GNN-Transformer Cooperative Architecture for Trustworthy Graph Contrastive Learning

Jianqing Liang, Xinkai Wei, Min Chen, Zhiqiang Wang, Jiye Liang*

Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 030006, Shanxi, China
liangjq@sxu.edu.cn, {202322407046,202222409003}@email.sxu.edu.cn, {wangzq,ljy}@sxu.edu.cn

Abstract

Graph contrastive learning (GCL) has become a hot topic in the field of graph representation learning. In contrast to traditional supervised learning relying on a large number of labels, GCL exploits augmentation strategies to generate multiple views and positive/negative pairs, both of which greatly influence the performance. Unfortunately, commonly used random augmentations may disturb the underlying semantics of graphs. Moreover, traditional GNNs, a type of widely employed encoders in GCL, are inevitably confronted with over-smoothing and over-squashing problems. To address these issues, we propose GNN-Transformer Cooperative Architecture for Trustworthy Graph Contrastive Learning (GTCA), which inherits the advantages of both GNN and Transformer, incorporating graph topology to obtain comprehensive graph representations. Theoretical analysis verifies the trustworthiness of the proposed method. Extensive experiments on benchmark datasets demonstrate state-of-the-art empirical performance.

Code — https://github.com/a-hou/GTCA

Introduction

Compared with traditional supervised learning, self-supervised learning eliminates the dependence on labels. As one of the most representative methods of self-supervised learning, contrastive learning has been widely applied in the graph domain, i.e., Graph Contrastive Learning (GCL). GCL utilizes multiple views and positive/negative pairs for node/-graph representations (Zhu et al. 2020; Peng et al. 2020; Zhu, Sun, and Koniusz 2021; Yin et al. 2022; Liu et al. 2023), the performance of which are influenced by augmentation strategies, graph encoders and loss functions.

Different from computer vision domain where augmented images obtained with cropping, rotation, and other strategies usually retain the same semantic as the original images (Shorten and Khoshgoftaar 2019), graph augmentation will disturb the underlying semantics of graphs, which may affect the performance on downstream tasks (Li et al. 2022). Figure 1 (a) shows that even after randomly cropping or changing the color of an image, the underlying semantic still remains the same, which we can still easily recognize. Figure 1 (b) presents that removal of an atom or edge from the molecules may change their structures, resulting in different compounds. In order to address this issue, a number of augmentation strategies on graphs spring up. GCA (Zhu et al. 2021) introduces learnable adaptive augmentation strategies with the structural and attribute characteristics of graphs for GCL. GCS (Wei et al. 2023) employs gradient-based strategies to differentiate between semantics and environment within graphs. It perturbs the environment while maintaining the semantics to obtain positive pairs, and perturbs the semantics while preserving the environment to obtain negative pairs. However, effective augmentation strategies must achieve a balance between the preservation of graph structural information and the generation of meaningful views (Suresh et al. 2021). Poorly designed augmentations may result in redundant or irrelevant information generation, leading to inferior performance.

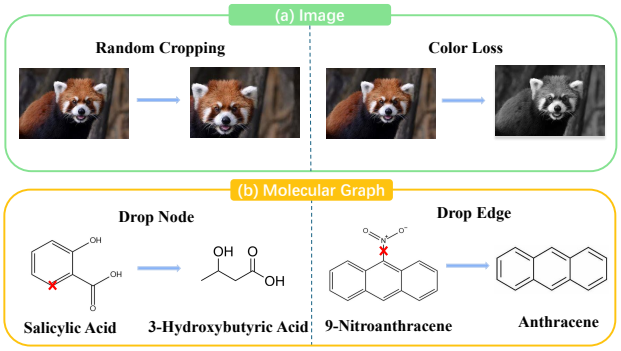

Figure 1: Augmentations on image (a) keep the semantic information, while augmentations on graphs (b) change the underlying semantic information.

Most of the existing GCL methods exploit GNNs as graph encoders (Veličković et al. 2019; Hassani and Khasahmadi 2020; Mo et al. 2022; Lee, Lee, and Park 2022; Shen et al. 2023). GNNs iteratively propagate, transform, and aggregate representations from topological neighbors to progressively update node features (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Zhang et al. 2022a). With multi-layer stacking, these local features are progressively integrated to form a comprehensive perception of the en-

*Corresponding author.

tire graph structure. While this mechanism is particularly effective in handling local dependencies, especially in capturing both direct and indirect relation between nodes, it fails to capture complex global structures and long-range dependencies within graphs (Xu et al. 2019; Garg, Jegelka, and Jaakkola 2020). In contrast, Graph Transformers (GTs) are capable of effectively capturing global dependencies between nodes with self-attention mechanisms (Veličković et al. 2018; Yun et al. 2019; Kreuzer et al. 2021). GTs can handle long-range dependencies within graphs, rather than being confined to local neighborhoods. By calculating attention scores between nodes, GTs update node representations with all other nodes in the graph, thereby effectively integrating global information. This mechanism enables GTs to model complex interactions and long-term dependencies between nodes. Therefore, they are more flexible and accurate in handling dependencies between distant nodes (Yun et al. 2019), which is ignored by existing GCL methods in choosing graph encoders to learn node/graph representations. However, the quadratic complexity of GTs limits the scalability. Currently, some works introduce linear attention mechanisms to improve scalability for large-scale graphs (Wu et al. 2022, 2024; Liang, Chen, and Liang 2024).

Existing GCL methods mostly use InfoNCE as their loss functions (Zhu et al. 2020; Zhang et al. 2022b, 2023), where each anchor point has only one positive pair. Therefore, when calculating the loss, nodes from the same class as the anchor point are pushed away. This may lead to ignorance of the potential information from similar nodes when handling node feature similarity tasks (Shen et al. 2023). While this design effectively encourages the model to distinguish between positive and negative pairs, it may lead to insufficient aggregation of nodes from the same class, thereby affecting the ability to represent graph data and capture graph structures.

