Knowledge and Information Systems (2022) 64:235–260
https://doi.org/10.1007/s10115-021-01635-8

REGULAR PAPER

Scalable self-supervised graph representation learning via enhancing and contrasting subgraphs

Yizhu Jiao[1] · Yun Xiong[1,2] · Jiawei Zhang[3] · Yao Zhang[1] · Tianqi Zhang[1] · Yangyong Zhu[1,2]

Received: 16 March 2021 / Revised: 25 November 2021 / Accepted: 26 November 2021 /
Published online: 19 January 2022

Abstract

Graph representation learning has attracted lots of attention recently. Existing graph neural networks fed with the complete graph data are not scalable due to limited computation and memory costs. Thus, it remains a great challenge to capture rich information in large-scale graph data. Besides, these methods mainly focus on supervised learning and highly depend on node label information, which is expensive to obtain in the real world. As to unsupervised network embedding approaches, they overemphasize node proximity instead, whose learned representations can hardly be used in downstream application tasks directly. In recent years, emerging self-supervised learning provides a potential solution to address the aforementioned problems. However, existing self-supervised works also operate on the complete graph data and are biased to fit either global or very local (1-hop neighborhood) graph structures in defining the mutual information-based loss terms. In this paper, a novel self-supervised representation learning method via Sub-graph Contrast, namely SUBG- CON, is proposed by utilizing the strong correlation between central nodes and their sampled subgraphs to capture regional structure information. Instead of learning on the complete input graph data, with

✉ Yizhu Jiao
yzjiao18@fudan.edu.cn

✉ Yun Xiong
yunx@fudan.edu.cn

Jiawei Zhang
jiawei@ifmlab.org

Yao Zhang
yaozhang18@fudan.edu.cn

Tianqi Zhang
tqzhang18@fudan.edu.cn

Yangyong Zhu
yyzhu@fudan.edu.cn

1 Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China

2 Shanghai Institute for Advanced Communication and Data Science, Fudan University, Shanghai, China

3 IFM Lab, Department of Computer Science, University of California, Davis, CA, USA

a novel data augmentation strategy, SUBG- CON learns node representations through a contrastive loss defined based on subgraphs sampled from the original graph instead. Besides, we further enhance the subgraph representation learning via mutual information maximum to preserve more topology and feature information. Compared with existing graph representation learning approaches, SUBG- CON has prominent performance advantages in weaker supervision requirements, model learning scalability, and parallelization. Extensive experiments verify both the effectiveness and the efficiency of our work. We compared it with both classic and state-of-the-art graph representation learning approaches. Various downstream tasks are done on multiple real-world large-scale benchmark datasets from different domains.

Keywords Self-supervised learning · Graph representation learning · Subgraph contrast · Graph neural networks

1 Introduction

Graph representation learning [10] has attracted much attention recently. Its basic idea is to extract the high-dimensional information in graph-structured data and embed it into low-dimensional vector representations. These node representation vectors can be potentially used in various downstream tasks such as node classification [17], link prediction [8], graph classification [19], and graph alignment [13]. Graph representation learning problems have been studied on graph data from many different domains such as social networks [3], chemical molecular graphs [20], and bio-medical brain graphs [32].

Most existing successful methods are based on graph neural networks (GNNs) [17,25,30, 33] , which learn nodes' contextualized representations via effective neighborhood information aggregation. These methods usually take a complete graph as the input, which can hardly be applied to large-scale graph data, e.g., Facebook and Twitter with millions or even billions of nodes. What's more, the inter-connected graph structure also prevents parallel graph representation learning, which is especially critical for large-sized graph data. In addition, most of these existing graph neural networks focus on supervised learning. They encode the graph structure into representation vectors with the supervision of label information. However, for real-world graph data, manual graph labeling can be very tedious and expensive, which becomes infeasible for large-scale graphs. To overcome this challenge, some works try unsupervised learning settings instead. They optimize models with objective functions defined for capturing node proximity [10] or reconstructing graph structures [18]. However, detached from supervision information, representations learned by such unsupervised approaches can hardly work well in downstream applications with specific task objectives [21].

Self-supervised learning [14] has recently emerged as a promising approach to overcome the dilemma of lacking available supervision. Its key idea is defining an annotation-free pretext task, which can generate surrogate training samples automatically without human annotation. These training samples are used to train an encoder for representation learning. In the field of computer vision, data augmentation [28], such as flipping or cropping, is commonly used for training sample generation, which can improve the generalization of self-supervised learning. However, due to the unordered vertexes and extensive connections in graph data, such existing techniques mentioned above cannot work anymore and new data augmentation methods for graph data specifically are needed.

Self-supervised graph representation learning is a new research problem, but there're still existing some prior works on this topic, e.g., Deep Graph Infomax [31] and Graphical

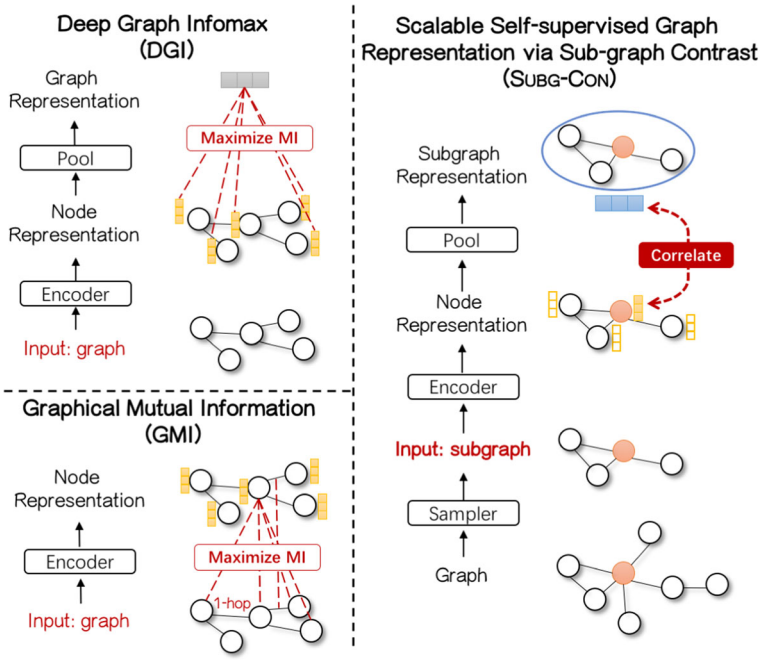


Fig. 1 An illustration of DGI (upper left), GMI (bottom left), and our proposed SUBG- CON (right). Little colored squares denote learnt representations. Red nodes denote central nodes of context subgraphs. SUBG-CON utilizes the strong correlation between central nodes and their context subgraphs sampled from the original graph. Note that SUBG- CON encodes the sampled subgraphs while the other two methods take the complete graph as the input. Besides, SUBG- CON captures structure information from regional neighborhoods instead of tending to be biased in fitting either the overall or very local (1-hop neighbor) graph structures

Mutual Information [23] (even though these approaches pose themselves as unsupervised models initially). Deep Graph Infomax (DGI) [31] introduces a global level pretext task to discriminate actual node representations from the corrupted ones based on the global graph representation. Graphical Mutual Information (GMI) [23] is centered about local structures by maximizing mutual information between the hidden representation of each node and the original features of its directly adjacent neighbors. As illustrated in the left of Fig. 1, these works tend to be biased in fitting either the overall or very local (1-hop neighbor) graph structures in defining the mutual information based loss terms, which would harm the quality of learned representations. Besides, these self-supervised works adopt a graph neural network as the encoder and also need to take the complete graph as the input, which restricts their scalability on large-sized graphs.

Intuitively, nodes and their regional neighbors are more correlated while other nodes that are very far away hardly influence them, especially in large-scale graphs. Therefore, subgraphs consisting of regional neighborhoods play a critical role to provide structure contexts for node representation learning. In this paper, we propose a novel scalable self-supervised graph representation via Sub-graph Contrast, SUBG- CON [35]. It takes strong correlation between central nodes and their regional subgraphs (involving both direct neighbors and other nodes that are further away) into consideration as illustrated in Fig. 1. More specifically, we introduce a data augmentation strategy on graphs firstly, including global-based and local-based subgraph sampling. The central nodes together with their closely related

surrounding nodes are sampled from the original graph to compose context subgraphs. Then, these subgraphs are fed into graph neural network encoders to obtain the representations of central nodes and subgraphs after pooling. Finally, a contrastive loss is introduced in the latent space to train the encoder to distinguish the generated positive and negative samples (to be introduced later), so that nodes with different regional structures can be well differentiated. Besides, to preserve topology and feature information as much as possible in subgraph representations, we enhance subgraph representation learning by maximizing the mutual information between the subgraph representations and the original context subgraphs. This enhanced graph representation learning method is denoted as SUBG- CON$^+$. Compared with previous methods operating on the complete graph structure, SUBG- CON and SUBG- CON$^+$ can capture regional information in context subgraphs of smaller sizes and simpler structures with lower time and space costs. Besides, based on sampled subgraph instances, SUBG- CON and SUBG- CON$^+$ are easy to parallelize, which is critical for large-sized graph data.

