Graph InfoClust: Maximizing Coarse-Grain Mutual Information in Graphs

Costas Mavromatis(✉) and George Karypis

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA
{mavro016,karypis}@umn.edu

Abstract. This work proposes a new unsupervised (or self-supervised) node representation learning method that aims to leverage the *coarse-grain* information that is available in most graphs. This extends previous attempts that only leverage *fine-grain* information (similarities within local neighborhoods) or *global* graph information (similarities across all nodes). Intuitively, the proposed method identifies nodes that belong to the same clusters and maximizes their mutual information. Thus, coarse-grain (cluster-level) similarities that are shared between nodes are preserved in their representations. The core components of the proposed method are (i) a jointly optimized clustering of nodes during learning and (ii) an Infomax objective term that preserves the mutual information among nodes of the same clusters. Our method is able to outperform competing state-of-art methods in various downstream tasks, such as node classification, link prediction, and node clustering. Experiments show that the average gain is between 0.2% and 6.1%, over the best competing approach, over all tasks. Our code is publicly available at: https://github.com/cmavro/Graph-InfoClust-GIC.

1 Introduction

Graph structured data naturally emerge in various real-world applications. Such examples include social networks, citation networks, and biological networks. The challenge, from a data representation perspective, is to encode the high-dimensional, non-Euclidean information about the graph structure and the attributes associated with the nodes and edges into a low dimensional embedding space. The learned embeddings (a.k.a. representations) can then be used for various tasks, e.g., node classification, link prediction, community detection, and data visualization. In this paper, we focus on *unsupervised* (or self-supervised) representation learning methods that estimate node embeddings without using any labeled data but instead employ various self-supervision approaches. These methods eliminate the need to develop task-specific graph representation models, eliminate the cost of acquiring labeled data, and can lead to better representations by using large unlabeled datasets.

K. Karlapalem et al. (Eds.): PAKDD 2021, LNAI 12712, pp. 541–553, 2021.
https://doi.org/10.1007/978-3-030-75762-5_43

Various approaches have been developed to self-supervise graph representation learning. Many of them optimize the embeddings based on a loss function that ensures that pairs of nearby nodes are closer in the embedding space compared to pairs of distant nodes, e.g., DeepWalk [16] and GraphSAGE [4]. A key difference between such methods is whether they use graph neural network (GNN) encoders to compute the embeddings. These encoders insert an additional inductive bias that nodes share similarities with their neighbors and lead to significant performance gains.

Since GNN encoders already preserve similarities between neighboring nodes, Deep Graph Infomax (DGI) [21] uses a different loss function as self-supervision. This self-supervision encourages each node to be mindful of the global graph properties, in addition to its local neighborhood properties. Specifically, DGI maximizes the mutual information (MI) between the representation of each node (*fine-grain* representations) and the *global* graph representation, which corresponds to the summary of all node representations. DGI has shown to estimate superior node representations and is considered to be among the best unsupervised node representation learning approaches. However, in many graphs, besides their global structure, there is additional structure that can be captured. For example, nodes tend to belong to (multiple) clusters that represent topologically near-by nodes; or some nodes may share similar structural roles with topologically distant nodes. In such cases, methods that simultaneously preserve these *coarse-grain* interactions allow node representations to encode richer structural information.

Motivated by this observation, we developed *Graph InfoClust* (GIC), an unsupervised representation learning method that extracts coarse-grain information by identifying nodes that belong to the same clusters. Then, GIC learns node representations by maximizing the mutual information of nodes and their cluster-derived summaries, which preserves the coarse-grain information in the embedding space. Furthermore, since the node representations can help identify better clusters (both w.r.t. communities and also their role), we do not rely on an a priori clustering solution. Instead, the cluster-level summaries are obtained and jointly optimized by a differentiable K-means clustering [23] in an end-to-end fashion. We evaluated GIC on seven standard datasets using node classification, link prediction, and clustering as the downstream tasks. Our experiments show that in eleven out of thirteen dataset-task combinations, GIC performs better than the best competing approach and its average improvement over DGI is 0.9, 2.6, and 15.5% points for node classification, link prediction, and clustering, respectively. These results demonstrate that by leveraging cluster summaries, GIC is able to improve the quality of the estimated representations.

