Adaptive and Context-rich Generative Self-supervised Learning on Graphs

Yijun Tian[1], **Chuxu Zhang**[2], **Ziyi Kou**[1], **Zheyuan Liu**[1],
Xiangliang Zhang[1], **Nitesh V. Chawla**[1]

[1]University of Notre Dame
[2]University of Connecticut
meetyijun@gmail.com, chuxu.zhang@uconn.edu, zkou@alumni.nd.edu, {zliu29, xzhang33, nchawla}@nd.edu

Abstract

Generative self-supervised learning on graphs has emerged as a popular learning paradigm and demonstrated its efficacy in handling non-Euclidean data. However, several remaining issues limit the capability of existing methods: 1) the disregard of uneven node significance in masking, 2) the underutilization of holistic graph information, 3) the ignorance of semantic knowledge in the representation space due to the exclusive use of reconstruction loss in the output space, and 4) the unstable reconstructions caused by the large volume of masked contents. In light of this, we propose ACE-GSL, an adaptive and context-rich graph self-supervised learning framework to address these issues from the perspectives of adaptivity, integrity, complementarity, and consistency. Specifically, we first develop an adaptive feature mask generator to account for the unique significance of nodes and sample informative masks (**adaptivity**). We then design a ranking-based structure reconstruction objective joint with feature reconstruction to capture holistic graph information and emphasize the topological proximity between neighbors (**integrity**). After that, we present a bootstrapping-based similarity module to encode the high-level semantic knowledge in the representation space, complementary to the low-level reconstruction in the output space (**complementarity**). Finally, we build a consistency assurance module to provide reconstruction objectives with extra stabilized consistency targets (**consistency**). Extensive experiments demonstrate that ACE-GSL achieves state-of-the-art performance over 28 methods on 20 datasets across 3 tasks.

Introduction

Although Graph Neural Networks (GNNs) have demonstrated exceptional effectiveness in handling graph data and have achieved great success in a wide range of graph mining tasks, many of them adhere to the supervised or semi-supervised learning settings, where labels are required to guide the learning procedure (Qiao et al. 2024; Wang, Jin, and Derr 2022). To alleviate the strict requirements on labels, self-supervised learning on graphs is proposed and has emerged as one of the most exciting learning paradigms for GNNs (Liu, Xia, and Huang 2024; Yuan et al. 2024; Wu et al. 2023). The key insight behind self-supervised learning on graphs is to learn node or graph representations based on supervision signals derived from the data itself without relying on human-annotated labels (Wang et al. 2024b; He et al. 2025). The learned representations can then be utilized in various downstream applications and tasks.

As for self-supervised learning on graphs, contrastive-based methods are one of the most prevalent and widely used approaches (Qiu et al. 2020; You et al. 2020). Since the insights behind contrastive learning are to maximize the agreement between contrastive data samples, the success of contrastive methods relies heavily on the derivation of these samples (Zhu et al. 2021). However, due to the fact that the majority of constructed data samples are based on heuristics and prior knowledge, model performance can vary between different tasks and data (You et al. 2021; Trivedi et al. 2022). In addition, negative sampling, as a common strategy to obtain negative data samples for most of the contrastive objectives, often entails arduous designs and cumbersome constructions from graphs (Velickovic et al. 2019; Zhu et al. 2020). Therefore, deriving high-quality contrastive data samples becomes a crucial aspect of contrastive method designs.

Generative-based self-supervised learning methods naturally circumvent the aforementioned issues of building contrastive data samples by directly reconstructing the input graph data (Kipf and Welling 2016; Liu et al. 2020; Wu et al. 2021; Tian et al. 2023b). In particular, masked autoencoders are introduced and achieved outstanding performance by eliminating a significant percentage of the input data and using the eliminated data to guide training (Tan et al. 2023; Tian et al. 2023a; Wang et al. 2024a). However, existing works have several limitations that hinder their capability in fully modeling the graph data and learning strong representations.

First, existing methods disregard the uneven node significance in masking. The random sampling of the masked nodes has shown to work, but since not all nodes contain the same amount of information, assuming a uniform probability distribution over all input nodes is sub-optimal. The second limitation is the under-utilization of holistic graph information. Graphs are non-euclidean data with complex structural information and features. To learn effective node representations that fully encode the graph content and topology, merely reconstructing the edges or features is far from sufficient. In addition, current methods ignore the semantic knowledge in the representation space due to the exclusive use of reconstruction loss in the output space. Since the reconstruction loss focuses on the target in the output space while only the

learned representations are utilized in the downstream tasks, a gap exists between the learning objective and what would be used ultimately. Moreover, masking a large portion of input data introduces implicit uncertainties into the model and causes unstable reconstructions, resulting in unsatisfactory model performance.

To address the above challenges, we propose ACE-GSL, an adaptive and context-rich graph self-supervised learning framework from the following four perspectives: adaptivity, integrity, complementarity, and consistency. Specifically, we first develop an adaptive feature mask generator to incorporate the node significance in masking. Compared to random masking, the designed adaptive masking generator samples more nodes that contain rich information and are hard to reconstruct. We then design a ranking-based structure reconstruction objective with the feature reconstruction loss to fully encode integral graph knowledge and emphasize the topological proximity between neighbor nodes. After that, a bootstrapping-based similarity module is proposed to complementarily integrate the high-level semantics from the latent space into the reconstruction in the output space. Finally, we build a consistency assurance module to promote stable learning by providing the original objectives with additional consistency labels. To fully evaluate our model, we conduct extensive experiments on three graph learning tasks. From the evaluation results, we conclude that ACE-GSL can effectively address the aforementioned challenges and achieve state-of-the-art performance. To summarize, the contributions of this paper are as follows:

- We point out that existing methods suffer from four limitations that undermine their capability: the disregard of uneven node significance in masking, the under-utilization of holistic graph information, the ignorance of semantic knowledge, and the unstable reconstructions caused by the large volume of masked contents.
- To address these limitations, we propose ACE-GSL, a novel generative self-supervised learning framework on graphs. ACE-GSL develops multiple novel components to tackle the above four issues from the perspectives of adaptivity, integrity, complementarity, and consistency.
- Extensive experiments demonstrate that ACE-GSL achieves state-of-the-art performance over 28 methods across on 20 datasets 3 graph learning tasks.

