Generative Subgraph Contrast for Self-Supervised Graph Representation Learning

Yuehui Han, Le Hui, Haobo Jiang, Jianjun Qian*, and Jin Xie*

Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education
Jiangsu Key Lab of Image and Video Understanding for Social Security
PCA Lab, School of Computer Science and Engineering
Nanjing University of Science and Technology, China
{hanyh, le.hui, jiang.hao.bo, csjqian, csjxie}@njust.edu.cn

Abstract. Contrastive learning has shown great promise in the field of graph representation learning. By manually constructing positive/negative samples, most graph contrastive learning methods rely on the vector inner product based similarity metric to distinguish the samples for graph representation. However, the handcrafted sample construction (e.g., the perturbation on the nodes or edges of the graph) may not effectively capture the intrinsic local structures of the graph. Also, the vector inner product based similarity metric cannot fully exploit the local structures of the graph to characterize the graph difference well. To this end, in this paper, we propose a novel adaptive subgraph generation based contrastive learning framework for efficient and robust self-supervised graph representation learning, and the optimal transport distance is utilized as the similarity metric between the subgraphs. It aims to generate contrastive samples by capturing the intrinsic structures of the graph and distinguish the samples based on the features and structures of subgraphs simultaneously. Specifically, for each center node, by adaptively learning relation weights to the nodes of the corresponding neighborhood, we first develop a network to generate the interpolated subgraph. We then construct the positive and negative pairs of subgraphs from the same and different nodes, respectively. Finally, we employ two types of optimal transport distances (i.e., Wasserstein distance and Gromov-Wasserstein distance) to construct the structured contrastive loss. Extensive node classification experiments on benchmark datasets verify the effectiveness of our graph contrastive learning method. Source code is available at https://github.com/yh-han/GSC.git.

Keywords: Graph Representation Learning, Contrastive Learning, Subgraph Generation, Optimal Transport Distance.

1 Introduction

Graph representation learning [11] has received intensive attention in recent years due to its superior performance in various downstream tasks, such as

[*] Corresponding authors

node/graph classification [17,19], link prediction [41] and graph alignment [7]. Most graph representation learning methods [10,17,31] are supervised, where manually annotated nodes are used as the supervision signal. Since the acquisition of supervision signals is time-consuming and labor-intensive, these methods are difficult to be applied to real scenarios.

Recent efforts have been devoted to unsupervised graph representation learning [15,18,22,34,35,43]. Among these methods, graph contrastive learning is a powerful manner for learning node or graph representation. By manually constructing positive/negative samples based on the perturbation (e.g., attribute masking, nodes shuffling or edge perturbation), it aims to enforce similar samples to be closer and dissimilar samples far from each other. However, dropping edges or masking node attributes randomly may change the original properties of the graph. For example, by adding or discarding graph nodes to construct positive samples, edge perturbation based methods [9,30,44,45] may change the local geometric structures of the original graph, resulting in generating dissimilar positive samples. Thus, with GNNs for feature extraction, the features of positive samples cannot be guaranteed to be as close as possible in the graph contrastive learning framework. Moreover, since the perturbation-based positive/negative sample augmentation methods are dataset-specific, it is difficult to adaptively select the suitable augmentation method for the specific dataset. Besides, the readout function is usually used to construct the vector-wise similarity metric between nodes/graphs, which ignores the structures of the graph [12,14,32]. Thus, the vector inner product based similarity metrics cannot characterize the graph difference well.

In this paper, instead of manually constructing contrastive samples, we propose a novel subgraph generation based contrastive learning framework for efficient self-supervised graph representation learning, where the optimal transport distance is employed to capture the difference between the subgraphs for robust similarity evaluation. Specifically, we first sample the neighbor subgraph of the center node based on the breadth first search (BFS). We then develop a subgraph generation network to adaptively generate subgraphs whose nodes are interpolated in the feature space with the learned weights. For each node, we can assign different attentional weights to the neighboring nodes to obtain the weighted node so that the formed subgraph can capture the intrinsic geometric structure of the graph. Consequently, we construct the positive pair with the sampled subgraph and the generated subgraph of the same center node and the negative pair with the sampled and generated subgraphs of different center nodes. Finally, based on the constructed positive/negative subgraphs, we formulate the structured contrastive loss to learn the node representation with the Wasserstein distance and Gromov-Wasserstein distance [2]. The structured contrastive loss can minimize the geometry difference between the positive subgraphs and maximize the difference between the negative subgraphs. Experimental results on five benchmark node classification datasets demonstrate that our proposed graph contrastive learning method can yield good classification performance.

To summarize, the main contributions include:

- We propose a novel adaptive sample generation based contrastive learning framework for self-supervised graph representation learning.
- We develop a subgraph generation module to adaptively generate contrastive subgraphs with neighborhood interpolation.
- We employ the optimal transport distance as the similarity metric for subgraphs, which can distinguish the contrastive samples by fully exploiting the local attributes (i.e., features and structures) of the graph.

2 Related Work

2.1 Graph Neural Networks

The purpose of graph neural networks (GNNs) is to use graph structures and node features to learn the node representations. Formally, classical GNNs follow a two-step processing: neighborhood node aggregation and feature transformation. It first updates the node representations by aggregating the representations of its neighboring nodes as well as its representations. Then, the representations of each node are mapped into a new feature space by the shared linear transformation. Graph Convolutional Network (GCN) [17] employs a weighted sum of the 1-hop neighboring node features to update the node features, where the weights of each node come from the node degree. Graph Attention Network (GAT) [31] calculates the weights by using the interaction between the neighboring nodes to replace the node degree. However, they usually need the complete graph as the input. Therefore, limited by the hardware resources, these methods are not suitable to be applied to large-scale graph data. To solve this issue, Hamilton et al [10] propose the sampling-based method, GraphSAGE. They first sample the neighborhood nodes for the mini-batch of the center nodes and update the node features by aggregating the sampled neighborhood nodes. Then, the batch nodes are iteratively updated until the entire graph is updated. These methods mainly focus on supervised learning and require a lot of manual labels. However, the acquisition of manually annotated labels is costly in labor and time.