2 Notation, Definitions, and Problem Statement

Let $G := \{\mathcal{V}, \mathcal{E}\}$ denote a graph with N nodes and $|\mathcal{E}|$ edges, where $\mathcal{V} := \{v_1, \ldots, v_N\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. The connectivity is represented with the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, with $\mathbf{A}_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and

$\mathbf{A}_{i,j} = 0$, otherwise. Let $\mathbf{x}_i \in \mathbb{R}^F$ be the feature vector associated with node v_i and $\mathbf{X} \in \mathbb{R}^{N \times F}$ be the matrix that stores these features across all nodes. Here, we denote vectors by bold lower-case letters and matrices by bold upper-case letters. We also use the terms *representation* and *embedding* interchangeably.

Let $\mathbf{H} := [\mathbf{h}_1, \ldots, \mathbf{h}_N] \in \mathbb{R}^{N \times D}$ be the *node embedding matrix* of G, where $\mathbf{h}_i \in \mathbb{R}^D$ is the *node embedding vector* for v_i The goal of *node representation learning* is to learn a function (encoder), $f : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^{N \times D}$, such that $\mathbf{H} = f(\mathbf{A}, \mathbf{X})$. Once learned, $\mathbf{H}$ can be used as an input feature matrix to downstream tasks such as node classification, link prediction, and clustering.

3 Graph InfoClust (GIC)

3.1 Motivation and Overview

Preliminaries, DGI Framework. DGI employs a loss function that encourages node embeddings to contain information about the global graph properties. It does so by training the GNN-encoder to maximize the mutual information (MI) between the representation of each node $\mathbf{h}_i$ (fine-grain representation) and a summary representation $\mathbf{s} \in \mathbb{R}^D$ of the entire graph (global summary). Maximizing the precise value of mutual information is intractable; instead, DGI maximizes the Jensen-Shannon MI estimator that maximizes MI's lower bound [6]. This estimator acts like a binary cross-entropy (BCE) loss, whose objective maximizes the expected log-ratio of the samples from the joint distribution (positive examples) and the product of marginal distributions (negative examples). The positive examples are pairings of $\mathbf{s}$ with $\mathbf{h}_i$ of the real input graph $\mathcal{G} := (\mathbf{A}, \mathbf{X})$, but the negatives are pairings of $\mathbf{s}$ with $\tilde{\mathbf{h}}_i$, which are obtained from a fake/corrupted input graph $\tilde{\mathcal{G}} := (\tilde{\mathbf{A}}, \tilde{\mathbf{X}})$ (assuming the same number of nodes). The graph summary $\mathbf{s}$ is obtained by averaging all nodes' representations followed by the logistic sigmoid nonlinearity $\sigma(\cdot)$, as $\mathbf{s} = \sigma\left(\frac{1}{N}\sum_{k=1}^{N} \mathbf{h}_i\right)$. A discriminator $d : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ is used to assign higher scores to the positive than the negative examples. The Jensen-Shannon-based BCE objective is expressed as

$$\mathcal{L}_1 = \sum_{i=1}^{N} \mathbb{E}_{(\mathbf{X},\mathbf{A})}\Big[\log d(\mathbf{h}_i, \mathbf{s})\Big] + \sum_{i=1}^{N} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\Big[\log\big(1 - d(\tilde{\mathbf{h}}_i, \mathbf{s})\big)\Big], \tag{1}$$

which corresponds to a noise-contrastive objective between positive and negative examples. This type of objective has been proven to effectively maximize mutual information between $\mathbf{h}_i$ and $\mathbf{s}$ [6,21].

GIC Framework. Optimizing the node representations based on Eq. (1) encourages the encoder to preferentially encode information that is shared across all nodes. Such approach ensures that the computed representations do not encode the noise that may exist in some neighborhoods—the noise will be very different from the global summary. However, for exactly the same reason, it will also fail to encode information that is different from the global summary but is over-represented in small parts of the graph (e.g., the neighborhoods of some

