SPGP: Structure Prototype Guided Graph Pooling

Anonymous Author(s)
Affiliation
Address
email

Abstract

While graph neural networks (GNNs) have been successful for node classification tasks and link prediction tasks in graph, learning graph-level representations still remains a challenge. For the graph-level representation, it is important to learn both representation of neighboring nodes, i.e., aggregation, and graph structural information. A number of graph pooling methods have been developed for this goal. However, most of the existing pooling methods utilize k-hop neighborhood without considering explicit structural information in a graph. In this paper, we propose Structure Prototype Guided Pooling (SPGP) that utilizes prior graph structures to overcome the limitation. SPGP formulates graph structures as learnable prototype vectors and computes the affinity between nodes and prototype vectors. This leads to a novel node scoring scheme that prioritizes informative nodes while encapsulating the useful structures of the graph. Our experimental results show that SPGP outperforms state-of-the-art graph pooling methods on graph classification benchmark datasets in both accuracy and scalability.

1 Introduction

Graph neural networks (GNNs) have been successfully applied to graph-structured data for node classification tasks [22, 14, 41] and link prediction tasks [48, 46]. Most of the existing GNNs use graph convolution to aggregate neighbor information at the node-level. However, graph-level representation requires additional condensation of an entire graph. For this, a number of graph pooling methods have been developed and used for learning a compact embedding of an entire graph. Capturing global or hierarchical structures in a graph is important to learn accurate representations of graphs of diverse characteristics such as size, connectivity, or density. Due to the structural diversity of graphs, developing efficient and effective graph pooling methods remains a challenge. The main difficulty is how to consider node features and graph structures simultaneously.

To address these issues of the graph pooling methods, a number of methods have been developed to capture global or hierarchical structures of a graph. Global pooling methods [38, 24, 49] obtain graph-level representations directly from node-level representations via permutation invariant readout functions. They handle global interactions between nodes, but they are inherently flat and are not powerful enough to learn the hierarchical structures of the graph. Recently, hierarchical graph pooling methods have been proposed, reducing gradually a graph to smaller pooled graphs while maintaining essential structures of a graph. To construct the pooled graphs, a number of techniques have been explored based on a concept of the clustering algorithms [43, 25, 45], a priority of node features [9] and integration of structural information [23, 47, 10], and end-to-end structure learning [34, 51].

However, these approaches extract graph structural information from k-hop neighborhood, which faces two technical issues. First, graph pooling based on k-hop neighborhood depends on k, which is often an arbitrary value. When the value of k is small, the receptive field of a k-hop neighborhood is relatively limited to local information which does not encapsulate the global structural information of a graph [11]. On the other hand, with a large value of k, irrelevant nodes can be easily included within

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

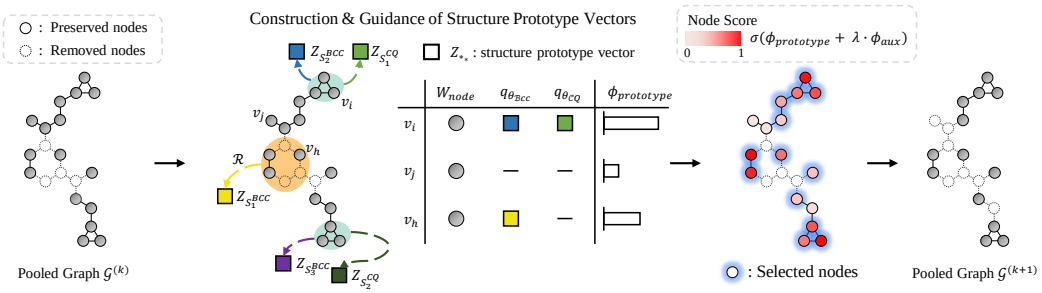


Figure 1: Concept of SPGP: SPGP first constructs the prototype vectors $Z_{S_*^{BCC}}$ and $Z_{S_*^{CQ}}$ by utilizing biconnected components and cliques as the structure prototypes. The global structural importance of a node is calculated based on the information of the node itself (W_{node}) and the affinities between the node and the prototype vectors, which are measured by relation modules q_{θ_s}. Finally, with an auxiliary score, SPGP extracts important nodes to generate the pooled graph.

the receptive field. This observation can be even worse in scale-free networks such as technological and biological networks [3, 42, 29]. Second, explicit graph structures, that can be difficult to represent with k-hop neighborhood, are overlooked. For example, biconnected component (BCC), one of the important graph structures in graph theory, is an important candidate graph structure. Mutual interactions within the BCC can cover long-range information with low noise. Thus, BCC is widely used to effectively capture the global structural information of a graph in various tasks such as weighted vertex cover or minimum cost flow [16, 35, 15]. Another promising candidate is a set of cliques. It is well known that the clique set is closely related to the global topology of the graph [31, 33]. Due to these global topology-related properties, cliques are being utilized for predicting graph-level properties in various fields (e.g., in chemistry [4] and biology [36]). These observations motivate us to develop a graph pooling method that utilizes explicit graph structures beyond k-hop neighborhood.

In this paper, we propose the Structure Prototype Guided Pooling (SPGP) that extracts informative nodes and obtains more accurate graph representations by utilizing the overlooked graph structures related to the global information of the graph. Given the prior graph structures, SPGP considers them as structure prototypes and computes their information through learnable prototype vectors in an end-to-end fashion. To guide the pooling, SPGP prioritizes nodes by measuring the affinity between the nodes and the prototype vectors. We extensively validate SPGP on real-world graphs in widely used benchmark datasets including the large-scale Open Graph Benchmark (OGB) dataset, where SPGP outperforms existing graph pooling methods in both accuracy and scalability. Our paper's main contributions are:

- We propose a new graph pooling that utilizes prior graph structures, which are hard to represent as k-hop neighborhood, to make more accurate graph representations with global structural information.
- We show that SPGP can capture structural information better than the existing pooling methods. Additionally, SPGP efficiently addresses the isolated node problem of graph pooling, without computationally expensive procedures.
- We evaluate SPGP on widely used benchmark datasets for graph classification under the extensive experiments such as 20 different seeds with 10-fold cross-validation on each seed. We also demonstrate SPGP outperforms over the state-of-the-art graph pooling methods.

2 Related Works

2.1 Graph Neural Networks

Recently, various GNNs have been proposed to compensate for the shortcomings of graph representation learning through matrix factorization [32] or random walk [13]. These GNNs can be categorized into two domains: spectral and spatial. Spectral methods generally utilize graph Fourier transform with approximations for handling large graphs [5, 22]. Spatial methods [14, 41, 37], on the other

hand, focus on the spatial relationship of nodes, such as the convolution of an image. The main idea is a *message passing* algorithm in which a node iteratively aggregates the node's neighborhood information and updates its information.

2.2 Graph Pooling

The pooling operation performs downsampling on the graph to learn the graph representation efficiently. There are two branches of the pooling operations: global pooling and hierarchical pooling. The global pooling methods [38, 24, 49] summarize the entire graph at once by permutation invariant readout functions. Since these methods do not capture the hierarchical structures in the graph, hierarchical pooling methods have been proposed. DiffPool [43] uses node embeddings to smoothly assign nodes into clusters to construct a new graph topology. TopK Pooling [9] selects the top k-th nodes by scoring the nodes with their hidden representations. SAGPool [23] is an extension of TopK Pooling that calculates node scores by considering neighborhood information via an additional GNN. ASAP [34] clusters local subgraphs by capturing local extremum information with maintaining graph connectivity. GSAPool [47] learns structural feature-based information by mixing node information and structural information. HGP-SL [51] performs attention based structure learning on the pooled graph. Unlike these methods, which focus only on k-hop neighborhood, our approach uses more global structural information through prior graph structures.

3 SPGP: Structure Prototype Guided Pooling

In this section, we introduce our proposed method SPGP for graph pooling as illustrated in Figure 1. Given prior graph structures such as BCC or cliques, SPGP computes the significance of the graph structures in the form of learnable prototype vectors using node representations with contextual information. The importance of these structure prototype vectors is then distributed to the nodes belonging to the graph structures. Further, with an auxiliary node scoring scheme, SPGP prioritizes the structural importance of nodes to select the top scoring nodes for creating the pooled graph. In the following, we describe the SPGP approach in detail.

