Multi-view Tensor Graph Neural Networks Through Reinforced Aggregation

Xusheng Zhao, Qiong Dai, Jia Wu, *Senior Member, IEEE* Hao Peng, *Member, IEEE* Mingsheng Liu, Xu Bai, Jianlong Tan, Senzhang Wang, Philip S. Yu, *Fellow, IEEE*

Abstract—Graph Neural Networks (GNNs) have yielded fruitful results in learning multi-view graph data. However, it is challenging for existing GNNs to capture the potential correlation information (PCI) among the graph structure features of multiple views. It is also challenging to adaptively identify valuable neighbors for node feature fusion in different views. To this end, we propose a novel Reinforced Tensor Graph Neural Network (RTGNN) framework to more effectively perform multi-view graph representation learning through reinforcing inter- and intra-graph aggregation. Specifically, RTGNN first uses tensor decomposition to extract the graph structure features (GSFs) of each view in the common feature space. These GSFs contain the PCI of multiple views and alleviate fusion conflicts that may be caused by differences between view feature spaces in cross-view feature fusion. Since fusing the features of all neighbor nodes may harm the features of the center node, we filter the irrelevant neighbors to improve the performance of intra-graph aggregation in each view. Concretely, a reinforcement learning (RL)-guided scheme is developed to automatically calculate the optimal filtering threshold for each view, avoiding tedious manual updates and infeasible back propagation updates. Experimental results and analysis on five datasets show that RTGNN surpasses the best multi-view graph representation baselines and achieves the maximum 14.26% performance improvement in terms of F1. The code link is https://github.com/RingBDStack/RTGNN.

Index Terms—Multi-view, graph neural network, tensor decomposition, reinforcement learning.

1 Introduction

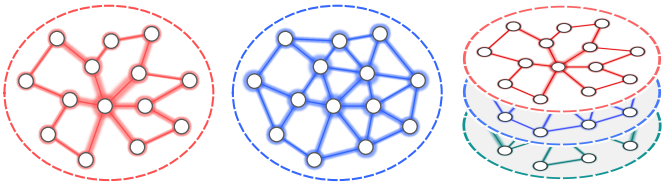
Large amounts of data, in reality, can naturally be stored as graphs, consisting of nodes and edges. If an instance of graph-structured data contains the same nodes but discrepant structures in different views/modalities, it can be defined as a multi-view graph. For example, we can convert a patient's brain into multiple brain graphs that share the same nodes but different edges in different medical imaging views, where nodes represent brain regions and edges represent relations between the pairs of regions [1]. Then, we refer to the set of brain graphs of the patient as a multi-view graph. As a hot topic in the research of multi-view graphs, multi-view graph representation learning/embedding has attracted considerable attention in many applications, such as traffic prediction [2], clinical medicine [3], and so on. Most existing works in multi-view graph representation learning aim to combine information from multiple distinct views and embed each instance into a high-quality representation with low dimensionality for downstream machine learning or data mining tasks, such as classification [3], [4], clustering [5], [6], [7], and so on.

Existing multi-view graph representation learning works can be generally categorized into two types from the perspective of model architecture depth. As stated in work [1], shallow models, such as models based on tensor decomposition [8], [5] and random walk [9], [10], have limited capability in learning feature representations of multi-view graphs with complex structures. In contrast, deep models, such as [1], [3], [11], [4] often utilize Deep Neural Networks (DNNs) to learn highly non-linear and structure-preserving representations for multi-view graphs. Among them, Graph Neural Networks (GNNs)-based models are prevalent and compelling because they take advantage of the representation capability of deep learning and migrate traditional convolution operations from Euclidean space to topological graphs with irregular domains [12], [13], [14].

Since most original GNNs, such as Graph Convolutional Network (GCN) [12], GraphSAGE [15] and Graph Attention Network (GAT) [16] are designed to deal with single-view graphs, it is inconvenient to apply them to multi-view graph scenes. Specifically, given a multi-view graph, the straightforward solution is to consider only a single or combined view and ignore the other views [11], [17], [18], which may lose some potential correlation information (PCI) among the graph structure features of multiple views or the graph structure features (GSFs) of each view. Some works in [19], [6], [4], [3], [20] extended the original GNN to a multi-channel version to process each graph separately and then aggregate the graph features from all the views. Although they can capture the GSFs of each view, they always ignore the PCI of multiple views. Recently, [21] proposed a GNN-based model with tensor networks to capture the spatial and

- *Xusheng Zhao, Qiong Dai, Xu Bai, and Jianlong Tan are with the Institute of Information Engineering, Chinese Academy of Sciences, and the School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China. E-mail: {zhaoxusheng, daiqiong, baixu, tanjianlong}@iie.ac.cn.*
- *Jia Wu is with the Department of Computing, Macquarie University, Sydney, Australia. E-mail: jia.wu@mq.edu.au.*
- *Hao Peng is with the School of Cyber Science and Technology, Beihang University, Beijing, China. E-mail: penghao@buaa.edu.cn.*
- *Mingsheng Liu is with the Shijiazhuang Institute of Railway Technology, Shijiazhuang, China. E-mail: liums601001@sina.com.*
- *Senzhang Wang is with the School of Computer Science and Engineering, Central South University, Changsha, China. E-mail: szwang@csu.edu.cn.*
- *Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, IL, USA. E-mail: psyu@uic.edu.*

Manuscript received July, major revised November, minor revised December 2021, accepted January 2022. (Corresponding author: Hao Peng)



(a) Day view graph (b) Month view graph (c) Tensor graph

Fig. 1: A toy example of multi-view graphs with three views in an urban traffic scene.

temporal information of dynamic spatiotemporal graphs. It is customized for mining the temporal information across multiple graphs in a single view and cannot be directly applied to capture PCI in the multi-view graph. In addition, previous works in [22], [23], [24], [25], [26] have pointed out and verified that usually, not all neighbors are valuable for intra-graph aggregation of GNNs, such as GCN and GAT. Some aggregated neighbor nodes may provide interference information that harms the representation learning of the center node. And some works in [25], [27], [28], [29] have alleviated this problem by filtering these irrelevant neighbors. However, they either ignore the influence of edge weights on the importance of neighbor nodes or manually specify the number of neighbors that need to be retained (a.k.a. the filtering threshold).

To extend GNNs to the multi-view graph, we need to address the following two major challenges. First, it is difficult for original GNNs to effectively capture PCI due to the non-negligible heterogeneity between different views. For example, Fig. 1 shows a multi-view graph whose nodes and edges represent stations and traffic flows among them, respectively. It contains the traffic graph in the current day view (i.e., Fig. 1(a)) and the traffic graph in the past month view (i.e., Fig. 1(b)). The two graphs in different views share the same set of nodes but with discrepant structure features in sparsity and edge weight distribution. It is naturally hard to mine the PCI encoded by the two views with different view feature spaces and may easily cause fusion conflicts in cross-view feature fusion. The second challenge is that it is tricky to adaptively identify valuable neighbors for intra-graph aggregation in each view. Applying the rich edge information in multi-view graph data to assist neighbor importance ranking has rarely been studied. In addition, in a multi-view graph, different views may correspond to different optimal neighbor filtering thresholds. Since treating the thresholds as hyperparameters lacks generalization capability, we usually need to manually update them during the experiment, which is laborious and time-consuming. This motivates us to investigate: *can we design a scheme that utilizes the edge information to great effect to assist in ranking the importance of neighbors and automatically find the best filtering thresholds for different views?*

To tackle the above two challenges, a novel Reinforced Tensor Graph Neural Network framework, i.e., RTGNN, is proposed for multi-view graph representation learning in this work. RTGNN contains four major learning steps: **1) PCI Extraction.** Inspired by the fact that tensor decomposition can acknowledge the underlying correlations among different modes of a tensor and explore the factor features in each mode [30], [8], [5], RTGNN utilizes a bridge module based on tensor decomposition algorithms to deal with the first challenge. Similar to Fig. 1(c), the bridge module first stacks the graphs of different views as a tensor [5], [31], [32] and then introduces the Higher-order Singular Value Decomposition (HOSVD) [33] to extract the GSFs containing PCI in the common feature space. It is worth noting that this module is not limited by the number of views and reduces the risk of fusion conflicts that may be caused by inconsistent feature spaces when performing cross-view graph feature fusion (i.e., inter-graph aggregation). **2) Neighbor Filtration.** We propose an RL-guided module that includes a novel neighbor importance measure and a filtering threshold calculator to solve the second challenge. Concretely, the measure employs the edge weights and edge distances in the graph data to rank the importance of neighbors. Since the filtering thresholds are not directly involved in the model training process, using back propagation to update them is infeasible. Moreover, it is also infeasible to evaluate the thresholds within an epoch. Therefore, reinforcement learning (RL) [34] is applied in the threshold calculator to automatically search for the best filtering threshold for each view. **3) Intra-graph Aggregation.** After obtaining the GSFs of each view and the most valuable neighbors that have undergone the above reinforcement operations, we employ a multi-view GNN to aggregate the features of neighbors in a view-parallel manner to further improve GSFs. Compared with the tensor-based methods, the multi-view GNN better explores the hierarchical patterns of the graphs of different views. **4) Inter-graph Aggregation.** Following the principle of node feature aggregation within a graph, we employ a node-aware manner to fully fuse graph features from different views. At the last layer of our RTGNN, we vectorize the output in order to get the low-dimensional multi-view graph representations.

Our three key contributions are generalized as below:

- We propose a multi-view graph representation learning framework RTGNN with two novel modules. On the one hand, the bridge module takes advantage of tensor decomposition in capturing the PCI encoded by multiple views. On the other hand, the RL-guided module guides the intra-graph aggregation of the multi-view GNN, thereby further improving the GSF mining of each view.
- To the best of our knowledge, this is the first work to introduce HOSVD into the GNN for multi-view graph representation learning. We are also the first to introduce edge information for neighbor filtering in the intra-graph aggregation of the multi-view GNN.
- We conduct plentiful experiments on five real multi-view graph datasets. The corresponding results and analysis indicate that the proposed RTGNN is superior to the best baselines. Besides, RTGNN can not only process natural multi-view graphs (e.g., multi-view brain graphs), but also assist in internal multi-view analysis of heterogeneous graph network datasets (e.g., DBLP).

We format the paper as follows: Sec. 2 describes the preliminary knowledge. Sec. 3 presents the specific technical details of our RTGNN. Sec. 4 gives the relevant deployments and results of the experiments. Sec. 5 introduces related work. Sec. 6 concludes the work of this paper.

TABLE 1: Forms and interpretations of necessary notations.

