SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

April 2026, Vol. 69, Iss. 4, 142103:1–142103:15
https://doi.org/10.1007/s11432-024-4688-6

T3A: robust graph contrastive learning with tensorized augmentation, aggregation and approximation

Tianchi LIAO[1,2], Yujing LAI[1], Tong WANG[1], Chuan CHEN[1*] & Zibin ZHENG[2]

[1]*School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China*
[2]*School of Software Engineering, Sun Yat-sen University, Zhuhai 519082, China*

Received 20 August 2024/Revised 19 August 2025/Accepted 18 November 2025/Published online 26 March 2026

Abstract Recently, graph contrastive learning (GraphCL) has received extensive attention owing to its unsupervised training paradigm and powerful representation learning capability. However, existing GraphCL methods have seldom been researched for robustness, and their performance degrades significantly against adversarial attacks. Therefore, it is significant to consider the robustness of GraphCL. In this paper, a robust GraphCL model with tensorized augmentation, aggregation, and approximation (T3A) is proposed to defend against attacks and noise. First, a tensorized multi-graph augmentation is designed to enable the encoder to derive more essential and intrinsic consistency across multiple augmented graphs, distinct from the single-graph augmentation used in existing studies, which is vulnerable to perturbations. Second, the augmented views are fed into a tensorized graph aggregator, which is trained using the extended contrastive loss in matrix form to fuse information from multiple graphs. Finally, tensorized low-rank approximation is employed for the tensor representation to further extract the low-rankness of the graph, thereby improving model robustness. Extensive experiments are conducted on five datasets under three different adversarial attacks to demonstrate the effectiveness and robustness of T3A.

Keywords graph contrastive learning, robust graph learning, low-rank approximation, tensor, augmentation

Citation Liao T C, Lai Y J, Wang T, et al. T3A: robust graph contrastive learning with tensorized augmentation, aggregation and approximation. Sci China Inf Sci, 2026, 69(4): 142103, https://doi.org/10.1007/s11432-024-4688-6

1 Introduction

Graph neural networks (GNNs) are a powerful family of methods for graph representation learning, achieving significant performance across a range of graph tasks, such as node classification [1] and link prediction [2]. Their core is the message-passing scheme, which can simultaneously incorporate both node attributes and graph topology.

Owing to the scarcity of labels, graph contrastive learning (GraphCL), an unsupervised method, has attracted considerable attention in recent years [3–5]. GraphCL aims to extract essential information by maximizing the mutual information across two augmented views of a graph. The resulting node representations can be employed for a variety of downstream tasks with high flexibility.

Recent studies have seldom focused on robustness in GraphCL, which is crucial to the performance on downstream tasks. GraphCL typically adopts GNNs as the backbone. However, GNNs are vulnerable to adversarial attacks [6] due to their message-passing scheme, which propagates slight noise to surrounding neighboring nodes and recursively affects a larger portion of the graph. Therefore, GraphCL frameworks are also sensitive to noise and attacks, which can significantly degrade the performance of downstream tasks. An experiment is conducted to demonstrate the concept, as shown in Figure 1. The classical GraphCL model deep graph infomax (DGI) [7] under a *nettack* attack [8], where the perturbation number on each targeted node is only one, shows a decrease in the node classification task of 3.5%, 0.64%, and 5.78% on the three datasets. This implies that the quality of the node representations produced by the GraphCL model deteriorates noticeably after being attacked. Therefore, it is essential to focus on the robustness of GraphCL to mitigate the impact of attacks.

Graph augmentation is a critical part of GraphCL, where the augmented view is generated by performing perturbations, sampling, or other transformations on the original graph. Existing contrastive learning methods employ single-graph augmentation [7, 9], i.e., generating one transformed graph as the augmented view, as shown in Figure 2(a). However, as the encoder in contrastive learning is enforced to learn the intrinsic and essential consistency

* Corresponding author (email: chenchuan@mail.sysu.edu.cn)

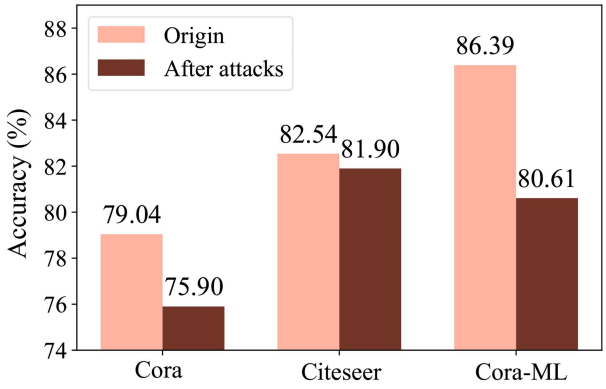

Figure 1 (Color online) Node classification accuracy of DGI before and after being attacked by *nettack* on three datasets.

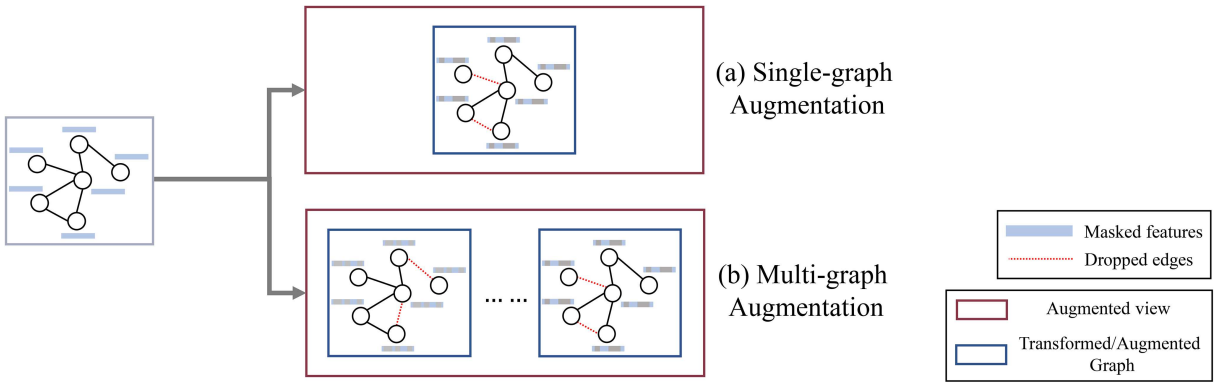

Figure 2 (Color online) Single-graph augmentation and multi-graph augmentation.

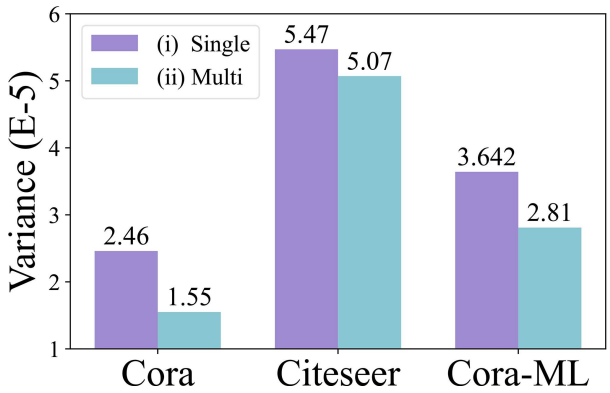

Figure 3 (Color online) Accuracy variances of DGI on Cora with single or multiple augmented graphs.

across different augmented graphs, more augmented graphs will allow the encoder to learn more intrinsic information from them. A case study was conducted to verify the effectiveness of using multiple augmented graphs. (i) The original graph generated an augmented graph by randomly masking node features and dropping edges. (ii) The original graph generated five augmented graphs by randomly masking node features and dropping edges, and experiments were performed for each. The experiments under both setups were conducted using the DGI on the Cora dataset, and the accuracy variances for the downstream node classification task were reported. The results are shown in Figure 3. The experiments using more augmented graphs resulted in lower variances, implying that the model performs more robustly and extracts more invariant intrinsic information. In contrast, existing contrastive models with single-graph augmentation learn from only two graphs in two views, making it difficult for them to defend against noise and attacks. Therefore, introducing more augmented graphs leads to a more robust performance compared with single-graph augmentation, enabling the encoder to mine more stable and intrinsic information and enhancing model robustness from the perspective of graph augmentation.

From the perspective of the encoder, the following problems must be considered. Capturing the correlations

among multiple augmented graphs and fusing the information between multiple graphs for enhancing the models robustness is challenging. Existing studies employ conventional GNNs as the encoder that can only accept one graph as an input at a time, and thus ignore the latent connections and fail to extract the common essential information of various augmented graphs, limiting the ability to learn robust node representations. Therefore, there is a demand for an encoder that can be fed with and fuse several graphs to exploit augmented graphs more efficiently and effectively, enhancing robustness. Conversely, many studies have shown that real-world graphs are universally low-rank [10], and most existing attack methods for graphs perform perturbations by introducing high-rank components of the graph data [11]. Therefore, low-rank approximation is an effective approach to resist noise and attacks, but maintaining the low-rankness of the graph in contrastive learning with multiple augmented graphs remains a challenge.

