Generative and Contrastive Paradigms Are Complementary for Graph Self-Supervised Learning

Yuxiang Wang[†,*] Xiao Yan[#,*] Chuang Hu[†,✉] Quanqing Xu[§] Chuanhui Yang[§]
Fangcheng Fu[‡] Wentao Zhang[‡] Hao Wang[†] Bo Du[†] Jiawei Jiang[†,✉]
[†] School of Computer Science, Wuhan University [#] Centre for Perceptual and Interactive Intelligence (CPII)
[§] OceanBase [‡] School of Computer Science, Peking University
[†] {nai.yxwang,handc,wanghao.cs,dubo,jiawei.jiang}@whu.edu.cn [#] yanxiaosunny@gmail.com
[§] {xuquanqing.xqq,rizhao.ych}@oceanbase.com [‡] {ccchengff,wentao.zhang}@pku.edu.cn

2024 IEEE 40th International Conference on Data Engineering (ICDE) | 979-8-3503-1715-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDE60146.2024.00234

***Abstract*—For graph self-supervised learning (GSSL), *masked autoencoder* (MAE) follows the generative paradigm and learns to reconstruct masked graph edges or node features while *contrastive learning* (CL) maximizes the similarity between augmented views of the same graph. Existing works utilize MAE and CL separately but we observe that the MAE and CL paradigms are complementary and propose the *graph contrastive masked autoencoder* (GCMAE) framework to unify them. Specifically, by focusing on local edges or node features, MAE cannot capture global information of the graph and is sensitive to particular edges and features. On the contrary, CL excels in extracting global information because it considers the relation between graphs. As such, we equip GCMAE with an MAE branch and a CL branch, and the two branches share a common encoder, which allows the MAE branch to exploit the global information extracted by the CL branch. To force GCMAE to capture global graph structures, we train it to reconstruct the entire adjacency matrix instead of only the masked edges as in existing works. Moreover, a discrimination loss is proposed for feature reconstruction, which improves the disparity between node embeddings rather than reducing the reconstruction error to tackle the feature smoothing problem of MAE. We evaluate GCMAE on four popular graph tasks (i.e., node classification, node clustering, link prediction, and graph classification) and compare it with 14 state-of-the-art baselines. The results show that GCMAE consistently provides good accuracy across these tasks, and the maximum accuracy improvement is up to 3.2% compared with the best-performing baseline.**

***Index Terms*—graph learning, self-supervised learning, masked autoencoder, contrastive learning**

I. Introduction

By modeling entities as nodes and the relations between the entities as edges, graphs prevail in many domains such as social networks [1, 2], finance [3], biology [4], and medicine [5]. Graph neural networks (GNNs) conduct message passing on the edges and utilize neural networks to aggregate messages on the nodes. GNNs yield good performance for various graph processing tasks, e.g., node classification, node clustering, link prediction, and graph classification [1, 6, 7, 8, 9, 10], which have long been hot topics in both database and data engineering [8, 11, 12, 13, 14, 15, 16, 17]. To train GNN models, *graph self-supervised learning* (GSSL) is becoming increasingly popular because it does not require label information [18, 19], which can be rare or expensive to obtain in practice. Existing works conduct GSSL with two main paradigms [20, 21], i.e., *masked autoencoder* (MAE) and *contrastive learning* (CL).

[✉] Corresponding author. [*] These authors contributed equally to this work.

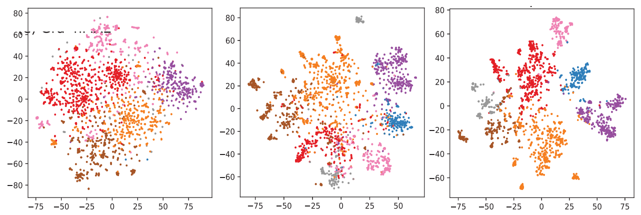
(a) CCA-SSG (NMI=56.38) (b) GraphMAE (NMI=58.33) (c) GCMAE (NMI=59.31)

Fig. 1. We conduct node clustering on the Cora graph and visualize the node embeddings learned by GraphMAE, CCA-SSG, and our GCMAE with t-SNE. GraphMAE and CCA-SSG represent MAE and CL methods, respectively. Nodes of the same category are plotted in the same color. We use Normalized Mutual Information (NMI) to evaluate the clustering effect, with a larger value indicating better accuracy.

Graph MAE methods usually come with an encoder and a decoder, which are both GNN models [21, 22, 23, 24, 25]. The encoder computes node embeddings from a corrupted view of the graph, where some of the edges or node features are masked; the decoder is trained to reconstruct the masked edges or node features using these node embeddings. Graph MAE methods achieve high accuracy for graph tasks but we observe that they still have two limitations. ❶ Graph MAE *misses global information* of the graph, which results in sub-optimal performance for tasks that require a view of the entire graph (e.g., graph classification) [26, 27]. This is because both the encoder and decoder GNN models use a small number of layers (e.g., 2 or 3), and a k-layer GNN can only consider the k-hop neighbors of each node [28]. Thus, both the encoder and decoder obtain information from the local neighborhood. ❷ Graph MAE is prone to *feature smoothing*, which means that neighboring nodes tend to have similar reconstructed features and harm the performance of tasks that rely on node features (e.g., node classification). This is because the encoder and decoder GNN models aggregate the neighbors to compute the embedding for each node; and due to this local aggregation, GNN models are widely known to suffer from the over-smoothing problem [28, 29], which also explains why GNN

TABLE I
PERFORMANCE COMPARISON WITH THE BEST-PERFORMING BASELINES IN EACH CATEGORY. *Others* ARE SPECIALIZED METHODS FOR EACH TASK. RESULTS IN THE PARENTHESES ARE THE RESULTS OF A NAIVE CONTRASTIVE MAE DESIGN VERSUS THE BASELINES. RESULTS OUT OF THE PARENTHESES ARE THE RESULTS OF OUR GCMAE.

Graph Task	vs. Contrastive	vs. MAE	Others
Node classification	4.8% (-8.2%)	2.2% (-9.6%)	12.0% (2.1%)
Link prediction	4.4% (-9.4%)	1.5% (-12.1%)	-
Node clustering	8.8% (-14.8%)	3.2% (-18.8%)	14.7% (-18.9%)
Graph classification	2.5% (-20.1%)	4.2% (-21.0%)	-

models normally do not use many layers.

Graph CL methods usually generate multiple perturbed views of a graph via data augmentation and train the model to maximize the similarity between positive pairs (e.g., an anchor node and its corresponding node in another view) and minimize the similarity between negative pairs (e.g., all other nodes except positive pairs) [30, 31]. As graph CL can contrast nodes that are far apart (not limited to 3 hops) in multiple views, it can capture global information of the entire graph. However, without the ability of reconstructing the node features and edges, CL cannot well learn local information for the nodes and edges, and thus may have sub-optimal performance compared to MAE for tasks that require such information (e.g., link prediction and node clustering) [18, 32].

The above analysis shows that graph MAE and CL methods can potentially complement each other in capturing both local and global information of the graph. Thus, combining MAE and CL may yield better performance than using them individually as in existing works. Figure 1 shows such an example with node clustering. As a representative graph CL method, CCA-SSG [33] shows poor clustering performance for the nodes due to the lack of local information. For GraphMAE [22], a state-of-the-art graph MAE method, the node clusters are more noticeable by capturing local information. However, by combining MAE and CL, the model yields the best node clustering and hence the highest accuracy among the three methods (see Figure 1(c)). Thus, the research problem becomes—*how to design a unified framework that enables MAE and CL to benefit from each other?*

As shown in Figure 2 (c), a *naive* and *straightforward* design is to splice MAE and CL together with two independent GNN models. Specifically, the MAE branch computes the reconstruct loss between the original graph and the graph decoded from the latent embeddings; the CL branch encodes corrupted graphs and contrasts the outputs of the corrupted graphs. This design allows the gradients from MAE and CL to update the node embeddings. However, as shown inside the parentheses of Table I, this naive design yields lower accuracy than both MAE and contrastive methods on all four graph tasks. This is because MAE and CL have different graph augmentation logic and learning objectives, and focus on local and global information, respectively. Without a proper design of the model structure, MAE and CL may contradict instead of benefiting each other.

To enjoy the benefits of both MAE and CL, we design the *graph contrastive masked autoencoder* (i.e., GCMAE) framework. We first show that MAE and CL can be unified in one mathematical framework despite their apparent differences. In particular, we express MAE as a special kind of CL, which uses masking for data augmentation and maximizes the similarity between the original and masked graphs. This analysis inspires us to use *an MAE branch* and *a CL branch* in GCMAE , and share *the encoder* for the two branches. The MAE branch is trained to reconstruct the original graph while the CL branch is trained to contrast two augmented views; and the shared encoder is responsible for mapping the graphs (both original and augmented) to the embedding spaces to serve as inputs for the two branches. With this design, the shared encoder performs input mapping and transfers information between the two branches. As such, local information and global information are condensed in the shared encoder and benefit both MAE and CL. Furthermore, to tackle the feature smoothing problem of MAE that makes the reconstructed features of neighboring nodes similar, we introduce a *discrimination loss* for model training, which enlarges the variance of the node embeddings to avoid smoothing. Instead of training MAE to reconstruct only the masked edges as in existing works, we reconstruct the entire adjacency matrix such that MAE can learn to capture global information of the graph structures.

We evaluate our GCMAE on four popular graph tasks, i.e., node classification, node clustering, link prediction, and graph classification. These tasks have essentially different properties and require different information in the graph. We compare GCMAE with 14 state-of-the-art baselines, including graph MAE methods such as GraphMAE [22], SeeGera [21], S2GAE [25], and MaskGAE [24], and graph CL methods such as GDI [2], MVGRL [34], GRACE [35], and CCA-SSG [33]. The results show that GCMAE yields consistently good performance on the four graph tasks and is the most accurate method in almost all cases (i.e., a case means a dataset plus a task). As shown in Table I, GCMAE outperforms both contrastive and MAE methods, and the improvements over the best-performing baselines are significant. We also conduct an ablation study for our model designs (e.g., shared encoder and the loss terms), and the results suggest that the designs are effective in improving accuracy.

