GMCL: Graph Mask Contrastive Learning for Self-Supervised Graph Representation Learning

1st Long Xu
National Key Laboratory of Information Systems Engineering
National University of Defense Technology
Changsha, China
xulong@nudt.edu.cn

2nd Zhiqiang Pan
National Key Laboratory of Information Systems Engineering
National University of Defense Technology
Changsha, China
panzhiqiang@nudt.edu.cn

3rd Honghui Chen*
National Key Laboratory of Information Systems Engineering
National University of Defense Technology
Changsha, China
chenhonghui@nudt.edu.cn

4th Tianjian Zhou
College of Systems Engineering
National University of Defense Technology
Changsha, China
zhoutianjian2000@nudt.edu.cn

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10650219

***Abstract*—Graph self-supervised learning is a task replete with potential. Previously, mainstream approaches in graph self-supervised learning were based on contrastive or generative tasks to extract graph embeddings, achieving commendable performance in downstream tasks such as node classification. However, these two methods have their respective strengths and limitations: (i) contrastive graph learning excels at capturing discriminative features of the graph but is more prone to overlooking structural representations of the graph itself; (ii) generative methods prioritize learning reconstructive features but struggle with large graphs and are susceptible to overfitting. Consequently, several simplistic fusion methods have been proposed, integrating encoded features from both approaches through concatenation, addition, or attention mechanisms for downstream tasks. However, these methods tend to be overly coarse, neglecting some unique information present in both categories of methods.**

To enhance the synergy of these two methods, we propose an edge perturbation data augmentation method to prevent the generation of spurious positive samples and a feature imputation decoder to complement contrastive features. We assess the model's performance on two downstream tasks in graph representation learning, and experimental results demonstrate that our proposed method outperforms baseline methods on ten publicly available datasets.

***Index Terms*—Graph Neural Networks, Self-Supervised Learning, Graph Representation Learning**

I. Introduction

Self-Supervised Learning (SSL) stands as a crucial subfield within the realm of machine learning, aiming to address the challenge of acquiring labels in supervised learning by learning from unlabeled data and autonomously generating labels or tasks [1]. While state-of-the-art methods leveraging large-scale data have found success in the Natural Language Processing (NLP) domain [2], graph data [3] [4], in comparison, grapples with substantial data scarcity due to the inherent difficulties in acquisition. Despite the absence of ample data, a plethora of self-supervised learning methods for graph has emerged [5] [6].

Existing self-supervised learning methods for graphs are largely divided into two categories: contrastive and generative. However, as mentioned in [7], generative methods exhibit better performance in understanding graph structures and features. Nonetheless, they struggle with handling large-scale graphs and are prone to overfitting. On the other hand, contrastive methods demonstrate stronger generalization capabilities but may overly focus on discriminative information, neglecting a more comprehensive learning of graph structures and feature representations [8]. In the domains of image and 3D point clouds [9], there have been works effectively combining these two approaches. However, in the field of graph-related research, such endeavors are still limited. Existing methods that integrate contrastive and generative approaches in graph self-supervised learning often involve simple concatenation, addition, or attention operations on the encoded features of both methods [8]. These techniques are predominantly applied to enhance model performance in competitions, lacking substantial improvements in network architecture for a more integrated fusion.

To address the aforementioned challenges more effectively, we introduced a Siamese architecture [10] to integrate BGRL with GraphMAE. One component is the graph mask autoencoder, which acquires the representation of the graph by reconstructing the mask information. To better incorporate the locally extracted relational features from the generative module into contrastive learning, we introduced a feature imputation decoder in the generative module to complement

the mask features and align them with the latent features of the contrastive learning module. The other component is the contrastive learning module, where we incorporated the BGRL model framework. We innovatively designed the data augmentation part to better integrate with the generative method.

Through the innovative module design described above, our GMCL model effectively leverages the strengths of both generative and contrastive methods, yielding promising results in graph representation learning tasks across several real-world datasets.

The contributions of this paper are summarized as follows:

1) We introduce a novel GMCL approach aimed at exploring the integration of generative and contrastive learning to enhance graph representation learning. The acquired representations not only capture local reconstruction information but also encompass discriminative information among diverse graph datasets.
2) To leverage the representations encoded by the generative module for contrastive learning, we propose a feature imputation decoder to complement the masked representations. Additionally, we introduce a graph data augmentation method with weak edge perturbation to prevent the generation of false positive samples, effectively elevating the encoding proficiency.
3) Our method effectively enhances the learning capabilities of graph self-supervised learning, outperforming current popular baseline methods across ten publicly available datasets.

II. Related Work

In this section, we briefly review two types of research relevant to our work, namely contrastive graph self-supervised learning and generative graph self-supervised learning.