Related Work

This work is closely related to generative and contrastive self-supervised learning on graphs.

Generative Self-supervised Learning on Graphs

The objective of generative self-supervised learning on graphs is to reconstruct missing input data. In prior studies, the performance of generative methods is inferior to that of contrastive methods (Kipf and Welling 2016; Park et al. 2019; Garcia Duran and Niepert 2017). For example, the earliest works VGAE and GAE (Kipf and Welling 2016) leverage a graph convolutional network as the encoder and use a dot product operator as the decoder. Later, GraphMAE (Hou et al. 2022) was proposed and has demonstrated great performance, which introduces masked autoencoder (He et al. 2022) in the graph domain by reconstructing node features, achieving outstanding performance. After that, many works are proposed to solve graph learning tasks following the pipeline of masked autoencoder (Hou et al. 2023; Wang et al. 2024a; Tan et al. 2023). For instance, AUG-MAE (Wang et al. 2024a) investigates the relationships between GAE and contrastive learning. However, existing methods disregard several challenges such as the ignorance of semantic knowledge in the representation space and the unstable reconstructions in the output space, which we address in this paper.

Contrastive Self-supervised Learning on Graphs

Contrastive learning, which encourages alignment between different augmentations or distributions, has achieved great success in the graph domain and has become a widely recognized paradigm for graph representation learning (Xu et al. 2021a; Qiu et al. 2020; You et al. 2020; Tian et al. 2024; Wan et al. 2024; Zhang et al. 2023; Tian et al. 2022; Zhang et al. 2022; Shi, Zhou, and Liu 2023). For example, GCC (Qiu et al. 2020) aligns the local structures of two sampled subgraphs. GraphCL (You et al. 2020) focuses on aligning different graph augmentations. AD-GCL (Suresh et al. 2021) introduces the alignment between the original graph and an adversarial augmented graph. However, the success of most contrastive methods relies on the choice of contrastive data samples (You et al. 2020; Zhu et al. 2021), where negative sampling is usually utilized. For instance, InfoGraph (Sun et al. 2019) and DGI (Velickovic et al. 2019) use corruptions to obtain negative pairs. GCC (Qiu et al. 2020) utilizes negative queues that are proposed in MoCo (He et al. 2020). GraphCL (You et al. 2020), GRACE (Zhu et al. 2020), and GCA (Zhu et al. 2021) leverage data samples within the same batch as negatives. Since the selection of different contrastive data samples can affect the performance greatly (You et al. 2021), the determination and construction of suitable contrastive samples play an important role in contrastive method designs. In this paper, we focus on the generative methods, which are more recent than the contrastive methods.

Method

In this section, we formally present ACE-GSL to address the limitations. ACE-GSL contains four innovative components.

Adaptive Feature Mask Generator

In order to account for the unique significance of nodes in masking, we design the adaptive feature mask generator to obtain informative masks, as shown in Figure 1 (b). By determining and assigning high probability values for node features with high reconstruction errors, the generator enables the model to focus on reconstructing nodes with hard features and rich information. In contrast to random sampling, we first utilize an auxiliary sampling network to estimate the categorical distribution for all nodes, and then obtain the masks based on the distribution. Given that sampling is a non-differentiable operation, we present a REINFORCE-based method to optimize the sampling network. In particular, we define the input graph as $G = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V}$ represents