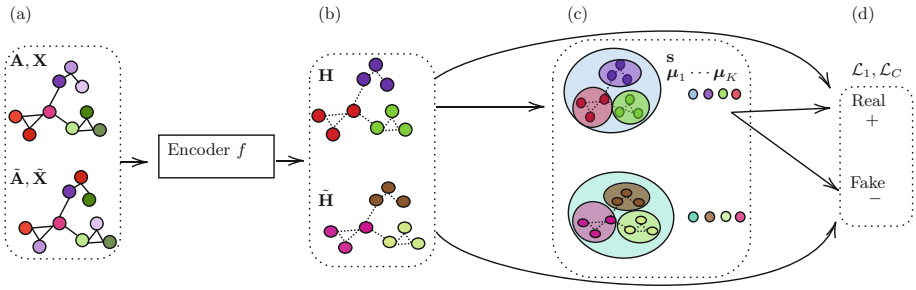

Fig. 1. GIC's framework. (a) A fake input is created based on the real one. (b) Embeddings are computed for both inputs with a GNN-encoder. (c) The global graph summary and the coarse-grain summaries are computed. (d) The goal is to discriminate between real and fake samples based on the computed summaries.

nodes). Capturing such information can be important for downstream tasks like link prediction and clustering.

Graph InfoClust (GIC) is specifically designed to address this problem. It postulates that the nodes belong to multiple clusters and learns node representations by simultaneously maximizing the MI between a node's (fine-grain) representation with that of the global graph summary and a coarse-grain summary derived from the clusters that it belongs to. Since this approach takes advantages of multiple entities within the graph, it leverages significantly more information that is present across the nodes and across different levels of the graph.

As Fig. 1 illustrates, GIC uses a GNN-based encoder to compute fine-grain node representations, which are then used to derive (i) the global summary, and (ii) the cluster-based summaries via a differentiable k-means clustering algorithm. These summaries are used in a contrastive-loss setting to define whether a node representation comes from the real or a fake graph. Specifically, GIC introduces a coarse-grain $\mathcal{L}_C$ loss to discriminate between real and fake samples based on the cluster-based summaries and uses the global-level $\mathcal{L}_1$ loss for the global summary, accordingly. GIC's overall objective is given by

$$\mathcal{L} = \alpha\mathcal{L}_1 + (1-\alpha)\mathcal{L}_C, \tag{2}$$

where $\alpha \in [0, 1]$ controls the relative importance of each component. The optimization of this objective leads to the maximization of the mutual information that is present in both fine-grain and coarse-grain levels as well as the global level of the graph.

3.2 Coarse-Grain Loss

Suppose we have computed a coarse-grain/cluster-derived summary $\mathbf{z}_i \in \mathbb{R}^D$ that is associated with v_i (described later in Sect. 3.3). Then, we can simply

maximize the mutual information between $\mathbf{z}_i$ and $\mathbf{h}_i$ in a similar manner with DGI. The new coarse-grain-based objective term is

$$\mathcal{L}_C = \sum_{i=1}^{N} \mathbb{E}_{(\mathbf{X},\mathbf{A})} \Big[\log g(\mathbf{h}_i, \mathbf{z}_i) \Big] + \sum_{i=1}^{N} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})} \Big[\log \big(1 - g(\tilde{\mathbf{h}}_i, \mathbf{z}_i)\big) \Big], \tag{3}$$

where positive examples are pairings of $\mathbf{h}_i$ with $\mathbf{z}_i$ and negatives are pairings $\tilde{\mathbf{h}}_i$ with $\mathbf{z}_i$. A discriminator $g : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is used to facilitate the optimization, as before, by assigning higher scores to the positive examples.

3.3 Coarse-Grain Summaries

Coarse-grain summaries summarize the information that is present in the corresponding (coarse-grain) clusters of nodes within the graph. Because nodes may belong to multiple clusters, and thus, may be present in multiple coarse-grain groups of the graph, it is advantageous to perform a soft-assignment to these clusters. Then, the coarse-grain summary $\mathbf{z}_i$ can be computed as a weighted average of the cluster centroids that node v_i belongs to.

Since the representations of the nodes can help identify better clusters (both w.r.t. communities and also their role), we optimize the clusters in an end-to-end fashion along with the node representations. Specifically, the cluster centroids $\boldsymbol{\mu}_k \in \mathbb{R}^D$ with $k = 1, \ldots, K$ (suppose K clusters) are obtained by a layer that implements a differentiable version of K-means clustering, as in ClusterNet [23], as follows. The clusters are updated by optimizing Eq. (3) via an iterative process by alternately setting

$$\boldsymbol{\mu}_k = \frac{\sum_i r_{ik} \mathbf{h}_i}{\sum_i r_{ik}} \quad k = 1, \ldots, K \tag{4}$$

