Edge but not Least: Cross-View Graph Pooling

Xiaowei Zhou[1,3], Jie Yin[2]✉, and Ivor W. Tsang[1,4]

[1] Australian Artificial Intelligence Institute (AAII), University of Technology Sydney, NSW 2007, Australia
Xiaowei.Zhou@student.uts.edu.au, ivor.tsang@uts.edu.au
[2] Discipline of Business Analytics, The University of Sydney, NSW 2006, Australia
jie.yin@sydney.edu.au
[3] Data61, CSIRO, NSW 2015, Australia
[4] Center for Frontier AI Research A*STAR, Singapore

Abstract. Graph neural networks have emerged as a powerful representation learning model for undertaking various graph prediction tasks. Various graph pooling methods have been developed to coarsen an input graph into a succinct graph-level representation through aggregating node embeddings obtained via graph convolution. However, because most graph pooling methods are heavily node-centric, they fail to fully leverage the crucial information contained in graph structure. This paper presents a cross-view graph pooling method (Co-Pooling) that explicitly exploits crucial graph substructures for learning graph representations. Co-Pooling is designed to fuse the pooled representations from both node view and edge view. Through cross-view interaction, edge-view pooling and node-view pooling mutually reinforce each other to learn informative graph representations. Extensive experiments on one synthetic and 15 real-world graph datasets validate the effectiveness of our Co-Pooling method. Our results and analysis show that (1) our method is able to yield promising results over graphs with various types of node attributes, and (2) our method can achieve superior performance over state-of-the-art pooling methods on graph classification and regression tasks.

Keywords: Graph Pooling · Graph Representation Learning.

1 Introduction

With widespread digitization occurring in various domains, a significant portion of data takes the form of graphs, such as social networks, chemical molecular graphs, and financial transaction networks. As such, learning effective graph representations plays a crucial role in a variety of tasks, such as drug discovery, molecule property prediction, and traffic forecast, etc. Recently, graph neural networks (GNNs) have emerged as state-of-the-art models for graph representation learning, including graph convolutional network (GCN) [9], graph attention network (GAT) [20], graph isomorphism network (GIN) [22], and GraphSAGE [8]. Most of these GNN models rely on message passing to learn the embedding of each node by aggregating and transforming the features of its neigh-

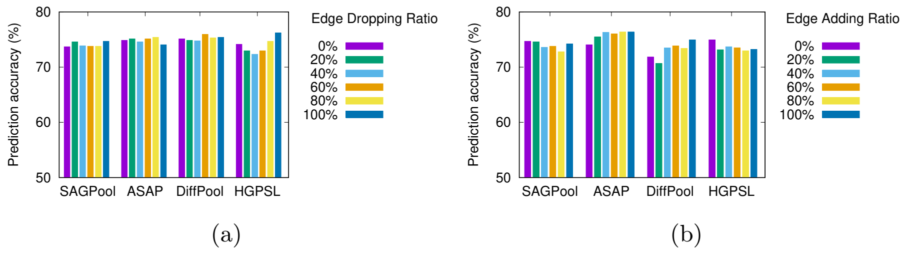


Fig. 1: Classification accuracy on PROTEINS with different edge ratios. Accuracy does not significantly drop when different ratios of edges are (a) randomly dropped from original graphs, or (b) randomly added from no-edge graphs.

bouring nodes. To obtain the representation of the entire graph, node embeddings are aggregated via a readout function or graph pooling methods [26,24,15,25]. Graph pooling methods coarsen an input graph into a compact vector-based representation for the entire graph, which is then used for graph prediction tasks, such as graph classification or graph regression.

To learn expressive graph representations, various graph pooling methods have been proposed in recent years. Sampling based methods (*e.g.*, SAGPool [10], ASAP [15], HGPSL [26]) calculate an importance score for each node and then select the top K important nodes to generate an induced subgraph. For example, SAGPool [10] selects nodes by learning importance scores via a self-attention mechanism. HGPSL [26] samples important nodes and uses an additional structure learning mechanism to learn new graph connectivity for the sampled nodes. Clustering based methods, like differentiable graph pooling (DiffPool) [24], learn an assignment matrix to cluster nodes into several super-nodes level by level. Then, a hierarchy of the induced subgraphs can be generated for representing the whole graph. Nonetheless, we argue that the existing pooling methods focus primarily on aggregating node-level information, so they fail to exploit key graph substructures for learning graph-level representations. The loss of information present in the global graph structure would hinder message passing in subsequent layers and consequently jeopardize the graph representation expressiveness.

