Multi-View Graph Pooling via Dominant Sets for Graph Classification

Waqar Ali[a,c], Sebastiano Vascon[a], Thilo Stadelmann[c,b], Marcello Pelillo[a]

[a]*Ca' Foscari University of Venice , Venice, 30170, Italy*
[b]*ECLT European Centre for Living Technology, Venice, 30123, Italy*
[c]*ZHAW Centre for Artificial Intelligence, Technikumstrasse 9 Winterthur, 8401, Switzerland*

Abstract

Graph pooling is a fundamental operation in Graph Neural Networks (GNNs), designed to simplify graphs by reducing the number of nodes and edges while preserving essential structural information for classification tasks. However, most existing pooling methods tend to overlook edge weights and rely on a single-view pooling strategy that focuses either on local or global topological information, failing to capture the full structural context of the graph. To address these limitations, this study introduces a novel Dominant Set Multi-View Pooling (DSMVPool) method featuring two main contributions. First, we propose a dominant-set cluster pooling approach that analyzes the overall graph architecture and connectivity patterns, identifies potential clusters using edge weight information, and generates a coarser graph view. In addition, we create two complementary pooled views by selecting the most representative nodes based on local topology and node features. Second, we design a fusion-view attention layer that integrates the coarser graph structure with the pooled graph views, enabling our method to simultaneously capture and combine global and local structural information and node features. Extensive experiments on four graph classification benchmarks, covering computer vision, chemical, biological, and social networks, demonstrate that DSMVPool achieves superior performance compared to state-of-the-art methods.

Keywords: Graph Neural Network, Graph Pooling, Dominant Set, Graph Classification, Multi-View Integration

1. Introduction

Graph Neural Networks (GNNs) have recently transformed the analysis of graph-structured data through message-passing mechanisms that aggregate information from neighboring nodes [1, 2]. These aggregated features are then used to generate node embeddings for various graph-related tasks, including node classification, graph classification, and link prediction [3, 4]. In the context of graph classification, graph pooling is an essential operation in GNNs for learning the representation of an entire graph. It maps the nodes or subgraphs into a compact representation, highlighting significant graph structures while improving computational efficiency [5, 6].

Early graph pooling methods use sum or average aggregation functions to generate graph-level representations for GNNs [7, 8]. Despite being straightforward, these methods often overlook the contextual and hierarchical information within graphs, which refers to the multi-level structural organization of nodes and edges, ranging from local neighborhoods and subgraphs to higher-level graph abstractions that capture complex interactions and dependencies. To address these limitations, recent studies have introduced hierarchical pooling methods that preserve graph substructures through local and global topological information [9, 10]. These hierarchical methods provide a more comprehensive graph representation by clustering or selecting informative nodes layer by layer [11, 12].

Node cluster pooling methods capture the connectivity patterns of the entire graph by grouping similar nodes into clusters and transferring each cluster into a single node to generate a coarsened graph [13]. Ying et al. [14] developed the first Differentiable Hierarchical Pooling (DiffPool) that learns a soft cluster assignment matrix for nodes using the GNN, which contains the probability values of nodes being assigned to clusters. DiffPool mainly focuses on node features to extract clusters and depends on a predefined cluster ratio. Similarly, the clique pooling method [15] targets global topological structures by identifying all maximal cliques within the graph. Methods [16] and [17] further improve clique pooling to extract the overlapping nodes between two cliques.

In node selection methods, the goal is to create a pooled graph by learning the significance scores of each node and then selecting a subset of nodes with high scores

[18, 9]. For example, Gao et al. [19] proposed a TopkPool method, which employs scalar projection values of node features to select the most important nodes. SAGPool [9] further enhanced the performance of TopkPool using self-attention weights, and MAC [18] used attention weights with a convolutional neural network to evaluate the importance of nodes.

Additionally, MuchPool [20] employs a multi-channel strategy and uses a graph convolution layer for integrating the different views to generate more robust graph representations. However, MuchPool depends on a prior number of clusters and does not consider edge weights in graph representations. Furthermore, Deep Hierarchical Transitive-Aligned Graph Kernels (GK-A) [21] introduces a graph kernel-based approach that incorporates edge weights in graph clustering. However, GK-A primarily focuses on aligning the graph structure and does not fully capture the interactions between node features and topological information. This lack of feature integration limits its effectiveness for tasks requiring both feature-rich and topological context, such as graph classification. Similarly, Graph Pooling Information Bottleneck (GPIB) [22] employs an information-theoretic principle to generate coarsened graphs by emphasizing relevant structural information. While GPIB improves the graph representations, it still does not fully leverage edge weights or multi-view pooling strategies, limiting its ability to integrate comprehensive node and edge information for better classification performance.

However, existing cluster pooling approaches mainly perform analysis on unweighted graphs, overlooking the nuanced dynamics of weighted graphs. In a weighted graph, edge weights quantify the similarity or strength of the connection between nodes. This information is crucial for capturing hierarchical structural patterns and improving performance in downstream classification tasks. Additionally, most traditional clustering-based pooling methods require a predefined cluster ratio to guide the pooling process. Furthermore, the above-mentioned pooling methods often fail to integrate the graph's comprehensive multi-view contextual information, leading to less robust graph representations [23].

To overcome these limitations, we propose DSMVPool, a Dominant Set Multi-View Graph Pooling method that incorporates both global and local topological information,

node features, and edge weights. Unlike MuchPool [20], which fuses features through a simple convolution layer, GK-A [21] and GPIB [22] rely on architectures that do not fully integrate feature-based edge weights or multi-view contextual information, resulting in limited adaptability to weighted graphs. In response, we introduce a node clustering mechanism based on the dominant set concept [24], enabling effective handling of edge-weighted graphs without relying on predefined cluster ratios. This flexibility allows our method to better capture meaningful clusters while generating a more robust graph coarsening representation $G_{coarser}$. We also generate two pooled views of the input graph G_{local} and $G_{feature}$ by extracting the most important nodes based on the graph's local topological information and node features, respectively.

Furthermore, the fusion-view attention mechanism in DSMVPool dynamically integrates different views (i.e., fuse $G_{coarser}$ with G_{local} and $G_{feature}$) of the graph, ensuring that only relevant node relationships are preserved. This attention mechanism improves the representation power of the pooled graph, addressing the drawbacks of previous methods, which often rely on static clustering and simplistic feature fusion strategies. Specifically, our contributions are four-fold:

- We propose a novel Dominant Set Multi-View Graph Pooling method, which simultaneously captures and integrates local topological information, coarser graph structures, and node features.

- For the first time, we use the Dominant Set clustering method to develop graph pooling. This has the advantage of exploiting edge weights (neglected by most of the pooling methods) and avoiding an a-priori fixed number of clusters, resulting in a more expressive graph coarsening.

- Furthermore, we design a fusion-view attention layer that refines the graph representations by integrating the coarser graph with local topological structures and node features-based pooled graphs.

- We conduct extensive experiments showing that DSMVPool enhances average accuracy by 1.10%, 0.62%, 1.03%, and 1.14% in chemical molecules, social networks, bio-informatics, and computer vision-based graph classification bench-

4

marks, respectively, compared to the state-of-the-art pooling approaches.

2. Related Work

Graph pooling methods can be broadly classified into two categories based on their design strategy: global pooling and hierarchical pooling. Hierarchical pooling methods, in turn, are further divided into two subcategories: single-view and multi-view graph pooling.

2.1. Global Pooling Approaches

Global pooling usually adopts summation or average operations to integrate the embeddings of all nodes, resulting in a single vector representation for the entire graph. The DGCNN [25] model first sorts the node embeddings and subsequently generates the graph representation by combining certain node embeddings. Furthermore, SortPooling [25] addresses the challenge of sequentially reading a graph by sorting the graph vertices in a consistent order, allowing traditional neural networks to process graphs directly. Recently, graph topological-based pooling procedures are introduced in [26], where Graclus and graph coarsening techniques are used as pooling modules. Global approaches perform pooling operations based only on node attributes, potentially resulting in the loss of hierarchical information.

2.2. Hierarchical Single View Graph Pooling

Hierarchical graph pooling methods aim to learn multi-level representations of graphs by building hierarchical GNNs. Based on their design, these methods can be broadly categorized into node selection and node clustering approaches [27, 28, 29]. The node selection pooling algorithms calculate the importance of nodes based on their features or the structural information of the graph and retain the nodes with the highest scores. For example, TopkPool [19], SAGPool [9], and AttPool [30] select the most significant nodes based on node attributes or attention scores to form a pooled graph for the next input layer. EdgePool [31] generates a pooled graph by integrating the edges of a given graph. It lacks flexibility, as it can only reduce the number of nodes by half with each iteration.

