Robust Tensor Graph Convolutional Networks via T-SVD based Graph Augmentation

Zhebin Wu
Sun Yat-Sen University
Guangzhou, China
wuzhb6@mail2.sysu.edu.cn

Lin Shu
Sun Yat-Sen University
Guangzhou, China
shulin@mail2.sysu.edu.cn

Ziyue Xu
Sun Yat-Sen University
Guangzhou, China
xuzy53@mail2.sysu.edu.cn

Yaomin Chang
Sun Yat-Sen University
Guangzhou, China
changym3@mail2.sysu.edu.cn

Chuan Chen
Sun Yat-Sen University
Guangzhou, China
chenchuan@mail.sysu.edu.cn

Zibin Zheng
Sun Yat-Sen University
Zhuhai, China
zhzibin@mail.sysu.edu.cn

ABSTRACT

Graph Neural Networks (GNNs) have exhibited their powerful ability of tackling nontrivial problems on graphs. However, as an extension of deep learning models to graphs, GNNs are vulnerable to noise or adversarial attacks due to the underlying perturbations propagating in message passing scheme, which can affect the ultimate performances dramatically. Thus, it's vital to study a robust GNN framework to defend against various perturbations. In this paper, we propose a Robust Tensor Graph Convolutional Network (RT-GCN) model to improve the robustness. On the one hand, we utilize multi-view augmentation to reduce the augmentation variance and organize them as a third-order tensor, followed by the truncated T-SVD to capture the low-rankness of the multi-view augmented graph, which improves the robustness from the perspective of graph preprocessing. On the other hand, to effectively capture the inter-view and intra-view information on the multi-view augmented graph, we propose tensor GCN (TGCN) framework and analyze the mathematical relationship between TGCN and vanilla GCN, which improves the robustness from the perspective of model architecture. Extensive experimental results have verified the effectiveness of RT-GCN on various datasets, demonstrating the superiority to the state-of-the-art models on diverse adversarial attacks for graphs.

KEYWORDS

graph neural networks, t-SVD, node classification, graph augmentation, robustness

ACM Reference Format:
Zhebin Wu, Lin Shu, Ziyue Xu, Yaomin Chang, Chuan Chen, and Zibin Zheng. 2022. 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 (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539436

KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539436

1 INTRODUCTION

Graphs are ubiquitous in a wide variety of real-world scenarios, ranging from social graphs [20, 35] to citation graphs [7, 27], biological graphs [18, 31], e-commerce graphs [22, 34], etc. Analyzing and mining valuable information from underlying graph data has been actively researched both in academia and industry. Among various graph mining techniques, graph neural networks (GNNs) [29], which explore neural networks for graphs, have attracted considerable attentions in recent years. State-of-the-art GNNs [10, 17, 33] follow the message-passing scheme, where node representations are generated by aggregating information from neighbors iteratively. Due to the striking performances, GNNs have become powerful tools in graph analysis and are widely applied to various downstream tasks such as node classification [26, 41].

However, recent researches have shown that GNNs are vulnerable to noise and adversarial attacks [4, 6, 45], i.e., subtle perturbations on node attributes and graph structures. Specifically, owing to the iterative message-passing scheme of GNNs, small perturbations in the graph are propagated to neighborhoods and further affect node representations in a wider range, which drastically degrade models' performance on downstream tasks. The lack of robustness of traditional GNNs has become a critical issue in applications of many real-world scenarios. Consider an e-commerce graph as an example, where nodes represent users and edges represent transactions. Fraudulent users created by a hacker constitute fake transactions to real users, which can easily propagate harmful information in the whole graph and mislead GNN models into estimating user creditability inaccurately. Therefore, it is of crucial importance to design GNN models that are robust against adversarial attacks.

A common solution for improving the robustness of GNN models is to generate denoised graphs by graph augmentation [28], such as adding/dropping edges, changing edge weights, etc. By employing graph augmentations, some spurious structures/attributes might be eliminated while the potential ones might be discovered and exploited, thus facilitating defending against adversarial attacks. For example, Entezari *et al.* [8] utilize singular value decomposition (SVD) to vaccinate GNN model. Wu *et al.* [37] recompute the edge weights between nodes by using Jaccard similarity or

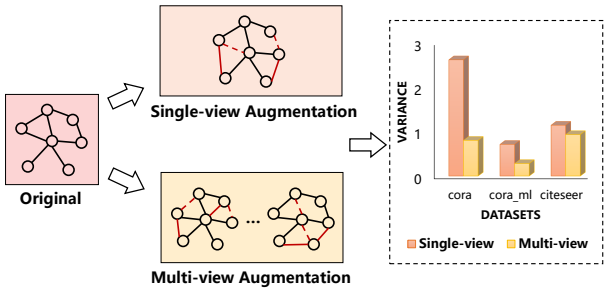
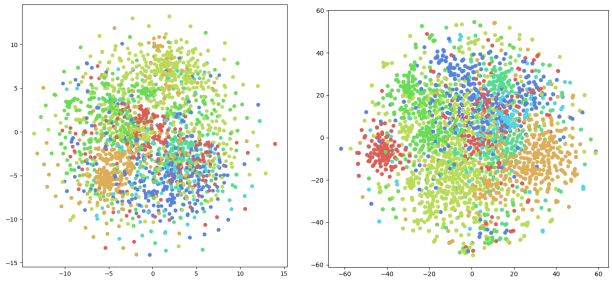


Figure 1: Case study and illustrations of differences between single-view and multi-view augmentation. Red solid lines represent the added edges and red dashed lines represent the dropped edges. The histogram depicts the variance of accuracy on node classification by utilizing single-view/multi-view augmentation respectively.



(a) Without LRA (b) With LRA

Figure 2: The visual results of Cora dataset using t-SNE [32] with/without low rank approximation (LRA) under *metattack* with 25% perturbation rate.

cosine similarity. The aforementioned methods focus on single-view augmentation, i.e., they generate only one augmented graph. However, we argue that single-view augmentation introduces high augmentation variance for learning node representations, which can be reduced by multi-view augmentation, i.e., generating multiple augmented graphs simultaneously. Fig. 1 illustrates an example of single-view and multi-view augmentation with adding and dropping edges, where red solid lines represent the added edges and red dashed lines represent the dropped edges. Empirically, single-view augmentation adds/drops edges in only one graph, which has large randomness and thus introduces high augmentation variance. On the contrary, multi-view augmentation simultaneously generates multiple graphs with different added/dropped edges, which is equivalent to the regularization of augmented graphs, thus having lower variance and stronger robustness. To verify our hypothesis, we further conduct a case study on three datasets. Concretely, we feed graphs generated by single-view and multi-view augmentation into GCN respectively, and perform node classification with the learned node embeddings, where accuracy is employed to measure the performance. Note that each graph generated by multi-view augmentation is trained through an independent GCN, and the ultimate accuracy is the average prediction of all GCNs. We repeat each experiment 10 times and the variance of accuracy is shown in the histogram of Fig. 1. Obviously, the variance of performances on multi-view augmentation is much lower than single-view augmentation. Hence, it is essential to utilize multi-view augmentation to reduce the augmentation variance, which helps improve the robustness of graph augmentation.