3.1 Notations and Problem Statement

The problem of graph classification considers a set of labeled graphs $D = \{(\mathcal{G}_1, y_1), \ldots, (\mathcal{G}_M, y_M)\}$, and we aim to learn a mapping $f : G \to Y$, where G is the set of graphs and Y is the set of graph labels. For an arbitrary graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, X_i)$, the node set $\mathcal{V}_i$ and the edge set $\mathcal{E}_i$ have n_i nodes and e_i edges, respectively. $X_i \in \mathbb{R}^{n_i \times \delta}$ represents the node feature matrix, where δ is the dimension of node features. $A_i \in \mathbb{R}^{n_i \times n_i}$ denotes the adjacency matrix. After the pooling operation on layer k, the number of nodes is changed as $n_i^{(k)}$, and the pooled graph $\mathcal{G}_i^{(k)}$ has the hidden representation matrix $H_i^{(k)} \in \mathbb{R}^{n_i^{(k)} \times d}$ and the adjacency matrix $A_i^{(k)} \in \mathbb{R}^{n_i^{(k)} \times n_i^{(k)}}$, where d is the dimension of hidden representations and $\mathcal{G}_i^{(0)} = \mathcal{G}_i$ and $H_i^{(0)} = X_i$. For SPGP, one or multiple types of prior graph structures can be used. Given a graph $\mathcal{G}_i$, the s-th type of graph structure is denoted as the structure prototype $SP_{s,i} = \{\mathcal{S}_{i,t}^s \mid \mathcal{S}_{i,t}^s \subset \mathcal{V}_i, t = 1 \ldots T_i\}$, where $\mathcal{G}_i$ has T_i graph structures of type s and $\mathcal{S}_{i,t}^s$ is a set of nodes belonging to the t-th graph structure of $SP_{s,i}$.

3.2 Guided Pooling with Structure Prototype Vectors

The core idea of graph pooling is to select the nodes that are important in terms of the graph as a whole. Unlike existing graph pooling methods, SPGP guides the learning process through the structure prototypes SP_s. SPGP learns a *node selector* using structure prototype vectors which constitute the graph structures based on the node representations with contextual information.

3.2.1 Node Representations with Contextual Information

In a graph, the structural information of a node can be enriched with contextual information of the node. Inspired by previous works on message passing network [18], we utilize additional graph convolution networks [22] to obtain node representations with contextual information. A contextual

node embedding $\widetilde{h}_v^{(k)}$ at layer k associated with node v can be formulated as:

$$\widetilde{h}_v^{(k)} = h_v^{(k)} + \text{LeakyReLU}\left(\boldsymbol{W}^{(k)}\text{Concat}(t_v^{(k)}, c_v^{(k)})\right) \tag{1}$$

where $h_v^{(k)} \in H^{(k)}$ and $\boldsymbol{W}^{(k)} \in \mathbb{R}^{2d\times d}$ (a bias is also used but omitted for clarity). The target node representation $t_v^{(k)}$ and the its context node representation $c_v^{(k)}$ can be expressed as:

$$t_v^{(k)} = \sum_{u\in\widehat{\mathcal{N}}_t^{(k)}(v)} \frac{e_{u,v}}{\sqrt{|\widehat{\mathcal{N}}_t^{(k)}(u)||\widehat{\mathcal{N}}_t^{(k)}(v)|}} \boldsymbol{W}_t^{(k)} h_u^{(k)} \tag{2}$$

$$c_v^{(k)} = \sum_{u\in\widehat{\mathcal{N}}_c^{(k)}(v)} \frac{e_{u,v}}{\sqrt{|\widehat{\mathcal{N}}_c^{(k)}(u)||\widehat{\mathcal{N}}_c^{(k)}(v)|}} \boldsymbol{W}_c^{(k)} h_u^{(k)} \tag{3}$$

where $\widehat{\mathcal{N}}_t^{(k)}(v)$ and $\widehat{\mathcal{N}}_c^{(k)}(v)$ denote the direct neighborhood and the context nodes of the node v, respectively, in the self-looped adjacency matrix $\widehat{A}^{(k)} = A^{(k)} + I^{(k)}$. The context nodes can be any k-hop ($k \geq 2$) neighborhood in the graph. In this paper, since global information of the graph is addressed through the structure prototypes, we simply employ the two-hop neighborhood. $e_{u,v}$ denotes edge weight between node u and node v if available, otherwise 1. $\boldsymbol{W}_t^{(k)}, \boldsymbol{W}_c^{(k)} \in \mathbb{R}^{d\times d}$ are learnable parameters.

3.2.2 Node Selection through Structure Prototype Vectors

Given a graph $\mathcal{G}_i^{(k)}$, SPGP learns the node importance by utilizing the structure prototypes $SP_{s,i}$ and the node representation $\widetilde{h}_v^{(k)}$. The global structural information contained in the $SP_{s,i}$ is distributed to each node through the learnable *structure prototype vectors* $Z_{\mathcal{S}_t^s}$, which can be expressed as: (For simplicity, we omit the superscript k and the subscript i as below.)

$$Z_{\mathcal{S}_t^s} = \mathcal{R}\left(\{\widetilde{h}_v \mid v \in \mathcal{S}_t^s\}\right) \tag{4}$$

where $\mathcal{R}$ is an aggregation function that integrates the representation of nodes belonging to $\mathcal{S}_t^s$ of the structure prototype SP_s. $\mathcal{R}$ can be any permutation-invariant methods such as simple average or attention mechanism. For computational efficiency, we simply utilize element-wise max operation to obtain the prototype vectors, i.e., $Z_{\mathcal{S}_t^s,j} = max_{v\in\mathcal{S}_t^s}\widetilde{h}_{v,j}$. In practice, despite the simple aggregator, the information in the global structure is well represented.

Prototype Score Then, using the structure prototype vectors $Z_{\mathcal{S}_t^s}$, a prototype score is calculated as:

$$\phi_{prototype}(v) = \sum_s q_{\theta_s}\left(\{Z_{\mathcal{S}_t^s} \mid v \in \mathcal{S}_t^s\}, \widetilde{h}_v\right) + \boldsymbol{W}_{node}\,\widetilde{h}_v \tag{5}$$

where $\boldsymbol{W}_{node} \in \mathbb{R}^{d\times 1}$ is a learnable parameter for extracting information of the node itself. $q_{\theta_s}(.)$ is relation module of the structure prototype SP_s that learns affinities between the prototype vectors and the node. Specifically, the relation module q_{θ_s} is calculated as:

$$q_{\theta_s}\left(\{Z_{\mathcal{S}_t^s} \mid v \in \mathcal{S}_t^s\}, \widetilde{h}_v\right) = \boldsymbol{W}_s\,\text{Concat}\left(\sum_{z\in Z_v} z,\ \ \widetilde{h}_v\right) \tag{6}$$

where Z_v is denoted as $\{Z_{\mathcal{S}_t^s} \mid v \in \mathcal{S}_t^s\}$. $\boldsymbol{W}_s \in \mathbb{R}^{2d\times 1}$ is a learnable parameter for extracting relationship between the prototypes and the node. If the node v does not belong to the structure prototype SP_s, the relation module q_{θ_s} generates zero, which means no affinity between the node and the structure prototype.

Auxiliary Score Inspired by [10], we utilize neighborhood information gain as an auxiliary score to obtain local structure information. Different from the previous work, we aggregate representations of the neighborhood with a learnable projection function.

$$\phi_{aux}(v) = \|\widetilde{h}_v - \sum_{u\in N_v} \boldsymbol{W}_{aux}\widetilde{h}_u\|_1 \tag{7}$$

$\boldsymbol{W}_{aux} \in \mathbb{R}^{d\times d}$ is a learnable projection that adjusts the discrepancy between the node v and the neighborhood. $\|\cdot\|_1$ is L_1-norm or Manhattan norm.

Total Score Finally, the node score $\phi(v)$ of SPGP is expressed as the prototype score with the auxiliary score to learn the structural information of the node v:

$$\phi(v) = \sigma\left(\phi_{prototype}(v) + \lambda\, \phi_{aux}(v)\right) \tag{8}$$

where σ denotes the activation function such as sigmoid or ReLU. $\lambda \in [0, 1]$ is the regularization for the auxiliary score.

