Graph Convolutional Networks with Adaptive Neighborhood Awareness

Mingjian Guang, Chungang Yan, Yuhua Xu, Junli Wang, and Changjun Jiang

Abstract—Graph convolutional networks (GCNs) can quickly and accurately learn graph representations and have shown powerful performance in many graph learning domains. Despite their effectiveness, neighborhood awareness remains essential and challenging for GCNs. Existing methods usually perform neighborhood-aware steps only from the node or hop level, which leads to a lack of capability to learn the neighborhood information of nodes from both global and local perspectives. Moreover, most methods learn the nodes' neighborhood information from a single view, ignoring the importance of multiple views. To address the above issues, we propose a multi-view adaptive neighborhood-aware approach to learn graph representations efficiently. Specifically, we propose three random feature masking variants to perturb some neighbors' information to promote the robustness of graph convolution operators at node-level neighborhood awareness and exploit the attention mechanism to select important neighbors from the hop level adaptively. We also utilize the multi-channel technique and introduce a proposed multi-view loss to perceive neighborhood information from multiple perspectives. Extensive experiments show that our method can better obtain graph representation and has high accuracy.

Index Terms—Adaptive neighborhood awareness, graph convolutional network, graph representation, multiple views

1 Introduction

Graph as a data structure is usually used to describe relationships between objects. Many real-life domains, such as chemical molecules [1] and social networks [2], can be represented by graphs. Learning effective knowledge from graphs has attracted extensive attention. Unfortunately, some successful deep learning methods for Euclidean data, such as convolutional neural network (CNN) [3], [4] or recurrent neural network (RNN) [5], cannot learn knowledge directly from graph data. To generalize these methods to graph data, some graph convolutional networks (GCNs) [6], [7], [8], [9] have been proposed and demonstrated superior performance in recent years. GCNs use graph convolutional operators to encode graph information into node representations in Euclidean space and then further transform these representations through some Euclidean space processing methods to apply to different downstream tasks, such as graph classification [10], [11], node classification [12], [13].

The neighborhood-aware process of graph nodes plays an essential role in learning node representations for GCNs [14], [15]. Because GCNs [8] assume that node representations are similar to their neighbors' (homophily assumption [16]) and learn each node's representation based on its neighbors' representations. The node representations will become indistinguishable if some irrelevant or other classes' neighbor information is learned, thereby affecting the model's performance. Therefore, a reasonable neighborhood-aware approach is needed.

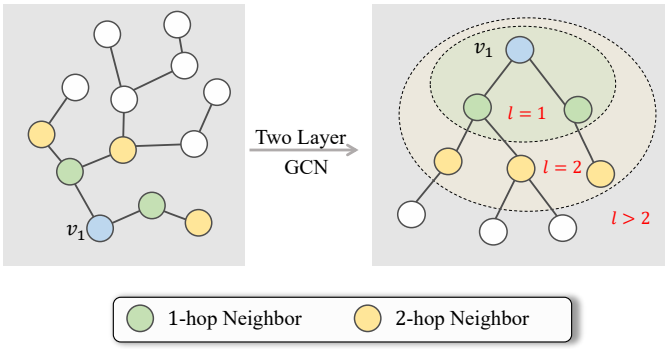


Fig. 1. Illustration of two-layer GCN neighborhood awareness, taking $v1$ as an example. The neighborhood awareness of other nodes is the same as that of $v1$, and l represents the lth graph convolutions.

To better perceive nodes' neighborhoods, the existing graph classification methods can be classified into three main categories. The first category is the greedy neighborhood-aware (GNA) method without distinguishing the importance of neighbor nodes. GCN [8] is a typical representative, and Fig. 1 shows its neighborhood-aware process. After each graph convolution, the neighborhood-aware range of all nodes is increased by one hop, and the same-hop nodes are treated equally. The hyperparameter l controls the farthest-aware hop of all nodes. The second category is the node-level neighborhood-aware (NLNA) approach, which selectively aggregates information from partial neighboring nodes using node sampling techniques [11], [17], [18] or attention mechanisms [19]. The third category is the hop-level neighborhood-aware (HLNA) method, concatenating the outputs of all convolutional layers (hop-level

- *This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4501700, and in part by the Shanghai Science and Technology Innovation Action Plan Project under Grant 22511100700. (Corresponding authors: Changjun Jiang; Junli Wang.)*

- *The authors are with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, and the National (ProvinceMinistry Joint) Collaborative Innovation Center for Financial Network Security, Tongji University, Shanghai 201804, China (e-mail: guangmingjian204@163.com; yanchungang@tongji.edu.cn; xuyuhua@tongji.edu.cn; junliwang@ tongji.edu.cn; cjjiang@tongji.edu.cn).*

representations) and computing their importance implicitly through a multi-layer perception [20], [21]. In summary, GNA is a greedy approach for aggregating neighborhood information. NLNA and HLNA methods concentrate on perceiving neighborhood information from local and global perspectives, respectively.

Although the above approaches are effective, they ignore two critical points. One is the lack of a hybrid neighborhood-aware (HNA) approach with node and hop levels. The hybrid approach can perceive neighborhoods from both global and local aspects. The other is ignoring the importance of perceiving neighborhoods from multiple views. Multi-view (MV) learning [22], [23] has achieved great success in some fields, such as image processing. This learning pattern is also very important for graph data, and the main reasons are as follows. When we observe the same graph from different views, each node's optimal neighborhood-aware range may be different. For example, for the same social network graph, a user has the closest relationship with the users who have been in contact with him within a week, and of course, it can also be within a day. It is easy to notice that, in this example, different views (a day or week) may affect our judgment of essential users [24]. Therefore, the multi-view neighborhood-aware method for GCNs is important and currently required.

This paper proposes a multi-view adaptive neighborhood-aware (MVANA) graph classification method to deal with the above two points. Specifically, for the node-level neighborhood-aware step, we propose three random feature masking (RFM) variants as graph data augmentation and utilize GraphSAGE [17] as the neighborhood aggregation method to encode the information of the augmented graph. The introduction of RFM involves removing features from certain neighboring nodes, which can promote the model to learn the correct node representation using only parts of the neighbors' features and improve the robustness. For the adaptive hop-level neighborhood-aware step, we propose a novel soft-adaptive neighborhood-aware method, which explicitly computes the importance of different hop-level representations and fuses all hop-level representations according to their importance. Combining the above two methods enables our method to perceive neighborhoods from local-global aspects and can also be regarded as a HNA approach. To perceive neighborhoods from multiple views, we employ multi-channel techniques since each channel can observe the same graph from a unique perspective. However, multiple-channel utilization can drop or even degenerate to a single channel when the parameters of multiple channels are similar. To this end, we also propose a multi-view loss to control the degree of diversity of different channels. In the experimental part, our method is applied to multiple graph classification datasets, including six public datasets and four large-scale datasets from a Chinese financial institution. Experiments show that our method achieves the best performance.

TABLE 1
Properties of Neighborhood-Aware Graph Classification Methods

	NLNA	Soft HLNA	Explicit HLNA	MV
GraphSAGE [17]	✓			
DropGIN [18]	✓			
GAT [19]	✓			
SortPool [31]	✓			
HGP-SL [32]	✓			
MAC [33]	✓			
gPool [34]	✓			
JK-Net [20]		✓		
SGC [21]		✓		
SSGC [35]		✓		
AP-GCN [36]			✓	
MVAGC [37]				✓
MVGRL [38]	✓			✓
Our	✓	✓	✓	✓

2 Related Work

CNN is good at dealing with Euclidean data and shows excellent success, but it cannot be directly applied to graphs of non-Euclidean data. To this end, spectral convolutional neural network (Spectral CNN) [25] employs the Fourier transform to transform a graph into spectral space for better recognition and classification of graph signals, which successfully generalizes the convolution operator to graph representation learning. However, Spectral CNN has a high computational complexity problem. Some spectral methods [26], [27] attempt to optimize it. GCN [8] introduces a convolution operator with symmetric Laplacian smoothing, which obtains graph representations faster and more accurately than previous methods. In recent years, GCN and its variants [17], [19] have gradually become the most important class in graph representation learning methods and have been successfully applied to numerous fields [28], [29], [30]. Neighborhood awareness is one of the most critical factors affecting the performance of GCNs [14], [15]. According to the neighborhood-aware pattern, most graph classification methods can be classified into GNA, NLNA, and HLNA methods. Next, we introduce these three categories.

GNA method aggregates all neighbors within a specific hop, and treats these neighbors equally. GCN [8] usually stacks multiple graph convolutional layers, and each graph convolution aggregates the representations of all its first-hop neighbors for each node. GIN [39] is a simple variant of GCN, which enhances the representation ability of the convolutional operator and can be approximately equal to the WL test. GIN is also a GNA approach because it employs the same aggregating method as GCN [8]. DCNN [40] defines a convolutional propagation operator utilizing matrix power series to encapsulate graph diffusion.

NLNA method usually adopts node sampling techniques or attention mechanisms to selectively aggregate information from partial neighboring nodes, treating neighbors differently. GAT [19] scores each neighbor's representation and computes their attention. GraphSAGE [17] randomly samples a fixed number of neighbors for each node. The random disturbance of the neighborhood can improve the robustness of a model. Some sampling-based pooling methods, such as gPool [34], SortPool [31], HGP-SL [32], ASAP [41], and MAC [33], score local neighbors and delete nodes with lower scores to form a coarsened

graph. Each convolution is performed on this new coarsened graph. Since these methods remove some nodes, only the remaining part of nodes participate in neighborhood-aware steps. NLNA mainly focuses on the importance of nodes on local neighborhoods.