To verify our argument, we select four state-of-the-art pooling methods: SAGPool [10], ASAP [15], DiffPool [24], and HGPSL [26] and analyze the influence of changing graph topological structure on the graph classification accuracy. We use PROTEINS, a macromolecule dataset containing rich structural information, as a case study, where we change graph topological structure by randomly dropping and adding edges with different ratios. As shown in Fig. 1, we find that the random edge manipulation does not cause a significant drop in the graph classification accuracy. Surprisingly, when there are no edges at all, *i.e.*, dropping 100% edges in Fig. 1(a) and adding 0% edges (no edges) in Fig. 1(b), the classification accuracy still retains at the same level as other edge ratios. In particular, for HGPSL (which implicitly uses edge information), the classification accuracy is the highest when all edges are removed. Our empirical studies indicate that

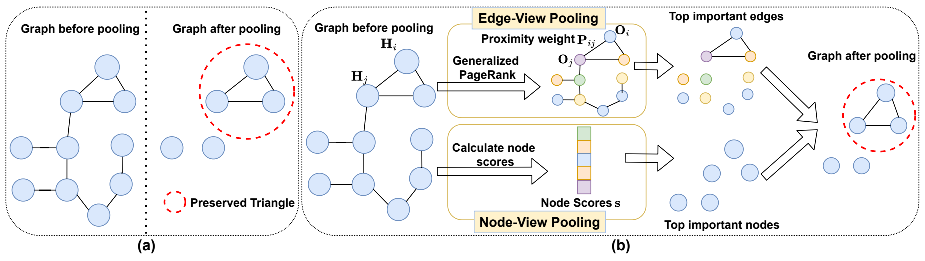


Fig. 2: (a) Examples of substructure (triangle) preserved by Co-Pooling. (b) Overview of our proposed Co-Pooling framework. Co-Pooling is composed of two complementary components – edge-view pooling and node-view pooling – that reinforce each other to better learn informative graph-level representations.

current graph pooling methods are heavily node-centric and lack the ability to fully leverage the crucial information contained in graph structure.

To fill this research gap, we propose a novel cross-view graph pooling method (Co-Pooling) that explicitly exploits graph substructures for learning graph-level representations. Our main motivations are twofold. First, we would like to capture crucial graph substructures through explicitly pruning unimportant edges in the graph. Key substructures, such as functional groups (*e.g.*, triangles, rings) in molecular networks, or cliques in protein-protein interaction networks and social networks, have been widely recognised as a crucial source for graph prediction tasks [13]. For example, in molecular chemistry, certain patterns of atoms (*e.g.*, triangles) are considered highly indicative for predicting certain molecule's properties [17]. As illustrated in Fig. 2(a), we need to preserve circular connectivity among three nodes in order to capture a triangle substructure. The crux is that, if graph pooling operates in a node-centric way or on the pairwise adjacency matrix, such higher-order, triangle circular connectivity cannot be properly reserved. Thus, we propose to preserve key substructures through learning higher-order proximity weights, which are then used to prune unimportant edges for substructure extraction. Second, apart from structural information, real-world graphs often have various types of node properties, such as one-hot attributes, real-valued attributes, or even no attributes (see Table 2). Hence, it is highly desirable for our pooling method to fuse useful information from both edge and node views and to make the best of node-level attributes when available.

Co-Pooling is composed of two key components: *edge-view pooling* and *node-view pooling*. Fig. 2(b) shows the overview of Co-Pooling. Edge-view pooling aims to preserve crucial graph substructures, which are informative for subsequent graph prediction tasks. This is achieved by capturing high-order structural and attribute proximity via generalized PageRank and then pruning the edges with lower proximity weights. For node-view pooling, an importance score is calculated for each node, and top-ranked important nodes are selected for pooling. The learning of graph pooling from the edge and node views mutually reinforces each other through exchanging proximity weights and the selected important nodes. The final pooled graph is obtained by fusing graph representations learned from these two views. Through cross-view interaction, Co-Pooling enables edge-view pooling and node-view pooling to complement each other towards learning expressive graph representations. Our contributions are summarised as follows:

– We investigate the ineffectiveness of the existing node-centric graph pooling methods in fully leveraging graph structure.
– We propose a novel graph pooling method (Co-Pooling) to learn graph representations by fusing the pooled graph from both node view and edge view. Co-Pooling has the ability to preserve crucial graph substructures and to handle different types of graphs (node-labeled/attributed/plain graphs).
– We validate the effectiveness of Co-Pooling in graph classification and regression tasks across one synthetic and 15 real-world graph datasets, demonstrating its competitive performance over state-of-the-art pooling methods.

2 Related Work

Graph pooling is a key component of GNNs for learning a vector representation of an input graph. The existing graph pooling methods can be divided into two categories: *sampling based pooling* and *clustering based pooling*.

Sampling based pooling methods generate a smaller induced graph by selecting the top important nodes according to certain importance scores of nodes. SortPooling [27] ranks the nodes according to node embeddings learned from graph convolution and stacks the embeddings of selected nodes as graph representation. SAGPool [10] uses a self-attention mechanism to calculate an importance score for each node and then chooses top-ranked nodes to induce the pooled graph. Ranjan et al. [15] propose adaptive structure aware pooling (ASAP) that updates node embeddings by aggregating the features of nodes in a local region and then calculates a fitness score for each node to select the top-k nodes. Gao et al. [7] propose neighborhood information gain as a criterion to select top important nodes and then constructs a coarsened graph from selected nodes. The above mentioned methods, however, do not fully leverage the crucial information contained in graph structure in the pooling process. HGPSL [26] takes one step forward to learn new connections between the selected nodes, but fails to capture crucial substructures contained in the original graph.

