GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner

Zhenyu Hou
Tsinghua University, China
houzy21@mails.tsinghua.edu.cn

Yufei He*
Beijing Institute of Technology, China
yufei.he@bit.edu.cn

Yukuo Cen
Tsinghua University, China
cyk20@mails.tsinghua.edu.cn

Xiao Liu
Tsinghua University, China
liuxiao21@mails.tsinghua.edu.cn

Yuxiao Dong†
Tsinghua University, China
yuxiaod@tsinghua.edu.cn

Evgeny Kharlamov
Bosch Center for Artificial Intelligence
evgeny.kharlamov@de.bosch.com

Jie Tang†
Tsinghua University, China
jietang@tsinghua.edu.cn

ABSTRACT

Graph self-supervised learning (SSL), including contrastive and generative approaches, offers great potential to address the fundamental challenge of label scarcity in real-world graph data. Among both sets of graph SSL techniques, the masked graph autoencoders (e.g., GraphMAE)—one type of generative methods—have recently produced promising results. The idea behind this is to reconstruct the node features (or structures)—that are randomly masked from the input—with the autoencoder architecture. However, the performance of masked feature reconstruction naturally relies on the discriminability of the input features and is usually vulnerable to disturbance in the features. In this paper, we present a masked self-supervised learning framework[1] GraphMAE2 with the goal of overcoming this issue. The idea is to impose regularization on feature reconstruction for graph SSL. Specifically, we design the strategies of multi-view random re-mask decoding and latent representation prediction to regularize the feature reconstruction. The multi-view random re-mask decoding is to introduce randomness into reconstruction in the feature space, while the latent representation prediction is to enforce the reconstruction in the embedding space. Extensive experiments show that GraphMAE2 can consistently generate top results on various public datasets, including at least 2.45% improvements over state-of-the-art baselines on ogbn-Papers100M with 111M nodes and 1.6B edges.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; • **Information systems** → **Data mining**.

KEYWORDS

Graph Neural Networks; Self-Supervised Learning; Graph Representation Learning; Pre-Training

ACM Reference Format:
Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. 2023. GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583379

1 INTRODUCTION

Graph neural networks (GNNs) have found widespread adoption in learning representations for graph-structured data. The success of GNNs has thus far mostly occurred in (semi-) supervised settings, in which task-specific labels are used as the supervision information, such as GCN [25], GAT [41], and GraphSAGE [13]. However, it is often arduously difficult to obtain sufficient labels in real-world scenarios, especially for billion-scale graphs [21, 22].

One natural solution to this challenge is to perform self-supervised learning (SSL) on graphs [30], where graph models (e.g., GNNs) are supervised by labels that are automatically constructed from the input graph data. Along this line, generative SSL models that aim to generate one part of the input graph from another part have received extensive exploration [9, 22, 24, 33, 43]. Among all alternatives, the masked graph modeling technique has been demonstrated as a powerful strategy of generative SSL on graphs [18, 21, 43]. Straightforwardly, it first corrupts the input graph by masking node features or edges and then learns to recover the original input.

Under the masked prediction framework, a very recent work introduces a masked graph autoencoder GraphMAE [18] for generative SSL on graphs, which yields outperformance over various baselines on 21 datasets for different tasks. Generally, an autoencoder is made up of an encoder, code/embeddings, and a decoder. The encoder maps the input to embeddings, and the decoder aims to

*This work was done when the author was visiting Tsinghua University.
†Yuxiao Dong and Jie Tang are the corresponding authors.
[1]The code is available at: https://github.com/THUDM/GraphMAE2.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583379

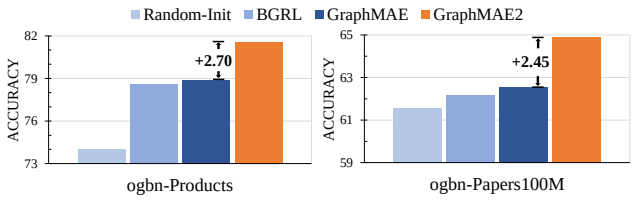

Figure 1: Linear probing results on ogbn-Products and ogbn-Papers100M. GraphMAE2 achieves a significant advantage over previous graph SSL methods on benchmarks with millions of nodes.

reconstruct the input based on the embeddings under a reconstruction criterion. The main idea of GraphMAE is to reconstruct the input node features that are randomly masked before encoding by using an autoencoding architecture. Its technical contribution lies in the design of 1) masked feature reconstruction and 2) fixed re-mask decoding, wherein the encoded embeddings of previously-masked nodes are masked again before feeding into the decoder.

Despite GraphMAE's promising performance, the reconstruction of masked features fundamentally relies on the discriminability [8, 45] of the input node features, i.e., the extent to which the node features are distinguishable. In practice, the features of nodes in a graph are usually generated from data that is associated with each node, such as the embeddings of content posted by users in a social network, making them an approximate description of nodes and thus less discriminative. Note that in vision or language studies, the reconstruction targets are usually a natural description of the data, i.e., pixels of an image and words of a document. Table 1 further shows that the performance of GraphMAE drops more significantly than the supervised counterpart when using less discriminative node features (w/ PCA). In other words, GraphMAE, as a generative SSL framework with feature reconstruction, is relatively more vulnerable to the disturbance of features.

In this work, we present GraphMAE2 with the goal of improving feature reconstruction for graph SSL. The idea is to impose regularization on target reconstruction. To achieve this, we introduce two decoding strategies: *multi-view random re-mask decoding* for reducing the overfitting to the input features, and *latent representation prediction* for having more informative targets.

First, instead of fixed re-mask decoding used in GraphMAE—re-masking the encoded embeddings of masked nodes, we propose to introduce randomness into input feature reconstruction with *multi-view random re-mask decoding*. That is, the encoded embeddings are randomly re-masked multiple times, and their decoding results are all enforced to recover input features. Second, we propose *latent representation prediction*, which attempts to reconstruct masked features in the embedding space rather than the reconstruction in the input feature space. The predicted embeddings of masked nodes are constrained to match their representations that are directly generated from the input graph. Both designs naturally work as the regularization on target construction in generative graph SSL.

Inherited from GraphMAE, GraphMAE2 is a simple yet more effective generative self-supervised framework for graphs that can be directly coupled with existing GNN architectures. We perform extensive experiments on public graph datasets representative of different scales and types, including three open graph benchmark datasets. The results demonstrate that GraphMAE2 can consistently offer significant outperformance over state-of-the-art graph SSL baselines under different settings. Furthermore, we show that both decoding strategies contribute to the performance improvements compared to GraphMAE. Excitingly, GraphMAE2 as an SSL method offers performance advantages over classic supervised GNNs across all datasets, giving rise to the premise of self-supervised graph representation learning and pre-training.

Table 1: Results with the original node features (raw) or PCA-processed node features (w/ PCA). *w/ PCA* represents that the input features are reduced to 50-dimensional continuous vectors using PCA, relatively less discriminative. GraphMAE can be more sensitive to the discriminability of input features than the supervised one. GAT is used as the backbone for all cases.