5

The graph cluster method globally assigns all nodes to a number of clusters but requires setting the number of clusters and then obtaining the basis vector of each cluster. This strategy better ensures the completeness of feature information due to the basis vectors of each cluster containing all node information. In [14], the authors proposed DiffPool to learn a soft assignment matrix using graph neural networks and mapping nodes to a set of clusters. In [15], the authors introduced clique-based graph pooling to capture the overall topological structures of the network effectively. This is achieved by dividing the graph into its possible cliques. Methods [16, 32] enhance the clique pooling technique to retrieve the overlapping nodes shared by two cliques.

Furthermore, the recent study SPGNN [27] extends traditional pooling techniques by using a graph convolution-based mechanism to learn the relative importance of subgraphs, thereby improving performance on heterogeneous graphs. However, it remains limited by its reliance on static, pre-defined graph partitions, which can hinder its adaptability to dynamic graph structures. MaxCutPool-E [33] tackles the MAXCUT problem in graph pooling by selecting a subset of nodes that maximize the cut between partitions. While this approach effectively addresses the challenge of downsampling graphs through maximizing edge cuts between node groups, it lacks differentiability. As a result, it cannot be easily integrated into end-to-end learning frameworks. Additionally, this approach primarily focuses on minimizing node redundancy, which might not always preserve critical graph features necessary for classification tasks.

SpreadEdge [34] improves on the MAXCUT-based pooling by introducing edge contraction techniques. This method focuses on preserving the graph's structural diversity, making it suitable for graphs with diverse edge distributions. However, SpreadEdge's reliance on edge contraction limits its flexibility when applied to graphs with varying edge densities, and its performance is less robust on graphs with heterogeneous or sparse edge weights. Furthermore, recent studies have introduced multi-view graph pooling methods, integrating sparse node selection and node cluster pooling methods to refine the graph structure for the pooled graph (see section 2.3).

Table 1: Comparison of graph pooling methods.

Properties	TopkPool	DiffPool	MAC	GPIB	MuchPool	**Our**
Sparse	✔	✘	✔	✔	✔	✔
Node aggregation	✘	✔	✔	✔	✔	✔
Graph clustering	✘	✔	✘	✔	✔	✔
Flexible number of cluster	✘	✘	✘	✘	✘	✔
Edge weights support	✘	✘	✘	✘	✘	✔

2.3. *Hierarchical Multi-View Graph Pooling*

Hierarchical multi-view graph pooling approaches aim to capture complementary contextual information from multiple graph perspectives, integrating both local and global graph properties as well as node features to form a more robust graph representation.

For example, Jinheon et al. [18] proposed Multistructure Attention Convolutional (MAC) pooling that incorporates multiple strategies to calculate the importance of nodes and uses an attention mechanism to update node representations. Multi-channel Motif-based Pooling (MPool) [35] further improves the node ranking by integrating the local and global graph structure and capturing the higher-order graph structure with motifs. Specifically, MPool contains two channels: the first ranks the nodes using motif adjacency, and the second performs spectral clustering using motif adjacency. It aggregates each channel's results into the final pooled graph. However, these methods do not perform node aggregation and neglect to consider the graph's local topological information during the pooling operation. Recently, Chen et al. [10] developed a novel topological pooling layer that leverages witness complex-based topological embedding mechanisms to integrate local and global topological information.

Additionally, MuchPool [20] combines DiffPool with TopkPool to capture graph local and global topological information and node features. To further improve the graph representation, GK-A [21] introduces Deep Hierarchical Transitive-Aligned Graph Ker-

nels, which aims to preserve both local and global graph structures through hierarchical transitive-aligned embeddings. While GK-A provides strong graph kernel performance by capturing structural correspondence information, its pooling mechanism primarily focuses on node-level matching rather than graph-level features, which limits its flexibility for dynamic graph structures. Moreover, despite its ability to capture structural patterns across graphs, GK-A does not account for edge weights. This limitation is critical in domains such as molecular graphs and social networks, where edge weights represent important relationship strengths. Additionally, GPIB [22] integrates information theory with graph pooling, proposing the Graph Pooling Information Bottleneck to generate task-specific graph representations. This approach utilizes a dynamic generation mechanism to extract node features and edge information, enhancing the sufficiency of the coarsened graph. However, it primarily focuses on minimizing irrelevant subgraph structures, and the noise injection method can cause performance issues when applied to sparse or heterogeneous graphs, where information loss may occur during noise filtering. Furthermore, it doesn't fully leverage the edge-weighted information, which is critical for graphs with varying node connectivity strengths.

However, most current multi-view graph pooling approaches only use a limited amount of contextual information from graphs, disregarding collaboration and input from many sources of information. Additionally, the node cluster pooling methods stated above only focus on unweighted graphs, which hinders their ability to adapt to weighted graphs during the clustering implementation process.

These limitations may constrain the potential use cases for pooling operations. Unlike MuchPool [20], which relies on DiffPool [14] and requires a fixed number of clusters, this study introduces a novel dominant set multi-view pooling method that uses edge weights to extract clusters in weighted graphs without relying on a predefined cluster ratio. This allows for more flexible and robust graph coarsening, making it adaptable to different graph structures. MuchPool [20] fuses features using a simple convolution layer, while MPool [35] employs a linear layer to integrate different contextual information. In contrast, we propose a novel attention-based fusion strategy combined with convolution to dynamically compute node relevance between the pooled and coarser graph structures. To further enhance this mechanism, we introduce a binary

8

relationship matrix that filters out non-significant node connections, ensuring that only contextually relevant pairs contribute to the fused embeddings.

See Table 1, which compares the properties of DSMVPool against other graph pooling operators. As highlighted, our approach uniquely supports all evaluated properties, including sparsity, node aggregation, graph clustering, flexibility in the number of clusters, and edge weight utilization.

3. Proposed Methodology

The proposed DSMVPool mainly uses three views to perform graph pooling operations, learn different contextual information about a graph, and then integrate the results of these three views. The first step of the Figure. 1 employs three graph pooling methods to deeply understand the graph's local and global topological structures and the node features. In this initial step, we generate two pooled views of the graph G_{local} and G_{feature} and one coarser view G_{coarser}. The second step implements the fusion-view attention layer that fuses the G_{coarser} with G_{local} and G_{feature}. The last step aggregates the results of views 1 and 3 to generate the final pooled graph G_{pool}. The following sections provide more detailed explanations of each step.

3.1. Mathematical Notations

This section defines the mathematical notations utilized in this study. Let's consider an arbitrary graph $G = (V, E, X, y)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $X \in \mathbb{R}^{n \times f}$ represents the node feature matrix with n as the number of nodes and f as the feature space's dimension, $E \subseteq V \times V$ the set of edges and y is the label associated to the graph G. The set of edges E is represented through the adjacency matrix $A \in \mathbb{R}^{n \times n}$. With X_i we refer to the features of the i-th node. This work focuses on the graph classification task. It involves a dataset $D = \{(G_1; y_1); (G_2; y_2), \ldots\}$, and aims to train a mapping function $f : G \rightarrow Y$ that assigns each graph to a label, where $G = \{G_1, G_2, \ldots, G_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ represent the set of input graphs and the set of labels, respectively.
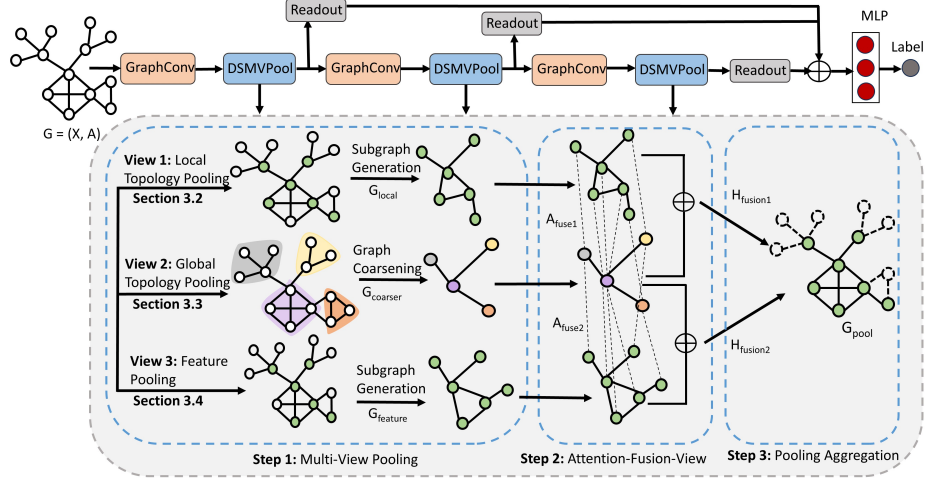
Figure 1: The architecture of the proposed DSMVPool. Step 1 applies three graph pooling methods that capture different contextual information of the graph and generate two pooled views, Glocal and Gfeature, and one coarser view, Gcoarser. Step 2 implements an attention-based fusion-view layer that integrates the three subgraphs—Gcoarser, Glocal, and Gfeature—by computing fusion relationship matrices A_{fuse1} and A_{fuse2} to generate the corresponding fusion feature embeddings, H_{fusion1} and H_{fusion2}. The last step aggregates H_{fusion1} and H_{fusion2} to produce the final pooled graph, Gpool.

