Collaborative Graph Contrastive Learning: Data Augmentation Composition May Not Be Necessary for Graph Representation Learning

Yuxiang Ren  
IFM Lab  
Department of Computer Science  
Florida State University, FL, USA  
yuxiang@ifmlab.org

Jiawei Zhang  
IFM Lab  
Department of Computer Science  
University of California, Davis, USA  
jiawei@ifmlab.org

ABSTRACT
Unsupervised graph representation learning is a non-trivial topic for graph data. The success of contrastive learning in the unsupervised representation learning of structured data inspires similar attempts on the graph. The current unsupervised graph representation learning and pre-training using the contrastive loss are mainly based on the contrast between handcrafted augmented graph data. However, the graph data augmentation is still not well-explored due to the uncontrollable invariance. In this paper, we propose a novel collaborative graph neural networks contrastive learning framework (CGCL), which uses multiple graph encoders to observe the graph. Features observed from different views act as the graph augmentation for contrastive learning between graph encoders, which avoid inducing any perturbation to guarantee the invariance. CGCL can handle both graph-level and node-level representation learning. Extensive experiments demonstrate the advantages of CGCL in unsupervised graph representation learning and the non-necessity of handcrafted data augmentation composition for graph representation learning. Our code are available at: Clickable link.

KEYWORDS
Graph Neural Network, Graph Representation Learning, Contrastive Learning

1 INTRODUCTION
Graphs have been proven to possess outstanding capabilities in representing diverse types of data from various research fields, including social networks [33], financial risk control [30], biological protein analysis [34], and intelligent transportation systems [51]. The advantage of graph-structured data in terms of information representation is that it can not only contain the attribute information of individual units (i.e., nodes) but also explicitly provide the connection information (i.e., edges) between these units. Due to this ability, many applications exhibit the favorable property of graph-structured data, which makes learning effective graph representations for downstream tasks a non-trivial problem.

Recently, graph neural networks (GNNs) [21, 35, 37, 47] demonstrate the state-of-the-art performance in both graph-level and node-level representation learning. At the node-level, GNNs [21, 37] focus on learning low-dimensional representations of nodes through integrating neighborhood information. When talking about the graph-level, GNNs [35, 47] seek to learn low-dimensional representations of the entire graph. In most scenarios, GNNs applied at the graph-level and node-level representation learning tasks are trained in supervised or semi-supervised ways. However, task-specific labels are scarce and unevenly distributed [35], and obtaining demanding labels is also extremely time-consuming and labor-intensive. Experiments for protein labeling, for example, necessitate a large workforce, material resources, and time. Besides, in financial risk control tasks, fraud labels are significantly fewer than normal labels.

For the obstacle of scarce and uneven labels, unsupervised graph representation learning has emerged as the critical technology to achieve breakthroughs. Graph kernels [19, 32, 46] can learn the representations in an unsupervised manner, but the handcrafted kernel features may lead to poor generalization performance. Self-supervised graph learning methods [18, 20, 49] define pretext tasks as their supervision to implement unsupervised graph representation learning. The pretext tasks are designed based on heuristics and presuppose a specific set of representational invariance (e.g., pairwise attribute similarity [20]), whereas downstream tasks may not meet this presupposition (e.g., A classification problem related pairwise attribute similarity [20]), whereas downstream tasks may not meet this presupposition (e.g., A classification problem related to the structure of the graph). In this way, the generality of learned representation can not be guaranteed. Recently, contrastive learning has shown attractive potentials in unsupervised representation learning, including natural language processing [8, 27] and visual representation [6, 13, 42]. In the graph domain, some works [26, 29, 38, 48] also have explored the mechanism, which tends to maximize feature consistency under differently augmented views to learn the desired invariance from these transformations. As a result, data augmentation technologies have a strong influence on contrastive learning performance.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Conference’17, July 2017, Washington, DC, USA  
© 2021 Copyright held by the owner/author(s).  
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM  
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1https://www.dropbox.com/sh/wvtozrr7bvlwidi/AACog- sUqYONmDXAvPO0qCa?dl=0
Data augmentation has achieved great success in image data where the invariance of various views (e.g., color-invariant, rotation-invariant, and resizing-invariant) are well-understood [6, 43]. However, due to the complex structural information and the coupling between nodes in the graph, the changes induced by the data augmentation are not easy to measure. For example, in image data, it is natural to evaluate what kind of data augmentation is more powerful (e.g., color distortion strength for color permuting [6]), but the situation of the graph data is much more complicated. Modifying the attributes of a node or removing an edge is not only related to the target node but also affects many other nodes [41], especially when GNNs follow the recursive neighborhood aggregation scheme mechanism [45]. Furthermore, the importance of each node and edge in the graph is far from equivalent, which differs from the importance of pixels in image data. For example, from the node-level perspective, changing the information of key nodes will cause its neighbor nodes to be affected so seriously that their semantic labels change. However, the data augmentation requires that the neighbor nodes keep stable [44, 48]. From the graph-level view, removing a critical edge is enough to change a graph from a connected graph to a disconnected one, making the augmented graph and the original graph have little learnable invariance. We provide an illustration in Figure 1 to show the unstable invariance between the original graph and augmented graphs under three different data augmentation methods (Node dropping, Edge perturbation, and Subgraph sampling). The upper part of each data augmentation method shows an augmented graph that is highly invariant to the original graph, while the lower part lacks invariance to the original graph.

We propose a graph contrastive learning framework that does not composite graph data augmentation strategies based on prior domain knowledge. Because current graph data augmentation strategies fail to ensure the invariance between the augmented graph/node and the original graph/node, our framework chooses to utilize the invariance of representations learned by different GNN models for contrastive learning from a novel perspective. Specifically, different GNN models adopt their unique "message-passing" schemes (e.g., spectral-based and spatial-based). However, the learned universal representations should be similar for the same graph/node due to the consistent structural and attribute information. Instead of learning the graph representation in a single embedding space invariant to the handcrafted augmentations, we utilize multiple GNN models to learn the representations in various embedding spaces, where the representation is invariant to the same graph/node’s representation in other embedding spaces. In essence, we regard the representations learned by each GNN model as a kind of data augmentation, and these augmented graphs are implemented through distinct embedding spaces and "message-passing" schemes. Because no handcrafted disturbance information is injected, data augmentation based on the collaboration of multiple GNN models can ensure invariance between augmented and original graphs/nodes. Besides, the collaboration of multiple GNN models can achieve a performance similar to ensemble learning. The invariance between representations learned by multiple models can benefit the final representation learning.

