Graph Transformation Augmentation for Contrastive Learning of Graph-Level Representation: An Initial Exploration

Tianchao Li
Department of Physics and Technology
UiT The Arctic University of Norway
litianchao1996@outlook.com

Yulong Pei
Department of Mathematics and Computer Science
Eindhoven University of Technology
y.pei.1@tue.nl

Abstract

Contrastive learning on the image data becomes a representative method of self-supervised learning to pre-train a neural encoder from data or model perspective(s). However, the data-perspective method in the graph domain is less explored because graph data augmentation is not as mature as image data augmentation. In this paper, we propose a transformation-based graph data augmentation, which is named Graph Transformation Augmentation (GTA). GTA will preserve the information of the graph spectrum instead of the subgraph information. GTA has two types: Permutation Augmentation and Orthonormal Augmentation. Finally, we experimentally validate the workability of GTA on self-supervised representation learning, and GTA counter-intuitively preserves the graph semantics.

1 Introduction

Self-supervised learning [23] has gained intensive attention, as an alternative direction to supervised learning due to the need for vast amounts of labeled data, which can be both time-consuming and expensive to acquire. SSL sufficiently leverages the underlying structure and relationships within the data to enable models to learn from data without explicit labels. Recent advances in SSL emerged in self-supervised representation learning. Self-supervised representation learning has been explored in Computer Vision, such as the variant of Autoencoder [10], and Contrastive Learning [3]. Contrastive Learning is to push the views (vectorized representation) via the neural encoder from the same sample closer and the views from different samples farther. Computer Vision has two types of data augmentation, such as spatial transformation augmentation (cropping, resizing, rotation [8], and cutout [6]), and appearance transformation augmentation (color distortion [12, 27], Gaussian blur, and Sobel filtering). SimCLR [3] experimentally shows contrastive learning benefits from stronger data augmentation (color distortion) than supervised learning. However, due to the complexity of graph data, graph data augmentation for Contrastive Learning mainly includes node/edge dropping, feature masking, and subgraph [34]. These types of graph data augmentation are similar to cropping, and the augmentation is relatively weak such that the graph semantics can be intuitively preserved to a great extent. Graph spectrum contains much structure information of the whole graph, so we transform the whole graph preserving the graph spectrum to obtain the graph data augmentation. In this way, the augmentation with sufficient information would be significantly strong because transformation should make it different from the original graph. Based on the conclusion of SimCLR, Contrastive Learning in the graph domain should benefit from this transformation-based graph augmentation. Therefore, we propose two types of graph data augmentation similar to rotation and appearance transformation respectively. Our contribution is in the following:

38th Conference on Neural Information Processing Systems (NeurIPS 2024), Machine Learning and Compression Workshop. This workshop does not have official proceedings and this paper is non-archival.

- We propose a transformation-based graph data augmentation named Graph Transformation Augmentation (GTA) preserving the graph spectrum. GTA transforms the original graph as a whole and includes two types: Permutation Augmentation and Orthonormal Augmentation.
- We adjust the GNN-based encoder for GTA considering the measure domain on the feature matrix. Precisely, after the node-level aggregation, the feature matrix is transformed back to the original domain.
- We experimentally validate GTA on Contrastive Learning, and prove that GTA counter-intuitively preserves graph semantics.

2 Preliminary

This paper considers the finite, unweighted and undirected graph: the number of nodes $N < +\infty$, and the adjacency matrix A is symmetric. A graph is denoted as $G = \{X, A\}$, where $X \in \mathbb{R}^{N \times d}$ is the feature matrix, and $A \in \{0, 1\}^{N \times N}$ is the adjacency matrix. In this graph, the feature of each node is d-dimensional, and the feature of the nth node is $x_n = X[n, :]$. $A_{i,j} = 1$ means the i-th node and j-th node are connected, and $A_{i,j} = 0$ represents no connectivity. Since A is symmetric and its entities are real, A has an orthonormal basis of eigenvectors with real eigenvalues [25].

Definition 1 (Graph Spectrum[1])**.** The set of graph eigenvalues of the adjacency matrix is called the spectrum of the graph.

The graph spectrum does contain surprisingly much information about the graph. For example, the examination of the eigenvalues and eigenvectors of their associated matrices (the adjacency matrix and Laplace matrix) can reveal combinatorial properties of graphs, and the eigenvalues provide an easy way to distinguish many pairs of non-isomorphic graphs, as the eigenvalues do not depend upon the ordering of the vertices [26]. Precisely, although two graphs with the same spectrum can not be isomorphic, two isomorphic graphs have the same spectrum. Therefore, spectrum-based machine learning algorithms [4] gain attention due to mathematically provable high-performance.