Through an empirical assessment on several benchmark graph datasets with different sizes from multiple diverse fields, we demonstrate that the representations learned by our proposed model are consistently competitive on various downstream tasks. It often outperforms both supervised and unsupervised strong baselines. We carefully studied the influence of different components in our framework on the model performance. Besides, we verify the efficiency of our proposed model, both on training time and computation memory, compared with state-of-the-art self-supervised methods that work on the complete graph.

To summarize, our major contributions include:

- We propose a novel self-supervised graph representation learning method via sub-graph contrast. It utilizes the correlation of central nodes and context subgraphs to capture regional graph structure information.
- We introduce a data augmentation strategy on graphs, which aims at increasing the training samples from the existing graph using subgraph sampling for self-supervised graph representation learning.
- By training with subgraphs of small sizes and simple structures, our proposed methods, SUBG- CON and SUBG- CON$^+$, require lower training time and computation memory costs for graph representation learning.
- Based on the sampled subgraph instances, our method enables parallel graph representation learning to further improve efficiency and scalability.
- Extensive experiments verify both the effectiveness and the efficiency of our work compared with prior unsupervised and supervised approaches on multiple real-world graph datasets from different domains.

2 Method

In this section, we will present our framework in a top-down fashion. It starts with an abstract overview of our specific subgraph-based representation learning setup, followed by an exposition of subgraph sampling based data augmentation, subgraph encoding for representations, enhanced subgraph representation learning, and our self-supervised pretext task for model optimization. Finally, we introduce parallel SUBG- CON briefly.

2.1 Subgraph-based self-supervised representation learning

Prior to going further, we first provide the preliminary concepts used in this paper. We assume a general self-supervised graph representation learning setup: For a graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, a set of node features are provided, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, where N is the number of nodes in the graph and $\mathbf{x}_i \in \mathbb{R}^F$ represents the features of dimension F for node i. We are also provided with relational information between these nodes in the form of an adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$. While $\mathbf{A}$ may consist of arbitrary real numbers (or even arbitrary edge features), in all our experiments we will assume the graphs to be unweighted, i.e., $\mathbf{A}(i, j) = 1$ if there exists an edge $i \to j$ in the graph and $\mathbf{A}(i, j) = 0$ otherwise.

Traditional graph representation methods target on training an encoder $\mathcal{E} : \mathbb{R}^{N \times F} \times \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times F'}$ to encode a complete graph, so that latent node representations $\mathbf{H} = \mathcal{E}(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times F'}$ can be produced, where F' is the dimension of latent representations. For convenience, we represent the learned representation of each node i as $\mathbf{h}_i$. These representations then are generated at once and retrieved for downstream tasks, such as node classification. However, due to limited computation time and memory, it remains a great challenge for traditional methods taking the complete graph structure as the input to handle large-scale graphs.

To overcome the limitation of traditional methods, we propose a novel subgraph-based representation learning approach. For a central node i, a subgraph sampler $\mathcal{S}$, i.e., a proxy of data augmentation, is designed to extract its context subgraphs $\mathbf{X}_i \in \mathbb{R}^{N' \times F}$ from the original graph. The context subgraph provides regional structure information for learning the representation of node i. $\mathbf{X}_i \in \mathbb{R}^{N' \times F}$ denotes the node features of the ith context subgraph. $\mathbf{A}_i$ denotes the relational information among node i and its neighbor nodes. N' indicates the context subgraph size. We target at learning a encoder for context subgraphs, $\mathcal{E} : \mathbb{R}^{N' \times F} \times \mathbb{R}^{N' \times N'} \to \mathbb{R}^{N' \times F'}$, which serves for acquiring node representations within context graphs. It should be noted that different from traditional methods, the input of the encoder are context subgraphs whose sizes are much smaller than the original graph. And node representations can be retrieved based on their context subgraph structures flexibly without the complete graph. Thus, by operating on sampled subgraph instances, SUBG- CON has prominent performance advantages in model learning scalability. Besides, it is easy to parallelize, which is critical for large-sized graph data.

Here we will focus on three key points for our subgraph-based self-supervised learning method: context subgraph extraction, subgraph encoding for representations, and the self-supervised pretext task for model optimization.

- For context subgraph extraction, the subgraph sampler $\mathcal{S}$ will serve as the proxy of data augmentation. It measures the importance scores of neighbors and samples a few closely related nodes to compose a context subgraphs which provide regional structure information for representation learning.
- For subgraph encoding, we target on encoding the structures and features of context subgraphs by the encoder $\mathcal{E}$, to produce the central node representations $\mathbf{h}_i$. The other key consequence is summarizing the subgraph centered around node i as the subgraph representation $\mathbf{s}_i$.
- For the self-supervised pretext task, it can optimize the encoder by taking advantage of the strong correlation between central nodes and their context subgraphs so that the regional information captured from the context subgraphs embeds into the central node representations.

2.2 Subgraph sampling based data augmentation

To overcome the dependence of manual labels, it is important for self-supervised learning to generate surrogate training samples automatically to train an encoder for representation learning. Data augmentation is a popular technique for training sample generation in computer vision. However, due to unordered vertexes and extensive connections, it hasn't been used explicitly in graph data. For self-supervised graph representation learning, we introduce the concept of data augmentation on graph formally here.

Definition 1 (*Data augmentation on graph*): Given a graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ denotes node features and $\mathbf{A}$ denotes relations, data augmentation is a strategy to produce a series of variant graphs $\mathcal{G}' = (\mathbf{X}', \mathbf{A}')$ using various transformations on features and relations of $\mathcal{G}$.

There are various transformations for graph data, such as node masking or feature corruption. In this paper, we adopt a subgraph sampling based data augmentation strategy. Because, intuitively, nodes and their regional neighborhoods are more correlated while long-distance nodes hardly influence them. This assumption is more reasonable as the size of graphs increases. Therefore, we sample a series of subgraphs including regional neighborhoods from the original graph as training data.

The most critical issue now is to sample a context subgraph, which can provide sufficient structure information for learning a high-quality representation for the central node. Here we provide several strategies for sampling subgraphs, analyze the algorithm complexity of different strategies, and evaluate them in the following experimental part.

2.2.1 Global structure based strategy

The first sampling strategy is based on personalized pagerank algorithm [12] as introduced in [38]. Considering the importance of different neighbors varies, for a specific node i, the subgraph sampler $\mathcal{S}$ first measures the importance scores of other neighbor nodes by personalized pagerank algorithm. Given the relational information between all nodes in the form of an adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$, the importance score matrix $\mathbf{S}$ can be denoted as

$$\mathbf{S} = \alpha \cdot (\mathbf{I} - (1 - \alpha) \cdot \bar{\mathbf{A}}), \tag{1}$$

where $\mathbf{I}$ is the identity matrix and $\alpha \in [0, 1]$ is a parameter which is always set as 0.15. $\mathbf{D}$ denotes as the corresponding diagonal matrix with $\mathbf{D}(i, i) = \sum_j \mathbf{A}(i, j)$ on its diagonal and $\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ denotes the column-normalized adjacency matrix. $\mathbf{S}(i, :)$ is the importance scores vector for node i, which indicates its correlation with other nodes.

It is noted that the importance score matrix S can be precomputed before model training starts. And we implement node-wise PPR to calculate importance scores to reduce computation memory, which makes our method more suitable to work on large-scale graphs.

2.2.2 Local structure based strategy

Sampling using the global structure can take the topological structure of the complete graph into consideration, but the computational complexity is relatively high. Inspired by [5], the second subgraph sampling strategy introduced here is a random walk based method, which captures local structural features to build the most relevant subgraph connecting a central node in a large graph. Similar to the first strategy, the extracted subgraphs contain the most important edges and the nodes. We measure the importance score of an edge with the expected

number of times it is visited along random walks connecting the central node. These expected times can reflect both local graph structures and the edge weights. These expected passage times can be obtained in a probability matrix with basic Markov chain theory [5]. But in the straightforward implementation, it is time-consuming since it relies on matrix inversions, which are usually performed with cubic time complexity in terms of the number of nodes in the graph.

To reducing the high computational demanding, here we approximate the probability matrix by repeated random walk experiments. For each central node, we perform random walks from it repeatedly and limit the walks to a maximal number of the specific step. Then, we count the average number of times that all nodes have been passed. The importance score matrix $\mathbf{S}$ can be obtained by measuring the average importance scores of different neighbor nodes.

Here, the step of walks and the number of central nodes can be typically fixed, whose magnitude of the value is less than the number of graph edges. Thus, this strategy essentially provides linear time complexity relative to the number of graph edges. By restricting the walking steps, it can be convenient from a computational point of view. Besides, it controls the level of locality when connecting central nodes. In this way, the extracted subgraphs will be avoided from being too diffused or too local.