A. Contrastive Graph Self-Supervised Learning

Contrastive graph self-supervised learning is a prevalent approach used for self-supervised learning on graph data. Contrastive graph learning algorithms train an encoder on a substantial amount of unlabeled graph data, enabling the extraction of informative graph representations that effectively capture the features inherent in the graph data. These representations are subsequently evaluated across various downstream tasks and applied in diverse real-world scenarios, such as molecular structures and social networks.

The developmental directions of contrastive learning methods in the field of graph can be broadly categorized into three main classes: data augmentation, contrastive hierarchies, and training procedures. In the computer vision (CV) domain, image data is typically augmented through operations such as cropping and rotation. In the context of graph data, there are generally two approaches. The first involves inducing some erosion and disruption operations on the graph structure, specifically by randomly removing nodes or randomly adding/deleting edges, as seen in GraphCL [11]. The second approach entails modifying graph features, including random masking of features or introducing noise to features, exemplified by methods like GRACE [12].

Contrastive graph self-supervised learning primarily emphasizes acquiring highly discriminative feature representations, extracting discriminative information from graphs while often overlooking the overall structure and detailed features within the graphs. In our study, the contrastive representations we utilize are derived from the feature imputation decoder of a generative model. This approach effectively incorporates the missing detailed intra-graph features into the training process, addressing a common deficiency in conventional graph contrastive learning.

B. Generative Graph Self-Supervised Learning

Generative graph self-supervised learning has evolved into another prominent paradigm in the domain of graph self-supervised learning. It entails the acquisition of graph representations through the reconstruction of missing input graph data. Within the field of NLP, the extraction of features by reconstructing or predicting words within sentences has consistently been a significant research focus. As Transformer models continue to advance in the CV domain, a multitude of generative image self-supervised learning models has emerged. An illustrative example is the MAE model, designed to recover and reconstruct RGB pixel values in images [13].

In the field of graphs, there are generally two major categories: autoregressive and autoencoding approaches [14]. The first category is commonly employed for graph generation, using an autoregressive method to construct graphs, thereby obtaining the joint probability of all information within the graph. Examples include GraphRNN [15] and GCPN [16]. There have been recent attempts in graph representation learning, such as GPT-GNN, which models the graph distribution through a generative approach, gradually predicting the features and connections of new nodes within the graph [17]. However, many graphs do not necessarily adhere to a fixed sequential representation, making autoregression less effective for such graph data.Another category involves the reconstruction of graph inputs, utilizing GNN as both encoders and decoders to extract graph features for downstream tasks, exemplified by models such as Graph Autoencoder (GAE) and Variational Graph Autoencoder (VGAE) [18].

Despite the commendable performance of generative graph self-supervised learning, it predominantly focuses on learning the reconstruction information within graphs, overlooking discriminative information across different graphs. Consequently, this paper conducts a specific investigation to address this issue.

III. Approach

We introduce the Graph Masked Contrastive Learning model, GMCL for brevity, delineated in Fig. 2. GMCL comprises two primary components: a generation module reconstructing the original graph from a masked graph and a contrastive module extracting discriminative insights within the graph. In the generation module, we begin by randomly

masking nodes in the graph and employ a GNN encoder to learn the representation of the masked graph. Subsequently, we subject the obtained representation to another round of random masking. This re-masked representation is then inputted into both the feature imputation decoder and the reconstruction decoder, yielding features for contrastive learning and reconstructed graph representations. Finally, we minimize the scaled cosine error between the reconstructed and perturbed graph representations, effectively capturing the graph's internal relational information [19]. Concerning the contrastive module, an encoder with a structure akin to the generation module is used to learn the local node information of the original graph. We then extract information via a prediction head and contrast it with the features output by the feature imputation decoder in the generation module, employing the scaled cosine error as the loss function. Following the execution of the contrastive module steps, discriminative information seamlessly integrates into the graph representation.

A. Generation Module

Our approach for the generative graph representation learning module consists of three key steps: 1) Encoding the masked node feature matrix $\mathbf{X}'$; 2) Employing feature imputation decoders and reconstruction decoders with a re-masking mechanism to obtain reconstructed node features for the masked portions and overall node features; 3) Minimizing the scaled cosine error between the reconstructed node features for the masked portions and the original graph node features.

1) Node Masking and Encoder Architecture: Initiating the edge perturbation data augmentation operation takes precedence, a process that will undergo meticulous analysis in the contrastive module. And then the masking operation is performed on the nodes in the graph. For the design of the encoder, we employed three distinct GNN methods: GAT, GCN, and GIN. In our experiments, we applied different types of encoders to various datasets, which will be elaborated upon in subsequent sections. Given the complexity of our model structure and with efficiency in mind, we devised an encoder denoted as $\mathcal{E}_\theta$ based on a dual-layer GNN architecture. This encoder is defined as follows:

$$\mathbf{H_1} = \mathcal{E}_\theta(\mathbf{A}, \mathbf{X}') \tag{1}$$

where $\mathbf{H_1}$ is the encoded features, and $\mathbf{A}$ is the adjacency matrix of the graph. $\mathbf{X}'$ is the masked node feature matrix, θ is the encoder parameter

Before the decoding phase, it is necessary to perform a secondary masking on the features of the encoded masked nodes. This procedure is implemented to compel the masked nodes to better exploit the encoding information from their unmasked neighboring nodes for self-reconstruction. A few dimensions of the masked node features are randomly chosen for masking in this step, with the selected feature positions being set to zero.