In order to address the issues above, we propose a novel method called GNN-Transformer Cooperative Architecture for Trustworthy Graph Contrastive Learning (GTCA). GTCA utilizes GCN and NodeFormer, a linear GT as graph encoders to generate node representation views without random augmentation strategies. In addition, it utilizes topological property of graphs to generate topology structure view. Furthermore, we design a novel loss function to exploit the intersection of the node representation views and topology structure view as positive pairs. The contributions are summarized follows:

- We utilize both GCN and NodeFormer as graph encoders to capture comprehensive perception of graphs, which has not been well explored in the field of GCL to our best knowledge.
- We generate topology structure view with topological property of graphs. Therefore, we introduce an augmentation-free strategy for GCL, which can enhance efficiency and avoid the potential risks of disturbing the underlying semantics of graphs.
- We design a novel contrastive loss function with multiple positive pairs for each node. Theoretical analysis and experimental results demonstrate the effectiveness of the GTCA method.

Related Work

Graph Contrastive Learning. Current GCL methods can be categorized into three mainstream paradigms: DGI framework (Veličković et al. 2019), InfoNCE framework (Oord, Li, and Vinyals 2018) and BGRL framework (Thakoor et al. 2022).

DGI aggregates the features of all nodes in the graph to obtain a global feature, then maximizes the mutual information between the global feature and the node features to learn node representations. Based on this framework, MVGRL (Hassani and Khasahmadi 2020) further advances the development of unsupervised graph contrastive learning through the adoption of graph diffusion and subgraph sampling methods. In spite of the competitive performance, global features in these methods may be insufficient to retain node-level embedding information.

GRACE (Zhu et al. 2020) first utilizes the InfoNCE loss to maximize the mutual information between positive pairs under two augmented views, while minimizing the mutual information between negative pairs to learn node representations. This principle is widely used in both node classification (Zhu et al. 2021; Zhang et al. 2023; Shen et al. 2023) and graph classification (You et al. 2020; Hassani and Khasahmadi 2020; Wei et al. 2023) tasks. While these methods achieve significant success, they still suffer from sampling bias issues.

BGRL uses the BYOL (Grill et al. 2020) method to remove negative pairs, thereby reducing computational complexity. However, it relies on graph augmentation to obtain positive pairs, which may disturb the underlying semantics of the graph. AFGRL (Lee, Lee, and Park 2022) extends BGRL (Thakoor et al. 2022) with an augmentation-free strategy and utilizes k-means for positive pair sampling. However, the randomness of k-means may influence the performance of down-stream tasks.

Graph Augmentation. The existing graph augmentation strategies include node dropping (You et al. 2020), edge perturbation (Qiu et al. 2020; Zhang et al. 2023), attribute masking (Zhu et al. 2021; Zhang et al. 2022b) and subgraph extraction (Hassani and Khasahmadi 2020). GRACE (Zhu et al. 2020) employs random edge perturbation and node feature masking to generate two views. GCA (Zhu et al. 2021) proposes an adaptive augmentation strategy which integrates both structural and attribute information. Similarly, GCS (Wei et al. 2023) utilizes the structural and semantic information to achieve adaptive graph augmentation. Furthermore, CI-GCL (Tan et al. 2024) proposes a community invariant GCL framework to maintain graph community structure. Despite the considerable success, bias is introduced due to the disturbance of the semantic.

Method

In this section, we provide a detailed description of GTCA, including graph encoders, node sampling and contrastive loss function. Figure 2 shows the model architecture of GTCA.