	Cora raw → *w/ PCA*	PubMed raw → *w/ PCA*
Supervised	83.0 → 82.3 (↓ 0.7)	78.0 → 77.0 (↓ 1.0)
GraphMAE	84.2 → 82.6 (↓ 1.6)	81.1 → 78.9 (↓ 2.2)
GraphMAE2	84.5 → 83.5 (↓ 1.0)	81.4 → 80.1 (↓ 1.3)

In addition, we extend GraphMAE2 to large-scale graphs with hundreds of millions of nodes, which have been previously less explored for graph SSL. We leverage local clustering strategies that can produce local and dense subgraphs to benefit GraphMAE2 (and GraphMAE) with masked feature prediction. Experiments on ogbn-Papers100M of 111M nodes and 1.6B edges suggest the simple GraphMAE2 framework can generate significant performance improvements over existing methods (Cf. Figure 1).

2 METHOD

In this section, we first revisit masked autoencoding for graph SSL and identify its deficiency in which the effectiveness of masked feature reconstruction can be vulnerable to the distinguishability of input node features. Then we present our GraphMAE2 to overcome the problem by imposing regularization on the feature decoding.

2.1 Masked Autoencoding on Graphs

Notations. Let $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$, where $\mathcal{V}$ is the node set, $N = |\mathcal{V}|$ represents the number of nodes, $\boldsymbol{A} \in \{0,1\}^{N\times N}$ is the adjacency matrix with each element $\boldsymbol{A}(i,j) = 1$ indicating that there exists an edge between v_i and v_j. $\boldsymbol{X} \in \mathbb{R}^{N\times d_{in}}$ is the input node feature matrix. In graph autoencoders, we use f_E to represent the GNN encoder such as GAT [41] and GCN [25]. And f_D represents the decoder which can be a multi-layer perceptron (MLP) or GNN. Denoting the hidden embedding $\boldsymbol{H} \in \mathbb{R}^{N\times d}$, the general goal of graph autoencoders is to learn representation $\boldsymbol{H}$ or a well-initialized f_E through reconstructing input node features or structure:

$$\boldsymbol{H} = f_E(\boldsymbol{A}, \boldsymbol{X}),\ \widetilde{\mathcal{G}} = f_D(\boldsymbol{A}, \boldsymbol{H}) \tag{1}$$

where $\widetilde{\mathcal{G}}$ denotes the reconstructed graph characteristics, which can be structure, node features or both.

Overview of masked feature reconstruction. The idea of masked autoencoder has seen successful practice in graph SSL [18]. As a

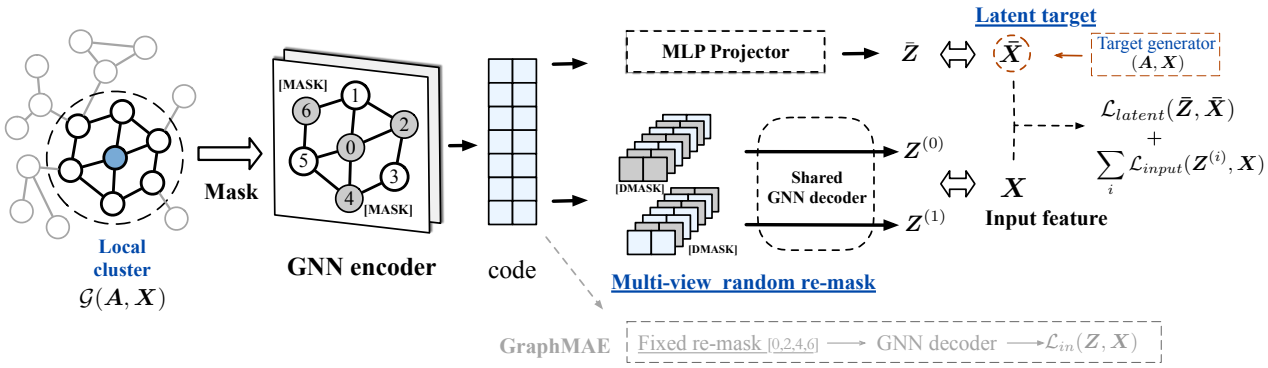

Figure 2: Overview of GraphMAE2 framework. For large-scale graphs, we first run local clustering to produce local clusters for each node as the preprocessing step. During the pre-training, GraphMAE2 corrupts the graph by masking input node features with a mask token [MASK] and then feeds the result to a GNN encoder to generate the code. The decoding involves two objectives: 1) we generate multiple corrupted codes by randomly re-masking the code several times, and they are all forced to reconstruct input features after GNN decoding. 2) we use an MLP as the decoder to predict latent target representations, which are produced by a target generator with the unmasked graph. As a comparison, GraphMAE is trained through input feature reconstruction only with a fixed re-mask decoding strategy.

form of more general denoising autoencoders, it removes a portion of data in the graph, e.g., node features or links, with the masking operation and learns to predict the masked content. And it has been demonstrated that reconstructing masked node features as the only pretext task could generate promising performance. In this work, we follow the paradigm of masked feature reconstruction and aim to further boost the performance by resolving the potential concerns in existing works.

Formally, we uniformly sample a subset of nodes $\widetilde{\mathcal{V}} \subset \mathcal{V}$ without replacement and replace their feature with a mask token [MASK], i.e. a learnable vector $\boldsymbol{x}_{[M]} \in \mathbb{R}^{d_{in}}$. And sampling with a relatively large mask ratio (e.g., 50%) helps eliminate redundancy in graphs and benefit performance. The features $\widetilde{\boldsymbol{x}}_i$ for node $v_i \in \mathcal{V}$ in the corrupted feature matrix $\widetilde{\boldsymbol{X}}$ can be represented as:

$$\widetilde{\boldsymbol{x}}_i = \begin{cases} \boldsymbol{x}_{[M]} & v_i \in \widetilde{\mathcal{V}} \\ \boldsymbol{x}_i & v_i \notin \widetilde{\mathcal{V}} \end{cases}$$

Then the corrupted graph $(\boldsymbol{A}, \widetilde{\boldsymbol{X}})$ is fed into the encoder f_E to generate representations $\boldsymbol{H}$. And the decoder f_D decodes the predicted masked features $\boldsymbol{Z}$ from $\boldsymbol{H}$. The training objective is to match the predicted $\boldsymbol{Z}$ with the original features $\boldsymbol{X}$ with a designated criterion, such as (scaled) cosine error.