We name this framework for multiple GNN models collaborative contrastive learning as Collaborative Graph Neural Networks Contrastive Learning (CGCL). CGCL is a general framework that can be applied for training both graph-level and node-level GNN models, such as GAT [37], GCN [21], GIN [45], DGCNN [50], and so on. CGCL working for graph-level and node-level representation learning differs slightly in that graph-level CGCL is based on batch-wise contrastive learning, whereas node-level CGCL employs a graph-wise mechanism. We validate the performance of CGCL on both graph classification tasks and node classification tasks over 12 benchmark graph datasets (9 for graph-level learning, 3 for node-level learning). Compared with the graph contrastive learning with data augmentation composition [48], CGCL demonstrates better generalization on various datasets and achieves better results without a handcrafted data augmentation composition. Experimental results reflect that existing data augmentation composition may be unnecessary for graph contrastive learning. Because augmenting the graph data induces a significant amount of noise, the performance improvement of the representation learning is unstable and varies across datasets.

The contributions of our work are summarized as follows:

- We propose a novel collaborative graph neural networks contrastive learning framework (CGCL) to reinforce unsupervised graph representation learning. CGCL requires no handcrafted data augmentation and composition compared with other graph contrastive learning methods.
- CGCL can work on both graph-level and node-level learning and show better generalization across different domains of datasets. The collaboration of multiple GNNs empowers CGCL with the
benefits such as ensemble learning, which enable to learn better representations.

- Extensive experiments show that CGCL has advantages in graph classification tasks and node classification tasks with unsupervised learning. In addition, we present the theoretical and empirical analysis of the performance of collaboration between different GNN models.

2 RELATED WORK

2.1 Graph Representation Learning

Graph representation learning has become a critical topic due to the ubiquity of graphs in real-world scenarios. As a data type containing rich structural information, many research works embed the graphs purely based on the structure. Node2vec [11] learns a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes. LINE [36] optimizes a carefully designed objective function that preserves both the local and global graph structures. Wang et al. [40] attempt to retrieve structural information through matrix factorization incorporating the community structure. In the recent past, graph neural networks [12, 21, 28, 35, 37, 47] have shown awe-inspiring capabilities in graph representation learning, which aggregate the neighbors’ information through neural networks to learn the latent representations [45]. Kipf et al. [21] propose Graph Convolutional Network (GCN) that extends convolution to graphs by a novel Fourier transformation. Graph Attention Network (GAT) [37] first imports the attention mechanism into graphs. Due to insufficient labeled data and out-of-distribution samples in training data, unsupervised graph representation learning gradually attracts researchers’ attention. DGI [38] is a general GNN for learning node representations in an unsupervised manner, which relies on maximizing mutual information between patch representations and corresponding high-level summaries of graphs. HDGI [29] extends the mutual information maximization-based mechanism to the heterogeneous graph representation learning. InfoGraph [35] extends the mutual information maximization to graph-level representations. Besides, several models [17, 18] work for pre-training in graph-structured data can help learn graph representations without labeled data.

2.2 Contrastive Learning

Contrastive learning has been used for unsupervised learning in vast fields, including natural language processing [8, 27] and visual representation [6, 13, 42], by training an encoder that can capture similarity from data. The contrastive loss is usually a scoring function that increases the score on the single matched instance and decreases the score on multiple unmatched instances [25, 42]. In the graph domain, GCC [26] utilizes contrastive learning to pre-train a model that can serve for the downstream graph classification task by fine-tuning. LCGNN [28] designs a label contrastive loss to implement the contrastive learning under the guide of graph labels. GraphCL [48] introduces data augmentation strategies for graph contrastive learning. The contrastive learning between augmented graphs can help improve the learned graph representations. Compared to GraphCL, our proposed method learns graph/node representations through collaborative contrastive learning between different GNN models instead of between manually augmented graphs.

2.3 Data Augmentation

In visual representation learning, data augmentation has been widely applied in both supervised and unsupervised methods [2, 6, 14]. The augmented data can be utilized to define contrastive prediction tasks, including global-to-local view prediction [2], neighboring view prediction [14] and so on. Chen et al. [6] perform simple random cropping to avoid changing the architecture when defining the contrastive prediction. The current works on graph data augmentation are relatively limited. GraphSFC [27] generates fake samples in low-density areas between subgraphs as augmented data and discriminates fake samples from the real in an adversarial way. Adversarial perturbations are generated to augment node features in [7]. NodeAug [41] regularizes the model prediction to ensure invariance when the data augmentation strategies induce changes. GraphCL [48] systematically investigates the impact of various handcrafted graph data augmentation compositions on graph representation learning. Current graph data augmentation methods primarily rely on injecting handcrafted perturbation into the graph-structured data, whereas CGCL proposed in this paper utilizes different graph encoder observation views of graph-structured data as a data augmentation method.

3 METHODOLOGY

In this section, we elaborate on the collaborative graph neural networks contrastive learning (CGCL) framework in detail for both graph-level and node-level representation learning. CGCL employs multiple GNN models as the graph encoders. Each of them works as a graph data augmentation function for other GNN models to improve the quality of learned representations. Each GNN model in CGCL updates its parameters through the contrastive learning [6] between its learned representation and other GNN models’ outputs. We provide the preliminaries about unsupervised graph-level and node-level representation learning first.

3.1 Preliminaries

The goal of unsupervised graph-level representation learning is to embed the entire graph into a low-dimensional vector. In comparison, node-level representation learning aims at learning the low-dimensional representation of each node. Both of them require no label information during the learning process. Formally, we define them as follows:

Unsupervised Graph-level Representation Learning Given a set of graphs $\mathcal{G} = \{G_1, G_2, \ldots \}$, the task is to learn a $d$-dimensional representation for each $G_i \in \mathcal{G}$. Here, $d$ is the pre-defined value indicating the number of dimensions of learned graph representations.

