E^2GCL: Efficient and Expressive Contrastive Learning on Graph Neural Networks

Haoyang LI[1], Shimin DI[1*], Lei CHEN[1,2], Xiaofang ZHOU[1]
[1]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[2]The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
hlicg@cse.ust.hk, sdiaa@connect.ust.hk, leichen@hkust-gz.edu.cn, zxf@cse.ust.hk

2024 IEEE 40th International Conference on Data Engineering (ICDE) | 979-8-3503-1715-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDE60146.2024.00071

***Abstract*—Recently, graph contrastive learning proposes to learn node representations from the unlabeled graph to alleviate the heavy reliance on node labels in graph neural networks (GNNs). The core idea is to generate diverse positive views and negative views according to local subgraphs. Then, GNNs take these views as supervised signals and train the model by maximizing the similarity between positive view pairs of each node and minimizing the similarity between positive and negative views. Regardless of the fruitful progress, existing graph contrastive learning approaches still suffer from *low-efficiency*, *insufficient-expressivity*, and *unpreserved-locality* issues. First, they train GNNs by all nodes, reducing the efficiency due to similar and redundant nodes. Second, they only use limited operations (e.g., edge deletion and feature masking) to generate positive views, thereby restricting their expressivity. Third, they uniformly delete edges and mask node features and may modify important edges and features, thereby damaging the important locality information of nodes. In this paper, we propose an efficient and expressive contrastive learning framework for GNNs, namely E^2GCL. Specifically, given a limited node budget, we select a set of representative nodes instead of all nodes to accelerate the GNNs training. Besides, we use three general operations (edge deletion, edge addition, and feature perturbation) to generate expressive and locality-preserved positive views based on edge and feature importance. Extensive experiments on various real-world datasets demonstrate the superior effectiveness and efficiency of our proposed E^2GCL.**

***Index Terms*—Graph Neural Network, Contrastive Learning**

I. Introduction

Graph neural networks (GNNs) [1], [2], [3], [4], [5], [6], [7], which propose to learn node representations by aggregating information from neighbors, have achieved great success on various tasks, such as node classification [8], [9], link prediction [10], [11], graph isomorphism [12], [13], subgraph counting [14], [15], and community search [16], [17]. Nevertheless, existing GNNs are typically built in a semi-supervised manner, which access limited task-specific labeled nodes for training [18], [19]. However, GNNs trained on limited labels may not generalize well on all nodes [20], [21], [22], [23], [24]. But it is expensive and time-consuming to annotate a large number of labeled nodes [25], [26], [27]. To alleviate the scarce label data issue, recent GNN works integrate contrastive learning [28], [29] to learn node representations from unlabeled graph data. These node representations can distinguish node similarities, which can be easily transferred to downstream tasks [21], [30].

*Corresponding Author

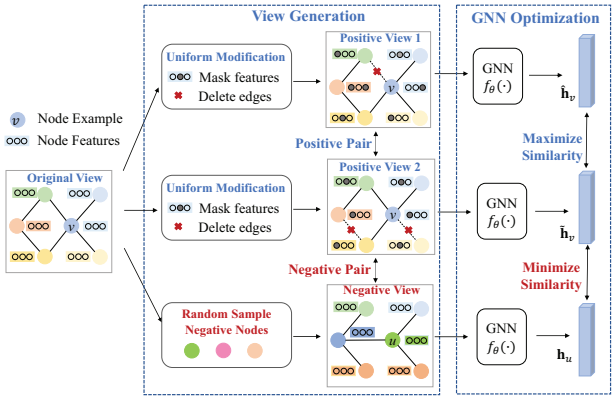


Fig. 1: An example of perturbation-based graph contrastive learning. Taking node v as an example, $\hat{\mathbf{h}}_v$ and $\tilde{\mathbf{h}}_v$ are representations of node v learned from two positive views, $\hat{\mathbf{h}}_u$ is the representation learned from the negative view.

Generally, contrastive learning approaches generate supervised signals from the unlabeled graph and then use these supervised signals to optimize GNNs. Specifically, they generate diverse positive views (i.e., different local subgraphs) for each node v as supervised signals. These positive views are expected to preserve the important locality information (i.e., local edges and node features) of node v in the original graph. Besides, these contrastive learning approaches randomly sample nodes from the graph as the negative views for each node v, expecting that these sampled nodes have different node locality information with the node v. Then, as illustrated in Fig. 1, the parameters of GNNs are optimized by maximizing the representation similarity between positive view pairs of each node, and minimizing the representation similarity between positive and negative views. In such a way, the pre-trained GNNs can distinguish the similarity between nodes.

Based on the techniques that are employed to generate positive views for each node, current contrastive learning approaches can be classified into three types, i.e., *similarity-based* [31], [32], [33], *diffusion-based* [34], [35], and *perturbation-based* [36], [20], [25], [37], [24], [38] approaches. Based on node features and structures, similarity-based approaches [31], [32], [33] first select nodes in the original graph that are similar to the target node. Then, they take the local subgraphs of these similar nodes as the

2375-026X/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDE60146.2024.00071

positive views for the target node. Instead of selecting the local subgraphs of existing nodes, diffusion-based and perturbation-based approaches generate positive views for the target node by reconstructing its local subgraph. More specifically, diffusion-based approaches [34], [35] modify graph structure (edge deletion/addition) based on the global graph topological information, such as the personalized PageRank (PPR) [39]. Perturbation-based approaches [36], [20], [25], [37], [24], [38] make modifications on both structures and features.

Despite the fruitiness of contrastive learning on GNNs, there still exist three limitations of existing approaches [34], [20], [25], [24], [38], i.e., *low-efficiency*, *insufficient-expressivity*, and *unpreserved-locality* issues. First, existing contrastive learning approaches [34], [20], [25], [24], [38] generate positive views for all nodes and use all nodes to optimize GNN parameters. However, since there exist similar nodes in the graph, it is redundant to use all nodes for training.

Second, there are various graph augmentation operations that can be utilized to generate positive or negative views [22], [24], such as edge deletion/addition, node dropping/addition, feature perturbation/dropping/masking, etc. Intuitively, more augmentation operations can generate more expressive positive views. However, as shown in Tab. I, current approaches [34], [20], [25], [38], [40] only use part of operations for simplicity purposes, which will hinder the expressiveness of contrastive learning approaches and thereby decreases their effectiveness.

Third, when generating positive views for a node, existing approaches take each edge and each feature dimension equally, and then modify them uniformly [34], [20], [24], [38]. However, the impact of different edges and features should be different. Consequently, existing approaches may remove important edges and node features, and thus deteriorate the intrinsic locality patterns of the node, thereby damaging the quality of the learned node representations [25], [41]. As shown in Tab. I, we summarize existing works from three perspectives: the nodes for training, the expressivity of generated views, and the locality to be persevered. None of existing approaches can cover all these aspects.

To address the above *low-efficiency*, *insufficient-expressivity*, and *unpreserved-locality* issues, a contrastive learning framework is expected to include the following three parts: (1) *a node selector* chooses a subset of the most informative nodes that can fully represent the entire graph. Such a way can avoid generating views for all nodes to train GNNs, thereby alleviating the *low-efficiency* issue. (2) *a view generator* uses more augmentation operations, such as all possible operations, to generate positive views, addressing the *insufficient-expressivity* issue. (3) to address the *unpreserved-locality* issue, the *view generator* targets to modify unimportant edges and perturb unimportant node features to generate positive views for each selected node. Then, the generated positive views can maintain the intrinsic local pattern of each node and preserve the important locality information. Additionally, each pair of positive views should be diverse, i.e., they have different unimportant edges and features. Such a way enables the pre-trained GNN to be invariant to noise [40], [42]. However, to achieve this framework, we need to address the following technique challenges.

TABLE I: The summary of existing contrastive learning approaches for GNNs. FP: Feature Perturbation, FM: Feature Mask, ED: Edge Deletion, EA: Edge Addition, Exp.: Expressivity, LP: Locality-preserved.

Type	Model	Optimization		Positive Views		
		Train Nodes	Efficiency	Operation	Exp.	LP
Similarity	AFGCL [33]	All	×	-	×	×
	AFGRL [31]	All	×	-	×	×
Diffusion	MVGRL [34]	All	×	EA, ED	×	✓
	MVGCC [35]	All	×	EA, ED	×	✓
Perturbation	GCC [38]	All	×	ED	×	×
	GraphCL [24]	All	×	EM, ED	×	×
	GRACE [20]	All	×	FM, ED	×	×
	BGRL [36]	All	×	FM, ED	×	×
	ADGCL [37]	All	×	ED	×	×
	GCA [25]	All	×	FM,ED	×	✓
	E^2GCL(Ours)	Subset	✓	FP, ED, EA	✓	✓

- There lacks a metric for measuring the representativity of a subset of nodes towards the other nodes under the contrastive learning manner, i.e., without labels. Moreover, given a node budget, the representative node subset should be selected efficiently regarding the exponential node combinations. Otherwise, the saved time from training GNNs on fewer number of nodes will be totally or even over-consumed by the node selection process.
- Even though more augmentation operations tend to generate more expressive positive views, the model complexity will increase as the number of used operations increases [24], [22]. Therefore, we should select a minimal subset of operations from all operations that can generate the same expressive positive views as all augmentation operations, thereby reducing model complexity.
- Due to the lack of node labels, it is difficult to measure the edge and node feature importance to further preserve the important locality information. Besides, given a set of augmentation operations, the space of positive views are exponential. Therefore, we need an efficient algorithm to generate diverse and locality-preserved positive views.

To address the aforementioned technique challenges, we propose an Efficient and Expressive Contrastive Learning framework for GNNs, namely E^2GCL, which incorporates edge and feature importance to generate locality-preserved positive views. First, under the contrastive learning manner, we first theoretically analyze the representability of nodes under the contrastive learning setting, i.e., we cannot access node labels when pre-training GNNs. Then, we formulate a cluster-based coreset selection problem to select top-k representative nodes, and prove that this problem is NP-hard and propose an efficient sampling-based algorithm with an approximation guarantee to solve it. In view generator, we select three augmentation operations (edge deletion, edge addition, and feature perturbation) and theoretically show that these three operations are the minimal set of operations required to generate the same expressive views as all graph augmentation operations. Third, we formulate a view generation problem that aims to generate diverse and locality-preserved views for each node by the three afore selected operations. We prove

that this problem is NP-hard as well and propose an edge and feature-aware sampling algorithm. Specifically, we measure the edge and feature importance from node centrality and node similarity, and then sample important edges and features to generate positive views. Such a way enables us to generate locality-preserved and diverse positive views. Overall, our contributions are summarized as follows.

- We propose an efficient and expressive contrastive learning framework for GNNs, namely E^2GCL. Generally, it consists of two components, i.e., representative node selector and locality-preserved view generator.
- In node selector, we first theoretically analyze the representability of nodes under the contrastive learning setting, and then formulate a cluster-based coreset selection problem to select top-k representative nodes. Then, we prove that this problem is NP-hard and propose a sampling-based greedy algorithm with an approximation guarantee.
- In view generator, we formulate a view generation problem that generates expressive, diverse, and locality-preserved positive views. Then, we prove that this problem is another NP-hard and propose an efficient sampling algorithm.
- We conduct experiments on various real-world datasets. Extensive empirical studies demonstrate the superior performance and efficiency of E^2GCL compared with baselines.

II. Preliminary and Related Works

In this section, we introduce graph neural networks (GNNs) and contrastive learning on GNNs. Formally, $G(V, \mathbf{A}, \mathbf{X})$ denotes a graph, where V, $\mathbf{A} \in \{0,1\}^{|V|\times|V|}$, $\mathbf{X} \in \{0,1\}^{|V|\times d_x}$, denote nodes, adjacency matrix, and node features, respectively. $N_v = \{u : \mathbf{A}[u][v] = 1\}$ is the neighbors of v.