Recently, Graph Neural Networks (GNNs) [15, 29, 32] have become the mainstream method to analyze graph data due to their promising performance. The message-passing scheme [9] can summarize existing GNNs. Firstly, they iteratively aggregate information from the node's neighborhood following the message-passing mechanism. Considering a K-layer GNN $f(\cdot)$, the propagation of the kth layer is represented as:

$$\mathbf{a}_n^{(k)} = \texttt{AGGREGATION}^{(k)}\left(\{\mathbf{h}_{n'}^{(k-1)}\} : n' \in \mathcal{N}(n)\right), \mathbf{h}_n^{(k)} = \texttt{COMBINE}^{(k)}\left(\mathbf{h}_n^{(k-1)}, \mathbf{a}_n^{(k)}\right), \quad (1)$$

where $\mathbf{h}_n^{(k)}$ is the embedding of the nth node of the graph at the kth layer with $\mathbf{h}_n^{(0)} = x_n$, $\mathcal{N}(n)$ is the set of the nth node's neighborhood, and $\texttt{AGGREGATION}^{(k)}(\cdot)$ and $\texttt{COMBINE}^{(k)}(\cdot)$ are component functions of the kth GNN layer. After K-layer propagation, the $\texttt{READOUT}(\cdot)$ function summarizes the node-level representations and outputs the graph-level representation:

$$f(G) = \texttt{READOUT}(\{\mathbf{h}_n^{(K)}, n \in \{1, 2, \ldots, N\}\}). \quad (2)$$

The main task of this paper is to validate our augmentation technique to pre-train the GNN-based encoder via contrastive learning.

3 Related Works

Self-supervised Representation Learning is to represent data samples utilizing an embedding space without supervision. One straightforward method is reconstruction-based using the variant of the autoencoder [16, 22, 5, 10]. The other method is Contrastive Learning which brings similar samples closer while pushing dissimilar ones farther away without supervision from data and/or model perspective(s) [3, 34, 31]. In the graph domain, graph data augmentation plays an important role in the data-perspective Contrastive Learning. The goal of graph data augmentation is to generate augmented graph(s) that can enrich or enhance the preserved information from the given graph(s) [7]. Structure-oriented graph data augmentations are widely implemented in graph contrastive learning,

[1] In physics, the eigenvalues of the Laplacian matrix of a graph are sometimes known as the graph's spectrum.

including edge-dropping [28], node-dropping [36], graph-sampling [14, 37]. These methods will use the substructure of the graph as the augmentation, which destroys the completeness of the graph compared to the original. This completeness is vital for molecule data and less for social data.

4 Methodology

This section illustrates one graph augmentation technique first, and it is named Graph Transformation Augmentation (GTA) by preserving the graph spectrum. We denote the augmented graph as $\widehat{G} = \{\widehat{X}, \widehat{A}\}$. Next, we initially adapt this technique to graph-level representation learning via contrastive learning based on GraphCL [34].

4.1 Graph Transformation Augmentation

As Section 2 demonstrates, the graph spectrum implicitly contains much graph information. Therefore, the augmented graph should preserve this important information, namely, the spectrum of the augmented graph should be identical to the spectrum of the original. To preserve the spectrum of the original graph, the original adjacency matrix will be transformed to its similar matrix as the augmentation adjacency matrix based on Definition 2, namely $\widehat{A} = M^{-1}AM$. Concurrently, the feature matrix should experience synchronous transformation. However, the shape of the feature matrix is usually not identical to the adjacency matrix, so the similarity transformation cannot straightforwardly be implemented on the feature matrix.

Definition 2 (Matrix Similarity [2])**.** In the linear algebra, two $n \times n$ matrices A and B are called similar if there exists an invertible $n \times n$ matrix M such that

$$B = M^{-1}AM, \tag{3}$$

where A and B have the same eigenvalues and normally different eigenvectors.

The internal representation depends on the covariance matrix $\Sigma^x = XX^\intercal = V\Lambda V^\intercal$, where eigenvalues Λ decide the strength of mapping and eigenvectors V decide the transformation mode [21]. It is worthy to be observed that Σ^x and A are in identical shape, and Σ^x represents relations between nodes like adjacency matrix A. Therefore, Σ^x is chosen for transformation instead of X, in which the strength would be preserved but the transformation mode would be changed. At this step, the augmented feature covariance matrix is $\widehat{\Sigma}^x = M^{-1}\Sigma^x M = \widehat{X}^\intercal\widehat{X}$, so this should be further factorized for augmented feature $\widehat{X}$. However, $\widehat{\Sigma}^x$ is factorizable as long as it is a symmetric semi-positive definite matrix. As Theorem 1 demonstrates, the matrix M should be invertible such that the augmented feature $\widehat{X}$ can be calculated. The proof of Theorem 1 is in Appendix A.2.1

Theorem 1. For any invertible matrix $M \in \mathbb{R}^{N\times N}$, the generated covariance matrix $\widehat{\Sigma}^x$ of the augmentation graph's node features $\widehat{X}$ is a symmetric semi-positive definite matrix, such that $\widehat{\Sigma}^x$ can be factorized as the augmentated graph's node feature $\widehat{X}$ because of $\widehat{\Sigma}^x = \widehat{X}\widehat{X}^\intercal$, where $\widehat{X} \in \mathbb{R}^{N\times d}$ and $\widehat{\Sigma}^x \in \mathbb{R}^{N\times N}$.