As such the sampling strategy computes edge and node relevance from random walks connecting the central nodes. A subgraph is obtained by keeping only those edges above a minimal importance threshold. In our experiments, the relevance importance is automatically fixed such that the subgraphs extracted by the selected edges is weakly connected.

Besides, the importance scores for edges can also be computed by this strategy and serve as new edge weights. It can then be run on the input graph with updated weights. This iterative process may be repeated a number of times to increase the discrimination between more and less important edges. We can use a parameter to determine how often it is iterated. In the subsequent process of encoding the subgraphs, the newly generated edge weights can be token into consideration to obtain better representations of the subgraphs.

So far, according to two sampling strategies, the importance score matrix $\mathbf{S}$ can be acquired. For a specific node i, the subgraph sampler $\mathcal{S}$ chooses top-K important neighbors to constitute a subgraph with the score matrix $\mathbf{S}$. The index of chosen nodes can be denoted as

$$idx = top\_rank(\mathbf{S}(i, :), K),$$

where top_rank is the function that returns the indices of the top K values and K denotes the size of context graphs.

The subgraph sampler $\mathcal{S}$ will process the original graph with the node index to obtain the context subgraph $\mathcal{G}_i$ of node i. Its adjacency matrix $\mathbf{X}_i$ and feature matrix $\mathbf{A}_i$ are denoted, respectively, as

$$\mathbf{X}_i = \mathbf{X}_{idx,:}, \quad \mathbf{A}_i = \mathbf{A}_{idx,idx},$$

where $\cdot_{idx}$ is an indexing operation. $\mathbf{X}_{idx,:}$ is the row-wise (i.e., node-wise) indexed feature matrix. $\mathbf{A}_{idx,idx}$ is the row-wise and col-wise indexed adjacency matrix corresponding to the induced subgraph.

So far, we can acquire the context subgraph $\mathcal{G}_i = (\mathbf{X}_i, \mathbf{A}_i) \sim \mathcal{S}(\mathbf{X}, \mathbf{A})$ for any specific node i. For large-sized input graphs, this procedure can support parallel computing to further improve efficiency. These context subgraphs produced by data augmentation can be decomposed into several mini-batches and fed to train SUBG- CON.

2.3 Encoding subgraph for representations

Given the context subgraph $\mathcal{G}_i = (\mathbf{X}_i, \mathbf{A}_i)$ of a central node i, the encoder $\mathcal{E} : \mathbb{R}^{N' \times F} \times \mathbb{R}^{N' \times N'} \rightarrow \mathbb{R}^{N' \times F'}$ encodes it to obtain the latent representations matrix $\mathbf{H}_i$ denoted as

$$\mathbf{H}_i = \mathcal{E}(\mathbf{X}_i, \mathbf{A}_i),$$

Here we adopt graph neural networks (GNN), a flexible class of node embedding architectures, as the encoder $\mathcal{E}$. Node representations are generated by aggregating information from neighbors. We study the impact of different graph neural networks in the experiments and will discuss later. The central node embedding $\mathbf{h}_i$ is picked from the latent representations matrix $\mathbf{H}_i$

$$\mathbf{h}_i = \mathcal{C}(\mathbf{H}_i),$$

where $\mathcal{C}$ denotes the operation picking out the central node embedding.

As mentioned before, the other key consequence is summarizing the subgraph centered around node i as the context subgraph representation $\mathbf{s}_i$. In order to obtain the subgraph-level summary vectors, we leverage a readout function, $\mathcal{R} : \mathbb{R}^{N' \times F'} \rightarrow \mathbb{R}^{F'}$, and use it to summarize the obtained node representations into a subgraph-level representation, $\mathbf{s}_i$, denoted as

$$\mathbf{s}_i = \mathcal{R}(\mathbf{H}_i).$$

So far, the representations of central nodes and context subgraphs are produced, which will play a key role in the generation of positive and negative samples for self-supervised pretext tasks.

2.4 Enhanced subgraph representation learning

We plan to utilize the strong correlation between central nodes and subgraphs to design self-supervised learning pretext tasks. Therefore, the quality of subgraph representations is crucial for self-supervised learning. According to the previous section, currently, the subgraph representations are obtained after a readout function based on the node representation, which is difficult to ensure that the subgraph representations can fully inherit the structural information. Therefore, to preserve the topology and feature information as much as possible in the subgraph representations, we want to enhance subgraph representation learning by maximizing the mutual information between the subgraph representations and the original context subgraphs.

For any specific node i, given its context subgraph $\mathcal{G}_i = (\mathbf{X}_i, \mathbf{A}_i)$ and subgraph representation $\mathbf{s}_i$, the mutual information between them is denoted as $I(\mathbf{s}_i; \mathcal{X}_i)$. According to the information theory, $I(\mathbf{s}_i; \mathbf{X}_i)$ can be defined as

$$I(\mathbf{s}_i; \mathbf{X}_i) = \int_{\mathbf{s}} \int_{\mathbf{X}} p(\mathbf{s}_i, \mathbf{X}_i) \log \frac{p(\mathbf{s}_i, \mathbf{X}_i)}{p(\mathbf{s}_i)\, p(\mathbf{X}_i)} \mathrm{d}\mathbf{s}_i \mathrm{d}\mathbf{X}_i,$$

where $p(\mathbf{X}_i)$ denotes the empirical probability distribution of node features $\mathbf{X}_i$; $p(\mathbf{s}_i)$ denotes the probability of $\mathbf{s}_i$; and $p(\boldsymbol{h}_i, \boldsymbol{X}_i)$ is the joint distribution.

Here we suppose the conditional probability $p(\mathbf{s}_i \mid \mathbf{X}_i)$ is multiplicative. Inspired by [23], $I(\mathbf{s}_i; \mathbf{X}_i)$ can be decomposed as a weighted sum of mutual information between the subgraph and its nodes, namely

$$I\left(\mathbf{s}_i; \mathbf{X}_i\right) = \sum_{j}^{n_i} w_{ij} I\left(\mathbf{s}_i; \mathbf{x}_j\right),$$

where $\mathbf{x}_j$ is the initial feature of the j-th node in the subgraph $\mathcal{G}_i$. n_i is the number of all nodes in $\mathcal{G}_i$. The weight w_{ij} satisfies $\frac{1}{n_i} \leq w_{ij} \leq 1$ for each j. It can be set to a fixed value, for example, the edge importance in the second subgraph sampling strategy. But by introducing more subgraph structure attributes, we train a learnable weight to relate its underlying distribution with the subgraph topology. More specifically, we denote the mutual information between the subgraph representations and the original context subgraphs as

$$I\left(\mathbf{s}_i; \mathcal{G}_i\right) := \sum_{j}^{n_i} w_{ij} I\left(\mathbf{s}_i; \mathbf{x}_j\right) + I\left(w_{ij}; \mathbf{a}_{ij}\right),$$

where $\mathbf{a}_{ij}$ represents the score of node j in the importance matrix of the subgraph $\mathcal{G}_i$. $w_{ij} = \sigma\left(\mathbf{s}_i^{\mathrm{T}} \mathbf{h}_j\right)$ indicates the relevance between the subgraph representation and the node representation. We hope to enhance the quality of learning by maximizing the mutual information between the original subgraphs and the representations. The objective function and training details will be introduced later.

2.5 Contrastive learning via central node and context subgraph

The key idea of self-supervised contrastive learning is defining an annotation-free pretext task and then generating positive and negative samples. The encoder can be trained by contrasting positive and negative examples. As the prerequisite for ensuring the quality of learned representations, if the pretext task can fully inherit the rich information in graphs, we can obtain better representations to support subsequent mining tasks without additional guidance.

Intuitively, nodes are dependent on their regional neighborhoods and different nodes have different context subgraphs. This assumption is even more reasonable in large-scale graphs. At the same time, the complete structure of large-scale graphs is still hard to handle by existing node representation learning methods. Therefore, we consider the strong correlation between central nodes and their context subgraphs to design a self-supervision pretext task. The architecture of SUBG- CON is fully summarized in Fig. 2.

Our approach for learning the encoder relies on, for a specific central node, contrasting its real context subgraph with a fake one. Specifically, for the node representation, $\mathbf{h}_i$, that captures the regional information in the context subgraph, we regard the context subgraph representation $\mathbf{s}_i$ as positive sample. On the other hand, for a set of subgraph representations, we employ a function, $\mathcal{P}$, to corrupt them to generate negative samples, denoted as

$$\{\widetilde{\mathbf{s}}_1, \widetilde{\mathbf{s}}_2 ..., \widetilde{\mathbf{s}}_M\} \sim \mathcal{P}(\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_m\}),$$

where m is the size of the representation set. The corruption strategy determines the differentiation of nodes with different contexts, which is crucial for some downstream tasks, such as node classification.