HLNA method calculates the neighbors' importance from the hop level to perform neighborhood-aware steps. Euclidean data usually uses Euclidean distance to measure the distance of data, but graphs use hops to define the distance of graph nodes. With a node as the center, its high-hop neighbors are generally considered weakly connected to the node. The outputs of different layers of GCN perceive different hop-level neighborhoods respectively. Some methods, such as JK-Net [20], SGC [21], and SSGC [35], concatenate the outputs of all convolutional layers (hop-level representations) and implicitly calculate the importance of all hop-level representations through multi-layer perception. Liu et al. [42] propose an adaptive residual graph neural network (GNN) to perform HLNA steps and enhance the robustness of GNN towards abnormal node features. Huang et al. [43] replace the skip connections and integrate recursive units into the information aggregation process of GCN to alleviate the aggregation of noisy neighborhood information for node classification tasks. Because these methods learn attention in an end-to-end pattern, they can also be called soft HLNA. AP-GCN [36] adaptively calculates the farthest awareness hop of each node, and explicitly knows how many hop neighbors are perceived for each node. Because this method knows exactly which neighborhoods to remove, it can also be called explicit HLNA. Unlike NLNA, HLNA performs neighborhood-aware steps from a global perspective and equally treats neighbors on the same hop.

Graph data augmentation techniques [44], [45], [46], [47], [48] typically apply various transformations to the original graph, effectively expanding the dataset to alleviate the overfitting problem or serving as a crucial approach in creating diverse views [49] for contrastive learning [50]. Therefore, some graph representation learning methods often combine graph data augmentation methods and showcase their effectiveness. For instance, some methods [18], [51], [52], [53] employ Dropout to remove nodes or features to regularize the network and enhance model robustness [54]. Besides, some approaches employ node or feature masking [55], [56], [57] to augment graph data, yielding effective performance. To enable graph encoders to perceive graph structures better, some studies [55], [58] utilize reconstruction losses for nodes or features to supervise the network's ability to reconstruct masked entities.

Currently, some multi-view methods have achieved great success in other fields. IBRNet [22] uses a multi-layer perception and a ray transformer to learn information on multiple source views to handle novel view synthesis tasks in image processing. DMVST-Net [23] extracts image information from temporal-, spatial-, and semantic-view respectively to obtain representations with multiple-view information. MVAGC [37] employs a multi-view graph convolutional layer to generate views that depict diverse graph structures, achieved through the metric learning technique. GraphDIVE [49] exploits the parameter perturbation strategy to obtain different views of the input and employs multiple experts to perform predictions for each view. This work is insightful and is the first to address the class imbalance issue in graph classification methods, demonstrating exceptional performance. Khan et al. [59] introduce a novel merging subspace representations technique to combine multi-view network data. They employ GNNs to learn node representations on the merged graph and effectively apply this approach to address global poverty issues. MVGRL [38] introduces a graph self-supervised learning technique that acquires node and graph-level representations through contrasting embeddings from two structural perspectives of graphs (immediate neighbors and graph diffusion). MV-HetGNN [60] leverages multi-view representation learning to extensively capture intricate heterogeneity and semantics in the local structure, resulting in versatile and comprehensive node representations for heterogeneous graphs. This approach is insightful and demonstrates superior performance across various heterogeneous graph tasks.

Table 1 summarizes the properties and categories of neighborhood-aware graph classification methods. Different from most methods, our method is a HNA method and can perform neighborhood-aware steps from multiple views.

TABLE 2
Summary of Major Mathematical Notations

Notations	Mathematical Meanings
G	input graph
V, E	node and edge set of G
v_i	ith node of G
$\boldsymbol{X}$	node feature matrix of G
$\boldsymbol{H}$	node representation matrix of G
$\boldsymbol{x}_i$	node feature of v_i
$\boldsymbol{h}_i^l$	node representation of lth layer for v_i
p, d	number of classes and features
n	number of nodes for G
L, C	number of convolutional layers and channels
$\odot$	Hadamard product
$\mathrm{Agg}(\cdot)$	aggregation function

3 THE PROPOSED APPROACH

In this section, we introduce the details of our approach. Section 3.1 introduces some symbol definitions. Section 3.2 illustrates the general framework of our approach. The remaining subsections are important components of our approach.

3.1 Preliminaries

Notation A graph can be represented as $G = (V, E, \boldsymbol{X})$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes, $E \subseteq V \times V$ is a set of edges, $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is a node feature matrix. Each node $v_i \in V$ corresponds to a d-dimensional feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$, where $\boldsymbol{x}_i$ is the ith row of $\boldsymbol{X}$. And $\mathcal{N}(v)$ denotes all first-hop neighbor nodes of v, i.e., $\mathcal{N}(v) = \{u \in V \mid (u, v) \in E\}$.

Given a set of graphs $\mathcal{G} = \{G_1, \ldots, G_m\}$ and a set of labels $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$ for graph classification task, where $\boldsymbol{y}_i \in \mathbb{R}^p$ is a one-hot encoded label vector, and p is the number of classes. Let $\mathcal{Y}_{ij}$ be the probability that the ith graph belongs to the jth class, where $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, p\}$, i.e., the jth element of $\boldsymbol{y}_i$. The goal of the

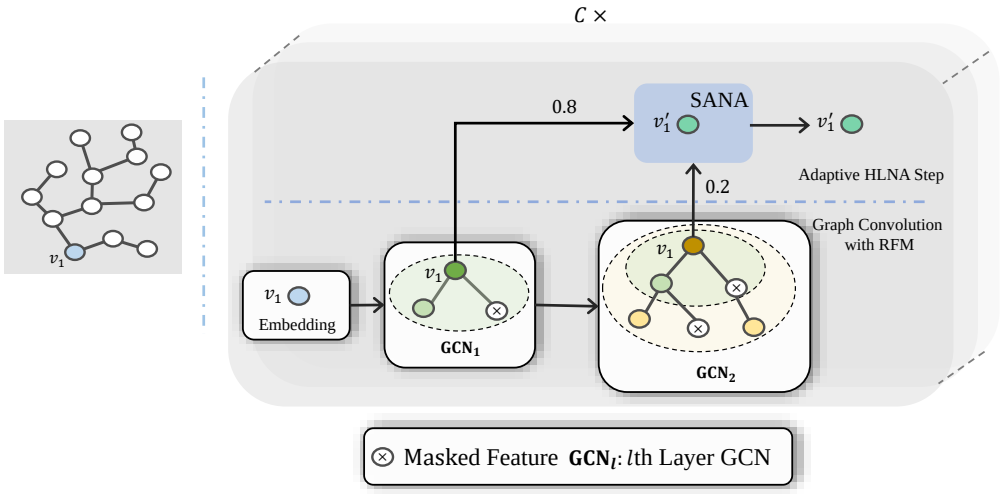


Fig. 2. Schematic figure of neighborhood-aware process of our encoder, taking node v_i as an example. The numbers on the connecting lines represent the attention size.

graph classification task is to find a mapping relationship $f_g : \mathcal{G} \to \mathcal{Y}$.

Graph Convolutional Network GCNs can be used to encode the graph topology and node features of G as a node representation matrix $\boldsymbol{H} \in \mathbb{R}^{n \times d}$ through L graph convolution layers, where d is the hidden layer dimension. The process of graph convolutional can usually be referred to the message passing [61]. The formula for message passing can be summarized in the following form. For the lth graph convolutional layer, the node representation vector of v_i can be obtained as follows:

$$\boldsymbol{h}_i^l = \sigma\left(\boldsymbol{W}_l \cdot \mathrm{Agg}\left(\left\{\boldsymbol{h}_j^{l-1}, \forall v_j \in \mathcal{N}(v_i) \cup v_i\right\}\right)\right) \quad (1)$$

where Agg is an aggregation function (e.g., LSTM, mean, GRU), σ represents an activation function (e.g., Relu [62]), and $\boldsymbol{W}_l$ is a learnable parameter matrix. For convenience, we simplify (1) as follows:

$$\boldsymbol{H}^l = \mathrm{GCN}_l(V, E, \boldsymbol{H}^{l-1}; \boldsymbol{W}_l) \quad (2)$$

where $\boldsymbol{H}^l$ and GCN_l are the node representation matrix and graph convolutional operator of lth layer, respectively.

3.2 The Overall Framework

In this work, we propose MVANA to enable adaptive neighborhood awareness for GCNs. The main steps are as follows. First, we build a multi-channel HNA encoder, which perceives the neighborhood from multiple views and node-hop levels, and encodes a graph in non-Euclidean space into a node representation tensor in Euclidean space. Next, a readout function is utilized to compress graph nodes and obtain a graph representation. Finally, we use the cross-entropy loss to shrink the distance between the predicted and ground-truth values. We also employ a multi-view loss function to control the diversity among multiple views. In the following subsections, we introduce these three critical components by taking graph G as an example. Specifically, Section 3.3 introduces the multi-channel HNA encoder; the readout function and loss are introduced in Sections 3.4 and 3.5, respectively.