3.2. Local Topology Pooling (View 1)

This pooling view learns the graph's structural importance by ranking nodes according to their local neighborhood connectivity. Given a graph $G = (V, E, X)$, we reduce the set of nodes considering the most important ones while preserving the original connectivity, hence generating the pooled graph G_{local}. In this context, the notion of a node's neighborhood can be defined as the explicit connections within a graph or the affinities within node embeddings. To assess the node's importance, we use a graph attention layer (GAT) [36] to capture the local topological structures. Mathematically, the significance of each node is derived through the following formulation:

$$L = \text{attention}(GAT(X, A)); \quad L_{\text{idx}} = \{i \mid L_{\text{i}} > p\} \tag{1}$$

where L_i shows the attention score for a node i, A denotes the adjacency matrix of the graph and X represents the initial node features. In L_{idx} we preserve only those nodes

having an attention score greater than the pooling ratio p. The pooled graph $G_{local} = (V_L, E_L, X_L)$ is then defined as $V_L = \{v_i | i \in L_{idx}\}$ is the set of nodes, $E_L \subseteq V_L \times V_L \cap E$ hence we use the original graph connectivity, and the set of features $X_L = \{X_i | i \in L_{idx}\}$.

3.3. Global Topology Pooling (View 2)

This pooling view aims to generate a coarser graph view $G_{coarser}$ by capturing the graph's global topological information. Current cluster pooling techniques like DiffPool [14] primarily use unweighted graphs to perform pooling operations, neglecting edge weights and typically requiring a predefined cluster ratio to find clusters within a graph. Edge weights play a crucial role in understanding the topological structures within graphs, such as in a molecular graph where edges indicate chemical bonds of different strengths or in a social network graph where edges represent the frequency of interactions among individuals [21]. In these cases, edge weight information is significant for describing the graph's topological structures and can affect how clusters form. Inspired by a dominant set-based clustering method [24] that combines concepts from graph theory and evolutionary game theory to identify clusters in data using edge weights, we adopt this robust concept for graph pooling. Therefore, we design a dominant set cluster pooling method to capture the graph's global topological information and identify all possible dominant set clusters. Following that, we transfer each dominant set cluster into a single node (supernode) and aggregate the feature within each cluster to generate $G_{coarser}$. Next, we provide a detailed explanation of the dominant set notion and describe how we used this concept in graph pooling to generate the graph coarsening.

3.3.1. Dominant Set Clustering:

The Dominant Set (DS) clustering extends the concept of identifying maximal cliques to edge-weighted graphs. In this context, the DS method is used to coarsen the input graph grouping nodes having similar features. The DS clustering is different from other graph coarsening approaches, such as DiffPool and Mincutpool, because it is not dependent on any predefined cluster ratio or a-priori number of clusters, and it works on weighted graphs. Therefore, DS pooling has strong advantages over the competitors, allowing the structures to emerge spontaneously from the graphs, resulting

in more natural, expressive, and representative clusters. We then construct an undirected edge-weighted graph $G = (V, E, w)$ without self-loops, where the nodes V correspond to the graph's nodes, represented by feature vectors. The edges $E \subseteq V \times V$ are the pairwise relations between nodes and their weight function $\omega : E \rightarrow \mathbb{R}_{\geq 0}$ calculates pairwise similarities. The $n \times n$ symmetric adjacency matrix $A = (a_{ij})$ summarizes G:

$$a_{ij} = \begin{cases} w(i, j) & \text{if } (i, j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Typically, every clustering method is expected to exhibit two essential properties: high intra-cluster homogeneity and low inter-cluster homogeneity. These properties are crucial for effectively segregating and grouping objects. They directly influence the combinatorial formulation of DS, as detailed in [24]. Pavan et al. established an intriguing connection between clusters, dominant sets, and local solutions of the following quadratic problem [24]:

$$\begin{aligned} \text{maximize} \quad & \mathbf{h}^T A \mathbf{h} \\ \text{subject to} \quad & \mathbf{h} \in \triangle^n \end{aligned} \tag{3}$$

where A is the similarity matrix of the graph and $\mathbf{h}$ is the so-called *characteristic vector* which lies in the n-dimensional simplex $\triangle^n$, that is, $(\sum_i \mathbf{h}_i = 1, \forall i \, h_i \geq 0)$. If $\mathbf{x}$ is a strict local solution of (3) then its support $\sigma(\mathbf{h}) = \{i \in V | h_i > 0\}$ is a dominant set [37]. In order to extract a DS, a local solution of (3) must be found. A well-known method to solve this problem is to use a result from evolutionary game theory [38] known as *replicator dynamics* (RD) (see Equation. 4).

$$h_i(t + 1) = h_i(t) \frac{(A\mathbf{h}(t))_i}{\mathbf{h}(t)^T A \mathbf{h}(t)} \tag{4}$$

RD is a dynamical system that conducts a selection process on the elements of the vector $\mathbf{h}$. Upon convergence of Equation. 3 $(\| (\mathbf{h}(t) - \mathbf{h}(t + 1)) \|_2 \leq \epsilon)$, ϵ is minimum threshold value and specific elements will emerge $(h_i > 0)$ while others will vanish $(h_i = 0)$. Convergence of the process is assured when the matrix A is non-negative and symmetric. The dynamical system commences at the barycenter of the simplex, and its elements are updated using Equation. 4. At convergence, a dominant set is identified

using the support of **h**, the selected nodes are removed from the graph (referred to as the "peeling-off" strategy), and the process iterates again on the remaining nodes until all nodes are assigned to a cluster.

3.3.2. Dominant Set Pooling:

Given an unweighted graph $G = (V, E, X)$ with associated features X for each node, we first construct a weighted version of it $G = (V, E, X, W)$ and then extract all the clusters using the DS method. Given two nodes i and j, we compute the edge weights using the cosine similarity of node features $X_i \in X$ and $X_j \in X$ and refine the similarities through a Gaussian kernel.

$$s_{ij} = \frac{X_i \cdot X_j}{\|X_i\|\|X_j\|}; \quad W_{ij} = \exp\left(-\frac{(1 - s_{ij})^2}{2\sigma^2}\right) \tag{5}$$

The σ is a scale parameter (after several experiments, we set it as 1.0).

Given the weight W we used the DS's peel-off strategy to extract all dominant set clusters $C = \{c_1, c_2, ..., c_k\}$ from the graph $G = (V, E, X)$ and generate the $G_{\text{coarser}} = (\hat{V}, \hat{E}, \hat{X})$ representation with updated features. Here, $\hat{V} = \{\hat{v}_1, \hat{v}_2, ..., \hat{v}_k\}$ is the set of supernodes representing the k clusters, $\hat{E}$ is the new set of edges connecting supernodes, and $\hat{X}$ are the features associated to each supernodes $\hat{V}$. These sets are defined as follows:

$$\hat{X}_i = \alpha(\{X_v | v \in c_i\}) \tag{6}$$

$$(c_i, c_j) \in \hat{E} \quad \text{iff} \quad \exists v \in c_i \quad \text{and} \quad u \in c_j | (v, u) \in E \tag{7}$$

where α represents a node features aggregation functions (max or average). An edge between supernodes i and j is added to $\hat{E}$ iff an edge already insists between a pair of nodes belonging to the clusters i and j in the un-coarsened graph G.

3.4. Node Feature Pooling (View 3)

Beyond the structural information, graphs often come with node features that can broadly describe their properties. For example, in chemical molecule graphs, node features depict the type of atoms essential for predicting the graph properties. Therefore, it is a valuable source for highlighting the significance of a node inside a graph. So, the main goal of this pooling view is to select the most important nodes based solely on

their feature values. Given a graph $G = (V, E, X)$, we reduce the set of nodes considering the most important ones while preserving the original connectivity, hence generating the pooled graph $G_{feature}$. For this purpose, we directly use a Multi-Layer Perceptron (MLP)[1] to calculate the node importance score directly from node features:

$$F = (MLP(X)); \quad F_{idx} = \{i|\ F_i > p\} \tag{8}$$

where F_i shows the feature score for node i and X represents the initial feature matrix. In *view1* pooling, Equation. 1 already considered the local topology and node features. However, the pooling operations can cause the graph to become sparse and result in isolated nodes (not connected to any other node). This can negatively impact the message passing in the subsequent layers. As a result, the information from these isolated nodes cannot be aggregated by Equation. 1. However, Equation. 8 for node feature learning is independent of structural information, meaning that the presence or absence of edges between nodes does not impact the learning process. Consequently, in a sparse graph, this pooling view layer can effectively learn the information of these isolated nodes. The pooled graph $G_{feature} = (V_F, E_F, X_F)$ is then defined as $V_F = \{v_i | i \in F_{idx}\}$ is the set of nodes, $E_F \subseteq V_F \times V_F \cap E$ hence we use the original graph connectivity, and the set of features $X_F = \{X_i | i \in F_{idx}\}$.