A. Graph Neural Network

Current researchers [43], [44], [45], [46], [47], [48], [49] propose to learn a low dimensional representations for nodes to facilitate the downstream task. One representative approach is GNNs. Specifically, given graph structure $\mathbf{A}$ and node features $\mathbf{X}$ of a graph G as inputs, the encoder GNNs f_θ [1], [17], [50], [51], [52], [53], [54], [55] propose to learn low dimensional representation $\mathbf{H} \in \mathbb{R}^{|V|\times d_h}$ for nodes V by recursively aggregating information from their neighbors. Taking the representative Graph Convolutional Network (GCN) [18] as an example, the *l+1-th* node representations $\mathbf{H}^{l+1} \in \mathbb{R}^{|V|\times d_{l+1}}$ can be computed as:

$$\mathbf{H}^{l+1} = \sigma(\mathbf{A}_n \mathbf{H}^l \mathbf{W}^l), \tag{1}$$

where σ denotes a non-linear function (e.g., ReLU [56]), $\mathbf{H}^l \in \mathbb{R}^{|V|\times d_l}$ and $\mathbf{W}^l \in \mathbb{R}^{d_l \times d_{l+1}}$ denotes node representations and parameters in the *l-th* layer, respectively. $\mathbf{A}_n$ is a normalized adjacency matrix [18]. Initially, $\mathbf{H}^0 = \mathbf{X}$. $\theta = \{\mathbf{W}^0, \cdots, \mathbf{W}^L\}$ denotes the parameters of GNN. For simplicity, given a GNN model f_θ and a graph G, we use $\mathbf{H} = f_\theta(G)$ to denote the node representations of V learned on G. Similarly, $\mathbf{h}_v = f_\theta(G_v)$ denotes v's representation learned from its L-hop local subgraph $G_v(V_v, \mathbf{A}_v, \mathbf{X}_v)$.

Then, the node representation $\mathbf{H}$ can be used for various downstream tasks, such as node classification, link prediction, and graph classification. Specifically, a simple decoder (e.g., l_2 linear regression) q_φ takes $\mathbf{H}$ as inputs to prediction node class, link probability, and graph class. In general, given labeled training data $TD_{task} = \{G_{task}, LD_{task}, \mathbf{Y}_{task}\}$, the GNN f_θ and the decoder q_φ can be optimized by minimizing task training loss $l_{task}(\cdot)$ as follows.

$$\varphi^*, \theta^* = \arg\min_{\varphi,\theta} l_{task}(q_\varphi, f_\theta, TD_{task}) \tag{2}$$

We introduce how to apply Eq. (2) to downstream tasks.

- *Node classification:* TD_{task} is $\{G, V_{la}, \mathbf{Y}_N\}$, where $V_{la} \subset V$ with labels $\mathbf{Y}_N \in \{0,1\}^{|V_{la}|\times|\mathcal{Y}|}$ denotes labeled training nodes. q_φ takes $\mathbf{H}$ to predict label score $\mathbf{S}$ for nodes V, i.e., $\mathbf{S} = q_\varphi(\mathbf{H}) \in [0,1]^{|V|\times|\mathcal{Y}|}$. The task loss [18], [57] can be defined as $l_{nc}(q_\varphi, f_\theta, TD_{task}) = -\sum_{v\in V_{la}} \ln \mathbf{S}[v][y_v]$.
- *Link prediction:* TD_{task} is $\{G, E, \mathbf{Y}_E\}$, where E contains node pairs and $\mathbf{Y}_E \in \{0,1\}^{|E|}$ denotes whether there is an edge between $(v,u) \in E$. The decoder $q_\varphi(\cdot)$ can predict the probability that there is an edge between v and u based on $\mathbf{h}_v$ and $\mathbf{h}_u$, i.e., $p_{v,u} = q_\varphi([\mathbf{h}_v, \mathbf{h}_u]) \in [0,1]$. The task loss [58], [59] can be defined as $l_{lp}(q_\varphi, f_\theta, TD_{task}) = -\sum_{(v,u)\in E} (\ln p_{v,u} + \sum_{u'\in Neg_v} \ln(1 - p_{v,u'}))$, where Neg_v is negative samples of v.
- *Graph classification:* TD_{task} is $\{\{G_i\}_{i=1}^n, \{G_i\}_{i=1}^n, \mathbf{Y}_G\}$, where a set of graph $\{G_i\}_{i=1}^n$ are labeled by $\mathbf{Y}_G \in \{0,1\}^{n\times|\mathcal{Y}|}$. The representation $\mathbf{z}_i$ of entire graph G_i can be summarized on the node representation $\mathbf{H}_i$ by a READOUT function [34], i.e., $\mathbf{z}_i = \mathsf{READOUT}(\mathbf{H}_i)$. One example of READOUT is SUM function, i.e., $\mathbf{z}_i = \sum_{v\in V_i} \mathbf{H}_i[v]$. Then, the decoder $q_\varphi(\cdot)$ can predict class score of each graph G_i based $\mathbf{z}_i$, i.e., $\mathbf{s}_i = q_\varphi(\mathbf{z}_i) \in [0,1]^{|\mathcal{Y}|}$. The task loss [34], [60] can be defined as $l_{gc}(q_\varphi, f_\theta, TD_{task}) = -\sum_{i=1}^n \ln \mathbf{s}_i[y_i]$.

Also, researchers propose variants of GCN to improve its efficiency [61], [62], [63], [64], [65], [66] and effectiveness [19], [67], [68], [69], [70] by sampling its neighbors, learning weight for edges, and automatically searching architecture.

TABLE II: Summary on important notations.

Symbols	Meanings
$G(V, \mathbf{A}, \mathbf{X})$	The graph G
$\mathbf{A}_n$	The normalized adjacency matrix of $\mathbf{A}$
N_v^l, N_v	The l-hop and 1-hop neighbors of node v
D_v, D	Degree of node v, the average node degree
$\mathbf{h}_v, \mathbf{H}$	The representation of v and all nodes
$\mathbf{R}, \mathbf{r}_v$	Raw aggregated information from neighbors
l_{nc}, l_{cl}	Node classification loss, and contrastive loss
f_θ, q_φ	The encoder GNN f_θ and simple decoder q_φ
k	Node budget for selecting a coreset
C_i,$C_{V_s,i}$	Cluster i, nodes in V_s belonging to cluster i
n_c, n_s	Cluster number and sampled node number
$RS(V_s)$	Representativity score of selected nodes V_s
$\varphi_c(v)$	Node centrality of node v
$w^e_{v,u}$	The edge score between node v and u
$w^f_{\mathbf{x}_v[i]}$	The i-th dimension feature score of v

B. Graph Contrastive Learning

In real-world applications, it is hard to annotate enough labels $\mathbf{Y}_{task}$ in Eq. (2) to train a generalizable GNN model. Recent GNN works integrate contrastive learning [21], [28], [29], [71], [72], [73] to learn high-quality node representations from unlabeled graph data, which can be easily transferred to downstream tasks [21], [30], [74]. Alg. 1 illustrates the framework of contrastive learning.

1) Pre-train Node Representations with Contrastive Learning: Generally, graph contrastive learning (GCL) approaches generate different local subgraphs (e.g., $\hat{G}_v$ and $\tilde{G}_v$) as the positive views for each node v. These positive views are expected to preserve the important locality information of the node v in the origin local graph G_v. In another word, the representation $\hat{\mathbf{h}}_v$ (resp. $\tilde{\mathbf{h}}_v$) learned from $\hat{G}_v$ (resp. $\tilde{G}_v$) is similar to $\mathbf{h}_v$ learned from G_v. Similarly, existing approaches sample a set of nodes Neg_v from the graph as negative views for each node v, assuming that these sampled nodes are dissimilar to v. Then, the GNN f_θ are optimized by the contrastive loss $l_{cl}(\cdot)$ in Eq. (3), which maximizes the similarity between each positive view pair ($\hat{G}_v$ and $\tilde{G}_v$), and minimizes the similarity between positive views ($\hat{G}_v$ and $\tilde{G}_v$) and negative views (Neg_v):

$$\theta^* = \arg\min_{\theta} \frac{1}{|V|} \sum_{v \in V} l_{cl}(f_\theta, \hat{G}_v, \tilde{G}_v, Neg_v) \quad (3)$$

$$s.t.\ \tilde{G}_v, \hat{G}_v = \arg \min_{\tilde{G}_v, \hat{G}_v} l_{vg}(G, v, \mathcal{T}), \forall v \in V, \quad (4)$$

where $l_{cl}(\cdot)$ is the constrastive loss, $l_{vg}(\cdot)$ is the view generation loss on positive view pair, $\mathcal{T}$ is a set of augmentation operations to generate views. One typical $l_{cl}(\cdot)$ can be defined based on euclidean distance as follows:

$$\|\hat{\mathbf{h}}_v - \tilde{\mathbf{h}}_v\|_2^2 - \frac{1}{2|Neg_v|} \sum_{\mathbf{h}'_v \in \{\tilde{\mathbf{h}}_v, \hat{\mathbf{h}}_v\}} \sum_{u \in Neg_v} \|\mathbf{h}'_v - \mathbf{h}_u\|_2^2. \quad (5)$$

Therefore, minimizing $l_{cl}(\cdot)$ is to maximize the similarity between positive views ($\tilde{\mathbf{h}}_v, \hat{\mathbf{h}}_v$) and minimize the similarity between positive views with negative views ($\mathbf{h}_u \in Neg_v$). The contrastive learning process is shown in line 1-5 in Alg. 1.

Existing contrastive learning approaches mainly differ in positive view generation in Eq. (4), especially their customized $l_{vg}(\cdot)$ and $\mathcal{T}$. Based on their technique, current contrastive learning approaches can be classified into three types. First, similarity-based approaches [31], [27], [32], [33], [41], [76] do not use graph augmentation operations, i.e., $\mathcal{T} = \emptyset$. Several researchers [31], [32], [33], [76] select each neighbor $u \in N_v$ of node v or the node $u \in V$ whose features are similar to v and take the local subgraph G_u as node v's positive view. Also, recent researchers [27], [41] propose to add small noise ϵ on the learned node representation $\mathbf{h}_v$ as the learned positive representations $\hat{\mathbf{h}}_v$, i.e., $\hat{\mathbf{h}}_v = \mathbf{h}_v + \epsilon$. Second, based on PPR technique [39], [77], [78] and Gaussian kernel [79], diffusion-based approaches [34], [35] take edge addition and deletion to modify the local subgraph G_v to generate positive views $\hat{G}_v$ and $\tilde{G}_v$, respectively. In addition to structure modification, perturbation-based approaches [36], [20], [25], [37], [24], [38] uniformly mask node features in local graph G_v. In particular, recent researchers [72], [80] propose to use various pretext task, such as predicting the shortest path between nodes, to learn general node representations.

Algorithm 1: GCL framework with downstream task

Input: Graph $TD_{task} = \{G_{task}, LD_{task}, \mathbf{Y}_{task}\}$, the encoder GNN f_θ, the decoder q_φ, epoch number T
Output: The encoder f_{θ^*} and the decoder q_{φ^*}

```
// GCL without labels
for i = 1 to T do
    for each G(V, A, X) ∈ G_task do
        for v ∈ V do
            G̃_v, Ĝ_v = arg min_{G̃_v, Ĝ_v} l_vg(G, v, T)
        θ* = arg min_θ (1/|V|) Σ_{v∈V} l_cl(f_θ, Ĝ_v, G̃_v, Neg_v)
// Decoder training with labels
φ* = arg min_φ l_task(q_φ, f_θ*, TD_task)
Return the trained encoder f_θ* and the decoder q_φ*
```

2) Fine-tune Node Representations on Downstream Tasks: Then, the GNN f_{θ^*} pre-trained in the contrastive manner can be used in node classification, link prediction and graph classification tasks. As shown in line 6 in Alg. 1 and Sec. II-A, given limited labeled training data $TD_{task} = \{G_{task}, D_{task}, \mathbf{Y}_{task}\}$ and the fixed GNN f_{θ^*}, the decoder q_φ can be optimized by minimizing task training loss $l_{task}(\cdot)$ as follows.

$$\varphi^* = \arg\min_{\varphi} l_{task}(q_\varphi, f_{\theta^*}, TD_{task}) \quad (6)$$

Then, the generalizable node representations learned from unlabeled graphs are fine-tuned to facilitate downstream tasks.

C. Coreset Selection

Consider a task with model $\mathcal{M}_\theta$ where parameters $\theta \in \Theta$, a training dataset $TD = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where features $\mathbf{x}_i \in \mathbb{R}^{d_x}$ and label $\mathbf{y}_i \in \{0,1\}^{|Y|}$, and a loss function $l(\theta, \mathbf{x}_i, \mathbf{y}_i)$, the model $\mathcal{M}_\theta$ can be optimized by minimizing the loss on training data TD as $\theta^* = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n l(\theta, \mathbf{x}_i, \mathbf{y}_i)$. Recently, the parameters θ can be optimized by stochastic gradient decent (SGD) approach, i.e, $\theta = \theta - \alpha \cdot \frac{1}{n} \sum_{i=1}^n \nabla_\theta l(\theta, \mathbf{x}_i, \mathbf{y}_i)$ where α is a learning rate.