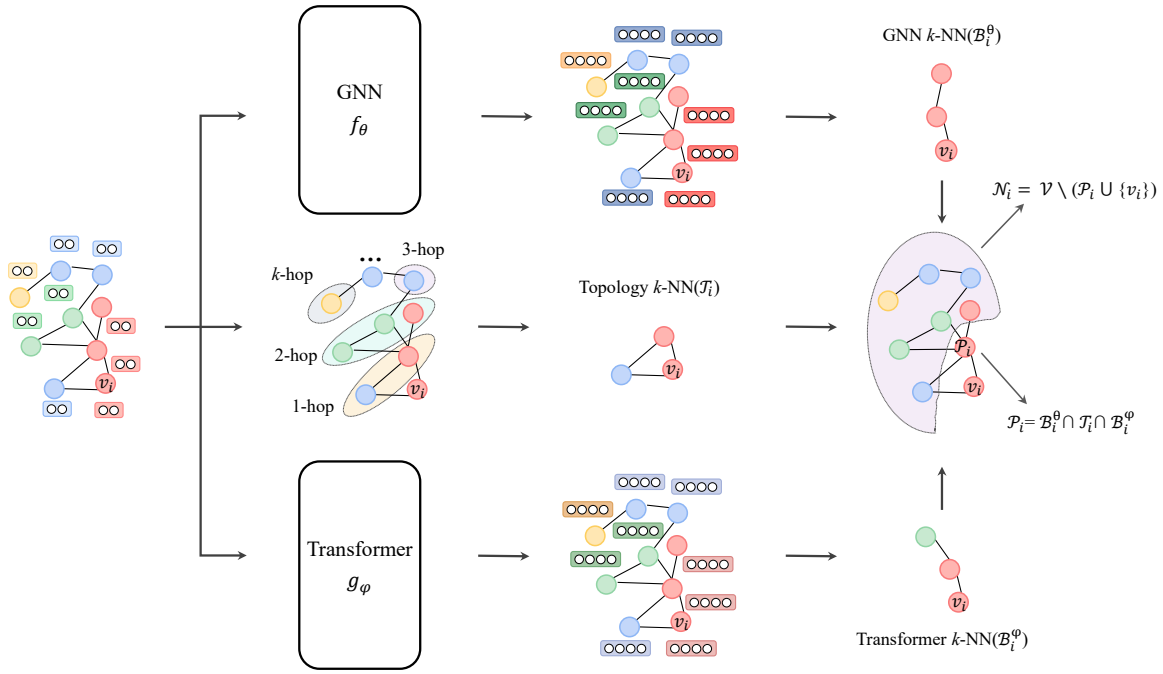

Figure 2: The architecture of GTCA. Given a graph, the GNN encoder f_θ and Transformer encoder g_φ generate node embeddings $\boldsymbol{H}_\theta$ and $\boldsymbol{H}_\varphi$. We apply two node embeddings to obtain k-NNs of v_i, i.e., $\mathcal{B}_i^\theta$, $\mathcal{B}_i^\varphi$, intersects which with the topological k-NNs $\mathcal{T}_i$ to obtain positive and negative pairs for node v_i separately, i.e., $\mathcal{P}_i$ and $\mathcal{N}_i$. Finally, we employ the contrastive loss to achieve that the anchor nodes close to positive pairs and far from negative pairs.

Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph, where $\mathcal{V} = \{v_1, \cdots, v_N\}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent the node set and the edge set respectively. We denote the feature matrix and the adjacency matrix as $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and $\boldsymbol{A} \in \{0,1\}^{N \times N}$, where $\boldsymbol{x}_i \in \mathbb{R}^F$ is the feature of v_i, and $\boldsymbol{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$. We aim to learn a GNN encoder $f_\theta(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N \times D}$ and a Transformer encoder $g_\varphi(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N \times D}$. They take the node features and structures as input, and generate node embeddings in low dimensionality, i.e., $D \ll F$. We denote $\boldsymbol{H}_\theta = f_\theta(\boldsymbol{X}, \boldsymbol{A})$ as GNN node representations and $\boldsymbol{H}_\varphi = g_\varphi(\boldsymbol{X}, \boldsymbol{A})$ as Transformer node representations, where $\boldsymbol{h}_i^\theta$ is GNN embedding of node v_i, and $\boldsymbol{h}_i^\varphi$ is the Transformer embedding of node v_i .

Graph Encoders

We employ GCN (Kipf and Welling 2017) as an encoder to capture the local structural information of the nodes. GCN utilizes a series of graph convolution layers to aggregate information from the neighbors. We update each layer according to the following equation:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\, \boldsymbol{H}^{(l)}\, \boldsymbol{W}^{(l)}\right) \tag{1}$$

where $\boldsymbol{H}^{(l)}$ denotes the feature matrix at layer l, $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix with self-loops, $\boldsymbol{W}^{(l)}$ represents the learnable weight matrix for layer l, and σ is a non-linear activation function, i.e., ReLU. We apply a 2-layer GCN to obtain $\boldsymbol{H}_\theta$.

Meanwhile, we utilize the NodeFormer (Wu et al. 2022) as the Transformer encoder with the following equation:

$$\mathbf{h}_i^{(l+1)} \approx \sum_{j=1}^{N} \frac{\phi(\mathbf{q}_i/\sqrt{\tau})^\top \phi(\mathbf{k}_j/\sqrt{\tau}) e^{g_j/\tau}}{\sum_{w=1}^{N} \phi(\mathbf{q}_i/\sqrt{\tau})^\top \phi(\mathbf{k}_w/\sqrt{\tau}) e^{g_w/\tau}} \cdot \mathbf{v}_j \tag{2}$$

where $\phi(\cdot)$ denotes a kernel function and $\mathbf{q}_i = \boldsymbol{W}_Q^{(l)} \mathbf{h}_i^{(l)}$, $\mathbf{k}_j = \boldsymbol{W}_K^{(l)} \mathbf{h}_j^{(l)}$ and $\mathbf{v}_j = \boldsymbol{W}_V^{(l)} \mathbf{h}_j^{(l)}$. We can obtain $\boldsymbol{H}_\varphi$ with NodeFormer.

Contrastive learning is an important tool for multi-view learning. Inspired by (Huang et al. 2021), we present our finding with respect to multi-view learning.

Theorem 1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph dataset of N nodes drawn i.i.d. according to an unknown distribution $\mathcal{D}$. Let $\mathcal{M}, \mathcal{N}$ be two distinct subsets of $[K]$, where K is the number of multiple views. Assume empirical risk minimizers $(\hat{h}_\mathcal{M}, \hat{g}_\mathcal{M})$ and $(\hat{h}_\mathcal{N}, \hat{g}_\mathcal{N})$, training with the $\mathcal{M}$ and $\mathcal{N}$ views separately. Then, for all $1 > \delta > 0$, with probability at least $1 - \frac{\delta}{2}$:*

$$r\left(\hat{h}_\mathcal{M} \circ \hat{g}_\mathcal{M}\right) - r\left(\hat{h}_\mathcal{N} \circ \hat{g}_\mathcal{N}\right) \leq$$

$$\gamma_\mathcal{G}(\mathcal{M}, \mathcal{N}) + 8L\mathfrak{R}_N(\mathcal{H} \circ \mathcal{G}_\mathcal{M}) + \frac{4C}{\sqrt{N}} + 2C\sqrt{\frac{2\ln(2/\delta)}{N}} \tag{3}$$

where

$$\gamma_{\mathcal{G}}(\mathcal{M},\mathcal{N}) \triangleq \eta(\hat{g}_{\mathcal{M}}) - \eta(\hat{g}_{\mathcal{N}}) \qquad \square \tag{4}$$

Remark. First, $\gamma_{\mathcal{G}}(\mathcal{M},\mathcal{N})$ of (4) trades off the quality between latent representations learning from $\mathcal{M}$ and $\mathcal{N}$ views with respect to the graph dataset $\mathcal{G}$. Theorem 1 bounds the difference of population risk training with two different subsets of views with $\gamma_{\mathcal{G}}(\mathcal{M},\mathcal{N})$, which validates our intuition that more views is superior. Second, for the commonly used function classes, Radamacher complexity for a node of size N, $\mathfrak{R}_N(\mathcal{F})$ is generally bounded by $\sqrt{C(\mathcal{F})/N}$, where $C(\mathcal{F})$ represents the intrinsic property of function class $\mathcal{F}$. Third, (3) can be written as $\gamma_{\mathcal{G}}(\mathcal{M},\mathcal{N}) + \mathcal{O}(\sqrt{\frac{1}{N}})$ in order terms. This indicates that as the number of node increases, the performance with different views mainly depends on its latent representation quality.