3.3 Multi-Channel HNA Encoder

In this subsection, we introduce our proposed multi-channel HNA encoder. Our main goal is to adopt HNA and multi-view learning to improve the encoding ability of GCN. Given a graph G, the proposed graph convolutional encoding process is shown in Fig. 2. Macroscopically, our encoder consists of three modules: graph convolution with RFM, adaptive HLNA step, and multi-channel structure. Next, we introduce these three modules separately.

Graph Convolution with RFM: Some NLNA methods typically employ node sampling techniques to alter the distribution of neighbors and utilize specific graph convolutional operators to learn the representation of new neighborhood distribution, such as [11], [17], [18], [31]. However, the authors in [63] highlight concerns that node sampling may result in the removal of crucial nodes, potentially leading to a decline in performance for graph-level tasks. For instance, removing a carbon atom from a benzene ring would transform the cyclic structure into a chain structure. Therefore, inspired by masked autoencoders (MAE) [55], [64], we opt for utilizing RFM to perturb the neighborhood distribution, as an alternative to node sampling. This approach can be seen as a milder form of node sampling, as it does not disrupt the underlying graph structure. Furthermore, node features represent essential information for nodes, and altering these features essentially changes node information, which impacts the graph convolutional layer's aggregation of neighborhood information.

Specifically, this module uses RFM to perturb the features of G to improve the robustness of our encoder in HLNA. The features of some nodes in G are randomly set to 0, which is RFM. As shown in Fig. 2, each node only perceives part of its neighbors' features. Training in this challenging environment, node representations are obtained with less reliance on all features. Moreover, similar to Dropout [65] to randomly delete neurons, RFM also has the effect of regularizing the network. Next, we formally describe RFM.

Formally, we randomly sample from a binary Bernoulli

distribution with masking ratio q to perform RFM, i.e,

$$\delta_i = \text{Bernoulli}(1-q) \tag{3}$$

where $\delta_i \in \{0,1\}$ is the masked factor for $\boldsymbol{x}_i$. And

$$\boldsymbol{x}_i' = \delta_i \boldsymbol{x}_i \tag{4}$$

where $\boldsymbol{x}_i'$ is the masked feature vector of v_i, and $\boldsymbol{X}'$ is the masked feature matrix of G.

We design three RFM variants, each involving distinct utilization methods, to better exploit the potential of RFM. These variants are as follows: 1) *Training Feature Masking* (TFM): Similar to the common setting of Dropout [65], we apply RFM during training, but not during testing. Excluding it during testing reduces disturbance to the graph data, allowing the model to access more node features and make more accurate predictions. 2) *Simple Feature Masking* (SFM): Inspired by [18], the authors employ Dropout [65] on both training and test sets to ensure that the two sets exhibit similar distributions. SFM follows their setting by applying RFM to both sets. 3) *Mixed Feature Masking* (MFM): Inspired by ensemble learning [66], [67], MFM uses a mixed strategy to enhance diversity across different views. Specifically, different channels employ RFM with distinct masking ratios during both training and testing. MFM can sometimes exhibit the characteristics of both TFM and SFM, for the following reasons. Firstly, MFM applies RFM to both sets, ensuring a consistent distribution between the two sets, reflecting the characteristics of SFM. Secondly, for specific channels where the masking ratio is relatively small or even zero, the model can perceive nearly all features in the test set, reflecting the characteristics of TFM.

With $\boldsymbol{X}'$, we then utilize GCN to encode G, and performs the graph size normalization [68] operation to normalize node representations as follows:

$$\boldsymbol{H}^l = \frac{1}{n}\text{GCN}_l(V, E, \boldsymbol{H}^{l-1}; \boldsymbol{W}_l) \tag{5}$$

where $\boldsymbol{H}^0 = \boldsymbol{X}'$, and we leverage GraphSAGE [17] to implement GCN. We also perform a BN [69] operator as follows:

$$\boldsymbol{H}^l = \frac{\boldsymbol{H}^l - \text{E}[\boldsymbol{H}^l]}{\sqrt{\boldsymbol{\sigma}_{\boldsymbol{H}^l} + \epsilon}} \odot \gamma + \xi \tag{6}$$

where $\boldsymbol{\sigma}$ denotes the variance, ϵ is a constant that prevents the denominator from becoming 0, γ and ξ are two scale- and shift-related learnable parameters. The convergence speed of GCN can both be increased by using these two normalizations.

Adaptive HLNA Step: In this module, we propose a soft-adaptive neighborhood-aware (SANA) method for HLNA. The main idea is to traverse each hop-level representation $\boldsymbol{H}^l$ and calculate a selection factor between 0 and 1 for this representation. A representation with a large selection factor is important. When the selection factor equals 0, its corresponding hop-level representation is discarded. We employ an aggregate representation $\boldsymbol{M}$ recording and fusing all hop-level representations selected by the selection factors.

Specifically, we concatenate $\boldsymbol{M}^l$ and hop-level representation $\boldsymbol{H}^{l+1}$, and transform them to capture the interrelationship between the two, i.e.,

$$\boldsymbol{K} = [\boldsymbol{M}^l, \boldsymbol{H}^{l+1}] \cdot \boldsymbol{W}_k \in \mathbb{R}^{n \times d} \tag{7}$$

where $\boldsymbol{W}_k \in \mathbb{R}^{2d \times d}$ is a learnable parameter matrix, $\boldsymbol{M}^1 = \boldsymbol{H}^1$, and $[\cdot]$ is a concatenation operator. Then, we score $\boldsymbol{H}^{l+1}$ and $\boldsymbol{M}^l$as follows:

$$\boldsymbol{s}_1 = [\boldsymbol{H}^{l+1}, \boldsymbol{K}] \cdot \boldsymbol{W}_z \tag{8}$$

$$\boldsymbol{s}_2 = [\boldsymbol{M}^{l}, \boldsymbol{K}] \cdot \boldsymbol{W}_z \tag{9}$$

where $\boldsymbol{W}_z \in \mathbb{R}^{2d \times 1}$ is a learnable parameter matrix, $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are the importance scores of $\boldsymbol{H}^{l+1}$ and $\boldsymbol{M}^l$, respectively. The selection factors can be obtained as follows:

$$\boldsymbol{z}_i = \frac{\exp(\boldsymbol{s}_i)}{\sum_{j=1}^{2}\exp(\boldsymbol{s}_j)} \in \mathbb{R}^n, \quad i \in \{1,2\} \tag{10}$$

where $\boldsymbol{z}_i$ is the selection factor. With selection factors, we selectively aggregate the hop-level representations of the current layer as follows:

$$\boldsymbol{M}^{l+1} = \boldsymbol{z}_1 \odot \boldsymbol{H}^{l+1} + \boldsymbol{z}_2 \odot \boldsymbol{M}^l. \tag{11}$$

After $L-1$ iterative aggregation, all hop-level representations are soft-adaptive aggregated into $\boldsymbol{M}^L$.

We have completed the exposition of the computational steps of the SANA method. Next, we will provide supplementary explanations regarding the operational principles of the proposed SANA implementation for the HLNA step in a formal way, as well as illustrate why it qualifies as a soft and explicit HLNA method, as indicated in Table 1.

Suppose we have a graph $G = (V, E, \boldsymbol{X})$ with $|V| = 1$, and the optimal step of propagation obtained by adaptive propagation [36] is l ($l \in [1, \ldots, L]$). Because the selection factor $\boldsymbol{z}_i$ follows a binomial Bernoulli distribution, so $\boldsymbol{z}_2 = 1 - \boldsymbol{z}_1$. And according to the above implementation steps, we can recursively derive:

$$\boldsymbol{M}^L = \boldsymbol{z}_1^L \odot \boldsymbol{H}^L + (1-\boldsymbol{z}_1^L) \odot \left(\boldsymbol{z}_1^{L-1} \odot \boldsymbol{H}^{L-1} + \left(1-\boldsymbol{z}_1^{L-1}\right) \odot \ldots\right)$$

where $\boldsymbol{z}_i^l$ denotes the selection factor of lth layer. If $[\boldsymbol{z}_1^1, \ldots, \boldsymbol{z}_1^l] = [1, \ldots, 1]$ and $[\boldsymbol{z}_1^{l+1}, \ldots, \boldsymbol{z}_1^L] = [0, \ldots, 0]$, then the soft adaptive propagation would reduce to adaptive propagation. For the cases where $|V| > 1$, we only need to expand $\boldsymbol{z}_i$ to a vector, which can also make SANA method reduce to adaptive propagation.

In summary, this implementation shares similarities with JK-Net [20], thus qualifying as a soft HLNA method. However, different from JK-Net [20], our distinguishing feature lies in the ability to know the importance and dominance of various hop-level representations through the selection factors' values. Therefore, our SANA method can also be regarded as an explicit HLNA method.