To solve these challenges, a robust GraphCL model is proposed to improve robustness with tensorized multi-graph augmentation (i.e., generating multiple transformed graphs as the augmented view, as shown in Figure 2(b)), tensorized graph aggregator, and tensorized low-rank approximation (T3A). First, a tensorized multi-graph augmentation is proposed to combine multiple transformed graphs to obtain augmented views in tensor form. The multi-graph augmentation method improves upon the previous single-graph augmentation, which lacks robustness. Second, a tensorized graph aggregator that exploits t-product operations to fuse and encode multiple graphs within a view is introduced, obtaining the node representations in matrix form with more robustness and generalization. Moreover, the existing vector-format contrastive loss is extended to be applied to the matrix-format representations. Finally, an f-product induced tensorized low-rank approximation is applied to the output from the aggregator to mine the low-rankness and derive the final robust node representation.

In summary, the contributions of the paper are as follows.

- A robust GraphCL model with tensorized multi-graph augmentation and tensorized graph aggregator optimized with an extended contrastive loss to mine intrinsic consistency across multiple augmented graphs is proposed, which fully leverages the advantages of multiple augmented graphs to enhance the robustness of GraphCL.
- Tensorized low-rank approximation is employed in GraphCL to extract the low-rank components of the graph to further yield more robust node representations.
- Extensive experiments are conducted on five datasets under three different adversarial attacks, illustrating the effectiveness and robustness of T3A compared with existing contrastive learning models.

2 Related work

2.1 Graph contrastive learning

Owing to the remarkable success in computer vision and natural language processing, contrastive learning has been applied to graphs and has attracted considerable research interest. The classical method DGI [7] maximizes the mutual information between patch representations and higher-level graph representations. GMI [12] measures the correlation between the input graph and the output representation in terms of node features and topology. GRACE [13] generates two augmented views by corruption and maximizes the agreement of node representation in two views. GCA [14] proposes an adaptive graph augmentation method to neglect the unimportant topology and node features. MERIT [15] adapts Siamese networks in computer vision to the graph domain and designs a multi-scale contrastive learning. ProGCL [16] focuses on the problem of false negative examples in contrastive learning and proposes a measure for negatives hardness. These methods improve contrastive learning from different aspects, but all remain within the inherited architecture of single-graph augmentation. RT-GCN [17] proposes a multi-view augmentation method for supervised tasks, but it can only tackle supervised tasks and fails to cope with more generalized unsupervised scenarios such as contrastive learning.

2.2 Robust graph learning

Recently, some studies have addressed the problem of robustness in graphs to improve the resistance of models against attacks and perturbations. RGCN [18] adopts Gaussian distributions as the hidden representations of nodes in each convolutional layer, allowing the model to automatically absorb the effects of adversarial changes through the variances of these distributions. GNNGuard [19] locates suspicious neighbors and reduces their importance, while assigning higher weights to edges connecting similar nodes within the message aggregation framework. ProGNN [10] defends against adversarial attacks by introducing constraints such as low-rankness and sparsity. However, few studies have considered the robustness problem in GraphCL, which is in an unsupervised scenario different from the aforementioned studies.

2.3 Low-rank approximation in various fields

Low-rankness is the common property of realistic data [20], and low-rank approximation is crucial for recovering the missing/error data and removing the perturbed noises, which is widely applied in various fields. Ref. [21] introduced t-product with Fourier convolution, and further proposed tensor singular value decomposition (t-SVD) to study the low-rankness of the data by defining the tensor tube rank. In [22], in order to reduce the computation of SVD, a tensor is decomposed into two smaller tensors by t-product and the low tube rank of the tensor is guaranteed. Ref. [23] considered the complexity of realistic data and embedded a deep neural network (DNN) into the t-SVD framework to improve the performance on the low-rank approximation. Ref. [24] proposed an effective and novel tensor decomposition method based on convolution calculation, which naturally extends the low-rank tensor model to the deep learning model.

2.4 Low-rank approximation in graph

GNNs have achieved remarkable success in learning node representations. However, recent studies have shown that noise inherent in real-world graph data can severely impair their effectiveness. Consequently low-rank techniques have been proposed as an effective means of mitigating such noise. In [25], a graph convolution with learnable low-rank local filters was proposed and proven to be more expressive than conventional methods. Some studies are devoted to speeding up low-rank approximations for robust graph models [26]. In [27], a robust and innovative node representation learning approach, termed low-rank regularized graph contrastive learning, was introduced, significantly enhancing model performance. For semi-supervised learning tasks, Yang et al. [28] proposed a low-rank tensor-decomposition-based GNN. The method constructs node attribute tensors from selected similar subgraphs and performs low-rank tensor decomposition to integrate long-range information. The use of the tensor nuclear norm facilitates the capture of long-range dependencies between the original network and the selected similar subgraphs. Entezari et al. [29] proposed a low-rank adversarial attack on graph embeddings, capable of degrading the classification performance of both graph neural networks and tensor-based node embeddings. Their findings demonstrate that low-rank attacks are conspicuous, whereas high-rank attacks are less perceptible. Building on low-rank information in graphs, Yang et al. [30] proposed an effective and parameter-efficient approach for transferring pre-trained GNNs to different graph domains. However, few methods have addressed the problem of low-rankness deficiency due to the attacks arising at the graph data level.

3 Preliminary

3.1 Notations

A graph is defined as $G = (V, E, X, A)$, where $V = \{v_1, v_2, ..., v_N\}$ and $E \subset V \times V$ represent the set of nodes and edges, respectively. $X = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_N] \in \mathbb{R}^{N\times F}$ is the feature matrix of all nodes, where each row $\mathbf{x}_i \in \mathbb{R}^F$is the feature vector of the corresponding node v_i. $A \in \{0, 1\}^{N\times N}$ denotes the adjacency matrix, where $A_{i,j} = 1$ if an edge exists between nodes v_i and v_j, else $A_{i,j} = 0$. For simplicity, a graph is also usually defined as $G = (X, A)$.

We use Euler letters to represent a tensor. The $\times_n$ denotes the mode-n tensor matrix product, i.e., for an N-order tensors $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and a matrix $W \in \mathbb{R}^{J\times I_n}$, the product along the n-th mode is defined as $\mathcal{Y}$, and the dimension of the resulting tensor $\mathcal{Y}$ is $I_1 \times \cdots \times J \times \cdots \times I_N$. $\mathcal{X}^{(i)} \in \mathbb{R}^{I_1\times I_2}$ denotes the i-th frontal slice of a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times I_3}$. Δ is the face-wise product between two tensors [31], i.e., $\mathcal{C} = \mathcal{A} * \mathcal{B} \Leftrightarrow \mathcal{C}^{(i)} = \mathcal{A}^{(i)} * \mathcal{B}^{(i)}$.

3.2 Tensor algebra

We first introduce t-product which can be implemented by the discrete Fourier transform for less computational resource [21] as follows.

Definition 1 (t-product [22])**.** The t-product of $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2\times n_4\times n_3}$ is a tensor $\mathcal{C} \in \mathbb{R}^{n_1\times n_4\times n_3}$ given by $\mathcal{C} = \mathcal{A} * \mathcal{B} = ((\mathcal{A} \times_3 F)\Delta(\mathcal{B} \times_3 F)) \times_3 F^{-1} = (\tilde{\mathcal{A}}\Delta\tilde{\mathcal{B}}) \times_3 F^{-1}$, where $F \in \mathbb{R}^{n_3\times n_3}$ is the discrete Fourier transform matrix and $F^{-1}$is the inverse matrix.

At present, most tensor factorization methods rely on multilinear operations. For instance, the t-product operation is based on linear transformations such as the discrete Fourier transform (DFT) or the discrete cosine transform (DCT). These linear transformations are applied to map the tensor into a low-rank representation, where the matrix rank of the transformed frontal slices is then used to define the tensor rank. However, given the complex and diverse topological structures of real-world data, the mapping between the original tensor and its optimal low-rank

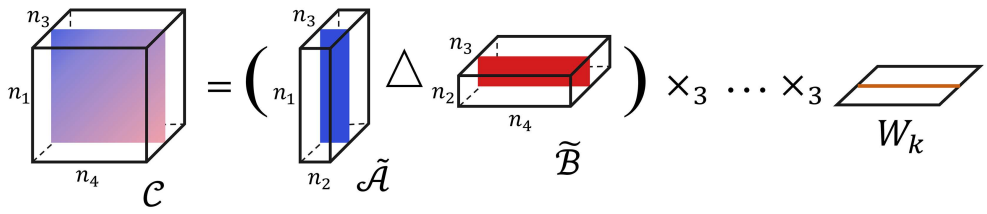

Figure 4 (Color online) Figurative expressions of Definition 2.

representation is likely to be nonlinear and hierarchical, which cannot be adequately captured by linear transformations. Therefore, we replace the linear transformation with DNNs [23]. We then introduce the f-product operation involved in the low-rank approximation. The calculation process is shown in Figure 4.