Problems in masked feature reconstruction. Despite the excellent performance, there exists potential concern for masked node feature reconstruction due to the inaccurate semantics of node features. A recent study [8] shows that the performance of GNNs on downstream tasks can be significantly affected by the distinguishability of node features. In masked feature reconstruction, less discriminative reconstruction targets might cause misleading and harm the learning. To verify this assumption, we conduct pilot experiments by comparing the results using original features with less discriminative features. To induce information loss on features, we compress the features by mapping the original features to low dimensional space, i.e., 50 dimensions, using PCA. Table 1 shows the results. We observe that the performance of GraphMAE degrades more significantly than the supervised counterpart when using the compressed features. The results indicate that the performance of learning through input feature reconstruction tends to be more vulnerable to the discriminability of the features.

In CV and NLP, where the philosophy of masked prediction has groundbreaking practices, their inputs are exact descriptions of data without loss of semantic information, e.g., pixels for images and words for texts. However, the input $\boldsymbol{X}$ of graphs could inevitably and intrinsically contain unexpected noises since they processed products from various raw data, e.g., texts or hand-crafted features. The input $\boldsymbol{X}$ is and generated by various feature extractors. For example, the node features of Cora [49] are bag-of-words vectors, ogbn-Arxiv [20] averages word embeddings of word2vec, and MAG240M [19] are from pretrained language model. Their discriminability is constrained to the expressiveness of the feature generator and could inherit the substantial noise in the generator. In masked feature reconstruction, the objective of recovering less discriminative node features can guide the model to fit inaccurate targets and unexpected noises, bringing potential negative effects.

2.2 The GraphMAE2 Framework

We present GraphMAE2 to overcome the aforementioned issue. It follows the masked prediction paradigm and further incorporates regularization to the decoding stage to improve effectiveness.

To improve feature reconstruction, we propose to randomly re-mask the encoded representations multiple times and force the decoder to reconstruct input features from the corrupted representations. Then to minimize the direct effects of input features, we also enforce the model to predict representations of masked nodes in the embedding space beyond the input feature space. Both strategies serve as regularization to avoid the model over-fitting to the input features. Moreover, we extend GraphMAE2 to large graphs and propose to sample densely-connected subgraphs to accommodate with GraphMAE2's training, The overall framework of GraphMAE2 is illustrated in Figure 2.

Multi-view random re-mask decoding. From the perspective of input feature reconstruction, we introduce randomness in the decoding and require the decoder to restore the input $\boldsymbol{X}$ from different and partially observed embeddings.

The decoder maps the latent code $\boldsymbol{H}$ to the input feature space to reconstruct $\boldsymbol{X}$ for optimization. GraphMAE [18] shows that using a GNN as the decoder achieves better performance than using MLP, and the GNN decoder helps the encoder learn high-level latent code when recovering the high-dimension and low-semantic features. The main difference is that GNN involves propagation and recovers the input relying on neighborhood information. Based on this characteristic of the GNN decoder, instead of the fixed re-mask decoding used in GraphMAE, we propose a *multi-view random re-mask decoding* strategy. It randomly re-masks the encoded representation before they are fed into the decoder, which resembles the random propagation in semi-supervised learning [10]. Formally, we resample a subset of nodes $\widetilde{\mathcal{V}} \subset \mathcal{V}$ following a uniform distribution. $\widetilde{\mathcal{V}}$ is different from the input masked nodes $\overline{\mathcal{V}}$ and nodes are equally selected for re-masking regardless of whether they are masked before. Then corrupted representation matrix $\widetilde{\boldsymbol{H}}$ is built from $\boldsymbol{H}$ by replacing the $\boldsymbol{h}_i$ of node $v_i \in \widetilde{\mathcal{V}}$ with another shared mask token [DMASK], i.e., a learnable vector $\boldsymbol{h}_{[M]} \in \mathbb{R}^d$:

$$\widetilde{\boldsymbol{h}}_i = \begin{cases} \boldsymbol{h}_{[M]} & v_i \in \widetilde{\mathcal{V}} \\ \boldsymbol{h}_i & v_i \notin \widetilde{\mathcal{V}} \end{cases}$$

Then the decoder would reconstruct the input $\boldsymbol{X}$ from the corrupted $\widetilde{\boldsymbol{H}}$. The procedure is repeated several times to generate K different re-masked nodes sets $\{\widetilde{\mathcal{V}}^{(j)}\}_{1,\dots,K}$ and corresponding corrupted representations $\{\widetilde{\boldsymbol{H}}^{(j)}\}_{1,\dots,K}$. Each view contains different information after re-masking, and they are all enforced to reconstruct input node features. The randomness of decoding serves as regularization preventing the network from memorizing unexpected patterns in the input $\boldsymbol{X}$, and thus the training would be less sensitive to the disturbance in the input feature. Finally, we employ the scaled cosine error [18] to measure the reconstruction error and sum over the errors of the K views for training:

$$\mathcal{L}_{input} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{j=1}^{K} \sum_{v_i \in \widetilde{\mathcal{V}}} \left(1 - \frac{\boldsymbol{x}_i^\top \boldsymbol{z}_i^{(j)}}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{z}_i^{(j)}\|}\right)^\gamma \tag{2}$$

where $\boldsymbol{x}_i$ is the i-th row of $\boldsymbol{X}$, $\boldsymbol{z}_i^{(j)}$ is the i-th row of predicted feature $\boldsymbol{Z}^{(j)} = f_D(\boldsymbol{A}, \widetilde{\boldsymbol{H}}^{(j)})$, and $\gamma >= 1$ is the scaled coefficient. In this work, the decoder f_D for feature reconstruction consists of a light single-layer GAT. Therefore, this strategy is very efficient and only incurs negligible computational costs.

Latent representation prediction. In line with the mask-then-predict, the focus of this part is on constructing an additional informative prediction target that is minimally influenced by the direct effects of input features. To achieve this, we propose to perform the prediction in representation space beyond input feature space.

Considering that the neural networks can essentially serve as denoising encoders [32] and encode high-level semantics [5, 56], we propose to employ a network as the target generator to produce latent prediction targets from the unmasked graph. Formally, we denote the GNN encoder as $f_E(\cdot;\theta) = f_E$. We also define a projector $g(\cdot;\theta)$, corresponding to the decoder f_D in input feature reconstruction, to map the code $\boldsymbol{H}$ to representation space for prediction. θ denotes their learnable weights. The target generator network shares the same architecture as the encoder and projector but uses a different set of weights, i.e., $f'_E(\cdot;\xi)$ and $g'(\cdot;\xi)$. During the pretraining, the unmasked graph is first passed through the target generator to produce target representation $\bar{\boldsymbol{X}}$. Then the encoding results $\boldsymbol{H}$ of the masked graph $\mathcal{G}(\boldsymbol{A}, \widetilde{\boldsymbol{X}})$ are projected to representation space, resulting in $\bar{\boldsymbol{Z}}$ for latent prediction:

$$\bar{\boldsymbol{Z}} = g(\boldsymbol{H};\theta),\ \bar{\boldsymbol{X}} = g'(f'_E(\boldsymbol{A}, \boldsymbol{X};\xi);\xi) \tag{3}$$