Unsupervised Node-level Representation Learning Given a graph $G = (\mathcal{V}, \mathcal{E})$ and the initial feature matrix of all nodes $\mathbf{X} \in \mathbb{R}^{\mathcal{V} \times D}$, and $D$ is the dimension of initial features. The unsupervised node-level representation learning is to learn the low dimensional node representations $\mathbf{H} \in \mathbb{R}^{\mathcal{V} \times d}$ that contains both structural information of $G$ and node attributes from $\mathbf{X}$. Here, $d$ is the number of dimensions of node representations.
Input Graphs

Minibatch

... which define the "message-passing" schemes. For the graph-level model can work for graph representation learning when taking a

representation learning tasks, GNNs need an extra READOUT(·) to summarize the representation of \( G \) from node representations:

\[
h_G = \text{READOUT}\left( \{ h_n^{(l)} : n \in {V, l \in L} \} \right)
\]

(2)

The key to CGCL is to choose various GNNs for collaborative learning. Technically, any GNN working on graph-level tasks can be used as one of the graph encoders here. However, the collaborations of different types of GNNs have a significant impact on the performance of CGCL. We analyze the GNN models selection in Section 4.2.3 from experimental results. Here, we propose three empirical insights first:

- The difference between the embedding spaces of various GNNs is uneven. The performance of the collaboration is harmed when there are too many or too few disparities.
- The READOUT (pooling) method has a significant influence on the graph-level representation. In the supervised learning, pooling methods are strongly sensitive to the specific task, but the selected pooling method may not be suitable for unsupervised representation learning.
- Employing more GNNs as graph encoders can help achieve better performance. This is equivalent to learning more essential invariance from multi-view representations.

For graph-level representation learning, the graph encoder candidates include GIN [45], GAT [37], GCN [21], and DGNN [50] in this paper. They can be combined for collaboration with CGCL. It should be noted that the READOUT of each model should be unified (e.g., global add pooling) instead of using the default method in the original work.

3.2.3 Graph representation as the data augmentation. The graph representations learned by different GNNs essentially act the data augmentation in CGCL. However, the data augmentation mechanism used here differs from that of GraphCL [48], which is designed manually based on priors of graph-structured data. CGCL achieves this kind of data augmentation through different “message-passing” schemes, equivalent to observing a graph from multiple views. We provide the detailed analysis of this insight in Section 3.4 in a theoretical way.

3.2.4 Batch-wise contrastive loss. For graph-level learning, we utilize the training batch to conduct contrastive learning. Given a minibatch of \( N \) graphs \( \{ G_1, G_2, \ldots, G_N \} \) as the input of \( k \) GNN models \( \{ M_1, M_2, \ldots, M_k \} \). Contrastive learning can be considered as learning an encoder for a dictionary look-up task [13]. We take \( M_1 \) as an example. The representations \( \{ h_{G_i}^{M_1}, h_{G_j}^{M_1}, \ldots, h_{G_N}^{M_1} \} \) learned by \( M_1 \) act as query graphs, while the representations learned by other models are key graphs. Between query and key graphs, the contrastive loss can be represented as:

\[
\left\{ (h_{G_i}^{M_1}, h_{G_j}^{M_2}) \mid i \in [1, N], j \in [2, k] \right\}
\]

(3)

Therefore, each graph encoded by \( M_1 \) has \( k - 1 \) positive samples. In essence, the positive graph pair indicates the representations of the same graph but learned by different graph encoders. Meanwhile, the other pairs are negative. Here, we utilize the InfoNCE [25] to calculate the contrastive loss for each graph encoder. We define the contrastive loss between two graph encoders \( M_p \) and \( M_q \) first, so the contrastive loss \( \mathcal{L}_{M_p}(M_q) \) can be represented as:

Figure 2: CGCL for graph-level representation learning. GNN Model 1, 2, · · · , \( k \) are different types of graph neural networks. The minibatch graphs are embedded by them and obtain low-dimensional representations. In order to optimize the corresponding GNN model, a contrastive loss is calculated for each GNN model. For instance, GNN Model 1 uses representations learned by itself as query graphs and representations learned by all other GNN models as key graphs to compute Model 1 contrastive loss.

In the following section, we introduce CGCL handling two tasks, respectively.

3.2 CGCL for Graph-level Learning

3.2.1 Framework Overview. We show in Figure 2 the overall framework of CGCL in processing graph-level representation learning task. The GNN models \( \{ M_1, M_2, \ldots, M_k \} \) used by CGCL as graph encoders are able to encode entire graphs, and the output is the representation of each graph. The graph representations learned by \( M_i \) are utilized by \( \{ M_1, M_2, \ldots, M_{i-1}, M_{i+1}, \ldots, M_k \} \) as graph data augmentations. A contrastive loss for each GNN model is defined to enforce the invariance between the positive pairs, which we will define in Section 3.2.4. Each GNN model can be updated based on the defined contrastive loss. After learning, each trained GNN model can work for graph representation learning when taking a graph as input.

3.2.2 GNN-based graph encoder. Given a set of graphs, CGCL needs to encode them into vectorized representations. Graph neural networks (GNNs) [12, 21, 37] have demonstrated their outstanding ability in encoding graph-structured data. In CGCL, we mainly consider GNNs as graph encoders. GNNs follow the recursive neighborhood aggregation (“message-passing”) scheme [45] to encode the node’s structural and attribute information. Given a graph \( G = (V, E) \) and the initial feature matrix of all nodes \( X \). The aggregation process at the \( l \)-th layer of a GNN can be represented as:

\[
a_n^{(l)} = \text{AGGREGATION}^{(l)}\left( \{ h_n^{(l-1)} : n' \in N(n) \} \right), \\
h_n^{(l)} = \text{COMBINE}^{(l)}\left( h_n^{(l-1)}, a_n^{(l)} \right)
\]

(1)

