Knowledge-Based Systems 299 (2024) 112112

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Graph Contrastive Multi-view Learning: A Pre-training Framework for Graph Classification

Michael Adjeisah [a,b], Xinzhong Zhu [b,c,*], Huiying Xu [b,c], Tewodros Alemu Ayall [d]

[a] National Centre for Computer Animation, Bournemouth University, Bournemouth, Pool, BH12 5BB, United Kingdom
[b] College of Computer Science and Technology, Zhejiang Normal University, Jinhua, Zhejiang, 321004, China
[c] Artificial Intelligence Research Institute of Beijing Geekplus Technology Co. Ltd., Beijing, 100101, China
[d] School of Natural and Computing Sciences & Interdisciplinary Centre for Data and AI, University of Aberdeen, Aberdeen, AB24 3UE, United Kingdom

ARTICLE INFO

Keywords:
Contrastive learning
Graph classification
Graph neural network
Multi-view representation learning
Pre-trained embeddings

ABSTRACT

Recent advancements in node and graph classification tasks can be attributed to the implementation of contrastive learning and similarity search. Despite considerable progress, these approaches present challenges. The integration of similarity search introduces an additional layer of complexity to the model. At the same time, applying contrastive learning to non-transferable domains or out-of-domain datasets results in less competitive outcomes. In this work, we propose maintaining domain specificity for these tasks, which has demonstrated the potential to improve performance by eliminating the need for additional similarity searches. We adopt a fraction of domain-specific datasets for pre-training purposes, generating augmented pairs that retain structural similarity to the original graph, thereby broadening the number of views. This strategy involves a comprehensive exploration of optimal augmentations to devise multi-view embeddings. An evaluation protocol, which focuses on error minimization, accuracy enhancement, and overfitting prevention, guides this process to learn inherent, transferable structural representations that span diverse datasets. We combine pre-trained embeddings and the source graph as a beneficial input, leveraging local and global graph information to enrich downstream tasks. Furthermore, to maximize the utility of negative samples in contrastive learning, we extend the training mechanism during the pre-training stage. Our method consistently outperforms comparative baseline approaches in comprehensive experiments conducted on benchmark graph datasets of varying sizes and characteristics, establishing new state-of-the-art results.

1. Introduction

The benefit of graph data is the ability to model complex data relationships that are deeply structured to suit inferences about indirect facts and indirectly related information. Edges are just as influential and complex as vertices/nodes, making graphical representation learning emerge as a practical approach to graph data exploration in recent years. It has been significant in the transition of graph vertices, edges, or subgraphs to low-dimensional dense embeddings [1] that maintain the structural features and attributes of the graphs. The retained embeddings can then serve as input to standard machine learning models for downstream tasks. Therefore, graph data has become increasingly widespread based on the neighborhood aggregation task [2], inspired by recent advances in deep learning [3,4]. Furthermore, multiple works in graph-based studies use GNN variants to predict linkages, nodes, and graphs for link prediction [5], node classification [6–9], graph classification [10–12], and have shown remarkable state-of-the-art performance.

However, GNN research has faced obstacles as a result of the very limited data sizes of traditional graph datasets after several years of development. Such impediments include overfitting and over-smoothing [13–15], unrealistic and arbitrary data splitting, non-rigorous evaluation metrics, and, in general, the neglect of validation sets [16–18]. Recalling the standard GCN for node classification reveals that they are usually superficial, limiting the number of layers to 2 [7]. Therefore, an attempt to go deeper is still ingrained in these impediments. Most recently, several works have proposed research on the development of deep GNN models [9,13,19,20] for node classification, ranging from node sampling [21–23], layer sampling [8,24,25], and subgraph sampling [26,27]. Nevertheless, such research still falls in graphs' highly limited data sizes, leading to graph data augmentation such as adversarial perturbation [28–31] and node and structural perturbation [1,2,32], all other methods. Multi-view representation learning is well established for deep neural networks, as the operation

* Corresponding author at: College of Computer Science and Technology, Zhejiang Normal University, Jinhua, Zhejiang, 321004, China.
E-mail addresses: madjeisah@bournemouth.ac.uk (M. Adjeisah), zxz@zjnu.edu.cn (X. Zhu).

https://doi.org/10.1016/j.knosys.2024.112112
Received 27 October 2022; Received in revised form 16 March 2023; Accepted 9 June 2024
Available online 11 June 2024

is frequently used in Computer Vision (CV) and Natural Language Processing (NLP) [33]. Therefore, the critical topic is that the operation should be adopted in the GNN structure to improve performance. However, in contrast to other data, where the structure is encoded by position, the structure of graphs is encoded by intermittent node connectivity [7] and is usually characterized by a non-euclidean space. Thus, various nodes are influenced by each other due to the links built by the edges. Consequently, it produces unwanted outcomes, such as intractable augmentation volumes and somewhat modified ground-truth labels [34].

Various augmentation strategies have emerged for multi-view construction; however, finding a suitable method for learning better representations of various graphs is still under exploration. The hypothesis is that the most beneficial pairs of augmentation types can be data-specific [35]. For example, for node classification tasks, adding nodes poses challenges in labeling and assigning features and connectivity of new nodes, while removing nodes reduces the data available. It is also well known that some augmentation strategies benefit social networks but damage some biochemical molecules [35].

Present work. Inspired by the above-listed, and graph augmentation strategies, we study the potential of contrasting multi-views to enhance performance. We generate relatively more views and collectively search for the best-view representation for various datasets. The idea is to perturb the graph structure to form positive pair embeddings given various input graphs. In particular, the structure of a paired augmented is similar to the original graph, increasing the view. For pre-training tasks, Hu et al. [36] performed a similarity search in data that are not transferable to out-of-domain data, nevertheless, it yielded low competitive results in various tasks. GraphCL [2] and MVGRL [37] are similar to our approach but different in implementation. Instead of pre-training GNN on different datasets [2,38] from the dataset for classification to obtain correlated views, we used a fraction of the same dataset during pre-training.

Also, MVGRL and other works assume that a particular augmentation pair works for all datasets. In our case, we collectively search for the best-view representation to learn the intrinsic and transferable structural representations on a particular dataset. Furthermore, the MVGRL mechanism transforms a sample of the source graph into a correlated view based on two dedicated GNNs, limiting the default view to two. Our work is not limited to the number of view generation. Nevertheless, significant view representations are data-specific, hence; the search for the best duo of views during pre-training. We present multi-view construction based on self-supervised learning for graph classification tasks. It composes different augmentations to generate additional structural views of a sample graph that engage contrastive loss to estimate mutual information (MI) to group the embeddings of the positive pair while simultaneously pushing the negative ones away [39]. The pre-trained embeddings and the source graph are finally fed into GNN variants for downstream tasks, as shown in Fig. 1.

The major contributions of this work are as follows:

- We present a novel **G**raph **C**ontrastive multi-view learning using a **P**re-training framework (GCP). The method takes the source graph $\mathcal{G}_s$ and the pre-trained embeddings ($\mathcal{G}_t^{(a)}$ and $\mathcal{G}_t^{(b)}$) as input to evaluate a downstream task..
- We hypothesize that keeping the task domain-specific alleviates the extra similarity search in the non-transferable to the out-of-domain tasks and can benefit downstream tasks.
- With the prolonged training mechanism for more negative samples of each sample, our approach relatively balances the small information-to-noise ratio in high-order neighbors for a more varied split of total datasets, avoiding over-smoothing.
- Extensive experiments on benchmark graph datasets of different sizes with distinct characteristics show that our approach outperforms the compared baselines. The source code is available online[1] for reproducibility.

[1] https://github.com/Madjeisah/GCP

The rest of the paper is as follows. Section 1 introduces our work and problem statements and highlights the main contributions. Various related works on Graph Data Augmentation (GDA) and graph contrastive learning and their core functions in GNN are thoroughly discussed in Section 2. While Section 3 focuses on the GCP framework, Section 4 presents various graph datasets, experimental analysis, and visualization of results. Finally, we conclude this work in Section 5.