2) The Design of the Decoder: Subsequently, an introduction to the decoder module is provided, consisting of two decoders: the feature imputation decoder and the reconstruction decoder. The purpose of this module is to map the encoded visible and masked features to the features of the encoder in the contrastive module and the original graph. Due to differences in the mapping space, the decoders are separated into two types. The reconstruction decoder $\mathcal{G}_{rc}$ is designed to learn the reconstruction of masked node features. We utilize the entire node feature matrix $\mathbf{H}'$ after secondary masking to reconstruct the features of the masked nodes. This decoder facilitates the model in learning the comprehensive representation of each node. The reconstruction decoder is implemented as a single-layer graph neural network (GNN).

$$\mathbf{z}'_m = \mathcal{C}hoose(\mathcal{G}_{rc}(\mathbf{H}')) \tag{2}$$

where $\mathcal{C}hoose$ is a selection function used to filter out only the reconstructed features of the masked node, and $\mathbf{z}'_m$ is a feature prediction for the masked node.

In order to align with the outputs of the encoder in the contrastive module, it is necessary to restore the features of the masked nodes. Hence, we employ the feature imputation decoder $\mathcal{G}_f$, which shares the same structure as the reconstruction decoder. However, the parameters of the two decoders are not shared because they map to different target spaces. The final feature decoder yields the restored features of the masked nodes, denoted as $\mathbf{Z}_s$.

$$\mathbf{Z}_s = \mathcal{G}_f(\mathbf{H}') \tag{3}$$

3) Loss Function Formulation: We use the scaled cosine error (SCE) as the loss function and compute the loss only for the masked features on the original and reconstructed maps, as defined below:

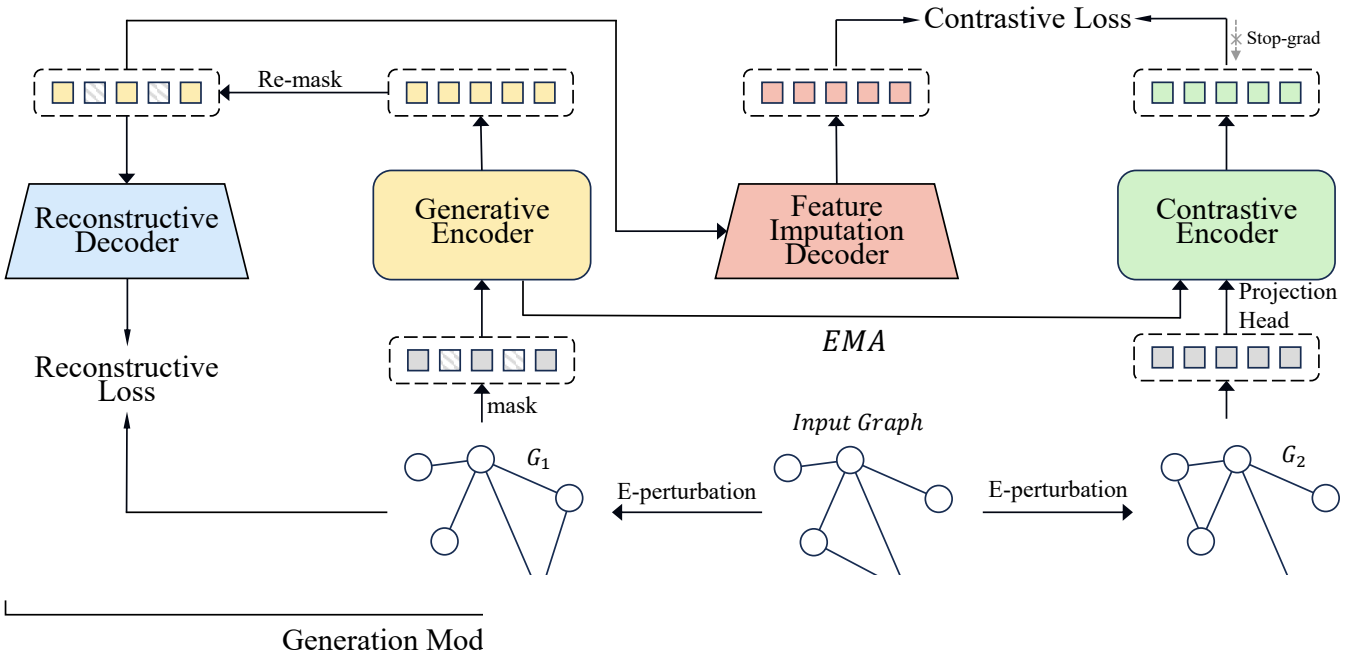
$$L_r = \frac{1}{N_{mask}} \sum_{\boldsymbol{v}_i \in \boldsymbol{V}'} \left(1 - \frac{\mathbf{z}_i^\top \mathbf{z}'_i}{\| \mathbf{z}_i \| \cdot \| \mathbf{z}'_i \|}\right)^\beta \tag{4}$$

where N_{mask} is the number of masked nodes, β is the scaling factor, $\mathbf{z}_i$ is the original map node features, $\mathbf{z}'_i$ is the predicted reconstructed node features, and L_r is the value of the loss function of the generation module.