To summarize, we make the following contributions:

- We observe the limitations of existing graph MAE and CL methods for graph self-supervised learning and propose to combine MAE and CL for enhanced performance.
- We propose the GCMAE framework, which comes with tailored model designs such as *shared encoder* for information sharing between MAE and CL, and *discrimination loss* to address feature smoothing.
- We conducted extensive experiments to evaluate GCMAE and compare it with state-of-the-art methods, demonstrating that GCMAE is general across various graph tasks and achieves significantly better accuracy.

II. Preliminaries

In this part, we introduce the basics of MAE and CL for GSSL to facilitate our formulation. The discussions on specific MAE and CL methods are provided in Section VI. We use $G = (\boldsymbol{V}, \boldsymbol{E})$ to denote a graph, where $\boldsymbol{V} = \{v_1, v_2, \cdots, v_N\}$ is the set of nodes with $|\boldsymbol{V}| = N$ and $\boldsymbol{E}$ is the set of edges. The adjacency matrix of the graph is $\boldsymbol{A} \in \{0, 1\}^{N\times N}$, and the node feature matrix is $\boldsymbol{X} \in \mathbb{R}^{N\times d}$, where $x_i \in \mathbb{R}^d$ is the feature vector of v_i with dimension d and $\boldsymbol{A}_{ij} = 1$ if edge $(v_i, v_j) \in \boldsymbol{E}$.

A. Contrastive Learning for GSSL

CL has been very popular for GSSL due to its good performance for various application scenarios, such as social network analysis [33], molecular detection [36], and financial deception [37]. Generally, graph CL follows an "augmenting-contrasting" pattern, where the graph views are corrupted via the data augmentation strategies $\mathcal{T}_1$ and $\mathcal{T}_2$ (e.g., node feature masking and node dropping), and then a GNN encoder f_θ is used to encode the corrupted views into node embeddings. The goal of CL is to maximize the mutual information between the two corrupted views, which is achieved by maximizing the similarity of node embeddings as a surrogate. Thus, CL does not require label information and conducts learning by distinguishing between node embeddings in different views. Figure 2 (a) shows the framework of graph contrastive learning. The objective of CL is essentially to maximize the *similarity function* between two node embeddings encoded through a GNN encoder f_θ, which can be formulated as:

$$\max_{\theta} \mathop{\mathbb{E}}_{x\sim\mathcal{D}} S\left(f_\theta\left(\mathcal{T}_1\left(\boldsymbol{A}, \boldsymbol{X}\right)\right), f_\theta\left(\mathcal{T}_2\left(\boldsymbol{A}, \boldsymbol{X}\right)\right)\right), \tag{1}$$

where θ represents the network weights, node feature vector x follows the input data distribution $\mathcal{D}$, and $S(\cdot, \cdot)$ is the similarity function to measure the similarity between node embeddings.

B. Masked Autoencoder for GSSL

Different from graph CL methods, graph MAE follows a "masking-reconstructing" pattern. The overview of graph MAE methods is shown in Figure 2 (b). In general, graph MAE methods first randomly mask node features or edges with mask patches $\boldsymbol{M}$, where $x \odot \boldsymbol{M}$ represents visible tokens and $x \odot (1 - \boldsymbol{M})$ means masked tokens. Graph MAE leverages the encoder f_θ to encode the visible tokens into node embeddings, and then the decoder d_ϕ attempts to reconstruct the masked tokens by decoding from the node embeddings. Thus, we wish to maximize the similarity between the masked tokens and the reconstructed tokens:

$$\max\nolimits_{\theta,\phi} \mathop{\mathbb{E}}_{x\sim\mathcal{D}} S(d_\phi(h) \odot (1-\boldsymbol{M}), x \odot (1-\boldsymbol{M})), h = f_\theta(x \odot \boldsymbol{M}), \tag{2}$$

where $\odot$ means element-wise product, h is the node embedding learned by the encoder, and $S(\cdot, \cdot)$ is the *similarity function* for MAE modeling.

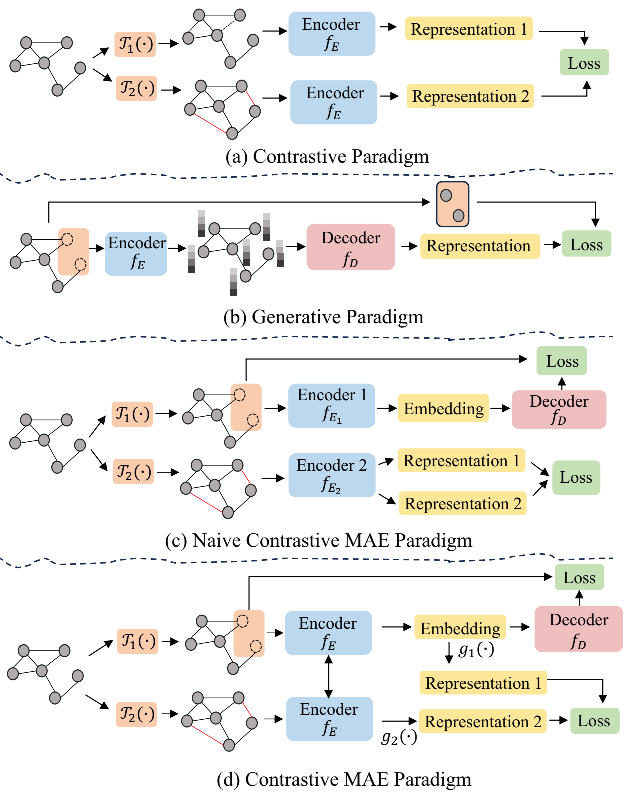

Fig. 2. Architectures of naive contrastive MAE paradigm, contrastive paradigm, generative paradigm, and our GCMAE for graph self-supervised learning.

III. The GCMAE Framework

Since graph MAE cannot benefit from global information, it can only learn from limited neighbor nodes, which may eventually lead to similar graph representations. We are inspired by the successful application of CL, where global information can be learned by contrasting the anchor node with distant nodes. The overall framework of GCMAE is shown in Figure 3.

A. Unifying CL and MAE for GSSL

Even though contrastive and generative approaches have achieved individual success, there is a lack of systematic analysis regarding their correlation and compatibility in one single framework. Motivated as such, our aim is to explore a *unified* Contrastive MAE paradigm to combine contrastive paradigm and MAE paradigm. From Equation 2, we can conclude that graph MAE essentially maximizes the similarity between node embeddings reconstructed via decoder and the masked tokens, which is formally analogous to Equation 1. In other words, both the contrastive paradigm and the generative paradigm maximize the similarity between two elements within the function.

Let us still focus on Equation 2, graph MAE wishes to find a theoretically optimal decoder that can reconstruct the masked tokens losslessly. Suppose that we can achieve the decoder $d'_{\phi'}$ parameterized by ϕ' that satisfies $d'_{\phi'}\left(f_\theta\left(x \odot \left(1 - \boldsymbol{M}\right)\right)\right) \cdot (1 - \boldsymbol{M}) \approx x \odot (1 - \boldsymbol{M})$ as closely as possible. Then, we transform Equation 2 into the following form:

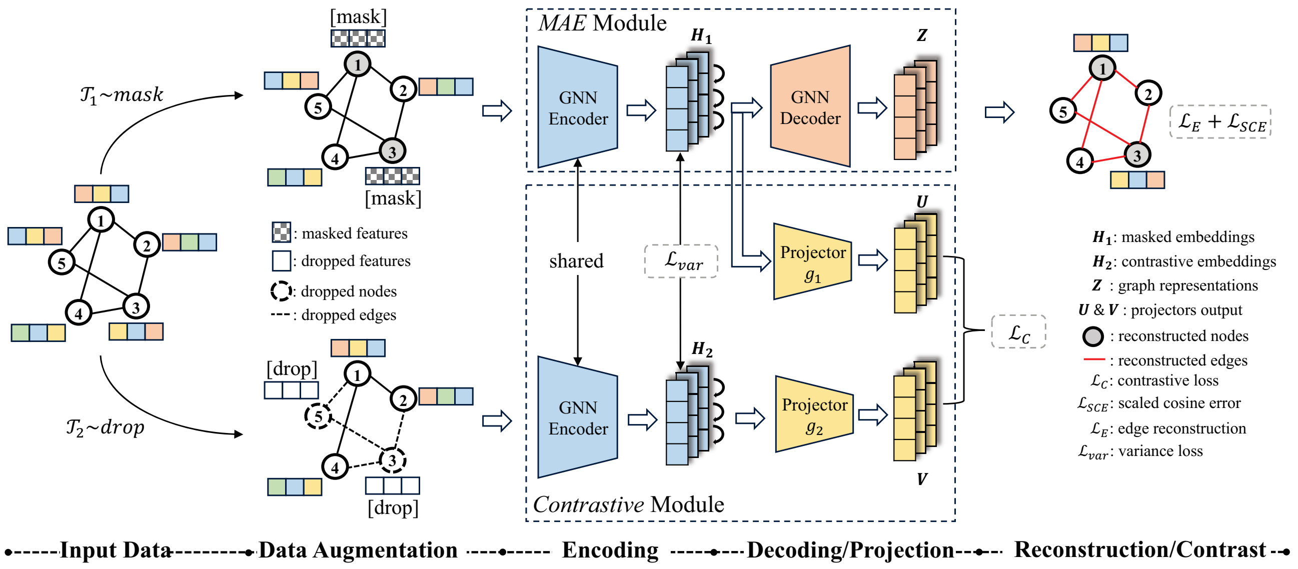

Fig. 3. Structure of our GCMAE framework, which consists of an MAE branch and a contrastive branch with a shared encoder.