2. Related works

2.1. Graph data augmentation

Depending on the difference in the various graph augmentation strategies, graph augmentation can be divided mainly into two categories [40], including the topology-based method (GCA [1], GAUG-O [6], AdaEdge [19], DropEdge [20]) and the feature-based augmentation method (NodeAug [34], FLAG [41], G-GNN [42], LA-GNN [43]).

The former usually perturbs **A** to generate different graph structures. For example, AdaEdge Adaptive Edge Optimization AdaEdge [19] adapts the graph topology by iteratively adding the intra-class edges and removing the inter-class edges during training. DropEdge [20] randomly removes a substantial proportion of the edges of the input graph by generating different random distorted copies of the original graph. GAUG-O [6] engages in a dual step, which is to obtain edge [44] probabilities for all potential and existing edges via the edge predictor function followed by adding/removing new or existing edges utilizing the predicted edge probabilities to construct a modified graph to serve as input to a GNN node classifier. GCA [1] proposes a joint adaptive GDA strategy at the topology and node attribute level, i.e., edge disturbance and attribute masking by specifying essential edges and feature dimensions through prominent criteria.

The latter focuses on exploiting the influential nodes **X** by placing a probability on the trivial ones to improve classification performance. G-GNN [42], which preserves global information learned, is known for its performance, easy implementation, and theoretical understanding. It constructs the global information as the global structure and global attribute features to each node and pre-trained them for a final parallel GNN-based model to learn distinct characteristics from the pre-trained and original features. Local augmentation for GNN (LA-GNN) [43] preserves the locality of node representations by their subgraph structure. Node-Parallel Augmentation (NodeAug) [34] offered three distinct augmentation methods by (1) adjusting the node attributes, (2) the graph structure, and (3) introducing a subgraph mini-batch training for resourceful execution. The GraphANGEL [45] framework presents an adaptive and structure-aware essential intrinsic connection-based subgraph sampling between random walk and graph convolution. The approach exploits the unique feature of random-walk that combines time and diverse structural efforts of the nodes to propose a lightweight component to flexibly assess the appropriate depth of neighborhood exploration for the target nodes. Observably, the technique adaptively embeds the structural prerogative information of a node and its vital neighborhood simultaneously for more acceptable structure-aware graph representation learning. Free Large-scale Adversarial Augmentation on Graphs (FLAG) [41], Graph Adversarial Training (GraphAT) [46], and Batch Virtual Adversarial Training (BVAT) [31] have recently performed remarkably on graph data.

2.2. Contrastive multi-view learning

Most recent research has focused on finding appropriate augmentation techniques to understand adequate representations for various graph applications through contrastive learning [47–49]. However, the unfastened query is how to develop a graph contrastive learning model from scratch for graph representation tasks [50]. Qiu et al. [38] proposed a self-supervised GNN pre-training framework that captures the topological properties of universal networks across multiple networks.

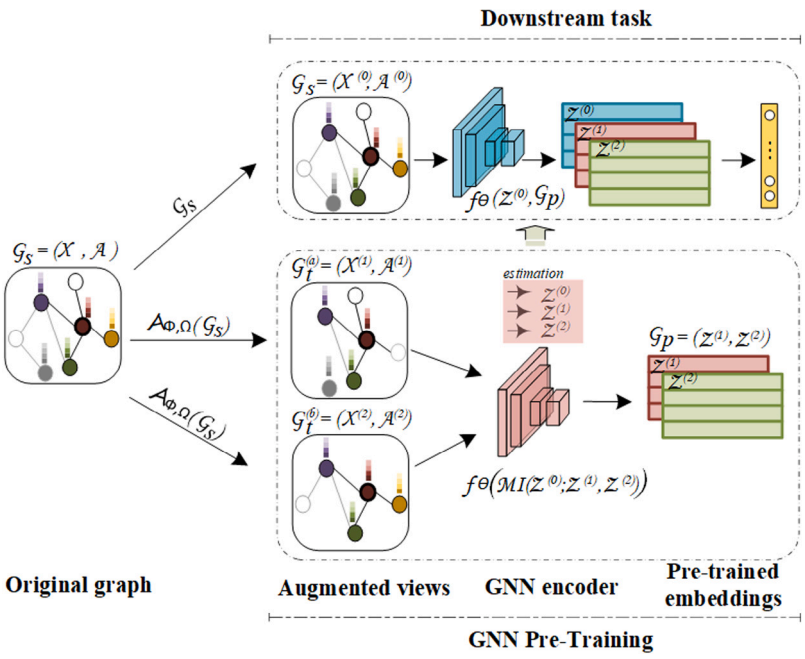

Fig. 1. The proposed model for GCP. We engage a GNN encoder to generate pre-trained embeddings by composing different augmentations Φ and Ω for additional structural views $\mathcal{G}_t^{(a)}$ and $\mathcal{G}_t^{(b)}$ from a source graph $\mathcal{G}_s$. A contrastive loss (infoNCE) estimates the mutual information between the positive information from augmented data and all the negative information in the mini-batch. The positive embeddings are paired into a single data object and pre-trained $\mathcal{G}_p$. Finally, the source graph $\mathcal{G}_s$ and the pre-trained embeddings $\mathcal{G}_p$ are engaged for downstream tasks.

The approach engages pre-training tasks such as random subgraph sampling instance discrimination in and across networks, leveraging graph contrastive learning to empower GNN to learn the intrinsic and transferable structural representations. The authors hypothesize that representing universal and transferable structural patterns of graphs across networks can achieve competitive and somewhat better performance than its task-specific and trained-from-scratch counterparts. You et al. [2] systematically study the various combinations of graph augmentation for GNN pre-training to address data heterogeneity challenges by obtaining correlated views. For the representation invariant to specialized perturbation and comprehended for various graph-structured data [51], the method leverages contrastive graph learning for pre-training. The approach examines whether and when the application of different data augmentation aids contrastive learning [52] with a detailed observation summary and derived insights.

Hassani & Khasahmadi [37] introduce a self-supervised approach to learning representations at the node and graph level by contrasting the structural views of the graphs. The work mainly focused on engaging diffusion and node sampling augmentation strategies to learn better representations. They established that feature-space augmentations could be problematic as many benchmarks do not carry initial node features, degrading performance. DSGC [53] took advantage of hyperbolic and Euclidean space to perform graph contrastive learning among views generated in both spaces to represent graph data in the embedding spaces. SimGRACE [54] observed that graph data can preserve their semantics well during encoder perturbations; therefore, the perturbed version of an input graph is used as two encoders to generate two correlated views.

The node attribute and topology contain extensive and various information about the structure, respectively, and are equally paramount for definitive performance in many circumstances. Furthermore, the negotiation between augmented information sources in various works benefits intrinsic data structure learning. However, finding the right augmentation strategies for better multi-view representation learning in various graph applications is pivotal in GNN. Therefore, applying appropriate augmentation for best-view estimation and the original graph can infuse the corresponding priors on the data distribution, aiding the downstream performance. Also, regarding the significance of negative samples in contrastive learning, our approach prolongs the training process with more epochs for more negative samples for each sample during pre-training to make the split of total datasets more varied. More significantly, such operations enhance performance, provide better generalizable features, and avoid over-fitting. While MVGR [37] engaged in an end-to-end approach with two dedicated graph encoders, one for each view, limiting the views to two by default, we generated multiple views with various augmentation techniques using the upstream (pre-train) method. Finally, in contrast, we eliminate the extra effort for similarity search [38] by using a fraction of the dataset during pre-training for primary structural similarity and transferability.

2.3. Graph pre-training

Some machine learning applications require a model to accurately make predictions on test sets that are different from the training distribution in an insufficient task-specific domain. An effective conceptualization of this situation is to pre-train a model on abundant task-related data and then fine-tune it on a downstream task of interest. Pre-training has proven to be an important infrastructure capability that can support many different use cases in CV, and NLP, such as classification, and generation. It has recently been adopted to enhance the power of GNNs. Many self-supervised pretext tasks seldom notice the integral training objective between the pretext and downstream tasks because of the universal encoding of graph representation. Therefore, fine-tuning for adapting the pre-trained model to downstream problems is often cost-effective.