and

$$r_{ik} = \frac{\exp(-\beta \cos(\mathbf{h}_i, \boldsymbol{\mu}_k))}{\sum_k \exp(-\beta \cos(\mathbf{h}_i, \boldsymbol{\mu}_k))} \quad k = 1, \ldots, K, \tag{5}$$

where $\cos(\cdot, \cdot)$ denotes the cosine similarity between two instances and β is an inverse-temperature hyperparameter; $\beta \to \infty$ gives a binary value for each cluster assignment. The gradients propagate only to the last iteration of the forward-pass updates in order to ensure convergence [23]. Finally, the cluster-derived summary, i.e., coarse-grain summary, associated with v_i in Eq. (3) is given by $\mathbf{z}_i = \sigma\left(\sum_{k=1}^{K} r_{ik} \boldsymbol{\mu}_k\right)$, where r_{ik} is the degree that v_i is assigned to cluster k and $\boldsymbol{\mu}_k$ is the centroid of the kth cluster, as described before, followed by a sigmoid nonlinearity.

3.4 Fake Input and Discriminators

When the input is a single graph, we opt to corrupt the graph by row-shuffling the original features $\mathbf{X}$ as $\tilde{\mathbf{X}} := \text{shuffle}([\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N])$ and $\tilde{\mathbf{A}} := \mathbf{A}$ (see Fig. 1a).

In the case of multiple input graphs, it may be useful to randomly sample a different graph from the training set as negative examples.

As the discriminator function d in $\mathcal{L}_1$, we use a bilinear scoring function, followed by a logistic sigmoid nonlinearity, which converts scores into probabilities, as $d(\mathbf{h}_i, \mathbf{s}) = \sigma(\mathbf{h}_i^T \mathbf{W} \mathbf{s})$, where $\mathbf{W}$ is a learnable scoring matrix. We use an inner product similarity, followed by $\sigma(\cdot)$, as the discriminator function g in $\mathcal{L}_C$, $g(\mathbf{h}_i, \mathbf{z}_i) = \sigma(\mathbf{h}_i^T \mathbf{z}_i)$. Here, we replace the bilinear scoring function used for g by an inner product, since it dramatically reduces the memory requirements and worked better in our case.

4 Related Work

Many unsupervised/self-supervised graph representation learning approaches follow a contrastive learning paradigm. Their objective is to give a higher score to positive examples and a lower to negative examples, which acts as a binary classification between positives and negatives. Based on the selection of positive/negative examples, methods are able to capture fine-grain similarities (DeepWalk [16], node2vec [3], GraphSAGE [4], GAE/VGAE [9], ARGVA [13], GMI [15]) or global similarities (DGI [21], MVGRL [5]) that are shared between nodes.

For example, in DeepWalk, node2vec and GraphSAGE, positive examples are representations of node pairs that co-occur in short random walks while negatives are representations of distant nodes. In GAE/VGAE and ARGVA, positive examples are representations of incident nodes and negatives are representations of random pairs of nodes. ARGVA uses additional positive/negative pairs as it discriminates at the same time whether a latent node representation comes from the prior (positive) or the graph encoder (negative). GMI has two types of positive/negative examples: First, by pairing a node representation with its input structure and attributes (positive) and with the input of a random node (negative), and second, by obtaining them as in GAE. MVGRL [5] works in a same manner with DGI, but it augments the real graph to get two additional versions of the graph (real and fake), which doubles the pairs of positive/negative examples. Methods that capture additional global properties (like DGI and MVGRL) are shown to estimate superior node representations.

5 Experimental Methodology and Results

5.1 Methodology and Configuration

Datasets. We evaluated the performance of GIC using seven commonly used benchmarks (Table 1). CORA, CiteSeer, and PubMed [25] are three citation networks, CoauthorCS and CoauthorPhysics are co-authorship graphs, and AmazonComputer and AmazonPhoto [18] are segments of the Amazon co-purchase graph.

Table 1. Datasets statistics of the entire graph and the largest connected component (LCC) of the graph.

