W is set to 10, 80, 20 and, 50 in Cora, Citeseer, Pubmed, and CS, respectively; U is set to 7, 7, 6, and 4 in the four datasets, respectively. Besides, for the optimal transportation solver, E_{ot} is set to 1,000 and μ is set to 20.

4.2 Node Clustering (RQ1)

To demonstrate the performance of the proposed approach, we first evaluate it on an unsupervised task—we conduct node clustering algorithms on top of the learned node embeddings. In this experiment, we employ kMeans as the clustering method. We run the algorithm for ten (10) times and report the averaged performance as well as standard deviation. We choose two widely used metrics accuracy and **Normalized Mutual Information (NMI)** in reporting the performance.

Baselines. For a comprehensive comparison, we compare our methods against various unsupervised methods. These methods can be grouped into four categories.

- Traditional methods that only make use of input features. We run two methods kMeans and **Spectral Clustering (SC)** directly on the input features, which means that no graph structures are used at all.
- Network embedding methods that use graph structures only.
  - DeepWalk [31] is a representative random-walk-based method, which generates node embeddings by sampling random walks on graphs and feeds them into language models.
  - DNGR [5] adopts a random surfing model to capture the graph structures. These methods only utilize graph structural information and neglect the input features.

Table 3. Performance of Node Clustering on the Four Datasets in Terms of Accuracy and Normalized Mutual Information (NMI)

Method	Cora		Citeseer		Pubmed		CS	
	Accuracy	NMI	Accuracy	NMI	Accuracy	NMI	Accuracy	NMI
*k*Means	34.65±0.83	25.42±0.83	38.49±0.99	30.47±0.85	33.37±1.39	57.35±0.54	56.70±2.86	50.23±2.05
SC	36.26±1.15	25.64±1.38	46.23±1.27	33.70±0.76	59.91±1.46	58.61±0.92	52.69±2.45	54.57±2.30
DeepWalk	46.74±1.38	38.06±0.62	36.15±0.99	26.70±0.71	61.86±1.26	47.06±0.71	55.78±0.25	39.17±0.22
DNGR	49.24±1.18	37.29±1.36	32.59±1.15	44.19±1.07	45.35±1.15	17.90±0.89	42.12±0.23	55.21±0.18
GAE	53.25±0.87	41.97±1.26	41.26±0.95	29.13±0.87	64.08±1.36	49.26±1.09	54.57±2.01	58.73±2.48
VGAE	55.95±0.74	41.50±0.76	44.38±1.00	31.88±0.94	65.48±0.64	50.95±1.25	57.94±1.78	60.10±2.12
MGAE	63.43±1.35	38.01±1.33	63.56±1.03	39.49±0.64	43.88±0.88	41.98±0.79	61.91±2.03	62.54±1.01
ARGE	64.00±1.01	61.90±1.49	57.30±1.34	54.60±0.65	59.12±0.68	58.41±0.99	62.12±2.10	63.11±0.98
ARVGE	63.80±1.05	62.70±0.63	54.40±1.20	52.90±0.58	58.22±0.82	23.04±1.07	61.18±1.99	62.75±1.25
DANE	70.27±1.25	68.93±0.76	47.97±1.44	45.28±1.43	69.42±1.00	65.10±0.81	62.13±2.03	63.72±1.02
AGC	68.92±0.87	65.61±1.04	67.00±0.23	**62.48±0.52**	69.78±1.45	68.72±1.36	63.23±1.67	62.16±1.58
AGE	74.74±1.22	72.34±0.83	58.58±0.57	55.91±0.23	49.63±0.81	34.58±0.35	60.57±1.12	54.23±1.05
DGI	65.28±0.73	58.90±1.22	60.37±1.31	55.63±0.78	51.22±1.29	46.73±0.76	63.00±1.78	58.55±1.21
GMI	67.10±0.91	65.75±2.01	59.82±0.11	56.47±0.82	OOM	OOM	OOM	OOM
CLEAR	**77.37±2.52**	**75.24±2.88**	**67.30±0.55**	62.20±0.58	**71.03±0.13**	**70.72±0.13**	**69.01±1.32**	**63.82±1.02**

- Attributed graph clustering models that use both structures and attributes.
  - Graph autoencoders (GAE [21], VGAE [21], and MGAE [43]) use GCN [22] as the encoder and enforce the model to reconstruct graph structures specified by a graph proximity matrix (e.g., the adjacency matrix that represents one-order proximities).
  - DANE [14] employs two autoencoders to preserve proximities for both graph structures and node attributes.
  - AGC [52] directly applies graph convolutions to the input features and runs spectral clustering on the obtained embeddings.
- Deep graph embedding models that use both structures and attributes.
  - AGE [10] applies a Laplacian smoothing filter to alleviate the high-frequency noise in node features and employs an adaptive encoder for better node embeddings.
  - DGI [41] applies contrastive learning techniques that aims to maximize mutual information between global graph embeddings and local node embeddings.
  - GMI [30] proposes graphical mutual information, which measures the correlation between the graph and the embeddings from both structural and feature aspects.