Forms	Interpretations/Definitions
$\mathcal{G}; \mathcal{G}_{train}$	The multi-view graphs; The training set in $\mathcal{G}$
$\mathcal{A}; \mathcal{F}$	The weighted adjacency tensor; The feature tensor
$\mathcal{A}^+; \mathcal{Z}$	The trainable adjacency tensor; The transformation tensor
$G; N; E$	The graph; The node set; The edge set
$\mathbf{X}; \mathbf{Y}$	The node label matrix; The multi-view graph label matrix
$\mathbf{A}; \mathbf{F}$	The weighted adjacency matrix; The feature matrix
$\widehat{\mathbf{A}}; \widehat{\mathbf{F}}$	The normalized version of $\mathbf{A}$; The enhanced version of $\mathbf{F}$
$\mathbf{E}$	The final feature matrix of $\mathcal{G}$
$\mathbf{U}; \mathbf{V}$	These notations indicate tensor factor matrices
$\mathbf{I}; \mathbf{Z}$	The square matrix; The feature transformation matrix
M	The total number of multi-view graphs
$V; \lvert N\rvert$	The total number of views; The nodes' number in G
D	The feature dimension of $\mathbf{F}/\mathbf{E}$
L	The total number of layers of the multi-view GNN
P	The total number of epochs
$T; S$	The filtering threshold; The step size of RL action
n	The node element belonging to N
$i; j; k; k'; l; l^*; p$	These notations represent index variables
$AGG_{intra}(\cdot)$	The intra-graph aggregation function
$AGG_{inter}(\cdot)$	The inter-graph aggregation function
$\lVert\cdot\rVert_F; \lVert\cdot\rVert_2$	The F-norm of the matrix; The 2-norm of the matrix
$TRAN(\cdot)$	The transpose operation of the matrix/tensor
$NORM(\cdot)$	The normalization operation of the matrix
$FNN(\cdot)$	The fully connected neural network
$IMP(k, k')$	The importance of node $n_{k'}$ to node n_k
$DIST(k, k')$	The distance between n_k and its neighbor $n_{k'}$
$RANK(k, k')$	The ranking of the importance of node $n_{k'}$ to node n_k
AVG	The average neighbor importance
$REW; TER$	The RL reward; The terminal condition
$\sigma(\cdot); \min(\cdot)$	The activation function; The mathematical min function
$\otimes; \times$	The combination operation; The mode product operation
$\mathcal{L}_{edge}; \mathcal{L}_{GNN}$	The measure loss; The GNN loss
$\mathcal{L}_{all}$	The overall loss of RTGNN
$\Theta; \mu$	The model parameter set; The regularization coefficient

2 PRELIMINARIES

First, we define the multi-view graphs and the multi-view graph representation learning. Next, we give the intra- and inter-graph feature aggregation process of the multi-view GNN. The forms and interpretations of all necessary notations throughout our work are stated in Table 1.

Definition 1. Multi-view Graphs. We denote a graph as $G = \{N, E, \mathbf{X}, \mathbf{A}, \mathbf{F}\}$, where $N = \{n_k\}|_{k=1}^{|N|}$ is the node set, $E \subseteq N \times N$ is the edge set, $\mathbf{X}(k)$ is the label of the k-th node n_k. Let $\mathbf{A} \in \mathbb{R}^{|N|\times|N|}$ represents the weighted adjacency matrix of G, so that $\mathbf{A}(k, k')$ encodes the relation between n_k and $n_{k'}$ if there is an edge, and 0 otherwise. Let $\mathbf{F} \in \mathbb{R}^{|N|\times D}$ be the feature matrix of G, where the k-th row of $\mathbf{F}$, i.e., $\mathbf{F}(k)$ is the feature vector of n_k with dimension D. Hence, a V-view graph is denoted as a set $\{G_j\}|_{j=1}^{V}$, where each graph is extracted from a specific view, and all graphs share the same set of nodes N but different sets of edges, i.e., inconsistent E and $\mathbf{A}$. The dataset contains M multi-view graphs is $\mathcal{G} = \{\mathcal{G}_i\}|_{i=1}^{M}$, where $\mathcal{G}_i = \{G_{i,j}\}|_{j=1}^{V}$ is the i-th multi-view graph associated with a class label $\mathbf{Y}(i)$, and $G_{i,j}$ denotes the graph in the j-th view of $\mathcal{G}_i$.

Definition 2. Multi-view Graph Representation Learning. We study multi-view graph representation learning in this work, which can be defined as follows: given the multi-view graphs $\mathcal{G} = \{\mathcal{G}_i\}|_{i=1}^{M}$, we aim to embed them into a low-dimensional and high-quality feature matrix $\mathbf{E} \in \mathbb{R}^{M\times D^{(L)}}$, which allows multi-view graphs with different labels to be easily separated. $\mathbf{E}(i)$ indicates the feature representation/vector of $\mathcal{G}_i$.

Definition 3. Multi-view GNN. This is a GNN framework designed for learning multi-view graph representations. Given $\mathcal{G}$, we can denote its accompanying weighted adjacency tensor and feature tensor as $\mathcal{A} \in \mathbb{R}^{M\times V\times|N|\times|N|}$ and $\mathcal{F} \in \mathbb{R}^{M\times V\times|N|\times D}$, respectively. Therefore, $\mathcal{A}(i)$ and $\mathcal{F}(i)$ are the weighted adjacency tensor and feature tensor of the i-th multi-view graph $\mathcal{G}_i$. In addition, we stipulate that the adjacency matrix $\mathcal{A}(i, j)$ and feature matrix $\mathcal{F}(i, j)$ of the graph $G_{i,j}$ are equivalent to $\mathbf{A}_{i,j}$ and $\mathbf{B}_{i,j}$, respectively. For instance, the feature aggregation process of the multi-view GNN at the l-th layer for the i-th multi-view graph $\mathcal{G}_i$ can generally be expressed as follows:

$$\begin{aligned}\mathbf{F}_{i,1}^{(l)}(k) &= \sigma\left(AGG_{inter}^{(l)}\left(\left\{\mathbf{F}_{i,j}^{(l^*)}(k)\right\}\Big|_{j=1}^{V}\right)\right),\\ \mathbf{F}_{i,j}^{(l^*)}(k) &= \sigma\left(AGG_{intra,j}^{(l)}\left(\left\{\mathbf{F}_{i,j}^{(l-1)}(k') : \mathbf{A}_{i,j}(k, k') > 0\right\}\right)\right),\end{aligned} \tag{1}$$

where $\mathbf{F}_{i,j}^{(l-1)}(k')$ is the representation of the k'-th node $n_{k'}$ at the $(l-1)$-th layer, $\mathbf{A}_{i,j}$ is the weighted adjacency matrix in the j-th view and remains unchanged at all layers, and $\mathbf{F}^{(l^*)}$ comprises the transformed feature representations between the l- and $(l-1)$-th layer. $AGG_{intra,j}^{(l)}$ is the intra-graph aggregation function (e.g., convolution and attention) at the l-th layer, whose subscript j means using models with the same architecture but different parameters to handle each divergent view. $AGG_{inter}^{(l)}$ is the inter-graph function (e.g., concatenation and mean) used to fuse graph features from different views and one for each layer.

3 METHODOLOGY

Fig. 2 illustrates the proposed learning framework RTGNN, which contains the bridge module, RL-guided module, and feature aggregation operations. Next, we will introduce each component of RTGNN in detail.

3.1 Bridge Module for PCI Extraction

Although tensor analysis has shown excellent performance in multi-view graph mining [32], [31], few efforts combine tensor decomposition algorithms with GNNs to facilitate multi-view graph representation learning. Inspired by tensor decomposition's success in projecting the initial tensor into a common space that encodes the potential correlations among different modes [5], [8], we propose a bridge module based on tensor decomposition, which aims to enhance the initial GSFs and alleviate the fusion conflicts during the inter-graph aggregation of the multi-view GNN. Different from the existing feature space alignment techniques of different views applied to GNNs [35], the bridge module does not roughly use linear transformation to align multiple feature spaces but finds a new common space that retains the initial features of each view. Moreover, the GSFs generated by this module also contain the PCI of multiple views, which is tricky to achieve with other GNN-based methods. Considering that the core tensor produced by Tucker decomposition (i.e., HOSVD [33]) can capture further correlation information while reflecting most of the properties of the original tensor (which is lacking in other algorithms such as CANDECOMP/PARAFAC (CP) decomposition [36]), we apply Tucker decomposition to complete the proposed bridge module.

First, we define each multi-view graph $\mathcal{G}_i$ in the dataset $\mathcal{G} = \{\mathcal{G}_i\}|_{i=1}^{M}$ as an instance. Then we convert all the accompanying feature matrices of $\mathcal{G}$ into a feature tensor comprising four modes: instances, views, nodes and features.

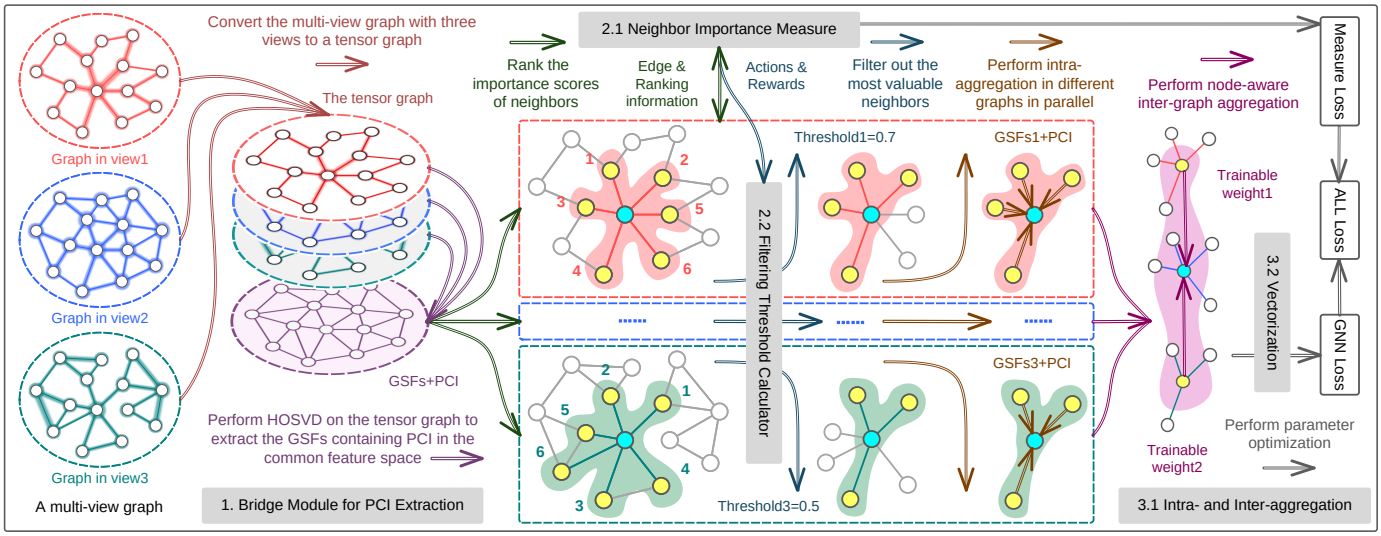


Fig. 2: Overview framework of our RTGNN. Given a multi-view graph with three views, the bridge module (Sec. 3.1) first use HOSVD to capture the PCI of multiple views in the common feature space. Next, the RL-guided module (Sec. 3.2) combines the neighbor importance measure and RL to automatically select the most valuable neighbors. At the aggregation stage, we perform parallel intra-graph aggregation (Sec. 3.3.1) on the three graphs and then utilize a node-aware manner to complete the inter-graph aggregation (Sec. 3.3.2). The optimization steps are presented in Sec. 3.4.

The feature tensor is denoted as $\mathcal{F} \in \mathbb{R}^{M \times V \times |N| \times D}$, where $\mathcal{F}(:, j, :, :) \in \mathbb{R}^{M \times |N| \times D}$ is its j-th slice on the second mode. The HOSVD process of $\mathcal{F}$ is defined as follows:

$$\begin{aligned} &\min_{\mathcal{C}, \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \mathbf{U}_4} \left\| \mathcal{F} - \mathcal{C} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \times_4 \mathbf{U}_4 \right\|_F^2, \\ &s.t. \quad TRAN(\mathbf{U}_i)\mathbf{U}_i = \mathbf{I}, i \in \{1, 2, 3, 4\}, \end{aligned} \tag{2}$$

where $\times_1$, $\times_2$, $\times_3$ and $\times_4$ indicate mode product operations. $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$ and $\mathbf{U}_4$ are the factor matrices (i.e., the principal components) in each mode of $\mathcal{F}$. $\mathcal{C}$ is the core tensor whose entries reveal the level of interaction between the diverse modes. $TRAN = TRAN_{1,2}$ represents the transpose operation of the matrix, and $\mathbf{I}$ represents a square matrix. We further derive the minimization formula for building the view-dependent graph features as follows:

$$\min_{\widehat{\mathbf{F}}} \sum_{i=1}^{M} \sum_{j=1}^{V} \left\| \mathbf{F}_{i,j} \times_1 TRAN(\mathbf{U}) \times_2 TRAN(\mathbf{V}) - \widehat{\mathbf{F}}_{i,j} \right\|_F^2, \tag{3}$$

where $\mathbf{U} = \mathbf{U}_3$ and $\mathbf{V} = \mathbf{U}_4$ are the last two orthogonal factor matrices of HOSVD on $\mathcal{F}$. More precisely, $\mathbf{U}$ and $\mathbf{V}$ are used as functional matrices to project graph representations to a common feature space, reducing the heterogeneity gap between different views. $\widehat{\mathbf{F}}_{i,j}$ is the enhanced reliable feature matrix corresponding to $\mathbf{F}_{i,j}$, which involves potential correlation information of multiple views. To avoid notation confusion, we use $\mathcal{F}^{(0)} \in \mathbb{R}^{M \times V \times |N| \times D^{(0)}}$ instead of $\widehat{\mathcal{F}}^{(0)}$ in the following content to represent the input feature tensor of the multi-view GNN.