B. Contrastive Module

We proceed to delineate specific operations and procedures of the contrastive self-supervised graph representation learning module now. For the contrastive graph representation learning module, we divide the process into three steps: 1) data augmentation of the graph to generate positive samples; 2) encoding of the generated positive sample graphs; 3) extraction of features through a projection head, followed by training through the comparison of features with those decoded by the feature completion decoder of the generation module.

1) E-perturbation Technique: Initially, we engage in edge perturbation data augmentation for graphs. Conventional graph contrast models typically employ two distinct random operations, such as node shuffling, node and edge dropout or addition, and so forth [20]. However, in our experiments, we observed detrimental effects of such data augmentation



operations on the model. We attribute this to th
of the generation module and contrast module
from the typical contrastive method input. Ext
is applied to the input of the generation modu
random data augmentation operations are per
lead to a significant disparity in the positive sar
contrastive learning, rendering them nearly ur
sulting in spurious positive samples. Consequer
contrastive training on such spurious positive pa
model from learning genuine discriminative ir

To mitigate the generation of false positive s
introduce a novel edge perturbation data augmentation method tailored for the GMCL model, as illustrated in Fig. 3. The core idea involves initially subjecting the original graph to random edge perturbation at a certain ratio to obtain an augmented graph. Subsequently, a smaller ratio is employed to randomly replace operations from the preceding edge perturbation, resulting in another augmented graph. Specifically, the process begins by randomly selecting node pairs with the condition of unidirectional edges, no self-loops, and no repetition. A smaller subset of node pairs is then randomly replaced from the previously chosen pairs, adhering to the aforementioned conditions, to yield two sets of node pairs. For pairs with existing edges, the edges are removed, and for pairs without edges, edges are added. This completes the data augmentation operation, termed $E\text{-}perturbation$, within our model.

$$(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) = E\text{-}perturbation(\mathbf{X}, \mathbf{A}) \tag{5}$$

where $(\mathbf{X}, \mathbf{A})$ is the original map and $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}})$ is the data enhanced map.

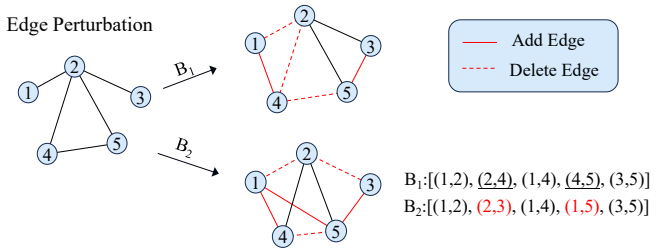


Fig. 2. Schematic diagram of $E\text{-}perturbation$.B_1, B_2 represent the selected set of perturbed edges, where underlining indicates the operation of selecting the replaced edge in the set of edges, and red marking indicates the operation of replacing it with another edge.

2) The Design of the Encoder: In the design of the encoder, we have retained the same encoder structure as the generative module. However, in the contrastive module, encoder parameter updates do not rely on backpropagation but are rather executed through exponential moving average (EMA) [21] operations conducted by the generative encoder.

$$\varphi \leftarrow \delta\varphi + (1-\delta)\theta \tag{6}$$

where θ, φ are the encoder parameters in the generation and comparison modules, respectively, and δ is the decay rate that controls the distance maintained by φ and θ.

This encoder is defined as

$$\mathbf{H_2} = \mathcal{E}_{\varphi}(\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}) \tag{7}$$

where $\mathbf{H}_2$ are the coded features, $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{X}}$ are the data-enhanced neighbourhood and node feature matrices, respectively, and φ are the encoder parameters.

3) Loss Function Formulation: Finally, we obtain features $\widetilde{\mathbf{H}}$ through a projection head, filtering out some low-level details to enhance the model's generalization ability [22]. Subsequently, we utilize the scaled cosine loss as the loss function for the contrastive module, effectively minimizing the distance between positive samples through the following equation.

$$L_c = \frac{1}{N}\sum_{i=0}^{N-1}\left(1 - \frac{\mathbf{Z}_{(s,i)}\mathbf{H}_{(2,i)}^{\top}}{\parallel \mathbf{Z}_{(s,i)} \parallel \cdot \parallel \mathbf{H}_{(2,i)}^{\top} \parallel}\right)^{\gamma} \tag{8}$$

where N is the number of nodes, γ is the scaling factor, $\mathbf{Z}_{(s,i)}$ is the embedding representation obtained by the i node through the feature decoder, $\mathbf{H}_{(2,i)}^{\top}$ is the feature obtained by the i node through the encoder in the comparison module, and L_c is the value of the loss function in the comparison module.

