MultiNet: Adaptive Multi-Viewed Subgraph Convolutional Networks for Graph Classification

Xinya Qin[1]*, **Lu Bai**[1]†, **Lixin Cui**[2], **Ming Li**[3,4], **Hangyuan Du**[5], **Edwin R. Hancock**[6]
[1]School of Artificial Intelligence, Beijing Normal University, Beijing, China.
[2]School of Information, Central University of Finance and Economics, Beijing, China.
[3]Zhejiang Institute of Optoelectronics, Jinhua, China;
[4]Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, China.
[5]School of Computer and Information Technology, Shanxi University, Taiyuan, China.
[6]Department of Computer Science, University of York, York, United Kingdom.

Abstract

The problem of over-smoothing has emerged as a fundamental issue for Graph Convolutional Networks (GCNs). While existing efforts primarily focus on enhancing the discriminability of node representations for node classification, they tend to overlook the over-smoothing at the graph level, significantly influencing the performance of graph classification. In this paper, we provide an explanation of the graph-level over-smoothing phenomenon, and propose a novel Adaptive Multi-Viewed Subgraph Convolutional Network (MultiNet) to address this challenge. Specifically, the MultiNet introduces a local subgraph convolution module that adaptively divides each input graph into multiple subgraph views. Then a number of subgraph-based view-specific convolution operations are applied to constrain the extent of node information propagation over the original global graph structure, not only mitigating the over-smoothing issue but also generating more discriminative local node representations. Moreover, we develop an alignment-based readout that establishes correspondences between nodes over different graphs, thereby effectively preserving the local node-level structure information and improving the discriminative ability of the resulting graph-level representations. Theoretical analysis and empirical studies show that the MultiNet mitigates the graph-level over-smoothing and achieves excellent performance for graph classification.

1 Introduction

Graph data analysis has become an important research area for deep learning, and is widely employed in various fields, including bioinformatics [36, 17], social networks [21, 22], and recommendation systems [33, 9]. However, traditional neural networks are originally designed for grid-like data, such as images or sequences. They struggle to handle graphs due to the irregular structures and non-Euclidean properties of graph data. To address these limitations, Graph Neural Networks (GNNs) have emerged as powerful tools for learning on graph-structured data.

In recent years, Graph Convolutional Networks (GCNs) have become the most popular architecture in GNN research. The early spectral-based GCNs, such as the Spectral GNN [7], ChebNet [13], and Vanilla GCN [18], interpret graphs as signals and define convolution operations in the spectral domain. However, these methods are computationally expensive and rely on the eigenvalue spectrum of the graph Laplacian matrix, making them less scalable. To improve the efficiency, the spatial-based

*Email: XinyaQin@mail.bnu.edu.cn
†Corresponding Author, Email: bailu@bnu.edu.cn

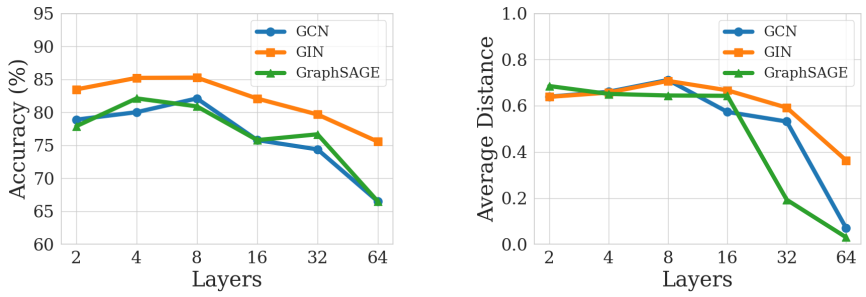
39th Conference on Neural Information Processing Systems (NeurIPS 2025).

GCNs have been introduced, such as the GraphSAGE [16] and GIN [34] that perform convolutions by aggregating the information from neighboring nodes without spectral decomposition. While these spatial GCN approaches are usually computationally efficient, they still suffer from several challenges.

One notable challenge for most existing GCNs is the over-smoothing phenomenon [20, 24, 8], where an increasing number of layers leads to excessive propagation of node features across the global graph structure. Since the information diffuses through multiple hops, the initially discriminative local node-level structural features tend to be indiscriminative, resulting in increasingly similar node representations. This homogenization weakens the ability of GCNs to capture local node differences and ultimately degrades the performance on node classification. To overcome the shortcoming, a variety of mitigation strategies have been developed, including the normalization [25], regularization [40], dynamic connections [26, 27], and residual connections [19, 10].

Nevertheless, most existing studies mainly investigate the convergence of node representations within a single graph structure. The impact of over-smoothing at the graph level has not been sufficiently explored by now. In this paper, we argue that deep GCNs also tend to generate extremely similar representations for different graphs, leading to the graph-level over-smoothing and ultimately influencing the performance of graph classification. This issue is due to the following two key factors. First, when the node information propagates over multiple layers, the resulting node representations tend to be increasingly similar, leading to a significant information loss of local distinct nodes. For graphs having similar node information distributions, this will cause the node representations over all graphs to converge. Second, the unordered nature of graph nodes forces GNNs to employ permutation-invariant global readout functions, such as summation, mean, or max pooling, for generating graph-level representations. However, the simplicity of these operations treats all nodes equally, thereby further ignoring the distinct structural information of local nodes associated with similar node representations. As a result, the graphs with structurally distinct local-level node features may still produce similar graph-level representations, thereby decreasing the discriminative power of graph classification models.

To indicate how the graph-level over-smoothing phenomenon influences the performance of GCNs, Figure 1 reports the graph classification accuracy and the average cosine distance between graph representations (defined in Section 5.3) with varying network depths, associated with three popular GCN models. The results reveal that both the classification accuracies and the representation diversities simultaneously degrade with the deeper network layers (i.e., more than eight layers), and indicate that the models struggle to distinguish between different graph structures. This observation provides empirical evidence that the graph-level over-smoothing arising in GCNs has a negative influence on graph classification.



(a) The graph classification accuracy.

(b) The average distance.

Figure 1: The impact of increasing layers based on the MUTAG dataset.

.

