Pattern Recognition Letters
journal homepage: www.elsevier.com

Hierarchical Glocal Attention Pooling for Graph Classification

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

[a] *Ca' Foscari University of Venice, Via Torino, 155, Mestre, Venice, 30170, Italy*
[b] *ZHAW Centre for Artificial Intelligence, Technikumstrasse 9, Winterthur, 8401, Switzerland*
[c] *ECLT European Centre for Living Technology, Sestiere Dorsoduro, 3911, Venice, 30123, Italy*

ABSTRACT

Graph pooling is an essential operation in Graph Neural Networks that reduces the size of an input graph while preserving its core structural properties. Existing pooling methods find a compressed representation considering the Global Topological Structures (e.g., cliques, stars, clusters) or Local information at node level (e.g., top-k informative nodes). However, an effective graph pooling method does not hierarchically integrate both Global and Local graph properties. To this end, we propose a dual-fold Hierarchical Global Local Attention Pooling (HGLA-Pool) layer that exploits the aforementioned graph properties, generating more robust graph representations. Exhaustive experiments on nine publicly available graph classification benchmarks under standard metrics show that HGLA-Pool significantly outperforms eleven state-of-the-art models on seven datasets while being on par for the remaining two.

1. Introduction

Graph Neural Networks (GNNs) have recently gained significant attention due to their ability to process graph-structured data effectively. They have shown effectiveness in various tasks of classifying graphs and learning graph representations (Kipf & Welling, 2016), including understanding and predicting interactions between molecules and proteins (Wang et al., 2022; Réau et al., 2023a) that can lead to significant advancements in drug discovery, analyzing structure and dynamics of interactions in social networks (Khemani et al., 2024), operating on knowledge graphs to enable Retrieval-Augmented Generation (RAG) (Mavromatis & Karypis, 2024) or enable the detection of visual objects from their context (Han et al., 2022), e.g., to support document accessibility (Schmitt-Koopmann et al., 2022). Within the scope of graph classification, the Graph Pooling (GP) methods play a pivotal role in GNN architectures that map the set of nodes or subgraphs into a compact representation to capture a meaningful structure of the entire graph (Pang et al., 2021). Early GP methods, referred to as global methods (Bai et al., 2019; Simonovsky & Komodakis, 2017), are the simplest approaches, which reduce the size of the input graph by performing a sum or average of all nodes without considering the hierarchical information of the graph and may lose feature information. Recently, several advanced hierarchical GP methods, such as node cluster and node selection methods (Chen & Gel, 2023; Gao & Ji, 2019; Luzhnica et al., 2019), have been proposed to tackle the limitations of global pooling and obtain state-of-the-art performance.

The node cluster methods like DiffPool (Ying et al., 2018) and CliquePool (Luzhnica et al., 2019) capture the Global Topological Structure (GTS) of a graph, which extracts the overall architecture and connectivity patterns of the entire network by partitioning the graph into clusters. Then, each cluster transfers into a supernode to create a coarsened graph. However, the CliquePool (Luzhnica et al., 2019) method has limitations in extracting overlapping nodes between cliques of the same size and treats all cliques equally informative without considering node features. On the other hand, node selection methods such as TopKPool, SAGPool, and Topological Pooling (Xu et al., 2022; Lee et al., 2019; Chen & Gel, 2023) identify the most significant nodes based on their feature values or attention scores while dropping unnecessary nodes. These methods primarily focus on the graph's Local Topological Structure (LTS), which includes the analysis of individual nodes, their attributes,

**Corresponding author:
e-mail: waqar.ali@unive.it (Waqar Ali),
sebastiano.vascon@unive.it (Sebastiano Vascon), stdm@zhaw.ch (Thilo Stadelmann), pelillo@unive.it (Marcello Pelillo)

and neighborhoods. However, there is a lack of an effective GP method that combines the graph's global and local properties hierarchically to improve the graph representation.

To address the limitations of existing GP methods, this study proposes Hierarchical GlobalLocal Attention Pooling (HGLA-Pool), a novel pooling method designed to capture graphs' local and global properties. The proposed approach is structured as a two-fold process, each fold addressing specific limitations of existing GP approaches. The fold-1 leverages the idea of cliques that perform clique pooling (Luzhnica et al., 2019), which aims to encapsulate the graph's GTS. In this fold, we design a rule-based method to enhance the clique representation by identifying the overlapping nodes between cliques. Identifying overlapping nodes is crucial in various real-world scenarios, such as molecular biology, where a molecule might serve as a shared binding site for multiple proteins, and in social network analysis, where individuals often belong to multiple social circles, including professional networks and personal connections (Wang et al., 2022; Khemani et al., 2024). Additionally, we introduce a novel dynamic fusion method that incorporates graph structure information and node features to calculate scores for each clique and select the topk significant cliques using this fusion score. Furthermore, we recognize that every node within a selected clique does not contribute equally to the graph representation during the pooling operation. We argue that the most informative nodes from all ranked cliques should be captured in GP. Therefore, in fold-2, we develop a LocalPool layer on top of fold-1 using multi-attention to pinpoint and emphasize the most informative nodes within the ranked cliques, ensuring a more meaningful representation of the overall graph for pooling operation. The hierarchical architecture of our method, as illustrated in Figure 1, reflects this dual-fold approach. We summarized our contributions as follows:

- This study introduces HGLA-Pool, a novel pooling layer that sequentially integrates the global structural properties of the graph with the local node's properties.
- We develop a rule-based method to extract the overlapping nodes between cliques. We also design a novel dynamic fusion method that leverages graph structural information and node features to identify the most relevant global structures (cliques). Furthermore, this study develops a multi-attention LocalPool to capture the local node's properties.
- We experimentally prove that considering both sources of information (global and local) yields a better-learned representation. We consistently outperformed 11 state-of-the-art models on seven diverse and challenging benchmarks.