2.2 Graph Contrastive Learning

Graph contrastive learning has recently been considered a promising approach for self-supervised graph representation learning. Its main objective is to train the encoder with an annotation-free pretext task. The trained encoder can transform the data into low-dimensional representations, which can be used for downstream tasks. The basic idea of graph contrastive learning aims at embedding positive samples close to each other while pushing away each embedding of the negative samples. In general, we can divide graph contrastive learning into two categories: pretext task based and data augmentation based methods.

Pretext Task. In graph contrastive learning, many early works design pretext tasks from the scale of the contrastive samples, i.e., node, subgraph or graph. Inspired by Deep InfoMax (DIM) [13], Deep Graph Infomax (DGI) [32] and Mutual Information Graph (INFOGRAPH) [29] learn the representations of nodes or graph by maximizing mutual information between the node and global graph. Also based on the contrast of nodes and graph, Multi-View Graph Representation Learning (MVGRL) [12] expands DGI to multiple views. By adding the cross-view contrast between the representation of nodes and graph, MVGRL further enhances the guidance performance of the pretext task. In order to avoid the problem of sharing positive samples (global graph) among multiple nodes in these methods, some works try to construct exclusive positive sample for each sample. Graphical Mutual Information (GMI) [23] proposes to maximize the mutual information between the neighborhood in input and the center node in output. Sub-graph Contrast (Subg-Con) [14] learns node features by taking the induced subgraphs of the center node as the input of the encoder and treating the center node and context subgraph as the contrastive sample pairs. This method can also alleviate the problem of memory overload caused by large-scale graphs. By treating top-k similar nodes from T-hop neighbors as positive samples, Augmentation-Free Graph Contrastive Learning (AF-GCL) [33] proposes the augmentation-free methods. Graph Contrastive Coding (GCC) [27] proposes to contrast between subgraphs. It takes the subgraphs from the same r-ego network as positive samples and subgraphs from the different r-ego networks as negative samples. However, GCC only considers the structure information neglecting the node features.

In addition to the scale of the contrastive samples, some works design pretext tasks to better exploit contrastive information. To solve the problem of false-negative samples, [42] proposes to jointly perform representation learning and clustering, where feature representation and clustering can be promoted from each other. With the same motivation, Curriculum Contrastive Learning (CuCo) [5] proposes a scoring function to sort the negative samples from easy to hard, and a pacing function to automatically select the negative samples in the training process. For better selection of positive samples, Augmentation-Free Graph Representation Learning (AFGRL) [20] proposes to discover the positive node that share the local structural information and the global semantics. Besides, Local-instance and Global-semantic Learning (GraphLoG) [36] proposes to capture the local similarities and the global semantic clusters to learn the whole-graph representation.

Data Augmentation. Data augmentation based graph contrastive learning methods usually design different perturbation manners (e.g., attribute masking, nodes shuffling or edge perturbation) to construct contrastive samples. Deep Graph Contrastive Representation Learning (GRACE) [44] augments the graph by setting the probability of edge removal and node features mask. Then, it takes the corresponding nodes of the augmented graph as positive samples and all the other nodes as negative samples. Graph Contrastive Learning (GraphCL) [9] proposes the sample augmentation manner from the subgraph level. For the induced subgraphs of the center nodes, it employs two stochastic perturbations and a shared encoder to produce two representations of the same node. You et al propose [39] for molecular property prediction in chemistry and protein function prediction in biology. They systematically study the effects of various combinations of graph augmentations on multiple datasets, and found that the choice of data augmentation is closely related to the specific datasets. However, these perturbation-based methods heavily rely on handcraft settings. To solve this problem, various efforts have been made. On the one hand, some works try to optimize the setting of perturbation probability. Based on the node centrality, Graph Contrastive Learning with Adaptive Augmentation (GCA) [45] can adaptively learn the probability of edge removal. Besides, by adding more noise to the unimportant node features, it can enforce the model to recognize underlying semantic information. Based on the min-max principle, Adversarial Graph Contrastive Learning (AD-GCL) [30] proposes a trainable edge-dropping graph augmentation manner. On the other hand, some works try to optimize the choice of perturbation methods. Joint Augmentation Optimization (JOAO) [38] proposes to adaptively select data augmentation manners of graph by adversarial training. With the same motivation, Automated Graph Contrastive Learning (AutoGCL) [37] proposes to adaptively select data augmentation manners of nodes by the learnable graph view generator.

3 Proposed Method

In this section, we present our subgraph generation based graph contrastive learning method. As shown in Fig. 1, based on the sampled subgraphs with the breadth first search, we first adaptively generate the contrastive subgraphs to construct positive/negative samples. Then, we employ the optimal transport distance (i.e., Wasserstein distance and Gromov-Wasserstein distance) to formulate the contrastive loss between the constructed samples.

3.1 Adaptive Subgraph Generation

Before introducing our method, we first provide the preliminary concepts about our graph representation learning. Let $G = (V, E)$ represent an undirected graph, where V and E denote the vertex set and the edge set, respectively. The feature matrix of the graph is denoted as $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i \in R^C$ is the feature of the node i, C represents the dimension of input features and N is the number of the nodes. The adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ indicates the topological structure of the graph where if node i and j are linked, $\boldsymbol{A}_{ij} = 1$, otherwise, $\boldsymbol{A}_{ij} = 0$. Let $\mathcal{G}_i = (V_i, E_i)$ denote the induced subgraph of center node i, where V_i and E_i represent the vertex set and the edge set of the induced subgraph i, respectively. We denote the adjacency matrix of subgraph i induced from graph as $\boldsymbol{A}_i \in R^{k\times k}$, where k is the number of nodes of subgraph i. The goal of self-supervised graph representation learning is to learn the nodes embeddings $\boldsymbol{H} = \varepsilon(\boldsymbol{A}, \boldsymbol{X})$ via an encoder $\varepsilon : R^{N\times C} \times R^{N\times N} \to R^{N\times F}$ without supervised information, where F is the dimension of embeddings.