Now we identify M to satisfy the constraint on the adjacency matrix. As aforementioned, the graph we consider is undirected, so the augmented graph should be undirected as well. Therefore, the augmented adjacency matrix $\widehat{A}$ should be symmetric. Theorem 2 proves that the orthonormal matrix M can make the augmented adjacency matrix symmetric. The proof of Theorem 2 is in Appendix A.2.2. On this condition that M is an orthonormal matrix, we have $\widehat{\Sigma}^x = (M^\intercal X)(M^\intercal X)^\intercal = \widehat{X}\widehat{X}^\intercal$. We can obtain $\widehat{X} = M^\intercal X$ without factorization.

Theorem 2. For any orthonormal matrix $M \in \mathbb{R}^{N\times N}$, the augmented graph $\widehat{G} = \{\widehat{X}, \widehat{A}\}$ is an undirected graph as the same as the original graph $G = \{X, A\}$, where $X, \widehat{X} \in \mathbb{R}^{N\times d}$ and $A, \widehat{A} \in \mathbb{R}^{N\times N}$

Here, we have to claim $M \in \mathbb{R}^{N\times N}$. If the Fourier matrix is implemented, the augmented feature matrix will become $M^\intercal X$ with complex values. To our best knowledge, current GNNs cannot handle complex features, except for embedding the real and imaginary parts together as a feature vector. Such embedding will double the feature dimension during contrastive learning, and the model cannot

compute the representation with the original data as the input. Although the Fourier matrix will transform the feature to the Fourier domain, which is a stable and decent transformation, due to the capability of GNNs, we will not consider it. To summarize, we can use two types of the orthonormal matrix M: permutation matrix and random orthonormal matrix in $\mathbb{R}^{N\times N}$. After transformation using the orthonormal matrix M, the augmented graph is $\widehat{G} = \{M^{\mathrm{T}}X, M^{\mathrm{T}}AM\}$

The example augmentation via two types of transformation matrices on one one-hot encoding CO_2 molecule is visualized in Figure 1: the graph via the top transformation with one permutation matrix is named Permutation Augmentation; the graph via the bottle transformation with one random orthonormal matrix is named Orthonormal Augmentation. Since the random orthonormal matrix is non-singular, the orthonormal augmentation is a complete graph as long as there is no isolated part in the original graph. Therefore, we will regard the orthonormal augmentation as a fully connected graph. Such fully-connected graph would require more GPU memory for training. Similar to the rotation matrix to achieve image rotation [8] in the visual domain, the permutation matrix is utilized to rotate the original graph. The orthonormal augmentation transforms the original graph (feature and connection) into a random domain. From the result-oriented perspective, the orthonormal augmentation is similar to appearance transformation [12, 27] in Computer Vision. Intuitively, the orthonormal augmentation significantly differs from the original graph, and less graph semantics can be preserved especially the connectivity. Given the original graph $G = \{X, A\}$ and the orthonormal augmentation $\widehat{G} = \{\widehat{X}, \widehat{A}\}$, we can obtain a new adjacency matrix $A' = A \odot \widehat{A}$ as the adjacency matrix, where $\odot$ represents the Hadamard product. A' is more similar to A compared to $\widehat{A}$ because A' and A represent the same connectivity except the weight and $\widehat{A}$ means full connectivities. We call $\widehat{G}' = \{\widehat{X}, A'\}$ as Sparsized Orthonormal Augmentation (SOA). However, we can reconstruct G from $\widehat{G}$ using (computed) M instead of from G'. Therefore, our augmentation counter-intuitively preserves the original graph semantics. Furthermore, edge-dropping and node-dropping methods will destroy the graph's completeness, especially in the molecule data. For instance, dropping one connectivity or one atom usually means one different molecule or nothing. Our method transforms the graph as a whole, such that this problem can be solved. Therefore, in this paper, we experiment on the molecule dataset.

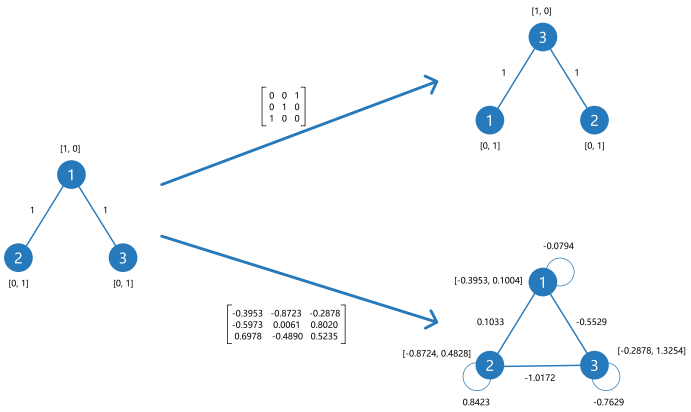

Figure 1: The left graph represents a molecule CO_2 using one-hot encoding as the atom feature; The right-top graph is one augmentation graph via one permutation matrix: The right-bottom graph is one augmentation graph via one random orthonormal matrix.

