MCGC: an MLP-based supervised Contrastive learning framework for Graph Classification

Xiao Yue
Department of Computer Science
Oakland University
Rochester, USA
xiaoyue@oakland.edu

Bo Liu
School of Mathematical and Computational Sciences
Massey University
Palmerston North, New Zealand
b.liu@massey.ac.nz

Andrew Meng
Troy high school
Troy, USA
andrewmeng114@gmail.com

Guangzhi Qu
Department of Computer Science
Oakland University
Rochester, USA
gqu@oakland.edu

Abstract

Graph Neural Networks (GNNs) have been widely used for tasks involving graph-structured data. These networks create matrix representations of graphs by aggregating nodes information from neighbor nodes recursively. Integrating with contrastive learning, graph contrastive learning has shown enhanced performance on graph-level tasks. However, architectures of graph contrastive learning frameworks become complicated due to the sophisticated structures of GNN-based encoder and necessity of both encoder and projection head. In this paper, we proposed a significantly simplified MLP-based supervised contrastive learning framework for graph classification tasks, coined as MCGC, which does not incorporate any GNN layers. Experimental results on graph benchmark datasets and ablation studies indicate that, despite not utilizing GNN layers, our framework achieved comparable or even superior performance on graph classification tasks against some state-of-the-art models.

1 Introduction

The growing accessibility of graph data has a significant impact on various domains, including social analysis(1), biology, transportation and financial systems(2). Nevertheless, unlike data with orderly arranged representations such as images, nature property of lacking euclidean structure in graph data poses challenges for applying machine learning techniques on graphs. Inspired by neural networks(3) which efficiently extract patterns from large and high-dimensional datasets, variants of graph neural networks(4; 5) have been developed such as graph auto-encoder(6), graph recurrent neural networks(7; 8), graph attention networks(9), graph isomorphism networks(10) and graph convolutional networks. Graph neural networks are exploited for tasks involving graph-structured data by combining graph networks and neural networks, thereby extending existing machine learning techniques to the realm of graph data. Features of a node and structural information of the node's neighborhoods are synthesized to create an vector representation of a node by aggregating nodes information from neighboring nodes recursively, therefore a matrix representation of a graph can be obtained through readout functions such as a pooling layer or an aggregation function. GNNs have been widely adopted as standard and state-of-the-art tools for processing diverse categories of graphs in practice.

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

In recent years, interest in integrating contrastive learning with graph learning(11) is aroused, inspired by the success of contrastive learning in computer vision(12) and natural language processing domains. Most of graph contrastive learning frameworks utilize GNN layers as encoders to generate matrix representations of a graph, because of GNNs' exceptional capabilities in handling graph data. However, the architectures of graph contrastive learning frameworks become complicated due to the sophisticated structures of GNN layers and necessity of both encoder (GNN layers) and projection head (a MLP), leading the total number of model parameters to reach hundreds of thousands even millions. Additionally, the computational cost of graph operations, such as aggregations and propagations within GNN layers, can be expensive and unnecessary in certain scenarios. In this paper, we proposed a significantly simplified MLP-based supervised contrastive learning framework for graph classification tasks, coined as MCGC, which does not incorporate any GNN layers. GNN-based encoder in general frameworks is replaced by a feature extraction layer, generating primitive vector representations of nodes in a graph without relying on GNN related operations. A sum aggregator after the projection head produces final embedding of a graph. By eliminating the GNN layer, performance of our framework solely rely on the MLP and contrastive learning. Furthermore, we applied supervised contrastive learning methodology(13) that fully exploits label information provided to improve performance of contrastive learning in graph level. To validate effectiveness of our framework, we conducted experiments on several graph learning benchmark datasets. Experimental results indicate that, despite not utilizing any GNN layers, our framework achieved comparable or even superior performance on graph classification tasks against some state-of-the-art models. Contributions of this paper are presented as follows:

- We develop a feature extraction layer to generate vector representations of nodes in a graph as inputs for projection head. A sum aggregator is utilized to produce a final embedding of a graph.
- We investigate approaches of not utilizing GNN layers in graph contrastive learning. As a result, we propose MCGC, a significantly simplified graph contrastive learning framework that does not incorporate any GNN layers.
- We employ supervised contrastive learning technique on MCGC framework to fully exploit label information in contrastive learning.