The construction of contrastive samples is critical in graph contrastive learning. Most graph contrastive learning methods generate positive and negative

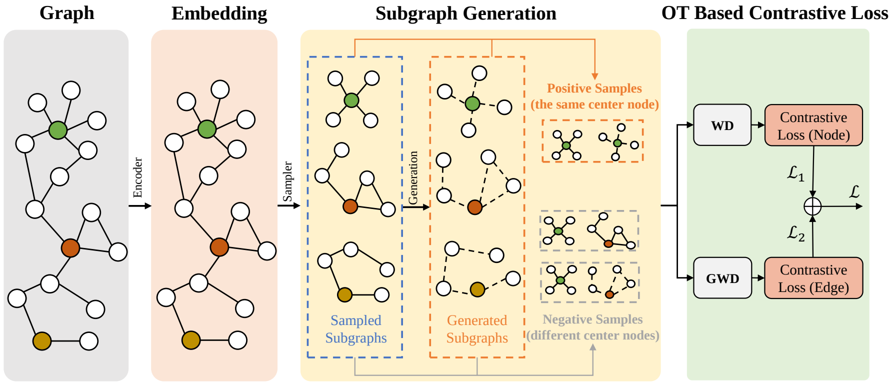


Fig. 1. The architecture of our method. We first employ an encoder to obtain the node embeddings. Based on the BFS sampling, we obtain the subgraphs of each node. Next, we use the proposed generation module to generate the contrastive samples of the sampled subgraphs. Then we take the sampled and generated subgraphs with the same center node as the positive samples while the subgraphs with the different center nodes as the negative samples. In order to fully exploit local structure information of the graph, we further introduce two types of optimal transport distances (i.e., Wasserstein distance and Gromov-Wasserstein distance) to calculate the similarity between the subgraphs. Finally, we use the combination of the WD-based contrastive loss $\mathcal{L}_1$ and GWD-based contrastive loss $\mathcal{L}_2$ to train the network.

samples with the perturbation of nodes, edges, or graphs. The perturbation operation may lose important information or even destroy the intrinsic structures of the graph. Thus, the constructed samples may be not discriminative enough to train the contrastive learning model. In order to construct more effective contrastive samples, we propose a learnable subgraph generation module to generate positive/negative subgraph samples. It is expected that the generated subgraphs can characterize the intrinsic local structures of the graph well.

The proposed generation module can adaptively generate the contrastive subgraph of the sampled subgraph. (For the specific sampling process, please refer to the supplementary materials). As shown in Fig. 2 (a)(b), based on the local structure information interpolation, we first generate the subgraph nodes. Then, we generate the edges of the subgraph based on the interpolated nodes.

For a specific sampled subgraph node i, we can formulate the interpolation-based generation as:

$$\hat{\boldsymbol{h}}_i = \sum_{j=1}^{\mathcal{N}_i} a_j \boldsymbol{h}_j \tag{1}$$

where $\boldsymbol{h}_j \in R^F$ is the representations of neighborhood node of center node i. $j \in \mathcal{N}_i$, $\mathcal{N}_i$ is the neighborhood of node i in the graph. $\hat{\boldsymbol{h}}_i \in R^F$ is the representations of generated subgraph node. a_j is the learned relationship weight between the neighborhood node j and the center node i. For each sampled subgraph node, we perform the interpolation based on learned neighborhood relation weights to generate new nodes. As for the learned relation weight a_j, we can define it as:

$$a_j = \frac{exp(\theta(\boldsymbol{h}_i, \boldsymbol{h}_j))}{\sum_{k=1}^{\mathcal{N}_i} exp(\theta(\boldsymbol{h}_i, \boldsymbol{h}_k))} \tag{2}$$

where $\theta(\boldsymbol{h}_i, \boldsymbol{h}_j)$ represents the relationship between center node i and neighborhood node j. We can define the $\theta(\boldsymbol{h}_i, \boldsymbol{h}_j)$ as:

$$\theta(\boldsymbol{h}_i, \boldsymbol{h}_j) = \mathrm{LeakyReLU}(\boldsymbol{W}_\theta[\boldsymbol{W}_\phi \boldsymbol{h}_i \| \boldsymbol{W}_\phi \boldsymbol{h}_j]) \tag{3}$$

where LeakyReLU is the activation function (with negative input slope 0.2), $\boldsymbol{W}_\theta \in R^{1\times 2F}$ and $\boldsymbol{W}_\phi \in R^{F\times F}$ are the weight matrixes to be learned, and $\|$ represents the feature concatenation.

As shown in Fig. 2 (c), based on the generated nodes, we directly generate the edges of the contrastive subgraph. For the node s_i and s_j in the subgraph, $s_i, s_j = 1, 2, ..., k$, k is the number of subgraph nodes, the generated edge between nodes s_i and s_j of the subgraph can be denoted as:

$$\hat{\boldsymbol{A}}(s_i, s_j) = \varphi(\hat{\boldsymbol{h}}_{s_i}, \hat{\boldsymbol{h}}_{s_j}) \tag{4}$$

where $\hat{\boldsymbol{A}}$ is the adjacency matrix of generated subgraph, $\hat{\boldsymbol{h}}_{s_i}$ is the generated features of subgraph node s_i. $\varphi(.,.)$ is the similarity calculation function, here, we use the cosine similarity, i.e., $\varphi(\hat{\boldsymbol{h}}_{s_i}, \hat{\boldsymbol{h}}_{s_j}) = \frac{\hat{\boldsymbol{h}}_{s_i}^T \hat{\boldsymbol{h}}_{s_j}}{\|\hat{\boldsymbol{h}}_{s_i}\|_2 \|\hat{\boldsymbol{h}}_{s_j}\|_2}$.