3.5. Fusion-View Attention Convolution

In the first step, illustrated in Figure 1, we generate two pooled views and one coarser view of the input graph G_{local}, $G_{feature}$, and $G_{coarser}$. Next, we need to fuse these three pooled graphs, each reflecting the graph's distinct contextual aspects. To achieve this, we employ an attention-based fusion mechanism inspired by MuchPool [20]. Specifically, we compute attention scores between the node embeddings of the local and feature views with the coarser global view, which enables us to capture the affinity between node pairs across these views. The fusion-view operation can be mathematically expressed as:

$$H_{view} = \sigma\left([H_{view} + A_{fuse} \cdot H_{coarser}] \cdot W\right) \tag{9}$$

---

[1]The MLP is composed of two linear layers of dimensions 64 and 1, respectively, interleaved with a ReLU function.

where $H_{\text{view}} \in \mathbb{R}^{p \times f}$ denotes the node embedding matrix of either G_{local} or G_{feature}, $H_{\text{coarser}} \in \mathbb{R}^{k \times f}$ represents the node embedding matrix of G_{coarser} (see Equation. 6 and 7) and $A_{fuse} \in \mathbb{R}^{p \times k}$ is the fusion relationship matrix that reflects the attention-based connections between the views, where p and k are the node numbers in the two pooled graphs and G_{coarser}, respectively.

The fusion relationship matrix A_{fuse} is computed by concatenating the node embeddings of G_{local}, and G_{coarser}, as well as those of G_{feature} and G_{coarser}, followed by a linear transformation and an activation function to produce raw attention scores. These raw attention scores reflect the importance of each node in the fusion process.

However, to avoid introducing excessive connections between nodes in the two views, which would result in a noisy fusion process, we apply a binary relationship matrix $A_{\text{relation}} \in \{0, 1\}^{p \times k}$, where p and k represent the number of nodes in G_{view} and G_{coarser}, respectively. The binary mask ensures that the attention mechanism only focuses on contextually significant node pairs (i.e., nodes that share meaningful relationships within the graph topology).

Thus, the final fusion relationship matrix A_{fuse} is obtained by element-wise multiplication of the raw attention scores and the binary relationship matrix A_{relation}. This process ensures that the fusion operation is guided by the graph's inherent structural properties, which improves the quality of the pooled graph. The updated feature embeddings are then passed to the next step for further aggregation. The message-passing architecture within GNNs can be expressed as:

$$H^{(t)} = \sigma \left(\left[H^{(t-1)} + A H^{(t-1)} \right] W^{(t)} \right) \in \mathbb{R}^{n \times f} \tag{10}$$

where $H^{(t)}$ denotes the hidden node embeddings computed after t steps of the GNN, $A \in \{0, 1\}^{n \times n}$ denotes the adjacency matrix, and $W^{(t)} \in \mathbb{R}^{n \times f}$ denotes a learnable weight matrix and the node embeddings $H^{(t-1)}$ generated from the previous message-passing step. In our case, we can use A_{fuse} and H_{coarser} as the adjacency matrix and embedding matrix of the fused graph separately in Equation. 10 and generate two new embeddings H_{fusion1} and H_{fusion2}.

15

3.6. Pooling Aggregation Operation

Using the fusion-view layer, we generate two fused pooled graph embeddings from views 1 to 3. The indices of the selected nodes in view 1 are denoted as L_{idx}, and the embedding matrix following the fusion-view operation step is denoted as H_{fusion1}. Similarly, the indices of the selected nodes and the embedding matrix in view 3 are F_{idx} and H_{fusion2}, respectively. Next, we aggregate these fused graph embeddings to minimize the loss for graph classification tasks. We perform the following formulations to aggregate the pooled graphs:

$$\text{Idx} = L_{\text{idx}} \cup F_{\text{idx}} \tag{11}$$

The above equation allows for extracting the induced graph, G_{pool}, from the selected nodes. The adjacency matrix of this pooled graph is calculated as:

$$A_{pool} = A_{\text{Idx, Idx}} \tag{12}$$

where $A_{\text{Idx, Idx}}$ is the row-wise and col-wise indexed adjacency matrix, and $A_{pool} \in \mathbb{R}^{P \times P}$ is the new adjacency matrix of the pooled graph. The node feature matrix for the aggregated pooled graph, X_{pool}, is defined as:

$$X_{\text{pool}}[i,:] = \begin{cases} H_{fusion1}[i,:] & \text{if } i \in L_{\text{idx}} \\ \frac{1}{2}\left(H_{fusion1}[i,:] + H_{fusion2}[i,:]\right) & \text{if } i \in L_{\text{idx}} \cap F_{\text{idx}} \\ H_{fusion2}[i,:] & \text{if } i \in F_{\text{idx}} \end{cases} \tag{13}$$

where $X_{\text{pool}} \in \mathbb{R}^{P \times f}$ is the node feature matrix for the aggregated pooled graph.

3.7. Hierarchical DSMVPool Architecture and Readout Function

We integrate multiple GCN layers with DSMVPool layers to make a hierarchical pooling architecture and perform graph classification tasks. Figure 1 illustrates the hierarchical framework consisting of three blocks with GCN and DSMVPool layers, and each block takes a graph as input and generates a pooled graph with updated feature and adjacency matrices. Next, the pooled graph is fed into a readout function to combine all node representations and generate a single graph embedding as follows:

$$Z_j = \left[\max_{1 \le i \le N^l} X_{ij}^l\right] \forall j \in [0, d]; \quad R^l = \left[\frac{1}{N^l} \sum_{i=1}^{N^l} x_i^l \;\; \| \;\; Z\right] \tag{14}$$

16

where $Z \in \mathbb{R}^d$ vector contains the maximum values of each feature dimension across all nodes N^l at layer l-th, X_i^l is the feature vector of i-th node, and $\|$ denotes concatenation. The readout $R^l \in \mathbb{R}^{2*d}$ concatenates the two (average and max) feature representations. Finally, this graph embedding is fed into a multi-layer perceptron classifier to make predictions. Algorithm 1 summarizes the proposed method.

3.8. Complexity Analysis

The computation complexity of our dominant set method is $O(k * t * n^2)$, where k is the number of clusters, n is the number of nodes, and t is the number of iterations of the replication dynamics with $t << n$ and k depends on the unknown number of clusters in the graph. For space complexity, our method requires $O(n^2)$ to memorize the similarity matrix. If the graph is very large, we might use a function to compute the edge weight on the fly; hence, the space needed concerns the vector h (Equation. 4), which requires $O(n)$.

Table 2: Characteristics and Statistics of eight datasets.

Classification	Datasets	#Graphs	#Nodes	Avg Nodes	Avg Edges	Classes
Biological	Proteins	1,113	43, 471	39.06	72.82	2
	D&D	1,178	334,925	284.32	715.66	2
Chemical	Mutagen	4,337	131,488	30.32	30.77	2
	COX2	467	19,249	41.22	43.45	2
	BZR	405	14,478	35.75	38.36	2
Social Networks	REDDIT-M-12K	11,929	4,669,129	391.41	456.89	11
	Github_starg	12,725	1,448,038	113.79	234.64	2
	IMDB-Binary	1000	19,770	19.77	96.53	2
Computer Vision	MSRC_21	43,643	563	77.52	198.32	20

Algorithm 1 Dominant Set Multi-View Graph Pooling (DSMVPool)

Input: Graph $G = (V, X, w)$, where V is the set of nodes, X is the node feature matrix, w is the set of weights

Output: Pooled graph G' with updated adjacency matrix A' and feature matrix X'

1: Utilize A and X to calculate the local attention scores L by Eq. (1)

2: Select the indices L_{idx} of top k nodes based on local attention scores with pooling ratio p and generate G_{local} graph:

$$L_{idx} = \{i \mid L_i > p\}$$
$$G_{local} \leftarrow L_{idx}$$

3: Compute edge weights using the node features X by Eq. (5)

4: Initialize the set of dominant clusters $C = \emptyset$

5: **for** each node $v_i \in V$ **do**

6:      $C_i \leftarrow$ Replicator Dynamics by Eqs. (2) - (4)

7:      $C \leftarrow C \cup C_i$

8: **end for**

9: Generate coarser graph using dominant set clusters

$$G_{coarser} \leftarrow C$$

10: Compute node feature importance scores by Eg. 8

11: Select the indices F_{idx} of top k nodes based on feature scores with pooling ratio p and generate $G_{features}$ graph:

$$F_{idx} = \{i \mid F_i > p\}$$
$$G_{feature} \leftarrow F_{idx}$$

12: Compute fused embeddings $H_{fusion1}$ and $H_{fusion2}$ by integrating G_{local}, $G_{feature}$, and $G_{coarser}$ by Eq. (9)

13: Aggregate fused embeddings to form the final node feature matrix X' by Eq. (12)

14: Construct the final adjacency matrix A' of the pooled graph by Eq. (13)

15: **Return** the pooled graph G' with X' and A'

---

4. Experiments

This section reports the experimental details and evaluates the performance of the DSMVPool. For experiments, we selected four diverse dataset categories: chemical molecules, biological networks, social networks, and computer vision. These datasets are benchmarks against which we compare DSMVPool with state-of-the-art competitors and baseline methods. Table 2 shows the details of the datasets. Additionally, we conduct ablation studies to evaluate the contribution of each view in the DSMVPool. We also visualize the results of various pooling layers to see how baselines and the proposed pooling method reduce the nodes.

