Data Mining and Knowledge Discovery (2023) 37:228–254
https://doi.org/10.1007/s10618-022-00879-4

Informative pseudo-labeling for graph neural networks with few labels

Yayong Li[1] · **Jie Yin**[2] · **Ling Chen**[1]

Received: 15 December 2021 / Accepted: 11 September 2022 / Published online: 9 November 2022

Abstract

Graph neural networks (GNNs) have achieved state-of-the-art results for semi-supervised node classification on graphs. Nevertheless, the challenge of how to effectively learn GNNs with very few labels is still under-explored. As one of the prevalent semi-supervised methods, pseudo-labeling has been proposed to explicitly address the label scarcity problem. It is the process of augmenting the training set with pseudo-labeled unlabeled nodes to retrain a model in a self-training cycle. However, the existing pseudo-labeling approaches often suffer from two major drawbacks. First, these methods conservatively expand the label set by selecting only high-confidence unlabeled nodes without assessing their informativeness. Second, these methods incorporate pseudo-labels to the same loss function with genuine labels, ignoring their distinct contributions to the classification task. In this paper, we propose a novel informative pseudo-labeling framework (InfoGNN) to facilitate learning of GNNs with very few labels. Our key idea is to pseudo-label the most informative nodes that can maximally represent the local neighborhoods via mutual information maximization. To mitigate the potential label noise and class-imbalance problem arising from pseudo-labeling, we also carefully devise a generalized cross entropy with a class-balanced regularization to incorporate pseudo-labels into model retraining. Extensive experiments on six real-world graph datasets validate that our proposed approach

Responsible editor: Albrecht Zimmermann, Peggy Cellier

✉ Jie Yin
jie.yin@sydney.edu.au

Yayong Li
yayongli@outlook.com

Ling Chen
Ling.Chen@uts.edu.au

[1] Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW, Australia

[2] Discipline of Business Analytics, The University of Sydney, Sydney, NSW, Australia

significantly outperforms state-of-the-art baselines and competitive self-supervised methods on graphs.

Keywords Graph neural networks · Pseudo-labeling · Mutual information maximization

1 Introduction

Graph neural networks (GNNs) have emerged as state-of-the-art models for undertaking semi-supervised node classification on graphs (Hamilton et al. 2017; Kipf and Welling 2017; Veličković et al. 2018; Wu et al. 2019). The aim of these models is to leverage a small subset of labeled nodes together with a large number of unlabeled nodes to train an accurate classifier. Most modern GNNs rely on an iterative message passing procedure that aggregates and transforms the features of neighboring nodes to learn node embeddings, which are then used for node classification. However, under extreme cases where very few labels are available (*e.g.*, only a handful of labeled nodes per class), popular GNN architectures, such as graph convolutional networks (GCNs) typically with two layers, are ineffective in propagating the limited training labels to learn discriminative node embeddings, resulting in inferior classification performance. Recently, a central theme of latest studies has attempted to improve classification accuracy by designing deeper GNNs or new network architectures (Qu et al. 2019; Verma et al. 2020). However, the challenge of how to effectively learn GNNs with few labels is still under-explored.

Recently, pseudo-labeling, also called self-training, has been proposed as one prevalent semi-supervised method to explicitly tackle the label scarcity problem on graphs. Pseudo-labeling expands the label set by assigning a pseudo-label to high-confidence unlabeled nodes, and iteratively retrains the model with both given labels and pseudo-labels. Li et al. (2018) first proposed a self-trained GCN that chooses top-K high-confidence unlabeled nodes to enlarge the training set for model retraining. Sun et al. (2020) pointed out the ineffectiveness of shallow GCNs in propagating label information under few-label settings. A multi-stage approach was then proposed, which applies deep clustering techniques to assign pseudo-labels to unlabeled nodes with high prediction confidence. Zhou et al. (2019) proposed a dynamic self-training framework, which assigns a soft label confidence on the pseudo-label loss to control its contribution to gradient update.

Despite offering promising results, the existing pseudo-labeling approaches on GNNs have not fully explored the power of self-training, due to two major limitations. First, these methods impose strict constraints that only unlabeled nodes with high prediction probabilities are selected for pseudo-labeling. However, these selected nodes often exhibit similar information conveyed by the given labels, causing information redundancy in the expanded label set. On the contrary, if unlabeled nodes with lower prediction probabilities are allowed to enlarge the label set, more pseudo-label noise would be incurred to significantly degrade the classification performance. This creates a dilemma for pseudo-labeling to achieve desirable performance improvements. Second, current methods treat pseudo-labels and genuine labels equally important. They

are all incorporated into the same loss function, such as the standard cross entropy loss, for node classification, neglecting their distinct contributions to the classification task. In the presence of unreliable or noisy pseudo-labels, model performance might deteriorate during retraining.

Motivated by the above observations, we propose a novel informative pseudo-labeling framework called **InfoGNN** for semi-supervised node classification with few labels. Our aim is to fully harness the power of self-training by incorporating more pseudo-labels, but alleviating possible negative impact caused by noisy (i.e., incorrect) pseudo-labels. To address *information redundancy*, we define node informativeness via neural estimation of mutual information (MI) between a node and its local context subgraph in the embedding space. Our method offers two advantages: (1) It provides an informativeness measure to select unlabeled nodes for pseudo-labeling, such that the added pseudo-labels can bring in more information gain. (2) It implicitly encourages each node to approximate its own local neighborhood and depart away from other neighborhoods. The intuition behind is that an unlabeled node is considered informative when it can maximally reflect its local neighborhood. By integrating this informativeness measure with model prediction probabilities, our approach enables to selectively pseudo-label nodes with maximum performance gain. To mitigate the negative impact of *noisy pseudo-labels*, we adapt a generalized cross entropy loss to pseudo-labels for improving model robustness. This loss allows to maximize the pseudo-labeling capacity while minimizing the model collapsing risk. Moreover, to cope with the potential *class-imbalance problem* caused by pseudo-labeling under extremely few-label settings, we propose a class-balanced regularization that regularizes the number of pseudo-labels to keep relative equilibrium in each class.

Our main contributions can be summarized as follows:

- Our study analyzes the ineffectiveness of the existing pseudo-labeling strategies and proposes a novel pseudo-labeling framework for semi-supervised node classification with extremely few labels.
- Our approach has unique advantages to incorporate an MI-based informativeness measure for pseudo-label candidate selection and to alleviate the negative impact of noisy pseudo-labels via a generalized cross entropy loss.
- We validate our proposed approach on six real-world graph datasets of various types, demonstrating its superior performance over state-of-the-art baselines.

2 Related works

2.1 Graph learning with few labels

GNNs have emerged as a new class of deep learning models on graphs (Kipf and Welling 2017; Veličković et al. 2018). The principle of GNNs is to learn node embeddings by recursively aggregating and transforming features from local neighborhoods (Wu et al. 2019). Node embeddings are then used as input to any differentiable prediction layer, for example, a softmax layer for node classification. Recently, a series of semi-supervised GNNs, such as GCNs and their variants, have been proposed

for node classification. The success of these models relies on a sufficient number of labeled nodes for training. How to train GNNs with a very small set of labeled nodes has remained a challenging task.

Pseudo-labeling on graphs. To tackle label scarcity, pseudo-labeling has been proposed as one of the prevalent semi-supervised methods. It refers to a specific training regime, where the model is bootstrapped with additional labeled data obtained by using confidence-based thresholding methods (Lee 2013; Rosenberg et al. 2005). Recently, pseudo-labeling has shown promising results on semi-supervised node classification. Li et al. (2018) proposed a self-trained GCN that enlarges the training set by assigning pseudo-labels to top-K confidence unlabeled nodes, and then re-trains the model using both given and pseudo-labels. The pseudo-labels are generated by another random walk model rather than the GNN itself. A similar approach was proposed in Zhan and Niu (2021). Sun et al. (2020) showed that a shallow GCN is ineffective in propagating label information under few-label settings, and proposed a multi-stage self-training framework that relies on a deep clustering model to assign pseudo-labels. Zhou et al. (2019) proposed a dynamic pseudo-labeling approach called DSGCN that selects unlabeled nodes with prediction probabilities higher than a pre-specified threshold for pseudo-labeling, and assigns soft label confidence as label weight.

We argue that current pseudo-labeling methods on GNNs share two major problems: information redundancy and noisy pseudo-labels. This work is proposed to explicitly address these pitfalls. Our focus is upon developing a robust pseudo-labeling framework that allows to expand the pseudo-label set with more informative nodes, and to mitigate the negative impact of noisy pseudo-labels.