Here, \( h_n^{(l-1)} \) is the representation of node \( n \) at the \((l-1)\)-th layer with \( h_n^{(0)} = X[n] \), and \( N(n) \) is the set of neighboring nodes of node \( n \). The difference between each GNN is mainly reflected in the use of unique AGGREGATION(·) and COMBINE(·) functions, which define the “message-passing” schemes. For the graph-level representation learning, we show in Figure 2 the overall hand-crafting two tasks, respectively.
\[ \mathcal{L}_{M_p}(M_q) = -\sum_{j=1}^{N} \log \frac{\exp \left( \frac{\mathbf{h}^{M_p}_{ij} \cdot \mathbf{h}^{M_q}_{ij}}{\tau} \right)}{\sum_{n=1}^{N} \exp \left( \frac{\mathbf{h}^{M_p}_{ij} \cdot \mathbf{h}^{M_q}_{ij}}{\tau} \right)} \]  

(4)

Here, \( \tau \) is the temperature hyper-parameter that controls the concentration level of the distribution [15]. In this contrastive loss, \( M_p \) acts as the query encoder (i.e., providing query graphs), while \( M_q \) provides key graphs as data augmentation. When considering all collaborative graph encoders, the contrastive loss of \( M_p \) can be denoted as:

\[ \mathcal{L}_{M_p} = \sum_{M_q \in \{M_q\}_{i \neq p}} \mathcal{L}_{M_p}(M_q) \]  

(5)

\( M_p \) can be updated based on the contrastive loss \( \mathcal{L}_{M_p} \) through back-propagation. Other GNNs follow the same process to compute their own contrastive loss and get updated through minibatch training iteratively. The detailed collaborative learning process of CGCL is described in Algorithm 1.

3.2.5 CGCL helps GNN pre-training. CGCL does not rely on specific tasks labels so that helps the pre-training of the GNN. The GNNs trained through CGCL can embed the graph data universally. For specific downstream tasks, the trained GNNs can be fine-tuned in an end-to-end manner, which can help overcome the challenges of scarce task-specific labeled data and out-of-distribution samples in training set [17].

Discussion Several existing contrastive learning models [13, 28, 42] choose to maintain a memory bank, which can break the consistencies problem becomes more serious. Therefore, we drop the constraints from batch size. The memory bank needs to have a large consistency problem becomes more serious. Therefore, we drop the memory bank in CGCL, but adopt batch-wise learning.

It is also worth discussing that the idea of CGCL is close to ensemble learning [52], which helps improve machine learning results by combining multiple models. As the meta-algorithms, traditional ensemble methods work on combining several machine learning techniques into one predictive model in order to decrease variance (e.g., Bagging) and bias (e.g., Boosting) [53]. Think back to CGCL, each graph encoder utilizes its unique "message-passing" scheme to analyze the graph data, while CGCL contributes to eliminate the variance and bias between the same graph’s representation from different graph encoders and promote them to reach a consensus. For unsupervised graph representation learning, the contrastive loss can indicate this consensus and guide the ensemble learning process.

3.3 CGCL for Node-level Learning

3.3.1 Framework Overview. The main difference for node-level learning is that the representation learning of nodes requires all nodes in the same graph as input for GNN-based graph encoders and cannot be randomly divided into batches because of node dependency [3]. Therefore, CGCL employs graph-wise contrastive learning for node-level representation learning, which means the contrastive loss is computed within each complete graph.

3.3.2 Graph-wise contrastive loss. Given a graph \( G = (V, E) \) and the initial feature matrix \( X \) as the input of \( k \) node-level GNN models \( \{M_1, M_2, \ldots, M_k\} \). Taking \( M_p \) as an example, node representations \( \{\mathbf{h}^{M_p}_{V_1}, \mathbf{h}^{M_p}_{V_2}, \ldots, \mathbf{h}^{M_p}_{V_N}\} \) learned by it based on Equation 1 are query nodes, where \( v \in V \). The node representations learned by another model \( M_q \) serve as key nodes. The contrastive loss between \( M_p \) and \( M_q \) within the graph \( G \) is:

\[ \mathcal{L}_{M_p}(M_q, G) = -\frac{1}{|V|} \sum_{j=1}^{|V|} \log \frac{\exp \left( \frac{\mathbf{h}^{M_p}_{V_j} \cdot \mathbf{h}^{M_q}_{V_j}}{\tau} \right)}{\sum_{k \neq j} \exp \left( \frac{\mathbf{h}^{M_p}_{V_j} \cdot \mathbf{h}^{M_q}_{V_k}}{\tau} \right)} \]  

(6)

Considering all graph encoder, the contrastive loss of \( M_p \) can be calculated similar to Equation 5:

\[ \mathcal{L}_{M_p}(G) = \sum_{M_q \in \{M_q\}_{i \neq p}} \mathcal{L}_{M_p}(M_q, G) \]  

(7)

The contrastive loss can be computed for the datasets containing multiple graphs by summing the loss on all graphs. Each model can be updated by their contrastive loss, and the node representations learned by each model will reach a consensus finally.

3.4 Theoretical Analysis of CGCL

You et al. [48] build a bridge between the contrastive loss and mutual information maximization, which can help depict the essence of contrastive learning. We follow a similar way to analyze CGCL. For simplicity, we analyze CGCL with two graph encoders \( M_p \) and \( M_q \). For one sampled graph minibatch \( \{G_j\}_{j=1}^N \), the CGCL loss can be

\[
\frac{1}{|V|} \sum_{j=1}^{N} \left( \log \left( \frac{\exp \left( \frac{\mathbf{h}^{M_p}_{ij} \cdot \mathbf{h}^{M_q}_{ij}}{\tau} \right)}{\sum_{k \neq j} \exp \left( \frac{\mathbf{h}^{M_p}_{ij} \cdot \mathbf{h}^{M_q}_{ik}}{\tau} \right)} \right) \right)
\]
We continue to write the Equation 8 as the expectation form:

\[ \hat{t} \]

rewritten as:

\[ \mathcal{L}_{\text{MLP}} (M_\beta) = - \frac{1}{N} \sum_{i=1}^{N} \log \left( \frac{e^{h_{G_i}^{M_\beta}}}{\sum_{j=1}^{N} e^{h_{G_i}^{M_\beta}}} \right) \]

or

\[ \mathcal{L}_{\text{MLP}} (M_\beta) = - \frac{1}{N} \sum_{i=1}^{N} \log \left( \frac{e^{h_{G_i}^{M_\beta}}}{\sum_{j=1}^{N} e^{h_{G_i}^{M_\beta}}} \right) + N \log N \]

Here, \( f_{M_\beta} (\cdot) \) and \( s_{M_\beta} (\cdot) \) are graph encoders \( M_\beta \) and \( M_\beta \) accordingly.

