Graph Communal Contrastive Learning

Bolian Li
Tianjin University
Tianjin, China
libolian@tju.edu.cn

Baoyu Jing
University of Illinois at Urbana-Champaign
IL, USA
baoyuj2@illinois.edu

Hanghang Tong
University of Illinois at Urbana-Champaign
IL, USA
htong@illinois.edu

ABSTRACT
Graph representation learning is crucial for many real-world applications (e.g. social relation analysis). A fundamental problem for graph representation learning is how to effectively learn representations without human labeling, which is usually costly and time-consuming. Graph contrastive learning (GCL) addresses this problem by pulling the positive node pairs (or similar nodes) closer while pushing the negative node pairs (or dissimilar nodes) apart in the representation space. Despite the success of the existing GCL methods, they primarily sample node pairs based on the node-level proximity yet the community structures have rarely been taken into consideration. As a result, two nodes from the same community might be sampled as a negative pair. We argue that the community information should be considered to identify node pairs in the same communities, where the nodes insides are semantically similar. To address this issue, we propose a novel Graph Communal Contrastive Learning (gCool) framework to jointly learn the community partition and learn node representations in an end-to-end fashion. Specifically, the proposed gCool consists of two components: a Dense Community Aggregation (DeCA) algorithm for community detection and a Reweighted Self-supervised Cross-contrastive (ReSC) training scheme to utilize the community information. Additionally, the real-world graphs are complex and often consist of multiple views. In this paper, we demonstrate that the proposed gCool can also be naturally adapted to multiplex graphs. Finally, we comprehensively evaluate the proposed gCool on a variety of real-world graphs. The experimental results show that the gCool outperforms the state-of-the-art methods.

CCS CONCEPTS
• Computing methodologies → Unsupervised learning; Learning latent representations; • Mathematics of computing → Information theory; Graph algorithms.

KEYWORDS
self-supervised learning, graph contrastive learning, community detection

Permission to make digital or hard copies of all or part 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 components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:
Bolian Li, Baoyu Jing, and Hanghang Tong. 2022. Graph Communal Contrastive Learning. In Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512208

1 INTRODUCTION
Graph representation learning is crucial in many real-world applications, including predicting friendship in social relationship [76], recommending products to potential users [31] and detecting social opinions [60, 61]. However, traditional graph representation learning methods [19, 28, 58] demand labeled graph data of high quality, while such data is too expensive to obtain and sometimes even unavailable due to privacy and fairness concerns [25, 26]. To address this problem, recent Graph Contrastive Learning (GCL) [15, 16, 28, 33, 49, 71, 78] proposes to train graph encoders by distinguishing positive and negative node pairs without using external labels. Among them, node-level GCL methods regard all node pairs as negative samples, which pulls closer positive node pairs (representations of the same node in two views) and pushes away negative node pairs (representations of different nodes) [64]. The variants of GCL [16, 56, 72] employ mutual information maximization to contrast node pairs and have achieved state-of-the-art performance on many downstream tasks, such as node classification [2], link prediction [34] and node clustering [52].

Despite the success of these methods, most of the existing GCL methods primarily focus on the node-level proximity but fail to explore the inherent community structures in the graphs when sampling node pairs. For example, MVGRL [16], GCA [78] and GraphCL [71] regard the same nodes in different graph views as positive pairs; [16] regards the neighbors as positive node pairs; DGI [59] and HDI [21] produce negative pairs by randomly shuffling the node attributes; [14, 49, 74] regard negative samples as different sub-graphs. However, the structural information such as the community structures in the graphs has not been fully explored yet. The community structures can be found in many real graphs [66]. For example, in social graphs, people are grouped by their interests [53]. The group of people with the same interest tend to be densely connected by their interactions, while the people with different interests are loosely connected. Therefore, the people in the same interest community are graphically similar and treating them as negative pairs will introduce graphic errors to the node representations. To address this problem, we firstly propose a novel dense community aggregation (DeCA) algorithm to learn community partition based on structural information in the graphs. Next, we introduce a novel reweighted self-supervised contrastive (ReSC) training scheme to pull the nodes within the same community closer to each other in the representation space.
Additionally, in the real graphs, nodes can be connected by multiple relations [22, 24, 44, 69], and each relation can be treated as a graph view (known as multiplex graphs [10]). One way for graph representation learning methods to be adapted to the multiplex graphs is to learn the representations of each graph view separately and then combining them with a fusion model [21]. However, this approach ignores the graphic dependence between different graph views. For example, IMDB [62] can be modeled as a multiplex movie graph connected by both co-actor and co-director relations. These two kinds of connections are not independent since personal relations between actors and directors would affect their participation in movies. To address this problem, we apply contrastive training that is consistent with the situation of the single-relation graph, applying pair-wise contrastive training on each pair of graph views and using a post-ensemble fashion to combine model outputs of all views in the down-stream tasks.

The main contributions of this paper are summarized as follows:

- We propose a novel Dense Community Aggregation (DeCA) algorithm to detect structurally related communities by end-to-end training.
- We propose a Rewighted Self-supervised Cross-contrastive (ReSC) training scheme, which employs community information to enhance the performance in down-stream tasks.
- We adapt our model to multiplex graphs by pair-wise contrastive training, considering the dependence between different graph views.
- We evaluate our model on various real-world datasets with multiple down-stream tasks and metrics. Our model achieves the state-of-the-art performance.

