GraphMAE: Self-Supervised Masked Graph Autoencoders

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

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

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

Yuxiao Dong*
Tsinghua University
yuxiaod@tsinghua.edu.cn

Hongxia Yang
DAMO Academy, Alibaba Group
yang.hyx@alibaba-inc.com

Chunjie Wang
BirenTech Research
cjwang@birentech.com

Jie Tang*
Tsinghua University
jietang@tsinghua.edu.cn

ABSTRACT

Self-supervised learning (SSL) has been extensively explored in recent years. Particularly, generative SSL has seen emerging success in natural language processing and other AI fields, such as the wide adoption of BERT and GPT. Despite this, contrastive learning—which heavily relies on structural data augmentation and complicated training strategies—has been the dominant approach in graph SSL, while the progress of generative SSL on graphs, especially graph autoencoders (GAEs), has thus far not reached the potential as promised in other fields. In this paper, we identify and examine the issues that negatively impact the development of GAEs, including their reconstruction objective, training robustness, and error metric. We present a masked graph autoencoder GraphMAE[1] that mitigates these issues for generative self-supervised graph learning. Instead of reconstructing graph structures, we propose to focus on feature reconstruction with both a masking strategy and scaled cosine error that benefit the robust training of GraphMAE. We conduct extensive experiments on 21 public datasets for three different graph learning tasks. The results manifest that GraphMAE—a simple graph autoencoder with careful designs—can consistently generate outperformance over both contrastive and generative state-of-the-art baselines. This study provides an understanding of graph autoencoders and demonstrates the potential of generative self-supervised learning on graphs.

CCS CONCEPTS

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

KEYWORDS

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

*Yuxiao Dong and Jie Tang are the corresponding authors.
[1]The code is publicly available at https://github.com/THUDM/GraphMAE.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539321

ACM Reference Format:
Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539321

1 INTRODUCTION

Self-supervised learning (SSL), which can be generally categorized into generative and contrastive methods [23], has found widespread adoption in computer vision (CV) and natural language processing (NLP). While contrastive SSL methods have experienced an emergence in the past two years, such as MoCo [13], generative SSL has been gaining steadily increasing significance thanks to several groundbreaking practices, such as the well-established BERT [4] and GPT [30] in NLP as well as the very recent MAE [12] in CV.

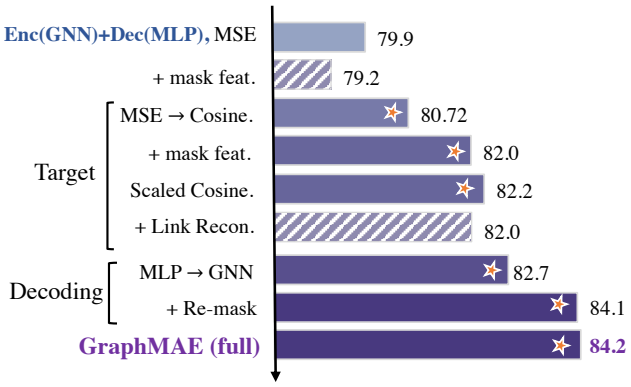
However, in the context of graph learning, contrastive SSL has been the dominant approach, especially for two important tasks—node and graph classifications [11, 29, 40]. Its success has been largely built upon relatively complicated training strategies. For example, the bi-encoders with momentum updates and exponential moving average are usually indispensable to stabilize the training of GCC [29] and BGRL [37]. Additionally, negative samples are necessary for most contrastive objectives, often requiring arduous labor to sample or construct from graphs, e.g., GRACE [60], GCA [61], and DGI [40]. Finally, its heavy reliance on high-quality data augmentation proves to be the pain point of contrastive SSL, e.g., CCA-SSG [57], as graph augmentation is mostly based on heuristics whose effectiveness varies drastically from graph to graph.

Self-supervised graph autoencoders (GAEs) can naturally avoid the aforementioned issues in contrastive methods, as its learning objective is to directly reconstruct the input graph data [6, 20]. Take VGAE for example, it [20] targets at predicting missing edges. EP [6] instead proposes to recover vertex features. GPT-GNN [17] proposes an autoregressive framework to perform node and edge reconstruction iteratively. Later GAEs, including ARVGA [26], MGAE [42], GALA [27], GATE [31], and AGE [3], majorly focus on the objectives of link prediction and graph clustering.

Dilemmas. Despite their simple forms and various recent attempts on them [3, 16, 18, 27, 31, 55], the development of self-supervised GAEs has been thus far lagged behind contrastive learning. To date,

Methods	Feat. Loss	AE	No Struc.	Mask Feat.	GNN Decoder	Re-mask Dec.	Space
VGAE [20]	n/a	✓	-	-	-	-	$O(N^2)$
ARVGA [26]	n/a	✓	-	-	-	-	$O(N^2)$
MGAE [42]	MSE	✓	-	✓	-	-	$O(N)$
GALA [27]	MSE	✓	✓	-	✓	-	$O(N)$
GATE [31]	MSE	✓	-	-	✓	-	$O(N)$
AttrMask [16]	CE	✓	✓	✓	-	-	$O(N)$
GPT-GNN [17]	MSE	-	-	✓	-	-	$O(N)$
AGE [3]	n/a	✓	-	-	-	-	$O(N^2)$
NodeProp [18]	MSE	✓	✓	✓	-	-	$O(N)$
GraphMAE	SCE	✓	✓	✓	✓	✓	$O(N)$

(a) Technical comparison between generative SSL methods.



(b) The effect of GraphMAE designs on the performance on Cora dataset.

Figure 1: Comparison between generative SSL methods and the effect of GraphMAE design. *AE*: autoencoder methods; *No Struct.*: no structure reconstruction objective; *Mask Feat.*: use masking to corrupt input features; *GNN Decoder*: use GNN as the decoder; *Re-mask Dec.*: re-mask encoder output before fed into decoder; *Space*: run-time memory consumption; *MSE*: Mean Squared Error; *SCE*: Scaled Cosine Error; *CE*: Cross-Entropy Error; *SCE* represents our proposed Scaled Cosine Error.

there have been no GAEs succeeding to achieve a comprehensive outperformance over contrastive SSL methods, especially on node and graph classifications, which have been significantly advanced by neural encoders, e.g., graph neural networks. To bridge the gap, we analyze existing GAEs and identify the issues that may negatively affect the progress of GAEs. Note that though previous GAEs may have individually tackled one or two of these issues below, none of them collectively deals with the four challenges.

First, the structure information may be over-emphasized. Most GAEs leverage link reconstruction as the objective to encourage the topological closeness between neighbors [3, 17, 20, 26, 31, 42]. Thus, prior GAEs are usually good at link prediction and node clustering, but unsatisfactory on node and graph classifications.

Second, feature reconstruction without corruption may not be robust. For GAEs [3, 20, 26, 27, 31] that leverage feature reconstruction, most of them still employ the vanilla architecture that risks learning trivial solutions. However, the denoising autoencoders [41] that corrupt input and then attempt to recover it have been widely adopted in NLP [4], which might be applicable to graphs as well.