C. Integrated Loss Function for Joint Optimization

The final overall learning loss is a weighted sum of the reconstruction loss L_r of the generative module and the comparison loss L_c, defined as

$$L = \alpha L_c + (1-\alpha)L_r \tag{9}$$

where α is the weight hyperparameter for both losses and L is the total loss.

IV. Experiments

In this section, we conduct a performance comparison between GMCL and several leading-edge methods in the current domain of self-supervised graph learning. The evaluation encompasses two foundational tasks: node clustering and graph clustering. The experimental substantiating the effectiveness of our previously introduced theoretical approach.

A. Research questions

To better evaluate the performance of GMCL, we addressed four research questions:

(RQ1) Can the proposed GMCL model outperform competitive baselines in node clustering tasks?

(RQ2) Can the proposed GMCL model outperform competitive baselines in graph clustering tasks?

(RQ3) Do the two innovative modules in the GMCL model contribute significantly to its performance?

(RQ4) How does the performance of the GMCL model vary under different hyperparameter values α and mask ratios?

B. Experimental Setting

1) Datasets: We conducted performance assessments of GMCL on node clustering and graph clustering tasks across five benchmark datasets. Node clustering tasks involved three citation network datasets—Cora, CiteSeer, and Pubmed [23] along with a biological graph dataset, PPI, and a social network dataset, Reddit. For graph clustering tasks, we utilized three biological graph datasets—MUTAG, PROTEINS, NCI1 and two movie datasets—IMDB-B and IMDB-M [24].

2) Experimental setup: Across the ten publicly available datasets, we employed a dual-layered GNN encoder for feature extraction and a single-layered GNN as the decode. Specific GNN methods used for different datasets will be detailed in the subsequent ablation experiments section. For the five node-level task datasets, each hidden layer of the encoder integrated layer normalization [25] and PRELU [26] activation. For the graph-level tasks, the applied operations were batch normalization [27] and PRELU. Experimental standardization led to a fixed hidden layer size of 1024 for Pubmed and PPI, while the other eight datasets were set to a size of 512. For the parameter β, a grid search was conducted within the space: 0, 0.05, ..., 1. Across all datasets, Adam optimizer was utilized, with a learning rate of $1e^{-4}$ for PPI and graph-level tasks, and $1e^{-3}$ for the rest. Training epochs varied among datasets, not individually listed here, with an early stopping epoch set at 30. Reddit and PPI datasets underwent testing in the inductive setting, while the remaining datasets were evaluated under the transductive setting.

3) Evaluation: We address node classification and graph classification tasks independently in this study. For node classification tasks, we utilize the standard split ratios of publicly available datasets. In the unsupervised scenario, we train the encoder using the GMCL method proposed in this paper. Upon completion of training, we fix the parameters of the encoder, generate node features, and subsequently fit a linear classifier on these features for evaluation. In the context of graph classification tasks, post the generation of graph features using the GMCL model, we predict graph labels by fitting a LIBSVM classifier [28].

C. Results for Node Classification

We conducted a comparative analysis of GMCL against three categories of baseline tasks: (i) generative self-supervised learning methods, encompassing GAE [18], GPT-GNN [17], GATE [29], and GraphMAE; (ii) contrastive self-supervised learning methods, including DGI [7], MVGRL [30], GRACE [12], BGRL [31], InfoGCL [32], and CCA-SSG [33]; (iii) supervised methods, such as GCN and GAT. In instances where experimental parameters were provided, we meticulously replicated the experimental results using identical parameters. For models lacking publicly disclosed parameters, we determined the parameters and derived results through hyperparameter search.

Table I illustrates the results, demonstrating that the GMCL model consistently outperformed all baseline tests. For instance, in the CiteSeer dataset, GMCL outperforms the GraphMAE and the InfoGCL, respectively. This indicates that GMCL does not merely benefit from the merits of one paradigm but rather the compounded effect of both. In the PPI network, where biological graphs present unique challenges due to their intricate local and global structures. The balance between local feature capture and global structural understanding is critical in this domain, and GMCL's dual-strategy learning adeptly navigates this complexity. For large-scale graphs such as Reddit, GMCL's performance is particularly

TABLE I

EXPERIMENT RESULTS IN UNSUPERVISED REPRESENTATION LEARNING FOR NODE CLASSIFICATION. WE REPORT THE MICRO-F1 SCORE FOR PPI AND ACCURACY FOR THE OTHER DATASETS.

