Supplemental Materials: Self-Supervised Learning of Graph Neural Networks: A Unified Review

Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, Shuiwang Ji, *Senior Member, IEEE*

APPENDIX A
HETEROGENEOUS AND DYNAMIC GRAPHS

Compared to typical homogeneous graphs, the heterogeneous graphs further include attributes indicating types of nodes and edges that contain richer topological and feature information. Generally, typical contrastive and predictive frameworks can still be adapted to learn representations from heterogeneous and dynamic graphs, as long as a proper graph encoder, such as message passing neural networks with edge attributes and R-GCNs [1]. To better utilize the information in node and edge types, recent work proposes contrastive methods with view generations and objectives specifically designed for heterogeneous graphs. In particular, for the view generation, HeCo [2] proposes to generate the network schema and meta-paths as two views in the contrastive framework for heterogeneous graphs, as introduced in Section 3.3.3. Moreover, for the computation of contrastive objectives, HGNN [3] proposes to adopt asymmetric projection heads for the representation of two nodes with different types. For predictive methods, the meta-paths in heterogeneous graphs are adopted as additional self-supervision to help improve the performance of downstream tasks. The dynamic graph can be considered as a special case of heterogeneous where the additional attributes contain temporal information indicating the time when the nodes and edges are constructed in a continuous form. Tian et al. [4] propose the time-aware GNN encoder for dynamic graphs and the DDGCL framework where two temporal subgraphs obtained at different time points are adopted as two views. While still following the general contrastive framework, the specific design of view generation, encoders, and objectives for homogeneous and dynamic graphs can bring additional performance gain compared to general contrastive methods on homogeneous graphs.

APPENDIX B
COMPARISONS BETWEEN CONTRASTIVE AND PREDICTIVE MODELS

As described in Section 1, the major methodological difference between contrastive methods and predictive methods is whether paired samples are required for training, as contrastive methods contrast negative pairs from positive ones. They both aim to learn encoders that compute informative representations. For contrastive learning, the goal is achieved by maximizing mutual information between representations of different parts of the data. In other words, the mutual information $I(v_i, f(v_i))$ between any given view v_i and its representation $f(v_i)$ is maximized only if $I(f(v_i), f(v_j))$ is maximized, ideally, to $I(v_i, v_j)$ for any views of a given graph. For predictive methods, the goal is achieved by learning representations that preserve (by being able to predict) certain properties or characteristics of the original graph.

Empirically, contrastive methods are usually more computationally expensive compared to predictive methods but generally outperform most predictive methods in terms of downstream classification performance. On the other hand, recent predictive methods based on invariance-regularization can achieve performance on par with the SOTA contrastive methods. The design of pretext learning tasks in predictive methods is more flexible so more domain knowledge can be included to benefit the learning of representations. However, most predictive methods are less theoretically guided compared to contrastive methods grounded on mutual information, as discussed in Section 7, which may lead to the reduced performance mentioned above.

APPENDIX C
GRAPH ENCODERS IN CONTRASTIVE LEARNING

Graph encoders are usually constructed based on graph neural networks (GNNs) following a neighborhood aggregation strategy, where the representation of a node is iteratively updated by combining the its own representation with the aggregated representation over its neighbors. Formally, the k-th layer of GNN is:

$$\boldsymbol{x}_v^{(k)} = \text{COMBINE}^{(k)}(\boldsymbol{x}_v^{(k-1)}, \boldsymbol{a}_v^k), \tag{1}$$

$$\boldsymbol{a}_v^k = \text{AGGREGATE}^{(k)}\left((\boldsymbol{x}_v^{(k-1)}, \boldsymbol{x}_u^{(k-1)}) : u \in \mathcal{N}(v)\right), \tag{2}$$

where $\boldsymbol{x}_v^{(k)}$ denotes the feature vector of node v at the k-th layer, and $\mathcal{N}(v)$ is a set of neighbor nodes of v. Graph encoders mainly differ from their aggregation strategies. $\text{COMBINE}^{(k)}(\cdot)$ and $\text{AGGREGATE}^{(k)}(\cdot)$ are component functions that determine types of GNNs, such as

Graph Convolutional Networks (GCNs) [5], Graph Attention Networks (GATs) [6] and Graph Isomorphism Networks (GINs) [7].