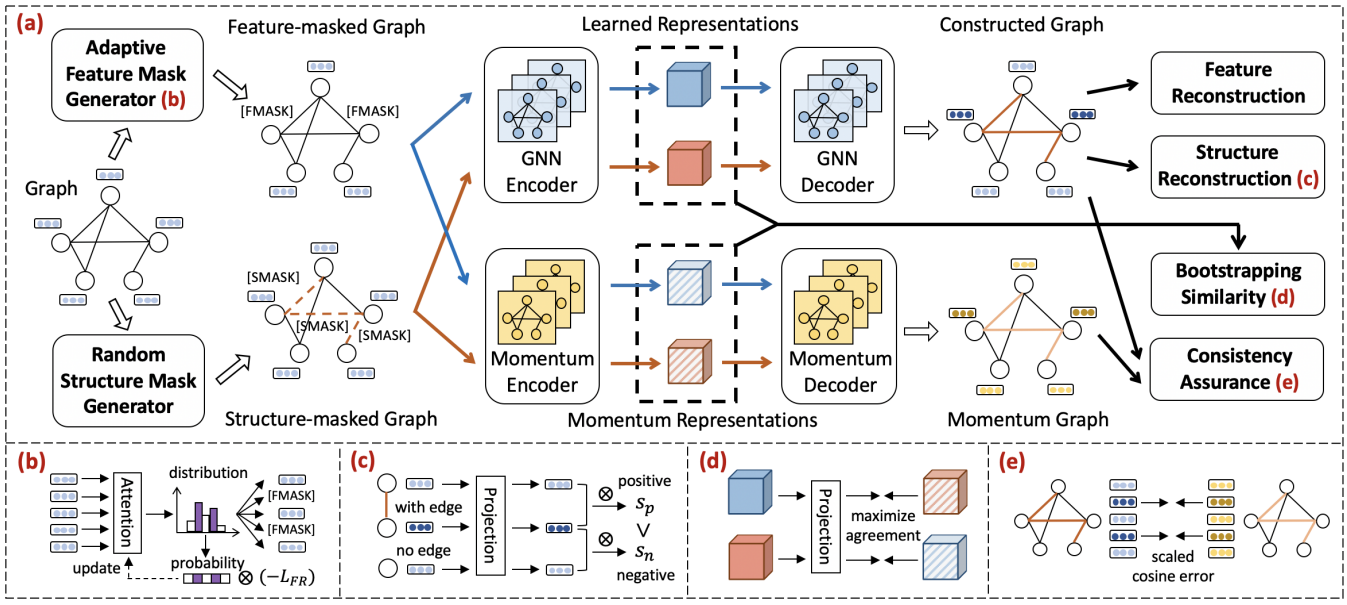

Figure 1: (a) The overall framework of ACE-GSL: we first design mask generators to obtain masked graphs and then send them into the encoder/decoder pipeline to learn representations. Then, the learned representations are taken for the reconstruction and consistency assurance objectives in the output space. In addition, we calculate bootstrapping similarity in the representation space to capture the high-level semantic knowledge. (b) Adaptive feature mask generator: assigning high probability values for features with high reconstruction errors. (c) Ranking-based structure reconstruction: encouraging the connected nodes to be relatively similar. (d) Bootstrapping-based similarity: maximizing the agreement between learned and momentum representations. (e) Consistence assurance: minimizing the scaled cosine error between the constructed and momentum graph.

the node set, $\mathcal{E}$ indicates the edge set, and X denotes the node features. To calculate and enable the model to consider the node significance, we first use a multi-head attention network (MHA) followed by a simple feed-forward neural network (FFN) and a Softmax activation to calculate the probability scores $P = \text{Softmax}\left(\text{FFN}[\text{MHA}(X)]\right)$.

We then define a categorical distribution over P and draw the set of masked nodes $\widetilde{\mathcal{V}}$ without replacement, where $\widetilde{\mathcal{V}} \in \mathcal{V}$. The number of nodes in $\widetilde{\mathcal{V}}$ is determined by the feature mask rate p_f, where $|\widetilde{\mathcal{V}}| = |\mathcal{V}| \times p_f$. After that, We design a learnable mask token [FMASK] to mask the node features. For each node $v \in \widetilde{\mathcal{V}}$, we have its mask feature $X_{[\text{FMASK}]} \in \widetilde{X}$, where $\widetilde{X}$ denotes the masked feature matrix. Correspondingly, for each $\widetilde{X}^v \in \widetilde{X}$, the process is formulated as follows:

$$\widetilde{X}^v = \begin{cases} X_{[\text{FMASK}]} & \text{if } v \in \widetilde{\mathcal{V}} \\ X^v & \text{if } v \notin \widetilde{\mathcal{V}}. \end{cases} \tag{1}$$

Feature Reconstruction. After we acquire the masked feature matrix $\widetilde{X}$, we denote the feature-masked graph as $\widetilde{G}_f$ where $\widetilde{G}_f = (\mathcal{V}, \mathcal{E}, \widetilde{X})$, which is later send into the encoder f_E to obtain the learned node representations $H_1 = f_E(\mathcal{E}, \widetilde{X})$. To encourage the encoder to learn informative representations without relying on the decoder's capability for reconstruction, we apply another mask token $[DM]$ on H_1 to obtain $\widetilde{H}_1$ before sending it into the decoder, where $[DM]$ holds the same masked node indices as $[FMASK]$.

Next, we send $\mathcal{E}$ and $\widetilde{H}_1$ into the decoder f_D to obtain the reconstructed node features Z_1 under feature masking. The process is as follows:

$$\widetilde{H}_1 = \begin{cases} h_{[DM]} & \text{if } v \in \widetilde{\mathcal{V}} \\ h^v & \text{if } v \notin \widetilde{\mathcal{V}}, \end{cases} \quad Z_1 = f_D(\mathcal{E}, \widetilde{H}_1). \tag{2}$$

Subsequently, for each node $v \in \widetilde{\mathcal{V}}$, we compute the loss between the reconstruction $Z_1^v \in Z_1$ and the ground-truth node features $X^v \in X$ with a scaling factor α. In particular, we define the feature reconstruction loss $\mathcal{L}_{\text{FR}}$ as follows:

$$\mathcal{L}_{\text{FR}} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v \in \widetilde{\mathcal{V}}} \left(1 - \frac{X^v \cdot Z_1^v}{\|X^v\| \times \|Z_1^v\|}\right)^{\alpha}. \tag{3}$$

Optimizing Adaptive Feature Mask Generator. Since the sampling process is non-differentiable, to optimize the adaptive feature mask generator, we present a REINFORCE-based method. Specifically, we introduce a sampling loss $\mathcal{L}_{\text{sample}}$, in which we consider the sampling process as *action*, both the encoder and the decoder as *environment*, and the feature reconstruction loss $\mathcal{L}_{\text{FR}}$ as *return*. To encourage the generator to focus on nodes that are hard to reconstruct, we optimize the generator by maximizing the expected feature reconstruction error $\mathbb{E}[\mathcal{L}_{FR}]$: $\mathcal{L}_{\text{sample}} = -\mathbb{E}[\mathcal{L}_{\text{FR}}] = -\sum_{v \in \widetilde{\mathcal{V}}} log(P^v) \cdot \mathcal{L}_{\text{FR}}^v$, where P^v is the probability score for node v and $\mathcal{L}_{\text{FR}}^v$ is the feature reconstruction error for node v. To prevent precision errors caused by small values, we take the logarithm for P^v. In addition, we stop the gradient updates from $\mathcal{L}_{\text{sample}}$ to

propagate through the encoder and decoder with the aim of avoiding duplicate computation.