Node Sampling

We compute the cosine similarity matrix of $\boldsymbol{H}_\theta$ and $\boldsymbol{H}_\varphi$ separately to obtain the k-NN sets. Specifically, we obtain the k-NN node set $\mathcal{B}_i^\theta$ of node v_i with the cosine similarity between $\boldsymbol{h}_i^\theta$ and $\boldsymbol{H}_\theta$. Similarly, we obtain the k-NN node set $\mathcal{B}_i^\varphi$ for node v_i with the cosine similarity between $\boldsymbol{h}_i^\varphi$ and $\boldsymbol{H}_\varphi$. Furthermore, we compute the topological k-NN matrix via nodes sorting according to the number of hops from other nodes to node v_i in ascending order, resulting in the topological k-NN set $\mathcal{T}_i$ for node v_i. Finally, we obtain the positive set $\mathcal{P}_i$ for node v_i from the intersection of $\mathcal{B}_i^\theta$, $\mathcal{B}_i^\varphi$, and $\mathcal{T}_i$:

$$\mathcal{P}_i = \mathcal{B}_i^\theta \cap \mathcal{B}_i^\varphi \cap \mathcal{T}_i \tag{5}$$

Meanwhile, we treat all other nodes as negative pairs.

$$\mathcal{N}_i = \mathcal{V} \setminus (\mathcal{P}_i \cup \{v_i\}) \tag{6}$$

Compared with existing augmentation-based contrastive learning models, GTCA has several advantages. First, GTCA discards random or heuristic augmentation strategies which may disturb the graph structure. Second, in contrast to existing GCL methods that utilize shared-weight GNNs, GTCA uses GNN and Transformer encoders to learn diverse graph feature representations.

Contrastive Loss Function

According to (Oord, Li, and Vinyals 2018), the InfoNCE is a lower bound of the true Mutual Information (MI). Recent GCL methods use InfoNCE as the loss function (Zhu et al. 2020; Zhang et al. 2023; Guo et al. 2023; Yu et al. 2024). Figure 3 gives an example of the InfoNCE loss. For a given anchor node, InfoNCE uses only one node to generate a positive pair and utilizes all the other nodes to generate negative pairs. This may lead to ignorance of the potential information of similar nodes.

To address this issue, we design a novel contrastive loss function. Let $\boldsymbol{h}_i^\theta$ and $\boldsymbol{h}_i^\varphi$ denote the ℓ_2-normalized GCN and NodeFormer embeddings of v_i respectively. Select $\boldsymbol{h}_i^\theta$ as the anchor node, its positives come from three sources: (1) its NodeFormer embedding $\boldsymbol{h}_i^\varphi$; (2) GCN embeddings $\{\boldsymbol{h}_j^\theta \mid v_j \in \mathcal{P}_i\}$; (3) NodeFormer embeddings $\{\boldsymbol{h}_j^\varphi \mid v_j \in \mathcal{P}_i\}$. That is, the total number of positive pairs associated with node v_i is $2|\mathcal{P}_i|+1$. Ultimately, the contrastive loss function is as follows:

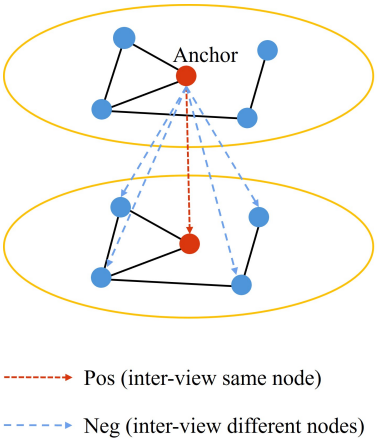

Figure 3: The red nodes denote the positive pair and the blue dotted lines with arrows are negative pairs of the anchor.

$$\ell(\boldsymbol{h}_i^\theta) = -\log \frac{e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_i^\varphi)/\tau} + \sum_{j\in\mathbf{P}_i}(e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\theta)/\tau} + e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\varphi)/\tau})}{e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_i^\varphi)/\tau} + \sum_{j\neq i}(e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\theta)/\tau} + e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\varphi)/\tau})} \tag{7}$$

where $s(\cdot,\cdot)$ is the cosine similarity and τ is a temperature parameter. For a given query node $v_i \in \mathcal{V}$, we compute the cosine similarity as follows:

$$s(\boldsymbol{h}_i^\theta, \boldsymbol{h}_i^\varphi) = \frac{\boldsymbol{h}_i^\theta \cdot \boldsymbol{h}_i^\varphi}{\|\boldsymbol{h}_i^\theta\|\|\boldsymbol{h}_i^\varphi\|}, \forall v_j \in \mathcal{V} \tag{8}$$

The last two terms of the denominator of Equation (7) are decomposed as:

$$\sum_{j\neq i} e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\theta)/\tau} = \sum_{j\in\mathbf{P}_i} e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\theta)/\tau} + \sum_{j\in\mathbf{N}_i} e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\theta)/\tau} \tag{9}$$

$$\sum_{j\neq i} e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\varphi)/\tau} = \sum_{j\in\mathbf{P}_i} e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\varphi)/\tau} + \sum_{j\in\mathbf{N}_i} e^{s(\boldsymbol{h}_i^\theta,\boldsymbol{h}_j^\varphi)/\tau} \tag{10}$$