Hu et al. [36], developed strategies using self-supervised methods to train an expressive GNN at the level of the respective nodes and the entire graphs to simultaneously learn transferable local and global knowledge. GPPT [55] adopted the masked edge prediction to first pre-train GNNs and proposed a graph that prompts the functionality to modify the standalone node into a token pair, and perceptually reformulate the downstream node classification task as edge prediction. Ziniu Hu et al. [56], factorized the probability of graph generation

into attribute and edge generation by modeling both components to capture the inherent habituation between node attributes and graph structure during pre-training. To capture intrinsic patterns underlying the graph structures, Pengyong Li et al. [57] proposed an effective self-supervised representation [58] strategy to expressly pre-trained GNN at both the node level and the graph level. Pairwise Half-Graph Discrimination (PHD) can overcome the challenges of transferability and can generalize well to different GNN tasks. The L2PGNN [59] approach attempts to learn how to fine-tune during the pre-training process with transferable prior knowledge by encoding both local and global information in the prior using a dual adaptation mechanism at node and graph levels.

3. Methodology

3.1. Problem formulation

Given a source graph $\mathcal{G}_s$ and a target graph $\mathcal{G}_t$, the graph pair aims to store multiple graphs in a single data object to ensure the correct batching behavior across all those graphs. Similarly, the approach can be extended to the composition of different augmentations with the GNN encoder from the original graph and pair them into a single data object. Generally, assume $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph, with vertices or node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and edge set $\mathcal{E}$. The adjacency matrix of $\mathcal{G}$ is the $\mathbf{A} \in \mathbb{R}^{n \times n}$ matrix such that:

$$\mathbf{A}_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in \mathcal{E}; \\ 0, & otherwise. \end{cases} \tag{1}$$

Thus binary matrix $\mathbf{A}$ in a graph with edges of n nodes is such that, if there is an edge between node i and j, then $\mathbf{A}_{ij} = 1$, else $\mathbf{A}_{ij} = 0$. Hence $\mathbf{A}$ is also written as $\mathbf{A} = [\mathbf{A}_{ij}]$. The attribute matrix $\mathbf{X}$ is represented as $\mathbf{X} \in \mathbb{R}^{n \times q}$, where the ith row denotes v_i's attributes and q denotes the total amount of the attributes.

One exemplary procedure in this work is that we generate multi-views on a fraction of a given source graph with self-supervised learning composing different augmentation strategies, e.g., edge perturbation and diffusion. We then estimated the mutual information between the source and augmented graphs. The positive samples of the augmented graph are then stored/paired in a single data object to ensure the correct batching behavior across the graphs and finally pre-trained. Finally, we engage the pre-trained embeddings and the source graph as input, whereby a classifier head is fine-tuned for downstream task evaluation. Notably, the model constitutes two modules; the lower module (pre-training stage) and the upper module (downstream stage).

3.2. Detailed approach

This section details our proposed GCP model as shown in Fig. 1. Specifically, we illustrate the modules of GCP by explaining the multi-view establishment for the graph encoder, presenting the self-supervised contrastive learning, elaborating on how to get the positive and negative samples, and finally a description of the overall training procedure.

3.2.1. Generating multi-view

Assuming that graph $\mathcal{G}_s$ is the source graph with label $\mathcal{Y}_L$, we can generate views via augmentation strategies to get $\mathcal{G}_t^{(a)}$ and $\mathcal{G}_t^{(b)}$, the first and second target augmented graph to label $\mathcal{Y}_L$, respectively with;

$$\mathcal{A}_{\Phi,\Omega}(\mathcal{G}_s) := (\mathcal{A}_{\Phi,\Omega}(\mathcal{G}_t^{(a)}), \mathcal{A}_{\Phi,\Omega}(\mathcal{G}_t^{(b)})), \tag{2}$$

where Φ and Ω are some augmentation parameters. The approach is similar to other works [2,51,60–62], however, while the other works limit the number of augmentations, our model allows for various choices of augmentation strategy without restrictions. Similarly, the encoder framework allows for various choices of the GNN architecture using the Deep InfoMax approach [63].

Algorithm 1: The algorithm for multi-view generation

Input: The source graph $\mathcal{G}_s$
Params: Augmentation parameters $\mathcal{A}_{\Phi,\Omega}(\cdot)$, epoch α, pre-training sample size β, InfoNCE loss.
Output: $\mathcal{G}_p$
Initialize $\mathcal{A}_{\Phi,\Omega}$, Θ
for *each* α **do**
  if β *is set* **then**
    // Generate views via augmentation
    $\mathcal{G}_t^{(a)} \leftarrow \mathcal{A}_{\Phi,\Omega}(\mathcal{G}_s)$ // Eq. (2)
    $\mathcal{G}_t^{(b)} \leftarrow \mathcal{A}_{\Phi,\Omega}(\mathcal{G}_s)$ // Eq. (2)
    // View estimation
    Get READOUT anchor of $\mathcal{G}_s$
    Get READOUT positives of $\mathcal{G}_t^{(a)}$, and $\mathcal{G}_t^{(b)}$
    Estimate $\{\mathcal{G}_s, \mathcal{G}_t^{(a)}, \mathcal{G}_t^{(b)}\}$ as $\{\mathcal{Z}^{(0)}, \mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}\}$
    // Get views after estimation
    Get $\{\mathcal{Z}^{(0)}, \mathcal{Z}^{(1)}, \mathcal{Z}^{(2)}\}$ after view estimation
    // Compute Mutual Info (MI)
    Compute MI for estimated views
    Update the InfoNCE objective function
    Repeat the process until convergence
    Pair views $\mathcal{Z}^{(1)}$ and $\mathcal{Z}^{(2)}$ into $\mathcal{G}_p$
  end
end
return $\mathcal{G}_p$

3.2.2. Multi-view estimation

The GNN encoder aims to find the density ratio to estimate the mutual information between these graphs to generate pre-train embeddings. With a non-linear learned mapping, the aim is to obtain the READOUT anchor and positive information to maximally preserve the mutual information between graphs $\mathcal{G}_s$, and $\mathcal{G}_t^{(a)}$, and $\mathcal{G}_s$ and $\mathcal{G}_t^{(b)}$. Notably, the GNN READOUT operation aggregates the features of all the nodes in the graph to produce a single, fixed-size vector that summarizes the entire graph. This vector serves as a fixed point that captures the overall structure of the graph. To obtain such similarities between these graphs, we use the function $I(\mathcal{G}_s; \mathcal{G}_t^{(a)}, \mathcal{G}_t^{(b)})$ which allows us to detect the number of similar points employing conditional and joint probability as:

$$I(\mathcal{G}_s; \mathcal{G}_t^{(a)}, \mathcal{G}_t^{(b)}) = \sum_{t,t',s} p(t,t',s) \log \frac{p(t,t'|s)}{p(t,t')}, \tag{3}$$

where $s \in \mathcal{G}_s$, $t \in \mathcal{G}_t$ and $t' \in \mathcal{G}_t^{(a)}$, respectively, and $p(\cdot)$ is a conditional probability function. By maximizing mutual information between encoder representations, we extract the underlying latent variables that the inputs have in common [64]. To this end, we find the density ratio that preserves mutual information between s and between t, and between s and t' and concatenates them. Approximately, we can define

$$f(s,t,t') = \frac{p(t,t'|s)}{p(t,t')}, \tag{4}$$

where f denotes the density ratio. To efficiently estimate mutual information between views, we train the GNN encoder using the InfoNCE objective function.