We continue to write the Equation 8 as the expectation form:

\[ \mathcal{L}_{\text{MLP}} (M_\beta) = - \frac{1}{2} \sum_{i,j} \log \left( \frac{e^{h_{G_i}^{M_\beta}}}{\sum_{j=1}^{N} e^{h_{G_i}^{M_\beta}}} \right) \]

\[ \mathcal{L}_{\text{MLP}} (M_\beta) = - \frac{1}{2} \sum_{i,j} \log \left( \frac{e^{h_{G_i}^{M_\beta}}}{\sum_{j=1}^{N} e^{h_{G_i}^{M_\beta}}} \right) + N \log N \]

Here, \( P (f_{M_\beta} (G), s_{M_\beta} (G)) \) and \( T (\cdot, \cdot) \) is a learnable score function which we apply the inner product together with temperature factor \( \tau \). Based on the conclusion from Hjelm et al. [16], optimizing Equation 10 is equivalent to maximizing a lower bound of the mutual information between the representations \( h_{G_i}^{M_\beta} = f_{M_\beta} (G) \).

GraphCL [48] defines a general framework for existing graph contrastive learning methods, where the general loss can be formulated as:

\[ \mathcal{L} = \mathbb{E}_{G_i} \left[ - \mathbb{E}_{f_i (G_i), f_j (G_j)} \right] + \log (\mathbb{E}_{G_i} e^{T(\hat{f}_i (G_i), f_j (G_j))}) \]

\[ \mathcal{L} = \mathbb{E}_{G_i} \left[ - \mathbb{E}_{f_i (G_i), f_j (G_j)} \right] + \log (\mathbb{E}_{G_i} e^{T(\hat{f}_i (G_i), f_j (G_j))}) \]

4 EXPERIMENTS

In this section, we first verify the effectiveness of the graph-level representations learned by CGCL in the setting of unsupervised learning and pretrain-finetune learning [48] in the graph classification tasks. Then we demonstrate the node representations learned by CGCL in the task of node classification [38]. In addition, we analyze the source of the performance improvement brought by CGCL and visualize the learned node representations in Section 4.3. At last part, we verified the reliability of the collaboration between graph encoders through convergence analysis.

4.1 Datasets

For graph-level representation learning, we conduct experiments on 9 widely used benchmark datasets: NCI1, PROTEINS, D&D, MUTAG, COLLAB, IMDB-B, IMDB-M, RDT-B, RDT-M. From the dataset, we collect 3 well-known benchmark dataset: Cora, CiteSeer and Pubmed [31]. For all the above datasets, we reach them with the support of the PyTorch Geometric Library. Additional details of these datasets are in Appendix 6.1.

4.2 Graph-level Representation Learning

4.2.1 Unsupervised Graph Representation Learning

We first evaluate the representations learned by CGCL in the setting of unsupervised learning. We follow the same process of InfoGraph [35], where representations are learned by models without any labels and then fed them to a SVM from sklearn to evaluate the graph classification performance. We select three categories of models as our comparison methods:

- **Kernel-based method**: Weisfeiler-Lehman subtree kernel (WL) [32], Graphlet Kernel (GL), and Deep Graph Kernel (DGK) [46]: They first decompose graphs into sub-components based on the kernel definition, then learn graph embeddings in a feature-based manner.

- **Graph-encoding methods**: Sub2vec [1], Node2vec [11], Graph2vec [24]: They extend document embedding neural networks to learn representations of entire graphs.

- **Graph neural networks methods**: InfoGraph [35]: It is an unsupervised with the pooling operator for graph representation learning base on the mutual information maximization. GCC [26]: It follows pre-training and fine-tuning paradigm for graph representation learning. GarphCL [48]: It implements contrastive learning between augmented graphs to obtain graph representation in an unsupervised manner.

Table 1: Graph classification accuracy on all datasets. We use bold red number to denote the best result on each dataset.

| Methods | NCI1 | PROTEINS | D&D | MUTAG | COLLAB | IMDB-B | IMDB-M | RDT-B | RDT-M |
|---------|------|----------|-----|-------|--------|--------|--------|-------|-------|
| WL      | 76.65±1.99 | 72.92±0.56 | 76.44±2.35 | 80.72±3.00 | 72.30±3.44 | 46.93±0.46 | 68.82±0.41 | 46.00±0.21 |
| Kernels | GL   | 62.48±2.11 | 72.25±4.49 | 72.54±3.83 | 81.66±2.11 | 72.84±0.28 | 65.87±0.98 | 77.34±0.18 | 41.01±0.17 |
| Graph   | Node2vec | 48.39±1.81 | 57.94±5.37 | 67.12±4.52 | 72.63±10.20 | 69.03±7.13 | 69.03±7.13 | 69.03±7.13 | 69.03±7.13 |
| Embed   | Sub2vec | 52.84±1.47 | 53.03±5.55 | 59.34±8.01 | 61.05±15.80 | - | 55.26±1.54 | 36.67±0.83 | 71.48±0.41 | 36.68±0.42 |
| Graph   | Graph2vec | 75.22±4.83 | 73.30±2.05 | 71.98±3.54 | 83.15±9.25 | - | 71.10±0.54 | 50.44±0.87 | 75.78±1.03 | 47.86±0.26 |
| Graph   | GCC_corse | 76.25±1.26 | 74.49±1.71 | 72.85±1.78 | 89.01±1.15 | 78.55±1.78 | 72.60±1.72 | 50.45±1.61 | 72.60±1.72 |
| GNNS    | GraphCL | 77.87±0.41 | 74.39±0.45 | 78.62±0.40 | 86.80±1.34 | 71.36±1.15 | 71.14±0.44 | - | 89.59±0.84 | 55.99±0.28 |