The rest of the paper is organized as: preliminaries (Sec. 2), methodology (Sec. 3), experiments (Sec. 4), related works (Sec. 5) and conclusion (Sec. 6).

2 PRELIMINARIES

In this section, we introduce the similarity measurement used in this paper (Sec. 2.1), the basic concepts of contrastive learning (Sec. 2.2) and community detection (Sec. 2.3). Then we give a brief definition of attributed multiplex graphs in Sec. 2.4. The notations used in this paper are summarized in Table 1.

2.1 Similarity Measurement

We adopt two classic similarity functions to determine the similarity between two nodes: exponent cosine similarity [50]

\[
\delta_c(x_1, x_2) = \exp \left\{ \frac{x_1^T x_2}{\|x_1\| \|x_2\|} \right\}, \tag{1}
\]

in which we exponentiate the cosine similarity for non-negativity, and Gaussian RBF similarity [3]

\[
\delta_c(x_1, x_2) = \exp \left\{ -\|x_1 - x_2\| ^2 / \tau^2 \right\}, \tag{2}
\]

where \(\tau\) is the sensitivity factor.

2.2 Contrastive Learning

Contrastive learning aims at distinguishing similar (positive) and dissimilar (negative) pairs of data points, encouraging the agreement of the similar pairs and the disagreement of the dissimilar ones [8]. The general process dating back to [1] can be described as: For a given data point \(x\), let \(x^1\) and \(x^2\) be differently augmented data points derived from \(x\). \(z^1\) and \(z^2\) are the representations by passing \(x^1\) and \(x^2\) through a shared encoder. The mutual information [9] (or its variants such as [43]) between the two representations is maximized. The motivation of contrastive learning is to learn representations that are invariant to perturbation introduced by the augmentation schemes [65], and thus are robust to inherent noise in the dataset.

A widely used objective for contrastive learning is InfoNCE [43], which applies instance-wise contrastive objective by sampling the same instances in different views as positive pairs and all other pairs between different views as negative pairs. The limitation of InfoNCE is that it uniformly samples negative pairs, ignoring the semantic similarities between some instances.

2.3 Community Detection

Communities in graphs are groups of nodes that are more strongly connected among themselves than others. Community structures widely exist in many real-world graphs, such as sociology, biology and transportation systems [41]. Detecting communities in a graph has become a primary approach to understand how structure relates to high-order information inherent to the graph [20].
### Table 1: Notations.

| Symbol | Description |
|--------|-------------|
| $G$    | attributed graph |
| $\mathcal{G}$ | attributed multiplex graph |
| $E$    | edge set |
| $\tilde{E}$ | augmented edge set |
| $X$    | attribute matrix |
| $\tilde{X}$ | augmented attribute matrix |
| $X'$   | attribute matrix of view $r$ |
| $A$    | adjacency matrix |
| $\tilde{A}$ | extended adjacency matrix |
| $C_r$  | community |
| $\Phi$ | community centroid matrix |
| $R$    | community assignment matrix |
| $F$    | community density matrix |

### 2.3.1 Formulation.

For a graph $G = (V, E)$, with its node set $V = \{v_1, v_2, ..., v_M\}$ and edge set $E = \{(v_i, v_j), (v_i, v_j), ..., (v_M, v_M)\}$, we define one of its communities $C$ as a sub-graph: $C = (V^C, E^C)$, where $V^C \subseteq V$ and $E^C = E \cap (V^C \times V^C)$.

### 2.3.2 Modularity.

A widely used evaluation for community partitions is the modularity [42], defined as:

$$ m = \frac{1}{2M} \sum_{i,j} A[i, j] \frac{d[i, j]}{2M} \left[ r(i, j) \right] $$

where $A$ is the adjacency matrix, $d[i]$ is the degree of the $i$-th node, and $r(i, j)$ is 1 if nodes $i$ and $j$ are in the same community, otherwise is 0. The modularity measures the impact of each edge to the local edge density ($d[i, j]/2M$ represents the expected local edge density), and is easily disturbed by the changing of edges.

### 2.3.3 Other notations.

We define the edge count function over the adjacency matrix:

$$ E(C) = \sum_{i,j} \left[ A^C[i, j] \neq 0 \right] $$

where $A^C$ is the adjacency matrix for community $C$. The edge density function compares the real edge count to the maximal possible number of edges in the given community $C_k$:

$$ d(k) = \frac{E(C_k)}{|C_k|(|C_k| - 1)} $$

### 2.4 Attributed Multiplex Graph

Multiplex graphs are also known as multi-dimensional graphs [39] or multi-view graphs [12, 23, 55], and are comprised of multiple single-view graphs with shared nodes and attributes but different graph structures (usually with different types of link) [6]. Formally, an attributed multiplex graph is $\mathcal{G} = \{G^1, G^2, ..., G^R\}$, where $R \in \mathbb{N}_+$ and each $G^r = (V, E^r)$ is an attributed graph. If the number of views $R = 1$, $\mathcal{G} = \{G^1\}$ is equivalent to the attributed graph $G^1$.