Besides, as most individuals in real-world graphs are tend to be connected with a small number of neighbors, forming some united communities, low-rankness remains a universal property of the graph's topological structure [13, 42]. And note that current successful adversarial attacks for graphs mainly increase the number of neighbors dissimilar to the target nodes, which correspond to the high-rank component of graph data [8]. Thus, the low-rank approximation is a simple but effective method to eliminate perturbations. As shown in Fig. 2, the distribution of nodes in Cora dataset [25] under *metattack* [46] with 25% perturbation rate, approximated by low rank has a more clear community relationship than the other one. As a consequence, how to employ multi-view augmentation to reduce the augmentation variance as well as maintain the low-rank property plays a significant role in defending against adversarial attacks, which improves the robustness from the perspective of graph preprocessing.

With multi-view augmented graphs as input, modern GNNs, e.g., the most representative Graph Convolutional Network (GCN) [17], can only process multiple augmented graphs one after another separately and aggregates the multi-view embeddings at last, which neglects the correlation among different views in graph convolution operator. Augmented from the same original graph, the graphs of different views are supposed to have similar low-rank components, i.e., they are low-rank across the view dimensions. Thus, it's vital to capture the low-rank inter-view information to improve the robustness from the perspective of model architecture.

In order to address the aforementioned problems, we propose Robust Tensor Graph Convolutional Networks (RT-GCN) in this paper. Concretely, we firstly acquire multi-view augmented graphs by utilizing random edge-adding/dropping technique, and utilize Tensor Singular Value Decomposition (T-SVD) to approximate the augmented graphs with low-rankness to improve their robustness. Then, tensor GCN (TGCN) is proposed to convolute the multi-view graph in inter-view and intra-view simultaneously to learn a more robust representation.

The contributions of this paper are summarized as follows:

- We propose a T-SVD based graph augmentation method, which obtains a low tubal rank approximation of a multi-view graph augmented by random edge dropping/adding technique.
- To learn the inter-view and intra-view information of augmented multi-view graph simultaneously, we propose a TGCN model and analyze the relationship between TGCN and vanilla GCN.
- Extensive experimental results on various datasets demonstrate the effectiveness of the proposed model and the superiority to the state-of-the-art methods on different kinds of adversarial attacks for graphs.

The rest of this paper is organized as follows. In Section 2, we introduce some relevant works. In Section 3, we introduce the notations and some preliminaries used in this paper. We depict our framework elaborately in Section 4 and show our experimental results in Section 5. Finally, we conclude this paper in Section 6.

2 RELATED WORK

2.1 Graph Neural Networks

With the rapid development of deep learning, graph neural networks (GNNs) have gained growing interests in recent years, which extend neural networks for graph-structured data. Owing to the striking performance, GNNs have a wide variety of applications in real-world scenarios. Representative methods of GNNs include GCN [17], which designs a layer-wise propagation rule based on a first-order approximation of spectral graph convolutions. Simple Graph Convolution (SGC) [36] reduces the GCN layer to a single linear transformation and still achieves competitive performance. GraphSAGE [10] samples and aggregates features from neighbors to generate node representations. Graph Attention Network (GAT) [33] employs a self-attention mechanism to measure contributions of different neighbors when aggregating information. Nevertheless, traditional GNNs learn representations by aggregating neighbors' information iteratively, which are vulnerable to adversarial attacks and lack in robustness.

2.2 GNNs for Defending Adversarial attacks

To address the fragility problem of GNNs, many researches are proposed recently and achieve incredible performance. GCN-Jaccard [45] and GCN-SVD [8] utilize corresponding graph augmentation methods with prior knowledge, e.g., Jaccard similarity, cosine similarity, low-rankness, etc., to obtain a considerably clean graph from the poisoned one. GNNGUARD [38] estimates the neighbor importance and introduces a layer-wise memory mechanism in the training procedure to reduce the weights of suspicious edges. RGCN [44] adopts Gaussian distribution as hidden node representations in the convolutional layer to absorb the changes of adversarial attacks in the variance of Gaussian distribution automatically. ProGNN [13] defends adversarial attacks by imposing low-rankness, sparsity and feature smoothness regularizations on the objective function to constraint the target graph. Though tremendous achievements have been made by the above methods, they hardly consider reducing the augmentation variance or contingency introduced by adversarial attacks.

2.3 Low-rankness in Data Mining

Low-rankness is a universal property of most matrices, which is able to depict the correlations among rows or columns. Besides, low-rankness is widely applied in many data mining tasks, e.g., matrix completion [12], robust principal component analysis [2], etc. However, the low-rankness of matrix can hardly capture the information across higher dimensions when the data is beyond 2-D.

A tensor is often known as an extension of a 1-D array or 2-D matrix, which can offer a high-dimensional storage structure for various data. Thus, it's a powerful tool to process and analyze the multi-dimensional data [3, 5, 21]. Kilmer *et al.* [15] proposed a novel tensor-tensor product (t-product) instead of traditional mode-n matrix multiplication. And based on this, the tensor singular value decomposition (T-SVD) is further proposed to study the low-rankness of tensor data globally. Recently, T-SVD is widely used in multi-dimensional data mining and has achieved incredible performance, e.g., robust tensor completion problem [30, 39, 43], tensor robust principal component analysis [23], etc.

Some recent works have introduced the tensor algebraic operations into the GNN schemes to process the dynamic graphs [24] or multi-relational graph [11]. However, since the tensor-based operators remain unexplored, the relationship between the proposed tensor-based GCN and vanilla GCN [17] remains unclear.

3 NOTATIONS AND PRELIMINARIES

3.1 Notations

In this paper, a graph is defined as $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is the set of N nodes and $E \subset V \times V$ denotes the set of edges between nodes. Let $A \in \mathbb{R}^{N\times N}$ denotes the adjacency matrix and $D_{i,i} = \sum_j A_{i,j}$ represents the diagonal degree matrix, where $A_{i,j}$ indicates that node i and node j are neighbours if $A_{i,j} = 1$ else $A_{i,j} = 0$. We denote the node feature matrix as $X \in \mathbb{R}^{N\times F}$, where F is the dimension of the node features. Alternatively, the graph can be also represented as $G = (A, X)$.

Tensors are symbolized by Euler letters, e.g., let $\mathcal{G} \in \mathbb{R}^{N\times N\times T}$ is a third-order tensor, which denotes a T-view graph. Each frontal slice $\mathcal{G}(:, :, i) \in \mathbb{R}^{N\times N}$ of $\mathcal{G}$ represents the i-th-view graph, which can be denoted as $G^{(i)}$ for simplicity.

3.2 Graph Neural Networks

GNNs have received great attentions in recent years due to their characteristics of solving nontrivial problems on graph data. However, in this paper, we focus on the most popular and effective GNN model proposed by Kipf *et al.* [17], GCN, for semi-supervised node classifications. Besides, we extend the matrix-based propagation framework to tensor-based one in this paper. Specifically, GCN in [17] is of two GCN layers, whose propagation function f_θ can be formulated as:

$$f_\theta(A, X) = \text{softmax}(\hat{A}\sigma(\hat{A}XW_1)W_2), \tag{1}$$

where $\hat{A} = \hat{D}^{-\frac{1}{2}}(A + I)\hat{D}^{-\frac{1}{2}}$ represents the symmetric normalized $A+I$ with added self-loops, $\hat{D}_{ii} = 1+\sum_j A_{ij}$ is the diagonal element of degree matrix $\hat{D}$ correspondingly, and $\sigma(\cdot)$ denotes a nonlinear activation, e.g., $ReLU(\cdot)$. θ is the parameters set including W_1 and W_2.