Third, the mean square error (MSE) can be sensitive and unstable. To the best of our knowledge, all existing GAEs with feature reconstruction [17, 18, 27, 31, 42] have adopted MSE as the criterion without additional precautions. However, MSE is known to suffer from varied feature vector norms and the curse of dimensionality [5] and thus can cause the collapse of training for autoencoders.

Fourth, the decoder architectures are of little expressiveness. Most GAEs [3, 16–18, 20, 26, 42] leverage MLP as their decoders. However, the targets in language are one-hot vectors containing rich semantics, while in graphs, most of our targets are less informative feature vectors (except for some discrete attributes such as those in chemical graphs). In this case, this trivial decoder (MLP) may not be strong enough to bridge the gap between encoder representations and decoder targets for graph features.

Contributions. In light of the above observations, the goal of this work is to examine to what extent we can 1) mitigate the issues faced by existing GAEs and 2) further enable GAEs to match or outperform contrastive graph learning techniques. To this end, we present a masked graph autoencoder GraphMAE for self-supervised graph representation learning. By identifying the critical components in GAEs, we add new designs and also improve existing strategies for GraphMAE, unleashing the power of autoencoders for graph learning. Figure 1a summarizes the different design choices between GAEs and GraphMAE. Specifically, the performance of GraphMAE largely benefits from the following critical designs (See Figure 1b for their contributions to performance improvements):

Masked feature reconstruction. Different from most GAEs' efforts in structure reconstruction, GraphMAE only focuses on reconstructing features with masking, whose effectiveness has been extensively verified in CV [12] and NLP [4, 30]. Our empirical studies suggest that with a proper error design, masked feature reconstruction can substantially benefit GAEs.

Scaled cosine error. Instead of using MSE as existing GAEs, GraphMAE employs the cosine error, which is beneficial when feature vectors vary in their magnitudes (as is often the case for node attributes in graphs). On top of it, we further introduce a scaled cosine error to tackle the issue of imbalance between easy and hard samples during reconstruction.

Re-mask decoding. We leverage a re-mask decoding strategy that re-masks the encoder's output embeddings of masked nodes before they are fed into the decoder. In addition, GraphMAE proposes to leverage more expressive graph neural nets (GNNs) as its decoder in contrast to previous GAEs' common usage of MLP.

Overall, GraphMAE is a simple generative self-supervised method for graphs without additional cost. We conduct extensive experiments on 21 datasets for three different graph learning tasks, including node classification, graph classification, and transfer learning. The results suggest that equipped with the simple designs above, GraphMAE can generate performance advantages over state-of-the-art contrastive SSL approaches across three tasks.

Moreover, in many cases, GraphMAE can match or sometimes outperform supervised baselines, further demonstrating the potential of self-supervised graph learning, particularly generative methods.

2 RELATED WORK

According to model architectures and objective designs, self-supervised methods on graphs can be naturally divided into contrastive and generative domains.

2.1 Contrastive Self-Supervised Graph Learning

Contrastive self-supervised learning , which encourages alignment between label-invariant distributions and uniformity across other distributions, has been the prevalent paradigm for graph representation learning in the last two years. Its success relies heavily on the elaborate designs of the following components:

Negative sampling. In pursuit of uniformity, negative sampling is a must for most contrastive methods. Mutual information based DGI [40] and InfoGraph [34] leverage corruptions to construct negative pairs. GCC [29] follows the MoCo-style [13] negative queues. GRACE [60], GCA [61] and GraphCL [54] use in-batch negatives. Despite recent attempts for negative-sample-free contrastive learning, strong regularization from architecture designs (e.g., BGRL [37]) or in-batch feature decorrelation (e.g. CCA-SSG [57]) is necessary in their practice.

Architectures. Contrastive methods can be unstable in early-stage training, and thus architecture constraints are important to tackle the challenge. Asymmetric bi-encoder designs including momentum update [29], EMA, and Stop-gradient [37] are widely adopted.

Data augmentation. High-quality and informative data augmentation plays a central role in the success of contrastive learning, including feature-oriented (partial masking [16, 18, 37, 54, 60], shuffling [40]), proximity-oriented (diffusion [11, 19], perturbation [17, 54, 56]), and graph-sampling-based (random-walk [11, 29, 54], uniform [56], ego-network [33]) augmentations.

However, while augmentation in CV is usually human comprehensible, it is difficult to interpret in graphs. Without a theoretical understanding of handcrafted graph augmentation strategies, it remains unverified whether they are label-invariant and optimal.

2.2 Generative Self-Supervised Graph Learning

Generative self-supervised learning aims to recover missing parts of the input data. It can be further classified into autoregressive and autoencoding two families. In previous literature, generative methods' performance on graph representation learning falls behind contrastive methods by a large margin.

Graph autoregressive models. Autoregressive models decompose joint probability distributions as a product of conditionals. In supervised graph generation, previous researchers have proposed GraphRNN [52], GCPN [51]. For graph representation learning, GPT-GNN [17] is a recent attempt to leverage graph generation as the training objective. However, since most graphs do not present inherent orders, autoregressive methods make little sense on them.

Graph autoencoders (GAEs). Autoencoders [14] are designed to reconstruct certain inputs given the contexts and do not enforce any decoding orders as autoregressive methods do. The earliest works trace back to GAE and VGAE [20] which take 2-layer GCN as encoder and dot-product for link prediction decoding. EP [6] proposes to recover vertex features using mean squared error without input corruption. Later GAEs mostly adopt the structural reconstruction (e.g., ARVGA [26]) following VGAE, or a combination of structural and feature reconstruction (e.g., MGAE [42], GALA [27] and GATE [31]) as their objectives. And NWR-GAE [36] jointly predicts the node degree and neighbor feature distribution.

Regardless of the successful applications in link prediction and graph clustering, due to existing GAEs' reconstruction of structure or/and features without masking, their results on node/graph classification benchmarks are usually unsatisfactory. Therefore, our goal in this work is to identify the weaknesses of existing GAE designs and rejuvenate the idea of self-supervised GAEs on graph representation learning for classification.

GraphMAE v.s. Attribute-Masking. Recently, some works [18, 24, 55, 59] have been dedicated to surveying a wide range of many self-supervision objectives' effectiveness on GNNs, including masked feature reconstruction (namely Attribute-Masking). However, their performance lags far behind state-of-the-art contrastive methods because other critical defects of existing GAEs are not handled. We present a rough comparison of algorithms and results between GraphMAE and them in Appendix A.

3 THE GraphMAE APPROACH

In this section, we present the self-supervised masked graph autoencoder framework—GraphMAE—to learn graph representations without supervision based on graph neural networks (GNNs). We introduce the critical components that differ GraphMAE from previous attempts on designing graph autoencoders (GAEs).

3.1 The GAE Problem and GraphMAE

Briefly, an autoencoder usually comprises an encoder, code (hidden states), and a decoder. The encoder maps the input data to code, and the decoder maps the code to reconstruct the input under the supervision of a reconstruction criterion. For graph autoencoders, they can be formalized as follows.