We show an example of attributed multiplex graph in Fig. 2.

### 3 METHODOLOGY

In this section, as important components of the proposed $\phi$CoolL model, we first introduce Dense Community Detection ($\text{DeCA}$) algorithm in Sec. 3.1, and then in Sec. 3.2, we propose the Reweighted Self-supervised Cross-contrastive ($\text{ReSC}$) training scheme. Further discussion on adaption to multiplex graphs is presented in Sec. 3.3.

#### 3.1 Dense Community Aggregation

Node-level GCL methods are vulnerable to the problem caused by pairing structurally close nodes as negative samples. For this issue, we propose a novel Dense Community Aggregation (DeCA) for detecting structurally related communities of a single graph view. Our method is inspired by the modularity [42] in graphs, which measures the local edge density in communities. However, modularity is easily perturbed by variations of edges [36], which limits its robustness in detecting communities (with edges dynamically changing in each epochs). Therefore, our goal is to enhance the robustness of modularity, and to further extend the modularity by maximizing edge density in each community, while minimizing the edge density between different communities. The DeCA is carried out by end-to-end training and is illustrated in Fig. 3.

![Figure 2: An example of attributed multiplex graph. It consists of two views with shared node attributes and different sets of edges.](image)

![Figure 3: Dense Community Aggregation. Each node is softly assigned to community centroids with probabilities, and the community centroids are learned in end-to-end training with Eq. 10.](image)
First, we assign each node in the graph softly to community centroids with probabilities (computed by similarity function Eq. 1 or Eq. 2). Specifically, we define a community assignment matrix \( R \), where each \( R[i, :] \) is a normalized similarity vector measuring the distance between the \( i \)-th node and all community centroids. Formally, \( R \) is computed by

\[
R = \text{normalize}(\delta(f_\theta(X, A), \Phi)),
\]
where \( \delta(\cdot) \) is the similarity function defined in Sec. 2.1, \( f_\theta(\cdot) \) is the graph encoder with parameters \( \theta \), and \( \text{normalize}(\cdot) \) normalizes the probability of each edge by dividing it by the sum of all probabilities and holds \( \sum_j R[i, j] = 1 \) for each \( i \).

Second, we employ two objectives for training community partition (see Table 1 and Sec. 2.3.3 for symbol definitions): intra-community density

\[
D_{\text{Intra}} = \frac{1}{N} \sum_{i,j,k} [A[i, j] - \hat{d}(k)] R[i, k] R[j, k],
\]
and inter-community density

\[
D_{\text{Inter}} = \frac{1}{N(N-1)} \sum_{i,j,k_1,k_2} A[i, j] R[i, k_1] R[j, k_2].
\]

These two objectives measure the impact of each edge on the community edge density (rather than the local edge density used in modularity [42]). Specifically, in Eq. 7 and Eq. 8, the term \( A[i, j] - \hat{d}(k) \) and \( A[i, j] - \hat{d}(k) \) represent the gap between real local density \( (A[i, j]) \) and the expected density \( (\hat{d}(k) \) for intra-community and \( \hat{d}(k) \) for inter-community). The centroid matrix \( \Phi \) will be updated to reach a reasonable community partition, by minimizing the joint objective:

\[
L_{\text{DeCA}}(R) = \lambda_w D_{\text{Inter}} - D_{\text{Intra}},
\]
where \( \lambda_w \) is the co-efficient. Moreover, for computational efficiency, we slightly modify the form of \( L_{\text{DeCA}} \) in our actual implementation (shown in Appendix C). Finally, we compute the \( L_{\text{DeCA}} \) objectives for two graph views, and carry out dense community aggregation simultaneously for them:

\[
L_{\text{DeCA}} = \frac{1}{2} \left[ L_{\text{DeCA}}(R^1) + L_{\text{DeCA}}(R^2) \right].
\]

### 3.2 Reweighted Self-supervised Cross-contrastive Training

We propose the Reweighted Self-supervised Cross-contrastive (ReSC) training scheme in this section. We firstly apply graph augmentations to generate two graph views and secondly apply node contrast and community contrast simultaneously to consider both node-level and community-level information. We introduce node-community pairs as additional negative samples to address the problem of pairing nodes in the same communities as negative samples.

#### 3.2.1 Graph augmentation

We define the graph augmentation \( T \) as randomly masking attributes and edges with a specific probability [78]. First, we define the attribute masks as random noise vector \( m \), where each dimension of it is independently drawn from a Bernoulli distribution: \( m[i] \sim \text{Bernoulli}(1 - p_o) \). The augmented attribute matrix is computed by

\[
\tilde{X} = [X[1, :] \odot m; X[2, :] \odot m; \ldots; X[N, :] \odot m]^T,
\]
where \( \odot \) is the Hadamard product [18] and \( X \) is the original attribute matrix. Second, we generate the augmented edge set \( \tilde{E} \) by randomly dropping edges from the original edge set \( E \) [77] with probability

\[
P\left( (v_1, v_2) \in \tilde{E} \right) = 1 - p_e, \forall (v_1, v_2) \in E.
\]

We apply the above graph augmentations (denoted as \( t^1, t^2 \sim T \) respectively for 2 independent augmentations) to generate two graph views: \( (X^1, A^1) = t^1(X, A) \) and \( (X^2, A^2) = t^2(X, A) \) for \( t^1, t^2 \sim T \). Then, we obtain their representations by a shared GCN encoder: \( Z^1 = f_\theta(X^1, A^1) \) and \( Z^2 = f_\theta(X^2, A^2) \).