3.3 T-product

For tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, we define two operators *unfold* and *fold* as:

$$\text{unfold}(\mathcal{A}) = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n_3)} \end{bmatrix}, \text{fold}(\text{unfold}(\mathcal{A})) = \mathcal{A},$$

where the *unfold* operator flattens tensor $\mathcal{A}$ to a matrix of size $n_1n_3 \times n_2$ and *fold* is its inverse operator. Also, we define the block

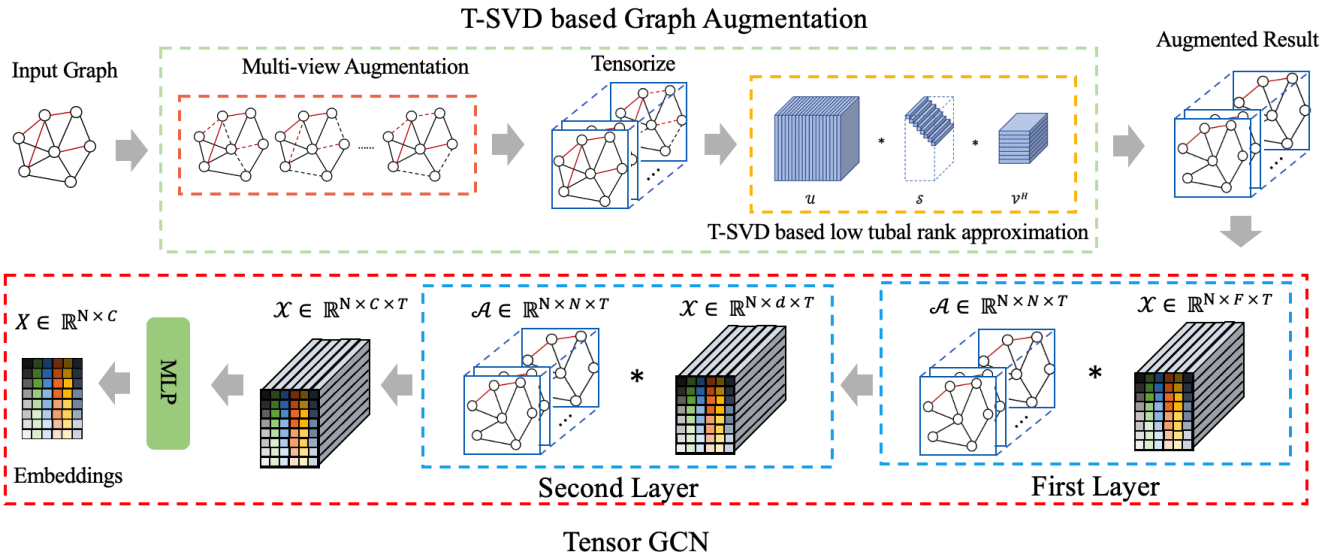


Figure 3: The framework of RT-GCN, including T-SVD based graph augmentation and tensor GCN.

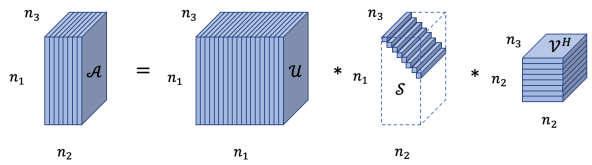


Figure 4: Illustration of T-SVD.

circulant matrix $bcirc(\mathcal{A}) \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}$ of tensor $\mathcal{A}$ as:

$$\text{bcirc}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & A^{(n_3)} & \dots & A^{(2)} \\ A^{(2)} & A^{(1)} & \dots & A^{(3)} \\ \vdots & \vdots & \vdots & \vdots \\ A^{(n_3)} & A^{(n_3-1)} & \dots & A^{(1)} \end{bmatrix}.$$

Given another tensor $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$, then the t-product $\mathcal{A} * \mathcal{B}$ is defined as [15]:

$$\mathcal{C} = \mathcal{A} * \mathcal{B} = \text{fold}(\text{bcirc}(\mathcal{A}) \cdot \text{unfold}(\mathcal{B})) \tag{2}$$

where $\mathcal{C} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$. From the matrix multiplication perspective, the third-order tensor $\mathcal{A}$ can be regarded as a matrix of size $n_1 \times n_2$ with each element being a tube that lies in the third dimension, where a tube of tensor is defined as a vector of size $1 \times 1 \times n_3$. Note that t-product can be implemented by Discrete Fourier Transform for saving computational resource [15], which is depicted in Appendix A due to the page limitation.

3.4 Tensor Singular Value Decomposition

Matrix SVD is one of the most popular matrix decomposition methods, which can be mathematically formulated as:

$$A = U \Sigma V^T, \tag{3}$$

where $A \in \mathbb{R}^{n_1 \times n_2}$ is the underlying matrix, $U \in \mathbb{R}^{n_1 \times n_1}$ and $V \in \mathbb{R}^{n_2 \times n_2}$ are orthogonal matrices consisting of singular vectors, and $(\cdot)^T$ denotes transpose. Σ is a diagonal matrix whose diagonal elements are sorted in descending order.

Analogous to the matrix SVD, T-SVD [14, 15] is proposed based on the aforementioned t-product. Given $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and it can be decomposed as:

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^H, \tag{4}$$

where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$, $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are orthogonal tensors [15], i.e., $\mathcal{U} * \mathcal{U}^H = \mathcal{I}$ and $\mathcal{V} * \mathcal{V}^H = \mathcal{I}$. Each frontal slice of $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is diagonal and $(\cdot)^H$ means conjugate transpose.

Fig. 4 shows the decomposition process of T-SVD intuitively. Note that when $n_3 = 1$, T-SVD can degenerate to matrix SVD.

Definition 1 (Tensor Tubal Rank). *[14] For a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, it's tubal rank is defined as the number of nonzero singular tubes of $\mathcal{S}$, where $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^H$. Normally, $r < \min(n_1, n_2)$ when low tubal rank-r approximation are conducted. For $j > r$, the element of $S_{jj}^{(i)}$ will be zero.*

4 PROPOSED FRAMEWORK

In this section, we will introduce our proposed RT-GCN model through two perspectives of robustness on graph preprocessing and model architecture, respectively.

4.1 T-SVD Based Graph Augmentation

4.1.1 Graph Augmentation. Data augmentation is an effective approach to artificially enlarge the diversity of dataset by using various translations without modifying the model architecture but improving the generalization of the whole model. Random erasing [40] has acquired magnificent achievements in image classification by randomly erasing pixels within the images. Albeit simple, it

outperforms many convolutional neural networks (CNN) based benchmarks. Considering the underlying relation between CNNs and GNNs, we utilize random edge dropping/adding technology analogously in our model.