3.3. View pairing

During MI estimation, the anchor gap and the positive have the same identity information, but the negative has different information. The contrastive loss $\mathcal{L}_{NCE}$ function minimizes the anchor and positive

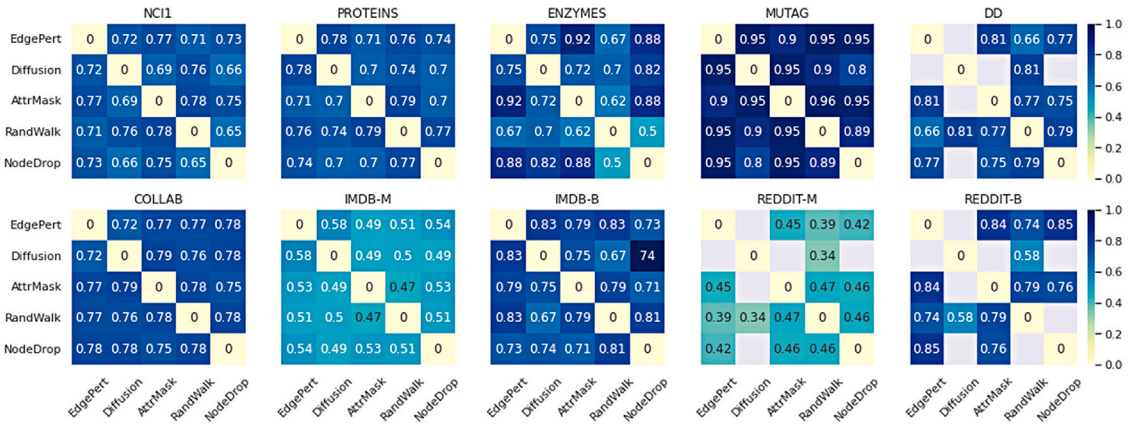

Fig. 2. Accuracy of different augmentation pairs for multi-views on the biochemical molecules (**Top**) and social network (**Bottom**) datasets. Warmer colors indicates best accuracy and empty space signifies out-of-memory (OOM) resources.

gap while maximizing the gap between the anchor and negative ones. Thus, multiple views are generated, of which the one identical to the given data point constitutes the positive part, and other views far from the data point are considered negative. While the contrastive loss function forces the embeddings of positive views to cluster within a high-order neighbor, the model simultaneously forces the embeddings of negative views away [65]. For simplicity, we adopt a utility function[2] to return a pair of positive samples from the augmented graphs $\mathcal{G}_t^{(a)}$ and $\mathcal{G}_t^{(b)}$. In particular, the augmented graph $\mathcal{G}_t^{(a)}$ and $\mathcal{G}_t^{(b)}$ are passed through the same encoder, resulting in two embedding matrices $\mathcal{Z}^{(1)}$ and $\mathcal{Z}^{(2)}$ that are adaptively fused into $\mathcal{G}_p$. The algorithm for our multiview generation is shown in Algorithm 1.

To understand which views contain augmented-rich information, we pair augmentation methods such as edge perturbation, diffusion, attribute masking, node dropping, and random walk subgraph extraction for view generation. We then engaged a standard evaluation protocol of the embeddings concerning model training uncertainties based on accuracy, error rate, over-fitting, over-smoothing, and computation for quality representation learning. We used parameters such as the pre-training sample size β within the range of 10, 20, 30, and 50% with augmentation parameters Φ and Ω on several runs. Eventually, GCP augments graph data with evaluation protocol as guidance, preserving semantics during augmentation and achieving better generalization because of the search-based evaluation protocol. Detailed experimental results are shown in Section 4.4.

4. Experimental results

This section begins by presenting the datasets and the experimental setup utilized in this work. We then conduct extensive experiments on graph classification tasks to evaluate the preeminence of the proposed GCP and compared it with previous state-of-the-art works to exhibit the model's effectiveness. Finally, we rationalize performance investigations to further verify the significance of composing views relatively more than the original graph and its augmentation.

The aim of this work is to answer the following research questions (RQ) to instantiate each component in GCP:

- **RQ1**: Can pre-trained embeddings preserve local and global information to improve structural similarity and transferability?
- **RQ2**: Considering the importance of negative samples in contrastive learning, how effective is the prolonged training in GCP?
- **RQ3**: How significant are the GCP default views, and does increasing the views beyond that introduce any helpful information?
- **RQ4**: Can the model extend its layer beyond the usual 2-3 layers without over-smoothing?
- **RQ5**: How efficient is GCP in terms of training time and memory cost?

Table 1
Statistics of graph classification benchmarks.

		Graphs	Nodes	Edges	Classes
Biochem.	NCI1	4110	29.87	32.30	2
	PROTEINS	1113	39.06	72.82	3
	D&D	1178	284.32	715.66	89
	ENZYMES	600	32.63	64.14	3
	MUTAG	188	17.93	19.79	2
Social networks	COLLAB	5000	74.49	2457.78	3
	IMDB-M	1500	13.00	65.94	3
	IMDB-B	1000	19.77	96.53	2
	REDDIT-M	4999	508.82	594.87	5
	REDDIT-B	2000	429.63	497.75	2

4.1. Datasets and experimental setup

We use 10 publicly available graph datasets. NCI1 [66], PROTEINS [67], D&D [68], MUTAG [69], which contains mutagenic compounds, and ENZYMES [70] are biochemical molecular graph networks. For graph social networks, we use; COLLAB, IMDB-BINARY, and IMDB-MULTI, connecting actors/actresses (nodes) based on movie appearances (edges), REDDIT-BINARY and REDDIT-5K, connecting users (nodes) through responses (edges) in Reddit online discussions [71]. Dataset statistics are summarized in Table 1.

4.2. Training and evaluation protocol

The ultimate idea is that the GNN pre-training concern is to learn an operation that maps a vertex to a low-dimensional feature vector, making it similar in structure and transferability. Thus, a function that can map vertices with similar network topologies close to each other in the vector space and compatible with vertices and graphs. [38].

We implemented our GCP framework using PyTorch[3], PyTorch Geometric[4], and Deep Graph Library[5] packages in all experiments. We

[2] https://pytorch-geometric.readthedocs.io/en/latest/notes/batching.html

[3] https://pytorch.org/tutorials/beginner/basics/autogradqs_tutorial.html
[4] https://github.com/pyg-team/pytorch_geometric
[5] https://www.dgl.ai/

Table 2
Test accuracy on graph classification task. We performed 10 runs to evaluate the performance of the model and report the mean accuracy and standard deviation. The best performances are highlighted in **bold** and underlined as the second best.

	NCI1	PROTEINS	D&D	ENZYMES	MUTAG
GCN	69.8 ± 2.2	72.8 ± 3.7	71.4 ± 4.5	65.2 ± 6.4	85.6 ± 5.8
DGCNN	76.4 ± 1.7	72.9 ± 3.5	76.6 ± 4.3	38.9 ± 5.7	–
DiffPool	76.9 ± 1.9	73.7 ± 3.5	75.0 ± 3.5	59.5 ± 5.6	–
ECC	76.2 ± 1.4	72.3 ± 3.4	72.6 ± 4.1	29.5 ± 8.2	–
GIN	80.0 ± 1.4	73.3 ± 4.0	75.3 ± 2.9	59.6 ± 4.5	–
GraphSAGE	76.0 ± 1.8	73.0 ± 4.5	72.9 ± 2.0	58.2 ± 6.0	–
InfoGraph	76.2 ± 1.1	74.44 ± 0.3	72.85 ± 1.8	–	89.01 ± 1.1
GraphCL	77.87 ± 0.4	74.39 ± 0.5	78.62 ± 0.4	–	86.90 ± 1.3
GCC	–	–	–	–	–
MVGRL	–	–	–	–	89.7 ± 1.1
JOAO	78.07 ± 0.47	74.55 ± 0.41	77.40 ± 1.15	–	87.67 ± 0.79
Mixup	81.60 ± 0.41	74.03 ± 0.51	76.48 ± 0.58	–	–
SimGRACE	74.0 ± 1.9	74.3 ± 3.5	76.8 ± 2.9	–	–
TGNN	–	71.0 ± 0.7	70.8 ± 0.9	–	–
GCP+GCN	81.10 ± 0.1	**80.56 ± 2.31**	82.43 ± 0.25	58.33 ± 0.28	87.02 ± 0.04
GCP+GraphSAGE	**82.86 ± 1.1**	76.89 ± 1.36	**85.63 ± 1.5**	**67.41 ± 0.17**	**89.76 ± 0.52**
GCP+GIN	77.16 ± 0.08	72.26 ± 0.21	78.83 ± 0.21	57.10 ± 0.48	84.22 ± 0.51
	COLLAB	IMDB-M	IMDB-B	REDDIT-M5K	REDDIT-B
GCN	70.2 ± 1.5	51.9 ± 3.8	74.0 ± 3.4	52.2 ± 1.5	50.0 ± 0.0
DGCNN	71.2 ± 1.9	45.6 ± 3.4	69.2 ± 3.0	49.2 ± 1.2	87.8 ± 2.5
DiffPool	68.9 ± 2.0	45.6 ± 3.4	68.4 ± 3.3	53.8 ± 1.4	89.1 ± 1.6
ECC	OOM	43.5 ± 3.1	67.7 ± 2.8	OOM	OOM
GIN	75.6 ± 2.3	48.5 ± 3.3	71.2 ± 3.9	56.1 ± 1.7	89.9 ± 1.9
GraphSAGE	73.9 ± 1.7	47.6 ± 3.5	68.8 ± 4.5	50.0 ± 1.3	84.3 ± 1.9
InfoGraph	70.65 ± 1.1	–	73.0 ± 0.8	53.46 ± 1.0	82.5 ± 1.4
GraphCL	71.4 ± 1.1	–	71.14 ± 0.4	55.9 ± 0.3	86.90 ± 1.3
GCC	81	52	75	53.0	87.8
MVGRL	–	51.2 ± 0.5	74.2 ± 0.7	–	84.5 ± 0.6
JOAO	69.33 ± 0.36	–	70.83 ± 0.25	56.03 ± 0.27	86.42 ± 1.45
Mixup	77.0 ± 2.2	49.9 ± 3.2	–	57.8 ± 1.7	–
SimGRACE	74.74 ± 0.28	–	–	53.97 ± 0.64	88.86 ± 0.62
TGNN	67.7 ± 0.4	42.9 ± 0.8	72.8 ± 1.7	43.8 ± 1.0	76.3 ± 1.3
GCP+GCN	78.34 ± 0.07	57.48 ± 3.10	**76.41 ± 2.31**	54.2 ± 0.04	86.7 ± 0.02
GCP+GraphSAGE	**83.20 ± 0.21**	**59.48 ± 0.31**	76.25 ± 0.93	54.2 ± 0.24	89.12 ± 2.07
GCP+GIN	75.8 ± 0.06	53.56 ± 0.03	71.86 ± 3.1	**57.86 ± 2.3**	**89.97 ± 2.1**