#### 3.2.2 Node contrast

After generating two graph views, we employ node contrast and community contrast simultaneously to learn the node representations. We introduce a contrastive loss based on the InfoNCE [43] to contrast node-node pairs:

\[
I_{\text{InfoNCE}}(Z^1; Z^2) = -\log \sum_i \delta(Z[i, :], Z^2[i, :]), \quad (13)
\]

We apply the node contrast symmetrically for the two graph views:

\[
L_{\text{node}} = \frac{1}{2} \left[ I_{\text{InfoNCE}}(Z^1; Z^2) + I_{\text{InfoNCE}}(Z^2; Z^1) \right], \quad (14)
\]

which distinguishes negative pairs in the two views, and enforces maximizing the consistency between positive pairs [35].

#### 3.2.3 Community contrast

The community contrast is based on the result of DeCA in Sec. 3.1. First, we obtain the community centers by training the randomly initialized community centroids matrix \( \Phi \) with Eq. 10. Second, we adopt a re-weighted cross-contrastive objective to contrast node representations of one view with the community centroids of the other view (a cross-contrastive fashion). We are inspired by the cross-prediction scheme in [4] and introduce the cross-contrast into InfoNCE objective [43] to maximize the community consistency between two views, enforcing that the node representations in one community are pulled close to those of the counterpart community in the other view. Formally, the community contrast is computed by

\[
I_{\text{com}}(Z, \Phi) = -\log \sum_i \delta(Z[i, :], \Phi[k_i, :]) + \sum_{k, e} \sum_{k_i, k_e} w(i, k) \cdot \delta(Z[i, :], \Phi[k, :]), \quad (15)
\]

where the \( i \)-th node is assigned to the \( k_i \)-th community. Here, \( w(i, k) = \exp \left\{ -\gamma ||Z[i, :] - \Phi[k, :]||^2 \right\} \) is the RBF weight function (slightly different from Gaussian RBF similarity in Eq. 2), which gives more attention to the more similar community pairs. In this objective, the similarity of node representations within the same communities are maximized since they are contrasted positively to the same centroids, while in different communities, node representations are separated by negative contrast. Similarly, we compute the contrastive objective symmetrically for the two generated graph views:

\[
L_{\text{com}} = \frac{1}{2} \left[ I_{\text{com}}(Z^1, \Phi^2) + I_{\text{com}}(Z^2, \Phi^1) \right], \quad (16)
\]

where \( Z^1 \) and \( \Phi^2 \) are the node representation matrices of view 1 and view 2 respectively.
3.2.4 Joint objective. We propose the $\alpha$-decay coefficient to combine $L_{\text{node}}$, $L_{\text{DeCA}}$ and $L_{\text{com}}$ into a joint objective:

$$L = L_{\text{node}} + \alpha(t)L_{\text{DeCA}} + [1 - \alpha(t)]L_{\text{com}}. \quad (17)$$

where the coefficient $\alpha(t) = \exp \left(-t/\eta\right)$ would decay smoothly with the proceeding of training ($t$ refers to the epoch). We observe that the community partition would be stabilized within a few hundreds of epochs by DeCA training (see Sec. 4.3), while the training of the gCoolL model usually takes thousands of epochs. To this end, we apply the $\alpha$-decay to focus mainly on training community partition at first, and gradually divert the focus to learning node representations.

In summary, the ReSC training process is shown in Algorithm 1.

### Algorithm 1: The ReSC training.

**Inputs:** attribute matrix $X$, adjacency matrix $A$;  
**Output:** graph encoder $f_\theta()$;  
**for** $t = 1,2,......$ **do**  
  Sample two random graph augmentations $t^1, t^2 \sim T$;  
  Generate two graph views, $(X^1, A^1) = t^1(X, A)$ and $(X^2, A^2) = t^2(X, A)$;  
  Encode the two views by a shared GCN encoder, $Z^1 = f_\theta(X^1, A^1)$ and $Z^2 = f_\theta(X^2, A^2)$;  
  Obtain $L_{\text{node}}, L_{\text{DeCA}}$ and $L_{\text{com}}$ to compute the joint loss: $L = L_{\text{node}} + \alpha(t)L_{\text{DeCA}} + [1 - \alpha(t)]L_{\text{com}}$;  
  Update parameters $\theta$ by stochastic gradient descent on joint objective $L$.  
**end**

3.3 Adaptation to Multiplex graphs

The proposed gCoolL framework can be naturally adapted to multiplex graphs with a few modifications on the training and inference process. We apply contrastive training between different graph views, which consider the dependence between them.

3.3.1 Training. In multiplex graphs, we no longer need to generate graph views through graph augmentation, since the different views in a multiplex graph are naturally multi-viewed data. We propose to detect communities (DeCA) and learn node representations (ReSC) on each pair of views. The modified training process is shown in Algorithm 2.

