A COMPARSION WITH OTHER ARCHITECTURES

Different architectures of hierarchical graph neural networks are shown in Figure 5.

B PSEUDO ALGORITHMS OF MSGNN

Algorithm 2: MSGNN Algorithm

Input: Graph (A^0, X^0); Subgraph size s; Pooling ratio r; Number of epochs E; Number of iterations (scales) L;
Output: Graph label $\hat{y}$

for $e = 1, \ldots, E$ **do**
  for $l = 1, \ldots, L$ **do**
    // Graph convolution
    $H^l = \sigma(\widetilde{A}^{l-1} X^{l-1} W^{l-1})$
    // Subgraph sampling
    Sample pivot, star, and cutting nodes: V_{sam};
    $\mathcal{G}^l_{\text{sub(sam)}} = \text{BFS}((A^{l-1}, X^{l-1}), V_{\text{sam}})$;
    // Subgraph selection
    $\mathcal{G}^l_{\text{sub(sel)}} = \text{top-}k(\mathcal{G}^l_{\text{sub(sam)}}, H^l)$;
    // Subgraph evolution
    $\mathcal{G}^l_{\text{sub(evo)}} = \text{SubGraphEvo}(\mathcal{G}^l_{\text{sub(sel)}})$;
    $ind = \{i\}, \forall v_i \in V(\mathcal{G}^l_{\text{sub(evo)}})$;
    $A^l = A^{l-1}(ind, ind)$;
    $X^l = H^l(ind, :)$;
    // Graph readout
    $R^l = \text{READOUT}(\{x^l_i\}^{n^l}_{i=1})$;
  end
  // Clustering
  $(A^{L+1}, X^{L+1}) = \text{CLUSTER}((A^0, H^1), (A^L, X^L))$;
  $R^{L+1} = \text{READOUT}(\{x^{L+1}_i\}^{n^{l+1}}_{i=1})$;
  // Graph classification
  $R = \|^{L+1}_{l=1} R^l$;
  $\hat{y} = \text{MLP}(R)$;
  Train MSGNN: $\mathcal{L} = -\sum^n_{i=1} y_i \log \hat{y}_i$.
end

C DATASETS

D&D [10] and **PROTEINS** [6] contain graphs of protein structures labeled as enzymes or non-enzymes. In the D&D dataset, a node represents an amino acid and two nodes are connected by an edge if the distance is less than 6 Angstroms apart. In the PROTEINS dataset, nodes are secondary structure elements, and an edge connecting two nodes indicates that they are in an amino acid sequence or in a close 3D space. **NCI1** and **NCI109** [42] are two biological datasets screened for activity against non-small cell lung cancer and ovarian cancer cell lines. **MUTAG** [18] is a molecular dataset where labels are based on the mutagenic effects on bacteria. In these three datasets, each graph represents a chemical compound, with nodes and edges representing atoms and chemical bonds, respectively. **IMDB-B** and **IMDB-M** are movie-collaboration datasets encompassing actor/actress and genre information sourced from IMDB. In these two datasets, the nodes symbolize the actor/actress entities, while the edges denote instances where they co-appear in the same movies. The statistics of the datasets are summarized in Table 5.

Table 5: Statistics of datasets.

Dataset	Number of Graphs	Number of Classes	Avg. Nodes	Avg.Edges	Node Labels
D&D	1178	2	384.31	715.65	82
PROTEINS	1113	2	39.06	72.81	3
NCI1	4110	2	29.87	32.30	37
NCI109	4127	2	29.68	32.13	38
MUTAG	188	2	17.93	39.59	7
IMDB-B	1000	2	19.80	96.50	–
IMDB-M	1500	3	13.00	65.90	–

D BASELINES

For *flat pooling*, two methods are compared:

Set2Set [41] implements a pooling operation by aggregating node features through the LSTM.

DGCNN [54] sorts nodes in a consistent order based on their structural roles. By ranking nodes, DGCNN learns the global graph topology instead of summing node features.

Hierarchical pooling methods are further considered:

DiffPool [50] groups nodes into clusters by different assignment functions, and re-generates edges among these clusters.

Graph U-net [13] selects top-ranked nodes for pooling through a project vector.

SAGPool [22] considers both node features and graph topology to select top-ranked nodes through GNNs.

ASAP [31] samples representative clusters from local neighborhoods to form a pooled graph.

GIB [51] proposes Graph Information Bottleneck to search informative and compressive subgraphs.

GXN [23] aims to learn the multi-scale features of graphs through feature-crossing layers.

PAS [44] utilizes neural architecture search (NAS) to search the optimal architecture in search space for graph classification.

GMT [2] uses a multi-head attention based global pooling layer to capture the interaction between nodes and can be extended to the clustering-based methods for hierarchical graph pooling.