utilized the various augmentation techniques and adopted a utility function to return a pair of graphs to ensure correct batching behavior and pre-train them on a GeForce RTX 3090 GPU with 24 GB memory. The dataset is divided into 70%, 20%, and 10% for the training, validation, and test set, respectively.

4.2.1. Pre-training

Our goal of pre-training is to generate additional views based on the learned knowledge, given a fraction of the input datasets, to facilitate downstream tasks. Specifically, we first pre-train a GNN and use the pre-trained embeddings and the original graph as input to initialize models for downstream tasks. We used 20% of the train set during pre-training and specified 3 layers for the graph encoder for both training stages. At the same time, an ablation study of different data percentages and layers was conducted. We train the model with 1000 epochs, InfoNCE [64] loss to efficiently estimate mutual information between views, Adam optimizer, a learning rate of 0.001, and a batch size of 64 epochs, as shown in Table Table 3. Engaging data augmentation techniques such as diffusion and subgraph, we adopt a GNN variant for our encoder model to obtain multiple views on all datasets. We randomly select a subset of the data as the validation set.

4.2.2. Fine-tuning

Following GCC's full fine-tuning strategy, we use the pre-trained embeddings and the original graph as input, and a classifier head is then fine-tuned for downstream task evaluation. We engage similar parameters during pre-training except for Cross-Entropy loss since it is a classification task with 200 epochs. To alleviate the adverse influence of randomness, we replicate each experiment 10 times and report the average values and the corresponding standard deviations on the test set in Table Table 2. Detailed experimental results and analysis are shown in Section 4.4.

Table 3
Pre-training hyper-parameter settings: Loss ($\mathcal{L}$), Epoch (Ep), Optimizer (Opt), Learning rate (Lr), Batch size (B_{size}), Feature dimension (F_{dim}) and number of layers (#L) The default number of layers is set to 3; however, the best performance of each dataset is achieved based on the depth (number of layers) of the GCP model architecture as shown in Table 4.

Hyper-parameters settings		
	Pre-training	Classification
$\mathcal{L}$	InfoNCE	Cross-Entropy
Ep	1000	200
Opt	Adam	
Lr	0.001	
B_{size}	64	
F_{dim}	128	
#L	3	

4.3. Compared methods

We compared the GCP method with semi-supervised, unsupervised, and self-supervised methods including GCN [7], Deep Graph Convolutional Neural Network (DGCNN) [72], DiffPool [10], Edge-Conditioned Convolution (ECC) [73], Graph Isomorphism Network (GIN) [74], GraphSAGE [22], Infograph [35], GraphCL [2], GCC [38], MVGRL [37] JOint Augmentation Optimization (JOAO) [75], Mixup [76], SimGRACE [54] and TGNN [12].

4.4. Results and analysis

The optimal combinations of augmentation types may vary depending on the specific dataset. Consequently, we conducted preliminary experiments to generate additional augmentations and systematically identify each dataset's most effective view representations. To achieve this, we created a heat map reflecting accurate results from the initial runs across all datasets. This approach provides an informative overview of which views encapsulate augmentation-rich information for each dataset. The best are selected for final execution, as shown in Figure Fig. 2. After different runs, we observed that the position of the augmentation method had no effect. Therefore, the results in the lower half are similar to those in the upper half. Identical augmentations are not considered for this work (see Section Section 4.5.4), and the empty space means out-of-memory (OOM) resources. Clearly, the experimental analysis answers the research question (**RQ1**).

Table 2 presents the accuracy performance of GCP and other related methods in ten graph classification datasets. The results of other methods from GCN to GraphSAGE are directly copied from Federico Errica et al. [77], and the rest are from the corresponding works. The results show that our GCP methods perform much better than most compared methods due to the appropriate augmentation search for multiple view generation. In general, attribute masking is performed across various datasets. In particular, applying attribute masking, a feature-space augmentation strategy, achieves better performance in denser graphs [2]. For example, the accuracy of NCI1, PROTEINS, and MUTAG is the result of combining the random walk and node attribute masking from the initial experiment in Fig. 2 as such views show superior performance. Similarly, edge perturbation combined with node attribute masking and diffusion paired with random walk recorded the highest score for the ENZYMES and D&D datasets. Regardless, such views combination runs to overfitting on the corresponding dataset. With random-walk and node attribute masking, the GCP model can combat over-fitting.

Likewise, the initial experiment shows that the random-walk and node attribute masking technique registered the highest accuracy for COLLAB but ran into over-fitting, similar to previous cases. Engaging edge perturbation and node attribute masking combat the over-fitting issue. Overall, edge perturbation and node attribute masking work satisfactorily for all social network datasets to generate the best-view representation.

An observation is that diffusion paired with other methods for view generation is computationally expensive, mainly when applied to a large dataset that leads to OOM resources, as shown in Fig. 2. For REDDIT-M, REDDIT-R, and D&D datasets, we can see that diffusion paired with edge perturbation, node attribute masking, or node dropping runs out of memory on GeForce RTX 3090 GPU. Also, node dropping and subgraph on REDDIT-R. Therefore, the introduction of a mini-batch subgraph training for creative execution in the NodeAug model [34] is of great interest. Notably, graph mini-batch algorithms are vital in training GNNs on large-scale datasets. Note that even though the initial experiment serves as a guide for the GCP model, we also pay attention to uncertainties such as over-fitting. Thus, selecting the best-view representation depends on accuracy and model performance in combatting over-fitting from initial runs. Furthermore, the depth of our model architecture as shown in Table 4 plays a vital role in improving the accuracy performance of various datasets. See Section 4.5.5 for a detailed analysis.

4.5. Ablation study

4.5.1. Hyperparameter sensitivity

The essential hyperparameter for the GCP setup is *lr*. In general, it controls the model's response to the estimated error per model weight update. Although optimization like the Adams optimizer has a theoretical guarantee of convergence in many optimization problems, it remains an open challenge in contrastive learning. Especially, the considerable beneficial pairs of augmentation varieties can be data-specific. Therefore, we search for a hyperparameter (specifically *lr*) that functions in various datasets. In our experiments, we investigate the effects of *lr* on model performance. We performed a grid search on *lr*s 1e-1, 1e-2, and 1e-3 with 100 epochs for pre-training. Fig. 3 shows the behavior changes for *lr*s 1e-1 and 1e-2, and their inability to learn in the NCI1 and PROTEINS datasets. Therefore, the configuration of the model indicates that a moderate *lr* of 1e-3 results in a satisfactory level of empirical convergence across various datasets.

