SOLA-GCL: Subgraph-Oriented Learnable Augmentation Method for Graph Contrastive Learning

Tianhao Peng[1], Xuhong Li[2], Haitao Yuan[3*], Yuchen Li[2, 4], Haoyi Xiong[2*]
[1]Beihang University
[2]Baidu Inc.
[3]Nanyang Technological University
[4]Shanghai Jiao Tong University
pengtianhao@buaa.edu.cn, lixuhong@baidu.com, yuchenli@sjtu.edu.cn
haitao.yuan@ntu.edu.sg, haoyi.xiong.fr@ieee.org

Abstract

Graph contrastive learning has emerged as a powerful technique for learning graph representations that are robust and discriminative. However, traditional approaches often neglect the critical role of subgraph structures, particularly the intra-subgraph characteristics and inter-subgraph relationships, which are crucial for generating informative and diverse contrastive pairs. These subgraph features are crucial as they vary significantly across different graph types, such as social networks where they represent communities, and biochemical networks where they symbolize molecular interactions. To address this issue, our work proposes a novel subgraph-oriented learnable augmentation method for graph contrastive learning, termed SOLA-GCL, that centers around subgraphs, taking full advantage of the subgraph information for data augmentation. Specifically, SOLA-GCL initially partitions a graph into multiple densely connected subgraphs based on their intrinsic properties. To preserve and enhance the unique characteristics inherent to subgraphs, a graph view generator optimizes augmentation strategies for each subgraph, thereby generating tailored views for graph contrastive learning. This generator uses a combination of intra-subgraph and inter-subgraph augmentation strategies, including node dropping, feature masking, intra-edge perturbation, inter-edge perturbation, and subgraph swapping. Extensive experiments have been conducted on various graph learning applications, ranging from social networks to molecules, under semi-supervised learning, unsupervised learning, and transfer learning settings to demonstrate the superiority of our proposed approach.

Introduction

Graph structures are used to represent data such as social networks (Zhou et al. 2020), molecules (Li et al. 2023b; Zhang et al. 2023b), and traffic flows (Yuan and Li 2021; Yuan et al. 2023). Graph neural networks (GNNs) (Kipf and Welling 2017; Velickovic et al. 2018; Xu et al. 2019; Yun et al. 2019; Li et al. 2022) have become popular for graph representation learning (Lin et al. 2023; Tang et al. 2023; Li et al. 2023f; Cai et al. 2023; Yuan et al. 2021; Li et al. 2025), but they require labeled data. To learn from unlabeled data, contrastive learning has been adapted to graph data (You et al. 2020, 2021; Velickovic et al. 2019; Suresh et al. 2021; Yin et al. 2022). This involves generating two views by perturbing the graph and learning representations by maximizing feature consistency (You et al. 2020, 2021; Velickovic et al. 2019; Li et al. 2023c). Graph-structured data are more complex than image or text data due to their properties and distribution shifts (Hassani 2022; Li et al. 2023e; Yuan, Cong, and Li 2024). Methods like DGI (Velickovic et al. 2019), GRACE (Zhu et al. 2020), and GraphCL (You et al. 2020) use random augmentations, but lack automatic selection of augmentation policies, affecting the graphs' semantic integrity.

Recent advances in Graph Contrastive Learning (GCL) leverage methods like JOAO (You et al. 2021) and LG2AR (Hassani and Ahmadi 2022) adaptively select one augmentation strategy from a predefined pool for specific graph data, followed by random graph view generation process such as node masking and edge dropping. While these methods improve the adaptability of selecting augmentation strategies for individual graphs, their reliance on random view generation may change the original graph semantics. Despite some advancements with end-to-end methods like AD-GCL and AutoGCL, which introduce learnable augmentations, the full potential of subgraph information is still largely unexplored. Methods like MSSGCL (Liu et al. 2023) and SUBG-CON (Jiao et al. 2020) attempt to address this by focusing on the relationships between sampled subgraphs and the entire graph or the central nodes, respectively. However, they still struggle to capture the intra-subgraph characteristics and inter-subgraph relationships (e.g., communities in social networks). Furthermore, while these learnable methods manage to preserve the semantics of original graphs, they lack adaptability across different datasets due to their reliance on uniform augmentation strategies (You et al. 2021, 2020; Li et al. 2024; Lyu et al. 2024), which ultimately limits their flexibility. Here, we conclude the limitations of existing methods as follows.

- *Loss of intra-subgraph and inter-subgraph information:* Graphs contain important details within subgraphs (intra-subgraph characteristics) and between different subgraphs (inter-subgraph relationships). For instance, in social networks, distinct communities exhibit unique characteristics, while in chemistry, combinations of functional

*Both Haitao Yuan and Haoyi Xiong are corresponding authors.

groups determine molecular functionalities. Previous studies have not effectively captured the critical information both within and between subgraphs. *How to effectively capture both intra-subgraph characteristics and inter-subgraph relationships* remains an ongoing challenge.

- *Poor adaptability and losing semantic information:* Different graph structures necessitate distinct augmentation strategies. For example, molecular graphs may require edge perturbation, whereas social community graphs might benefit more from node dropping (You et al. 2021, 2020). Methods like JOAO, which lack a learnable view generation process, can offer improved adaptability but at the cost of losing semantic information. Conversely, methods with uniform augmentation strategies that include a learnable view generation process, such as AD-GCL, preserve semantic information but often fail to adapt effectively across diverse datasets. *Improving the adaptability of GCL methods while preserving semantic integrity* remains a significant challenge.

To address above challenges, We propose a novel **S**ubgraph-**O**riented **L**earnable **A**ugmentation method for **G**raph **C**ontrastive **L**earning (SOLA-GCL) to tackle existing challenges by leveraging subgraph information for data augmentation. The process begins by dividing the original graph into connected subgraphs using partitioning algorithms, such as the Louvain method (Blondel et al. 2008) for community graphs or RDKit (Landrum 2013) for molecular structures, to identify functional groups. A graph view generator then applies multiple augmentation strategies (node dropping, edge perturbation, subgraph swapping) to these subgraphs, assembling them into a new graph view. This involves a subgraph augmentation selector, which learns the optimal augmentation strategy distribution; a subgraph view generator for implementing these strategies; and a subgraph view assembler. Our method demonstrated superior performance in extensive graph classification experiments across various learning tasks, outperforming state-of-the-art graph contrastive learning approaches.