Our work is also related to learning with label noise, but most of the existing label noise methods cannot be adopted to handle noisy pseudo-labels due to two reasons. First, some methods require an extra clean set (Xiao et al. 2015) or other noise information (Han et al. 2018) (*e.g.*, the noise rate), which is unavailable in our setting; Second, other methods often involve designing an extra network for label noise modeling (Goldberger and Ben-Reuven 2016; Sukhbaatar et al. 2015) or require some other algorithmic modifications (Li et al. 2019; Ren et al. 2018). These modifications might be inconsistent with our learning objectives, thereby degrading model performance, and also incur extra computational complexity.

Graph few-shot learning. Originally designed for image classification, few-shot learning focuses on classification tasks where a classifier is adapted to accommodate new classes unseen during training, given only a few labeled examples for each class (Snell et al. 2017). Several recent studies (Ding et al. 2020; Huang and Zitnik 2020) have attempted to generalize few-shot learning to graph domains. For example, Ding et al. (2020) proposed a graph prototypical network for node classification, which learns a transferable metric space via meta-learning, such that the model can extract meta-knowledge to achieve good generalization ability on the target few-shot classification task. Huang and Zitnik (2020) proposed to transfer subgraph-specific information and learn transferable knowledge via meta gradients.

Although few-shot learning and our work both tackle the label scarcity problem, their problem settings and learning objectives are fundamentally different. In few-shot learning, the training and test sets typically reside in different class spaces. Hence, few-

shot learning aims to learn transferable knowledge to enable rapid generalization to new tasks. On the contrary, our work follows the transductive GNN setting where the training and test sets share the same class space.

Graph self-supervised learning. Our work is related to self-supervised learning on graphs (Velickovic et al. 2019), which also investigates how to best leverage the unlabeled data. However, there is a clear distinction in the objectives: the primary aim of self-supervised learning is to learn node/graph representations by designing pretext tasks without label-related supervision, such that the generated representations could facilitate specific classification tasks (Liu et al. 2022). For example, You et al. (2020a) showed that self-supervised learning can provide regularization for graph-related classification tasks. This work proposed three pretext tasks (*i.e.*, node clustering, graph partitioning, and graph completion) based on graph properties. Other works attempted to learn better node/graph representations through creating contrastive views, such as local node vs. global graph view in Velickovic et al. (2019), or performing graph augmentation (Zhu et al. 2020). In contrast, our work resorts to augmenting label-specific supervision via pseudo-labeling for semi-supervised node classification.

2.2 Mutual information maximization

The infomax principle was first proposed to encourage an encoder to learn effective representations that share maximized mutual information (MI) with the input (Belghazi et al. 2018; Hjelm et al. 2019). Recently, the idea of MI maximization has been applied to improve graph representations. Velickovic et al. (2019) applied MI maximization to learn node embeddings by contrasting local subgraphs with high-level, global graph representations. Qiu et al. (2020) proposed to learn intrinsic and transferable structural representations by contrasting subgraphs from different graphs via a discriminator. Hassani and Khasahmadi (2020) contrasted node representations from a local view with graph representations from a global view to learn more informative node embeddings. In our context, we leverage the idea of contrastive learning to maximize the MI between each node and its neighboring context. The estimated MI enables to select more representative unlabeled nodes in local neighborhoods for pseudo-labeling so as to advance model performance.

3 Problem statement

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ represents an undirected graph, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ denotes a set of n nodes, and $\mathcal{E}$ denotes a set of edges that connect pairs of nodes. $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$ denotes the node feature matrix, and $\mathbf{x}_i \in \mathcal{R}^d$ is a d-dimensional feature vector of node v_i. The graph structure is represented by the adjacent matrix $A \in \mathbb{R}^{n \times n}$, where $A(i, j) \in \{0, 1\}$. Since it is costly to acquire plentiful node labels due to restricted access or privacy concerns, real-world graphs often suffer from the label scarcity problem, with only a handful of labels provided for training. For example, the label rates on two real-world graphs, Coauther_CS (Shchur et al. 2018) and Wikics (Mernyei and Cangea 2020), are only 0.29% and 1.7%, respectively.

Thus, we assume that only a small portion of nodes are labeled in the node set, with $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|L|}$ denoting the set of labeled nodes. And most nodes remain to be unlabeled, denoted by $\mathcal{U}$. $\mathbf{y}_i = \{y_{i1}, y_{i2}, \ldots, y_{ic}\}$ is the one-hot encoding of node v_i's class label, and c is the number of classes.

We consider a semi-supervised node classification problem (Kipf and Welling 2017; Veličković et al. 2018) under a pseudo-labeling paradigm, which is formally defined as follows:

Problem 1 *Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ together with a small subset of labeled nodes $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{|L|}$, we aim to design a strategy $\mathbb{1}(\cdot)$ for expanding the label set from unlabeled nodes, a method $\mathbb{Y}(\cdot)$ for generating reliable pseudo-labels, and an exclusive loss function $\ell_U(\cdot)$ for pseudo-labels, such that $\mathbb{1}(\cdot)$, $\mathbb{Y}(\cdot)$ and $\ell_U(\cdot)$ can be combined with the task-specific loss $\ell_L(\cdot)$ to maximize the classification performance of GNN $f_\Theta(\cdot)$. This problem can be formulated as*

$$\min_{\Theta} \mathcal{J} = \sum_{\mathbf{x}_i \in \mathcal{L}} \ell_L(\mathbf{y}_i, f_\Theta(\mathbf{x}_i)) + \sum_{\mathbf{x}_i \in \mathcal{U}} \ell_U(\mathbb{Y}(\mathbf{x}_i), f_\Theta(\mathbf{x}_i)) \cdot \mathbb{1}(\mathbf{x}_i). \tag{1}$$

The notations used in the paper can be found in Appendix A.

4 Methodology

4.1 Framework overview

The primary aim of our work is to develop a robust pseudo-labeling framework for GNN training with few labels. As shown in Fig. 1, our proposed InfoGNN framework comprises of three key modules: (1) the GNN encoder; (2) the informativeness estimator; and (3) the pseudo-label selector. Taking a graph as input, the GNN encoder is first utilized to learn node embeddings as well as to estimate class predictions and confidence scores. Then, the informativeness estimator closely follows to produce node informativeness scores for unlabeled nodes. Finally, according to informativeness and confidence scores, informative nodes are selected for pseudo-labeling and model retraining. During GNN retraining phase, besides the standard cross entropy (SCE) loss applied on the given labels, a generalized cross entropy (GCE) loss is applied on pseudo-labels to improve model robustness against potential noise. A class-balanced regularization (CBR) is used to mitigate the potential class-imbalance problem arising during pseudo-labeling.

4.2 The GNN encoder

The GNN encoder in our framework learns node embeddings and generates class prediction probabilities, which reflect model confidence for making predictions. Any GNN that focuses on node classification can be utilized here for embedding learning and classification. The GNN encoder learns node embeddings by recursively aggregating and transforming node features from neighborhoods. In our work, we utilize

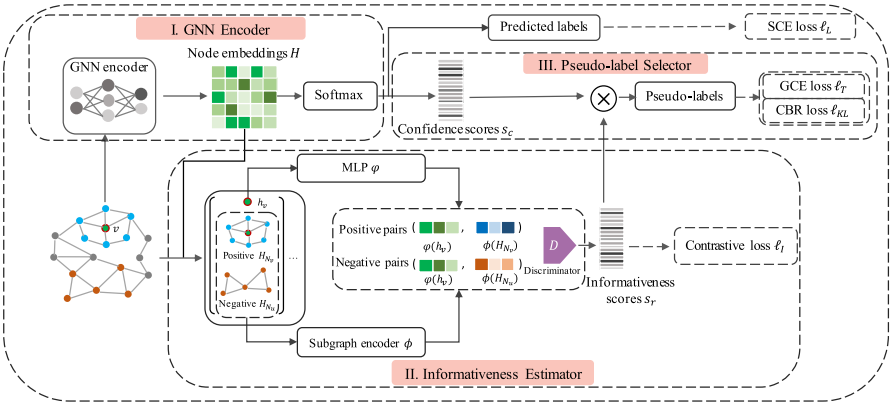


Fig. 1 Overview of the proposed InfoGNN framework, comprising of three main modules: the GNN encoder, informativeness estimator, and pseudo-label selector. The GNN encoder is responsible for generating node embeddings and estimating confidence scores. Then, the informativeness estimator closely follows, in charge of measuring node informativeness and producing quantitative scores. Finally, according to both confidence and informativeness scores, informative nodes are selected for pseudo-labeling and model retraining

GCN (Kipf and Welling 2017) as our GNN encoder $f_\Theta(\cdot)$ parameterized by Θ. For $v \in \mathcal{V}$, node embedding at *k-th* layer's propagation can be obtained by:

$$h_v^k = \sigma\Big(\sum_{v' \in \mathcal{N}_v} (\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2})_{v,v'}\Theta^{k-1}h_{v'}^{k-1}\Big), \tag{2}$$

$\sigma(\cdot)$ is the activation function, $\tilde{A} = A + I$ is the adjacency matrix of $\mathcal{G}$ with added self-connections. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and Θ^{k-1} is a layer-specific trainable weight matrix. We use the SCE loss to optimize GCN for node classification:

$$\ell_L(\mathbf{y}, f_\Theta(\mathbf{x})) = -\sum_{i \in \mathcal{L}} \mathbf{y}_i log(f_\Theta(\mathbf{x}_i)). \tag{3}$$

Finally, according to class prediction probabilities, we obtain a *confidence score* for each node v:

$$s_c(v) = \max_j f_\Theta(\mathbf{x}_v)_j. \tag{4}$$

The confidence score $s_c(v)$ is utilized for node selection in combination with the informativeness score, which is detailed below.