Ranking-based Structure Reconstruction

We also design a structure reconstruction objective to capture holistic graph knowledge and emphasize the topological proximity between neighbors, as shown in Figure 1 (c). Since reconstructing the edges via a strict binary classification loss might force the model to focus on the explicit structures while ignoring the implicit relevance between nodes, we design a ranking-based objective to incorporate the relative node similarities and distances. The objective involves comparing the preference scores between nodes connected by an edge against the scores between nodes without any connections. To demonstrate, we start by introducing a random structure mask generator to obtain structure mask $[SMASK]$. Specifically, we randomly sample a subset of masked edges $\mathcal{E}_{mask} \in \mathcal{E}$ following Bernoulli distribution, i.e., $\mathcal{E}_{mask} \sim \text{Bernoulli}(p_s)$, where p_s denotes the structure mask rate. For edges in $\mathcal{E}_{mask}$, we mask them with $[SMASK]$, deriving the structure-masked graph $\widetilde{G}_s = (\mathcal{V}, \widetilde{\mathcal{E}}, X)$, where $\widetilde{\mathcal{E}}$ represents the remained visible edges after masking with $\widetilde{\mathcal{E}} = \mathcal{E} - \mathcal{E}_{mask}$. Then, we feed $\widetilde{\mathcal{E}}$ and node features X into the encoder f_E to obtain the learned node representations H_2 under structure masking. After that, we send $\widetilde{\mathcal{E}}$ and H_2 into the decoder f_D to generate the reconstructed node features Z_2 under structure masking. The process is formulated as follows:

$$H_2 = f_E(\widetilde{\mathcal{E}}, X), \quad Z_2 = f_D(\widetilde{\mathcal{E}}, H_2). \tag{4}$$

After we obtain Z_2, we can leverage it to calculate the preference scores between nodes, and further use the scores to define the structure reconstruction objective. To illustrate, given two connected nodes v_i and v_j, we encourage the inner product similarity between v_i and v_j to be higher than the inner product similarity between v_i and a random negative node $v_{j'}$. Accordingly, we formulate the structure reconstruction loss $\mathcal{L}_{\text{SR}}$ as follows:

$$\mathcal{L}_{\text{SR}} = \sum_{(v_i, v_j) \in \widetilde{\mathcal{E}}} \max\{0, 1 - \text{sim}(Z_2^i, Z_2^j) + \text{sim}(Z_2^i, Z_2^{j'})\}, \tag{5}$$

where $Z_2^i, Z_2^j, Z_2^{j'}$ are learned representations under structure masking for $v_i, v_j, v_{j'}$, respectively.

Bootstrapping-based Similarity

We propose a bootstrapping-based similarity module (Figure 1 (d)) to capture high-level semantic knowledge, which enables the model to iteratively bootstrap the prior encoder outputs as learning targets and facilitate enhanced representations. In particular, we refine the learned representations via a bootstrapping procedure by predicting the output of a momentum encoder, i.e., predicting the momentum representations that are generated by a slowly moving exponential average of the original encoder. We start by introducing a momentum encoder f_{E^*} with the same model structure as the original encoder f_E. We set the parameters for f_{E^*} as an updated exponential moving average of f_E. We then feed the feature-masked graph $\widetilde{G}_f = (\mathcal{V}, \mathcal{E}, \widetilde{X})$ and the structure-masked graph $\widetilde{G}_s = (\mathcal{V}, \widetilde{\mathcal{E}}, X)$ into f_{E^*} to obtain the momentum representations H_1^* and H_2^*, repestively. The process is formulated as follows:

$$H_1^* = f_{E^*}(\mathcal{E}, \widetilde{X}), \quad H_2^* = f_{E^*}(\widetilde{\mathcal{E}}, X). \tag{6}$$

As momentum representations, H_1^* and H_2^* provide dynamically deeper semantics by considering prior knowledge via bootstrapping. To enable better learning, we transform the learned representations H_1 and H_2 via a shared simple projection network proj (e.g., MLP):

$$H_1' = \text{proj}(H_1), \quad H_2' = \text{proj}(H_2), \tag{7}$$

where H_1', H_2' are the transformed representations of H_1, H_2, respectively. Next, we encourage the cross-masking similarity by making H_1' on $\widetilde{G}_f$ closer to H_2^* on $\widetilde{G}_s$, and H_2' on $\widetilde{G}_s$ closer to H_1^* on $\widetilde{G}_f$. Given that the momentum representations are generated via the bootstrapping procedure, we name the similarity as the bootstrapping-based similarity. We promote this similarity for each node $v \in \mathcal{V}$ and formally define the loss $\mathcal{L}_{\text{BS}}$ as follows:

$$\mathcal{L}_{\text{BS}} = -\frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \Bigg(\frac{(H_1')^v \cdot (H_2^*)^v}{\|(H_1')^v\| \times \|(H_2^*)^v\|} + \frac{(H_1^*)^v \cdot (H_2')^v}{\|(H_1^*)^v\| \times \|(H_2')^v\|} \Bigg). \tag{8}$$