C.1 Node-Level and Graph-Level Representations

The most straight-forward way to obtain the node-level representation $\boldsymbol{h}_v$ for node v is to directly use the node feature at the final layer K of the encoder [8, 9, 10], *i.e.*, $\boldsymbol{h}_v = \boldsymbol{x}_v^{(K)}$. One may also adopt skip connections or jumping knowledge [11] to generate node-level representation. However, the node-level representation produced by concatenating node features from all layers have different dimension from node features. To avoid such inconsistency in vector dimension, [12, 13] and [14] concatenate node features of all layers, followed by a linear transformation:

$$\boldsymbol{h}_v = \text{CONCAT}([\boldsymbol{x}_v^{(k)}]_{k=1}^{K})\boldsymbol{W}, \tag{3}$$

where $\boldsymbol{W} \in \mathbb{R}^{(\sum_k d_k)\times d}$ is the weight matrix used to shrink the dimension size of $\boldsymbol{h}_v$.

The READOUT function is considered as the key operation to compute the graph-level representation $\boldsymbol{h}_{graph}$ given the node-level representations $\boldsymbol{H}$ of the graph. For the sake of node permutation invariance, summation and averaging are most commonly used READOUT functions. Sun et al. [12], You et al. [13] and Hassani and Khasahmadi [14] employ sum over all the nodes' representations as

$$\boldsymbol{h}_{graph} = \text{READOUT}(\boldsymbol{H}) = \sigma(\sum_{v=1}^{|V|} \boldsymbol{h}_v), \tag{4}$$

where $|V|$ denotes the total number of nodes in the given graph, and σ is either sigmoid function, multi-layer perceptron or identity function. Veličković et al. [8] and Jiao et al. [10] employ mean pooling READOUT that averages all the node-level representations as

$$\boldsymbol{h}_{graph} = \text{READOUT}(\boldsymbol{H}) = \sigma(\frac{1}{|V|}\sum_{v=1}^{|V|} \boldsymbol{h}_v). \tag{5}$$

C.2 Effects of Graph Encoders

Typically, GNN-based encoders are not constrained on choices of GNN types and most frameworks [14, 15] allow various choices. However, some studies have more thorough considerations of GNN types. InfoGraph [12] adopts GIN to achieve less inductive bias for graph-level applications. GraphCL [13] finds GIN outperforms GCN and GAT in semi-supervised learning on node classification tasks. Hu et al. [9] observe that the most expressive GIN achieves the best performance with pre-training, although GIN has slightly inferior performance than the less expressive GNNs without pre-training. That is, GIN achieves the highest performance gain of pre-training. This observation agrees with [16] that fully-utilization of pre-training requires an expressive model as limited expressive models can harm performance and the observation [17] that the quality of learned representations is impacted more by the choice of encoder than the objective. On the contrary, for SUBG-CON [10], GCN-based encoders outperform other GNN-based encoders as GCN is more suitable to handle subgraphs than more expressive GIN and GAT. In addition, Veličković et al. [8] and Zhu et al. [18] employ different encoders on different learning tasks, *i.e.*, GCN for transductive learning tasks, GraphSage-GCN or a mean-pooling layer with skip connection for inductive learning on larges Reddit, and mean-pooling layers with skip or dense skip connections for inductive learning on multiple graphs PPI. These observations imply that different contrastive learning frameworks and methods may prefer distinct GNN types for encoders. Even for the same framework, encoder choices may vary when applying to different datasets.

APPENDIX D
COMPARISON OF CONTRASTIVE OBJECTIVES

Among all contrastive objectives for graphs, the JS estimator $\widehat{\mathcal{I}}^{(JS)}$ and InfoNCE $\widehat{\mathcal{I}}^{(\text{NCE})}$ based on lower-bounds to mutual information are most commonly used. Regarding the two estimators generally, Hjelm et al. [19] empirically shows that InfoNCE generally outperforms the JS estimator in most cases. However, compared to the JS estimator, InfoNCE is more sensitive to the number of negative samples N and requires a large number of negative samples in order to be competitive. Consequently, when the number of negative samples, *i.e.*, the mini-batch size, is limited, the performance of InfoNCE could be limited and the JS estimator may become a better choice.

In addition, Hassani and Khasahmadi [14] perform ablation studies comparing the three objectives $\widehat{\mathcal{I}}^{(\text{DV})}, \widehat{\mathcal{I}}^{(\text{JS})}$ and $\widehat{\mathcal{I}}^{(\text{NCE})}$ with batch size ranged from 32 to 256 for graph classification and ranged from 2 to 8 for node classification. They show that the JS estimator generally leads to the best performance among all objectives on graph classification datasets, while InfoNCE (or NT-Xent) achieves overall best performance on node-classification tasks.