2. Related Work

Related work includes existing GP studies focusing on global and hierarchical techniques. Global pooling methods usually use sum or mean functions to aggregate the node features that generate a single vector representation for the entire graph. For example, Vinyal et al. (Vinyals et al., 2015) proposed a Set2Set framework that uses the Long Short-Term Memory model to generate graph representation by identifying informative nodes. Authors (Zhang et al., 2018) developed a novel GNN layer to capture multi-level node features and a sort pooling that sorts these features to keep more graph information. However, global methods ignore the graph's hierarchical information during the pooling operations. Meanwhile, hierarchical methods leverage graph structures and node features for the pooling. Based on designing properties, the hierarchical methods can be classified into two main classes: node selection and node cluster.

Node selection methods reduce the graph size by selecting the most informative nodes based on their node features or attention scores. Hongyang et al. (Gao & Ji, 2019) developed a TopkPool that uses node scalar values on a trainable projection vector as the node score. SAGPool (Lee et al., 2019) further adopts the GNN layer to calculate node scores. Jinheon et al. (Xu et al., 2022) proposed Multistructure Attention Convolutional (MAC) pooling that incorporates dual-node scoring strategies to obtain the importance of nodes. These methods are considered more computationally efficient as they only require calculating an informative node's score, but they ignore the graph's GTS information. The node cluster method captures the global properties by finding the clusters within the graph. DiffPool (Ying et al., 2018) uses the GNN layer output to learn a differentiable soft cluster assignment to extract the clusters. In (Luzhnica et al., 2019), the authors introduced a CliquePool method to learn the GTS by extracting the maximal cliques from the graph. Quasi-CliquePool (Ali et al., 2023) further improves the CliquePool by capturing the overlapping nodes between two cliques using the replicator dynamic algorithm. (Islam et al., 2023) developed a Motif-based pooling method that extracts the high-order graph structure by combining cluster and selection pooling operations. The motif method treated all motif structures equally without considering the node features. Graph Multiset Transformer (GMT) pooling incorporates node structural dependencies with a multi-head attention mechanism to enhance the graph representation by identifying node interaction. Jinlong et al. (Du et al., 2021) design a Multi-channel pooling (MuchPool) that combines TopkPool and DiffPool methods to create a more comprehensive hierarchical representation of graphs. However, the MuchPool method suffers from considerable computational complexity due to the simultaneous operation of three distinct channels to capture node feature information, global and local structural information. A recent study (Chen & Gel, 2023) introduced a Wit-TopoPool, which integrates a witness complex-based topological embedding mechanism with a global pooling layer. This approach aims to extract comprehensive and discriminative topological information from graphs.

In conclusion, the current landscape in pooling methods demonstrates a notable gap: the lack of an effective pooling approach that sequentially combines node selection and node cluster techniques to yield a robust graph representation. Addressing this gap, we proposed a novel HGLA-Pool method that captures the graph's local and global properties with node features to generate a more robust graph representation.

3. Methodology

This section explains our proposed HGLA-Pool, shown in Figure. 1. The HGLA-Pool performs pooling operations in two-fold, capturing the graph's global and local properties and then aggregating the results to form a new pooled graph. The following sections present a detailed description of the proposed HGLA-Pool.

3.1. Preliminaries

This section defines mathematical notations and concepts about GNNs, which will be used in this study.

We represent the input graph as $G = (V, E)$, where V denotes the nodes and E the edges in the graph. The connection between nodes of G can be represented by an adjacency matrix $A \in \mathbb{R}^{N\times N}$, where $N = |V|$ is the number of nodes. The matrix $X \in \mathbb{R}^{N\times d}$ represents the node features matrix, where d is the dimension of the feature space. The GP operation changes the number of nodes at each pooling layer, thus, we further identify the graph at the l-th layer as $G'^l = (V'^l, E'^l)$. The node feature and adjacency matrices of the pooled graph represent as $X'^l \in \mathbb{R}^{N'^l\times d'^l}$ and $A'^l \in \mathbb{R}^{N'^l\times N'^l}$.

Given a graph dataset $D = \{(G_1, Y_1), (G_2, Y_2), \ldots\}$, where G_i and Y_i are the ith graphs and the set of labels corresponding to graphs, respectively. The principle of the graph classification task is to learn a mapping function, such as $f : G \rightarrow Y$.

Several GNN architectures have been developed, such as Graph Convolutional Network (GCN) Kipf & Welling (2016) and Graph Attention Network (GAT) Veličković et al. (2017). They can learn node representations by propagating and aggregating information from a node and its neighbors, achieving outstanding performance in various graph-related tasks. Therefore, this study adopts hybrid graph convolution operators to extract significant cliques (i.e., a graph is a subset of nodes such that every two distinct nodes are adjacent) and node representations from multiple perspectives. In the following, we briefly outline the mechanisms of GCN (Kipf & Welling, 2016) and GAT (Veličković et al., 2017).

GCNs learn node representations by considering topology information and node features in a graph. The basic operation of a GCN layer can be described as follows:

$$X^{(l+1)} = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}X^{(l)}W^{(l)}) \tag{1}$$

where $\hat{A}$ is the adjacency matrix with added self-loops, $\hat{D}$ is the degree matrix, $X^{(l)}$ is the feature matrix at layer l, $W^{(l)}$ is the trainable weight matrix, and σ is an activation function.

While GATs introduce an attention mechanism to GNNs, allowing the model to assign different importance to different nodes within a neighborhood. The expression of GAT is given as follows:

$$X^{(l+1)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)} a_{ij}^{k} W^{k} X_j\right) \tag{2}$$

where K represents the number of heads, X_j is the feature vector of node v_j, which is a neighbor of node v_i, a_{ij}^{k} is the attention weight between X_i and X_j, and W^k is a trainable weight matrix for head k.

3.2. Fold 1: Global Topological Structure Learning

In fold 1, we aim to capture the GTS information of the graph. For this purpose, we use the CliquePool (Luzhnica et al., 2019) with a modified version of the Bron-Kerbosch algorithm (Cazals & Karande, 2008) to extract all cliques from the input graph G, denoted as a set $C = \{c_1, c_2, ..., c_i\}$. This fold is designed to overcome two significant challenges of CliquePool (Luzhnica et al., 2019): extracting overlapping nodes between cliques of the same size and ranking cliques based on their node features.