However, it is time-consuming to optimize the model on large training datasets. Recent researchers observe that similar data instances exist in the training data [81], [82], [83], [84]. Therefore, they propose to select a subset of nodes (i.e., coreset) with size k from all the training data and train the model on the coreset [85], [86], [87], [88], [82], [83], [84]. Existing effective coreset selection approaches utilize gradient approximation approaches [85], [87], [82], [89]. They expect the gradients on all training data TD and on the coreset $S \subseteq TD$ to be as similar as possible, which can be defined as follows.

$$\min_S \max_{\theta \in \Theta} \|\sum_{i=1}^n \nabla_\theta l(\theta, \mathbf{x}_i, \mathbf{y}_i) - \sum_{j=1}^{|S|} \lambda_j \nabla_\theta l(\theta, \mathbf{x}_j, \mathbf{y}_j)\| \quad (7)$$

$$s.t.\ |S| \leq k, \sum_{j=1}^{|S|} \lambda_j = |TD|, \text{and } \lambda_j \in \mathbb{N},\ \forall j \in \{1, \cdots, |S|\}$$

where $\lambda_j \in \mathbb{N}$ is the weight of instance $(\mathbf{x}_j, \mathbf{y}_j)$, which is the number of nodes that it can represent. Since more similar gradients can achieve similar optimized parameters, thereby coreset approaches can perform similar on test data.

Nevertheless, existing coreset selection approaches [5], [85], [86], [87], [88], [82], [83], [84], [89] are not suitable for contrastive learning. They need labels for instances, while there are no instance labels in contrastive learning as introduced in Sec. II-B1 and Alg. 1 (lines 1-5). As a result, none of the existing approaches analyze how to extend Eq. (7) to select a coreset under graph contrastive learning setting.

III. Node Selector

As introduced in Sec. II-B, existing GCL models utilize all nodes to pre-train GNNs, restricting the efficiency due to the redundant nodes in the graph. In this section, we first theoretically analyze the gradient approximation of the coreset for GCL and formulate a cluster-based coreset selection problem. Then, we prove that this problem is NP-hard and propose an efficient algorithm with theoretical guarantee.

A. Theoretical Analysis

Based on Eq. (3), we first transform the coreset selection problem of supervised learning in Eq. (7) to graph contrastive learning. Formally, given a graph $G(V, \mathbf{A}, \mathbf{X})$, node budget k, GNNs model f_θ with parameters $\theta \in \Theta$, and contrastive loss $l_{cl}(\cdot)$, the target is to select a coreset $V_s \subseteq V$ by minimizing the follows.

$$\min_{V_s \in V} \max_{\theta \in \Theta} \| \sum_{v \in V} \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \sum_{u \in V_s} \lambda_u \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u) \| \quad (8)$$
$$s.t.\ |V_s| \le k,\ \sum_{u \in V_s} \lambda_u = |V|, \text{and } \lambda_u \in \mathbb{N},\ \forall u \in V_s$$

However, it is infeasible to directly solve the optimization problem. It is because it requires us to calculate the gradient of the loss function over each parameter $\theta \in \Theta$, which is too expensive. Therefore, following [85], we propose how to quickly measure the gradient difference between all training nodes V and the selected coreset V_s . Specifically, we define a mapping function $\gamma : V \to V_s$ in that the node $v \in V$ are represented by the selected node $u \in V_s$. Then, for arbitrary parameters $\theta \in \Theta$, we can get the following inequality based on triangle inequality.

$$\begin{aligned} &\| \sum_{v \in V} \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \sum_{u \in V_s} \lambda_u \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u) \| \quad (9) \\ &\le \| \sum_{v \in V} (\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_{\gamma(v)}, \tilde{\mathbf{h}}_{\gamma(v)})) \| \\ &\le \sum_{v \in V} \| \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_{\gamma(v)}, \tilde{\mathbf{h}}_{\gamma(v)}) \| \end{aligned}$$

The upper bound of Eq. (9) is minimized when each node $v \in V$ is assigned to the node $u \in V_s$ with the most gradient similarity, i.e., $\gamma(v) = \arg\min_{u \in V_s} \|(\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u))\|$. Hence, we can get the inequality:

$$\begin{aligned} &\| \sum_{v \in V} \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \sum_{u \in V_s} \lambda_u \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u) \| \\ &\le \sum_{v \in V} \min_{u \in V_s} \| \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u) \| \quad (10) \end{aligned}$$

To bound the estimation error for all $\theta \in \Theta$, we consider a worst-case approximation of the estimation error $\|\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u)\|$. Specifically, without loss of generality, we utilize the representative GCN for analysis. Following [90], [91], [92], we relax the nonlinear function $\sigma(\cdot)$ in Eq. (1) of as a linear function, i.e., $\mathbf{H} = \mathbf{A}_n^L \mathbf{X}\theta$. The relaxed GCN can capture the key idea of GNNs: aggregating information from neighbors. Also, we set $l_{cl}(\cdot)$ in Eq. (5) as $l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) = \|\hat{\mathbf{h}}_v - \tilde{\mathbf{h}}_v\|_2^2$ by excluding the randomly sampled negative nodes. Then, we can get Theorem 1.

Theorem 1. *Let the raw aggregated information difference of each node $v \in V$ between each sampled positive view $\hat{G}_v$ and the original graph G_v be bound with ϵ, i.e., $\|\mathbf{A}_n^L \mathbf{X}[v] - \hat{\mathbf{A}}_n^L \hat{\mathbf{X}}[v]\| \le \epsilon$. Given the graph $G(V, \mathbf{A}, \mathbf{X})$, an L-layer GCN f_θ with parameters $\theta \in \Theta$, the contrastive loss function $l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) = \|\hat{\mathbf{h}}_v - \tilde{\mathbf{h}}_v\|_2^2$, the gradient difference between nodes v and u can be bounded as: $\|\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u)\| \le c\|\mathbf{A}_n^L \mathbf{X}[v] - \mathbf{A}_n^L \mathbf{X}[u]\| + 4c\epsilon$, where the constant $c = 8\epsilon \cdot \max_{\theta \in \Theta} \|\theta\|$.*

Proof. For Similarity, we use $\mathbf{r}_v$ (resp. $\mathbf{R}$) to denote $\mathbf{A}_n^L \mathbf{X}[v]$ (resp. $\mathbf{A}_n^L \mathbf{X}$). The derivative $\nabla_\theta l_{cl}(\hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v)$ can be computed as:$\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) = (\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v)^\top (\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v)\theta$. Then, we can get the following inequality.

$$\begin{aligned} &\|\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u)\| \\ &= \|(\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v)^\top (\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v)\theta - (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u)^\top (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u)\theta\| \\ &\le \|\theta\| \|(\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v)^\top (\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v) - (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u)^\top (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u)\| \\ &\le \|\theta\| \|((\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v)^\top - (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u)^\top)(\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v) + \\ &\qquad (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u)^\top ((\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v) - (\hat{\mathbf{r}}_u - \tilde{\mathbf{r}}_u))\| \\ &\le \|\theta\| (\|\triangle \mathbf{r}_v\| + \|\triangle \mathbf{r}_u\|) \|\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v - \hat{\mathbf{r}}_u + \tilde{\mathbf{r}}_u\| \\ &\le \|\theta\| (\|\triangle \mathbf{r}_v\| + \|\triangle \mathbf{r}_u\|) (\|\hat{\mathbf{r}}_v - \hat{\mathbf{r}}_u\| + \|\tilde{\mathbf{r}}_v - \tilde{\mathbf{r}}_u\|) \\ &\le 8\epsilon \|\theta\| (\|\mathbf{r}_v - \mathbf{r}_u\| + 4\epsilon) \quad (11) \end{aligned}$$

Thus, under all $\theta \in \Theta$, the difference $\|\nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_v, \tilde{\mathbf{h}}_v) - \nabla_\theta l_{cl}(\theta, \hat{\mathbf{h}}_u, \tilde{\mathbf{h}}_u)\| \le 8\epsilon \max_{\theta \in \Theta} \|\theta\| (\|\mathbf{r}_v - \mathbf{r}_u\| + 4\epsilon)$. □

Based on Theorem 1, we obtain the upper bound of coreset selection on graph contrastive learning. Theorem 1 indicates that the gradient difference between two nodes v and u can be bounded by the raw aggregated information difference $\|\mathbf{A}_n^L \mathbf{X}[v] - \mathbf{A}_n^L \mathbf{X}[u]\|$, which is independent of the GNN parameter θ. Therefore, based on Theorem 1, we can transform the Eq. (8) as follows.

$$RS(V_s) = \min_{V_s \subseteq V, |V_s| \le k} \sum_{v \in V} \min_{u \in V_s} \|\mathbf{A}_n^L \mathbf{X}[v] - \mathbf{A}_n^L \mathbf{X}[u]\| \quad (12)$$

However, there are two issues to directly optimizing Eq. (12): (1) It requires compare the distance between

all pairs of nodes to select k nodes, resulting in a time complexity at least $O(k|V|^2)$. Such a computational burden makes it impractical to scale well on large graphs. (2). The selected coreset may be class-imbalanced. Specifically, we can partition nodes V into n_c clusters $\mathcal{C} = \{C_i\}_{i=1}^{n_c}$ by KMeans based on $\mathbf{A}_n^L\mathbf{X}$. Then, we can transform $\sum_{v\in V}\min_{u\in V_s}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u]\|$ to $\sum_{i=1}^{n_c}\sum_{v\in C_i}\min_{u\in V_s}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u]\|$. Thus, since $\sum_{v\in C_i}\min_{u\in V_s}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u]\|$ can bring larger value for large communities, it tends to select node in the larger community $C_i \in \mathcal{C}$ and ignore nodes in smaller communities. As a result, the selected coreset under a limited budget becomes class-imbalanced. The selected nodes from overrepresented classes may dominate the node representation learning process, potentially leading to suboptimal node representation learning for nodes in underrepresented classes. Also, class-imbalanced coresets impede the ability of GCL model to learn discriminative node representations that can distinguish nodes in different classes. It is because the comparisons between nodes in overrepresented classes and nodes in underrepresented classes are limited. Consequently, the node representations learned on class-imbalance coreset may lead to poor performance on downstream tasks.

B. Cluster-based Corset Selection Problem

To solve these two issues, we formulate a cluster-based coreset selection problem. The basic idea is to focus on measuring the distance between each pair of nodes in the same cluster while using a relaxed distance metric to compute the distance between nodes in different clusters. Given node clusters $\mathcal{C} = \{C_i\}_{i=1}^{n_c}$, Eq. (12) can be bounded as follows.

$$\begin{aligned}\sum_{v\in V}\min_{u\in V_s}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u]\| &= \sum_{i=1}^{n_c}\sum_{v\in C_i}\min_{u\in V_s}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u]\| \\ \leq \sum_{i=1}^{n_c}\sum_{v\in C_i}\min(&\min_{u_1\in C_{V_s,i}}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u_1]\|, \\ &\min_{u_2\in V_s\setminus C_{V_s,i}}(\|\mathbf{c}_i-\mathbf{A}_n^L\mathbf{X}[u_2]\|+d_i^{max})) \end{aligned} \tag{13}$$

where $C_{V_s,i} = \{v|v \in C_i, v \in V_s\}$ is nodes in coreset V_s belonging to cluster i. $\mathbf{c_i}$ is the center vector of cluster i. d_i^{max} is the maximum distance between nodes in cluster i and the center of cluster i, i.e., $d_i^{max} = \max_{v\in C_i}\|\mathbf{c}_i - \mathbf{A}_n^L\mathbf{X}[v]\|$.

By optimizing the right formula in Eq. (13), we can exclusively compute the distance between nodes within the same cluster $C_i \in \mathcal{C}$ and the distance between each node and n_c cluster centers. It is more efficient compared to computing the distance between all pairs of nodes. Also, the distance between nodes in different clusters is relaxed to a larger value. Therefore, given a cluster C_i, the formula tends to use the node in C_i rather than the node in other clusters to represent the other nodes in C_i. Such a way encourage nodes in coreset from different clusters, thereby alleviating the class-imbalance issue. In summary, we formally define the cluster-based corset selection problem for graph contrastive learning as follows.