3.2.3 Generation of Pooled Graph

Based on the node scores $\phi(.)$, SPGP generates the pooled graph $\mathcal{G}_i^{(k+1)}$ at layer $k+1$ as:

$$\text{idx} = \text{Top-K}(\phi(v^{(k)}), p) \tag{9}$$

$$H^{k+1} = H^k\,[\text{idx},\ :] \odot \phi(.), \quad A^{k+1} = A^k\,[\text{idx},\ \text{idx}] \tag{10}$$

where $p \in (0, 1]$ is the pooling ratio. Top-K [9] selects the top $\lceil pn_i^{(k)} \rceil$ nodes based on the score ϕ. The idx is indices of selected nodes for the pooled graph. $\odot$ is broadcasted hadamard product. Note that, the end product of node scores $\phi(.)$ is a discrete selection of nodes, which is not trainable by back-propagation. Therefore, the score $\phi(.)$ is multiplied on hidden representations to make the SPGP learnable in an end-to-end fashion.

3.3 Well-studied Structure Prototypes: BCC and Cliques

SPGP can use any graph structure related to global information of the graph. In this paper, as an example of the structure prototypes, we utilize two well-studied structures related to the overall graph topology in graph theory: *Biconnected Component (BCC)* and *Cliques*.

Given a graph $\mathcal{G}_i$, a set of biconnected components $SP_{BCC,i}$ and a set of cliques $SP_{CQ,i}$ are obtained by the graph traversal algorithms [17, 50]. Specifically, for biconnected components, we discard the biconnected components with fewer than three nodes to avoid simple biconnected components, i.e., single edges. Then, we define the structure prototype for biconnected components of $\mathcal{G}_i$ as:

$$SP_{BCC,i} = \{\mathcal{S}_{i,p}^{BCC} \mid \mathcal{S}_{i,p}^{BCC} \subset \mathcal{V}_i, |\mathcal{S}_{i,p}^{BCC}| \geq 3, p = 1 \ldots P_i\} \tag{11}$$

where $\mathcal{G}_i$ has P_i biconnected components and $\mathcal{S}_{i,p}^{BCC}$ is a set of nodes belonging to the p-th biconnected component of $\mathcal{G}_i$.

For cliques, we also discard cliques with less than three nodes. A set of cliques can characterize the overall topology of the graph [33], so if adjacent cliques share more than half of the nodes, they are merged. Then, we define the structure prototype for cliques of $\mathcal{G}_i$ as:

$$SP_{CQ,i} = \{\mathcal{S}_{i,q}^{CQ} \mid \mathcal{S}_{i,q}^{CQ} \subset \mathcal{V}_i, |\mathcal{S}_{i,q}^{CQ}| \geq 3, q = 1 \ldots Q_i\} \tag{12}$$

where $\mathcal{G}_i$ has Q_i the merged cliques and $\mathcal{S}_{i,q}^{CQ}$ is a set of nodes belonging to the q-th cliques of $\mathcal{G}_i$.

The time complexity of finding BCC is $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$. Cliques can be efficiently searched on each biconnected component, not the entire graph [15]. In addition, graph traversal is performed only once during preprocessing, not for model training.

4 Theoretical and Computational Analysis

4.1 SPGP can discriminate structural role of nodes

We investigate the theoretical understanding on the importance of learning structural information. Following existing studies on the importance of graph structure [44, 28], we assume that all nodes have the same feature vector, which means that the only information we can learn from the graph is structural information. In general, nodes belonging to a local level subgraph tend to have same or similar feature vectors due to homophily [26, 7]. In this case (e.g., naphthalene), it is difficult to learn the structural role between nodes in the subgraph using node features. Note that if the structural meaning of the nodes on the graph is different, efficient graph pooling should be able to identify the difference in the amount of information of each node.

Theorem 1. *Given a regular graph that has prior graph structures,*

(i) Standard k-hop neighborhood based graph pooling methods cannot distinguish the nodes.

(ii) SPGP can assign different node scores based on the structural role of nodes.

Proof. See Appendix B.1 for proof. □

Motivated by the above analysis, we believe that providing the pooling with prior graph structures beyond k-hop neighborhood can better learn the structural information of the graph. Therefore, to extract and model the structure prototype information in an end-to-end fashion, we formulate SPGP based on the learnable structure prototype vectors.

To further support the observation in more general situations, we measure the diversity of node scores in the regular graphs with degree perturbations. That is, we generate more realistic graphs with structural diversity from regular graphs (See Appendix B.2 for details). Figure 2 shows that, as in Theorem 1, SPGP learns the structural roles of nodes in regular graphs, but the other pooling methods have failed. Furthermore, as the degree perturbation increases, that is, as the distribution of node degree on the graph diversifies, SPGP captures the structural role of nodes better than the other methods.

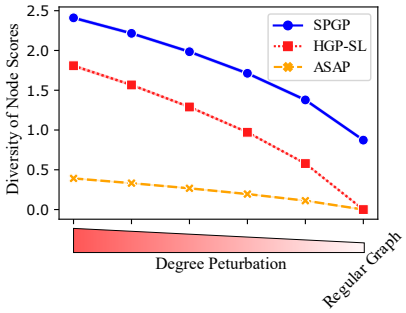


Figure 2: Node Score's diversity of SPGP, HGP-SL, and ASAP on regular graphs with degree perturbations.

4.2 Efficient Handling of Isolated Nodes

SPGP is a node selection based graph pooling which can cause isolated nodes during the pooling. Since isolated nodes cannot aggregate the information of neighbors, they may result in loss of graph structural information and degenerate graph classification performances. To address this issue, the existing graph pooling methods, such ASAP [34] and HGP-SL [51], perform 'structure learning' that refines an adjacency matrix of a pooled graph. However, this operation is an additional task performed after graph pooling and it is computationally expensive. The computational complexity of ASAP is $O(n^3)$ time at a cost of $O(n^2)$ space. HGP-SL is $O(dn^2)$ time at a cost of $O(n^2)$ space. Meanwhile, our SPGP does not require an additional step for addressing the isolated nodes. During the graph pooling, Equation 6 aggregates and updates structural information of nodes belonging to structure prototypes, even if they are isolated. Furthermore, the computational complexity of Equation 6 is $O(sdn)$ time at a cost of $O(dn)$ space, $n \gg s, d$, which is more efficient for the large-scale graphs. Total computational complexity of SPGP is discussed in Appendix A.3.

5 Experiment Setup

5.1 Datasets

For fair comparisons, we use the widely used benchmark datasets for graph classification. Specifically, datasets from biochemical domains of TUDataset [27] (DD, PROTEINS, NCI1, NCI109, and FRANKENSTEIN; ranging from 1K to 4K graphs) are utilized for performance evaluation. Furthermore, two large-scale benchmark datasets (ogbg-molhiv with 41.1K graphs and ogbg-molpcba with 437.9K graphs) are utilized from Open Graph Benchmark (OGB) [19]. Meanwhile, in real-world datasets, especially in the biochemical domains, most graphs contain more than one BCC or clique. For example, in all datasets except FRANKENSTEIN, more than 97% of graphs have these structures (DD: 100%, PROTEINS: 99.9%, NCI1: 98.0%, NCI109: 97.6%, ogbg-molhiv: 100%, ogbg-molpcba: 99.9%). Even in the chimera dataset, FRANKENSTEIN, the ratio is 82.6%. Detailed information about the datasets is described in the Appendix C.1.

5.2 Baselines

We compare the performance of SPGP with state-of-the-art global pooling (i.e., Set2Set [38], Global-Attention [24], and SortPool [49])) or hierarchical graph pooling methods (i.e., DiffPool [43],

Table 1: Test Accuracy (%) performance on TUDataset. Average accuracy and standard deviation is reported for 20 random seeds. The best result and its comparable results ($p > 0.05$) from t-test for each dataset are shown in bold. The second best results on each category are shown as underlined. Compared with the previous graph pooling methods, the proposed SPGP outperforms the baselines.