The main contributions of this study are as follows:

- A novel graph contrastive learning framework, termed SOLA-GCL, is proposed. To the best of our knowledge, this is the first work to build learnable generative augmentation policies that specifically focus on the intra-subgraph characteristics and inter-subgraph relationships within graphs.
- The proposed SOLA-GCL introduces an end-to-end differentiable training algorithm, enabling automatic augmentation strategy selection and graph view generation.
- Extensive experiments are conducted on a variety of graph classification datasets with semi-supervised, unsupervised, and transfer learning settings, showcasing the robustness and effectiveness of our SOLA-GCL framework on graph classification tasks.

Related Work

In this section, we first introduce the studies relevant to our work from the perspectives of graph neural networks, graph partition algorithms, and graph contrastive learning (GCL) algorithms upon views generated by various graph data augmentation strategies. Later, we discuss the unique contributions made by this work compared to previous studies.

Graph Neural Networks

Graph neural networks (GNNs) are the extension of the neural network models onto graph data (Wu et al. 2021; Li et al. 2023d; Zhang et al. 2023a; Yuan, Li, and Bao 2022; Xiong et al. 2024). Most existing GNNs adopt the message-passing framework and use permutation-invariant local aggregation schemes to update node representations. For instance, GCN (Kipf and Welling 2017) averages features of all neighboring nodes. GAT (Velickovic et al. 2018) uses an attention mechanism to assign different weights to neighboring nodes. GraphSAGE (Hamilton, Ying, and Leskovec 2017) samples fixed-size neighbors of a node and aggregates their features for realizing fast and scalable GNN training. GIN (Xu et al. 2019) adjusts the weight of the central node by a learnable parameter to distinguish different graph structures based on the graph embedding. ResGCN (Pei et al. 2022) combine the residual connection with GCN to build deeper GNNs.

In this study, we employ two state-of-the-art GNNs, GIN (Xu et al. 2019) and ResGCN (Pei et al. 2022), as our backbone GNNs, following the existing graph contrastive learning literature (You et al. 2020, 2021; Yin et al. 2022). We believe our work could complement with the line of research on GNN while incorporating advanced GNN models for potential improvement in future studies.

Graph Partition Algorithm

Graph partitioning divides a graph into communities based on node and edge connectivity. Densely connected nodes form communities, while sparsely connected nodes do not. Spectral clustering approximates graph partition solutions using eigenvalue decomposition on the normalized graph Laplacian, but it is computationally expensive. The Louvain method (Blondel et al. 2008) quickly optimizes modularity, a measure of intra-community edge density relative to inter-community edges. The Girvan–Newman (GN) algorithm (Girvan and Newman 2002) identifies communities by removing edges iteratively. However, these methods often ignore node features, despite their richness in recent applications. SGCN (Wang et al. 2021) addresses this by detecting community centers without prior labels. RDKit is a software suite for cheminformatics, computational chemistry, and predictive modeling, which offers graph partition algorithms for molecular graphs (Landrum 2013). In our work, we use the Louvain method (Blondel et al. 2008) and RDKit (Landrum 2013) for graph partitioning before GCL training, but our framework is adaptable to other partition algorithms, including deep learning-based approaches.

GCL with Data Augmentation

In recent years, GCL with data augmentation has attracted significant attention for self-supervised graph learning. The main idea is to maximize the agreement between representations of a graph in augmented views (Li et al. 2023a).

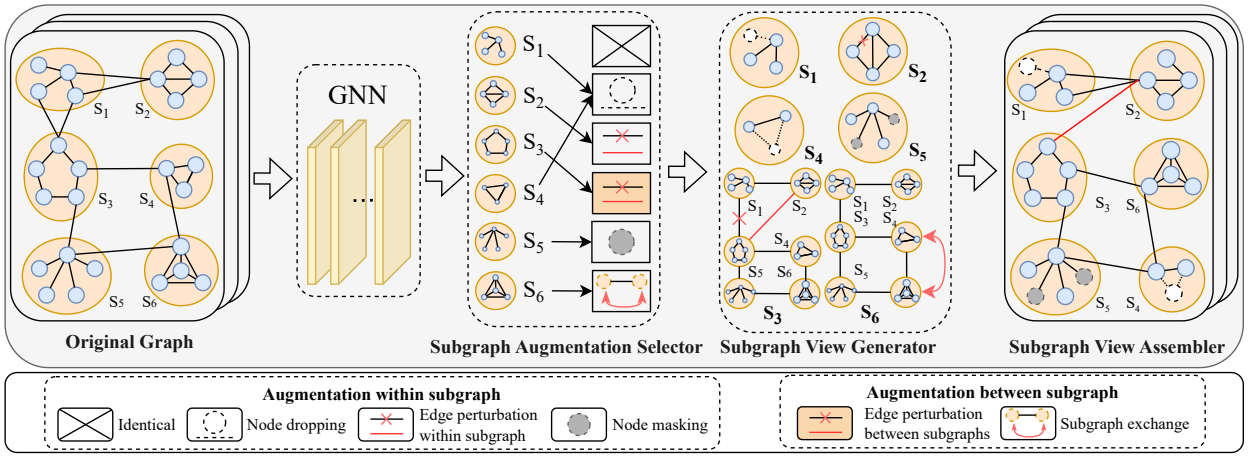

Figure 1: An illustration of the proposed SOLA-GCL framework. The graph view generator is composed of three critical components: the subgraph augmentation selector, the subgraph view generator, and the subgraph view assembler. The subgraph augmentation selector learns to choose the optimal augmentation strategy for each subgraph, and the subgraph view generator outputs augmented subgraph views according to the selected strategies. The subgraph view assembler constructs an augmented graph view based on these augmented subgraph views.

Recently, GRACE (Zhu et al. 2020) introduces two general types of augmentations (edge perturbation and attribute masking), while GraphCL (You et al. 2020) proposes four (node dropping, edge perturbation, attribute masking, and subgraph sampling), applying these strategies randomly, which limits task adaptability. To address this, JOAO (You et al. 2021) and GPA (Zhang et al. 2022) adaptively select suitable augmentation strategies by learning the distribution of graph datasets. However, these methods often apply random augmentations to specific graphs, resulting in sub-optimal performance. To enhance adaptability in GCL models, AD-GCL (Suresh et al. 2021) uses adversarial training to perturb edges and generate graph views, while AutoGCL (Yin et al. 2022) focuses on perturbing nodes for augmented views. Despite their success, these approaches often overlook the importance of subgraphs. Methods like SUBG-CON (Jiao et al. 2020) and MSSGCL (Liu et al. 2023) have explored subgraph sampling for contrastive learning by generating global and local views at different scales. However, they typically fail to leverage the intra-subgraph characteristics and inter-subgraph relationships fully. To address these limitations, we propose a subgraph-centered method that generates augmented subgraph views and assembles them into an augmented graph, utilizing subgraph information in a learnable manner.