3.2 RL-guided Module for Neighbor Filtration

In this part, we propose a novel measure that considers edge information to calculate the neighbor node importance and an RL-guided scheme to automatically obtain the optimal filtering threshold for each view. After eliminating the interference of irrelevant neighbors on each center node in each view, RTGNN improves the intra-graph aggregation performance of the multi-view GNN.

3.2.1 Neighbor importance Measure

Previous works in [25], [27] have verified that same-label neighbors are valuable while different-label ones are generally of little value in the feature aggregation of the center node of the GNN. For example, in the graph-based fraud detection task [27], since fraudster nodes usually interact with normal nodes to achieve behavioral camouflage, aggregating all neighbor features may lead to indistinguishable fraudster representations. Although some works in [22], [37] have proposed unsupervised measures (e.g., cosine similarity) to filter neighbors, it is still tricky to distinguish between fraudsters with camouflaged features and normal users. Hence, we need a neighbor importance measure using supervised signals from labels. Considering that the multi-view graphs contain rich edge information, a new measure, which utilizes edge weights and node labels is developed to distinguish each node's valuable neighbors.

Inspired by [25], in which a perceptron is set to predict annotations, we utilize a Fully Connected Neural Network (FNN) with a single layer as the node label predictor and the 2-norm $\|\cdot\|_2$ to calculate the node similarity (i.e., Euclidean distance/edge distance) between the center node and each of its neighbors. Then we combine edge weights and distance information to infer the importance of neighbors. Given the graph $G_{i,j}$ in the j-th view of the i-th instance, the importance of the neighbor node $n_{k'}$ to the center node n_k at the l-th layer is calculated as follows:

$$\begin{aligned} IMP_j^{(l)}(k, k') &= \left| \mathbf{A}_{i,j}(k, k') \right| \otimes \left(1 - DIST_j^{(l)}(k, k') \right), \\ DIST_j^{(l)}(k, k') &= NORM \left(\left\| \begin{matrix} \sigma\left(FNN_j^{(l)}\left(\mathbf{F}_{i,j}^{(l)}(k) \right) \right) \\ \sigma\left(FNN_j^{(l)}\left(\mathbf{F}_{i,j}^{(l)}(k') \right) \right) \end{matrix} \right\|_2 \right), \end{aligned} \tag{4}$$

where $\left| \mathbf{A}_{i,j}(k, k') \right|$ is the edge weight of n_k and $n_{k'}$, and the outputs of IMP and $DIST$ are both non-negative values. $\otimes$ is a combination operation, and we use multiplication in this work. To optimize the above label predictors (i.e., FNN

functions) and the multi-view GNN together while avoiding interference to the training of GNN, we add an independent calculation path to update the FNN parameters. We denote the training set in the multi-view graphs $\mathcal{G}$ by $\mathcal{G}_{train}$, then the cross entropy loss of the l-th layer's FNN functions can be written as follows:

$$\mathcal{L}_{edge}^{(l)} = -\sum_{\mathcal{G}_{train}} \sum_{j=1}^{V} \sum_{k=1}^{|N|} \mathbf{X}(k) \log\left(\sigma\left(FNN_j^{(l)}\left(\mathbf{F}_{i,j}^{(l)}(k)\right)\right)\right). \quad (5)$$

During training, the measure is optimized by supervised signals from node labels. It guarantees fast convergence within the first few training epochs and helps improve the accuracy of importance calculations.

3.2.2 Filtering Threshold Calculator

After getting the importance scores of all neighbor nodes in each view, we rank them first and then find the most valuable neighbors according to the filtering thresholds. With the following motivations, we introduce a filtering threshold calculator via reinforcement learning to assist the neighbor filtering process. First, when processing a single-view graph, we can define a hyperparameter about the filtering threshold and keep tuning to locate the best value. However, in a multi-view graph, different views often correspond to different optimal neighbor filtering thresholds. It is laborious and inefficient to find the optimal filtering thresholds for different views simultaneously through manual adjustment, especially when the number of views is enormous. Second, since the filtering thresholds do not directly participate in the model training process, it is infeasible to use back propagation from the training loss of the multi-view GNN to update them. Finally, the filtering threshold of each view should be dynamically adjusted within the interval of two adjacent epochs to find the optimal value.

Taking the k-th node n_k as an example, we first collect the set containing the importance scores of its neighbor nodes $\left\{IMP_j^{(l)}(k,k') : \mathbf{A}_{i,j}(k,k') > 0\right\}$ and the l-th layer's filtering threshold $T_j^{(l)} \in [0,1]$ in the j-th view, where 0 and 1 indicate that the neighbors are all dropped or kept, respectively. Then we rank the neighbors of node n_k in order of importance from large to small and treat the top $T_j^{(l)}$ percent of neighbors as the optimal aggregation objects. Concretely, we define the problem of finding optimal thresholds as a Two-Armed Bandit $\left\{\left\{act_1, act_2\right\}, REW, TER\right\}$ [34], where the calculator and the measure can be viewed as the "player" and "slot machine", respectively. $\left\{act_1, act_2\right\}$ is the action space containing two actions, i.e., increase or decrease the filtering threshold. REW and TER are functions of reward and termination, respectively.

• **Action:** The action represents how this calculator updates the threshold based on the reward. Here we use the ϵ-greedy strategy to plus or minus one step size S to the filtering threshold in each action.

• **Reward:** The disparity between the mean importances of two adjacent epochs determines the reward. The average neighbor importance in the j-th view at the l-th layer for the p-th epoch is calculated as follows:

$$AVG_j^{(l)[p]} = \sum_{\mathcal{G}_{train}} \frac{\sum_{\mathbf{A}_{i,j}^{(l^*)[p]}(k,k')>0} IMP_j^{(l)}(k,k')}{\sum_{n_k \in N} T_j^{(l)[p]} |N(k)|}, \quad (6)$$

where $T_j^{(l)[p]}$ is the current filtering threshold, $N(k) = \left\{n_{k'} : \mathbf{A}_{i,j}(k,k') > 0\right\}$ represents the initial neighbor set of n_k, $T_j^{(l)[p]}|N(k)|$ is the number of neighbors retained after filtering, and $\mathbf{A}_{i,j}^{(l^*)[p]}$ is the filtered adjacency matrix of $G_{i,j}$, whose calculation is defined as follows:

$$\mathbf{A}_{i,j}^{(l^*)[p]}(k,k') = \begin{cases} 1, & RANK_j^{(l)}(k,k') \le T_j^{(l)[p]}|N(k)| \\ 0, & \text{otherwise} \end{cases}, \quad (7)$$

where $RANK_j^{(l)}(k,k') \in \left[1, |N(k)|\right]$ indicates the ranking of importance score $IMP_j^{(l)}(k,k')$. Hence, the immediate reward is determined as follows:

$$REW_j^{(l)[p]} = \begin{cases} -1, & AVG_j^{(l)[p]} \le AVG_j^{(l)[p-1]} \\ +1, & AVG_j^{(l)[p]} > AVG_j^{(l)[p-1]} \end{cases}. \quad (8)$$

If the average importance of the newly selected neighbors is larger than the previous one, we set the reward to positive and increase the threshold by one step size, and vice versa.

• **Termination:** The termination condition TER is:

$$\left|\sum_{p-10}^{p} REW_j^{(l)[p]}\right| \le 1. \quad (9)$$

It means that the module has converged in the last ten epochs, and the following training process will keep the current threshold. It is worth noting that our RTGNN calculates the neighbor importance scores and the filtering threshold of each view in a view-parallel manner.

3.3 Feature Aggregation

Aggregation operations of RTGNN include node feature fusion in each graph and a novel cross-view graph feature fusion. For example, at the l-th layer, the feature tensor of the multi-view graphs $\mathcal{G}$ is transferred as follows:

$$\mathcal{F}^{(l-1)} \xrightarrow{AGG_{intra}} \mathcal{F}^{(l^*)} \xrightarrow{AGG_{inter}} \mathcal{F}^{(l)}, \quad (10)$$

where $\mathcal{F}^{(l)} \in \mathbb{R}^{M \times V \times |N| \times D^{(l)}}$ means the output and input of the l- and $(l+1)$-th layer of the multi-view GNN model, respectively, and $\mathcal{F}^{(l^*)} \in \mathbb{R}^{M \times V \times |N| \times D^{(l^*)}}$ represents the l-th layer's transformed feature tensor, $D^{(l)}$ and $D^{(l^*)}$ are the node feature dimensions of $\mathcal{F}^{(l)}$ and $\mathcal{F}^{(l^*)}$, respectively. As the number of layers starts from 1, the initial input of this part is $\mathcal{F}^{(0)}$ (from Sec. 3.1).

3.3.1 Intra-graph Aggregation

We have obtained the enhanced adjacency tensor through the neighbor ranking and filtering operations (i.e., Eq. (7)), which is denoted as $\mathcal{A}^{(l^*)} \in \mathbb{R}^{M \times V \times |N| \times |N|}$ for simplicity. Here we take GCN as an example to implement the intra-graph aggregation. With the adjacency tensor $\mathcal{A}^{(l^*)}$ and feature tensor $\mathcal{F}^{(l-1)}$, the l-th layer's detailed node feature propagation of AGG_{intra} is as follows:

$$\mathcal{F}^{(l^*)} = \sigma\left(\widehat{\mathcal{A}}^{(l^*)} \mathcal{F}^{(l-1)} \mathcal{Z}_{intra}^{(l)}\right). \quad (11)$$

If we focus on the single-view graph in the j-th view of the i-th instance $\mathcal{G}_i$, Eq. (11) is refined into the following form:

$$\mathbf{F}_{i,j}^{(l^*)} = \sigma\left(\widehat{\mathbf{A}}_{i,j}^{(l^*)} \mathbf{F}_{i,j}^{(l-1)} \mathbf{Z}_{intra,i,j}^{(l)}\right), \quad (12)$$

where $\mathcal{Z}_{intra}^{(l)} \in \mathbb{R}^{M \times V \times D^{(l-1)} \times D^{(l^*)}}$ is the layer-view-specific (each view has an independent matrix at each layer) transformation tensor that changes the node feature vectors' dimension from $D^{(l-1)}$ to $D^{(l^*)}$, $\mathcal{Z}_{intra}(i,j) = \mathbf{Z}_{intra,i,j}$, and $\widehat{\mathbf{A}}_{i,j}^{(l^*)} \in \mathbb{R}^{|N| \times |N|}$ is the normalized version of $\mathbf{A}_{i,j}^{(l^*)}$.