The remaining sections of this paper are organized as follows. In Section 2, we provide an overview of the related work on graph neural networks and graph contrastive learning. Details of approaches utilized in our framework are presented in Section 3. Section 4 presents our experiments, including performance comparison between our framework and other state-of-the-art models, exploration on numbers of parameters, exploration on augmentations on MCGC, as well as ablation studies. Finally, we draw brief conclusions in Section 5.

2 Related Work

2.1 Graph neural networks

Graph neural networks primarily follow the neighborhood information aggregation algorithm to produce a representation of a graph. They aggregate information of node's neighborhood to obtain a new representation of the node iteratively. The representation of a entire graph is then formed by readout functions such as a pooling function or an aggregator. As a prominent GNN variant, graph convolutional networks (GCNs) are generally divided into two main categories(14): spectral-based and spatial-based, according to their algorithmic properties. Spectral-based graph convolutional networks(15; 16; 17; 18) apply convolutional operations on graphs in Fourier domain. On the other hand, due to the restriction of fixed graph structures on convolutional operations in Fourier domain, spatial-based graph convolutional networks(19; 20; 21; 22; 23; 24) bypass it by alternative convolutional operations which aggregate nodes' information from their neighbor nodes. Apart from GCNs, other variants of GNNs has been developed for applying machine learning techniques on graphs. Graph attention network (GAT)(9) utilizes novel convolution-style approach with attention mechanism. Xu et al. proposed graph isomorphism network (GIN) to maximize the discriminative power of a GNN. GraphSAGE (25) applys an inductive representation learning algorithm on large graphs by sampling neighborhood and aggregating information. Graph neural networks are specifically designed for handling graph-related tasks and have shown excellent performance in

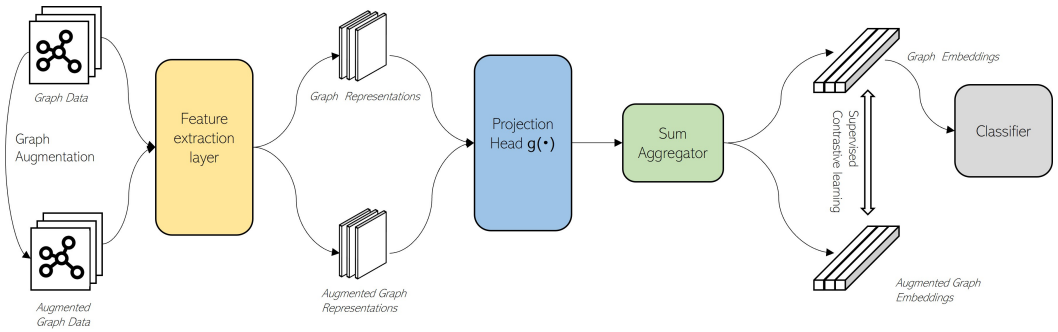

Figure 1: Overview of MCGC

various applications. However, we aim to simplify the framework architecture and investigate the effectiveness of not using GNN layers in graph contrastive learning. Therefore, GNN layers are not incorporated into our framework, allowing us to explore alternative approaches for graph learning.

2.2 Graph Contrastive Learning

As a self-supervised learning technique, contrastive learning aims to maximize the feature consistency of positives in the latent space, while minimizing feature consistency of negatives. There has been a surge of interest in leveraging contrastive learning to graph-related tasks, driven by its notable success in the fields of computer vision and natural language processing. DGI(26) utilizes an approach of learning node representations within graph-structured data by contrastive learning. As an extension of DGI, InfoGraph(27) aims at graph-level learning tasks by utilizing mutual information maximization. You et al. proposed GraphCL framework for GNN pre-training and analyzed influence of data augmentations(11). Additionally, graph contrastive learning is explored in (28; 29; 30; 31; 32).

2.3 Supervised contrastive learning

Contrastive learning, as a self-supervised learning technique, requires no label information for training, making it particularly valuable for datasets with a scarcity of labels. Khosla et al. (13) proposed an extension of contrastive learning that exploits labels to enable supervised training. In contrast to self-supervised contrastive learning, which considers only augmentations of the sample as positives, supervised contrastive learning treats all samples belonging to the same class as positives, as well as their augmentations. Conversely, samples and augmentations from different classes are treated as negatives. Supervised contrastive learning exhibits improved robustness to corruption. Inspired by it, Zheng et al.(33) presented a weakly supervised contrastive learning framework with only two projection heads on image classification. A supervised contrastive learning objective for fine-tuning language learning models in few shot learning is demonstrated in (34).