Proposed CGCL/GN: 77.89±0.54 | 76.28±0.31 | 79.37±0.47 | 89.05±1.42 | 73.28±1.22 | 73.11±0.74 | 51.73±1.37 | 91.31±1.22 | 54.47±1.08 |
CGCL denotes Gin [45] trained under the proposed framework CGCL. The graph encoder candidates include GAT [37], GCN [21], and DGCNN [50], which are combined with Gin to conduct collaborative contrastive learning. The representations learned by CGCL are extracted for the downstream graph classification. We test the task ten times and report the average accuracy and the standard deviation. In our experiments, only two graph encoders or three graph encoders collaborative learning settings are used. The result of CGCL Gin we report in Table 1 is the best, and the results of other assembly are reported in Table 3. More detailed information about training settings can be viewed in Appendix 6.3. From the results in Table 1, CGCL achieves the best result on the six datasets. In particular, compared with GraphCL, GraphCL has an advantage in the dataset RDT-M, but not in some datasets with a low average node degree. This observation is also in line with our illustration in Figure 1, because when the graph is small and not dense enough, the perturbation from the data augmentation is more likely to damage the invariance, e.g., a greater probability of affecting the graph connectivity by dropping a single node.

4.2.2 Pretrain-Finetune Graph Representation Learning. In this setting, CGCL first conducts collaborative contrastive learning to pretrain the graph encoders. Then we use 10% labeled graph data of the whole dataset to finetune the pretrained graph encoder end-to-end for the downstream graph classification tasks. Here, we utilize GAE [22], InfoGraph [35], GCC [26], and GraphCL [48] to pretrain GCN for comparison. GAE focuses on the graph structure reconstruction and InfoGraph utilizes the local-global mutual information maximization. GCC and GraphCL pretrain the graph encoder based on the contrastive learning between augmented graph data. CGCL employs GCN and Gin as graph encoders to fulfill the designed contrastive learning in each dataset. For fairness, we finetune GCN with 10% labeled graph and report the graph classification accuracy. More implementation details can be referred to in Appendix 6.3. Table 2 shows the results of combining two or three graph encoders on 3 datasets. We focus on the performance of unsupervised graph representation learning of CGCL Gin. From Table 3, we can find that the collaboration of three graph encoders is better than that of the two encoders. The phenomenon is consistent with the conclusion proposed by GraphCL [48], that is, the harder data augmentation benefits the performance of contrastive learning. Here, the consensus reached by more graph encoders is a kind of harder data augmentation. In addition, we find that DGCNN cannot help train Gin better through collaboration than GCN and GAT. When we analyze the structure of DGCNN, we notice that the pooling layer of DGCNN is forcibly closed to the graph classification task and needs to be trained in an end-to-end way. This ungeneralized design makes the pooling layer can not be well trained in unsupervised learning. In comparison, we assign a global add pooling layer to GAT, GCN and Gin. It also reflect that the READOUT method has a great influence on the graph level representation, and we should keep the graph encoder use the same pooling layer to diminish the difference of embedding spaces. Here, we additionally use a non-GNN method Set2Set [39] for comparison, and the results show that it does not collaborate well with the GNN models. The reason should also be that the embedding space is too different, which makes it difficult for the graph encoders to reach a consensus on representations.

4.3 Node-level Representation Learning
4.3.1 Node classification Tasks. For the effectiveness of CGCL on node-level representation learning, we use the node classification task to evaluate. The representations learned through unsupervised methods are fed into a downstream classifier. GAT and GCN are utilized as the graph encoders in CGCL. DGI [38] and GraphCL [48] are used as unsupervised comparison methods. GCN and GAT trained in a semi-supervised manner are compared with CGCL as well. We conduct the experiments on three datasets (Cora, Citeseer, and PubMed) with random splits, where 40 nodes per class from labeled nodes in the training set, 500 nodes are used as the validation set, and 1000 nodes for testing. Many existing works conduct experiments based on the standard split [21, 38], here we have to explain why we use 40 nodes per class for training instead of 20 nodes in the standard split. Since we use contrastive learning to discriminate each node, the distribution of representations learned by CGCL is sparser than those under semi-supervised conditions. The distance between node will be larger even if nodes have the

### Table 2: Graph classification accuracy in the setting of pretrain-finetune learning on all datasets. We use bold red number to denote the best result on each dataset.

| Methods           | NCII | PROTEINS | D&D | COLLAB | MMDB-B | IMDB-M | RDT-B | RDT-M |
|-------------------|------|----------|-----|--------|--------|--------|-------|-------|
| GAE               | 74.36±0.24 | 70.51±1.17 | 74.54±0.68 | 73.09±0.19 | - | - | 87.69±0.40 | 53.58±0.13 |
| InfoGraph         | 74.86±0.26 | 72.27±0.40 | 75.78±0.34 | 73.76±0.29 | 72.23±0.21 | - | 88.66±0.95 | 53.61±0.31 |
| GraphCL           | 74.63±0.25 | 74.17±0.34 | 76.17±1.37 | 74.23±0.21 | 74.23±0.31 | 74.23±0.31 | 89.11±0.19 | 52.55±0.45 |
| GCC               | 79.20±1.48 | 72.49±3.42 | 74.57±2.84 | 71.65±2.91 | 71.34±1.98 | 49.19±2.76 | 84.34±3.62 | 50.25±1.18 |
| GCLCGCN           | 75.41±0.52 | 74.98±0.48 | 76.23±0.39 | 72.59±0.48 | 72.74±0.30 | 71.82±0.34 | 85.98±0.24 | 50.30±0.30 |

### Table 3: Graph classification accuracy on all datasets with different graph encoder assemblies. The bold red number denotes the best learning result of Gin on each dataset.