Clustering based pooling methods learn an assignment matrix to cluster nodes into super-nodes. DiffPool [24] learns a differentiable soft cluster assignment, which is used to group nodes into several clusters in the subsequent layer. HaarPooling [21] relies on compressive Haar transform filters to generate the induced graph of smaller size. HAP [11] uses master-orthogonal attention to learn a soft assignment to cluster nodes. SUGAR [18] samples several subgraphs from the input graph and clusters the top important subgraphs into super-nodes via reinforcement learning. However, it is highly dependent on the sampling strategy used to obtain useful subgraph candidates.

Most of current graph pooling methods operate on a single node view; they are unable to fully leverage crucial graph structure. Although preliminary attempts (*e.g.*, EdgePool [3] and EdgeCut [5]) have been made to pool the input graph from an edge view, these methods simply rely on local connectivity to calculate pairwise edge scores. In contrast, our edge-view pooling leverages higher-order structural and attribute proximity to measure the importance of edges, which is fed to further guide the selection of important nodes for node-view pooling. To the best of our knowledge, we are the first to propose a cross-view graph pooling method, which enables pooling to fuse useful information from both edge and node views towards learning informative graph representations.

3 Methodology

In this section, we first introduce preliminaries and notations, and then present the details of our proposed cross-view pooling method.

3.1 Preliminaries and Notations

Suppose we are given m input graphs $\mathbb{G} = \{G^{(0)}, G^{(1)}, \cdots, G^{(m)}\}$ and their corresponding targets $\mathbb{Y} = \{y^{(0)}, y^{(1)}, \cdots, y^{(m)}\}$. For graph classification, $y^{(i)}$ is a discrete class label; for graph regression, $y^{(i)}$ is a continuous target variable $y^{(i)} \in \mathbb{R}$. An arbitrary graph $G^{(g)}$ is represented as $(\mathcal{V}^{(g)}, \mathcal{E}^{(g)}, \mathbf{X}^{(g)})$. For simplicity, $(\mathcal{V}^{(g)}, \mathcal{E}^{(g)}, \mathbf{X}^{(g)})$ is also noted as $(\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the node attribute matrix, where $n = |\mathcal{V}|$ is the number of nodes and d is the dimension of node attributes; $\mathbf{A}$ is the adjacent matrix, if there is an edge between node i and j, $\mathbf{A}_{ij} = 1$; otherwise $\mathbf{A}_{ij} = 0$. $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacent matrix with self-loops.

In this work, we use graph convolution network (GCN) as our backbone to learn representations for graphs. The graph convolution operation is defined as:

$$\mathbf{H} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta} \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{n \times f}$ is node embedding after convolution, f is the dimension of node embedding; $\hat{\mathbf{D}}_{ii} = \sum_{j=0} \hat{\mathbf{A}}_{ij}$ is diagonal degree matrix; $\mathbf{\Theta}$ is a learnable parameter.

After node embeddings are learned, graph pooling aims to generate a vector representation for the whole graph. To facilitate downstream graph prediction tasks, the learned graph representation is expected to preserve the information conveyed by both graph structure and node attributes.

3.2 Cross-View Graph Pooling: Co-Pooling

The key idea of Co-Pooling is to preserve crucial “signals" that are beneficial to downstream graph prediction tasks. Unlike previous studies that dominantly focus on node-level information, we take both node and edge views to preserve crucial substructures reflected by graph structure and node attributes. To this end, we propose to perform graph pooling from both edge and node views.

As shown in Fig. 2(b), our proposed Co-Pooling framework consists of two complementary components: edge-view pooling and node-view pooling. Edge-view pooling prunes unimportant edges to capture meaningful substructures (*e.g.*, triangles). Node-view pooling, on the other hand, further selects top-ranked important nodes. Through cross-view interaction, edge-view pooling and node-view pooling reinforce each other to induce informative graph representations.

Edge-View Pooling The key objective of edge-view pooling is to preserve crucial substructures contained in the original graph. Extracting useful substructures requires to incorporate high-order structural and node attribute information. Thus, we propose to use generalized PageRank (GPR) [2] to jointly optimize node attribute and topological information extraction.