As to the objective, related works use a noise-contrastive type objective with a standard binary cross-entropy loss between positive examples and negative examples [31]. However, as these context subgraphs are extracted from the same original graph and overlap with each other, we suppose that it can be harmful for representation learning if positive and negative examples to be distinguished absolutely. Therefore, we use the margin triplet loss [27] for

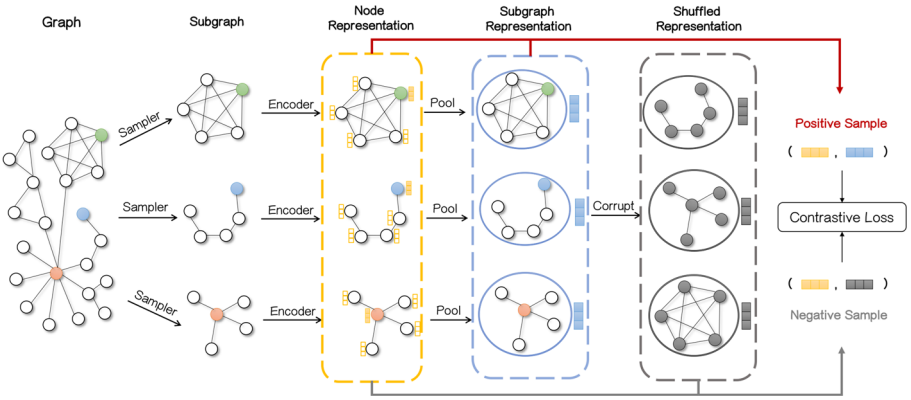


Fig. 2 Architecture of SUBG- CON. A series of context subgraphs are sampled from the original graph and fed into the encoder to obtain the representations of central nodes and subgraphs after pooling. For a specific node, the representation of its context subgraph is regarded as a positive sample. And we corrupt the subgraph representation to produce a negative sample. The contrastive loss in the latent space will force the encoder to recognize positive and negative samples so that different nodes can be well discriminated based on regional structure information

Algorithm 1 Optimization Algorithm.

Input: A graph $\mathcal{G}$ with input feature $\mathbf{X}$ and adjacency matrix $\mathbf{A}$; Subgraph sampler $\mathcal{S}$; Encoder $\mathcal{E}$; Readout function $\mathcal{R}$; Corruption function $\mathcal{P}$.

Precompute importance score matrix $\mathbf{S}$ according to Eq. 1.
while not converge **do**
  Sample context subgraphs $\{(\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2), ..., (\mathbf{X}_m, \mathbf{A}_M)\}$ where $\mathbf{H}_i = \mathcal{S}(\mathbf{X}_i, \mathbf{A}_i)$ and M is the number of the subgraphs.
  for all each subgraph $(\mathbf{X}_i, \mathbf{A}_i)$ **do**
    Encode the subgraph to obtain latent representation matrixes $\mathbf{H}_i = \mathcal{E}(\mathbf{X}_i, \mathbf{A}_i)$.
    Obtain the central node representation $\mathbf{h}_i = \mathcal{C}(\mathbf{H}_i)$.
    Summarize the subgraph representation through the readout function $\mathbf{s}_i = \mathcal{R}(\mathbf{H}_i)$.
  end for
  Corrupt the subgraph representations to generate negative examples for the corresponding node representations $\{\widetilde{\mathbf{s}}_1, \widetilde{\mathbf{s}}_2, ..., \widetilde{\mathbf{s}}_M\} = \mathcal{P}(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_M)$.
  Update parameters of $\mathcal{E}$ and $\mathcal{R}$ by applying gradient descent to maximize Eq.2 and Eq.3.
end while

model optimization so that positive and negative samples can be well discriminated to some extent and high-quality representations can be obtained. The loss is denoted as

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{(\mathbf{X},\mathbf{A})}(-\max(\sigma(\mathbf{h}_i\mathbf{s}_i) - \sigma(\mathbf{h}_i\widetilde{\mathbf{s}}_i) + \epsilon, 0)), \tag{2}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function and ϵ is margin value.

In addition, as introduced in the previous section, we attempt to enhance the subgraph representations by maximizing mutual information. The generated positive and negative samples are also used to optimize the mutual information of the subgraph and the representations. Specifically, the objective function is defined as

$$\mathcal{L}' = \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{(\mathbf{X},\mathbf{A})}(I(\mathbf{s}_i; \mathcal{G}_i) - I(\widetilde{\mathbf{s}}_i; \mathcal{G}_i)) \tag{3}$$

We summarize the steps of the procedure of our approach in Algorithm 1. It is noted that, to distinguish the variant of SUBG- CON with enhanced subgraph representation learning, we denote it as SUBG- CON^{+}.

In the training phase, we set the number of the subgraphs as M, the subgraph size as K, the number of features as F, and the number of nonzeros in the adjacency matrix of subgraphs as $\|A\|_0$. Based on these, for the proposed method, its time complexity is $O(M\|A\|_0 F)$ and its space complexity is $O(MKF)$.

2.6 Parallelizability

Compared with existing methods that input the complete graph data, it is parallelizable to operate on context subgraphs. On the one hand, subgraph extraction is easy to parallelize. Several random workers (in different threads, processes, or machines) can simultaneously explore different parts of the same graph for context subgraphs extraction. On the other hand, without the need for global computation that needs the whole graph structure, it becomes possible to encoder multiple subgraphs synchronously to obtain the representations of central nodes and subgraphs. Benefit from the parallelizability, our model can be scaled efficiently on larger-size graphs.

3 Experiment

In this section, we conduct extensive experiments to verify both the effectiveness and the efficiency of SUBG- CON on a variety of tasks on multiple real-world datasets from different domains, such as node classification and link prediction. In each case, SUBG- CON is used to learn node representations in a fully unsupervised manner. We compare our approach with prior unsupervised and supervised strong baselines. Besides, we analyze the design of our architecture, including the encoder architecture and the objective function. We also do experiments about the efficiency including training time and memory usage. Reducing the number of training subgraphs and parallelization are studied to further improve efficiency. Lastly, parameter sensitivity analysis helps to choose suitable parameters for our approach.

3.1 Datasets

To assess the effectiveness of the representation learned by our work, we conduct experiments on multiple real-world datasets from different domains. We choose three popular small-scale datasets widely used in related works [17] (Cora, Citeseer, and Pubmed) and three large-scale datasets to verify the scalability of our approach (PPI, Flickr, and Reddit) [17,36]. It includes three citation networks, two social networks, and a protein network. Further information on the datasets is found in Table 1.

We set up the experiments on the following benchmark classification tasks: (1) classifying research papers into topics on the Cora, Citeseer, and Pubmed citation networks; (2) classifying protein roles within protein–protein interaction (PPI) networks, requiring generalization to unseen networks; (3) categorizing types of images based on the descriptions and common properties of Flickr online; (4) predicting the community structure of a social network modeled with Reddit posts.

Table 1 Dataset statistics

	Dataset	Type	Nodes	Edges	Degree	Features	Classes
Small scale	Cora	Citation	2708	5429	4.0	1433	7
	Citeseer	Citation	3327	4732	2.8	3703	6
	Pubmed	Citation	19,717	44,338	4.5	500	3
Large scale	PPI	Protein	56,944	818,716	28.8	50	121
	Flickr	Social	89,250	899,756	20.2	500	7
	Reddit	Social	232,965	11,606,919	99.6	602	41

3.2 Experimental settings

Encoder design For six different datasets, we study the impact of different graph neural networks (described below) and employ distinct encoders appropriate to that setting.

For Cora, Citeseer, Pubmed, and PPI, we adopt a one-layer Graph Convolutional Network (GCN) with skip connections [37] as our encoder, with the following propagation rule:

$$\mathcal{E}(\mathbf{X}, \mathbf{A}) = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W} + \hat{\mathbf{A}}\mathbf{W}_{skip}),$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix with inserted self-loops and $\hat{\mathbf{D}}$ is its corresponding degree matrix. For the nonlinearity σ, we apply the parametric ReLU (PReLU) function [11]. $\mathbf{W}$ is a learnable linear transformation applied to every node and $\mathbf{W}_{skip}$ is a learnable projection matrix for skip connections.

For Reddit and Flickr, we adopt a two-layer GCN model as our encoder, with the following propagation rule:

$$GCN(\mathbf{X}, \mathbf{A}) = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}),$$
$$\mathcal{E}(\mathbf{X}, \mathbf{A}) = GCN(GCN(\mathbf{X}, \mathbf{A}), \mathbf{A}),$$

where the latent representations produced by the first layer of GCN are fed as the input of the second layer.