The minimization of Equation (7) will pull positive pairs closer and push negative pairs away. Since the two views are symmetric, we define the loss for the feature embedding $\boldsymbol{h}_i^\varphi$ corresponding to node v_i in the other view similarly to Equation (7). Then, the final loss function is as follows:

$$\mathcal{L} = \frac{1}{2N}\sum_{i=1}^{N}\left[\ell(\boldsymbol{h}_i^\theta) + \ell(\boldsymbol{h}_i^\varphi)\right] \tag{11}$$

Proposition 1. *Given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, encoders f_θ, g_φ and a contrastive loss function $\mathcal{L}$ defined in Equation (11). $\mathcal{B}_i^\theta$, $\mathcal{B}_i^\varphi$, and $\mathcal{T}_i$ are k-NNs of node v_i, $i = 1, 2, \cdots, N$ under two node representation views and one topology structure view. Equation (5) generates the most trustworthy positive pairs for GCL.*

Algorithm 1: GTCA

Input: The adjacency matrix $\boldsymbol{A}$, the feature matrix $\boldsymbol{X}$, and the number of training epochs J

Output: Feature matrix $\boldsymbol{H}$

for epoch in 1 to J **do**
  Generate GNN embeddings $\boldsymbol{H}_\theta$ and NodeFormer embeddings $\boldsymbol{H}_\varphi$ with GNN encoder f_θ , NodeFormer encoder g_φ, adjacency matrix $\boldsymbol{A}$ and feature matrix $\boldsymbol{X}$;
  Generate GNN k-NN node set $\mathcal{B}_i^\theta$, Nodeformer k-NN node set $\mathcal{B}_i^\varphi$ and topological k-NN set $\mathcal{T}_i$ with $\boldsymbol{H}_\theta$, $\boldsymbol{H}_\varphi$ and adjacency matrix $\boldsymbol{A}$;
  Calculate positive pairs $\mathcal{P}_i$ and negative pairs $\mathcal{N}_i$, $i = 1, \cdots, N$ with Equation (5) and Equation (6);
  Compute loss $\mathcal{L}$ with Equation (11);
  Apply gradient descent to minimize $\mathcal{L}$ and update parameters;
end for
Calculate the final output feature matrix with Equation (12);
teturn $\boldsymbol{H}$ for downstream tasks;

Proof. It is obvious that the intersection set contains the minimum number of positive pairs with specific $\mathcal{B}_i^\theta$, $\mathcal{B}_i^\varphi$, and $\mathcal{T}_i$. The numerator of Equation (7) sums the exponential calculation with respect to $\mathcal{P}_i$, whereas the denominator sums the exponential calculation with respect to both $\mathcal{P}_i$ and $\mathcal{N}_i$. Therefore, when calculating $\mathcal{P}_i$ with Equation (5), we get the lower bound of the numerator, and also the upper bound of the contrastive loss in Equation (7). When we minimize the upper bound of the loss in Equation (11), trustworthy GCL can be achieved. □

At each training epoch, GTCA first generates two node feature representation matrices $\boldsymbol{H}_\theta$ and $\boldsymbol{H}_\varphi$ with GCN encoder f_θ and NodeFormer encoder g_φ, respectively. Then, we calculate the positive and negative pairs with Equation (5) and Equation (6). Finally, we minimize the objective in Equation (11) to update the parameters of f_θ and g_φ. We obtain two trained feature matrices, $\boldsymbol{H}'_\theta$ and $\boldsymbol{H}'_\varphi$. The final output feature matrix $\boldsymbol{H}$ is obtained with the normalized feature matrices $\boldsymbol{H}'_\theta$ and $\boldsymbol{H}'_\varphi$ according to a weight parameter λ:

$$\boldsymbol{H} = \lambda \cdot \boldsymbol{H}'_\theta + (1 - \lambda) \cdot \boldsymbol{H}'_\varphi \tag{12}$$

The final output feature matrix $\boldsymbol{H}$, a linear combination of $\boldsymbol{H}'_\theta$ and $\boldsymbol{H}'_\varphi$, is applied to downstream node classification tasks. λ is a tunable weight parameter. Algorithm 1 summarizes the overall procedure of GTCA.

Experiments

Datasets

To validate the effectiveness of the GTCA method, we perform extensive experiments on 5 benchmark datasets for node classification including a commonly used citation network, i.e., Cora (Sen et al. 2008), a reference network constructed based on Wikipedia, i.e., Wiki-CS (Mernyei and Cangea 2020), a co-authorship network, i.e., Coauthor-CS (Shchur et al. 2018), and two product co-purchase networks, i.e., Amazon-Computers and Amazon-Photo (Shchur et al. 2018). We list the detailed statistics of these datasets in Table 1.

Datasets	# Nodes	# Edges	# Features	# Labels
Cora	2,708	5429	1,433	7
Wiki-CS	11,701	216,123	300	10
Coauthor-CS	18,333	81894	6,805	15
Amazon-Computers	13,752	245,861	767	10
Amazon-Photo	7,650	119081	745	8

Table 1: Statistics of the datasets.

Baselines

We compare GTCA with 12 state-of-the-art methods, including 2 semi-supervised GNNs, i.e., GCN (Kipf and Welling 2017) and GAT (Veličković et al. 2018), 2 semi-supervised GCL methods, i.e., CGPN (Wan et al. 2021b) and CG3 (Wan et al. 2021a), and 8 self-supervised GCL methods, i.e., DGI (Veličković et al. 2019), GMI (Peng et al. 2020), MVGRL (Hassani and Khasahmadi 2020), GRACE (Zhu et al. 2020), GCA (Zhu et al. 2021), SUGRL (Mo et al. 2022), AFGRL (Lee, Lee, and Park 2022) and NCLA (Shen et al. 2023).