Discussion on Most Relevant Works

Our GCL framework introduces a learnable graph view generator that focuses on subgraphs to enhance graph understanding (Adhikari et al. 2018; Zhu et al. 2021a). The most relevant works to our study are JOAO and MSSGCL, from the perspectives of model adaptability and subgraph information modeling, respectively.

From the perspective of model adaptability, both JOAO (You et al. 2021) and our SOLA-GCL learn the probability distribution of data augmentation strategies. While JOAO applies the random graph view generation process that risks altering the original graph's semantics, our SOLA-GCL generates new graph views in a learnable manner, thus preserving the semantic integrity. From the perspective of subgraph information modeling, our proposed SOLA-GCL method differs significantly from MSSGCL (Liu et al. 2023), which focuses on the relationship between the original graph and its sampled subgraph. In contrast, SOLA-GCL method delves deeper, fully exploiting both the crucial intra-subgraph characteristics and inter-subgraph relationships by applying learnable targeted augmentation strategies within and between subgraphs.

Methodology

This section elaborates on the proposed SOLA-GCL framework for graph classification in detail. With an overview shown in Fig. 1, SOLA-GCL first partitions the graph into several densely connected subgraphs, and then trains the subgraph augmentation selector and subgraph view generator jointly in an end-to-end manner to generate graph views for graph contrastive learning.

The Basic GNN Module

SOLA-GCL employs a vanilla GNN with L local aggregation layers, enabling nodes to access information from L-hop neighbors, as the basic module. Thus, for a given node $v \in \mathcal{V}$, the l-th layer's calculation formula in an L-layer GNN (l=1, 2, ..., L) is as follows:

$$h_v^{(l)} = \text{UPDATE}^{(l)}\left(h_v^{(l-1)}, \underset{\forall u \in \mathcal{N}(v)}{\text{AGGREGATE}^{(l)}}\left(\left\{h_u^{(l-1)}\right\}\right)\right) \tag{1}$$

where $h_v^{(l)}$ is the feature of a node v in the lth layer, and $h_v^{(0)}$=x_v, x_v is the original feature vector of node v;

AGGREGATE$^{(l)}(\cdot)$ and UPDATE$^{(l)}(\cdot)$ represent the feature aggregation function (e.g., mean, LSTM, and max pooling) and feature update function (e.g., linear-layer combination and MLP) (Ding et al. 2022), respectively; $\mathcal{N}(v)$ represents the one-hop neighboring node set of a node v.

Learnable Graph View Generator

Given a graph G=$(\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V}$=$\{v_1, v_2, ..., v_N\}$ denotes a node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is an edge set, $X \in \mathbb{R}^{N \times d}$ represents node features, the graph view generator aims to create an augmented view without semantic labels during training. With an overview shown in Fig. 1, the graph view generator is composed of three critical components: the ***subgraph augmentation selector***, the ***subgraph view generator***, and the ***subgraph view assembler***. The subgraph augmentation selector aims to select the optimal augmentation strategy for each subgraph, then the subgraph view generator generates augmented views for subgraphs, and the subgraph view assembler generates an augmented graph view by combining the augmented subgraph views.

Subgraph Augmentation Selector In studying different types of networks such as social and module graphs, it becomes crucial to extract and differentiate key information specific to each network type. To effectively implement GCL on various dataset-specific subgraphs, we utilize graph partition algorithms like the Louvain algorithm (Blondel et al. 2008), which partitions a graph into densely connected subgraphs. For a graph G=$(\mathcal{V}, \mathcal{E}, X)$, the partition process can be formally expressed as follows: $\{\mathbf{S_1}, \mathbf{S_2}, ..., \mathbf{S_k}\}$=ALGO$(G)$, where $\mathbf{S_i}$ represents the i-th subgraph partitioned by algorithm ALGO$(\cdot)$. Later, an L-layer GNN processes node attributes to extract embeddings, which are then aggregated to form subgraph embeddings used for selecting appropriate augmentation strategies from a pool (node drop, feature mask, edge perturbation, subgraph swap) as detailed in . The Gumbel-Softmax technique (Jang, Gu, and Poole 2017) is employed to probabilistically assign augmentation operations to these partitions, enhancing the distinctiveness of the subgraph analysis.

For each subgraph $\mathbf{S}$, the selection of the augmentation strategy can be formulated as follows:

$$p_{\mathbf{S}} = \text{FUNC}(\text{READOUT}\left(\left\{h_v^{(L)} : v \in \mathbf{S}\right\}\right)) \quad (2)$$

$$f_{\mathbf{S}} = \text{GumbelSoftmax}\left(p_{\mathbf{S}}\right) \quad (3)$$

where $h_v^{(L)}$ denotes the embedding of node v. READOUT denotes a function that summarizes over all node embeddings in subgraph S. FUNC is a function (e.g. MLP or a linear layer) that transforms the embedding of the subgraph to the probability distribution $p_{\mathbf{S}}$. This distribution represents the probabilities across all potential augmentations for each subgraph. $f_{\mathbf{S}}$ represents the augmentation choice for subgraph $\mathbf{S}$. $f_{\mathbf{S}}$ is a differentiable one-hot vector sampled via Gumbel-Softmax using the reparameterization trick.

Subgraph View Generator For each subgraph, the subgraph view generator would select an augmentation strategy based on $f_{\mathbf{S}}$, and generate a subgraph view for the subgraph. In this study, we use five augmentation strategies, including three intra-subgraph strategies (node dropping, feature masking, and intra-edge perturbation) and two inter-subgraph strategies (inter-edge perturbation and subgraph swapping). The details are as follows.