Given a target graph $G = (A, X)$, random edge dropping/adding is conducted in preprocessing stage. Each edge in the graph is dropped or added with a certain probability p to obtain the augmented graph. In order to reduce the augmentation variance, we utilize multi-view augmentation to produce $T-1$ augmented graphs. The original target graph and $T-1$ augmented graphs can be constructed to a third-order tensor $\mathcal{G} = (\mathcal{A}, \mathcal{X})$ of size $N \times N \times T$ by folding them in the third order, where $\mathcal{A} \in \mathbb{R}^{N\times N\times T}$ represents an adjacency tensor and each frontal slice $A^{(i)} (i \in [1, T])$ of it is the corresponding single-view adjacency matrix. Similarly, $\mathcal{X} \in \mathbb{R}^{N\times F\times T}$ represents T-view node feature tensor.

4.1.2 Low Tubal Rank Tensor Approximation. It is vital to denoise before graphs are input to the downstream models as slight perturbations can reduce the performance dramatically. Noises or adversarial attacks for graphs usually impose unnoticeable perturbations on nodes or edges, which is tend to correspond to the high-rank component of the graph [8]. And it has been verified that real-world graphs are naturally low-rank [13, 42] because wherein nodes are often gathered in cluster way according to a common topic or belonging to the same community.

Hence, to discard high-rank noises or perturbations in graphs, low-rank approximation has been verified as an effective way to improve robustness in [8, 13]. Truncated SVD of matrices is a prominent approach applied in dimension reduction to extract the low-rank component of matrix data. Usually, computing the rank-r approximation of matrix A can be formulated as:

$$A_r = U_r \Sigma_r V_r^T = \sum_{i=1}^{r} u_i \sigma_i v_i^T, \tag{5}$$

where r is rank of matrix A_r derived from SVD, U_r and V_r denotes the top r singular vectors of matrix U and V, respectively. Σ_r is the diagonal matrix who only contains top r elements of singular values.

Nevertheless, when the structure of data gets beyond 2-D, traditional SVD can hardly consider the low-rankness along different dimensions. For augmented multi-view graphs, the low-rankness of not only the frontal slices but also the lateral slices can be studied. Thus, to capture the inter-view consistency and intra-view neighbor information simultaneously, T-SVD is served as the most promising method according to the entangling ability of t-product. Let $\mathcal{A} \in \mathbb{R}^{N\times N\times T}$ represent the augmented adjacency tensor and it can be low tubal rank approximated as:

$$\mathcal{A}_r = \mathcal{U}_r * \mathcal{S}_r * \mathcal{V}_r^H, \tag{6}$$

where r is the tubal rank of tensor $\mathcal{A}_r$. $\mathcal{U}_r$, $\mathcal{S}_r$ and $\mathcal{V}_r$ are similar to matrix ones by regarding each tensor tube as an element in the first frontal slice.

4.2 T-product Based Tensor GCN

Inspired by the most popular spatial GCN [17] on single-view graph for its effectiveness and heuristics, this paper extends the dimension of model to third order, called tensor GCN (TGCN). Existing GCN models can only process the multi-view graph one view after another separately and fuse the inter-view information at the final step, which ignores the correlation among different views in graph convolution operator. As a result, it's inevitable to miss a number of key features during the propagation stage due to the activation function, which decreases the robustness of model.

Let $\mathcal{A} \in \mathbb{R}^{N\times N\times T}$ denotes the adjacency tensor of a T-view graph with each frontal slice $A^{(i)} = \mathcal{A}(:, :, i)$ representing a single-view graph. Correspondingly, $\hat{\mathcal{D}} \in \mathbb{R}^{N\times N\times T}$ is the diagonal degree tensor with each frontal slice $\hat{D}^{(i)}$ representing the diagonal degree matrix of $\hat{A}^{(i)} = A^{(i)} + I$. Also, $\mathcal{X} \in \mathbb{R}^{N\times F\times T}$ is the feature tensor with each frontal slice $X^{(i)}$ representing the i-th afeature matrix of i-th-view graph. Hence, we reformulate the aforementioned two-layer GCNs' propagation function Eq. (1) in the tensor way as:

$$f_\theta(\mathcal{A}, \mathcal{X}) = \text{softmax}((\hat{\mathcal{A}} * \sigma(\hat{\mathcal{A}} * \mathcal{X} * \mathcal{W}_1) * \mathcal{W}_2) \times_3 W_3), \tag{7}$$

where * denotes the t-product, $\hat{\mathcal{A}} = \text{fold}(\hat{D}^{(i)^{-\frac{1}{2}}} (\mathcal{A}+\mathcal{I})^{(i)} \hat{D}^{(i)^{-\frac{1}{2}}})$, $\times_3$ denotes matrix multiplication along the third dimension and θ is the set of parameters including weights $\mathcal{W}_1$, $\mathcal{W}_2$ and W_3. Different from tensor weights $\mathcal{W}_1 \in \mathbb{R}^{F\times H\times T}$ and $\mathcal{W}_2 \in \mathbb{R}^{H\times K\times T}$, where H is the dimension of embeddings of hidden layer and K is the output size, matrix weight W_3 is used to aggregate the final embeddings, which a vector of size $T \times 1$ indeed.

4.3 Analysis of Relationship Between Vanilla GCN and Tensor GCN

It's well-known that message passing framework is GNNs' pillar. Kipf *et al.* [17] proposed a fast approximate convolution on graphs in the spatial domain (for sake of distinction, we named it vanilla GCN in this paper), which is a linear formulation indeed. Especially, t-product is t-linear combination of tensors from the tube-wise perspective [14], which inspires us to study not only TGCN but also the possible mathematical relationships between vanilla GCN and TGCN.

Considering the fact that t-product can be computed by linear DFT according to Appendix A, it's intuitive that the embedding of TGCN is not a linear combination of embeddings of vanilla GCNs. In other words, the result of t-product is not simply the linear combination of the products of matrix multiplications between each corresponding frontal slices in the spatial domain. To prove this, we take the above notations as an example and the node embedding in final layer of vanilla GCN is:

$$H_2^{(i)} = \hat{A}^{(i)} \sigma(\hat{A}^{(i)} X^{(i)} W_1^{(i)}) W_2^{(i)}, \tag{8}$$

where i denotes the i-th graph, i.e., the number of GCNs needed is T. Similarly, the node embedding in final layer of TGCN is:

$$\mathcal{H}_2 = \hat{\mathcal{A}} * \sigma(\hat{\mathcal{A}} * \mathcal{X} * \mathcal{W}_1) * \mathcal{W}_2. \tag{9}$$

On account of the performance of nonlinear activation function $\sigma(\cdot)$, i.e., $ReLu(\cdot)$, it's apparent that the embeddings in the final layer are not linear correlation. However, when the activation function is linear or there are no activation functions in GCN and TGCN, the embedding obtained by TGCN is still not a linear combination of ones obtained by T vanilla GCNs indeed. To verify our proposition, we just consider the embeddings in the first layer without activation

function, without losing generality. Supposing there exists a linear transform matrix Q of size $T \times T$, which satisfies the following equation:

$$\mathcal{H}_1 = \hat{\mathcal{A}} * \mathcal{X} * \mathcal{W}_1 = \text{fold}(\hat{A}^{(i)} X^{(i)} W_1^{(i)}) \times_3 Q. \tag{10}$$