Definition 1 (Cluster-based Coreset Selection Problem)**.** *Given graph $G(V, \mathbf{A}, \mathbf{X})$, budget k, the GNN with L, and node clusters $\mathcal{C} = \{C_i\}_{i=1}^{n_c}$, the target is to select a node coreset $V_s \subseteq V$ whose size $|V_s| \leq k$ by minimizing the objective:*

$$\begin{aligned}\min_{V_s\subseteq V,|V_s|\leq k}\sum_{i=1}^{n_c}\sum_{v\in C_i}\min(&\min_{u_1\in C_{V_s,i}}\|\mathbf{A}_n^L\mathbf{X}[v]-\mathbf{A}_n^L\mathbf{X}[u_1]\|, \\ &\min_{u_2\in V_s\setminus C_{V_s,i}}(\|\mathbf{c}_i-\mathbf{A}_n^L\mathbf{X}[u_2]\|+d_i^{max})) \end{aligned} \tag{14}$$

where $C_{V_s,i}$ is the set of nodes in coreset V_s belonging to cluster i. $\mathbf{c_i}$ is the center vector of cluster i. d_i^{max} is the maximum distance between nodes in cluster i and the center of cluster i, i.e., $d_i^{max} = \max_{v\in C_i}\|\mathbf{c}_i - \mathbf{A}_n^L\mathbf{X}[v]\|$.

Algorithm 2: Greedy Node Selection Algorithm

Input: Graph $G(V, \mathbf{A}, \mathbf{X})$, budget k, and GNN layer L
Output: Selected nodes V_s with weights $\{\lambda_v|v \in V_s\}$

1 $V_s \leftarrow \emptyset$; $\mathbf{R} = \mathbf{A}_n^L\mathbf{X}$
2 $C = \mathbf{KMeans}(V, \mathbf{R}, n_c)$
3 **while** $|V_s| < k$ **do**
4 $V_{sam} = \mathbf{Random\_Sample}(V \setminus V_s, n_s)$
5 **for** $v \in V_{sam}$ **do**
6 $RS(V_s \cup \{v\}) \leftarrow$ **Eq.** (14)
7 $\triangle RS(v|V_s) = RS(V_s) - RS(V_s \cup \{v\})$
8 $v^* = \arg\max_{v\in V_{sam}} \triangle RS(v|V_s)$
9 $V_s \leftarrow V_s \cup \{v^*\}$
10 **for** $v \in V_s$ **do** $\lambda_v = |\{u|v = \arg\min_{v_1\in V_s} d(v_1, u), v \in V|$;
11 **Return** Selected nodes V_s with weights $\{\lambda_v|v \in V_s\}$

Theorem 2. *The coreset selection problem is NP-hard.*

Proof. In general, we prove the coreset selection problem is NP-hard by the reduction from the k-cluster problem [93]. We put the full proof in our technique report [94] Appx. A1 □

C. Sampling-based Greedy Algorithm

Theorem 2 indicates that the cluster-based coreset selection problem is NP-hard. Therefore it is unlikely to obtain the optimal solution of this problem in polynomial time unless P=NP. In this subsection, we propose an efficient sampling-based greedy algorithm to address this problem. Specifically, we first define the marginal representative score gain brought by each node v if we select it into V_s. Formally, given selected nodes V_s, the marginal representative score gain of a node v can be defined as $\triangle RS(v|V_s) = RS(V_s) - RS(V_s \cup \{v\})$.

As shown in Alg. 2, we will first initialize the representative node set V_s as an empty set and compute the hidden representations $\mathbf{R} = \mathbf{A}_n^L\mathbf{X}$ of nodes aggregating from neighbors (line 1). Then, we sample n_s nodes V_{sam} from the un-selected nodes $V \setminus V_s$. Finally, we will select the representative nodes with the maximum marginal representative gain $\triangle RS(v|V_s)$ into V_s from the sampled nodes V_{sam} (line 4-9). We will repeat the selection process until until exceeding the node budget k. Then, we assign each node in $u \in V$ to the most similar node $v \in V_s$ and set the weight of each node $v \in V_s$ as the number of assigned nodes (line 10).

Theorem 3. *Given $n_s = \frac{n}{k}\log\frac{1}{\epsilon}$, Alg. 2 can achieve an approximation ratio of $1 - 1/e - \epsilon$.*

Proof. Let V_i and V_{opt} denote the solution of Alg. 2 at the i-th iteration and the optimal solution of Eq. (14), respectively. We first prove that the expectation of V_{i+1} satisfies $\mathbb{E}(RS(V_{i+1})) \geq (1-(1-\frac{1-\epsilon}{k})^{i+1})RS(V_{opt})$. Then, we prove that $RS(V_s) \geq (1-1/e-\epsilon)\cdot RS(V_{opt})$. Due to space limit, we put the full proof in our technique report [94] Appx. A2. □

Time Complexity. Let cluster number be $n_c = \mu|V|$ where $\mu \in [0,1]$. In line 1, it takes $O(\bar{D}^L|V|d_x)$ to compute the hidden representations $\mathbf{R}$ [95], where $\bar{D}$ is the average degree. In line 2, it takes O($\mu|V|^2 d_x$) to cluster nodes. in line 4-9, it takes $O(|V|+|V|/\mu+\mu|V|n_s d_x)$ to select one representative node. Line 10 takes $O(k|V|)$ Thus, the total time complexity is $O(\bar{D}^L|V|d_x+\mu|V|^2 d_x+k(|V|+|V|/\mu+\mu|V|n_s d_x))$

IV. View Generator

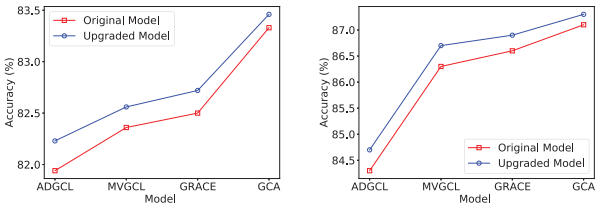
In this section, we first provide an insight to generate more expressive positive views. Second, we formulate the view generation problem that aims to generate both diverse and locality-preserved positive views.

A. Insights on Expressive Positive Views

Intuitively, more augmentation operations tend to create more expressive positive views. However, as shown in Tab. I, existing contrastive learning approaches only use limited operations, thereby restricting their ability to learn high-quality node representation and damaging their effectiveness on downstream tasks. We conduct experiments to further reveal this idea. Without loss of generality, we take four representative contrastive learning models as examples, including ADGCL [37], MVGRL [34], GRACE [20], and GCA [25]. We conduct node classification on two widely used datasets, namely Cora [96] and Amazon-computers [97]. As shown in Tab. I, these original models, ADGCL, MVGRL, GRACE, and GCA only cover {ED}, {EA,ED}, {ED,FM}, and {ED,FM}, respectively. We upgrade four models by adding {FP,EA}, {FP}, {EA,FP}, and {EA,FP} for them, respectively. As shown in Fig. 2, the blue line is above the red line, indicating that each ungraded model that uses more operations performs better than its original model on both datasets.

To enable the model to generate expressive views, one straightforward way is to employ all augmentation operations, including edge deletion/addition, node dropping/addition, feature perturbation/dropping/masking, subgraph sampling, etc. [22], [24]. However, as the number of operations increases, the model becomes more complex and the view generation process becomes less efficient [22], [98]. To overcome this issue, we provide Prop. 1 to demonstrate that three operations, including edge addition, edge deletion, and feature perturbation, can generate the same view space as all operations, which ensures the expressiveness of generated positive views.

Proposition 1. *In general, existing graph augmentation operation set $\mathcal{T}$ includes edge deletion, edge addition, feature masking, feature perturbation, feature dropping, node dropping, node addition, and subgraph sampling [22], [24]. Then, given $G(V,\mathbf{A},\mathbf{X})$, the positive view space of each node v generated by the three general operations, i.e., edge addition, edge deletion, and feature perturbation, is the same as the view space generated by all augmentation operations $\mathcal{T}$.*



(a) Cora. (b) Amazon-Computers.

Fig. 2: Motivational experiments on more operations.

Proof. In brief, we prove that the view $\hat{G}_v$ of a node v generated by any operation combination from $\mathcal{T}$ can be generated by three general operations. Due to space limit, we put the full proof in our technique report [94] Appx. A3. □

B. Positive View Generation Problem

Intuitively, these positive views of each node v should preserve its important locality information in the original graph, i.e., important local edges and node features. This important locality information contains the intrinsic patterns of each node, which can distinguish the differences among nodes. Then, GNNs optimized by maximizing the similarity of positive views in Eq. (3) are expected to learn node representations that capture nodes' intrinsic patterns. Given a pre-trained GNN model f_θ, the difference between node v's original local graph G_v and its positive view $\hat{G}_v$ can be measured by the difference between node representations $\mathbf{h}_v$ and $\hat{\mathbf{h}}_v$, i.e., $\|\mathbf{h}_v - \hat{\mathbf{h}}_v\|_2$, where $\mathbf{h}_v = f_\theta(G_v)$. A smaller $\|\mathbf{h}_v - \hat{\mathbf{h}}_v\|_2$ indicates that $\hat{G}_v$ can preserve more important locality information of node v.

In addition to preserving locality information, each positive view pair should be diverse, i.e., they should have different unimportant edges and features. In such a way, the pre-trained GNNs can be insensitive to various noises. One intuitive way of measuring the diversity of each pair of positive views $\hat{G}_v(\hat{\mathcal{V}}_v,\hat{\mathbf{A}}_v,\hat{\mathbf{X}}_v)$ and $\tilde{G}_v(\tilde{\mathcal{V}}_v,\tilde{\mathbf{A}}_v,\tilde{\mathbf{X}}_v)$ is based on their raw aggregated features, $\hat{\mathbf{A}}^L_{v,n}\hat{\mathbf{X}}_v$ (i.e., $\hat{\mathbf{r}}_v$) and $\tilde{\mathbf{A}}^L_{v,n}\tilde{\mathbf{X}}_v$ (i.e., $\tilde{\mathbf{r}}_v$), where $\hat{\mathbf{A}}^L_{v,n}$ is the normalized adjacency matrix for positive view $\hat{G}_v$. Specifically, As introduced in Eq. (1) in Sec. II-A, GNNs learn each node representation $\hat{\mathbf{h}}_v$ by a transformation on the raw aggregated features $\hat{\mathbf{A}}^L_{v,n}\hat{\mathbf{X}}_v$. Thus, if $\hat{\mathbf{A}}^L_{v,n}\hat{\mathbf{X}}_v$ and $\tilde{\mathbf{A}}^L_{v,n}\tilde{\mathbf{X}}_v$ are different while $\hat{\mathbf{h}}_v$ and $\tilde{\mathbf{h}}_v$ are the similar, we regard that $\hat{G}_v$ and $\hat{G}_v$ are diverse and contain different noise. Formally, we define our view generation objective as follows.

Definition 2 (View Generation Problem)**.** *Given selected nodes V_s, the GNN f_θ with L layers, the graph $G(V,\mathbf{A},\mathbf{X})$, and three general augmentation operations $\mathcal{T}_s$, including edge deletion, edge addition, and feature perturbation, the target is to generate two diverse and node locality-preserved positive views $\hat{G}_v(\hat{V}_v,\hat{\mathbf{A}}_v,\hat{\mathbf{X}}_v)$ and $\tilde{G}_v(\tilde{V}_v,\tilde{\mathbf{A}}_v,\tilde{\mathbf{X}}_v)$ for each node v by minimizing the following objective $l_{vg}(G,v,\mathcal{T}_s)$.*

$$\min_{\hat{G}_v,\tilde{G}_v} \|\hat{\mathbf{h}}_v - \mathbf{h}_v\|_2 + \|\tilde{\mathbf{h}}_v - \mathbf{h}_v\|_2 - \|\hat{\mathbf{r}}_v - \tilde{\mathbf{r}}_v\|_2 \qquad (15)$$

where $\mathbf{h}_v$*,* $\hat{\mathbf{h}}_v$*, and* $\tilde{\mathbf{h}}_v$ *are learned by* f_θ *on the original graph* G*, two positive views* $\hat{G}_v$*, and* $\tilde{G}_v$*, respectively.*

Theorem 4. *The view generation problem is NP-hard.*

Proof. We prove the view generation problem is NP-hard by the reduction from the Subset Sum Problem (SSP) [99]. We put the full proof in our technique report [94] Appx. A4. □

C. Sampling Algorithm

Theorem 4 shows that it is unlikely to generate two optimal positive views in polynomial time unless P=NP. Therefore, we propose an efficient edge-aware and feature-aware sampling algorithm. The basic idea is first to collect neighbor candidates for each selected node. Then, we compute the edge score between candidate neighbors and the target node and compute the feature score of each node. Based on these two scores, we can sample neighbor candidates with higher scores for connections and sample node features with lower scores for perturbations. In such a way, these positive views can preserve the important locality information. Moreover, the sampling process will produce variance on two positive views, which ensures the diversity of two positive views.