(1) **Node dropping** is conditioned on the node representations to decide which nodes within a subgraph to drop. Given the subgraph $\mathbf{S}$, the process of node dropping can be formulated as:

$$p_v = \text{FUNC}(h_v^{(L)}) \quad (4)$$

$$f_v = \text{GumbelSoftmax}\left(p_v\right) \quad (5)$$

$$\tilde{X}_{drop}, \tilde{\mathcal{E}}_{drop} = \text{AUG}_{drop}(X, \mathcal{E}, f_v, f_{\mathbf{S}}) \quad (6)$$

where FUNC is a function (e.g. MLP or a linear layer) that transforms the embeddings of nodes to the probability distribution p_v. This distribution indicates the likelihood of each node being dropped or retained. f_v is a one-hot vector sampled from this distribution via Gumbel-Softmax, $\text{AUG}_{drop}(X, \mathcal{E}, f_v, f_{\mathbf{S}})$ is the augmentation function that outputs the augmented node features $\tilde{X}_{drop}$ and edges $\tilde{\mathcal{E}}_{drop}$. The underlying assumption is that missing part of nodes does not damage the semantic information of the graph.

(2) **Feature masking** is to mask the feature of nodes within a subgraph. The process is similar to node dropping augmentation. Feature masking implies that the absence of some node features does not affect the semantics.

(3) **Intra-edge perturbation** is conditioned on head and tail nodes to decide which edges that are within a subgraph to add or remove. It would be a heavy burden for back-propagation to predict the full adjacency matrix when dealing with large-scale graphs. To achieve efficient computation, we randomly sample negative edges within subgraphs. The underlying prior is that the semantic meaning of the graph is robust to the variance of edges.

(4) **Inter-edge perturbation** focuses on perturbing the edges between two subgraphs $\mathbf{S}_i$ and $\mathbf{S}_j$. Similarly, we first randomly sample $|\mathcal{E}|$ negative edges between subgraphs, and the process of inter-edge perturbation can be formulated as:

$$p_e = \text{FUNC}(h_v^{(L)} \parallel h_u^{(L)} \parallel h_{\mathbf{S}_i}^{(L)} \parallel h_{\mathbf{S}_j}^{(L)} : v \in \mathbf{S}_i, u \in \mathbf{S}_j) \quad (7)$$

$$f_e = \text{GumbelSoftmax}\left(p_e\right) \quad (8)$$

$$\tilde{\mathcal{E}}_{inter} = \text{AUG}_{inter}(\mathcal{E} \cup \tilde{\mathcal{E}}, f_e, f_{\mathbf{S}}) \quad (9)$$

where $h_{\mathbf{S}_i}^{(L)}$ is the embedding of subgraph $\mathbf{S}_i$. FUNC is a function (e.g. MLP or a linear layer) that transform the embeddings of edges to the probability distribution p_e. This distribution indicates the likelihood of each edge that are between the subgraph $\mathbf{S}_i$ and $\mathbf{S}_j$ being dropped or retained. $\parallel$ denotes the concatenation operation, f_e is a one-hot vector sampled from this distribution via Gumbel-Softmax, and $\text{AUG}_{inter}(\mathcal{E} \cup \tilde{\mathcal{E}}, f_e, f_{\mathbf{S}})$ is the augmentation function that outputs the augmented edge table $\tilde{\mathcal{E}}_{inter}$. The inter-edge perturbation differs from the intra-edge perturbation in that it takes into account the subgraph embedding when calculating the edge probability distribution, as shown in Eq. (7).

(5) **Subgraph swapping** is to swap the position of the subgraphs by changing the edges between subgraphs. The

process of subgraph swapping can be formulated as:

$$\tilde{\mathcal{E}}_{sub} = \text{AUG}_{sub}(\mathcal{E}, f_{\mathbf{S}}) \tag{10}$$

where $\text{AUG}_{sub}(\mathcal{E}, f_{\mathbf{S}})$ is the augmentation function that outputs the augmented edge table $\tilde{\mathcal{E}}_{sub}$. It believes that most of the semantic meaning of the graph can be preserved in its local structure. The augmentation function $\text{AUG}(\cdot)$ integrates the node attribute (and adjacency matrix) with the f_v (and f_e, the one-hot vector for edges sampled via Gumbel-Softmax) using differentiable operations such as multiplication. Consequently, the gradients of the weights of the subgraph view generator are retained in the augmented node features (edges) and can be computed using back-propagation.

Subgraph View Assembler For a partitioned graph $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_k\}$, we denote the augmented subgraph view of $\mathbf{S}_i$ as $\tilde{\mathbf{S}}_i$=$(\tilde{X}_{\mathbf{S}_i}, \tilde{\mathcal{E}}_{\mathbf{S}_i})$. The augmented graph view $\tilde{G}$=$(\tilde{X}, \tilde{\mathcal{E}})$ can be obtained via $\tilde{X}$=$\text{ASSEMBLE}_x(\tilde{X}_{\mathbf{S}_i})$ and $\tilde{\mathcal{E}}$=$\sum_{i=1}^{k}(\tilde{\mathcal{E}}_{\mathbf{S}_i})$, where ASSEMBLE_x is a matrix computation operation. For the augmented graph, the edge table and node features both participate in the gradient computation, and the parameters of the graph view generator can be updated in a differentiable manner. Therefore, our graph view generator is end-to-end differentiable.

Complexity Analysis

In SOLA-GCL, we use GIN as the graph embedding model, with a complexity of $O(L\cdot(E_{\text{avg}}\cdot C_{\text{agg}}+N_{\text{avg}}\cdot C_{\text{update}}))$, where L is the number of layers, E_{avg} is the average number of edges, N_{avg} is the average number of nodes, C_{agg} is the cost of aggregation, and C_{update} is the cost of updating node features. For the subgraph augmentation selector, the complexity is approximated as $O((N_{\text{avg}}+k_{\text{avg}})\cdot H)$, where H is the feature dimension, and k_{avg} is the average number of subgraphs. This includes an average pooling layer, a linear layer, and a Gumbel-Softmax operation. The subgraph view generator has a complexity of $O(H\cdot(N_{\text{avg}}+E_{\text{avg}}))$, involving a linear layer and a Gumbel-Softmax to generate augmented node features and edge tables. For the subgraph view assembler, the complexity is $O(k_{\text{avg}}\cdot N_{\text{avg}}\cdot H+E_{\text{avg}})$, where assembling node features has a complexity of $O(k_{\text{avg}}\cdot N_{\text{avg}}\cdot H)$ and assembling the edge table is $O(E_{\text{avg}})$, summing up the edges across all subgraphs.