Definition 2 (f-product [23])**.** The f-product of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ is a tensor $\mathcal{C} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$ given by $\mathcal{C} = \mathcal{A} *_f \mathcal{B} = g(f(\mathcal{A})\Delta f(\mathcal{B})) = g(\tilde{\mathcal{A}}\Delta\tilde{\mathcal{B}})$, where $f(\cdot)$ is a DNN, and $g(\cdot)$ is the inverse transform of $f(\cdot)$. Given the matrices $\{W_k \in \mathbb{R}^{n_3 \times n_3}\}_{k=1}^{L}$ and a nonlinear scalar function $\sigma(\cdot)$, $f(\mathcal{X}) = \sigma(\cdots\sigma(\sigma(\mathcal{X} \times_3 W_1) \times_3 W_2)\cdots \times_3 W_{L-1}) \times_3 W_L$. When $k = 1$, $f(\cdot)$ degenerates into the linear form.

With Definition 2, we can characterize the low-rankness of a low-rank tensor by factorizing it as follows.

Theorem 1 (Low-rank tensor factorization)**.** Assume that $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a 3-order tensor and the tube rank of $\mathcal{X}$ is $rank_f(\mathcal{X}) \triangleq \max\limits_{i=1,2,\cdots,n_3} \{rank(f(\mathcal{X})^{(i)})\}$. When $rank_f(\mathcal{X}) = r$, $\mathcal{X}$ can be factored as two tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times r \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{r \times n_2 \times n_3}$ such that $\mathcal{X} = \mathcal{A} *_f \mathcal{B}$ hold.

Definition 1 is the final form of t-product. Here we talk about the detailed explanation of the process of Fourier convolution. For tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we define following operators:

$$\text{bcirc}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & A^{(n_3)} & \cdots & A^{(2)} \\ A^{(2)} & A^{(1)} & \cdots & A^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(n_3)} & A^{(n_3-1)} & \cdots & A^{(1)} \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}, \quad \text{bdiag}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & & & \\ & A^{(2)} & & \\ & & \ddots & \\ & & & A^{(n_3)} \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}, \tag{1}$$

and

$$\text{vec}(\mathcal{A}) = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n_3)} \end{bmatrix}, \quad \text{fold}_v(\text{vec}(\mathcal{A})) = \mathcal{A}, \quad \text{fold}_d(\text{bdiag}(\mathcal{A})) = \mathcal{A}. \tag{2}$$

The t-product of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ is a tensor $\mathcal{C} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$ given by $\mathcal{C} = \mathcal{A} * \mathcal{B} = \text{fold}_v(\text{bcirc}(\mathcal{A}) \times \text{vec}(\mathcal{B}))$. Since the block circulant matrix $\text{bcirc}(\mathcal{A})$ can be diagonalized by the discrete Fourier transform matrix $F_{n_3} \in \mathbb{R}^{n_3 \times n_3}$, i.e., $\text{bdiag}(\hat{\mathcal{A}}) = (F_{n_3} \otimes I_{n_1}) \times \text{bcirc}(\mathcal{A}) \times (F_{n_3}^{\text{H}} \otimes I_{n_2})$, where $\otimes$ is the Kronecker product. It follows that

$$\begin{aligned} \mathcal{A} * \mathcal{B} &= \text{fold}_v(\text{bcirc}(\mathcal{A}) \times \text{vec}(\mathcal{B})) \\ &= \text{fold}_v((F_{n_3}^{\text{H}} \otimes I_{n_1}) \times \text{bdiag}(\hat{\mathcal{A}}) \times (F_{n_3} \otimes I_{n_2}) \times \text{vec}(\mathcal{B})) \\ &= \text{fold}_v((F_{n_3}^{\text{H}} \otimes I_{n_1}) \times \text{bdiag}(\hat{\mathcal{A}}) \times \text{vec}(\hat{\mathcal{B}})) \\ &= \text{fold}_d((F_{n_3}^{\text{H}} \otimes I_{n_1}) \times \text{bdiag}(\hat{\mathcal{A}}) \times \text{bdiag}(\hat{\mathcal{B}})) \\ &= F_{n_3}^{\text{H}}[\text{fold}_d(\text{bdiag}(\hat{\mathcal{A}}) \times \text{bdiag}(\hat{\mathcal{B}}))] \\ &= ((\mathcal{A} \times_3 F_{n_3})\Delta(\mathcal{B} \times_3 F_{n_3})) \times_3 F_{n_3}^{-1} \\ &= (\hat{\mathcal{A}}\Delta\hat{\mathcal{B}}) \times_3 F_{n_3}^{-1}, \end{aligned} \tag{3}$$

where A^{H} denotes the conjugate transpose of A.

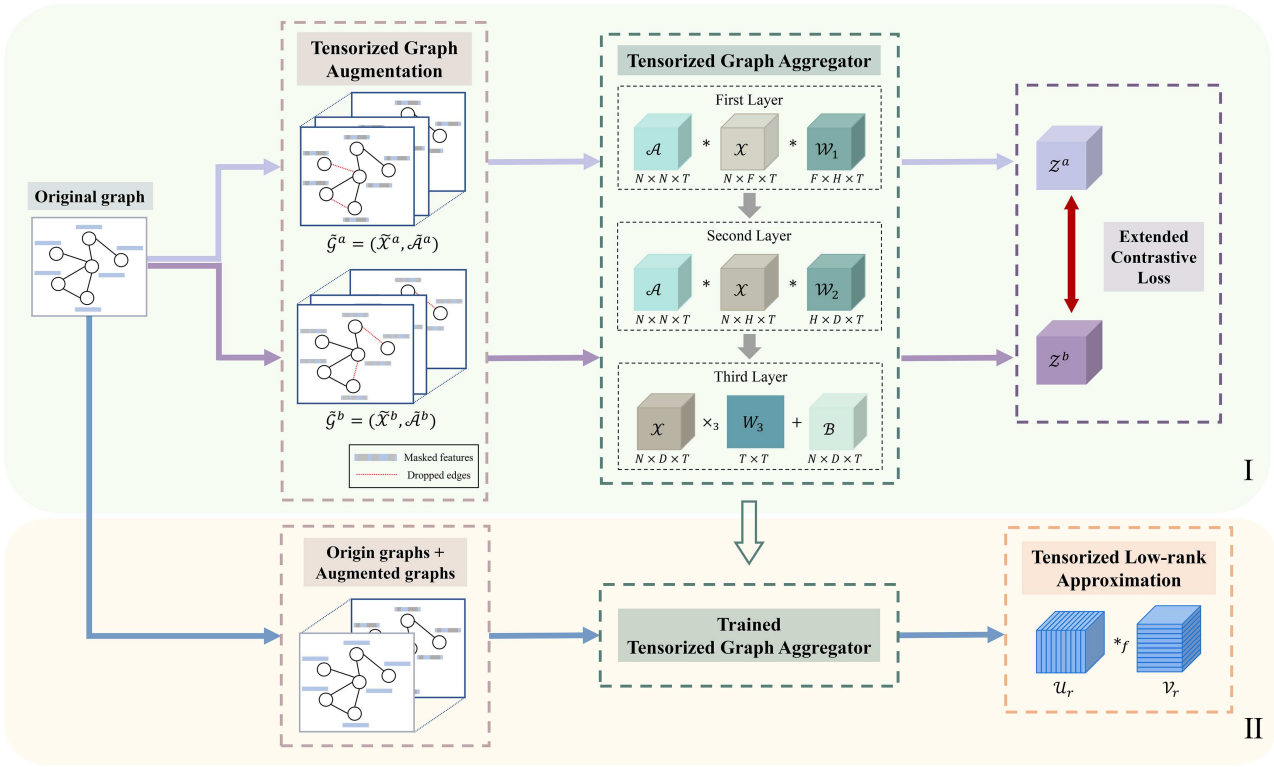

Figure 5 (Color online) The overview of T3A. In the green part I of Figure 5, after tensorized multi-graph augmentation, the parameters in the tensorized graph aggregator are optimized with the extended contrastive loss. In the yellow part II, the trained tensorized graph aggregator with fixed parameters is employed to yield tensor representations which are further optimized with the low-rank approximation.