Let $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$ denote a graph, where $\mathcal{V}$ is the node set, $N = |\mathcal{V}|$ is the number of nodes, $\boldsymbol{A} \in \{0,1\}^{N\times N}$ is the adjacency matrix, and $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ is the input node feature matrix. Further, given f_E as the graph encoder, f_D as the graph decoder, and $\boldsymbol{H} \in \mathbb{R}^{N\times d_h}$ denoting the code encoded by the encoder, the goal of general GAEs is to reconstruct the input as

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

where $\mathcal{G}'$ denotes the reconstructed graph, which could be either reconstructed features or structures or both.

Despite their versatile applications in NLP and CV, autoencoders' progress in graphs, especially for classification tasks, is relatively insignificant. To bridge the gap, in this work, we aim to identify and rectify the deficiencies of existing GAE approaches, and subsequently present the GraphMAE—a masked graph autoencoder—to further the idea and design of GAEs and generative SSL in graphs.

GraphMAE. The overall architecture of GraphMAE is illustrated in Figure 2. Its core idea lies in the reconstruction of masked node features. And we introduce a re-mask decoding strategy with GNNs,

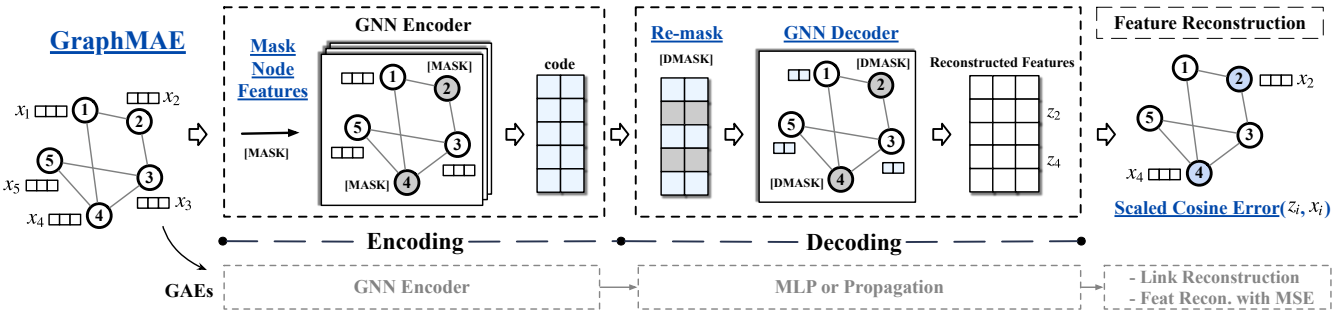


Figure 2: Illustration of GraphMAE and the comparison with GAE. We underline the key operations in GraphMAE. During pre-training, GraphMAE first masks input node features with a mask token [MASK]. The corrupted graph is encoded into code by a GNN encoder. In the decoding, GraphMAE re-masks the code of selected nodes with another token [DMASK], and then employs a GNN, e.g., GAT, GIN, as the decoder. The output of the decoder is used to reconstruct input node features of masked nodes, with the scaled cosine error as the criterion. Previous GAEs usually use a single-layer MLP or Laplacian matrix in the decoding and focus more on restoring graph structure.

rather than the widely-used MLP in GAEs, as the decoder to empower GraphMAE. To have a robust reconstruction, we also propose to use a scaled cosine error as the criterion. Figure 1a summarizes the technical differences between GraphMAE and existing GAEs.

In detail, the backbones for f_E and f_D can be any type of GNNs, such as GCN [21], GAT [39], or GIN [45]. As our encoder f_E processes the whole graph $\boldsymbol{A}$ with partially observed node features $\widetilde{\boldsymbol{X}}$, resonating to the backbones in other generative SSL methods (e.g., BERT and MAE), GraphMAE prefers a more expressive GNN encoder on features for different tasks. For instance, GAT is more expressive in node classification, and GIN provides a better inductive bias for graph-level applications (See Tables 6 and 4).

3.2 The Design of GraphMAE

In this part, we explore how to design a generative self-supervised graph pre-training framework that can match and outperform state-of-the-art contrastive models. Specifically, we discuss different strategies via answering the following four questions:

- **Q1**: What to reconstruct in GAEs?
- **Q2**: How to train robust GAEs to avoid trivial solutions?
- **Q3**: How to arrange the decoder for GAEs?
- **Q4**: What error function to use for reconstruction?

These questions concern the designs of the reconstruction objective, robust learning, loss function, and model architecture in GAEs, the answers to which enable us to develop GraphMAE.

Q1: Feature reconstruction as the objective. Given a graph $\mathcal{G} = (\mathcal{V}, \boldsymbol{A}, \boldsymbol{X})$, a GAE could target reconstructing either the structure $\boldsymbol{A}$ or the features $\boldsymbol{X}$, or both of them. Most classical GAEs [20, 26] focus on the tasks of link prediction and graph clustering, and thus usually choose to reconstruct $\boldsymbol{A}$—a target commonly used in network embeddings [9, 28, 35]. More recent GAEs [27, 31, 42] tend to adopt a combined objective of reconstructing both features and structure, which unfortunately does not empower GAEs to produce as significant progress in node and graph classifications as autoencoders have done in NLP and CV.

A very recent study shows that simple MLPs distilled from trained GNN teachers can work comparably to advanced GNNs on node classification [58], indicating the vital role of features in such tasks. Thus, to enable GraphMAE to achieve a good performance on classification, we adopt feature reconstruction as the training objective. Our empirical examination also shows that the explicit prediction of structural proximity has no contributions to the downstream classification tasks in GraphMAE (See Figure 1b).

Q2: Masked feature reconstruction. When the code's dimension size is larger than input's, the vanilla autoencoders have risks to learn the notorious "identity function"—the trivial solution—that makes the learned code useless [41]. Relatively speaking, it is not a severe problem in CV since the image input is usually high-dimensional. However, in graphs, the node feature dimension size is typically quite small, making it a real challenge to train powerful feature-oriented GAEs. Unfortunately, existing GAEs that incorporate the reconstruction of features as their objective commonly ignore the threat [3, 20, 26, 27, 31].

The denoising autoencoder [41], which corrupts the input data on purpose, is a natural option to eliminate the trivial solution. Actually, the idea of employing masking as the corruption in masked autoencoders has found wide applications in CV [1, 12] and NLP [4]. Inspired by their success, we propose to adopt masked autoencoders as the backbone of GraphMAE.

Formally, we sample a subset of nodes $\widetilde{\mathcal{V}} \subset \mathcal{V}$ and mask each of their features with a mask token [MASK], i.e., a learnable vector $\boldsymbol{x}_{[M]} \in \mathbb{R}^d$. Thus, the node feature $\widetilde{\boldsymbol{x}}_i$ for $v_i \in \mathcal{V}$ in the masked feature matrix $\widetilde{\boldsymbol{X}}$ can be defined as:

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

The objective of GraphMAE is to reconstruct the masked features of nodes in $\widetilde{\mathcal{V}}$ given the partially observed node signals $\widetilde{\boldsymbol{X}}$ and the input adjacency matrix $\boldsymbol{A}$.