In summary, the overall complexity of the framework is $O(L\cdot(E_{\text{avg}}\cdot C_{\text{agg}}+N_{\text{avg}}\cdot C_{\text{update}})+k_{\text{avg}}\cdot N_{\text{avg}}\cdot H+E_{\text{avg}})$

Graph Contrastive Learning in SOLA-GCL

In this work, we define contrastive loss $\mathcal{L}_{\text{cl}}$, similarity loss $\mathcal{L}_{\text{sim}}$, and classification loss $\mathcal{L}_{\text{cls}}$. The contrastive loss enforces maximizing the consistency between positive pairs z_i, z_j compared with negative pairs. The similarity loss minimizes the mutual information between the views generated by the two view generators. The classification loss is used in the semi-supervised learning task to encourage the graph view generator to generate label-preserving augmentations.

For the contrastive loss, we follow the previous works (Chen et al. 2020; You et al. 2020; Yin et al. 2022) and use the normalized temperature-scaled cross entropy loss (NT-XEnt) (Sohn 2016). The cosine similarity function is defined as $\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)$=$\frac{\boldsymbol{z}_i\cdot\boldsymbol{z}_j}{\|\boldsymbol{z}_i\|_2\cdot\|\boldsymbol{z}_j\|_2}$. During the training process, a data batch of M graphs is randomly sampled and we pass the batch to the two graph view generators to obtain $2M$ graph views. The two augmented views from the same input graph are regarded as the positive view pair. We denote $\ell_{(i,j)}$ as the instance-level contrastive loss between a positive pair of samples (i, j), and the contrastive loss of this data batch $\mathcal{L}_{\text{cl}}$ can be formulated as:

$$\ell_{(i,j)} = -\log\frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1,k\neq i}^{2M}\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)} \tag{11}$$

$$\mathcal{L}_{\text{cl}} = \frac{1}{2M}\sum_{k=1}^{N}[\ell(2k-1, 2k)+\ell(2k, 2k-1)] \tag{12}$$

where τ is the temperature parameter. The final loss is computed across all positive pairs per batch.

For the similarity loss, we simultaneously minimize the mutual information between the augmentation selections generated by the two subgraph augmentation selectors and between the views generated by the two subgraph view generators. During the view generation process, the subgraph augmentation selector outputs a sampled state matrix S indicating the corresponding augmentation operation of each subgraph, and the subgraph view assembler outputs the edge table $\overline{\mathcal{E}}$. As the augmentation process involves edge negative sampling, the length of edge table $\overline{\mathcal{E}}$ is not equal between the two view generators. We transform the edge table $\overline{\mathcal{E}}$ to adjacency matrix A to compute the similarity loss between views. For a graph G, we denote the sampled state matrix of each view generator as S_1, S_2, and the adjacency matrix as A_1, A_2. The similarity loss can be formulated as:

$$\mathcal{L}_{\text{sim}} = \text{sim}(S_1, S_2)+\text{sim}(A_1, A_2) \tag{13}$$

For the classification loss, we employ the cross entropy loss ℓ_{cls}. Give a graph sample G with its corresponding class label y, and its augmented views denoted as $\tilde{G}_1$ and $\tilde{G}_2$, with CLS representing the classifier, the classification loss ℓ_{cls} is expressed as follows:

$$\mathcal{L}_{\text{cls}} = \ell_{cls}(\text{CLS}(G), y)+\ell_{cls}(\text{CLS}(\tilde{G}_1), y)+\ell_{cls}(\text{CLS}(\tilde{G}_2), y) \tag{14}$$

$\mathcal{L}_{\text{cls}}$ is utilized in the pre-training phase of the semi-supervised learning task to encourage the view generator to generate label-preserving augmentations.

Experiments

In this section, we assess the performance of SOLA-GCL on graph classification across various applications by comparing it against state-of-the-art (SOTA) methods in semi-supervised, unsupervised, and transfer learning tasks. We also explore the effectiveness of SOLA-GCL through visualizations of subgraph importance. Additionally, we conduct an ablation study to determine the impact of different components and provide analyses on running time to demonstrate its efficiency.

Model	MUTAG	NCI1	COLLAB	IMDB-B	IMDB-M	Infections	Facebook	DBLP
GL	81.66±2.11	-	-	65.87±0.98	-	-	-	-
WL	80.72±3.00	80.01±0.50	-	72.30±3.44	-	-	-	-
DGK	87.44±2.72	80.31±0.46	-	66.96±0.56	-	-	-	-
node2vec	72.63±10.20	54.89±1.61	54.57±0.37	38.60±2.30	-	-	-	-
sub2vec	61.05±15.80	52.84±1.47	55.26±1.54	55.26±1.54	-	-	-	-
graph2vec	83.15±9.25	73.22±1.81	71.10±0.54	71.10±0.54	-	-	-	-
InfoGraph	89.01±1.13	76.20±1.06	70.65±1.13	73.03±0.87	49.80±1.50	53.50±1.38	52.35±1.12	50.72±1.51
GraphCL	86.80±1.34	77.87±0.41	71.36±1.15	71.14±0.44	49.20±1.62	50.50±1.04	49.05±0.69	48.81±1.10
JOAO	87.35±1.02	78.07±0.47	69.50±0.36	70.21±3.08	-	-	-	-
JOAOv2	87.67±0.79	72.99±0.75	70.40±2.21	71.60±0.86	48.73±1.54	47.00±2.09	48.54±1.01	50.35±1.77
AD-GCL	-	69.67±0.51	73.32±0.61	71.57±1.01	49.87±1.44	52.50±1.08	52.66±1.67	48.21±1.93
AutoGCL	88.64±1.08	82.00±0.29	70.12±0.68	73.30±0.40	48.47±1.88	49.00±1.43	49.83±1.50	49.40±1.74
SimGRACE	89.01±1.31	79.12±0.44	71.72±0.82	71.30±0.77	-	-	-	-
MSSGCL	89.68±0.57	81.45±0.48	73.48±0.83	73.14±0.38	-	-	-	-
Ours	**90.49±2.22**	**82.07±1.32**	**73.80±1.32**	**74.20±1.89**	**51.27±1.05**	**53.50±1.92**	**52.97±1.42**	**53.25±1.90**

Table 1: Comparison with the existing methods for unsupervised learning. **Bold** numbers indicate the best performance, while blue numbers highlight the second best.