3 proposed framework

In this section, details of our proposed framework are presented. Figure 1 shows the proposed MCGC framework. Firstly, graph augmentations are generated for subsequent stages. Then MCGC exploits the feature extraction layer and sum aggregator as alternatives to GNN-based encoder to create embeddings of graphs and augmented graphs, as shown in Figure 2. Supervised contrastive learning method is utilized for maximizing the consistency among the positives and minimizing the consistency among the negatives. Lastly, after the projection head is optimized by supervised contrastive learning loss function, embeddings of graphs are fed into a classifier for final predictive results.

3.1 Notations

Consider a graph $\mathcal{G}_g = (\mathcal{N}_g, \mathcal{E}_g), \mathcal{G}_g \in \mathcal{D}$ in a graph dataset $\mathcal{D}$, where $\mathcal{N}_g$ denotes the set of all nodes and $\mathcal{E}_g$ denotes the set of all edges in $\mathcal{G}_g$. Let N_g to be the quantity of nodes and E_g to be the quantity

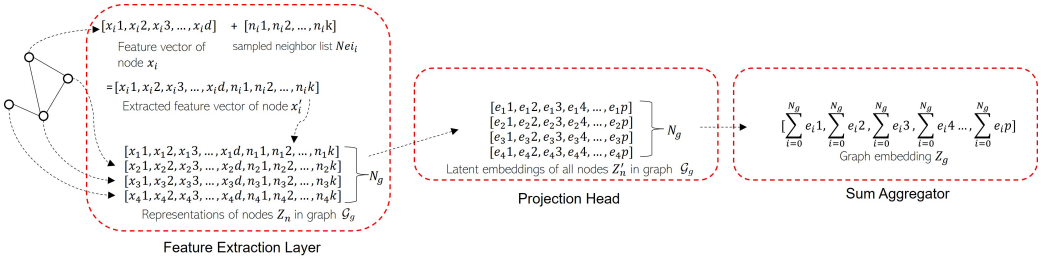

Figure 2: Building a balanced matrix representation by feature extraction layer and pooling layer

of edges in graph $\mathcal{G}_g$. Each node $n_i \in \mathcal{N}_g$ has a d-dimensional feature vector denoted as $x_i \in \mathbb{R}^d$. In this paper, a graph $\mathcal{G}_g$ is assumed to have no self loop. Matrix A^g denotes the adjacent matrix of the graph $\mathcal{G}_g$ where $A^g_{ij} = 1$ indicates nodes n_i and n_j are adjacent.

3.2 Feature extraction layer

Non-Euclidean graph data inherently lack consistent features, therefore extra layers are required to generate orderly arranged representations of graphs, which are mainly achieved by GNN layers with learnable weight matrices in a GNN-base framework. However, GNN layers are not exploited in our framework to demonstrate non-necessity in certain graph classification tasks. Therefore, we build a feature extraction layer to firstly convert a raw graph data to representations of nodes in this graph. Section *Feature extraction layer* in figure 2 shows the approaches of building representations of nodes in a graph by the feature extraction layer. Consider a graph $\mathcal{G}_g$, given the feature vector $x_i \in \mathbb{R}^d$ of a node $n_i \in \mathcal{N}_g$ and the set of its adjacent nodes $AD_i = \{x_j | A^g_{ij} = 1\}$, we randomly sample k nodes from AD_i to derive topology information of graph. Indices of these sampled nodes $\{n_i 1, n_i 2, ...n_i k\}$ are concatenated to build a sampled neighbor list Nei_i of node n_i. Extracted feature vector x_i' of node n_i is built by concatenating x_i and Nei_i. $Z_N \in \mathbb{R}^{N_g \times (d+k)}$ denotes a matrix of representations of all nodes in graph $\mathcal{G}_g$, created by a vertical concatenation of extracted feature vectors of all nodes $Z_N = \|_{i=1}^{N_g} x_i'$. Here, the value of k is either specified manually or set as max degree of all nodes $n_i \in \mathcal{N}_g$.