Consistency Assurance

With the aim of alleviating the inconsistent reconstruction caused by the large portion of different masked contents, we design a consistency assurance module to provide the model with extra stabilized consistency targets, as shown in Figure 1 (e). To demonstrate, we introduce a self-distillation learning paradigm by distilling the knowledge from a momentum decoder teacher to the original decoder student, with the aim of improving the student by learning from the teacher. In particular, we force the original decoder student to learn from the momentum decoder teacher by matching the reconstructions. We find that the momentum decoder teacher functions in the form of model ensembling similar to Polyak-Ruppert averaging with exponential decay, and utilizing Polyak-Ruppert averaging for model ensembling is a standard way for enhancing model capability (Polyak and Juditsky 1992; Jean et al. 2014), which further enables the original student decoder to learn and develop high-quality reconstructions. In specific, we denote the momentum decoder teacher as f_{D^*}, and equip it with the same model structure as the original decoder student f_D. The parameters of f_{D^*} are updated by the exponential moving average of f_D. Then, we send the edge set $\mathcal{E}$ and the momentum representations H_1^* (from Eq. 6) into f_{D^*} to obtain the momentum reconstructions Z_1^*. The process is shown as follows:

$$Z_1^* = f_{D^*}(\mathcal{E}, H_1^*). \tag{9}$$

Next, we compare the original reconstructions Z_1 and the momentum reconstructions Z_1^*, and further encourage them

Types	Methods	Cora	CiteSeer	PubMed	Ogbn-arxiv	PPI
Supervised	GCN	81.5	70.3	79.0	71.74±0.29	75.7±0.1
	GAT	83.0±0.7	72.5±0.7	79.0±0.3	72.10±0.13	97.30±0.20
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	69.68±0.13	63.80±0.20
	MVGRL	83.5±0.4	73.3±0.5	80.1±0.7	-	-
	GRACE	81.9±0.4	71.2±0.5	80.6±0.4	-	69.71±0.17
	BGRL	82.7±0.6	71.1±0.8	79.6±0.5	70.30±0.14	73.63±0.16
	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	70.16±0.22	73.34±0.17
	GraphMAE	84.2±0.4	73.4±0.4	81.1±0.4	70.37±0.11	74.50±0.29
	GraphMAE2	84.4±0.5	73.4±0.3	81.4±0.5	70.43±0.24	74.51±0.36
	S2GAE	84.1±0.5	73.3±0.4	81.2±0.3	70.71±0.24	74.51±0.32
	AUG-MAE	84.3±0.4	73.2±0.4	81.2±0.4	70.41±0.24	74.31±0.32
	ACE-GSL	**85.4±0.4**	**74.6±0.4**	**82.6±0.3**	**72.43±0.18**	**75.43±0.27**

Table 1: Node classification performance comparison. We report accuracy (%) for all datasets except Micro-F1 (%) score for PPI dataset. The best and second-best results are highlighted in bold and underlined, respectively.

to be similar via a scaled cosine error. Since we stop the gradient from the momentum decoder, Z_1 is forced to learn from Z_1^*. We employ the consistency assurance for every node $v \in \widetilde{\mathcal{V}}$ and define the consistency assurance loss $\mathcal{L}_{\text{CA}}$ as follows:

$$\mathcal{L}_{\text{CA}} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v \in \widetilde{\mathcal{V}}} (1 - \frac{Z_1^v \cdot (Z_1^*)^v}{\|Z_1^v\| \times \|(Z_1^*)^v\|})^\beta, \tag{10}$$

where β is the scaling factor. Theoretically, if $(Z_1^*)^v$ and Z_1^v are very similar, then $\mathcal{L}_{\text{CA}}$ will be small, encouraging the model to learn more from the reconstruction objectives. However, if $(Z_1^*)^v$ and Z_1^v differ a lot, then $\mathcal{L}_{\text{CA}}$ will be large, facilitating the model to move in a direction that favors consistency provided by prior knowledge. In addition, the utilization of the scaling factor β enables the decoder to down-weight the contributions of similar reconstruction pairs when $\beta > 1$ while focusing more on those pairs that are distinct. Finally, we train the model by combining the above objectives.

Experiments

In this section, we conduct extensive experiments to validate the effectiveness of ACE-GSL, including node classification, graph classification, transfer learning, ablation study, and parameter sensitivity.

Experimental Setup

Datasets. We conduct experiments on 20 datasets. For the node classification task, we employ five public benchmark datasets, including Cora, CiteSeer, PubMed, Ogbn-arxiv, and PPI. For the graph classification task, we report the performance on seven benchmark datasets: IMDB-M, IMDB-B, PROTEINS, COLLAB, MUTAG, REDDIT-B, and NCI1. For the transfer learning task, we conduct experiments on eight benchmark datasets including BBBP, Tox21, ToxCast, SIDER, ClinTox, MUV, HIV, and BACE.

Baselines. In total, we compare ACE-GSL against 28 methods, including GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), DGI (Velickovic et al. 2019), MVGRL (Hassani and Ahmadi 2020), GRACE (Zhu et al. 2020), BGRL (Thakoor et al. 2022), InfoGCL (Xu et al. 2021a), CCA-SSG (Zhang et al. 2021), GAE (Kipf and Welling 2016), GPT-GNN (Hu et al. 2020b), GATE (Salehi and Davulcu 2020), GIN (Xu et al. 2019), DiffPool (Ying et al. 2018), WL (Shervashidze et al. 2011), DGK (Yanardag and Vishwanathan 2015), graph2vec (Narayanan et al. 2017), Infograph (Sun et al. 2019), GraphCL (You et al. 2020), JOAO (You et al. 2021), No-pretrain, ContextPred (Hu et al. 2020a), AttrMasking (Hu et al. 2020a), Infomax (Linsker 1988), GraphLoG (Xu et al. 2021b), GraphMAE (Hou et al. 2022), GraphMAE2 (Hou et al. 2023), S2GAE (Tan et al. 2023), AUG-MAE (Wang et al. 2024a).