The aim of this paper is to develop a novel **Adaptive Multi-Viewed Subgraph Convolutional Network (MultiNet)**, that can mitigate the influence of the graph-level over-smoothing problem for graph classification tasks. The key innovation lies in a local subgraph convolution module as well as an alignment-based readout mechanism. The former can restrict the excessive message propagation, preserving local distinct node features and alleviating the feature homogenization. The latter maintains the local information and structural information for the resulting graph-level

representations, enhancing the ability to distinguish different graphs. The main contributions of this work are summarized as follows:

- We propose a local subgraph convolution module that can adaptively divide the original input graph into multiple structurally distinct subgraph views. Within each subgraph view, we define an attention-based convolution operation to capture the discriminative local node features. Then we introduce a feature fusion strategy that integrates the complementary information from different subgraph perspectives. This design can effectively preserve the local distinct node information and alleviate the node-level over-smoothing within each individual graph.
- We develop an alignment-based readout mechanism to overcome the limitations of traditional global readout functions in preserving local features and structural information. By establishing consistent cluster-level correspondences across different graphs, our method enables the use of a more expressive readout function during graph aggregation, thereby enhancing the discriminative capability of the learned graph representations. This mechanism effectively preserves local features and structural distinctions extracted from the local subgraph convolution modules within the graph-level representation, thus alleviating the graph-level over-smoothing problem.
- We provide both theoretical and empirical analyses, and demonstrate that the proposed MultiNet can effectively alleviate the graph-level over-smoothing. Experiments on benchmark datasets indicate that the MultiNet can outperform state-of-the-art methods on graph classification tasks.

This paper is organized as follows. Section 2 discusses either the node-level or the graph-level over-smoothing problem. Section 3 details the MultiNet model. Section 4 analyzes the mitigation effect of the MultiNet. Section 5 presents experiments and results. Section 6 concludes this work.

2 The Over-smoothing Problem for GCNs

2.1 The Node-level Over-Smoothing Phenomenon

The over-smoothing phenomenon is a fundamental challenge in GCNs, where repeated message passing causes node representations to become indistinguishable, ultimately impairing performance on both node-level and graph-level tasks. Various works provide different perspectives on the underlying mechanisms of the node-level over-smoothing. For example, [20] models graph convolutional layers as a form of Laplacian smoothing, showing that deep networks drive the node representations toward a space where they become indistinguishable. From a spectral viewpoint, [24] interprets each graph convolution layer as a low-pass filter that attenuates the high-frequency components of the graph signal. While effective for noise reduction, excessive filtering blurs discriminative information, thereby hindering effective classification.

To address this issue, a range of strategies have been introduced. DropEdge [25] applies stochastic edge removal as a form of implicit regularization. PairNorm [40] stabilizes feature diversity by preserving pairwise distances between node embeddings. GraphCON [26] reformulates message passing as a nonlinear dynamical system to enhance representation stability. G^2 [27] introduces a gradient-based gating mechanism that adaptively halts message propagation per node. Residual-based methods like Res-GCN[19] and GCNII [10] incorporate skip connections to retain initial features throughout the network. However, these approaches primarily target at node-level representations and do not restrict the diffusion extent of information. In this work, we revisit the over-smoothing problem from a graph-level perspective and present a method that effectively alleviates this phenomenon.

2.2 The Graph-Level Over-smoothing Phenomenon

While prior studies have primarily focused on the node-level over-smoothing problem, the graph-level over-smoothing phenomenon remains underexplored. Although [31] introduces a strategy to alleviate over-smoothing and applies it to graph classification tasks, it still primarily targets the node-level issue and lacks experiments explicitly addressing graph-level over-smoothing. We argue that excessive convolution across same-domain graphs can lead to indistinguishable graph representations, where the representations of different graphs become excessively similar after the readout stage, making it difficult for the model to differentiate between graphs and thereby impairing graph classification performance. Here we begin with a formal definition, followed by theoretical analysis.

Definition 1 (The Graph-Level Over-Smoothing) Let $\mathcal{G} = \{G_1, G_2, \ldots, G_N\}$ denote a set of undirected and connected graphs, and let $H \in \mathbb{R}^{N\times d}$ be the graph representation matrix after readout operation and $H_i \in \mathbb{R}^d$ be the graph representation of graph G_i. Define a similarity measure $\mu : \mathbb{R}^{N\times d} \to \mathbb{R}_{\geq 0}$ satisfying the following properties:

- a) $\exists c \in \mathbb{R}^d$ with $H_i = c$ for all graphs i $\in \{1, \cdots N\} \Leftrightarrow \mu(H) = 0$, for $H \in \mathbb{R}^{N\times d}$;
- b) $\mu(H + Z) \leq \mu(H) + \mu(Z),$ for all $H, Z \in \mathbb{R}^{N\times d}$.

If the representations of different graphs after multiple graph convolution layers tend to become indistinguishable, i.e., $\mu(H) \to 0$, we say that **the graph-level over-smoothing** has occurred.

Assumption 1 (The Node Feature Distribution Consistency) Given any two graphs $G_i, G_j \in \mathcal{G}$ with initial node feature matrices X_i and X_j, we assume that their feature distributions are statistically consistent, especially in datasets drawn from a shared domain or labeling space. Specifically, the average feature vectors x_i, x_j satisfy $x_i \approx x_j$, indicating that the node features across graphs follow similar distributions.

In graph classification datasets, samples typically originate from similar domains, and node features are represented as sparse one-hot encodings. Hence, it is reasonable to assume that the distributions of node features across different graphs are largely consistent. In subsection 5.3 and Table 2, we provide empirical evidence showing that node features in graph classification datasets from a specific domain exhibit distributional consistency, thereby supporting the validity of Assumption 1.

Theorem 1 Under Assumption 1, excessive message passing leads to the graph-level over-smoothing.

Proof: According to the analysis in [20], the spectral radius of the normalized adjacency matrix $\tilde{A}_i$ is 1, and its eigenvalues $\{\lambda_k\}_{k=1}^{n_i}$ satisfy $\lambda_k \in (-1, 1]$ [18], where n_i is the number of nodes in graph G_i. Therefore, as the number of layers L increases, the terms corresponding to eigenvalues with $|\lambda_k| < 1$ decay exponentially. As $L \to \infty$, only the spectral component corresponding to the dominant eigenvalue $\lambda_k = 1$ remains, leading to the over-smoothing problem. In this case, considering the simplest form of a graph convolution layer, where the activation functions of each layer are linear and the weight matrix is ignored, the node representations $X_i^{(L)}$ after L layer convolutions will eventually converge to the direction related to the average feature vectors, i.e.,

$$\lim_{L\to\infty} X_i^{(L)} = \mathbf{1}_{n_i} x_i^\top, \tag{1}$$

where $\mathbf{1}_{n_i}$ is the all-ones vector. This convergence indicates that the feature differences between nodes are smoothed out, leading to the loss of local-level information. Moreover, common readout functions (such as summation, averaging, or maximizing) treat all nodes in the graph equally, losing the ability to capture the structural information of the graph. For instance, consider the averaging readout used to obtain the graph-level representations:

$$H_i = \frac{1}{n_i} \sum_{k=1}^{n_i} X_{ik}^{(L)}, \tag{2}$$

where k is the node index. Since node features tend to converge, using such a global readout function is equivalent to directly averaging the original features, rendering the convolution operation in GCNs meaningless. According to Assumption 1, $x_i \approx x_j$, we have $H_i \approx H_j$, which leads to $\mu(H) \to 0$. Therefore, the graph representations tend to become indistinguishable, and the graph-level over-smoothing has occurred. □