However, it is non-trivial to measure the edge score and feature score. Taking the edge score as an example, one straightforward way of measuring the score of an existing (resp. un-existing) edge regarding node v is to compute the node v's representation difference after deleting (resp. adding) the edge. Nevertheless, this process is inefficient [100], [90], because we need to recompute the node representation after each edge modification. Thus, we propose two metrics to measure edge and feature score efficiently as follows.

1) Edge Score: We measure the edge score from the perspective of node influence and node similarity. Intuitively, if the neighbor u of node v is more influential, the node u tends to affect the node representation of v. For example, in a citation network, the high influential pioneer work cited by a paper is more helpful in distinguishing the label of the paper. Therefore, It has a higher risk of destroying the locality intrinsic pattern of node v if we add a new edge (resp. delete an existing edge) between node v with an influential node. Formally, following [101], [25], we can compute the influential score of each node u based on node degree centrality: $\varphi_c(u) = \log(D_u + 1)$, where D_u is the degree of node u. Besides, the edge score also depends on the similarity between these two connected nodes. According to current studies [90], [91], [100], [102], if we delete (resp. add) edges with the nodes that are more similar (resp. dissimilar) with node v, the representation of v tends to be changed. Therefore, based on node centrality and node similarity, the edge score between the target node v and its neighbor candidate u can be computed as follows:

$$w^e_{v,u} = \begin{cases} \beta \cdot \exp(\varphi_c(u) + Sim(v,u)), & u \in N_v \\ (1-\beta) \cdot \exp(-\varphi_c(u) + Sim(v,u)), & u \in V \setminus N_v \end{cases},$$

where $Sim(v,u) = c - \|\mathbf{x}_v - \mathbf{x}_u\|$ is the feature similarity between v and u, and the constant $c = \max_{(v,u)\in\mathcal{E}} \|\mathbf{x}_v - \mathbf{x}_u\|$ and $\mathcal{E}$ denote all edges.

Algorithm 3: Positive View Generation Algorithm

Input: Graph $G(V, \mathbf{A}, \mathbf{X})$, layer numer L, selected nodes V_s, the neighbor ratio $\hat{\tau}$ and $\tilde{\tau}$, and feature perturbation hyperparameters $\hat{\eta}$ and $\tilde{\eta}$
Output: Positive view set $\hat{PG}$ and $\tilde{PG}$

1 $\hat{PG}, \tilde{PG} \leftarrow \emptyset, \emptyset$
2 **for** $v \in V_s$ **do**
3 $\hat{G}_v \leftarrow \{v, \hat{\mathbf{A}}_v, \hat{\mathbf{X}}_v\}, \tilde{G}_v \leftarrow \{v, \tilde{\mathbf{A}}_v, \tilde{\mathbf{X}}_v\}$
4 **for** $l = 0$ ***to*** $L-1$ **do**
5 **for** $u \in \hat{\mathcal{N}}_v^l$ **do**
6 $V_u^N = N_u^1 \cup N_u^2$
7 $\hat{\mathcal{N}}_u = \textbf{Sample}(V_u^N, P(\cdot|u, V_u^N), \hat{\tau}|N_u|)$
8 **for** $u_1 \in \hat{\mathcal{N}}_u$ **do** $\hat{\mathbf{A}}_v[u][u_1] = 1, \hat{\mathbf{X}}_v[u_1] = \mathbf{X}[u_1]$
9 **for** $u \in \tilde{\mathcal{N}}_v^l$ **do**
10 $V_u^N = N_u^1 \cup N_u^2$
11 $\tilde{\mathcal{N}}_u = \textbf{Sample}(V_u^N, P(\cdot|u, V_u^N), \tilde{\tau}|N_u|)$
12 **for** $u_2 \in \tilde{\mathcal{N}}_u$ **do** $\tilde{\mathbf{A}}_v[u][u_2] = 1, \tilde{\mathbf{X}}_v[u_2] = \mathbf{X}[u_2]$
13 **for** $u \in \hat{\mathcal{V}}_v$ **do**
14 **for** $i = 1$ ***to*** d_x **do** $\hat{\mathbf{X}}_v[u][i] \leftarrow$ **Eq.** (16)
15 **for** $u \in \tilde{\mathcal{V}}_v$ **do**
16 **for** $i = 1$ ***to*** d_x **do** $\tilde{\mathbf{X}}_v[u][i] \leftarrow$ **Eq.** (16)
17 $\hat{PG} = \hat{PG} \cup \{\hat{G}_v\}, \tilde{PG} = \tilde{PG} \cup \{\tilde{G}_v\}$
18 **Return:** Positive view set $\hat{PG}$ and $\tilde{PG}$

2) Feature Score: We measure the node feature score from the perspective of feature frequency and node influence. Intuitively, if a feature appears more frequently in the influential nodes, this feature is more important [25]. Thus, the global importance w_i^f of the i-th dimension feature can be defined as $w_i^f = \sum_{v\in V} \varphi_c(v) \cdot |\mathbf{x}_v[i]|$, where $\mathbf{x}_v[i]$ is node v's i-th dimension feature. Besides, as we discussed above, one neighbor with high centrality tends to affect the target node's representation. Thus, if the neighbor u has larger centrality, the probability of perturbing its features should be lower. Formally, given a target node v, the score of the i-th dimension feature of its neighbor u can be computed as $w^f_{\mathbf{x}_v[i]} = w_i^f \cdot \varphi_c(v)$.

3) Edge-aware and Feature-aware Sampling Algorithm: The basic idea is to generate two diverse and node locality-preserved positive views for each selected node by sampling nodes with higher edge scores as neighbors and perturbing unimportant node features accordingly. The sampling algorithm is summarized in Alg. 3. Specifically, for each selected node $v \in V_s$, we first initialize its two positive views $\hat{G}_v$ and $\tilde{G}_v$ (line 3), where each positive view only contains the target node v. The adjacency matrix $\hat{\mathbf{A}}_v$ and $\tilde{\mathbf{A}}_v$ are empty, and node features $\hat{\mathbf{X}}_v$ and $\tilde{\mathbf{X}}_v$ only contain the node feature $\mathbf{x}_v$. Then, we sample neighbors of node v from 1-hop to L-hop iteratively for the two positive view $\hat{G}_v$ and $\tilde{G}_v$ (line 4-12). Particularly, for each node $u \in N_v^l$, we only consider the 1-hop and 2-hop neighbors of u in the original graph as its neighbor candidates $V_u^N = N_u^1 \cup N_u^2$. Such a way avoids introducing too much noise. Then, we independently sample $\hat{\tau}|N_u|$ edges from V_u^N, where the probability of each node $u_1 \in V_u^N$ that is selected as the neighbor of u is defined as the normalized edge score as $P(u_1|u, V_u^N) = \frac{w^e_{u,u_1}}{\sum_{u_2\in V_u^N} w^e_{u,u_2}}$. Similarly, we can independently sample $\tilde{\tau}|N_u|$ nodes from V_u^N

as the neighbors for the positive view $\tilde{G}_v$. We will repeat the sampling process until we obtain the L-hop neighbors of node v for both positive views. After finishing sampling edges for both positive views, we will perturb the features of each node in two positive views (line 13-16). Specifically, for each node $u \in \hat{\mathcal{V}}_v$ in positive view $\hat{G}_v$,we can perturb the i-th dimensional feature $\mathbf{x}_u[i]$ as follows:

$$\hat{\mathbf{x}}_u[i] = \mathbf{x}_u[i] + \hat{m}_u[i] \cdot (2 \cdot \textbf{Uniform}(0,1) - 1) \cdot \mathbf{x}_u[i], \quad (16)$$

where $\hat{m}_u[i] \sim Bernoulli(\hat{p}_{\mathbf{x}_u[i]}) \in \{0,1\}$ is a mask that decides whether to perturb $\mathbf{x}_u[i]$. Also, $\hat{p}_{\mathbf{x}_u[i]} = \hat{\eta} \cdot \frac{w^f_{max} - w^f_{\mathbf{x}_u[i]}}{w^f_{max} - w^f_{mean}}$ is the normalized probability of perturbing the node u's i-th dimensional feature $\mathbf{x}_u[i]$, where two constants $w^f_{max} = \max_{v \in V} w^f_{\mathbf{x}_v[i]}$ and $w^f_{mean} = \sum_{v \in V} w^f_{\mathbf{x}_v[i]}/|V|$ are the max and mean value of i-th feature score of all nodes, and $\hat{\eta}$ is a hyperparameter to control the maximum probability to perturb node features. Additionally, $2 \cdot \textbf{Uniform}(0,1) - 1 \in [-1,1]$ is to control the magnitude of feature perturbation. Generally, a node feature with a lower score is more likely to be perturbed. Similarly, we can perturb the node features for positive view $\tilde{G}_v$ with the hyperparamter $\tilde{\eta}$. Finally, the algorithm will return the positive views $\hat{PG}$ and $\tilde{PG}$ for all selected nodes.

Time Complexity. The edge score and feature score of candidate nodes can be pre-computed and used for all selected nodes V_s, which takes $O(|V_s|\bar{D}^{L+1} + |V|d_x)$ time. Given an L-layer GNN, it takes $O(\bar{D}^{L+1})$ to construct the structure of two positive views for each selected node (line 3-12). Then, it takes $O(\bar{D}^L d_x)$ to perturb node features (line 13-16). Therefore, the total complexity of generating positive views for all nodes is $O(|V|d_x + |V_s|(\bar{D}^{L+1} + \bar{D}^L d_x))$.

Remarks: As described in Sec. IV-C, Alg. 3 computes edge scores and feature scores only based on node features and degrees, which are extracted from the raw graph data. In other words, it does not depend on GNN parameters or GNN outputs, making it suitable for generating views for any GNNs.

V. Experiments

In this section, we compare our proposed framework E^2GCL against state-of-the-art baselines. Also, we visualize the selected nodes in technique report [94] Appx. B4.

A. Experiment Setting

1) Tasks and Datasets: Following current approaches [22], [20], [25], [34], [98], we utilize the node classification task to evaluate our framework on seven widely used benchmark datasets, namely, Cora [96], Citeseer [96], Computers [97], Photo [97], CS [97], Arxiv [103], and Products [104]. The detailed statistics are in Tab. III and the detailed descriptions are in our technique report Appx. B2.

2) Evaluation Protocol: We follow the evaluation protocol of previous state-of-the-art contrastive learning approaches [22], [20], [24], [25], [34], [98]. As illustrated in Alg. 1 (line 1-5), existing contrastive learning approaches, including our proposed E^2GCL, first pre-train the encoder GNN f_θ in the contrastive learning manner, i.e., without labels. Particularly, existing approaches use all nodes to train GNNs,

TABLE III: The datasets statistics. $\overline{\textbf{Degree}}$ is the average degree per node. **#Feature** is the feature dimension.

Dataset	#Node	#Edge	$\overline{\textbf{Degree}}$	#Feature	#Class
Cora	2,708	5,278	3.89	1,433	7
Citeseer	3,327	4,552	2.74	3,703	6
Photo	7,650	119,081	31.13	745	8
Computers	13,752	245,861	35.76	767	10
CS	18,333	81,894	8.93	6,805	15
Arxiv	169,343	1,166,243	13.77	128	40
Products	1,569,960	264,339,468	336.74	200	107

i.e., $V_s = V$, while our proposed E^2GCL selects V_s under node budget k as introduced in Sec. III. Second, we evaluate the effectiveness of the contrastive learning approaches by a simple l_2-regularized linear decoder q_φ (line 6 in Alg. 1). This procedure allows node labels for the node classification task. For each dataset, we randomly split all nodes V into 10%, 10%, and 80% for training, validation, and testing, respectively. Besides, for fair comparisons, we run the evaluation process 10 times on different data splits, and report the average test accuracy and standard deviation.