| Dataset assembly | GIN | GCN | GAT | DGCNN | Set2Set |
|------------------|-----|-----|-----|-------|---------|
| PROTEINS         | 75.84 | 75.04 | - | 67.85 | - |
| GIN+DGCNN        | 72.5 | 72.92 | 72.76 | - | - |
| GIN+GCN+GAT      | 76.28 | 75.32 | 74.37 | - | - |
| GIN+GCN+DGCNN    | 74.08 | 74.85 | - | - | - |
| GIN+GCN+GAT      | 87.67 | 84.83 | - | - | - |
| GIN+GCN           | 87.96 | 86.72 | - | - | - |
| GIN+GAT           | 88.83 | 85.29 | - | - | - |
| GIN+Set2Set      | 85.15 | 78.35 | - | - | - |
| GIN+GCN+GAT      | 86.91 | 85.71 | 84.12 | - | - |
| GIN+GCN+DGCNN    | 89.05 | 85.74 | 86.94 | - | - |
| GIN+DGCNN        | 76.34 | 76.16 | - | - | - |
| GIN+GAT           | 76.39 | 72.58 | - | - | - |
| GIN+Set2Set      | 76.35 | 75.97 | - | - | - |
| D&D               | 73.42 | 73.42 | - | - | - |
| GIN+GAT           | 79.37 | 77.21 | 78.18 | - | - |
| GIN+DGCNN        | 77.44 | 74.40 | 74.85 | - | - |
same label, so using 20 nodes are not enough to train a classifier. However, this does not mean that the representations learned by CGCL are not effective because a qualified representation is to have stronger expressive ability and distinguishability. For the sake of fairness, we apply the same settings for all methods. We run the experiments 10 times and report the mean and standard deviation of accuracy. Other detailed experimental parameters are provided in Appendix 6.4. The node classification accuracy are presented in Table 4. DGI(64) denotes that the GCN trained by DGI has the 64-dimensional hidden feature, while the original DGI in [38] has the 512-dimensional hidden feature. All other methods set the dimension of the hidden feature as 64. GraphCL(A+I) means that AttrMask and Identical are used for the data augmentation. N and S represent NodeDrop and Subgraph, respectively. CGCL_GCN and CGCL_GAT beat all unsupervised methods. For Cora and PubMed, they even surpass semi-supervised learning methods, which have labeled data in training. The results fully demonstrate that for node-level representation learning, it may not be necessary to apply handcrafted data augmentation composition for contrastive learning similar to GraphCL. The data augmentation induced by the graph encoder enables the GNN model to learn important representations from multiple views.

4.3.2 Explore Source of Performance Improvement. In order to further verify that the collaboration of different GNN models can bring performance improvements and the effectiveness of data augmentation based on graph encoders, we design the experiment using two same GNN models as graph encoders. We take two GCN and two GAT to collaborate in CGCL to work on node-level tasks. We set two kinds of initialization for them, i.e., Xavier initialization [10] and Normal distribution initialization ($\mu=10$, $\sigma=0$). The experimental results are shown in Table 5. The 'Assembly' represents the composition of GNN with initialization methods. From the results in Table 5, it can be found that the same GNN model does not bring performance improvement because the same type of GNN learns graphs with the same ‘message-passing’ scheme. Even though they are initialized in different ways, no augmented graph data can be provided. In contrast, the results also demonstrate that the performance improvement of collaborative contrastive learning comes from the data augmentations that are node representations learned by different graph encoders.

4.3.3 Visualization of Learned Representations. In Figure 3, we visualize the learned representations by GCN, DGI, and CGCL on the CiteSeer dataset by t-SNE [4]. The representations of GCN are extracted from the last convolution layer. From this figure, we can find that fewer discrete nodes learned by CGCL_GCN than DGI, and the boundaries are clearer. Compared with semi-supervised GCN, nodes with the same label in CGCL_GCN is not as dense as GCN, but the completely overlap of nodes with different labels has eased, which is brought by discriminating each node through contrastive loss. We also present the visualization of representations on the Cora dataset in the Appendix 6.5.
In this paper, we propose a novel collaborative graph neural networks contrastive learning framework. DCCG aims at addressing the invariance issue present in current handcrafted graph data augmentation methods. Different from manually constructing augmented graphs, DCCG uses multiple GNN-based encoders to observe the graph, and representations learned from different views are used as the augmentation of the graph-structured data. This data augmentation method avoids inducing any perturbation so that the invariance can be guaranteed. Using the contrastive learning between the graph encoders, they can reach a consensus on the graph data feature and learn the general representations in an unsupervised manner. Extensive experiments not only verify the advantages of DCCG in both unsupervised graph-level and node-level representation learning, but also demonstrate the non-necessity of handcrafted data augmentation composition for graph contrastive learning.

REFERENCES

[1] Biju Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature learning for subgraphs. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 170–182. Springer, 2018.
[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. arXiv preprint arXiv:1906.09910, 2019.
[3] Jiayi Bai, Yuxiang Ren, and Jiawei Zhang. Ripple walk training: A subgraph-based training framework for large and deep graph neural networks. In 2021 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2021.
[4] Andreas Buja, Diane Cook, and Deborah F. Swayne. Interactive high-dimensional data visualization. Journal of computational and graphical statistics, 5(3):78–99, 1996.
[5] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Graph representation learning: a survey. APSIPA Transactions on Signal and Information Processing, 9, 2020.
[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pages 1597–1607. PMLR, 2020.
[7] Zhiqie Deng, Yinpeng Dong, and Jun Zhu. Batch virtual adversarial training for graph convolutional networks. arXiv preprint arXiv:1902.09192, 2019.
[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[9] Ming Ding, Jie Tang, and Ji Zhang. Semi-supervised learning on graphs with generative adversarial nets. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pages 913–922, 2018.
[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
[11] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pages 855–864, 2016.
[12] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02243, 2017.
[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9729–9738, 2020.
[14] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In International Conference on Machine Learning, pages 4182–4192. PMLR, 2020.
[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.
[16] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670, 2018.
[17] Yuxiang Ren, Hao Zhu, Jiawei Zhang, Peng Dai, and Liefeng Bo. Ensemfdet: An ensemble method for feature detection based on bipartite graph. In 2021 IEEE 37th International Conference on Data Engineering (ICDE), pages 2039–2044. IEEE, 2021.
[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.
[19] Petar Veličković, Guillaume Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[20] Zhiqin Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. arXiv preprint arXiv:2004.12848, 2019.
[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018.