Specifically, by ignoring weights momentarily, the element-wise tensor graph convolution operator in spatial domain can be performed as:

$$\begin{aligned}
&\text{unfold}(\mathcal{C}) = \text{unfold}(\mathcal{A} * \mathcal{X}) = \text{bcirc}(\mathcal{A}) \cdot \text{unfold}(\mathcal{X}) \\
&= \begin{bmatrix} A^{(1)} & A^{(T)} & \cdots & A^{(2)} \\ A^{(2)} & A^{(1)} & \cdots & A^{(3)} \\ \vdots & \vdots & \vdots & \vdots \\ A^{(T)} & A^{(T-1)} & \cdots & A^{(1)} \end{bmatrix} \cdot \begin{bmatrix} X^{(1)} \\ X^{(2)} \\ \vdots \\ X^{(T)} \end{bmatrix} \\
&= \begin{bmatrix} A^{(1)}_{1,:} X^{(1)}_{:,1} + \cdots + A^{(2)}_{1,:} X^{(T)}_{:,1}, \cdots, A^{(1)}_{1,:} X^{(1)}_{:,F} + \cdots + A^{(2)}_{1,:} X^{(T)}_{:,F} \\ A^{(1)}_{2,:} X^{(1)}_{:,1} + \cdots + A^{(2)}_{2,:} X^{(T)}_{:,1}, \cdots, A^{(1)}_{2,:} X^{(1)}_{:,F} + \cdots + A^{(2)}_{2,:} X^{(T)}_{:,F} \\ \vdots \\ A^{(T)}_{N,:} X^{(1)}_{:,1} + \cdots + A^{(1)}_{T,:} X^{(T)}_{:,1}, \cdots, A^{(T)}_{N,:} X^{(1)}_{:,F} + \cdots + A^{(1)}_{N,:} X^{(T)}_{:,F} \end{bmatrix},
\end{aligned} \tag{11}$$

where $\mathcal{C}$ and $C^{(i)}$s denote the results of TGCN. According to Eq. (11), note that the element of $C^{(i)}$, e.g., $C^{(1)}_{m,n} = A^{(1)}_{m,:} X^{(1)}_{:,n} + A^{(T)}_{m,:} X^{(2)}_{:,n} + \cdots + A^{(2)}_{m,:} X^{(T)}_{:,n}$, is entangled by all the frontal slice of $\mathcal{A}$ and $\mathcal{X}$. What's more, $C^{(2)}_{m,n} = A^{(2)}_{m,:} X^{(1)}_{:,n} + A^{(1)}_{m,:} X^{(2)}_{:,n} + \cdots + A^{(3)}_{m,:} X^{(T)}_{:,n}$ is another combination between $A^{(i)}$s and $X^{(i)}$s. In a word, all these $C^{(i)}$s follow a circulant formulation so as to fuse inter-view information roundly. The vector $C^{(i)}_{m,:} = [A^{(i)}_{m,:} X^{(1)}_{:,1} + \cdots + A^{(i+1)}_{m,:} X^{(T)}_{:,1}, \cdots, A^{(i)}_{m,:} X^{(1)}_{:,F} + \cdots + A^{(i+1)}_{m,:} X^{(T)}_{:,F}]$ is able to regarded as the underlying embedding of node m in i-th view.

However, vanilla spatial graph convolution performs as matrix multiplication on a single-view graph as:

$$\bar{C}^{(i)} = A^{(i)} X^{(i)}, \tag{12}$$

where $\bar{C}^{(i)}$s denote the results of vanilla GCNs. Then the element-wise formulation is $\bar{C}^{(i)}_{m,n} = A^{(i)}_{m,:} X^{(i)}_{:,n}$, which only consists of the corresponding row and column information of A and X in i-th view, respectively. And the embedding of node m in i-th view is $\bar{C}^{(i)}_{m,:} = [A^{(i)}_{m,:} X^{(i)}_{:,1}, \cdots, A^{(i)}_{m,:} X^{(i)}_{:,F}]$. Transform fold$(\bar{C}^{(i)})$ to result tensor $\hat{\mathcal{C}}$ by multiplying Q along the third dimension:

$$\hat{\mathcal{C}} = \text{fold}(\bar{C}^{(i)}) \times_3 Q = \begin{bmatrix} \bar{C}^{(1)}_{:,1}, \cdots, \bar{C}^{(T)}_{:,1} \\ \bar{C}^{(1)}_{:,2}, \cdots, \bar{C}^{(T)}_{:,2} \\ \vdots \\ \bar{C}^{(1)}_{:,F}, \cdots, \bar{C}^{(T)}_{:,F} \end{bmatrix} \times \begin{bmatrix} Q_{1,1}, \cdots, Q_{1,T} \\ Q_{2,1}, \cdots, Q_{2,T} \\ \cdots \\ Q_{T,1}, \cdots, Q_{T,T} \end{bmatrix}, \tag{13}$$

where the embedding of node m in i-th view can be computed by:

$$\begin{aligned}
\hat{C}^{(i)}_{m,:} &= \left(\begin{bmatrix} \bar{C}^{(1)}_{m,1}, \cdots, \bar{C}^{(T)}_{m,1} \\ \bar{C}^{(1)}_{m,2}, \cdots, \bar{C}^{(T)}_{m,2} \\ \vdots \\ \bar{C}^{(1)}_{m,F}, \cdots, \bar{C}^{(T)}_{m,F} \end{bmatrix} \times \begin{bmatrix} Q_{1,i} \\ Q_{2,i} \\ \vdots \\ Q_{T,i} \end{bmatrix} \right)^T \\
&= \begin{bmatrix} A^{(1)}_{m,:} X^{(1)}_{:,1} \times Q_{1,i} + A^{(2)}_{m,:} X^{(2)}_{:,1} \times Q_{2,i} + \cdots + A^{(T)}_{m,:} X^{(T)}_{:,1} \times Q_{T,i} \\ A^{(1)}_{m,:} X^{(1)}_{:,2} \times Q_{1,i} + A^{(2)}_{m,:} X^{(2)}_{:,2} \times Q_{2,i} + \cdots + A^{(T)}_{m,:} X^{(T)}_{:,2} \times Q_{T,i} \\ \vdots \\ A^{(1)}_{m,:} X^{(1)}_{:,F} \times Q_{1,i} + A^{(2)}_{m,:} X^{(2)}_{:,F} \times Q_{2,i} + \cdots + A^{(T)}_{m,:} X^{(T)}_{:,F} \times Q_{T,i} \end{bmatrix}^T
\end{aligned} \tag{14}$$

Comparing $C^{(i)}_{m,:}$ with $\hat{C}^{(i)}_{m,:}$, generally speaking, there doesn't exist a linear transform matrix Q to transform fold$(\bar{C}^{(i)})$ to $\mathcal{C}$, generically (meaning that the conclusion remains true except for a set of measure zero with respect to the Lebesgue measure.) The extreme situation remains when all the feature matrices $X^{(i)}$ are the same, all the elements of Q are equal to 1. In a word, the embedding of TGCN is a linear combination of ones of vanilla GCNs, only when the activation function is linear or there is no activation functions, and all the feature matrices are the same.