Category	Methods	DD	PROTEINS	NCI1	NCI109	FRANKENSTEIN
Baseline	GCN	68.33 ± 1.30	73.83 ± 0.33	73.92 ± 0.43	72.77 ± 0.57	60.47 ± 0.62
Global	Set2Set	71.60 ± 0.87	72.16 ± 0.43	66.97 ± 0.74	61.04 ± 2.69	61.46 ± 0.47
	Global-Attention	71.38 ± 0.78	71.87 ± 0.60	$\underline{69.00 \pm 0.49}$	67.87 ± 0.40	61.31 ± 0.41
	SortPool	$\underline{71.87 \pm 0.96}$	$\underline{73.91 \pm 0.72}$	68.74 ± 1.07	$\underline{68.59 \pm 0.67}$	$\underline{63.44 \pm 0.65}$
Hierarchical	DiffPool	66.95 ± 2.41	68.20 ± 2.02	62.32 ± 1.90	61.98 ± 1.98	60.60 ± 1.62
	TopK	75.01 ± 0.86	71.10 ± 0.90	67.02 ± 2.25	66.12 ± 1.60	61.46 ± 0.84
	SAGPool	76.45 ± 0.97	71.86 ± 0.97	67.45 ± 1.11	67.86 ± 1.41	61.73 ± 0.76
	ASAP	$\underline{76.87 \pm 0.70}$	$\underline{74.19 \pm 0.79}$	71.48 ± 0.42	70.07 ± 0.55	$\underline{66.26 \pm 0.47}$
	GSAPool	76.07 ± 0.81	73.53 ± 0.74	70.98 ± 1.20	70.68 ± 1.04	60.21 ± 0.69
	HGP-SL	75.16 ± 0.69	74.09 ± 0.84	75.97 ± 0.40	74.27 ± 0.60	63.80 ± 0.50
	iPool$_{greedy}$	75.39 ± 0.76	73.12 ± 0.55	76.03 ± 0.60	$\underline{74.96 \pm 0.58}$	62.16 ± 0.35
	iPool$_{local}$	75.22 ± 0.70	73.56 ± 0.59	$\underline{76.49 \pm 0.62}$	74.64 ± 0.39	62.24 ± 0.55
	SPGP$_{BCC}$	76.01 ± 0.92	74.10 ± 0.66	77.36 ± 0.31	76.89 ± 0.35	66.86 ± 0.67
	SPGP$_{CQ}$	76.77 ± 0.80	$\mathbf{75.04 \pm 0.76}$	78.57 ± 0.34	$\mathbf{77.23 \pm 0.48}$	67.72 ± 0.51
	SPGP$_{BCC+CQ}$	$\mathbf{77.76 \pm 0.73}$	$\mathbf{75.20 \pm 0.74}$	$\mathbf{78.90 \pm 0.27}$	$\mathbf{77.27 \pm 0.32}$	$\mathbf{68.92 \pm 0.71}$

TopK [9], SAGPool [23], ASAP [34], GSAPool [47], HGP-SL [51], and iPool [10]). We report the results, except for GSAPool, HGP-SL, iPool$_{greedy}$, and iPool$_{local}$, from [34] and measure the performance of GSAPool and HGP-SL based on the released source codes. We implement both of iPool variants by modifying the code of HGP-SL. Brief explanations of the hierarchical graph pooling methods are described in Appendix C.2

5.3 Experimental Details

In case of TUDataset, following [34], we adopt the extensive experimental designs for performance evaluation. In short, the performances are averaged over the results based on 20 different random seeds. In addition, on each seed, 10-fold cross-validation is performed and data split configurations are the same in [34]. In case of OGB datasets, we utilize the scaffold splitting scheme and report the average results on the 20 different random seeds. Hyperparameters of SPGP are selected by grid search on the validation data. Settings of hyperparameter values are shown in the Appendix C.3.

6 Results

This section attempts to answer the following questions:

- **Q1** Whether can prior graph structures improve graph classification performances of SPGP compared to baselines? (Section 6.1)
- **Q2** How does a regularization λ affect the performances over various datasets? (Section 6.2)
- **Q3** Can SPGP capture meaningful structures during pooling? (Section 6.3)
- **Q4** How does SPGP perform compared to baselines on large-scale datasets? (Section 6.4)

6.1 Performance Comparison on TUDataset (Q1)

Table 1 reports the experimental results of SPGP with baseline models on TUDataset. From the table, we find that our SPGP$_{BCC+CQ}$ significantly outperforms all baselines in terms of graph classification accuracy. SPGP$_{BCC+CQ}$ is up to 9.90% and 2.66% better than the second-best method in global pooling and hierarchical pooling baselines, respectively. We notice that the guided pooling using prior graph structures based on structure prototypes, in SPGP, is the main contribution to performance improvements against baselines.

To further investigate the effects of structure prototypes, we conduct ablation studies on SPGP with only one structure prototype. (Due to space limit, an ablation study for without structure prototypes is discussed in Appendix C.4.) In Table 1, SPGP$_{BCC}$ and SPGP$_{CQ}$ are variants of SPGP that use only BCCs and cliques respectively. Interestingly, both variants of SPGP achieve better classification

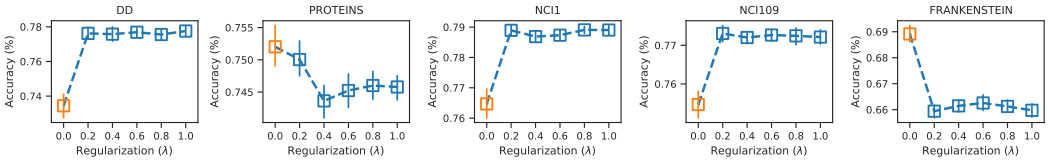
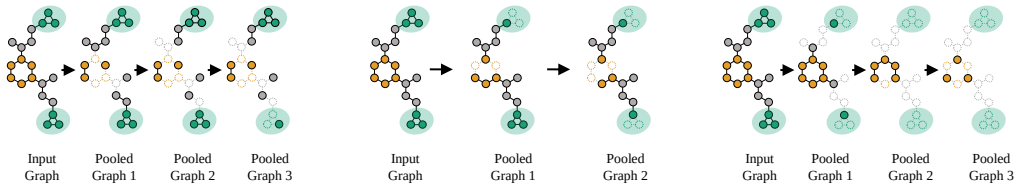


Figure 3: Performance of SPGP on the TUDataset by varying the regularization parameter (λ). Average accuracy and standard deviation for 20 random seeds are shown as box plots. The orange box plot denotes $\lambda = 0.0$. The blue box plots denote $\lambda \geq 0.2$.



(a) SPGP (b) HGP-SL (c) ASAP

Figure 4: Visualization of different pooling methods. The input graph is sampled from FRANKENSTEIN dataset. Orange nodes represent the nodes belong to a biconnected component. Green nodes represent the nodes belong to both a biconnected component and a clique. We omit newly added edges in HGP-SL and ASAP for better visibility.

performances than all baselines except ASAP on DD dataset, which is also comparable. This means that providing additional graph structural information to the model during the graph pooling is useful for learning more accurate graph-level representations. Also, since the set of cliques can be either the BCC itself or a part of the BCC, the two graph structures can be used together as structure prototypes to provide nodes with more valuable structural information. For this reason, SPGP$_{BCC+CQ}$ achieves better performances than SPGP$_{BCC}$ and SPGP$_{CQ}$.

6.2 Effects of Regularization λ (Q2)

We investigate the effects of regularization λ on the performance of TUDataset. We use the SPGP$_{BCC+CQ}$ model and the λ in the range 0.0, 0.2, 0.4, 0.6, 0.8, 1.0. Figure 3 shows that the need for ϕ_{aux} depends on the dataset. In PROTEINS and FRANKENSTEIN, consideration of ϕ_{aux}, i.e., $\lambda \geq 0.2$, negatively affects the performance. To further support our observation, we investigate the closeness centrality of graphs in the datasets (See Supplementary Figure 1 and Appendix C.6). Interestingly, the graphs from PROTEINS and FRANKENSTEIN have a larger average of closeness centrality than the other datasets. Note that a node with a higher closeness centrality has the shortest distance to all other nodes, so the node's information propagates more easily throughout the graph. For this reason, ϕ_{aux} that considers information of local neighborhood seems to degrade the performance. In summary, the effects of ϕ_{aux} and the regularization λ may depend on graph properties such as closeness centrality.