We apply a uniform random sampling strategy without replacement to obtain masked nodes. In GNNs, each node relies on its neighbor nodes to enhance/recover its features [7]. Random sampling with a uniform distribution helps prevent a potential bias center, i.e., one's neighbors are neither all masked nor all visible. Additionally, similar to MAE [12], a relatively large mask ratio (e.g.,

50%) is necessary to reduce redundancy in the attributed graphs in most cases and thus form a challenging self-supervision to learn meaningful node representations.

The use of [MASK], on the other hand, potentially creates a mismatch between training and inference since the [MASK] token does not appear during inference [49]. To mitigate the discrepancy, BERT proposes to not always replace "masked" words with the actual [MASK] token, but with a small probability (i.e., 15% or smaller) to leave it unchanged or to substitute it with another random token. Our experiments find that the "leave-unchanged" strategy actually harms GraphMAE's learning, while the "random-substitution" method could help form more high-quality representations.

Q3: GNN decoder with re-mask decoding. The decoder f_D maps the latent code $\boldsymbol{H}$ back to the input $\boldsymbol{X}$, and its design would depend on the semantic level [12] of target $\boldsymbol{X}$. For example, in language, since targets are one-hot missing words with rich semantics, usually a trivial decoder such as MLP is sufficient [4]. But in vision, previous studies [12] discover that a more advanced decoder (e.g., the Transformer model [38]) is necessary to recover pixel patches with low-level semantics.

In graphs, the decoder reconstructs relatively less informative multi-dimensional node features. Traditional GAEs employ either no neural decoders or a simple MLP for decoding with less expressiveness, causing the latent code $\boldsymbol{H}$ to be nearly identical to input features. However, it has no merit to learn such trivial latent representations because the goal is to embed input features with meaningful compressed knowledge. Therefore, GraphMAE resorts to a more expressive single-layer GNN as its decoder. The GNN decoder can recover the input features of one node based on a set of nodes instead of only the node itself, and it consequently helps the encoder learn high-level latent code.

To further encourage the encoder to learn compressed representations, we propose a *re-mask decoding* technique to process the latent code $\boldsymbol{H}$ for decoding. We replace $\boldsymbol{H}$ on masked node indices again with another mask token [DMASK], i.e., the decoder mask, with $\boldsymbol{h}_{[M]} \in \mathbb{R}^{d_h}$. Specifically, the re-masked code $\widetilde{\boldsymbol{h}}_i$ in $\widetilde{\boldsymbol{H}} = \text{REMASK}(\boldsymbol{H})$ can be denoted as

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

With the GNN decoder, a masked node is forced to reconstruct its input feature from the neighboring unmasked latent representations. Similar to encoders, our empirical examination suggests that the GAT and GIN encoders are good options for node classification and graph classification, respectively. Note that the decoder is only used during the self-supervised training stage to perform the node feature reconstruction task. Therefore, the decoder architecture is independent of the encoder choice and can use any type of GNN.

Q4: Scaled cosine error as the criterion. The feature reconstruction criterion varies for masked autoencoders [4, 12] in different domains. In NLP and CV, the *de facto* criterion is to predict discrete token indices derived from tokenizers using cross entropy error. An exception is the MAE work [12] in CV, which directly predicts pixels in the masked patches using the mean square error (MSE); Nevertheless in fact, pixels are naturally normalized to 0–255, functioning similarly to tokenizers. But in graphs, it remains unexplored how to define a universal tokenizer.

In GraphMAE, we propose to directly reconstruct the raw features for each masked node, which can be challenging due to the multi-dimensional and continuous nature of node features. Existing GAEs with feature reconstruction have adopted MSE as their criterion [18, 27, 42]. But in preliminary experiments, we discover that the MSE loss can be minimized to nearly zero and may not be enough for feature reconstruction, which may (partly) explain why few existing GAEs use feature reconstruction as their only training objective. To this end, we found that MSE could suffer from the issues of *sensitivity* and *low selectivity*. Sensitivity means that MSE is sensitive to vector norms and dimensionality [5]. Extreme values in certain feature dimensions can also lead to MSE's overfit on them. Low selectivity represents that MSE is not selective enough to focus on those harder ones among imbalanced easy-and-hard samples.

To handle its sensitivity, we leverage the cosine error as the criterion to reconstruct original node features, which gets rid of the impact of dimensionality and vector norms. The l_2-normalization in the cosine error maps vectors to a unit hyper-sphere and substantially improves the training stability of representation learning. This benefit is also observed by some contrastive learning methods like BYOL [8].

To improve its selectivity, we further the cosine error by introducing the scaled cosine error (SCE) for GraphMAE. The intuition is that we can down-weight easy samples' contribution in training by scaling the cosine error with a power of $\gamma \geq 1$. For predictions with high confidence, their corresponding cosine errors are usually smaller than 1 and decay faster to zero when the scaling factor $\gamma > 1$. Formally speaking, given the original feature $\boldsymbol{X}$ and reconstructed output $\boldsymbol{Z} = f_D(\boldsymbol{A}, \widetilde{\boldsymbol{H}})$, we define SCE for GraphMAE as

$$\mathcal{L}_{\text{SCE}} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v_i \in \widetilde{\mathcal{V}}} \left(1 - \frac{\boldsymbol{x}_i^T \boldsymbol{z}_i}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{z}_i\|}\right)^{\gamma}, \ \gamma \geq 1, \tag{2}$$

which is averaged over all masked nodes. The scaling factor γ is a hyper-parameter adjustable over different datasets. This scaling technique could also be viewed as an adaptive sample reweighing, and the weight of each sample is adjusted with the reconstruction error. This error is also famous in the field of supervised object detection as the focal loss [22].

In summary, GraphMAE is a simple and scalable self-supervised graph learning framework. Figure 1 illustrates how each of its design choices directly impacts the performance of the self-supervised GraphMAE framework. By identifying the negative components and designing new strategies, GraphMAE unleashes the power of autoencoders for self-supervised graph pre-training.

3.3 Training and Inference

The overall training flow of GraphMAE is summarized by Figure 2. First, given an input graph, we randomly select a certain proportion of nodes and replace their node features with the mask-token [MASK]. We feed the graph with partially observed features into the encoder to generate the encoded node representations. In decoding, we re-mask the selected nodes and replace their features with another token [DMASK]. Then the decoder is applied to the

re-masked graph to reconstruct the original node features with the proposed scaled cosine error.

For downstream applications, the encoder is applied to the input graph without any masking in the inference stage. The generated node embeddings can be used for various graph learning tasks, such as node classification and graph classification. For graph-level tasks, we use a non-parameterized graph pooling (readout) function, e.g., MaxPooling and MeanPooling, to obtain the graph-level representation $\boldsymbol{h}^g = \text{READOUT}(\{\boldsymbol{h}_i, v_i \in \mathcal{G}_g\})$. In addition, similar to [16], GraphMAE also enables robust transfer of pre-trained GNN models to various downstream tasks. In the experiments, we show that GraphMAE achieves competitive performance in both node-level and graph-level applications.