Implementation Details. We report the performances of mean and standard deviation across five runs for node and graph classification, and across ten runs for transfer learning. For the node classification task, the inductive setting is taken from GraphSAGE (Hamilton, Ying, and Leskovec 2017) where the testing is employed on unseen nodes. For the graph classification task, to associate each graph with input features, we utilize node labels as the features for datasets MUTAG, PROTEINS, and NCI1, while leveraging node degrees as the features for datasets IMDB-M, IMDB-B, REDDIT-B, and COLLAB. For the transfer learning task, we follow the experimental setting in (Hu et al. 2020a). In order to simulate the real-world use cases, the downstream datasets are split using scaffold split. The atom number and chirality tag are employed as node features, which provide information about the properties of individual atoms in a molecule. Additionally, the interactions between atoms in the molecule are represented by using bond type and direction as edge features. The experiments are implemented using PyTorch and DGL library (Wang et al. 2019). We optimize the model using Adam (Kingma and Ba 2015).

Types	Methods	IMDB-M	IMDB-B	PROTEINS	COLLAB	MUTAG	REDDIT-B	NCI1
Supervised	GIN	52.3±2.8	75.1±5.1	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	46.95±0.46	72.30±3.44	72.92±0.56	-	80.72±3.00	68.82±0.41	80.31±0.46
	DGK	44.55±0.52	66.96±0.56	73.30±0.82	-	87.44±2.72	78.04±0.39	80.31±0.46
Self-supervised	graph2vec	50.44±0.87	71.10±0.54	73.30±2.05	-	83.15±9.25	75.78±1.03	73.22±1.81
	Infograph	49.69±0.53	73.03±0.87	74.44±0.31	70.65±1.13	89.01±1.13	82.50±1.42	76.20±1.06
	GraphCL	48.58±0.67	71.14±0.44	74.39±0.45	71.36±1.15	86.80±1.34	86.80±1.34	77.87±0.41
	JOAO	49.20±0.77	70.21±3.08	74.55±0.41	69.50±0.36	87.35±1.02	85.29±1.35	78.07±0.47
	InfoGCL	51.40±0.80	75.10±0.90	-	80.00±1.30	**91.20±1.30**	-	80.20±0.60
	GraphMAE	51.63±0.52	75.52±0.66	75.30±0.39	80.32±0.46	88.19±1.26	88.01±0.19	80.40±0.30
	GraphMAE2	51.80±0.60	73.88±0.53	74.86±0.34	77.59±0.22	86.63±1.33	76.84±0.21	78.56±0.26
	S2GAE	51.49±0.62	74.62±0.82	74.91±0.38	79.46±0.28	86.96±1.28	87.86±0.22	80.42±0.32
	AUG-MAE	51.35±0.78	75.47±0.64	74.97±0.37	80.54±0.50	87.72±1.31	88.12±0.45	80.20±0.60
	ACE-GSL	**53.13±0.42**	**76.82±0.63**	**77.26±0.24**	**82.04±0.16**	88.43±1.28	**89.92±0.22**	**81.20±0.20**

Table 2: Graph classification performance comparison. We report accuracy (%) for all datasets.

Node Classification

We conduct node classification in two scenarios to fully evaluate the model, i.e., inductive for PPI and transductive for other datasets. The results are shown in Table 1. According to the table, we find that our model ACE-GSL outperforms all the baselines across all datasets. In particular, traditional generative self-supervised models such as GAE and GPT-GNN perform poorly by simply focusing on the reconstruction of input data, while contrastive self-supervised models such as DGI and InfoGCL perform slightly better by maximizing the agreement between different distributions or augmentations. Among the baselines, the masked autoencoder-based methods GraphMAE2 and S2GAE have the best results in most cases, demonstrating the success of utilizing the masking strategy in self-supervised learning. However, they show several limitations that can significantly affect their learning capabilities. By addressing these issues, ACE-GSL achieves the best performance compared to all methods. This validates the significance of addressing the identified limitations and proves the effectiveness of our model.

Graph Classification

We report the performance of graph classification task in Table 2. We observe that our model ACE-GSL achieves the best performance across various datasets except for one dataset. To illustrate, the dataset MUTAG has the smallest size with only hundreds of graphs, providing the least amount of information to learn, which leads to unstable results with a large variance. Moreover, we can find that ACE-GSL can consistently outperform the current state-of-the-art methods for both mean accuracy and standard deviation. This demonstrates the effectiveness of our method and the necessity of obtaining stable reconstructions via the proposed consistency assurance module. In addition, considering that the node features of the graphs in these seven datasets are one-hot vectors, the exceptional performance of ACE-GSL manifests that the proposed model can learn meaningful representations even when the provided information is insufficient.

Dataset		Node Classification		Graph Classification	
		Cora	PubMed	IMDB-M	PROTEINS
Decoder	MLP	73.0±1.2	80.5±0.4	51.62±0.31	75.82±0.39
	GCN	75.7±1.8	80.5±0.9	51.91±0.38	75.63±0.41
	GIN	74.7±1.7	81.3±0.4	**53.13±0.42**	**77.26±0.24**
	GAT	**85.4±0.4**	**82.6±0.3**	51.14±0.39	75.83±0.43

Table 3: Effect of decoder backbones.

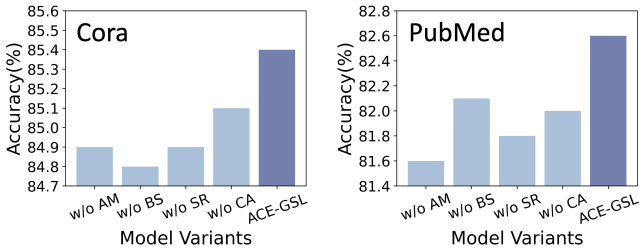

Figure 2: Ablation studies of model components.