4.2 Contrastive Learning with Graph Transformation Augmentation

Our contrastive learning mainly follows the framework of GraphCL [34] to maximize the agreement between the augmented views, generated by the GNN-based encoder with the augmented graphs as the input. Our framework also mainly consists of four components:

(1) Graph Augmentation The given graph G will be transformed by the random permutation and orthonormal matrix to obtain the permutation augmentation $\widehat{G}_p$ and orthonormal augmentation $\widehat{G}_o$ respectively, as a positive pair.

(2) GNN-based Encoder A GNN-based encoder $f_{\theta_1}(\cdot)$ with learnable parameters $\theta_1 \in \Theta_1$ consists of two parts: node-level $\texttt{AGGREGATION}(\cdot)$ and $\texttt{COMBINE}(\cdot)$, and $\texttt{READOUT}(\cdot)$ summarizing all the node features as the vectorized graph-level representation (h_i, h_j). When we use one orthonormal augmentation to compute its graph-level representation, after the node-level aggregation with feature matrix A_r, we compute $A_o = MA_r$ as the input of the following $\texttt{READOUT}(\cdot)$, because the feature A_r is in a random domain, and this matrix multiplication will transform the feature back to the original domain (Backward Transformation (BT)). In the pre-train state, we use the adjacency matrix of $\widehat{G}_p$ as the edge weight, and for the downstream task, we use the $\mathbf{0}$ matrix as the edge weight matrix of the original graph. In this way, the permutation augmentation's representation differs from the original graph's when we use a permutation-invariant $\texttt{READOUT}(\cdot)$, and the unreal edge weight will not affect the aggregation scheme using the real graph. Any GNN architecture considering edge weights and features can implemented on contrastive learning with GTA.

(3) Projection Head A non-linear transformation $g_{\theta_2}(\cdot)$ with the learnable parameter $\theta_2 \in \Theta_2$ projects the graph-level representation to another latent space to calculate the contrastive loss, as advocated in GraphCL [34]. We apply a two-layer perceptron (MLP) to obtain z_i, z_j.

(4) Contrastive Loss Function The contrastive loss function is denoted as $\mathcal{L}$, which is defined to maximizing the consistency between positive pairs z_i, z_j and minimizing the consistency between both and negative pairs. We use NT-Xent (the normalized temperature-scaled cross-entropy loss) as the loss function. We re-annotate z_i, z_j as $z_{n,i}$, $z_{n,j}$ to index the n-th graph in the dataset or minibatch. So NT-Xent for the n-th graph is defined as:

$$l_n = -\log \frac{\exp(sim(z_{n,i}, z_{n,j}))/\tau}{\sum_{n'=1,n'\neq n}^{N} \exp(sim(z_{n,i}, z_{n',j}))/\tau}, \tag{4}$$

where τ denotes the temperature parameter, $sim(z_{n,i}, z_{n,j}) = z_{n,i}^{\intercal} z_{n,j} / \|z_{n,i}\| \|z_{n,j}\|$ denotes the cosine similarity function, and the negative pairs are the other $N-1$ augmented graphs. So, the contrastive loss function is defined as:

$$\mathcal{L} = \sum_{n=1}^{N} l_n. \tag{5}$$

This loss function will guide the representations of GTAs tending to invariance to (permutation) transformation via the permutation matrix and the orthonormal matrix. Due to the strong randomness of the orthonormal matrix, theoretically, the training should require a sufficiently large number of epochs for transformation invariance. Minimizing this loss is equivalent to maximizing the mutual information between h_1 and h_2 [34]. As the discussion in Section A.3, GTA will implicitly help the GNN-based encoder achieve the InfoMax Principle [18, 17] through contrastive learning.

5 Experiments

This section will validate our method on molecule datasets using GINE to achieve node-level aggregation. The datasets include `NCI1`, `PROTEINS` and `MUTAG` from `TUDataset` [19]. In the experiment, the GNN-based encoder is trained without supervision, and evaluates the quality of the encoder based on the performance on the downstream task following the setting in [34].