### Algorithm 2: The ReSC training on multiplex graphs.

**Inputs:** attribute matrix $X$, adjacency matrices $A^1,...,A^F$;  
**Output:** graph encoder $f_\theta()$;  
**for** $t = 1,2,......$ **do**  
  Encode the $R$ graph views by a shared GCN encoder, $Z^1 = f_\theta(X^1, A^1)$, $Z^2 = f_\theta(X^2, A^2)$, ......  
  Obtain $L_{\text{node}}, L_{\text{DeCA}}$ and $L_{\text{com}}$ to compute the joint loss for each pair of views ($G^i, G^j$) such that $i \neq j$;  
  Update parameters $\theta$ by stochastic gradient descent on joint objective $L$ in Eq. 17 (adding up objectives of each pair of views).  
**end**

4.1 Experimental Setup

4.1.1 Datasets. We use 6 real graphs, including Amazon-Computers, Amazon-Photo, Coauthor-CS, WikiCS, IMDB and DBLP, to evaluate the performance on node classification and node clustering. The detailed statistics of all datasets are summarized in Table 2.

- **Amazon-Computers** and **Amazon-Photo** [54] are two graphs of co-purchase relations, where nodes are products and there exists an edge between two products when they are bought together. Each node has a sparse bag-of-words attribute vector encoding product reviews.
- **Coauthor-CS** [54] is an academic network based on the Microsoft Academic Graph. The nodes are authors and edges are co-author relations. Each node has a sparse bag-of-words attribute vector encoding paper keywords of the author.
- **WikiCS** [40] is a reference network from Wikipedia. The nodes are articles and edges are hyperlinks between the articles. Node attributes are computed as the average of GloVe [47] word embedding of articles.
- **IMDB** [62] is a movie network with two types of links: co-actor and co-director. The attribute vector of each node is a bag-of-words feature of movie plots. The nodes are labeled with Action, Comedy or Drama.
- **DBLP** [62] is a paper graph with three types of links: co-author, co-paper and co-term. The attribute vector of each node is a bag-of-words feature vector of paper abstracts.

| Dataset          | Type       | Edges   | Nodes   | Attributes | Classes |
|------------------|------------|---------|---------|------------|---------|
| Amazon-Computers | co-purchase| 245,778 | 13,381  | 767        | 10      |
| Amazon-Photo     | co-purchase| 119,043 | 7,487   | 745        | 8       |
| Coauthor-CS      | co-author  | 81,894  | 18,333  | 6,805      | 15      |
| WikiCS           | reference  | 216,123 | 11,701  | 300        | 10      |
| IMDB             | co-actor   | 66,428  | 13,788  | 3,550      | 2,000   | 3       |
| DBLP             | co-paper   | 144,738 | 90,145  | 7,907      | 2,000   | 4       |
4.1.2 Evaluation protocol. We follow the evaluations in [78], where each graph encoder is trained in a self-supervised way. Then, the encoded representations are used to train a classifier for node classification, and fit a K-means model for comparing baselines on node clustering. We train each graph encoder for five runs with randomly selected data splits (for WikiCS, we use the given 20 data splits) and report the average performance with the standard deviation.

For the node classification task, we measure the performance with Normalized Mutual Information (NMI) score: $\text{NMI} = 2I(\hat{Y}; Y)/(H(\hat{Y}) + H(Y))$, where $\hat{Y}$ and $Y$ refer to the predicted cluster indexes and class labels respectively, and $\text{ARI}$: $\text{ARI} = RI - \mathbb{E}[RI]/(\max{RI} - \mathbb{E}[RI])$, where $RI$ is the Rand Index [51], which measures the similarity between two cluster indexes and class labels.

4.1.3 Baselines. We compare the gCool methods (gCool$G$ for exponent cosine similarity and gCool$E$ for Gaussian RBF similarity) with two sets of baselines:

- **On single-view graphs**, we consider baselines as 1) traditional methods including node2vec [13] and DeepWalk [48], 2) supervised methods including GCN [28], and 3) unsupervised methods including MVGRL [16], DGI [59], HDI [21], graph autoencoders (GAE and VGAE) [29] and GCA [78].

- **On multiplex graphs**, we consider baselines as 1) methods with single-view representations including node2vec [13], DeepWalk [48], GCN [28] and DGI [59], and 2) methods with multi-view representations including CMNA [7], MNE [70], HAN [62], DMI [44] and HDMI [21]. Additionally, we compare different clustering baselines including K-means, Spectral biclustering (SBC) [30] and modularity [42] to show the effectiveness of our proposed DeCA (DeCA$G$ for exponent cosine similarity and DeCA$E$ for Gaussian RBF similarity).

4.1.4 Implementation. We use a 2-layer GCN [28] as the graph encoder for each deep learning baseline. We use Adam optimizer [27] to optimize each model. Before applying the ReSC objective, we pass the representations of two graph views through a shared projection module (a 2-layer MLP) with the same input, hidden-layer and output dimensions. The number of communities is set according to the number of classes in each graph. More implementation details are listed in Appendix A.