The encoder and projector network are trained to match the output of the target generator on masked nodes. Of particular interest, encouraging the correspondence of unmasked nodes would bring slight benefits to our framework. This may attribute to the masking operation implicitly serving as a special type of augmentation. We learn the parameters θ of the encoder and projector by minimizing the following scaled cosine error with gradient descent.

$$\mathcal{L}_{latent} = \frac{1}{N} \sum_{i}^{N} \left(1 - \frac{\bar{\boldsymbol{z}}_i^\top \bar{\boldsymbol{x}}_i}{\|\bar{\boldsymbol{z}}\| \cdot \|\bar{\boldsymbol{x}}\|}\right)^\gamma \tag{4}$$

And the parameters of target generator ξ are updated via an exponential moving average of θ [28] using weight decay τ:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta \tag{5}$$

The target generator shares similarities with the teacher network in self-knowledge distillation [5, 56] or contrastive methods [12]. But there exist differences in both the motivation and implementation: GraphMAE2 aims to direct the prediction of masked nodes with output from the unmasked graph as the target. In contrast, knowledge-distillation and contrastive methods target maximizing the consistency of two augmented views. The characteristic is that our method does not rely on any elaborate data augmentations and thus has no worry about whether the augmentations would alter the semantics in particular graphs.

Training and inference. The overall training flow of GraphMAE2 is summarized in Figure 2. Given a graph, the original graph is passed through the target generator to generate the latent target $\bar{\boldsymbol{X}}$. Then we randomly mask the features of a certain portion of nodes and feed the masked graph with partially observed features into the encoder $f_E(\boldsymbol{A}, \tilde{\boldsymbol{X}};\theta)$ to generate the code $\boldsymbol{H}$. Next, the decoding consists of two streams. On the one hand, we apply the multi-view random re-masking to replace re-masked nodes in $\boldsymbol{H}$ with [DMASK] token, and the results are fed into the decoder f_D to reconstruct the input $\boldsymbol{X}$. On the other hand, another decoder g is adapted to predict the latent target $\bar{\boldsymbol{X}}$. We combine the two losses with a mixing coefficient λ during training:

$$\mathcal{L} = \mathcal{L}_{input} + \lambda \mathcal{L}_{latent} \tag{6}$$

Note that the time and space complexity of GraphMAE2 is linear with the number of nodes N, and thus it can scale to extremely large graphs. When applying to downstream tasks, the decoder and target generator are discarded, and only the GNN encoder is used to generating embeddings or finetuned for downstream tasks.

Extending to large-scale graph Extending self-supervised learning to large-scale graphs is of great practical significance, yet few efforts have been devoted to this scenario. Existing graph SSL works

focus more on small graphs, and current works [13, 39] concerning large graphs simply conduct experiments based on existing graph sampling developed under the supervised setting, e.g., neighborhood sampling [13] or ClusterGCN [7]. Though it is a feasible implementation, there exist several challenges that may affect the performance under the self-supervised setting:

(1) Self-supervised learning generally benefits from relatively larger model capacity, i.e., wider and deeper networks, whereas GNNs suffer from the notorious problem of over-smoothing and over-squashing when stacking more layers [1, 27]. One feasible way to circumvent the problems is to decouple the receptive field and depth of GNN by extracting a local subgraph [52].
(2) In the context of masked feature prediction, GraphMAE2 has a preference for a well-connected local structure since each node would rely on aggregating messages of its neighboring nodes to generate embedding and reconstruct features.

Most popular sampling methods tend to generate highly sparse yet wide subgraphs as regularization in supervised setting [53, 60], or only bear shallow GNNs in inference [7, 13]. In light of these defects, we imitate the idea from [52] and are motivated to construct densely connected subgraphs for GraphMAE2 to tackle the scalability on large-scale graphs. Thus, we utilize local clustering [2, 36] algorithms to seek local and dense subgraphs. Local clustering aims to find a small cluster near a given seed in the large graph. And it has been proven to be very useful for identifying structures at small-scale or meso-scale [23, 26]. Though many local clustering algorithms have been developed, we leverage the popular spectral-based PPR-Nibble [2] for efficient implementation. PPR-Nibble adopts the personalized PageRank (PPR) vector $\boldsymbol{p}_i$, which reflects the significance of all nodes $\mathcal{V}$ in the graph for the node v_i, to generate a local cluster for a given node v_i. Previous works [50, 59] provide a theoretical guarantee for the quality of the generated local cluster of PPR-Nibble. The theorem in [50, 59] (described in Appendix A.1) indicates that the algorithm can generate local clusters of a relatively small conductance, which meets our expectations for densely connected local subgraphs. In our work, we select the k-largest elements in $\boldsymbol{p}_i$ to form a local cluster for node v_i for computational efficiency.

The PPR-Nibble can be implemented efficiently through fast approximation, and the computational complexity is linear with the number of nodes. One by-product is that this strategy decreases the discrepancy between training and inference since they are both conducted on the extracted subgraphs. In GraphMAE2, the self-supervised learning is conducted upon all nodes within a cluster. In downstream finetuning or inference, we generate the prediction or embedding for node v_i using the local cluster induced by v_i.

3 EXPERIMENTS

In this section, we compare our proposed self-supervised framework with state-of-the-art methods in the setting of unsupervised representation learning and semi-supervised node classification. In this work, we focus on the node classification task, which aims to predict unlabeled nodes. Note that GraphMAE2 is a general SSL method and can be applied to various graph learning tasks.

Table 2: Statistics of datasets.

Datasets	#Nodes	#Edges	#Features
Cora	2,485	5,069	1,433
Citeseer	2,110	3,668	3,703
Pubmed	19,717	44,324	300
ogbn-Arxiv	169,343	1,166,243	128
ogbn-Products	2,449,029	61,859,140	100
MAG-Scholar-F	12,403,930	358,010,024	128
ogbn-Papers100M	111,059,956	1,615,685,872	128

Table 3: Linear probing results on large-scale datasets with mini-batch training. We report accuracy(%) for all datasets. *Random-Init* represents a random-initialized model without any self-supervised pretraining.

	Arxiv	Products	MAG	Papers100M
MLP	55.50±0.23	61.06±0.08	39.11±0.21	47.24±0.31
SGC	66.92±0.08	74.87±0.25	54.68±0.23	63.29±0.19
Random-Init	68.14±0.02	74.04±0.06	56.57±0.03	61.55±0.12
CCA-SSG	68.57±0.02	75.27±0.05	51.55±0.03	55.67±0.15
GRACE	69.34±0.01	79.47±0.59	57.39±0.02	61.21±0.12
BGRL	70.51±0.03	78.59±0.02	57.57±0.01	62.18±0.15
GGD[1]	-	75.70±0.40	-	63.50±0.50
GraphMAE	71.03±0.02	78.89±0.01	58.75±0.03	62.54±0.09
GraphMAE2	**71.89**±0.03	**81.59**±0.02	**59.24**±0.01	**64.89**±0.04

[1] The source code of GGD is not released and its results on Arxiv and MAG-Scholar-F are not reported in the paper.