3.3 Projection head and sum aggregator

Projection head $g_\theta(\cdot)$, which is a non-linear transformation with parameters θ, transforms the matrix of representations of all nodes $Z_N \in \mathbb{R}^{N_g \times (d+k)}$ to latent embeddings of nodes $g_\theta(Z_N) = Z'_N \in \mathbb{R}^{N_g \times p}$ that are utilized for calculating contrastive learning losses in latent space, where p indicates output size of projection head. MLP is broadly chosen as projection head in existing graph contrastive learning approaches (11; 28), which transforms representations generated by a preceding GNN-based encoder to embeddings in contrastive learning latent space. However, the encoder is removed in our framework, causing projection head to be the only learnable transformation. Therefore, functionalities of encoder and projection head in other works are combined into a single projection head in our framework. Afterwards, a readout function aims to build an embedding of the entire graph according to embeddings of individual nodes for classification tasks on graph level. The sum aggregator is utilized in MCGC due to its better expressive power than mean and max aggregators(10). Graph embedding $Z_g \in \mathbb{R}^p$ of graph $\mathcal{G}_g$ is obtained by a sum aggregator $F_{sum}()$ which calculates summations through the feature dimension $Z_g = F_{sum}(Z'_N) = \sum_{N_g} Z'_N$.

3.4 Graph supervised contrastive learning

Contrastive learning has gained significant attention in the field of machine learning and has shown promising performance, particularly in computer vision and natural language processing domain. One commonly used contrastive learning loss is InfoNCE(35) that is presented as equation 1.

$$\mathcal{L}(t_i) = -log \frac{exp(z_i \cdot z_j / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)} \tag{1}$$

Here, t_i represents a sample in dataset and $A(i)$ indicates a set of all samples excluding t_i, symbol $\cdot$ denotes the inner product, and τ denotes the temperature parameter which has been well explored in (36). As a self-supervised learning loss function, it doesn't exploit label information during training since augmentations are the only positives for sample t_i. As an extension of equation 1, supervised contrastive learning loss function is proposed in (13) which is generalized to an arbitrary number of positives, shown in equation 2.

$$\mathcal{L}_{Sup}(t_i) = \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{exp(z_i \cdot z_p/\tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a/\tau)} \tag{2}$$

Here, $P(i)$ indicates a sample set containing all samples that share the same label with t_i, and $|P(i)|$ denotes cardinality of $P(i)$. According to equation 2, we treat all samples with a same label as positives, while all other samples are negatives. The gradient of the loss function in equation 2 is presented as equation 3.

$$\frac{\partial \mathcal{L}}{\partial z_i} = \frac{1}{\tau} \left\{ \sum_{p \in P(i)} z_p (P_{ip} - \frac{1}{|P(i)|}) + \sum_{n \in N(i)} z_n P_{in} \right\} \tag{3}$$

Where $N(i)$ is a sample set containing all samples that have different labels from sample t_i and P_{it} is defined in equation 4.

$$P_{it} = \frac{exp(z_i \cdot z_t/\tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a/\tau)} \tag{4}$$

Therefore the gradient of loss function in equation 2 is calculated based on all positive and negative samples. By maximizing the consistency among the positives and minimizing the consistency among the negatives, graph embeddings with the same label tend to be closer together in the latent space. In our framework, projection head $g_\theta(\cdot)$ and sum aggregator take matrix of representations of all nodes Z_N as an input and transform it to latent space as a graph embedding Z_g, represented as $F_{sum}(g_\theta(Z_n)) = Z_g$. Graph embedding is utilized as z_i in equation 2 to calculate contrastive loss in latent space. Therefore, projection head $g_\theta(\cdot)$ is optimized by equation 5.

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{\mathcal{G}_i \in \mathcal{D}} \mathcal{L}_{Sup}(F_{sum}(g_\theta(Z_n))) \tag{5}$$

3.5 Graph augmentation

Inspired by the image augmentation methods such as rotation and cropping used in image contrastive learning, graph data augmentation methods aim to generate new graph data by applying certain transformations to the original graphs while preserving their semantic information. Previous research has explored and proposed various methods for graph augmentations (11). Here we utilize two existing augmentation methods *Edge dropping* and *Feature masking*, as well as a new augmentation method *Index shuffling* which is specifically designed for our framework.

Edge dropping augmentation method randomly discards certain edges from a given graph $\mathcal{G}_g$. The underlying assumption behind this method is that removing certain edges does not significantly alter the semantic information of the graph. This method introduces variability into the graph data while preserving important patterns and relationships within the graph.