Multi-Channel Structure: The above formulas only use one channel. We also adopt multi-channel structure to perceive the neighborhood from multiple views. Specifically, for each channel c, we calculate the output of cth channel encoder $\boldsymbol{Z}_c$ as follows:

$$\boldsymbol{Z}_c = \alpha \boldsymbol{H}_c^L + (1-\alpha)\boldsymbol{M}_c^L \tag{12}$$

where α is a hyperparameter to weigh the importance of graph convolution with RFM and adaptive HLNA step. And the encoder can degenerate into a network without SANA when $\alpha = 1$. Finally, the aggregated representations on all

channels are concatenated together as the output $\boldsymbol{Z}$ of our encoder, i.e.,

$$\boldsymbol{Z} = [\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_C] \in \mathbb{R}^{C \times n \times d}. \tag{13}$$

In summary, this subsection introduces our encoder, which encodes graph information into node representation tensor $\boldsymbol{Z}$ in Euclidean space. Our encoder adopts three main modules, the first two are used to realize HNA, and the last one is used to multi-view learning.

3.4 Readout Function

In this subsection, our goal is to adopt a readout function to compress node representation tensor $\boldsymbol{Z}$ to a graph representation, i.e., $f_{re} : \boldsymbol{Z} \in \mathbb{R}^{C \times n \times d} \rightarrow \mathbb{R}^d$. Because 2D convolution and pooling operators have demonstrated excellent performance in extracting 3D tensors, we use them to extract and compress $\boldsymbol{Z}$. Unfortunately, the dimension of $\boldsymbol{Z}$ varies for distinct graphs, because graphs have varying numbers of nodes, i.e., n is an unknowable value. Due to the uncertain input dimension, we can not calculate the output dimension of the 2D convolution operator and define the input dimension of the subsequent fully connected layers. Therefore, we utilize our previous work [10], which leverages the attention mechanism [70], [71], to map $\boldsymbol{Z}$ to a fixed-dimension tensor before input into the 2D convolution operator.

Formally, for each channel c, we perform an attention transformation to $\boldsymbol{Z}_c \in \mathbb{R}^{n \times d}$ as follows:

$$\boldsymbol{Z}'_c = \text{Attention}(\boldsymbol{Z}_c \boldsymbol{W}^k, \boldsymbol{Z}_c \boldsymbol{W}^v) \tag{14}$$

where $\boldsymbol{W}^k \in \mathbb{R}^{d \times d'}$ and $\boldsymbol{W}^v \in \mathbb{R}^{d \times d'}$ are learnable parameter matrices, and

$$\text{Attention}(\boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}^{\mathrm{T}} \text{Softmax}(\boldsymbol{K}) \tag{15}$$

So, $\boldsymbol{Z}'_c \in \mathbb{R}^{d' \times d'}$ can be obtained by (14) and (15). We can observe that n is reduced during the transformation, and adding a new variable d' that can be set as a hyperparameter. Then, we can obtain attention representation $\boldsymbol{Z}' \in \mathbb{R}^{C \times d' \times d'}$ by concatenating the output of all channels $\boldsymbol{Z}' = [\boldsymbol{Z}'_1, \ldots, \boldsymbol{Z}'_C]$.

With $\boldsymbol{Z}'$, we continue to compress it by 2D convolution and pooling operators. Let Conv2D and Pool denote the convolution and pooling operators, respectively. The process can be formally written as follows:

$$\boldsymbol{y}' = \text{MLP}\left(\text{Pool}\left(\text{Conv2D}\left(\text{Pool}\left(\text{Conv2D}\left(\boldsymbol{Z}'\right)\right)\right)\right)\right) \tag{16}$$

where MLP denotes multi-layer perception, y' is the output of our readout function and our prediction for G, the convolutional kernel size is 5, and the pooling size is 2.

3.5 Loss

The loss function can direct a network to learn parameters and plays a critical role as the learning target. A good loss function can make the model more general. In this subsection, we introduce our loss function.

With the previous subsections' calculations, we can obtain all prediction values for $\mathcal{G}$. Let $\mathcal{Y}' = \{\boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_m\}$ be the set of predictions for all graphs on the training set obtained by our graph encoder and readout function. We perform the Softmax function to obtain predicted probabilities for different classes, i.e.,

$$\hat{\mathcal{Y}} = \text{Softmax}(\mathcal{Y}'). \tag{17}$$

Then, we can simply use cross-entropy loss to calculate the distance between these predictions and the ground truth labels to optimize the network parameters, i.e.,

$$\mathcal{L}_{\text{cross}} = -\sum_{i=1}^{m}\sum_{j=1}^{p} \mathcal{Y}_{ij} \log(\hat{\mathcal{Y}}_{ij}). \tag{18}$$

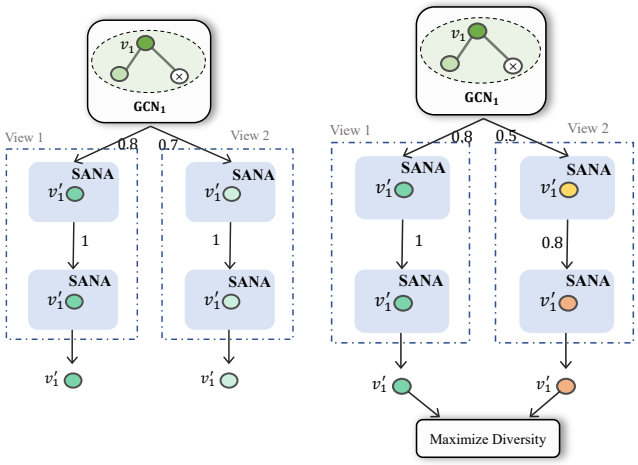
However, this may be suboptimal, since we cannot control the diversity among multiple views. For example, this loss may learn a model as shown in Fig. 3(a), and the neighborhood-aware process of the two views may be very similar. Diversity is a very important factor for ensemble learning [66] and multi-view learning [72] to improve performance.

To alleviate the above problems, we additionally add a multi-view loss to supervise and control the diversity among multiple views, as shown in Fig. 3(b). The loss function is penalized and encourages more diversity in the outputs of multiple channels. Specifically, we define the multi-view loss as follows:

$$\mathcal{L}_{\text{view}} = \frac{1}{C}\sum_{i=1}^{C} \text{Div}^2(\boldsymbol{Z}^i) \tag{19}$$

where $\boldsymbol{Z}^i$ is our encoder's output for G_i, and $\text{Div}(\cdot)$ is a function that measures the diversity among the multi-channel output matrices of $\boldsymbol{Z}^i$. Next, we introduce the implementation and principle of this function.

Because for the same input graph, the outputs of the channels are quite different when the parameters of each channel are inconsistent. Therefore, we assume that the lower the similarity among output matrices, the higher the channel diversity. And we can provide feedback on the diversity of channel parameters by measuring the similarity between channel outputs. The Pearson Correlation Coefficient [73] is an effective way to measure the similarity of



(a) cross-entropy loss (b) multi-view loss (ours)

Fig. 3. Illustration of the multi-view loss. Our loss can encourage diversity among different views. (a) cross-entropy loss. (b) multi-view loss (ours).

Algorithm 1 MVANA with MFM

Require:
The graph dataset, $\mathcal{G} = \{(G_1, \boldsymbol{y}_1), \ldots, (G_m, \boldsymbol{y}_m)\}$;
The number of epochs, T;
The number of channels, C;
Two factors, α, β;
Ensure:
A classifier for graph classification;
for $t = 1$ to T **do**
  for $i = 1$ to m **do**
    for $c = 1$ to C **do**
      Sample a masking ratio q_c;
      Mask $\boldsymbol{X}$ of G_i with masking ratio q_c, and obtain $\boldsymbol{X}'$ by (3), (4);
      Encode G_i with $\boldsymbol{X}'$ and obtain $\boldsymbol{Z}_c$ by (5)–(12);
      $\boldsymbol{Z}'_c = \text{Attention}(\boldsymbol{Z}_c\boldsymbol{W}^k, \boldsymbol{Z}_c\boldsymbol{W}^v)$;
    end for
    $\boldsymbol{Z}^i = [\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_C]$;
    Calculate the diversity of $\boldsymbol{Z}^i$ by (20);
    $\boldsymbol{Z}'^i = [\boldsymbol{Z}'_1, \ldots, \boldsymbol{Z}'_C]$;
    Transform $\boldsymbol{Z}'^i$ by (16) and obtain $\boldsymbol{y}'_i$;
  end for
  $\mathcal{Y}' = \{y'_i \mid i \in [1, \ldots, m]\}$;
  $\hat{\mathcal{Y}} = \text{Softmax}(\mathcal{Y}')$;
  Calculate the multi-view loss $\mathcal{L}_{\text{view}}$ by (19);
  Update the parameters in the network by $\frac{1}{m}\nabla[\beta\mathcal{L}_{\text{view}} - \sum_{i=1}^{m}\sum_{j=1}^{p}\mathcal{Y}_{ij}\log(\hat{\mathcal{Y}}_{ij})]$;
end for

features or matrices, and we use it to implement the function $\text{Div}(\cdot)$, i.e.,

$$\text{Div}(\boldsymbol{Z}^i) = \frac{\text{E}\left[\left(\boldsymbol{Z}_1^i - \boldsymbol{\mu}_{\boldsymbol{Z}_1^i}\right)\cdots\left(\boldsymbol{Z}_C^i - \boldsymbol{\mu}_{\boldsymbol{Z}_C^i}\right)\right]}{\boldsymbol{\sigma}_{\boldsymbol{Z}_1^i}\cdots\boldsymbol{\sigma}_{\boldsymbol{Z}_C^i}} \tag{20}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent the mean and covariance, respectively. $\text{Div}(\cdot) \in [-1, 1]$, the similarity among matrices is the lowest when its value equals 0, and the matrices have the highest similarity when it is equal to -1 or 1. Therefore, $\text{Div}^2(\cdot)$ in (19) can be used to measure the diversity among channels.