3 MultiNet: The Adaptive Multi-Viewed Subgraph Convolutional Network

3.1 The Overall Framework

In this paper, we propose a novel MultiNet to alleviate the graph-level over-smoothing problem. As shown in Figure 2, the overall framework of the MultiNet consists of three main components: the Global Graph Convolution, the Local Graph Convolution, and the Alignment-Based Readout. First, the **Global Graph Convolution** layer performs initial feature abstraction. Then, the **Local Subgraph Convolution Module** divides the graph into multiple subgraph views and applies adaptive subgraph convolutions to extract discriminative local features, followed by integrating information across views. Finally, the **Alignment-Based Readout** module aligns nodes across different graphs to generate a more discriminative graph-level representation.

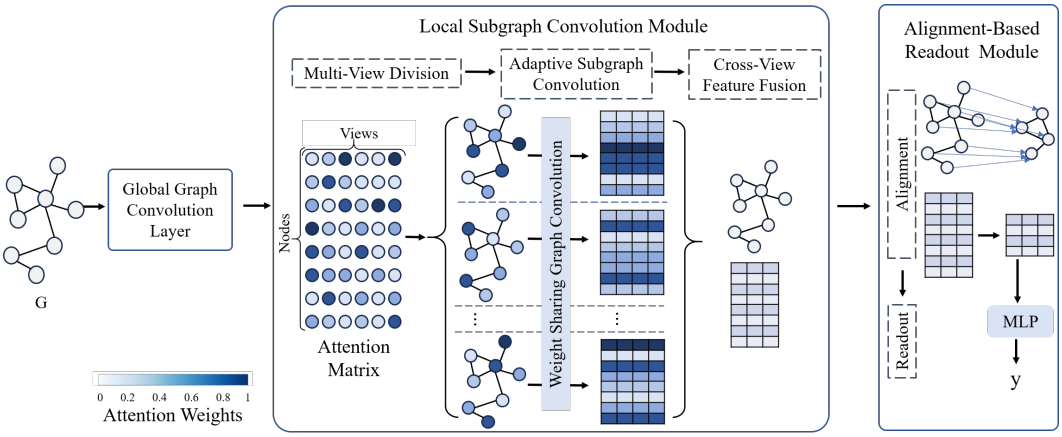


Figure 2: Overall framework of the proposed MultiNet.

3.2 The Global Graph Convolution Operation

Assume $G(A, X)$ denote the input sample graph, where $A \in \mathbb{R}^{n\times n}$ is the adjacency matrix and $X \in \mathbb{R}^{n\times d}$ is the node feature matrix. For attributed graphs, the node features are initialized using one-hot encodings of node labels. For unattributed graphs, we use one-hot encodings of node degrees instead. In our proposed MultiNet, we begin by applying a global graph convolution layer to the input graph $G(A, X)$ to perform initial feature extraction, producing the initial node features $X^{(0)}$. Here we employ the basic convolution operation of the Vanilla GCN defined by [18], i.e.,

$$X^{(0)} = \sigma\left(\tilde{A}XW_G\right). \tag{3}$$

Here W_G is the learnable weight matrix of the graph convolution layer, and $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU). The normalized adjacency matrix $\tilde{A}$ is computed as $\tilde{A} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$, where $\hat{A} = A + I$ includes self-loops and $\hat{D}$ is the degree matrix of $\hat{A}$.

3.3 The Local Subgraph Convolution Module

We define a novel local subgraph convolution module to adaptively capture local structural patterns from the multiple subgraph views. Specifically, the proposed local subgraph convolution module consists of three key components, i.e., the Multi-View Division, the Adaptive Subgraph Convolution, and the Cross-View Feature Fusion.

3.3.1 The Multi-Viewed Subgraph Division

We propose a multi-view division strategy that adaptively assigns the graph into multiple local subgraphs. Specifically, we use a graph convolution layer to learn an attention weight matrix $P \in \mathbb{R}^{n\times m}$, which assigns the n nodes of the original graph to m distinct subgraph views. This process enables diverse local substructure modeling and emphasizes the relative importance of nodes across different perspectives. Formally,

$$P = \text{softmax}(\tilde{A}X^{(0)}W_P), \tag{4}$$

where $W_P \in \mathbb{R}^{d\times m}$ is a learnable weight matrix for view assignment. The softmax function ensures that the attention weights across all views for each node sum to 1.

3.3.2 The Adaptive Subgraph Convolution Operation

Based on the learned attention matrix, we divide the original graph into multiple subgraph views and apply adaptive subgraph convolutions within each view. Unlike the standard aggregation scheme of GCNs, we adopt a constrained message-passing strategy, where the information propagation of each node is regulated by its attention score. While this operation shares similarities with the Graph Attention Network (GAT, [32]), the attention scores here are learned globally, reflecting the

overall importance of each node within this view, rather than computed from local neighborhoods. Specifically, in the l-th layer of subgraph convolution, where $l \in \{1, 2, \ldots, L\}$, let $X_{ij}^{(l)}$ denote the representation of node i in the j-th subgraph view, and P_{ij} be the corresponding attention score. The neighbors of node i are denoted as $\{i_1, i_2, \ldots, i_k\}$, with their representations $\{X_{i_1 j}^{(l)}, X_{i_2 j}^{(l)}, \ldots, X_{i_k j}^{(l)}\}$ and attention scores $\{P_{i_1 j}, P_{i_2 j}, \ldots, P_{i_k j}\}$. The updated representation of node i after the convolution operation is computed as

$$X_{ij}^{(l+1)} = (P_{ij} \times X_{ij}^{(l)} + P_{i_1 j} \times X_{i_1 j}^{(l)} + \cdots + P_{i_k j} \times X_{i_k j}^{(l)}) \cdot W^{(l)}, \tag{5}$$

where $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l+1)}}$ denotes the convolution weight at the l-th layer. The convolution weight is shared across subgraph views. This process can also be formulated in matrix form as

$$X_j^{(l+1)} = \sigma\left((\tilde{A} X_j^{(l)}) \odot P_j W^{(l)}\right), \tag{6}$$

where $\odot$ denotes element-wise multiplication, P_j is the attention weight vector for view j.