$$\max_{\theta,\phi,\phi'} S(d_\phi(h) \odot (1-\boldsymbol{M}) - d'_{\phi'}(h) \odot (1-\boldsymbol{M})), \tag{3}$$

where ϕ' is optimized in the following form:

$$\begin{aligned}\phi' = \arg\max_{\phi'} \mathop{\mathbb{E}}_{x'\sim\mathcal{D}} & S(d'_{\phi'}(f_\theta(x' \odot (1-\boldsymbol{M}))) \\ & \odot (1-\boldsymbol{M}) - x' \odot (1-\boldsymbol{M})),\end{aligned} \tag{4}$$

where x' is the feature token reconstructed by the optimal decoder $d'_{\phi'}$. Notice that since d and d' have the same architecture, we let $d = d'$. Inspired by [38], we further simplify Equation 3, and define the loss function for MAE:

$$\mathcal{L}(h_1, h_2, \phi, \phi') = \max_{\phi,\phi'} S(d_\phi(h_1), d_{\phi'}(h_2)) \odot (1-\boldsymbol{M}), \tag{5}$$

where h_1, h_2 are the hidden embeddings derived from two augmentation strategies:

$$\begin{cases} h_1 = f_\theta(\mathcal{T}_1(x)) = f_\theta(x \odot \boldsymbol{M}), \\ h_2 = f_\theta(\mathcal{T}_2(x)) = f_\theta(x \odot (1-\boldsymbol{M})) \end{cases} \tag{6}$$

In this way, we rewrite the generative paradigm into a form similar to the contrastive paradigm, both with data augmentation and optimizing a similarity function.

In order to integrate MAE and CL into one optimization framework, we propose a novel self-supervised paradigm, called *Contrastive MAE*. As shown in Figure 2(d), the MAE branch is trained to reconstruct the masked tokens using similarity function S_1 while the CL branch is trained to contrast two augmented views using similarity function S_2:

$$\begin{aligned}\mathcal{L}(x, \boldsymbol{M}, h_1, h_2, \theta, \phi) = \max_{\theta,\phi} & S_1(d_\phi(h_1), x \odot (1-\boldsymbol{M})) \\ & + S_2(h_1, h_2).\end{aligned} \tag{7}$$

Summary. Despite the recent attempts by contrastive MAE methods [39, 40], understanding their underlying rationale remains an open question. Therefore, we provide a theoretical analysis showing that the nature of the generative paradigm is essentially similar to that of CL. Rather than treating them separately, these two paradigms can potentially complement each other, compensating for their respective shortcomings and obtaining better performance. Moreover, both the generative and contrastive paradigms share similar optimization objectives, which enables us to optimize them within a unified framework.

B. Structure of the GCMAE Framework

Objective: In this paper, we aim to unify graph MAE and CL into a single framework and further reveal the intrinsic relation between graph MAE and CL, where CL can help graph MAE to achieve global information, through which we hope to further improve the performance of graph MAE on various downstream tasks.

Overview: We propose a novel graph SSL framework to incorporate CL into graph MAE. As shown in Figure 3, our framework consists of two core modules: the MAE Module and the Contrastive Module, which are used to reconstruct masked nodes and contrast two corrupted views, respectively. To achieve our design goal, we introduce three main components:

- **Shared encoder.** We take the masked graph in Graph MAE as an augmented view, and then generate another view by randomly dropping nodes. The encoding phase is shown in Figure 3, we leverage a shared encoder to connect both modules and encode the corrupted views into hidden node embeddings. In this way, the contrastive module can transfer the global information to the MAE module. The node embedding generated from MAE mod-

ule is used to calculate feature reconstruction loss using Scaled Cosine Error (SCE) $\mathcal{L}_{SCE}$, and another node embedding is leveraged to calculate contrastive loss $\mathcal{L}_C$.
- **Discrimination loss.** Low discriminative input node features will cause graph MAE to generate similar node representations. Thus, we propose a novel discrimination loss $\mathcal{L}_{Var}$, which improves feature discriminability by increasing the variance between node hidden embeddings (i.e., the output of the shared encoder).
- **Adjacency matrix reconstruction.** In order to further improve performance on link prediction and node clustering, we reconstruct the entire graph and calculate the adjacency matrix reconstruction error $\mathcal{L}_E$. This is because only reconstructing the masked edges is not enough to learn the entire graph structures.

Therefore, the overall loss function for GCMAE is defined as follows:

$$\mathcal{J} = \mathcal{L}_{SCE} + \alpha\mathcal{L}_C + \lambda\mathcal{L}_E + \mu\mathcal{L}_{Var}, \tag{8}$$

where α, λ and μ are hyper-parameter used to adjust the weights. In summary, the contrastive module learns global information by contrasting two corrupted views and transfers it to the MAE module with a shared encoder. Before decoding, a discrimination loss is leveraged to improve the discrimination between node embeddings to avoid yielding similar node representations. We reconstruct both the node features and the adjacency matrix in the MAE module.

IV. Model Designs

In this section, we present a detailed description of the proposed GCMAE. We introduce each core component individually with the following three questions in mind.

- **Q1:** How to learn global information for graph MAE?
- **Q2:** How to train a decent encoder to learn the entire graph structures?
- **Q3:** How to enhance the feature discrimination?

Recently, GraphMAE [22] has attracted great attention in graph SSL due to its simple but effective framework. Therefore, we choose GraphMAE as the backbone in this paper. For a node set $\boldsymbol{V}$, we randomly select a subset of node $\widetilde{\boldsymbol{V}} \subseteq \boldsymbol{V}$ and mask their corresponding features $\boldsymbol{X}$, i.e., setting its feature values to 0. We sample $\widetilde{\boldsymbol{V}}$ following a specific distribution, i.e., Bernoulli distribution. This masking strategy is a common but effective strategy in a wide range of previous applications [18, 41]. We consider this node masking operation as a random data augmentation in graph CL methods [7, 42]. Then the node feature $\hat{x_i}$ for node $v_i \in \widetilde{\boldsymbol{V}}$ in the masked feature mask $\hat{\boldsymbol{X}}$ can be defined as follow:

$$\hat{x}_i = \begin{cases} 0, & v_i \in \widetilde{\boldsymbol{V}} \\ x_i, & v_i \in \boldsymbol{V} \end{cases} \tag{9}$$

Further, given an encoder f_E and GNN decoder f_D:

$$\boldsymbol{H_1} = f_E(\boldsymbol{A}, \hat{\boldsymbol{X}}) \quad \boldsymbol{Z} = f_D(\boldsymbol{A}, \boldsymbol{H_1}), \tag{10}$$

where $\boldsymbol{H_1} \in \mathbb{R}^{N\times d^h}$ is the hidden embedding of the feature masking view encoded by f_E and then used as the input of GNN decoder f_d to obtain the node representations $\boldsymbol{Z} \in \mathbb{R}^{N\times d'}$. d^h is the dimension of hidden embedding. Then we calculate the SCE loss:

$$\mathcal{L}_{\text{SCE}} = \frac{1}{|\widetilde{V}|}\sum_{v_i\in\widetilde{V}}(1 - \frac{x_i^T h_i}{\|x_i\|\cdot\|z_i\|})^\gamma, \gamma > 1, \tag{11}$$

where γ is leveraged to adjust the hyperparameter of loss convergence speed, h_i is the hidden embedding vector of node v_i, and z_i is the node representation of node v_i.

A. Shared Encoder

We reveal the intrinsic correlation between CL and Graph MAE and mathematically analyze the feasibility of jointly optimizing them in Section III-A. However, how to efficiently transfer global information without complicated structural design is still an unsolved problem. Therefore, we ask here:

how to transfer global information from the contrastive module to the MAE module?

An intuitive solution is to train two encoders to encode the CL branch and the MAE branch separately, and then fuse the learned node embeddings. However, this inevitably faces two issues: ❶ training two codes independently cannot realize information transfer between branches, ❷ the strategy of fusion embedding will introduce an unnecessary design burden. Moreover, the accuracy performance depends on the lower bound of either embedding, and low-quality node embeddings will directly affect the performance.

To tackle the problem, we introduce a shared encoder that simultaneously encodes two augmented views to learn local and global information. The contrastive module leverages the shared encoder to transfer the global information from the entire graph to assist Graph MAE in learning meaningful representations and fine-tuning local network parameters. We do not need an additional embedding fusion strategy, since the shared encoder can directly yield the unify node embeddings. Thus, we first use the shared encoder f_E to encode the corrupted graph $\bar{\boldsymbol{A}}$ augmented by random node dropping into hidden embeddings:

$$\boldsymbol{H_2} = f_E(\bar{\boldsymbol{A}}, \boldsymbol{X}), \tag{12}$$

where $\boldsymbol{H_2} \in \mathbb{R}^{N\times d^h}$. Then, we utilize two projectors g_1, g_2 to map $\boldsymbol{H_1}$,$\boldsymbol{H_2}$ into different vector spaces:

$$\boldsymbol{U} = \sigma(g_1(\boldsymbol{H_1}, \psi_1)), \quad \boldsymbol{V} = \sigma(g_2(\boldsymbol{H_2}, \psi_2)), \tag{13}$$

where $\sigma(\cdot)$ is a nonlinear activation function. The projector consists of a simple two-layer perceptron model with bias ψ_1, ψ_2. We use the classical InfoNCE as the contrastive loss function, defined for each positive sample pair (u_i, v_i) as:

$$\ell_c = -\log\frac{\exp(s(u_i^a, v_i^b)/\tau)}{\sum_{j=1}^n[\exp(s(u_i^a, v_j^a))/\tau + \exp(s(u_i^a, v_j^b)/\tau)]}, \tag{14}$$

where τ is the temperature parameter and $s(\cdot)$ is the cosine similarity between u_i and v_i, while $u_i^a \in \boldsymbol{U}$ and $u_j^b \in \boldsymbol{V}$ denote the projected vector of v_i and v_j, respectively. Since the two projectors are symmetric, the loss for the node pair (v_i, u_i) is defined similarly. The overall objective of the contrastive module is to maximize the average mutual information of all positive sample pairs in the two views. Formally, it is defined as:

$$\mathcal{L}_C = \frac{1}{2N}\sum_{i=1}^{N}[\ell_c(u_i, v_i) + \ell_c(v_i, u_i)]. \tag{15}$$