So far, we obtain the generated contrastive subgraph that contains the nodes features and edges. Essentially, we use the adaptively generated samples to replace the perturbation-based samples. Different from perturbation-based method that randomly discards the information of the graph, the proposed generation module could maintain the integrity of the graph. Our generation module, by assigning the learned attentional weights to the neighborhood nodes, can adaptively exploit the intrinsic geometric structure of the graph and generate more effective contrastive samples. Besides, since the similarity between adjacent nodes in the graph is an inherent attribute, there is a strong correlation between the central node and its neighborhood. Therefore, the generated subgraphs by neighborhood interpolation are inherently similar to the original subgraphs. And it is reasonable and effective to treat the generated subgraph as positive sample.

3.2 OT Distance Based Contrastive Learning

Most graph contrastive learning methods use node pairs or node-subgraph pairs or subgraph pairs as contrastive samples. Particularly, the features of subgraphs can be extracted with the readout function. Thus, these methods mainly employ the vector-wise similarity metrics to calculate the similarity between these samples. However, the vector inner product based similarity metrics cannot fully exploit the local structures of the graph to characterize the graph difference well. Instead of using the vector-wise similarity metrics, in our method, we introduce the

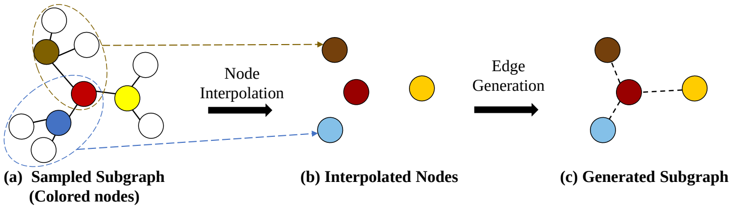


Fig. 2. The proposed interpolation-based adaptive subgraph generation module. For the sampled subgraph in (a), we use the neighborhood of each subgraph node to interpolate the new node in the subgraph (b). Based on the interpolated nodes features, we then generate the edges between the interpolated nodes. Finally, we obtain the generated subgraph (c).

optimal transport distance (i.e., Wasserstein distance and Gromov-Wasserstein distance) as the similarity metric for contrastive subgraphs. Therefore, we can accurately characterize the geometric difference between the subgraphs.

Wasserstein distance (WD). WD is commonly used for matching two discrete distributions (e.g., two sets of node embeddings) [2]. It can represent the cost of converting one subgraph to another by counting the difference between all node pairs in the two subgraphs. In our settings, WD is employed to measure the similarity between the nodes of the subgraphs. The WD for similarity calculation between subgraphs can be described as follows.

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ represent discrete distributions of two subgraphs, where $\boldsymbol{u} = \{u_1, u_2, ..., u_n\}$ and $\boldsymbol{v} = \{v_1, v_2, ..., v_m\}$, $\sum_{i=1}^{n} u_i = \sum_{j=1}^{m} v_j = 1$, n and m are the number of the subgraph nodes, respectively. The WD between the two discrete distributions $\boldsymbol{u}$ and $\boldsymbol{v}$ can be defined as:

$$D_w(\boldsymbol{u}, \boldsymbol{v}) = \min_{\boldsymbol{T} \in \pi(\boldsymbol{u}, \boldsymbol{v})} \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{T}_{ij} c(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}) \tag{5}$$

where $\pi(\boldsymbol{u}, \boldsymbol{v})$ represents all the joint distributions between two subgraphs nodes. $c(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}) = exp(-\varphi(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j})/\tau)$ denotes the transport cost between node i in subgraph 1 and node j in subgraph 2, $\boldsymbol{h}_{1i}$ and $\boldsymbol{h}_{2j}$ represent the node features, τ is a temperature parameter, and $\varphi(.,.)$ denotes the cosine similarity between the node features. The matrix $\boldsymbol{T}$ represents the transport plan, where $\boldsymbol{T}_{ij}$ denotes the amount of mass shifted from u_i to v_j. And $\boldsymbol{T}$ can be achieved by applying the Sinkhorn algorithm [6,26] with an entropic regularizer [1].

$$\min_{\boldsymbol{T} \in \pi(\boldsymbol{u}, \boldsymbol{v})} \sum_{i=1}^{n} \sum_{j=1}^{m} \boldsymbol{T}_{ij} c(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}) + \beta H(\boldsymbol{T}) \tag{6}$$

where $H(\boldsymbol{T}) = \sum_{i,j} \boldsymbol{T}_{ij} log \boldsymbol{T}_{ij}$, and β is the hyperparameter controlling the importance of the entropy term.

WD-based contrastive loss. Based on the Wasserstein distance, we define the loss as:

$$\mathcal{L}_1 = \frac{-1}{N(M+1)} \sum_{i=1}^{N} [log(exp(-D_w(\boldsymbol{s}_i, \boldsymbol{s}_p)/\tau)) + \sum_{j=1}^{M} log(1 - exp(-D_w(\boldsymbol{s}_i, \boldsymbol{s}_{nj})/\tau))] \tag{7}$$

where N is the number of sampled subgraphs, M is the number of negative samples of each subgraph, and τ is a temperature parameter. (s_i, s_p) denotes the positive sample pair, (s_i, s_{nj}) denotes the negative sample pair. To speed up the calculation efficiency, we only randomly select two negative samples, i.e., $M = 2$, one from the sampled subgraphs and the other from the generated subgraphs.