Results and analysis. The performance is summarized in Table 3 with the highest performance highlighted in bold. We report the performance of baselines in accordance with their original papers [14, 52]. From the table, it is evident that our proposed CLEAR surpasses other baseline methods in terms of accuracy on all four datasets. It is worth mentioning that we exceed the existing state-of-the-art model by a large margin of over 7% in terms of absolute accuracy improvements on Cora.

The results can be analyzed in three aspects. First, we observe that traditional algorithms such as *k*Means and spectral clustering, which simply rely on node attributes perform poorly on graph data. Second, conventional network embedding methods outperform traditional clustering methods, but their performance is still inferior to that of attributed graph clustering methods. This demonstrates the power of modern deep learning techniques on graphs that can better leverage both graph structures and node attributes. Last, it is observed that our proposed method outperforms attributed graph clustering baselines by considerable margins. Previous graph clustering methods merely perform node representation learning on the node level, while

Table 4. Performance of Node Classification on the Four Datasets in Terms of Micro-F1 (Mi-F1) and Macro-F1 (Ma-F1)

Method	Cora		Citeseer		Pubmed		CS	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
DeepWalk	$75.68_{\pm 0.20}$	$74.98_{\pm 0.73}$	$50.52_{\pm 0.72}$	$46.45_{\pm 0.69}$	$80.47_{\pm 0.39}$	$78.73_{\pm 0.61}$	$74.12_{\pm 0.01}$	$70.42_{\pm 0.04}$
node2vec	$74.77_{\pm 0.65}$	$72.56_{\pm 0.29}$	$52.33_{\pm 0.07}$	$48.32_{\pm 0.12}$	$80.27_{\pm 0.41}$	$78.49_{\pm 0.31}$	$75.69_{\pm 0.20}$	$72.54_{\pm 0.11}$
GraRep	$75.68_{\pm 0.74}$	$74.41_{\pm 0.23}$	$48.17_{\pm 0.53}$	$45.89_{\pm 0.01}$	$79.51_{\pm 0.42}$	$77.85_{\pm 0.24}$	$72.12_{\pm 0.21}$	$71.95_{\pm 0.43}$
ANE	$72.03_{\pm 0.24}$	$71.50_{\pm 0.72}$	$58.77_{\pm 0.15}$	$54.51_{\pm 0.47}$	$79.77_{\pm 0.44}$	$78.75_{\pm 0.10}$	$81.21_{\pm 0.76}$	$74.55_{\pm 0.71}$
GAE	$76.91_{\pm 0.42}$	$75.73_{\pm 0.14}$	$60.58_{\pm 0.25}$	$55.32_{\pm 0.11}$	$82.85_{\pm 0.65}$	$83.28_{\pm 0.28}$	$82.15_{\pm 1.12}$	$75.67_{\pm 0.88}$
VGAE	$78.88_{\pm 0.58}$	$77.36_{\pm 0.21}$	$61.15_{\pm 0.38}$	$56.62_{\pm 0.29}$	$82.99_{\pm 0.28}$	$82.40_{\pm 0.02}$	$83.25_{\pm 1.06}$	$\mathbf{78.21_{\pm 0.25}}$
DANE	$78.67_{\pm 0.74}$	$77.48_{\pm 0.70}$	$64.44_{\pm 0.20}$	$60.43_{\pm 0.18}$	$\mathbf{86.08_{\pm 0.67}}$	$\mathbf{85.79_{\pm 0.15}}$	$83.11_{\pm 0.45}$	$77.01_{\pm 0.21}$
AGE	$74.79_{\pm 0.73}$	$73.09_{\pm 0.73}$	$63.47_{\pm 0.50}$	$58.85_{\pm 0.02}$	$81.69_{\pm 0.74}$	$81.32_{\pm 0.44}$	$75.68_{\pm 0.89}$	$75.12_{\pm 2.01}$
DGI	$82.53_{\pm 0.20}$	$81.09_{\pm 0.35}$	$68.76_{\pm 0.23}$	$63.58_{\pm 0.73}$	$85.98_{\pm 0.59}$	$85.66_{\pm 0.07}$	$84.35_{\pm 1.28}$	$67.13_{\pm 2.14}$
GMI	$82.19_{\pm 0.13}$	$80.84_{\pm 0.48}$	$69.44_{\pm 0.53}$	$\mathbf{63.81_{\pm 0.12}}$	OOM	OOM	OOM	OOM
CLEAR	$\mathbf{82.56_{\pm 0.32}}$	$\mathbf{81.16_{\pm 0.64}}$	$\mathbf{69.56_{\pm 0.72}}$	$61.59_{\pm 0.24}$	$85.76_{\pm 0.52}$	$83.49_{\pm 0.02}$	$\mathbf{88.84_{\pm 1.01}}$	$75.92_{\pm 0.98}$