Proposition 4.1. *Let $\mathcal{A} \in \mathbb{R}^{N \times N \times T}$ be a T-view graph and each frontal slice $A^{(i)} \in \mathbb{R}^{N \times N}$ is the i-th-view graph. Then the embedding obtained by TGCN is generically a nonlinear combination of ones learned by an independent vanilla GCN from each view with respect to the Lebesgue measure.*

In addition, the vector $C^{(i)}_{m,:}$ illustrates that TGCN is able to capture the inter-view information against vanilla GCN. That is, each frontal slice of $\mathcal{H}_1$, e.g., the aforementioned simple embedding $C^{(i)}_{m,:} = [A^{(i)}_{m,:} X^{(1)}_{:,1} + \cdots + A^{(i+1)}_{m,:} X^{(T)}_{:,1}, \cdots, A^{(i)}_{m,:} X^{(1)}_{:,F} + \cdots + A^{(i+1)}_{m,:} X^{(T)}_{:,F}]$, collects information from all the other augmented graphs and maintains the original intra-view neighbour aggregating mechanism, which demonstrates TGCN's superiority. Moreover, inter-view information aggregating before activating is able to consider more features of nodes than after activating, because activation function filters some important features. Taking $ReLU(\cdot)$ as an example, the latter formulation can only aggregate the non-negative features from different views, while the former one is able to consider the negative features.

5 EXPERIMENTS

5.1 Experimental Settings

5.1.1 Datasets. Following [13, 45, 46], we validate the proposed model on the largest connected component (LCC) of three benchmark datasets, including three citation graphs, i.e., Cora [25], Cora-ML [1] and Citeseer [9]. The concrete statistics of datasets are shown in Table 2.

5.1.2 Baselines. Following the adversarial attacks repository DeepRobust [19], we compare our RT-GCN model with the state-of-the-art GNN and defense models on three datasets to evaluate the robustness performance:

Table 1: The results (Accuracy±Std) of node classification under non-targeted attack (*metattack*).

Dataset	Pbt Rate(%)	GCN	GAT	GCN-SVD	GCN-Jaccard	RGCN	ProGNN	RT-GCN
Cora	0	83.34±0.33	83.73±0.77	78.02±0.36	82.47±0.43	84.40±0.84	82.98±0.23	**85.34±0.37**
	5	77.31±0.75	79.89±0.90	78.19±0.70	79.33±0.49	79.23±0.45	**82.27±0.45**	82.02±0.40
	10	70.72±1.66	74.13±1.58	72.54±1.04	77.11±0.74	73.35±0.57	79.03±0.59	**80.58±0.28**
	15	64.38±2.07	70.95±1.10	69.23±0.96	75.05±0.70	68.48±0.69	76.40±1.27	**79.83±0.17**
	20	54.84±1.63	58.25±1.22	63.56±0.55	73.60±1.44	58.78±0.16	73.32±1.56	**76.44±0.70**
	25	50.14±1.07	53.22±1.43	62.58±0.81	72.22±1.27	52.61±0.67	69.72±1.69	**72.27±0.69**
Cora-ML	0	85.85±0.30	85.65±0.32	83.62±0.06	84.74±0.22	86.52±0.30	85.38±0.14	**86.83±0.08**
	5	79.96±0.34	81.59±0.32	82.10±0.13	80.12±0.36	81.55±0.27	83.57±0.13	**85.23±0.21**
	10	64.45±0.42	76.27±0.67	81.69±0.39	75.45±0.43	74.65±0.53	81.20±0.51	**84.06±0.64**
	15	54.18±0.82	58.68±1.72	57.99±1.37	57.42±0.46	54.39±0.48	78.29±0.19	**79.04±0.60**
	20	45.04±0.74	42.02±1.53	48.64±0.30	46.80±0.53	46.53±0.53	**74.86±0.55**	73.10±0.77
	25	48.80±0.91	47.06±2.21	48.34±2.10	48.53±1.28	49.95±0.46	**76.15±0.15**	72.59±1.05
Citeseer	0	71.87±0.83	73.17±1.08	69.44±0.38	72.33±0.68	71.69±0.58	73.28±0.69	**75.04±0.21**
	5	70.40±1.03	72.96±0.68	67.72±1.24	69.38±1.40	71.06±1.88	72.93±0.57	**74.87±0.24**
	10	67.03±1.06	70.60±0.64	68.83±0.71	69.10±1.02	67.64±0.43	72.51±0.75	**75.53±0.17**
	15	63.94±1.27	68.45±1.55	64.44±0.92	66.09±0.91	63.66±1.12	72.03±1.11	**74.88±0.20**
	20	62.03±1.79	60.16±1.06	58.73±0.83	59.96±0.58	56.05±0.68	70.02±2.28	**72.61±0.46**
	25	56.04±2.29	61.66±2.23	56.94±1.31	60.05±1.21	57.99±0.73	68.95±2.78	**74.43±0.45**

Table 2: The statistics of the largest connected components (LCC) of the three benchmark datasets respectively.

Name	N_{LCC}	E_{LCC}	Classes	Features
Cora	2485	5069	7	1433
Cora-ML	2810	7981	7	2879
Citeseer	2110	3668	6	3707

- **GCN** [17]: This is one of the most representative GCN models owing to its effectiveness, which leverages the spatial graph convolution operator to generate embeddings.
- **GAT** [33]: Graph Attention Network (GAT) aggregates neighbours according to the coefficients calculated by multi-head self-attention mechanism.
- **GCN-SVD** [8]: This method preprocesses the adjacency matrix by discarding the high-rank component of underlying graph through low-rank approximation and feeds the preprocessed adjacency matrix as input of the GCN model.
- **GCN-Jaccard** [37]: This method preprocesses the adjacency matrix by eliminating the underlying noise, i.e., edges connected with dissimilar nodes, according to the Jaccard similarity of node features. Then, the preprocessed adjacency matrix is fed into the GCN model to generate ultimate embeddings. Thus, GCN-Jaccard is not suitable to datasets without features such as Polblogs.
- **RGCN** [44]: RGCN adopts Gaussian distribution as hidden node representations to absorb the changes of adversarial attacks in the variance of Gaussian distribution and imposes penalty on the nodes with high variance.
- **ProGNN** [13]: ProGNN explores the low-rankness, sparsity and feature smoothness of target graphs. This is an end-to-end method that learns graph structure and GNN parameters simultaneously.

5.1.3 Parameter Settings. For each target graph, we randomly split all nodes into three parts, i.e., 10% nodes for training, 10% nodes for validation and the remaining 80% nodes for testing. The final performance is the average of 10 runs for each experiment. All the parameters of baselines are tuned to get preferable performance in most situations or the same as authors' original implementations. Adam optimizer [16] with learning rate as 0.01 is utilized to train RT-GCN. We augment the target graph to five views in all experiments, i.e., $T = 5$.