[46] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pages 1365–1374, 2015.

[47] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In Advances in neural information processing systems, pages 4800–4810, 2018.

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems, 33, 2020.

[49] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In International Conference on Machine Learning, pages 10871–10880. PMLR, 2020.

[50] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[51] Fan Zhou, Qing Yang, Ting Zhong, Dajiang Chen, and Ning Zhang. Variational graph neural networks for road traffic prediction in intelligent transportation systems. IEEE Transactions on Industrial Informatics, 2020.

[52] Zhi-Hua Zhou. Ensemble learning. Encyclopedia of biometrics, 1:270–273, 2009.

[53] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. Artificial intelligence, 137(1-2):239–263, 2002.
6 APPENDIX ON REPRODUCIBILITY

In this section, we provide detailed information of the experimental settings and configurations to ensure the reproducibility of our work.

6.1 Dataset Details
The datasets tested for graph level representation learning are described in Table 6. The dataset used in the node-level representation learning task are listed in Table 7. We get the dataset from Pytorch Geometric library.

Table 6: Detailed statistics of datasets for graph-level representation learning

| Datasets | Category       | Classes | Graph Num. | Avg. Node | Avg. Degree |
|----------|----------------|---------|------------|-----------|-------------|
| NCI1     | Biochemical Molecules | 2       | 4110       | 29.87     | 1.08        |
| PROTEINS | Biochemical Molecules | 2       | 1113       | 39.06     | 1.86        |
| D&D      | Biochemical Molecules | 2       | 1178       | 284.32    | 715.66      |
| MUTAG    | Biochemical Molecules | 2       | 188        | 17.93     | 19.79       |
| COLLAB   | Social Networks   | 3       | 5000       | 74.49     | 32.99       |
| RDT-BINARY | Social Networks   | 2       | 2000       | 429.63    | 1.15        |
| RDT-MULTI-5K | Social Networks   | 5       | 4999       | 508.52    | 594.87      |
| IMDB-BINARY | Social Networks   | 2       | 1000       | 19.77     | 96.53       |
| IMDB-MULTI | Social Networks   | 3       | 1500       | 13.00     | 65.94       |

Table 7: Detailed statistics of datasets for node-level representation learning

| Dataset  | Nodes | Edges | Classes | Attributes |
|----------|-------|-------|---------|------------|
| Cora     | 3,327 | 4,732 | 7       | 1,433      |
| Citeseer | 19,717| 44,338| 3       | 500        |

6.2 Experiments Environment
We run the experiments on a Dell PowerEdge T630 Server with 2 20-core Intel CPUs and 256GB memory and the other Server with 3 GTX-1080 ti GPUs. All models are implemented in Pytorch 1.4.0, torch-geometric 1.6.0, cudatoolkit 10.1.243, and scikit-learn 0.23.1. Our code is available at: Anonymous link. The other version of code with a detailed README and execution script will be available soon.

6.3 Settings of graph-level representation learning

6.3.1 Unsupervised graph-level representation learning. The detailed hyper-parameters used in unsupervised graph representation learning are shown in Table 8. The parameters are the same for any graph encoder assembly in the same dataset. For the classifier SVM, it searches the parameter “C” in \{1, 10, 100, 1000, 10000\}.

6.3.2 Pretrain-Finetune graph-level representation learning. The pretrain process uses the same hyper-parameters listed in Table 8. In finetuning step, we finetune the model with Adam optimizer, which learning rate is 0.0005 and weight decay is 0.0001. We optimize the model 120 steps. All datasets use the same hyper-parameters.

6.4 Settings of node-level representation learning

The detailed hyper-parameters used in unsupervised node representation learning are shown in Table 9. The classifier used in this task is MLP.

6.5 Visualization of node representations

The visualization of representations on the Cora dataset is shown in Figure 5. The results reflected on the Cora dataset are consistent with the results on CiteSeer shown in Figure 3. Nodes with the same label in CGCL is not as dense as GCN, but the boundaries of the communities where the same label nodes exist are clearer than DGI.
Figure 5: The learned node representations on the Cora dataset visualized by t-SNE [4]

Table 8: Hyper-parameters for unsupervised graph-level representation learning

| Dataset   | random seed | learning rate | weight decay | hidden size | dropout ratio | batch size | temperature | node attribute |
|-----------|-------------|---------------|--------------|-------------|---------------|------------|--------------|----------------|
| PROTEINS  | 888         | 0.01          | 0.001        | 128         | 0.5           | 256        | 0.07         | Default        |
| D&D       | 888         | 0.05          | 0.0001       | 128         | 0.5           | 256        | 0.01         | Default        |
| MUTAG     | 888         | 0.001         | 0.0001       | 128         | 0.5           | 256        | 0.07         | Default        |
| COLLAB    | 888         | 0.01          | 0.0001       | 128         | 0.5           | 256        | 0.3          | Constant       |
| RDT-BINARY| 888         | 0.05          | 0.0001       | 128         | 0.5           | 256        | 0.1          | Constant       |
| RDT-MULTI-5K | 888   | 0.05          | 0.0001       | 128         | 0.5           | 256        | 0.1          | Constant       |
| IMDB-BINARY| 888       | 0.05          | 0.0001       | 128         | 0.5           | 256        | 0.5          | OneHotDegree   |
| IMDB-MULTI| 888         | 0.05          | 0.0001       | 128         | 0.5           | 256        | 0.1          | OneHotDegree   |

Table 9: Hyper-parameters for unsupervised node-level representation learning

| Dataset   | random seed | learning rate | weight decay | hidden size | dropout ratio | temperature |
|-----------|-------------|---------------|--------------|-------------|---------------|-------------|
| Cora      | 123         | 0.001         | 0.0001       | 64          | 0.5           | 0.5         |
| CiteSeer  | 123         | 0.01          | 0.0001       | 64          | 0.5           | 0.8         |
| PubMed    | 123         | 0.01          | 0.0001       | 64          | 0.5           | 0.5         |