4.3 Candidate selection for pseudo-labelling

Current pseudo-labeling methods typically select unlabeled nodes based on model confidence or uncertainty (Zhou et al. 2019; Li et al. 2018). They pseudo-label only nodes

with high prediction probabilities, preventing adding noisy pseudo-labels for model retraining. However, such high-confidence nodes often carry redundant information conveyed by the given labels, resulting in the limited capacity to improve model performance. Thus, along with model confidence, we propose to take node informativeness into account for pseudo-label selection so as to maximally boost model performance. To this end, a key problem lies in how to measure node informativeness.

Informativeness measure by MI maximization. We define node informativeness as the representativeness of a node in reference to its contextual neighborhood. The intuition behind is that a node is considered informative when it could maximally represent its surrounding neighborhood while minimally reflecting other arbitrary neighborhoods. Hence, the representativeness of a node can be measured by the mutual information (MI) between the node itself and its neighborhood with positive correlation. On account of this, we employ *MI maximization* techniques (Belghazi et al. 2018) to estimate the MI by measuring how much one node can represent its surrounding neighborhood and discriminate an arbitrary neighborhood. This provides a score for quantifying the informativeness of each node. This principle is formulated as a *subgraph-based contrastive learning* task, which contrasts each node with its positive and negative context subgraphs.

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ with learned node embeddings H, for each node $v \in \mathcal{V}$, we define its *positive* subgraph as a local r-hop subgraph $\mathcal{N}_v$ centered at node v, and its *negative* subgraph as an r-hop subgraph $\mathcal{N}_u$ centered at an arbitrary node u. The mutual information $I^{\mathcal{G}}(v)$ between node v and its neighborhood can then be measured by a GAN-like divergence (Nowozin et al. 2016) as follows,

$$I^{\mathcal{G}} \geq \hat{I}^{\mathcal{G}} = \max_{\omega} \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} [MI_{\omega}(h_v, H_{\mathcal{N}_v}) + MI_{\omega}(h_v, H_{\mathcal{N}_u})], \tag{5}$$

where h_v is node v's embedding generated from the GNN encoder, $H_{\mathcal{N}_v}$ and $H_{\mathcal{N}_u}$ are the embedding sets of subgraphs centered at node v and u, respectively. MI_{ω} is a trainable neural network parameterized by ω. $MI_{\omega}(h_v, H_{\mathcal{N}_v})$ indicates the affinity between positive pairs, while $MI_{\omega}(h_v, H_{\mathcal{N}_u})$ indicates the discrepancy between negative pairs. Our objective is to estimate $I^{\mathcal{G}}$ by maximizing $MI_{\omega}(h_v, H_{\mathcal{N}_v})$ while minimizing $MI_{\omega}(h_v, H_{\mathcal{N}_u})$, which is in essence a contrastive objective.

This contrastive objective is achieved by employing a discriminator as shown in Fig. 1. At each iteration, after obtaining the learned node embeddings H, both positive and negative subgraphs for each node are first sampled and paired. Then, those nodes and their corresponding paired subgraphs are passed on to a discriminator $\mathcal{D}(\cdot)$ after being separately processed by a multilayer perception (MLP) encoder $\varphi(\cdot)$ and a subgraph encoder $\phi(\cdot)$. This discriminator finally produces an informativeness score for each node by distinguishing a node's embedding from its subgraph embedding.

Formally, we specify $MI_{\omega}(h_v, H_{\mathcal{N}_v}) = \mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v}))$. Here, $\varphi(\cdot)$ is an MLP encoder parameterized by $\omega^{(\varphi)}$ for node embedding transformation. $\phi(\cdot)$ is a *subgraph encoder* that aggregates embeddings of all nodes in the subgraph to generate an embedding of the subgraph, which is implemented using a one-layer GCN on an r-hop subgraph:

$$\phi(H_{\mathcal{N}_v}) = \sigma\left(\sum_{v' \in \mathcal{N}_v} (D_r^{-1/2} A_r D_r^{-1/2})_{v,v'} h_{v'} \omega^{(\phi)}\right), \tag{6}$$

where $\omega^{(\phi)}$ is a learnable parameter. A_r is the r-hop adjacent matrix, obtained by $A_r = Bin(AA_{r-1} + A_{r-1})$. $Bin(\cdot)$ is a binary function that guarantees $A_r(i, j) \in \{0, 1\}$. D_r is the corresponding degree matrix of A_r. Notice that we use an r-hop adjacent matrix A_r, instead of the original adjacent matrix A, for feature aggregation, and the aggregated embedding of the centre node is used as subgraph embedding. For the discriminator $\mathcal{D}(\cdot)$, we implement it using a bilinear layer:

$$\mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v})) = \sigma(\varphi(h_v)\omega^{(\mathcal{D})}\phi(H_{\mathcal{N}_v})^T), \tag{7}$$

where $\omega^{(\mathcal{D})}$ is a learnable parameter. To enable the discriminator $\mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v}))$ to measure the affinity between node v and its corresponding local subgraph $\mathcal{N}_v$, we minimize the binary cross entropy loss between positive and negative pairs, which is formulated as the *contrastive loss*:

$$\ell_I = -\frac{1}{|\mathcal{V}|}\sum_{v \in \mathcal{V}}[log\mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v})) + log(1 - \mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_u})))]. \tag{8}$$

By minimizing ℓ_I, the discriminator could maximally distinguish a node from any arbitrary subgraphs that it does not belong to in the embedding space. This process is equivalent to maximizing their MI in the sense of Eq. (5).

Pseudo-labeling. The discriminator $\mathcal{D}(\cdot)$ measures the affinity between each node and its local subgraph. We utilize this affinity to define the *informativeness score* for each node:

$$s_r(v) = \mathcal{D}(\varphi(h_v), \phi(H_{\mathcal{N}_v})), \tag{9}$$

where $s_r(v)$ indicates to what extent a node could reflect its neighborhood, and a higher score means that the node is more informativeness. Therefore, by considering both the informativeness score and model prediction confidence, we derive the selection criterion to construct the pseudo-label set $\mathcal{U}_p$:

$$\mathcal{U}_p = \{v \in \mathcal{U} | (s_r(v) + s_c(v))/2 > k, s.t.\ s_c(v) > k\}, \tag{10}$$

where $s_c(v)$ is the confidence score as in Eq. (4), and k is a hyperparameter whose value can be empirically determined (See Fig. 4b in Sect. 5.6). We then produce the pseudo-labels for $\mathcal{U}_p$ by utilizing the GNN encoder $f_\Theta(\cdot)$:

$$\hat{\mathbf{y}}_v = \underset{j}{\text{argmax}}\ f_\Theta(\mathbf{x}_v)_j;\ v \in \mathcal{U}_p, \tag{11}$$

where the pseudo-label $\hat{\mathbf{y}}_v$ is actually the predicted label by the GNN encoder.

4.4 Mitigating noisy pseudo-labels

During model retraining, existing pseudo-labeling methods regard the given labels and pseudo-labels as equally important, so an identical loss function, *e.g.*, the SCE loss, is applied. However, with more nodes added in, it is inevitable to introduce unreliable or noisy (*i.e.*, incorrect) pseudo-labels. If the same SCE loss is applied on unreliable pseudo-labels, it would degrade model performance. This is because, the SCE loss implicitly weighs more on the difficult nodes whose predictions deviate away from the supervised labels during gradient update (Zhang and Sabuncu 2018; Van Rooyen et al. 2015). This is beneficial for training with clean labels and ensures faster convergence. However, when there exist noisy pseudo-labels in the label set, more emphasis would be put on noisy pseudo-labels as they are harder to fit than correct ones. This would ultimately cause the model to overfit incorrect labels, thereby degrading model performance.