3) Baselines: We follow existing approaches to compare representative and state-of-the-art baselines in the three categories: (1) Supervised and Semi-supervised learning approaches, including MLP and GCN [18], are trained by the end-to-end manner on node labels in Eq. (3). (2) Traditional unsupervised learning approaches, including DeepWalk [32] and Node2Vec [105]. (2) Graph contrastive learning approaches, including GAE [106], VGAE [106], DGI [107], AFGRL [31], BGRL [36], MVGRL [34], GRACE [20], GCA [25]. The last two categories follow Alg. 1 for evaluation. They first use **all nodes** to optimize their encoder to learn node representations from the unlabeled graphs. Then, they use these learned representations to train a simple l_2-regularized linear decoder for node classification. Different from existing works, our proposed E^2GCL selects V_s under **node budget** $\boldsymbol{k}$ and only use V_s to pre-train GNNs. Due to space limit, the baseline details are list in technique report [94] Appx. B3.

4) Hyperparameter Setting: For the baselines, we use their default hyper-parameter setting. Also, unsupervised and contrastive learning baselines use all nodes to learn representations from unlabeled graphs. For our proposed E^2GCL, due to different datasets with different sizes, we use the node ratio r to control the node budget k, i.e., $k = r \cdot |V|$. By default, we set $r = 0.4$ for seven datasets. Also, we take a 2-layer GCN [18] as our encoder model. Besides, based on validation data, we tune cluster number in Alg. 2 $n_c \in \{100, 200, \cdots, 1000\}$ for the large Products data and tune $n_c \in \{30, 60, \cdots, 180\}$ for the other datasets. We tune sampling number in Alg. 2 $n_c \in \{200, 400, \cdots, 2000\}$ for the large Products data and tune $n_s \in \{100, 200, \cdot, 1000\}$ for the other datasets. We tune the neighbor sampling number $\hat{\tau}$ and $\tilde{\tau}$, and feature perturbation $\hat{\eta}$ and $\tilde{\eta}$ in Alg. 3 from $\{0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$. We set the upper value of the neighbor and feature as 1.4, since sampling more edges and perturbing more node features will introduce noise. The training batch is 500 for all approaches.

B. Main Results

1) Effectiveness Evaluation: As shown in Tab. IV and Tab. V, our proposed E^{2}GCL outperforms the other baselines on all seven datasets. Specifically, supervised approaches, including GCN and MLP, do not achieve superior performance. It is because these models only rely on limited training labels to optimize GNNs, thereby degrading the generalization of the learned node representations. Unlike supervised approaches, traditional unsupervised and current GCL approaches first learn node representations from the unlabeled graph and optimize a simple decoder with training labels as shown in Alg. 1. Such a way enables the learned node representations to be more generalizable. Particularly, traditional contrastive learning approaches, including DeepWalk and Node2Vec, achieve unsatisfying performance, because they only rely on graph structure and neglect node features to learn node representations. Instead, by incorporating both node features and graph structure, current GCL approaches, such as GCA, MVGRL, GRACE, and AFGRL, achieve satisfying performance.

E^{2}GCL outperforms these contrastive learning approaches, even though we only use the top-k nodes to optimize GNNs. It is because redundant nodes exist in a graph, which allows us to select representative nodes to represent the entire graph. Then, GNNs optimized by the positive views of these representative nodes can learn generalizable representations as well as GNNs optimized by all nodes. Besides, compared with current works, E^{2}GCL uses three general augmentation operations (edge addition, edge deletion, and feature perturbations) to generate more expressive, diverse, and importance information-preserved positive views for each node.

2) Efficiency Evaluation: First, we use widely used training time-accuracy curve to comprehensively evaluate the efficiency of E^{2}GCL. The total training time includes the time of training node selection, view generation, and GNN optimization. For simplicity, we show the time-accuracy curve of E^{2}GCL on Cora and Citeseer data, since similar trends are observed on other datasets. Also, without loss of generality, we only compare the most effective contrastive learning baselines, including AFGRL, BGRL, MVGRL, GRACE, and GCA. As illustrated in Fig. 3, E^{2}GCL converges faster and achieves better performance than baselines at the same time, demonstrating superior efficiency.

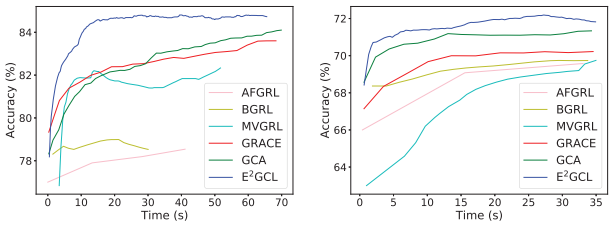
Second, we report the top-k node selection time (ST), the total training time (TT) of CGL models convergence (line 1-5 in Alg. 1), and the accuracy of node classification on the two large data, i.e., Arxiv and Products. Specifically, as shown in Tab. V, the node selection is efficient, which is only a relatively small portion of the total training time. Also, the total training time on E^{2}GCL convergence is smaller than other baselines. It is because E^{2}GCL selects a coreset of nodes with size k that can represent all nodes V. Besides, E^{2}GCL can generate expressive positive views. Such two ways ensure that E^{2}GCL can converge faster and perform the best.

C. Ablation Study

1) The Framework: Here, we compare the version of **E^{2}GCL** (**E^{2}GCL**$_{S,I}$) with the other three variants.

TABLE IV: Node classification performance (Accuracy $\pm$ Std). DW and N2V are DeepWalk and Node2Vec, respectively.

Model	Cora	Citeseer	Photo	Computers	CS
MLP	57.15±0.86	57.98±0.55	80.57±0.24	76.04±0.71	90.10±0.49
GCN	82.46±0.72	70.93±0.51	92.15±0.68	86.15±0.63	92.59±0.45
DW	72.93±0.91	52.67±0.65	88.10±0.94	83.31±0.52	81.94±0.98
N2V	71.61±0.85	54.06±0.61	87.85±0.86	83.36±0.57	83.25±0.78
GAE	78.35±0.58	67.36±0.63	90.61±0.43	81.62±0.76	89.77±0.59
VGAE	80.33±0.76	70.89±0.46	91.42±0.35	84.26±0.67	91.90±0.43
DGI	81.24±0.67	70.46±0.50	90.49±0.41	82.31±0.37	92.03±0.24
BGRL	79.52±0.56	70.06±0.96	91.35±0.40	86.10±0.51	90.07±0.29
AFGRL	81.94±0.73	70.38±1.15	92.23±0.24	87.46±0.56	93.04±0.16
MVGRL	82.36±0.55	71.23±0.54	90.98±0.26	87.24±0.47	92.36±0.31
GRACE	82.31±0.56	70.65±0.32	91.38±0.81	86.74±0.25	92.41±0.15
GCA	83.33±0.47	71.47±0.72	92.24±0.21	87.36±0.38	92.50±0.13
E^{2}GCL	**84.06±0.21**	**71.86±0.61**	**93.02±0.47**	**88.92±0.52**	**93.15±0.19**



(a) Cora. (b) Citeseer.

Fig. 3: Efficiency evaluation by accuracy-time curve.

E^{2}GCL$_{A,U}$ uses All nodes to train GNNs, and it Uniformly modifies edges and perturbs node features to generate positive views for nodes. **E^{2}GCL**$_{S,U}$ uses the Selected representative nodes to train GNNs, and it Uniformly modifies edges and perturbs node features. Also, **E^{2}GCL**$_{S,I}$ uses the Selected representative nodes to train GNNs, and it modifies edges and perturbs node features according to the edge and node feature Importance. As illustrated in Tab. VI, **E^{2}GCL**$_{A,U}$ and **E^{2}GCL**$_{S,U}$ achieve unsatisfying performance, since they uniformly modify edges and node features and thereby cannot preserve the important locality information of nodes. Particularly, **E^{2}GCL**$_{S,I}$ achieves the best and comparable performance with **E^{2}GCL**$_{A,I}$, even though **E^{2}GCL**$_{S,I}$ only train GNN on the top-k nodes. It is because these selected nodes can fully represent the entire graphs. Overall, since the training time decreases as the number of training nodes decreases, the best and comparable results on partial nodes imply the practicability of our proposed framework.

2) Node Selector: To verify that the selector can select representative nodes, we compare five baselines, i.e., Random, Degree, KMeans, KCG [108], and Grain [5]. Specifically, Random randomly selects k nodes from all nodes. KMeans first separates nodes into 10 clusters, and then randomly selects k nodes from each cluster. Degree defines the probability that each node v can be selected as $\frac{\log(D_v+1)}{\sum_{u\in V}\log(D_u+1)}$, where D_v is the node v's degree. Then, Degree samples k nodes from all nodes based on the degree-based probability. KCG and Grain are semi-supervised settings, i.e., they incorporate node labels to train models and compute the node similarity accordingly. To fit the contrastive settings (i.e., without labels), KCG and Grain use aggregated raw features to compute node similarity. As shown in Tab. VII, our proposed framework outperforms the other baselines. It demonstrates that our node selector

TABLE V: Results on two large data, i.e., Arxiv and Products. $\overline{\mathbf{ST}}$ and $\overline{\mathbf{TT}}$ denote the average **S**election **T**ime (seconds) and the total **T**raining **T**ime (seconds). $\overline{\mathbf{ST}}$ is " $-$ " for baselines since they use all nodes for training without selection. " $\sim$ " denotes that the model cannot converge in 3 days.

Model	Arxiv			Products		
	Accuracy	ST	TT	Accuracy	ST	TT
AFGRL	43.14±0.39	-	7,338.5	26.51±0.02	-	147,923.2
MVGCL	43.95±0.23	-	8,246.2	$\sim$	$\sim$	$\sim$
GRACE	43.37±0.31	-	7,781.3	26.28±0.03	-	208,261.9
GCA	44.76±0.29	-	6,292.9	26.91±0.03	-	193,825.7
$\mathbf{E^2GCL}$	**45.26±0.26**	70.5	**3,106.8**	**27.21±0.02**	4,219.2	**82,195.7**

can select more representative nodes than the other baselines, ensuring the effectiveness of GNNs.

3) Edge-aware and Feature-aware View Generator: To further verify the effectiveness of view generator in Sec. IV, we compare our edge-aware and feature-aware sampling with a uniform sampling manner. Specifically, we compare three variants of $\mathbf{E^2GCL\backslash F\backslash S}$, $\mathbf{E^2GCL\backslash S}$, and $\mathbf{E^2GCL\backslash F}$. Specifically, $\mathbf{E^2GCL\backslash F\backslash S}$ perturbs each dimension of node **F**eatures and modifies graph **S**tructure uniformly, i.e., the probabilities of perturbing node features and sampling an edge are the same to all nodes. Similarly, $\mathbf{E^2GCL\backslash S}$ samples edge uniformly while perturbing node features based on feature score. $\mathbf{E^2GCL\backslash F}$ perturbs node feature uniformly while sampling edges based on edge score.

As shown in Tab. VIII, our edge-aware and feature-aware $\mathbf{E^2GCL}$ outperforms the other variants, especially $\mathbf{E^2GCL\backslash F\backslash S}$ with uniform sampling. In addition, $\mathbf{E^2GCL\backslash S}$ and $\mathbf{E^2GCL\backslash F}$ are superior to $\mathbf{E^2GCL\backslash F\backslash S}$. It demonstrates that our generator can preserve the important locality information based on measuring both feature and edge scores, and thereby can learn high-quality node representation. In particular, $\mathbf{E^2GCL\backslash F}$ is better than $\mathbf{E^2GCL\backslash S}$. It is because modifying one important edge will have a greater impact on the quality of node representation than modifying one important feature [90], [91], [100].

D. Parameter Sensitivity

1) Node Budget k: We evaluate the effect of node budget on five datasets, including Cora, Citeseer, Photo, Computers, and CS. Recall that the node budget $k = r \cdot |V|$ is controlled by the node rate r. Here we study the influence of different node budget by setting node rates $r \in \{1, \frac{1}{2}, \cdots, \frac{1}{2^9}, \frac{1}{2^{10}}\}$ on the all five datasets. As shown in Fig. 4 (a), as the node budget decreases, the node classification accuracy of five datasets first maintains the similar result as GNNs trained on all nodes and then drops. It reveals that there exist redundant nodes in the graph, which provides an opportunity to select a node subset to represent the entire graph. In such a way, GNNs trained on the node subset can achieve similar performance as GNNs trained on all nodes. In particular, as the node budget decreases significantly, the performance on Computers and Photo decreases more drastically than the other three datasets. It is because the nodes in Computers and Photo are more similar than those of the other datasets. Thus, limited selected nodes cannot help GNNs learn distinguishable node representations, thus damaging the performance on downstream tasks.