Experimental Settings

We employ the above methods for node classification. For Cora dataset, we follow (Yang, Cohen, and Salakhudinov 2016) to randomly select 20 nodes per class for training, 500 nodes for validation, and the remaining nodes for testing. For Wiki-CS, Coauthor-CS, Amazon-Computers and Amazon-Photo datasets, we follow (Liu, Gao, and Ji 2020) to randomly select 20 nodes per class for training, 30 nodes per class for validation, and the remaining nodes for testing. We perform 20 random splits of training, validation, and testing on each dataset and report the average performance of all algorithms across these splits. All experiments are implemented in PyTorch and conducted on a server with NVIDIA GeForce 3090 (24GB memory each). Table 2 shows the hyperparameter settings of GTCA on 5 datasets.

Datasets	k	E	λ	lr
Cora	520	440	0.7	0.005
Wiki-CS	500	400	0.8	0.001
Coauthor-CS	240	420	0.4	0.001
Amazon-Computers	550	512	0.8	0.001
Amazon-Photo	510	512	0.7	0.001

Table 2: Hyperparameter settings of GTCA on 5 datasets. lr is the learning rate.

Methods	Cora	Wiki-CS	Coauthor-CS	Amazon-Computers	Amazon-Photo
GCN	79.6±1.8	67.3±1.5	90.0±0.6	76.4±1.8	86.3±1.6
GAT	81.2±1.6	68.6±1.9	90.9±0.7	77.9±1.8	86.5±2.1
CGPN	74.0±1.7	66.1±2.1	83.5±1.4	74.7±1.3	84.1±1.5
CG3	80.6±1.6	68.0±1.5	90.6±1.0	77.8±1.7	89.4±1.9
DGI	82.1±1.3	69.1±1.4	92.0±0.5	78.8±1.1	83.5±1.2
GMI	79.4±1.2	67.8±1.8	88.5±0.8	76.1±1.2	86.7±1.5
MVGRL	82.4±1.5	69.2±1.2	91.5±0.6	78.7±1.7	89.7±1.2
GRACE	79.6±1.4	67.8±1.4	90.0±0.7	76.8±1.7	87.9±1.4
GCA	79.0±1.4	67.6±1.3	90.9±1.1	76.9±1.4	87.0±1.9
SUGRL	81.3±1.2	68.7±1.1	91.2±0.9	78.2±1.2	90.5±1.9
AFGRL	78.6±1.3	68.0±1.7	91.4±0.6	77.7±1.1	89.2±1.1
NCLA	82.2±1.6	**70.3±1.7**	91.5±0.7	**79.8±1.5**	90.2±1.3
GTCA	**82.5±1.3**	69.7±1.5	**92.5±0.6**	79.2±1.4	**90.5±1.2**

Table 3: Node classification accuracy (%) comparison on 5 datasets.

Node Classification Results

Table 3 shows node classification accuracy on 5 benchmark graph datasets. On the whole, GTCA demonstrates superior performance across all 5 datasets. GTCA ranks first on three datasets and ranks second on the other two datasets.

The remarkable performance of GTCA can be attributed to two key aspects. First, GTCA introduces an augmentation-free strategy, which avoids the potential risk of disturbing the underlying semantics of graphs. Second, GTCA employs a novel sampling strategy to generate trustworthy positive pairs and negative pairs.

Ablation Study

Table 4 presents the ablation study results of GTCA and its variants without $\mathcal{T}_i$ and with different graph encoders including GNNs f_θ, f_φ and NodeFormers g_θ, g_φ. We observe that there is an obvious decline in performance without topology structure view on 5 datasets. This indicates that topological information is essential for enhancing node classification accuracy. In terms of graph encoders, compared with GTCA, the use of either GCN-GCN or NodeFormer-NodeFormer results in inferior performance, which validates the reasonability of the module design. Meanwhile, we can observe that GTCA_g_θ, g_φ ranks recond on the Coauthor-CS dataset, which has the largest number of nodes among the 5 datasets, due to the fact that Transformer-based models can effectively capture long-range dependencies.

	Cora	Wiki-CS	Coauthor-CS	Amazon-Computers	Amazon-Photo
GTCA w/o $\mathcal{T}_i$	79.8	69.0	92.0	75.7	86.1
GTCA_f_θ, f_φ	80.4	69.1	90.0	78.1	90.1
GTCA_g_θ, g_φ	77.6	68.8	92.1	76.9	88.8
GTCA	**82.5**	**69.7**	**92.5**	**79.2**	**90.5**

Table 4: Node classification accuracy (%) of different components on 5 datasets.

Graph Encoders Analysis

Figure 4 shows that with a single graph encoder, correct ratio of positive pairs is lower than 40%. With both GCN and NodeFormer as graph encoders, the correct ratio of positive pairs varies from 30% to 60%. When GTCA exploits intersection of the k-NN neighborhood of GCN, Nodeformer node representation and topology structure views, the correct ratio of positive pairs exceeds 80%. This highlights the soundness of the current graph encoder design.

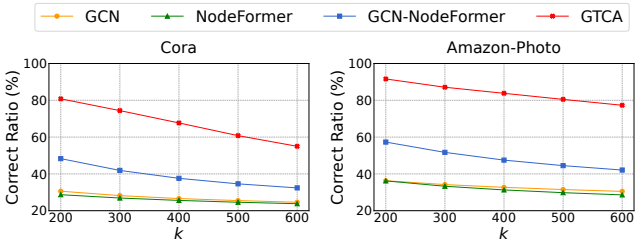

Figure 4: Correct ratio of positive pairs with various k values on Cora and Amazon-Photo datasets.