B. Adjacency Matrix Reconstruction

Graph data exhibits a unique property of complex topology when compared to images and text, as graph contains complex graph topology and node features. Several studies have focused on adjacency matrix reconstruction [43, 44]. Since it not only helps models pay more attention to link relationships but also facilitates the encoder to get rid of the constraints brought by fitting extreme feature values [45]. By incorporating the adjacency matrix, the overall objective lightens the weight of feature reconstruction, thereby eliminating the incomplete learned knowledge caused by a single objective. Different from MaskGAE [24] which reconstructs limited edges, we reconstruct node features and adjacency matrices at the same time, which can help the model capture the global graph structures rather than particular edges or paths. In this work, we employ the output representation of the decoder to directly reconstruct the adjacency matrix:

$$\ell_{MSE} = \frac{1}{N^2}\sum_{i,j}(\hat{\boldsymbol{A}}_{ij} - \boldsymbol{A}_{ij})^2, \hat{\boldsymbol{A}} = \left\langle \boldsymbol{Z}, \boldsymbol{Z}^T \right\rangle = \boldsymbol{Z}\boldsymbol{Z}^T, \tag{16}$$

where $\hat{\boldsymbol{A}}_{ij}$ is the probability of an edge existing between nodes v_i and v_j. The MSE function measures the distance between the reconstructed adjacency matrix and the original adjacency matrix, ensuring consistency between the latent embedding and the input graph structure. However, due to the discretization and sparsity of the adjacency matrix, solely using MSE would make the model overfit to zero values. Therefore, we adopt Cross-Entropy (CE) to determine the existence of an edge between two nodes:

$$\ell_{BCE} = -\frac{1}{N^2}\sum_{i,j}[\boldsymbol{A}_{ij} \cdot \log \hat{\boldsymbol{A}}_{ij} + (1 - \boldsymbol{A}_{ij}) \cdot \log(1 - \hat{\boldsymbol{A}}_{ij})]. \tag{17}$$

In addition to the above issues, graph data is low-density and the degree distribution follows a power-law distribution, which cannot be adequately estimated by simple MSE and CE. The coexistence of both low-degree and high-degree nodes further renders Euclidean spaces invalid and hampers the training speed. To address this problem, we put forth the Relative Distance (RD) loss function:

$$\ell_{DIST} = -\log\frac{\sum_{(i,j)\in \boldsymbol{E}} D(z_i, z_j)}{\sum_{(i,j)\notin \boldsymbol{E}} D(z_i, z_j)}, \tag{18}$$

where $D(\cdot,\cdot)$ defines the distance between node v_i and node v_j, and z is the node representation of the decoder in the MAE module. The numerator and denominator in the RD loss function correspond to the sum of distances between adjacent nodes and non-adjacent nodes, respectively. Reconstructing the original degree distribution is an NP-hard problem. Inspired by CL, we are no longer obsessed with learning the original target distribution, but shift to a proxy task of evaluating node similarity. The final adjacency matrix reconstruction error is a combination of the three loss functions mentioned above:

$$\mathcal{L}_E = \ell_{MSE} + \ell_{BCE} + \ell_{DIST}. \tag{19}$$

C. Discrimination Loss

Graph MAE approaches have achieved good classification performance, however, the presence of difficult-to-distinguish dimensions in the features can lead to deteriorated training. In comparison to the sparsity and discretization characteristics of graph data, pixel and text feature vectors possess a higher information density and more significant discrimination between each other. Low discrimination in graph data is primarily due to the node features being vector representations of text, which are often compressed representations of keywords obtained through feature extractors [22, 23, 46].

Previous research has demonstrated a strong correlation between feature discrimination and Graph MAE performance [23]. When reconstructing node features, indistinguishable features can lead to over-smoothing of the model. In other words, graph MAE attempts to reconstruct the less informative node features, which results in model collapse and drives the learned representations to be similar. This observation further inspires us to design a novel loss function to narrow down the gap between Graph MAE and feature discrimination.

To address the above challenges, we introduce a variance-based discrimination loss, which aims to assist the encoder in learning discriminative embeddings and cushion the impact caused by erroneous information to the network. More importantly, this variance term enforces different node representations within the same embedding matrix to be diverse, compensating for the lack of original feature discrimination. The regularized standard deviation is defined as follows:

$$\mathcal{L}_{Var}(h, \epsilon) = \sqrt{\mathrm{Var}(h) + \epsilon}, \tag{20}$$

where ϵ is a small scalar to prevent numerical instability when reducing to 0 values. $\mathrm{Var}(h)$ is the variance of hidden embeddings (i.e., the output of the shared encoder). This regularization encourages the encoder to map inputs to a different space within a specific range of variances, thereby preventing the model from collapsing to the same vector. Algorithm 1 summarizes the overall training process of GCMAE with all the loss terms.

Complexity. Recall that the framework of GraphMAE [22] includes an encoder and decoder using the GAT [2]. In this way, the total complexity of computing hidden embeddings is $O(LNd^2 + LMd)$, where L is the number of layers, and M is the number of edges. Worse, GAT needs to take the entire adjacency matrix as input when encoding the input graph. However, we adopt SAGE [1] convolution layer to encode node embeddings, it only requires a small number of nodes sampled from neighbors for message passing. Thus, the overall time complexity for GCMAE is $O(r^L nd^2)$, where r is

Algorithm 1 The training procedure of GCMAE

```
Input: Graph $G = (\boldsymbol{A}, \boldsymbol{X})$, shared encoder $f_E$, decoder $f_D$,
    projectors $g_1, g_2$, feature masking $\mathcal{T}_1$, node dropping $\mathcal{T}_2$,
    Epochs $E$
Output: Trained GNN encoder $f_E$
1: Initialize the GNN encoder $f_E$
2: for each $epoch \in E$ do
3:     Generate corrupted views $\mathcal{T}_1(G)$, $\mathcal{T}_2(G)$
4:     $\boldsymbol{H_1} = f_E(\mathcal{T}_1(G)), \quad \boldsymbol{H_2} = f_E(\mathcal{T}_2(G))$
5:     Decoding from hidden embedding $\boldsymbol{Z} = f_D(\boldsymbol{A}, \boldsymbol{H_1})$
6:     $\boldsymbol{U} = g_1(\boldsymbol{H_1}), \quad \boldsymbol{V} = g_2(\boldsymbol{H_2})$
7:     Calculate loss function $\mathcal{J} = \mathcal{L}_{SCE} + \mathcal{L}_C + \mathcal{L}_E + \mathcal{L}_{Var}$
8:     Update $f_E$ with $\mathcal{J}$
9: end for
```

TABLE II
STATISTICS OF THE DATASETS USED FOR NODE CLASSIFICATION, LINK PREDICTION, AND NODE CLUSTERING.

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2,708	10,556	1,433	7
Citeseer	3,327	9,228	3,703	6
PubMed	19,717	88,651	500	3
ogbn-products	2,449,029	61,859,140	100	47

the number of neighbors being sampled for each node. Here, $n << N$ and L is a small value, because deeper layers will lead to over-smoothing. Our efficiency experiment will show that the training time of GCMAE is much shorter than that of GraphMAE.

D. Limitations

Although GCMAE enjoys the advantages of both CL and MAE, and thus provides strong performance as we will show in Section V, it still reveals several limitations. One drawback of GCMAE is that its training time may be relatively long because it uses two branches for CL and MAE and learns to reconstruct the entire adjacency matrix. As the adjacency matrix contains many edges for large graphs, the time consumption could be high. To alleviate this problem, we sample multiple subgraphs from the original graph for reconstruction. As we will report in Section V-D, the training time of GCMAE is comparable to the baseline methods.

V. EXPERIMENTAL EVALUATION

We extensively evaluate our GCMAE along with state-of-the-art baselines on 4 graph tasks, that is, node classification, link prediction, node clustering, and graph classification. We aim to answer the following research questions:

- RQ1: How does GCMAE perform comparing to the baselines in terms of the *accuracy for the graph tasks*?
- RQ2: Does GCMAE *achieve its design goals*, i.e., improving MAE with the contrastive learning paradiagm?
- RQ3: It the training of GCMAE *efficient*?
- RQ4: How do *the design choices and the hyper-parameters* affect the performance of GCMAE?

TABLE III
STATISTICS OF THE DATASETS USED FOR GRAPH CLASSIFICATION.

Dataset	# Graphs	# Classes	Avg. # Nodes
IMDB-B	1,000	2	19.8
IMDB-M	1,500	3	13
COLLAB	5,000	3	74.5
REDDIT-B	2,000	2	429.7
NCI1	4,110	2	29.8
ogbg-molhiv	41,127	2	25.5
ogbg-molbbbp	2.039	2	24.1
Peptides-func	15,535	10	150.1

A. Experiment Settings

Datasets. We conduct experiments on 12 public graph datasets, which are widely used to evaluate graph self-supervised learning methods [22, 35, 46]. In particular, the 4 datasets in Table II are used for node classification, link prediction, and node clustering, and the 8 datasets in Table III are used for graph classification. Note that each dataset in Table II is a single graph while each dataset in Table III contains many small graphs for graph classification. These graphs have a wide range of nodes, edges, and classes. We use OceanBase 4.2.1 [47] to store these graph datasets.