Algorithm 1 RTGNN: Reinforcement Learning Guided Tensor Graph Neural Network

Input: The multi-view graphs $\mathcal{G}$ including the node labels $\mathbf{X}_{i,j}$ of each graph $G_{i,j}$ and the label matrix $\mathbf{Y}$
Output: The high-quality feature matrix $\mathbf{E}$ of $\mathcal{G}$

$\mathcal{F}^{(0)} \leftarrow$ Eq. (2) and Eq. (3) ▷ Bridge Module
for $p = 1, \cdots, P$ **do**
  for $l = 1, \cdots, L$ **do**
    for $j = 1, \cdots, V$ **do** ▷ RL-guided Module
      $IMP_j^{(l)[p]} \leftarrow$ Eq. (4) ▷ Measure
      $\mathcal{L}_{edge}^{(l)} \leftarrow$ Eq. (5) ▷ Measure Loss
      if Eq. (9) is False **then** ▷ Calculator
        $T_j^{(l)[p]} \leftarrow$ Eq. (6) and Eq. (8)
      end if
      $\mathcal{A}^{(l^*)[p]}(:, j, :, :) \leftarrow$ Eq. (7)
      $\mathcal{F}^{(l^*)[p]}(:, j, :, :) \leftarrow$ Eq. (11) ▷ Intra-graph Agg
    end for
    $\mathcal{F}^{(l)[p]} \leftarrow$ Eq. (13) ▷ Inter-graph Agg
  end for
  $\mathbf{E}^{[p]} \leftarrow$ Eq. (14) ▷ Feature Vectorization
  $\mathcal{L}_{GNN} \leftarrow$ Eq. (15) ▷ GNN Loss
  $\mathcal{L}_{all} \leftarrow$ Eq. (16) ▷ Overall Loss
end for

3.3.2 Inter-graph Aggregation

After converting the feature tensor $\mathcal{F}^{(l-1)}$ to the feature tensor $\mathcal{F}^{(l^*)} \in \mathbb{R}^{M \times V \times |N| \times D^{(l^*)}}$, we complete the inter-graph aggregation of the multi-view GNN in a novel node-aware manner. Concretely, we construct a trainable adjacency tensor $\mathcal{A}^+ \in \mathbb{R}^{M \times |N| \times V \times V}$ to enable RTGNN to better fuse and adapt multiple features encoded by the multi-view graphs $\mathcal{G}$. With the feature tensor $\mathcal{F}^{(l^*)}$, we define the inter-graph aggregation function AGG_{inter} at the l-th layer as follows:

$$TRAN_{2,3}(\mathcal{F}^{(l)}) = \sigma\left(\mathcal{A}^{+(l)} TRAN_{2,3}(\mathcal{F}^{(l^*)}) \mathcal{Z}_{inter}^{(l)}\right), \quad (13)$$

where $TRAN_{2,3}$ exchanges of the second and third mode of a tensor, so that $TRAN_{2,3}(\mathcal{F}^{(l)}) \in \mathbb{R}^{M \times |N| \times V \times D^{(l)}}$. Similar to $\mathcal{Z}_{intra}^{(l)}$, $\mathcal{Z}_{inter}^{(l)} \in \mathbb{R}^{M \times |N| \times D^{(l^*)} \times D^{(l)}}$ is a layer-specific feature transformation tensor. Compared with the general view-level weighted summation, the adjacency tensor $\mathcal{A}^+$ captures the combined weights of diverse views at a higher granularity (node-level), resulting in the high learning flexibility of RTGNN. If prior knowledge about the dependencies among different views is available, it can be applied to initialize $\mathcal{A}^+$ through fine-tuning or freezing. RTGNN is not limited to the above convolution aggregations, and some of its variants are tested in Sec. 4. At the last layer of the model, we vectorize its output $\mathcal{F}^{(L)} \in \mathbb{R}^{M \times V \times |N| \times D^{(L)}}$ to obtain the final feature matrix of $\mathcal{G}$. The calculation process implemented by mean pooling is defined as follows:

$$\mathbf{E}(i) = \frac{1}{V|N|} \sum_{j=1}^{V} \sum_{k=1}^{|N|} \mathcal{F}^{(L)}(i, j, k, :), \quad (14)$$

where $\mathbf{E}(i) \in \mathbb{R}^{1 \times D^{(L)}}$ is the i-th row of $\mathbf{E}$ and the $D^{(L)}$-dimensional feature representation of the i-th multi-view graph instance $\mathcal{G}_i$, and the vectorization process described above can also be achieved by other methods, such as max pooling and concatenation.

TABLE 2: Statistics for all datasets in the experiments.

Dataset	Instances	Classes	Features	Views
HIV	70	2	90	fMRI&DTI
BP	97	2	82	fMRI&DTI
BikeDC	72	4	267	weekday&weekend&month
PROTEINS	1000	2	80	sequence&molecule interaction
DBLP	4067	4	334	APA&APTPA&APVPA

3.4 Optimization

We learn the parameters of the multi-view GNN by minimizing the discrepancy between the estimated and the given matrix $\mathbf{Y}$ during the training phase, where the i-th row of $\mathbf{Y}$ (i.e., $\mathbf{Y}(i)$) is the vector label in the one-hot encoding form of the i-th multi-view graph $\mathcal{G}(i) \in \mathcal{G}_{train}$. The cross entropy loss of the multi-view GNN is as follows:

$$\mathcal{L}_{GNN} = -\sum_{\mathcal{G}_{train}} \mathbf{Y}(i) \log\left(FNN^{(L+1)}\left(\mathbf{E}(i)\right)\right). \quad (15)$$

Together with the measure loss defined in Eq. (5), we denote the overall loss of our RTGNN as follows:

$$\mathcal{L}_{all} = \mathcal{L}_{GNN} + \sum_{l=1}^{L} \mathcal{L}_{edge}^{(l)} + \mu \sum_{l=1}^{L+1} \left\|\Theta^{(l)}\right\|_2, \quad (16)$$

where the third term is used to avoid overfitting, $\Theta^{(l)}$ is the parameter set of the model at the l-th layer, and μ is the regularization coefficient. The overall objective function can be efficiently optimized with gradient descent and back propagation algorithms. The detailed algorithm of our RTGNN is presented in Alg. 1.

4 Experiments

First of all, we describe the experimental setups, including datasets (Sec. 4.1), baselines and evaluation metrics (Sec. 4.2), and implementation details (Sec. 4.3). Then, we conduct a series of classification tasks, clustering tasks, ablation studies, hyperparameter analysis and discussion to solve five key research questions (RQ):

- RQ1. How does RTGNN compare with the state-of-the-art (SOTA) baselines on classification tasks? (Sec. 4.4)
- RQ2. How does our RTGNN compare with the SOTA baselines on clustering tasks? (Sec. 4.5)
- RQ3. How much do the bridge and RL-guided module included in RTGNN improve the representation learning capabilities of GNNs for multi-view graphs? (Sec. 4.6)
- RQ4. How do important hyperparameters in RTGNN affect the performance of representation learning? (Sec. 4.7)
- RQ5. What are the advantages and potential limitations of the proposed RTGNN? (Sec. 4.8)

4.1 Datasets and Processing

Five multi-view graph datasets covering four domains (i.e., brain science, urban traffic, bioinformatics, and academics) are used for evaluation. Table 2 gives the statistics of them, whose more details are described as follows:

- **Human Immunodeficiency Virus (HIV):** The original brain dataset is collected by Chicago Early HIV Infection Study at Northwestern University [38] from two views, i.e., functional magnetic resonance imaging (fMRI) and diffusion tensor imaging (DTI). There are idiosyncratic differences between these two views. Concretely, the structures

of fMRI-derived graphs encode functional activities among brain regions. In contrast, in DTI-derived graphs, the structures capture the white matter fiber pathways that connect different brain regions. We randomly select 35 HIV patients and 35 healthy controls, where these two groups have no difference in the character portraits, such as gender, age and educational base. Similar to the processing process proposed in [8], we process HIV data according to the following steps. First, the fMRI data is processed by the DPARSF [39]. All brain images are realigned to the first volume, and we perform time rectification and normalization to standard MNI templates. Then, a Gaussian kernel is used to smooth the normalized brain images spatially. The band-pass filtering and linear trending analysis are also performed to eliminate the interference from the low-frequency drift or high-frequency noise. After that, for each brain, the gray matter is divided into 116 anatomical volumes of interest through the automatic anatomical labeling atlas, where each atlas represents a specific brain region. Finally, by eliminating 26 cerebellar regions and calculating the correlations (as edges) among the remaining 90 labeled brain regions (as nodes), we can obtain the brain graphs of all subjects. Similarly, we use the FSL toolbox [40], including distortion correction, noise removal, and over-sampling from each voxel's main diffusion direction distribution to deal with DTI. Then we construct the DTI-derived brain graphs that share the same nodes as the fMRI-derived brain graphs. The two-view brain graph of a subject is regarded as an instance.

• **Bipolar Disorder (BP):** This brain dataset [41] also contains fMRI and DTI views, including 52 patients with bipolar disorder and 45 age and gender matched healthy controls. Following the work [8], we use the CONN toolbox [42] to build brain graphs with 82 labeled nodes. Considering that the connections of the initial brain graphs obtained through the above processing may not all be effective, we follow the work in [3] to generate more reliable weighted adjacency tensors for the HIV and BP datasets.

• **Capital Bikeshare Data (BikeDC):** This urban traffic dataset [1] comes from the Washington D.C. Bicycle System, from which we collect station traffic records over six years (i.e., from 2015 to 2020). Inspired by the work in [20], we delete some bicycle stations with no records or few records and then cluster them into four categories with a total of 267 coarse stations based on their location attributes (latitude, longitude, etc.). For any pair of coarse stations (as nodes), we can obtain the different traffic flows (as edges) between them in three temporal views (i.e., weekday, weekend, and month), where the inflow and outflow are the numbers of bicycles checked in and out of the station, respectively. Therefore, we treat the three-view traffic graph extracted within a month as an instance whose label is the corresponding season (i.e., spring, summer, autumn or winter).

• **PROTEINS:** This is a bioinformatics dataset [43] that requires accurate distinction between enzymatic and non-enzymatic protein molecules. Inspired by the work [44], we collect 1000 molecules from the original dataset as instances, where each molecule instance is processed as a graph with sequence view and molecule interaction view, and the nodes are four types of amino acids.

1. https://www.capitalbikeshare.com/system-data

• **DBLP:** This is a computer science bibliography dataset. We adopt a subset of DBLP extracted by [45], which contains 4057 authors (A), 20 venues (V), 8789 terms (T) and 14328 papers (P). The authors are labeled in four categories, i.e., database, data mining, information retrieval and machine learning. Notably, DBLP is not a natural multi-view dataset but a heterogeneous graph network. Unlike the above four multi-view scenes, we apply a relation-based graph splitting strategy to generate three homogeneous graphs with different semantics. Specifically, we treat authors as the graph's nodes and develop three view relations: *A-P-A* links two co-authors. *A-P-T-P-A* links two persons whose papers containing same terms. *A-P-V-P-A* links two persons whose papers are published in the same venue.

4.2 Baselines and Metrics

We choose the following ten SOTA graph-based methods or GNN-based methods designed for multi-view graph representation learning as baselines.

• **Graph-based Methods: 1) CoRegSC** [46] is proposed for multi-view clustering through co-regularized. Here, the centroid-based strategy is utilized for learning multi-view graphs' representations. **2) MultiNMF** [47] is a clustering method that applies non-negative matrix factorization, the goal of which is to find a scheme that provides a suitable clustering implementation across multiple views. **3) RMSC** [48] is a multi-view spectral clustering method with great robustness and considering the application of low rank and sparse decomposition. **4) AMGL** [49] is a representation learning method that learns each graph's best weight without additional parameters. **5) M2E** [5] is a multi-view multi-graph representation learning framework based on partially-symmetric tensor decomposition.

• **GNN-based Methods: 1) GCN** [12] is a graph representation learning model that performs convolution operations in the graph Fourier domain. **2) GAT** [16] is a graph representation learning model that performs convolution operations on topological graphs based on an incorporated attention mechanism. **3) HAN** [29] is a heterogeneous graph learning model that learns relation-specific representations from different homogeneous graphs and uses attention techniques to fuse them into the final feature matrix. **4) MAGNN** [28] is a heterogeneous graph representation learning model that utilizes linear transformation to learn node attributes and applies a special encoder to aggregate node-level information. **5) TensorGCN** [31] is a recently proposed tensor GCN-based representation learning method for multi-view graph scenes.