To address this issue, we propose to apply the negative Box-Cox transformation (Box and Cox 1964) to the loss function $\ell_U(\cdot)$ on pseudo-label set $\mathcal{U}_p$, inspired by (Zhang and Sabuncu, 2018). The transformed loss function is given as follows:

$$\begin{aligned} \ell_U(\hat{\mathbf{y}}_i, f_\Theta(\mathbf{x}_i)) &= \frac{1 - f_\Theta(\mathbf{x}_i)_j^q}{q}; \mathbf{x}_i \in \mathcal{U}_p, \\ \hat{\mathbf{y}}_i &= \underset{j}{\operatorname{argmax}}\, f_\Theta(\mathbf{x}_i)_j, \end{aligned} \tag{12}$$

where $q \in (0, 1]$, $\hat{\mathbf{y}}_i$ is the pseudo-label. To further elaborate how this loss impacts parameter update, we have its gradient as follows:

$$\frac{\partial \ell_U(\hat{\mathbf{y}}_i, f_\Theta(\mathbf{x}_i))}{\partial \Theta} = f_\Theta(\mathbf{x}_i)_j^q \left(-\frac{1}{f_\Theta(\mathbf{x}_i)_j} \nabla_\Theta f_\Theta(\mathbf{x}_i)_j \right), \tag{13}$$

where $f_\Theta(\mathbf{x}_i)_j \in (0, 1]$ for $\forall i$. Compared with the SCE loss, ℓ_U actually weighs each gradient by an additional $f_\Theta(\mathbf{x}_i)_j^q$, which reduces the gradient descending on those unreliable pseudo-labels with lower prediction probabilities. In fact, $\ell_U(\hat{\mathbf{y}}_i, f_\Theta(\mathbf{x}_i))$ can be regarded as the *generalization of the SCE loss and the unhinged loss*. It is equivalent to SCE when q approaches zero, and becomes the unhinged loss when q is equal to one. Thus, this loss allows the network to collect more additional information from a larger amount of pseudo-labels while alleviating their potential negative effect.

In practice, we apply a truncated version of $\ell_U(\cdot)$ to filter out the potential impact from unlabeled nodes with low prediction probabilities, given by:

$$\ell_T(\hat{\mathbf{y}}_i, f_\Theta(\mathbf{x}_i)) = \begin{cases} \ell_U(k), & f_\Theta(\mathbf{x}_i)_j \le k \\ \ell_U(\hat{\mathbf{y}}_i, f_\Theta(\mathbf{x}_i)), & f_\Theta(\mathbf{x}_i)_j > k, \end{cases} \tag{14}$$

where $k \in (0, 1)$, and $\ell_U(k) = (1 - k^q)/q$. Formally, the truncated loss version is derived as:

$$\ell_T(\hat{\mathbf{y}}, f_\Theta(\mathbf{x})) = \sum_{i \in \mathcal{U}} \lambda_i \ell_U(\hat{\mathbf{y}}_i, f_\Theta(\mathbf{x}_i)) + (1 - \lambda_i)\ell_U(k), \tag{15}$$

where $\lambda_i = 1$ if $i \in \mathcal{U}_p$, otherwise $\lambda_i = 0$. ℓ_T is referred to as the generalized cross entropy (GCE) loss. Intuitively, when the prediction probability of one node is lower than k, the GCE loss would be a constant. As the gradient of a constant loss is zero, this node would not contribute to gradient update, thus eliminating the negative effect of pseudo-labels with low confidence.

4.5 Class-balanced regularization

Under extreme cases where only very few labels are available for training, severe class-imbalance problems would occur during pseudo-labeling. That means, one or two particular classes might dominate the whole pseudo-label set, thus conversely impacting model retraining. To mitigate this issue, we propose to apply a Kullback–Leibler (KL) divergence between the pseudo-label distribution and a default label distribution for class-balanced regularization (CBR):

$$\ell_{KL} = \sum_{j=1}^{c} p_j log \frac{p_j}{\overline{f(X)}_j}, \tag{16}$$

where p_j is the default probability of class j. Since the true label distribution is unknown, we apply a uniform distribution for this regularization. That is, we set the probability of each class as $p_j = 1/c$ in our work. $\overline{f(X)}_j$ is the mean value of class prediction probability distribution over pseudo-labels, which is calculated as:

$$\overline{f(X)}_j = \frac{1}{|\mathcal{U}_p|} \sum_{\mathbf{x}_i \in \mathcal{U}_p} f(\mathbf{x}_i)_j. \tag{17}$$

It is worth noting that, under the uniform distribution assumption, we do not attempt to approximate the real label distribution, which is unknown a priori during training. Instead, we expect to regularize the class distribution in the pseudo-label set to be more uniformly distributed, preventing only one or two classes from dominating the selected pseudo-labels. Accordingly, hyperparamter β is employed to control the impact of ℓ_{KL} as in Eq. (19). More empirical analysis on the impact of class-balanced regularization will be provided in Sect. 5.6.

4.6 Model training and computational complexity

Our proposed InfoGNN framework is given by Algorithm 1, which consists of one pre-training phase and one formal training phase. The pre-training phase (Step 2-4) is used to train a parameterized GNN with the given labels. Accordingly, network

parameters are updated by:

$$\ell_{pre} = \ell_L + \alpha \ell_I. \tag{18}$$

During the formal training phase, the pre-trained GNN is first applied to generate the prediction probability and informativeness score for each node, which are then used to produce pseudo-labels (Step 5–7). Finally, both given and pseudo- labels are used to retrain the GNN by minimizing the following loss function (Step 8):

$$\ell = \ell_L + \ell_T + \alpha \ell_I + \beta \ell_{KL}. \tag{19}$$

Algorithm 1: Training InfoGNN with few labels

Input: Graph $G = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, α, β, r, q and k, Initialized network parameters $\{\Theta^0, \omega^0\}$
Output: updated model parameters $\{\Theta^t, \omega^t\}$

1 **for** $t = 0; t < epochs; t = t + 1$ **do**
2 **if** $t < start\_epoch$ **then**
3 Pre-train the network according to Eq. (18);
4 **else**
5 Generate node prediction probabilities $f_\Theta(\mathbf{x}_i)$;
6 Generate informativeness scores based on Eq. (9);
7 Construct pseudo-label set based on Eq. (10);
8 Update network parameters based on Eq. (19);
9 **return** *model parameters* $\{\Theta^t, \omega^t\}$

In terms of computational complexity, by comparison with GNN models based on the SCE loss, InfoGNN incurs slightly extra computational overhead in its attempt to mitigate label noise. This is mainly due to the calculation of the contrastive loss ℓ_I with subgraph encoder. Since we utilize a one-layer GCN as the subgraph encoder on an r-hop subgraph, its computational complexity is linear with the number of edges $\mathcal{O}(|E_r|)$, where E_r is the number of edges in the r-hop subgraph. This is reasonably acceptable.

5 Experiments

To validate the effectiveness of the proposed pseudo-labeling framework, we carry out extensive experiments on six real-world graph datasets to compare against state-of-the-art baselines. We also conduct the ablation studies and sensitivity analyses to better understand the key ingredients of our approach.

5.1 Datasets

Our experiments use six real-world graph datasets from three different domains:

Table 1 Details of six benchmark datasets

Dataset	Nodes	Edges	Classes	Features
Citeseer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
Dblp	17,716	105,734	4	1,639
Wikics	11,701	216,123	10	300
Coauthor_CS	18,333	81,894	15	6,805
Coauthor_Phy	34,493	247,962	5	8,415

- *Citation networks:* Cora, Citeseer (Kipf and Welling 2017) and Dblp[1] (Bojchevski and Günnemann 2018) are citation networks, where each node indicates a paper with a certain label and edges indicates the citation links among papers. Node features are bag-of-words vectors of papers.
- *Webpage networks:* Wikics[2] (Mernyei and Cangea 2020) is a network of computer science related Webpages. Nodes represent articles and edges represent hyperlinks between articles. Node features are mean vectors of GloVe word embeddings of articles.
- *Coauther networks:* Coauther-CS and Coauther-Phy[3] (Shchur et al. 2018) are coauthor networks in computer science and Physics. Nodes denote authors and edges denote whether two authors coauthor a paper. Node features are keywords from the author's papers.

Detailed dataset statistics are listed in Table 1.

5.2 Baselines

For comparison, we use a total of 12 state-of-the-art methods as baselines. Since all methods are built upon the original GCN (Kipf and Welling 2017), we compare against GCN as the benchmark. The other 11 recently proposed methods on graphs are used as strong competitors, which can be categorized into two groups:

- *Pseudo-labeling methods*: M3S (Sun et al. 2020), Self-training (Li et al. 2018), Co-training (Li et al. 2018), Union (Li et al. 2018), Intersection (Li et al. 2018), and DSGCN (Zhou et al. 2019);
- *Self-supervised methods*: Super-GCN (Kim and Oh 2021), GMI (Peng et al. 2020), SSGCN-clu (You et al. 2020b), SSGCN-comp (You et al. 2020b), and SSGCN-par (You et al. 2020b).