Overlapping Nodes Extraction: Overlapping nodes substantially impact various real-world scenarios, including molecular biology, where a particular molecule may act as a shared binding site for multiple proteins, and in social network analysis, a person may be part of both a professional network (Wang et al., 2022; Réau et al., 2023b). We design a rule-based algorithm using three conditions to handle nodes that belong to multiple cliques during the clique extraction process: 1) If a node is already assigned to a clique, it is only added to the current clique being considered if the sizes of the existing and current clique are equal or current clique size is bigger than existing clique in terms of nodes, 2) If a node is connected to multiple cliques, in that case, we evaluate whether this specific node has maximum connectivity with the nodes of an associated clique. If it does and is not already part of the associated clique, the node is integrated into that clique, and 3) If nodes of a clique have already been assigned to larger cliques, then we remove that clique. This criterion ensures that the node's association with the clique is not arbitrary but is based on a meaningful and substantial connectivity pattern. In Figure. 1, the fold-1 demonstrates the extraction of overlapping nodes; see the cliques 1, 2, 3, and 4 with overlapping nodes.

The existing clique pooling method (Luzhnica et al., 2019) partitions the graph into cliques based on the graph's topological structure without considering the node features. Furthermore, this method treats all cliques equally, which can be problematic because not all cliques are equally significant; some are rich in information, and others might merely consist of a single node, thus carrying significantly less information to subsequent pooling layers. To solve these issues, we developed a novel dynamic fusion method to calculate each clique's importance using graph structure information and node features.

Graph Structure Score: The graph structure can provide useful information on the importance of nodes, such as node degrees and the shortest paths between different nodes. Hence, we borrow the node centrality definition to illustrate each clique's importance within the graph. We introduce a heuristic method to calculate a score for each clique using graph structure information: fusion of degree centrality and unique neighbor count. In the first step, we compute the mean degree centrality $\bar{D}_{c_i}$ of all nodes in c_i. Next, we calculate the number of unique neighbors $\bar{U}_{C_i}$ of a clique c_i by collecting all neighbors of nodes in c_i and removing any nodes that are part of this clique itself. Finally, the structure score S_{c_i} of each clique is derived as a weighted sum of these two $\bar{D}_{c_i}$ and $\bar{U}_{C_i}$ metrics, parameterized by β and calculated as follows:

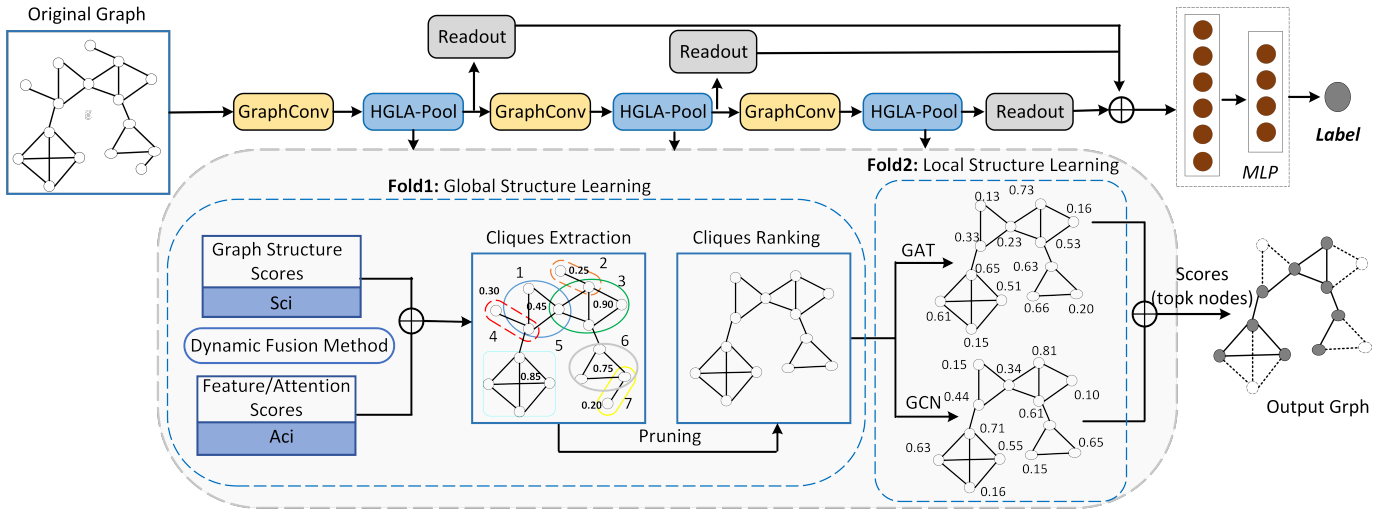


Fig. 1: The hierarchical architecture of the HGLA-Pool integrates with GNN for graph classification. The grey lightbox demonstrates the overall workflow of HGLA-Pool, which contains two folds. The **fold-1** performs three steps: 1) capture GTS by finding all maximal cliques within the graphs; 2) find overlapping nodes between two cliques to obtain more reasonable clique representations; and 3) incorporate the dynamic fusion method to rank the cliques and form a new pooled graph. Meanwhile, the **fold-2** takes this pooled graph as input and captures the LTS by selecting several significant nodes from ranked cliques based on scoring criteria (e.g., GCN and GAT). The readout function is applied after each pair of GNN and HGLA-Pool layers to generate the graph-level representation.

$$\bar{D}_{c_i} = \frac{1}{|c_i|}\sum_{v\in c_i} d_v; \quad \bar{U}_{c_i} = \left|\cup_{v\in c_i} N(v) - c_i\right| \quad S_{c_i} = \beta\times\bar{D}_{c_i} + (1-\beta)\times\bar{U}_{c_i}; \tag{3}$$

Node Feature Score: In addition to topological information, the node features information contributes significantly to defining the graph properties (Kong et al., 2023). For instance, in chemical molecules, node features represent atom types, which are essential for predicting the graph's characteristics. Therefore, it can be an effective resource to show the clique's importance within a graph. For this purpose, we use a graph convolution layer to calculate importance scores for each node. The calculation procedure is outlined as follows:

$$Z(v) = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW); \quad Z_{c_i} = \sum_{v\in c_i} Z(v) \tag{4}$$

where $Z(v) \in \mathbb{R}^{N\times 1}$ represents the importance scores and $\sigma(.)$ is a nonlinear activation function (e.g. softmax). The $\tilde{A} \in \mathbb{R}^{N\times N} = A + I$ shows the adjacency matrix with self-loops, $\tilde{D} \in \mathbb{R}^{N\times N} = \sum_j \tilde{A}_{ij}$ is the degree matrix of $\tilde{A}$ and $W \in \mathbb{R}^{d\times 1}$ is a trainable weight parameter. Various aggregation functions like max and sum can be employed to calculate the score for each clique. In our implementation, we choose the sum function to aggregate the importance scores of the nodes within the corresponding clique c_i. Given a clique c_i, the importance clique score Z_{c_i} is computed in equation 4.