Moreover, Jiao et al. [10] compares the non-bound based triplet margin loss, the logistic loss [20] as an equivalence of the JS estimator and the BPR loss as an equivalence of the InfoNCE with $N = 1$ under their graph contrastive framework. Their results show that the JS estimator and the BPR loss (the InfoNCE loss with $N = 1$) are similarly effective in their method, while the triplet margin loss achieves the best performance among the three objectives. The results indicate that the triplet margin loss can still be effective, given some certain views of the graphs, when the positive pairs and negative pairs should not be discriminated absolutely.

APPENDIX E
SUMMARY OF SSL METHODS FOR GNNS

We summarize contrastive methods and predictive methods reviewed in this survey in Supplementary Table 1 and Supplementary Table 2, respectively.

APPENDIX F
OVERVIEW OF THE SSLGRAPH LIBRARY WITHIN DIG

An overview of the developed DIG-sslgraph library is shown in Supplementary Figure 1.

TABLE 1
Summary of contrastive methods for GNNs in chronological order. Columns categorize the methods by their learning objectives, level of representations for contrast (G: graph, N: node), the type of view generation, and the level of their majorly targeted downstream tasks. We also include notes to show their specific constraints or related literature in other domains such as vision for additional references, when applicable. For downstream tasks, the task of link prediction is also considered as node-level as it is based on representations of a pair of nodes. *The triplet loss is equivalent to the NT-Xent (InfoNCE) loss where the number of negative samples equals to one.

Method	Objective	Rep. Levels	View Generation	Targeted Downstream Tasks	Other Notes
DGI [8]	JSE	G-N	Identical	Node-level	Deep Infomax [19]
InfoGraph [12]	JSE	G-N	Identical	Graph-level	Deep Infomax [19]
Hu et al. [9]	JSE	G-G	Subgraphs	Graph-level	–
GMI [21]	JSE	N-N	Identical	Node-level	–
GCC [15]	InfoNCE	G-G	Subgraphs	Graph-level	–
SUBG-CON [10]	Triplet*	G-G	Subgraphs	Graph-level	–
GRACE [18]	InfoNCE	N-N	Structural & Feature	Node-level	Molecular Graph
MVGRL [14]	JSE	G-N	Structural & Subgraphs	Node-level	–
GraphCL [13]	InfoNCE	G-G	Random	Graph-level	SimCLR [22]
GCA [23]	InfoNCE	N-N	Structural & Subgraphs	Node-level	–
PHD [24]	JSE	G-G	Subgraphs	Graph-level	Molecular Graph
PT-HGNN [3]	InfoNCE	N-N	Structural	Node-level	Heterogeneous Graph
HeCo [2]	InfoNCE	N-N	Subgraphs (Schema & Meta-path)	Node-level	Heterogeneous Graph
InfoGCL [25]	InfoNCE	G-G/G-N/N-N	Optimized (searched)	Graph-level/Node-level	Tian et al. [26]
AD-GCL [27]	InfoNCE	G-G	Optimized (learned)	Graph-level	Tian et al. [26]

TABLE 2
Summary of predictive methods for GNNs. Columns categorize the methods by their sources of supervision, sub-categories, pretext tasks, and paradigms of utilizing self-supervision. In the training paradigm column, *URL* denotes unsupervised representation learning, *Pretrain* denotes unsupervised pretraining, and *Auxiliary* denotes auxiliary learining.

Method	Source of Supervision	Sub-category	Pretext Task	Training Paradigm
GAE [28]	Reconstruction	Graph autoencoder	Adjacency reconstruction	URL
VGAE [28]		Variational autoencoder	Adjacency reconstruction	URL
MGAE [29]		Denoising autoencoder	Node feature reconstruction	URL
ARGA/ARVGA [30]		Variational autoencoder	Adjacency reconstruction	URL
GALA [31]		Graph autoencoder	Node feature reconstruction	URL
SIG-VGA [32]		Variational autoencoder	Adjacency reconstruction	URL
GPT-GNN [33]		Auto-regressive reconstruction	Node and edge reconstruction	URL
SuperGAT [34]		Graph autoencoder	Adjacency reconstruction	Auxiliary
SimP-GCN [35]		Graph autoencoder	Node-pair similarity rec.	Auxiliary
BGRL [36]	Invariance regularization	–	Pseudo-contrastive	URL
CCA-SSG [37]		–	Correlation reduction	URL
LaGraph [38]		–	Latent graph prediction	URL
S^2GRL [39]	Graph properties	Statistical property	K-hop connectivity prediction	URL
GROVER [40]		Statistical and domain knowledge-based	Motif, contextual property pred.	Pretrain
Hwang et al. [41]		Topological property	Meta-path prediction	Auxiliary
M3S [42]	Pseudo-labels	Self-training	Pseudo-label prediction	URL
IFC-GCN [43]		Self-training	Pseudo-label prediction	Pretrain