Transfer Learning

We assess the model transferability and show the results in Table 4. Based on the table, we find that ACE-GSL achieves the best results and significantly outperforms all baselines across all datasets. ACE-GSL has an average ROC-AUC score of 76.6, outperforming the best baseline S2GAE with score of 74.1. This demonstrates the strong transferability of our model. In addition, we notice that even though S2GAE has a decent average ROC-AUC score, it can only achieve a better result compared to other baselines on three out of eight datasets, while our model ACE-GSL exhibits a stable major improvement on all eight datasets. This again verifies the effectiveness of our model and reflects its incredible learning capacity even in transfer learning task for different datasets.

Methods	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Average
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
GraphMAE2	71.5±1.5	75.8±0.9	65.0±0.8	59.7±0.6	78.9±2.8	78.6±1.2	76.2±2.3	81.2±1.3	73.4
S2GAE	72.3±1.0	75.8±0.5	64.7±0.6	61.3±1.8	80.7±1.7	79.0±2.5	78.2±1.0	80.6±0.8	74.1
AUG-MAE	71.6±1.4	75.4±0.6	63.8±0.4	60.1±1.2	80.1±1.6	76.4±2.5	77.4±1.1	81.8±1.1	73.3
ACE-GSL	**73.5±0.8**	**77.3±0.5**	**65.9±0.4**	**65.2±0.6**	**83.2±1.6**	**80.8±1.2**	**80.3±0.8**	**86.7±1.0**	**76.6**

Table 4: Transfer learning performance comparison. We report ROC-AUC scores (%) for all datasets.

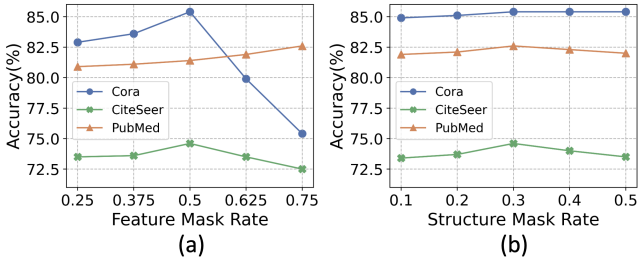

Figure 3: The performance of ACE-GSL *w.r.t.* different values of feature mask rate p_f and structure mask rate p_s.

Ablation Studies

Effect of decoder backbones. We compare with different decoder backbones such as MLP, GCN, GIN, and GAT to validate the effectiveness of ACE-GSL in Table 3. We observe that the usage of GNNs as the decoder achieves better performances than using MLP in most cases. For the node classification, GAT performs the best, while for the graph classification, GIN works the best. Therefore, for the sake of uniformity, we select GAT for all node classification datasets and GIN for all graph classification datasets.

Effect of model components. Since ACE-GSL contains various model components (i.e., adaptive feature mask generator (AM), ranking-based structure reconstruction (SR), bootstrapping-based similarity module (BS), and consistency assurance module (CA)), we analyze the contributions of different components by removing each of them independently. According to Figure 2, the decreasing performances of removing each component demonstrate the effectiveness of each component in enhancing the model.

Parameter Sensitivity

Impact of feature mask rate. The feature mask rate p_f controls the percentages of masked nodes and we show the impact in Figure 3 (a). We find that increasing the rate generally enhances performance. We ascribe this improvement to the comprehensive modeling of the data itself, whereas the use of a low feature mask rate simplifies the learning task and could prevent the model from fully capturing the knowledge. However, further increasing the value could degrade the performance. This is because applying an extreme mask rate provides the model with insufficient information, making it more difficult for the model to encode and concentrate on meaningful knowledge. In addition, the variation in the optimal mask rate for the PubMed dataset can be attributed to the information redundancy present in the graph structure of the dataset, caused by the large node degrees or high homogeneity, allowing the model to recover node features from a small number of neighbor nodes.

Impact of structure mask rate. The structure mask rate p_s controls the percentages of masked edges and we show the impact in Figure 3 (b). We observe that the performance reaches the optimal when $p_s = 0.3$. For the Cora dataset, the performance stays the same when further increasing the value. However, for other datasets, as the mask rate increases, the performance of the model begins to decrease, although the drop is not significant. In general, the choices of different p_s have little impact on the final model performance, demonstrating the robustness of the proposed model. We thus conclude that ACE-GSL is insensitive to the decisions of p_s. We ascribe this advantage to the strong learning capacity of model components, which enables ACE-GSL to function effectively across various structure mask rates.

Conclusion

In this paper, we identify and investigate four significant limitations of existing methods that undermine their capability. To address these limitations, We design ACE-GSL, a novel generative self-supervised learning framework on graphs. ACE-GSL integrates several innovative components from the perspectives of adaptivity, integrity, complementarity, and consistency. Extensive experiments across 20 datasets and 3 graph learning tasks demonstrate that ACE-GSL consistently outperforms 28 competing methods.

Acknowledgments

This work was supported in part by National Science Foundation under the NSF Center for Computer Assisted Synthesis (C- CAS), grant CHE-2202693.

References

Garcia Duran, A.; and Niepert, M. 2017. Learning graph representations with embedding propagation. In *NeurIPS*.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

Hassani, K.; and Ahmadi, A. H. K. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML*.

He, D.; Zhao, J.; Guo, R.; Feng, Z.; Huo, C.; Jin, D.; Pedrycz, W.; and Zhang, W. 2025. Distill & Contrast: A New Graph Self-Supervised Method With Approximating Nature Data Relationships. *IEEE Transactions on Knowledge and Data Engineering*.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollar, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *CVPR*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. B. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*.