Table 4
Depth of **GCP model architecture** based on the number of layers (#L) in improving the accuracy performance on various datasets.

#L	Three-Layer	Four-Layer	Five-Layer
Datasets	IMDB-B REDDIT-M5K ENZYMES MUTAG	IMDB-M D#D REDDIT-MB	NCI1 PROTEINS COLLAB

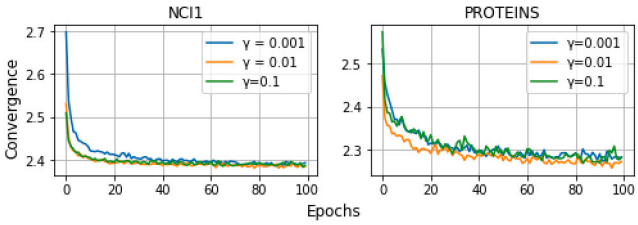

Fig. 3. GCN experiential training loss convergence of Adam optimizer in GCP on datasets NCI1 and PROTEINS with different γ values.

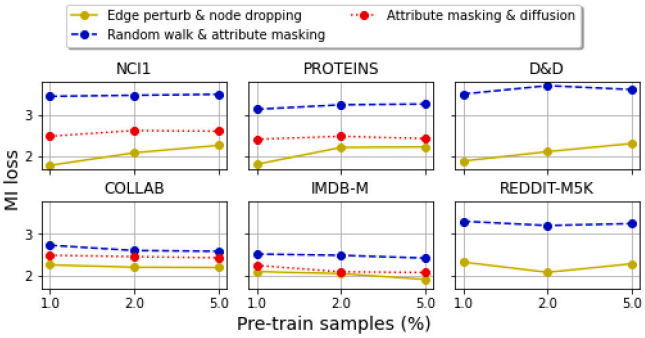

Fig. 4. Fraction of training samples (10%, 20%, and 50%) with various view pairs with GraphSAGE. Attribute masking and diffusion are OOM on D&D and REDDIT-M5K datasets.

4.5.2. Effect of pre-training samples

Fig. 4 shows the performance of our model trained with various fractions of training samples to estimate the error rate. Generally, during pre-training, 10% of the training samples perform better on biochemical molecules and can prevent overfitting in a self-supervised setting. It relatively balances the small information-to-noise ratio in high-order neighbors, preventing over-smoothing. For social networks, more training samples are required for better error estimation. The rationale is that the semantics of molecular graphs are delicate. Therefore, we opt for a 20% training sample for all datasets.

4.5.3. Training epochs and batch-sizes (RQ2)

Likewise, training longer with more epochs provides more negative samples for each sample, making the split of the total datasets more varied. Especially considering the importance of negative samples in contrastive learning, we investigated the effects of various epochs {100, 200, 300, 5000, and 1000} and batch sizes {64, 128, 256, and

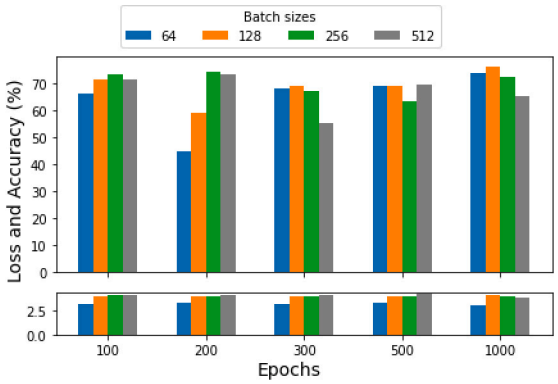

Fig. 5. GraphSAGE version of GCP performance with various epochs and batch sizes on PROTEINS dataset.

Table 5
Effect of the number of views on graph classification task based on the GCN. The **bold** highlight is the default view used in our model.

#V	PROTEINS	D&D	COLLAB
2	72.35 ± 0.28	79.61 ± 1.12	75.63 ± 0.02
3	**76.89 ± 1.36**	**85.63 ± 1.5**	**79.27 ± 0.15**
4	75.12 ± 0.24	76.33 ± 1.02	73.78 ± 0.11

512} during classification, as shown in Fig. 5. Overall, a batch size of 64 performs uniformly with low error rate estimation across various datasets. Therefore, we can further enhance the performance of GCP with batch size = 64 when we train the epoch for a longer time.

4.5.4. Effect of number of views (RQ3)

Again, we investigate the number of views (**#V**) to evaluate performance in a downstream task. During fine-tuning, we use three main views by default. Thus, pairing augmentation techniques extend the **#V**s by anchoring the original view as pre-trained embeddings. We use the pre-trained embeddings and the data's original view to perform the final classification task. To create more views, we further sampled views for each training sample in each mini-batch for contrastive representation. However, Table 5 indicates that extending the views does not introduce useful information, affecting performance. Compared to MVGRL [37], we can increase the number of views by default to three. Similarly, identical views are ignored in our experiments, as they carry similar information about the graph structure and do not lead to good performance [38]. Instead, it causes over-smoothing by increasing the noise in the small information-to-noise ratio in high-order neighbors.

4.5.5. Effect of number of layers (RQ4)

The standard GCN for classifying nodes and graphs reveals that they are usually superficial and limit the number of layers to 2 [7]. However, Bielak et al. [51] believe that the network layer can be extended on a large dataset, where they extended their encoder model to a 3-layer GCN model on some datasets. In this work, we can increase the number of layers in our proposed model without over-smoothing because of the structural similarity and transferability. Thus, using a fraction of the dataset during pre-training has similar local network topologies in the vector space, making it compatible with the vertices and graph when fine-tuning. This approach saved us the extra effort to search for similarity [38]. Table 6, shows the ablation performance of the depth of the GCP model architecture by experimenting with 2, 3, 4, and 5 layers. While a 3-layer most satisfactory benefits our proposed method, increasing the number of layers beyond that degrades the model's performance and also overfits in various GNN models. Note that performance on PROTEINS increases with the GCN model and on NCI1 and COLLAB with the GraphSAGE architecture beyond 3 layers. Intuitively, this can be attributed to the immense value of edge density in some datasets. When datasets are more densely connected, their representations become indistinguishable when the model goes deep [9]. Although there is an observable performance gain beyond the 3-layer encoder, the training procedure shows inconsistency, with numerous peaks in both validation loss and accuracy in some datasets.

Furthermore, the GCN model is known for its performance in transductive settings. In contrast, GraphSAGE is known for its generalized aggregator, mini-batch training, and fixed-size neighbor sampling algorithm to accelerate the training process, hence being more scalable in the inductive setting. Although our model is not currently designed for deeper GNNs, systematic observations highlight strategies to mitigate overfitting. These insights will inform our future work, allowing us to effectively utilize pre-trained embeddings and extend the depth of our GNNs.

4.5.6. Efficiency of GCP (RQ5).

Table 7, shows a comparison of GCP performance with state-of-the-art methods including GraphCL, JOAOv2 and SimGRACE in terms of training time and memory overhead. The results of other methods are directly copied from SimGRACE [54]. Training time refers to the time of the pre-training stage, and memory overhead refers to the estimated memory consumption of model parameters and all hidden batch representations of GCP. Observably, SimGRACE runs faster than our model, but the next competitive model for SimGRACE is GCP compared to JOAOv2 and GraphCL. Similarly, in terms of memory computation, our model is competitive in transductive settings, but not enough for inductive tasks. In particular, while the GCP approach is effective in improving the performance of graph classification tasks, it requires more computational memory on large-scale social graphs, such as COLLAB and REDDIT-B. It is important to state that the compared models are executed on different GPU distribution systems and also as layers increase, the more complex the model architecture becomes, the greater the complexity.