4.2 Quantitative Results

4.2.1 Overall performance. We present the quantitative results of node classification on single-view graphs (Table 3), node clustering on single-view graphs (Table 4), and node classification on multiplex graphs (Table 5). gCool outperforms the state-of-the-art methods on multiple tasks, and even outperforms the supervised methods. Moreover, the proposed DeCA (used to cluster nodes based on its community partition) outperforms traditional clustering methods and the former modularity [42] on encoded representations (shown in Table 6), which illustrates that DeCA is more suitable for deep-learning-based clustering and addresses the instability of modularity.

4.2.2 Ablation study. We compare the different combinations of $L_{\text{node}}$ (Eq. 14), $L_{\text{DeCA}}$ (Eq. 10) and $L_{\text{com}}$ (Eq. 16) on WikiCS in terms of node classification and node clustering. The results are listed in Table 7. We find that, first, applying node contrast is crucial to the performance (given that the combinations with $L_{\text{node}}$ outperform others) since it makes the representations more distinguishable, second, community contrast is beneficial to the overall performance (given that $L_{\text{node}} + L_{\text{com}}$ outperforms $L_{\text{node}}$) since it considers community information in contrastive training, and third, the performance of community contrast is boosted significantly when it is used with DeCA (comparing $L_{\text{com}}$ with $L_{\text{DeCA}} + L_{\text{com}}$). Moreover, we also show the visual ablation study in Table 8.

4.3 Visual Evaluations

We illustrate the significance of DeCA by visualizing the edge density and class entropy of the assigned communities. We evaluate each checkpoint for five times and show the means and deviations. We compare the results with traditional clustering methods (K-means and Spectral biclustering [30]) and the former modularity [42]. We also visualize the node representations for ablation study.

4.3.1 Edge density. The edge density is based on Eq. 5 and computed by the average density of all communities:

$$ED = \frac{1}{K} \sum_{k=1}^{K} d(k).$$

It is used to measure how DeCA learns the community partition that maximizes the intra-community density (see Sec. 3.1). Fig. 4 shows that, after a few hundreds of epochs, DeCA stably outperforms other clustering methods.

![Figure 4: Visualization of edge density comparison.](image-url)
Table 3: Overall performance on node classification (in percentage).

| Dataset          | Amazon-Computers | Amazon-Photo | Coauthor-CS | WikiCS |
|------------------|------------------|-------------|-------------|--------|
|                  | Micro-F1         | Macro-F1    | Micro-F1    | Macro-F1 | Micro-F1 | Macro-F1 |
| RawFeatures      | 73.82±0.01       | 70.10±0.04  | 78.45±0.04  | 76.10±0.01 | 90.40±0.02 | 89.01±0.06 |
| Node2vec         | 84.38±0.08       | 82.63±0.08  | 89.72±0.08  | 87.39±0.07 | 85.11±0.06 | 82.93±0.11 |
| DeepWalk         | 85.63±0.09       | 84.02±0.10  | 89.36±0.10  | 86.92±0.02 | 84.71±0.23 | 82.63±0.19 |
| GCA              | 82.51±0.23       | 90.47±0.32  | 91.74±0.09  | 89.93±0.09 | 93.04±0.28 | 91.02±0.38 |
| MVGRL            | 83.88±0.50       | 79.30±0.42  | 91.60±0.24  | 89.31±0.16 | 92.08±0.68 | 90.78±0.68 |
| DGI              | 85.43±0.13       | 80.74±0.25  | 90.09±0.10  | 88.70±0.16 | 89.98±0.14 | 86.73±0.17 |
| GAE              | 85.18±0.21       | 83.33±0.17  | 91.68±0.14  | 89.66±0.09 | 90.00±0.75 | 88.31±0.68 |
| VGE              | 82.24±0.08       | 90.04±0.17  | 92.39±0.21  | 91.05±0.13 | 92.08±0.08 | 90.11±0.06 |
| GCA              | 86.44±0.25       | 83.72±0.12  | 93.18±0.12  | 92.05±0.17 | 93.32±0.02 | 91.65±0.03 |
| gCool_e          | 88.83±0.14       | 87.42±0.28  | 92.79±0.17  | 91.57±0.29 | 93.31±0.01 | 91.63±0.03 |
| gCool_e          | 88.74±0.09       | 87.53±0.26  | 93.18±0.12  | 92.05±0.17 | 93.32±0.02 | 91.65±0.03 |

Table 4: Overall performance on node clustering.

| Dataset          | Amazon-Computers | Amazon-Photo | Coauthor-CS | WikiCS |
|------------------|------------------|-------------|-------------|--------|
|                  | NMI              | ARI         | NMI         | ARI    |
|                  | ±                 |              | ±           |        |
| GCA              | 0.74±0.040       | 0.29±0.000  | 0.37±0.030  | 0.26±0.030 |
| DGI              | 0.34±0.014       | 0.30±0.000  | 0.42±0.014  | 0.30±0.000 |
| HAN              | 0.65±0.010       | 0.49±0.000  | 0.53±0.040  | 0.49±0.000 |
| GA              | 0.44±0.000       | 0.52±0.000  | 0.63±0.000  | 0.52±0.000 |
| VGE              | 0.42±0.000       | 0.44±0.000  | 0.61±0.000  | 0.44±0.000 |
| GCA              | 0.42±0.000       | 0.42±0.000  | 0.61±0.000  | 0.42±0.000 |
| gCool_L_e        | 0.47±0.018       | 0.27±0.024  | 0.63±0.000  | 0.52±0.000 |
| gCool_L_e        | 0.67±0.070       | 0.83±0.040  | 0.73±0.005  | 0.62±0.007 |