Fusion Score: To achieve a more robust score for each clique, it is necessary to use both graph structure and node feature information fully. So, we combine S_{c_i} with the Z_{c_i} score using a learnable parameter α to obtain a fuse score F_{c_i} (see leftmost side of equation 5), thus providing a more comprehensive representation of the graph's topology. After obtaining the F_{c_i} scores, we used them to select cliques that the pooling operator should preserve. In detail, we first rank the cliques according to their fusion score F_{c_i}, then a subset of top-ranked cliques are selected for constructing the pooled graph as follows:

$$F_{C_i} = \alpha \times Z_{C_i} + (1-\alpha)\times S_{C_i}; \quad C'_j = \text{rank}(F_{C_i}, \lceil C_r \times |C| \rceil) \tag{5}$$

where rank(.) denotes the function that returns the important cliques, C_r is the clique ratio, and $|C|$ is the total cliques. We update the node feature based on selected clique nodes (see leftmost side of equation 6). We construct a new adjacency matrix A' where only the edges between the nodes in the selected cliques are retained. For this purpose, we define an edge mask matrix E_m, which is structured to correspond with the A using node mask M_i. Each element in E_m is set to 1 if both corresponding nodes (v_i and v_j in M_i) are part of the selected cliques and set to 0 otherwise. This can be represented mathematically as:

$$X' = X \odot M; \quad A' = A \odot E_m \tag{6}$$

where X' is the new node feature matrix, X is the original node feature matrix, $\odot$ represents the broadcasted element-wise product, and M is a vector where each entry corresponds to a node in the graph. The dimension of M is the same as the number of nodes in the graph. Each entry in M specifies whether or not the corresponding node is part of a selected clique (1 for nodes in C'_k and 0 for all other nodes). A' is the new adjacency matrix, and A is the original edge index matrix. The first fold's output serves as an input for the subsequent fold. The information provided by this fold acts as a contextual framework that enables the second fold to identify local structures effectively.

3.3. Fold 2: Local Topological Structure Learning

This section presents how our proposed LocalPool layer leverages multi-attention mechanisms to learn LTS information by extracting the most informative nodes from ranked cliques. Existing pooling methods like CliquePool and KPlexPool (Luzhnica et al., 2019; Bacciu et al., 2021) extract all possible cliques from the input graph and often transfer each clique

into a supernode to form a pooled graph. This transformation aims to reduce the graph size and complexity by aggregating the information within each clique. However, this transformation does result in a loss of information at the individual node level within each clique. The fine-grained details of interactions and relationships between nodes within a clique are not preserved in the pooled representation. This loss of information can potentially affect the downstream tasks, especially if the node-level information is crucial for classification tasks.

To address this, we propose a LocalPool layer to refine the graph further by selecting the most important nodes from ranked cliques. This refinement and selection process allows our pooling operation to help preserve the key information within each clique while reducing the graph size and complexity. Our LocalPool approach significantly diverges from existing methods like SAGPool (Lee et al., 2019). Specifically, SAGPool relies on a single strategy for computing node importance, yielding less robust node rankings. Inspired by the MAC method Xu et al. (2022), we adopt a dual-strategy-based pooling layer to calculate node importance, as illustrated in Figure 1. First, we employ a GCN layer to calculate importance scores for the nodes. This choice stems from the GCN model's ability to learn node representations by aggregating information from neighboring nodes, offering a localized perspective. GCNs aggregate feature information from a node's local neighborhood, providing a stable and robust representation by averaging the features of neighboring nodes. Simultaneously, we employ the Graph GAT layer to compute attention scores. GATs introduce an attention mechanism that assigns different importance to different nodes within a neighborhood, capturing more intricate relationships. GATs can adapt to varying graph structures by learning the attention coefficients, making them versatile and capable of capturing complex patterns and dependencies that might be missed by the uniform aggregation of GCNs. Given that GCN and GAT extract graph information from distinct viewpoints—GCN provides a robust, averaged view of a node's neighborhood, and GAT offers a nuanced, dynamically weighted view. we consider this process a multi-attention strategy for calculating node importance scores. By aggregating the scores from both GCN and GAT (using operations such as sum or max), we achieve a more comprehensive and robust node importance score, which enables our pooling method to select the most important nodes. We consider this process as a multi-attention strategy, and it is calculated as follows:

$$S_{N_i} = \text{Aggregation}\Big(\sigma\big(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)} {a_{ij}}^k W^k X'_j\big), \sigma\big(\tilde{D}^{-1/2}\tilde{A}'\tilde{D}^{-1/2}X'W\big)\Big) \tag{7}$$

where S_{N_i} represents a multi-attention score for a node, and Aggregation is an operation such as max, mean, and sum. We use the max function to select the maximum value from each row (e.g., for each node, choose the maximum score from either the GCN or GAT model). The $W^k \in \mathbb{R}^{d\times 1}$ and $W \in \mathbb{R}^{d\times 1}$ are parameter weight matrices and used for linear transformations of node features. We identify and select high-score nodes from the output graph of Fold-1 to construct a pooled graph using the scores S_{N_i}. The details of this process are as follows:

$$idx = topk(S_{N_i}, \lceil P_r * N' \rceil); \quad X^{l+1} = X'(idx,:) \odot S_{N_i}(idx,:); \quad A^{l+1} = A'(idx, idx) \tag{8}$$

where *topk* is the function that returns the indices of the top $N^{l+1} = \lceil P_r * N' \rceil$ values. The $X'(idx,:)$ perform the row-wise (node-wise) indexed node feature matrix, $S(idx,:)$ represents the row-wise indexed node importance score matrix, and $A'(idx, idx)$ is the row-wise and col-wise indexed adjacency matrix. X^{l+1} and A^{l+1} represent the new node feature and adjacency matrices, respectively.