5. Conclusion

This work systematically composes multi-views from a sample graph engaging a pre-trained framework. Our approach collectively searches for the best augmentation for building multi-views to learn the intrinsic and transferable structural representations of the domain datasets. The hypothesis is to keep the task within the domain dataset, which equalizes the effort of similarity search in the nontransferable to the out-of-domain tasks. Considering the importance of negative samples in contrastive learning, we prolong the pre-training process for more negative samples for each sample to obtain a more varied split of total datasets, which relatively balances the small information-to-noise ratio in high-order neighbors. Extensive experiments on ten graph classification tasks demonstrate that our method outperforms the compared baselines with new state-of-the-art results, making better generalizable features and avoiding over-smoothing.

Future work: As a result of prolonged training for more negative samples for each sample, our approach relatively balances the small information-to-noise ratio in high-order neighbors. However, there is no high information-to-noise ratio in low-order neighbors. Therefore, increasing the network layer or going deeper can cause an interaction between high-order neighbors to bring over-smoothing. The ultimate question is; What are the trade-offs between a small information-to-noise ratio in high-order neighbors and a high information-to-noise ratio in low-order neighbors in fixing the interaction that brings over-smoothing? Our future work will explore a regularization effect in GNN models with deep probabilistic layers, as the functionality is well known in NLP and CV. Second, there is no theoretical interpretability to the information-to-noise ratio phenomenon and how to measure it for context-rich information in GNN. We hope to expand our domain to provide a more theoretical guarantee. Finally, another interest that is worth focusing on includes devising our method for the scalability of large-scale datasets.

Table 6
Effect of the number of layers (*#L*) on graph classification task based on GCN and GraphSAGE. The default *#Ls* used in our model is highlighted in **bold**. The blue numbers indicate the improvement when the layer increases.

GCN					GraphSAGE			
#L	NCI1	PROTEINS	COLLAB	IMDB-M	NCI1	PROTEINS	COLLAB	IMDB-M
2	77.40 ± 0.15	71.83 ± 2.09	79.87 ± 0.11	56.30 ± 1.22	77.91 ± 0.08	73.51 ± 0.12	79.13 ± 0.02	58.22 ± 0.09
3	**81.10 ± 0.1**	**73.71 ± 0.12**	**78.43 ± 0.07**	**55.19 ± 0.3**	**81.56 ± 0.12**	**76.89 ± 1.36**	**79.27 ± 0.15**	**56.22 ± 0.41**
4	80.18 ± 1.08	76.88 ± 0.03	81.96 ± 1.02	57.48 ± 3.10	82.45 ± 0.14	73.71 ± 0.44	78.20 ± 0.14	59.48 ± 0.31
5	80.45 ± 0.18	80.56 ± 2.31	80.78 ± 0.11	55.63 ± 0.13	82.86 ± 1.1	61.61 ± 0.10	83.20 ± 0.21	56.81 ± 1.21

Table 7
Comparative analysis of model efficiency on three graph datasets. We engaged the GraphSAGE model as a backbone for comparison. The best ones are emphasized in **bold** and the underlined ones indicate the second best.

Dataset	Method	GPU	Training time	Memory
PROTEINS	GraphCL		111 s	1231 MB
	JOAO		4088 s	1403 MB
	SimGRACE	32 GB V100	**46 s**	**1175 MB**
	GCP	24 GB RTX3090	82 s	1179 MB
COLLAB	GraphCL	–	1033 s	10199 MB
	JOAO	–	10742 s	7303 MB
	SimGRACE	32 GB V100	**378 s**	**6547 MB**
	GCP	24 GB RTX3090	915 s	7483 MB
REDDIT-B	GraphCL	–	917 s	4135 MB
	JOAO	–	10278 s	3935 MB
	SimGRACE	32 GB V100	**280 s**	**2729 MB**
	GCP	24 GB RTX3090	724 s	5727 MB

CRediT authorship contribution statement

Michael Adjeisah: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Xinzhong Zhu:** Resources, Writing – review & editing, Supervision, Funding acquisition. **Huiying Xu:** Resources, Supervision. **Tewodros Alemu Ayall:** Software, Validation, Formal analysis, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used are academic data online.

Acknowledgments

This work was supported by Zhejiang Normal University Postdoctoral Research Fund (Grant No. ZC304021943), the China Natural Science Foundation (Project No. 61976196), Zhejiang Provincial Natural Science Foundation (Project No. LZ22F030003) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 900025.

References

[1] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, Liang Wang, Graph contrastive learning with adaptive augmentation, in: Proceedings of the Web Conference 2021, 2021, pp. 2069–2080.

[2] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, Yang Shen, Graph contrastive learning with augmentations, Adv. Neural Inf. Process. Syst. 33 (2020) 5812–5823.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, Jeff Dean, Distributed representations of words and phrases and their compositionality, Adv. Neural Inf. Process. Syst. 26 (2013).

[5] Muhan Zhang, Yixin Chen, Link prediction based on graph neural networks, Adv. Neural Inf. Process. Syst. 31 (2018) 5165–5175.

[6] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, Neil Shah, Data augmentation for graph neural networks, in: The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), 2021, pp. 11015–11023.

[7] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations (ICLR-2017), 2017.

[8] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, Quanquan Gu, Layer-dependent importance sampling for training deep and large graph convolutional networks, Adv. Neural Inf. Process. Syst. 32 (2019) 11249–11259.

[9] Meng Liu, Hongyang Gao, Shuiwang Ji, Towards deeper graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 338–348.

[10] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, Jure Leskovec, Hierarchical graph representation learning with differentiable pooling, Adv. Neural Inf. Process. Syst. 31 (2018) 4800–4810.

[11] Xiao Luo, Wei Ju, Meng Qu, Chong Chen, Minghua Deng, Xian-Sheng Hua, Ming Zhang, Dualgraph: Improving semi-supervised graph classification via dual contrastive learning, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 699–712.

[12] Wei Ju, Xiao Luo, Meng Qu, Yifan Wang, Chong Chen, Minghua Deng, Xian-Sheng Hua, Ming Zhang, TGNN: A joint semi-supervised framework for graph-level classification, in: Lud De Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, International Joint Conferences on Artificial Intelligence Organization, 2022, pp. 2122–2128, Main Track.

[13] Qimai Li, Zhichao Han, Xiao-Ming Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 3538–3545.

[14] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, Stefanie Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 5453–5462.

[15] Johannes Klicpera, Aleksandar Bojchevski, Stephan Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, in: 7th International Conference on Learning Representations, ICLR-19, 2019.

[16] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, Xavier Bresson, Benchmarking graph neural networks, 2020, arXiv preprint arXiv:2003.00982.

[17] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, Stephan Günnemann, Pitfalls of graph neural network evaluation, 2018, arXiv preprint arXiv:1811.05868.

[18] Federico Errica, Marco Podda, Davide Bacciu, Alessio Micheli, A fair comparison of graph neural networks for graph classification, in: Proceedings of the International Conference on Learning Representations (ICLR-20), 2019.

[19] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, Xu Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, (04) 2020, pp. 3438–3445.

[20] Yu Rong, Wenbing Huang, Tingyang Xu, Junzhou Huang, Dropedge: Towards deep graph convolutional networks on node classification, 2019, arXiv preprint arXiv:1907.10903.

[21] Jianfei Chen, Jun Zhu, Le Song, Stochastic training of graph convolutional networks with variance reduction, in: 35th International Conference on Machine Learning, (ICML-18), 2018, pp. 1503–1532.

[22] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017) 1024–1034.

[23] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[24] Wenbing Huang, Tong Zhang, Yu Rong, Junzhou Huang, Adaptive sampling towards fast graph representation learning, Adv. Neural Inf. Process. Syst. 31 (2018) 4558–4567.

[25] Jie Chen, Tengfei Ma, Cao Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, in: 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings, 2018.

[26] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, Cho-Jui Hsieh, 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.

[27] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, Viktor Prasanna, Graphsaint: Graph sampling based inductive learning method, in: Proceedings of International Conference on Learning Representations (ICLR-20), 2020.

[28] Xiaoyun Wang, Minhao Cheng, Joe Eaton, Cho-Jui Hsieh, Felix Wu, Attack graph convolutional networks by adding fake nodes, 2018, arXiv preprint arXiv: 1810.10751.

[29] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, Le Song, Adversarial attack on graph structured data, in: International Conference on Machine Learning, PMLR, 2018, pp. 1115–1124.