Compared with readout function-based manner, WD can exploit similar information among all nodes and distinguish the contrastive samples more effectively. Therefore, with the WD-based contrastive loss, we can maximize the similarity between the nodes across the positive subgraphs and minimize the similarity between the nodes across the negative subgraphs.

Gromov-Wasserstein distance (GWD). Unlike WD, which can directly calculate the distance of node pairs between two subgraphs, GWD [4,25] can be used when we can only get the distances between pairs of nodes within each subgraph. GWD can be used to calculate the distance between node pairs within the subgraph, as well as to measure the differences in these distances across the subgraphs. That is to say, GWD can measure the distances between node pairs within each subgraph compare to those in the counterpart subgraph. Therefore, GWD can be used to capture the similarity between the edges of the subgraphs. The GWD for similarity calculation between subgraphs can be described as:

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ represent discrete distributions of two subgraphs, where $\boldsymbol{u} = \{u_1, u_2, ..., u_n\}$ and $\boldsymbol{v} = \{v_1, v_2, ..., v_m\}$, $\sum_{i=1}^{n} u_i = \sum_{j=1}^{m} v_j = 1$, n and m is the number of subgraph nodes. The GWD between the two discrete distributions $\boldsymbol{u}$, $\boldsymbol{v}$ can be defined as:

$$D_{gw}(\boldsymbol{u}, \boldsymbol{v}) = \min_{\boldsymbol{T} \in \pi(\boldsymbol{u}, \boldsymbol{v})} \sum_{i,i',j,j'} \boldsymbol{T}_{ij} \boldsymbol{T}_{i'j'} \hat{c}(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}, \boldsymbol{h}_{1i'}, \boldsymbol{h}_{2j'}) \tag{8}$$

where $\pi(\boldsymbol{u}, \boldsymbol{v})$ denotes all the joint distributions, the matrix $\boldsymbol{T}$ represents the transport plan between two subgraphs, $\boldsymbol{T}_{ij}$ denotes the amount of mass shifted from u_i to v_j. $\hat{c}(\boldsymbol{h}_{1i}, \boldsymbol{h}_{2j}, \boldsymbol{h}_{1i'}, \boldsymbol{h}_{2j'}) = \|c(\boldsymbol{h}_{1i}, \boldsymbol{h}_{1i'}) - c(\boldsymbol{h}_{2j}, \boldsymbol{h}_{2j'})\|_2$ is the cost function to measure the edge difference between two subgraphs. $c(.,.)$ represents the distance between nodes within the subgraph.

Given the adjacent matrix of the sampled subgraph, for the nodes s_1 and s_2 of sampled subgraph s, the distance $c(\boldsymbol{h}_{s_1}, \boldsymbol{h}_{s_2})$ can be defined as:

$$c(\boldsymbol{h}_{s_1}, \boldsymbol{h}_{s_2}) = exp(-\boldsymbol{A}_s(s_1, s_2)/\tau) \tag{9}$$

where $\boldsymbol{A}_s(s_1, s_2)$ represents the connection relationship between node s_1 and node s_2 of the sampled subgraph, τ is a temperature parameter.

The distance between the generated subgraph nodes can be defined as:

$$c(\hat{\boldsymbol{h}}_{s_1}, \hat{\boldsymbol{h}}_{s_2}) = exp(-\hat{\boldsymbol{A}}_s(s_1, s_2)/\tau) \tag{10}$$

where $\hat{\boldsymbol{A}}_s(s_1, s_2) = \varphi(\hat{\boldsymbol{h}}_{s_1}, \hat{\boldsymbol{h}}_{s_2})$ represents the connection relationship between the node s_1 and node s_2 of generated subgraph, $\varphi(.,.)$ represents the consine similarity, τ is a temperature parameter.

GWD-based contrastive loss. Based on the Gromov-Wasserstein distance, we define the loss as:

$$\mathcal{L}_2 = \frac{-1}{N(M+1)} \sum_{i=1}^{N} [log(exp(-D_{gw}(\boldsymbol{s}_i, \boldsymbol{s}_p)/\tau)) + \sum_{j=1}^{M} log(1 - exp(-D_{gw}(\boldsymbol{s}_i, \boldsymbol{s}_{nj})/\tau))] \tag{11}$$

where N is the number of sampled subgraphs, M is the number of negative samples of each subgraph and we also set $M = 2$, τ is a temperature parameter. (s_i, s_p) and (s_i, s_{nj}) denote positive and negative sample pairs. The GWD-based contrastive loss can maximize the similarity between the edges of the positive subgraphs and minimize the similarity between the edges of the negative subgraphs so that the geometry difference between the subgraphs can be captured.

Finally, we obtain the final loss function $\mathcal{L}$, which is defined as follows:

$$\mathcal{L} = \lambda \mathcal{L}_1 + (1 - \lambda)\mathcal{L}_2 \tag{12}$$

where λ is the hyper-parameter for controlling the importance of different loss functions. Here, we set $\lambda = 0.5$. To the best of our knowledge, we are the first to introduce OT into subgraph-based graph contrastive learning. We use WD to exploit contrastive information based on the subgraph node features, and GWD to exploit contrastive information based on the edges of the subgraph. Therefore, the OT-based contrastive loss can better guide the training of the encoder.

4 Experiment

In this section, extensive experiments are conducted to evaluate the performance of our method in a self-supervised manner on the transductive and inductive node classification tasks. And we compare our method with other baselines, including unsupervised and supervised methods. Besides, we conduct ablation studies to verify the effectiveness of our proposed method.