our method exploits underlying cluster structures in graphs to guide representation learning. Additionally, we utilize cluster information to reduce the impact of noisy inter-class edges, which further improves the quality of embeddings. The improvements show that our proposed CLEAR helps generate embeddings that better preserve cluster structures.

Note that the proposed CLEAR is slightly inferior to AGC on Citeseer in terms of NMI score. However, CLEAR still outperforms it in terms of accuracy and on other datasets by a considerable margin. In all, these results verify the effectiveness of our proposed CLEAR.

4.3 Node Classification (RQ1)

We further evaluate the quality of embeddings generated by CLEAR on a supervised task—node classification. After training the model, we conduct node classification on the learned node representations. For a fair comparison, we closely follow the same experimental settings as Gao and Huang [14]. Specifically, we train a linear logistic regression classifier with ℓ_2 regularization. For training/test set splitting, we randomly select 10% nodes as the training set and the remaining nodes are left for the test set. Then, we use cross-validation to select the best model. We report the performance on the test set in terms of two widely used metrics, **Micro-averaged F1-score (Mi-F1)** and **Macro-averaged F1-score (Ma-F1)**. As with the previous experiment, we report the averaged performance and standard deviation of ten (10) runs.

Baselines. In node classification, we include two sets of baseline algorithms: (1) traditional network embedding methods, which only leverage graph structures and ignore node attributes, and (2) attributed graph representation learning methods, which use both graph structures and node attributes. The former category includes representative random-walk-based methods DeepWalk [31], node2vec [17], and GraRep [4]. The latter one includes attributed network embedding methods ANE [19] and DANE [14], and graph neural networks GAE, VGAE [21], DGI [41], AGE [10], and GMI [30].

Results and analysis. We report the performance in Table 4, with the best performance highlighted in boldface. The baseline performance is reported as in their original papers [14]. In general, the results confirm the effectiveness of the proposed method. As shown in the table, the proposed CLEAR performs best on the Cora and Citeseer datasets, compared with state-of-the-art baselines and shows competitive performance on Pubmed and CS, compared with other graph representation learning methods.

As with the conclusions drawn from the experiment of node clustering, traditional network embedding methods such as DeepWalk and node2vec perform worse than deep-learning-based graph representation learning methods, which highlights the importance of incorporating node attributes when training the model. Instead of merely leveraging graph structures, GNNs combine information of graph topology and node attributive information, resulting in better node embeddings. Our proposed method further utilizes cluster information in self-training and refines graph topology by removing inter-class edges that potentially hinder the model from preserving cluster structures in the embedding space. The proposed method produces better node embeddings, so that it achieves significant improvement over existing GNN-based methods.

Note that while DANE is a strong baseline on Pubmed, our proposed CLEAR still outperforms it in terms of accuracy and Macro-F1 score on the other two datasets. We observe that the NMI between cluster labels and ground-truth labels on Pubmed is the lowest among the four datasets, which can help explain the slightly inferior performance of CLEAR on Pubmed. Through the topology refining procedure that is based on cluster labels, our proposed CLEAR may accidentally remove informative edges, which results in performance loss. Recent work DGI and GMI marry the power of contrastive learning into graph representation learning. However, they only optimize node representations in the latent space, which neglect cluster structures that align with the intrinsic properties of the graphs. On the contrary, our proposed CLEAR significantly outperforms them on graph clustering tasks and achieves comparable performance with them in node classification. This can be attributed to our novel approach of leveraging cluster-aware self-training and refining graph topology, which not only enhances cluster structures but also provides more discriminative representations that can benefit node classification tasks. By refining the graph topology and strengthening intra-class edges, our method can generate more informative node embeddings, ultimately leading to improved performance in both tasks.

4.4 Ablation Studies (RQ2)

To further validate the proposed cluster-aware topology refining procedure and justify our architectural design choice, we conduct ablation studies by removing specific components in the topology refining module. Then, we conduct node clustering using the same setting described in previous sections.