3.1 Evaluating on Large-scale Datasets

Datasets. The experiments are conducted on four public datasets of different scales, varying from hundreds of thousands of nodes to hundreds of millions. The statistics are listed in Table 2. In the experiments, we follow the official splits in [20] for ogbn-Arxiv/Products/Papers100M. As for MAG-Scholar-F, we randomly select 5%/5%/40% nodes for training/validation/test, respectively.

For ogbn-Products [20] and MAG-Scholar-F [3], their node features are generated by first extracting bag-of-words vectors from the product descriptions or paper abstracts and then conducting Principal Component Analysis (PCA) to reduce the dimension. ogbn-Arxiv and ogbn-Papers100M [20] are both citation networks, and they leverage word2vec model to obtain node features by averaging the embeddings of words in the paper's title and abstract.

Baselines. We compare GraphMAE2 with state-of-the-art self-supervised graph learning methods, including contrastive methods, GRACE [57], BGRL [39], CCA-SSG [54], and GGD [55] as well as a generative method GraphMAE [18]. Other methods are not compared because they are not scalable to large graphs, e.g., MVGRL [14], or the source code has not been released, e.g., InfoGCL [48]. As stated in [40], random models can have a strong inductive bias on graphs and are non-trivial baselines. Therefore, we also report the results of the randomly-initialized GNN model and Simplified Graph Convolution (SGC) [46], which simply stacks

Table 4: Results of fine-tuning the pretrained GNN with 1% and 5% labeled training data on large-scale datasets. We report accuracy(%) for all datasets. *Random-Init* represents a random-initialized model without any self-supervised pretraining.

	Arxiv		Products		MAG		Papers100M	
Label ratio	1%	5%	1%	5%	1%	5%	1%	5%
Random-Init$_{SAINT}$	63.45±0.32	67.67±0.42	72.23±0.44	75.21±0.35	43.55±0.15	51.03±0.13	56.47±0.23	60.63±0.22
CCA-SSG	64.14±0.21	68.32±0.32	75.89±0.43	78.47±0.42	42.62±0.15	51.32±0.11	55.68±0.24	59.78±0.08
GRACE	64.53±0.47	69.21±0.45	**77.13**±0.64	79.67±0.54	43.59±0.13	51.35±0.12	55.45±0.23	59.38±0.15
BGRL	65.05±1.17	69.01±0.34	76.32±0.54	79.46±0.43	43.92±0.11	51.69±0.15	55.12±0.23	60.40±0.54
Random-Init$_{LC}$	64.79±0.45	67.89±0.27	71.87±0.34	74.42±0.43	43.66±0.14	50.86±0.12	57.48±0.23	61.41±0.25
GraphMAE	65.78±0.69	69.78±0.26	75.87±0.43	79.21±0.33	48.33±0.18	53.12±0.12	58.29±0.15	62.00±0.12
GraphMAE2	**66.86**±0.53	**70.16**±0.28	76.98±0.36	**80.52**±0.23	**49.01**±0.15	**53.58**±0.11	**58.69**±0.38	**62.87**±0.64

Table 5: Experimental results on small-scale datasets. We report accuracy(%) for all datasets.

	Cora	CiteSeer	PubMed
GCN	81.5	70.3	79.0
GAT	83.0±0.7	72.5±0.7	79.0±0.3
GAE	71.5±0.4	65.8±0.4	72.1±0.5
DGI	82.3±0.6	71.8±0.7	76.8±0.6
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
BGRL	82.7±0.6	71.1±0.8	79.6±0.5
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
GGD	83.9±0.4	73.0±0.6	81.3±0.8
GraphMAE	84.2±0.4	73.4±0.4	81.1±0.4
GraphMAE2	**84.5**±0.6	73.4±0.3	**81.4**±0.5

the propagated features of different orders, to examine whether the SSL learns a more effective propagation paradigm. Comparing with them can reflect the contributions of self-supervised learning. To extend to large graphs for baselines, we adopt GraphSAINT [53] sampling strategy, which is proved to perform better than widely-adopted Neighborhood Sampling [13] in many cases. GraphMAE and GraphMAE2 are trained based on the presented local clustering algorithm. For all baselines, we employ Graph Attention Network (GAT) [41] as the backbone of the encoder f_E and the decoder for input feature reconstruction f_D.

Evaluation. We evaluate our approach with two setups: (i) linear probing and (ii) fine-tuning. For *linear probing*, we first generate node embeddings with the pretrained encoder. Then we discard the encoder and train a linear classifier using the embeddings under the supervised setting. For *fine-tuning*, we add a linear classifier on top of node representations and fine-tune all parameters under the semi-supervised setting. We randomly sample 1% and 5% labels from the training set to finetune the pretrained model, aiming to test the ability to transfer knowledge learned from unlabeled data to facilitate the downstream performance with a few labels. For both cases, we run the experiments for 10 trials with random seeds and report the average accuracy and standard variance.

Results. The results of linear probing are illustrated in Table 3. And we interpret the result from 3 aspects. First, GraphMAE2 achieves better results than all self-supervised baselines across all datasets. This manifests that the proposed method can learn more discriminative representations under the unsupervised setting. Notably, GraphMAE2 improves upon GraphMAE by a margin of 1.91% and 2.35% (absolute difference) on MAG-Scholar-F and Papers100M. These results demonstrate the significance of the proposed improvement. Second, our approach, together with most baselines, outperforms the randomly initialized, untrained model by a large margin. This demonstrates that the designed self-supervised pretext task guides the model to better capture the semantic and structural information than the untrained model. As a comparison, improper self-supervised signals can lead the model to perform even worse, yet this phenomenon is ignored in most previous studies. Third, GraphMAE2 consistently generates better performance than SGC. Despite the fact that methods based on decoupled propagation have achieved promising results in the full-supervised setting with the assistance of self-training, we demonstrate that graph neural networks, like GAT, could still be more powerful at generating node representations in the unsupervised setting.

Table 4 shows the results of finetuning the pretrained model in the semi-supervised setting. On the one hand, it is observed that self-supervised pretraining of GraphMAE2 benefits downstream supervised training with significant performance gains. In ogbn-Products, with the pre-trained model, the performance achieves improvement by above 5.1%. And finetuning with only 5% of data generates comparative performance (80.52%) to many supervised learning methods in OGB leaderboard[2] with all 100% of training data, e.g., GraphSAINT: 80.27, Cluster-GAT: 79.23%. On the other hand, our approach remarkably achieves state-of-the-art performance for all benchmarks. The only exception is on the Products dataset with 1% training data, where GraphMAE2 slightly underperforms GRACE yet still achieves the second-best result. It should be noted that in ogbn-Papers100M, only GraphMAE2 and GraphMAE generate better performance than the random-initialized model, while all contrastive baselines fail to bring improvement with pretraining. One possible reason is that the data augmentation techniques used in baselines fail in this dataset.