4.1 Datasets

In order to evaluate the effectiveness of our method, we conduct experiments on five benchmark datasets of three real-world networks. Following [10,17], three tasks are performed: (1) classifying the topics of the documents on the citation network datasets of Cora, Citeseer and Pubmed [28]; (2) classifying protein roles of protein-protein interaction (PPI) networks [46], and generalizing to unseen networks; (3) predicting the community structure of the social network on Reddit posts [40]. More detailed descriptions can be found in the supplementary materials.

4.2 Implementation details

Due to different attributes of datasets, we employ distinct encoders for three experimental settings, i.e., transductive learning for the small graph, inductive learning for the large graph and multiple graphs. For more specific information of encoders may be found in [32], or please refer to the supplementary materials.

All experiments are implemented using PyTorch [3] and the geometric deep learning extension library [8]. The experiments are conducted on a single TITAN RTX GPU. Our method is used to learn node representations in a self-supervised manner, followed by evaluating the learned representations with the node classification task. This is performed by training and testing a simple linear (logistic regression) classifier in the downstream tasks using the learned representations. We train the model by minimizing the loss function provided in Eq. (12). And we use Adam optimizer [16] with an initial learning rate of 0.0001 (especially, 0.0005 on Pubmed). The dimension of node representations is 1024 (256 with 4 heads for PPI). In order to avoid the excessive calculation, in every epoch, we randomly sample some subgraphs to calculate the OT distance for the loss calculation. Besides, parameter $\boldsymbol{T}$ is shared by WD and GWD. The detailed parameter settings can be found in the supplementary materials.

4.3 Results

Classification results. We choose four state-of-the-art graph contrastive learning methods to evaluate graph embeddings, DGI [32], GMI [23], GraphCL [9] and Subg-Con [14]. And two traditional unsupervised methods, DeepWalk [24] and the unsupervised variant of GraphSAGE [10] are also compared with our method. Specially, we also provide results for training the classifier on the raw input features. Besides, we report the results of three supervised graph neural networks, GCN [17], GAT [31] and GraphSAGE [10]. For the node classification task, we employ mean classification accuracy to evaluate the performance on Cora, Citeseer, and Pubmed datasets, while the micro-averaged F1 score for the Reddit and PPI datasets.

The evaluation results on the five datasets are listed in Table. 1. The results demonstrate that our method has achieved good performance across all five datasets. As can be seen in Table. 1, our method successfully outperforms all the competing graph contrastive learning approaches, which implies the potential of our proposed method for the node classification task. Compared with the traditional subgraph perturbation based GraphCL, our method has the gain of at least 1% improvement on all data sets, even 3.2% on PPI. This indicates that our subgraph generation module can effectively capture the intrinsic local structures of the graph. We further observe that the performance of our method is better than other vector inner product based methods, which verifies that our OT-based similarity metric can effectively characterize the graph difference by exploiting local structures of the graph. From this table, one can also see that although the labels of the nodes are used in the supervised graph representation learning

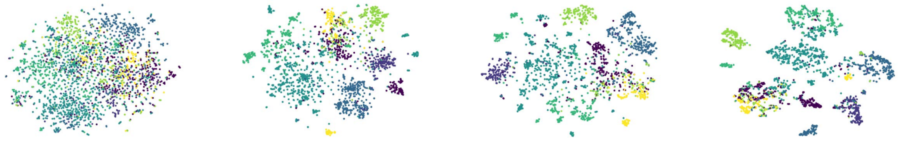
Table 1. Performance comparison of node classification with different methods on the transductive and inductive tasks. The third column illustrates the data used by each algorithm in the training phase, where X, A, and Y denote features, adjacency matrix, and labels, respectively. For simple expression, we abbreviate our method as GSC.

	Algorithm	Data	Transductive			Inductive	
			Cora	Citeseer	Pubmed	PPI	Reddit
Supervised	GCN [17]	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	81.4±0.6	70.3±0.7	76.8±0.6	51.5±0.6	93.3±0.1
	GAT [31]	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	83.0±0.7	72.5±0.7	79.0±0.3	**97.3±0.2**	-
	GraphSAGE [10]	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	79.2±1.5	71.2±0.5	73.1±1.4	51.3±3.2	92.1±1.1
Unsupervised	Raw features	$\boldsymbol{X}$	56.6±0.4	57.8±0.2	69.1±0.2	42.5±0.3	58.5±0.1
	DeepWalk [24]	$\boldsymbol{A}$	67.2	43.2	65.3	52.9	32.4
	GraphSAGE [10]	$\boldsymbol{X}, \boldsymbol{A}$	75.2±1.5	59.4±0.9	70.1±1.4	46.5±0.7	90.8±1.1
	DGI [32]	$\boldsymbol{X}, \boldsymbol{A}$	82.3±0.6	71.8±0.7	76.8±0.6	63.8±0.2	94.0±0.1
	GMI [23]	$\boldsymbol{X}, \boldsymbol{A}$	83.0±0.3	73.0±0.3	79.9±0.2	65.0±0.0	95.0±0.0
	GraphCL [9]	$\boldsymbol{X}, \boldsymbol{A}$	83.6±0.5	72.5±0.7	79.8±0.5	65.9±0.6	95.1±0.1
	Subg-Con [14]	$\boldsymbol{X}, \boldsymbol{A}$	83.5±0.5	73.2±0.2	81.0±0.1	66.9±0.2	95.2±0.0
	GSC (ours)	$\boldsymbol{X}, \boldsymbol{A}$	**84.6±0.1**	**73.7±0.1**	**82.1±0.2**	**69.1±0.3**	**95.3±0.1**

methods, the proposed graph contrastive learning method can still outperform most of the supervised methods.