3.3.3 The Cross-Viewed Feature Fusion

We fuse features from all subgraph views to obtain comprehensive node representations. Specifically, we concatenate all view-specific node features and process them through a Multi-Layer Perceptron (MLP) to produce a unified node feature matrix $X^{(L)} \in \mathbb{R}^{n \times d'}$ with feature dimension d', i.e.,

$$X^{(L)} = \text{MLP}([X_1^{(L)} \parallel X_2^{(L)} \parallel \cdots \parallel X_m^{(L)}]). \tag{7}$$

3.4 The Alignment-based Readout

Inspired by some works that study node alignment mechanisms in graphs [2, 1, 12], we propose a structure-aware graph readout mechanism based on alignment. This mechanism helps avoid the limitations of simple global readout functions, which tend to neglect the distinct structural information and local differences between nodes, thus alleviating the graph-level over-smoothing phenomenon. By establishing a reliable ordered correspondence for node clusters, this mechanism leverages a more expressive MLP for graph-level readout.

Specifically, we first assign nodes to s clusters. To ensure that clusters are aligned in a consistent order across different samples, we learn a shared cluster assignment matrix S using a unified mapping function for all graphs. Concretely, a graph convolution layer with weight matrix $W_S \in \mathbb{R}^{d' \times s}$ is employed to generate the assignment probability matrix $S \in \mathbb{R}^{n \times s}$, i.e.,

$$S = softmax(\tilde{A} X^{(L)} W_S). \tag{8}$$

Then, we use this matrix to perform alignment for node features, i.e.,

$$\tilde{X} = S^\top X^{(L)}. \tag{9}$$

Since the aligned node features $\tilde{X} \in \mathbb{R}^{s \times d'}$ are now order-consistent across samples, we can apply an MLP to assign weights to the aligned clusters and compute the graph representation $y \in \mathbb{R}^{d'}$, assigning different importance weights to different clusters, i.e.,

$$y = \text{MLP}(\tilde{X}). \tag{10}$$

4 Theoretical Analysis: How the MultiNet Works?

The proposed MultiNet mitigates the graph-level over-smoothing phenomenon from two aspects: First, the local subgraph convolution module limits the extent of node information propagation, reducing the loss of local node-level information and alleviating the node-level over-smoothing problem. Second, the structure-aware readout mechanism aligns nodes across different graphs and employs a more expressive MLP for graph readout, thereby reducing the loss of structural information and mitigating the graph-level over-smoothing phenomenon.

Specifically, in the local subgraph convolution module, the propagation kernel in the j-th view defined in Equation 6 can be equivalently expressed as:

$$\tilde{A}_j = \text{diag}(P_j)\, \tilde{A}\, \text{diag}(P_j) \tag{11}$$

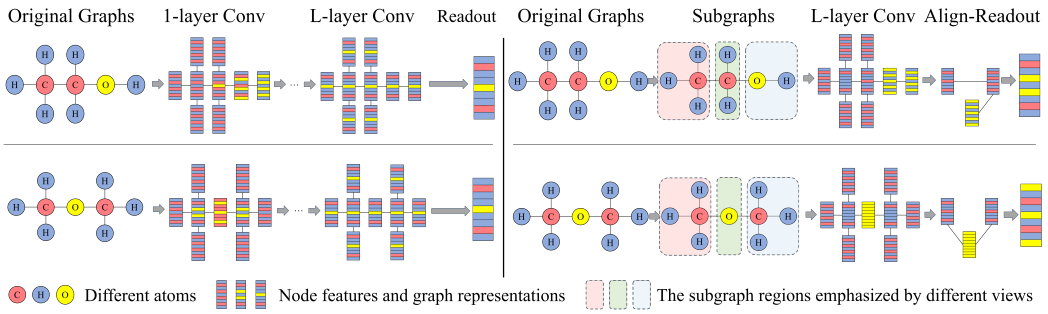


Figure 3: Toy examples to compare the conventional GCNs (left) and our MultiNet (right).

Here, $\mathrm{diag}(P_j)$ is the diagonal matrix of P_j, with values in the range $[0, 1]$. Since P_j is not a vector with all ones, the spectral radius of $\tilde{A}_j$ is smaller than that of $\tilde{A}$, i.e.,

$$\rho(\tilde{A}_j) < \rho(\tilde{A}) = 1, \tag{12}$$

where the spectral radius is denoted by $\rho(\cdot)$. As the number of layers increases, node features no longer converge to the dominant eigenvector direction, but retain components from multiple eigenvector directions. From the perspective of information propagation, this design effectively limits the extent of feature diffusion, allowing the model to focus on specific local regions with higher attention weights, thus mitigating uncontrolled propagation and preserving distinct local node information. While some information may still flow between the low-attention nodes, such propagation is effectively suppressed by the attention weights, which significantly limit its influence. While deeper layers may reduce feature magnitudes, this scaling issue can be addressed by normalization methods such as LayerNorm. Moreover, distinct attention vectors P_j in each view guide the model to focus on different local regions. Fusing these diverse perspectives further enhances both the expressiveness and robustness of the node representations.

Moreover, unlike global readout functions and traditional matrix decomposition or low-rank approximation methods, the alignment-based readout mechanism does not simply compress node features. Instead, it adaptively establishes correspondences between node clusters, enabling the use of a more expressive readout function such as an MLP. This design preserves permutation invariance while maintaining distinct structural information, thereby enhancing the discriminative power of graph representations and effectively mitigating the graph-level over-smoothing problem.

Here we select two representative toy examples of molecular graphs—ethanol and dimethyl ether—to compare the conventional GCNs with our proposed MultiNet model (see Figure 3). As shown on the left, in the traditional GCNs, multi-layer message passing leads to the homogenization of node features, resulting in the loss of crucial local-level information. Subsequent readout operations (e.g., sum or mean pooling) further overlook structural differences between molecules, ultimately rendering the two distinct molecular graphs indistinguishable. In contrast, as shown on the right, our MultiNet leverages a subgraph convolution module that adaptively identifies and focuses on discriminative functional groups, guiding the convolutional process toward structurally distinct local regions and effectively reducing the similarity among node features. Furthermore, by incorporating a structure-aware node alignment strategy and employing a more expressive MLP, our method enhances the discriminative power of the final graph representations.