We run all experiments 10 times with different random seeds, and report the mean Micro-F1 scores. Due to algorithmic design, the number of selected pseudo-labels might vary among different methods. Thus, we report the best performance of each baseline method with its optimized hyperparameters.

[1] https://github.com/abojchevski/graph2gauss

[2] https://github.com/pmernyei/wiki-cs-dataset/raw/master/dataset

[3] https://github.com/shchur/gnn-benchmark

Table 2 Details of hyperparameters

Given labels (per class)	α	β	k
{1, 3, 5}	1.0	1.0	0.55
{10, 15, 20, 30, 40, 50}	0.2	0.2	0.55

5.3 Experimental setup

Model specification. For fair comparison, all baselines are adapted to use a two-layer GCN with 16 units of the hidden layer. The hyperparameters are set the same with GCN (Kipf and Welling 2017), with the L2 regularization of 5×10^{-4}, learning rate of 0.01, ReLU activation, and dropout rate of 0.5. For subgraph encoder $\phi(\cdot)$, we use a one-layer GCN with the c-dimension output, where c is number of classes. Both positive and negative subgraphs share the same subgraph encoder. $\varphi(\cdot)$ is a one-layer MLP with the c-dimension output. The discriminator $\mathcal{D}(\cdot)$ is a one-layer bilinear network with one-dimension output, and it uses the sigmoid activation function. The stability analysis of the discriminator can be found in Appendix B.

Following the setup of self-training methods (Zhou et al. 2019), we split each dataset into training and test sets. We randomly choose {1, 3, 5, 10, 15, 20, 30, 40, 50} nodes per class for training as different settings, and the remaining nodes are used for testing. The performance of different methods is assessed on the test set for comparison.

Hyperparameter specification. We specify hyperparameters conforming to the following rules: Generally, a larger α and β value would be beneficial to model training when the given labels are scarce, while smaller α and β values are more likely to achieve better performance as the number of given labels increases. For k, we fix its value to 0.55 for all settings. The specification of the three hyperparameters are summarized in Table 2. In terms of q, we empirically find that our model has relatively lower sensitivity to q with the regularization of loss ℓ_I, so its value is fixed under most of the settings. Specifically, we set $q = 1.0$ when one label per class is given, and $q = 0.1$ for all other label rates. The best r value for subgraph embedding in loss ℓ_I depends on the edge density of the input graph. Particularly, we apply $r = 3$ for edge-sparse graphs (Cora, Citeseer, Dblp, Coauther_CS), $r = 2$ for Wikics, and $r = 1$ for Coauther_Phy.

Implementation details. When training InfoGNN, we first pre-train the network to generate reliable predictions using Eq. (18) for 200 epochs, and then proceed with formal training using the full loss function Eq. (19) for another 200 epochs. During formal training, in order to get a steady model, we allow the model to update the pseudo-label set every 5 epochs using Eq. (10). When updating the pseudo-label set, we use the mean scores of unlabeled nodes in its last 10 training epochs, rather than the current prediction and informativeness scores. Our framework is implemented using Pytorch. All experiments are run on a machine powered by Intel(R) Xeon(R) Gold 6126 @ 2.60GHz CPU and 2 Nvidia Tesla V100 32GB Memory Cards with Cuda version 10.2.

5.4 Comparison with state-of-the-art baselines

The mean Micro-F1 scores of all methods w.r.t. various label rates are reported in Tables 3 and 4. Table 3 focuses on the cases where the given labels are very sparse, whereas Table 4 reports on the cases with relatively higher label rates. The best performer is highlighted by **bold**, and the second best performer is highlighted by *underline* on each setting. We perform paired t-test between the Micro-F1 scores achieved by InfoGNN and the best baseline methods, where we use •(◦) to indicate that InfoGNN is significantly better (worse) than the compared baseline methods at 95% significance level.

Table 3 reports the overall performance comparison under the severe label sparsity settings. Overall, our proposed InfoGNN method outperforms other baseline methods by a large margin at almost all the settings. Compared with GCN, InfoGNN averagely achieves a performance improvement of 12.1%, 9.2%, 8.0%, 6.3%, 4.1%, and 3.5% on the six datasets, when 1, 3, 5, 10, 15, and 20 nodes per class are labeled, respectively. In particular, InfoGNN achieves better classification results with lower label rates. At the presence of less than 10 labeled nodes per class, InfoGNN succeeds in achieving similar Micro-F1 scores as GCN uses 20 labeled nodes per class over all datasets. As for self-supervised baselines, their performance is inconsistent across different datasets. For example, SSGCN-clu achieves advantageous results on Coauthor_CS and Coauthor_Phy, but yields undesirable results on the other four datasets. SSGCN-Comp performs poorly on Wikics. This is due to the fact that specific pretext tasks designed by SSGCN do not generalize well on graphs with different properties.

Table 4 further compares the performance of all methods w.r.t. higher label rates, with 30, 40, and 50 given labels per class. As can be seen, as the number of given labels per class increases beyond 20, Micro-F1 scores of all methods continue to increase but with a declining growth rate. In the meanwhile, the advantages of pseudo-labeling methods gradually diminish as compared to the original GCN. However, our InfoGNN still outperforms other baselines in most cases, especially on Cora and Citeseer. For example, when 50 labeled nodes per class are given on Cora, our InfoGNN achieves a Micro-F1 score of 85.3%, markedly outperforming the second best performer (SSGCN-clu) and GCN by 1.6% and 2.4%, respectively. This proves that our InfoGNN is able to effectively alleviate the information redundancy problem when label information is relatively sufficient.

5.5 Ablation study

To further analyze how different components of the proposed InfoGNN take effect, we conduct a series of ablation experiments. Due to space limit, we only report experimental results on the settings where 3 and 10 nodes are labeled per class. The ablations are designed as follows:

– *InfoGNN-I*: Only ℓ_I is applied based on GCN, which is used to evaluate the role of the contrastive loss;

Table 3 The micro-F1 performance comparison with low label rates

Method	Cora						Citeseer					
	1	3	5	10	15	20	1	3	5	10	15	20
GCN	0.418	0.616	0.685	0.742	0.784	0.797	0.381	0.504	0.569	0.602	0.660	0.682
Super-GCN	0.522	0.673	0.720	0.760	0.788	0.799	0.499•	0.610	0.665•	0.700•	0.706•	0.712
GMI	0.502	0.672	0.715	0.757	0.783	0.797	0.497	0.568	0.621	0.632	0.670	0.683
SSGCN-clu	0.407	0.684	0.739	0.776	0.797•	0.810•	0.267	0.388	0.507	0.616	0.634	0.647
SSGCN-comp	0.451	0.609	0.676	0.741	0.772	0.794	0.433	0.547	0.638	0.682	0.692	0.709
SSGCN-par	0.444	0.649	0.692	0.734	0.757	0.770	0.457	0.578	0.643	0.693	0.705	0.716•
Cotraining	0.533	0.661	0.689	0.741	0.764	0.774	0.383	0.469	0.563	0.601	0.640	0.649
Selftraining	0.399	0.608	0.693	0.761	0.789	0.793	0.324	0.463	0.526	0.647	0.683	0.685
Union	0.505	0.663	0.713	0.764	0.792	0.797	0.366	0.491	0.560	0.631	0.663	0.667
Intersection	0.408	0.596	0.674	0.736	0.770	0.775	0.337	0.497	0.582	0.671	0.694	0.699
M3S	0.439	0.651	0.688	0.754	0.763	0.789	0.307	0.515	0.635	0.674	0.683	0.695
DSGCN	0.596•	0.712•	0.745•	0.777•	0.792	0.795	0.463	0.613•	0.652	0.674	0.681	0.684
InfoGNN	**0.601**	**0.735**	**0.776**	**0.792**	**0.813**	**0.828**	**0.540**	**0.652**	**0.717**	**0.721**	**0.725**	**0.733**
Method	Dblp						Wikics					
	1	3	5	10	15	20	1	3	5	10	15	20
GCN	0.472	0.583	0.627	0.652	0.688	0.718	0.384	0.550	0.638	0.682	0.712	0.720
Super-GCN	0.472	0.583	0.685	0.708	0.729	0.738	0.399	0.552	0.599	0.683	0.712	0.721
GMI	0.544	0.597	0.656	0.728	0.739	0.754	0.325	0.484	0.546	0.654	0.683	0.700
SSGCN-clu	0.369	0.528	0.649	0.692	0.721	0.744	0.335	0.579	0.627	0.694	0.714	0.725
SSGCN-comp	0.458	0.525	0.598	0.634	0.674	0.707	0.224	0.261	0.358	0.381	0.343	0.356
SSGCN-par	0.418	0.545	0.639	0.683	0.708	0.733	0.332	0.593	0.659	0.706	0.732	0.740