[2]https://ogb.stanford.edu/docs/leader_nodeprop/

Table 6: Ablation studies of GraphMAE2 key components.

	Products	MAG	Papers100M
GraphMAE2	81.59±0.02	59.24±0.01	64.89±0.04
w/o random remask	81.04±0.03	59.01±0.02	64.16±0.02
w/o latent rep pred.	80.01±0.02	58.87±0.02	62.98±0.01
w/o input recon.	76.88±0.02	55.20±0.02	59.20±0.00
GraphMAE	78.89±0.01	58.75±0.03	62.54±0.09

Table 7: Ablation study on sampling strategy. "SAINT" refers to GraphSAINT, "Cluster" refers to Cluster-GCN, and "LC" refers the presented local clustering algorithm.

	Strategy	Products	MAG	Papers100M
GRACE	*SAINT*	79.47±0.59	57.39±0.02	61.21±0.12
BGRL	*SAINT*	78.59±0.02	57.57±0.01	62.18±0.15
GraphMAE2	*SAINT*	80.96±0.03	58.75±0.03	64.21±0.11
GraphMAE2	*Cluster*	79.35±0.05	58.05±0.02	63.77±0.11
GraphMAE2	*LC*	81.59±0.02	59.24±0.01	64.89±0.12

3.2 Evaluating on Small-scale Datasets

Experimental setup. We also report results on small yet widely adopted datasets, i.e., Cora, Citeseer, and PubMed [49], to show the generality of our method. We follow the public data splits as [14, 42].

We compare GraphMAE2 with state-of-the-art self-supervised graph learning methods, including contrastive methods, DGI [42], MVGRL [14], GRACE [57], BGRL [39], InfoGCL [48], CCA-SSG [54], and GGD [55] as well as generative methods GAE [24], GraphMAE [18]. For the evaluation, we employ the linear probing mentioned above and report the average performance of accuracy on the test nodes based on 20 random initialization. The GNN encoder and decoder both use standard GAT as the backbone, and an MLP is employed as the representation projector g.

Results. From Table 5, we can observe that our approach generally outperforms all baselines in all datasets, suggesting that GraphMAE2 serves as a general and effective framework for graph self-supervised learning on graphs of varied scales. We observe that the improvement over GraphMAE is not as significant as that in the experiments of large graphs. We guess that the reason lies in the construction of input node features. Bag-of-word vectors behave more like discrete features as words in text and pixels image and, thus are less noisy as reconstruction targets. And this may partially support our assumption that GraphMAE2 is more advantageous than GraphMAE when there is more noise in the data.

3.3 Ablation Studies

We further conduct ablation studies to verify the contributions of the designs in GraphMAE2. We choose linear probing for evaluation.

Ablation on the learning framework. We study the influence of the proposed two strategies—latent representation prediction and multi-view random re-masking. The results are shown in Table 6 , where the "w/o random re-mask" represents that we adopt the fixed re-masking strategy as GraphMAE. It is observed that the two strategies both contribute to performance improvement. This demonstrates the effectiveness and further supports our motivation for the effects of input feature quality. Latent representation prediction brings more benefits as the accuracy drops more when the component is removed, e.g., -1.58% in ogbn-Products and -1.91% in ogbn-Papers100M, than the multi-view random re-masking, e.g., -0.45% and -0.73%. The target generator network provides valuable guidance and constraints on the encoded representation.

We also conduct an experiment by totally removing the input feature reconstruction, and the training only involves latent representation prediction. In such cases, the learning degrades to self-knowledge distillation without heavy data augmentation and causes a significant drop in performance. The network may fall into a trivial solution and may learn collapsed representation as the results are worse than GraphMAE or even worse than the random-initialized model in MAG-Scholar-F and ogbn-Papers100M. This indicates that feature reconstruction substantially supports SSL, and the proposed two strategies serve as auxiliaries to help overcome the deficiency. Overall speaking, the results confirm that the superior performance of GraphMAE2 comes from the design rather than any individual contribution.

Ablation on sampling strategies. Table 7 shows the influence of different sampling strategies. We compare local clustering against two popular subgraph sampling algorithms–ClusterGCN [7] and GraphSAINT [53]. Neighborhood sampling is not included since it is not friendly to masked feature reconstruction, especially with GNN decoder. The local clustering is conducive to the excellent performance of GraphMAE2, as our algorithm shows an advantage over GraphSAINT and Cluster-GCN with 0.57% and 1.49% improvement on average. Recall that GraphSAINT tends to sample nodes globally, and thus the subgraph is more sparse. Although ClusterGCN generates large and connected partitions, it suffers from high information loss as edges between clusters are abandoned. And the results indicate that the densely-connected local subgraph produced can generate better representations. In addition, we compare our approach with the strongest baselines using the same sampling strategy. And GraphMAE2 still generates a 1.49% advantage in ogbn-Products, demonstrating its effectiveness.

Ablation on model capacity. The effects of model capacity have attracted significant attention in other fields like CV [15] and NLP [4] as it is demonstrated that SSL can largely benefit from increasing model parameters. We take an interest in whether the scaling law of model capacity also applies to GNNs. Specifically, we employ a GAT as the encoder and explore the influence of depth and width. And experiments are conducted on ogbn-Products of around 2 million nodes. The results are shown in Figure 3. Increasing the hidden size drives the model to achieve better performance. Doubling the hidden size leads to a performance improvement of nearly 2% in accuracy when the hidden size does not exceed 1024. But further enlarging the width only brings very marginal gain.

Another way to increase the capacity is to stack more network layers. Figure 3 shows that increasing the depth can slightly boost the performance as the accuracy increases by 0.65% when the number of network layers is increased from 2 to 4. And the benefits

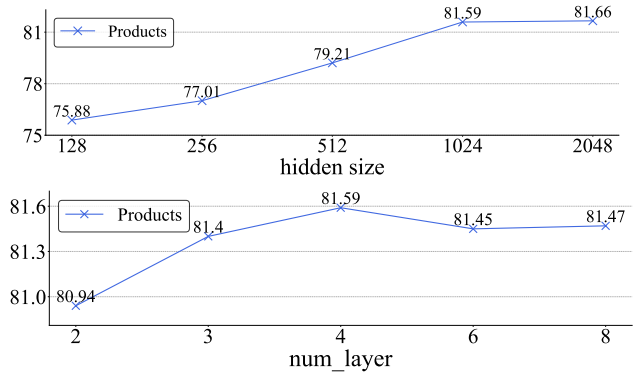

Figure 3: Ablation study on hidden size and the number of GNN layers. The effects of width are more significant than depth.

would diminish when stacking more layers. One possible reason is that deeper GNNs are harder to optimize, while current downstream tasks or semantics of homogeneous structured data benefit little from more complex network architecture. It is observed that the influence of the depth is less remarkable than the width of GNN.