	Classes	Features	Nodes	Edges	Label rate	Nodes LCC	Edges LCC	Label rate LCC
CORA	7	1,433	2,708	6,632	0.0517	2,485	5,069	0.0563
CiteSeer	6	3,703	3,327	4,614	0.0324	2,110	3,668	0.0569
PubMed	3	500	19,717	44,324	0.0030	19,717	44,324	0.0030
CoauthorCS	15	6,805	18,333	81,894	0.0164	18,333	81,894	0.0164
CoauthorPhysics	5	8,415	34,493	247,962	0.0029	34,493	247,962	0.0029
AmazonComputer	10	767	13,752	287,209	0.0145	13,381	245,778	0.0149
AmazonPhoto	8	745	7,650	143,663	0.0209	7,487	119,043	0.0214

Label rate is the fraction of nodes in the training set for node classification tasks

Hyper-parameter Tuning and Model Selection. As the encoder function f_{GNN} we use the graph convolution network (GCN) [8] with the propagation rule at layer l: $\mathbf{H}^{(l+1)} = \text{PReLU}\big(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{\Theta}\big)$, where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix with self-loops, $\hat{\mathbf{D}}$ is the diagonal degree matrix of $\hat{\mathbf{A}}$, $\mathbf{\Theta} \in \mathbb{R}^{F \times D}$ is a learnable matrix, PReLU denotes the nonlinear parametric rectified linear unit, and $\mathbf{H}^{(0)} = \mathbf{X}$.

We use an one-layer GCN-encoder ($l = 1$) and iterate the cluster updates in Eq. (4) and (5) for 10 times. Since GIC's cluster updates are performed in the unit sphere (cosine similarity), we row-normalize the embeddings before the downstream task.

GIC's learnable parameters are initialized with Glorot [2] and the objective is optimized using the Adam [7] with a learning rate of 0.001. We train for a maximum of $2k$ epochs, but the training is early terminated if the training loss does not improve in 20 or 50 consecutive epochs. The model state is reset to the one with the best (lowest) training loss. For each dataset-task pair, we perform model selection based on the validation set of the corresponding task (except for clustering, in which we use the link prediction task, instead). We set $\alpha \in \{0.25, 0.5, 0.75\}$ (regularization of the two objective terms), $\beta = \{10, 100\}$ (softness of the cluster assignments) and $K \in \{32, 128\}$ (number of clusters) to train the model, and keep the parameters' triplet that achieved the best result on the validation set. We determine these parameters once for each dataset-task pair, so that the corresponding results are reported based on the *same* triplet.

Competing Approaches and Implementation. We compare the performance of GIC against seventeen unsupervised and six semi-supervised methods and variants. The results for all the competing methods, except DGI and sometimes GMI and MVGRL, were obtained directly from [5,13,15,18]. We name VGAE-best and ARGVA-best the best performing variant of VGAE and ARGVA methods, respectively. We implemented GIC using the Deep Graph Library (DGL) [22] and PyTorch [14], as well by modifying DGI's original implementation (https://github.com/cmavro/Graph-InfoClust-GIC). All experiments were performed on a Nvidia Geforce RTX-2070 GPU on a i5-8400 CPU and 32 GB RAM machine.

Table 2. Mean node classification accuracy (with standard deviations) in % over 20 runs and for two different train/val sets: balanced and imbalanced. The datasets are randomly split in each run.

Train/Val.	Unsupervised				Semi-supervised	
	GIC		DGI		Best	Worst
	Imbalanced	Balanced	Imbalanced	Balanced	Balanced	
CORA	**81.7**(1.5)	80.7(1.1)	80.2(1.8)	80.0(1.3)	81.8(1.3)	76.6(1.9)
CiteSeer	**71.9**(1.4)	70.8(2.0)	71.5(1.3)	70.5(1.2)	71.9(1.9)	67.5(2.3)
PubMed	77.3(1.9)	**77.4**(1.9)	76.2(2.0)	76.8(2.3)	78.7(2.3)	76.1(2.3)
CoauthorCS	**89.4**(0.4)	89.3(0.7)	89.0(0.4)	88.7(0.8)	91.3(2.3)	85.0(1.1)
Coauth.Phys	**93.1**(0.7)	92.4(0.9)	92.7(0.8)	91.8(1.0)	93.0(0.8)	90.3(1.2)
Am.Comp.	**81.5**(1.0)	79.5(1.4)	79.0(1.7)	77.9(1.8)	83.5(2.2)	78.0(19.0)
Am.Photo	**90.4**(1.0)	89.0(1.6)	88.2(1.7)	86.8(1.7)	91.4(1.3)	85.7(20.3)

Results are reported on the largest connected component (LCC) of the graph. Semi-supervised methods: GCN [8], GAT [20], GraphSAGE [4], and MoNet [12].