Sensitivity Analysis

Figure 5 illustrates the node classification accuracy of GTCA on Cora and Amazon-Photo datasets with various hyperparameters, respectively. The optimal range of k is between 300 and 600. When k is too small, GTCA has inferior performance due to a scarcity of positive pairs. Conversely, a large k introduces too much redundant information. Thus, it is necessary to search for a proper k. In addition, increasing embedding dimension E improves classification accuracy. Moreover, the hyperparameter λ is crucial for model performance. Initially, an increase in λ improves node classification accuracy. However, when λ exceeds 0.7, the accuracy will decrease.

Embeddings Visualization

To provide a more intuitive presentation of the node embeddings, we utilize PCA (Abdi and Williams 2010) to visual-

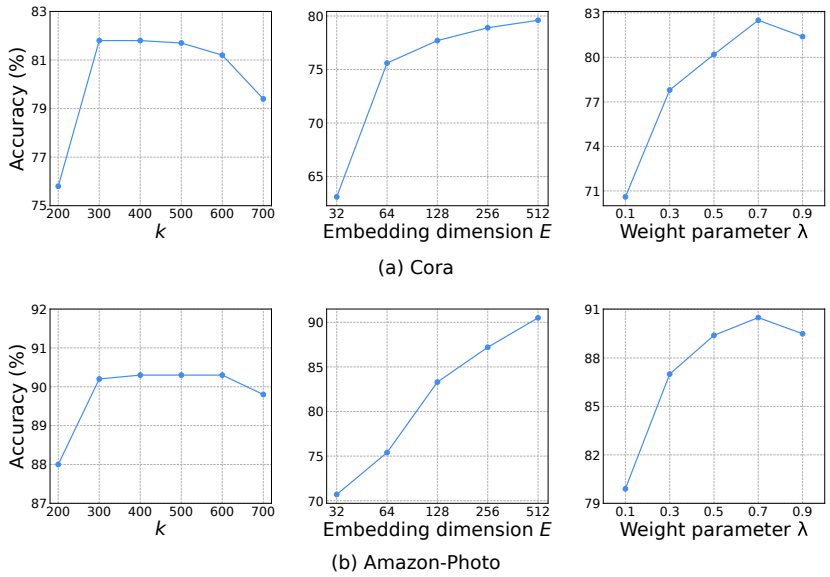

Figure 5: Accuracy vs. hyperparameters k, E and λ on Cora and Amazon-Photo datasets.

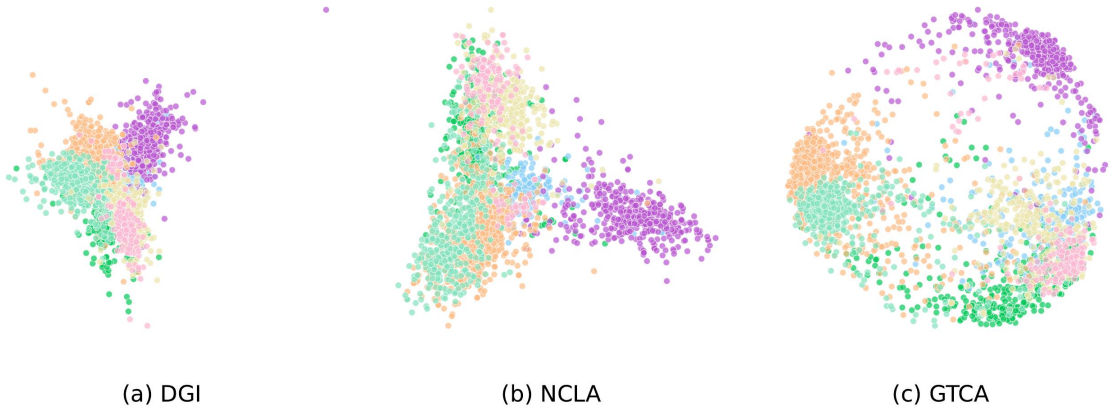

Figure 6: Visualization of DGI, NCLA and GTCA embeddings on Cora dataset with PCA.

ize the node embeddings of DGI in Figure 6 (a), NCLA in Figure 6 (b) and GTCA in Figure 6 (c). Different colors represent different categories. Compared with DGI and NCLA, GTCA can distinguish different classes of nodes much more effectively.

Conclusion

Most of the existing GCL methods utilize graph augmentation strategies, which may perturb the underlying semantics of graphs. Furthermore, they utilize GNNs as graph encoders, which inevitably results in the occurrence of over-smoothing and over-squashing issues. To address these issues, we propose GNN-Transformer Cooperative Architecture for Trustworthy Graph Contrastive Learning (GTCA). GTCA uses GCN and NodeFormer as encoders to generate node representation views. In addition, it utilizes topological property of graphs to generate the topology structure views. Theoretical analysis and experimental results demonstrate the effectiveness of GTCA.

While GTCA has been proved effective, it still faces challenges in terms of computational complexity. Specifically, its quadratic complexity poses a significant computational burden when handling large-scale graph data. In future work, we aim to explore more efficient techniques, such as parallel processing and distributed systems.

Acknowledgements

This work is supported by the National Science and Technology Major Project (2020AAA0106102) and National Natural Science Foundation of China (No.62376142, U21A20473, 62306205).

References

Abdi, H.; and Williams, L. J. 2010. Principal Component Analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 433–459.

Garg, V.; Jegelka, S.; and Jaakkola, T. 2020. Generalization and Representational Limits of Graph Neural Networks. In *Proceedings of International Conference on Machine Learning*, 3419–3430.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. 2020. Bootstrap Your Own Latent-A New Approach to Self-Supervised Learning. *Advances in Neural Information Processing Systems*, 21271–21284.

Guo, X.; Wang, Y.; Wei, Z.; and Wang, Y. 2023. Architecture Matters: Uncovering Implicit Mechanisms in Graph Contrastive Learning. In *Proceedings of Advances in Neural Information Processing Systems*, 28585–28610.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of Advances in Neural Information Processing Systems*.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive Multi-View Representation Learning on Graphs. In *Proceedings of International Conference on Machine Learning*, 4116–4126.