• **Our Methods: 1)** Two variants of RTGNN, namely **RTGNN-mean** and **RTGNN-att**, are proposed for classification or clustering tasks. They apply mean and attention operations instead of convolution operations in the inter-graph aggregation, respectively. **2)** Four additional variants of RTGNN are proposed for ablation studies. **RTGNN-m0** is equivalent to RTGNN with two modules removed. **RTGNN-m1** and **RTGNN-m2** represent that RTGNN only utilizes the bridge and RL-guided module, respectively. **RTGNN-m3** only uses the proposed measure and manually sets the filtering thresholds.

To measure the effectiveness of all these methods, we employ two frequently applied metrics, i.e., Macro-F1 (Ma-

TABLE 3: Experiment results (%) of classification tasks on four datasets. The best results of all methods are shown in bold. The best results of all baselines are shown in italics. ↑ indicates the relative improvement of RTGNN to the best baseline.

Methods		Train%	HIV		BP		BikeDC		DBLP	
			Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1
Graph-based Methods	CoRegSC	20%	49.34 ± 03.17	50.51 ± 04.72	46.31 ± 08.02	54.00 ± 02.63	20.47 ± 04.67	35.29 ± 03.72	80.63 ± 00.28	81.61 ± 00.93
		60%	59.19 ± 07.16	60.14 ± 09.56	54.13 ± 07.24	57.82 ± 04.36	29.05 ± 03.75	43.52 ± 05.99	82.90 ± 00.15	83.88 ± 00.33
	MultiNMF	20%	48.85 ± 06.94	49.19 ± 03.81	53.84 ± 09.24	58.22 ± 05.60	22.24 ± 02.75	37.35 ± 02.29	78.43 ± 00.43	79.20 ± 00.70
		60%	53.36 ± 08.89	54.71 ± 04.19	55.10 ± 07.58	58.69 ± 06.22	26.21 ± 02.77	42.35 ± 04.40	82.02 ± 00.20	82.57 ± 00.19
	RMSC	20%	51.24 ± 06.93	51.69 ± 06.47	43.95 ± 08.19	53.33 ± 03.29	20.38 ± 03.58	34.11 ± 02.99	83.46 ± 00.51	83.82 ± 00.18
		60%	62.50 ± 09.54	63.12 ± 06.07	53.96 ± 06.98	57.39 ± 03.79	30.57 ± 06.14	45.88 ± 08.64	84.89 ± 00.27	85.03 ± 00.32
	AMGL	20%	48.26 ± 08.68	49.48 ± 03.06	33.82 ± 10.68	51.11 ± 04.24	19.21 ± 05.97	35.58 ± 05.17	80.52 ± 00.21	81.38 ± 00.91
		60%	56.60 ± 09.54	58.76 ± 08.47	34.28 ± 09.31	52.17 ± 05.49	25.84 ± 04.85	43.52 ± 06.55	84.50 ± 00.37	85.11 ± 00.13
	M2E	20%	53.98 ± 03.03	56.25 ± 05.13	51.47 ± 07.45	55.55 ± 03.71	22.85 ± 05.22	38.23 ± 04.36	83.45 ± 00.12	83.82 ± 00.18
		60%	63.75 ± 03.75	65.00 ± 05.00	54.70 ± 06.98	56.95 ± 05.97	30.21 ± 03.75	45.88 ± 06.85	85.74 ± 00.05	86.41 ± 00.09
GNN-based Methods	GCN	20%	55.60 ± 03.03	56.56 ± 03.93	50.54 ± 08.94	55.77 ± 03.50	22.75 ± 03.28	39.11 ± 03.49	89.12 ± 00.17	89.94 ± 00.16
		60%	65.73 ± 04.85	67.50 ± 04.67	57.24 ± 07.12	59.56 ± 05.84	30.73 ± 03.57	48.23 ± 04.40	90.17 ± 00.05	90.40 ± 00.12
	GAT	20%	55.98 ± 04.73	56.25 ± 02.82	50.58 ± 06.98	55.55 ± 02.81	23.83 ± 04.33	39.70 ± 03.54	91.06 ± 00.03	91.19 ± 00.07
		60%	65.69 ± 04.19	68.75 ± 04.14	57.57 ± 05.88	60.00 ± 05.07	31.98 ± 04.73	48.23 ± 06.33	91.94 ± 00.18	92.35 ± 00.09
	HAN	20%	58.90 ± 05.04	60.00 ± 03.64	*56.18 ± 06.31*	*59.33 ± 04.97*	*25.40 ± 02.26*	40.00 ± 03.52	91.92 ± 00.08	92.42 ± 00.07
		60%	69.55 ± 04.23	70.62 ± 04.88	*60.29 ± 02.50*	*62.60 ± 03.47*	33.07 ± 05.16	50.00 ± 03.94	92.10 ± 00.04	93.56 ± 00.12
	MAGNN	20%	56.66 ± 04.75	58.43 ± 03.14	54.01 ± 05.91	57.11 ± 02.63	24.04 ± 03.53	37.05 ± 04.20	*93.30 ± 00.27*	*93.82 ± 00.32*
		60%	69.84 ± 04.59	71.25 ± 05.00	59.74 ± 06.62	62.17 ± 05.16	32.10 ± 06.47	49.41 ± 06.55	*94.35 ± 00.06*	*94.58 ± 00.41*
	TensorGCN	20%	*59.02 ± 04.38*	*60.31 ± 02.81*	53.07 ± 07.54	56.88 ± 04.68	25.30 ± 02.68	*40.29 ± 03.73*	92.40 ± 00.24	93.08 ± 00.11
		60%	*70.98 ± 04.36*	*72.50 ± 05.00*	60.29 ± 08.42	61.73 ± 08.43	*33.99 ± 08.56*	*50.58 ± 09.18*	93.30 ± 00.13	93.99 ± 00.09
Our Methods	RTGNN -mean	20%	64.48 ± 05.97	66.87 ± 05.44	57.13 ± 09.55	59.77 ± 05.74	31.84 ± 06.01	44.41 ± 03.82	94.13 ± 00.07	94.58 ± 00.06
		60%	72.23 ± 08.59	73.75 ± 08.29	63.86 ± 05.51	64.34 ± 05.43	45.35 ± 09.68	54.11 ± 10.12	95.19 ± 00.41	95.48 ± 00.36
	RTGNN -att	20%	65.26 ± 07.05	67.18 ± 05.45	58.09 ± 03.15	60.44 ± 02.39	33.48 ± 03.98	45.58 ± 01.97	94.30 ± 00.12	94.76± 00.30
		60%	73.07 ± 06.88	74.37 ± 05.89	65.67 ± 05.50	66.08 ± 05.43	46.27 ± 08.05	55.88 ± 05.42	95.21 ± 00.16	95.56 ± 00.05
	RTGNN	20%	**65.74 ± 05.75**	**67.50 ± 06.43**	**60.70 ± 04.62**	**61.11 ± 04.44**	**35.22 ± 07.25**	**45.88 ± 03.76**	**94.50 ± 00.13**	**94.91 ± 00.15**
		60%	**75.14 ± 08.18**	**76.25 ± 08.29**	**66.12 ± 06.00**	**66.52 ± 06.16**	**48.25 ± 05.10**	**57.05 ± 04.59**	**95.42 ± 00.26**	**95.62 ± 00.44**
Gain		20%	**6.72** ↑	**7.19** ↑	**4.52** ↑	**1.78** ↑	**9.82** ↑	**5.59** ↑	**1.20** ↑	**1.09** ↑
		60%	**4.16** ↑	**3.75** ↑	**5.83** ↑	**3.92** ↑	**14.26** ↑	**6.47** ↑	**1.07** ↑	**1.04** ↑

F1) and Micro-F1 (Mi-F1), to expose the classification performance. Besides, we apply normalized mutual information (NMI) and adjusted rand index (ARI) to quantitatively evaluate/verify the clustering performance.

4.3 Implementation Details

For our RTGNN and its six variants, we first study the parameters related to the bridge and RL-guided module. For the bridge module, we need to consider the rank of HOSVD and the dimension of the common feature space. Specifically, since the projecting process mentioned in Sec. 3.1 only involves the factor matrices (i.e., $\mathbf{U}$ and $\mathbf{V}$ in Eq. (3)) of the last two modes (i.e., nodes and features) and different factor matrices can be calculated independently in HOSVD, we only need to consider the calculation of $\mathbf{U}$ and $\mathbf{V}$. Considering that the 1-mode product of $TRAN(\mathbf{U})$ and the initial feature matrix $\mathbf{F}$ affects the dimension of the node mode of the corresponding enhanced feature matrix $\widehat{\mathbf{F}}$, $\mathbf{U}$ should be a square matrix whose dimension is equal to the number of nodes, i.e., $\mathbf{U} = \mathbf{I} \in \mathbb{R}^{|N|\times|N|}$. Unlike $\mathbf{U}$, the factor matrix $\mathbf{V}$ of the feature mode of the initial feature tensor $\mathcal{F}^{(0)}$ determines the dimension of the common feature space after 2-mode product. We argue that the enhanced feature matrix $\widehat{\mathbf{F}}$ should maintain the same feature dimension as the initial feature matrix $\mathbf{F}$ to ensure the fairness of subsequent ablation studies. Therefore, we still retain all the singular values in the process of calculating $\mathbf{V}$. In particular, because multi-view datasets like HIV and BP do not contain feature tensors, we follow the standard strategy used in previous multi-view graph analysis [1], [5], [3], i.e., the initial adjacency tensor is viewed as the input feature tensor of RTGNN. Hence, the dimension of the common feature space in the multi-view datasets is equal to the number of nodes. For the RL-guided module, we start the filtering threshold calculator at the end of the second epoch and the initial filtering threshold for each view to 0.5. Finally, the number of GNN layers is set to 1.

For GCN and GAT, we follow the previous works in [29], [28] to perform them on each view separately and report the best results. For all methods for spectral clustering, we build RBF kernel matrices with width equal to the median distance between all multi-view graphs. For all the deep learning methods, we employ the Adam optimizer with a learning rate of 0.005 and the weight decay of 0.001. Furthermore, we train them for 100 epochs with the dropout rate of 0.5 and utilize early stopping with a patience of 5. For methods involving attention mechanisms, we set the number of attention heads to 8 and the dimension of the attention vector to 128. For a fair comparison, We set the dimension $D^{(L)}$ of the final feature matrix $\mathbf{E} \in \mathbb{R}^{M\times D^{(L)}}$ obtained by all the above methods to 64. We tune all baselines' parameters through the policies developed in their works and report their performance with the optimal settings. We divide the first four datasets into training, validation, and test sets according to the ratio of 3:1:6, and divide the DBLP dataset according to the ratio/split of 1:1:8. Each experiment is repeated 20 times using the same data splits and take the mean values as the results.

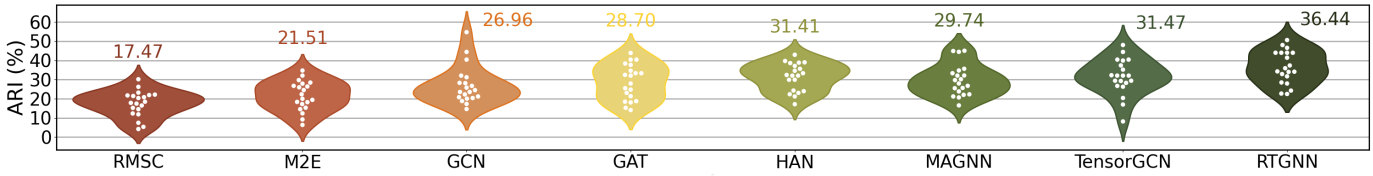


Fig. 3: Visualization of clustering experiments on the BikeDC dataset. The white dots represent the results of 20 repeated experiments, and the values represent the average results under the metric ARI (%).