4.1. Competitors and Experimental Settings

4.1.1. Graph Neural Networks

We select four GNN architectures to conduct comparative experiments: GCN [39], GAT [36] and GraphConv [40].

4.1.2. Graph Pooling Methods

In this work, we compare our proposed DSMVPool method with a range of existing graph pooling approaches, including both global and hierarchical strategies. For global graph pooling, we evaluate SortPool [25] and Set2set [41], which are widely used methods for generating a graph-level representation by aggregating node features. We also include several advanced single-view pooling methods, such as TopkPool [19], which selects the most significant nodes based on their feature importance; SAGPool [9], which uses self-attention to evaluate node significance. Other single-view methods like DiffPool [14], CliquePool [15],Quasi-CliquePool [16], MaxCutPool-E [33], and SPGNN [42] focus on clustering nodes based on their topological or feature similarities, often with fixed cluster sizes or topological constraints. SpreadEdge [34] employs various forms of edge-based to enhance the representation learning process.

For multi-view graph pooling, we incorporate more complex methods that integrate multiple perspectives of graph structures. MuchPool [20], MPool [35], and and MAC [18] leverage multi-channel strategies, using convolutional layers to combine different graph representations. Additionally, GK-A [21], and GPIB [22] employ graph

topology-driven pooling strategies to enhance the graph representation by considering multiple views simultaneously. Finally, Wit-TopoPool [10] utilizes a topology-aware pooling method that combines global and local structures to improve graph classification performance.

These baselines provide a comprehensive comparison against the DSMVPool method, which aims to better capture graph representations by integrating global and local topological views, as well as node feature information, through a fusion-view attention layer.

4.1.3. Experimental Settings

DSMVPool is implemented using the PyTorch framework and PyTorch Geometric library [43]. We used the provided node features to generate the edge weights for our dominant set cluster pooling method (see Section 3.3). To ensure a fair comparison, we follow numerous prior research studies [9, 16, 20], employing tenfold cross-validation to assess the performance of the models listed above and provide average accuracy and standard deviation. For all the competitors we used the same GCN backbone [40] as a message-passing function to fairly emphasize the contributions of each pooling method. We utilize the official source codes of all methods provided by the authors and tune hyperparameters to reproduce the results according to the requirements in their papers. Hyperparameters are fine-tuned within specified ranges, such as embedding dimensions in the range of {64, 128, 256}, learning rate in the range of {0.01, 0.001, 0.0001}, batch size in the range of {32, 64, 128, 256} and pooling ratio in the range {0.5, 0.6, 0.7, 0.8, 0.9}. We utilize the Adam optimizer to initialize our model and apply a negative log-likelihood loss function for training. We implement patience and an early stopping criterion, which stops the training process if the loss value of the validation set does not decrease for 50 consecutive epochs. [2].

---

[2]We reproduced the accuracy numbers of all baseline pooling methods using the source codes provided by the authors. For reproducibility, the implementation of our DSMVPool and all baselines are available on this link

Table 3: Comparison of DSMVPool and baselines on chemical and computer vision datasets. The highest score is in **bold**, and the second highest score is in <u>underline</u> ("-" datasets are missing in the original papers).

Class	Methods	COX2	BZR	MUTAG.	MSRC.
GNNs	GCN	$78.16_{\pm0.85}$	$80.01_{\pm2.39}$	$78.18_{\pm2.58}$	$84.04_{\pm6.40}$
	GAT	$78.37_{\pm0.66}$	$80.74_{\pm2.52}$	$78.99_{\pm2.44}$	$88.80_{\pm4.38}$
	GraphConv	$80.01_{\pm4.43}$	$82.12_{\pm2.89}$	$79.18_{\pm3.13}$	$88.66_{\pm3.47}$
Global Pooling	SortPool	$78.20_{\pm0.02}$	$79.70_{\pm0.07}$	$75.80_{\pm2.43}$	$74.70_{\pm0.40}$
	Set2set	$78.21_{\pm0.02}$	$78.88_{\pm0.02}$	$78.00_{\pm1.19}$	$87.50_{\pm3.19}$
Single-View Pooling	SAGPool	$78.37_{\pm1.86}$	$81.24_{\pm4.01}$	$77.25_{\pm3.03}$	$87.92_{\pm3.24}$
	DiffPool	$77.60_{\pm2.70}$	$78.90_{\pm0.40}$	$71.80_{\pm0.15}$	$83.30_{\pm0.51}$
	SPGNN	–	–	$78.70_{\pm0.67}$	$89.10_{\pm0.93}$
	CliquePool	$78.37_{\pm1.86}$	$82.17_{\pm2.25}$	$78.47_{\pm1.62}$	$88.72_{\pm1.14}$
	QuasiPool	<u>$80.15_{\pm2.13}$</u>	$82.21_{\pm2.58}$	$79.53_{\pm1.58}$	$88.15_{\pm2.28}$
	MaxCutPool-E	$74.84_{\pm4.62}$	$82.09_{\pm2.01}$	$79.00_{\pm1.00}$	$89.12_{\pm3.09}$
	SpreadEdge	$79.10_{\pm2.31}$	$81.19_{\pm2.22}$	$76.00_{\pm4.00}$	–
Multi-View Pooling	MAC	$78.36_{\pm0.16}$	<u>$82.30_{\pm2.58}$</u>	<u>$80.33_{\pm2.01}$</u>	<u>$89.75_{\pm2.12}$</u>
	MuchPool	$79.27_{\pm6.09}$	$80.28_{\pm6.93}$	$78.75_{\pm2.48}$	$86.85_{\pm3.61}$
	MPool	$78.10_{\pm0.10}$	$78.70_{\pm0.11}$	$79.60_{\pm3.70}$	$87.52_{\pm0.54}$
	Wit-TopoPool	$80.65_{\pm3.05}$	$79.75_{\pm0.14}$	$73.88_{\pm1.29}$	$89.16_{\pm4.06}$
	GK-A	–	–	$77.42_{\pm0.29}$	–
	GPIB	$78.32_{\pm2.01}$	$80.71_{\pm2.12}$	$77.91_{\pm2.21}$	–
Proposed	**DSMVPool**	**$81.13_{\pm2.96}$**	**$83.58_{\pm2.75}$**	**$81.39_{\pm1.98}$**	**$90.89_{\pm2.5}$**
	Gain	**+0.98**	**+1.28**	**+1.06**	**+1.14**

4.2. Performance Comparison

We evaluated our proposed DSMVPool method alongside various baseline methods on eight different datasets for the graph classification task. The results, including accuracy and standard deviation, are presented in Tables 3 and 4. DSMVPool outperforms all baseline methods in the domains of social networks, chemical molecules, biological networks, and computer vision.

In the chemical molecular domain, for example, DSMVPool achieves a 1.28% improvement over the best baseline on the BZR dataset, 0.98% on COX2, and 1.06% on Mutagenicity. These improvements are particularly significant given that the chemical compound datasets have a lower average number of edges compared to others, as shown in Table 2. This sparsity in edge connections poses a challenge for pooling methods,

Table 4: Comparison of DSMVPool and baselines on Biological and social networks datasets. The highest score is in **bold**, and the second highest score is in underline (OOR referred to as out-of-resources and "-" referred datasets are missing in the original papers).