3.4. Hierarchical HGLA-Pool Architecture and Readout Layer

We employ a hierarchical pooling architecture, integrating multiple GCN layers with HGLA-Pool layers to perform graph classification tasks. Figure 1 shows the architecture consisting of three blocks with GCN and HGLA-Pool layers. Each block receives a graph as input and generates a pooled graph with a new feature and adjacency matrices. Concurrently, we use mean-pooling and max-pooling as a readout layer after each block to aggregate all the node representations to obtain a single graph embedding as follows:

$$\Gamma_j = \Big[\max_{1\le i\le N^l} X^l_{ij}\Big] \forall j \in [0, d]; \quad R^l = \Big[\frac{1}{N^l}\sum_{i=1}^{N^l} x^l_i \quad \| \quad \Gamma\Big] \tag{9}$$

where N^l is the number of nodes at layer l-th, x_i is the feature vector of i-th node, and $\|$ denotes concatenation. The $\Gamma \in \mathbb{R}^d$ vector contains the maximum values of each feature dimension across all nodes. 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.

4. Experiments and Analysis

This section presents a comprehensive evaluation and quantitative analysis of our proposed HGLA-Pool's effectiveness. We perform exhaustive experiments across four diverse dataset categories: chemical molecular structures (Wale et al., 2008) (including Mutagenicity, NCI-1, NCI-109, COX2, and BZR), social networks (Bacciu et al., 2021) (Reddit-Multi-12K), biological networks (Dobson & Doig, 2003) (DD and Proteins) and computer vision (Irwin et al., 2012) (MSRC_21). Table 2 shows the statistics of each dataset. Additionally, we conduct ablation studies to evaluate the individual contributions of each fold within the HGLA-Pool layer. To further analyze the robustness of our approach, we also explore the impact of hyperparameter variations on the performance of our pooling method.

4.1. Baselines and Experimental Settings

We select three GNN architectures to conduct comparative experiments, including GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), and GraphConv (Morris et al., 2019) and eleven state-of-the-art hierarchical and global pooling approaches such as DiffPool (Ying et al., 2018), SortPool (Zhang et al., 2018), TopkPool (Gao & Ji, 2019), SAGPool (Lee et al., 2019), CliquePool (Luzhnica et al., 2019), GMT (Baek et al., 2021), MuchPool (Du et al., 2021), MPool (Islam et al., 2023),

Table 1: Comparison of HGLA-Pool and baselines. The highest score is in **bold**, and the second highest score is in underline (OOR referred to as out-of-resources).

Class	Baselines	PROTEINS	D&D	NCI-1	NCI-109	COX2	BZR	MUTAGEN	RED-M-12K	MSRC-21
GNNs	GCN[2017]	73.77 ± 5.59	75.13 ± 4.14	74.68 ± 3.09	73.47 ± 2.22	78.16 ± 0.85	80.01 ± 2.39	78.18 ± 2.54	23.84 ± 0.92	84.04 ± 6.40
	GAT[2018]	75.30 ± 5.23	76.91 ± 2.68	75.16 ± 3.36	73.78 ± 3.38	78.37 ± 0.66	80.74 ± 2.52	78.99 ± 2.44	21.73 ± 0.03	88.80 ± 4.38
	GraphConv[2018]	74.04 ± 5.07	76.57 ± 3.98	78.15 ± 1.86	77.01 ± 2.52	80.79 ± 4.43	83.51 ± 2.89	79.18 ± 2.24	37.53 ± 1.97	88.66 ± 2.49
Pooling	DiffPool[2018]	77.62 ± 4.97	74.31 ± 2.15	70.70 ± 0.22	70.20 ± 0.24	77.60 ± 2.70	78.90 ± 0.04	71.80 ± 0.15	OOR	83.30 ± 0.51
	SortPool[2018]	74.66 ± 5.08	67.47 ± 6.23	70.58 ± 3.68	68.87 ± 2.38	78.20 ± 0.02	79.70 ± 0.07	75.80 ± 2.43	42.50 ± 0.36	74.70 ± 0.40
	TopkPool[2019]	73.32 ± 6.09	73.85 ± 4.63	74.84 ± 4.62	75.09 ± 2.37	77.93 ± 2.79	81.49 ± 4.17	79.56 ± 1.96	37.56 ± 3.39	88.45 ± 3.70
	SAGPool[2019]	73.50 ± 4.56	72.49 ± 2.87	74.26 ± 1.96	74.94 ± 3.13	78.37 ± 1.86	81.24 ± 4.01	79.25 ± 3.03	36.79 ± 3.53	87.92 ± 3.24
	CliquePool [2019]	73.56 ± 2.86	74.81 ± 3.87	77.26 ± 1.53	76.49 ± 1.14	78.37 ± 1.86	82.17 ± 2.25	78.47 ± 1.62	36.11 ± 2.13	88.72 ± 1.14
	GMT[2021]	75.09 ± 0.59	78.72 ± 0.59	74.21 ± 1.88	71.38 ± 2.03	80.86 ± 0.41	82.25 ± 0.70	80.53 ± 0.11	44.06 ± 0.09	90.53 ± 0.34
	MuchPool[2021]	78.52 ± 3.89	76.48 ± 7.01	74.70 ± 2.25	71.84 ± 2.66	79.27 ± 6.01	80.28 ± 6.93	74.75 ± 2.48	OOR	86.85 ± 3.61
	MPool[2023]	**79.30 ± 3.30**	**81.20 ± 2.10**	77.40 ± 1.90	73.50 ± 2.50	78.10 ± 0.10	78.70 ± 0.11	79.60 ± 3.70	OOR	87.52 ± 0.54
	MAC[2023]	76.08 ± 3.55	79.13 ± 4.70	77.60 ± 1.66	75.84 ± 1.86	78.36 ± 0.16	82.76 ± 5.91	80.33 ± 1.49	42.67 ± 2.23	89.75 ± 0.22
	Quasi-CliquePool[2023]	75.68 ± 1.38	75.30 ± 3.30	78.21 ± 1.83	77.38 ± 2.23	80.15 ± 2.12	82.87 ± 2.58	79.53 ± 1.58	38.21 ± 2.43	88.15 ± 2.22
	Wit-TopoPool[2023]	75.88 ± 5.60	71.30 ± 0.37	70.58 ± 0.29	69.71 ± 0.25	81.73 ± 3.50	79.75 ± 0.14	73.88 ± 1.29	33.87 ± 1.01	89.16 ± 4.00
Proposed	**HGLA-Pool+Z_C**	76.58 ± 3.34	77.57 ± 3.01	80.80 ± 1.24	**79.80 ± 2.30**	82.86 ± 3.55	85.93 ± 3.08	**82.40 ± 1.58**	**45.61 ± 2.15**	91.30 ± 2.39
	HGLA-Pool+F_C	73.33 ± 3.18	75.30 ± 3.95	**81.02 ± 1.90**	79.10 ± 1.10	**83.30 ± 3.57**	**87.43 ± 3.87**	82.25 ± 1.56	43.30 ± 2.03	**92.19 ± 2.23**