Baselines. We compare GCMAE with 14 state-of-the-art methods on the four graph tasks, which can be classified into the following three categories:

- **Supervised methods** directly train their models with labeled data. We choose two classical GNN models, i.e., GCN [48] and GAT [2], for node classification.
- **Contrastive methods** learn to generate the node embeddings by discriminating positive and negative node pairs. Then, the embeddings serve as input for a separate model (e.g., SVM), which is trained for the downstream task with labeled data. Following GraphMAE [22], we use DGI [46], MVGRL [34], GRACE [35], and CCA-SSG [33] for node classification, link prediction, and node clustering. For graph classification, we choose four graph-level contrastive methods, i.e., Infograph [42], GraphCL [18], JOAO [32], and InfoGCL [49].
- **Masked autoencoder (MAE) methods** adopt a "mask-reconstruct" structure to learn the node embeddings. Like the contrastive methods, a separate model is trained for each downstream task. Following SeeGera [21], we use four graph MAE models, i.e., GraphMAE [22], SeeGera [21], S2GAE [25], and MaskGAE [24] for node classification, link prediction, clustering, and graph classification.
- **Deep clustering methods** are specially designed for node clustering, with design objectives tailored for clustering to guide model training. We choose three such methods, i.e., GC-VGE [6], SCGC [50], and GCC [51].

Note that some baselines may not apply for a task, e.g., the supervised methods only work for node classification. Thus, we apply the baselines for the tasks when appropriate. For contrastive and MAE methods, we use LIBSVM [52] to train SVM classifiers for node classification and graph classification

TABLE IV
NODE CLASSIFICATION ACCURACY SCORES FOR THE EVALUATED METHODS. FOR EACH GRAPH, WE MARK THE MOST ACCURATE METHOD IN **BOLDFACE** AND THE RUNNER-UP WITH UNDERLINE.

	Method	Cora	Citeseer	PubMed	ogbn-products
Supervised	GCN	81.48±0.58	70.34±0.62	79.00±0.50	71.63±0.41
	GAT	82.99±0.65	72.51±0.71	79.02±0.32	73.72±0.32
Contrastive	DGI	82.36±0.62	71.82±0.76	76.82±0.62	74.81±0.30
	MVGRL	83.48±0.53	73.27±0.56	80.11±0.77	OOM
	GRACE	81.86±0.42	71.21±0.53	80.62±0.43	OOM
	CCA-SSG	84.03±0.47	72.99±0.39	81.04±0.48	75.43±0.12
MAE	GraphMAE	85.45±0.40	72.48±0.77	82.53±0.14	78.34±0.11
	SeeGera	85.56±0.25	72.81±0.13	83.01±0.32	78.68±0.38
	S2GAE	86.15±0.25	74.54±0.06	86.79±0.22	79.22±0.36
	MaskGAE	87.31±0.05	75.10±0.07	86.33±0.26	78.48±0.42
ConMAE	GCMAE	**88.82±0.11**	**76.77±0.02**	**88.51±0.18**	**80.19±0.15**

following GraphMAE [22] and SeeGera [21]. We use 5-fold cross-validation to evaluate the performance of the tasks. For link prediction, we fine-tune the final layer of the model using cross-entropy following MaskGAE [24]. For node clustering, we apply *K-means* [53] on the node embeddings. We use the Adam optimizer with a weight decay of 0.0001 for our method, and set the initial learning rate as 0.001.

Performance metrics. We are mainly interested in the accuracy of the evaluated methods and use well-established accuracy measures for each task. Following the evaluation metrics of previous methods [6, 22, 24], we adopt the Accuracy score (ACC) for node classification, the Area Under the Curve (AUC) and Average Precision (AP) for link prediction, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) for node clustering, and Accuracy score (ACC) for graph classification. For all these measures, larger values indicate better performance. For each case (i.e., dataset plus task), we report the average accuracy and standard deviation for each method over 5 runs with different seeds.

B. Model Performance on Graph Tasks (RQ1)

Node classification. Table IV reports the accuracy scores of our GCMAE and the baselines for node classification. We observe that GCMAE is the most accurate method on all datasets. Compared with the best-performing baseline, the accuracy improvements of GCMAE are 1.7% on Cora, 2.2% on Citeseer, 2.0% on PubMed, and 2.1% on ogbn-products, respectively. More detailedly, the supervised methods (i.e., GCN and GAT) perform the worst because they can only utilize label information while the other methods use self-supervised learning to introduce more supervision signals. The graph MAE methods generally perform better than the contrastive methods because node classification relies on the local information of each node (i.e., a local task), and graph MAE is better at capturing local information by learning to reconstruct individual node features and masked edges. Similar patterns are also observed in the results of other tasks, i.e., link prediction and node clustering. The fact that GCMAE outperforms both contrastive and MAE methods

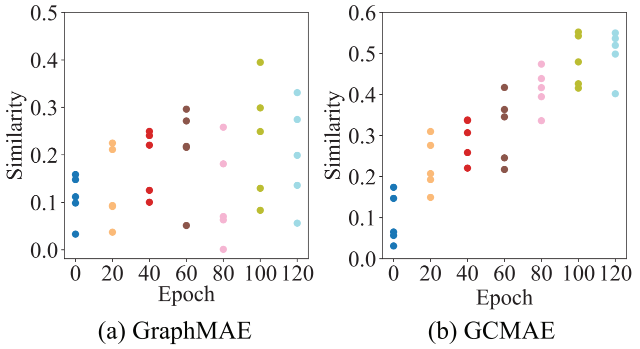

Fig. 4. The embedding similarity between a node and its 5-hop neighbors w.r.t. the number of training epochs.

suggests that our model designs allow us to enjoy the benefits of both paradigms.

Link prediction. Following SeeGera [21] and S2GAE [25], we do not report the results of supervised methods for link prediction. Since DGI [46], MVGRL [34], and GraphMAE [22] do not report relevant experimental results, we train a network for each of them and output the prediction results.

The results of the link prediction task are shown in Table V. GCMAE achieves the best prediction results (except AP on PubMEd), with an average improvement of 1.1% on AUC and 0.8% when compared to the runner-up MaskGAE. The performance of GCMAE exceeds all contrastive methods by a large margin, with an average increase of 6.1% in AUC and 5.8% in AP. Compared to contrastive methods, we use adjacency matrix reconstruction as part of the total objective, forcing the model to pay more attention to graph structures. GraphMAE [22] performs poorly on link prediction, which indicates that only reconstructing node features can lead to performance degradation on link-level tasks. In contrast, MaskMAE [24] takes edges as reconstruction objectives, which is consistent with the downstream tasks, and unsurprisingly becomes the strongest method among all baselines. Based on the observation, the adjacency matrix reconstruction brings more performance improvement to link prediction than edge reconstruction, because the model can capture more meaningful global structures of the graph.

Node clustering. As shown in Table VI, GCMAE achieves the best results among all baselines on the node clustering task. Compared with the MAE method, the improvements range from 0.2% to 7.4% regarding NMI and from 0.5% to 8.1% regarding ARI. Since GCMAE can learn the feature and structure difference between nodes of different clusters from global information, it helps the model clarify the boundaries of clusters. We can observe that the performance gap between the contrastive method and the MAE method on node clustering is not large, unlike node classification and link prediction. This is because the goal of node clustering is to divide the data set into different clusters, in other words, maximizing the similarity of intra-cluster nodes and expanding the difference of inter-cluster nodes. This goal is similar to the intrinsic mechanism of CL. Moreover, we choose 3 deep node clustering methods

TABLE V
LINK PREDICTION ACCURACY FOR THE EVALUATED METHODS. FOR EACH GRAPH, WE MARK THE MOST ACCURATE METHOD IN **BOLDFACE** AND THE RUNNER-UP METHOD WITH UNDERLINE.

	Method	Cora		Citeseer		PubMed		ogbn-products	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP
Contrastive	DGI	93.88±1.00	93.60±1.14	95.98±0.72	96.18±0.68	96.30±0.20	95.65±0.26	70.01±0.32	69.47±0.95
	MVGRL	93.33±0.68	92.95±0.82	88.66±5.27	89.37±4.55	95.89±0.22	95.53±0.30	OOM	OOM
	GRACE	93.46±0.71	92.74±0.48	92.07±0.51	90.32±0.57	96.11±0.13	95.37±0.25	OOM	OOM
	CCA-SSG	93.88±0.95	93.74±1.15	94.69±0.95	95.06±0.91	96.63±0.15	95.97±0.23	80.50±0.68	80.85±0.38
MAE	GraphMAE	90.70±0.01	89.52±0.01	70.55±0.05	74.50±0.04	69.12±0.01	87.92±0.01	84.56±0.61	84.64±0.41
	SeeGera	95.50±0.71	95.92±0.68	97.04±0.47	97.33±0.46	97.87±0.20	97.88±0.21	-	-
	S2GAE	95.05±0.76	95.01±0.62	94.85±0.49	94.84±0.23	98.45±0.03	98.22±0.05	85.83±0.09	85.89±0.14
	MaskGAE	96.66±0.17	96.29±0.23	98.00±0.23	98.25±0.16	99.06±0.05	**98.99±0.06**	84.98±0.33	85.10±0.43
ConMAE	GCMAE	**98.00±0.03**	**97.74±0.37**	**99.48±0.18**	**99.46±0.23**	**99.14±0.27**	98.82±0.13	**87.98±0.25**	**87.66±0.11**

TABLE VI
NODE CLUSTERING ACCURACY FOR THE EVALUATED METHODS. FOR EACH GRAPH, WE MARK THE MOST ACCURATE METHOD IN **BOLDFACE** AND THE RUNNER-UP METHOD WITH UNDERLINE.