Graph kernel-based methods are also compared:

SHORTPATH [5] defines expressive graph kernels which are based on the shortest path.

WL [34] leverages the Weisfeiler–Lehman (WL) algorithm as a subroutine to extract subtree features for graph classification.

E IMPLEMENTATION DETAILS

In the network depth study, to examine the impact of network depth on the MSGNN performance, we use different network depths on the graph classification task. We set the number of iterations in the range of $[2, 5]$ and the pooling ratio $[0.7, 0.6]$, $[0.7, 0.6, 0.5]$,

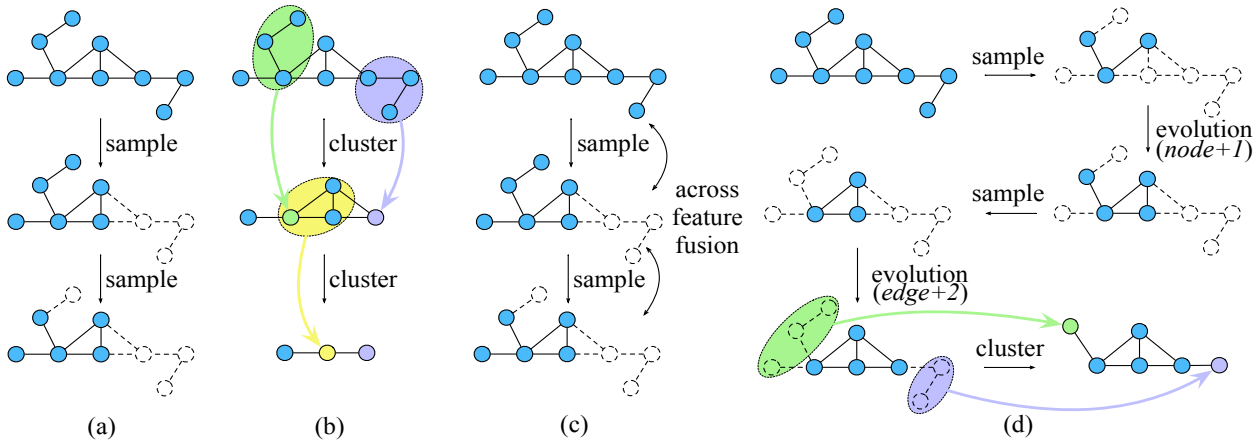

Figure 5: Architectures of hierarchical graph neural networks. (a) Graph U-net samples nodes across scales to downsize the graph. (b) Diffpool clusters nodes hierarchically to coarsen the graph. (c) GXN leverages cross-scale feature fusion to promote information flow. (d) Our MSGNN not only samples key substructures at multiple scales but also clusters nodes to preserve the graph topology. Notably, MSGNN employs evolution to find suitable substructures for downstream tasks.

$[0.7, 0.6, 0.5, 0.4]$, and $[0.7, 0.6, 0.5, 0.4, 0.3]$. We obtain the best classification performance when the depth is 3. In the efficiency study, to assess the efficiency of subgraph evolution with respect to graph size, we compare the evolution time of MSGNN on the D&D and MUTAG. D&D contains large graphs, while MUTAG consists of smaller ones. As expected, the evolution of large graphs naturally takes longer due to the increased number of candidate nodes and edges.

F CASE STUDY

We conduct a case study of the ablation experiment on the MUTAG dataset as shown in Figure 6. "W/o evolution" only selects subgraphs without evolving them. If important subgraphs are not selected at a specific scale, they will certainly not be preserved after multi-scale pooling. The lack of key subgraphs that are critical for classification during pooling leads to wrong prediction results. However, MSGNN achieves subgraph reversibility by remedying subgraphs through subgraph evolution. We can see that MSGNN selects subgraphs by the top-k method, and some important subgraphs are not selected at a certain scale. But the subgraphs are evolved under the guidance of downstream tasks through the algorithm we designed. In this way, a part of important nodes are added to enrich subgraph structures, which play an essential role in classification. As a result, we get correct classification results. The clustering layer retains the information of unimportant nodes through clusters. We can observe that it is the presence of these clusters that allows the five ring structures of the original graph to be preserved, suggesting that the clustering layer plays a vital role in maintaining the topology of the graph.