To be specific, we first update node embeddings via GPR to capture the information from multi-hop neighbours. As shown in Eq. (2), node embeddings are updated by multiplying a GPR weight β_t at each step t. When $t = 0$, we have $\mathbf{H}^0 = \mathbf{H}$; when $t > 0$, we have $\mathbf{H}^t = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{t-1}$. Through GPR, node embeddings propagate T steps, and the GPR weight β_t is learned at each step. Thus, the contribution of each propagation step towards node embeddings can be learned adaptively. The GPR operation of T steps helps incorporate the information from multi-hop neighbours to learn expressive node embeddings.

$$\mathbf{O} = \sum\nolimits_{t=0}^{T} \beta_t \mathbf{H}^t. \tag{2}$$

After updating node embeddings via GPR, we calculate pairwise proximity weights that reflect high-order structural and attribute proximity between nodes. This process can be illustrated using Eq. (3), where $\mathbf{O}_i$ and $\mathbf{O}_j$ are the updated embeddings of node i and node j by GPR. We first transform node embeddings $\mathbf{O}_i$ and $\mathbf{O}_j$ via a linear transformation parameterized with $\mathbf{W}$. Then, the transformed embeddings are concatenated and fed to another linear transformation with learnable parameters $\mathbf{a}$. Finally, the proximity weight $\mathbf{P}_{ij}$ between node i and node j is obtained via a Sigmoid function. To preserve the original adjacency of graphs, we multiply the proximity weight with the adjacent matrix $\mathbf{A}$.

$$\mathbf{P}_{ij} = \sigma(\mathbf{a}^T[\mathbf{W}\mathbf{O}_i \| \mathbf{W}\mathbf{O}_j]) \odot \mathbf{A}_{ij}, \tag{3}$$

where $\mathbf{P}_{ij}$ is the proximity weight between node i and node j; σ is Sigmoid function; $\|$ represents the concatenation operation; $\mathbf{a}$ and $\mathbf{W}$ are learnable parameters; $\odot$ represents matrix element-wise multiplication; $\mathbf{A}_{ij} = 1$ *or* 0 indicates whether or not there is an edge connecting node i and node j.

According to the proximity weight $\mathbf{P}_{ij}$ of each node pair, we can obtain the proximity matrix $\mathbf{P}$ for all node pairs. For undirected graphs, we average the proximity weights at symmetric positions by $\mathbf{P}_{\text{sym}} = (\hat{\mathbf{P}} + \hat{\mathbf{P}}^T)/2$.

For a specific prediction task, the edges constituting discriminative substructures are expected to have higher proximity weights. Conversely, less important edges would have lower proximity weights. Thus, we prune unimportant edges with low proximity weights during edge-view pooling. For a given edge preserving ratio γ, we have the pruned proximity matrix $\mathbf{P}_{\text{prune}} = \text{Top}_{\gamma}(\mathbf{P}_{\text{sym}})$, where $\text{Top}_{\gamma}()$ is the operation that preserves the top γ percentage of edges with high proximity weights. Accordingly, we update the adjacent matrix to reflect the removal of edges. The pruned proximity matrix signifies certain crucial substructures preserved by pruning unimportant edges. The pruned proximity matrix is further fed to node-view pooling to guide the selection of important nodes.

Node-View Pooling For node-view pooling, the aim is to select the top K important nodes for coarsening the input graph. To better exploit the connectivity between nodes, we take the pruned proximity matrix from edge-view pooling to compute an importance score for each node, given by

$$\mathbf{s} = \hat{\mathbf{D}}_{\text{prune}}^{-1/2} \hat{\mathbf{P}}_{\text{prune}} \hat{\mathbf{D}}_{\text{prune}}^{-1/2} \mathbf{H} \mathbf{1}^T, \tag{4}$$

where $\mathbf{s}$ is the score vector for all nodes; $\hat{\mathbf{D}}_{\text{prune}}$ is the diagonal degree matrix of $\hat{\mathbf{P}}_{\text{prune}}$, $\hat{\mathbf{P}}_{\text{prune}} = \mathbf{P}_{\text{prune}} + \mathbf{I}$; and $\mathbf{1}^T$ is a vector with all entries as one.

Based on node importance scores, we select the top $K = n \times \epsilon$ nodes, where ϵ is the node pooling ratio. For selected nodes, we can obtain their indices and corresponding node embeddings.

Edge-Node View Interaction To enable edge-view pooling and node-view pooling to reinforce each other, Co-Pooling exchanges the pruned proximity matrix and the indices of selected nodes, which serve as the mediator for the interaction between two views.

For node-view pooling, the pruned proximity matrix from edge-view pooling is used to calculate the important score for each node. The pruned proximity matrix better reflects higher-order structural and attribute proximity between nodes, thus providing a better measure than the original adjacent matrix to quantify the importance of nodes contained in certain substructures. After obtaining the node scores, we select the top-K important nodes as the pooled graph, *i.e.*, $\mathbf{H}(\text{indices}, :)$, where $(\text{indices}, :)$ indicates index selection operation.

For edge-view pooling, the indices of selected nodes obtained from node-view pooling are used to aggregate node embeddings from neighborhoods based on the pruned proximity matrix. The node selection process in node-view pooling is useful for further extracting meaningful substructures, as less important nodes are removed. The pooled representation from edge-view pooling is obtained as $\mathbf{P}_{\text{prune}}(\text{indices}, :)\mathbf{H}$.