4 Framework

The proposed T3A is illustrated in Figure 5 and consists of four components: (1) tensorized multi-graph augmentation, (2) tensorized graph aggregator, (3) extended contrastive loss in matrix format and (4) tensorized low-rank approximation. Tensorized multi-graph augmentation first generates two augmented views in tensor form. Then the augmented views are fed into the tensorized graph aggregator which is trained with the extended contrastive loss in matrix form to fuse multiple augmented graphs. Finally, the f-product induced low-rank approximation for the tensor output from the trained tensorized graph aggregator is performed to mine the low-rankness of the node representations for more robustness.

4.1 Tensorized multi-graph augmentation

Graph augmentation implements transformations on the original graph to enforce the encoder to explore the essential and intrinsic information consistent in different augmented graphs. While the existing contrastive learning studies adopt single-graph augmentation which is considered to introduce high variance, we propose the tensorized multi-graph augmentation that exploits multiple augmented graphs.

We first describe the technique of feature masking and edge dropping in single-graph augmentation, which is simple and widely effective [32].

Feature masking. Given a graph $G = (X, A)$, the features of nodes in the graph are randomly masked. For the F-dimensional features of the nodes, each dimension is masked with 0 with the probability p_x. The augmented feature matrix $\tilde{X}$ is calculated as

$$\tilde{X} = X \odot M, \ M_{i,j} = \begin{cases} 0, & \text{the } j\text{-th column is selected as mask,} \\ 1, & \text{otherwise,} \end{cases} \tag{4}$$

where M is a masking matrix in which each column has the probability of p_x to be selected as a mask. $\odot$ is the Hadamard product. Feature masking motivates the encoder to infer the missing semantics of the nodes from the context.

Edge dropping. Given a graph $G = (X, A)$, the edges in the graph are randomly dropped. For all edges in the graph, each edge is dropped with the probability p_e to obtain an augmented adjacency matrix $\tilde{A}$. The augmented adjacency matrix $\tilde{A}$ is calculated as

$$\tilde{A} = A \odot (1 - \mathcal{D}), \tag{5}$$

where $\mathcal{D}$ obeys Bernoulli distribution $\mathcal{D} \sim \mathcal{B}(p_e)$. If $\mathcal{G}$ is an undirected graph and the corresponding adjacency matrix A is symmetric, we allow $\tilde{A}$ to be asymmetric, in which case there is only unidirectional information propagation between certain nodes. Edge dropping prompts the encoder to study the structural semantics so that the encoder is immune to connection perturbations. Therefore, $\tilde{X}$ and $\tilde{A}$ constitute a single-graph augmented view denoted as $\tilde{G} = (\tilde{X}, \tilde{A})$.

We propose the tensorized multi-graph augmentation to exploit multiple transformed graphs. T augmented graphs are first generated with the above single-graph augmentation. The $\tilde{X}$ and $\tilde{A}$ in these T augmented graphs are stacked in the third order to form a third-order tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{N\times F\times T}$ and $\tilde{\mathcal{A}} \in \mathbb{R}^{N\times N\times T}$, respectively. In $\tilde{\mathcal{X}}$ or $\tilde{\mathcal{A}}$, each frontal slice is a single-graph augmented feature matrix or adjacency matrix. $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{A}}$ constitute an tensorized augmented view $\tilde{\mathcal{G}} = (\tilde{\mathcal{X}}, \tilde{\mathcal{A}})$. The above procedure is performed twice to construct two tensorized augmented views $\tilde{\mathcal{G}}^a = (\tilde{\mathcal{X}}^a, \tilde{\mathcal{A}}^a)$ and $\tilde{\mathcal{G}}^b = (\tilde{\mathcal{X}}^b, \tilde{\mathcal{A}}^b)$. By introducing multi-graph augmentation, the encoder is enabled to extract more intrinsic consistency and promote the robustness of the model from the graph augmentation perspective.

4.2 Tensorized graph aggregator

After the tensorized augmented views are constructed, it is necessary to encode and produce node representations through an encoder to further extract intrinsic correlations. However, existing GNNs can only accept a single graph as input, ignoring the latent connections between multiple augmented graphs and reducing the robustness of the model.

To solve the above problems, we introduce a tensorized graph aggregator that is capable of fusing information among multiple graphs:

$$f_\theta(\mathcal{X}, \mathcal{A}) = \mathrm{softmax}((\mathcal{A} * \sigma(\mathcal{A} * \mathcal{X} * \mathcal{W}_1) * \mathcal{W}_2) \times_3 W_3 + \mathcal{B}), \tag{6}$$

where $*$ denotes the t-product, $\times_3$ denotes the multiplication along the third dimension between the tensor and the matrix. $\sigma(\cdot)$ is a nonlinear activation. The model's parameter θ includes $\mathcal{W}_1 \in \mathbb{R}^{F\times H\times T}$, $\mathcal{W}_2 \in \mathbb{R}^{H\times D\times T}$, $W_3 \in \mathbb{R}^{T\times T}$ and $\mathcal{B} \in \mathbb{R}^{N\times D\times T}$ where H is the dimension of hidden layers and D is the dimension of the output layer. As in Definition 1, the t-product performs a transformation in the Fourier domain on the third dimension of the tensor. The third dimension of the input tensor corresponds to multiple augmented graphs, so the tensorized graph aggregator is capable of fusing and encoding multi-graph information.

The output of the tensorized graph aggregator is a representation tensor of size $N \times D \times T$, where the node representation of each node is the corresponding upper slice of size $D \times T$. The output representation tensor for the two views $\tilde{\mathcal{G}}^a = (\tilde{\mathcal{X}}^a, \tilde{\mathcal{A}}^a)$ and $\tilde{\mathcal{G}}^b = (\tilde{\mathcal{X}}^b, \tilde{\mathcal{A}}^b)$ are defined as $\mathcal{Z}^a \in \mathbb{R}^{N\times D\times T}$ and $\mathcal{Z}^b \in \mathbb{R}^{N\times D\times T}$, respectively, and the corresponding output representation of the node v are defined as $Z_v^a \in \mathbb{R}^{D\times T}$ and $Z_v^b \in \mathbb{R}^{D\times T}$. The representation of each node merges information from various graphs, and thus the contrastive learning framework is able to mine the intrinsic consistency across multiple augmented graphs for improving robustness and stability.

4.3 Contrastive loss in matrix form

The contrastive loss aims to maximize the mutual information of two views, constructing two representations of the same node corresponding to two views as positive pairs and representations of different nodes as negative pairs. It extracts the invariant intrinsic consistency of the graph through encouraging similarity of representations in positive pairs and reducing similarity of representations in negative pairs.

The node representations produced by the tensorized graph aggregator are in matrix form and contain distinct information in the two dimensions. The node representation is of size $D \times T$. The dimension of size T corresponds to different augmented graphs which are fused by the tensorized graph aggregator, and thus this dimension contains inter-graph information. Correspondingly, the dimension of size D contains intra-graph information. Therefore, the two dimensions are required to be distinguished in the contrastive loss. However, the general graph contrastive learning loss [7, 9, 12] can only tackle node representations in vector form.

We extend the general contrastive loss for node representations in matrix form. The contrastive loss of a node v is

$$\mathcal{L}_s(v) = \mathcal{L}_s^{a,b}(v) + \mathcal{L}_s^{b,a}(v), \tag{7}$$

$$\mathcal{L}_s^{i,j}(v) = -\log\frac{\exp(s(Z_v^i, Z_v^j)/\tau)}{\exp(s(Z_v^i, Z_v^j)/\tau) + \sum_{u\neq v} F_s^{i,j}(v,u)}, \tag{8}$$

$$F_s^{i,j}(v,u) = \exp(s(Z_v^i, Z_u^j)/\tau) + \exp(s(Z_v^i, Z_u^i)/\tau), \tag{9}$$

where i, j are the two views and i is the anchor view. In the first term of (7), i corresponds to the view a and j corresponds to the view b. In the second term of (7), i corresponds to the view b and j corresponds to the view a. Z_v^i and Z_v^j are two representations of node v in views i and j. With Z_v^i as the anchor representation, the representation Z_v^j of the same node v in the different view j is a positive sample, and the representations Z_u^i and Z_u^j of other nodes u in the same view i and in the different view j are negative samples. The F function computes the similarity of negative pairs. τ is the temperature hyperparameter. The contrastive loss aims at increasing the similarity of the representations of the same node in both views and decreasing the similarity of the representations of different nodes simultaneously. $s(\cdot,\cdot)$ is a similarity function between matrices (in previous studies it is between vectors), which is designed as two variants to distinguish the two dimensions. The first variant is as

$$s_r(Z_v^i, Z_v^j) = \frac{\langle Z_v^i W_r, Z_v^j W_r\rangle}{\|Z_v^i W_r\|_2 \|Z_v^j W_r\|_2}, \tag{10}$$

where $W_r \in \mathbb{R}^{T\times T}$ is the projection matrix to capture the inter-graph information. The similarity is calculated after projecting each row of size T of the node representations into the same space.