Table 3 continued

Method	Dblp						Wikics					
	1	3	5	10	15	20	1	3	5	10	15	20
Cotraining	0.545	0.646	0.634	0.674	0.703	0.701	0.367	0.584	0.645	0.692	0.724	0.737
Selftraining	0.437	0.580	0.634	0.707	0.738	0.759	0.350	0.602	**0.655**○	0.701	0.725	0.738
Union	0.485	0.618	0.652	0.712	0.737	0.746	0.351	0.584	0.646	0.694	0.723	0.740●
Intersection	0.458	0.581	0.566	0.665	0.715	0.734	0.359	0.599	0.654	0.706●	0.726●	0.740●
M3S	0.547	0.635	0.672	0.733	0.749●	0.752	0.401	0.593	0.621	0.685	0.711	0.734
DSGCN	0.587●	**0.671**○	0.720●	0.738●	0.744	0.764●	0.414●	0.607●	0.635	0.705	0.716	0.728
InfoGNN	**0.596**	0.669	**0.746**	**0.765**	**0.773**	**0.787**	**0.460**	**0.610**	0.650	**0.723**	**0.740**	**0.742**
Method	Coauther_CS						Coauther_Phy					
	1	3	5	10	15	20	1	3	5	10	15	20
GCN	0.640	0.799	0.847	0.893	0.901	0.909	0.700	0.849	0.868	0.901	0.912	0.918
Super-GCN	0.668	0.841	0.869	0.895	0.897	0.897	0.688	0.848	0.891	0.908	0.923	0.923
GMI	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
SSGCN-clu	**0.770**○	**0.886**○	0.890●	0.905●	0.908	0.911	**0.889**○	0.923●	0.930●	0.935●	0.936●	0.936●
SSGCN-comp	0.711	0.858	0.888	0.904	0.907	0.909	0.798	0.892	0.904	0.927	0.921	.928
SSGCN-par	0.737	0.860	0.881	0.898	0.901	0.903	0.824	0.915	0.919	0.925	0.931	0.931
Cotraining	0.643	0.745	0.810	0.849	0.864	0.885	0.758	0.842	0.850	0.898	0.891	0.917
Selftraining	0.592	0.770	0.828	0.873	0.892	0.895	0.744	0.865	0.890	0.908	0.914	0.921
Union	0.621	0.772	0.812	0.856	0.864	0.885	0.750	0.855	0.870	0.908	0.902	0.910
Intersection	0.650	0.775	0.851	0.887	0.893	0.898	0.612	0.763	0.854	0.901	0.904	0.926
M3S	0.648	0.818	0.879	0.897	0.909●	0.912●	0.828	0.868	0.895	0.914	0.922	0.930
DSGCN	0.743○	0.829	0.863	0.879	0.883	0.892	0.781	0.812	0.862	0.896	0.908	0.916
InfoGNN	0.683	0.865	**0.892**	**0.906**	**0.913**	**0.918**	0.842	**0.924**	**0.934**	**0.938**	**0.942**	**0.942**

OOM indicates out-of-memory on a 32GB GPU

Table 4 The Micro-F1 performance comparison with higher label rates

Method	Cora			Citeseer			Dblp		
	30	40	50	30	40	50	30	40	50
GCN	0.816	0.825	0.829	0.695	0.708	0.716	0.743	0.753	0.770
Super-GCN	0.812	0.828	0.836	0.720	0.728	0.737	0.760	0.767	0.775
GMI	0.806	0.815	0.820	0.692	0.695	0.701	0.784•	**0.794**○	0.794•
SSGCN-clu	0.822•	0.829•	0.837•	0.682	0.683	0.680	0.756	0.766	0.775
SSGCN-comp	0.804	0.819	0.830	0.718	0.729	0.739•	0.744	0.752	0.761
SSGCN-par	0.784	0.791	0.798	0.724•	0.732•	0.738	0.751	0.762	0.769
Cotraining	0.804	0.820	0.823	0.675	0.684	0.697	0.716	0.726	0.736
Selftraining	0.807	0.821	0.818	0.696	0.706	0.710	0.777	0.775	0.782
Union	0.807	0.819	0.827	0.688	0.691	0.694	0.757	0.764	0.757
Intersection	0.800	0.818	0.821	0.705	0.712	0.716	0.745	0.765	0.769
M3S	0.792	0.807	0.815	0.713	0.716	0.721	0.765	0.769	0.774
DSGCN	0.798	0.809	0.816	0.684	0.684	0.685	0.784	0.786	0.786
InfoGNN	**0.835**	**0.848**	**0.853**	**0.735**	**0.737**	**0.742**	**0.789**	0.792	**0.795**
Method	Wikics			Coauther_CS			Coauther_Phy		
	30	40	50	30	40	50	30	40	50
GCN	0.752•	0.761•	0.764	0.901	0.900	0.903	0.924	0.932	0.933
Super-GCN	0.742	0.752	0.763	0.908	0.909	0.909	0.929	0.930	0.933
GMI	0.713	0.730	0.746	OOM	OOM	OOM	OOM	OOM	OOM

Table 4 continued

Method	Wikics			Coauther_CS			Coauther_Phy		
	30	40	50	30	40	50	30	40	50
SSGCN-clu	0.738	0.745	0.747	0.914	0.915	0.915	0.938•	0.939•	0.940
SSGCN-comp	0.361	0.375	0.412	0.909	0.918	0.922•	0.928	0.933	0.937
SSGCN-par	0.741	0.750	0.755	0.906	0.908	0.908	0.933	0.933	0.934
Cotraining	0.750	0.756	0.765	0.889	0.895	0.898	0.926	0.924	0.927
Selftraining	0.743	0.760	**0.768**○	0.901	0.901	0.904	0.932	0.932	0.932
Union	0.752•	0.761	0.765	0.893	0.901	0.898	0.921	0.931	0.925
Intersection	0.748	0.765	0.767	0.896	0.898	0.905	0.927	0.927	0.932
M3S	0.745	0.755	0.763	0.916•	0.920•	0.922•	0.935	0.937	0.940
DSGCN	0.751	0.759	0.763	0.893	0.896	0.897	0.916	0.920	0.922
InfoGNN	**0.754**	**0.764**	0.766	**0.919**	**0.922**	**0.923**	**0.943**	**0.944**	**0.945**

OOM indicates out-of-memory on a 32GB GPU

– *InfoGNN-IT*: Both ℓ_I and ℓ_T are applied, which is utilized to evaluate the impact of the GCE loss by comparing with InfoGNN-I. Note that only model confidence scores are used here for ℓ_T, *i.e.*, $\mathcal{U}_p = \{v \in \mathcal{U} | f(\mathbf{x}_v)_j > k\}$;
– *InfoGNN-ITS*: On the basis of InfoGNN-IT, the informativeness score, *i.e.*, Eq.(10), is also applied for ℓ_T, which is to test the efficacy of the informativeness score by comparing with InfoGNN-IT. The impact of the ℓ_{KL} loss can be revealed by comparing with InfoGNN.

The ablation results are reported in Table 5, under two settings where the number of given labels per class is 3 and 10, respectively. The constrastive loss ℓ_I seems to make similar contributions with both label rates, achieving an average improvement of 3.7% and 3.9% over GCN on the six datasets. On top of ℓ_I, the use of GCE leads to further performance improvements. Taking Wikics as an example, GCE further boosts the accuracy by 3.8% and 3.0% on the basis of ℓ_I, with 3 and 10 given labels per class, respectively. By comparing the performance of InfoGNN-IT and InfoGNN-ITS, we can find that the informativeness scores make distinct contributions under the two settings; for example, InfoGNN-ITS achieves an improvement of 3.3% with 3 given labels per class in contrast to an improvement of 0.5% with 10 given labels per class on Citeseer. This is because an increasing number of training labels counteracts the effect of informativeness scoring. The similar phenomenon can also be observed on the contribution of ℓ_{KL}. It also plays a more significant role at a lower label rate, where imbalanced predictions are more likely to occur.

5.6 Hyperparameter sensitivity analysis

We also conduct experiments to test the impact of hyperparameters (α, β, q, k and r) on the performance of InfoGNN. We take turns to test the effect of each hyperparameter while fixing the values of the rest. Due to space limit, we only report the results when 3 and 10 labels per class are given for training.