4.4.1 Impact of the Proposed Topology Refining Module. First, to further validate the proposed cluster-aware topology refining module, we conduct ablation studies by removing this module. We term the resulting model as CLEAR− hereafter. To compare the performance of the original CLEAR and CLEAR−, we conduct node clustering using the same setting described in previous sections, where the performance is reported in Figure 3. From the figure, we observe that the topology refining module improves the performance of CLEAR− on node clustering by considerable margins in terms of three evaluation metrics, i.e., Micro-F1, Macro-F1, and NMI, which once again verifies its effectiveness. Moreover, we calculate graph purity against ground-truth classes to reflect the modification to topology of the original graph. From the figure, it is apparent that the proposed topology refining procedure is able to alleviate the impact of noisy inter-class edges and further better preserve cluster structures.

4.4.2 Impact of Different Schemes of Topology Refining. To further validate the proposed topology refining schemes, we perform ablation studies by comparing the model performance with different components of the refining module enabled. We report the clustering accuracy of the following three variants: (1) CLEAR-Add, which only adds intra-class edges, (2) CLEAR-Remove, which only removes inter-class connections, and (3) CLEAR-Hybrid, which is our proposed module with both schemes enabled. The performance of the three variants is presented in Figure 4.

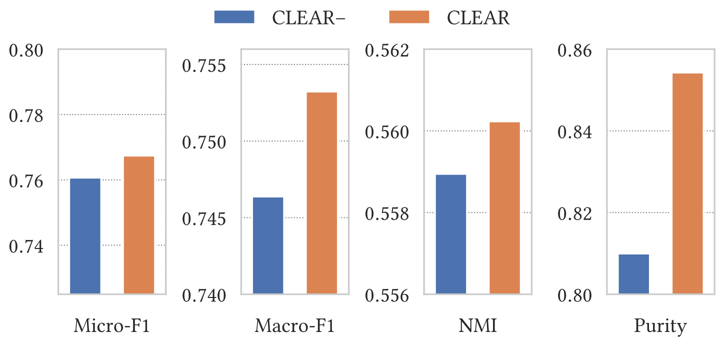


Fig. 3. Performance of node clustering and graph purity on the Cora dataset with and without the topology refining module.

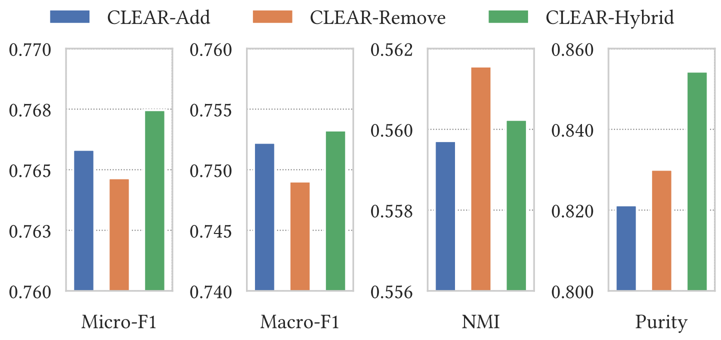


Fig. 4. Performance of node clustering on Cora with different schemes enabled in the topology refining module.

From the figure, it is clear that enabling both schemes benefits model performance in terms of Micro-F1, Macro-F1, and Purity. However, we note that, CLEAR-Remove outperforms CLEAR-Hybrid in terms of NMI slightly. This may be explained from the fact that CLEAR-Hybrid introduces some noisy edges when adding intra-class edges, as the model makes wrong prediction about the ground-truth classes. In summary, the proposed hybrid scheme generally outperforms better, compared with CLEAR-Add and CLEAR-Remove, which justifies our design choice of the proposed topology refining module.

4.5 Discussions of Hard and Soft Topology Refining Schemes (RQ3)

Following the ablation study of the proposed topology refining scheme, we further conduct additional experiments using a soft topology refining scheme. For the proposed topology refining scheme, we regard the edge deletion as a "hard" operation, where the intra-class edges will be *completely* removed for node representation learning. Considering the discrepancy between our model prediction about clusters and ground-truth labels, contrary to hard removal, we may consider an alternative "soft" scheme, where one intra-class edge are reassigned probabilities that express the strength of connection. In this experiment, for each edge (v_i, v_j), we reassign each intra-class edge with a weight $\boldsymbol{A}'_{ij} = \boldsymbol{c}_i^\top \boldsymbol{c}_j$; other edge weights are not modified. Since in the original graph, we represent each edge by $\boldsymbol{A}_{ij} = 1$, our soft modification $\boldsymbol{A}'_{ij} < 1$, which is able to reduce the connection of intra-class nodes.