TABLE VI: The effectiveness evaluation on the framework.

	Cora	Citeseer	Photo	Computers	CS
$\mathbf{E^2GCL}_{A,U}$	82.89±0.77	70.27±0.39	88.15±0.75	81.82±0.54	92.02±0.17
$\mathbf{E^2GCL}_{S,U}$	83.26±0.42	70.62±0.68	87.71±0.71	82.08±0.53	92.27±0.26
$\mathbf{E^2GCL}_{A,I}$	83.91±0.24	**72.14±0.56**	**93.11±0.17**	88.74±0.38	93.02±0.15
$\mathbf{E^2GCL}_{S,I}$	**84.06±0.21**	71.86±0.61	93.02±0.47	**88.92±0.52**	**93.15±0.19**

TABLE VII: Evaluation on different selection strategy.

	Cora	Citeseer	Photo	Computers	CS
Random	81.22±0.65	67.71±1.49	91.36±0.57	87.05±0.76	91.21±0.50
Degree	82.30±0.69	68.61±0.56	91.71±0.35	87.39±0.43	91.82±0.24
KMeans	82.49±0.56	70.52±0.60	92.30±0.29	88.10±0.24	92.10±0.14
KCG	82.61±0.49	70.27±0.54	92.46±0.25	87.81±0.19	92.32±0.26
Grain	83.21±0.67	70.94±0.58	92.65±0.36	88.26±0.35	92.64±0.21
Ours	**84.06±0.21**	**71.86±0.61**	**93.02±0.47**	**88.92±0.52**	**93.15±0.19**

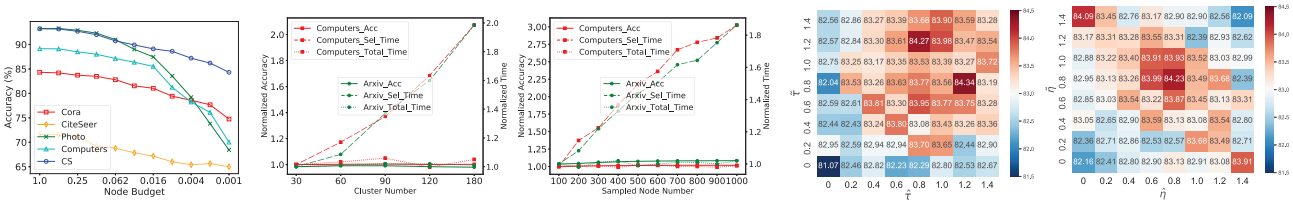
TABLE VIII: Evaluation on view generator sampling strategy.

	Cora	Citeseer	Photo	Computers	CS
$\mathbf{E^2GCL\backslash F\backslash S}$	82.67±0.56	70.40±0.67	86.02±0.51	81.52±0.56	91.98±0.14
$\mathbf{E^2GCL\backslash S}$	82.81±0.56	70.94±0.52	88.79±0.42	86.09±0.49	92.61±0.20
$\mathbf{E^2GCL\backslash F}$	83.21±0.59	71.30±0.45	92.51±0.28	88.41±0.32	92.82±0.19
$\mathbf{E^2GCL}$	**84.06±0.21**	**71.86±0.61**	**93.02±0.47**	**88.92±0.52**	**93.15±0.19**

2) Cluster Number n_c and Sampled Node Number n_s: We evaluate the effect of cluster number n_c and the sampled node number n_s in Alg. 2. First, we vary $n_c \in \{30, 60, 90, 120, 180\}$ on Computers and Arxiv and show node classification accuracy in y_1-axis, the node selection time and the total training time (including node selection, view generation, and GNN optimization) in y_2-axis. We set the node sampling number $n_s = 300$ to default. To put the results of two data into one picture, we normalize the accuracy and the time by dividing the result by the first variant $n_c = 30$. As shown in Fig. 4 (b), as the n_c increases, the selection time (dashed lines) increases, since comparisons between cluster centers and sampled nodes increase. Note that since the selection time includes pre-processing time, such as raw representation computation and clustering, selection time does not increase linearly with sampling number. Also, The total training time (dotted lines) and the accuracy (solid lines) change slightly with different n_c. It demonstrates that the top-k selected nodes based on different n_c have limited impacts on accuracy and time. In particular, the increases of selection time does not significantly affect the total time, because the selection time, as demonstrated in Section V-B2, is a relatively small portion of the total training time.

Second, we vary $n_s \in \{100, 200, \cdots, 1000\}$ on Computers and Arxiv and set the cluster number $n_c = 120$ to default. Similarly, we normalize the accuracy and the time by dividing the result by the first variant $n_s = 100$. As shown in Fig. 4 (c), as the n_s increases, the selection time (dashed lines) increases, since comparisons between cluster centers and sampled nodes increase as well. Also, the total training time (dotted lines) slightly changes, indicating that n_s has limited impacts on the total training time of $\mathrm{E^2GCL}$. Additionally, the accuracy (solid lines) on both datasets first rises before stabilizing. It demonstrates that sampling can speed up the selection process with satisfactory performance and that it is not necessary to employ all nodes when choosing the representative node.

3) The Hyperparameter $\hat{\tau}$ and $\tilde{\tau}$: We vary $\hat{\tau}, \tilde{\tau} \in \{0, 0.2, 0.6, 0.4, 0.8, 1.0, 1.2, 1.4\}$. Higher $\hat{\tau}$ and $\tilde{\tau}$ indicate



(a) Node budget $k = r \cdot |V|$ (b) Cluster number n_c (c) Sampled node number n_s (d) Edge rate $\hat{\tau}$ and $\tilde{\tau}$ (e) Feature rate $\hat{\eta}$ and $\tilde{\eta}$

Fig. 4: Parameter sensitivity evaluation.

sampling more neighbors for each node. Without loss of generality, we show the evaluation on Cora, since the observations are similar on the other datasets. As shown in Fig. 4 (d), as $\hat{\tau}$ and $\tilde{\tau}$ are smaller, the accuracy is unsatisfied, because sampling a few neighbors for each node cannot preserve the node locality information. In addition, as $\hat{\tau}$ and $\tilde{\tau}$ increase, the accuracy first increases and then decreases. The reason is that sampling more neighbors for each node can preserve node locality information. Besides, due to the sampling variance, two diverse positive views have different structures and node features. Thus, GNNs optimized on these locality-preserved and diverse positive views can perform effectively. However, as $\hat{\tau}$ and $\tilde{\tau}$ are very larger, the accuracy decreases, because sampling too many neighbors may introduce noise.

4) The Hyperparameter $\hat{\eta}$ and $\tilde{\eta}$: Similar to Sec. V-D3, we vary $\hat{\eta}, \tilde{\eta} \in \{0, 0.2, 0.6, 0.4, 0.8, 1.0, 1.2, 1.4\}$ on Cora. The larger $\hat{\eta}$ and $\tilde{\eta}$ tend to perturb more node features. As illustrated in Fig. 4 (e), as $\hat{\eta}$ and $\tilde{\eta}$ increase, the accuracy first increases and then decreases. It is because our generator tends to perturb the unimportant features and maintain the important features. Thus, when $\hat{\eta}$ and $\tilde{\eta}$ initially increase, the generated positive views can preserve the node's important locality information and are diverse, thereby enhancing the performance of GNNs. Also, when both $\hat{\eta}$ and $\tilde{\eta}$ are very large, the accuracy decreases, because higher $\hat{\eta}$ and $\tilde{\eta}$ will cause important features to be perturbed. Thus, positive views lose the important locality information in the original graph.

E. Additional Experiments

To explore more generality of our E^2GCL, we conduct experiments on link prediction and graph classification tasks. We compare E^2GCL with these effective contrastive learning baselines, including AFGRL, BGRL, MVGRL, GRACE, and GCA. The setting of GCL models is the same in Sec. V-A4.

1) Link Prediction Task: We utilize three datasets, i.e., Photo [97], Computer [97], and CS [97] to evaluate our framework and baselines. Specifically, given all edges, we randomly select 70%/10%/20% edges as the training/validation/testing edges. Note that when optimizing GNNs by E^2GCL and other GCL baselines, we exclusively retained the training edges in the graph to prevent potential leakage of validation and testing information. As shown in Tab. IX, E^2GCL outperforms all baselines, demonstrating that E^2GCL is also generalizable on the link prediction task.

2) Graph Classification Task: Following [34], [60], [109], we use the SUM function as the READOUT function, i.e., $\mathbf{z}_i = \sum_{v \in V_i} \mathbf{H}_i[v]$. We utilize three benchmark datasets for evaluation, including NCI1 [110], PTC_MR [110], and Proteins [110]. The data statistics and descriptions are in our technique report [94] Appx-Tab. X. Specifically, for each dataset, we randomly select 70%/10%/20% graphs as the training/validation/testing data. We set the node budget $k_i = r|V_i|$ for each graph G_i. As shown in Tab. IX, E^2GCL outperforms all baselines, demonstrating that E^2GCL is also generalizable for the graph classification task. It is because E^2GCL can learn high-quality node representations for each node. As a result, the representation of each graph summarized on node representations is also high-quality, leading to more accurate graph class predictions.

TABLE IX: Results on link prediction and graph classification.

	Link Prediction			Graph Classification		
	Photo	Computer	CS	NCI1	PTC_MR	Proteins
AFGCL	71.87±1.95	72.95±1.82	66.95±0.59	74.79±2.07	69.84±2.17	76.77±1.98
BGRL	71.74±2.02	72.30±1.79	65.92±0.86	74.12±2.51	68.21±2.95	76.12±2.61
MVGCL	71.49±1.71	72.92±1.64	66.61±0.72	74.71±2.12	69.21±2.76	76.57±1.87
GRACE	71.71±1.91	72.64±1.59	66.45±0.77	74.57±1.82	68.88±2.23	76.89±1.93
GCA	72.30±1.68	73.21±1.82	67.32±0.83	75.13±1.69	70.12±1.76	76.96±2.32
E^2GCL	**72.41±1.72**	**73.57±1.58**	**67.66±0.78**	**75.57±1.75**	**70.55±2.19**	**77.12±1.76**

VI. Conclusion

In this paper, we propose an efficient and expressive contrastive learning framework for GNNs, namely E^2GCL, which consists of two components, i.e., node selector and view generator. First, instead of all nodes, the node selector selects a limited number of nodes that can represent the entire graph to train GNNs. Second, the view generator employs three general operations (edge deletion, edge addition, and feature perturbation) to generate expressive, diverse, and locality-preserved positive views for selected nodes based on edge and feature importance. The superior effectiveness and efficiency of our proposed E^2GCL are demonstrated by extensive experiments on various downstream tasks.

Acknowledgment

Lei Chen's work is partially supported by National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant and HKUST-Webank joint research lab grants. Xiaofang Zhou's work is partially supported by the JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust, HKUST-China Unicom Joint Lab on Smart Society, and HKUST-HKPC Joint Lab on Industrial AI and Robotics Research.

REFERENCES

[1] Y. Cui, K. Zheng, D. Cui, J. Xie, L. Deng, F. Huang, and X. Zhou, "Metro: a generic graph neural network framework for multivariate time series forecasting," *Proceedings of the VLDB Endowment*, vol. 15, no. 2, pp. 224–236, 2021.

[2] Y. Chen, X. Li, G. Cong, C. Long, Z. Bao, S. Liu, W. Gu, and F. Zhang, "Points-of-interest relationship inference with spatial-enriched graph neural networks," *Proceedings of the VLDB Endowment*, vol. 15, no. 3, pp. 504–512, 2021.

[3] A. Vretinaris, C. Lei, V. Efthymiou, X. Qin, and F. Özcan, "Medical entity disambiguation using graph neural networks," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2310–2318.

[4] X. Wang, D. Lyu, M. Li, Y. Xia, Q. Yang, X. Wang, X. Wang, P. Cui, Y. Yang, B. Sun *et al.*, "Apan: Asynchronous propagation attention network for real-time temporal graph embedding," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2628–2638.

[5] W. Zhang, Z. Yang, Y. Wang, Y. Shen, Y. Li, L. Wang, and B. Cui, "Grain: improving data efficiency of gra ph neural networks via diversified in fluence maximization," *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2473–2482, 2021.