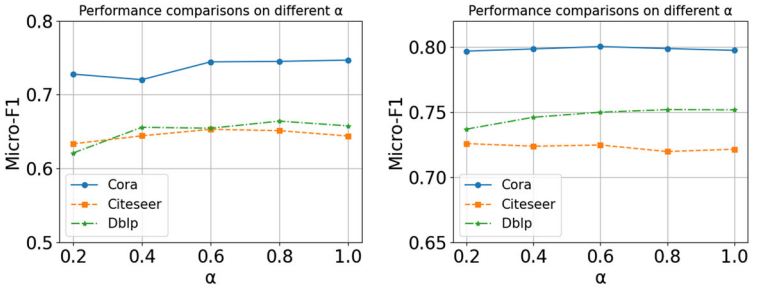
Hyperparameter α controls the contribution of the contrastive loss ℓ_I to the total loss, whose impact is shown in Fig. 2. With 3 given labels per class, we find that a larger α value can lead to better performance before α reaches 0.6. After that, the performance remains stable with very slight changes. With 10 labels per class provided, except on Dblp, the changes of α values do not largely impact model performance on Cora and Citeseer. This indicates that, when label information is very limited, our model requires stronger structural regularization to generate discriminative node embeddings. On the other hand, when label information is relatively sufficient, model training is dominated by the supervised loss from the given labels. Thus, the role of ℓ_I is more profound when given labels are scarce.

Figure 3 shows performance comparisons w.r.t. different values of β. With only 3 labels per class provided, since the class-imbalance problem is more likely to occur during pseudo-labeling, our model favors a larger β value for class-balance regularization in the pseudo-label set, as shown in Fig. 3a. As β increases from 0.1 to 1.0, our model boosts its classification accuracy by around 3% on Citeseer and Cora. When 10 labels per class are given, the class-imbalance problem is less likely to arise as more

Table 5 The Micro-F1 performance comparisons with various ablation studies

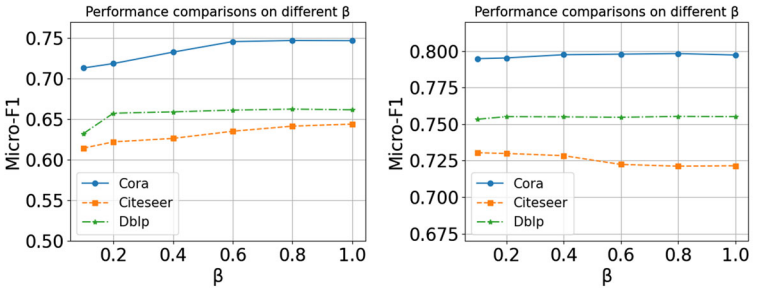
Method	Cora		Citeseer		Dblp		Wikics		Coauther_cs		Coauther_phy	
	3	10	3	10	3	10	3	10	3	10	3	10
GCN	0.616	0.742	0.504	0.602	0.583	0.652	0.550	0.682	0.799	0.893	0.849	0.901
InfoGNN-I	0.681	0.764	0.583	0.694	0.598	0.739	0.549	0.695	0.824	0.887	0.885	0.928
InfoGNN-IT	0.697	0.790	0.590	0.723	0.618	0.768	0.587	0.725	0.827	0.892	0.899	0.937
InfoGNN-ITS	0.720	0.792	0.623	0.728	0.646	0.766	0.593	0.723	0.827	0.886	0.906	0.937
InfoGNN	0.735	0.792	0.652	0.721	0.669	0.765	0.610	0.723	0.865	0.906	0.924	0.938

OOM indicates out-of-memory on a 32GB GPU



(a) α with 3 given labels per class **(b)** α with 10 given labels per class

Fig. 2 Sensitivity analysis w.r.t. hyperparameter α



(a) β with 3 given labels per class **(b)** β with 10 given labels per class

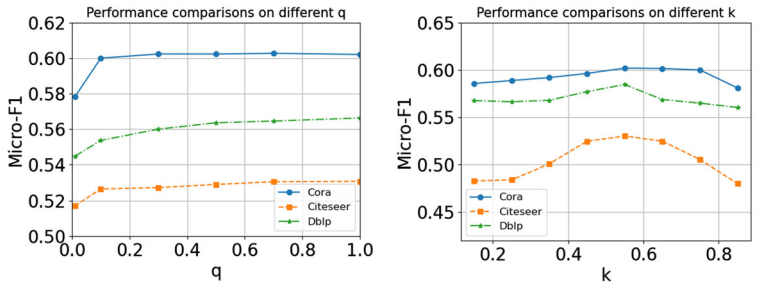
Fig. 3 Sensitivity analysis w.r.t. hyperparameter β

label information can be exploited. Overall, the change in β values does not largely impact model performance.

Hyperparameter q is the generalization coefficient in ℓ_T. Figure 4a illustrates model performance changes with an increase of q when one label per class is given. We can see that, as q rises, the performance of our method shows a gradual increase on the three datasets. This is because the severe lack of label information is more probable to incur noise in pseudo-labels. A larger q value could then decay the gradient update on unreliable samples, thereby improving model robustness. On the other hand, when q descends near zero, the GCE loss approaches close to SCE, and the model has a significant performance drop. This further proves the superiority of the GCE loss over SCE when only few labels are given for training.

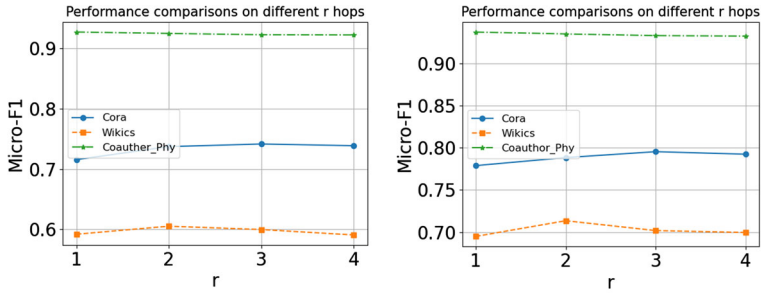
Hyperparameter k is the threshold for ℓ_T, which controls the number of unlabeled nodes selected for pseudo-labeling. Figure 4b shows model performance by varying k with one given label per class. As we can see, a medium k value achieves better accuracy, whereas too small or too large values degrade model performance.

Hyperparameter r indicates the number of hops for generating positive and negative subgraphs to calculate informativeness measures. Figure 5 shows how r affects model performance on the three datasets (Cora, Wikics, and Coauthor_Phy) with diverse topology characteristics. As depicted in this figure, for edge-sparse graphs (*e.g.*, Cora), a larger r tends to result in better performance. For edge-dense graphs



(a) q with one given label per class **(b)** k with one given label per class

Fig. 4 Sensitivity analysis w.r.t. the generalization coefficient q and the threshold k in ℓ_T



(a) r-hops with 3 given labels per class **(b)** r-hops with 10 given labels per class

Fig. 5 Sensitivity analysis w.r.t. number of hops r for sampled subgraphs

(*e.g.*, Coauther_Phy), a smaller r is more likely to exert better performance. For graphs with medium edge density (*e.g.*, Wikics), the best results are achieved with a medium r. When r exceeds 3 hops, model performance has a slight drop on all three datasets. In summary, our method favors a smaller subgraph scale on dense graphs, but a relatively larger scale on sparse graphs for sampling sufficient structural contexts in local neighborhoods.

6 Conclusion and future work

In this paper, we propose an informativeness augmented pseudo-labeling framework, called InfoGNN, to address semi-supervised node classification with few labels. We argue that current pseudo-labeling approaches on GNNs suffer from two major pitfalls: information redundancy and noisy pseudo-labels. To address these issues, we propose to quantify node informativeness based on MI estimation maximization. Taking both informativeness and prediction confidence into account, more informative unlabeled nodes are selected for pseudo-labeling. We then adapt a generalized cross entropy loss on pseudo-labels to mitigate the adverse effect of unreliable pseudo-labels. Furthermore, we apply a class-balanced regularization in response to the potential class-imbalance problem arising from pseudo-labeling. Extensive experimental results and ablation studies verify the effectiveness of our proposed framework, and demon-

strate its superior performance over state-of-the-art baseline models, especially under very few-label settings.

For future work, we plan to extend our proposed method from two aspects. First, although our proposed method employs a GCE loss to mitigate the negative effect of unreliable pseudo-labels, inaccurate model predictions still inevitably incur accumulated errors during pseudo-labeling. Thus, we will devise better strategies to combat such confirmation biases for pseudo-labeling. Second, we will generalize our current model to heterophilous graphs. Unlike homophilous graphs, where linked nodes tend to share the same labels, heterophilous graphs typically have edges connecting nodes from different classes. Therefore, this requires new informativeness measures for estimating node importance on heterophilous graphs.