Dataset	Cora	CiteSeer	PubMed	PPI	Reddit
GCN	81.5	70.3	79.0	75.7±0.1	95.3±0.1
GAT	83.0±0.7	72.5±0.7	79.0±0.3	97.30±0.20	96.0±0.1
GAE	71.5±0.4	65.8±0.4	72.1±0.5	-	-
GPT-GNN	80.1±1.0	68.4±1.6	76.3+0.8	-	-
GATE	83.2±0.6	71.8±0.8	80.9±0.3	-	-
GraphMAE	84.2±0.4	73.4±0.4	81.1±0.4	74.50±0.29	96.01±0.08
DGI	82.3±0.6	71.8±0.7	76.8±0.6	63.80±0.20	94.0±0.10
MVGRL	83.5±0.4	73.3±0.5	80.1±0.7	-	-
GRACE	81.9±0.4	71.2±0.5	80.6±0.4	69.71±0.17	94.72±0.04
BGRL	82.7±0.6	71.1±0.8	79.6±0.5	73.63±0.16	94.22±0.03
InfoGCL	83.5±0.3	73.5±0.4	79.1±0.2	-	-
CCA-SSG	84.0±0.4	73.1±0.3	81.0±0.4	73.34±0.17	95.07±0.02
GMCL	**84.6±0.3**	**73.9±0.5**	**82.2±0.4**	**74.61±0.37**	**96.46±0.26**

Models that do not give results because no code is given or there is not enough memory.

TABLE II

EXPERIMENT RESULTS IN UNSUPERVISED REPRESENTATION LEARNING FOR GRAPH CLASSIFICATION. WE REPORT ACCURACY (%) FOR ALL DATASETS.

Dataset	IMDB-B	IMDB-M	PROTEINS	MUTAG	NCI
WL	72.30±3.44	46.95±0.46	72.92±0.56	80.72±3.00	80.31±0.46
DGK	66.96±0.56	44.55±0.52	73.30±0.82	87.44±2.72	80.31±0.46
GIN	75.1±5.1	52.3±2.8	76.2±2.8	89.4±5.6	82.7±1.7
DiffPool	72.6±3.9	-	75.1±3.5	85.0±10.3	-
GCC	72.0	49.4	-	-	-
graph2vec	71.10±0.54	50.44±0.87	73.30±2.05	83.15±9.25	73.22±1.81
Infograph	73.03±0.87	49.69±0.53	74.44±0.31	89.01±1.13	76.20±1.06
GraphCL	71.14±0.44	48.58±0.67	74.39±0.45	86.80±1.34	77.87±0.41
JOAO	70.21±3.08	49.20±0.77	74.55±0.41	87.35±1.02	78.07±0.47
InfoGCL	75.10±0.90	51.40±0.80	-	**91.20±1.30**	80.20±0.60
MVGRL	74.20±0.70	51.20±0.50	-	89.70±1.10	-
GraphMAE	75.52±0.66	51.63±0.52	75.30±0.39	88.19±1.26	80.40±0.30
GMCL	**76.60±0.71**	**52.01±0.83**	**75.94±0.54**	89.96±0.68	**81.17±0.24**

The reported results of baselines are from previous papers if available.

telling. The model's effectiveness in harnessing both local and global information translates into an impressive ability to scale and manage the complexity inherent in large community structures. The accuracy achieved by GMCL on Reddit suggests that the model effectively learns representations that are both informative and discriminative, even when the graph's size and intricacy intensify.

D. Results for Graph Classification

For graph classification tasks, we conduct a comparative analysis of GMCL against three categories of baseline tasks: (i) classical graph kernel methods, which encompass Weisfeiler-Lehman sub-tree kernel (WL) [34] and deep graph kernel (DGK) [35]; (ii) unsupervised and contrastive methods, including GCC [36], graph2vec [37], Infograph [38], GraphCL [11], JOAO [39], MVGRL [30], and InfoGCL [32]; (iii) supervised methods, exemplified by GIN [40] and DiffPool [41]. The results of these baseline models are directly referenced from publicly available outcomes in various publications, and the utilized public datasets align with those employed in graph classification experiments within the literature of these models.

In the context of graph classification tasks, we have conducted a comparative assessment of GMCL against three categories of baseline tasks, as delineated in Table II. The results demonstrate that GMCL outperforms all baseline models on four of the five datasets, exhibiting commendable performance on the remaining dataset. The node features for these datasets are represented as one-hot vectors, signifying either node labels or node degrees. Notably, these datasets incorporate significantly less node information compared to datasets tailored for node-level tasks. The findings imply that GMCL, even when confronted with limited node feature information, can still effectively capture meaningful graph features, yielding excellent performance in graph classification tasks.

E. Ablation Studies

In order to better validate the role of each modular component in GMCL, we conducted ablation experiments on it.

TABLE III
ABLATION STUDIES OF THE E-PERTURBATION AND FEATURE IMPUTATION DECODER.