[6] R. Zhu, K. Zhao, H. Yang, W. Lin, C. Zhou, B. Ai, Y. Li, and J. Zhou, "Aligraph: a comprehensive graph neural network platform," *Proceedings of the VLDB Endowment*, vol. 12, no. 12, pp. 2094–2105, 2019.

[7] C. Zheng, H. Chen, Y. Cheng, Z. Song, Y. Wu, C. Li, J. Cheng, H. Yang, and S. Zhang, "Bytegnn: efficient graph neural network training at large scale," *Proceedings of the VLDB Endowment*, vol. 15, no. 6, pp. 1228–1242, 2022.

[8] R. Yang, J. Shi, X. Xiao, Y. Yang, and S. S. Bhowmick, "Homogeneous network embedding for massive graphs via reweighted personalized pagerank," *Proceedings of the VLDB Endowment*, vol. 13, no. 5, pp. 670–683, 2020.

[9] L. Wei, H. Zhao, and Z. He, "Designing the topology of graph neural networks: A novel feature fusion perspective," *arXiv preprint arXiv:2112.14531*, 2021.

[10] H. LI, S. Di, and L. Chen, "Revisiting injective attacks on recommender systems," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29 989–30 002, 2022.

[11] Z. Wang, S. Di, and L. Chen, "Autogel: An automated graph neural network with explicit link information," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 509–24 522, 2021.

[12] C. T. Duong, T. D. Hoang, H. Yin, M. Weidlich, Q. V. H. Nguyen, and K. Aberer, "Efficient streaming subgraph isomorphism with graph neural networks," *Proceedings of the VLDB Endowment*, vol. 14, no. 5, pp. 730–742, 2021.

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

[14] X. Liu, H. Pan, M. He, Y. Song, X. Jiang, and L. Shang, "Neural subgraph isomorphism counting," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1959–1969.

[15] G. Bouritsas, F. Frasca, S. P. Zafeiriou, and M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[16] J. Gao, J. Chen, Z. Li, and J. Zhang, "Ics-gnn: lightweight interactive community search via graph neural network," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 1006–1018, 2021.

[17] Y. Jiang, Y. Rong, H. Cheng, X. Huang, K. Zhao, and J. Huang, "Query driven-graph neural networks for community search: from non-attributed, attributed, to interactive attributed," *Proceedings of the VLDB Endowment*, vol. 15, no. 6, pp. 1243–1255, 2022.

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

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

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

[21] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. Yu, "Graph self-supervised learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[22] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[23] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[25] Y. Zhu *et al.*, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080.

[26] W. Zhang, Y. Shen, Y. Li, L. Chen, Z. Yang, and B. Cui, "Alg: fast and accurate active learning framework for graph convolutional networks," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2366–2374.

[27] J. Xia, L. Wu, J. Chen, B. Hu, and S. Z. Li, "Simgrace: A simple framework for graph contrastive learning without data augmentation," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1070–1079.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[29] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[30] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *Technologies*, vol. 9, no. 1, p. 2, 2020.

[31] H. Wang, J. Zhang, Q. Zhu, and W. Huang, "Augmentation-free graph contrastive learning," *arXiv preprint arXiv:2204.04874*, 2022.

[32] B. Perozzi *et al.*, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[33] N. Lee, J. Lee, and C. Park, "Augmentation-free self-supervised learning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7372–7380.

[34] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4116–4126.

[35] J. Yuan, H. Yu, M. Cao, M. Xu, J. Xie, and C. Wang, "Semi-supervised and self-supervised classification with multi-view graph neural networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2466–2476.

[36] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko, "Bootstrapped representation learning on graphs," in *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021.

[37] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[38] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160.

[39] T. Haveliwala, "Efficient computation of pagerank," Stanford, Tech. Rep., 1999.

[40] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "Infogcl: Information-aware graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30 414–30 425, 2021.

[41] Y. Zhang, H. Zhu, Z. Song, P. Koniusz, and I. King, "Costa: covariance-preserving feature augmentation for graph contrastive learning," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2524–2534.

[42] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 6827–6839, 2020.

[43] Y. Zhang, Q. Yao, L. Yue, X. Wu, Z. Zhang, Z. Lin, and Y. Zheng, "Emerging drug interaction prediction enabled by flow-based graph neural network with biomedical network," *Nature Computational Science*, 2023.

[44] X. Zhang, Y. Shen, and L. Chen, "Feature-oriented sampling for fast and scalable gnn training," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 723–732.

[45] Y. Zhang, Z. Zhou, Q. Yao, X. Chu, and B. Han, "Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3446–3457.

[46] W. Lin, F. He, F. Zhang, X. Cheng, and H. Cai, "Initialization for network embedding: A graph partition approach," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 367–374.

[47] D. Lin, S. Sun, J. Ding, X. Ke, H. Gu, X. Huang, C. Song, X. Zhang, L. Yi, J. Wen *et al.*, "Platogl: Effective and scalable deep graph learning system for graph-enhanced real-time recommendation," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 3302–3311.

[48] W. Lin, "Large-scale network embedding in apache spark," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3271–3279.

[49] W. Xiao, H. Zhao, V. W. Zheng, and Y. Song, "Vertex-reinforced random walk for network embedding," in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 595–603.

[50] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. L. Reutter, and J.-P. Silva, "The expressive power of graph neural networks as a query language," *ACM SIGMOD Record*, vol. 49, no. 2, pp. 6–17, 2020.

[51] Y. Zheng, H. Wang, Z. Wei, J. Liu, and S. Wang, "Instant graph neural networks for dynamic graphs," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22, New York, NY, USA, 2022, p. 2605–2615.

[52] H. Liu, S. Lu, X. Chen, and B. He, "G3: when graph neural networks meet parallel graph processing systems on gpus," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2813–2816, 2020.

[53] D. Yao, Y. Gu, G. Cong, H. Jin, and X. Lv, "Entity resolution with hierarchical graph attention networks," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 429–442.

[54] Y. Han, C. Chai, J. Liu, G. Li, C. Wei, and C. Zhan, "Dynamic materialized view management using graph neural network," 2023.

[55] H. Li and L. Chen, "Cache-based gnn system for dynamic graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 937–946.

[56] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

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

[58] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 519–527.

[59] H. Li and L. Chen, "Early: Efficient and reliable graph neural network for dynamic graphs," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–28, 2023.

[60] T. Chen, S. Bian, and Y. Sun, "Are powerful graph neural nets necessary? a dissection on graph classification," *arXiv preprint arXiv:1905.04579*, 2019.

[61] H. Wang, M. He, Z. Wei, S. Wang, Y. Yuan, X. Du, and J.-R. Wen, "Approximate graph propagation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1686–1696.

[62] R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. S. Bhowmick, "Scaling attributed network embedding to massive graphs," *Proceedings of the VLDB Endowment*, vol. 14, no. 1, pp. 37–49, 2020.

[63] X. Zhang, Y. Shen, Y. Shao, and L. Chen, "Ducati: A dual-cache training system for graph neural networks on giant graphs with the gpu," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–24, 2023.

[64] Y. Park, S. Min, and J. W. Lee, "Ginex: Ssd-enabled billion-scale graph neural network training on a single machine via provably optimal in-memory caching," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 2626–2639, 2022.

[65] W. Huang *et al.*, "Adaptive sampling towards fast graph representation learning," in *Advances in neural information processing systems*, 2018, pp. 4558–4567.

[66] N. Liao, D. Mo, S. Luo, X. Li, and P. Yin, "Scara: scalable graph neural networks with feature-oriented optimization," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 3240–3248, 2022.

[67] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Tackling over-smoothing for general graph convolutional networks," *arXiv preprint arXiv:2008.09864*, 2020.

[68] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, and C. S. Jensen, "Decoupled dynamic spatial-temporal graph neural network for traffic forecasting," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 2733–2746, 2022.

[69] S. Di and L. Chen, "Message function search for knowledge graph embedding," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2633–2644.

[70] Z. Wang, S. Di, and L. Chen, "A message passing neural network space for better capturing data-dependent receptive fields," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 2489–2501.

[71] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang, "Self-supervised learning on graphs: Deep insights and new direction," *arXiv preprint arXiv:2006.10141*, 2020.

[72] W. Jin, X. Liu, X. Zhao, Y. Ma, N. Shah, and J. Tang, "Automated self-supervised learning for graphs," in *10th International Conference on Learning Representations (ICLR 2022)*, 2022.

[73] Y. Wang, J. Zhang, H. Li, Y. Dong, H. Yin, C. Li, and H. Chen, "Clusterscl: cluster-aware supervised contrastive learning on graphs," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1611–1621.

[74] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, and B. Cui, "Contrastive learning for sequential recommendation," in *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 2022, pp. 1259–1273.

[75] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[76] Y. Mo, L. Peng, J. Xu, X. Shi, and X. Zhu, "Simple unsupervised graph representation learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7797–7805.

[77] S. Wang, Y. Tang, X. Xiao, Y. Yang, and Z. Li, "Hubppr: effective indexing for approximate personalized pagerank," *Proceedings of the VLDB Endowment*, vol. 10, no. 3, pp. 205–216, 2016.

[78] D. Lin, R. C.-W. Wong, M. Xie, and V. J. Wei, "Index-free approach with theoretical guarantee for efficient random walk with restart query," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 913–924.

[79] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[80] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li, "Self-supervised learning on graphs: Contrastive, generative, or predictive," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[81] H. Liu, S. Di, and L. Chen, "Incremental tabular learning on heterogeneous feature space," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–18, 2023.

[82] C. Chai, J. Liu, N. Tang, J. Fan, D. Miao, J. Wang, Y. Luo, and G. Li, "Goodcore: Data-effective and data-efficient machine learning through coreset selection over incomplete data," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–27, 2023.

[83] J. Wang, C. Chai, N. Tang, J. Liu, and G. Li, "Coresets over multiple tables for feature-rich and data-efficient machine learning," *Proceedings of the VLDB Endowment*, vol. 16, no. 1, pp. 64–76, 2022.

[84] C. Chai, J. Liu, N. Tang, G. Li, and Y. Luo, "Selective data acquisition in the wild for model charging," *Proceedings of the VLDB Endowment*, vol. 15, no. 7, pp. 1466–1478, 2022.

[85] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6950–6960.

[86] J. Huang, R. Huang, W. Liu, N. Freris, and H. Ding, "A novel sequential coreset method for gradient descent algorithms," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4412–4422.

[87] Y. Li, Y. Shen, and L. Chen, "Camel: Managing data for efficient stream learning," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1271–1285.

[88] B. Mirzasoleiman, K. Cao, and J. Leskovec, "Coresets for robust training of deep neural networks against noisy labels," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 465–11 477, 2020.

[89] C. Chai, J. Wang, Y. Luo, Z. Niu, and G. Li, "Data management for machine learning: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4646–4667, 2022.
[90] H. Li, S. Di, Z. Li, L. Chen, and J. Cao, "Black-box adversarial attack and defense on graph neural networks," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1017–1030.
[91] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.
[92] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.
[93] O. Bachem, M. Lucic, and S. Lattanzi, "One-shot coresets: The case of k-clustering," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 784–792.
[94] H. Li, S. DI, L. Chen, and X. Zhou. (2023) E^2GCL: Efficient and expressive contrastive learning on graph neural networks technique report. [Online]. Available: https://anonymous.4open.science/r/ICDE_E2GCL_Technique_Report/
[95] W.-L. Chiang, X. Liu, S. Si, Y. Li *et al.*, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.
[96] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.
[97] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *arXiv preprint arXiv:1811.05868*, 2018.
[98] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 121–12 132.
[99] S. Martello and P. Toth, "Approximation schemes for the subset-sum problem: Survey and experimental analysis," *European Journal of operational research*, vol. 22, no. 1, pp. 56–69, 1985.
[100] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019.
[101] A. Saxena and S. Iyengar, "Centrality measures in complex networks: A survey," *arXiv preprint arXiv:2011.07190*, 2020.
[102] K. Li, Y. Liu, X. Ao, and Q. He, "Revisiting graph adversarial attack and defense from a data distribution perspective," in *The Eleventh International Conference on Learning Representations*, 2022.
[103] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.
[104] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.
[105] A. Grover *et al.*, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.
[106] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
[107] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *International Conference on Learning Representations*, 2018.
[108] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *International Conference on Learning Representations*, 2018.
[109] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.
[110] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. [Online]. Available: www.graphlearning.io
[111] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.
[112] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.