Similarly, the node matrix is projected column-wise with another projection matrix $W_c \in \mathbb{R}^{D\times D}$ to capture the intra-graph information. So the second variant of $s(\cdot,\cdot)$ is as

$$s_c(Z_v^i, Z_v^j) = \frac{\langle W_c Z_v^i, W_c Z_v^j\rangle}{\|W_c Z_v^i\|_2 \|W_c Z_v^j\|_2}. \tag{11}$$

The first variant is calculated row-wise while the second variant is calculated column-wise, capturing different information.

The final loss function is organized with the two variants as

$$\mathcal{L} = \sum_v \mathcal{L}_{s_r}(v) + \mathcal{L}_{s_c}(v). \tag{12}$$

Therefore, by projecting the two dimensions into separate spaces to distinguish the inter-graph and intra-graph information in node representations, the general contrastive loss is extended to the matrix form and the consistency in the node representations is captured. Through training, the essential consistency across multiple augmented graphs is fully explored, and thus more robust node representations are generated.

4.4 Tensorized low-rank approximation

After training the tensorized graph aggregator, the next step is to derive the node representations to be applied in downstream tasks. Adopt T_1 original graphs and generate T_2 single-graph augmented graphs ($T_1 + T_2 = T$), then stack them randomly at the third order to form a third-order tensor. The third-order tensor is fed into the tensorized graph aggregator for yielding the node representation tensor $\mathcal{Z}$.

Before feeding the node representations into the downstream task, we employ a tensorized low-rank approximation to extract the low-rank components of the tensor data for further robustness enhancement. Recall that real-world graphs universally exhibit low-rankness, where nodes with the same label are likely to be clustered in the same community [10]. Thus, the existing graph attack methods tend to introduce the high-rank components of the graph [11]. Therefore, the low-rank approximation can eliminate the noise or perturbations for the graph.

Inspired by [23], considering the complexity and diversity of graph data, there is most probably a nonlinear transformation between the graph and the optimal low-rank representation. Therefore, we employ f-product induced tensorized low-rank factorization, which embeds DNNs and captures the nonlinear transformation for a better low-rank structure for the graph. Specifically, we perform a low-rank approximation on the representation tensor $\mathcal{Z}$ as

$$\mathcal{Z} = \mathcal{U}_r *_f \mathcal{V}_r, \tag{13}$$

where r is the tube rank of $\mathcal{Z}$. $\mathcal{U}_r$ and $\mathcal{V}_r$ are tensor factorizations of rank r. $*_f$ denotes the f-product. After the tensorized low-rank approximation which further extracts the low-rank components of the graph from the resulting representations, more robust node representations are derived for downstream tasks.

Table 1 Statistics of datasets.

Dataset	$\#N$	$\#E$	#Features	#Labels	Homophily level
Cora	2708	5429	1433	7	0.810
Citeseer	3327	4732	3703	6	0.736
Cora-ML	2995	8416	2879	7	0.810
PubMed	19717	44324	500	3	0.802
Actor	7600	30019	931	5	0.219

4.5 Complexity analysis

We analyze the time complexity and space complexity of the T3A model by the number of floating point operations (FLOPs) and space overhead required by the T3A model by separately considering the four main components mentioned above. For Section 4.1, the primary computation lies in generating augmented graphs, with a time complexity of $\mathcal{O}(T(NF+E))$. By employing sparse matrix storage, the space complexity is $\mathcal{O}(TNF+TE)$. In Section 4.2, the major computational cost arises from the tensorized graph aggregator. For the first t-product layer, the time complexity is $\mathcal{O}(TN^2F+TN^2\log T+TNFH+TNF\log T)$, and similarly, the second and third layers each require $\mathcal{O}(TN^2H+TN^2\log T+TNDH+TND\log T)$ and $\mathcal{O}(NT^2D)$. Given that $D<H<F\ll N$, the overall time complexity can be simplified to $\mathcal{O}(TN^2+T^2N)$. The space complexity in this module is dominated by the models output and weight parameters, amounting to $\mathcal{O}(NFT+NDT+FHT+HDT+T^2)$. Section 4.3 primarily involves the computation of the contrastive loss, with a time complexity of $\mathcal{O}(NT^2D+ND^2T+B^2TD)$, where B denotes the batch size, and a space complexity of $\mathcal{O}(N^2+NDT)$. Finally, in Section 4.4, the computational cost is mainly due to the tensor decomposition process, yielding a time complexity of $\mathcal{O}(TNrD+NDT^2)$ and a space complexity of $\mathcal{O}(NrT+rDT+T^2L)$.

5 Experiments

5.1 Experimental setup

5.1.1 *Datasets and baselines*

We validate the performance of the proposed model T3A on four homophilic graphs: Cora [33], Citeseer [34], Cora-ML [35], PubMed [36], and one heterophilic graph data: Actor [37], whose statistics are summarized in Table 1.

To comprehensively validate the effectiveness of the proposed T3A approach, it is compared with some baselines, including four supervised learning methods: GCN [38], SGC [39], GAT [40], and ADEdgeDrop (ADED) [41], as well as eight contrastive learning methods, including DGI [7], GMI [12], GRACE [12], GCA [14], MERIT [15], ProGCL [16], S3GCL [42], and GRANCE [3].

5.1.2 *Implementation*

All the experiments were implemented in PyTorch. The feature masking probability p_x and edge dropping probability p_e for both views were set to 0.2 and 0.3, respectively. The dimensions of the hidden and output layers were set to 512 and 16, respectively. The augmentation time T was fixed to 16. The learning rates for the five datasets were chosen from {0.0005, 0.001, 0.003, 0.005, 0.01} based on performance. The codes are available online: https://github.com/JaneYul/T3A.

The downstream task is node classification. According to the evaluation scheme in [7], each model was first trained under the unsupervised setting, and the resulting embeddings of the training nodes were used to train a logistic regression classifier and tested on the test nodes. The nodes in each graph were randomly divided, where 10% served for training, 10% for validation, and the remaining 80% for testing. All experiments were repeated 20 times, and their average accuracies and standard deviations are recorded.

5.2 Experimental results

5.2.1 *Comparative study*

Table 2 demonstrates the node classification accuracy and standard deviation of five datasets, where the baselines are constructed with single-graph augmentation. The experimental results demonstrate that T3A consistently outperforms all unsupervised baselines and surpasses supervised methods, such as SGC, GCN, and GAT. Moreover, in most cases, T3A achieves lower standard deviations than the baselines, indicating greater stability. This superior

Table 2 Node classification accuracy (%) (mean ± std) of five datasets. X, A, and Y denote the node feature matrix, adjacency matrix, and label information available for the methods, respectively. The top-performing unsupervised model is in bold.

Method	Training data	Cora	Citeseer	Cora-ML	PubMed	Actor
SGC	X, A, Y	80.6±0.8	69.1±0.7	81.9±0.8	74.12±0.5	22.64±1.1
GCN	X, A, Y	81.8±0.5	71.2±0.7	85.9±0.6	80.90±0.3	28.64±1.4
GAT	X, A, Y	82.57±1.0	71.96±1.0	84.29±1.1	79.51±0.3	25.44±0.9
ADED	X, A, Y	83.67±0.6	73.28±1.2	85.07±1.0	83.12±0.2	29.12±1.7
DGI	X, A	82.60±0.4	68.80±0.7	85.33±0.4	77.59±0.3	27.33±1.6
GMI	X, A	83.03±0.7	70.84±0.4	83.15±0.5	78.03±1.7	26.24±0.5
GRACE	X, A	83.30±0.4	72.10±0.5	85.37±0.7	80.28±0.9	28.24±0.2
GCA	X, A	83.28±0.8	72.43±0.7	85.64±0.6	81.84±1.1	27.85±0.7
MERIT	X, A	83.48±0.5	72.31±0.4	85.72±0.2	79.28±0.2	28.02±0.7
ProGCL	X, A	83.54±0.7	72.50±0.3	86.17±0.3	80.66±0.2	27.81±0.8
S3GCL	X, A	83.67±1.2	72.19±0.8	85.53±1.3	81.13±1.4	26.55±0.9
GRANCE	X, A	83.32±0.3	71.98±0.6	85.27±0.3	81.62±0.5	27.82±0.5
T3A	X, A	**83.81±0.4**	**73.01±0.3**	**86.38±0.3**	**82.41±0.6**	**28.81±0.5**

Table 3 Comparison of the FLOPs (in GFLOPs) and the memory cost (in GB) of different methods on Cora, Citeseer, Cora-ML, PubMed, and Actor.