	Method	Cora		Citeseer		PubMed		ogbn-products	
		NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
Contrastive	DGI	52.75±0.94	47.78±0.65	40.43±0.81	41.84±0.62	30.03±0.50	29.78±0.28	45.01±0.35	16.14±0.29
	MVGRL	54.21±0.25	49.04±0.67	43.26±0.48	42.73±0.93	30.75±0.54	30.42±0.45	OOM	OOM
	GRACE	54.59±0.32	48.31±0.63	43.02±0.43	42.32±0.81	31.11±0.48	30.37±0.51	OOM	OOM
	CCA-SSG	56.38±0.62	50.62±0.90	43.98±0.94	42.79±0.77	32.06±0.40	31.15±0.85	46.81±0.22	17.21±0.35
MAE	GraphMAE	58.33±0.78	51.64±0.41	45.17±1.12	44.73±0.55	32.52±0.53	31.48±0.39	46.94±0.38	17.91±0.33
	S2GAE	56.25±0.43	50.21±0.44	44.82±0.56	44.51±0.94	31.48±0.35	30.86±0.60	47.26±0.45	18.46±0.30
	MaskGAE	59.09±0.26	52.19±0.51	45.46±0.77	45.68±0.42	33.91±0.35	32.64±0.68	48.17±0.33	18.06±0.22
Clustering	GC-VGE	53.57±0.30	48.15±0.45	40.91±0.56	41.51±0.32	29.71±0.53	29.76±0.66	45.28±0.34	15.82±0.45
	SCGC	56.10±0.72	51.79±1.59	45.25±0.45	46.29±1.13	-	-	-	-
	GCC	59.17±0.28	52.57±0.41	45.13±0.68	45.05±0.93	32.30±0.48	31.23±0.44	47.09±0.36	18.58±0.39
ConMAE	GCMAE	**59.31±0.12**	**52.98±0.21**	**45.84±0.58**	**46.54±0.89**	**34.98±0.85**	**33.76±0.61**	**50.61±0.18**	**20.25±0.19**

as the baseline, and GCMAE can still outperform them both in NMI and ARI. This means that we can obtain high-quality node embeddings for the node clustering task without deliberately tailoring a clustering loss to guide the training process.

Graph classification. SeeGera [21] and MaskGAE [24] are not chosen as baselines due to the source codes are unavailable. Table VII reports all the experimental results for graph classification. We can see that the contrastive methods and the graph MAE methods achieve comparable performance on the graph classification task. In contrast, GCMAE achieves the highest accuracy on all datasets when compared to all the baselines. Our method benefits from both MAE and CL, and therefore can effectively distinguish the differences between graph-level embeddings by comparing multiple corrupted views. Compared to GraphMAE [22], by enhancing feature discrimination through the discrimination loss, our proposed method is able to learn meaningful representations from features with limited information content, such as one-hot vectors based on labels or degrees. Overall, GCMAE does not only perform well on node-level (i.e., node classification and node clustering) and link-level (i.e., link prediction) tasks, but also can generalize well to graph-level downstream tasks. The results clearly demonstrate the effectiveness of the proposed GCMAE framework and validate our claims.

C. Anatomy of the Design Goals (RQ2)

GCMAE learns global information. To verify whether CL can help graph MAE to obtain global information, we visualize the similarity between nodes in GraphMAE and our GCMAE, respectively. Specifically, we randomly select distant nodes that are 5-hops away from the target node and then calculate the similarity between them. We can observe that as the training epoch increases, the similarity of target nodes to distant nodes remains at a low level, which means that MAE is prone to learn node embeddings from local information, according to the result shown in Figure 4 (a). When we combine CL and graph MAE, GCMAE can gradually improve the similarity between target nodes to distant nodes. In other words, CL can make up for the shortcoming of GNN layers that are too shallow, and potentially help graph MAE surpass the constraints of local receptive field and acquire global information, letting graph nodes gain useful knowledge from nodes or edges that are out of GNN's aggregation scope.

Note that as the number of training epochs increases, the similarity between the target node and the distant node does

TABLE VII
GRAPH CLASSIFICATION ACCURACY FOR THE EVALUATED METHODS. FOR EACH DATASET, WE MARK THE MOST ACCURATE METHOD IN **BOLDFACE** AND THE RUNNER-UP METHOD WITH UNDERLINE.

	Method	IMDB-B	IMDB-M	COLLAB	REDDIT-B	NCI1	ogbg-molbbbp	ogbg-molhiv	Peptides-func
		ACC					ROC-AUC		AP
Contrastive	Infograph	73.03±0.87	49.69±0.53	70.65±1.13	82.50±1.42	76.20±1.06	57.90±0.03	73.38±0.07	62.89±0.54
	GraphCL	71.14±0.44	48.58±0.67	71.36±1.15	89.53±0.84	77.87±0.41	62.02±0.01	72.84±0.02	63.71±0.10
	JOAO	70.21±3.08	49.20±0.77	69.50±0.36	85.29±1.35	78.07±0.47	61.41±0.11	73.57±0.11	64.33±0.15
	MVGRL	74.20±0.70	51.20±0.50	OOM	84.50±0.60	OOM	62.11±0.02	72.75±0.56	63.92±0.38
	InfoGCL	75.10±0.90	51.40±0.80	80.00±1.30	OOM	80.20±0.60	62.84±0.02	74.76±0.01	64.62±0.55
MAE	GraphMAE	75.52±0.66	51.63±0.52	80.32±0.46	88.01±0.19	80.40±0.30	62.97±0.12	76.01±0.02	65.17±0.25
	S2GAE	75.76±0.62	51.79±0.36	81.02±0.53	87.83±0.27	80.80±0.24	63.31±0.16	75.92±0.01	65.70±0.21
ConMAE	GCMAE	**75.78±0.23**	**52.49±0.45**	**81.32±0.32**	**91.75±0.22**	**81.42±0.30**	**66.52±0.01**	**78.58±1.17**	**67.24±0.23**

TABLE VIII
NODE CLASSIFICATION ACCURACY USING DIFFERENT DESIGNS FOR THE ENCODER OF GCMAE.

	Cora	Citeseer	PubMed
MAE Encoder	84.14	73.17	81.83
Con. Encoder	68.46	60.46	57.61
Fusion Encoder	85.61	71.71	78.63
Shared Encoder	**88.82**	**76.77**	**88.51**

TABLE IX
END-TO-END TRAINING TIME OF REPRESENTATIVE METHODS. "S" MEANS SECONDS AND "H" MEANS HOURS.

Method	Cora	Citeseer	PubMed	ogbn-products
CCA-SSG	**2.2(s)**	**1.9(s)**	**4.6(s)**	**739.6(s)**
GraphMAE	152.8(s)	93.1(s)	1270.1(s)	1.3(h)
MaskGAE	26.3(s)	40.5(s)	52.7(s)	1037.4(s)
GCMAE	28.6(s)	55.3(s)	508.9(s)	2296.2(s)

not continue to increase. As shown in Figure 4, after a certain number of epochs (i.e., epoch=90), the similarity tends to be stable in (0.4-0.6). Therefore, the model will not encounter the over-smoothing issue.

The shared encoder effectively transfers global information. We study the impact of shared encoder and unshared encoder on our model to evaluate whether the shared encoder can transfer global information. Therefore, we conduct the node classification task on Cora, Citeseer, and PubMed with different types of encoder. Table VIII presents the accuracy results. We can find that "MAE Encoder" outperforms the other two independently parameterized encoders by a significant margin, but does not surpass the shared-parameter encoder. Note that, by only using "MAE Encoder", our method degenerates to GraphMAE [22]. The "Con. Encoder" does not perform as well as expected, which may be due to the excessive corruption of the input graph caused by a high mask ratio, leading to the failure of the contrastive encoder. "Fusion Encoder" represents the average sum of the embeddings generated by the "MAE Encoder" and "Con. Encoder". Naturally, "Fusion Encoder" may suffer from collapsed contrastive encoder, which results in suboptimal results. This is particularly important that the "Shared Encoder" achieves the best classification performance, which means CL can convey global information to the MAE module through the shared-parameter encoder, aiding MAE in perceiving long-range node semantics.

D. The Training Efficiency of GCMAE(RQ3)

In order to study whether unifying CL and graph MAE will increase the training time consumption, we conduct the node classification task on 4 datasets: Cora, Citeseer, PubMed, and ogbn-products, and report the total time consumption. Among the contrastive baselines, we choose CCA [33] since it achieves the best node classification performance. Similarly, GraphMAE [22] and MaskGAE [24] methods are chosen due to their superior performance in the MAE methods. Table IX shows the time consumption of all methods under the parameter setting with the highest node classification accuracy, that is, the sum of pre-training time and the fine-tuning time for downstream tasks.

We can observe that CCA-SSG has the least time consumption, which is due to the use of *canonical correlation analysis* to optimize the calculation between the embeddings of the two views, which greatly reduces the time consumption caused by large matrix operations. Our GCMAE is on average 2× faster than GraphMAE. This is because GraphMAE uses GAT [2] as an encoder, and GAT needs to take the entire adjacency matrix as input when encoding the input graph. Even if we try to reduce the dimension of the hidden embeddings, it still introduces unacceptable time consumption when encountering large-scale graphs. Unlike GraphMAE, the overall time consumption of our method is similar to that of MaskGAE. Since we both use GraphSAGE [1] to encode node embeddings, which can sample multiple subgraphs from a large-scale graph for mini-batch training without inputting the entire graph into the network. However, our method is still slower than MaskGAE, because we reconstruct the entire adjacency matrix instead of only reconstructing partial edges like MaskGAE. Overall, the efficiency performance of GCMAE is comparable to prior works, and there is not a significant increase in time consumption by combining graph MAE and CL.

E. Ablation Study and Parameter Sensitivity (RQ4)

In this part, we conduct an ablation study for the designs of GCMAE and explore the influence of the parameters.

The components of GCMAE. To study the effectiveness

TABLE X
ABLATION STUDY FOR GCMAE'S DESIGNS. NC, LP, CLU, AND GC REFERS TO THE TASK OF NODE CLASSIFICATION, LINK PREDICTION, NODE CLUSTERING, AND GRAPH CLASSIFICATION, RESPECTIVELY. THE PERFORMANCE METRICS (E.G., AUC) ARE IN THE PARENTHESES. WE USE CORA FOR NC, LP, AND CLU, AND REDDIT-B FOR GC.