4 EXPERIMENTS

In this section, we demonstrate that GraphMAE is a general self-supervised framework for various graph learning tasks, including:

- Unsupervised representation learning for *node classification*;
- Unsupervised representation learning for *graph classification*;
- *Transfer learning* on molecular property prediction.

Extensive experiments on various datasets are conducted to evaluate the performance of GraphMAE against state-of-the-art (SOTA) contrastive and generative methods on these three tasks. In each task, we follow exactly the same experimental procedure, e.g., data splits, evaluation protocol, as the standard settings [16, 34, 40, 57].

4.1 Node Classification

Setup. The node classification task is to predict the unknown node labels in networks. We test the performance of GraphMAE on 6 standard benchmarks: Cora, Citeseer, PubMed [48], Ogbn-arxiv [15], PPI, and Reddit. Following the inductive setup in GraphSage [10], the testing for Reddit and PPI is carried out on unseen nodes and graphs, while the other networks are used for transductive learning.

For the evaluation protocol, we follow the experimental setting in [11, 37, 40, 57]. First, we train a GNN encoder by the proposed GraphMAE without supervision. Then we freeze the parameters of the encoder and generate all the nodes' embeddings. For evaluation, we train a linear classifier and report the mean accuracy on the test nodes through 20 random initializations. We follow the public data splits [11, 40, 57] of Cora, Citeseer, and PubMed. The graph encoder f_E and decoder f_D are both specified as standard GAT. Detailed hyper-parameters can be found in Appendix A.

Results. We compare GraphMAE with SOTA contrastive self-supervised models, DGI [40], MVGRL [11], GRACE [60], BGRL [37], InfoGCL [44], and CCA-SSG [57], as well as supervised baselines GCN and GAT. We also report the results of previous generative self-supervised models, GAE [20], GPT-GNN [17], and GATE [31]. We report results from previous works with the same experimental setup if available. If results are not previously reported and codes are provided, we implement them based on the official codes and conduct a hyper-parameter search. Table 1 lists the results. GraphMAE achieves the best or competitive results compared to the SOTA self-supervised approaches in all benchmarks. Notably, GraphMAE outperforms existing generative methods by a large margin. The results in the inductive setting of PPI and Reddit suggest the self-supervised GraphMAE technique provides strong generalization to unseen nodes.

4.2 Graph Classification

Setup. For graph classification, we conduct experiments on 7 benchmarks: MUTAG, IMDB-B, IMDB-M, PROTEINS, COLLAB, REDDIT-B, and NCI1 [47]. Each dataset is a collection of graphs where each graph is associated with a label. Node labels are used as input features in MUTAG, PROTEINS, and NCI1, whereas node degrees are used in IMDB-B, IMDB-M, REDDIT-B, and COLLAB.

For the evaluation protocol, after generating graph embeddings with GraphMAE's encoder and readout function, we feed encoded graph-level representations into a downstream LIBSVM [2] classifier to predict the label, and report the mean 10-fold cross-validation accuracy with standard deviation after 5 runs. We adopt GIN [45], which is commonly used in previous graph classification works, as the backbone of encoder and decoder.

Results. In addition to classical graph kernel methods—Weisfeiler-Lehman sub-tree kernel (WL) [32] and deep graph kernel (DGK) [47], we also compare GraphMAE with SOTA unsupervised and contrastive methods, GCC [29], graph2vec [25], Infograph [34], GraphCL [54], JOAO [53], MVGRL [11], and InfoGCL [44]. The supervised baselines, GIN [45] and DiffPool [50], are also included. Per graph classification research tradition, we report results from previous papers if available. The results are shown in Table 2. We find that GraphMAE outperforms all self-supervised baselines on five out of seven datasets and has competitive results on the other two. The node features of these seven datasets are all one-hot vectors representing node-labels or degrees, which are considered to be less informative than node features in node classification. The results manifest that generative auto-encoders could learn meaningful information and offer potentials in graph-level tasks.

4.3 Transfer Learning

Setup. To evaluate the transferability of the proposed method, we test the performance on transfer learning on molecular property prediction, following the setting of [16, 53, 54]. The model is first pre-trained in 2 million unlabeled molecules sampled from the ZINC15 [33], and then finetuned in 8 classification benchmark datasets contained in MoleculeNet [43]. The downstream datasets are split by scaffold-split to mimic real-world use cases. Input node features are the atom number and chirality tag, and edge features are the bond type and direction.

For the evaluation protocol, we run experiments for 10 times and report the mean and standard deviation of ROC-AUC scores (%). Following the default setting in [16], we adopt a 5-layer GIN as the encoder and a single-layer GIN as the decoder.

Results. We evaluate GraphMAE against methods including Infomax, AttrMasking and ContextPred [16] , and SOTA contrastive learning methods, GraphCL [54], JOAO [53], and GraphLoG [46]. Table 2 shows that the performance on downstream tasks is comparable to SOTA methods, in which GraphMAE achieves the best average scores and has a small edge over previous best results in two tasks. This demonstrates the robust transferability of GraphMAE.

Table 1: Experiment results in unsupervised representation learning for node classification. We report the Micro-F1 (%) score for PPI and accuracy (%) for the other datasets.

	Dataset	Cora	CiteSeer	PubMed	Ogbn-arxiv	PPI	Reddit
Supervised	GCN	81.5	70.3	79.0	71.74±0.29	75.7±0.1	95.3±0.1
	GAT	83.0±0.7	72.5±0.7	79.0±0.3	72.10±0.13	97.30±0.20	96.0±0.1
Self-supervised	GAE	71.5±0.4	65.8±0.4	72.1±0.5	-	-	-
	GPT-GNN	80.1±1.0	68.4±1.6	76.3±0.8	-	-	-
	GATE	83.2±0.6	71.8±0.8	80.9±0.3	-	-	-
	DGI	82.3±0.6	71.8±0.7	76.8±0.6	70.34±0.16	63.80±0.20	94.0±0.10
	MVGRL	83.5±0.4	73.3±0.5	80.1±0.7	-	-	-
	GRACE[1]	81.9±0.4	71.2±0.5	80.6±0.4	71.51±0.11	69.71±0.17	94.72±0.04
	BGRL[1]	82.7±0.6	71.1±0.8	79.6±0.5	71.64±0.12	73.63±0.16	94.22±0.03
	InfoGCL	83.5±0.3	**73.5±0.4**	79.1±0.2	-	-	-
	CCA-SSG[1]	84.0±0.4	73.1±0.3	81.0±0.4	71.24±0.20	73.34±0.17	95.07±0.02
	GraphMAE	**84.2±0.4**	73.4±0.4	**81.1±0.4**	**71.75±0.17**	**74.50±0.29**	**96.01±0.08**

The results not reported are due to unavailable code or out-of-memory.

[1] Results are from reproducing using authors' official code, as they did not report the results in part of these datasets. The result of PPI is a bit different from what the authors' reported. This is because we train the linear classifier until convergence, rather than for a small fixed number of epochs during evaluation for PPI, using the authors' official code.

Table 2: Experiment results in unsupervised representation learning for graph classification. We report accuracy (%) for all datasets.