5 Experiments

5.1 The Experimental Setup

We evaluate the proposed MultiNet model on several benchmark graph classification datasets, including MUTAG, PTC_MR, ENZYMES, PROTEINS, DD, IMDB-B, and IMDB-M. These datasets cover two primary domains: bioinformatics (Bio) and social networks (SN). All experiments are implemented in PyTorch and PyTorch Geometric, and executed on an NVIDIA GeForce RTX 3090 GPU (24GB VRAM). For all datasets, we use a local subgraph convolution module with 3 to 4 layers, set the number of views to 8, the node embedding dimension to 32, and the number of aligned nodes

Table 1: Classification accuracy (In % $\pm$ standard error) on benchmark datasets.

Method	MUTAG	PTC_MR	ENZYMES	PROTEINS	DD	IMDB-B	IMDB-M	Rank
DGCNN	84.0±6.7	58.3±7.0	38.9±5.7	72.9±3.5	76.6±4.3	69.2±3.0	45.6±3.4	12.40
DiffPool	79.8±7.1	60.8±7.0	59.5±5.6	73.7±3.5	75.0±3.5	68.4±3.3	45.6±3.4	10.80
ECC	75.4±6.2	55.7±3.3	29.5±8.2	72.3±3.4	72.69±4.1	67.7±2.8	43.5±3.1	15.60
GIN	84.7±6.7	58.8±5.5	**59.6±4.5**	73.3±4.0	75.3±2.9	71.2±3.9	48.5±3.3	7.40
GraphSAGE	83.6±9.6	60.1±4.7	58.2±6.0	73.0±4.5	72.9±2.0	68.8±4.5	47.6±3.5	10.80
DGK	82.66±1.45	57.32±1.13	53.4±0.9	71.68±0.50	78.50±0.22	66.96±0.56	44.55±0.52	10.57
1-RWNN	89.2±4.3	—	56.7±5.2	74.7±3.3	77.6±4.7	70.8±4.8	47.8±3.8	6.33
2-RWNN	88.1±4.8	—	57.4±4.9	74.1±2.8	76.9±4.6	70.6±4.4	48.8±2.9	7.16
3-RWNN	88.6±4.1	—	57.6±6.3	74.3±3.3	77.4±4.9	70.7±3.9	47.8±3.5	6.50
GKNN-WL	85.73±2.70	59.29±2.54	—	74.94±1.10	—	69.70±2.20	47.87±1.78	7.00
GKNN-GL	85.24±2.28	60.13 ±1.94	—	75.36±1.12	—	69.90±2.20	45.67±1.22	7.40
RWGK	80.77±0.72	55.91±0.37	22.37±0.35	74.20±0.40	71.70±0.47	67.94±0.77	46.72±0.30	13.14
SPGK	83.38±0.31	56.55±0.53	29.00±0.48	75.10±0.50	78.45±0.26	71.26±1.04	51.33±0.57	6.71
GK	81.66±0.11	—	24.87±0.22	71.67±0.55	78.45±0.26	65.87±0.98	45.42±0.87	14.17
WLSK	82.88±0.57	56.05±0.51	52.75±0.44	73.52±0.43	**79.78±0.36**	71.88±0.77	49.50±0.49	7.14
JTQK	85.50±0.55	57.39±0.46	56.41±0.42	72.86±0.41	79.49±0.32	72.45±0.81	50.33±0.49	6.00
ASK	87.50±0.65	—	—	—	70.38±0.22	—	50.12±0.51	9.33
EDBMK	86.35	56.75	36.85	—	78.19	—	—	8.25
QBMK	88.55±0.43	59.38±0.36	—	—	77.60±0.47	—	—	5.00
MultiNet	**89.81±1.46**	**62.65±0.88**	54.83±1.55	**76.40±0.87**	78.90±0.51	**76.49±0.60**	**51.93±0.25**	**2.28**

during readout to 8, with ReLU as the activation function. The model is trained using the Adam optimizer, with hyperparameters such as learning rate and number of epochs tuned via validation. To ensure statistical robustness, we perform ten runs of 10-fold cross-validation and report the mean classification accuracy along with the standard deviation [3].

5.2 Experiments on Graph Classification

We compare the MultiNet with several advanced GNNs and graph kernels. Specifically, **The advanced GNNs include**: five baseline models (the DGCNN [39], DiffPool [38], ECC [30], GIN [34], and GraphSAGE [16]) as well as six additional advanced models (the DGK [37], p-RWNN [23] (with $p = 1, 2, 3$), GKNN-WL, and GKNN-GL [11]). **The graph kernels include**: the RWGK [15], SPGK [6], GK [29], WLSK [28], JTQK [4] (where $q = 2$), ASK [5], EDBMK [35], and QBMK [3]. The classification accuracy and standard error are reported in Table 1, and the last column represents the average rank. For the baseline deep learning methods, we report the results from the fair comparison [14] or reproduce them following its evaluation protocol. For other models, we adapt their best results from their original papers.

The MultiNet achieves the best or near-best classification accuracy on most benchmark datasets, demonstrating the effectiveness of the proposed multi-view adaptive propagation mechanism in enhancing the performance of graph classification. The advantages of the MultiNet are summarized as follows: 1) **Compared with the graph kernel methods**, the MultiNet provides an end-to-end framework that adaptively extracts features from different views, eliminating the need for handcrafted kernel design and feature engineering. 2) **Compared to the traditional GNN-based methods**, our MultiNet restricts the extent of message propagation through node-level attention weights, enabling the model to focus on view-specific critical local features. This effectively suppresses redundant neighbor information and reduces noise. Moreover, MultiNet maintains multiple parallel subgraph convolutions, allowing information to propagate adaptively in different directions. This design captures diverse structural patterns and improves generalization across various types of graph data. 3) **Compared to the models that use simple readout functions (e.g., the GIN)**, the MultiNet introduces an alignment-based readout strategy that considers local node features and the overall graph structure when generating graph representations, thereby further enhancing the expressiveness.