APPENDIX G
EFFICIENCY AND DOWNSTREAM ACCURACY OF DIG IMPLEMENTATIONS

We compare the efficiency and downstream accuracy between individual original implementations and DIG implementations for SSL methods in Supplementary Table 3. All results are obtained from running corresponding implementations under the unsupervised setting on the same environment and device with a single NVIDIA V100 GPU. We use the standard dataset split for training and test [44] for all datasets. Note that the left column of downstream accuracy are reproduced results using the official code provided by the authors and may differ from the original results reported in their paper due to environment difference, randomness, training/test data split, or any unreleased tricks. For example, the original GRACE code uses different datasets split with individual prefixed random seeds for each dataset. To assure fair comparisons, we perform evaluation of the original GRACE under the standard datasets split. Regarding the efficiency, the DIG-sslgraph implementations consume significantly less training time for GraphCL, MVGRL, and InfoGraph, due to more efficient view generation computations. Besides, DIG consumes the same level of GPU memory as original implementations for all four methods. Although DIG-sslgraph does not focus on efficiency optimization, we will keep improving the efficiency in future released versions.

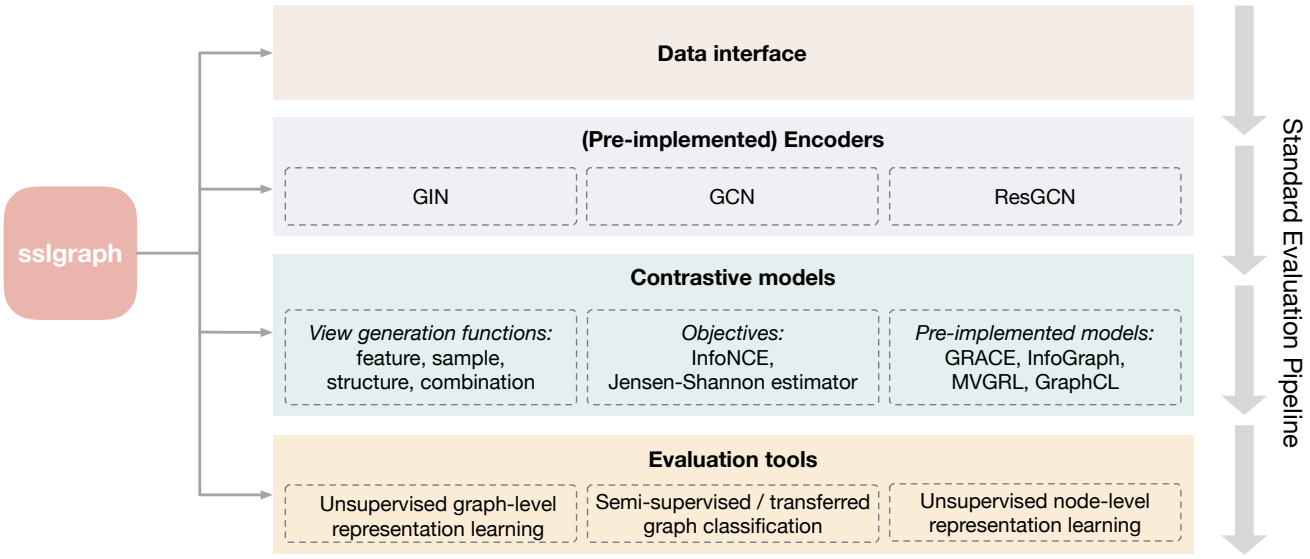

Fig. 1. An overview of the developed sslgraph library within DIG: Dive into Graphs. The library provides a standardized evaluation framework consisting of customizable framework and pre-implemented models for contrastive methods, data interface and evaluation tools for both contrastive and predictive methods.

TABLE 3
Comparisons on efficiency and downstream accuracy between individual original implementations and DIG implementations for SSL methods. Efficiencies are compared in terms of in terms of training time and GPU memory. Four pre-implemented methods, GraphCL, MBGRL, InfoGraph, and GRACE, are compared.