	Dataset	IMDB-B	IMDB-M	PROTEINS	COLLAB	MUTAG	REDDIT-B	NCI1
Supervised	GIN	75.1±5.1	52.3±2.8	76.2±2.8	80.2±1.9	89.4±5.6	92.4±2.5	82.7±1.7
	DiffPool	72.6±3.9	-	75.1±3.5	78.9±2.3	85.0±10.3	92.1±2.6	-
Graph Kernels	WL	72.30±3.44	46.95±0.46	72.92±0.56	-	80.72±3.00	68.82±0.41	80.31±0.46
	DGK	66.96±0.56	44.55±0.52	73.30±0.82	-	87.44±2.72	78.04±0.39	80.31±0.46
Self-supervised	graph2vec	71.10±0.54	50.44±0.87	73.30±2.05	-	83.15±9.25	75.78±1.03	73.22±1.81
	Infograph	73.03±0.87	49.69±0.53	74.44±0.31	70.65±1.13	89.01±1.13	82.50±1.42	76.20±1.06
	GraphCL	71.14±0.44	48.58±0.67	74.39±0.45	71.36±1.15	86.80±1.34	89.53±0.84	77.87±0.41
	JOAO	70.21±3.08	49.20±0.77	74.55±0.41	69.50±0.36	87.35±1.02	85.29±1.35	78.07±0.47
	GCC	72.0	49.4	-	78.9	-	**89.8**	-
	MVGRL	74.20±0.70	51.20±0.50	-	-	89.70±1.10	84.50±0.60	-
	InfoGCL	75.10±0.90	51.40±0.80	-	80.00±1.30	**91.20±1.30**	-	80.20±0.60
	GraphMAE	**75.52±0.66**	**51.63±0.52**	**75.30±0.39**	**80.32±0.46**	88.19±1.26	88.01±0.19	**80.40±0.30**

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

Table 3: Experiment results in transfer learning on molecular property prediction benchmarks. The model is first pre-trained on ZINC15 and then finetuned on the following datasets. We report ROC-AUC scores (%).

	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Avg.
No-pretrain	65.5±1.8	74.3±0.5	63.3±1.5	57.2±0.7	58.2±2.8	71.7±2.3	75.4±1.5	70.0±2.5	67.0
ContextPred	64.3±2.8	75.7±0.7	63.9±0.6	60.9±0.6	65.9±3.8	75.8±1.7	77.3±1.0	79.6±1.2	70.4
AttrMasking	64.3±2.8	**76.7±0.4**	**64.2±0.5**	61.0±0.7	71.8±4.1	74.7±1.4	77.2±1.1	79.3±1.6	71.1
Infomax	68.8 ±0.8	75.3 ±0.5	62.7 ±0.4	58.4 ±0.8	69.9±3.0	75.3 ±2.5	76.0 ±0.7	75.9 ±1.6	70.3
GraphCL	69.7±0.7	73.9±0.7	62.4±0.6	60.5±0.9	76.0±2.7	69.8±2.7	**78.5±1.2**	75.4±1.4	70.8
JOAO	70.2±1.0	75.0±0.3	62.9±0.5	60.0±0.8	81.3±2.5	71.7±1.4	76.7±1.2	77.3±0.5	71.9
GraphLoG	**72.5±0.8**	75.7±0.5	63.5±0.7	**61.2±1.1**	76.7±3.3	76.0±1.1	77.8±0.8	**83.5±1.2**	73.4
GraphMAE	72.0±0.6	75.5±0.6	64.1±0.3	60.3±1.1	**82.3±1.2**	**76.3±2.4**	77.2±1.0	83.1±0.9	**73.8**

Table 4: Ablation studies of the decoder type, re-mask and reconstruction criterion on node- and graph-level datasets.

	Dataset	Node-Level			Graph-Level	
		Cora	PubMed	Arxiv	MUTAG	IMDB-B
COMP.	GraphMAE	84.2	81.1	71.75	88.19	75.52
	w/o mask	79.7	77.9	70.97	82.58	74.42
	w/o re-mask	82.7	80.0	71.61	86.29	74.42
	w/ MSE	79.1	73.1	67.44	86.30	74.04
Decoder	MLP	82.2	80.4	71.54	87.16	73.94
	GCN	81.3	79.1	71.59	87.78	74.54
	GIN	81.8	80.2	71.41	88.19	75.52
	GAT	84.2	81.1	71.75	86.27	74.04

To summarize, the self-supervised GraphMAE method achieves competitive performance on node classification, graph classification, and transfer learning across 21 benchmarks. Note that we do not customize a dedicated GraphMAE for each task. The consistent results on the three tasks demonstrate that GraphMAE is an effective and universal self-supervised graph pre-training framework for various applications.

4.4 Ablation Studies

To verify the effects of the main components in GraphMAE, we further conduct several ablation studies. Without loss of generalization, we choose three datasets from node classification and two datasets from graph classification for experiments.

Effect of reconstruction criterion. We study the influence of reconstruction criterion, and Table 4 shows the results of MSE and the SCE loss function. Generally, the input features in node classification lie in continuous or high-dimensional discrete space, containing more discriminative information. The results manifest that SCE has a significant advantage over MSE, with an absolute gain of 1.5% ∼ 8.0%. In graph classification, pre-training with either MSE or SCE improves accuracy. The input features are discrete one-hot encoding in these benchmarks, representing either node degrees or node labels. Reconstructing one-hot encoding with MSE is similar to classification tasks, thus MSE also works. Nevertheless, SCE offers a performance edge (though limited) over MSE.

Figure 3 shows the influence of scaling factor γ. We observe that $\gamma > 1$ offers benefits in most cases, especially in node classification. However, in MUTAG, a larger γ value harms the performance. In our experiments, we notice that the training loss in node classification is much higher than that in graph classification. This further demonstrates that aligning continuous vectors in a unit sphere is more challenging. Therefore, scaling γ brings improvements.

Effect of mask and re-mask. Masking plays an important role in the proposed GraphMAE method. GraphMAE employs two masking strategies — masking input feature before encoder, and re-masking encoded code before decoder. Table 4 studies the designs. We observe a significant drop in performance if not masking input features, indicating that masking inputs is vital to avoid the trivial solution. For the re-mask strategy, the accuracy drops by 0.1%∼1.9% without it. Re-mask is designed for the GNN decoder and can be regarded as regularization, which makes the self-supervised task more challenging.

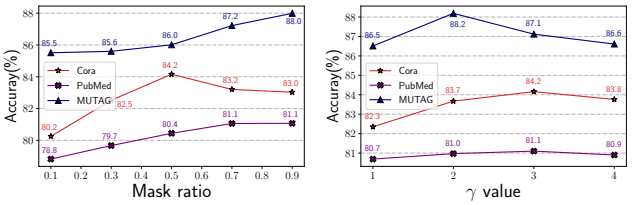


Figure 3: Ablation studies of mask ratio and scaling factor.