TABLE 4: Experiment results (%) on the PROTEINS dataset for multi-view graph classification.

Methods	PROTEINS			
	Ma-F1		Mi-F1	
	20%	60%	20%	60%
GCN	71.06 ± 0.24	74.60 ± 0.69	72.16 ± 0.21	76.25 ± 0.67
GAT	71.32 ± 0.59	75.16 ± 0.85	72.19 ± 0.49	75.58 ± 0.94
HAN	71.98 ± 0.59	*76.04 ± 0.51*	72.58 ± 0.62	*76.83 ± 0.50*
MAGNN	72.19 ± 0.59	75.97 ± 1.25	72.66 ± 0.65	76.58 ± 1.24
TensorGCN	*72.19 ± 0.60*	75.93 ± 0.69	*72.75 ± 0.58*	76.50 ± 0.67
RTGNN	**73.28 ± 0.48**	**77.08 ± 1.18**	**73.81 ± 0.54**	**77.79 ± 1.19**
Gain	**1.51** ↑	**1.37** ↑	**1.46** ↑	**1.25** ↑

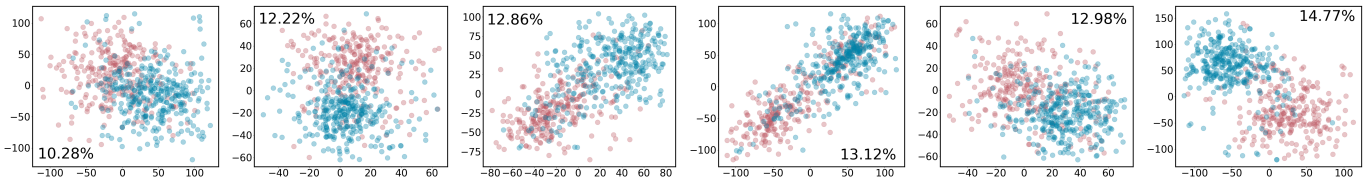
TABLE 5: Experiment results (%) of clustering tasks on two brain science datasets.

Methods		Metrics	HIV	BP
Graph-based Methods	CoRegSC	NMI	18.51 ± 06.53	12.22 ± 06.62
		ARI	11.01 ± 06.32	07.53 ± 08.68
	MultiNMF	NMI	14.21 ± 07.68	14.17 ± 06.26
		ARI	08.33 ± 06.49	11.58 ± 07.11
	RMSC	NMI	15.85 ± 08.54	13.65 ± 08.57
		ARI	07.73 ± 04.06	10.50 ± 05.02
	AMGL	NMI	09.62 ± 07.72	09.10 ± 07.46
		ARI	06.90 ± 07.05	05.27 ± 09.58
	M2E	NMI	19.52 ± 08.46	13.33 ± 10.84
		ARI	07.86 ± 05.12	12.55 ± 10.12
GNN-based Methods	GCN	NMI	*20.67* ± 09.70	13.49 ± 05.18
		ARI	11.61 ± 07.42	11.88 ± 04.47
	GAT	NMI	20.16 ± 07.17	13.65 ± 08.57
		ARI	11.83 ± 05.35	12.59 ± 09.05
	HAN	NMI	23.73 ± 07.17	*18.51 ± 08.86*
		ARI	17.59 ± 14.75	13.34 ± 08.19
	MAGNN	NMI	19.30 ± 10.96	16.20 ± 06.08
		ARI	*20.33 ± 11.98*	13.68 ± 07.63
	TensorGCN	NMI	*24.38 ± 13.50*	17.27 ± 06.93
		ARI	19.44 ± 09.14	*13.98 ± 07.93*
Our Method	RTGNN	NMI	**35.02 ± 11.43**	**19.70 ± 06.38**
		ARI	**30.86 ± 10.81**	**17.26 ± 09.80**
Gain		NMI	**10.64** ↑	**1.19** ↑
		ARI	**10.53** ↑	**3.28** ↑

4.4 Classification Results (RQ1)

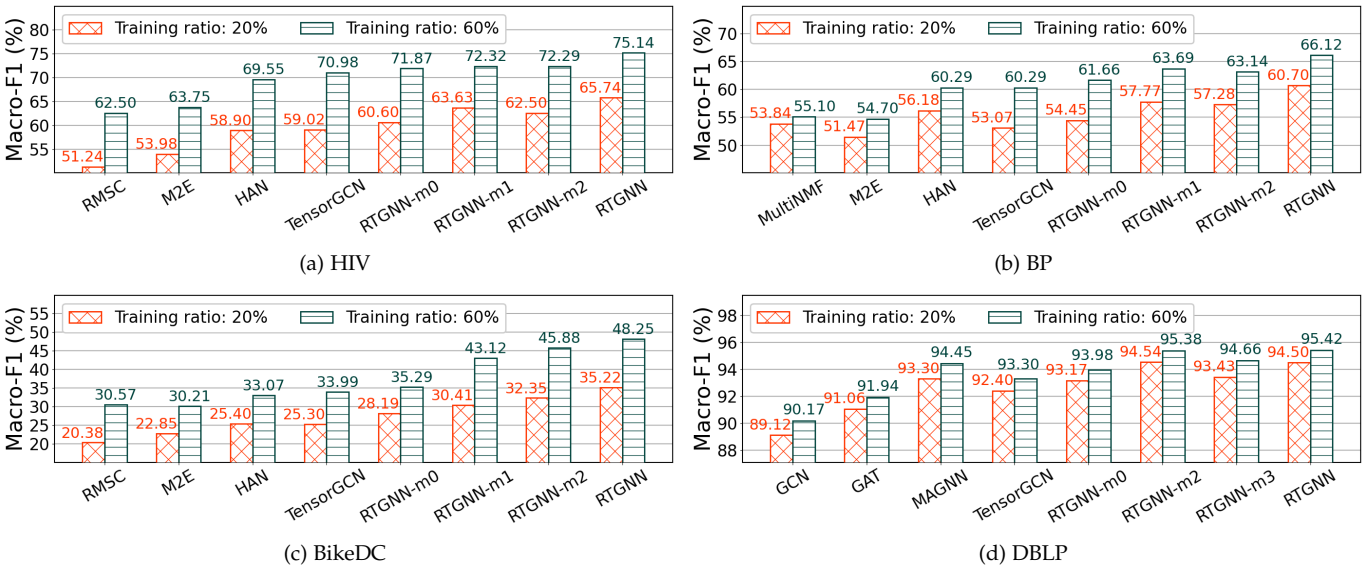
We conduct experiments on all datasets to compare the performance of different methods on classification tasks. Following the work [28], we feed the low-dimensional feature vectors of every method into Support Vector Machine (SVM) at different training ratios (i.e., 20% and 60%) to achieve classification. Considering that we only provide the test set to the linear SVM, the training and testing ratios here only concern the test set. From the results shown in Table 3, we can draw six conclusions:

1) The proposed RTGNN consistently outperforms all baselines on the four datasets (i.e., HIV, BP, BikeDC and DBLP) and achieves the maximum 14.26% performance improvement over the second-best method. This is because the two modules contained in RTGNN effectively reinforce the aggregation operations. First, the bridge module takes advantage of tensor decomposition in capturing the PCI of multiple views and reduces the risk of graph feature fusion conflicts in the inter-graph aggregation. Second, the RL-guided module facilitates the GSF mining (i.e., intra-graph aggregation) by dynamically finding the most valuable neighbors for each graph, which is infeasible for all baselines. **2)** Among all the baselines, GCN and GAT are two methods designed for single-view graph representation learning. We find that they both achieve lower F1 scores than the other multi-view GNN-based methods. This phenomenon demonstrates that different views can be complementary, and combining the information of multiple views may produce a better graph feature matrix than that of a single view. The extended multi-channel GNN is more suitable for processing multi-view graphs than traditional GNNs. **3)** Compared with HAN and MAGNN, TensorGCN performs better on the HIV and BikeDC but slightly worse on the BP dataset. The possible reason is that the cross-view adjacency tensor applied in TensorGCN encodes the interaction between graph structures, thereby achieving better inter-graph aggregation than the weighted summation, especially when the number of views is large. Adopting a similar fusion strategy, our node-aware inter-graph aggregation optimizes the cross-views adjacency tensors through model training rather than prior knowledge to avoid degenerating into a mean operation when there are only two views. Therefore, RTGNN always beats all its variants. **4)** On the DBLP dataset, MAGNN stands out among all baselines because it can judiciously transcribe the heterogeneous node features. Even though the classification results of MAGNN are remarkable, RTGNN still obtains higher scores than them. Considering that RTGNN only uses homogeneous node features, the above phenomenon again verifies the effectiveness of the proposed RL-guided module. Furthermore, it confirms that our RTGNN can handle not only natural multi-view graph data but also assist in internal multi-view analysis of heterogeneous graph networks. **5)** The four multi-view graph-based baselines (i.e., CoRegSC, MultiNMF, RMSC, and AMGL) have a commonality, i.e., the learned view features are all represented by vectors. Unfortunately, for the multi-view graph, the topological feature information is hardly persevered by the flattened dense feature vectors, which could be the underlying reason why they are always weaker than the GNN-based methods. **6)** By applying a tensor decomposition algorithm to encode multi-view graph-structured data, M2E explores multiple



(a) RMSC (b) M2E (c) HAN (d) MAGNN (e) TensorGCN (f) RTGNN

Fig. 4: Visualization of graph representations learned by six methods on the PROTEINS. The red dots represent enzyme molecules, the blue dots represent non-enzyme molecules, and the values is the average results under the metric NMI (%).



(a) HIV (b) BP

(c) BikeDC (d) DBLP

Fig. 5: Visualization of ablation studies related to classification tasks on the HIV, BP, BikeDC, and DBLP.

views and graphs simultaneously in the common feature space like our RTGNN, which performs better than the other multi-view graph-based methods. Nevertheless, M2E cannot fully exploit the graph structural features of each view like the GNN-based methods. These observations indicate the necessity of PCI and GSF mining for reinforcing multi-view graph representations and verify our motivation for building the bridge module and a series of feature aggregation operations.

Since there are fewer instances in HIV, BP, and BikeDC, we further verify the scalability of RTGNN on the PROTEINS dataset. The classification results in Table 4 again verify that RTGNN is superior to the excellent GNN-based baselines on the larger multi-view graph dataset.

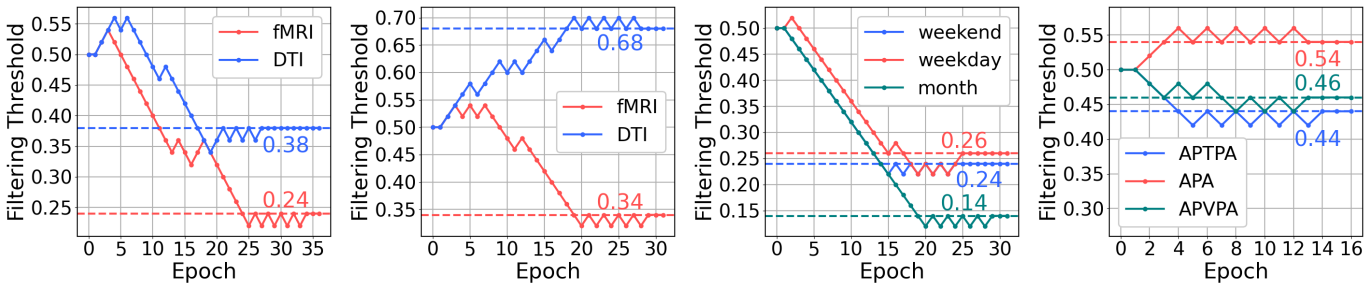
4.5 Clustering Results (RQ2)

We compare the performance of multiple methods on multi-view graph clustering tasks (i.e., the HIV and BP dataset). Following the previous works in [48], [49], [5], [29], [28], we input the low-dimensional features of multi-view graphs into the K-means technique. The number of classes/categories per multi-view dataset determines the number of clusters in K-means (e.g., 2 for both the HIV and BP). As mentioned in Sec. 4.3, we repeat each clustering experiment 20 times to overcome the disturbance of center initialization on the results. From the results shown in Table 5, conclusions similar to those of classification experiments can be drawn, which proves that our RTGNN learns more distinguishable representations than the state-of-the-art baselines. Specifically, RTGNN achieves the maximum 10.63% performance improvement over the best baseline. To better illustrate the feature quality of the multi-view graphs, we visually compare some methods on the BikeDC and PROTEINS dataset. As shown in Fig. 3, compared with baselines, RTGNN has more minor fluctuations while obtaining higher ARI values. Based on t-SNE, Fig. 4 illustrates that RTGNN learns higher quality representations with less overlap and better separation of groups.