Method	Cora		Citeseer		Cora-ML		PubMed		Actor	
	FLOPs	Mem.	FLOPs	Mem.	FLOPs	Mem.	FLOPs	Mem.	FLOPs	Mem.
DGI	2.48G	1.42G	5.56G	4.87G	3.29G	3.12G	11.87G	14.63G	3.42G	3.01G
GCA	0.36G	0.93G	0.92G	3.01G	0.56G	1.23G	2.60G	13.87G	1.83G	1.16G
T3A	0.66G	1.33G	1.32G	4.91G	1.04G	3.27G	6.23G	17.43G	2.15G	2.13G

performance can be attributed to the synergy of its four key modules. Specifically, the multi-graph augmentation enables the model to learn more stable and consistent features across multiple views, thereby reducing the variance caused by single perturbations. The tensorized aggregator, through the t-product, fuses cross-graph information and extracts shared representations across augmented graphs, which enhances robustness. The matrix-form contrastive loss jointly captures intra-graph and inter-graph consistency, preserving discriminative power at a finer granularity. Finally, the tensorized low-rank approximation effectively removes high-rank noise and preserves the inherent low-rank structure of the graph, further improving resistance to adversarial attacks. Together, these modules enable T3A to significantly outperform existing methods in both standard tasks and adversarial settings.

Furthermore, the performance gains on the Cora dataset are smaller than those on other datasets, primarily because Cora has fewer nodes and edges, as well as lower feature dimensionality, which limits the capacity of multi-graph augmentation and the tensorized aggregator to capture richer information. Nevertheless, despite the relatively limited improvements, T3A consistently achieves lower standard deviations than the baselines on Cora, indicating that our model maintains stronger stability.

5.2.2 *Computational complexity study*

We compared the FLOPs and memory consumption of the classical algorithm DGI and the graph augmentation method GCA on the Cora, Citeseer, Cora-ML, PubMed, and Actor datasets, as shown in Table 3. The results show that DGI exhibits the highest FLOPs. Owing to the smaller number of augmented graphs generated, GCA requires substantially less memory than our method. Although our method does not have a memory advantage over commonly used augmentation approaches, its memory overhead remains within an acceptable range. Moreover, as demonstrated in Section 5.2.3, the robustness of these competing methods is limited.

5.2.3 *Robustness study*

To further demonstrate the robustness of T3A against adversarial attacks, attack experiments were conducted on homophily graphs (Cora, Citeseer, Cora-ML, and PubMed), comparing T3A with the baseline methods. The experimental setup follows research in the field of supervised learning [10, 17]. The following three approaches are adopted to attack the original graph.

• **Targeted attack** aims at attacking the target nodes and degrading the performance of the model on these specific nodes. The state-of-the-art targeted attack, *nettack*, was used [8].

Table 4 Node classification accuracy (%) under *metattack*, where the best result is in bold and the second best is underlined.

Dataset	PTB rate	DGI	GMI	GRACE	GCA	MERIT	ProGCL	S3GCL	GRANCE	T3A
Cora	0	83.89	84.00	84.26	84.71	84.46	84.77	85.02	84.71	**84.85**
	0.1	81.88	81.89	82.37	83.23	83.52	83.84	83.22	83.67	**84.07**
	0.3	76.07	76.13	78.74	79.32	78.53	**79.87**	79.13	79.65	79.70
	0.5	73.24	71.53	76.31	76.25	75.24	76.49	76.09	75.52	**76.62**
	0.7	71.42	70.93	74.3	74.29	72.16	74.55	74.22	74.82	**75.31**
	0.9	70.79	68.88	73.14	72.73	70.76	73.31	73.43	73.53	**74.40**
Citeseer	0	74.54	72.92	73.70	75.94	75.88	75.97	76.01	75.87	**76.58**
	0.05	74.09	72.31	72.2	75.11	75.22	76.15	75.27	75.81	**76.26**
	0.1	72.37	72.03	70.68	74.1	73.85	74.62	74.02	74.73	**75.92**
	0.15	72.67	71.06	69.91	73.36	73.39	74.1	74.28	73.79	**75.16**
	0.2	70.69	69.67	69.00	71.23	72.13	73.08	73.82	73.31	**74.55**
	0.25	70.37	69.41	69.02	70.65	71.48	72.56	72.63	72.36	**73.86**
Cora-ML	0	85.38	84.8	84.91	85.8	85.77	86.16	86.21	86.17	**86.45**
	0.05	80.53	78.46	78.68	80.86	80.82	82.32	82.43	82.11	**82.98**
	0.1	78.19	76.7	77.94	79.29	79.32	79.81	79.34	79.53	**80.87**
	0.15	77.04	75.13	76.79	78.22	77.39	78.91	78.43	78.21	**79.54**
	0.2	76.38	74.09	76.51	76.57	76.17	77.68	77.31	78.01	**78.77**
	0.25	75.12	73.92	75.25	75.36	75.91	77.03	76.41	77.26	**78.42**
PubMed	0	82.46	81.85	82.93	**83.52**	83.29	83.11	83.32	83.19	83.24
	0.05	80.32	79.26	81.31	81.23	81.29	81.34	81.71	81.74	**81.98**
	0.1	76.77	75.14	78.25	78.16	78.23	78.27	79.32	79.39	**80.33**
	0.15	74.24	72.33	76.14	75.97	75.83	76.15	76.87	76.85	**77.52**
	0.2	72.21	70.31	73.92	74.26	74.07	74.16	75.28	75.24	**76.31**
	0.25	69.93	68.41	71.31	72.63	72.01	71.98	72.75	72.25	**73.38**

• **Nontargeted attack** attempts to degrade the overall performance of the model over the whole graph, distinct from targeted attack. The *metattack* [43] was employed as the nontargeted attack owing to its representativeness.

• **Random attack** randomly selects some node pairs and flips their connectivity (i.e., removes existing edges and connects nonadjacent nodes). This can be treated as injecting random noise into a clean graph by randomly adding noisy edges [10].

The above three approaches attack the original graph with different perturbation rates, and then models are evaluated on the attacked graph. Following [10], all attack experiments are conducted on the largest connected component of the benchmark datasets. In the *nettack* experiment, as it targets specific nodes, evaluations are conducted on these nodes using the same split as in [10]. In the other two attack experiments, the split is the same as in the comparative study. The original methods are employed, as well as default parameters for each model. In addition, the results under no attack are reported for all datasets.

The experimental results under *metattack* are shown in Table 4, where the *metattack* perturbation (PTB) rate is set from 0 to 0.25 on Citeseer, Cora-ML, and PubMed, and from 0 to 0.9 on Cora owing to our interests in the performance of these methods with high perturbation rates under *metattack*. T3A is superior to the other competitors in most settings. Another observation derived from the results is that a method that performs better in the comparative study without perturbation may not perform equally well in attack experiments; e.g., DGI performs better than GMI on the four datasets; however, it underperforms GMI in the comparative study. In addition, with a high perturbation rate of 0.9, the results of all methods do not yield significantly low results owing to the inherent robustness of the contrastive learning paradigm.

The accuracy of node classification under *nettack* is shown in Figure 6. The number of perturbations on each targeted node ranges from 0 to 5. The proposed method achieves the best performance with a different number of perturbations. Moreover, the improvement of T3A over the most competitive baseline is more significant when the number of perturbations is larger. Specifically, T3A has 0.15%, 1.16%, 1.34%, and 0.81% accuracy improvement compared with the second-best baseline for the four datasets with a perturbation number of one. When the number of perturbations is increased to five, the improvement rises to 2.79%, 8.46%, 2.32%, and 1.98%, respectively. This indicates the superior robustness of our method against high perturbation rates, and such superiority is also observed in the experiments of *metattack* and random attacks.