Model	BBBP	ToxCast	SIDER	ClinTox	HIV	BACE
No Pretrain	65.8±4.5	63.4±0.6	57.3±1.6	58.0±4.4	75.3±1.9	70.1±5.4
Infomax	68.8±0.8	62.7±0.4	58.4±0.8	69.9±3.0	76.0±0.7	75.9±1.6
EdgePred	67.3±2.4	64.1±0.6	60.4±0.7	64.1±3.7	76.3±1.0	79.9±0.9
AttrMasking	64.3±2.8	64.2±0.5	61.0±0.7	71.8±4.1	77.2±1.1	79.3±1.6
ContextPred	68.0±2.0	63.9±0.6	60.9±0.6	65.9±3.8	77.3±1.0	79.6±1.2
GraphCL	69.68±0.67	62.40±0.57	60.53±0.88	75.99±2.65	78.47±1.22	75.38±1.44
JOAOv2	71.39±0.92	63.16±0.45	60.49±0.74	80.97±1.64	77.51±1.17	75.49±1.27
AD-GCL	70.01±1.07	63.07±0.72	**63.28±0.79**	79.78±3.52	78.28±0.97	78.51±0.80
AutoGCL	73.36±0.77	63.47±0.38	62.51±0.63	80.99±3.38	78.35±0.64	**83.26±1.13**
Ours	**74.08±0.82**	**64.50±0.41**	62.79±0.91	**81.49±1.31**	**78.68±1.01**	82.64±1.01

Table 2: Comparison with the existing methods for transfer learning.

Experimental Settings

We compare SOLA-GCL with the SOTA methods on 16 datasets under unsupervised, semi-supervised and transfer learning settings. The baseline models are as follows: (1) three SOTA kernel-based methods: graphlet kernel (GL) (Shervashidze et al. 2009), Weisfeiler-Lehman subtree kernel (WL) (Shervashidze et al. 2011), and deep graph kernel (DGK) (Yanardag and Vishwanathan 2015); (2) six unsupervised graph-level representation learning methods: node2vec (Grover and Leskovec 2016), sub2vec (Adhikari et al. 2018), graph2vec (Narayanan et al. 2017), EdgePred (Hu et al. 2020), AttrMasking (Hu et al. 2020), and ContextPred (Hu et al. 2020); (3) six classic GCL methods that randomly generate graph views: InfoGraph (Sun et al. 2020), Infomax (Velickovic et al. 2019), GCA (Zhu et al. 2021b), GraphCL (You et al. 2020), JOAO (You et al. 2021), and JOAOv2 (You et al. 2021); (4) four learning-based GCL method: AD-GCL (Suresh et al. 2021), AutoGCL (Yin et al. 2022), SimGRACE (Xia et al. 2022), and MSSGCL (Liu et al. 2023).

Performance Comparison

Unsupervised Representation Learning. As shown in Table 1, several observations can be made: (1) The consistent top performance across diverse datasets suggests that our method is not only versatile but also highly effective in extracting meaningful patterns from complex graph structures, surpassing both traditional graph learning methods and SOTA deep learning approaches. (2) Traditional methods like WL and DGK focus on graph kernels and rely on explicit feature engineering. They lack the adaptability of deep learning models, which can automatically extract features through training. (3) Deep learning methods like AD-GCL and AutoGCL show varying performance across datasets, illustrating the nuanced capabilities and limitations of these approaches. (4) MSSGCL can potentially capture nuanced relationships within the graph data by focusing on subgraph and graph features. In contrast, our approach leverages a wider spectrum of subgraph interactions, not merely focusing on local versus global relationships but also extensively exploring the intricate relationships within and between subgraphs. The superior performance across various datasets demonstrate the effectiveness of our SOLA-GCL.

Transfer Learning. According to results shown in Table 2, three observations can be made: (1) Our model consistently achieves SOTA or second-best performance across all evaluated datasets, demonstrating its robustness and effectiveness under the transfer learning setting. (2) In the SIDER and BACE datasets, our model trails behind AD-GCL and AutoGCL, likely due to their more effective handling of sparse and imbalanced data typical in side effect prediction and biochemical field. (3) AD-GCL and AutoGCL often outperform other models like GraphCL and JOAO due to their advanced strategies in a learnable manner. In contrast, GraphCL and JOAO typically employ more generalized augmentation and optimization strategies, which may not effectively capture the nuanced details of the data during large-scale pre-training.

Semi-Supervised Learning. For semi-supervised learning, we perform the semi-supervised graph classification exper-

Model	PROTEINS	DD	NCI1	COLLAB
Full Data	78.25±1.61	80.73±3.78	83.65±1.16	83.44±0.77
10% Data	69.72±6.71	74.36±5.86	75.16±2.07	74.34±2.00
10% GAE	70.51±0.17	74.54±0.68	74.36±0.24	75.09±0.19
10% Infomax	72.27±0.40	75.78±0.34	74.86±0.26	73.76±0.29
10% ContextPred	70.23±0.63	74.66±0.51	73.00±0.30	73.69±0.37
10% GCA	73.85±5.56	76.74±4.09	68.73±2.36	74.32±2.30
10% GraphCL	74.21±4.50	76.65±5.12	73.16±2.90	75.50±2.15
10% JOAO	72.13±0.92	75.69±0.67	74.48±0.27	75.30±0.32
10% JOAOv2	73.31±0.48	75.81±0.73	74.86±0.39	75.53±0.18
10% AD-GCL	73.96±0.47	77.91±0.73	**75.18±0.31**	75.82±0.26
10% AutoGCL	75.65±2.40	77.50±4.41	73.75±2.25	**77.16±1.48**
10% SimGRACE	74.03±0.51	76.48±0.52	74.60±0.41	74.74±0.28
10% MSSGCL	75.76±0.52	78.89±0.18	74.77±0.31	76.02±0.13
10% **Ours**	**77.63±3.10**	**79.43±4.31**	74.94±1.96	76.97±1.31

Table 3: Comparison with existing methods and different strategies for semi-supervised learning.

	MUTAG	NCI1	COLLAB	IMDB-B
Louvain	90.49±2.22	82.07±1.32	73.80±1.32	74.20±1.89
GN	89.88±2.98	81.56±2.11	72.32±1.88	74.10±2.84
SOLA-GCL-P	88.33±2.04	79.93±1.74	70.92±1.91	71.70±2.80
SOLA-GCL-S	88.80±3.46	81.58±1.57	71.86±1.63	74.00±2.63
SOLA-GCL-R	88.80±3.24	80.95±1.75	69.38±1.96	73.70±2.84

Table 4: Ablation study on the critical modules.

iments on TUDataset (Morris et al. 2020). As shown in Table 3, two observations can be made: (1) Our SOLA-GCL consistently achieves SOTA or comparative performance compared to other models due to its effective use of subgraph-specific information, enhancing learning from limited labeled data across various datasets. (2) MSSGCL ranks second on PROTEINS and DD because of its effective multi-scale subgraph sampling strategy. However, it falls short of SOLA-GCL which better captures and utilizes subgraph interactions.