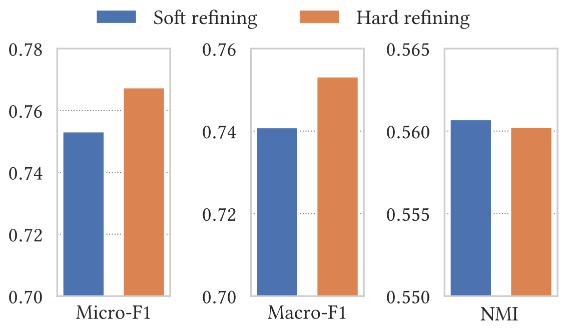


Fig. 5. Performance of node clustering on Cora with hard and soft topology refining schemes.

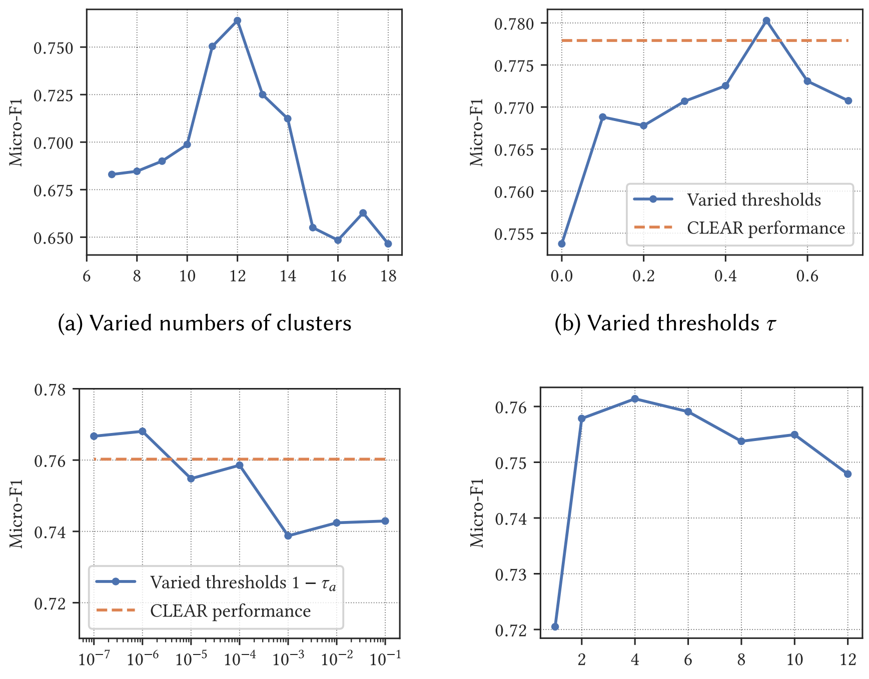
The results are shown in Figure 5. We observe that our proposed hard scheme evidently outperforms its soft variant in terms of Micro-F1 and Macro-F1, and performs slightly lower in terms of NMI. The result provides the rationale of using a hard removal scheme. The reason why the soft topology refining scheme performs worse than the hard scheme may be explained from the fact that via the soft removal scheme, there are still many inter-class edges remained, which deteriorate the quality of node embeddings.

4.6 Parameter Sensitivity Analysis (RQ4)

In this section, we examine the impact of four key parameters in our model, i.e., the cluster size, the two thresholds for topology refining, and the interval of reassigning clusters. We conduct node clustering on the Cora dataset by varying these four parameters independently. While one hyper-parameter studied in the sensitivity analysis is changed, the other hyper-parameters remain the same as previously described.

4.6.1 Impact of the Cluster Size K. To investigate the influence of cluster numbers on our model, we run CLEAR by varying the number of clusters from 7 to 18. The results on Cora with different numbers of clusters are plotted in Figure 6(a). From the figure, we observe that the model performance first benefits from the increase of cluster numbers, but soon the performance decreases. This indicates that the over-clustering strategy does boost the performance of CLEAR, since it can alleviate inconsistency between the clusters discovered in self-training and real-world datasets. Specifically, when we enforce each cluster to be equally balanced, classes in real-world graphs usually vary greatly in their sizes. However, dividing nodes into too many clusters will in turn deteriorate the performance, since the proposed cluster-aware topology refining mechanism will unnecessarily remove informative inter-cluster edges.

4.6.2 Impact of the Threshold τ_r in Topology Refining. To further investigate the impact of τ_r on the model performance, we run CLEAR by setting τ_r from 0 to 0.7, with a constant interval of 0.1. From the results in Figure 6(b), we observe that clustering accuracy is first boosted from the increase of τ_r, then it stops increasing and decreases. This can be explained that a higher threshold may result in the accidental removal of possibly useful intra-cluster edges. The observation is consistent with our previous study on the impact of cluster size. Moreover, we note that the performance achieved with our proposed scheme that selecting τ_r dynamically is close to the highest performance when directly fixing τ_r to a certain value, which prove the validity of the dynamic selection scheme. Since in the real world, ground-truth labels may be inaccessible, it is infeasible to fix τ_r to be the best value based on performance.