5.2 Results

Node Classification. In unsupervised methods, the learned node embeddings are passed to a downstream classifier that is based on logistic regression. Following [18], we set the embedding dimensions to $D = 64$, unless otherwise stated. Also, we sample 20×#classes nodes as the train set, 30×#classes nodes as the validation set, the remaining nodes are the test set for the classifier. The sets are either uniformly drawn from each class (balanced sets) or randomly sampled (imbalanced sets). In the Planetoid split [25], 1,000 nodes of the remaining nodes are only used for testing. We use a logistic regression classifier, which is trained with a learning rate of 0.01 for 300 or 1k epochs with Adam optimizer and Glorot initialization.

Table 2 shows the performance of GIC compared to DGI and semi-supervised methods. Leveraging cluster information benefits datasets as CORA and PubMed, where GIC achieves a mean classification accuracy gain of more than 1% over DGI. In CiteSeer, CoauthorCS and CoauthorPhysics, the gain is slightly lower, but still more than 0.4%, since the abundant attributes of each node makes the cluster extraction more challenging. In AmazonComputers and AmazonPhoto, GIC performs significantly better than DGI with a gain of more than 2%, on average. Due to the large edge density of these datasets, the GNN-encoder aggregates information from multiple other nodes leading to representations very similar to the global summary. In such cases, GIC's objective term is responsible for making the representations to contain different aspects of information. In all cases, GIC performs better than the worst performing semi-supervised method with a gain of more than 1.5% and as high as 4.3%.

Table 3 further illustrates the performance of GIC compared to recently developed unsupervised methods based on MI maximization. The table shows

Table 3. Mean node classification accuracy (with standard deviations) for the Planetoid split.

	GIC		DGI		GMI		MVGRL	
	$D*$	$D = 64$	$D*$	$D = 64$	$D*$	$D = 64$	$D*$	$D = 64$
CORA	**83.2**(0.4)	81.4(0.5)	82.3(0.6)	79.2(0.7)	83.0(0.3)	77.7(0.4)	–	76.9(0.4)
CiteSeer	**73.4**(0.4)	71.6(0.6)	71.8(0.7)	69.0(0.9)	73.0(0.3)	68.4(0.9)	73.3(0.5)	70.8(0.7)
PubMed	**80.3**(0.6)	78.3(1.3)	76.8(0.6)	77.5(1.2)	80.1(0.2)	OOM	80.1(0.7)	78.5(0.8)

$D*$: Results obtained by the corresponding papers. For GIC, $D* = 300, 400, 150$.
"OOM": out of GPU memory. "–": Results were not reported for the Planetoid split.

Table 4. Link prediction scores: Mean area Under Curve (AUC) score [1] and average Precision (AP) score [19] with standard deviations over 10 runs (in %). Top: $D = 16$, Bottom: $D*$.

	CORA		CiteSeer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
DeepWalk [16]	83.1	85.0	80.5	83.6	84.4	84.1
VGAE-best [9]	91.4 ± 0.01	92.6 ± 0.01	90.8 ± 0.02	92.0 ± 0.02	96.4 ± 0.00	96.5 ± 0.00
ARGVA-best [13]	92.4	93.2	92.4	93.0	**96.8**	**97.1**
DGI [21]	89.8 ± 0.8	89.7 ± 1.0	95.5 ± 1.0	95.7 ± 1.0	91.2 ± 0.6	92.2 ± 0.5
GIC	**93.5** ± 0.6	**93.3** ± 0.7	**97.0** ± 0.5	**96.8** ± 0.5	93.7 ± 0.3	93.5 ± 0.3
DGI	94.8 ± 0.7	95.2 ± 0.8	98.5 ± 0.4	98.4 ± 0.3	93.9 ± 0.4	93.9 ± 0.4
GMI [15]	95.1 ± 0.3	95.6 ± 0.2	97.8 ± 0.1	97.4 ± 0.2	OOM	OOM
GIC	**96.0** ± 0.2	**96.1** ± 0.4	**98.9** ± 0.2	**99.0** ± 0.1	95.5 ± 0.1	95.6 ± 0.2

For DGI only, we set $D = 32$ which greatly improves its results compared to $D = 16$.

that leveraging additional coarse-grain information (GIC, MVGRL) helps better than leveraging extra fine-grain information (GMI). Moreover, GIC is the only method that consistently performs better than DGI across all datasets with varying embedding size D. The performance improvement for other methods (GIC, MVGRL) is evident only for large D, e.g., $D = 512$ in the papers.

