Sub-graph Contrast for Scalable Self-Supervised Graph Representation Learning

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

[1]*Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, China*
[2]*Shanghai Institute for Advanced Communication and Data Science, Fudan University, China*
[3]*IFM Lab, Department of Computer Science, Florida State University, FL, USA*
Email: [1]{yzjiao18, yunx, yzhang18, tqzhang18, yyzhu}@fudan.edu.cn, [3]jiawei@ifmlab.org

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 a novel data augmentation strategy, SUBG-CON learns node representations through a contrastive loss defined based on subgraphs sampled from the original graph instead. 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 compared with both classic and state-of-the-art graph representation learning approaches on multiple real-world large-scale benchmark datasets from different domains.

Index Terms—**Self-Supervised Learning; Graph Representation Learning; Subgraph Contrast; Graph Neural Networks**

I. INTRODUCTION

Graph representation learning [1] 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 [2], link prediction [3], graph classification [4], and graph alignment [5]. Graph representation learning problems have been studied on graph data from many different domains such as social networks [6], chemical molecular graphs [7], and bio-medical brain graphs [8].

✉ Corresponding author.

Most existing successful methods are based on graph neural networks (GNNs) [2], [9]–[11] , 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 [1] or reconstructing graph structures [12]. However, detached from supervision information, representations learned by such unsupervised approaches can hardly work well in downstream applications with specific task objectives [13].

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 and generating surrogate training samples automatically to train an encoder for representation learning. In the field of computer vision, data augmentation [15], 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 [16] and Graphical Mutual Information [17] (even though these approaches pose themselves as unsupervised models initially). Deep Graph Infomax (DGI) [16] 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) [17] is centered about local struc-
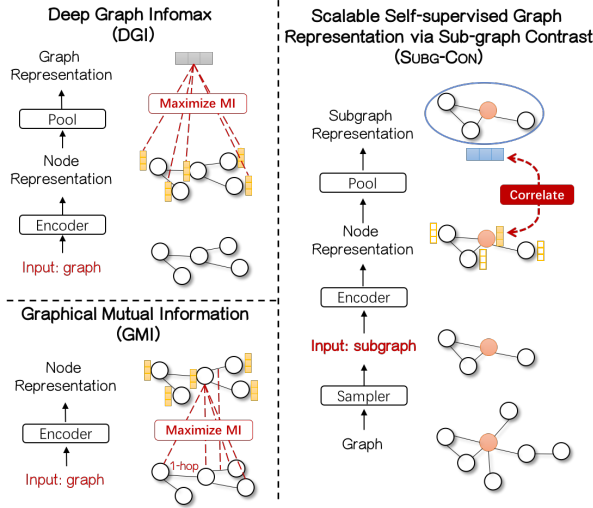
Fig. 1. An illustration of DGI (upper left), GMI (bottom left), and our proposed SUBG-CON (right). Little colored squares denote learnt representations. Red nodes denote central nodes of context subgraphs. SUBG-CON utilizes the strong correlation between central nodes and their context subgraphs sampled 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.

tures 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. It takes the 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 based on subgraph sampling firstly. 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. Compared with the previous methods operating on the complete graph structure, 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 is 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 SUBG-CON model are consistently competitive on node classification tasks, often outperforming both supervised and unsupervised strong baselines. Besides, we verify the efficiency of SUBG-CON, 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, SUBG-CON method requires 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.

II. RELATED WORK

A. 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 [18]. Existing graph neural networks have led to advances in multiple successful applications across different domains [2], [9], [18], [19]. 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 randomly sampled nodes in the mini-batch [6], [20]. 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.

B. 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 [3], [21], [22] or reconstructing graph structures [12], [20]. 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.

C. 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 [23]–[25]. However, there are a few works of literature about self-supervised methods for graph representation learning so far. Deep graph infomax [16] aims to train a node encoder that maximizes mutual information between node representations and the pooled global graph representation. Graphical mutual information [17] 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.

III. 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, and our self-supervised pretext task for model optimization. Finally, we introduce parallel SUBG-CON briefly.

A. 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 $i \rightarrow j$ in the graph and $\mathbf{A}(i, j) = 0$ otherwise.

Traditional graph representation methods target on training an encoder $\mathcal{E} : \mathbb{R}^{N \times F} \times \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times F'}$ 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 leant 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, $\mathcal{E} : \mathbb{R}^{N' \times F} \times \mathbb{R}^{N' \times N'} \rightarrow \mathbb{R}^{N' \times F'}$, 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.

B. 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 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 follow the subgraph sampling based on personalized pagerank algorithm [26] as introduced in [27]. 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 colum-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.

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 minibatches and fed to train SUBG-CON.