[30] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, Tom Goldstein, Adversarial training for free!, Adv. Neural Inf. Process. Syst. 32 (2019) 3358–3369.

[31] Zhijie Deng, Yinpeng Dong, Jun Zhu, Batch virtual adversarial training for graph convolutional networks, in: ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data, 2019.

[32] Susheel Suresh, Pan Li, Cong Hao, Jennifer Neville, Adversarial graph augmentation to improve graph contrastive learning, Adv. Neural Inf. Process. Syst. 34 (2021).

[33] Michael Adjeisah, Guohua Liu, Douglas Omwenga Nyabuga, Richard Nuetey Nortey, Jinling Song, Pseudotext injection and advance filtering of low-resource corpus for neural machine translation, Comput. Intell. Neurosci. 2021 (2021).

[34] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, Bryan Hooi, Nodeaug: Semi-supervised node classification with data augmentation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 207–217.

[35] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, Jian Tang, Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization, in: International Conference on Learning Representations (ICLR-20), 2020.

[36] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, Jure Leskovec, Strategies for pre-training graph neural networks, in: International Conference on Learning Representations (ICLR-20), 2020.

[37] Kaveh Hassani, Amir Hosein Khasahmadi, Contrastive multi-view representation learning on graphs, in: International Conference on Machine Learning, PMLR, 2020, pp. 4116–4126.

[38] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, Jie 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] Shuiqiao Yang, Borui Cai, Taotao Cai, Xiangyu Song, Jiaojiao Jiang, Bing Li, Jianxin Li, Robust cross-network node classification via constrained graph mutual information, Knowl.-Based Syst. (2022) 109–852.

[40] Michael Adjeisah, Xinzhong Zhu, Huiying Xu, Tewodros Alemu Ayall, Towards data augmentation in graph neural network: An overview and evaluation, Comp. Sci. Rev. 47 (2023) 100527.

[41] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, T. Goldstein, FLAG: adversarial data augmentation for graph neural networks 2020, 2021, arXiv preprint arXiv:2010.09891.

[42] Dan-Hao Zhu, Xin-Yu Dai, Jia-Jun Chen, Pre-train and learn: Preserving global information for graph neural networks, J. Comput. Sci. Tech. 36 (6) (2021) 1420–1430.

[43] Songtao Liu, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, Dinghao Wu, Local augmentation for graph neural networks, 2021, arXiv preprint arXiv:2109.03856.

[44] Li Wang, Wei Huang, Miao Zhang, Shirui Pan, Xiaojun Chang, Steven Weidong Su, Pruning graph neural networks by evaluating edge properties, Knowl.-Based Syst. 256 (2022) 109847.

[45] Jingshu Peng, Yanyan Shen, Lei Chen, Graphangel: Adaptive and structure-aware sampling on graph neural networks, in: 2021 IEEE International Conference on Data Mining, ICDM, 2021, pp. 479–488.

[46] Fuli Feng, Xiangnan He, Jie Tang, Tat-Seng Chua, Graph adversarial training: Dynamically regularizing based on graph structure, IEEE Trans. Knowl. Data Eng. 33 (6) (2019) 2493–2504.

[47] Xiaohui Chen, Yukun Li, Aonan Zhang, Li-Ping Liu, Nvdiff: Graph generation through the diffusion of node vectors, 2022, arXiv preprint arXiv:2211.10794.

[48] Cheng Wu, Chaokun Wang, Jingcao Xu, Ziyang Liu, Kai Zheng, Xiaowei Wang, Yang Song, Kun Gai, Graph contrastive learning with generative adversarial network, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 2721–2730.

[49] Yonghui Yang, Zhengwei Wu, Le Wu, Kun Zhang, Richang Hong, Zhiqiang Zhang, Jun Zhou, Meng Wang, Generative-contrastive graph learning for recommendation, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 1117–1126.

[50] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, Xiang Zhang, Infogcl: Information-aware graph contrastive learning, Adv. Neural Inf. Process. Syst. 34 (2021) 30414–30425.

[51] Piotr Bielak, Tomasz Kajdanowicz, Nitesh V. Chawla, Graph barlow twins: A self-supervised representation learning framework for graphs, Knowl.-Based Syst. (2022) 109–631.

[52] Guanyi Chu, Xiao Wang, Chuan Shi, Xunqiang Jiang, CuCo: Graph representation with curriculum contrastive learning, in: IJCAI, 2021, pp. 2300–2306.

[53] Haoran Yang, Hongxu Chen, Shirui Pan, Lin Li, Philip S. Yu, Guandong Xu, Dual space graph contrastive learning, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1238–1247.

[54] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, Stan 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.

[55] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, Xin Wang, Gppt: Graph pre-training and prompt tuning to generalize graph neural networks, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1717–1727.

[56] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, Yizhou Sun, Gpt-gnn: Generative pre-training of graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1857–1867.

[57] Pengyong Li, Jun Wang, Ziliang Li, Yixuan Qiao, Xianggen Liu, Fei Ma, Peng Gao, Seng Song, Guotong Xie, Pairwise half-graph discrimination: A simple graph-level self-supervised strategy for pre-training graph neural networks, 2021, arXiv preprint arXiv:2110.13567.

[58] Xiao Luo, Wei Ju, Meng Qu, Yiyang Gu, Chong Chen, Minghua Deng, Xian-Sheng Hua, Ming Zhang, Clear: Cluster-enhanced contrast for self-supervised graph representation learning, IEEE Trans. Neural Netw. Learn. Syst. (2022).

[59] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, Chuan Shi, Learning to pre-train graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, (5) 2021, pp. 4276–4284.

[60] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, Liang Wang, Deep graph contrastive representation learning, in: ICML Workshop on Graph Representation Learning and beyond, 2020.

[61] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, Liang Wang, Graph contrastive learning with adaptive augmentation, in: Proceedings of the Web Conference 2021, 2021, pp. 2069–2080.

[62] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, Michal Valko, Bootstrapped representation learning on graphs, in: ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.

[63] Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, Yoshua Bengio, Learning deep representations by mutual information estimation and maximization, in: ICLR 2019, ICLR, 2019.

[64] Aaron van den Oord, Yazhe Li, Oriol Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv preprint arXiv:1807.03748.

[65] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, Huajun Chen, Federated knowledge graph completion via embedding-contrastive learning, Knowl.-Based Syst. 252 (2022) 109459.

[66] Nikil Wale, Ian A. Watson, George Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, Knowl. Inf. Syst. 14 (3) (2008) 347–375.

[67] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S.V.N. Vishwanathan, Alex J. Smola, Hans-Peter Kriegel, Protein function prediction via graph kernels, Bioinformatics 21 (suppl_1) (2005) i47–i56.

[68] Paul D. Dobson, Andrew J. Doig, Distinguishing enzyme structures from non-enzymes without alignments, J. Mol. Biol. 330 (4) (2003) 771–783.

[69] Nils Kriege, Petra Mutzel, Subgraph matching kernels for attributed graphs, in: International Conference on Machine Learning, 2012, pp. 291–298.

[70] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, Dietmar Schomburg, BRENDA, the enzyme database: updates and major new developments, Nucl. Acids Res. 32 (suppl_1) (2004) D431–D433.

[71] Pinar Yanardag, S.V.N. Vishwanathan, Deep graph kernels, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1365–1374.

[72] Muhan Zhang, Zhicheng Cui, Marion Neumann, Yixin Chen, An end-to-end deep learning architecture for graph classification, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[73] Martin Simonovsky, Nikos Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3693–3702.
[74] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka, How powerful are graph neural networks? in: 7th International Conference on Learning Representations, ICLR-2019, 2019.
[75] Yuning You, Tianlong Chen, Yang Shen, Zhangyang Wang, Graph contrastive learning automated, in: International Conference on Machine Learning, PMLR, 2021, pp. 12121–12132.
[76] Wang Yiwei, Wang Wei, Liang Yuxuan, Cai Yujun, Hooi Bryan, Mixup for node and graph classification, 2021, pp. 3663–3674.
[77] Federico Errica, Marco Podda, Davide Bacciu, Alessio Micheli, A fair comparison of graph neural networks for graph classification, 2019, arXiv preprint arXiv: 1912.09893.