Class	Methods	RED-M	Github-S	IMDB-B	D&D	PROT.
GNNs	GCN	$23.84_{\pm1.92}$	$60.30_{\pm3.34}$	$60.10_{\pm5.34}$	$75.13_{\pm4.14}$	$73.77_{\pm5.59}$
	GAT	$21.73_{\pm1.03}$	$60.87_{\pm2.14}$	$52.00_{\pm2.12}$	$76.91_{\pm2.68}$	$75.30_{\pm5.23}$
	GraphConv	$37.53_{\pm2.34}$	$62.31_{\pm2.33}$	$70.12_{\pm2.78}$	$76.57_{\pm3.98}$	$74.04_{\pm5.07}$
Global Pooling	SortPool	$42.50_{\pm0.03}$	$61.91_{\pm2.24}$	$39.21_{\pm1.90}$	$67.47_{\pm6.23}$	$74.66_{\pm5.08}$
	Set2set	$41.50_{\pm0.06}$	$62.01_{\pm2.19}$	$70.00_{\pm4.50}$	$72.30_{\pm3.95}$	$71.12_{\pm5.09}$
Single View Pooling	SAGPool	$36.79_{\pm3.53}$	$63.87_{\pm2.03}$	$70.65_{\pm3.36}$	$72.49_{\pm2.87}$	$73.50_{\pm4.56}$
	DiffPool	OOR	OOR	$68.40_{\pm0.51}$	$74.31_{\pm2.15}$	$77.62_{\pm4.97}$
	SPGNN	$36.11_{\pm2.13}$	$64.10_{\pm2.44}$	$\underline{72.77}_{\pm0.32}$	$79.16_{\pm0.58}$	$76.00_{\pm0.53}$
	CliquePool	$36.11_{\pm2.13}$	$63.56_{\pm2.54}$	$70.00_{\pm3.71}$	$74.81_{\pm3.87}$	$73.56_{\pm2.86}$
	QuasiPool	$38.21_{\pm2.43}$	$64.45_{\pm2.76}$	$70.30_{\pm1.45}$	$75.30_{\pm3.30}$	$75.68_{\pm1.38}$
	MaxCutPool-E	$39.21_{\pm1.90}$	$64.76_{\pm2.01}$	$70.01_{\pm2.90}$	$77.00_{\pm3.00}$	$74.00_{\pm4.00}$
	SpreadEdge	–	–	$71.80_{\pm1.50}$	$76.10_{\pm3.50}$	$75.10_{\pm3.10}$
Multi View Pooling	MAC	$\underline{42.67}_{\pm2.23}$	$66.76_{\pm2.54}$	$57.20_{\pm0.92}$	$79.13_{\pm4.70}$	$76.08_{\pm3.55}$
	MuchPool	OOR	OOR	$65.10_{\pm6.65}$	$76.48_{\pm7.01}$	$78.52_{\pm3.89}$
	MPool	OOR	OOR	$71.02_{\pm3.57}$	$\underline{80.20}_{\pm2.10}$	$\underline{79.30}_{\pm3.30}$
	Wit-TopoPool	$33.87_{\pm1.01}$	$\underline{67.11}_{\pm2.13}$	$71.15_{\pm2.06}$	$71.30_{\pm0.37}$	$75.88_{\pm5.60}$
	GA-A	$38.32_{\pm0.15}$	–	$72.45_{\pm0.28}$	$78.65_{\pm0.27}$	$75.18_{\pm0.31}$
	GAIB	–	–	$71.25_{\pm2.12}$	$78.89_{\pm0.62}$	$76.37_{\pm0.59}$
Proposed	**DSMVPool**	$\mathbf{44.61}_{\pm2.15}$	$\mathbf{68.86}_{\pm2.35}$	$\mathbf{72.89}_{\pm2.98}$	$\mathbf{80.91}_{\pm2.14}$	$\mathbf{79.84}_{\pm2.53}$
	Gain	+1.94	+1.75	+0.12	+0.71	+0.54

but the fusion-view layer in DSMVPool captures both local and global graph structures effectively, enabling better information extraction from sparse graph topologies.

Furthermore, DSMVPool consistently surpasses GCN-based global pooling methods across all datasets. This highlights the method's ability to generate robust graph representations, emphasizing the necessity of integrating hierarchical pooling layers into the graph learning process. Notably, DSMVPool and other node cluster pooling methods, such as DiffPool, CliquePool, Quasi-CliquePool, and MPool, outperform traditional GNN models. On the other hand, methods like CliquePool and DiffPool do not always outperform simpler node selection pooling methods, such as SAGPool and TopkPool, underscoring the importance of combining both local and global information for effective graph classification.

When comparing with multi-view graph pooling baselines, such as MuchPool [20], MPool [35], and Wit-TopoPool [10], DSMVPool demonstrates superior performance across all datasets. Particularly, MuchPool and MPool leverage multi-channel and motif-based pooling strategies, yet DSMVPool still outperforms these methods, achieving significant improvements. Notably, Wit-TopoPool, which incorporates topological information into the pooling process, performs well but does not match the performance of DSMVPool.

In addition, DSMVPool also outperforms two state-of-the-art multi-view graph pooling methods: GK-A [21] and GPIB [22]. GK-A uses graph kernels to capture high-order node relationships, while GPIB focuses on propagating node importance via graph-based importance propagation. While both methods are effective at integrating different graph views, they do not achieve the same level of performance as DSMVPool, particularly in handling the complexity of graph structures across various domains.

These results highlight DSMVPool's effectiveness in integrating multi-view information, combining global and local topological properties with node feature data through its innovative fusion-view attention mechanism. DSMVPool significantly enhances graph representations across various datasets. This is evidenced by a 1.14% improvement in a computer vision dataset such as MSRC21. Even on the IMDB-Binary dataset, which has fewer nodes and graphs, DSMVPool achieves a modest 0.12% increase. Notably, when evaluated on large-scale datasets such as REDDIT-MULTI-12k and Github_starg, which represent real-world social network data with thousands of nodes and graphs, DSMVPool outperforms the baselines, showing an impressive 1.14% improvement on REDDIT-MULTI-12k and 1.75% on Github_starg. This further underscores the robustness of DSMVPool, particularly in the social network domain, where the complexity of relationships between nodes presents a significant challenge for graph pooling methods.

These results indicate that DSMVPool consistently outperforms the baseline pooling approaches across eight distinct benchmarks. Notably, it also improves performance on datasets with fewer nodes, demonstrating the effectiveness of integrating both local and global graph features.

4.3. Ablation Study

This section presents an ablation study on DSMVPool to better understand the contributions of its various components, particularly the fusion-view attention layer and three pooling views. We introduce several variations of DSMVPool by removing different views and the fusion mechanism, and evaluate their impact on performance across different graph datasets. For convenience, we name the DSMVPool method without the *local*, *global*, *feature*, and *fusion-view attention layer* as $DSMVPool_{NLV}$, $DSMVPool_{NGV}$, $DSMVPool_{NFV}$, and $DSMVPool_{NFAL}$, respectively.

We perform experiments on five different-scale graph datasets, ranging from small graphs to large graphs, to observe how each variation affects performance. The results presented in Table 5 highlight the considerable impact of our dominant set-based clustering pooling and fusion-view attention layer, particularly within the domains of chemical molecules, biological and computer vision graphs, since capturing the global structure with local or node features is especially useful for the classifier to distinguish the graphs.

Notably, when DSMVPool is evaluated without the fusion-view attention layer ($DSMVPool_{NFAL}$), the performance drops by 1.0% and 2.0%, particularly on datasets with complex structures like Proteins, MSRC_21 and BZR, respectively. The fusion layer is critical for integrating local, global, and feature-based information, allowing the model to more effectively represent graphs for classification tasks. Furthermore, it can be observed that $DSMVPool_{NLV}$ outperforms $DSMVPool_{NFV}$ in datasets characterized by sparse graph structures, such as those involving chemical molecules, because the local topology pooling operation may generate isolated nodes, which could adversely affect message passing in subsequent network layers.

Overall, DSMVPool effectively learns both the local and global topological information of the graph, integrating node features and utilizing a fusion-view attention layer to select and combine information from the most relevant nodes, thereby generating robust graph representations that significantly improve performance in classification tasks.

Table 5: Effect of different graph pooling views in DSMVPool.

Architecture	MUTAGEN	PROTEINS	BZR	IMDB-B	MSRC_21
DSMVPool$_{NLV}$	79.82 ± 2.63	78.65 ± 2.51	79.02 ± 1.16	70.80 ± 4.41	88.47 ± 4.24
DSMVPool$_{NGV}$	78.47 ± 2.97	76.56 ± 2.79	80.00 ± 5.48	71.00 ± 4.08	87.93 ± 3.51
DSMVPool$_{NFV}$	80.61 ± 1.70	78.21 ± 2.70	78.77 ± 1.59	71.70 ± 3.03	87.75 ± 5.03
DSMVPool$_{NFAL}$	81.01 ± 1.40	78.87 ± 2.40	81.23 ± 2.79	71.98 ± 3.03	89.80 ± 5.12
DSMVPool	**81.39 ± 2.01**	**79.84 ± 2.53**	**83.58 ±3.15**	**72.10 ± 3.40**	**90.89 ± 2.5**

4.4. Sensitivity Analysis of Pooling Ratio

We conducted a sensitivity analysis of the pooling ratio parameter in DSMVPool to assess its impact on model performance across various datasets. The pooling ratio determines the fraction of nodes retained during the pooling operation, and the results of this analysis are illustrated in Figure 2.