Hou, Z.; He, Y.; Cen, Y.; Liu, X.; Dong, Y.; Kharlamov, E.; and Tang, J. 2023. GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner. In *WWW*.

Hou, Z.; Liu, X.; Dong, Y.; Wang, C.; Tang, J.; et al. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *KDD*.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; and Leskovec, J. 2020a. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.-W.; and Sun, Y. 2020b. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*.

Jean, S.; Cho, K.; Memisevic, R.; and Bengio, Y. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Linsker, R. 1988. Self-Organization in a Perceptual Network. *Computer*.

Liu, X.; Zhang, F.; Hou, Z.; Wang, Z.; Mian, L.; Zhang, J.; and Tang, J. 2020. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218*.

Liu, Y.; Xia, L.; and Huang, C. 2024. Selfgnn: Self-supervised graph neural networks for sequential recommendation. In *SIGIR*.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Park, J.; Lee, M.; Chang, H. J.; Lee, K.; and Choi, J. Y. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *ICCV*.

Polyak, B. T.; and Juditsky, A. B. 1992. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*.

Qiao, H.; Wen, Q.; Li, X.; Lim, E.-P.; and Pang, G. 2024. Generative semi-supervised graph anomaly detection. In *NeurIPS*.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*.

Salehi, A.; and Davulcu, H. 2020. Graph Attention Auto-Encoders. In *ICTAI*.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*.

Shi, Y.; Zhou, K.; and Liu, N. 2023. Engage: Explanation guided data augmentation for graph representation learning. In *ECML PKDD*.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial graph augmentation to improve graph contrastive learning. In *NeurIPS*.

Tan, Q.; Liu, N.; Huang, X.; Choi, S.-H.; Li, L.; Chen, R.; and Hu, X. 2023. S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *WSDM*.

Thakoor, S.; Tallec, C.; Azar, M. G.; Munos, R.; Velickovic, P.; and Valko, M. 2022. Large-Scale Representation Learning on Graphs via Bootstrapping. In *ICLR*.

Tian, Y.; Aziz, M.; Wang, A.; Palumbo, E.; and Bouchard, H. 2024. Structural podcast content modeling with generalizability. In *WWW*.

Tian, Y.; Dong, K.; Zhang, C.; Zhang, C.; and Chawla, N. V. 2023a. Heterogeneous Graph Masked Autoencoders. In *AAAI*.

Tian, Y.; Zhang, C.; Guo, Z.; Huang, C.; Metoyer, R.; and Chawla, N. V. 2022. RecipeRec: A Heterogeneous Graph Learning Model for Recipe Recommendation. In *IJCAI*.

Tian, Y.; Zhang, C.; Guo, Z.; Zhang, X.; and Chawla, N. V. 2023b. NOSMOG: Learning noise-robust and structure-aware mlps on graphs. In *ICLR*.

Trivedi, P.; Lubana, E. S.; Yan, Y.; Yang, Y.; and Koutra, D. 2022. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *WWW*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Lio, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR*.

Wan, G.; Tian, Y.; Huang, W.; Chawla, N. V.; and Ye, M. 2024. S3GCL: Spectral, swift, spatial graph contrastive learning. In *ICML*.

Wang, L.; Tao, X.; Liu, Q.; and Wu, S. 2024a. Rethinking graph masked autoencoders through alignment and uniformity. In *AAAI*.

Wang, M.; Zheng, D.; Ye, Z.; Gan, Q.; Li, M.; Song, X.; Zhou, J.; Ma, C.; Yu, L.; Gai, Y.; Xiao, T.; He, T.; Karypis, G.; Li, J.; and Zhang, Z. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315*.

Wang, Y.; Jin, W.; and Derr, T. 2022. Graph neural networks: Self-supervised learning. *Graph Neural Networks: Foundations, Frontiers, and Applications*.

Wang, Z.; Wang, X.; Deng, H.; Liu, N.; Pan, S.; and Hu, C. 2024b. Uncovering the redundancy in graph self-supervised learning models. In *NeurIPS*.

Wu, L.; Lin, H.; Liu, Z.; Liu, Z.; Huang, Y.; and Li, S. Z. 2023. Homophily-Enhanced Self-Supervision for Graph Structure Learning: Insights and Directions. *IEEE Transactions on Neural Networks and Learning Systems*.

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

Xu, D.; Cheng, W.; Luo, D.; Chen, H.; and Zhang, X. 2021a. Infogcl: Information-aware graph contrastive learning. In *NeurIPS*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Xu, M.; Wang, H.; Ni, B.; Guo, H.; and Tang, J. 2021b. Self-supervised graph-level representation learning with local and global structure. In *ICML*.

Yanardag, P.; and Vishwanathan, S. 2015. Deep graph kernels. In *KDD*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph contrastive learning automated. In *ICML*.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. In *NeurIPS*.

Yuan, X.; Tian, Y.; Zhang, C.; Ye, Y.; Chawla, N. V.; and Zhang, C. 2024. Graph cross supervised learning via generalized knowledge. In *KDD*.

Zhang, C.; Huang, C.; Li, Y.; Zhang, X.; Ye, Y.; and Zhang, C. 2022. Look Twice as Much as You Say: Scene Graph Contrastive Learning for Self-Supervised Image Caption Generation. In *CIKM*.

Zhang, C.; Huang, C.; Tian, Y.; Wen, Q.; Ouyang, Z.; Li, Y.; Ye, Y.; and Zhang, C. 2023. When sparsity meets contrastive models: less graph data can bring better class-balanced representations. In *ICML*.

Zhang, H.; Wu, Q.; Yan, J.; Wipf, D.; and Yu, P. S. 2021. From canonical correlation analysis to self-supervised graph neural networks. In *NeurIPS*.

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

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*.