Finally, the loss function with cross-entropy and multi-view loss is defined as follows:

$$\mathcal{L} = \mathcal{L}_{\text{cross}} + \beta\mathcal{L}_{\text{view}} \tag{21}$$

where β is the diversity loss learning factor, which can control the degree of diversity. In fact, we can also add a reconstruction loss [55] to recover the masked features. However, considering time efficiency and computational complexity, we do not employ this technique in this work. The training process of MVANA with MFM is shown in Algorithm 1.

3.6 Time Complexity

The encoder part is our main work, so this subsection mainly discusses the time complexity of our encoder. The main sources of time complexity of our encoder are RFM, SANA, and graph convolution, respectively. MFM has the largest time complexity among the three RFMs, because it performs the feature masking C times. Probabilistic sampling can be implemented by the Alias method [74] with $\mathcal{O}(1)$ time complexity. Therefore, the time complexity of MFM is $\mathcal{O}(Cn)$. Next, we introduce the time complexity of SANA. (7)–(9) use matrix multiplication operations and their time complexity is $\mathcal{O}(2nd^2 + 2nd + 2nd) = O(2nd^2 + 4nd)$. The time complexity of Softmax is $\mathcal{O}(n)$. The time complexity of (11) is also $\mathcal{O}(n)$. So the time complexity of SANA is $\mathcal{O}(2nd^2 + 4nd + 2n)$, and the time complexity of SANA with C channel is $\mathcal{O}(2Cnd^2 + 4Cnd + 2Cn)$. The time complexity of GCN's graph convolutional operator is $\mathcal{O}(|E|d^2)$ [8]. Ours is $\mathcal{O}(C|E|d^2)$, since our encoder has C channels.

In summary, the time complexity of our encoder is $\mathcal{O}(Cn+2Cnd^2+4Cnd+2Cn+C|E|d^2)$. In general, $|E| > n$, so the complexity is approximately $\mathcal{O}(C|E|d^2)$.

4 EXPERIMENTS

In this section, we present experiments for MVANA. We begin by providing the statistics of the datasets our used. Then, in Section 4.2, we introduce the experimental setup. To examine our method's performance, we conduct experimental validation in Section 4.3 using publicly available small datasets. We also evaluate the effectiveness of each component through ablation experiments in Section 4.4. Because we introduced three RFM variants in the preceding sections, Section 4.5 undertakes a comprehensive comparative analysis for three variants of RFM and two variants of Dropout. To comprehend the characteristics of each important parameter, we plot variation curves of several hyperparameters and discuss them in Section 4.6. Finally, Sections 4.7 and 4.8 analyze our method's performance in fraud detection and provide visualizations of graph representations to enhance understanding of our model, respectively.

4.1 Datasets

We evaluate MVANA on six public datasets and four large-scale datasets from a Chinese financial institution. The public datasets are PROTEINS [75], DD [75], NCI1 [76], NCI109 [76], Mutagenicity [76], and REDDIT-MULTI-12K [77]. The financial fraud datasets are four months of data from financial institutions named Fraud-Dec, Fraud-Apr, Fraud-May, and Fraud-Jun, respectively. These fraud datasets have two

TABLE 3
Dataset Information

Datasets	Dom	$\lvert\mathcal{G}\rvert$	Avg. $\lvert V\rvert$	Avg. $\lvert E\rvert$	p
PROTEINS	Bio	1 113	39.06	72.82	2
DD	Bio	1 178	284.32	715.66	2
NCI1	Mole	4 110	29.87	32.3	2
NCI109	Mole	4 127	29.68	32.13	2
Mutagenicity	Mole	4 337	30.32	30.77	2
REDDIT-MULTI-12K	SN	11 929	391.41	456.89	11
Fraud-Dec	Fraud	691 661	8.12	32.39	2
Fraud-Apr	Fraud	1 243 035	11.77	48.70	2
Fraud-May	Fraud	1 216 299	11.05	45.26	2
Fraud-Jun	Fraud	1 042 714	9.84	39.73	2

TABLE 4
Performance on Public Datasets. Most Reported Results Come From [33], [55] (OOM: Out of Memory on 32GB Tesla V100)

Baselines	PROTEINS	DD	NCI1	NCI109	Mutagenicity	REDDIT-MULTI-12K
GCN	73.22±3.98	71.82±6.86	69.46±2.81	68.36±1.99	79.92±2.00	40.62±1.87
Set2Set	73.40±4.21	70.98±4.78	72.75±3.11	69.30±4.03	80.13±1.99	40.79±2.29
JK-Net	73.22±3.98	71.82±6.86	69.46±2.81	68.36±1.99	79.92±2.00	40.62±1.87
APGCN	73.94±3.25	78.74±5.60	74.22±4.96	69.58±7.59	80.69±1.50	44.67±1.50
GraphSAGE	72.86±4.66	72.41±4.10	73.02±3.26	72.01±1.69	79.99±2.72	39.26±2.03
GAT	72.24±4.10	71.31±4.89	70.22±2.35	65.66±4.83	78.81±3.53	40.37±1.79
SortPool	74.66±5.08	67.47±6.23	70.58±3.68	68.87±2.38	75.81±2.43	40.83±1.98
DiffPool	73.62±5.44	76.24±5.19	71.03±3.04	70.04±2.80	79.71±2.31	42.36±2.13
EdgePool	73.04±4.64	72.33±4.48	73.94±2.88	72.23±1.87	79.92±1.72	41.52±2.16
gPool	73.14±4.55	75.56±4.74	70.92±3.31	71.51±2.86	75.91±3.60	42.09±2.07
SAGPool	74.84±4.92	77.35±4.31	75.16±2.58	73.69±2.70	78.40±1.76	41.27±2.35
HGP-SL	74.57±3.78	78.61±5.86	75.40±3.90	74.05±2.32	79.76±1.90	42.25±1.97
ASAP	74.66±4.48	77.42±5.12	72.60±2.02	71.29±1.21	77.96±1.65	41.53±2.09
GMT	75.09±5.90	78.72±5.90	74.21±1.88	71.38±2.03	80.26±2.20	42.49±2.18
DropGIN	74.39±5.09	OOM	77.59±1.71	77.08±1.73	80.77±1.53	OOM
GraphMAE	76.30±6.1	77.67±2.40	80.40±0.30	74.33±1.07	78.23±1.72	41.22±1.48
MVAGC	75.64±2.58	78.34±4.29	76.23±2.39	74.97±2.46	80.18±1.65	41.37±2.36
MVGRL	75.12±4.58	OOM	80.17±2.67	77.73±1.32	80.07±1.27	OOM
MAC	76.28±3.55	79.13±4.70	77.62±1.16	75.84±1.86	80.33±1.49	42.67±2.23
ours	**77.26±3.77**	**80.90±3.72**	**81.44±2.30**	**79.06±1.48**	**83.08±1.80**	**49.68±1.52**

classes representing whether the transaction is fraudulent. Table 3 shows the statistics information of our used datasets. The main information includes dataset names (Datasets), the domains (Dom), the number of graphs ($|\mathcal{G}|$), the average number of nodes per graph (Avg. $|V|$), the average number of edges per graph (Avg. $|E|$) and the number of classes (p). The public datasets come from three different domains, including bio-informatics (Bio), chemical molecules (Mole), and social networks (SN). Most of them are binary classification datasets, including a multi-classification dataset REDDIT-MULTI-12K.

4.2 Experimental Setup

To evaluate the performance of our proposed method, we adopt the same experimental setup to compare state-of-the-art baselines fairly and efficiently. Next, we introduce our experimental setup from the three parts of training settings, baselines, and hyperparameter settings.

Training Settings We first introduce the training settings on public datasets. Following the previous work [10], [11], [32], [33], we use the 10-fold cross-validation method and early stopping technique. Specifically, we divide the datasets into training, validation, and test sets. Our used seed for partitioning the dataset is consistent with [33]. The maximum number of training epochs is 300, and the patience is 50. The models' results evaluate on the test set is the one that performs best on the validation set. The reported results are mean and standard deviation of accuracy.

The fraud datasets are large-scale imbalanced datasets, unlike the used public datasets, so the training settings are slightly different. Specifically, as some literature recommended [78], we use AUC as the metric to measure the model's performance. Models need much time to perform training, verification, and testing on large-scale fraud data sets, so we use fixed random partition instead of the 10-fold cross-validation method.

Baselines To effectively evaluate our method, we adopt many state-of-the-art baselines. Some reported results on the public dataset are from a published paper [33], [55]. The two GNA methods are used, including GCN [8] and Seq2Seq [79]. We use ten NLNA methods, including GraphSAGE [17], SortPool [31], DiffPool [80], EdgePool [81], gPool [34], SAGPool [11], HGP-SL [32], MAC [33], ASAP [41], and GMT [82]. We also use two representative HLNA methods, including JK-Net [20] and APGCN [36]. Two methods that utilize graph augmentation techniques to enhance GNNs are included as baselines, namely DropGIN [18] and GraphMAE [55]. Additionally, we compare MVANA with two multi-view methods, including MVAGC [37] and MVGRL [38]. In addition to the above methods, we also select three fraud detection methods, including MLP [83], RNN [84], and LSTM [85].