Feature masking augmentation method randomly masks some elements in the feature vectors of certain nodes, based on the assumption that a small percentage of missing values is unlikely to significantly impact the performance of the model. It allows researchers to investigate the model's robustness to missing feature values and assess its ability to handle noisy data.

Nodes may lack meaningful indices in many graphs, leading to the assignments of random indices when storing them into a data structure. Although the randomness of these indices does not alter the graph's underlying structure, it can introduce variations in the adjacency matrix, as well as extracted feature vectors of nodes x'_i. Therefore, we propose the *index shuffling* augmentation method which randomly shuffles the indices of all nodes within a graph, resulting in the regeneration of adjacency matrix and extracted feature vectors of all nodes.

Note that most of random argumentation methods, including *Edge dropping* and *Feature masking*, are risky to alter graph structure and generate misleading views (37), especially in some sensitive fields like chemoinformatics where even a slight change in a compound can result in a significant

variation in its chemical properties. However, *Index shuffling* is considered risk-free, as it builds isomorphisms of original graph to retain graph semantics. Isomorphisms are not commonly utilized as augmentations in other graph contrastive learning frameworks due to the abilities of GNN-based encoders on identifying isomorphisms such as GIN(10). However, absence of GNN-based encoder in our framework makes isomorphism augmentations are possible to be valuable because of their high similarity with original graph.

4 Experiments

4.1 Datasets

We assessed the effectiveness of our proposed approaches by evaluating performance on multiple benchmark graph datasets for graph classification tasks. These datasets were derived from the TU Graph dataset(38), a comprehensive collection of benchmark datasets for graph learning tasks. It contains a diverse range of graph datasets, including small molecules, bioinformatics, computer visions, social networks and synthetic graphs. We selected 6 datasets from TU dataset: MUTAG(39), PROTEINS(40), DD(41), NCI1(42), IMDB_BINARY (43) and IMDB_MULTI (43). By testing the 10-fold cross validation accuracies of our framework on these datasets, we assessed the effectiveness of our proposed approaches. Table 1 shows detailed statistics of these datasets.

Table 1: Statistics of datasets

	Number of Graphs	Classes	Average Nodes	Average Edges
MUTAG	188	2	17.93	19.79
PROTEINS	1113	2	39.06	72.82
DD	1178	2	284.32	715.66
NCI1	4110	2	29.87	32.30
IMDB_BINARY	1000	2	19.77	96.53
IMDB_MULTI	1500	3	13.00	65.94

4.2 Settings

Our framework is implemented using Pytorch with Python version 3.9. We employed an linear layer whose output dimension is *256* as the projection head $g_\theta(\cdot)$. Additionally, a two-layer MLP is utilized as the classifier. For contrastive learning, learning rate is set as *0.001* and temperature parameter τ is set as *0.07*. Graph datasets in Table 1 are selected to evaluate the performance of MCGC. For each dataset, we conducted 10-fold cross validation to obtain the average and standard deviation of accuracies. Performance of our framework is evaluated and compared with GK kernels(44), WL kernels(45), DGCNN(22), GIN(10), GraphCL(11) and InfoGraph(27). Specifically, as two graph contrastive learning models in our experiments, GraphCL adopts GCN as its encoder, while GIN are selected as encoder in InfoGraph.

4.3 Experimental results and PCA visualization analysis

Table 2 shows experimental results and comparisons with other models. According to Table 2, MCGC demonstrates comparable performance on **DD** and **IMDB_MULTI** datasets. Moreover, it exhibits superior performance on **PROTEINS**, **MUTAG** and **IMDB_BINARY** datasets compared to other models. However, when considering the **NCI1** dataset, MCGC's classification performance is found to be inferior to other models. In addition, the standard deviations of accuracies of MCGC are larger than others, caused by randomness from indices assignments and neighboring nodes sampling, as well as graph augmentations. It suggests that robustness of MCGC should be further improved to minimize the interference from randomness.