Visualization

To validate the effectiveness of our SOLA-GCL framework (Fig.2), we trained a view generator on the MUTAG dataset(Debnath et al. 1991; Kriege and Mutzel 2012), a real-world chemical dataset. We evaluate SOLA-GCL's ability to identify important subgraphs, such as carbon rings and NO_2 groups, known to be mutagenic (Debnath et al. 1991). The visualization results explain the capability of SOLA-GCL as it has successfully identified the carbon rings and NO_2 groups as critical subgraphs. The results emphasize the importance of capturing subgraph features and relationships among subgraphs, and prove the effectiveness of our SOLA-GCL.

Ablation Study

The proposed method first partitions the graph into a set of subgraphs, and then jointly trains the subgraph augmentation selector and the subgraph view generator to generate graph views. In this subsection, we present an ablation study on the unsupervised learning task to evaluate the contribution of the three components as shown in Table 4. We use "P" as a suffix of the model name (SOLA-GCL-P) to denote the model without a subgraph partition module, a suffix "S" (SOLA-GCL-S) to denote the model with a random selector for subgraph augmentation strategy rather than a learnable one, and a suffix "R" (SOLA-GCL-R) to denote the model with a random

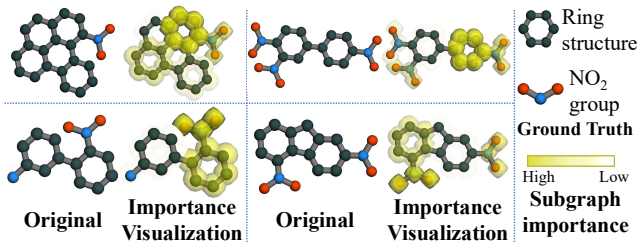

Figure 2: We visualize the graphs in the MUTAG dataset, the deeper color indicates the more important subgraphs and the ground truth is ring structure and NO_2 group.

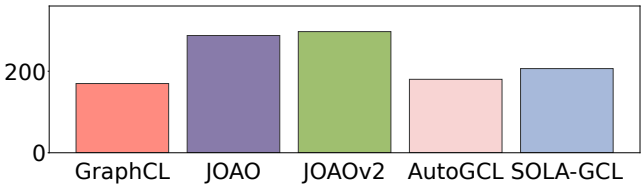

Figure 3: Comparisons in terms of real running time. Each model is trained for 100 epochs on each dataset and the average training time of one epoch is reported.

subgraph view generator (e.g., random node dropping in one subgraph) rather than a learnable one. For variations of the subgraph partitioning algorithm, we evaluate SOLA-GCL using the Louvain algorithm and Girvan-Newman (GN) algorithm respectively, and provide the average subgraph number of a partitioned graph in each dataset. The results indicate that the SOLA-GCL framework performs well across different subgraph partitioning algorithms, highlighting its effectiveness and robustness. Furthermore, the extensive experiments across several datasets demonstrate the robustness and adaptability of SOLA-GCL, indicating its performance is resilient to variations in the partitioned subgraph sizes.

Efficiency Analysis

As shown in Fig. 3, we present the comparisons on training time consumption for each epoch in the training process. While GraphCL and AutoGCL show the lowest training time consumption, SOLA-GCL strikes an optimal balance between efficiency and effectiveness. It achieves better performance than JOAO and JOAOv2 and consumes less training time, demonstrating a significant advantage in both performance and efficiency.

Conclusion

We present SOLA-GCL, a novel data augmentation method for graph contrastive learning. In contrast to existing methods that overlook the importance of the critical role of subgraph structures, SOLA-GCL takes a comprehensive approach by considering the information of intra-subgraph characteristics and inter-subgraph relationships. This is achieved through the joint training of a subgraph augmentation selector and a subgraph view generator, enabling the generation of learnable graph views. It consistently outperforms existing methods across various datasets, highlighting its effectiveness.

References

Adhikari, B.; Zhang, Y.; Ramakrishnan, N.; and Prakash, B. A. 2018. Sub2Vec: Feature Learning for Subgraphs. In *PAKDD*.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10): P10008.

Cai, L.; Mao, X.; Xiao, Y.; Wu, C.; and Lan, M. 2023. An effective and efficient time-aware entity alignment framework via Two-aspect three-view label propagation. In *IJCAI*, 5021–5029.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, 1597–1607.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2): 786–797.

Ding, K.; Xu, Z.; Tong, H.; and Liu, H. 2022. Data Augmentation for Deep Graph Learning: A Survey. *ACM SIGKDD Explorations Newsletter*, 24(2): 61–77.

Girvan, M.; and Newman, M. E. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12): 7821–7826.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*, 855–864.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, volume 30, 1024–1034. Long Beach, CA, USA: Curran Associates, Inc.

Hassani, K. 2022. Cross-Domain Few-Shot Graph Classification. In *AAAI*, 6856–6864.

Hassani, K.; and Ahmadi, A. H. K. 2022. Learning Graph Augmentations to Learn Graph Representations. *CoRR*, abs/2201.09830.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; and Leskovec, J. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.

Jiao, Y.; Xiong, Y.; Zhang, J.; Zhang, Y.; Zhang, T.; and Zhu, Y. 2020. Sub-graph Contrast for Scalable Self-Supervised Graph Representation Learning. In *ICDM*, 222–231. IEEE.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 1–14. Toulon, France.

Kriege, N. M.; and Mutzel, P. 2012. Subgraph Matching Kernels for Attributed Graphs. In *ICML*.

Landrum, G. 2013. Rdkit documentation. *Release*, 1(1-79): 4.

Li, X.; Wu, X.; Kong, L.; Zhang, X.; Huang, S.; Dou, D.; and Xiong, H. 2023a. ContRE: A Complementary Measure for Robustness Evaluation of Deep Networks via Contrastive Examples. In *ICDM*, 309–318. IEEE.

Li, Y.; Xiong, H.; Kong, L.; Bian, J.; Wang, S.; Chen, G.; and Yin, D. 2024. GS^{2}P: a generative pre-trained learning to rank model with over-parameterization for web-scale search. *Mach. Learn.*, 113(8): 5331–5349.