(a) Varied numbers of clusters

(b) Varied thresholds τ

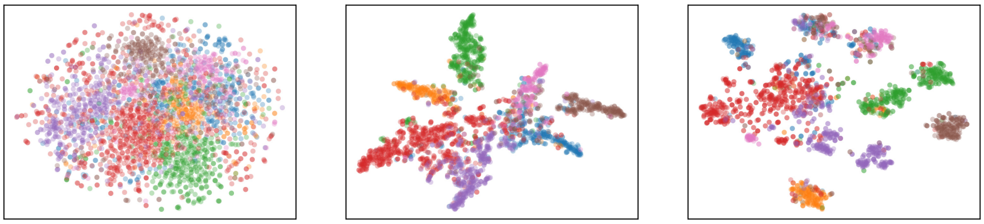
(c) Varied thresholds τ_a

(d) Varied intervals of updating clusters

Fig. 6. Sensitivity analysis under different cluster sizes, thresholds τ, τ_a, and cluster updating intervals U, in terms of node clustering accuracy on the Cora dataset.

4.6.3 Impact of the Threshold τ_a in Topology Refining. To investigate the impact of τ_a on the performance of CLEAR, we run CLEAR by setting τ_a to different values and report the clustering accuracy on Cora. Due to the high sparse nature of edges in the original graph, we set $(1 - \tau_a)$ from 10^{-1} to 10^{-7} by taking exponential scales. We report the performance under different τ_a in Figure 6(c). From the figure, we can see that model performance first benefits from the increase of $(1 - \tau_a)$, indicating adding more edges, but soon the accuracy decreases. The performance gain when $(1-\tau_a)$ is set to 10^{-7} or 10^{-6} verifies the effectiveness of our proposed adding edge scheme for topology refining. While the model benefits from the adding edge scheme initially, the performance becomes inferior to the base model when $(1 - \tau_a)$ is large. This can be attributed to the fact that a large $(1 - \tau_a)$ will result in a dense neighborhood, which leads to the over-smoothing problem and tends to bring noise into node representations.

4.6.4 Impact of the Cluster Reassignment Interval U. To investigate the impact of the interval of cluster reassignment U, we run CLEAR in varied U values with node clustering accuracies on Cora reported in Figure 6(d). From the results, we can make observations such that model performance first benefits from the increase of U, but soon the accuracy levels off. The performance gain when U increases can be explained by the fact that reassignments can result in more reliable pseudo-labels, which can better guide the learning of our model. This is consistent with our motivation that the learning of model and the pseudo-labels can benefit from the progress of each other and jointly boost the quality of learned representations. However, adjusting cluster assignments too frequently may bring instability to model training and thus leads to inferior model performance.



(a) Raw features (b) Embeddings by GCN (c) Embeddings by CLEAR

Fig. 7. Visualization of raw features and embeddings learned with CLEAR on the Cora dataset. T-SNE [39] is applied to project features and embeddings into two-dimensional spaces. Each node is colored with its corresponding ground-truth class label.

4.7 Visualizing Node Embeddings

Finally, we provide qualitative results by visualizing the learnt embeddings. Specifically, we leverage t-SNE [39] to project the embeddings onto a two-dimensional space and plot them with colors corresponding to the class of each node. The node embeddings are extracted from the penultimate layer of a CLEAR model that is pre-trained on the Cora dataset. For comparison, we also present the visualization of the raw node features in Figure 7(a) as well as embeddings trained with a supervised GCN in Figure 7(b). From the figures, we observe that the representations learned with CLEAR exhibit discernible clusters in the projected two-dimensional space. Note that node colors correspond to seven *ground-truth* node classes, which shows that the produced embeddings are highly discriminative across seven classes in Cora. Compared to the supervised counterpart, the embedding space learned by CLEAR is more well-clustered, verifying that CLEAR is able to extract and preserve essential information of the graphs.

5 CONCLUSION AND FUTURE WORK

In this article, we have developed a novel cluster-aware self-training and refining model (CLEAR) for unsupervised graph representation learning, in which we train GNNs without human annotations. Specifically, CLEAR performs clustering on the node embeddings and updates GNN parameters by predicting cluster assignments of nodes. Then, we propose an equipartition strategy to reassign cluster assignments to avoid downgraded solution. Moreover, we leverage a novel graph topology refining scheme that strengthens intra-class edges and isolates nodes from different clusters based on cluster labels to improve node embedding quality. Comprehensive experiments on two benchmark tasks using real-world datasets have been conducted. The results demonstrate the superior performance of our proposed CLEAR over state-of-the-art baselines.