C. Encoding Subgraph For Representations

Given the context subgraph $\mathcal{G}_i = (\mathbf{X}_i, \mathbf{A}_i)$ of a central node i, the encoder $\mathcal{E} : \mathbb{R}^{N' \times F} \times \mathbb{R}^{N' \times N'} \to \mathbb{R}^{N' \times F'}$ 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, $\mathcal{R} : \mathbb{R}^{N' \times F'} \to \mathbb{R}^{F'}$, 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.
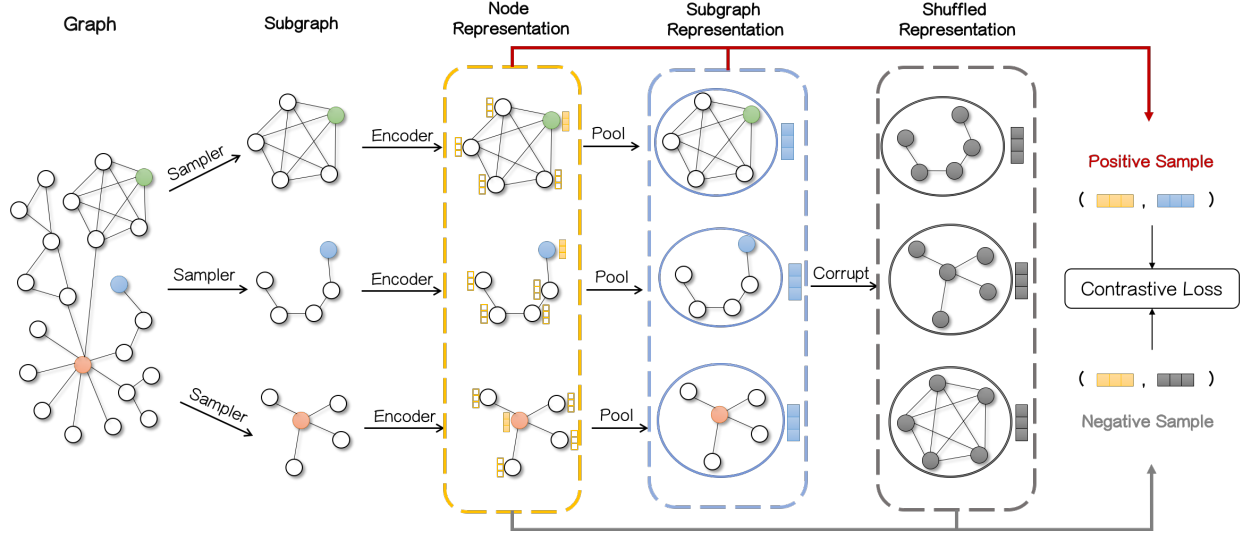
Fig. 2. Architecture of SUBG-CON. A series of context subgraph are sampled from the original graph and fed into the encoder to obtain the representations of central nodes and subgraphs after pooling. For a specified node, its context subgraph representation is regarded as positive sample while other subgraph representations randomly sampled are regarded as negative samples. 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.

D. 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 by 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 be-

---

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 $\mathcal{P}$.

1: Precompute importance score matrix $\mathbf{S}$ according to Eq. 1.
2: **while** not converge **do**
3:     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 size of the subgraph set.
4:     **for all** each subgraph $(\mathbf{X}_i, \mathbf{A}_i)$ **do**
5:         Encode the subgraph to obtain latent representation matrixes $\mathbf{H}_i = \mathcal{E}(\mathbf{X}_i, \mathbf{A}_i)$.
6:         Obtain the central node representation $\mathbf{h}_i = \mathcal{C}(\mathbf{H}_i)$.
7:         Summarize the subgraph representation through the readout function $\mathbf{s}_i = \mathcal{R}(\mathbf{H}_i)$.
8:     **end for**
9:     Corrupt the subgraph representations to generate negative examples for the corresponding node representations $\{\widetilde{\mathbf{s}}_1, \widetilde{\mathbf{s}}_2, ..., \widetilde{\mathbf{s}}_M\} = \mathcal{P}(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_M)$.
10:     Update parameters of $\mathcal{E}$ and $\mathcal{R}$ by applying gradient descent to maximize Eq. 2.
11: **end while**

---

tween positive examples and negative examples [16]. 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 [28] for 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

Table I. Dataset statistics

	Dataset	Type	Nodes	Edges	Degree	Features	Classes	Train / Val / Test
Small-scale	Cora	Citation network	2,708	5,429	4.0	1,433	7	0.05 / 0.18 / 0.37
	Citeseer	Citation network	3,327	4,732	2.8	3,703	6	0.04 / 0.15 / 0.30
	Pubmed	Citation network	19,717	44,338	4.5	500	3	0.003 / 0.03 / 0.05
Large-scale	PPI	Protein network	56,944	818,716	28.8	50	121	0.79 / 0.11 / 0.10
	Flickr	Social network	89,250	899,756	20.2	500	7	0.50 / 0.25 / 0.25
	Reddit	Social network	232,965	11,606,919	99.6	602	41	0.66 / 0.10 / 0.24

denoted as

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

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function and ϵ is margin value. We summarize the steps of the procedure of our approach in Algorithm 1.