Task	NC (ACC)	LP (AUC)	CLU (NMI)	GC (ACC)
GraphMAE	85.5	90.6	58.2	88.0
w/o Con.	87.3	93.1	58.9	88.4
w/o Stru. Rec.	86.0	91.1	58.5	89.7
w/o Disc. Loss	87.0	92.9	58.4	89.3
GCMAE	**88.8**	**97.8**	**59.3**	**91.6**

of each component, we compare our complete framework GCMAE with GraphMAE [22] and three variants: "w/o Con.", "w/o Stru. Rec." and "w/o Disc." on all four graph tasks, where node classification, link prediction and node clustering use the Cora dataset, and graph classification uses the REDDIT-B dataset. The results are presented in Table X, where "w/o Con." means the removal of the CL loss, "w/o Stru. Rec." refers to our method without adjacency matrix reconstruction, and "w/o Disc." is our method without feature discrimination loss. We can find that the removal of any of these components leads to a decrease in performance for all tasks. Interestingly, even if we totally remove the contrastive module, our method still outperforms the GraphMAE [22]. This is because the adjacency matrix reconstruction provides rich graph structure information and the discrimination loss improves the feature discrimination among nodes.

Effect of the hyper-parameters. To study the influence of different hyper-parameter settings on the model, we conduct sensitivity experiments for the mask rate and drop rate on node classification. We report the performance in Figure 5, where the x-axis, y-axis, and z-axis represent the feature mask rate p_{mask}, the drop node rate p_{drop}, and the F1-Score value, respectively. When p_{mask} is large (0.5-0.8), the model performance remains within a satisfactory range. A higher mask rate means lower redundancy, which can help the encoder recover missing node features from a few neighboring nodes. Moreover, changing p_{mask} has a significant effect on the accuracy of GCMAE while the influence of p_{drop} is smaller. This is because MAE is more effective than contrastive learning for the node classification task, which is evidenced by Table IV. p_{mask} and p_{drop} serve to control the influences of the MAE module and the contrastive module within the overall framework, respectively. As such, adjusting p_{mask} to control MAE has a more significant effect on the accuracy of GCMAE for node classification.

The impact of network width and depth on performance has attracted significant attention in the CL methods [35, 54]. As shown in Figure 6, the accuracy of GCMAE first improves but then deteriorates when increasing the hidden dimension of GNN. This means a moderate hidden dimension is crucial for the model to learn informative and compact node embeddings for downstream tasks. Meanwhile, we can observe that network depth also has a remarkable impact on model performance. Figure 6 shows that the accuracy of GCMAE

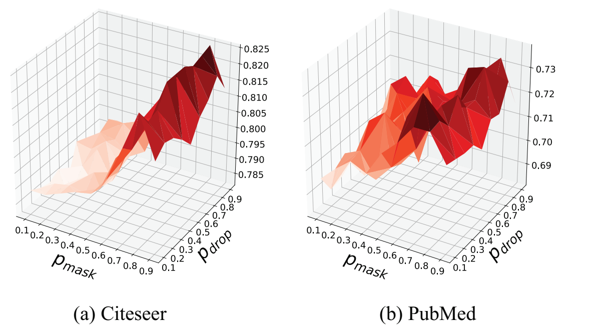

Fig. 5. The influence of edge mask rate p_{mask} and node drop node rate p_{drop} on the performance of node classification.

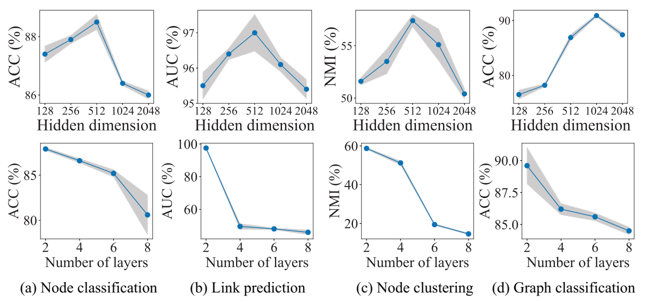

Fig. 6. The influence of the hidden dimension and the number of layers in the GNN models. We report the results of our GCMAE for all four tasks. We use the Cora dataset for node classification, link prediction and node clustering, and the Peptides-func dataset for graph classification.

is the highest with 2 GNN layers and gradually decreases as the number of layers increases. This is because GNN usually achieves a good balance between *receptive filed* and *over-smoothing* with a small number of layers (e.g., 2-3) [55, 56]. In particular, using more layers allows larger receptive filed and aggregating neighbors that are more hops away from the target node to collect information; in the meantime, using more layers tends to produce similar embeddings for the nodes (i.e., over-smoothing), which makes it difficult to distinguish different nodes and may hinder downstream tasks.

VI. RELATED WORK

In this part, we review representative works on contrastive and generative methods for graph self-supervised learning.

A. Graph Learning for Graph Processing

Graph processing tasks (e.g., node classification, link prediction, node clustering and graph classification) have been long studied in both database and data engineering [57, 58, 59, 60], and many systems are designed to train and serve graph learning models efficiently [61, 62, 63, 64, 65, 66]. Recently, graph learning is widely used to enhance graph processing tasks due to its good performance [13, 15, 67]. For instance, CLDG [11] uses self-supervised learning to contrast positive and negative samples from graphs to improve the accuracy of graph classification. GraphNAS [12] allows GNNs to focus on the most relevant and influential neighbors rather than aggregating information from the entire neighborhood. CG-KGR [14] enhances recommendation (i.e., link prediction) by propagating information over the user-item interaction graph using graph neural networks (GNNs). ANC [16] targets graph clustering and learns to encompass local structural information and edge weights into a consistent distance metric. In this paper, we combine the MAE and CL paradigms to improve the graph learning and thus may benefit various graph processing tasks.

B. Contrastive Methods for Graph SSL

Inspired by the success of CL in computer vision and natural language processing [68, 69, 70], many works develop contrastive methods for graph learning [34, 35, 46, 71, 72]. These methods generally produce multiple corrupted views of the graph via data augmentation and maximize the similarity between these views. For instance, GraphCL [18] adopts four types of graph augmentations (i.e., node dropping, edge perturbation, attribute masking, and subgraph sampling) to incorporate different priors and learns to predict whether two graphs are generated from the same graph. DGI [46] conducts CL using patches of a graph and uses a read-out function to compute graph-level embedding from the node embeddings. GRACE [35] corrupts both the graph topology and the node features such that contrast learning can capture more information. MVGRL [34] observes that simply increasing the number of views does not improve performance and proposes to maximize the mutual information among the node and graph representations in different views. CCA-SSG [33] leverages canonical correlation analysis to speedup the computation of contrastive loss among multiple augmented views and reduce model training time. We observe that the contrastive methods are good at capturing global information of the graph but poor in learning the local information for particular edges and nodes. Thus, by augmenting MAE with CL, our GCMAE outperforms the existing GSSL methods.

C. Generative SSL Method

Graph Autoencoder (GAE) is a classical self-supervised learning method, which encodes the graph structure information into a low-dimensional latent space and then reconstructs the adjacency matrix from hidden embeddings [73, 74, 75]. For example, DNGR [75] uses a stacked denoising autoencoder [76] to encode and decode the PPMI matrix via multilayer perceptrons. However, DNGR ignores the feature information when encoding node embeddings. Therefore, GAE [77] utilizes GCN [48] to encode node structural information and node feature information at the same time and then uses a dot-product operation for reconstructing the input graph. Variational GAE (VGAE) [77] learns the distribution of data, where KL divergence is used to measure the distance between the empirical distribution and the prior distribution. In order to further narrow the gap between the above two distributions, ARVGA [78] employs the training scheme of a generative adversarial network [79] to address the approximate problem. Later studies focused on leveraging feature reconstruction or additional auxiliary information [74, 78]. Unfortunately, most of them mainly perform well on a single task such as node classification or link prediction, since they are limited by a sufficient reconstruction objective. However, our GCMAE surpasses these GAE methods on various downstream tasks due to both reconstructing node features and edge.

Masked Autoencoders learn graph representations by masking certain nodes or edges and then reconstructing the masked tokens [23, 44, 80]. This strategy allows the graph to use its own structure and feature information in a self-supervised manner without expensive label annotations. Recently, GraphMAE [22] enforces the model to reconstruct the original graph from redundant node features by masking node features and applying a re-masking strategy before a GNN decoder. Instead of masking node features, MaskGAE [24] selects edges as the masked token and then reconstructs the graph edges or random masked path accordingly. S2GAE [25] proposes a cross-correlation decoder to explicitly capture the similarity of the relationship between two connected nodes at different granularities. SeeGera [21] is a hierarchical variational framework that jointly embeds nodes and features in the encoder and reconstructs links and features in the decoder, where an additional structure/feature masking layer is added to improve the generalization ability of the model. Based on the above observations, these graph MAE methods all suffer from inaccessible global information, resulting in sub-optimal performance. However, GCMAE surpasses these graph MAE methods, as unifying CL and graph MAE enjoys the benefits of both paradigms and yields more high-quality node embeddings.

VII. Conclusion

In this paper, we study two main paradigms for graph self-supervised learning, i.e., masked autoencoder and contrastive learning. We find that they have their own limitations but complement each other. Thus, we propose the GCMAE framework that jointly utilizes MAE and contrastive learning for learning both local and global information. GCMAE comes with tailored model designs including a shared encoder for information exchange, a discrimination loss to tackle feature smoothing, and an adjacency matrix reconstruction method to learn global information of the graph. We conduct extensive experiments to evaluate GCMAE on various graph tasks. The results show that GCMAE significantly outperforms state-of-the-art GSSL methods and is general across different graph tasks.

Acknowledgment

This work was sponsored by National Key R&D Program of China (No. 2023YFB2703600), Key R&D Program of Hubei Province (No. 2023BAB077), the Fundamental Research Funds for the Central Universities (No. 2042023kf0219), and CCF- AFSG Research Fund. This work was supported by Ant Group through CCF-Ant Research Fund.

References

[1] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

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

[3] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, "Pick and choose: a gnn-based imbalanced learning approach for fraud detection," in *Proceedings of the Web Conference 2021*, 2021, pp. 3168–3177.