We exploited the PCA dimensional reduction technique to visualize graph embeddings of **PROTEINS**, **MUTAG** and **IMDB_BINARY** in the contrastive learning latent space, as an auxiliary method to evaluate the effectiveness of proposed approaches. Figure 3 presents the distributions of graph

Table 2: Experimental results

	MUTAG	PROTEINS	DD	NCI1	IMDB_B	IMDB_M
GK	80.7 ± 1.4	71.0 ± 0.2	75.0 ± 0.8	64.5 ± 0.3	65.3 ± 0.9	44.6 ± 0.4
WL	84.6 ± 1.8	74.9 ± 0.6	77.4 ± 0.7	82.1 ± 0.7	72.4 ± 0.7	49.8 ± 0.5
DGCNN	85.8 ± 1.6	75.5 ± 0.9	$\mathbf{79.3 \pm 0.9}$	74.4 ± 0.4	70.0 ± 0.8	47.8 ± 0.9
GIN	89.5 ± 4.8	76.5 ± 2.4	75.6 ± 2.6	$\mathbf{82.9 \pm 1.5}$	75.4 ± 4.5	$\mathbf{51.2 \pm 2.4}$
GraphCL	87.0 ± 1.1	74.5 ± 0.4	78.1 ± 0.5	77.2 ± 0.6	70.9 ± 0.5	49.1 ± 0.4
InfoGraph	89.3 ± 1.0	75.0 ± 0.3	73.2 ± 1.6	76.8 ± 1.1	73.0 ± 0.6	49.4 ± 0.6
MCGC	$\mathbf{92.5 \pm 3.8}$	$\mathbf{78.5 \pm 3.0}$	77.6 ± 1.4	72.2 ± 1.4	$\mathbf{77.8 \pm 2.6}$	49.9 ± 1.6

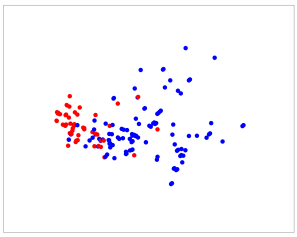

Figure 3: PCA visualization of MUTAG in 2D space

embeddings of **MUTAG** in 2D space after applying PCA method. Figure 3 shows that majority of red dots are clustered together and can be separated from most of blue dots, indicating effectiveness of exploiting embeddings for further classification tasks. However, the distributions of visualized embeddings of **PROTEINS** and **IMDB_BINARY** datasets are overlapped and not well separated, therefore the figures of them are not presented here, which can be caused by the sensitivity of PCA to outliers, a lower accuracy on these datasets, and probably a larger number of graphs in these datasets. Furthermore, we observed that a higher accuracy does not necessarily lead to a better separated visualized distribution of graph embeddings. Therefore performance of the models cannot be solely estimated by distributions of graph embeddings, and visualization should be considered as an auxiliary measurement only.

In conclusion, analysis of experimental results and PCA visualizations reveals several drawbacks of the proposed framework. First, it doesn't achieve higher performance on all datasets. Second, the framework is not comparatively robust enough to handle randomness. Lastly, the distributions of embeddings of some graph datasets may not be suitable for visualization using dimensional reduction method like PCA. These drawbacks guide us with the potential future improvements for our framework. Even though there are several drawbacks, MCGC provides effective approaches for exploiting supervised contrastive learning on graphs without relying on any GNN layers.

4.4 Exploration on numbers of parameters

To evaluate simplicity of MCGC in comparison to other models, we conducted parameters analysis experiments on MCGC, as well as two commonly used encoders in graph contrastive learning: the GCN encoder and the GIN encoder. Note that the GCN encoder and GIN encoder are typically connected with a separate projection head in most of graph contrastive learning framework, while MCGC does not require that. In our experimental setting, we set output dimension of MCGC as *256*. For evaluating the GCN encoder, we deployed a two-layer GCN encoder with hidden dimension *(128,256)*, along with a projection head whose output dimension is *256*. The GIN encoder had a hidden dimension of *256* and consisted of *3* layers. Output dimension of projection head for GIN encoder is set as *256*3*. Accuracies are skipped here since they are already presented in Table 2. Experimental results of parameters are presented in Table 3. Numbers of parameters in MCGC are significantly smaller than both GCN encoder and GIN encoder, highlighting the simplicity of the proposed model.