Table 5: Overall classification performance on multiplex graphs.

| Dataset          | IMDB             | DBLP            |
|------------------|------------------|-----------------|
|                  | Micro-F1         | Macro-F1        | Micro-F1    | Macro-F1    |
|                  | ±                 | ±               | ±           | ±           |
| HAN              | 0.60±0.070       | 0.71±0.071     | 0.70±0.070  | 0.71±0.071  |
| GCA              | 0.65±0.065       | 0.81±0.020     | 0.76±0.065  | 0.81±0.020  |
| DeepWalk         | 0.55±0.050       | 0.33±0.048     | 0.55±0.050  | 0.33±0.048  |
| GAE              | 0.61±0.061       | 0.71±0.074     | 0.61±0.061  | 0.71±0.074  |
| MVGRL            | 0.56±0.056       | 0.53±0.053     | 0.56±0.056  | 0.53±0.053  |
| DGI              | 0.57±0.057       | 0.56±0.056     | 0.57±0.057  | 0.56±0.056  |
| NME              | 0.60±0.059       | 0.71±0.076     | 0.60±0.059  | 0.71±0.076  |
| CMNMA            | 0.64±0.064       | 0.71±0.077     | 0.64±0.064  | 0.71±0.077  |
| DGI              | 0.65±0.065       | 0.81±0.020     | 0.65±0.065  | 0.81±0.020  |
| VGE              | 0.67±0.067       | 0.83±0.040     | 0.67±0.067  | 0.83±0.040  |
| gCool_L_e        | 0.67±0.070       | 0.83±0.040     | 0.67±0.070  | 0.83±0.040  |
| gCool_L_e        | 0.67±0.070       | 0.83±0.040     | 0.67±0.070  | 0.83±0.040  |

5 RELATED WORKS

In this section, we briefly review the related works, including graph contrastive learning (Sec. 5.1) and community detection (Sec. 5.2).

5.1 Graph Contrastive Learning

Graph contrastive learning [33] proposed to pull similar nodes into close positions in the representation space while pushing dissimilar nodes apart. [37, 64] comprehensively review the development of GCL. GRACE [77] proposed a simplified framework to maximize the agreement between two graph views. GCA [78] proposed an adaptive augmentation algorithm to generate graph views. GCC [49] designed a pre-training task to learn structural graph representations. HeCo [63] employed network schema and meta-path views.
In this paper, we propose a novel Graph Communal Contrastive Learning (gCool) framework to learn the community partition of structurally related communities by a Dense Community Aggregation (DeCA) algorithm, and to learn the graph representation with a reweighted self-supervised cross-contrastive (ReSC) training scheme considering the community structures. The proposed gCool consistently achieves the state-of-the-art performance on multiple tasks and can be naturally adapted to multiplex graphs. We show that community information is beneficial to the overall performance in graph representation learning.

5.2 Community Detection

Community detection is a primary approach to understand how structure relates to high-order information inherent to the graph. [20] reviewed the recent progress in community detection. CommlDGI [75] employed modularity for community detection. [32] introduced the identifiability of communities. RWM [38] used random walk to detect local communities. Destine [67, 68] detects dense subgraphs on multi-layered networks.

6 CONCLUSION

In this paper, we propose a novel Graph Communal Contrastive Learning (gCool) framework to learn the community partition of...
Fan-Yun Sun, Jordon Hoffman, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In International Conference on Learning Representations.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of machine learning research 9, 11 (2008).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In International Conference on Learning Representations.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In International Conference on Learning Representations.

Dong Wang, Jieun Li, Kaimin Xu, and Yizhen Wu. 2017. Sentiment community detection: exploring sentiments and relationships in social networks. Electronic Commerce Research 17, 1 (2017), 103–132.

Ruijie Wang, Zijie Huang, Shengzhong Liu, Huajie Shao, Dongxin Liu, Jinyang Li, Tianshi Wang, Dachun Sun, Shuochao Yao, and Tarek Abdelzaher. 2021. DyDiff-VAE: A Dynamic Variational Framework for Information Diffusion Prediction. 163–172.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In The World Wide Web Conference. 2022–2032.

Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised Heterogeneous Graph Neural Network with Co-contrastive Learning. arXiv preprint arXiv:2105.09111 (2021).

Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, Stan Li, et al. 2021. Self-supervised on Graphs: Contrastive, Generative, or Predictive. arXiv preprint arXiv:2105.07342 (2021).

Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. 2020. What Should Not Be Contrastive in Contrastive Learning. In International Conference on Learning Representations.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In International Conference on Learning Representations.

Zhe Xu. 2021. Dense subgraph detection on multi-layered networks. Ph.D. Dissertation.