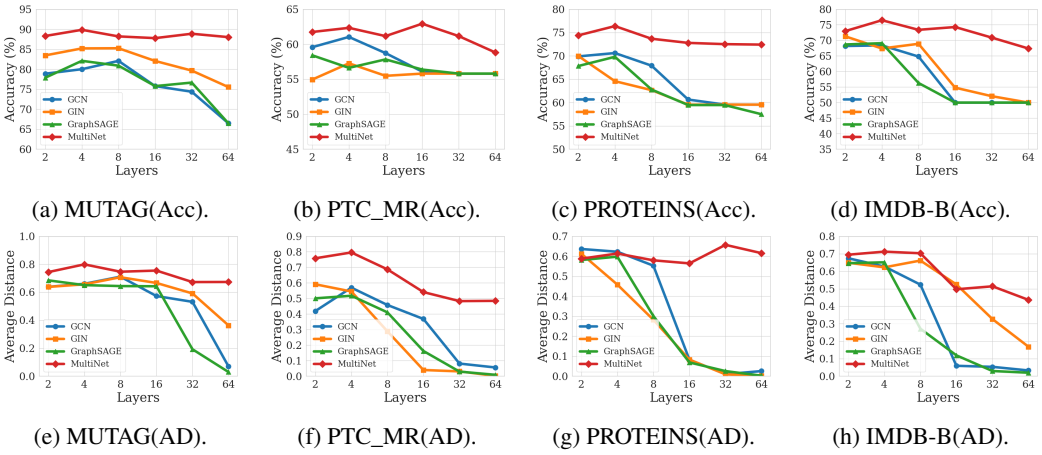
5.3 Evaluations on Mitigating the Over-Smoothing

We conduct experiments to validate the effectiveness of MultiNet in mitigating the graph-level over-smoothing. To quantify the differences between graph representations, we adopt the Average Cosine

[3] Code is available at https://github.com/Xiaoqin0421/MultiNet

Table 2: AD of initial node features for different datasets.

Dataset	MUTAG	PTC_MR	ENZYMES	PROTEINS	DD	IMDB-B	IMDB-M
AD	0.0331	0.1608	0.1315	0.1857	0.1437	0.6320	0.7328



(a) MUTAG(Acc). (b) PTC_MR(Acc). (c) PROTEINS(Acc). (d) IMDB-B(Acc).

(e) MUTAG(AD). (f) PTC_MR(AD). (g) PROTEINS(AD). (h) IMDB-B(AD).

Figure 4: Classification accuracies (%) and the AD values on the four datasets.

Distance (AD). Given the graph-level representation matrix H, AD is defined as:

$$\mu(H) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(1 - \frac{H_i^\top H_j}{\|H_i\| \|H_j\|}\right). \tag{13}$$

Although this metric does not satisfy conditions a) and b) in Definition 1, our primary concern is whether graph representations converge to similar directions. Therefore, we choose a metric that emphasizes directionality rather than vector magnitudes, which is similar to those used in the node-level over-smoothing analysis. A higher AD value indicates greater diversity among graph representations, suggesting that over-smoothing is less severe. To analyze the impact of initial features, we compute the average node features for each dataset and calculate their corresponding AD values (see Table 2). The results show that the AD values for the Bio datasets are below 0.2, supporting Assumption 1 that the initial node features in these datasets are highly similar.

Subsequently, we compare the proposed MultiNet with three standard GCNs (including the Vanilla GCN, GIN, and GraphSAGE) on three Bio datasets (including the MUTAG, PTC_MR, and PROTEINS datasets) and an SN dataset: the IMDB-B dataset. We set the number of convolutional layers to 2, 4, 8, 16, 32, and 64, and report both classification accuracy (%) and the AD values in Figure 4. In the MultiNet, the number of layers refers to the depth of the local subgraph convolution module. The results show that our model already outperforms other baselines under shallow configurations (e.g., L=2 4), demonstrating that the proposed subgraph convolution and the alignment-based readout are inherently effective without requiring deeper networks. As the network depth increases, the MultiNet experiences a smaller drop in accuracy and consistently maintains a higher AD value. In contrast, the AD values of the baseline models decrease significantly, indicating a more severe over-smoothing problem. These findings demonstrate that the MultiNet effectively mitigates the over-smoothing problem on the graph level. Even though Assumption 1 does not hold clearly on the IMDB-B dataset, the graph-level over-smoothing remains evident. This suggests that local node-level features and structural information play a crucial role in graph classification, which are effectively captured by the MultiNet. Notably, under deeper configurations (e.g., L=16 or L=32), the MultiNet achieves competitive or even superior classification accuracy compared to its shallower variants on certain datasets, which further demonstrates its ability to mitigate over-smoothing while preserving the advantages of deeper propagation.

5.4 Ablation Studies

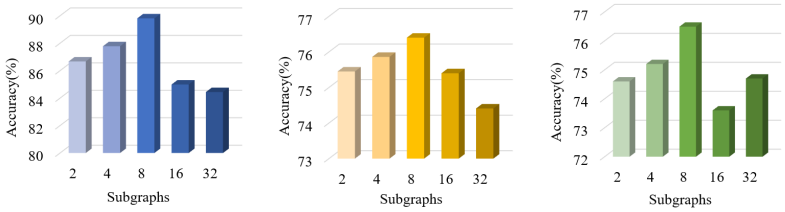
We conduct ablation studies by removing individual modules to evaluate the contribution of each component in the proposed MultiNet. Specifically, we compare the performance of the following three variants: **(1) w/o L**: removing the local subgraph convolution module; **(2) w/o F**: replacing the feature fusion module with simple feature addition; **(3) Avg. Readout**: replacing the alignment module with an average readout function; and **(4) Sum Readout**: replacing the alignment module with a sum readout function. The results in Table 3 indicate that the MultiNet consistently outperforms all ablated variants across benchmark datasets. These findings demonstrate the effectiveness of each component in our framework. The local subgraph convolution module enables adaptive multi-view feature extraction. The feature fusion module effectively integrates information from diverse perspectives. Meanwhile, the alignment module enhances the expressiveness of the graph-level representations. Together, these modules collectively contribute to the superior classification performance.

Table 3: Ablation experimental accuracy (In % $\pm$ standard error).