Method	Dataset	Time cost per training epoch		GPU memory consumption		Downstream accuracy	
		Original	DIG	Original	DIG	Original	DIG
GraphCL	NCI1	68.6915s	7.279s	1459MB	1499MB	0.7954 ± 0.0141	0.7961 ± 0.0143
	PROTEINS	5.223s	3.996s	1463MB	1607MB	0.7547 ± 0.0350	0.7637 ± 0.0290
	MUTAG	0.278s	0.446s	1431MB	1475MB	0.8991 ± 0.0495	0.9096 ± 0.0669
MVGRL	MUTAG	9.417s	1.602s	1917MB	2091MB	0.8778 ± 0.0665	0.8877 ± 0.0646
	PTC-MR	8.213s	2.986s	2827MB	2493MB	0.5755 ± 0.0670	0.5903 ± 0.0859
	IMDB-B	27.342s	8.557s	4459MB	4783MB	0.7450 ± 0.0377	0.7370 ± 0.0377
InfoGraph	MUTAG	0.585s	0.551s	1459MB	1459MB	0.8939 ± 0.0885	0.9041 ± 0.0927
	PTC-MR	0.808s	0.924s	1471MB	1445MB	0.6249 ± 0.0966	0.6196 ± 0.0422
	IMDB-B	9.356s	3.013s	1485MB	1465MB	0.7370 ± 0.0276	0.7400 ± 0.0313
GRACE	CORA	0.020s	0.064s	1701MB	1773MB	0.7869 ± 0.0013	0.7877 ± 0.0096
	CiteSeer	0.030s	0.083s	2001MB	2145MB	0.6858 ± 0.0004	0.6842 ± 0.0061
	PubMed	0.233s	0.345s	12703MB	13963MB	0.8217 ± 0.0005	0.8188 ± 0.0046

REFERENCES

[1] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[2] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 1726–1736.

[3] X. Jiang, T. Jia, Y. Fang, C. Shi, Z. Lin, and H. Wang, "Pre-training on large-scale heterogeneous graph," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 756—-766.

[4] S. Tian, R. Wu, L. Shi, L. Zhu, and T. Xiong, "Self-supervised representation learning on dynamic graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1814–1823.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[7] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[8] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and D. Hjelm, "Deep graph infomax," in *International Conference on Learning Representations*, 2019.

[9] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations*, 2020.

[10] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, "Sub-graph contrast for scalable self-supervised

graph representation learning," in *IEEE International Conference on Data Mining*. IEEE, 2020, pp. 222–231.

[11] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.

[12] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *International Conference on Learning Representations*, 2019.

[13] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 5812–5823.

[14] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[15] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "GCC: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1150–1160.

[16] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, no. 19, pp. 625–660, 2010.

[17] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," in *International Conference on Learning Representations*, 2020.

[18] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," in *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[19] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, Workshop Track Proceedings*, 2013.

[21] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference 2020*, 2020, pp. 259–270.

[22] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the International Conference on Machine Learning*, 2020.

[23] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of The Web Conference 2021*, 2021.

[24] P. Li, J. Wang, Z. Li, Y. Qiao, X. Liu, F. Ma, P. Gao, S. Song, and G. Xie, "Pairwise half-graph discrimination: A simple graph-level self-supervised strategy for pre-training graph neural networks," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021, pp. 2694–2700.

[25] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "Infogcl: Information-aware graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[26] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *arXiv preprint arXiv:2005.10243*, 2020.

[27] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[28] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

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

[30] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2609–2615.

[31] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6519–6528.

[32] A. Hasanzadeh, E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Semi-implicit graph variational auto-encoders," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 10 712–10 723.

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

[34] D. Kim and A. Oh, "How to find your friendly neighborhood: Graph attention design with self-supervision," in *International Conference on Learning Representations*, 2021.

[35] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. ACM, 2021.

[36] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko, "Large-scale representation learning on graphs via bootstrapping," *arXiv preprint arXiv:2102.06514*, 2021.

[37] H. Zhang, Q. Wu, J. Yan, D. Wipf, and S. Y. Philip, "From canonical correlation analysis to self-supervised graph neural networks," in *Advances in Neural Information Processing Systems*, 2021.

[38] Y. Xie, Z. Xu, and S. Ji, "Self-supervised representation learning via latent graph prediction," *arXiv preprint arXiv:2202.08333*, 2022.

[39] Z. Peng, Y. Dong, M. Luo, X.-M. Wu, and

Q. Zheng, "Self-supervised graph representation learning via global context prediction," *arXiv preprint arXiv:2003.01604*, 2020.

[40] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," in *Advances in Neural Information Processing Systems*, 2020.

[41] D. Hwang, J. Park, S. Kwon, K.-M. Kim, J.-W. Ha, and H. J. Kim, "Self-supervised auxiliary learning with meta-paths for heterogeneous graphs," *arXiv preprint arXiv:2007.08294*, 2020.

[42] K. Sun, Z. Lin, and Z. Zhu, "Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5892–5899.

[43] Z. Hu, G. Kou, H. Zhang, N. Li, K. Yang, and L. Liu, "Rectifying pseudo labels: Iterative feature clustering for graph representation learning," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, p. 720–729.

[44] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.