Dataset	Node_level			Graph_level	
	Cora	Citeseer	PubMed	IMDB-B	MUTAG
GMCL	84.6	73.9	82.2	76.60	89.96
w/o E-perturbation	84.0	72.9	81.5	76.03	88.38
w/o F-I Decoder	82.6	72.1	80.5	74.43	87.16

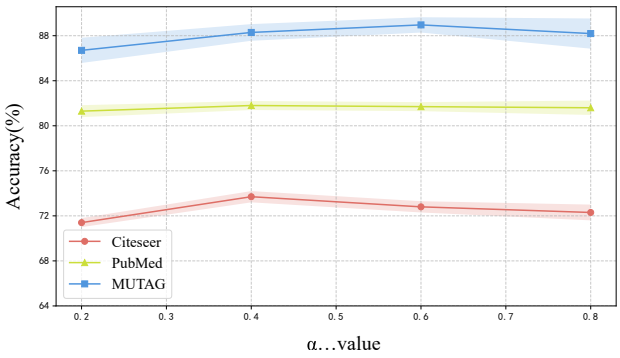
We selected three node classification datasets and two graph classification datasets for this part of the experiment.

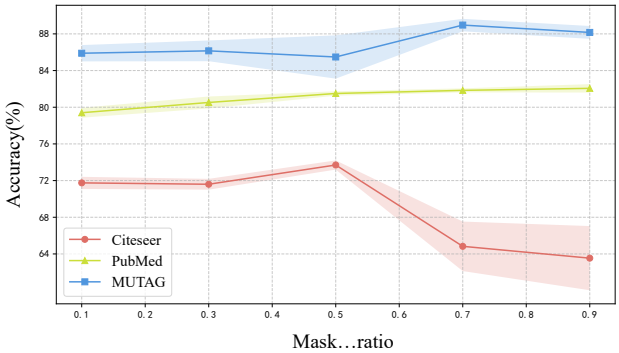
1) Effects of E-perturbation: In contrast to the extensive data augmentation commonly employed in graph contrastive learning for sample construction, we observe that moderate data augmentation yields superior results in the GMCL model. A comparison between the novel edge perturbation method introduced in this paper and conventional data augmentation methods in the model reveals, as shown in the Table III, that our proposed edge perturbation method outperforms typical data augmentation methods in the GMCL model. The edge perturbation method outlined in the paper generates more reliable positive samples, leading to improved model performance. As highlighted in the introduction to the contrastive module, this operation enhances the synergy between contrastive and generative self-supervised learning methods.

2) Effects of Feature Imputation Decoder: In order to assess the effectiveness of our feature completion decoder, we removed the decoder and utilized the output of the encoder in the generation module directly for contrastive training. The Table III highlights that the lack of a feature completion decoder yields markedly poorer results, potentially even trailing the performance of the two distinct algorithms. This is due to the incomplete semantics embedded in each masked embedding, posing significant challenges when using these features directly for contrastive learning. The incorporation of the feature completion decoder addresses this issue, thereby promoting the effectiveness of the contrastive module in learning discriminative features.

F. Parameter Analysis

1) Effects of Contrastive Module and Reconstruction Module: We examined the influence of the generation module and the contrastive module. The influence of the α weighting factor is illustrated in Fig. 4. It is evident that setting the weight to 0 transforms GMCL into a generative self-supervised model. As α increases, the optimal performance is observed at α = 0.4 on the Citeseer and Pubmed datasets, while on the MUTAG dataset, the optimal performance is achieved at α = 0.6. This phenomenon may be attributed to the richer node feature information in the Citeseer and Pubmed datasets, enabling the reconstruction of local information to capture more node and global information. In contrast, on the MUTAG dataset, the emphasis lies more on discriminative information





(b) Accuracy values with respect to Mask ratio on Citeseer, PubMed and MUTAG.

Fig. 3. The impact of parameter tuning on GMCL accuracy.

among different graph samples. This further affirms that the GMCL model effectively integrates the strengths of generative and contrastive self-supervised learning models.

2) Effects of Mask Ratio: Fig. 4 illustrates the impact of mask ratios on the performance of the model. It is evident that, in most scenarios, low mask ratios do not yield satisfactory results. It attributed to the challenge faced by the generation module in learning meaningful reconstruction features from such ratios. The optimal mask ratio varies across different datasets. For instance, on the Citeseer dataset, a mask ratio of 0.5 already achieves optimal results. However, on PubMed and MUTAG datasets, GMCL requires mask ratios of 0.7 or even 0.9 for optimal performance. This variation may be linked to the presence of repetitive information in the graphs. High node degrees or graph homogeneity can result in an excess of redundant information, allowing reconstruction features to rely on minimal neighboring node information for recovery, resulting in inadequate comprehension of global information. Conversely, when the graph contains less redundant information due to lower node degrees, more nodes are needed to provide information for reconstruction. In such cases, an excessively high mask ratio may impede the recovery of node features.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduce a novel graph self-supervised learning method called Graph Masked Contrastive Learn-