Method	MUTAG	PTC_MR	ENZYMES	PROTEINS	DD	IMDB-B	IMDB-M
MultiNet	**89.81±1.46**	**62.65±0.88**	**54.83±1.55**	**76.40±0.87**	**78.90±0.51**	**76.49±0.60**	**51.93±0.25**
w/o L	83.39±1.04	58.83±0.54	27.33±1.05	75.01±0.40	78.60±0.45	74.73±0.61	50.61±0.43
w/o F	88.13±2.13	59.51±2.19	50.17±1.20	74.83±0.90	78.58±0.36	72.20±1.06	50.80±1.45
Avg. Readout	86.11±1.97	61.79±1.51	52.33±2.09	74.83±0.42	78.63±0.25	73.52±1.05	50.13±0.32
Sum Readout	85.55±0.96	62.53±0.58	29.17±1.77	73.51±1.42	78.42±0.32	73.50±1.92	47.13±1.43

5.5 Impact of the Number of Subgraphs

We evaluate the impact of the number of subgraphs on model performance by conducting extensive experiments on the MUTAG, PROTEINS, and IMDB-B datasets, varying the number of subgraphs among 2, 4, 8, 16, and 32. The results in Figure 5 show that setting the number to 8 yields the highest accuracy across all three datasets, suggesting that a moderate division effectively captures rich structural information and consistently enhances performance. However, further increasing the number of subgraphs (to 16 and 32) leads to performance degradation, possibly due to the introduction of noise or overfitting from excessive division.



(a) MUTAG. (b) PROTEINS. (c) IMDB-B.

Figure 5: The experimental results of different numbers of subgraphs.

6 Conclusions

In this paper, we propose a novel MultiNet to mitigate the graph-level over-smoothing issue encountered in graph classification tasks. By incorporating a local subgraph convolution module, the MultiNet adaptively divides the input graph into multiple subgraph views. The proposed method extracts more discriminative local features by focusing on view-specific regions with higher attention weights, followed by cross-view node feature fusion to obtain comprehensive node representations. Additionally, we introduce an alignment-based readout mechanism that enhances the quality of the graph representations. Through theoretical analysis and experimental results, we demonstrate that the MultiNet effectively mitigates the graph-level over-smoothing and consistently outperforms existing state-of-the-art methods on graph classification tasks. In future work, we aim to extend our model to explore the over-smoothing problem in node classification tasks and investigate sparsification-based approaches to further reduce its spatial complexity.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62576371, T2122020, 62172370 and 62576198). Ming Li acknowledged the supports from the Jinhua Science and Technology Plan (No. 2023-3-003a).

References

[1] Lu Bai, Lixin Cui, and Edwin R. Hancock. A Hierarchical Transitive-Aligned Graph Kernel for Un-attributed Graphs. In *Proceedings of ICML*, pages 1327–1336, 2022.

[2] Lu Bai, Lixin Cui, Ming Li, Peng Ren, Yue Wang, Lichi Zhang, Philip S. Yu, and Edwin R. Hancock. AEGK: Aligned Entropic Graph Kernels Through Continuous-Time Quantum Walks. *IEEE Trans. Knowl. Data Eng.*, 37(3):1064–1078, 2025.

[3] Lu Bai, Lixin Cui, Ming Li, Yue Wang, and Edwin Hancock. QBMK: Quantum-based Matching Kernels for Un-attributed Graphs. In *Proceedings of ICML*, pages 2364–2374, 2024.

[4] Lu Bai, Luca Rossi, Horst Bunke, and Edwin R. Hancock. Attributed Graph Kernels Using the Jensen-Tsallis q-Differences. In *Proceedings of ECML-PKDD*, pages 99–114, 2014.

[5] Lu Bai, Luca Rossi, Zhihong Zhang, and Edwin R. Hancock. An Aligned Subtree Kernel for Weighted Graphs. In *Proceedings of ICML*, pages 30–39, 2015.

[6] K.M. Borgwardt and H. Kriegel. Shortest-Path Kernels on Graphs. In *Proceedings of ICDM*, pages 74–81, 2005.

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[8] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *Proceedings of AAAI*, pages 3438–3445, 2020.

[9] Huiyuan Chen, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. Graph Neural Transport Networks with Non-Local Attentions for Recommender Systems. In *Proceedings of WWW*, pages 1955–1964, 2022.

[10] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and Deep Graph Convolutional Networks. In *Proceedings of ICML*, pages 1725–1735, 2020.

[11] Luca Cosmo, Giorgia Minello, Alessandro Bicciato, Michael M. Bronstein, Emanuele Rodolà, Luca Rossi, and Andrea Torsello. Graph Kernel Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.*, pages 1–14, 2024.

[12] Lixin Cui, Lu Bai, Xiao Bai, Yue Wang, and Edwin R. Hancock. Learning Aligned Vertex Convolutional Networks for Graph Classification. *IEEE Trans. Neural Networks Learn. Syst.*, 35(4):4423–4437, 2024.

[13] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proceedings of NeurIPS*, pages 3844–3852, 2016.

[14] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A Fair Comparison of Graph Neural Networks for Graph Classification. In *Proceedings of ICLR*, 2020.

[15] Thomas Gartner, Peter A. Flach, and Stefan Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Proceedings of COLT*, pages 129–143, 2003.

[16] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Proceedings of NeurIPS*, pages 1025–1035, 2017.

[17] Bowen Jing, Stephan Eismann, Pratham N. Soni, and Ron O. Dror. Equivariant Graph Neural Networks for 3D Macromolecular Structure. *arXiv preprint arXiv:2106.03843*, 2021.

[18] Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*, 2017.

[19] Guohao Li, Matthias Müller, Guocheng Qian, Itzel C. Delgadillo, Abdulellah Abualshour, Ali Thabet, and Bernard Ghanem. DeepGCNs: Making GCNs Go as Deep as CNNs. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 6923–6939, 2023.

[20] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *Proceedings of AAAI*, 2018.

[21] Julian McAuley and Jure Leskovec. Learning to Discover Social Circles in Ego Networks. In *Proceedings of NeurIPS*, pages 539–547, 2012.

[22] Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. STGSN — a Spatial–Temporal Graph Neural Network framework for Time-Evolving Social Networks. *KBS*, 2021.

[23] Giannis Nikolentzos and Michalis Vazirgiannis. Random Walk Graph Neural Networks. In *Proceedings of NeurIPS*, pages 16211–16222, 2020.

[24] Hoang NT and Takanori Maehara. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters. *arXiv preprint arXiv:1905.09550*, 2019.

[25] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *Proceedings of ICLR*, 2020.

[26] T. Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-Coupled Oscillator Networks. In *Proceedings of ICML*, pages 18888–18909, 2022.

[27] T. Konstantin Rusch, Benjamin Paul Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra. Gradient Gating for Deep Multi-Rate Learning on Graphs. In *Proceedings of ICLR*, 2022.