Huang, Y.; Du, C.; Xue, Z.; Chen, X.; Zhao, H.; and Huang, L. 2021. What Makes Multi-Modal Learning Better than Single (Provably). In *Proceedings of Advances in Neural Information Processing Systems*, 10944–10956.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of International Conference on Learning Representations*.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. In *Proceedings of Advances in Neural Information Processing Systems*, 21618–21629.

Lee, N.; Lee, J.; and Park, C. 2022. Augmentation-Free Self-Supervised Learning on Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 7372–7380.

Li, S.; Wang, X.; Zhang, A.; Wu, Y.; He, X.; and Chua, T.-S. 2022. Let Invariant Rationale Discovery Inspire Graph Contrastive Learning. In *Proceedings of International Conference on Machine Learning*, 13052–13065.

Liang, J.; Chen, M.; and Liang, J. 2024. Graph External Attention Enhanced Transformer. In *Proceedings of International Conference on Machine Learning*.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards Deeper Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 338–348.

Liu, Y.; Zhao, Y.; Wang, X.; Geng, L.; and Xiao, Z. 2023. Multi-Scale Subgraph Contrastive Learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2215–2223.

Mernyei, P.; and Cangea, C. 2020. Wiki-cs: A Wikipedia-Based Benchmark for Graph Neural Networks. In *Proceedings of International Conference on Machine Learning*.

Mo, Y.; Peng, L.; Xu, J.; Shi, X.; and Zhu, X. 2022. Simple Unsupervised Graph Representation Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 7797–7805.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*.

Peng, Z.; Huang, W.; Luo, M.; Zheng, Q.; Rong, Y.; Xu, T.; and Huang, J. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *Proceedings of The Web Conference*, 259–270.

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 *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1150–1160.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI magazine*, 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of Graph Neural Network Evaluation. *arXiv preprint arXiv:1811.05868*.

Shen, X.; Sun, D.; Pan, S.; Zhou, X.; and Yang, L. T. 2023. Neighbor Contrastive Learning on Learnable Graph Augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9782–9791.

Shorten, C.; and Khoshgoftaar, T. M. 2019. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 1–48.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial Graph Augmentation to Improve Graph Contrastive Learning. In *Proceedings of Advances in Neural Information Processing Systems*, 15920–15933.

Tan, S.; Li, D.; Jiang, R.; Zhang, Y.; and Okumura, M. 2024. Community-Invariant Graph Contrastive Learning. In *Proceedings of International Conference on Machine Learning*.

Thakoor, S.; Tallec, C.; Azar, M. G.; Azabou, M.; Dyer, E. L.; Munos, R.; Veličković, P.; and Valko, M. 2022. Large-Scale Representation Learning on Graphs via Bootstrapping. In *Proceedings of International Conference on Learning Representations*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of International Conference on Learning Representations*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *Proceedings of International Conference on Learning Representations*.

Wan, S.; Pan, S.; Yang, J.; and Gong, C. 2021a. Contrastive and Generative Graph Convolutional Networks for Graph-Based Semi-Supervised Learning. In *Proceedings of AAAI Conference on Artificial Intelligence*, 10049–10057.

Wan, S.; Zhan, Y.; Liu, L.; Yu, B.; Pan, S.; and Gong, C. 2021b. Contrastive Graph Poisson Networks: Semi-Supervised Learning with Extremely Limited Labels. In *Proceedings of Advances in Neural Information Processing Systems*, 6316–6327.

Wei, C.; Wang, Y.; Bai, B.; Ni, K.; Brady, D.; and Fang, L. 2023. Boosting Graph Contrastive Learning Via Graph Contrastive Saliency. In *Proceedings of International Conference on Machine Learning*, 36839–36855.

Wu, Q.; Zhao, W.; Li, Z.; Wipf, D. P.; and Yan, J. 2022. Nodeformer: A Scalable Graph Structure Learning Transformer for Node Classification. In *Proceedings of Advances in Neural Information Processing Systems*, 27387–27401.

Wu, Q.; Zhao, W.; Yang, C.; Zhang, H.; Nie, F.; Jiang, H.; Bian, Y.; and Yan, J. 2024. Simplifying and Empowering Transformers for Large-Graph Representations. In *Proceedings of Advances in Neural Information Processing Systems*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *Proceedings of International Conference on Learning Representations*.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of International Conference on Machine Learning*, 40–48.

Yin, Y.; Wang, Q.; Huang, S.; Xiong, H.; and Zhang, X. 2022. Autogcl: Automated Graph Contrastive Learning via Learnable View Generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8892–8900.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. In *Proceedings of Advances in Neural Information Processing Systems*, 5812–5823.

Yu, Y.; Wang, X.; Zhang, M.; Liu, N.; and Shi, C. 2024. Provable Training for Graph Contrastive Learning. In *Proceedings of Advances in Neural Information Processing Systems*.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph Transformer Networks. In *Proceedings of Advances in Neural Information Processing Systems*.

Zhang, Y.; Zhu, H.; Meng, Z.; Koniusz, P.; and King, I. 2022a. Graph-Adaptive Rectified Linear Unit for Graph Neural Networks. In *Proceedings of the ACM Web Conference*, 1331–1339.

Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; and King, I. 2022b. COSTA: Covariance-Preserving Feature Augmentation for Graph Contrastive Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2524–2534.

Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; and King, I. 2023. Spectral Feature Augmentation for Graph Contrastive Learning and Beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11289–11297.

Zhu, H.; Sun, K.; and Koniusz, P. 2021. Contrastive Laplacian Eigenmaps. In *Proceedings of Advances in Neural Information Processing Systems*, 5682–5695.

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 *Proceedings of the Web Conference*, 2069–2080.