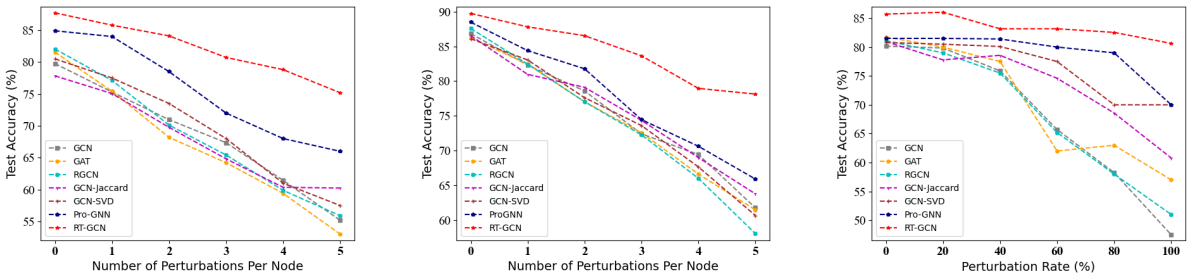
5.2 Defense Performance

To evaluate the robustness of the proposed method, we utilize three types of adversarial attacks to perturb original graph in this paper:

- **Targeted Attack**: Targeted attack focuses on specified targeted nodes and attempts to fool GNNs through these target nodes. In this paper, we use the state-of-the-art targeted attack, *nettack* [45] , to generate targeted attack on three datasets.
- **Non-targeted Attack**: On the contrary to targeted attack, non-targeted attack aims to degenerate the overall performance of model without any specified nodes. Similarly, we use the state-of-the-art non-targeted attack, *metattack* [46].
- **Random Attack**: Random attack is conducted by adding edges between nodes on the graph randomly [13].

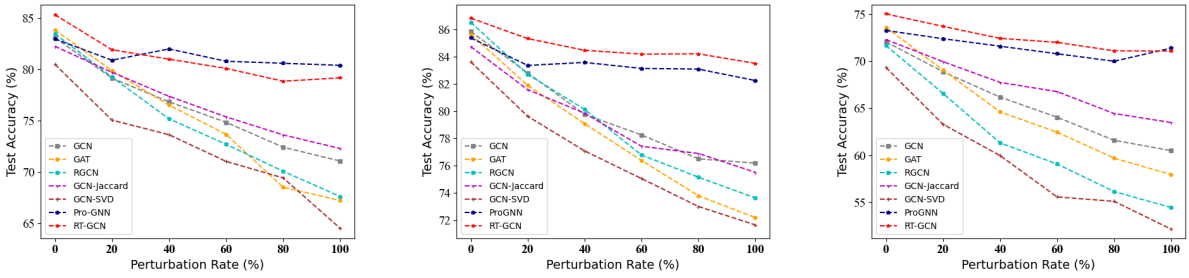
All datasets are firstly attacked by the aforementioned three types of attacks under different perturbation rates. Then, we evaluate the proposed method RT-GCN and other baselines on these poisoned graphs.

5.2.1 Defense Against Non-targeted Adversarial Attack. Metattack is chosen as the representative non-targeted adversarial attack method in our experiments and we use the default parameter settings in authors' original implementation for comparison methods.



(a) Nettack-Cora (b) Nettack-Cora-ML (c) Nettack-Citeseer

Figure 5: The performance of RT-GCN and the other baselines under targeted attacks.



(a) Random-Cora (b) Random-Cora-ML (c) Random-Citeseer

Figure 6: The performance of RT-GCN and the other baselines under random attacks.

To measure the performances of node classification tasks, we utilize classification accuracy as the evaluation criteria. *Metattack* has several variants, while we just consider the most destructive one, Meta-Self. In the experiments, the perturbation rates vary from 0 to 25% with a step of 5%. The results of all the models, i.e., classification accuracy with standard deviation, are shown in Table 1.

Note that RT-GCN outperforms the other models under different perturbation rates from 0 to 25% in almost all situations, which verifies the effectiveness of proposed model on resisting non-targeted adversarial attacks. Especially, the standard deviations of RT-GCN consistently stay at a considerably low level compared with other models. This demonstrates the superior robustness and stability of our model. Concretely, the performance of RT-GCN outperforms the state-of-the-art model, i.e., ProGNN, on different datasets. Although ProGNN explores the poisoned graph from low-rankness, sparsity and feature smoothness, it only focus on single-view augmentation to learn the original structure, which might introduce augmentation variance and contingency for learning node representations. Moreover, compared with preprocessing models GCN-Jaccard and GCN-SVD, T-SVD based graph augmentation has exhibited its superiority on multi-view augmentation. Compared to vanilla GCN, our model improves the performance on three datasets under 25% perturbation rate by 22%, 24% and 18%, respectively.

5.2.2 Defense Against Targeted Adversarial Attack. *Nettack* is adopted as the targeted adversarial attack in this experiment and all the parameter settings are default similarly. The number of perturbations on each targeted node varies from 0 to 5 with a step size of 1. GCN-SVD [8] has demonstrated that SVD based methods are designed for defending carefully-crafted high-rank adversarial attacks, i.e., *nettack*. As an extension, T-SVD based method illustrates that low-rank approximation is able to defend high-rank attacks again. As shown in Fig. 5, it's intuitive that RT-GCN particularly outperforms other methods when the perturbation rate is high.

5.2.3 Defense Against Random Adversarial Attack. In this experiment, we evaluate the robustness of RT-GCN on defending random attacks with perturbation rates varying from 0 to 100% with a step of 20%. On the contrary to *nettack*, it's obvious that random attack is not an unnoticeable attack because it doesn't consider any properties of graph. Thus, SVD based method GCN-SVD achieves the poorest performance under random attacks. However, as shown in Fig. 6, this disadvantage is greatly reduced by our RT-GCN model. Besides, the performance of RT-GCN is considerably competitive to ProGNN.

5.3 Ablation Study

To be aware of the framework of RT-GCN more clearly and figure out which component of RT-GCN playing the key role on maintaining robustness, we conduct the following ablation studies. Here, we only show the results on three datasets under *nettack* because of the page limitation.

Table 3: The results of ablation study of RT-GCN variants under *nettack* with 5 perturbations per node.

	RT_{-AUG}	RT_{-TSVD}	RT_{-TGCN}	RT_{ALL}
Cora	69.27±1.45	61.20±0.72	63.72±1.51	**75.18±0.65**
Cora-ML	75.39±0.84	65.16±0.81	65.18±0.27	**78.14±1.18**
Citeseer	80.79±0.48	71.90±3.95	80.85±0.21	**81.11±0.48**

5.3.1 Remove Random Adding/Dropping. It's important to break the shackle of single-view augmentation and improve the robustness of input data further. To explore the effectiveness of random adding/dropping technique, we duplicate the target graph and organize them into tensor forms. The results of variant RT_{-AUG} model without modifying graph under *nettack* are concluded in Table 3, where the effectiveness of augmentation technique is verified.

5.3.2 Remove T-SVD Based Low Tubal Rank Approximation. Low rank tensor approximation is an effective method to clean corrupted data. To evaluate the contributions of this component, we produce another variant RT_{-TSVD} removing T-SVD module after multi-view augmentation. Table 3 has shown that T-SVD based low tubal rank approximation is the indispensable component of RT-GCN.

5.3.3 Remove Tensor GCN. TGCN is able to convolute multi-view graph from the inter-view and intra-view perspectives simultaneously. To verify its effectiveness, we use five vanilla GCN models to learn node representations separately and the average of five predictions is utilized as the final performance. As shown in Table 3, TGCN is superior to multiple GCN models when there are multi-view graphs, which verifies our proposition about the relationship between vanilla GCN and TGCN.