Hyperparameter Settings In our experiments, we adopt the hyperparameter settings described below. The number of channels is fixed as 4. The hidden layer dimension is 32. The dropout and masking ratios are between 0 and 0.5. The maximum number of graph convolutional layers used is 2 or 3. α is between 0 and 1 with 0.2 steps. We select β from $\{0, 1\text{e}-4, 1\text{e}-3, 1\text{e}-2, 0.1\}$. And we choose learning rate lr from $\{1\text{e}-2, 1\text{e}-3, 5\text{e}-3, 1\text{e}-4, 5\text{e}-4\}$. According to the above experimental setup, we implement our method based on PYG [86] and push the MVANA code and experiment records on GitHub. [1]

4.3 Experimental Results of Public Datasets

Table 4 reports the experimental results on six public datasets. Most of the experimental results are from [33],

1. https://github.com/guangmingjian/MVANA.git

TABLE 5
Ablation Experiments

RFM	MV	SANA	PROTEINS	DD	Mutagenicity
			73.04±3.90	73.18±8.23	80.33±1.68
✓			74.21±5.49	77.42±4.55	80.98±1.69
	✓		74.39±3.40	77.67±2.96	80.91±1.70
		✓	74.12±4.45	77.08±3.02	80.65±2.12
	✓	✓	74.75±4.43	78.35±5.59	81.97±1.73
✓		✓	74.74±3.91	78.26±4.24	81.25±1.83
✓	✓		75.38±3.95	78.47±4.54	81.95±1.54
✓	✓	✓	**77.26±3.77**	**80.90±3.72**	**83.08±1.80**

except for the results of JK-Net [20] and APGCN [36]. Two additional results are obtained through the experimental setup of Section 4.2.

By comparing the baselines, we can draw the following conclusions. A reasonable neighborhood-aware step can improve the performance of GCN effectively, because NLNA and HLNA baselines are almost better than GNA methods. For the NLNA methods, sampling-based methods are better than attention-based methods, for example, GraphSAGE is better than GAT most of the time. APGCN and MAC are the best-performance methods in NLNA and HLNA, respectively. These two excellent methods have their applicable data sets, for example, APGCN performs better on Mutagenicity and REDDIT-MULTI-12K. For DropGIN [18] and GraphMAE [55], the two methods utilize the Dropout [65] and feature masking techniques, respectively. They exhibit performance advantages on certain datasets. However, the computational cost of DropGIN [18] is higher than that of GraphMAE [55]. We can observe that both multi-view baselines, MVAGC [37] and MVGRL [38], achieve impressive performance across all baselines, indicating the effectiveness of multi-view techniques for graph classification tasks. However, MVGRL [38] encounters the "out of memory" issue on some datasets, indicating the challenge of high memory consumption.

By comparing MVANA with baselines, the following conclusions can be obtained. Our method is effective because MVANA achieves the best performance on all datasets. Especially for the dataset REDDIT-MULTI-12K, the improvement is 5.01%. As an HNA method, MVANA can achieve excellent performance on datasets at which HLNA or NLNA are not good. Therefore, the HNA approach, which aims to draw on the strengths of both HLNA and NLNA, is a competitive approach.

4.4 Ablation Experiments

The previous subsection evaluates the overall performance of our model. But the previous results can not indicate whether the proposed components are effective, and which components play a crucial role for our model. In this subsection, we report the results of our ablation experiments to analyze the influence of individual components.

Table 5 reports the results of our ablation experiments. This work introduces three main components, including RFM, Multi-view (MV) learning, and SANA components. To validate the performance of each component, we conduct a series of experiments, including removing all components, individually adding our components to the base model, combining components in pairs, and using all components together. In Table 5, checkmarks indicate the experiments using the corresponding component. It is observable that adding each of our components individually (rows 2-4) results in performance improvements compared with the base model. This indicates the effectiveness of our proposed components. Among the three components, the MV component slightly outperforms the RFM component, as it demonstrates better performance on two datasets. Moreover, combining the proposed components in pairs results in performance improvements compared with using a single component, indicating compatibility and synergy between any two components. Ultimately, when all three components are used, our proposed MVANA achieves optimal performance, highlighting their collaborative relationship rather than conflict.

4.5 RFM Variants Analysis

Section 3.3 introduced three RFM variants. In this subsection, we study their effectiveness. Table 6 reports the performance of the three RFM variants. The results show that SFM, TFM, and MFM achieve the best performance on one, two, and three datasets, respectively. Therefore, these three variants exhibit certain applicability in distinct scenarios and can be considered as hyperparameters that fine-tune the model's performance. Although none of them exhibit the best performance across all datasets, MFM demonstrate a notable superiority. MFM achieves at least the second-best performance on all datasets and demonstrates the highest frequency of achieving the best performance. Consequently, we recommend MFM as the default hyperparameter choice since it consistently performs well in the majority of scenarios. The success of MFM also highlights the effectiveness of the mixed masking ratio in improving the model's performance.

To further validate the effectiveness of the RFM variants, we introduce two additional variants: utilizing Dropout node (DropNode) and Dropout feature (DropFeature) as replacements for the variant MFM of RFM. The experimental results are presented in Table 7. It can be observed that there is a decline in performance when we replace MFM with

TABLE 6
Performance for Three RFM Variants

	PROTEINS	DD	NCI1	NCI109	Mutagenicity	REDDIT-MULTI-12K
SFM	**77.26±3.77**	79.96±2.83	80.15±1.59	77.97±2.42	81.28±2.36	49.04±1.07
TFM	76.28±3.29	80.81±4.15	81.14±1.54	**79.06±1.48**	**83.08±1.80**	48.29±1.39
MFM	76.46±3.55	**80.90±3.72**	**81.44±2.30**	78.75±1.83	82.32±1.65	**49.68±1.52**

TABLE 7
Performance Comparison of RFM and Dropout

	PROTEINS	NCI1	NCI109	Mutagenicity
DropNode	75.20±3.62	78.83±1.42	78.12±2.25	79.18±1.46
DropFeature	75.38±2.91	80.49±1.77	78.36±1.49	81.07±1.79
MFM	**76.46±3.55**	**81.44±2.30**	**78.75±1.83**	**82.32±1.65**

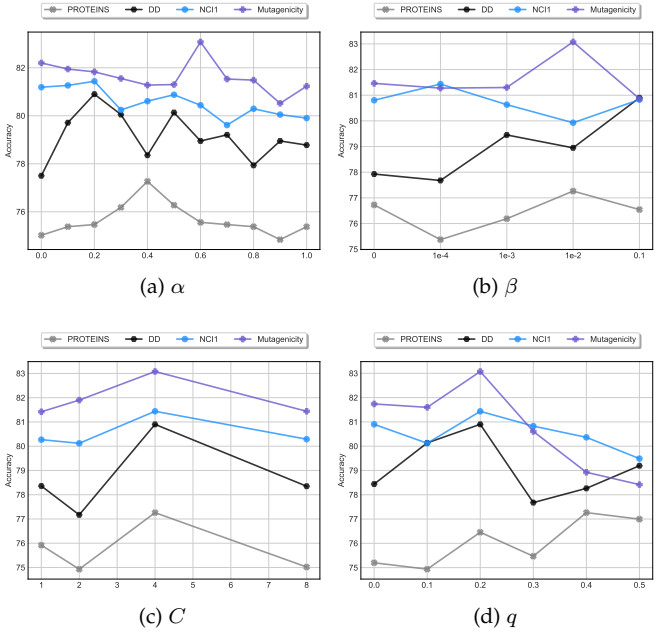
TABLE 8
AUC of Fraud Detection Datasets

	Fraud-Dec	Fraud-Apr	Fraud-May	Fraud-Jun
MLP	50.000	50.000	50.000	50.000
RNN	57.605	71.406	65.163	53.783
LSTM	61.623	75.611	67.258	57.465
GCN	62.953	70.310	77.409	76.442
GraphSAGE	67.132	79.339	79.426	75.345
GAT	63.949	71.149	78.897	75.608
SAGPool	60.941	70.989	79.231	73.726
MAC	63.751	76.808	68.847	72.681
MVANA	**82.242**	**85.025**	**85.577**	**80.089**

Dropout [65] techniques. In particular, the replacement of MFM with DropNode led to the most noticeable decrease in MVANA's performance. This suggests that node sampling might not be the suitable choice for graph classification problems, consistent with the conclusion of [63]. In conclusion, within the context of our method framework, the RFM variant demonstrates a certain degree of competitiveness when compared with Dropout [65] techniques.