Li, Y.; Xiong, H.; Kong, L.; Sun, Z.; Chen, H.; Wang, S.; and Yin, D. 2023b. MPGraf: a Modular and Pre-trained Graphformer for Learning to Rank at Web-scale. In *ICDM*, 339–348.

Li, Y.; Xiong, H.; Kong, L.; Wang, Q.; Wang, S.; Chen, G.; and Yin, D. 2023c. S^{2}phere: Semi-Supervised Pre-training for Web Search over Heterogeneous Learning to Rank Data. In *SIGKDD*, 4437–4448.

Li, Y.; Xiong, H.; Kong, L.; Wang, S.; Sun, Z.; Chen, H.; Chen, G.; and Yin, D. 2023d. LtrGCN: Large-Scale Graph Convolutional Networks-Based Learning to Rank for Web Search. In *ECML PKDD 2023,*, 635–651.

Li, Y.; Xiong, H.; Kong, L.; Zhang, R.; Dou, D.; and Chen, G. 2022. Meta hierarchical reinforced learning to rank for recommendation: a comprehensive study in moocs. In *ECML PKDD*, 302–317. Springer.

Li, Y.; Xiong, H.; Kong, L.; Zhang, R.; Xu, F.; Chen, G.; and Li, M. 2023e. MHRR: MOOCs Recommender Service With Meta Hierarchical Reinforced Ranking. *IEEE TSC*, 16(6): 4467–4480.

Li, Y.; Xiong, H.; Wang, Q.; Kong, L.; Liu, H.; Li, H.; Bian, J.; Wang, S.; Chen, G.; Dou, D.; and Yin, D. 2023f. COLTR: Semi-Supervised Learning to Rank With Co-Training and Over-Parameterization for Web Search. *IEEE TKDE*, 35(12): 12542–12555.

Li, Y.; Xiong, H.; Zhang, Y.; Bian, J.; Peng, T.; Li, X.; Wang, S.; Kong, L.; and Yin, D. 2025. RankElectra: Semi-supervised Pre-training of Learning-to-Rank Electra for Web-scale Search. In *SIGKDD*.

Lin, B.; Li, Y.; Gui, N.; Xu, Z.; and Yu, Z. 2023. Multi-View Graph Representation Learning Beyond Homophily. *ACM TKDD*.

Liu, Y.; Zhao, Y.; Wang, X.; Geng, L.; and Xiao, Z. 2023. Multi-scale subgraph contrastive learning. In *IJCAI*, 2215–2223.

Lyu, Z.; Li, Y.; Zhu, G.; Xu, J.; Poor, H. V.; and Cui, S. 2024. Rethinking Resource Management in Edge Learning: A Joint Pre-training and Fine-tuning Design Paradigm. *arXiv preprint arXiv:2404.00836*.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Pei, Y.; Huang, T.; van Ipenburg, W.; and Pechenizkiy, M. 2022. ResGCN: attention-based deep residual modeling for anomaly detection on attributed networks. *Mach. Learn.*, 111(2): 519–541.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9).

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, 488–495.

Sohn, K. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *NeurIPS*, 1849–1857.

Sun, F.; Hoffmann, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. In *NeurIPS*, 15920–15933.

Tang, H.; Liang, X.; Guo, Y.; Zheng, X.; Wu, B.; Zhang, S.; and Li, Z. 2023. Diffuse and Smooth: Beyond Truncated Receptive Field for Scalable and Adaptive Graph Representation Learning. *ACM TKDD*, 17(5): 1–25.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*, 1–12. Vancouver, BC, Canada.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR*.

Wang, X.; Li, J.; Yang, L.; and Mi, H. 2021. Unsupervised learning for community detection in attributed networks based on graph convolutional network. *Neurocomputing*, 456: 147–155.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE TNNLS*, 32(1): 4–24.

Xia, J.; Wu, L.; Chen, J.; Hu, B.; and Li, S. Z. 2022. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *WWW*, 1070–1079.

Xiong, H.; Bian, J.; Li, Y.; Li, X.; Du, M.; Wang, S.; Yin, D.; and Helal, S. 2024. When search engine services meet large language models: visions and challenges. *IEEE Transactions on Services Computing*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Yanardag, P.; and Vishwanathan, S. 2015. Deep graph kernels. In *SIGKDD*, 1365–1374.

Yin, Y.; Wang, Q.; Huang, S.; Xiong, H.; and Zhang, X. 2022. AutoGCL: Automated Graph Contrastive Learning via Learnable View Generators. In *AAAI*, 8892–8900.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph Contrastive Learning Automated. In *ICML*.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. In *NeurIPS*.

Yuan, H.; Cong, G.; and Li, G. 2024. Nuhuo: An Effective Estimation Model for Traffic Speed Histogram Imputation on A Road Network. *VLDB*, 17(7): 1605–1617.

Yuan, H.; and Li, G. 2021. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6(1): 63–85.

Yuan, H.; Li, G.; and Bao, Z. 2022. Route travel time estimation on a road network revisited: Heterogeneity, proximity, periodicity and dynamicity. *VLDB*, 16(3): 393–405.

Yuan, H.; Li, G.; Bao, Z.; and Feng, L. 2021. An effective joint prediction model for travel demands and traffic flows. In *ICDE*, 348–359.

Yuan, H.; Wang, S.; Bao, Z.; and Wang, S. 2023. Automatic road extraction with multi-source data revisited: completeness, smoothness and discrimination. *VLDB*, 16(11): 3004–3017.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph Transformer Networks. In *NeurIPS*, 11960–11970.

Zhang, X.; Tan, Q.; Huang, X.; and Li, B. 2022. Graph Contrastive Learning with Personalized Augmentation. *CoRR*, abs/2209.06560.

Zhang, Y.; Yao, Q.; Yue, L.; Wu, X.; Zhang, Z.; Lin, Z.; and Zheng, Y. 2023a. Emerging drug interaction prediction enabled by a flow-based graph neural network with biomedical network. *Nature Computational Science*, 3(12): 1023–1033.

Zhang, Y.; Zhou, Z.; Yao, Q.; Chu, X.; and Han, B. 2023b. Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning. In *SIGKDD*, 3446–3457.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81.

Zhu, Y.; Xu, Y.; Liu, Q.; and Wu, S. 2021a. An Empirical Study of Graph Contrastive Learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks*.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021b. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*, 2069–2080.