6.3 Visualization: What Structure Does SPGP Preserve? (Q3)

To answer the question, we visualize the effects of different pooling strategies of SPGP, HGP-SL, and ASAP. We randomly sample a graph, which has mutagenicity, from FRANKENSTEIN dataset. As shown in Figure 4, the chemical compound contains one aromatic ring (orange) and two functional groups of three-membered heterocycles (green) such as oxirane or aziridine. These two reactive groups are known as toxicophores, which are closely related to the mutagenic activity of the chemical compounds [20]. Figure 4(a) shows that SPGP well preserves both of the informative structures after pooling. Specifically, since the three-membered heterocycles are cliques as well as BCCs, SPGP potentially learns the importance of the structures through the guidance of structure prototypes. On the other hand, in Figure 4(b) and 4(c), HGP-SL and ASAP try to preserve graph topology information through structure learning, but they relatively focus on carbon chains and an aromatic ring, respectively.

Table 2: Performances on OGB dataset. Evaluation metrics for ogbg-molhiv and ogbg-molpcba are ROC-AUC (%) and average precision (%), respectively. Average metric and standard deviations for 20 random seeds are reported. The best result and its comparable results ($p > 0.05$) from t-test are shown in bold. GCN denotes reported results in the leader-board. GCN† denotes reproduced results in the 20 random seeds. OOM denotes out-of-memory error.

Methods	ogbg-molhiv	ogbg-molpcba
GCN	76.06 ± 0.97	20.20 ± 0.24
GCN†	75.86 ± 1.15	20.18 ± 0.22
HGP-SL	72.46 ± 1.55	18.64 ± 0.28
ASAP	76.89 ± 1.90	OOM
SPGP	$\mathbf{78.15 \pm 0.83}$	$\mathbf{23.95 \pm 0.97}$

6.4 Scalability on large-Scale OGB datasets (Q4)

To answer the question, we compare SPGP with two hierarchical pooling methods ASAP and HGP-SL on ogbg-molhiv and ogbg-molpcba from the large-scale OGB dataset. For fair comparisons, we experiment with SPGP, ASAP, and HGP-SL in the same setting (See Appendix C.3 for details). In both datasets, SPGP outperforms all baselines including the simple GCN model [22] without pooling operations. Rather, unlike HGP-SL, which degrades performances through pooling operations, SPGP can extract meaningful structures on the large-scale datasets. For ASAP, which is the best competitor in TUDataset, it shows performance improvements through pooling on ogbg-molhiv, but with out-of-memory errors on the larger dataset, ogbg-molpcba. Due to the efficiency of SPGP, it can be jointly used with advanced graph convolution layers such as GCN/GIN with virtual node and the results are discussed in Appendix C.7

To support this observation, we evaluate the GPU memory footprint of SPGP with the baselines on the graphs generated by two random graph models: Erdős–Rényi [12] and Barabási–Albert [1] (See Appendix C.5 for details). Figure 5 shows that SPGP is very effective and scalable compared to other methods in terms of memory usage. This is, as shown in Section 4.2, because ASAP and HGP-SL additionally learn the structure of the pooled graph to extract meaningful nodes, whereas SPGP learns global structural information with fewer operations through structure prototype vectors.

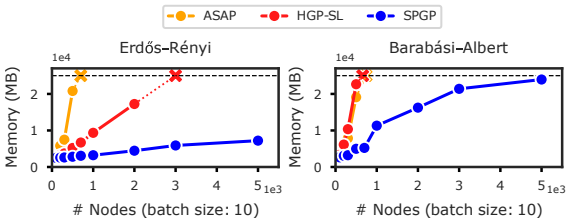


Figure 5: Memory footprint of SPGP compared with baselines. Memory usage is measured for graphs of various sizes generated from two random graph models. Black dashed line represents maximum memory of GPU. X denotes out-of-memory error.

7 Conclusions

Limitations There are potential limitations of SPGP that would be an important direction for future work. (i) Isolated nodes that do not belong to the structure prototypes are hard to address by SPGP. (ii) Given prior graph structures, SPGP is limited to utilizing information such as size and specific shape based on only node membership information. Other structural information may be useful. (iii) Learning the interactions between different types of structure prototypes remains a challenge.

Summary We propose a novel graph pooling method, SPGP, for graph-level classification. The key to success of SPGP is the consideration of informative global graph structures to obtain accurate graph-level representations. The structure prototypes guide the node scoring scheme during pooling. Experiments on multiple datasets and ablation studies show that utilization of the prior graph structures achieves better performance than existing pooling methods. In this study, two well-known structures in graph theory, BCC and cliques are utilized as the prior graph structures. In future research, SPGP can be easily extended to utilize other graph structures inspired by domain knowledge, e.g., network biomarkers for interpreting diseases and chemical building blocks in molecular graph generation.

References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[2] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 (suppl_1):i47–i56, 2005.

[3] Anna D Broido and Aaron Clauset. Scale-free networks are rare. *Nature communications*, 10 (1):1–10, 2019.

[4] National Research Council et al. *Mathematical challenges from theoretical/computational chemistry*. National Academies Press, 1995.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[6] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[7] Anna Evtushenko and Jon Kleinberg. The paradox of second-order homophily in networks. *Scientific Reports*, 11(1):1–10, 2021.

[8] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[9] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pages 2083–2092. PMLR, 2019.

[10] Xing Gao, Wenrui Dai, Chenglin Li, Hongkai Xiong, and Pascal Frossard. ipool–information-based pooling in hierarchical graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[11] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pages 3419–3430. PMLR, 2020.

[12] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[15] Dorit S Hochbaum. Why should biconnected components be identified first. *Discrete applied mathematics*, 42(2-3):203–210, 1993.

[16] Seok-Hee Hong, Quan Nguyen, Amyra Meidiana, Jiaxi Li, and Peter Eades. Bc tree-based proxy graphs for visualization of big graphs. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 11–20. IEEE, 2018.

[17] John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[18] W Hu, B Liu, J Gomes, M Zitnik, P Liang, V Pande, and J Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[19] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[20] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1):312–320, 2005.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[23] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International Conference on Machine Learning*, pages 3734–3743. PMLR, 2019.

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[25] Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 723–731, 2019.

[26] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[27] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

[28] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020.

[29] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2019.

[30] Francesco Orsini, Paolo Frasconi, and Luc De Raedt. Graph invariant kernels. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[31] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6:1–38, 2017.

[32] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

[33] Giovanni Petri, Martina Scolamiero, Irene Donato, and Francesco Vaccarino. Topological strata of weighted complex networks. *PloS one*, 8(6):e66506, 2013.

[34] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

[35] Zhenzhen Shao, Na Guo, Yu Gu, Zhigang Wang, Fangfang Li, and Ge Yu. Efficient closeness centrality computation for dynamic graphs. In *International Conference on Database Systems for Advanced Applications*, pages 534–550. Springer, 2020.

[36] Sriganesh Srihari, Chern Han Yong, and Limsoon Wong. *Computational prediction of protein complexes from protein interaction networks*. Morgan & Claypool, 2017.

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[38] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*, 2015.

[39] Nikil Wale and George Karypis. Acyclic subgraph based descriptor spaces for chemical compound retrieval and classification. Technical report, MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE, 2006.

[40] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[41] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[42] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[43] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4805–4815, 2018.

[44] Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning*, pages 7134–7143. PMLR, 2019.

[45] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[46] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34, 2021.

[47] Liang Zhang, Xudong Wang, Hongsheng Li, Guangming Zhu, Peiyi Shen, Ping Li, Xiaoyuan Lu, Syed Afaq Ali Shah, and Mohammed Bennamoun. Structure-feature based graph self-adaptive pooling. In *Proceedings of The Web Conference 2020*, pages 3098–3104, 2020.

[48] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[49] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[50] Yun Zhang, Faisal N Abu-Khzam, Nicole E Baldwin, Elissa J Chesler, Michael A Langston, and Nagiza F Samatova. Genome-scale computational approaches to memory-intensive applications in systems biology. In *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, pages 12–12. IEEE, 2005.

[51] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.

A Algorithm

A.1 Overall Structure of SPGP

The overall structure of SPGP is described in Algorithm 1.