To test the workability of our augmentation technique, we chose GraphCL as the benchmark, because it is a purely data-perspective method. In the combination of data-perspective and model-perspective methods like [33], we hard to identify whether our augmentation technique works. For comparison, we implement Graph Isomorphism Network (GIN) [32] as the backbone to achieve aggregation procedure. However, the orthonormal permutation will generate a weighted graph. To incorporate the edge weight $\mathbf{e}_{ij}$ on the edge connecting the ith and jth nodes in the aggregation procedure, we adapt the edge version of GIN (GINE) [13] using a Multilayer Perceptron (MLP) to map $\mathbf{e}_{ij}$ to the same dimension as the node feature. The operator is defined as:

$$\mathbf{h}_n^{(k)} = h_\Theta \left((1+\epsilon) \cdot \mathbf{h}_n^{(k-1)} + \sum_{j \in \mathcal{N}(n)} ReLU(\mathbf{h}_j^{(k-1)} + MLP(\mathbf{e}_{ij})) \right), \tag{6}$$

where we use Multilayer Perceptron (MLP) to represent $h_\Theta(\cdot)$.

Table 1 compares our method with GraphCL: GINE defined as Equation 6 as the GNN-based encoder, LSTM as the READOUT$(\cdot)$. On PROTEINS and MUTAG datasets, our method outperforms GraphCL, but GraphCL performs better on NCI1 dataset. This is potentially because its feature dimension is relatively high, and the edge weight should be mapped to the high dimension as the feature's. Table 2a demonstrates backward transformation will benefit contrastive learning: better performance with backward transformation on the downstream task on the NCI1 and PROTEINS datasets. Furthermore, Figure 2 and Figure 3 in Appendix respectively visualize the pre-train curves with/without backward transformation on PROTEINS. That pre-train curve with backward transformation decreases slightly faster in the beginning stage and oscillates less in the end stage. Table 2b shows the decreasing performance on the downstream task using SOA, so the orthonormal augmentation preserves more meaningful information compared to SOA. Table 3 illustrates the performances of GTA with LSTM (permutation-variant) and Mean Average (permutation-invariant) on NCI1 and PROTEINS datasets. It is clear that the GNN-based encoder with Mean Average performs worse than LSTM on the downstream task. Theoretically, GTA's permutation augmentation will guide the GNN-based encoder to be permutation-invariant during the pre-train. However, the permutation-invariant Mean Average will significantly weaken the effect of permutation augmentation.

Table 1: Comparing classification accuracy on representation learning methods with GraphDCL.

Dataset	GraphCL	*Ours*	Mean Diff.	Highest Acc.
NCI1	77.87±0.41	75.56±0.10	-2.31	75.96
PROTEINS	74.39±0.45	75.27±0.04	0.88	75.65
MUTAG	86.80±1.34	87.05±0.17	0.25	87.81

Table 2: Comparing the effect of Backward Transformation and Sparsized Orthonormal Augmentation on NCI1 and PROTEINS

(a) Comparing the effect of Backward Transformation on NCI1 and PROTEINS.

BT	Dataset	Accuracy
✓	NCI1	75.56±0.10
×	NCI1	74.16±0.10
✓	PROTEINS	75.27±0.04
×	PROTEINS	74.90±0.07

(b) Comparing the effect of Sparsized Orthonormal Augmentation on NCI1 and PROTEINS.

SOA	Dataset	Accuracy
×	NCI1	75.56±0.10
✓	NCI1	73.43±0.12
×	PROTEINS	75.27±0.04
✓	PROTEINS	74.98±0.03

Table 3: Comparing the effect of LSTM and Mean Average on NCI1 and PROTEINS.

READOUT	Dataset	Accuracy
LSTM	NCI1	75.56±0.10
Mean Average	NCI1	70.35±0.05
LSTM	PROTEINS	75.27±0.04
Mean Average	PROTEINS	71.86±0.02

6 Conclusion

We propose a transformation-based graph augmentation, Graph Transformation Augmentation, for graph contrastive learning. This method is experimentally validated. Although the improvement on the downstream task is not significant, GINE does not process orthonormal augmentation well. In the future, we will explore other more suitable GNN architecture especially Graph Transformer [35]. Furthermore, the eigenvalues of the graph Laplacian are also informative for the original graph. Therefore, the spectrum of graph Laplacian will be investigated. Finally, the contrastive loss in this paper inheriting from GraphCL [34] and SimpleCLR [3] ensures that the representations of different samples are distinct [23]. Recent $\mathbb{X}$-sample contrastive loss [24], extending the self-similarity of the sample to the similarity across samples, potentially benefits our methods.

References

[1] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

[2] Richard Bronson. *Matrix methods: An introduction*. Gulf Professional Publishing, 1991.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[4] Mihai Cucuringu, Ioannis Koutis, Sanjay Chawla, Gary Miller, and Richard Peng. Simple and scalable constrained clustering: a generalized spectral method. In *Artificial Intelligence and Statistics*, pages 445–454. PMLR, 2016.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Terrance DeVries. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[7] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2):61–77, 2022.

[8] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[10] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[11] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2018.

[12] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

[13] 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 (ICLR)*, 2020.