Table 2: Eight datasets' characteristics and Statistics. TG, AN, and AE denote the total number of graphs, average nodes, and average edges, respectively.

Classification	Datasets	TG	AN	AE	Classes
Biological	Proteins	1,113	39.06	72.82	2
	D&D	1,178	284.32	715.66	2
Chemical	NCI-1	4,110	29.87	32.30	2
	NCI-109	4,127	29.68	32.13	2
	Mutagen	4,337	30.32	30.77	2
	COX2	467	41.22	43.45	2
	BZR	405	35.75	38.36	2
Social	RED-M	11,929	391.41	456.89	11
Computer Vision	MSRC_21	563	77.52	198.32	20

MAC (Xu et al., 2022), Quasi-CliquePool (Ali et al., 2023), and Wit-TopoPool (Chen & Gel, 2023).

We implement HGLA-Pool using the PyTorch framework (Paszke et al., 2017) and the PyTorch Geometric library (Fey & Lenssen, 2019). To ensure a fair comparison, we follow many previous works (Lee et al., 2019; Ali et al., 2023; Du et al., 2021), employing tenfold cross-validation to evaluate the effectiveness of all models. We report the average accuracy and standard deviation. For consistency, we utilized the source codes provided by the authors of the baseline models and tuned the hyperparameters according to the specifications in their papers to reproduce the results. We use the same GCN (Morris et al., 2019) layer as a message-passing function for HGLA-Pool and all baselines.[1]

Hyperparameters are optimized within predefined ranges, including embedding dimensions {32, 64, 128, 256}, learning rate {0.01, 0.001, 0.0001, 0.05, 0.005, 0.0005}, batch size {64, 128}, pooling and clique ratios {0.5, 0.6, 0.7, 0.8, 0.9}. We employed the Adam optimizer for model initialization and a negative log-likelihood loss function for training. We also adopt patience and early stopping criterion, i.e., if the loss value of the validation set does not reduce for 50 consecutive epochs, then the training process will be stopped.

4.2. Performance Comparison

We evaluated the performance of our proposed method (HGLA-Pool with importance scores Z_C and HGLA-Pool with dynamic fusion scores F_C) and baseline methods on the nine benchmark datasets for the graph classification tasks. We outlined the classification accuracy with standard deviation in Table 1. In this comparison, HGLA-Pool method achieves superior performance among all other baselines in the chemical molecules, social networks, and computer vision domain datasets. For instance, in the case of the chemical molecular domain, our method outperforms the best baselines by 2.81% improvement for the NCI1, 2.42% for the NCI109, 3.92% for the BZR, 1.57% for the COX2 and 2.07% for the Mutagenicity. It is worth noting that the average number of edges in these datasets is much smaller than in the other datasets (see Table 2). This implies that these five chemical datasets are relatively sparse, presenting a significant challenge for pooling layers to learn meaningful representations. However, our designed dual-fold attention strategy can capture the graph's local and global information, enabling HGLA-Pool to extract information undeterred by the sparse graph structure effectively.

Furthermore, HGLA-Pool consistently outperforms GCN-based global pooling methods across datasets from all four domains. This superior performance underscores the efficacy of our approach in generating more meaningful graph representations, thereby emphasizing the value of incorporating hierarchical pooling layers into the learning process. Notably, HGLA-Pool and other node clustering techniques like CliquePool, Quasi-CliquePool, and MPool perform better than GNN-based models such as GCN, GAT, and GraphConv. This outcome suggests that capturing GTSs as maximal cliques or clusters is beneficial for enhancing graph representation learning. It is also noteworthy that CliquePool and DiffPool do not consistently outperform node selection pooling approaches such as SAGPool and TopkPool. This observation further substantiates the idea that integrating the graph's local and global properties can lead to more effective GP methods. We can also see from Table 1 that our proposed HGLA-Pool demonstrates considerable improvement in the multi-classes based datasets, with an increase of 1.66% and 1.55% on MSRC_21 and REDDIT-MULTI, respectively. Our analysis identified a significant proportion of isolated nodes and subgraphs in the biological dataset. Furthermore, the graphs in this dataset demonstrate a scarcity of structures resembling cliques. Our approach's effectiveness hinges

[1] We reproduced the numbers of all baselines using the source codes given by the authors, and the source codes for all baselines and our method are available on this link.