Lastly, the pooled representations from node-view pooling and edge-view pooling are fused to form the final graph representation as:

$$\mathbf{Z} = \mathbf{W}[\mathbf{P}_{\text{prune}}(\text{indices}, :)\mathbf{H} \| \mathbf{H}(\text{indices}, :)] \tag{5}$$

where $\mathbf{W}$ is a learnable parameter of linear transformation; $\|$ indicates the concatenation operator; and $\mathbf{Z}$ is the graph representation after pooling. Through edge-node view interaction, Co-Pooling enables edge-view pooling and node-view pooling to complement each other for learning informative graph representations.

4 Experiments

We evaluate the performance of Co-Pooling on three graph prediction tasks, including substructure counting, graph classification, and graph regression. For substructure counting (Section 4.1), we empirically assess the performance of Co-Pooling in preserving important substructures. For graph classification, we compare Co-Pooling against several state-of-the-art pooling methods under two settings: *attribute-complete graphs* (Section 4.2) and *attribute-incomplete graphs* (Section 4.3). Furthermore, we compare Co-Pooling against baseline pooling methods on graph regression (Section 4.5). The source code of Co-Pooling is available at: https://github.com/zhouxiaowei1120/Co-Pooling.

Baselines. As our focus is upon designing new graph pooling methods, we compare Co-Pooling with five state-of-the-art graph pooling methods rather than specially designed GNNs for graph classification. These baseline methods include: SAGPool [10], ASAP [15], DiffPool [24], HGPSL [26], and EdgePool [3]. When training DiffPool, we use the auxiliary link prediction loss function with entropy regularization as in the original paper. For comparison, all graph pooling methods are built on top of the same GCN architecture for downstream tasks.

4.1 Substructure Counting on Random Graphs

To verify the capacity of Co-Pooling in preserving graph substructures, we consider a substructure counting task, with the aim to count the number of triangles contained in random graphs.

Dataset. For substructure counting task, we use the synthetic **Syn-triangle** dataset [1], consisting of 5,000 Erdös-Renyi random graphs. Each graph contains 10 nodes and $p = 0.3$ is the probability that an edge exists. Akin to [1], we use 30%-20%-50% graphs as training-validation-test sets.

Experimental Setup. For training a regression model on Syn-triangle, we use GCN as the backbone and inject two pooling layers before an MLP layer. Adam optimizer with learning rate decay is used to train the model. The optimization stops if the validation loss doesn't decrease after 50 epochs. Following [1], we use L2 loss function and set the initial learning rate and weight decay as 0.02 and 0.001, respectively. The GPR operation step T is set as 3. We train the regression model with four different random seeds. The results are measured by mean square error (MSE) on the test data divided by the variance of the ground truth counts.

Table 1: Normalized MSE for substructure counting on Syn-triangle.

Methods	Syn-triangle
SAGPool	0.849±0.061
ASAP	0.701±0.140
DiffPool	0.762±0.194
HGPSL	0.878±0.079
EdgePool	0.704±0.009
Co-Pooling	**0.448±0.046**

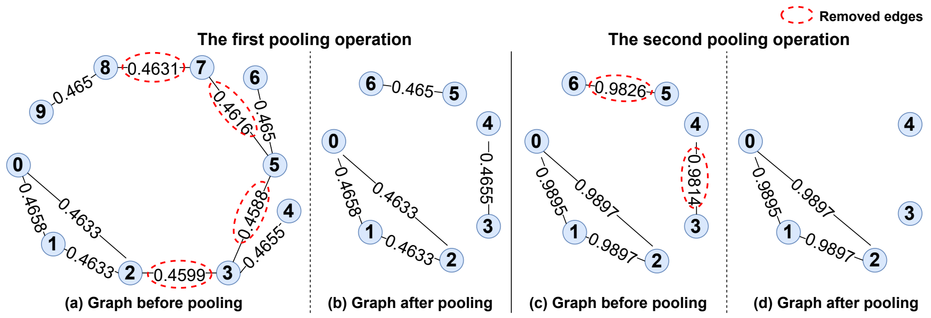


Fig. 3: Illustration of the pooled graphs by Co-Pooling.

Results. The MSE results on triangle counting are given in Table 1. As can be seen, Co-Pooling outperforms all baseline methods, yielding markedly smaller errors than the second best performer ASAP. This empirically verifies that Co-Pooling is able to preserve crucial substructures during the pooling process.

Fig. 3 gives an example to illustrate two pooling operations of Co-Pooling. For a given graph, the proximity weights of edges and the pooled graphs are shown in the figure. During the first pooling operation, four edges with lower proximity weights (marked in red ellipse) are removed, and afterwards, nodes 7, 8, and 9 with lower importance scores are further removed to generate the pooled graph. It is clear to find that the triangle substructure is preserved after the first pooling layer (see Fig. 3(b)). A similar process can be observed during the second pooling operation, where the triangle substructure is also preserved in the pooled graph (see Fig. 3(d)), which is highly indicative for triangle counting.

4.2 Graph Classification on Attribute-Complete Graphs

Datasets. We undertake graph classification on a total of 13 benchmark graph datasets with various attribute properties, including three *attributed graph* datasets with real-valued node attributes, five *labeled graph* datasets with only one-hot node attributes, and five *plain graph* datasets without node attributes. The detailed statistics about these datasets are listed in Table 2.