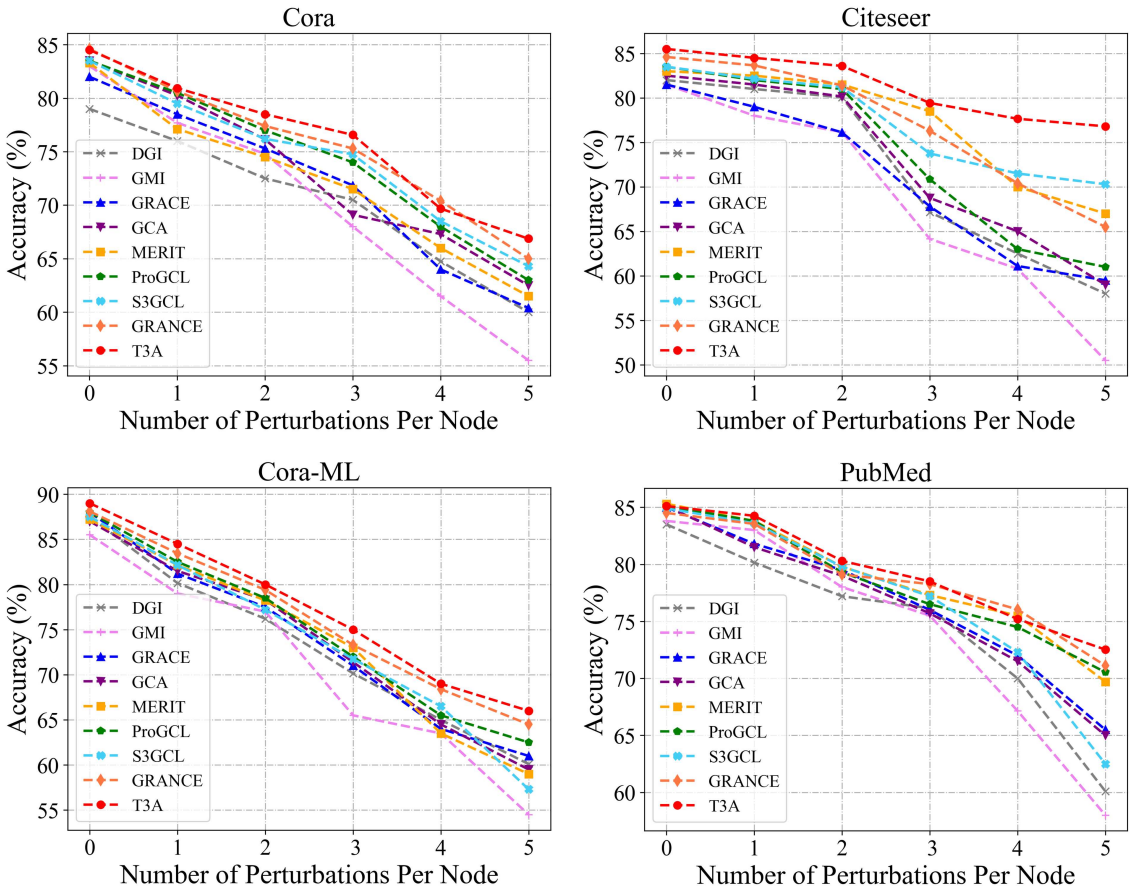

Figure 6 (Color online) Node classification accuracy under *nettack*, where the number of perturbations per node ranges from 0 to 5.

The experimental results under random attack are shown in Figure 7, where the perturbation rate is set from from 0 to 0.9. Our method exhibits better performance than baselines in most settings. On Citeseer, the average improvement of our method over the second-best method is the largest compared with that on other datasets under random attack, and the experimental results under *metattack* also exhibit the same observation. This may be because our method is more effective in processing graph data with higher-dimensional node features, which provide more consistent information to the tensorized graph aggregator.

5.2.4 *Parameter study*

The sensitivity of the augmentation times T is analyzed, and the performance is judged based on comparisons with various values of T under different perturbation rates. The value of T is varied from 8 to 24 in a stride of four. The experimental results under *nettack* are presented in Figure 8.

The value of T corresponding to the best performance is fixed to be 16 in all settings, which reflects the robustness of the model with respect to T. As T increases gradually, the performance gradually improves and then drops. A larger T means more essential information can be learned from augmented views. However, an excessive number of augmented graphs can add redundant information that degrades performance and makes optimization more difficult.

5.2.5 *Ablation study*

To further understand the contributions of each component in T3A, we designed four variants and report their experimental results on five datasets in Table 5. In T3A without (w/o) PTB, the original graph is not augmented by feature masking or edge dropping, and the augmented view is constructed directly with T original graphs. In T3A w/o TGA, the tensorized graph aggregator is replaced by T-independent GCNs to encode the tensorized augmented view. In T3A w/o CLM, the contrastive loss is performed in only one dimension. In T3AA w/o FLR, the f-product induced low-rank approximation is removed.

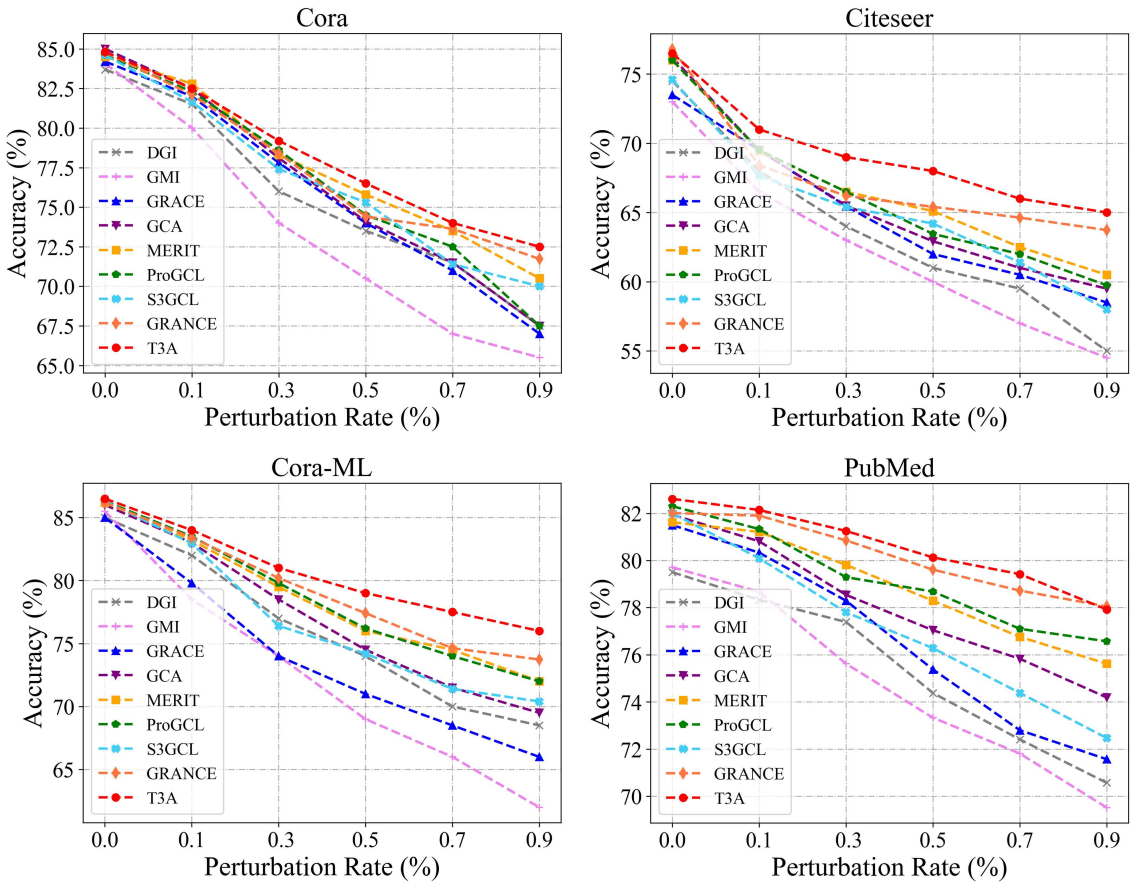

Figure 7 (Color online) Node classification accuracy under random attack, where the perturbation rate is from 0 to 0.9.

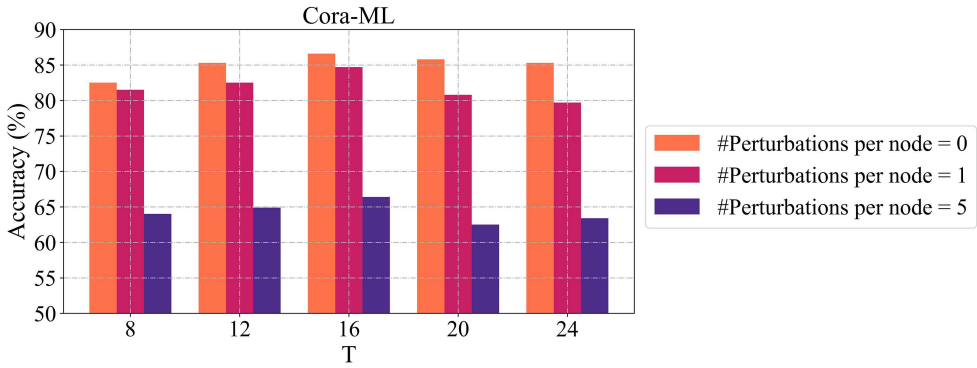

Figure 8 (Color online) Results of the parameter study with different T values.