Corruption functions The corruption function generates negative samples for our self-supervised task to make nodes with different contexts well distinguished, which is important for the node classification task. For convenience of computation, given a set of context subgraph representations, our corruption function shuffles them randomly. The subgraph representation of other central nodes is regarded as the negative sample so that nodes are closely related to their context subgraphs and weakly associated with other subgraphs. For learning node representations toward other kinds of tasks, the design of appropriate corruption strategies remains an area of open research. In the experiment, we also tried other several corruption functions, such as randomly sampling subgraphs, inversing the original subgraphs, and randomly modifying the initial node features and edges of the subgraphs.

Readout functions For all six experimental datasets, we employ the identical readout function with a simple averaging of all the nodes' features:

$$\mathcal{R}(\mathbf{H}) = \sigma(\frac{1}{N'}\sum_{i=1}^{N'}\mathbf{h}_i),$$

where σ is the logistic sigmoid nonlinearity. We assume that this simple readout is efficient for subgraphs of small sizes when we have found it to perform the best across all our experiments.

Table 2 Performance comparison with different methods on node classification

Algorithm	Available data	Cora	Citeseer	Pubmed
Raw features	**X**	56.6 ± 0.4	57.8 ± 0.2	69.1 ± 0.2
DeepWalk	**A**	67.2	43.2	65.3
Unsup-GraphSAGE	**X**, **A**	75.2 ± 1.5	59.4 ± 0.9	70.1 ± 1.4
DGI	**X**, **A**	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6
GMI	**X**, **A**	83.0 ± 0.3	73.0 ± 0.3	79.9 ± 0.2
GCN	**X**, **A**, **Y**	81.4 ± 0.6	70.3 ± 0.7	76.8 ± 0.6
GAT	**X**, **A**, **Y**	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
FastGCN	**X**, **A**, **Y**	78.0 ± 2.1	63.5 ± 1.8	74.4 ± 0.8
GraphSAGE	**X**, **A**, **Y**	79.2 ± 1.5	71.2 ± 0.5	73.1 ± 1.4
SUBG- CON	**X**, **A**	**83.5 ± 0.5**	**73.2 ± 0.2**	**81.0 ± 0.1**
SUBG- CON$^+$	**X**, **A**	**83.8 ± 0.5**	**73.5 ± 0.4**	**81.3 ± 0.1**
Algorithm	Available data	PPI	Flickr	Reddit
Raw features	**X**	42.5 ± 0.3	20.3 ± 0.2	58.5 ± 0.1
DeepWalk	**A**	52.9	27.9	32.4
Unsup-GraphSAGE	**X**, **A**	46.5 ± 0.7	36.5 ± 1.0	90.8 ± 1.1
DGI	**X**, **A**	63.8 ± 0.2	42.9 ± 0.1	94.0 ± 0.1
GMI	**X**, **A**	65.0 ± 0.0	44.5 ± 0.2	95.0 ± 0.0
GCN	**X**, **A**, **Y**	51.5 ± 0.6	48.7 ± 0.3	93.3 ± 0.1
GAT	**X**, **A**, **Y**	**97.3 ± 0.2**	**OOM**	**OOM**
FastGCN	**X**, **A**, **Y**	63.7 ± 0.6	48.1 ± 0.5	89.5 ± 1.2
GraphSAGE	**X**, **A**, **Y**	51.3 ± 3.2	**50.1 ± 1.3**	92.1 ± 1.1
SUBG- CON	**X**, **A**	**66.9 ± 0.2**	**48.8 ± 0.1**	**95.2 ± 0.0**
SUBG- CON$^+$	**X**, **A**	**67.7 ± 0.5**	**49.0 ± 0.3**	**95.3 ± 0.1**

The second column illustrates the data used by each algorithm in the training phase, where **X**, **A**, and **Y** denotes features, adjacency matrix, and labels, respectively
OOM Out of memory

Objective functions We compare the margin loss [27] against other commonly used contrastive loss functions, such as logistic loss [22], and bayesian personalized ranking (BPR) loss [26]. Table 3 shows these three objective function. Their impacts will be discussed later.

Sampling strategies In addition to the two subgraph sampling strategies mentioned in the method section, namely the global-based method and the local-based method, the experimental part also considers two naive strategies. One is to randomly sample a connected block from the neighborhood as a context subgraph for the central node. The scale of such subgraphs is the same with the previous two methods. The other is to contain all the 1-hop neighbors of the central node, that is, the 1-hop ego networks. We will explore the most suitable subgraph sampling strategy for different datasets.

Implementation details We implemented the baselines and SUBG- CON using PyTorch [15] and the geometric deep learning extension library [6]. The experiments are conducted on 8 NVIDIA TITAN Xp GPUs. SUBG- CON is used to learn node representations in a fully unsupervised manner, followed by evaluating the node-level classification with these representations. This is performed by directly using these representations to train and test a simple linear (logistic regression) classifier. In preprocessing, we perform row normalization

Table 3 Studied objective functions

Name	Objective function
Margin loss	$-\max(\sigma(\mathbf{hs}) - \sigma(\mathbf{h\widetilde{s}}) + \epsilon, 0)$
Logistic loss	$\log\sigma(\mathbf{hs}) + \log\sigma(-\mathbf{h\widetilde{s}})$
BPR loss	$\log\sigma(\mathbf{hs} - \mathbf{h\widetilde{s}})$

on Cora, Citeseer, PubMed following [17] and apply the processing strategy in [9] on Reddit, PPI, and Flickr. Especially, for PPI, suggested by [31], we standardize the learned embeddings before feeding them into the logistic regression classifier. During training, we use Adam optimizer [16] with an initial learning rate of 0.001 (specially, 10^{-5} on Citeseer and Reddit). The subgraph size is no more than 20 (specially, the subgraph size is 10 on Citeseer due to better performance). The dimension of node representations is 1024. The margin value ϵ for the loss function is 0.75.

Baselines We choose two state-of-art self-supervised methods, DGI [31] and GMI [23], which both learn graph embeddings by leveraging mutual information maximization. Two traditional unsupervised methods, DeepWalk [24] and unsupervised variants of GraphSAGE (abbreviated as Unsup-GraphSAGE) [9] are also compared with our model. Specially, we provide results for training the logistic regression on raw input features. Besides, we report experiment results on three supervised graph neural networks, GCN [17], GAT [30], FastGCN [3] and supervised GraphSAGE [9]. Notably, we reuse the metrics already reported in original papers or choose optimal hyper-parameters carefully after reproducing the code for different baselines in this paper to ensure the fairness of comparison experiments.

It is noted that, to verify the effectiveness of enhanced subgraph representation learning, we refer to the models with/without this part as SUBG- CON$^+$ and SUBG- CON, respectively.

Evaluation metrics For the node classification task, we provide the learned embeddings across the training set to the logistic regression classifier and give the results on the test nodes [23]. Followed [31], we adopt the mean classification accuracy to evaluate the performance for three benchmark datasets (Cora, Citeseer, and Pubmed), while the micro-averaged F1 score averaged is used for the other three larger datasets (PPI, Flickr, and Reddit). All datasets follow "fixed-partition " splits.

For the link prediction task, some edges are hidden in the input graph and the goal is to predict the existence of these edges based on the computed embeddings. We adopted AUC as the evaluation metrics, which is equal to the probability that a randomly chosen edge is ranked higher than a randomly chosen negative edge. The higher AUC score a approach reaches, the better performance it achieves. We report the AUC score averaged after 10 runs. All the existing links in the graph datasets are used as the positive link set. We sample a subset of unknown links among nodes in the original graph randomly as the negative link set, which is of the same size of the positive social link set. We randomly sample 50% of the positive and negative edges as train set, 5% as validation set, and 45% as test set.

3.3 Node classification

The results of our comparative evaluation experiments are summarized in Table 2. The results demonstrate our strong performance can be achieved across all six datasets. Our method successfully outperforms all the competing self-supervised approaches, thus verifying the potential of methods based on graph regional structure information in the node classification domain. Due to the enhanced subgraph representations, the performance of SUBG- CON$^+$

achieves further improvement. We also observe that all self-supervised methods are more competitive than traditional unsupervised baselines that rely on proximity-based objectives. It indicates our data augmentation strategy for self-supervised learning can make a greater contribution to models to capture high-level information in complex graphs even if these supervision signals are not frankly related to the node classification task. Besides, we particularly note that the DGI approach is competitive with the results reported for three supervised graph neural networks, even exceeding its performance on the Cora, Citeseer, Pubmed, and Reddit datasets. However, on PPI the gap is still large-we believe because our encoder is heavily dependent on node original features while available features on PPI are extremely sparse (over 40% of the nodes having all-zero features).