6 CONCLUSIONS

Improving the robustness of GNNs and protecting them from noise or adversarial attack is a crucial problem. In this paper, we propose a novel RT-GCN model, which consists of two major components, i.e., T-SVD based graph augmentation and tensor GCN, to tackle this problem. In addition, we illustrate that the embedding obtained by TGCN is generically a nonlinear combination of ones learned by an independent vanilla GCN from each view with respect to the Lebesgue measure. Extensive experiments show that RT-GCN is robust enough to resist various attacks and outperforms state-of-the-art baselines. We also hope that this tensor-based GNN study leads to a promising perspective in the field of robust GNN.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[2] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. 2011. Robust principal component analysis? *Journal of the ACM (JACM)* 58, 3 (2011), 1–37.

[3] Chuan Chen, Zhe-Bin Wu, Zi-Tai Chen, Zi-Bin Zheng, and Xiong-Jun Zhang. 2021. Auto-weighted robust low-rank tensor completion via tensor-train. *Information Sciences* 567 (2021), 100–115.

[4] Liang Chen, Jintang Li, Jiaying Peng, Tao Xie, Zengxu Cao, Kun Xu, Xiangnan He, and Zibin Zheng. 2020. A survey of adversarial learning on graphs. *arXiv preprint arXiv:2003.05730* (2020).

[5] Zitai Chen, Chuan Chen, Zibin Zheng, and Yi Zhu. 2019. Tensor decomposition for multilayer networks clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3371–3378.

[6] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[7] Travis Ebesu and Yi Fang. 2017. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 1093–1096.

[8] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. 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*. 169–177.

[9] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*. 89–98.

[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[11] Zhichao Huang, Xutao Li, Yunming Ye, and Michael K Ng. 2020. MR-GCN: Multi-Relational Graph Convolutional Networks based on Generalized Tensor Product.. In *IJCAI*. 1258–1264.

[12] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 665–674.

[13] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 66–74.

[14] Misha E Kilmer, Karen Braman, Ning Hao, and Randy C Hoover. 2013. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM J. Matrix Anal. Appl.* 34, 1 (2013), 148–172.

[15] Misha E Kilmer and Carla D Martin. 2011. Factorization strategies for third-order tensors. *Linear Algebra Appl.* 435, 3 (2011), 641–658.

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[18] Xiangyu Li, Weizheng Chen, Yang Chen, Xuegong Zhang, Jin Gu, and Michael Q Zhang. 2017. Network embedding-based representation learning for single cell RNA-seq data. *Nucleic acids research* (2017).

[19] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. 2020. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149* (2020).

[20] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2257–2270.

[21] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2012. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2012), 208–220.

[22] Ziyang Liu, Zhaomeng Cheng, Yunjiang Jiang, Yue Shang, Wei Xiong, Sulong Xu, Bo Long, and Di Jin. 2021. Heterogeneous Network Embedding for Deep Semantic Relevance Match in E-commerce Search. *arXiv preprint arXiv:2101.04850* (2021).

[23] Canyi Lu, Jiashi Feng, Yudong Chen, Wei Liu, Zhouchen Lin, and Shuicheng Yan. 2016. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5249–5257.

[24] Osman Asif Malik, Shashanka Ubaru, Lior Horesh, Misha E Kilmer, and Haim Avron. 2021. Dynamic Graph Convolutional Networks Using the Tensor M-Product. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 729–737.

[25] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[26] Hogun Park and Jennifer Neville. 2019. Exploiting Interaction Links for Node Classification with Deep Graph Neural Networks.. In *IJCAI*. 3223–3230.

[27] Chanathip Pornprasit, Xin Liu, Natthawut Kertkeidkachorn, Kyoung-Sook Kim, Thanapon Noraset, and Suppawong Tuarob. 2020. Convcn: A cnn-based citation network embedding algorithm towards citation recommendation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*. 433–436.

[28] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903* (2019).

[29] Ryoma Sato. 2020. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078* (2020).

[30] Guangjing Song, Michael K Ng, and Xiongjun Zhang. 2020. Robust tensor completion using transformed tensor singular value decomposition. *Numerical Linear Algebra with Applications* 27, 3 (2020), e2299.

[31] Chang Su, Jie Tong, Yongjun Zhu, Peng Cui, and Fei Wang. 2020. Network embedding in biomedical data science. *Briefings in bioinformatics* 21, 1 (2020), 182–197.

[32] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[34] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 839–848.
[35] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. 2017. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 327–335.
[36] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.
[37] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610* (2019).
[38] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. *arXiv preprint arXiv:2006.08149* (2020).
[39] Zemin Zhang and Shuchin Aeron. 2016. Exact tensor completion using t-SVD. *IEEE Transactions on Signal Processing* 65, 6 (2016), 1511–1526.
[40] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 13001–13008.
[41] Jun Zhou, Chaochao Chen, Longfei Zheng, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. 2020. Privacy-preserving graph neural network for node classification. *arXiv e-prints* (2020), arXiv–2005.
[42] Ke Zhou, Hongyuan Zha, and Le Song. 2013. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*. PMLR, 641–649.
[43] Pan Zhou, Canyi Lu, Zhouchen Lin, and Chao Zhang. 2017. Tensor factorization for low-rank tensor completion. *IEEE Transactions on Image Processing* 27, 3 (2017), 1152–1163.
[44] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1399–1407.
[45] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.
[46] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412* (2019).

A DISCRETE FOURIER TRANSFORM AND T-PRODUCT

Just like the circulant matrix can be diagonalized by normalized Discrete Fourier Transform (DFT) matrix, the t-product of tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ can be further simplified as:

$$\begin{aligned}\text{unfold}(\mathcal{C}) &= \text{bcirc}(\mathcal{A}) \cdot \text{unfold}(\mathcal{B}) \\ &= (F_{n_3}^H \otimes I_{n_1})(F_{n_3} \otimes I_{n_1}) \cdot \text{bcirc}(\mathcal{A}) \cdot (F_{n_3}^H \otimes I_{n_2}) \\ &\quad (F_{n_3} \otimes I_{n_2}) \cdot \text{unfold}(\mathcal{B}) \\ &= (F_{n_3}^H \otimes I_{n_1}) \cdot \text{bdiag}(\bar{\mathcal{A}}) \cdot (F_{n_3} \otimes I_{n_2}) \cdot \text{unfold}(\mathcal{B}) \\ &= (F_{n_3}^H \otimes I_{n_1}) \cdot \text{bdiag}(\bar{\mathcal{A}}) \cdot \text{unfold}(\bar{\mathcal{B}}),\end{aligned} \tag{15}$$

where F_{n_3} is the normalized DFT matrix, which is also an unitary matrix, i.e., $F_{n_3}^{-1} = F_{n_3}^H$. $\otimes$ denotes the Kronecker product and $\bar{\mathcal{A}}$ is the result of tensor $\mathcal{A}$ transformed along the third dimension by DFT, and $bdiag(\cdot)$ is the block diagonal matrix with its i-th block on the diagonal as the i-th frontal slice of $\bar{\mathcal{A}}$.

According to Eq. (15), t-product can be implemented by DFT for saving computational resource. Specifically, we can transform the two multipliers along the third dimension by DFT, and finish the frontal-slice-wise matrix multiplication in the Fourier domain. Then, the final result can be obtained by transforming inverse DFT on the product along third dimension, again.