According to observations, all the components of the model are significant. The tensorized graph aggregator is the most critical in the model, which proves the necessity of fusing multiple augmented graphs, where several GCNs cannot perform equally. In addition, the f-product-induced low-rank approximation for the tensor representation plays a secondary but important role, suggesting that the low-rank properties of the graphs are worth exploring and exploiting.

6 Conclusion

This study proposed a robust contrastive learning framework with multi-graph augmentation, employing a tensorized graph aggregator, an extended contrastive loss, and a low-rank approximation to encode, train, and maintain the low-rank structure of graphs. The paper provides a tensorization perspective to establish a connection between

Table 5 Results (%) of the ablation study on five datasets.

	Cora	Citeseer	Cora-ML	PubMed	Actor
T3A w/o PTB	82.60	71.89	84.97	79.58	25.42
T3A w/o TGA	72.01	62.74	67.91	71.47	22.98
T3A w/o CLM	82.89	72.28	85.24	81.35	28.13
T3A w/o FLR	82.45	72.02	84.68	80.14	27.63
T3A	83.81	73.01	86.38	82.41	28.81

tensor operators and the GraphCL framework. However, there are still some limitations in our work. For example, encoding tensor graphs with t-product operations suffers from scalability problems, which may be solved with other tensor operations. Future studies should design more efficient Fourier transform methods for the t-product by incorporating random projection and sparsification techniques to further reduce the computational complexity in the frequency domain. Incremental low-rank update algorithms must be explored, which enable the model to update tensor representations without full recomputation, thereby enhancing its practicality in large-scale graph scenarios. Moreover, the latent merits of the multi-graph structure should be studied, and the combinations of tensor techniques with graph learning should be explored. The extension of the T3A framework to inductive learning scenarios must also be explored to provide stronger scalability for the model.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2023YFB2703700), National Natural Science Foundation of China (Grant No. 62176269), and Guangzhou Science and Technology Program (Grant No. 2023A04J-0314).

References

1 Hamilton W L, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of the Advances in Neural Information Processing Systems, 2017. 1024–1034
2 Zhang M, Chen Y. Link prediction based on graph neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, 2018. 5171–5181
3 Zhuang S, Wu Z, Chen Z, et al. Refine then classify: robust graph neural networks with reliable neighborhood contrastive refinement. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2025. 13473–13482
4 Wang J, Li T R, Yang Y, et al. DiagLLM: multimodal reasoning with large language model for explainable bearing fault diagnosis. Sci China Inf Sci, 2025, 68: 160103
5 Chen J, Liao T, Chen C, et al. Improving message-passing GNNs by asynchronous aggregation. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024. 228–238
6 Dai H, Li H, Tian T, et al. Adversarial attack on graph structured data. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 1123–1132
7 Velickovic P, Fedus W, Hamilton W L, et al. Deep graph infomax. In: Proceedings of the 7th International Conference on Learning Representations, 2019
8 Zügner D, Akbarnejad A, Günnemann S. Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018. 6246–6250
9 Zhuo W, Tan G. Proximity enhanced graph neural networks with channel contrast. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, 2022. 2448–2455
10 Jin W, Ma Y, Liu X, et al. Graph structure learning for robust graph neural networks. In: Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2020. 66–74
11 Entezari N, Al-Sayouri S A, Darvishzadeh A, et al. All you need is low (rank): defending against adversarial attacks on graphs. In: Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining, 2020. 169–177
12 Peng Z, Huang W, Luo M, et al. Graph representation learning via graphical mutual information maximization. In: Proceedings of the Web Conference, 2020. 259–270
13 Zhu Y Q, Xu Y C, Yu F, et al. Deep graph contrastive representation learning. In: Proceedings of ICML Workshop on Graph Representation Learning and Beyond, 2020
14 Zhu Y, Xu Y, Yu F, et al. Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference, 2021. 2069–2080
15 Jin M, Zheng Y, Li Y, et al. Multi-scale contrastive siamese networks for self-supervised graph representation learning. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, 2021. 1477–1483
16 Xia J, Wu L, Wang G, et al. Progcl: rethinking hard negative mining in graph contrastive learning. In: Proceedings of the International Conference on Machine Learning, 2022. 24332–24346
17 Wu Z, Shu L, Xu Z, et al. Robust tensor graph convolutional networks via T-SVD based graph augmentation. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022. 2090–2099

18 Zhu D, Zhang Z, Cui P, et al. Robust graph convolutional networks against adversarial attacks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019. 1399–1407

19 Zhang X, Zitnik M. Gnnguard: defending graph neural networks against adversarial attacks. In: Proceedings of the Advances in Neural Information Processing Systems, 2020

20 Liao T, Wu Z, Chen C, et al. Tensor completion via convolutional sparse coding with small samples-based training. Pattern Recognition, 2023, 141: 109624

21 Kilmer M E, Martin C D. Factorization strategies for third-order tensors. Linear Algebra Its Appl, 2011, 435: 641–658

22 Zhou P, Lu C, Lin Z, et al. Tensor factorization for low-rank tensor completion. IEEE Trans Image Process, 2017, 27: 1152–1163

23 Luo Y, Zhao X L, Meng D, et al. Hlrtf: hierarchical low-rank tensor factorization for inverse problems in multi-dimensional imaging. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022. 19281–19290

24 Liao T, Yang J, Chen C, et al. A neural tensor decomposition model for high-order sparse data recovery. Inf Sci, 2024, 658: 120024

25 Cheng X, Miao Z, Qiu Q. Graph convolution with low-rank learnable local filters. In: Proceedings of the 9th International Conference on Learning Representations, 2021

26 Xu H, Xiang L, Yu J, et al. Speedup robust graph structure learning with low-rank information. In: Proceedings of the 30th ACM International Conference on Information; Knowledge Management, 2021. 2241–2250

27 Wang Y, Yang Y. Low-rank graph contrastive learning for node classification. 2024. ArXiv:2402.09600

28 Yang L, Shi R, Zhang Q, et al. Self-supervised graph neural networks via low-rank decomposition. In: Proceedings of Advances in Neural Information Processing Systems, 2023. 36: 34295–34307

29 Entezari N, Al-Sayouri S A, Darvishzadeh A, et al. All you need is low (rank) defending against adversarial attacks on graphs. In: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020. 169–177

30 Yang Z R, Han J, Wang C D, et al. Graphlora: structure-aware contrastive low-rank adaptation for cross-graph transfer learning. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2025. 1785–1796

31 Kolda T G, Bader B W. Tensor decompositions and applications. SIAM Rev, 2009, 51: 455–500

32 You Y, Chen T, Sui Y, et al. Graph contrastive learning with augmentations. In: Proceedings of the Advances in Neural Information Processing Systems, 2020

33 McCallum A K, Nigam K, Rennie J, et al. Automating the construction of Internet portals with machine learning. Inf Retrieval, 2000, 3: 127–163

34 Giles C L, Bollacker K D, Lawrence S. Citeseer: an automatic citation indexing system. In: Proceedings of the 3rd ACM International Conference on Digital Libraries, 1998. 89–98

35 Bojchevski A, Günnemann S. Deep Gaussian embedding of graphs: unsupervised inductive learning via ranking. In: Proceedings of the 6th International Conference on Learning Representations, 2018

36 Sen P, Namata G, Bilgic M, et al. Collective classification in network data. AI Mag, 2008, 29: 93–106

37 Pei H, Wei B, Chang K C C, et al. Geom-gcn: geometric graph convolutional networks. In: Proceedings of ICLR, 2020

38 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of ICLR, 2017

39 Wu F, Jr. A H S, Zhang T, et al. Simplifying graph convolutional networks. In: Proceedings of the 36th International Conference on Machine Learning, 2019. 6861–6871

40 Veličković P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of the ICLR, 2018

41 Chen Z, Wu Z, Sadikaj Y, et al. ADEdgeDrop: adversarial edge dropping for robust graph neural networks. IEEE Trans Knowl Data Eng, 2025, 37: 4948–4961

42 Wan G, Tian Y, Huang W, et al. S3gcl: spectral, swift, spatial graph contrastive learning. In: Proceedings of the Forty-first International Conference on Machine Learning, 2024

43 Zügner D, Günnemann S. Adversarial attacks on graph neural networks via meta learning. In: Proceedings of the 7th International Conference on Learning Representations, 2019