3.4 Link prediction

The results of our evaluation experiments on link prediction are summarized in Table 4. As we can see, both of our methods, SUBG- CON and SUBG- CON$^+$, outperform other strong baselines in most of the datasets. In Cora, Citeseer, and Pubmed, our method achieves a mean AUC score gain of nearly 2%. In Flickr, the gain is moderately lower, but still more than 1%, on average. However, in PPI and Reddit, GMI performs slightly better than our method with a gain of less than 0.5%. We explain that for dense graphs, proximity similarity is enough for predicting the potential connection between two nodes. Therefore, GMI that emphasizes learning the local structure can also take an advantage. Moreover, in most cases, the performance of our proposed methods lies in between the best and worst performing semi-supervised method. By comparing SUBG- CON and SUBG- CON$^+$, we can find that the latter can show obvious advantages on the six datasets owing to enhanced sub-graph representations.

3.5 Design of architectures

In this section, we will explore the design of each part of the framework, including , the subgraph encoder, the corruption function, the objective function, and the subgraph sampling strategy. For the convenience, all the experimental results reported in this section are on the node classification task .

3.5.1 Design of encoder

For better architecture and performance, we conducted experiments about the design of our encoder. We choose four different graph neural networks as the encoder to learn node representation, including graph convolutional network (GCN), graph convolutional network with skip connection (GCN + Skip), graph attention network (GAT) [30], graph isomorphism network (GIN) [34]. The experimental results are listed in Table 5.

As can be observed, GCN with skip connection can achieve the best performance on Citeseer, Pubmed, and PPI. Although GAT can be competitive on Cora, because GAT requires more training time and memory, we choose GCN with skip connection as our encoder finally. It is noted that, even if GCN is not the best choice on these three datasets, but compared in Table 2, our method with GCN as encoder still outperforms supervised GCN. For the other two larger datasets, Flickr and Reddit, 2-layer GCN is the best option. We assume higher-level information captured in the large-scale graphs can make contributions to improve the quality of the learned representations. To sum up, compared with the complete graphs with large scales and complex structures, subgraphs can be well encoded with simple graph neural

Table 4 Performance comparison with different methods on link prediction

Algorithm	Available data	Cora	Citeseer	Pubmed
Raw features	**X**	79.4 ± 0.01	75.5 ± 0.01	81.2 ± 0.02
DeepWalk	**A**	83.1 ± 0.01	80.5 ± 0.02	84.4 ± 0.00
Unsup-GraphSAGE	**X**, **A**	88.2 ± 0.9	94.1 ± 1.0	92.1 ± 1.4
DGI	**X**, **A**	89.8 ± 0.8	95.5 ± 1.0	91.2 ± 0.6
GMI	**X**, **A**	87.2 ± 1.0	93.5 ± 0.5	91.9 ± 0.7
GCN	**X**, **A**, **Y**	88.1 ± 0.8	95.3 ± 0.9	96.8 ± 0.6
GAT	**X**, **A**, **Y**	90.0 ± 0.4	97.6 ± 0.7	98.5 ± 0.2
FastGCN	**X**, **A**, **Y**	88.0 ± 1.5	94.5 ± 1.2	95.4 ± 0.8
GraphSAGE	**X**, **A**, **Y**	89.2 ± 1.3	95.2 ± 0.6	96.1 ± 0.4
SUBG- CON	**X**, **A**	**91.5 ± 0.6**	**97.1 ± 0.5**	**93.9 ± 0.2**
SUBG- CON$^+$	**X**, **A**	**92.1 ± 0.4**	**97.2 ± 0.6**	**94.2 ± 0.4**
Algorithm	Available data	PPI	Flickr	Reddit
Raw features	**X**	67.4 ± 0.1	74.3 ± 0.02	89.5 ± 0.01
DeepWalk	**A**	78.4 ± 0.01	79.3 ± 0.01	93.9 ± 0.01
Unsup-GraphSAGE	**X**, **A**	82.5 ± 0.7	81.6 ± 1.0	**96.0 ± 1.0**
DGI	**X**, **A**	84.8 ± 0.2	82.9 ± 0.5	95.8 ± 0.1
GMI	**X**, **A**	**86.0 ± 0.5**	83.5 ± 0.0	**96.0 ± 0.3**
GCN	**X**, **A**, **Y**	89.0± 1.0	84.2 ± 0.9	96.3 ± 0.1
GAT	**X**, **A**, **Y**	92.8 ± 0.7	**OOM**	**OOM**
FastGCN	**X**, **A**, **Y**	88.1± 1.0	85.0 ± 0.4	94.5 ± 0.2
GraphSAGE	**X**, **A**, **Y**	89.3 ± 0.5	85.6 ± 0.7	95.1 ± 0.3
SUBG- CON	**X**, **A**	85.8 ± 1.0	**84.8 ± 1.0**	95.4 ± 0.1
SUBG- CON$^+$	**X**, **A**	**86.4 ± 1.0**	**85.3 ± 0.8**	**96.0 ± 0.3**

Table 5 Comparison with different graph neural network encoders

Dataset	GCN	GCN+Skip	GAT	GIN
Cora	82.1	**83.5**	**83.5**	83.0
Citeseer	72.4	**73.2**	73.0	73.0
Pubmed	79.2	**81.1**	80.0	80.4
PPI	66.2	**66.9**	66.8	66.0
Flickr	**48.8**	48.2	48.7	48.3
Reddit	**95.2**	94.5	94.9	93.9

networks. More expressive GNNs, such as GAT and GIN, are less suitable to handle these subgraphs.

3.5.2 Effectiveness of objective function

We compare different objective functions and list the experiment results in Table 6. To make the comparisons fair, we tune the hyperparameters for all loss functions and report their best results. Table 6 shows that margin loss can achieve the best performance compared with other losses. We believe, as context subgraphs extracted from the same original graph can be

Table 6 Comparison with models trained with different objective functions

	Cora	Citeseer	Pubmed	PPI	Flickr	Reddit
Margin	**83.5**	**73.2**	**81.0**	**66.9**	**48.8**	**95.2**
Logistic	82.4	72.2	79.8	66.8	48.5	95.0
BPR	81.7	72.0	79.9	66.8	48.6	94.8

Table 7 Model performance via different corruption function

Function	Cora	Citeseer	Pubmed
Random	67.0 ± 0.8	52.7 ± 0.7	69.6 ± 1.0
Inverse	73.0 ± 0.5	63.8 ± 0.6	75.5 ± 0.5
Modify	$\mathbf{83.8 \pm 0.5}$	72.7 ± 0.4	$\mathbf{81.5 \pm 0.5}$
Shuffle	83.5 ± 0.5	$\mathbf{73.5 \pm 0.2}$	81.0 ± 0.1
Function	PPI	Flickr	Reddit
Random	43.9 ± 0.9	37.8 ± 0.4	73.6 ± 0.8
Inverse	53.6 ± 0.6	42.2 ± 0.4	83.8 ± 0.3
Modify	65.7 ± 0.6	$\mathbf{49.0 \pm 0.3}$	94.6 ± 0.1
Shuffle	$\mathbf{67.7 \pm 0.2}$	48.8 ± 0.1	$\mathbf{95.3 \pm 0.1}$

somewhat similar, it is not suitable to apply the loss functions that distinguish positive and negative examples absolutely.

3.5.3 Choose of corruption function

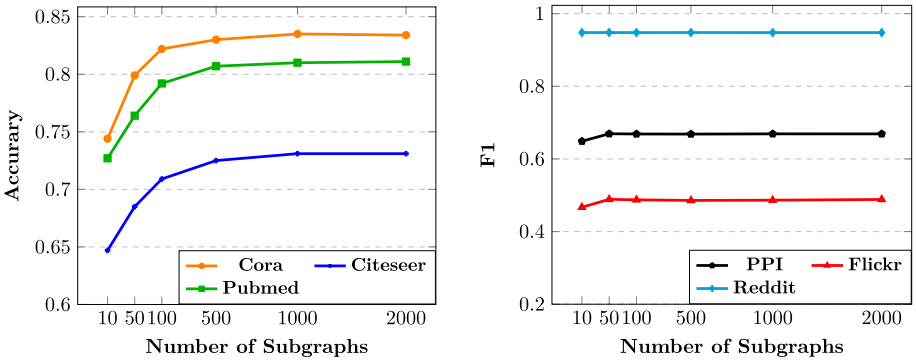
In Table 7, we can see the experimental results of different corruption functions. Since the corruption function determines the generation of positive and negative samples, it has a great impact on the experimental results. On some datasets, different corruption functions will cause a huge fluctuation of the evaluation scores for node classification. On the six datasets, the results of the two corruption functions, Modify and Shuffle, are relatively good. We believe that the best strategy is to make slight modifications and distinguish the existing subgraphs from others.

3.5.4 Choose of sampling strategies

We study the most suitable subgraph sampling strategies for different datasets. The experimental results are shown in Table 8. Compared with random sampling strategy, the methods which select important nodes according to the graph structure can achieve better results, because these subgraphs can provide better regional information. In addition, for large-scale graph datasets, the PPR-based method can outperform others. We assume the reason is the difficulty of extracting important nodes only by the local structure in the large graphs. Therefore, using global structures can contribute to node representation learning. In addition, although in most cases, the egonet strategy can also achieve good results, such a sampling method will increase the storage space requirements for dense graphs.