E. Parallelizability

Compared with existing methods that input the complete graph data, it is parallelizable to operate on context subgraph. 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.

IV. EXPERIMENT

In this section, we conduct extensive experiments to verify both the effectiveness and the efficiency of SUBG-CON on a variety of node classification tasks on multiple real-world datasets from different domains. 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.

A. 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 wide used in related works [2] (Cora, Citeseer and Pubmed) and three large-scale datasets to verify the scalability of our approach (PPI, Flickr, and Reddit) [2], [29]. It includes three citation networks, two social networks, and a protein network. All datasets follow "fixed-partition"

splits. Further information on the datasets can be found in Table I.

We set up the experiments on the following benchmark 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. **Source code** The source code of SUBG-CON is available in https://github.com/yzjiao/Subg-Con.

B. 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 [30] 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 perametric ReLU (PReLU) function [31]. $\mathbf{W}$ 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 towards other kinds of tasks, the

Table II. Performance comparison with different methods on node classification. The second column illustrates the data used by each algorithm in the training phase, where $\mathbf{X}$, $\mathbf{A}$, and $\mathbf{Y}$ denotes features, adjacency matrix, and labels, respectively. **OOM**: Out of memory.

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

design of appropriate corruption strategies remains an area of open research.

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

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

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 III. 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	$\log \sigma(\mathbf{hs} - \mathbf{h\widetilde{s}})$

Objective functions. We compare the margin loss [28] against other commonly used contrastive loss functions, such as logistic loss [32], and bayesian personalized ranking (BPR) loss [33]. Table III shows these three objective function. Their impacts will be discussed later.

Implementation details. All experiments are implemented in PyTorch [34] with Glorot initialization [35] and 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 on Cora, Citeseer, PubMed following [2], and apply the processing strategy in [20] on Reddit, PPI, and Flickr. Especially, for PPI, suggested by [16], we standardize the learned embeddings before feeding them into the logistic regression classifier. During training, we use Adam optimizer [36] 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 [16] and GMI [17], which both learn graph embeddings by leveraging mutual information maximization. Two traditional unsupervised methods, DeepWalk [21] and unsupervised variants of GraphSAGE (abbreviated as Unsup-GraphSAGE) [20] 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 [2], GAT [9], FastGCN [6] and supervised GraphSAGE [20]. 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.

Evaluation metrics. For the classification task, we provide the learned embeddings across the training set to the logistic regression classifier and give the results on the validation and test nodes [17]. Followed [16], 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).

C. Node Classification

The results of our comparative evaluation experiments are summarized in Table II. 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. We further 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 super-vision 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).

D. Design of Architectures

Table IV. 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

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) [9], graph isomorphism network (GIN) [18]. The experimental results are listed in Table IV.

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

Table V. 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

2) Effectiveness of Objective Function: We compare different objective functions and list the experiment results in Table V. To make the comparisons fair, we tune the hyperparameters for all loss functions and report their best results. Table V 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 somewhat similar, it is not suitable to apply the loss functions that distinguish positive and negative examples absolutely.
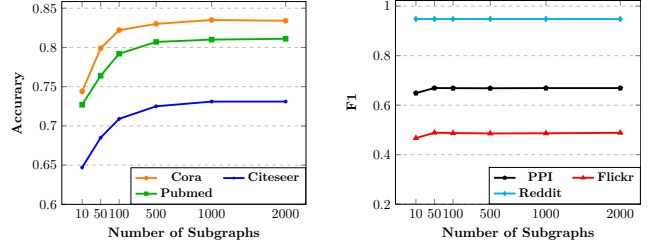
(a) Small-Scale Datasets   (b) Large-Scale Datasets

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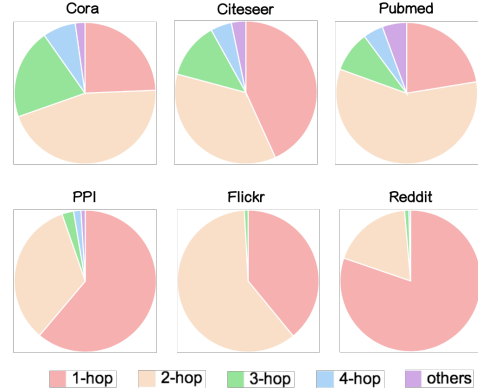

Fig. 4. Composition of context subgraphs for different datasets. The pie chart indicates the proportion of neighbors of different distances from central nodes in the context subgraphs.