Algorithm 1 Overall Structure of SPGP

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; Adjacency matrix A; Node feature matrix X; Pooling operation SPGP; Structure Prototype SP; Number of layers L, readout function Readout
Output: A predicted graph label $\widehat{y}$

$X_{aug} \leftarrow FA(X, SP)$ // Add global structure related features (Section A.2)
$H^1 \leftarrow$ LayerNorm(GCN(X_{aug}, A))
$H^1, A^1 \leftarrow$ SPGP(H^1, A, SP)
$HS \leftarrow$ Readout(H^1)
for i=2...L **do**
  $H^{i+1} \leftarrow$ LayerNorm(GCN(H^i, A^i))
  $H^{i+1}, A^{i+1} \leftarrow$ SPGP(H^i, A^i, SP)
  $HS \leftarrow HS +$ Readout(H^{i+1})
end for
out $\leftarrow$ MLP(HS) // Multi-Layer Perceptron
$\widehat{y} \leftarrow argmax$(out)
return $\widehat{y}$

Algorithm 2 Graph Pooling of SPGP($H^{(k)}$, $A^{(k)}$, SP, p)

$\widetilde{H}^{(k)} \leftarrow H^{(k)} +$ LeakyReLU $\left(\boldsymbol{W}^{(k)}\text{Concat}(T^{(k)}, C^{(k)})\right)$ according to Equation 1
Calculate $\phi_{prototype}$ according to Equation 5
Calculate ϕ_{aux} according to Equation 7
$\phi(v) \leftarrow \sigma\left(\phi_{prototype}(v) + \lambda\, \phi_{aux}(v)\right)$
idx $\leftarrow$ Top-K($\phi(v^{(k)}), p$)
$H^{(k+1)} \leftarrow H^{(k)}$ [idx, :] $\odot\, \phi(.)$
$A^{(k+1)} \leftarrow A^{(k)}$ [idx, idx]
return $H^{(k+1)}$, $A^{(k+1)}$

A.2 Augmentation of structure prototype related features

As another aspect of utilizing structure prototypes, we augment node features from the graph itself. The node features enhanced by the structure prototypes are expected to help in the pooling operation. Given a node v of a graph $\mathcal{G}_i$, the augmented node features are defined as:

- Size of biconnected components $\delta_1 : \max_{p \in N_v^{bcc}} \frac{|\mathcal{S}_{i,p}^{BCC}|}{|\mathcal{S}_{i,max}^{BCC}|}$
- Size of cliques $\delta_2 : \max_{q \in N_v^{cq}} \frac{|\mathcal{S}_{i,q}^{CQ}|}{|\mathcal{S}_{i,max}^{CQ}|}$
- Number of cliques $\delta_3 : \frac{|N_v^{cq}|}{|\mathcal{S}_i^{CQ}|}$

where $\mathcal{S}_{i,max}^{BCC}$ is the largest biconnected components in the graph $\mathcal{G}_i$. $\mathcal{S}_{i,max}^{CQ}$ is the maximum cliques in the graph $\mathcal{G}_i$. N_v^{bcc} and N_v^{cq} are biconnected components and cliques containing node v, respectively. This augmentation can be easily extended to other prior graph structures.

A.3 Computational Complexity

For a single layer of vanilla GCN, computational complexity is $O(md + nd^2)$. SPGP consists of four steps: (1) contextual node embedding, (2) prototype score, (3) auxiliary score, (4) topk pooling.

Assume that $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, $|\mathcal{V}| = n$, $|\mathcal{E}| = m$, $\mathcal{X} \in \mathbb{R}^{n \times d}$ at layer l, computational complexity of each step is as follow,

1. Node Presentation with Contextual Information - Equation 1: We calculates the contextual node embedding with right-to-left multiplication. Since adjacency matrix A is stored as a sparse matrix, the computational complexity is $O(md + nd^2)$. It matches the computational complexity of the vanilla GCN.
2. Prototype Score - Equation 5: The prototype score consists of two terms: relation module "q" and node's self-information. The computational complexity is $O(sdn + dn)$, where s is the number of prototypes and $s, d << n$.
3. Auxiliary Score - Equation 7: This procedure requires aggregation of neighborhood information. So, computational complexity is $O(md + nd^2)$.
4. Top-k pooling - Equation 9 and 10: This procedure filters the node feature matrix and graph structure by selected nodes. It takes $O(n + m)$.

In summary, the computational complexity of SPGP is $O(md + nd^2 + sdn)$. With L layers graph neural network with GCN and SPGP with pooling ratio p, total computational complexity is $O(\frac{1-p^L}{1-p}(md + nd^2 + sdn)) \approx O(L(md + nd^2 + sdn)$.

B Proof

B.1 Proof for Theorem 1

We assume that all nodes have the same feature vector, which means that the only information we can learn from the graph is structural information. We first prove that the standard GNN generates the same embedding for every node in a regular graph (Lemma 1). After that, we show that the SPGP can learn the structure role of nodes in a regular graph, but not the standard k-hop neighborhood based graph pooling methods (Theorem 1).

Lemma 1. *The standard GNN generates the same embedding for every node in a regular graph.*

Proof. The standard GNN updates node representations by following equations.

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} \mid u \in N(v)\})$$
$$h_v^{(k)} = \text{MERGE}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

Given a regular graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we show for an arbitrary iteration k and nodes $v_1, v_2 \in \mathcal{V}, v_1 \neq v_2$ that $h_{v_1}^{(k)} = h_{v_2}^{(k)}$. In iteration $k = 0$, $h_{v_1}^{(0)} = h_{v_2}^{(0)}$ is hold since all nodes are the same feature vector. Assume for induction that $h_{v_1}^{(k-1)} = h_{v_2}^{(k-1)}$ for arbitrary nodes v_1 and v_2. Let $a_{v_1}^{(k)} = \text{AGGREGATE}^{(k)}(\{h_{u_1}^{(k-1)} \mid u_1 \in N(v_1)\})$ and $a_{v_2}^{(k)} = \text{AGGREGATE}^{(k)}(\{h_{u_2}^{(k-1)} \mid u_2 \in N(v_2)\})$ be the aggregated messages from neighbors of v_1 and v_2. By the induction hypothesis, we know that $\{h_{u_1}^{(k-1)} \mid u_1 \in N(v_1)\} = \{h_{u_2}^{(k-1)} \mid u_2 \in N(v_2)\}$ and that $a_{v_1}^{(k)} = a_{v_2}^{(k)}$. Therefore, the input of $\text{MERGE}^{(k)}$ is identical in v_1 and v_2. This proves that $h_{v_1}^{(k)} = h_{v_2}^{(k)}$, and there by the lemma. □

Theorem 1. *Given a regular graph that has prior graph structures,*

(i) Standard k-hop neighborhood based graph pooling methods cannot distinguish the nodes.

(ii) SPGP can assign different node scores based on the structural role of nodes.

Proof. For the standard k-hop neighborhood based graph pooling methods, node representations for measuring the node score are based on the standard GNN. By Lemma 1, in the regular graph, all node representations are the same. Therefore, the existing pooling methods cannot differentiate between nodes, even if they aggregate k-hop neighbor information, since the amount of information they collect is the same. This proves the first part of the theorem. We prove that the second part by showing that Equation 5 and 6 can learn the additional information of nodes belonging to the prior graph structures. The structural roles contained in the given structure prototypes can be reflected in the nodes through the equations. Hence, SPGP can assign different node scores based on the given structural information of the structure prototypes. □

Supplementary Table1: Summary statistics of datasets. $\mathcal{V}_{avg}, \mathcal{E}_{avg}$ are the average number of nodes and edges per graph, respectively.

	Dataset	$\#Graphs$	$\#Classes$	$\mathcal{V}_{avg}$	$\mathcal{E}_{avg}$	$\#Tasks$
TUDataset	DD	1,178	2	284.3	715.7	1
	PROTEINS	1,113	2	39.1	72.8	1
	NCI1	4,110	2	29.9	32.3	1
	NCI109	4,127	2	29.7	32.1	1
	FRANKENSTEIN	4,337	2	16.9	17.9	1
OGB Dataset	ogbg-molhiv	41,127	2	25.5	27.5	1
	ogbg-molpcba	437,929	2	26.0	28.1	128

Supplementary Table 2: Preprocessing time for SPGP.