Table 8 Model performance via different subgraph sampling strategies

Strategy	Cora	Citeseer	Pubmed
Random	78.5 ± 1.5	65.8 ± 1.5	79.3 ± 1.3
Egonet	80.3 ± 0.5	69.5 ± 0.5	**81.0 ± 0.3**
Local-based	**83.5 ± 0.5**	**73.2 ± 0.2**	80.9 ± 0.3
Global-based	83.3 ± 0.3	72.5 ± 0.7	**81.0 ± 0.6**
Strategy	PPI	Flickr	Reddit
Random	63.4 ± 1.4	44.9 ± 1.0	89.5 ± 1.1
Egonet	66.4 ± 0.3	47.5 ± 0.3	92.0 ± 0.2
Local-based	66.0 ± 0.4	47.3 ± 0.4	95.0 ± 0.1
Global-based	**66.9 ± 0.2**	**48.8 ± 0.1**	**95.2 ± 0.0**



(a) Small-Scale Datasets **(b)** Large-Scale Datasets

Fig. 3 Effectiveness of training the encoder with different numbers of sampled subgraphs

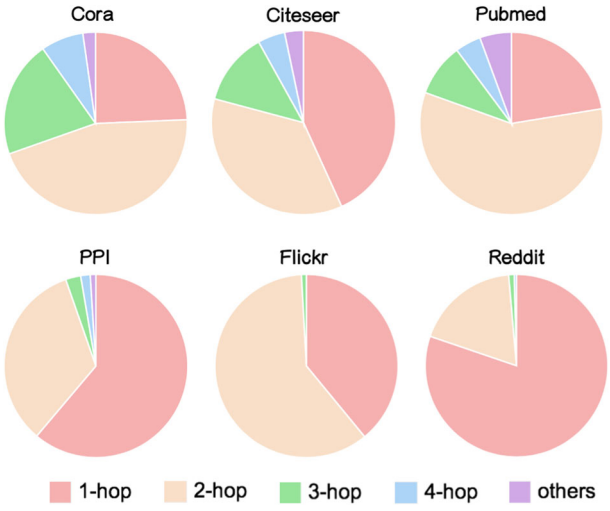


Fig. 4 Composition of context subgraphs for different datasets. The pie chart indicates the proportion of neighbors of different distances from central nodes in the context subgraphs

Table 9 Efficiency of SUBG- CON on three small-scale datasets

Dataset	Algorithm	Training Time	Memory
Cora	DGI	27s	3597MB
	GMI	104s	3927MB
	SUBG- CON	**14s**	**1586MB**
Citeseer	DGI	48s	4867MB
	GMI	410s	7605MB
	SUBG- CON	**12s**	**1163MB**
Pubmed	DGI	104s	10911MB
	GMI	1012s	12115MB
	SUBG- CON	**26s**	**975MB**

We train the encoder with 500 context subgraphs

Table 10 Efficiency of SUBG- CON on three large-scale datasets

Dataset	Algorithm	Training Time	Memory
PPI	DGI	44s	10171MB
	GMI	561s	12101MB
	SUBG- CON	**3s**	**1349MB**
Flickr	DGI	518s	5028MB
	GMI	1247s	9768MB
	SUBG- CON	**12s**	**1903MB**
Reddit	DGI	4071s	8517MB
	GMI	9847s	12098MB
	SUBG- CON	**25s**	**3805MB**

We train the encoder with 50 context subgraphs

3.6 Efficiency

3.6.1 Train with a few subgraphs

As context subgraphs have simple and similar structures, we assume that maybe extracting all subgraphs is unnecessary for training the encoder well. Therefore, we conducted some experiments about training the encoder with a few subgraphs sampled from the graph. The effectiveness of the number of sampled subgraphs on the six datasets is shown in Fig. 3. We observed that, for Cora, Citeseer, and Pubmed, about 500 subgraphs can provide sufficient information for the encoder while the other three datasets, PPI, Flickr, and Reddit, only require as few as 50 subgraphs. We believe the sparsity of the graph leads to the difference. The degree of nodes in Cora, Citeseer, and Pubmed is small; therefore, subgraphs extracted from these datasets can be of much different shape. On the contrary, PPI, Flickr, and Reddit are relatively denser and the context subgraphs likely composed of direct neighbors. Thus, the encoder can capture the structure easily. To verify our surmise, we observe the composition of subgraphs in different datasets as shown in Fig. 4. The observation guides us to train the encoder with a few subgraphs to accelerate the convergence of the loss function, which takes much less training time and computation memory.

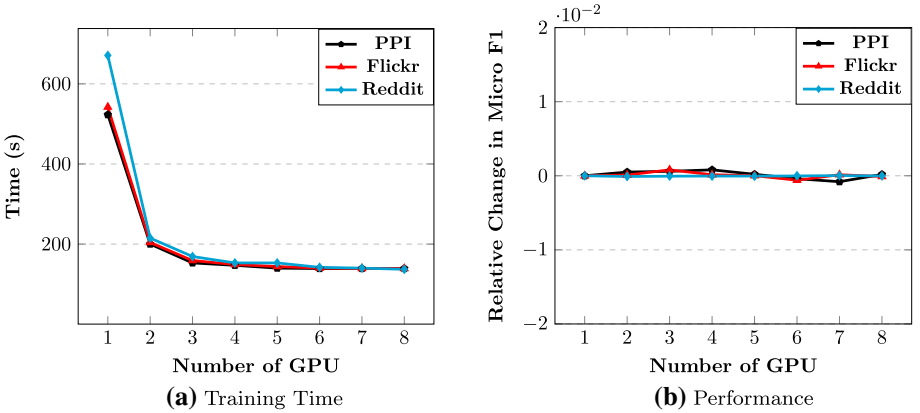


Fig. 5 Effects of parallelizing

3.6.2 Training time and memory cost

In Tables 9 and 10, we summarize the performance on the state-of-the-arts self-supervised methods over their training time and memory usage relative to that of our method on all the six datasets. The training time refers to the time for training the encoder (exclude validation). The memory refers to total memory costs of model parameters and all hidden representations of a batch. The two self-supervised baselines apply GCN as their encoders on Cora, Citeseer, and Pubmed, which cannot be trained on large-scale graphs due to excessive memory requirements. For other larger graphs, they choose GraphSAGE, a fast sampling-based graph neural network, for node representation learning. We use an early stopping strategy on the observed results on the validation set, with a patience of 20 epochs (specially, 150 epochs for Pubmed). According to the findings in the previous subsection, 500 subgraphs randomly sampled are used to train the encoder for three small-scale datasets in Table 9 while 50 subgraphs are used for three larger datasets in Table 10. We can clearly found ours methods can be trained much faster with much less computation memory than these baselines on all the datasets. In particular, our advantage of efficiency can be more prominent on large-scale graphs, especially on Reddit. We believe that compared to the whole graph structure, subgraphs of much small size can speedup encoder training. Besides, training with a few subgraphs can further reduce training time and memory usage.

3.6.3 Parallel computation

For complex realistic application scenarios, in the case when training with a small number of subgraphs doesn't work, SUBG- CON can be run efficiently in parallel. We set the number of subgraphs as 20,000 and set training epoch as 400, and run experiments using multiple GPUs on three large-scale datasets, PPI, Flickr, and Reddit. Figure 5 presents the effects of parallelizing. It shows the speed of processing these three data sets can be accelerated by increasing the number of GPUs (Fig. 5a). It also shows that there is no loss of predictive performance relative to the running our model serially (Fig. 5b). It has been demonstrated that this technique is highly scalable.

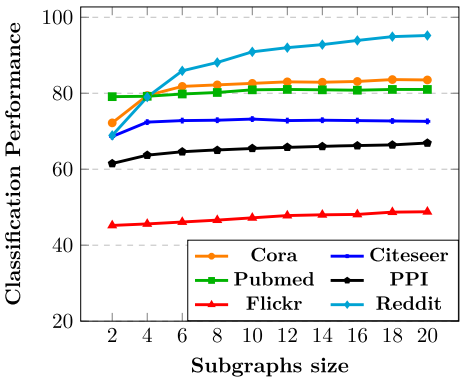


Fig. 6 Subgraph size analysis

3.7 Subgraph size analysis

Now we examine the influence of the size of context subgraphs in our framework on the six datasets. We adjust the subgraph size from 2 to 20 (including the central node) and evaluated the results as shown in Fig. 6. We observe that our model can achieve better performance with context subgraphs of a larger size in general. We believe that is because more regional structure information makes a contribution to high-quality latent representations. Due to the limited computation memory, we set the subgraph size as 20. However, there is an exception. As the size of subgraphs increases, the performance on Citeseer becomes better first, reaches the peak when the size is 10, and then goes down. We consider, due to the sparsity of Citeseer, the subgraphs composed of 10 nodes have sufficient context information. Larger subgraphs with complex structures will bring about more noise and deteriorate the process of representations learning. Thus, we set the subgraph size as 10 for Citeseer. It is noted that using very small subgraphs causes different impacts on different datasets. Specifically, when we train the encoder with subgraphs containing only two nodes (a central node and a closest related neighbor), the performance degrades on all the datasets. Especially, the decrease in F1 score on Reddit is up to 20 points. It indicates that Reddit is large in scale and complex in structure; therefore, a few neighbors are insufficient to be a proxy of relatively informative context. We should take it into consideration for model design.