[14] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*, pages 222–231. IEEE, 2020.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[16] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[17] Tianchao Li and Yulong Pei. Mole: Modular learning framework via mutual information maximization. In *ICML Workshop on Localized Learning (LLW)*, 2023.

[18] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[19] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[20] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270, 2020.

[21] A Saxe, J McClelland, and S Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Represenatations 2014*. International Conference on Learning Represenatations 2014, 2014.

[22] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[23] Ravid Shwartz Ziv and Yann LeCun. To compress or not to compress—self-supervised learning and information theory: A review. *Entropy*, 26(3):252, 2024.

[24] Vlad Sobal, Mark Ibrahim, Randall Balestriero, Vivien Cabannes, Diane Bouchacourt, Pietro Astolfi, Kyunghyun Cho, and Yann LeCun. $\mathbb{X}$-sample contrastive loss: Improving contrastive learning with sample similarity graphs. In *ICML 2024 Workshop on Foundation Models in the Wild*.

[25] Daniel Spielman. Spectral and algebraic graph theory. *Yale lecture notes, draft of December*, 4:47, 2019.

[26] Daniel A Spielman. Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 29–38. IEEE, 2007.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[28] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[30] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[31] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*, pages 1070–1079, 2022.

[32] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[33] Zelin Yao, Chuang Liu, Xueqi Ma, Mukun Chen, Jia Wu, Xiantao Cai, Bo Du, and Wenbin Hu. Dual-perspective cross contrastive learning in graph transformers. *arXiv preprint arXiv:2406.00403*, 2024.

[34] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[35] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[36] Sixiao Zhang, Hongxu Chen, Haoran Yang, Xiangguo Sun, Philip S Yu, and Guandong Xu. Graph masked autoencoders with transformers. *arXiv preprint arXiv:2202.08391*, 2022.

[37] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020.

A Appendix

A.1 Related Works

Self-Supervised Representation Learning Self-supervised Representation Learning is to represent data samples utilizing an embedding space without supervision. One straightforward method is reconstruction-based using the variant of the autoencoder [16]. However, the reconstruction task using the autoencoder lacks a real correct answer. The mask mechanism provides such an answer to the reconstruction task leading to better performance on the downstream task [22, 5, 10]. Furthermore, contrastive learning is proposed in Computer Vision [3]. It brings similar samples closer while pushing dissimilar ones farther away without supervision from data and/or model perspective. In the graph domain, GraphCL [34] is the representative of the data-perspective method using the data augmentation for the positive/negative pairs. SimGrace [31] augments the model by introducing Gaussian noise to obtain the pair for contrastive learning. The hybrid of two perspectives [33] improves the performance compared to a single perspective.

Graph Data Augmentation The goal of graph data augmentation is to generate augmented graph(s) that can enrich or enhance the preserved information from the given graph(s) [7]. This paper only discusses the *non-learnable* augmentations. Structure-oriented graph data augmentations are widely implemented in graph contrastive learning, including edge-dropping [28], node-dropping [36], graph-sampling [14, 37]. These methods will use the substructure of the graph as the augmentation, which destroys the completeness of the graph compared to the original. This completeness is vital for the molecule data and less for the social network. For example, the lack of only one edge or node leads to one different molecule the augmented graph represents.

A.2 Proof of Theorems

Since the definition of matrix similarity is derived from linear algebra, to inherit the property from linear algebra, we change the notation of the graph: use the column space of the feature matrix as the node feature. Specifically, a Graph is denoted as $G = \{X, A\}$, where $X \in \mathbb{R}^{d\times N}$ and $A \in \mathbb{R}^{N\times N}$.

A.2.1 Proof of Theorem 1

Proof of Theorem 1. Similar to Σ^x, $\widehat{\Sigma}^x$ is the covariance matrix of the augmentation graph's nodes $\widehat{X}$, so $M^{-1}\Sigma^x M$ should be symmetric semi-positive definite as well as Σ^x. Such that, $\widehat{\Sigma}^x$ can be factorized as $\widehat{X}^\intercal\widehat{X}$. Proving $\widehat{\Sigma}^x$ is a symmetric semi-positive definite matrix is equivalent to

$$\forall x, x^\intercal\widehat{\Sigma}^x x \geq 0, \tag{7}$$

where x is a N-dimensional column vector. Since Σ^x is symmetric semi-positive definite, Σ^x can be decomposed through eigendecomposition as

$$\Sigma^x = Q\Lambda Q^{-1} = Q\Lambda Q^\intercal, \tag{8}$$

where Q is the eigenvector and orthonormal matrix ($Q^{-1} = Q^\intercal$) and Λ is the eigenvalue and diagonal matrix. Since Σ^x is a symmetric semi-positive definite matrix, all the diagonal values of Λ are non-negative. So, we have the following equivalence relation:

$$\forall x, x^\intercal\widehat{\Sigma}^x x \geq 0 \Leftrightarrow \forall x, (Q^\intercal(M^{-1})^\intercal x)^\intercal\Lambda(Q^\intercal Mx) \geq 0, \tag{9}$$

where $\Leftrightarrow$ represents the equivalence relation, and $Q^\intercal(M^{-1})^\intercal x$ and $Q^\intercal Mx$ are the N-dimensional column vectors. $\Lambda(Q^\intercal Mx)$ means the component $(Q^\intercal Mx)_i$ is non-negatively scaled by Λ_{ii}. $(Q^\intercal(M^{-1})^\intercal x)^\intercal\Lambda(Q^\intercal Mx) \geq 0$ means after scale, the $Q^\intercal(M^{-1})^\intercal x$'s projection on $\Lambda(Q^\intercal Mx)$ have the same direction with $\Lambda(Q^\intercal Mx)$. Since all Λ_{ii} is non-negative, $\Lambda(Q^\intercal Mx)$ never has the opposite directional component with $Q^\intercal Mx$ after scale, namely,

$$\forall x, (Q^\intercal(M^{-1})^\intercal x)^\intercal(Q^\intercal Mx) \geq 0 \leftrightarrow \forall x, (Q^\intercal(M^{-1})^\intercal x)^\intercal\Lambda(Q^\intercal Mx) \geq 0. \tag{10}$$

Therefore, $\forall x, (Q^\intercal(M^{-1})^\intercal x)^\intercal(Q^\intercal Mx) \geq 0$ can guarantee $\forall x, x^\intercal\widehat{\Sigma}^x x \geq 0$. After computing $(Q^\intercal(M^{-1})^\intercal x)^\intercal(Q^\intercal Mx)$, we can have

$$\forall x, (Q^\intercal(M^{-1})^\intercal x)^\intercal(Q^\intercal Mx) = \|x\|^2, \tag{11}$$

where $\|x\|^2$ represents the length of x in the Euclidean space. So, Inequation 7 holds as long as $\widehat{\Sigma}^x$ is available, equivalently to M^{-1} is available. □

A.2.2 Proof of Theorem 2

Proof of Theorem 2. In this proof, the graph notation follows *Proof of Theorem 1*, and its result that M is invertible. So the augmentation's feature matrix is computed. In the following, we prove that what the extra property of the matrix M guarantees the augmentation graph is undirected.

If the augmentation graph $\widehat{G} = \{\widehat{X}, \widehat{A}\}$ is undirected as the original graph $G = \{X, A\}$, then the adjacency matrix $\widehat{A}$ is symmetric, namely $\widehat{A}^{\mathtt{T}} = \widehat{A}$. Due to $\hat{A} = M^{-1}AM$, we have the following equation:

$$M^{\mathtt{T}}A(M^{-1})^{\mathtt{T}} = M^{-1}AM \tag{12}$$

We rewrite this equation as:

$$A = (M^{\mathtt{T}})^{-1}M^{-1}AM((M^{-1})^{\mathtt{T}})^{-1} = (M^{-1})^{\mathtt{T}}M^{-1}AMM^{\mathtt{T}} \tag{13}$$

Since M is invertible, $(MM^{\mathtt{T}})^{-1}$ exits, and we can rewrite this equation as:

$$A = (MM^{\mathtt{T}})^{-1}A(MM^{\mathtt{T}}). \tag{14}$$

From Equation 14, A should equal to its new similar matrix $(MM^{\mathtt{T}})^{-1}A(MM^{\mathtt{T}})$ with the invertible matrix $MM^{\mathtt{T}}$ for symmetricity. Normally, A will not be identical to its similar matrix, except that M is the diagonal matrix. In this case, we cannot give a property of M that guarantees $MM^{\mathtt{T}}$ is an arbitrary diagonal matrix. If we make this constraint stricter that $MM^{\mathtt{T}}$ is a identity matrix I, namely $MM^{\mathtt{T}} = I$, a special diagonal matrix, then M is an orthonormal matrix ($M^{-1} = M^{\mathtt{T}}$), which guarantee Equation (14) holds.

□

A.3 InfoMax Principle in Contrastive Learning with GTAs

As [34] unifies a broad family of contrastive learning methods on graph-structured data, $\mathcal{L}$ is equivalent to the following dual representation of mutual information:

$$\mathcal{L} = \mathbb{E}_{\mathbb{P}_{\widehat{G}_i}}\left\{-\mathbb{E}_{\mathbb{P}_{(\widehat{G}_j|\widehat{G}_i)}}T(f_1(\widehat{G}_i), f_2(\widehat{G}_j)) + \log(\mathbb{E}_{\mathbb{P}_{\widehat{G}_j}}e^{T(f_1(\widehat{G}_i), f_2(\widehat{G}_j)))}\right\}, \tag{15}$$