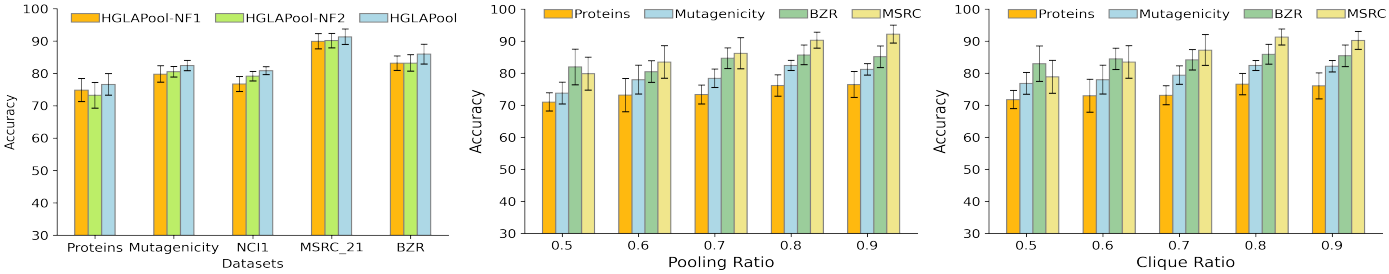


Fig. 2: Effect of dual folds and Hyperparameter analysis on HGLA-Pool performance.

significantly on identifying and ranking such cliques, leading to challenges in capturing meaningful hierarchical representation structures within these graphs. This structural difference between the dataset and our method's underlying assumption likely contributes to the observed performance decline. To sum up, our proposed HGLA-Pool consistently performs better on seven out of nine benchmarks than baseline pooling techniques.

4.3. Ablation Study

This subsection conducts an ablation study on HGLA-Pool by removing independently both folds to verify further where the performance improvement comes from. For convenience, we name the HGLA-Pool method without the first and the second fold as HGLAPool-NF1 and HGLAPool-NF2, respectively. We chose five different-scale graph datasets covering small and large graphs for these experiments. From Figure. 2 (see the leftmost graph), we can see that capturing GTS is crucial in the chemical molecule and MSRC datasets since the GLAPool-NF2 variant obtains a notable performance enhancement as graph attention facilitates the selection of the most informative cliques. Meanwhile, the GLAPool-NF1 variant outperforms GLAPool-NF2 in the PROTEINS as this dataset has limited clique-like structures, which poses challenges for our method in capturing meaningful hierarchical representation structures. This pattern suggests that the LTS with node features are more important for some datasets than the GTS. Overall, HGLA-Pool's ability to learn both the graph's local and global enables it to generate robust graph representations that significantly enhance performance in classification tasks.

4.4. Parameter Sensitivity Analysis

In this section, we investigate the sensitivities of the two main parameters, P_r and C_r, influencing the performance of the HGLA-Pool. Figure 2 (see second and third graph) summarizes the accuracy of HGLA-Pool under different combinations of these parameters across four different datasets. Our method achieves the highest accuracy for all four datasets when P_r=0.8 and C_r = 0.8, while performance drops with P_r=0.5 and C_r = 0.5. To further investigate these scenarios, we performed a more detailed analysis of the interaction between P_r and C_r, generating a heatmap (see Figure. 3) for each dataset, illustrating their combined influence. The heatmap reveals specific scenarios where certain combinations of P_r and C_r result in suboptimal performance, particularly in datasets with sparsely connected graphs like Mutagenicity. For instance, low values of P_r and C_r (e.g., $Pr = \{0.5, 0.6, 0.7\}$ and $Cr = \{0.5, 0.6, 0.7\}$) tend to exacerbate graph sparsity, leading to a loss of substantial graph structures during pooling, as depicted in Figure. 3. From our analysis, we also found that a C_r of at least 0.7 is essential for biological, social, and computer vision datasets, while a minimum C_r of 0.8 is necessary for chemical datasets due to their sparse nature. Therefore, the recommended small ranges for C_r and P_r are $\{0.7, 0.8\}$ for biological, social, and computer vision datasets and $\{0.8, 0.9\}$ for molecular datasets. Interestingly, the HGLA-Pool demonstrated stability, producing reasonably satisfactory results across most combinations of these parameters.

4.5. Graph Visualization

To further demonstrate the distinctiveness and superiority of our pooling method compared to existing techniques, we use the Networkx library to visualize the pooling outcomes of HGLA-Pool, CliquePool, TopkPool, and MuchPool. To provide a fair comparison, we build a hierarchical pooling architecture consisting of two layers and set a 0.8 for the C_r and P_r. We randomly selected a graph from the Mutagenicity dataset comprising 36 nodes for the demonstration. The input and pooled graphs of each method's first and second pooling layers are shown in the first row and the second row of Figure 4, respectively. The results demonstrate that the HGLA-Pool, CliquePool, and MuchPool mostly preserved the input graph's significant topological structure (ring and branch structures). In contrast, the results of the TopKPool are scattered with numerous isolated nodes, indicating a lack of structural preservation. The results obtained from the second pooling layer indicate that three baselines encounter challenges with preserving the underlying topology of the initial graph. However, HGLA-Pool can preserve reasonable topological structures, such as dual ring structures present in the initial graph. This underscores the effectiveness of HGLA-Pool since ring structures are crucial in the characterization of molecules.

4.6. Limitations

Our pooling method, rooted in the fundamental notion of cliques, effectively captures the graph's local and global characteristics. However, its efficacy is constrained when applied to datasets with non-clique structures, such as graphs containing many isolated nodes. For example, if the clique-based representation in fold-1 is inaccurate, it might lead to suboptimal outcomes in subsequent steps, such as clique ranking and LocalPool.