G NODE CLASSIFICATION

For node classification, we conduct experiments on D&D dataset [10]. We choose a graph with 1,704 nodes and 4,298 edges, where the node class is 20. In MSGNN, the number of nodes will decrease after multi-scale pooling. To classify all nodes in the graph, it is necessary to restore the original graph structure through graph unpooling. After L unpooling layers, we feed all node representations into the MLP classifier to perform node classification. Specifically, we design an inverse operation of the pooling layer, inspired by [13], to restore the graph. Firstly, we fill a zero matrix $O \in \{0\}^{n \times d}$ with node features X^L according to the original indices of the nodes. Then, we update the filled matrix through the graph convolution layers, in which the information is propagated from the nodes in the graph (A^L, X^L) to the padding nodes via the original graph structure. For the specific implementation of node classification, there are three points to note:

- Table 4 only reports operations of subgraph evolution during the pooling process, and graph unpooling is not involved.
- The clustering layer in MSGNN is not considered in the node classification task due to the need to maintain the original graph structure.
- For graph classification tasks, node features are generally initialized with one-hot vectors of node categories [13], while for node classification, in order to prevent label leakage, we use random initialization.

The classification accuracy is 82.31% on node classification. Combined with Table 2, it can be seen that our MSGNN performs well on a variety of tasks, suggesting the universality of the method.

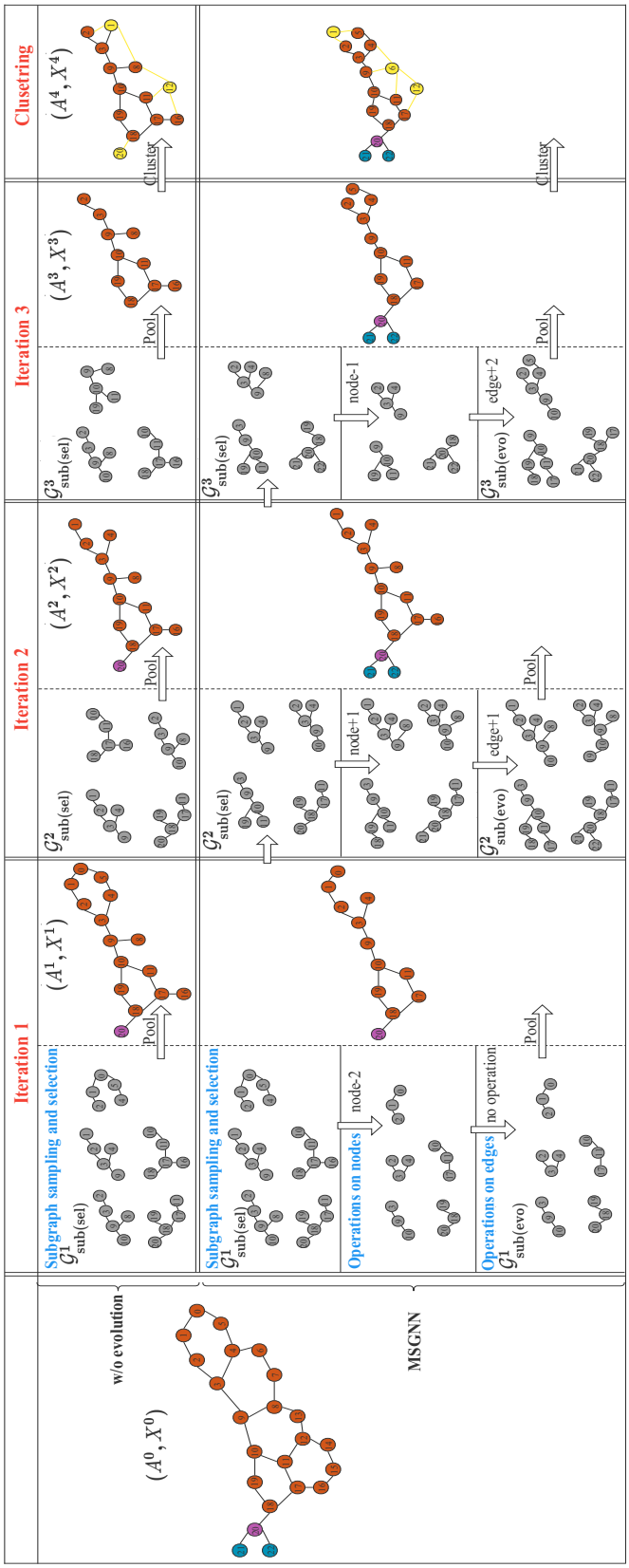

Figure 6: Case study of the ablation experiment. (A^0, X^0) **is a graph labeled as mutagenic from the MUTAG dataset. The row "w/o evolution" shows the process of pooling over three iterations and clustering. In each iteration, subgraph sampling and selection are performed. Finally, the prediction result is wrong. The rows "MSGNN" displays the process of subgraph sampling and selection, subgraph evolution (Operations on nodes and Operations on edges), and clustering. Lastly, the prediction result is correct. In the last column, the yellow nodes are the clusters.**