In general, DSMVPool integrates contextual information from local, global, and feature-based perspectives, making it more resilient to changes in the pooling ratio. Even with a pooling ratio as low as 0.5, DSMVPool consistently achieves promising performance across most datasets. However, the effect of the high pooling ratio is more pronounced in certain domains, particularly for chemical-based datasets like COX2 and Mutagen, which are relatively sparse in terms of node connections. The structure of these graphs, which resemble molecular configurations such as benzene rings and carbon chains, relies on maintaining key topological relationships. When many nodes are removed, the graph's structure is disrupted, leading to a drop in accuracy.

On the other hand, D&D, MSRC, and REDDIT-M-12K, which feature denser graphs, show a less drastic performance decline when the pooling ratio is reduced. These datasets are less sensitive to the removal of nodes, as their more connected structures can still retain essential information even at lower pooling ratios.

Despite these challenges, DSMVPool's ability to combine local, global, and feature-based graph representations ensures that, even with a 0.5 pooling ratio, the model retains a substantial amount of relevant information. This multi-view approach helps the model
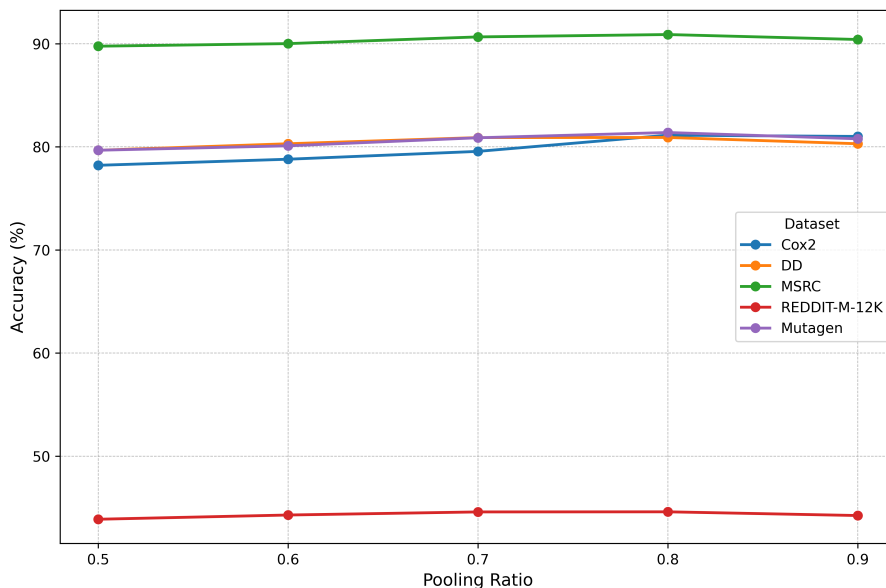
Figure 2: Sensitivity analysis of DSMVPool across different pooling ratios (p = 0.5 to 0.9) on five representative datasets (COX2, D&D, MSRC, REDDIT-M-12K, and Mutagen). The results indicate that DSMVPool achieves optimal performance at p = 0.8 (with a 0.2% reduction in the number of nodes) for most datasets. Sparse chemical datasets (COX2 and Mutagen) show sharper performance degradation at p = 0.5 due to disrupted molecular structures, while denser datasets (D&D, MSRC, REDDIT-M-12K) remain relatively stable.

maintain strong performance across different graph types, making DSMVPool robust to variations in the pooling ratio and adaptable to a wide range of graph structures.

4.5. Graph Visualization

To further demonstrate how DSMVPool outperforms baselines, we use networkx3 to visualize the pooling outcomes of the proposed approach and three other pooling methods: MuchPool, SAGPool, and CliquePool (see Figure 3). For a fair comparison, we make a three-layer hierarchical pooling architecture and set a 0.7 pooling ratio. We select a random graph from the Mutagenicity dataset containing 35 nodes. The first pooling layer of all methods shows that the DSMVPool, CliquePool, and MuchPool largely preserved the significant topological structure of the original graph, including ring and branch structures. However, the SAGPool contains several isolated nodes. The

26

findings from the second and third pooling layers demonstrate that SAGPool, Clique-Pool, and MuchPool struggle to generate pooled graphs with appropriate topological structures.
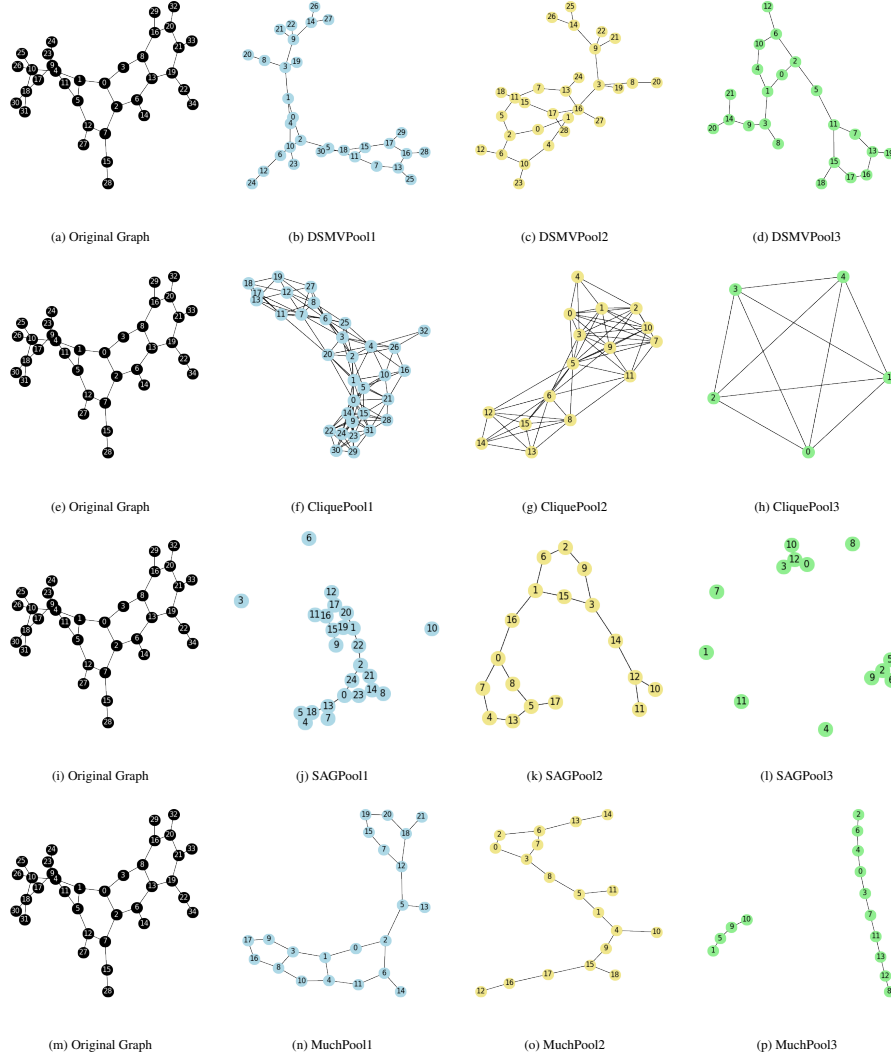

(a) Original Graph   (b) DSMVPool1   (c) DSMVPool2   (d) DSMVPool3

(e) Original Graph   (f) CliquePool1   (g) CliquePool2   (h) CliquePool3

(i) Original Graph   (j) SAGPool1   (k) SAGPool2   (l) SAGPool3

(m) Original Graph   (n) MuchPool1   (o) MuchPool2   (p) MuchPool3

Figure 3: Graph visualization of different pooling approaches. The graphs show that DSMVPool preserves the graph's original topological structures during the pooling operation.

Meanwhile, the second and third layers of our DSMVPool generate the pooled graph with reasonable topological structures, including dual ring structures in the original

27

graph. We can see the effectiveness of DSMVPool since ring structures are crucial in characterizing molecules. The visualization findings align with the results in Tables 3 and 4.

4.6. Limitations

The proposed DSMVPool effectively captures the graph's local and global characteristics with node features and integrates them to generate a more robust pooled graph. However, the fundamental dominant set approach relies on solving a quadratic optimization problem through replication dynamics, which does not allow direct gradient propagation. As a result, it is challenging to integrate DSMVPool into an end-to-end trainable framework.