Table 3: Number of parameters in models

	MUTAG	PROTEINS	DD	NCI1	IMDB_B	IMDB_M
GCN Encoder+Proj Head	99840	99456	110336	103680	98944	98944
GIN Encoder+Proj Head	529408	528640	550400	537088	527616	527616
MCGC Encoder	2048	1280	23040	9728	256	256

4.5 Exploration on augmentations

According to equation 1, augmentations are indispensable in self-supervised contrastive learning as the augmentations are only positive views of the current sample. Without these augmentations, the lack of positive views can result in a zero loss value, making it impossible to calculate gradients for the backward pass. However, since label information is exploited and all samples with a same label are treated as positives in supervised contrastive learning shown as equation 2, augmentations theoretically become unnecessary if batch size is large with respect to the number of classes. To investigate the impact of augmentation methods and number of augmentations on the MCGC framework, we conducted 10-fold cross validation on three datasets : **MUTAG**, **PROTEINS** and **IMDB_BINARY**, since MCGC achieved best performance on them. Three augmentation methods *Edge dropping*, *Feature masking* and *Index shuffling* are employed to generated different numbers of augmentations (1 augmentation, 2 augmentations and 3 augmentations). Besides, we also conducted experiments without any augmentation (No augmentation) which are used as the baseline. Experimental results on **MUTAG**, **PROTEINS** and **IMDB_BINARY** are shown in Figure 4, Figure 5 and Figure 6 respectively. According to experimental results depicted in Figure 4, Figure 5 and Figure 6, neither

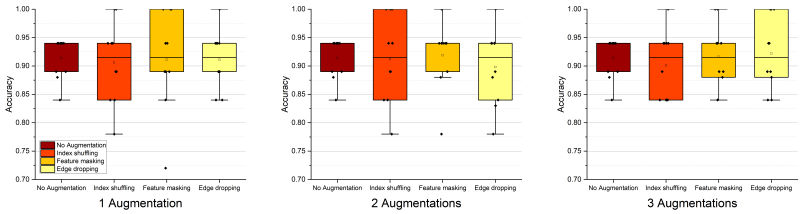

Figure 4: Augmentation experimental results on MUTAG dataset

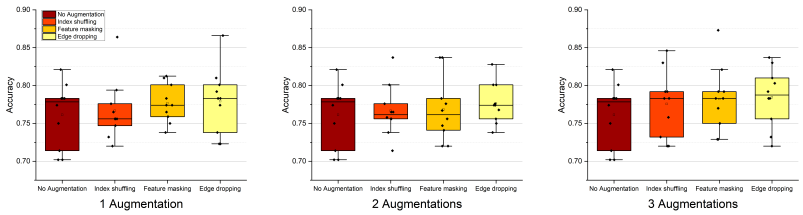

Figure 5: Augmentation experimental results on PROTEINS dataset

method nor number of augmentations has large impact on performance of MCGC on these three datasets. Considering introduction of randomness during the augmentation process, subtle variations on experimental results are inevitable. For **MUTAG** and **IMDB_BINARY** datasets, comparing to experimental results without augmentation, generating augmentations increases the range of the accuracy variation. However, range of the accuracy variation is larger due to the absence of augmentation on **PROTEINS**. Therefore, this suggests that effectiveness of augmentations may vary one dataset to another on MCGC due to properties of graph datasets, leading an interest for further investigation.

4.6 Ablation studies

We evaluated effectiveness of proposed framework by comparing with following three training strategies on three datasets: **MUTAG**, **PROTEINS** and **IMDB_BINARY** since MCGC achieved