Link Prediction. In link prediction, some edges are hidden in the input graph and the goal is to predict the existence of these edges based on the computed embeddings. The probability of an edge between nodes i and j is given by $\sigma(\mathbf{h}_i^T \mathbf{h}_j)$, where σ is the logistic sigmoid function. We follow the setup described in [9]: 5% of edges and negative edges as validation set, 10% of edges and negative edges as test set.

Table 4 illustrates the benefits of GIC for link prediction tasks. GIC outperforms DGI in all three datasets, since GIC's clustering is able to preserve and reveal useful interactions between nodes which may hint the existence of links between them. GIC also outperforms VGAE and ARGVA, in CORA and CiteSeer by 1%–2% and 4.5%–5.5%, respectively, even though these methods are specifically designed for link prediction tasks. In PubMed, which has fewer attributes to exploit, the performance of GIC is slightly worse than that of VGAE

Table 5. Clustering results with respect to the true labels.

	CORA			CiteSeer			PubMed		
	Acc	NMI	ARI	Acc	NMI	ARI	Acc	NMI	ARI
K-means	49.2	31.1	23.0	54.0	30.5	27.9	39.8	0.1	0.2
DeepWalk [16]	48,4	32.7	24.3	33.7	8.8	9.2	68.4	27.9	29.9
TADW [24]	56.0	44.1	33.2	45.5	29.1	22.8	35.4	0.1	0.1
VGAE-best [9]	60.9	43.6	34.7	40.8	17.6	12.4	67.2	27.7	27.9
ARGVA-best [13]	71.1	52.6	49.5	58.1	33.8	30.1	**69.0**	30.5	**30.6**
DGI [21]	59.0	38.6	33.6	57.9	30.9	27.9	49.9	15.1	14.5
GIC	**72.5**	**53.7**	**50.8**	**69.6**	**45.3**	**46.5**	67.3	**31.9**	29.1

Acc: accuracy, NMI [11]: normalized mutual information, ARI [11]: average rand index in percents (%).

and ARGVA. It is noteworthy, that GIC's proposed objective term works better than GMI which combines a mutual information objective term with a link prediction term.

Clustering. In clustering, the goal is to cluster together related nodes (e.g., nodes that belong to the same class) without any label information. The computed embeddings are clustered into $K = \#\text{classes}$ clusters with K-means. We set $D = 32$ and the evaluation is provided by external labels, the same used for node classification.

Table 5 illustrates GIC's performance for clustering. GIC performs better than other unsupervised methods in two out of three datasets (CORA and CiteSeer), and performs almost equally with ARGVA in PubMed. The gain over DGI is significantly large in all datasets, and can be as high as 15% to 18.5% for the NMI metric. Due to its interactive clustering, GIC can be considered an efficient method when the downstream task is clustering.

Ablation and Parameter Study

Effect of the Loss Function. We provide results w.r.t. silhouette score (SIL) [17], which is a clustering evaluation metric, of the learned representation after their t-SNE 2D projection [10]. Table 6 illustrates that using the novel objective term $\mathcal{L}_C$ ($\alpha \neq 1$) outperforms the baseline DGI objective $\mathcal{L}_1$ in all experiments. Primarily, using a single global vector to optimize the node representations may lead to certain pitfalls, especially when the embedding size and thus, the global vector's capacity, is low (Table 6a). Moreover, accounting for both cluster-level and graph-level information leads to better representations, while ignoring some of this information ($\alpha = 0$ or $\alpha = 1$) worsens the quality of the representations. This is also true when hyperparameters β and K are not optimized (Avg vs. Max in Table 6b), since the soft assignment of the clusters and their joint optimization during learning alleviates this need.

Table 6. SIL scores with varying embedding size D for CORA.

(a) SIL scores with varying embedding size D for CORA.

	$D = 16$	$D = 32$	$D = 512$
$\alpha = 0.5$ (GIC)	**0.195**	**0.229**	**0.257**
$\alpha = 1$ (DGI)	−0.121	−0.012	0.222

(b) Performance gain percentage (in %) w.r.t. SIL over DGI: $\beta \in \{10{,}100\}$, $K \in \{\#\text{classes}, 32, 128\}$.