ing (GMCL). The approach aims to enhance the quality of graph representation learning by integrating generative and contrastive self-supervised learning methods. In GMCL, we meticulously design the structure and input generation aspects to better fuse these two classes of models. Extensive experiments on node and graph classification datasets demonstrate that GMCL effectively improves the quality of representations learned through graph self-supervised learning. In our forthcoming endeavors, we plan to extend the work on GMCL along two promising directions: We aim to enhance the computational efficiency of GMCL. This may involve optimizing the existing algorithms for better hardware utilization, employing techniques like quantization and knowledge distillation, or leveraging more advanced parallel processing techniques. Extending GMCL to handle dynamic graphs is another avenue, where the model must adapt to graphs that evolve over time. This requires developing mechanisms that can update the graph representation incrementally as new nodes and edges are added or removed.

REFERENCES

[1] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, "Self-supervised representation learning: Introduction, advances, and challenges," *IEEE Signal Process. Mag.*, vol. 39, no. 3, pp. 42–62, 2022.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR 2013*, 2013.

[3] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[4] Z. Pan, F. Cai, W. Chen, and H. Chen, "Graph co-attentive session-based recommendation," *ACM Trans. Inf. Syst.*, vol. 40, no. 4, pp. 67:1–67:31, 2022.

[5] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 857–876, 2023.

[6] Z. Pan, F. Cai, W. Chen, C. Chen, and H. Chen, "Collaborative graph learning for session-based recommendation," *ACM Trans. Inf. Syst.*, vol. 40, no. 4, pp. 72:1–72:26, 2022.

[7] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR 2019*, 2019.

[8] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, and P. S. Yu, "Graph self-supervised learning: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5879–5900, 2023.

[9] Z. Qi, R. Dong, G. Fan, Z. Ge, X. Zhang, K. Ma, and L. Yi, "Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining," in *International Conference on Machine Learning (ICML)*, 2023.

[10] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *CVPR 2015*, 2015, pp. 4353–4361.

[11] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS 2020*, 2020.

[12] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *CoRR*, vol. abs/2006.04131, 2020.

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR 2022*, 2022, pp. 15 979–15 988.

[14] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li, "Self-supervised learning on graphs: Contrastive, generative, or predictive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4216–4235, 2023.

[15] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *ICML 2018*, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 5694–5703.

[16] J. You, B. Liu, Z. Ying, V. S. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *NeurIPS 2018*, 2018, pp. 6412–6422.

[17] Z. Hu, Y. Dong, K. Wang, K. Chang, and Y. Sun, "GPT-GNN: generative pre-training of graph neural networks," in *KDD '20*, 2020, pp. 1857–1867.

[18] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *CoRR*, vol. abs/1611.07308, 2016.

[19] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *KDD '22*, 2022, pp. 594–604.

[20] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," in *NeurIPS 2021*, 2021, pp. 15 920–15 933.

[21] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML 2019*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 6105–6114.

[22] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, vol. 119, 2020, pp. 1597–1607.

[23] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML 2016*, ser. JMLR Workshop and Conference Proceedings, vol. 48, 2016, pp. 40–48.

[24] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *SIGKDD 2015*, 2015, pp. 1365–1374.

[25] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV 2015,*, 2015, pp. 1026–1034.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML 2015*, vol. 37, 2015, pp. 448–456.

[28] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[29] A. Salehi and H. Davulcu, "Graph attention auto-encoders," in *ICTAI 2020*, 2020, pp. 989–996.

[30] K. Hassani and A. H. K. Ahmadi, "Contrastive multi-view representation learning on graphs," in *ICML 2020*, vol. 119, 2020, pp. 4116–4126.

[31] S. Thakoor, C. Tallec, M. G. Azar, M. Azabou, E. L. Dyer, R. Munos, P. Velickovic, and M. Valko, "Large-scale representation learning on graphs via bootstrapping," in *ICLR 2022*, 2022.

[32] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "Infogcl: Information-aware graph contrastive learning," in *NeurIPS 2021*, 2021, pp. 30 414–30 425.

[33] H. Zhang, Q. Wu, J. Yan, D. Wipf, and P. S. Yu, "From canonical correlation analysis to self-supervised graph neural networks," in *NeurIPS 2021*, 2021, pp. 76–89.

[34] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, 2011.

[35] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *SIGKDD 2015*, 2015, pp. 1365–1374.

[36] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "GCC: graph contrastive coding for graph neural network pre-training," in *KDD '20*, 2020, pp. 1150–1160.

[37] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *CoRR*, vol. abs/1707.05005, 2017.

[38] F. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR 2020*.

[39] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *ICML 2021*, ser. Proceedings of Machine Learning Research, vol. 139, 2021, pp. 12 121–12 132.

[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR 2019*, 2019.

[41] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *NeurIPS 2018*, 2018, pp. 4805–4815.