- **BZR-A** [19] is a dataset of chemical compounds for classifying biological activities. The node attributes are 3D coordinates of compound structures.
- **AIDS-A** [16] is composed of graphs representing molecular compounds. It contains two classes of graphs, which are against HIV or not.
- **FRANKENSTEIN** [14] consists of molecules as mutagens or non-mutagens for binary classification. The node attributes are 780-dimensional MNIST image vectors of pixel intensities, representing chemical atom symbols.
- **D&D** [12] and **PROTEINS** [12] include macromolecules as graph datasets in bioinformatics, which are for enzyme and non-enzyme classification task.
- **NCI1** [12] and **NCI109** [12] contain chemical compounds as small molecules, which are used for anticancer activity classification task.

Table 2: Details of graph datasets for graph classification evaluation.

Dataset	# Graphs	# Classes	Avg. $\lvert V \rvert$	Avg. $\lvert E \rvert$	Node Attributes	Type
BZR-A	405	2	35.75	38.36	Real-valued	*Attributed*
AIDS-A	2,000	2	15.69	16.20	Real-valued	*Attributed*
FRANKENSTEIN	4,337	2	16.90	17.88	Real-valued	*Attributed*
PROTEINS	1,113	2	39.06	72.82	Node label	*Labeled*
D&D	1,178	2	284.32	715.66	Node label	*Labeled*
NCI1	4,110	2	29.87	32.30	Node label	*Labeled*
NCI109	4,127	2	29.68	32.13	Node label	*Labeled*
MSRC_21	563	20	77.52	198.32	Node label	*Labeled*
COLLAB	5,000	3	74.49	2457.78	None	*Plain*
IMDB-B	1,000	2	19.77	96.53	None	*Plain*
IMDB-M	1,500	3	13.00	65.94	None	*Plain*
REDDIT-B	2,000	2	429.63	497.75	None	*Plain*
REDDIT-M	11,929	11	391.41	456.89	None	*Plain*

- **MSRC_21** [12] is a graph dataset constructed by semantic images. Each image is represented as a conditional Markov random field graph. Nodes in a graph represent the segmented superpixels in an image. If the segmented superpixels are adjacent, the corresponding nodes are connected. Each node is assigned a semantic label as node attribute.
- **COLLAB** [23] is a collection of scientific collaboration graphs, where the task is to classify the graphs into different research fields.
- **IMDB-B** [23] and **IMDB-M** [23] are two datasets for classifying graphs into movie genres. Each graph is an ego-network for each actor/actress.
- **REDDIT-B** and **REDDIT-M** [23] are two datasets generated from online discussions. Each graph represents a discussion thread where nodes indicate different users. If one user responds to another one, there is an edge between them. The task is to classify which section each discussion belongs to.

Baselines. Apart from other graph pooling baselines, we also compare with two ablated variants of Co-Pooling: Co-Pooling w/o GPR that removes generalized PageRank and Co-Pooling w/o NV that removes node-view pooling.

Experimental Setup. For all datasets, we use the same GNN architecture for a fair comparison. The GNN consists of three GCN layers, two pooling layers (constructed by different pooling methods), and three linear transformation layers. A softmax layer is then connected after the last linear transformation layer. Note that, the input to the first linear transformation layer is the concatenated features after each pooling layer.

Akin to prior work [24], we perform 10-fold cross-validation to train the GNN model. We randomly partition each dataset into training, validation, and test sets using a 80%-10%-10% split. We use Adam optimizer with early stopping; the optimization stops if the validation loss does not improve after 50 epochs.

Table 3: Graph classification accuracy on 13 graph datasets.

Dataset	SAGPool	ASAP	DiffPool	HGPSL	EdgePool	Co-Pooling (w/o GPR)	Co-Pooling (w/o NV)	Co-Pooling
BZR-A	82.95±4.91	83.70±6.00	83.93±4.41	83.23±6.51	83.43±6.00	81.00±5.82	81.69±5.80	85.67±5.29
AIDS-A	98.85±0.78	99.00±0.74	99.40±0.58	99.10±0.66	99.05±0.69	98.85±0.71	98.90±0.58	99.45±0.42
FRANK	60.94±2.90	66.73±2.76	65.08±1.50	62.19±1.74	62.99±2.21	64.01±1.70	67.00±2.37	64.15±1.34
D&D	76.91±3.42	77.84±3.41	78.01±2.70	77.33±4.22	76.66±2.05	75.81±3.81	77.00±5.04	77.85±2.21
PROTEINS	73.68±4.63	74.85±5.18	75.11±2.95	74.13±4.12	77.01±5.41	73.68±2.33	76.28±5.09	76.19±4.13
NCI1	71.51±4.51	76.59±1.71	74.14±1.43	73.48±2.42	78.39±2.43	77.25±2.11	79.15±2.04	78.66±1.48
NCI109	69.69±3.27	74.73±3.48	72.04±1.43	72.30±2.18	77.01±2.39	75.60±1.46	78.07±1.77	77.08±2.03
MSRC_21	90.22±2.82	90.41±3.91	90.41±3.58	88.97±4.78	90.05±3.02	91.64±2.79	91.29±3.70	92.54±2.63
COLLAB	70.58±2.31	72.84±1.84	72.18±1.68	74.20±2.72	-	74.82±2.10	68.90±5.59	77.30±2.29
IMDB-B	60.90±2.34	65.50±2.80	58.27±5.92	62.50±3.50	60.30±5.08	70.40±3.85	70.80±3.60	72.10±4.44
IMDB-M	39.80±3.39	45.93±4.03	40.00±4.52	40.53±4.88	44.27±4.50	47.60±4.55	44.80±3.94	49.07±3.28
REDDIT-B	83.55±4.53	-	84.61±2.42	-	88.35±2.31	88.90±2.00	88.00±4.69	89.35±1.25
REDDIT-M	40.56±3.30	-	41.21±1.96	-	-	46.84±2.26	49.02±1.56	46.85±2.62