Effect of mask ratio. Figure 3 shows the influence of mask ratio. In most cases, the reconstruction task with a low mask ratio (0.1) is not challenging enough to learn useful features. The optimal ratio varies across graphs. Results on Ogbn-arxiv and IMDB-B can be found in Appendix A. In Cora, increasing the mask ratio by more than 0.5 degrades the performance, while GraphMAE still works with a surprisingly high ratio (0.7∼0.9) in PubMed and MUTAG. This can be connected with the information redundancy in graphs. Large node degrees or high homogeneity may lead to heavy information redundancy, in which missing node features may be recovered from very few neighboring nodes with little high-level understanding of features and local context. In contrast, lower redundancy means that an excessively high mask ratio would make it impossible to recover features, thus degrading the performance.

Effect of decoder type. In Table 4, we compare different decoder types, including MLP, GCN, GIN, and GAT. The re-mask strategy is only used for GNN decoders. As the results show, using GNN decoder typically boosts the performance. Compared to MLP, which reconstructs original features from latent representations, GNN enforces masked nodes to extract information relevant to their original features from the neighborhood. One reasonable assumption is that GNN avoids the representation tending to be the same as original features. MLP also works in GraphMAE, which might be partly attribute to the usage of the SCE metric.

Among different GNNs, GIN performs better in graph-level tasks and GAT is a more reasonable option for node classification. It is observed that replacing GAT with GCN causes a significant drop, especially in Cora (∼2.9%) and PubMed (∼2.0%). We speculate that the attention mechanism matters in reconstructing continuous features with the re-mask strategy.

5 CONCLUSION

In this work, we explore generative self-supervised learning in graphs and identify the common issues that are faced by graph autoencoders. We present GraphMAE—a simple masked graph autoencoder—to address them from the perspective of reconstruction objective, learning, loss function, and model architecture. In GraphMAE, we design the masked feature reconstruction strategy with a scaled cosine error as the reconstruction criterion. We conduct extensive experiments on a wide range of node and graph classification benchmarks, and the results demonstrate the effectiveness and generalizability of GraphMAE. Our work shows that generative SSL can offer great potential to graph representation learning and pre-training, requiring more in-depth explorations for future work.

Acknowledgements. This work is supported by Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2020AAA0108400 and 2020AAA0108402, Natural Science Foundation of China (Key Program, No. 61836013), and National Science Foundation for Distinguished Young Scholars (No. 61825602).

REFERENCES

[1] Hangbo Bao, Li Dong, and Furu Wei. 2021. BEiT: BERT Pre-Training of Image Transformers. In *NeurIPS*.
[2] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *TIST* (2011), 1–27.
[3] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. 2020. Adaptive graph encoder for attributed graph embedding. In *SIGKDD*.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.
[5] Jerome H. Friedman. 2004. On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery* 1 (2004), 55–77.
[6] Alberto Garcia Duran and Mathias Niepert. 2017. Learning graph representations with embedding propagation. In *NeurIPS*.
[7] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.
[8] 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*.
[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.
[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
[11] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *ICML*.
[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377* (2021).
[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.
[14] Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length, and Helmholtz free energy. In *NeurIPS*.
[15] 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*.
[16] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. 2020. Strategies For Pre-training Graph Neural Networks. In *ICLR*.
[17] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *SIGKDD*.
[18] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. 2020. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141* (2020).
[19] Zekarias T Kefato and Sarunas Girdzijauskas. 2021. Self-supervised Graph Neural Networks without explicit negative sampling. In *WWW*.
[20] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV*.
[23] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *TKDE* (2021).
[24] Franco Manessi and Alessandro Rozza. 2021. Graph-based neural network models with multiple self-supervised auxiliary tasks. *Pattern Recognition Letters* (2021).
[25] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).
[26] S Pan, R Hu, G Long, J Jiang, L Yao, and C Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*.
[27] 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*.
[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.
[29] 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*.
[30] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).
[31] Amin Salehi and Hasan Davulcu. 2020. Graph Attention Auto-Encoders. In *ICTAI*. IEEE.
[32] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-Lehman graph kernels. *JMLR* (2011).
[33] Teague Sterling and John J Irwin. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling* (2015), 2324–2337.
[34] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.
[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[36] Mingyue Tang, Carl Yang, and Pan Li. 2022. Graph Auto-Encoder via Neighborhood Wasserstein Reconstruction. In *ICLR*.
[37] 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*.
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[40] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In *ICLR*.
[41] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.
[42] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*.
[43] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* (2018), 513–530.
[44] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. 2021. InfoGCL: Information-Aware Graph Contrastive Learning. *NeurIPS*.
[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *ICLR*.
[46] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. 2021. Self-supervised graph-level representation learning with local and global structure. In *ICML*.
[47] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *KDD*.
[48] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.
[49] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *NeurIPS* 32 (2019).
[50] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*.
[51] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. 2018. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS*.
[52] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*. PMLR, 5708–5717.
[53] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. In *ICML*.
[54] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *NeurIPS*.
[55] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2020. When does self-supervision help graph convolutional networks?. In *ICML*.
[56] Jiaqi Zeng and Pengtao Xie. 2021. Contrastive self-supervised learning for graph classification. In *AAAI*.
[57] 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*.
[58] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2022. Graph-less Neural Networks: Teaching Old MLPs New Tricks Via Distillation. In *ICLR*.
[59] Qikui Zhu, Bo Du, and Pingkun Yan. 2020. Self-supervised training of graph convolutional networks. *arXiv preprint arXiv:2006.02380* (2020).
[60] 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).
[61] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*.

A APPENDIX

Table 5: Comparison with other attributed-masking methods in node classification. "FT" means finetuning the model in downstream tasks, while "LP" represents training a linear classifier for classification.

	Cora	Citeseer	PubMed	Type
GAE	71.5	65.8	72.1	LP
GPT-GNN	80.1	68.4	76.3	LP
NodeProp	81.94	71.60	79.44	FT
RASSL-GCN[1]	83.80	72.95	*81.23	FT
GATE	83.2	71.8	80.9	LP
GraphMAE	84.16	73.35	81.10	LP

[1] PubMed dataset used is different from that in other baselines.

A.1 Results in the PPI Dataset

Figure 4 shows the results of self-supervised learning methods in PPI dataset. Previous studies often report a large gap between supervised and self-supervised results on PPI. We find that increasing the number of model parameters helps little in supervised setting, but could highly boost the performance in self-supervised setting. As the hidden size reaches 2048×4, GraphMAE even outperforms supervised counter-part, although the model would be too much larger. At least, this indicates that the gap could be filled.

The results of BGRL and GRACE in PPI are a bit different from those reported in [37]. This is because we train the linear classifier until convergence during evaluation, using the authors' official code. In [37], the classifier is trained only for a small fixed number of epochs, and the training does not converge.

Table 6: Experiment results using different encoder backbones in node classification.