4 RELATED WORK

In this section, we introduce related works about graph self-supervised learning and scalable graph neural networks.

4.1 Graph Self-Supervised Learning

Graph self-supervised learning (SSL) can be roughly categorized into two genres, including graph contrastive learning and graph generative learning, based on the learning paradigm.

Contrastive methods. Contrastive learning is an important way to learn representations in a self-supervised manner and has achieved successful practices in graph learning [14, 29, 35, 37, 42, 51, 55]. DGI [42] and InfoGraph [37] adopt the local-global mutual information maximization to learn node-level and graph-level representations. MVGRL [14] leverages graph diffusion to generate an additional view of the graph and contrasts node-graph representations of distinct views. GCC [35] utilizes subgraph-based instance discrimination and adopts InfoNCE as the pre-training objective with MoCo-style dictionary [16]. GRACE [57], GraphCL [51], and GCA [58] learn the node or graph representation by maximizing agreement between different augmentations. GGD [55] analyzes the defect of existing contrastive methods (i.e., improper usage of Sigmoid function) and proposes a group discrimination paradigm.

To avoid the expensive computation of negative samples, some researchers propose graph SSL methods that do not require negative samples. BGRL [39] uses an online encoder and a target encoder to contrast two augmented versions without negative samples. CCA-SSG [54] leverages a feature-level objective for graph SSL, inspired by Canonical Correlation Analysis methods.

Most graph contrastive learning methods rely on complex graph augmentation operators to generate two different views, which are used to be contrasted or correlated. However, the theoretical understanding of augmentation techniques on the graph SSL has not been well-studied. The choice of graph augmentation operators mostly depends on the empirical analysis of researchers. Although some works [44, 47] have made attempts to alleviate this reliance, it still remains further exploration.

Generative methods. Graph autoencoders (GAE) and VGAE [24] follow the spirit of autoencoder [17] to learn node representations. Following VGAE, most GAEs focus on reconstructing the structural information (e.g., ARVGA [33]) or adopt the reconstruction objective of both structural information and node attributes (e.g., MGAE [43], GALA [34]). NWR-GAE [38] designs a graph decoder to reconstruct the entire neighborhood information of graph structure. kgTransformer [31] applies the masked GAE to knowledge graph reasoning. However, these previous GAE models do not perform well on node-level and graph-level classification tasks. To mitigate the performance gap, GraphMAE [18] leverages masked feature reconstruction as the objective with auxiliary designs and obtains comparable or better performance than contrastive methods. In addition to graph autoencoders, inspired by the success of autoregressive models in natural language processing, GPT-GNN [22] designs an attributed graph generation task, including attribute and edge generation, for pre-training GNN models. Generative methods can alleviate the deficiency of contrastive rivals since the objective of generative ones is to directly reconstruct the input graph data.

4.2 Scalable Graph Neural Networks

There are two genres of methods for scalable GNNs. One is based on sampling that trains GNN models on sampled mini-batch data. GraphSAGE [13] adopts the neighbor sampling method to conduct the mini-batch training. FastGCN [6] performs layer-wise sampling and leverages importance sampling to reduce variance. GraphSAINT [53] and ClusterGCN [7] both produce a subgraph from the original graph for mini-batch training by graph partition or random walks. Another paradigm for scalable GNNs is to decouple the message propagation and feature transformation. SGC [46] removes the nonlinear functions and is equivalent to a pre-processing K-step propagation and a logistic regression on the propagated features. SIGN [11] extends SGC to stack the propagated results of different hops and graph filters, and then only trains MLPs for applications. These decoupled methods achieve excellent performance in the supervised setting but have no advantage in generating high-quality embeddings.

5 CONCLUSION

In this work, we explore graph self-supervised learning with masked feature prediction. We first examine its potential concern that the discriminability of input features hinders current attempts to achieve promising performance. Then we present a framework GraphMAE2 to address this issue by imposing regularization on the prediction. We focus on the decoding stage and introduce latent representation target and randomness to input reconstruction. The novel decoding strategy significantly boosts the performance in realistic large-scale benchmarks. Our work further supports that node-level signals could provide abundant supervision for masked graph self-supervised learning and deserves further exploration.

Acknowledgements. This research was supported by Natural Science Foundation of China(NSFC) 61825602 and 62276148, Tsinghua-Bosch Joint ML Center, and Zhipu.AI.

REFERENCES

[1] Uri Alon and Eran Yahav. 2020. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205* (2020).

[2] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *FOCS*. IEEE, 475–486.

[3] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *KDD*. 2464–2473.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*, Vol. 33.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *ICCV*. 9650–9660.

[6] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. In *ICLR*.

[7] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*. 257–266.

[8] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2022. Node Feature Extraction by Self-Supervised Multi-scale Neighborhood Prediction. In *ICLR*.

[9] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. 2020. Adaptive graph encoder for attributed graph embedding. In *KDD*. 976–985.

[10] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph random neural networks for semi-supervised learning on graphs. In *NeurIPS*.

[11] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198* (2020).

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *ICML*.

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *CVPR*.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

[17] Geoffrey E Hinton and Richard Zemel. 1993. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *NeurIPS*, J. Cowan, G. Tesauro, and J. Alspector (Eds.), Vol. 6. Morgan-Kaufmann.

[18] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *KDD*.

[19] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430* (2021).

[20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.

[21] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. In *ICLR*.

[22] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*.

[23] Lucas GS Jeub, Prakash Balachandran, Mason A Porter, Peter J Mucha, and Michael W Mahoney. 2015. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E* 91, 1 (2015).

[24] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[25] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[26] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[27] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *ICCV*. 9267–9276.

[28] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[29] Xiao Liu, Haoyun Hong, Xinghao Wang, Zeyi Chen, Evgeny Kharlamov, Yuxiao Dong, and Jie Tang. 2022. Selfkg: self-supervised entity alignment in knowledge graphs. In *WWW*. 860–870.

[30] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *TKDE* (2021).

[31] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. 2022. Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries. In *KDD*. 1120–1130.

[32] Yao Ma, Xiaorui Liu, Tong Zhao, Yozen Liu, Jiliang Tang, and Neil Shah. 2021. A unified view on graph neural networks as graph signal denoising. In *CIKM*.

[33] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*.

[34] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *ICCV*. 6519–6528.

[35] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *SIGKDD*.

[36] Daniel A Spielman and Shang-Hua Teng. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on computing* 42, 1 (2013), 1–26.

[37] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR'20*.

[38] Mingyue Tang, Carl Yang, and Pan Li. 2022. Graph Auto-Encoder via Neighborhood Wasserstein Reconstruction. *arXiv preprint arXiv:2202.09025* (2022).

[39] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. 2022. Large-Scale Representation Learning on Graphs via Bootstrapping. In *ICLR*.

[40] Puja Trivedi, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, and Danai Koutra. 2022. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *WWW*. 1538–1549.

[41] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[42] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In *ICLR*.

[43] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*. 889–898.