We also conduct node classification experiments with few contrastive training samples to evaluate our method. When using 70%, 50%, 30% and 10% contrastive samples on the Cora dataset, the performance of Subg-Con is 83.0%, 82.3%, 81.2% and 79.6%, while the performance of GraphCL is 83.2%, 82.5%, 81.6% and 80.1%, respectively. Nonetheless, by generating multiple positive samples for each original subgraph, the performance of our method is 84.6%, 84.1%, 83.6% and 82.9%, respectively. Particularly, in the case of 10% contrastive samples, our method can obtain the gain of 3.3%. Since Subg-Con can only construct one positive sample by pooling the top-k important neighbors for each subgraph, the performance is limited in the cases of few contrastive training samples. Different from Subg-Con, based on neighborhood interpolation, our subgraph generation module can generate multiple positive samples for each samples, which can effectively handle the situation of insufficient contrastive samples. Although GraphCL can also construct multiple positive samples, the perturbation-based data augmentation manner may change the original attributes of the graph. Compared with GraphCL, our neighborhood interpolation based generation module can effectively preserve the local structures of the graph. For more results, please refer to supplementary materials.

Visual results. As shown in Fig. 3, we also visualize the raw features and learned embeddings of the graphs with the t-SNE [21] plot for different graph contrastive learning methods, including DGI, Subg-Con and GSC (ours). From the visualization results in Fig. 3, one can see that the embeddings generated by GSC can exhibit closer clusters than the other three methods. This means that our graph contrastive learning can obtain more discriminative features.



(a) Raw features. (b) DGI. (c) Subg-Con. (d) GSC (ours).

Fig. 3. Visualization of t-SNE embeddings of raw features, DGI, Sub-Con, and GSC (ours) on Cora dataset.

4.4 Ablation studies

Effectiveness of generation module and OT distance. To further verify the effectiveness of different modules, we conduct three sets of comparative experiments on Cora, Citeseer and, Pubmed datasets. To verify the effectiveness of the OT distance based contrastive loss, we use the readout function to obtain the feature vectors of the subgraphs and calculate the vector-wise similarity. As can be seen from Table. 2, in comparison with Readout, the use of the OT distance can improve the classification performance. This demonstrates that the OT distance can effectively capture local structure information of the graph and distinguish different subgraphs.

To demonstrate the effectiveness of the generation module, we use the traditional perturbation on the subgraph to replace the generation module and calculate the similarity between subgraphs with the OT distance. From Table. 2, we can see that the classification performance of Generation + OT outperforms that of Perturbation + OT. This can verify that our generation module can effectively capture the intrinsic local structures of the graph. The generated samples can improve the performance of graph contrastive learning.

Table 2. Ablation studies on different modules

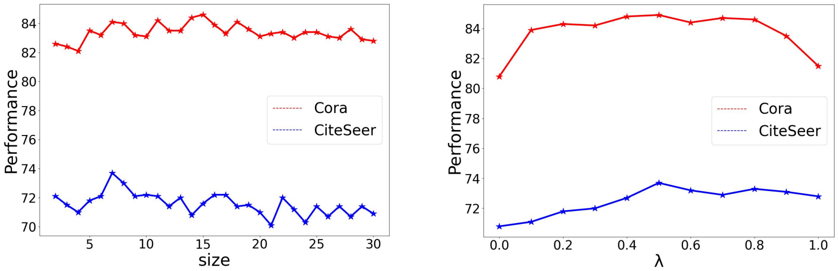
	Cora	Citeseer	Pubmed
Gene + Readout	83.6	72	78.5
Perturbation + OT	84.1	72.3	80.4
Gene + OT (ours)	**84.6**	**73.7**	**82.1**

Table 3. Ablation studies on different sampling methods

	Cora	Citeseer	Pubmed
Importance Score	83.5	71.68	80.6
Random Walk	84.2	72.4	81.2
BFS (ours)	**84.6**	**73.7**	**82.1**

Effectiveness of different sampling methods. We compare different subgraph sampling methods (i.e., importance score [14] and random walk [39]) and list the experiment results in Table 3. To make the comparisons fair, we sample subgraphs of the same size. As can be seen from Table. 3, BFS-based method can achieve the best performance compared with other sampling methods. This can verify that BFS-based sampled subgraphs can better cover local information and can be more beneficial to distinguish between subgraphs.

Influence of the subgraph size and parameter λ. To study the influence of the subgraph size in our method, conduct experiments on the Cora and Citeseer datasets by varying the numbers of subgraph nodes from 2 to 30. As can



(a) Different subgraph sizes

(b) Different λ

Fig. 4. Ablation studies of different subgraph sizes and parameter λ.

be seen from Fig. 4 (a), the classification performance is slightly fluctuated with different subgraph sizes. Besides, we vary the parameter λ from 0 to 1 to study the effects on the final classification accuracy. Different λ can control different weights of the loss terms. As shown in Fig. 4 (b), both node and edge features have effects on the final performance and the classification performance can be kept relatively stable in [0.4, 0.8]. The best performance can be achieved when λ is set to 0.5, where the contributions of node and edge features reach a balance.

5 Conclusion

In this paper, we proposed a novel subgraph generation method for graph contrastive learning. Based on the neighborhood interpolation, we developed the subgraph generation module, which can adaptively generate the contrastive subgraphs. Furthermore, we employed two types of optimal transport distances (i.e., Wasserstein distance and Gromov-Wasserstein distance) to calculate the similarity between subgraphs. In particular, generated subgraphs and the OT distance based similarity metric can effectively capture the intrinsic local structures of the graph to characterize the graph difference well. By conducting extensive experiments on multiple benchmark datasets, we demonstrate that our proposed graph contrastive learning method can yield better performance in comparison with the supervised and unsupervised graph representation learning methods.

Acknowledgments

The authors would like to thank reviewers for their detailed comments and instructive suggestions. This work was supported by the National Science Fund of China (Grant Nos. 61876084, 61876083, 62176124).