4.6 Hyper-Parameters Analysis

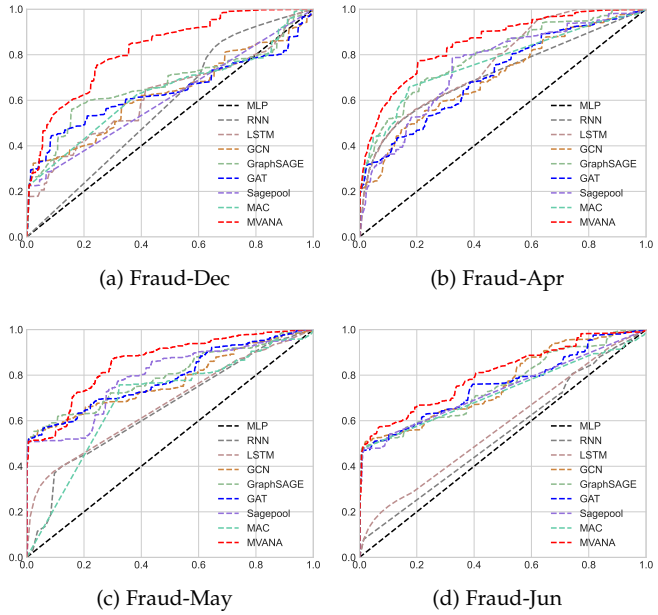
To make the model more scalable, our method introduces some hyper-parameters. In this subsection, we visualize the variation trends of four hyper-parameters to help us understand the characteristics of these hyper-parameters,



(a) α (b) β

(c) C (d) q

Fig. 4. Parameter Sensitivity Analysis. (a) α. (b) β. (c). C. (d). q.



(a) Fraud-Dec (b) Fraud-Apr

(c) Fraud-May (d) Fraud-Jun

Fig. 5. ROC curves for fraud datasets. (a) Fraud-Dec. (b) Fraud-Apr. (c) Fraud-May. (d) Fraud-Jun.

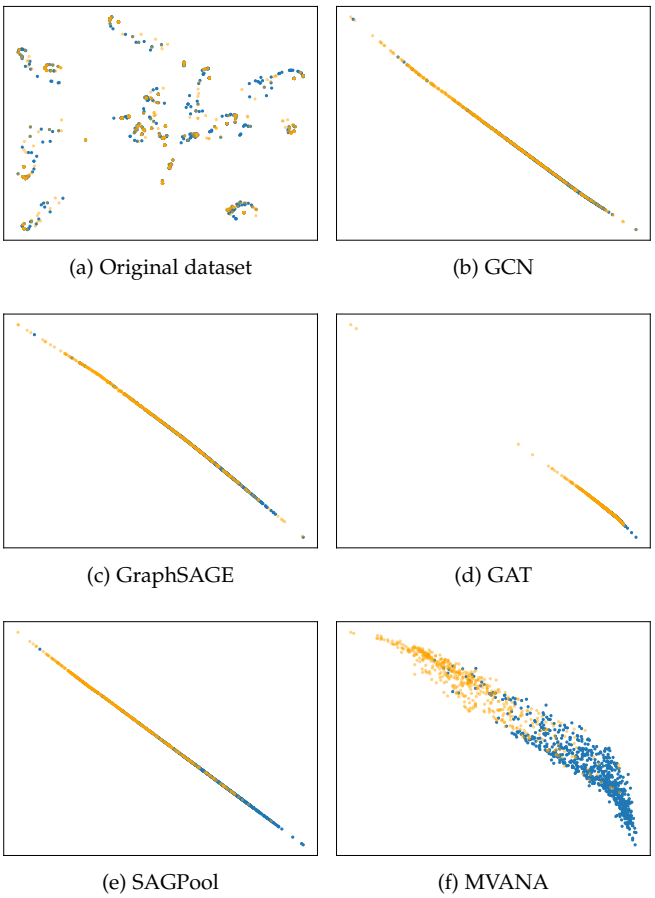
as shown in Fig. 4. We can observe the following conclusions. 1) The curve trend of the same hyper-parameter varies greatly on different data sets. For example, for the hyper-parameter q, the three data sets show a downward trend after $q = 0.2$, but the data set PROTEINS achieve the best value at $q = 0.4$. 2) Our model is sensitive to these parameters since the best, and worst performance gaps between hyper-parameters are between 1% and 4%. 3) The optimal values of hyper-parameters on different datasets are usually different, except C achieves the best performance on all datasets at 4. 4) $\alpha \in [0.2, 0.4], \beta \in \{1e-4, 1e-3, 1e-2, 0.1\}, C = 4$, and $q \in \{0.2, 0.4\}$ are respectively the ranges in which these hyper-parameters achieve the best performance.

4.7 Application to Financial Fraud Detection

In this section, we evaluate our method's performance when MVANA is applied to a financial fraud dataset from a Chinese financial institution, and the experimental results are shown in 6 and Table 7. We can observe that we achieve the best performance on all four datasets. MVANA's ROC curve is the outermost of all models, and we rank first in AUC indicators. In particular, the AUC of our method is 15% higher than the second-ranked method on the dataset Fraud-Dec. Therefore, our method also can be applied to large-scale financial fraud datasets.

4.8 Visualization Analysis

As shown in Fig. 6, we visualize the global graph representation learned by five models on datasets DD to better comprehend our method. To visualize the original dataset in 2D space, we utilize T-SNE dimensionality reduction. We extract the features of the baselines before feeding them to Softmax for visualization. The following are our conclusions. 1) The feature overlap of the original data set is



(a) Original dataset (b) GCN (c) GraphSAGE (d) GAT (e) SAGPool (f) MVANA

Fig. 6. Visualization of graph representations on dataset DD. (a) Original dataset. (b) GCN. (c) GraphSAGE. (d) GAT. (e) SAGPool. (f) MVANA.

very high, and the two classes are difficult to distinguish. It shows that the topological structure is very important for the classification of this data set, and the node features cannot provide enough classification information. 2) These GCNs can be regarded as graph encoders, which can encode the topological structure and node feature information of the graph into linearly separable global graph representations through multiple linear and nonlinear transformations. 3) The class boundary overlap of SAGPool is low, but the graph representations learned by other baselines have high-class overlap. 4) The graph representations learned by our method have a very high degree of separation at class boundaries, indicating that our method can learn graph representations well.

5 Conclusion

This work proposed a novel multi-view adaptive neighborhood-aware graph classification method to improve the ability of GCNs to perceive neighborhoods. Our method can adaptively perceive neighborhoods from both node and hop levels simultaneously, unlike most methods that only learn representations from a single level. Moreover, we proposed a multi-view loss, which can improve the neighborhood-aware diversity among different channels. We performed extensive experiments to evaluate the effectiveness of our method on public and financial institution's fraud detection datasets. The results indicated that our method was effective and could better encode graph information into a representation in Euclidean space.

References

[1] W. Jin, R. Barzilay, and T. S. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 2328–2337.
[2] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2412–2429, Apr. 2023.
[3] G. Huang, Z. Liu, G. Pleiss, L. van der Maaten, and K. Q. Weinberger, "Convolutional networks with dense connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8704–8716, Dec. 2022.
[4] J. R. Clough, N. Byrne, I. Oksuz, V. A. Zimmer, J. A. Schnabel, and A. P. King, "A topological loss function for deep-learning based image segmentation using persistent homology," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 8766–8778, Dec. 2022.
[5] M. O. Turkoglu, S. D'Aronco, J. Wegner, and K. Schindler, "Gating revisited: Deep multi-layer RNNs that can be trained," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4081–4092, 2022.
[6] C. E. Priebe, C. Shen, N. Huang, and T. Chen, "A simple spectral failure mode for graph convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8689–8693, Aug. 2022.
[7] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5103–5113, May 2022.
[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, Apr. 2017, pp. 1–14.
[9] F. Hu, Y. Zhu, S. Wu, W. Huang, L. Wang, and T. Tan, "GraphAIR: Graph representation learning with neighborhood aggregation and interaction," *Pattern Recognit.*, vol. 112, p. 107745, 2021.
[10] M. Guang, C. Yan, Y. Xu, J. Wang, and C. Jiang, "A multichannel convolutional decoding network for graph classification," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, 2023.
[11] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 3734–3743.
[12] A. Kazi, L. Cosmo, S.-A. Ahmadi, N. Navab, and M. Bronstein, "Differentiable graph module (DGM) for graph convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1606–1617, Apr. 2023.
[13] F. Hu, Y. Zhu, S. Wu, L. Wang, and T. Tan, "Hierarchical graph convolutional networks for semi-supervised node classification," in *Proc. Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4532–4539.
[14] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *Proc. Int. Conf. Res. Develop. Inf. Retrieval*, Jul 2020, pp. 1289–1298.
[15] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-aware message-passing GCN for recommendation," in *Proc. WWW*, Apr. 2021, pp. 1296–1305.
[16] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is homophily a necessity for graph neural networks?" in *Proc. Int. Conf. Learn. Representations*, Apr. 2022.
[17] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 1024–1034.
[18] P. A. Papp, K. Martinkus, L. Faber, and R. Wattenhofer, "DropGNN: Random dropouts increase the expressiveness of graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2021, pp. 21 997–22 009.
[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, Apr. 2018, pp. 1–12.
[20] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 5449–5458.
[21] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2019, pp. 6861–6871.

[22] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. AAAI Conf. Artif. Intell*, Feb. 2018, pp. 2588–2595.

[23] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. A. Funkhouser, "IBRNet: Learning multi-view image-based rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 4690–4699.

[24] Q. Wang, Z. Ding, Z. Tao, Q. Gao, and Y. Fu, "Generative partial multi-view clustering with adaptive fusion and cycle consistency," *IEEE Trans. Image Process.*, vol. 30, pp. 1771–1783, Jan. 2021.

[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, Apr. 2014, pp. 1–22.

[26] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, 2019.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2016, pp. 3837–3845.

[28] Z. Shao, Z. Zhang, F. Wang, and Y. Xu, "Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Aug. 2022, pp. 1567–1577.