	CORA		**CiteSeer**		**PubMed**	
	Avg	Max	Avg	Max	Avg	Max
$\alpha = 0$	3.3	18.3	7.8	19.3	15.8	22.1
$\alpha = .25$	22.0	32.5	**13.8**	**22.0**	21.4	32.4
$\alpha = .5$	**27.8**	**38.2**	12.0	21.4	43.1	49.4
$\alpha = .75$	25.7	30.9	7.5	10.3	**47.2**	**62.3**

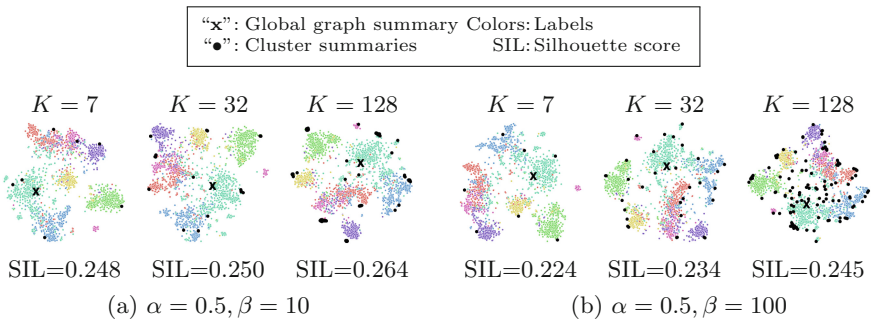

Fig. 2. t-SNE plots for CORA dataset and the corresponding silhouette scores (SIL). SIL score for $\alpha = 1$ is 0.212.

Visualization of the Clusters. In Fig. 2, we plot the t-SNE 2D projection of the learned node representations ($D = 64$) for the CORA dataset. A large β, e.g., $\beta = 100$ in Fig. 2b ($K = 128$), makes the distances between the cluster centers larger compared to a smaller one, e.g., $\beta = 10$ in Fig. 2a ($K = 128$). Increasing K generally helps, e.g., Fig. 2a ($K = 128$) compared to Fig. 2a ($K = 7$), however, that does not mean that all clusters will be distinct.

6 Conclusion

We have presented Graph InfoClust (GIC), an unsupervised graph representation learning method which relies on leveraging cluster-level content. GIC identifies nodes with similar representations, clusters them together, and maximizes their mutual information. This enables us to improve the quality of node representations with richer content and obtain better results than existing approaches for tasks like node classification, link prediction, clustering, and data visualization.

Acknowledgements. This work was supported in part by NSF (1447788, 1704074, 1757916, 1834251), Army Research Office (W911NF1810344), Intel Corp, and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

References

1. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms (1997)
2. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (2010)
3. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)
4. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems (2017)
5. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: Proceedings of International Conference on Machine Learning (2020)
6. Hjelm, R.D., et al.: Learning deep representations by mutual information estimation and maximization. In: International Conference on Learning Representations (2019)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014)
8. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
9. Kipf, T.N., Welling, M.: Variational graph auto-encoders. In: NIPS Workshop on Bayesian Deep Learning (2016)
10. Maaten, L.V.D., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. (2008)
11. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval (2008)
12. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5115–5124 (2017)
13. Pan, S., Hu, R., Fung, S.F., Long, G., Jiang, J., Zhang, C.: Learning graph embedding with adversarial training methods. IEEE Trans. Cybern. (2019)
14. Paszke, A., et al.: Automatic differentiation in pytorch (2017)
15. Peng, Z., et al.: Graph representation learning via graphical mutual information maximization. In: Proceedings of the Web Conference (2020)
16. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2014)
17. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
18. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
19. Su, W., Yuan, Y., Zhu, M.: A relationship between the average precision and the area under the ROC curve. In: Proceedings of the 2015 International Conference on the Theory of Information Retrieval (2015)
20. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
21. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: International Conference on Learning Representations (2019)

22. Wang, M., et al.: Deep graph library: towards efficient and scalable deep learning on graphs. In: ICLR Workshop (2019)
23. Wilder, B., Ewing, E., Dilkina, B., Tambe, M.: End to end learning and optimization on graphs. In: Advances in Neural and Information Processing Systems (2019)
24. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.: Network representation learning with rich text information. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
25. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning (2016)