[44] Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. 2022. Augmentation-Free Graph Contrastive Learning. In *AAAI*.

[45] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. 2022. Masked feature prediction for self-supervised visual pre-training. In *CVPR*. 14668–14678.

[46] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*. PMLR.

[47] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. 2022. SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation. In *WWW*. 1070–1079.

[48] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. 2021. Infogcl: Information-aware graph contrastive learning. *NeurIPS* 34 (2021).

[49] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*. PMLR, 40–48.

[50] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *KDD*. 555–564.

[51] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *NeurIPS*.

[52] Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. 2021. Decoupling the depth and scope of graph neural networks. In *NeurIPS*.

[53] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graphsaint: Graph sampling based inductive learning method. In *ICLR*.

[54] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. In *NeurIPS*.

[55] Yizhen Zheng, Shirui Pan, Vincent Cs Lee, Yu Zheng, and Philip S Yu. 2022. Rethinking and Scaling Up Graph Contrastive Learning: An Extremely Efficient Approach with Group Discrimination. In *NeurIPS*.

[56] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. 2022. ibot: Image bert pre-training with online tokenizer. In *ICLR*.

[57] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

[58] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*. 2069–2080.

[59] Zeyuan Allen Zhu, Silvio Lattanzi, and Vahab Mirrokni. 2013. A local algorithm for finding well-connected clusters. In *ICML*. PMLR, 396–404.

[60] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *NeurIPS*.

A APPENDIX

A.1 Theorem for Local Clustering.

Theorem A.1 (Theorem 1 in [59]; Theorem 4.3 in [50]). *Let $T \subset V$ be some unknown targeted cluster that we are trying to retrieve from an unweighted graph. Let η be the inverse mixing time of the random walk on the subgraph induced by T. Then there exists $T^g \subseteq T$ with $\text{vol}(T^g) \geq \text{vol}(T)/2$, such that for any seed $u \in T^g$, PPR-Nibble with $\alpha = \Theta(\eta)$ and $\epsilon \in \left[\frac{1}{10\,\text{vol}(T)}, \frac{1}{5\,\text{vol}(T)}\right]$ outputs a set S with*

$$\Phi(S) \leq \widetilde{O}\left(\min\left\{\sqrt{\Phi(T)}, \Phi(T)/\sqrt{\eta}\right\}\right).$$

Here, $\text{vol}(S) \triangleq \sum_{v_i \in \mathcal{S}} degree(v_i)$ is the graph volume, and $\Phi(S) \triangleq \frac{\sum_{v_i \in \mathcal{S}} \sum_{v_j \in \mathcal{V}-\mathcal{S}} \mathbf{A}(i,j)}{\min(\text{vol}(\mathcal{S}), \text{vol}(\mathcal{V}-\mathcal{S}))}$ defines the conductance of a non-empty node set $\mathcal{S} \subset \mathcal{V}$.

A.2 Finetuning Results for Supervised Learning

Table 8: Fine-tuning results of mini-batch training with full labels. We report accuracy(%) for all datasets.

	Arxiv	Products	MAG.	Papers100M
MLP	55.50	61.06	39.11	47.24
SGC	66.92	74.87	54.68	63.29
Random-Init$_{SAINT}$	71.88	81.15	57.11	66.36
Random-Init$_{LC}$	71.47	81.59	56.84	66.47
CCA-SSG	72.24	80.61	56.54	66.11
GRACE	72.54	82.45	57.23	66.45
BGRL	72.48	82.37	57.34	66.57
GraphMAE	72.38	81.89	59.77	**66.64**
GraphMAE2	**72.69**	**83.65**	**60.01**	**66.66**

Table 8 shows the results of finetuning the pretrained encoder with all labels in the training set. It is observed that self-supervised learning can still lead to improvements in the supervised setting. GraphMAE2 can bring 1.22%-3.17% improvement compared to the randomly initialized model on three datasets, i.e., ogbn-Arxiv, Products, and MAG-Scholar-F. The only exception is ogbn-Papers100M, in which the improvement is only 0.2% in accuray. The reason could be that the splitting strategy of this dataset (train:val:test = 78%:8%:14%) resulted in an overly well-labeled training set. It is worth mentioning that on ogbn-Products, our method achieves better results than the current best single model in OGB leaderboard[3], i.e., GAMLP: 83.54%.

A.3 Linear Probing Results on ogbn-Arxiv with Full-graph Training.

As ogbn-Arxiv dataset is relatively small, we can conduct full batch training and inference. Table 9 shows the results of GraphMAE2 as well as baselines under full batch training. Compared to GraphMAE, GraphMAE2 has a 0.2% improvement. It is worth noting that GraphMAE results are better when trained with full-batch than with mini-batch, while the mini-batch training results of GraphMAE2 are comparable to the full-batch results.

[3]https://ogb.stanford.edu/docs/leader_nodeprop/

Table 9: Linear probing results on ogbn-Arxiv with full-graph training. We report accuracy(%).

	Arxiv
GCN	71.74±0.29
GAT	72.10±0.13
DGI	70.34±0.16
GRACE	71.51±0.11
BGRL	71.64±0.12
CCA-SSG	71.24±0.20
GraphMAE	71.75±0.17
GraphMAE2	**71.95**±0.08

A.4 Implementation Notes

Running Environment. Our proposed framework is implemented via PyTorch. For our methods, the experiments are conducted on a Linux machine with 1007G RAM, and 8 NVIDIA A100 with 80GB GPU memory. As for software versions, we use Python 3.9, PyTorch 1.12.0, OGB 1.3.3, and CUDA 11.3.

Model Configuration. For our model and all baselines, we pretrain the model using AdamW Optimizer with cosine learning rate decay without warmup. More details about pre-training hyper-parameters are in Table 10.

Table 10: Hyper-parameters on large-scale datasets.

	Arxiv	Products	MAG	Papers100M
masking rate	0.5	0.5	0.5	0.5
re-masking rate	0.5	0.5	0.5	0.5
num_re-masking	3	3	3	3
coefficient λ	10.0	5.0	0.1	10.0
hidden_size	1024	1024	1024	1024
num_layer	4	4	4	4
lr	0.0025	0.002	0.001	0.001
weight_decay	0.06	0.06	0.04	0.05
max epoch	60	20	10	10

A.5 Baselines

For large datasets, we choose four baselines GRACE [57], BGRL [39], CCA-SSG [54] and GraphMAE [18]. To have a fair comparison, we download the public source code and use the GAT backbone. We adapted their code to integrate with sampling algorithms to run on large-scale graphs. For GGD [55], we can only report the results available in the original paper since the authors have not released the code. The sources of the codes used are as follows:

- BRGL: https://github.com/Namkyeong/BGRL_Pytorch
- GRACE: https://github.com/CRIPAC-DIG/GRACE
- CCA-SSG: https://github.com/hengruizhang98/CCA-SSG/
- GraphMAE:https://github.com/THUDM/GraphMAE