	Dataset	$\#Graphs$	Total Time (s)	Time per graph (s)
TUDataset	DD	1,178	97	0.0823
	PROTEINS	1,113	9	0.0081
	NCI1	4,110	12	0.0029
	NCI109	4,127	12	0.0029
	FRANKENSTEIN	4,337	292	0.0673
OGB Dataset	ogbg-molhiv	41,127	114	0.0027
	ogbg-molpcba	437,929	1,163	0.0027

B.2 Experiment on regular graphs with degree perturbations

To further support Theorem 1, we measure the diversity of node scores in the regular graphs with degree perturbations. We generate 100 random 6-regular graphs with 100 nodes. Then, for each graph, it goes through the process of perturbing the degree of nodes by changing the existing edge to another non-existent edge, i.e., edge rewiring. That is, starting with the regular graph, the degree distribution on the graph is diversified to give a difference in the amount of information around the k-hop neighborhood. As shown in Figure 2, in the regular graphs, only SPGP can capture node diversity. In addition, as the degree of perturbation progresses, the existing methods such as ASAP and HGP-SL also have distinguishing power of nodes, but are not as good as SPGP.

C Experimental Details

C.1 Datasets Details

Summary statistics for the datasets are shown in Supplementary Table 1.

TUDataset DD [6] and PROTEINS [2] are datasets of protein structures labeled according to whether they are enzymes or non-enzymes. The main difference between the two datasets is that the nodes are amino acids and protein secondary structures, respectively. NCI1 and NCI109 [39] are datasets of chemical compounds labeled according to whether they have activity against non-small cell lung cancer cell line and ovarian cancer cell line, respectively. The nodes in both datasets are one-hot encodings of atoms. FRANKENSTEIN [30] is a chimeric dataset of Mutagenicity and MNIST. The mutagenic effects of the chemical compounds are measured by Ames test (Salmonella typhimurium reverse mutation assay) and the most frequent atom symbols are remapped as high dimensional MNIST digit vectors. Compared to NCI1 and NCI109, the unique information of an atom is lost because different images with the same number are used for the same atom (e.g., many different MNIST images “1”).

OGB Dataset The two datasets "ogbg-molhiv" and "ogbg-molpcba" are chemical compound datasets for predicting molecular properties. They are largest datasets in MoleculeNet [40]. The performance of "ogbg-molhiv" is measured by ROC-AUC to predict whether a molecule inhibits

HIV virus replication. "ogbg-pcba" is a curated dataset from PubChem BioAssay (PCBA) providing biological activities of small molecules. The performance of "ogbg-pcba" is measured with average precision because the dataset contains very few positive labels, i.e., extremely biased (positive rate 1.40%) and various tasks (128 tasks).

Preprocessing Time of Structure Prototypes for SPGP The exploration of BCCs and cliques, which are the candidate structure prototypes of SPGP, only needs to be performed once in the preprocessing step, so they do not have a significant impact on model training. Each graph for preprocessing takes approximately 0.002 seconds to 0.08 seconds, and detailed information is described in Supplementary Table 2.

C.2 Brief Explanations of Hierarchical Graph Pooling Methods

Most of hierarchical graph pooling methods are node-selection based graph pooling, which requires an efficient node selection strategy. To select important nodes, node features and graph structure are needed to compute the node score. From this point of view, SPGP and the existing methods except for DiffPool, which is a node clustering approach rather than node selection, can be summarized as follows and Supplementary Table 3:

- TopK [9]: TopK utilizes only node features. Given a node representation matrix $H \in \mathbb{R}^{n \times d}$, TopK computes node scores with a trainable weight matrix $W \in \mathbb{R}^{d \times 1}$ by $\sigma(HW)$, where σ is a sigmoid function.
- SAGPool [23]: Unlike TopK, SAGPool utilizes the amount of information of the node itself and the information of neighboring nodes. To consider node features and graph structure, SAGPool computes node scores by using an additional graph convolution.
- GSAPool [47]: GSAPool separately measures node features and graph structural information. Node importance is calculated by a trainable weight matrix. Structural importance is calculated by an additional graph convolution. Then, node scores are then computed by fusing the two importance via a weighted sum.
- ASAP [34]: ASAP points out that, since previous methods use information from neighboring nodes when calculating node scores, neighborhoods around nodes with a large amount of information are also redundantly selected. To address this problem, ASAP proposes a new graph convolution called LEConv (Local Extrema Convolution). ASAP also performs structure learning to preserve graph connectivity as node selection can produce isolated nodes
- iPool [10]): Similar to ASAP, iPool also focuses on the relationship between a node and adjacent nodes. iPool computes node scores by L1-norm of the difference between the node representation and the summation of neighborhoods. Therefore, iPool is a parameter-free node selection strategy.
- HGP-SL [51]: HGP-SL is an extension of iPool. The node scores are calculated in the same way as iPool. After that, HGP-SL performs sparse softmax-based structure learning, which requires a lot of computational overhead to deal with isolated nodes.
- SPGP (ours): Existing node selection strategies used graph structure information, but are limited to local information. Therefore, we propose, SPGP, a novel node selection method utilizing prior graph structures related to the global structure. In addition, we try to alleviate the isolated node problem without heavy structure learning by enabling information exchange between nodes belonging to the same structure through the structure prototype.

In summary, the main difference between SPGP and existing methods is that prior graph structures are utilized in the node selection strategy by learnable structure prototypes. As shown in Table 1 and 2 of the main text, SPGP outperforms the other node selection strategies in the same GCN and TopK-based hierarchical pooling framework.

C.3 Training & Hyperparameters

TUDataset For all our experiments of SPGP on TUDataset, Adam [21] optimizer with l_2 regularizer of $5e-4$ is used. With different 20 seeds, on each seed, 10-fold cross-validation is performed with

Supplementary Table 3: Properties of node-selection based graph pooling methods.

Method	Node Feature	Structure Information		Handling Isolated Nodes
		Local	Global	
TopK	✓			
SAGPool	✓	✓		
GSAPool	✓	✓		
ASAP	✓	✓		✓
iPool	✓	✓		
HGP-SL	✓	✓		✓
SPGP (ours)	✓	✓	✓	✓

Supplementary Table 4: Hyperparameter tuning Summary for TUDataset.

Hyperparameter	Range
Learning rate	{0.1, 0.01, 0.005, 0.004, 0.003, 0.002, 0.001}
Batch size	{64, 128, 256}
Hidden dimension	{32, 64, 128, 256}
Number of layers	{2, 3, 4, 5, 6}
Pooling ratio	{0.5, 0.8, 0.9, 0.95}
Dropout rate	{0.0, 0.1, 0.2}
Regularization λ	{0.0, 0.2, 0.4, 0.6, 0.8, 1.0}
Learning rate Decaying Step	{25, 50}

80% for training and 10% for validation and test each. Models are trained for 100 epochs with stepLR scheduler (lr decay of 0.1 after every 25 epochs). The best hyperparameters are selected from a predefined range (Supplementary Table 4) and are provided in Supplementary Table 5. The model with the best validation accuracy is selected for testing. Our models are implemented based on PyTorch Geometric [8]. The experiments are conducted on a Linux server with 126GB memory, 40 CPUs, a single NVIDIA GeForce RTX 3090 or RTX 2080 Ti.

OGB Dataset For all our experiments of SPGP and baselines, Adam [21] optimizer is used. With different 20 seeds, we use the scaffold splitting scheme, which is the standard data split scheme provided by OGB dataset. Models are trained for 100 epochs with early stopping (patience = 30). The selected hyperparameters are described in Supplementary Table 6. In case of ogbg-molpcba, we use batch size as 1,024 for fast training and utilize BatchNorm instead of LayerNorm. The evaluator provided by the OGB dataset is used for evaluating model performances. The experiments are conducted in the same experimental environment as TUDataset.

C.4 Additional Ablation Study for Structure Prototype

For an additional ablation study, we modify SPGP, which removes the score for the first term, namely the structure prototype, in Equation 5, and evaluate it on the TUDataset. As shown in Supplementary Table 7, the graph classification performances are significantly degraded without the guidance of node selection through the structure prototype. Performance degradation is more severe in chemical datasets (NCI1, NCI109, and FRANKENSTEIN) than in biological datasets (DD and PROTEINS). This is because the major structure motifs of molecules are often expressed in BCCs or cliques, so the structure prototypes greatly contributes to performance improvement.