4.6 Ablation Studies (RQ3)

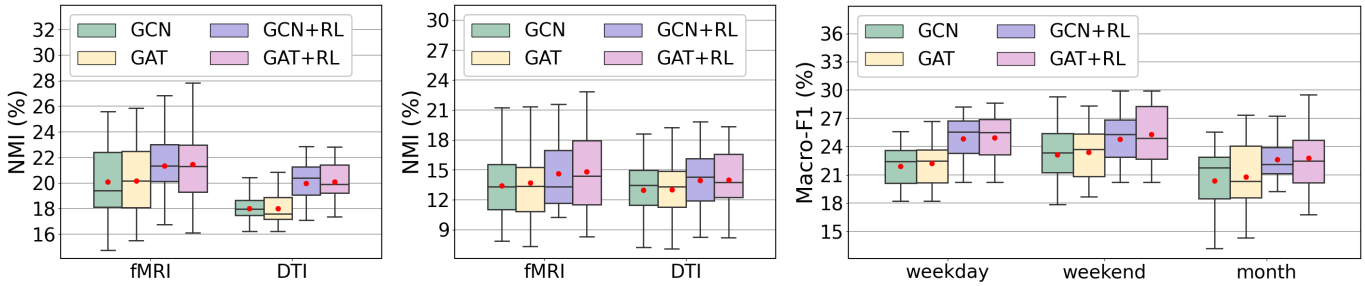
Furthermore, we perform a lot of ablation studies to explore the specific effects of the bridge module and RL-guided module included in our RTGNN on the above classification and clustering tasks. For classification tasks, we will focus on the experiments of RTGNN with two training ratios under the metric Macro-F1. For clustering tasks, we will focus on the experiments under the metric NMI. Fig. 5 illustrates the results of the ablation studies on four classification tasks, from which we draw the following five conclusions:

1) Although the variants of RTGNN cannot achieve the best performance after removing the bridge module or RL-guided module, they are still better than the strongest baselines on the HIV, BP and BikeDC dataset. This indicates that these two modules effectively improve the multi-view GNN and harmoniously coexist. **2)** On the HIV and BP, the bridge module is more important than the RL-guided



(a) HIV (b) BP (c) BikeDC (d) DBLP

Fig. 6: Visualization of the adaptive update process of filtering thresholds for different datasets and different views.



(a) HIV (b) BP (c) BikeDC

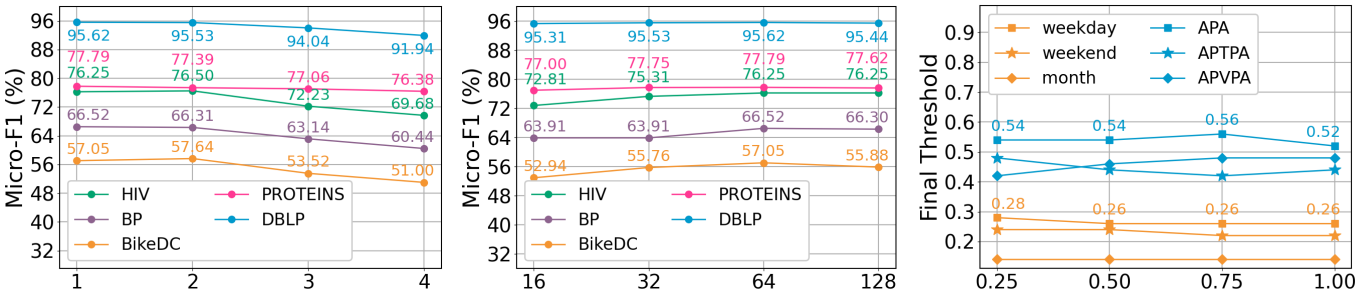
Fig. 7: Performance comparison between original GNNs and GNNs enhanced by the RL-guided module on clustering and classification in each view of different datasets. The red dot in each box indicates the corresponding average result.

module, which is reflected by the fact that RTGNN-m1 is better than RTGNN-m2 under different training ratios. In contrast, the RL-guided module shows its promising advantages on the BikeDC and DBLP. The first phenomenon can be explained by the large differences in the features of different views of the brain science datasets. RTGNN-m1 projects different view features into the common feature space, which alleviates the feature fusion conflicts in the inter-graph aggregation. For the second phenomenon, the possible reason is that the BikeDC and DBLP dataset have richer neighbor information, so RTGNN-m2, which removes the interference of irrelevant neighbors, stands out among the variants. **3)** From Fig. 6, we find that the RL-guided module simultaneously updates multiple filtering thresholds during the model training process and dynamically determine the stop time according to the convergence condition (i.e., Eq. (9)). **4)** The optimal filtering thresholds for different views in different datasets can vary greatly. Views that encode more edges may not necessarily be assigned larger filter thresholds. For example, in the brain science dataset, the edge density of the graphs in fMRI is often higher than that of DTI, and the thresholds of the former are smaller than those of the latter (as shown in Fig. 6(a) and Fig. 6(b)). These observations indicate that there may be some neighbors in the graph that are worthless for representation learning, and reveal the inefficiency and unreliability of manual settings and the robustness of our RTGNN to view changes. **5)** The last sub-graph in Fig. 5 corresponds to DBLP. Considering that there are no multiple natural views in DBLP, we directly use the initial features just like the previous works in [29], [28]. Specifically, we compare RTGNN-m2 with RTGNN-m3, equivalent to RTGNN-m2 without the threshold calculator. The results show that RTGNN-m3 is relatively weak but still surpasses the best baseline, verifying the effectiveness of the importance measure and node-aware inter-graph aggregation.

In addition, we compare the performance of the original GNNs and the GNNs enhanced by the RL-guided module on clustering and classification in different views of different datasets. It can be seen from the visualization results in Fig. 7 that the proposed RL-guided module improves the performance of the original GNNs in different views. This verifies that the RL-guided module has excellent transferability, i.e., it is suitable for both single-view and multi-view graph representation learning.

4.7 Hyperparameter Analysis (RQ4)

We study the impact of important hyperparameters of RTGNN, namely the number of GNN layers, the dimension of representations and the initial filtering thresholds. Specifically, we will focus on the classification experiments with the training ratio of 60% under the metric Micro-F1. Fig. 8(a) shows that as the number of GNN layers increases, the performance of RTGNN tends to decrease. We argue that this is caused by the over-smoothing problem [24] that GNN-based methods often face. From Fig. 8(b), it can be seen that the quality of the multi-view graph representation learning of RTGNN is not easily affected by the change of the representation dimension, especially on the PROTEINS and DBLP dataset. The sensitivity analysis of the filtering threshold initialization on the BikeDC and DBLP dataset illustrated in Fig. 8(c) attests that the proposed RL-guided module stably finds the optimal threshold interval regardless of the initial values.



(a) Number of GNN layers (b) Dimension of representations (c) Initial filtering threshold

Fig. 8: Parameter sensitivity analysis of RTGNN on the five datasets.

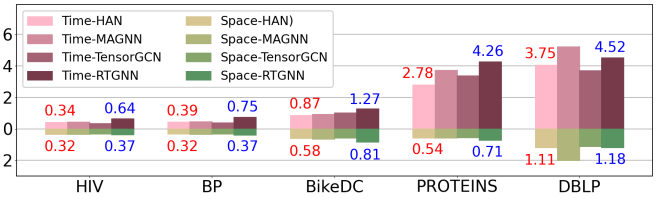


Fig. 9: Comparison of execution time (Minute) and space requirements (Gigabyte) of different methods. The blue values correspond to RTGNN, and the red values correspond to the optimal consumption of the baselines.

4.8 Discussion (RQ5)

Based on the multi-view GNN and two modules, RTGNN outperforms the baselines on five datasets. However, the introduction of modules may make RTGNN have potential limitations in terms of execution time and space requirements. For the bridge module, it requires additional runtime and memory space while using the core tensor to capture correlations. It takes $\mathcal{O}(MV|N|D\min(|N|, MVD) + \mathcal{O}(MV|N|D\min(D, MV|N|))$ to calculate the two projection matrices of the input tensor $\mathcal{F} \in \mathbb{R}^{M\times V\times|N|\times D}$. Then it needs to take $\mathcal{O}(MV|N|D\min(|N|, D))$ on the input tensor to enhance each feature matrix $\mathbf{F} \in \mathbb{R}^{|N|\times D}$. For the RL-guided module, it needs $\mathcal{O}(LV(M|N|^2D + 1))$ to achieve threshold calculation and neighbor filtering in all views for each epoch. And we take $\mathcal{O}(LMV|N|(2D\min(|N|, D, V)))$ for feature aggregation within the multi-view GNN and $\mathcal{O}(MV|N|)$ to perform vectorization. Hence, the computational complexity of RTGNN is $\mathcal{O}(LVM|N|^2D)$, which is acceptable but can still be improved in the future.

Fig. 9 illustrates the execution time and maximum memory space requirements of different methods of running an epoch on different datasets, where all experiments are performed on an NVIDIA Tesla P100 16G GPU. We observe that the time and space requirements of RTGNN on the five datasets are greater than (which is acceptable) the optimal consumption of the baselines. Considering the excellent performance of RTGNN, we argue that it is worth sacrificing a certain amount of time and space in exchange for improving the quality of representation learning.

5 Related Work

Existing literature can be roughly classified into tensor-based multi-view graph representation, tensor graph neural networks, and GNN models with neighbor selection.

5.1 Tensor-based Multi-view Graph Representation

Many tensor-based methods for multi-view graph representation learning have been proposed. For example, [50] proposed a graph-based multi-view representation integration method, which captures higher-order information by mapping the original graph to a set of cross-view tensor product graphs. [8] proposed a multi-view clustering framework, first stacking multi-view graphs into tensors, and then learning graph embeddings based on CP tensor decomposition. [5] presented a multi-view multi-graph representation framework with partially-symmetric tensor decomposition. However, most of the existing methods are based on shallow frameworks or models and cannot mine the deep features of multi-view graphs like GNNs.

5.2 Tensor Graph Neural Networks

The existing tensor graph neural networks are usually based on GCNs. [51] proposed a tensor GCN that globally models those sub-graphs factorized from a large graph. Recently, [32] proposed a tensor framework based on residual GCN layers, enabling robust learning for single- or multi-relational data when the underlying topology is perturbed. [31] represented the text semantics as a three-dimensional graph tensor and then performed two kinds of propagation learning on the network tensor for text classification. Although they all involve the combination of tensor techniques and multiplex GCNs, they only extend the matrix operations of GCN to the multi-channel tensor version. That is, they do not involve the utilization of Tucker decomposition to capture the PCI of the multi-view graphs, nor do they consider ranking and filtering the neighbor nodes in the graph to reinforce the intra-graph aggregation.

5.3 GNN Models with Neighbor Selection

Recently, GNNs with neighbor selection have received increasing attention. For example, [25] proposed a label-aware GCN, which filters irrelevant neighbors based on node labels to improve the node classification performances of GNN models (including GAT). To avoid the over-assimilation of the representations of different types of nodes in GNN aggregation, [26] proposed a multi-relational GNN framework that contains a relation-aware mechanism for neighbor filtering. However, the existing methods do not apply edge information widely existing in graph data to assist neighbor selection.