[28] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.*, pages 2539–2561, 2011.

[29] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of AISTATS*, pages 488–495, 2009.

[30] Martin Simonovsky and Nikos Komodakis. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In *Proceedings of CVPR*, pages 29–38, 2017.

[31] Stevan Stanovica, Benoit Gaüzère, and Luc Brun. Graph Neural Networks with Maximal Independent Set-based Pooling: Mitigating Over-smoothing and Over-squashing. *Pattern Recognit. Lett.*, 187:14–20, 2025.

[32] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *Proceedings of ICLR*, 2018.

[33] Yu Wang, Yuying Zhao, Yi Zhang, and Tyler Derr. Collaboration-Aware Graph Convolutional Networks for Recommendation Systems. In *Proceedings of WWW*, pages 91–101, 2023.

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *Proceedings of ICLR*, 2019.

[35] Lixiang Xu, Lu Bai, Xiaoyi Jiang, Ming Tan, Daoqiang Zhang, and Bin Luo. Deep Rényi Entropy Graph Kernel. *Pattern Recognit.*, 2021.

[36] Yujun Yan, Gao Li, and Danai Koutra. Size Generalizability of Graph Neural Networks on Biological Data: Insights and Practices from the Spectral Perspective. *arXiv preprint arXiv:2305.15611*, 2023.

[37] Pinar Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of KDD*, pages 1365–1374, 2015.

[38] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proceedings of NeurIPS*, page 4805–4815, 2018.

[39] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An End-to-end Deep Learning Architecture for Graph Classification. In *Proceedings of AAAI*, pages 4438–4445, 2019.

[40] Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling Oversmoothing in GNNs. *arXiv preprint arXiv:1909.12223*, 2019.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: This work introduces a novel MultiNet, as highlighted in the abstract and introduction, which aims to improve the performance of graph classification and address the graph-level over-smoothing.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [NA]

   Justification: This paper aims at mitigating the influence of the graph-level over-smoothing problem for graph classification tasks. Therefore, this paper does not refer to any limitations of the proposed work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 2.2 provides rigorous theoretical analysis of the graph-level over-smoothing.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 5.1 details key experimental parameters, and we provide complete code for reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release full code and replication guidelines.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 details key experimental parameters, with additional details provided in the released code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report standard errors, which are commonly used in graph classification experiments to reflect the variability and reliability of the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides sufficient information on the GPU resources (type and memory) used to reproduce the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer:[Yes]

   Justification: This work adheres to NeurIPS ethical guidelines.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This foundational research has no direct societal implications.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No high-risk assets are involved.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: Only publicly available graph classification datasets are used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer:[NA]

    Justification: This study does not involve human participants.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not applicable.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: No LLMs are used in methodology development.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Technical Appendices and Supplementary Material

A.1 Statistics of the Datasets

In Table 4, we summarize the statistics of the benchmark graph classification datasets used in this work, including the maximum (Max #Nodes) and average number of nodes (Mean #Nodes), total number of graphs (#Graphs), number of classes (#Classes), and the domain of each dataset.

Table 4: Statistics of the datasets used in our experiments

Dataset	MUTAG	PTC_MR	ENZYMES	PROTEINS	DD	IMDB-B	IMDB-M
Max #Nodes	28	64	126	620	5748	136	89
Mean #Nodes	17.93	14.29	32.63	39.06	284.3	19.77	13
#Graphs	188	344	600	1113	1178	1000	1500
#Classes	2	2	6	2	2	2	3
Domain	Bio	Bio	Bio	Bio	Bio	SN	SN

A.2 Visualization

To intuitively demonstrate the effectiveness of multi-view subgraph division, we showcase examples from real-world datasets in Figure 6, including MUTAG, PROTEINS, and IMDB-B. Each row corresponds to the same graph instance, while each column shows a specific subgraph view. The intensity of the node color represents the attention weights. The variation in color across different subgraphs suggests that the model adaptively regulates information propagation between subgraphs by capturing features from distinct local regions. This helps preserve local information and mitigates the over-smoothing of node features.

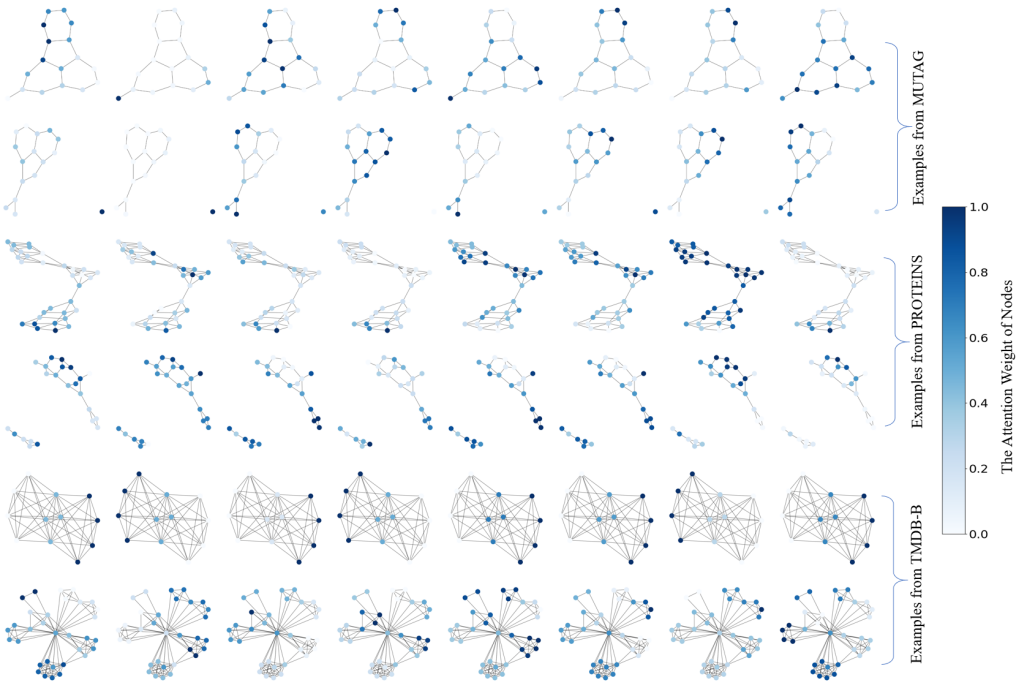


Figure 6: Examples of multi-view subgraph division from different datasets.