"-" means the results can not be obtained in an acceptable time, *i.e.* 24h.

The maximum epoch number is set as 300. Following [10], we use grid search to obtain optimal hyperparameters for each method. The ranges of different hyperparameters are set as follows: learning rate in {0.005, 0.0005, 0.001}, weight decay in {0.0001 0.001}, node pooling ratio in {0.5, 0.25}, hidden size in {128, 64}, dropout ratio in {0, 0.5}, and edge preserving ratio γ in {0.3, 0.6, 1.0}. Akin to [2], step T of GPR is set to 10. To implement the convolution operation on plain graphs without node attributes, we follow the implementation of DiffPool to pad each node with a constant vector, *i.e.* an all-one vector of 10 dimensions.

Comparison with State-of-the-art. Table 3 shows graph classification accuracy of all methods averaged over 10-fold cross-validation on 13 datasets. For a fair comparison, all baseline methods and our method are trained using the same training strategy. As we can observe, among all methods, our Co-Pooling method achieves the best performance on all datasets except on D&D and PROTEIN, where Co-Pooling achieves the second best performance. In particular, Co-Pooling significantly improves the best baseline method by 6.6%, 3.14%, 7.81%, 2.13%, and 1.74% on IMDB-B, IMDB-M, REDDIT-M, MSRC_21, and BZR-A, respectively. This proves the effectiveness of Co-Pooling in predicting different types of graphs with various attribute properties. It is worth noting that Co-Pooling achieves the best performance on all five datasets without node attributes. This shows the superiority of our method to complement node-view pooling with edge-view pooling, when node attributes are not informative.

When comparing different variants of our method, Co-Pooling consistently outperforms Co-Pooling w/o GPR on all datasets. This shows the importance

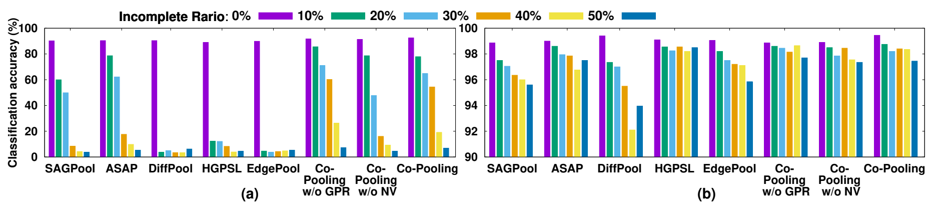


Fig. 4: Graph classification accuracy on (a) *labeled graph* dataset (MSRC_21) and (b) *attributed graph* dataset (AIDS-A) under attribute-incomplete settings.

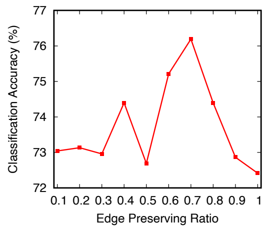
of using generalized PageRank to capture higher-order structural information. Co-Pooling yields higher accuracy than Co-Pooling w/o NV on most (8/13) of the datasets. This demonstrates the effectiveness of Co-Pooling in combining two complementary views. In particular, its performance gains on *attributed graphs* with real-valued node attributes are more significant than those on *labeled graphs* with one-hot attributes. This is because real-valued node attributes provide more accurate information to select important nodes for node-view pooling as opposed to one-hot attributes. This in turn reinforces edge-view pooling more effectively for learning the final graph representations.

4.3 Graph Classification on Attribute-Incomplete Graphs

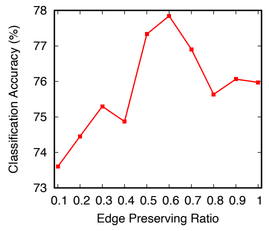
Next, we compare the performance of our method and all baselines on attribute-incomplete graphs. For attribute-incomplete graphs, a portion of nodes has completely missing attributes. This set of experiments is used to evaluate the effectiveness of our method in real-world scenarios, where attribute information for some nodes is inaccessible due to privacy or legal constraints.