5. Conclusion

This study developed a novel Dominant Set Multi-View Pooling approach for hierarchical graph representation learning. We developed a dominant set cluster pooling method to identify all potential clusters using edge weights, generating a coarser graph view. Furthermore, we generated two pooled graph views by selecting the most significant nodes based on the graph's local topological structures and node features importance. Next, we developed an attention-based fusion-view layer to fuse the coarser graph with the pooled graphs and perform aggregation to form the final pooled graph. We evaluated the performance of DSMVPool on ten graph-classification datasets, including four distinct domains. Our method outperforms baselines and competitors on eight datasets and achieves comparable results on the other two. The DSMVPool is not an end-to-end since the dominant set approach is not differentiable.

In the future, we aim to address the non-differentiability of the dominant set approach, which currently limits the end-to-end training of our model. One potential direction is to explore differentiable alternatives to the dominant set algorithm or to design approximations that could allow us to integrate the DSMVPool into a fully differentiable graph learning framework. Furthermore, we plan to extend DSMVPool to handle larger, more complex graphs with varying node and edge types, making it

applicable to a broader range of domains, such as analyzing protein structures for drug discovery. Specifically, we will focus on enhancing the ability of DSMVPool to model protein-ligand interactions, improving predictions for drug-target binding affinities, and identifying key structural features critical for drug design. Additionally, we intend to investigate how DSMVPool can be adapted to dynamic graphs, where the graph structure evolves over time, as seen in applications like social network analysis and real-time recommendation systems.

References

[1] A. Begga, W. Ali, G. Niculescu, F. Escolano, T. Stadelmann, M. Pelillo, Community-hop: Enhancing node classification through community preference, in: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2024, pp. 21–30.

[2] M. A. Abbas, W. Ali, F. Smarandache, S. S. Alshamrani, M. A. Raza, A. Alshehri, M. Ali, Residual Attention Augmentation Graph Neural Network for Improved Node Classification Residual Attention Augmentation Graph Neural Network for Improved Node Classification, Infinite Study, 2024.

[3] X. Chen, S. Li, R. Liu, B. Shi, J. Liu, J. Wu, K. Xu, Molecular graph contrastive learning with line graph, Pattern Recognition (2025) 111380.

[4] M. Réau, N. Renaud, L. C. Xue, A. M. Bonvin, Deeprank-gnn: a graph neural network framework to learn patterns in protein–protein interfaces, Bioinformatics 39 (1) (2023) btac759.

[5] W. Ju, Y. Gu, Z. Mao, Z. Qiao, Y. Qin, X. Luo, H. Xiong, M. Zhang, Gps: Graph contrastive learning via multi-scale augmented views from adversarial pooling, Science China Information Sciences 68 (1) (2025) 112101.

[6] A. Amouzad, Z. Dehghanian, S. Saravani, M. Amirmazlaghani, B. Roshanfekr, Graph isomorphism u-net, Expert Systems with Applications 236 (2024) 121280.

[7] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, W. Wang, Simgnn: A neural network approach to fast graph similarity computation, in: Proceedings of the twelfth ACM international conference on web search and data mining, 2019, pp. 384–392.

[8] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3693–3702.

[9] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in: International conference on machine learning, PMLR, 2019, pp. 3734–3743.

[10] Chen, Yuzhou and Gel, Yulia R, Topological pooling on graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 7096–7103.

[11] J. Sang, Y. Wang, W. Ding, Z. Ahmadkhan, L. Xu, Reward shaping with hierarchical graph topology, Pattern Recognition 143 (2023) 109746.

[12] W. Ali, S. Vascon, T. Stadelmann, M. Pelillo, Hierarchical glocal attention pooling for graph classification, Pattern Recognition Letters 186 (2024) 71–77.

[13] H. Zhang, L. Yu, G. Wang, S. Tian, Z. Yu, W. Li, X. Ning, Cross-modal knowledge transfer for 3d point clouds via graph offset prediction, Pattern Recognition 162 (2025) 111351.

[14] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, Advances in neural information processing systems 31 (2018).

[15] E. Luzhnica, B. Day, P. Lio, Clique pooling for graph classification, arXiv preprint arXiv:1904.00374 (2019).

[16] W. Ali, S. Vascon, T. Stadelmann, M. Pelillo, Quasi-cliquepool: Hierarchical graph pooling for graph classification, in: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, 2023, pp. 544–552.

[17] D. Bacciu, A. Conte, R. Grossi, F. Landolfi, A. Marino, K-plex cover pooling for graph neural networks, Data Mining and Knowledge Discovery 35 (5) (2021) 2200–2220.

[18] Y. Xu, J. Wang, M. Guang, C. Yan, C. Jiang, Multistructure graph classification method with attention-based pooling, IEEE Transactions on Computational Social Systems 10 (2) (2022) 602–613.

[19] H. Gao, S. Ji, Graph u-nets, in: international conference on machine learning, PMLR, 2019, pp. 2083–2092.

[20] J. Du, S. Wang, H. Miao, J. Zhang, Multi-channel pooling graph neural networks., in: IJCAI, 2021, pp. 1442–1448.

[21] X. Qin, L. Bai, L. Cui, M. Li, Z. Lyu, H. Du, E. Hancock, Dhtagk: Deep hierarchical transitive-aligned graph kernels for graph classification, Preprint-IJCAI 2025.

[22] J. Long, Z. Wang, T. Chen, L. Yang, Improving hierarchical graph pooling with information bottleneck, Journal of King Saud University Computer and Information Sciences 37 (3) (2025) 36.

[23] H. Du, W. Wang, L. Bai, Dual-channel embedding learning model for partially labeled attributed networks, Pattern Recognition 142 (2023) 109644.

[24] M. Pavan, M. Pelillo, Dominant sets and pairwise clustering, IEEE transactions on pattern analysis and machine intelligence 29 (1) (2006) 167–172.

[25] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An end-to-end deep learning architecture for graph classification, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 32, 2018.

[26] H. Yuan, S. Ji, Structpool: Structured graph pooling via conditional random fields, in: Proceedings of the 8th International Conference on Learning Representations, 2020.

[27] Y. Wang, W. Hou, N. Sheng, Z. Zhao, J. Liu, L. Huang, J. Wang, Graph pooling in graph neural networks: Methods and their applications in omics studies, Artificial Intelligence Review 57 (11) (2024) 294.

[28] S. Stanovic, B. Gaüzère, L. Brun, Graph neural networks with maximal independent set-based pooling: Mitigating over-smoothing and over-squashing, Pattern Recognition Letters 187 (2025) 14–20.

[29] S. Deng, G. Yang, Y. Yang, Z. Gong, C. Chen, X. Chen, Z. Hao, Module-based graph pooling for graph classification, Pattern Recognition 154 (2024) 110606.

[30] J. Huang, Z. Li, N. Li, S. Liu, G. Li, Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6480–6489.

[31] Y. Ma, S. Wang, C. C. Aggarwal, J. Tang, Graph convolutional networks with eigenpooling, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 723–731.

[32] D. Bacciu, A. Conte, R. Grossi, F. Landolfi, A. Marino, K-plex cover pooling for graph neural networks, Data Mining and Knowledge Discovery 35 (5) (2021) 2200–2220.

[33] Abate, Cristiano and Bianchi, Filippo Maria, MaxCutPool: Differentiable Feature-Aware MaxCut for Pooling in Graph Neural Networks, in: Proceedings of the International Conference on Learning Representations (ICLR), 2025.

[34] K. Limbeck, L. Mezrag, G. Wolf, B. Rieck, Geometry-aware edge pooling for graph neural networks, arXiv preprint arXiv:2506.11700 (2025).

[35] M. I. K. Islam, M. Khanov, E. Akbas, Mpool: Motif-based graph pooling, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2023, pp. 105–117.

[36] Veličković, Petar and Cucurull, Guillem and Casanova, Arantxa and Romero, Adriana and Liò, Pietro and Bengio, Yoshua, Graph Attention Networks, in:

Proceedings of the International Conference on Learning Representations (ICLR), 2018.

[37] M. Pavan, M. Pelillo, A new graph-theoretic approach to clustering and segmentation, in: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., Vol. 1, IEEE, 2003, pp. I–I.

[38] J. W. Weibull, Evolutionary game theory, MIT press, 1997.

[39] Kipf, Thomas N and Welling, Max, Semi-Supervised Classification with Graph Convolutional Networks, in: Proceedings of the International Conference on Learning Representations (ICLR), 2017.

[40] C. Morris, M. Ritzert, Martin, W. L. Hamilton, J. E. Lenssen, G. Rattan, M. Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, in: Proceedings of the AAAI conference, 2019, pp. 4602–4609.

[41] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: International conference on machine learning, PMLR, 2017, pp. 1263–1272.

[42] Z. Dong, M. Zhang, Y. Chen, Spgnn: Recognizing salient subgraph patterns via enhanced graph convolution and pooling, arXiv preprint arXiv:2404.13655 (2024).

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, Z. Yang, Edward, A. Lin, Zeming, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).