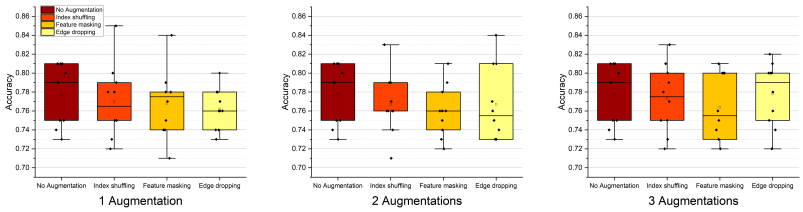

Figure 6: Augmentation experimental results on IMDB_B dataset

best performance on them.
We remove projection head and supervised contrastive learning in MCGC, as well as graph augmentations. Representations of nodes Z_n generated by feature extraction layer are directly fed into sum aggregator to produce a graph presentation for classification. Effectiveness of feature extraction layer is evaluated by this strategy. We denote this training strategy as *w/o SupCon*.
Instead of optimizing parameters of projection head by supervised contrastive learning loss, we modify our framework by following the methodology of the auto-encoder. Note that due to the restriction of not exploiting GNN layers, GAE (graph auto-encoder) and VGAE (variational graph auto-encoder) are not utilized in ablation studies. In this case, graph augmentations won't be generated due to absence of contrastive learning. We consider the projection head as the encoder to created latent embeddings of nodes from representations of nodes (Z_n). Additionally, a decoder is utilized to build reconstructed graph representations ($\hat{Z}_n$). The auto-encoder is pre-trained using the reconstruction error as the loss function, denoted as $\mathcal{L} = (Z_n, \hat{Z}_n)$. Node embeddings generated by the encoder are used for further classification tasks. It helps us to compare the contrastive learning method with other self-supervised learning method. This training strategy is referred to as *w/ AE, w/o SupCon*.
We substitute the supervised loss function (equation 2) by the self-supervised contrastive learning loss function (equation 1). In this case, label information won't be exploited during training projection head. It allows us to compare the effectiveness of self-supervised and supervised contrastive learning. This modified training strategy is denoted as *w/ SelfCon*.

Table 4: Experimental results on ablation studies

	MUTAG	PROTEINS	IMDB_BINARY
w/o SupCon	80.2 ± 7.2	72.9 ± 3.4	67.3 ± 3.4
w/ AE, w/o SupCon	79.6 ± 4.2	71.7 ± 4.0	65.2 ± 3.1
w/ SelfCon	88.8 ± 3.8	74.9 ± 2.9	73.4 ± 1.6
MCGC	92.5 ± 3.8	78.5 ± 3.0	77.8 ± 2.6

We employed three different training strategies and conducted 10-fold cross validation to obtain the average and standard deviation of accuracies of each training strategy. The experimental results on ablation studies are presented in Table 4. By comparing with three other training strategies, we conclude that approaches of our framework achieved best performance. Ablation studies demonstrate the effectiveness of proposed approaches. We observed that the *w/ SelfCon* strategy achieved much better performance compared to other two training strategies without contrastive learning, highlighting the effectiveness of contrastive learning in graph tasks.

5 Conclusions

In this paper, we proposed a significantly simplified supervised contrastive learning framework for graph classification tasks that does not rely on any GNN layers. The proposed framework simplifies structure of graph contrastive learning by eliminating the need for complex GNN layers. Furthermore, we have extended application of supervised contrastive learning to graph domain. Experimental results on graph benchmark datasets and ablation studies indicate that, despite not utilizing GNN layers, our framework achieved comparable or even superior performance on certain graph classification tasks against some state-of-the-art models. Even though some drawbacks are identified in experiments, approaches in proposed framework are effective for utilizing supervised contrastive learning on graphs without any GNN layers.

References

[1] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 635–644.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[5] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

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

[7] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[8] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

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

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

[11] 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, pp. 5812–5823, 2020.

[12] 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.

[13] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[14] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[16] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

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

[18] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016.

[19] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *arXiv preprint arXiv:1509.09292*, 2015.

[20] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*. PMLR, 2016, pp. 2014–2023.

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

[22] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[23] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "Deepergcn: All you need to train deeper gcns," *arXiv preprint arXiv:2006.07739*, 2020.

[24] X. Yue, G. Qu, B. Liu, and F. Zhang, "Edge utilization in graph convolutional networks for graph classification," in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2022, pp. 808–813.

[25] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[26] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *ICLR (Poster)*, vol. 2, no. 3, p. 4, 2019.

[27] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv preprint arXiv:1908.01000*, 2019.

[28] H. Liang, X. Du, B. Zhu, Z. Ma, K. Chen, and J. Gao, "Graph contrastive learning with implicit augmentations," *Neural Networks*, vol. 163, pp. 156–164, 2023.

[29] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 1294–1303.

[30] 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.

[31] N. Jovanović, Z. Meng, L. Faber, and R. Wattenhofer, "Towards robust graph contrastive learning," *arXiv preprint arXiv:2102.13085*, 2021.

[32] 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, pp. 15 920–15 933, 2021.

[33] M. Zheng, F. Wang, S. You, C. Qian, C. Zhang, X. Wang, and C. Xu, "Weakly supervised contrastive learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 042–10 051.

[34] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," *arXiv preprint arXiv:2011.01403*, 2020.

[35] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[36] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2495–2504.

[37] 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.

[38] 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

[39] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[40] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[41] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.

[42] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, pp. 347–375, 2008.

[43] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[44] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 488–495.

[45] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.