6 Conclusion

In this paper, we propose a novel multi-view GNN-based framework, namely RTGNN, for graph representation learning. To reinforce the inter-graph aggregation of the multi-view GNN, we propose a new bridge module based on HOSVD and node-aware aggregation. To reinforce the intra-graph aggregation, we propose the RL-guided module composed of a neighbor importance measure and a filtering threshold calculator. The two modules included in RTGNN can be easily ported to other methods. Experimental results and analysis demonstrate that RTGNN consistently surpasses SOTA baselines on five real-world multi-view scenes. In the future, we will focus on how to effectively combine tensor decomposition and GNNs to further reduce time and space requirements while ensuring performance.

Acknowledgment

The authors of this paper are supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grants 62002007, U20B2053 and 62172443, S&T Program of Hebei through grant 20310101D, the ARC DECRA Project (No. DE200100964), and Fundamental Research Funds for the Central Universities. Thanks for computing infrastructure provided by Huawei MindSpore platform. Philip S. Yu is partially supported by NSF under grants III-1909323, III-2106758, and SaTC-1930941.

References

[1] S. Wang, L. He, B. Cao, C.-T. Lu, P. S. Yu, and A. B. Ragin, "Structural deep brain network mining," in *KDD*. ACM, 2017, pp. 475–484.

[2] M. Zhang, T. Li, Y. Li, and P. Hui, "Multi-view joint graph representation learning for urban region embedding," in *IJCAI*, 2020, pp. 4431–4437.

[3] X. Zhang, L. He, K. Chen, Y. Luo, J. Zhou, and F. Wang, "Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson's disease," in *AMIA Annual Symposium Proceedings*. AMIA, 2018, pp. 1147–1156.

[4] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *ICML*. PMLR, 2020, pp. 4116–4126.

[5] Y. Liu, L. He, B. Cao, P. S. Yu, A. B. Ragin, and A. D. Leow, "Multi-view multi-graph embedding for brain network clustering analysis," in *AAAI*. AAAI Press, 2018, pp. 117–124.

[6] J. Cheng, Q. Wang, Z. Tao, D. Xie, and Q. Gao, "Multi-view attribute graph convolution networks for clustering," in *IJCAI*, 2020, pp. 2973–2979.

[7] C. Liu, Z. Liao, Y. Ma, and K. Zhan, "Stationary diffusion state neural estimation for multiview clustering," in *AAAI*, 2022, pp. 1–8.

[8] G. Ma, L. He, C.-T. Lu, W. Shao, P. S. Yu, A. D. Leow, and A. B. Ragin, "Multi-view clustering with graph embedding for connectome analysis," in *CIKM*. ACM, 2017, pp. 127–136.

[9] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*. ACM, 2014, pp. 701–710.

[10] H. Peng, R. Yang, Z. Wang, J. Li, L. He, P. S. Yu, A. Y. Zomaya, and R. Ranjan, "Lime: Low-cost incremental learning for dynamic heterogeneous information networks," *IEEE Transactions on Computers*, 2021.

[11] M. R. Khan and J. E. Blumenstock, "Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty," in *AAAI*, vol. 33, no. 01, 2019, pp. 606–613.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017, pp. 1–14.

[13] J. Li, H. Peng, Y. Cao, Y. Dou, H. Zhang, P. Yu, and L. He, "Higher-order attribute-enhancing heterogeneous graph neural networks," *IEEE TKDE*, no. 01, pp. 1–1, 2021.

[14] H. Peng, J. Li, Y. Song, R. Yang, R. Ranjan, P. S. Yu, and L. He, "Streaming social event detection and evolution discovery in heterogeneous information networks," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 5, pp. 1–33, 2021.

[15] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.

[16] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018, pp. 1–14.

[17] S. Li, W.-T. Li, and W. Wang, "Co-gcn for multi-view semi-supervised learning," in *AAAI*. AAAI Press, 2020, pp. 4691–4698.

[18] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *WWW*, 2020, pp. 3070–3076.

[19] Q. Liu, M. C. Kampffmeyer, R. Jenssen, and A.-B. Salberg, "Multi-view self-constructing graph convolutional networks with adaptive class weighting loss for semantic segmentation," in *CVPR Workshops*. IEEE, 2020, pp. 199–205.

[20] J. Sun, J. Zhang, Q. Li, X. Yi, Y. Liang, and Y. Zheng, "Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks," *IEEE TKDE*, no. 01, pp. 1–1, 2020.

[21] C. Jia, B. Wu, and X.-P. Zhang, "Dynamic spatiotemporal graph neural network with tensor network," *arXiv*, 2020.

[22] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the inconsistency problem of applying graph neural network to fraud detection," in *SIGIR*. ACM, 2020, pp. 1569–1572.

[23] Y. Hou, J. Zhang, J. Cheng, K. Ma, R. T. B. Ma, H. Chen, and M.-C. Yang, "Measuring and improving the use of graph information in graph neural networks," in *ICLR*, 2019.

[24] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *AAAI*. AAAI Press, 2020, pp. 3438–3445.

[25] H. Chen, Y. Xu, F. Huang, Z. Deng, W. Huang, S. Wang, P. He, and Z. Li, "Label-aware graph convolutional networks," in *CIKM*. ACM, 2020, pp. 1977–1980.

[26] H. Peng, R. Zhang, Y. Dou, R. Yang, J. Zhang, and P. S. Yu, "Reinforced neighborhood selection guided multi-relational graph neural networks," *TOIS*, 2021.

[27] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *CIKM*. ACM, 2020, pp. 315–324.

[28] X. Fu, J. Zhang, Z. Meng, and I. King, "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding," in *WWW*, 2020, pp. 2331–2341.

[29] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*. ACM, 2019, pp. 2022–2032.

[30] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[31] X. Liu, X. You, X. Zhang, J. Wu, and P. Lv, "Tensor graph convolutional networks for text classification," in *AAAI*. AAAI Press, 2020, pp. 8409–8416.

[32] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, "Tensor graph convolutional networks for multi-relational and robust learning," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6535–6546, 2020.

[33] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[34] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *ECML*, vol. 3720. Springer, 2005, pp. 437–448.

[35] Y. Cao, Z. Liu, C. Li, Z. Liu, J. Li, and T.-S. Chua, "Multi-channel graph neural network for entity alignment," in *ACL*. Association for Computational Linguistics, 2019, pp. 1452–1461.

[36] H. A. L. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.

[37] S. F. Yilmaz and S. S. Kozat, "Unsupervised anomaly detection via deep metric learning with end-to-end optimization," *arXiv*, 2020.

[38] A. B. Ragin, H. Du, R. Ochs, Y. Wu, C. L. Sammet, A. Shoukry, and L. G. Epstein, "Structural brain alterations can be detected early in hiv infection," *Neurology*, vol. 79, no. 24, pp. 2328–2334, 2012.

[39] C. Yan and Y. Zang, "Dparsf: A matlab toolbox for "pipeline" data analysis of resting-state fmri," *Frontiers in systems neuroscience*, vol. 4, p. 13, 2010.

[40] S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E. Behrens, H. Johansen-Berg, P. R. Bannister, M. De Luca, I. Drobn-

jak, D. E. Flitney *et al.*, "Advances in functional and structural mr image analysis and implementation as fsl," *Neuroimage*, vol. 23, pp. S208–S219, 2004.
[41] B. Cao, L. Zhan, X. Kong, P. S. Yu, N. Vizueta, L. L. Altshuler, and A. D. Leow, "Identification of discriminative subgraph patterns in fmri brain networks in bipolar affective disorder," in *International Conference on Brain Informatics and Health*. Springer, 2015, pp. 105–114.
[42] S. Whitfield-Gabrieli and A. Nieto-Castanon, "Conn: A functional connectivity toolbox for correlated and anticorrelated brain networks," *Brain connectivity*, vol. 2, no. 3, pp. 125–141, 2012.
[43] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, pp. i47–i56, 2005.
[44] N. Adaloglou, N. Vretos, and P. Daras, "Multi-view adaptive graph convolutions for graph classification," in *ECCV*. Springer, 2020, pp. 398–414.
[45] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," *NIPS*, vol. 22, pp. 585–593, 2009.
[46] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," *NIPS*, vol. 24, pp. 1413–1421, 2011.
[47] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *SDM*. SIAM, 2013, pp. 252–260.
[48] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in *AAAI*. AAAI Press, 2014, pp. 2149–2155.
[49] F. Nie, J. Li, X. Li *et al.*, "Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification," in *IJCAI*, 2016, pp. 1881–1887.
[50] L. Shu and L. J. Latecki, "Integration of single-view graphs with diffusion of tensor product graphs for multi-view spectral clustering," in *ACML*, vol. 45, 2015, pp. 362–377.
[51] T. Zhang, W. Zheng, Z. Cui, and Y. Li, "Tensor graph convolutional neural network," *arXiv*, 2018.

Xusheng Zhao is currently a Ph.D. candidate in the Institute of Information Engineering, Chinese Academy of Sciences and the School of Cyber Security, University of Chinese Academy of Sciences. His research interests include representation learning and reinforcement learning.

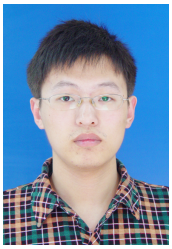
Qiong Dai is currently an Associate Professor in the Institute of Information Engineering, Chinese Academy of Sciences. Her current research interests include data mining, knowledge graph and collaborative computing.

Jia Wu received the Ph.D. degree in computer science from the University of Technology Sydney, Australia. Dr Wu is currently the Research Director for the AI-enabled Processes (AIP) Research Centre and an ARC DECRA Fellow in the School of Computing, Macquarie University, Sydney, Australia. His current research interests include data mining and machine learning. Since 2009, he has published 100+ refereed journal and conference papers, including TPAMI, TKDE, TNNLS, TMM, TKDD, NIPS, WWW, and KDD.

Dr. Wu was the recipient of SDM'18 Best Paper Award in Data Science Track, IJCNN'17 Best Student Paper Award, and ICDM'14 Best Paper Candidate Award. He is the Associate Editor of the ACM Transactions on Knowledge Discovery from Data (TKDD) and Neural Networks (NN).

Hao Peng is currently an Assistant Professor at the School of Cyber Science and Technology, Beihang University. His research interests include representation learning, social network mining and reinforcement learning. To date, Dr Peng has published over 70 research papers in top-tier journals and conferences, including the IEEE TKDE, TPDS, TC, ACM TOIS, TKDD, and Web Conference.

Mingsheng Liu is a Professor in Shijiazhuang Institute of Railway Technology, Shijiazhuang, China. His research interests include urban computing, big data and deep learning.



Xu Bai is an Engineering in the Institute of Information Engineering, Chinese Academy of Sciences. His current research interest include network security and social computing.

Jianlong Tan is a Professor in the Institute of Information Engineering, Chinese Academy of Sciences. His main research directions are network data flow management, algorithm design, massive regular expression matching and image matching algorithms.

Senzhang Wang is a Professor in the School of Computer Science and Engineering, Central South University. His current research interests include data mining, urban computing and social network analysis.

Philip S. Yu is a Distinguished Professor and the Wexler Chair in Information Technology at the Department of Computer Science, University of Illinois at Chicago. Before joining UIC, he was at the IBM Watson Research Center, where he built a world-renowned data mining and database department. He is a Fellow of the ACM and IEEE. Dr. Yu has published more than 1,100 referred conference and journal papers cited more than 142,000 times with an H-index of 175. He has applied for more than 300 patents. Dr. Yu was the Editor-in-Chiefs of ACM Transactions on Knowledge Discovery from Data (2011-2017) and IEEE Transactions on Knowledge and Data Engineering (2001-2004).