Zhe Xu, Si Zhang, Yinglong Xia, Liang Xiong, Jejun Xu, and Hanghang Tong. 2021. DESTINE: Dense Subgraph Detection on Multi-Layered Networks. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 3558–3562.

Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic Knowledge Graph Alignment. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 4564–4572.

Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. 2019. Deep spectral clustering using dual autoencoder network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 4066–4075.

Yuning You, Tianlong Chen, Yongdao Sun, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems 33 (2020), 5812–5823.

Hongming Zhang, Liwei Qiu, Luqiang Yi, and Hanghang Tong. 2021. Scalable Multiplex Network Embedding. In Proceedings of the 39th ACM International Conference on Information & Knowledge Management. 1843–1852.

Tianqi Zhang, Yun Xiong, Jiawei Zhang, Yao Zhang, Yizhu Jiao, and Yangyong Zhu. 2020. CommDGI: Community Detection Oriented Deep Graph Infomax. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 143–152.

Jiang Zhu, Baowen Wu, Bin Wu, and Weiwu Zhu. 2017. Emotional community detection in social network. IEICE Transactions on Information and Systems 100, 10 (2017), 2515–2525.

Yanqiao Zhu, Yichen Xu, Peng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020).

Yanqiao Zhu, Yichen Xu, Peng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In Proceedings of the Web Conference 2021. 2069–2080.
A IMPLEMENTATION DETAILS
We conduct all of our experiments with PyTorch Geometric 1.7.2 [11] and PyTorch 1.8.1 [45] on a NVIDIA Tesla V100 GPU (with 32 GB available video memory). All single-viewed graph datasets used in experiments are available in PyTorch Geometric libraries, and all multiplex graph datasets are available in 6. Our hyperparameter specifications are listed in Table 9. Those hyperparameters are determined empirically by grid search based on the settings of [78].

Table 9: Hyperparameter specifications.

| Hyperparameter | Amazon-Computers | Amazon-Photo | Coauthor-CS | WikiCS | IMDB | DBLP |
|----------------|------------------|--------------|-------------|--------|------|------|
| Training epochs | 2900 /1400       | 600          | 50          | 50     | 100  | 100  |
| Hidden-layer dimension | 2558      | 256          | 7105        | 384    | 2048 | 128  |
| Representation dimension | 300        | 128          | 1368        | 2048   | 100  | 100  |
| Activation function | ReLU            | ReLU         | PReLU       | ReLU   | ReLU | ReLU |
| η               | 1000            | 500          | 500         | 50     | 50   | 50   |
| τ               | 0.2             | 0.3          | 0.4         | 0.4    | 1.0  | 1.0  |
| λw             | 0.9             | 0.9          | 1.0         | 1.0    | 0.5  | 0.5  |
| γ               | 2e-5            | 8e-5         | 8e-5        | 8e-5   | 5e-4 | 2e-4 |
| p_u             | 0.25            | 0.1          | 0.3         | 0.1    | 0.1  | 0.1  |
| p_e             | 0.45            | 0.4          | 0.2         | 0.2    | 0.0  | 0.0  |

B NUMBER OF COMMUNITIES
We compare varying numbers of communities in the vicinity of the numbers of classes on each single-viewed graph dataset (refer to Fig. 6). Although the performances remain stable for nearly all settings, we recommend to set the number of communities according to the actual number of classes.

Figure 6: Comparison of community number on node classification.

C PARALLELIZATION OF DENSE COMMUNITY AGGREGATION
In DeCA, the intra-community density objective is computationally costly and hard to vectorize since the expected edge density is dependent on the community choice. To address this problem, we derive a lower bound of $D_{intra}$:

$$D_{intra} \geq \frac{1}{N} \sum_{i,j,k} A[i,j] \cdot \max_k \{d(x)\} R[i,k] R[j,k]$$

where $A = \max \{d(x)\} I$ is the extended adjacency matrix, and $\overline{D}_{intra} = \inf D_{intra}$ is the lower bound. We use $\overline{D}_{intra}$ to replace $D_{intra}$ in Eq. 9.

Next, we employ the community density matrix $F = R'AR$ and $\overline{F} = R'\overline{A}R$ to vectorize the objective $D_{inter}$ and $\overline{D}_{intra}$. The entry of $F$ is $F[u,v] = \sum_i R[i,u] \cdot \langle A \rangle[i,v] = \sum_i A[i,j] R[i,u] R[j,v]$, which naturally accords with the form of Eq. 7 and Eq. 8. Therefore, these two objectives can be re-formalized as

$$\overline{D}_{intra} = \frac{1}{N} tr(\overline{F}),$$

and

$$D_{inter} = \frac{1}{N(N-1)} \sum_{i,j} F[i,j] - tr(F).$$

Finally, the vectorized DeCA objective is

$l_{DeCA}(R) = \lambda_w D_{inter} - \overline{D}_{intra}$

$$= \frac{\lambda_w}{N(N-1)} \sum_{i,j} F[i,j] - tr(F) - \frac{1}{N} tr(\overline{F}),$$

which can be directly computed by the CUDA operators [45].

---

1https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html
2https://www.dropbox.com/s/460e78bjp8ht151/data.tar.gz?dl=0