[4] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *International Conference on Machine Learning*, 2018, pp. 1106–1114.

[5] S. Liu, T. Li, H. Ding, B. Tang, X. Wang, Q. Chen, J. Yan, and Y. Zhou, "A hybrid method of recurrent neural network and graph neural network for next-period prescription prediction," *International Journal of Machine Learning and Cybernetics*, vol. 11, pp. 2849–2856, 2020.

[6] L. Guo and Q. Dai, "Graph clustering via variational graph embedding," *Pattern Recognition*, vol. 122, p. 108334, 2022.

[7] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080.

[8] W. Zhang, X. Miao, Y. Shao, J. Jiang, L. Chen, O. Ruas, and B. Cui, "Reliable data distillation on graph convolutional network," in *Proceedings of the ACM on Management of Data (SIGMOD)*, 2020, pp. 1399–1414.

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

[10] M. Lu, Z. Han, S. X. Rao, Z. Zhang, Y. Zhao, Y. Shan, R. Raghunathan, C. Zhang, and J. Jiang, "Bright-graph neural networks in real-time fraud detection," in *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022, pp. 3342–3351.

[11] Y. Xu, B. Shi, T. Ma, B. Dong, H. Zhou, and Q. Zheng, "Cldg: Contrastive learning on dynamic graphs," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 696–707.

[12] H. Zhao, Q. Yao, and W. Tu, "Search to aggregate neighborhood for graph neural network," in *International Conference on Data Engineering (ICDE)*, 2021.

[13] S. Gurukar, N. Pancha, A. Zhai, E. Kim, S. Hu, S. Parthasarathy, C. Rosenberg, and J. Leskovec, "Multibisage: A web-scale recommendation system using multiple bipartite graphs at pinterest," *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 781–789, 2022.

[14] Y. Chen, Y. Yang, Y. Wang, J. Bai, X. Song, and I. King, "Attentive knowledge-aware graph convolutional networks with collaborative guidance for personalized recommendation," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 299–311.

[15] K. Liu, S. Wang, Y. Zhang, and C. Xing, "An efficient algorithm for distance-based structural graph clustering," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 1, no. 1, pp. 1–25, 2023.

[16] Z. Feng, M. Qiao, and H. Cheng, "Clustering activation networks," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 780–792.

[17] X. Miao, W. Zhang, Y. Shao, B. Cui, L. Chen, C. Zhang, and J. Jiang, "Lasagne: A multi-layer graph convolutional network framework via node-aware deep architecture," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1721–1733, 2021.

[18] 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.

[19] Y. Yin, Q. Wang, S. Huang, H. Xiong, and X. Zhang, "Autogcl: Automated graph contrastive learning via learnable view generators," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8892–8900.

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

[21] X. Li, T. Ye, C. Shan, D. Li, and M. Gao, "Seegera: Self-supervised semi-implicit graph variational auto-encoders with masking," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 143–153.

[22] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 594–604.

[23] Z. Hou, Y. He, Y. Cen, X. Liu, Y. Dong, E. Kharlamov, and J. Tang, "Graphmae2: A decoding-enhanced masked self-supervised graph learner," *arXiv preprint arXiv:2304.04779*, 2023.

[24] J. Li, R. Wu, W. Sun, L. Chen, S. Tian, L. Zhu, C. Meng, Z. Zheng, and W. Wang, "Maskgae: masked graph modeling meets graph autoencoders," *arXiv preprint arXiv:2205.10053*, 2022.

[25] Q. Tan, N. Liu, X. Huang, S.-H. Choi, L. Li, R. Chen, and X. Hu, "S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 787–795.

[26] E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, "Node feature extraction by self-supervised multi-scale neighborhood predic-

tion," *arXiv preprint arXiv:2111.00064*, 2021.

[27] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, "Masked feature prediction for self-supervised visual pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 668–14 678.

[28] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. 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, no. 04, 2020, pp. 3438–3445.

[29] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," *arXiv preprint arXiv:1909.12223*, 2019.

[30] S. Li, X. Wang, A. Zhang, Y. Wu, X. He, and T.-S. Chua, "Let invariant rationale discovery inspire graph contrastive learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 13 052–13 065.

[31] X. Cai, C. Huang, L. Xia, and X. Ren, "Lightgcl: Simple yet effective graph contrastive learning for recommendation," *arXiv preprint arXiv:2302.08191*, 2023.

[32] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 121–12 132.

[33] H. Zhang, Q. Wu, J. Yan, D. Wipf, and P. S. Yu, "From canonical correlation analysis to self-supervised graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 76–89, 2021.

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

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

[36] Y. Ma, Y. Chen, Y. Tian, C. Gu, and T. Jiang, "Contrastive study of in situ sensing and swabbing detection based on sers-active gold nanobush–pdms hybrid film," *Journal of Agricultural and Food Chemistry*, vol. 69, no. 6, pp. 1975–1983, 2021.

[37] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu, "The ubiquity of large graphs and surprising challenges of graph processing: extended survey," *The VLDB journal*, vol. 29, pp. 595–618, 2020.

[38] X. Kong and X. Zhang, "Understanding masked image modeling via learning occlusion invariant feature," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6241–6251.

[39] 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," *arXiv preprint arXiv:2302.02318*, 2023.

[40] Z. Huang, X. Jin, C. Lu, Q. Hou, M.-M. Cheng, D. Fu, X. Shen, and J. Feng, "Contrastive masked autoencoders are stronger vision learners," *arXiv preprint arXiv:2207.13532*, 2022.

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

[42] 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.

[43] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[44] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1857–1867.

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

[46] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *The International Conference on Learning Representations*, vol. 2, no. 3, p. 4, 2019.

[47] Z. Yang, C. Yang, F. Han, M. Zhuang, B. Yang, Z. Yang, X. Cheng, Y. Zhao, W. Shi, H. Xi *et al.*, "Oceanbase: a 707 million tpmc distributed relational database system," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, pp. 3385–3397, 2022.

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

[49] 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.

[50] Y. Liu, X. Yang, S. Zhou, X. Liu, S. Wang, K. Liang, W. Tu, and L. Li, "Simple contrastive graph clustering," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[51] C. Fettal, L. Labiod, and M. Nadif, "Efficient graph convolution for joint node representation learning and clustering," in *Proceedings of the Fifteenth ACM International conference on Web Search and Data Mining*, 2022, pp. 289–297.

[52] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[53] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, 2006, pp. 144–153.

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

[55] X. Wan, K. Xu, X. Liao, Y. Jin, K. Chen, and X. Jin, "Scalable and efficient full-graph gnn training for large graphs," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 1, no. 2, pp. 1–23, 2023.

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

[57] Q. Zhang, R.-H. Li, M. Pan, Y. Dai, G. Wang, and Y. Yuan, "Efficient top-k ego-betweenness search," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 380–392.

[58] N. Yudong, Y. Li, J. Fan, and Z. Bao, "Local clustering over labeled graphs: An index-free approach," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 2805–2817.

[59] W. Fan, L. Geng, R. Jin, P. Lu, R. Tugay, and W. Yu, "Linking entities across relations and graphs," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 634–647.

[60] Q. Zhang, J. Li, H. Zhao, Q. Xu, W. Lu, J. Xiao, F. Han, C. Yang, and X. Du, "Efficient distributed transaction processing in heterogeneous networks," *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1372–1385, 2023.

[61] J. Jiang, P. Xiao, L. Yu, X. Li, J. Cheng, X. Miao, Z. Zhang, and B. Cui, "Psgraph: How tencent trains extremely large-scale graphs with spark?" in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1549–1557.

[62] J. Jiang, Y. Yang, J. Wang, and J. Wu, "Jointly learning representations for map entities via heterogeneous graph contrastive learning," *arXiv preprint arXiv:2402.06135*, 2024.

[63] P. Gong, R. Liu, Z. Mao, Z. Cai, X. Yan, C. Li, M. Wang, and Z. Li, "gsampler: General and efficient gpu-based graph sampling for graph learning," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 562–578.

[64] P. Yin, X. Yan, J. Zhou, Q. Fu, Z. Cai, J. Cheng, B. Tang, and M. Wang, "Dgi: An easy and efficient framework for gnn model evaluation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 5439–5450.

[65] Z. Cai, X. Yan, Y. Wu, K. Ma, J. Cheng, and F. Yu, "Dgcl: An efficient communication library for distributed gnn training," in *Proceedings of the Sixteenth European Conference on Computer Systems*, 2021, pp. 130–144.

[66] Z. Cai, Q. Zhou, X. Yan, D. Zheng, X. Song, C. Zheng, J. Cheng, and G. Karypis, "Dsp: Efficient gnn training with multiple gpus," in *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, 2023, pp. 392–404.

[67] Y. Li, R. Yang, and J. Shi, "Efficient and effective attributed hypergraph clustering via k-nearest neighbor augmentation," *Proceedings of the ACM on Management of Data (SIGMOD)*, vol. 1, no. 2, pp. 1–23, 2023.

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

[69] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[70] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[71] J. Xia, L. Wu, G. Wang, J. Chen, and S. Z. Li, "Progcl: Rethinking hard negative mining in graph contrastive learning," *arXiv preprint arXiv:2110.02027*, 2021.

[72] W.-Z. Li, C.-D. Wang, H. Xiong, and J.-H. Lai, "Homogcl: Rethinking homophily in graph contrastive learning," *arXiv preprint arXiv:2306.09614*, 2023.

[73] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 889–898.

[74] A. Salehi and H. Davulcu, "Graph attention autoencoders," *arXiv preprint arXiv:1905.10715*, 2019.

[75] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[76] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1096–1103.

[77] T. N. Kipf and M. Welling, "Variational graph autoencoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[78] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018.

[79] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[80] S. Zhang, Y. Liu, Y. Sun, and N. Shah, "Graph-less neural networks: Teaching old mlps new tricks via distillation," *arXiv preprint arXiv:2110.08727*, 2021.