[29] M. Guang, C. Yan, J. Wang, H. Qi, and C. Jiang, "Benchmark datasets for stochastic petri net learning," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2021, pp. 1–8.

[30] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, and C. S. Jensen, "Decoupled dynamic spatial-temporal graph neural network for traffic forecasting," *VLDB Endow.*, vol. 15, no. 11, pp. 2733–2746, 2022.

[31] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2018, pp. 4438–4445.

[32] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang, "Hierarchical graph pooling with structure learning," in *Proc. AAAI Conf. Artif. Intell.*, Nov. 2019, pp. 1–9.

[33] Y. Xu, J. Wang, M. Guang, C. Yan, and C. Jiang, "Multistructure graph classification method with attention-based pooling," *IEEE Trans. Comput. Social Syst.*, pp. 1–12, 2022.

[34] H. Gao and S. Ji, "Graph U-Nets," in *Proc. Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 2083–2092.

[35] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *Proc. Int. Conf. Learn. Representations*, May 2021.

[36] I. Spinelli, S. Scardapane, and A. Uncini, "Adaptive propagation graph convolutional network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4755–4760, 2021.

[37] N. Adaloglou, N. Vretos, and P. Daras, "Multi-view adaptive graph convolutions for graph classification," in *Proc. Eur. Conf. Comput. Vis.*, vol. 12371, Aug. 2020, pp. 398–414.

[38] K. Hassani and A. H. K. Ahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. Int. Conf. Mach. Learn.*, vol. 119, Jul. 2020, pp. 4116–4126.

[39] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Representations*, May 2019.

[40] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2016.

[41] E. Ranjan, S. Sanyal, and P. P. Talukdar, "ASAP: Adaptive structure aware pooling for learning hierarchical graph representations," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2020, pp. 5470–5477.

[42] X. Liu, J. Ding, W. Jin, H. Xu, Y. Ma, Z. Liu, and J. Tang, "Graph neural networks with adaptive residual," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2021, pp. 9720–9733.

[43] B. Huang and K. M. Carley, "Inductive graph representation learning with recurrent graph neural networks," 2019, arXiv:1904.08035.

[44] K. Ding, Z. Xu, H. Tong, and H. Liu, "Data augmentation for deep graph learning: A survey," *SIGKDD Explor. Newsl.*, vol. 24, no. 2, pp. 61–77, 2022.

[45] M. Zhou and Z. Gong, "GraphSR: A data augmentation algorithm for imbalanced node classification," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2023, pp. 4954–4962.

[46] R. Duan, C. Yan, J. Wang, and C. Jiang, "Path-aware multi-hop graph towards improving graph learning," *Neurocomputing*, vol. 494, pp. 13–22, 2022.

[47] J. Zhang, D. Luo, and H. Wei, "MixupMxplainer: Meneralizing explanations for graph neural networks with data augmentation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Aug. 2023, pp. 3286–3296.

[48] R. Duan, C. Yan, J. Wang, and C. Jiang, "Class-homophilic-based data augmentation for improving graph neural networks," *Knowledge-Based Syst.*, vol. 269, p. 110518, 2023.

[49] F. Hu, L. Wang, Q. Liu, S. Wu, L. Wang, and T. Tan, "GraphDIVE: Graph classification by mixture of diverse experts," in *Proc. Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 2080–2086.

[50] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li, "Self-supervised learning on graphs: Contrastive, generative, or predictive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4216–4235, 2023.

[51] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2020.

[52] J. Shu, B. Xi, Y. Li, F. Wu, C. A. Kamhoua, and J. Ma, "Understanding Dropout for graph neural networks," in *WWW*, Apr. 2022, pp. 1128–1138.

[53] T. Chen, K. Zhou, K. Duan, W. Zheng, P. Wang, X. Hu, and Z. Wang, "Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2769–2781, 2023.

[54] S. Wager, S. Wang, and P. Liang, "Dropout training as adaptive regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2013, pp. 351–359.

[55] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "GraphMAE: Self-supervised masked graph autoencoders," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Aug. 2022, pp. 594–604.

[56] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 9240–9251.

[57] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2412–2429, 2023.

[58] D. Liu, S. Xu, X. Liu, Z. Xu, W. Wei, and P. Zhou, "Spatiotemporal graph neural network based mask reconstruction for video object segmentation," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2021, pp. 2100–2108.

[59] M. R. Khan and J. E. Blumenstock, "Multi-GCN: Graph convolutional networks for multi-view networks, with applications to global poverty," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 606–613.

[60] Z. Shao, Y. Xu, W. Wei, F. Wang, Z. Zhang, and F. Zhu, "Heterogeneous graph neural network with multi-view representation learning," *IEEE Trans. Knowl. Data Eng.*, pp. 1–13, 2022.

[61] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2017, p. 1263–1272.

[62] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 15, Apr. 2011, pp. 315–323.

[63] N. Lee, J. Lee, and C. Park, "Augmentation-free self-supervised learning on graphs," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2022, pp. 7372–7380.

[64] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 15 979–15 988.

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, Jun. 2014.

[66] J. R. Quinlan, "Bagging, Boosting, and C4.5," in *Proc. AAAI Conf. Artif. Intell.*, Aug 1996, pp. 725–730.

[67] M. Guang, C. Yan, G. Liu, J. Wang, and C. Jiang, "A novel neighborhood-weighted sampling method for imbalanced datasets," *Chin. J. Electron.*, vol. 31, no. 5, pp. 969–979, Sep. 2022.

[68] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," 2020, arXiv:2003.00982.

[69] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2015, p. 448–456.

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 5998–6008.
[71] S. Fan, G. Liu, and J. Li, "A heterogeneous graph neural network with attribute enhancement and structure-aware attention," *IEEE Trans. Comput. Social Syst.*, pp. 1–10, 2023.
[72] X. Cao, C. Zhang, H. Fu, S. Liu, and H. Zhang, "Diversity-induced multi-view subspace clustering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 586–594.
[73] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2000, pp. 359–366.
[74] M. D. Vose, "A linear algorithm for generating random numbers with a given distribution," *IEEE Trans. Software Eng.*, vol. 17, no. 9, pp. 972–975, 1991.
[75] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. Mol. Biol.*, vol. 330, no. 4, pp. 771–783, Jul. 2003.
[76] M. Togninalli, M. E. Ghisu, F. Llinares-López, B. Rieck, and K. M. Borgwardt, "Wasserstein weisfeiler-lehman graph kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2019, pp. 6436–6446.
[77] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1365–1374.
[78] C. Jiang, J. Song, G. Liu, L. Zheng, and W. Luan, "Credit card fraud detection: A novel approach using aggregation strategy and feedback mechanism," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3637–3647, 2018.
[79] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in *Proc. Int. Conf. Learn. Representations*, May 2016.
[80] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2018, pp. 4805–4815.
[81] F. Diehl, "Edge contraction pooling for graph neural networks," 2019, arXiv:1905.10990.
[82] J. Baek, M. Kang, and S. J. Hwang, "Accurate learning of graph representations with graph multiset pooling," in *Proc. Int. Conf. Learn. Representations*, May 2021.
[83] R. Patidar, L. Sharma *et al.*, "Credit card fraud detection using neural network," *Int. J. soft Comput. Eng.*, vol. 1, no. 32-38, 2011.
[84] B. Branco, P. Abreu, A. S. Gomes, M. S. Almeida, J. T. Ascensão, and P. Bizarro, "Interleaved sequence RNNs for fraud detection," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 3101–3109.
[85] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen, "Sequence classification for credit-card fraud detection," *Expert Syst. Appl.*, vol. 100, pp. 234–245, 2018.
[86] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," in *Proc. Int. Conf. Learn. Represent. Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 1–9.

Mingjian Guang received his B.S. degree in Software engineering from Mongolian university, Inner Mongolia, China, in 2018. He is currently pursuing the Ph.D. degree from the College of Electronic and Information Engineering, Tongji University.

His research interests include graph neural networks, deep learning, and fraud detection.

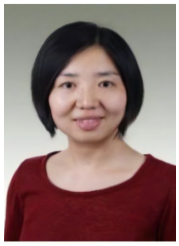
Chungang Yan received the Ph.D. degree from Tongji University, Shanghai, China, in 2006.

She is currently a Professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. She has authored or coauthored more than 100 papers in domestic and international academic journals and conference proceedings.

Her current research interests include graph neural networks, deep learning, and fraud detection.

Yuhua Xu received the B.S. degree and M.S. degree from Shandong University of Science and Technology, Qingdao, China, in 2017 and 2020, respectively. She is currently pursuing the Ph.D. degree from the College of Electronic and Information Engineering, Tongji University, Shanghai, China.

Her research interests include graph neural networks and graph representation learning



Junli Wang received the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2007.

She is currently an Associate Researcher at the College of Electronic and Information Engineering, Tongji University. Her research interests include text data analysis, deep learning, and artificial intelligence.

Changjun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Science, Beijing, China, in 1995.

He is currently the leader of the Key Laboratory of Embedded System and Service Computing (Tongji University), Ministry of Education, Shanghai, China. He is an academician of Chinese Academy of Engineering, an IET Fellow and an Honorary Professor with Brunel University London. He has been the recipient of one international prize and seven prizes in the field of science and technology.