Acknowledgements This work is supported by the USYD-Data61 Collaborative Research Project grant, the Australian Research Council under Grant DP210101347, and the China Scholarship Council under Grant 201806070131.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Appendix A: Table of symbols and notations

Table 6 summarizes a list of frequently used symbols and notations in this paper.

Table 6 A List of symbols and notations

Symbol	Description
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$	Undirected graph $\mathcal{G}$ with node set $\mathcal{V}$, edge set $\mathcal{E}$ and node feature matrix X
$\mathbf{x}_i$, $\mathbf{y}_i$	Feature vector and one-hot label of node v_i
$\mathcal{L}, \mathcal{U}$	Label set and unlabel set
$\mathcal{U}_p$	Selected pseudo-label set
h_v^k	Embedding of node v at k-th layer's propagation
A,$\tilde{A}$	Adjacent matrix and adjacent matrix with added self-connection
D, $\tilde{D}$	Degree matrix of A, $\tilde{A}$
A_r,D_r	r-hop adjacent matrix and the corresponding degree matrix
$s_c(v)$, $s_r(v)$	Confidence score and informativeness score for node v
$\mathcal{N}_v$	Neighbors of node v

Table 6 continued

Symbol	Description
$H_{\mathcal{N}_v}$	Neighboring node embedding matrix of node v
$\mathcal{D}(\cdot)$	Discriminator network
$\varphi(\cdot)$	MLP encoder
$\phi(\cdot)$	Subgraph encoder
$f_\Theta(\cdot)$	GNN parameterized by Θ
r	Number of hops for sampling subgraphs
$\ell_I(\cdot)$	Contrastive loss
$\ell_L(\cdot)$	Cross entropy (SCE) loss
$\ell_T(\cdot)$	Generalized cross entropy (GCE) loss
$\ell_{KL}(\cdot)$	Class-balanced regularization (CBR)

Appendix B: Stability analysis of the discriminator

In terms of the stability of the discriminator, we conduct a series of experiments to investigate the changes in its loss w.r.t. training epochs. The experiments are performed on Cora and Citeseer with 10 given labels per class. Figure 6 visually shows the changes in the loss during training. We can find that as the training proceeds, the loss rapidly decreases to a low level, and then stabilizes till the training ends. These results empirically show that the training of the discriminator is quite stable.

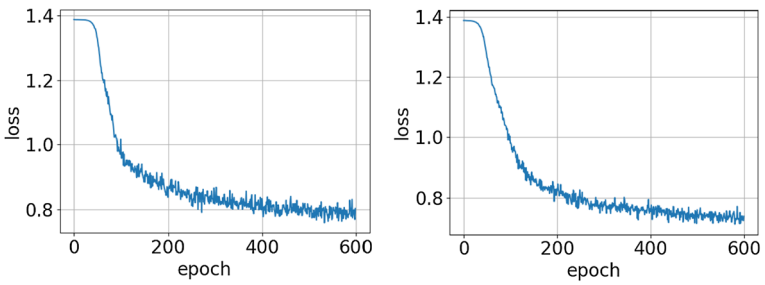


Fig. 6 The loss changes of the discriminator $\mathcal{D}(\cdot)$ with 10 given labels per class for training

References

Belghazi MI, Baratin A, Rajeshwar S, Ozair S, Bengio Y, Courville A, Hjelm D : Mutual information neural estimation. In: International conference on machine learning, pp 531–540. PMLR (2018)

Bojchevski A, Günnemann S (2018) Deep gaussian embedding of graphs: unsupervised inductive learning via ranking. In: International conference on learning representations

Box GE, Cox DR (1964) An analysis of transformations. J R Stat Soc: Ser B (Methodol) 26(2):211–243

Ding K, Wang J, Li J, Shu K, Liu C, Liu H (2020) Graph prototypical networks for few-shot learning on attributed networks. In: Proceedings of the 29th ACM international conference on information and knowledge management, pp 295–304

Goldberger J, Ben-Reuven E (2016) Training deep neural-networks using a noise adaptation layer. In: International conference on learning representations

Hamilton W, Ying Z, Leskovec J : Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp 1024–1034 (2017)

Han B, Yao Q, Yu X, Niu G, Xu M, Hu W, Tsang I, Sugiyama M (2018) Co-teaching: Robust training of deep neural networks with extremely noisy labels. In: Advances in neural information processing systems, pp 8527–8537

Hassani K, Khasahmadi AH (2020) Contrastive multi-view representation learning on graphs. In: International conference on machine learning, pp 4116–4126. PMLR

Hjelm RD, Fedorov A, Lavoie-Marchildon S, Grewal K, Bachman P, Trischler A, Bengio Y (2019) Learning deep representations by mutual information estimation and maximization. In: International conference on learning representations

Huang K, Zitnik M (2020) Graph meta learning via local subgraphs. In: Advances in neural information processing systems, pp 5862–5874

Kim D, Oh A (2021) How to find your friendly neighborhood: graph attention design with self-supervision. In: International conference on learning representations

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations

Lee DH, et al (2013) Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning. International conference on machine learning

Li J, Wong Y, Zhao Q, Kankanhalli MS (2019) Learning to learn from noisy labeled data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5051–5059

Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence, pp 3538–3545

Liu Y, Jin M, Pan S, Zhou C, Zheng Y, Xia F, Yu P (2022) Graph self-supervised learning: a survey. IEEE Trans Knowl Data Eng

Mernyei P, Cangea C (2020) Wiki-cs: A wikipedia-based benchmark for graph neural networks. arXiv preprint arXiv:2007.02901

Nowozin S, Cseke B, Tomioka R : f-gan: Training generative neural samplers using variational divergence minimization. In: Advances in Neural Information Processing Systems, pp. 271–279 (2016)

Peng Z, Huang W, Lu, M, Zheng Q, Rong Y, Xu T, Huang J (2020) Graph representation learning via graphical mutual information maximization. In: The web conference, pp 259–270

Qiu J, Chen Q, Dong Y, Zhang J, Yang H, Ding M, Wang K, Tang J (2020) Gcc: Graph contrastive coding for graph neural network pre-training. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1150–1160

Qu M, Bengio Y, Tang J (2019) Gmnn: Graph markov neural networks. In: International conference on machine learning, pp 5241–5250. PMLR

Ren M, Zeng W, Yang B, Urtasun R (2018) Learning to reweight examples for robust deep learning. In: International conference on machine learning, pp 4334–4343

Rosenberg C, Hebert M, Schneiderman H (2005) Semi-supervised self-training of object detection models. In: 2005 Seventh IEEE workshops on applications of computer vision **1**, pp 29–36

Shchur O, Mumme M, Bojchevski A, Günnemann S (2018) Pitfalls of graph neural network evaluation. In: Relational representation learning workshop, advances in neural information processing systems

Snell J, Swersky K, Zemel R (2017) Prototypical networks for few-shot learning. In: Advances in neural information processing systems, pp 4077–4087

Sukhbaatar S, Bruna J, Paluri M, Bourdev L, Fergus R (2015) Training convolutional networks with noisy labels. In: International conference on learning representations workshop

Sun K, Lin Z, Zhu Z (2020) Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In: Proceedings of the AAAI conference on artificial intelligence, pp 5892–5899

Van Rooyen B, Menon A, Williamson RC (2015) Learning with symmetric label noise: the importance of being unhinged. In: Advances in neural information processing systems, pp 10–18

Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: International conference on learning representations
Velickovic P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD (2019) Deep graph infomax. In: International conference on learning representations
Verma V, Qu M, Lamb A, Bengio Y, Kannala J, Tang J : Graphmix: Regularized training of graph neural networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence, pp 10024–10032 (2020)
Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K : Simplifying graph convolutional networks. In: International conference on machine learning, pp 6861–6871. PMLR (2019)
Xiao T, Xia T, Yang Y, Huang C, Wang X (2015) Learning from massive noisy labeled data for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2691–2699
You Y, Chen T, Sui Y, Chen T, Wang Z, Shen Y (2020) Graph contrastive learning with augmentations. In: Advances in neural information processing systems, pp 5812–5823
You Y, Chen T, Wang Z, Shen Y (2020) When does self-supervision help graph convolutional networks?. In: International conference on machine learning, pp 10871–10880. PMLR
Zhan K, Niu C (2021) Mutual teaching for graph convolutional networks. Futur Gener Comput Syst 115:837–843
Zhang Z, Sabuncu M (2018) Generalized cross entropy loss for training deep neural networks with noisy labels. In: Advances in neural information processing systems, pp 8778–8788 (2018)
Zhou Z, Zhang S, Huang Z (2019) Dynamic self-training framework for graph convolutional networks. arXiv preprint arXiv:1910.02684
Zhu Y, Xu Y, Yu F, Liu Q, Wu S, Wang L (2020) Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131