C.5 Memory Footprint Test

We evaluate the GPU memory footprint of SPGP with baselines on the graphs with mean degrees similar to ogbg-molpcba generated by two random graph models: Erdős–Rényi [12] and Barabási–Albert [1]. Erdős–Rényi model generates a graph constructed by connected nodes with respect to a probability p. In this experiment, a graph with n nodes have the probability $p = \delta/n$, which $\delta = 2.16$ is a mean degree per graph in ogbg-molpcba dataset. Barabási–Albert model

Supplementary Table 5: Selected hyperparameters for TUDataset.

Dataset	Learning rate	Batch size	Hidden dimension	Number of layers	Pooling ratio	Dropout rate	Regularization λ	Decaying Step
DD	0.002	128	256	6	0.95	0.1	1.0	50
PROTEINS	0.001	128	64	4	0.8	0.1	0.0	25
NCI1	0.001	64	256	2	0.8	0.0	0.8	25
NCI109	0.001	64	256	2	0.8	0.0	0.2	25
FRANKENSTEIN	0.001	64	64	3	0.8	0.1	0.0	25

Supplementary Table 6: Selected hyperparameters for OGB dataset.

Dataset	Learning rate	Batch size	Hidden dimension	Number of layers	Pooling ratio	Dropout rate (Pooling)	Dropout rate (Conv)	Regularization λ
ogbg-molhiv	0.001	256	256	6	0.8	0.5	0.2	1.0
ogbg-molpcba	0.001	1,024	600	3	0.8	0.1	0.3	0.0

Supplementary Table 7: Ablation Study for Effect of the Structure Prototype. Average test accuracy and standard deviation is reported for 20 random seeds.

Dataset	SPGP w/o prototype (A)	SPGP	Diff (A-B)
DD	76.69 ± 0.93	77.76 ± 0.73	-1.07
PROTEINS	73.96 ± 0.61	75.20 ± 0.74	-1.24
NCI1	74.63 ± 0.45	78.90 ± 0.27	-4.27
NCI109	73.88 ± 0.53	77.27 ± 0.32	-3.39
FRANKENSTEIN	61.39 ± 0.79	68.92 ± 0.71	-7.53

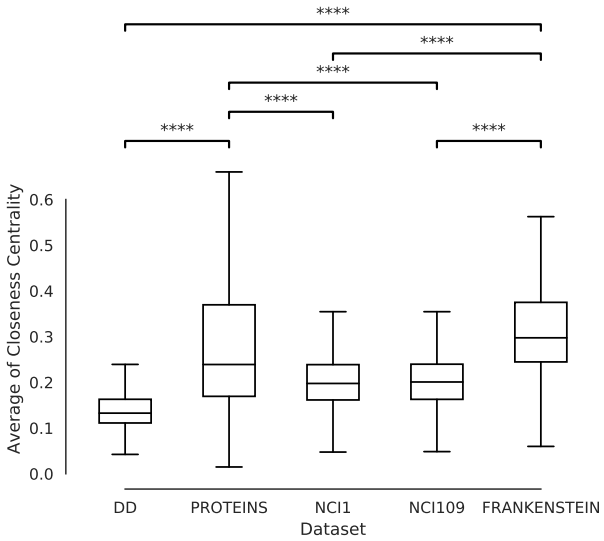
generates a scale-free network using a preferential attachment mechanism. Similar to the experiment of Erdős–Rényi model, newly added nodes preferentially are connected two nodes with high degree nodes. In both graph models, 1,000 random graphs are generated and the following experimental setup is used for evaluating the memory footprint: batch size = 10, # of layers = 3, # of hidden dimension = 300, pooling ratio = 0.8. The pooling ratio is chosen as the best performing hyperparameter setting in our experiments on the five small-scale and two large-scale benchmark datasets. When the pooling ratio is reduced from 0.8 to 0.5, the memory footprint required for each model decreased, but the pattern of increasing memory footprint with the graph size is observed as shown in Figure 5 of main text. Additionally, decreasing the pooling ratio reduces memory footprint but also results in decreased performance in benchmark dataset experiments.

C.6 Closeness Centrality Analysis of Datasets

Supplementary Figure 1 shows that the distributions of average of the closeness centrality are different between datasets. The average of closeness centrality in each graph is measured by averaging the values of closeness centrality of nodes in the graph. The statistical significance of the distribution difference is calculated by two-sided Mann-Whitney U test with Bonferroni correction. As shown in Supplementary Figure 1, PROTEINS and FRANKENSTEIN datasets have higher closeness centrality values than other datasets. The high closeness centrality of a node means that the length of the path required to propagate the information of that node to all other nodes is short. In other words, from a graph point of view, a large average closeness centrality means that the node's information is more globally spread. For this reason, it may not be necessary to incorporate the auxiliary score in the PROTEINS and FRANKENSTEIN datasets.

C.7 Additional Evaluation using ogbg-molpcba for GCN/GIN with Virtual Node

SPGP can be jointly used with advanced graph convolution layers such as GCN/GIN + virtual node. We performs SPGP with the official implementation of GCN/GIN + virtual node (VN) [19] on ogbg-molpbca dataset (450k molecules with 128 classification tasks). Supplementary Table 8 shows the average precision of validation and test dataset with two variants of SPGP model. First, after the last convolution layer, we add a single SPGP layer (p=0.8). Interestingly, we observe that the performance is improved compared to the GCN/GIN + virtual nodes. Inspired by this result, we also add one more



Supplementary Figure 1: Average closeness centrality of datasets. To compare distribution of two datasets, Mann-Whitney U test two-sided with Bonferroni correction is used. Statistical significance is denoted as symbols:
$* : 1 \times 10^{-2} < p \leq 5 \times 10^{-2}$,
$** : 1 \times 10^{-3} < p \leq 1 \times 10^{-2}$,
$*** : 1 \times 10^{-4} < p \leq 1 \times 10^{-3}$,
$**** : p \leq 1 \times 10^{-4}$

Supplementary Table 8: Performance of SPGP with GCN/GIN+VN.

Model	Validation AP	Test AP
GCN+VN (leaderboard)	0.2495 ± 0.0042	0.2424 ± 0.0034
GCN+VN+SPGP (last layer, $p = 0.8$)	0.2556 ± 0.0021	0.2504 ± 0.0032
GCN+VN+SPGP (last 2 layers, $p = 0.8$)	$\mathbf{0.2579 \pm 0.0044}$	$\mathbf{0.2538 \pm 0.0029}$
GIN+VN (leaderboard)	0.2798 ± 0.0025	0.2703 ± 0.0023
GIN+VN+SPGP (last layer, $p = 0.8$)	0.2818 ± 0.0032	0.2712 ± 0.0029
GIN+VN+SPGP (last 2 layers, $p = 0.8$)	$\mathbf{0.2848 \pm 0.0018}$	$\mathbf{0.2736 \pm 0.0016}$

SPGP layer (p=0.8) after the last second convolution layer, then further performance improvement is also observed in this case. This is a very interesting result demonstrating the effectiveness of SPGP.

D Broader Impact

Our work introduces a novel graph pooling method that utilizes prior graph structures to learn more accurate graph-level representations. While most of the existing graph pooling methods focus only on k-hop neighborhood in the graph structure information, our work uses explicit graph structures to provide more structural information to the model. Our experimental results show that the introduction of various global graph structures can improve graph classification performances. Thus, our work can provide insight that the use of domain knowledge is important in graph learning, and potentially have positive social impacts in various fields of study related to learning graph representation For example, biological pathways can be used to obtain patient embeddings from protein-protein interaction networks. Chemical motifs could play an important role in predicting molecular properties for AI-based drug discovery.

One potential negative societal impact is that the application of our work to specific fields may impose the burden of obtaining well-defined prior graph structures based on domain knowledge. To mitigate this, we introduce two well-known graph structures in graph theory: biconnected components and cliques. These two structures are good examples of containing global structural information in general graphs. Therefore, inspired by our work, it can be a good starting point when conducting research in a specific field.