References

1. Benamou, J.D., Carlier, G., Cuturi, M., Nenna, L., Peyré, G.: Iterative bregman projections for regularized transportation problems. SIAM Journal on Scientific Computing **37**(2), A1111–A1138 (2015) 8
2. Chen, L., Gan, Z., Cheng, Y., Li, L., Carin, L., Liu, J.: Graph optimal transport for cross-domain alignment. In: International Conference on Machine Learning. pp. 1542–1553. PMLR (2020) 2, 8
3. Chollet, F.: Deep learning with Python. Simon and Schuster (2017) 11
4. Chowdhury, S., Mémoli, F.: The gromov–wasserstein distance between networks and stable network invariants. Information and Inference: A Journal of the IMA **8**(4), 757–787 (2019) 9
5. Chu, G., Wang, X., Shi, C., Jiang, X.: Cuco: Graph representation with curriculum contrastive learning. In: IJCAI (2021) 4
6. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. Advances in neural information processing systems **26**, 2292–2300 (2013) 8
7. Faerman, E., Voggenreiter, O., Borutta, F., Emrich, T., Berrendorf, M., Schubert, M.: Graph alignment networks with node matching scores. Proceedings of Advances in Neural Information Processing Systems (NIPS) (2019) 2
8. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019) 11
9. Hafidi, H., Ghogho, M., Ciblat, P., Swami, A.: Graphcl: Contrastive self-supervised learning of graph representations. arXiv preprint arXiv:2007.08025 (2020) 2, 4, 11, 12
10. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1025–1035 (2017) 2, 3, 10, 11, 12
11. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 (2017) 1
12. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: International Conference on Machine Learning. pp. 4116–4126. PMLR (2020) 2, 4
13. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670 (2018) 3
14. Jiao, Y., Xiong, Y., Zhang, J., Zhang, Y., Zhang, T., Zhu, Y.: Sub-graph contrast for scalable self-supervised graph representation learning. arXiv preprint arXiv:2009.10273 (2020) 2, 4, 11, 12, 13
15. Jin, W., Derr, T., Liu, H., Wang, Y., Wang, S., Liu, Z., Tang, J.: Self-supervised learning on graphs: Deep insights and new direction. arXiv preprint arXiv:2006.10141 (2020) 2
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 11
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016) 2, 3, 10, 11, 12
18. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016) 2
19. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. In: International Conference on Machine Learning. pp. 3734–3743. PMLR (2019) 2

20. Lee, N., Lee, J., Park, C.: Augmentation-free self-supervised learning on graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 7372–7380 (2022) 4
21. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008) 12
22. Manessi, F., Rozza, A.: Graph-based neural network models with multiple self-supervised auxiliary tasks. Pattern Recognition Letters **148**, 15–21 (2021) 2
23. Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., Huang, J.: Graph representation learning via graphical mutual information maximization. In: Proceedings of The Web Conference 2020. pp. 259–270 (2020) 4, 11, 12
24. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014) 11, 12
25. Peyré, G., Cuturi, M., Solomon, J.: Gromov-wasserstein averaging of kernel and distance matrices. In: International Conference on Machine Learning. pp. 2664–2672. PMLR (2016) 9
26. Peyré, G., Cuturi, M., et al.: Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning **11**(5-6), 355–607 (2019) 8
27. Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., Tang, J.: Gcc: Graph contrastive coding for graph neural network pre-training. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1150–1160 (2020) 4
28. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI magazine **29**(3), 93–93 (2008) 10
29. Sun, F.Y., Hoffmann, J., Verma, V., Tang, J.: Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. arXiv preprint arXiv:1908.01000 (2019) 4
30. Suresh, S., Li, P., Hao, C., Neville, J.: Adversarial graph augmentation to improve graph contrastive learning. arXiv preprint arXiv:2106.05819 (2021) 2, 5
31. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017) 2, 3, 11, 12
32. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. arXiv preprint arXiv:1809.10341 (2018) 2, 3, 11, 12
33. Wang, H., Zhang, J., Zhu, Q., Huang, W.: Augmentation-free graph contrastive learning. arXiv preprint arXiv:2204.04874 (2022) 4
34. Wu, L., Lin, H., Gao, Z., Tan, C., Li, S., et al.: Self-supervised on graphs: Contrastive, generative, or predictive. arXiv preprint arXiv:2105.07342 (2021) 2
35. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020) 2
36. Xu, M., Wang, H., Ni, B., Guo, H., Tang, J.: Self-supervised graph-level representation learning with local and global structure. arXiv preprint arXiv:2106.04113 (2021) 4
37. Yin, Y., Wang, Q., Huang, S., Xiong, H., Zhang, X.: Autogcl: Automated graph contrastive learning via learnable view generators. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 8892–8900 (2022) 5
38. You, Y., Chen, T., Shen, Y., Wang, Z.: Graph contrastive learning automated. arXiv preprint arXiv:2106.07594 (2021) 5

39. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. Advances in Neural Information Processing Systems **33** (2020) 4, 13
40. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: Graphsaint: Graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931 (2019) 10
41. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. Advances in Neural Information Processing Systems **31**, 5165–5175 (2018) 2
42. Zhao, H., Yang, X., Wang, Z., Yang, E., Deng, C.: Graph debiased contrastive learning with joint representation clustering. IJCAI (2021) 4
43. Zhu, Q., Du, B., Yan, P.: Self-supervised training of graph convolutional networks. arXiv preprint arXiv:2006.02380 (2020) 2
44. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020) 2, 4
45. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021. pp. 2069–2080 (2021) 2
46. Zitnik, M., Leskovec, J.: Predicting multicellular function through multi-layer tissue networks. Bioinformatics **33**(14), i190–i198 (2017) 10