The study of unsupervised techniques for graph representation learning generally remains widely open. It is seen from this work that accurately predicting the cluster labels is crucial for successfully deploying the model. In our future work, we plan to further investigate combining other self-supervised methods, e.g., contrastive learning methods [53, 54], to better model the latent space of node embeddings and thereby improve the quality of node embeddings. Another possible direction for future work could be to integrate a hierarchical clustering technique, which would allow for a more refined representation of the internal structures. By considering sub-clusters at different levels of granularity, we may be able to better capture the fine-grained relationships between nodes while still maintaining the benefits of our cluster-aware self-training and topology refining approach.

APPENDIX

A DETAILS OF THE GREENKHORN ALGORITHM

The Greenkhorn algorithm [1] aims to solve the matrix scaling problem: given a non-negative matrix $\boldsymbol{A} \in \mathbb{R}_+^{N \times K}$, the goal is to find two vectors $\boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{y} \in \mathbb{R}^K$, such that the row sum and the column sum in $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{x})\boldsymbol{A}\,\mathrm{diag}(\boldsymbol{y})$ satisfy that

$$r(\boldsymbol{M}) = \boldsymbol{r}, \tag{8}$$

$$c(\boldsymbol{M}) = \boldsymbol{c}, \tag{9}$$

where $r(\boldsymbol{M}) = \boldsymbol{M}\mathbf{1}$, $c(\boldsymbol{M}) = \boldsymbol{M}^\top\mathbf{1}$, $\boldsymbol{r}$ and $\boldsymbol{c}$ is the required row/column sum.

The vanilla Sinkhorn-Knopp algorithm approximates the solution by alternatively normalizing the row and column sum of the matrix. Instead of normalizing all rows/columns at each iteration, Greenkhorn greedily selects one row or column to update according to a distance function, $\rho : \mathbb{R}^+ \times \mathbb{R}^+ \to [0, +\infty]$, which is defined as

$$\rho(a, b) = b - a + a \log \frac{a}{b}. \tag{10}$$

The details of the Greenkhorn algorithm are given in Algorithm 3, where E_{ot} is the number of iterations.

ALGORITHM 3: The Greenkhorn algorithm

```
1  function Greenkhorn($\boldsymbol{A}, \boldsymbol{r}, \boldsymbol{c}$):
2      $\boldsymbol{P} \leftarrow -\log(\mathrm{softmax}(\mathrm{MLP}(\boldsymbol{H})))$
3      $\boldsymbol{M}^{(0)} \leftarrow \boldsymbol{A}$
4      $\boldsymbol{x} \leftarrow \mathbf{0}, \boldsymbol{y} \leftarrow \mathbf{0}$
5      $\boldsymbol{M} \leftarrow \boldsymbol{M}^{(0)}$
6      for $epoch \leftarrow 1$ to $E_{ot}$ do
7          $I \leftarrow \mathrm{argmax}_i\ \rho(\boldsymbol{r}_i, r_i(\boldsymbol{M}))$
8          $J \leftarrow \mathrm{argmax}_j\ \rho(\boldsymbol{c}_i, c_i(\boldsymbol{M}))$
9          if $\rho(\boldsymbol{r}_I, r_I(\boldsymbol{M})) > \rho(\boldsymbol{c}_J, c_j(\boldsymbol{M}))$ then
10             $\boldsymbol{x}_I \leftarrow \boldsymbol{x}_I \cdot \frac{\boldsymbol{r}_I}{r_I(\boldsymbol{M})}$
11         else
12             $\boldsymbol{y}_I \leftarrow \boldsymbol{y}_I \cdot \frac{\boldsymbol{c}_J}{c_J(\boldsymbol{M})}$
13         $\boldsymbol{M} \leftarrow \mathrm{diag}(\boldsymbol{x})\boldsymbol{M}^{(0)}\mathrm{diag}(\boldsymbol{y})$
14     return $\boldsymbol{M}$
```

REFERENCES

[1] Jason Altschuler and Jonathan Weed. 2017. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems 30.* 1964–1974.

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. 2020. Self-labelling via simultaneous clustering and representation learning. In *Proceedings of the 8th International Conference on Learning Representations.*

[3] Piotr Bojanowski and Armand Joulin. 2017. Unsupervised learning by predicting noise. In *Proceedings of the 34th International Conference on Machine Learning*. 517–526.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management.* 891–900.