4 Related work

4.1 Graph neural networks

Graph neural networks use the graph structure as well as node features to learn node representation vectors. Existing graph neural networks follow a neighborhood aggregation strategy, which we iteratively update the representation of a node by aggregating representations of its neighboring nodes and combining with its representations [34]. Existing graph neural networks have led to advances in multiple successful applications across different domains [1,17,30,34]. However, they usually take a complete graph as input. Thus, they can hardly be applied to large-scale graph data. What's more, the inter-connected graph structure also prevents parallel graph representation learning, which is especially critical for large-sized graph data. To handle these issues, sampling-based methods are proposed to train GNNs based on mini-batch of nodes, which only aggregate the representations of a subset of ran-

domly sampled nodes in the mini-batch [3,9]. Although this kind of approaches reduces the computation cost in each aggregation operation, the total cost can still be large. Besides, these graph neural networks mainly focus on supervised learning and require the supervision of label information. It is intractable for them to handle unlabeled graphs, which are widely available in practically.

4.2 Unsupervised node representation learning

There is abundant literature in traditional unsupervised representation learning of nodes within graphs. Existing methods optimize models with random walk-based objectives [8,24,29] or reconstructing graph structures [9,18]. The underlying intuition is to train an encoder network so that nodes that are close in the input graph are also close in the representation space. Although these methods claim to capture node proximity, they still suffer from some limitations. Most prominently, they over-emphasizing proximity similarity, making it difficult to capture the inherent graph structural information. Besides, as these encoders already enforce an inductive bias that neighboring nodes have similar representations, it is unclear whether such objectives actually provide any useful signal for training an encoder. Thus, existing methods fail to solve real-world tasks as strongly as supervised methods do.

4.3 Self-supervised learning

Self-supervised learning has recently emerged as a promising approach to overcome the dilemma of lacking available supervision. Its key idea is defining an annotation free pretext task and generating surrogate training samples automatically to train an encoder for representation learning. A wide variety of pretext tasks have been proposed for visual representation learning [2,4,7]. However, there are a few works of literature about self-supervised methods for graph representation learning so far. Deep graph infomax [31] aims to train a node encoder that maximizes mutual information between node representations and the pooled global graph representation. Graphical mutual information [23] proposes to maximize the mutual information between the hidden representation of each node and the original features of its 1-hop neighbors. These works tend to be biased in fitting either the overall or very local (1-hop neighbor) graph structures in defining the mutual information based loss terms, which would harm the quality of learned representations. Besides, these self-supervised works also need to take the complete graph as the input, which restricts their scalability on large-sized graphs.

5 Conclusion

In this paper, we propose a novel scalable self-supervised graph representation via subgraph contrast, SUBG- CON. It utilizes the strong correlation between central nodes and their regional subgraphs for model optimization. Based on sampled subgraph instances, SUBG-CON has prominent performance advantages in weaker supervision requirements, model learning scalability, and parallelization.

Through an empirical assessment on multiple benchmark datasets, we demonstrate that the effectiveness and efficiency of SUBG- CON compared with both supervised and unsupervised strong baselines. In particular, it shows that the encoder can be trained well on the current popular graph datasets with a little regional information. It indicates that existing methods

may still lack the ability to capture higher-order information, or our existing graph dataset only requires low-order information to get good performance. We hope that our work can inspire more research on graph structure to explore the above problems.

Acknowledgements We appreciate the comments from anonymous reviewers which will help further improve our work. This work is funded in part by the National Natural Science Foundation of China Projects No. U1636207 and No. U1936213. This work is also partially supported by NSF through grant IIS-1763365, IIS-2106972 and by UC Davis.

References

1. Abu-El-Haija S, Perozzi B, Kapoor A, Alipourfard N, Lerman K, Harutyunyan H, Steeg GV, Galstyan A (2019) Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing. arXiv preprint arXiv:1905.00067
2. Asano YM, Rupprecht C, Vedaldi A (2019) A critical analysis of self-supervision, or what we can learn from a single image. arXiv preprint arXiv:1904.13132
3. Chen J, Ma T, Xiao C (2018) Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247
4. Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709
5. Dupont P, Callut J, Dooms G, Monette JN, Deville Y, Sainte B (2006) Relevant subgraph extraction from random walks in a graph. Universite Catholique de Louvain, UCL/INGI, Number RR, p 7
6. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch geometric. In: ICLR workshop on representation learning on graphs and manifolds
7. Gidaris S, Singh P, Komodakis N (2018) Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728
8. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 855–864
9. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp 1024–1034
10. Hamilton WL, Ying R, Leskovec J (2017) Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584
11. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034
12. Jeh G, Widom J (2003) Scaling personalized web search. In: Proceedings of the 12th international conference on World Wide Web, pp 271–279
13. Jiao Y, Xiong Y, Zhang J, Zhu Y (2019) Collective link prediction oriented network embedding with hierarchical graph attention. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 419–428
14. Jing L, Tian Y (2020) Self-supervised visual feature learning with deep neural networks: a survey. IEEE Trans Pattern Anal Mach Intell
15. Ketkar N (2017) Introduction to pytorch. In: Deep learning with python. Springer, pp 195–208
16. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
17. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
18. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv preprint arXiv:1611.07308
19. Lee J, Lee I, Kang J (2019) Self-attention graph pooling. arXiv preprint arXiv:1904.08082
20. Liao R, Zhao Z, Urtasun R, Zemel RS (2019) Lanczosnet: multi-scale deep graph convolutional networks. arXiv preprint arXiv:1901.01484
21. Meng L, Yang Bai J, Zhang J (2019) Latte: application oriented social network embedding. In: 2019 IEEE international conference on big data (Big Data), pp 1169–1174
22. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
23. Peng Z, Huang W, Luo M, Zheng Q, Rong Y, Xu T, Huang J (2020) Graph representation learning via graphical mutual information maximization. In: Proceedings of the web conference, pp 259–270

24. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 701–710
25. Qu M, Bengio Y, Tang J (2019) Gmnn: graph markov neural networks. In: International conference on machine learning, pp 5241–5250
26. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2012) Bpr: bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618
27. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 815–823
28. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):60
29. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077
30. Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2017) Graph attention networks. In: CoRR. arXiv:1710.10903
31. Veličković P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD (2018) Deep graph infomax. arXiv preprint arXiv:1809.10341
32. Wang S, He L, Cao B, Lu CT, Yu PS, Ragin AB (2017) Structural deep brain network mining. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 475–484
33. Wu F, Souza Jr AH, Zhang T, Fifty C, Yu T, Weinberger KQ (2019) Simplifying graph convolutional networks. In: ICML
34. Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks? arXiv preprint arXiv:1810.00826
35. Yizhu J, Yun X, Jiawei Z, Yao Z, Tianqi Z, Yangyong Z (2020) Sub-graph contrast for scalable self-supervised graph representation learning. In: 2020 IEEE international conference on data mining (ICDM). IEEE
36. Zeng H, Zhou H, Srivastava A, Kannan R, Prasanna V (2019) Graphsaint: graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931
37. Zhang J, Meng L (2019) Gresnet: graph residual network for reviving deep gnns from suspended animation. arXiv abs/1909.05729
38. Zhang J, Zhang H, Sun L, Xia C (2020) Graph-bert: only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140

Yizhu Jiao received an M.E. degree in software engineering from Fudan University, Shanghai, China, in 2021. Her research interests include graph mining and natural language processing.

Yun Xiong is a professor of computer science at Fudan University, Shanghai, China. She received the Ph.D. degree in computer and software theory from Fudan University. Her research interests include database and data mining.

Jiawei Zhang is an assistant professor in the Department of Computer Science at University of California, Davis. He obtained the Ph.D. degree from University of Illinois at Chicago in 2017. His research interest includes data mining and machine learning.

Yao Zhang is a post-doctoral of computer science at Fudan University, Shanghai, China. He received the Ph.D. degree from Fudan University. His research interests include data mining and machine learning.

Tianqi Zhang received an M.E. degree in computer science from Fudan University, Shanghai, China, in 2021. Her research interests include graph mining and community detection.

Yangyong Zhu is a professor of computer science at Fudan University, Shanghai, China. He received the Ph.D. degree in computer and software theory from Fudan University, China. His research interests include database and data mining.