where $T(\cdot,\cdot)$ is a learnable score function parameterized with the similarity function $sim(\cdot,\cdot)$, the projection head $g_{\theta_2}(\cdot)$, and temperature factor τ, and the mutual information between $h_i = f_i(\widehat{G}_i)$ and $h_i = f_j(\widehat{G}_j)$. In the implementation, $f_{\theta_1} = f_1 = f_2$. Based on the property of the dual representation of mutual information [1], Equation 15 can be rewritten as:

$$\min_{\theta_1\in\Theta_1,\theta_2\in\Theta_2}\mathcal{L} = \min_{\theta_1\in\Theta_1}\min_{\theta_2\in\Theta_2}\mathbb{E}_{\mathbb{P}_{\widehat{G}_i}}\left\{-\mathbb{E}_{\mathbb{P}_{(\widehat{G}_j|\widehat{G}_i)}}T(f_1(\widehat{G}_i), f_2(\widehat{G}_j)) + \log(\mathbb{E}_{\mathbb{P}_{\widehat{G}_j}}e^{T(f_1(\widehat{G}_i), f_2(\widehat{G}_j)))}\right\} \tag{16}$$

$$\leftrightarrow \max_{\theta_1\in\Theta_1} I(h_i, h_j), \tag{17}$$

When the parameters of the GNN-based encoder are considered, the functional space consists of $\Theta = \Theta_1 \cup \Theta_2$. So Equation 15 can be rewritten as:

$$\min_{\theta\in\Theta}\mathcal{L} = \min_{\theta\in\Theta}\mathbb{E}_{\mathbb{P}_{\widehat{G}_i}}\left\{-\mathbb{E}_{\mathbb{P}_{(\widehat{G}_j|\widehat{G}_i)}}T'(\widehat{G}_i, \widehat{G}_j) + \log(\mathbb{E}_{\mathbb{P}_{\widehat{G}_j}}e^{T'(\widehat{G}_i, \widehat{G}_j)})\right\} \tag{18}$$

$$\leftrightarrow I(\widehat{G}_i; \widehat{G}_j) \to H(G), \tag{19}$$

where $T(\cdot,\cdot)$ is a learnable score function parameterized with the similarity function $sim(\cdot,\cdot)$, the projection head $g_{\theta_2}(\cdot)$, temperature factor τ and the GNN-based encoder $f_{\theta_1}(\cdot)$. Therefore, we will have the following relation:

$$H(G) \leftarrow I(\widehat{G}_i; \widehat{G}_j) \leftrightarrow \min_{\theta\in\Theta}\mathcal{L} \leftrightarrow \max_{\theta_1\in\Theta_1} I(h_i; h_j). \tag{20}$$

This contrastive learning tends to achieve the transformation invariance, so we will have the following equivalence:

$$\max_{\theta_1\in\Theta_1} I(h; h_i) \leftrightarrow \max_{\theta_1\in\Theta_1} I(h; h_j) \leftrightarrow \max_{\theta\in\Theta_1} I(h_i; h_j), \tag{21}$$

where h is the representation of the original graph, and these quantities will converge to $H(G)$. It is natural that G contains more information of G compared to h_i, h_j. Simultaneously, the learning will enforce $I(h; G)$ tends to $H(G)$ with parameters $\theta_1 \in \Theta_1$, namely

$$\min_{\theta \in \Theta} \mathcal{L} \leftrightarrow \max_{\theta_1 \in \Theta_1} I(h; G). \tag{22}$$

Our augmentation technique makes the contrastive learning not only maximize the mutual information between representation vectors of two views but also implicitly achieves the InfoMax between the input and the latent representation. This InfoMax will enforce the encoder to extract as much information as possible in the input [18], and is widely implemented in self-supervised representation learning with good performance [11, 1, 30, 20].

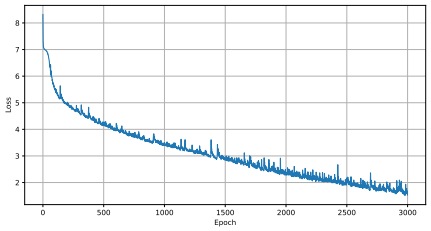

Figure 2: Pre-train curve with backward transformation on PROTEINS

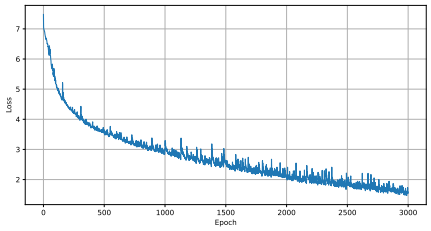

Figure 3: Pre-train curve without backward transformation on PROTEINS

A.4 Edge Attribute Augmentation

We also extend GTA on the edge attribute on MUTAG. Table 4 shows the improvement of GTA with the edge attribute.

Table 4: Comparing GTA with/without the edge attribute on MATUG.

Consider Edge Attributes	Accuracy
×	86.80±1.34
✓	86.30±0.99