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence.* 1145–1152.

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the 15th European Conference on Computer Vision*. 139–156.

[7] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 3438–3445.

[8] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning*. 1725–1735.

[9] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems 33*. 19314–19326.

[10] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. 2020. Adaptive graph encoder for attributed graph embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 976–985.

[11] Hejie Cui, Wei Dai, Yanqiao Zhu, Xiaoxiao Li, Lifang He, and Carl Yang. 2022. Interpretable graph neural networks for connectome-based brain disorder analysis. In *Proceedings of the 25th International Conference on Medical Image Computing and Computer Assisted Intervention (Lecture Notes in Computer Science, Vol. 13438)*. 375–385.

[12] Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26*. 2292–2300.

[13] Leyan Deng, Chenwang Wu, Defu Lian, Yongji Wu, and Enhong Chen. 2022. Markov-driven graph convolutional networks for social spammer detection. *IEEE Trans. Knowl. Data Eng.* (2022).

[14] Hongchang Gao and Heng Huang. 2018. Deep attributed network embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3364–3370.

[15] Pallabi Ghosh, Nirat Saini, Larry S. Davis, and Abhinav Shrivastava. 2021. Learning graphs for knowledge transfer with limited labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11151–11161.

[16] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. 2018. Unsupervised representation learning by predicting image rotations. In *Proceedings of the 6th International Conference on Learning Representations*.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[18] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74.

[19] Xiao Huang, Jundong Li, and Xia Hu. 2017. Accelerated attributed network embedding. In *Proceedings of the SIAM International Conference on Data Mining*. 633–641.

[20] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 148–156.

[21] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. Retrieved from arXiv:1611.07308v1.

[22] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

[23] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2017. Colorization as a proxy task for visual understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 840–849.

[24] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 3538–3545.

[25] Defu Lian, Zhihao Zhu, Kai Zheng, Yong Ge, Xing Xie, and Enhong Chen. 2023. Network representation lightening from hashing to quantization. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 5119–5131.

[26] Ralph Linsker. 1988. Self-organization in a perceptual network. *IEEE Comput.* 21, 3 (1988), 105–117.

[27] Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the 14th European Conference on Computer Vision*. 69–84.

[28] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. 2018. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*. 9359–9367.

[29] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2536–2544.

[30] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph representation learning via graphical mutual information maximization. In *Proceedings of the Web Conference (WWW'20)*. 259–270.

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 701–710.

[32] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 459–467.

[33] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 385–394.

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

[35] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. In *Proceedings of the NeurIPS Workshop on Relational Representation Learning*.

[36] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*. 5892–5899.

[37] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. 1067–1077.

[38] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. 2020. On mutual information maximization for representation learning. In *Proceedings of the 8th International Conference on Learning Representations*.

[39] Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9 (2008), 2579–2605.

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*.

[41] Petar Veličković, William Fedus, William L. Hamilton, and Pietro Liò. 2019. Deep graph infomax. In *Proceedings of the 7th International Conference on Learning Representations*.

[42] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 3670–3676.

[43] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. MGAE: Marginalized graph autoencoder for graph clustering. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. 889–898.

[44] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. 203–209.

[45] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*. 6861–6871.

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. Retrieved from arXiv:1901.00596v1.

[47] Teng Xiao, Zhengyu Chen, Donglin Wang, and Suhang Wang. 2021. Learning how to propagate messages in graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1894–1903.

[48] Zhilin Yang, William W. Cohen, and Ruslan R. Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*, Vol. 48. 40–48.

[49] Wayne W. Zachary. 1977. An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* 33, 4 (1977), 452–473.

[50] Jinghao Zhang, Yanqiao Zhu, Qiang Liu, Shu Wu, Shuhui Wang, and Liang Wang. 2021. Mining latent structures for multimedia recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 3872–3880.

[51] Richard Zhang, Phillip Isola, and Alexei A. Efros. 2016. Colorful image colorization. In *Proceedings of the 14th European Conference on Computer Vision*. 649–666.

[52] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. 2019. Attributed graph clustering via adaptive graph convolution. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 4327–4333.

[53] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. In *Proceedings of the International Conference on Machine Learning Workshop on Graph Representation Learning and Beyond (ICML'20)*.

[54] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference (WWW'21)*. 2069–2080.

[55] Yanqiao Zhu, Shichang Zhang, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Shu Wu, Yizhou Sun, and Wei Wang. 2021. A survey on graph structure learning: Progress and promise. Retrieved from https://arXiv:2103.03036v2

Received 18 March 2022; revised 27 April 2023; accepted 16 June 2023