E. Efficiency

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 are showed in the 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 showed 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.

2) Training time and memory cost: In Table VI and Table VII, 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

Table VI. Efficiency of SUBG-CON on three small-scale datasets. We train the encoder with 500 context subgraphs.

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**

Table VII. Efficiency of SUBG-CON on three large-scale datasets. We train the encoder with 50 context subgraphs.

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**

(a) Training Time (b) Performance

Fig. 5. Effects of parallelizing.

Fig. 6. Subgraph size analysis.

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 VI while 50 subgraphs are used for three larger datasets in Table VII. 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) 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 20000 and set training epoch as 400, and run experiments using multiple GPUs on three large-scale datasets, PPI, Flickr and Reddit. Fig. 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. 5(a)). It also shows that
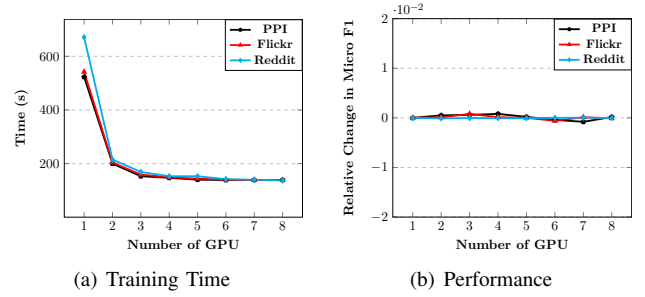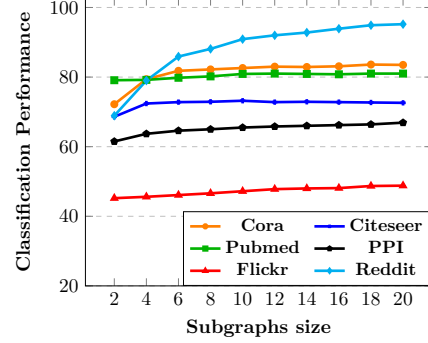
there is no loss of predictive performance relative to the running our model serially (Fig. 5(b)). It has been demonstrated that this technique is highly scalable.

F. 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 showed 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 of 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.

V. Conclusion

In this paper, we propose a novel scalable self-supervised graph representation via sub-graph 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.

VI. Acknowledgement

References

[1] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[4] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," *arXiv preprint arXiv:1904.08082*, 2019.

[5] Y. Jiao, Y. Xiong, J. Zhang, and Y. Zhu, "Collective link prediction oriented network embedding with hierarchical graph attention," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 419–428.

[6] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.

[7] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," *arXiv preprint arXiv:1901.01484*, 2019.

[8] S. Wang, L. He, B. Cao, C.-T. Lu, P. S. Yu, and A. B. Ragin, "Structural deep brain network mining," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 475–484.

[9] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *CoRR*, vol. abs/1710.10903, 2017. [Online]. Available: http://arxiv.org/abs/1710.10903

[10] F. Wu, A. H. Souza Jr, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019.

[11] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," in *International Conference on Machine Learning*, 2019, pp. 5241–5250.

[12] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[13] L. Meng, J. yang Bai, and J. Zhang, "Latte: Application oriented social network embedding," *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1169–1174, 2019.

[14] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[15] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[16] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *arXiv preprint arXiv:1809.10341*, 2018.

[17] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference 2020*, 2020, pp. 259–270.

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[19] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," *arXiv preprint arXiv:1905.00067*, 2019.

[20] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[21] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[23] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," *arXiv preprint arXiv:1803.07728*, 2018.

[24] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "A critical analysis of self-supervision, or what we can learn from a single image," *arXiv preprint arXiv:1904.13132*, 2019.

[25] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[26] G. Jeh and J. Widom, "Scaling personalized web search," in *Proceedings of the 12th international conference on World Wide Web*, 2003, pp. 271–279.

[27] J. Zhang, H. Zhang, L. Sun, and C. Xia, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020.

[28] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[29] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.

[30] J. Zhang and L. Meng, "Gresnet: Graph residual network for reviving deep gnns from suspended animation," *ArXiv, abs/1909.05729*, 2019.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[33] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[34] N. Ketkar, "Introduction to pytorch," in *Deep learning with python*. Springer, 2017, pp. 195–208.

[35] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.