	Cora	Citeseer	Pubmed	Ogbn-arxiv
BGRL (GCN)	82.7±0.6	71.1±0.7	79.4±0.6	71.64±0.12
BGRL (GAT)	82.8±0.5	71.1±0.8	79.6±0.5	70.07±0.02
CCA-SSG (GCN)	84.0±0.4	73.1±0.3	81.0±0.4	70.81±0.13
CCA-SSG (GAT)	83.8±0.5	72.6±0.7	79.9±1.1	71.24±0.20
GraphMAE (GCN)	82.9±0.6	72.5±0.5	81.0±0.5	**71.87±0.21**
GraphMAE (GAT)	**84.2±0.4**	**73.4±0.4**	**81.1±0.4**	71.75±0.17

A.2 Ablation on the Encoder Architecture

In node classification, GraphMAE uses GAT as the encoder. To have a fair comparison and investigate the influence of different GNN backbones, we compare the best baselines, BGRL and CCA-SSG, in node classification datasets using GCN and GAT as the encoder. The results are shown in Table 6. We observe that GraphMAE still outperforms the baselines with the same GAT backbone. In addition, the results manifest that attention mechanism would not always benefit graph self-supervised learning, as GCN is inferior to GAT, for instance, in (Cora, CCA-SSG) and (Ogbn-arxiv, GraphMAE). Under the training setting of GraphMAE, GAT could be a better option in most cases.

A.3 Implementation Details

A.3.1 Environment. Most experiments are conducted on Linux servers equipped with an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, 256GB RAM and NVIDIA 2080Ti GPUs. Experiments of Ogbn-arxiv and Reddit for node classificatin are conducted on Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz and NVIDIA 3090 GPUs, as they require large memory. Models of node and graph classification are implemented in PyTorch version 1.9.0, DGL version 0.7.2 (*https://www.dgl.ai/*) with CUDA version 10.2, scikit-learn version 0.24.1 and Python 3.7. For molecular property prediction, we implement our model based on the code in *https://github.com/snap-stanford/pretrain-gnns* with Pytorch Geometric 2.0.4 (*https://www.pyg.org/*).

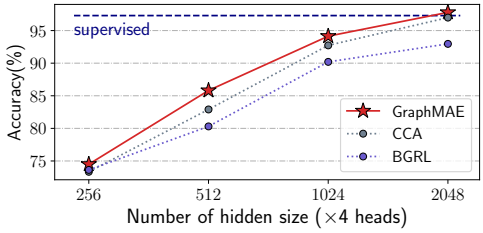


Figure 4: Performance on PPI using GAT with 4 attention heads, compared to other baselines. Self-supervised methods benefit much from larger model size, and GraphMAE could outperform supervised model.

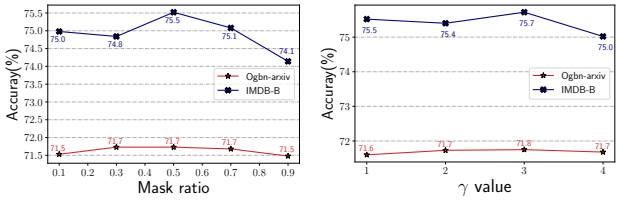


Figure 5: Ablation study of mask ratio and scaling factor γ in Ogbn-arxiv and IMDB-B.

A.3.2 Model Configuration. For node classification, we train the model using Adam Optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1\times10^{-8}$. The initial learning rate is set to 0.001, using cosine learning rate decay without warmup. We use PReLU as the non-linear activation. More details about hyper-parameters and datasets are in Table 7 and https://github.com/THUDM/GraphMAE.

For graph classification, we set the initial learning rate to 0.00015 with cosine learning rate decay for most cases. For the evaluation, the parameter C of SVM is searched in the sets $\{10^{-3}, ..., 10\}$. The hyper-parameters and statistics of datasets are in Table 8.

For transfer learning of molecule property prediction, we adopt a single-layer GIN as decoder, the mask rate is set to 0.25, and we

Table 7: Statistics and hyper-parameters for node classification datasets. "s" represents multi-class classification, and "m" means multi-label classification.

	Dataset	Cora	Citeseer	PubMed	Ogbn-arxiv	PPI	Reddit
Statistics	# nodes	2,708	3,327	19,717	169,343	56,944	232,965
	# edges	5,429	4,732	44,338	1,166,243	818,736	11,606,919
	# classes	7(s)	6(s)	3(s)	40(s)	121(m)	41(s)
Hyper-parameters	scaling factor γ	3	1	3	3	3	3
	masking rate	0.5	0.5	0.75	0.5	0.5	0.75
	replacing rate	0.05	0.10	0	0	0	0.15
	hidden_size	512	512	1024	1024	1024	512
	weight_decay	2e-4	2e-5	1e-5	0	0	2e-4
	max_epoch	1500	300	1000	1000	1000	500

Table 8: Statistics and hyper-parameters for graph classification datasets.

	Dataset	IMDB-B	IMDB-M	PROTEINS	COLLAB	MUTAG	REDDIT-B	NCI1
Statistics	# graphs	1,000	1,500	1,113	5,000	188	2,000	4,110
	# classes	2	3	2	3	2	2	2
	Avg. # nodes	19.8	13.0	39.1	74.5	17.9	429.7	29.8
Hyper-parameters	Scaling factor γ	1	1	1	1	2	1	2
	masking rate	0.5	0.5	0.5	0.75	0.75	0.75	0.25
	replacing rate	0.0	0.0	0.0	0.0	0.1	0.1	0.1
	hidden_size	512	512	512	512	32	512	512
	weight_decay	0	0	0	0	0.0	0.0	0
	max_epoch	60	50	100	20	20	100	300
	batch_size	32	32	32	32	64	8	16
	Pooling	mean	mean	max	max	sum	max	sum

Table 9: Statistics of datasets for molecular property prediction. "ZINC" is used for pre-training.

	ZINC	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE
# graphs	2,000,000	2,039	7,831	8,576	1,427	1,477	93,087	41,127	1,513
# binary prediction tasks	-	1	12	617	27	2	17	1	1
Avg. # nodes	26.6	24.1	18.6	18.8	33.6	26.2	24.2	24.5	34.1

pretrain the whole model for 100 epochs. For the finetuning, an Adam optimizer (learning rate: 0.001, batch size: 32) is employed to train the model for 100 epochs. We utilize a learning rate scheduler with fix step size, which multiplies the learning rate by 0.3 every 30 epochs for BACE only. Table 9 shows the statistics of datasets.

A.4 Baselines

For node classification, the results of supervised baselines of GCN in Reddit, and GAT in Ogbn-arxiv and Reddit are from CogDL(*https://cogdl.ai/*) if not reported before. For unsupervised baselines, GRACE [60], BGRL [37], CCA-SSG [57] are state-of-the-art contrastive learning methods in graph. GRACE and BGRL did not report the results in Cora, Citeseer, and PubMed of the public split. To have a fair comparison, we download the public source code and use the same GNN backbone as GraphMAE. We conduct hyper-parameter search for them and select the best results on the validation set. The results of CCA-SSG are the output of the official code after the bugs in the code are fixed. For MVGRL [11], we adopt DGL's reproducing results. We implement GPT-GNN [17] based on the official code for homogeneous networks, as the code is for heterogeneous networks, and report the results in Cora, Citeseer, and PubMed.

For graph classification and transfer learning of molecular property prediction, we adopt the results reported in previous papers if available. For the results of GraphCL [54] and JOAO [53] in IMDB-MULTI, we download the authors' official codes and keep hyper-parameters the same to get the output.