5. Conclusion

This study presents the HGLA-Pool method, an innovative approach for hierarchical graph representation. It employs a

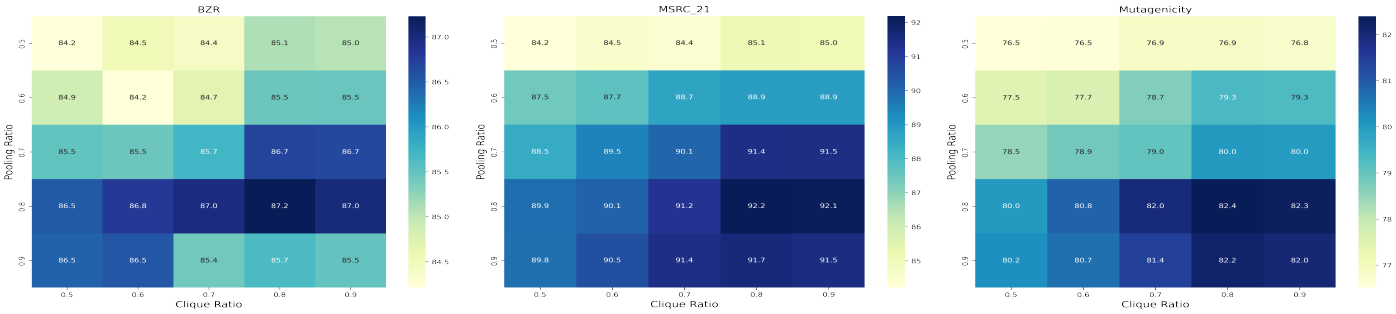
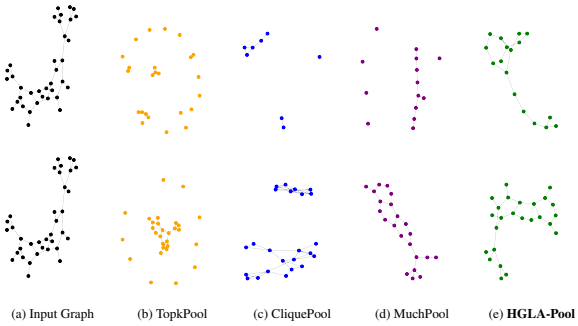


Fig. 3: Effect of different combinations of clique and pooling ratios on HGLA-Pool performance.



(a) Input Graph (b) TopkPool (c) CliquePool (d) MuchPool (e) **HGLA-Pool**

Fig. 4: Comparative graph visualization of different polling methods.

two-fold strategy to capture the graph's global and local properties sequentially. Fold-1 captures the global topological structure of a graph by identifying the cliques and overlapping nodes between cliques, coupled with a dynamic fusion scoring method to rank and select significant cliques for pooling. In Fold-2, a LocalPool layer employing a multi-attention mechanism selects the most informative nodes from these cliques, capturing the LTS. In the end, the outcomes of both folds are integrated sequentially to form a new pooled graph, which retains the original graph's structure and enhances classification performance. The effectiveness of HGLA-Pool has been rigorously assessed through extensive experiments across nine graph classification datasets spanning four distinct domains. Future work will enhance the clique pooling technique to capture a broader range of graph structures, further boosting the graph classification performance.

References

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

Bacciu, D., Conte, A., Grossi, R., Landolfi, F., & Marino, A. (2021). K-plex cover pooling for graph neural networks. *DMKD*, *35*, 2200–2220.

Baek, J., Kang, M., & Hwang, S. J. (2021). Accurate learning of graph representations with graph multiset pooling. *preprint arXiv:2102.11533*, .

Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., & Wang, W. (2019). Simgnn: A neural network approach to fast graph similarity computation. In *Proc. of the twelfth ACM ICWS-DM* (pp. 384–392).

Cazals, F., & Karande, C. (2008). A note on the problem of reporting maximal cliques. *Theoretical computer science*, *407*, 564–568.

Chen, Y., & Gel, Y. R. (2023). Topological pooling on graphs. In *Proc. of the AAAI* (pp. 7096–7103). volume 37.

Dobson, P. D., & Doig, A. J. (2003). Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, .

Du, J., Wang, S., Miao, H., & Zhang, J. (2021). Multi-channel pooling graph neural networks. In *IJCAI* (pp. 1442–1448).

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, .

Gao, H., & Ji, S. (2019). Graph u-nets. In *international conference on machine learning* (pp. 2083–2092). PMLR.

Han, K., Wang, Y., Guo, J., Tang, Y., & Wu, E. (2022). Vision GNN: An image is worth graph of nodes. *Advances in NIPSs*, *35*, 8291–8303.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., & Coleman, R. G. (2012). Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, *52*, 1757–1768.

Islam, M. I. K., Khanov, M., & Akbas, E. (2023). Mpool: Motif-based graph pooling. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 105–117). Springer.

Khemani, B., Patil, S., Kotecha, K., & Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, *11*, 18.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, .

Kong, Y., Li, J., Zhang, K., & Wu, J. (2023). Multi-scale self-attention mixup for graph classification. *Pattern Recognition Letters*, *168*, 100–106.

Lee, J., Lee, I., & Kang, J. (2019). Self-attention graph pooling. In *International conference on machine learning* (pp. 3734–3743). PMLR.

Luzhnica, E., Da, B., & Lio (2019). Clique pooling for graph classification. *arXiv preprint arXiv:1904.00374*, .

Mavromatis, C., & Karypis, G. (2024). GNN-RAG: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, .

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

Pang, Y., Zhao, Y., & Li, D. (2021). Graph pooling via coarsened graph infomax. In *Proc. of the 44th International ACM SIGIR* (pp. 2177–2181).

Paszke, A., Gross, S., Chintala, G., Yang, Z., Edward, Lin, A., Zeming, Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. *Automatic differentiation in pytorch*, .

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

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

Schmitt-Koopmann, F. M., Huang, E. M., Hutter, H.-P., Stadelmann, T., & Darvishy, A. (2022). FormulaNet: A benchmark dataset for mathematical formula detection. *IEEE Access*, *10*, 91588–91596.

Simonovsky, M., & Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3693–3702).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*, .

Vinyals, O., Bengio, S., & Kudlur, M. (2015). Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, .

Wale, N., Watson, I. A., & Karypis, G. (2008). Comparison of descriptor spaces for chemical compound retrieval and classification. *KISs*, *14*, 347–375.

Wang, Z., Liu, M., Luo, Y., Xu, Z., Xie, Y., Wang, L., Cai, L., Qi, Q., Yuan, Z., Yang, T. et al. (2022). Advanced graph and sequence neural networks for molecular property prediction and drug discovery. *Bioinformatics*, *38*.

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

Ying, Z., You, J., Morris, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Proc. NeuIPS*, *31*.

Zhang, M., Cui, Z., Neumann, M., & Chen (2018). An end-to-end deep learning architecture for graph classification. In *Proc. AIII*. volume 32.