Experimental Setup. We perform experiments on *attributed graph* dataset (AIDS-A) and *labeled graph* dataset (MSRC_21) as a case study. For each graph from the two datasets, we randomly select different ratios of nodes and remove their original node attributes, while keeping the rest of nodes unchanged. We define the ratio of nodes with all their attributes removed as the incomplete ratio. For example, if we remove all attributes for 10% of nodes, the incomplete ratio is 10%. The resulting attribute-incomplete graph datasets are randomly divided into training set (80%), validation set (10%), and test set (10%). We train the GNN model with different pooling methods on training set. The GNN model architecture and the best hyperparameters are the same as in Section 4.2. We report graph classification accuracy averaged over 10-fold cross-validation.

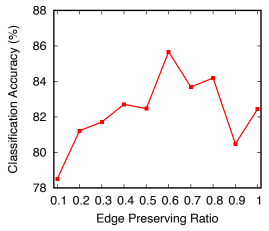
Comparison with State-of-the-art. Fig. 4 (a) compares the classification accuracy of all methods on attribute-incomplete MSRC_21 datasets. For all baseline methods, the classification accuracy drops significantly as the incomplete ratio increases from 0% to 50%. In contrast, the accuracy of Co-Pooling and its variants decreases at a much lower rate. Especially for DiffPool, HGPSL,



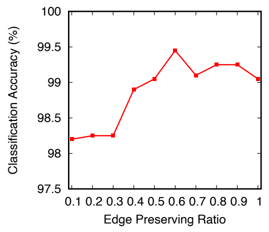
(a) PROTEINS



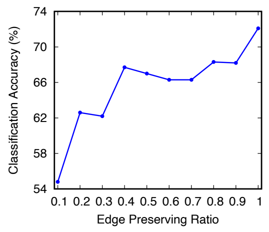
(b) D&D



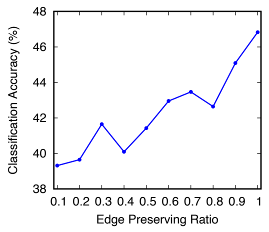
(c) BZR-A



(d) AIDS-A



(e) IMDB-B



(f) REDDIT-M

Fig. 5: Graph classification accuracy with different edge preserving ratios (γ).

and EdgePool, the classification accuracy drops by 3.73%, 12.33%, and 4.61%, respectively, although only 10% nodes have their attributes missing. With the 10% incomplete ratio, Co-Pooling and its variants can still achieve at least 77.93% accuracy. Compared with the best baseline method ASAP, Co-Pooling achieves an average of 8.62% accuracy increase on attribute-incomplete MSRC_21 datasets with different incomplete ratios (from 0% to 50%).

Fig. 4(b) compares graph classification accuracy of all methods on attribute-incomplete AIDS-A datasets. We can see that, for SAGPool, DiffPool, and EdgePool, the classification accuracy drops by 3.25%, 5.45%, and 3.2%, respectively, when the incomplete ratio increases from 0% to 50%. On the contrary, the accuracy of Co-Pooling and its variants drops by around 1.15% only. Our methods beat ASAP with all incomplete ratios. Compared with HGPSL, our methods achieve better performance with 0%, 10%, 20%, and 40% incomplete ratios, with higher average accuracy on all attribute-incomplete AIDS-A datasets.

The comparisons on attribute-incomplete graph datasets demonstrate the effectiveness of our method in handling graphs with missing node attributes. This further testifies the complementary advantage of our method by fusing node-view and edge-view pooling, especially when node attributes are less informative.

4.4 Parameter Sensitivity

The Co-Pooling method has the edge preserving ratio (γ) as an important parameter to determine the percentages of edges preserved during edge-view pooling. To investigate the effect of the edge preserving ratio (*i.e.*, γ) on the graph classification accuracy of Co-Pooling, we conduct empirical studies on six representative graph datasets, including two *labeled graph* datasets, two *attributed graph* datasets, and two *plain graph* datasets. On each dataset, we train the GNN model with an edge preserving ratio ranging from 10% to 100%. All other hyperparameters are set as the best values obtained in Section 4.2. We also use the same GNN model architecture and training strategy as in Section 4.2. We report the average classification accuracy on 10-fold cross-validation.

Fig. 5 plots the change in classification accuracy with respect to γ on the six datasets. On the two *labeled graph* datasets (PROTEINS and D&D), we find that keeping all edges ($\gamma = 1.0$) is not the best choice for graph classification. As shown in Fig. 5 (a) and (b), Co-Pooling achieves the highest classification accuracy when $\gamma = 0.7$ on PROTEIN, and $\gamma = 0.6$ on D&D, respectively. A similar phenomenon can also be observed on the two *attributed graphs* (BZR-A and AIDS-A). As shown in Fig. 5 (c) and (d), Co-Pooling yields the best performance when γ is set to 0.6 on the two datasets. The results on the four datasets indicate that not all edges are useful for graph classification when graphs have informative node attributes. Again, this confirms the effectiveness of our method in preserving crucial edge information through edge-view pooling and using this knowledge to further guide node-view pooling. On the other hand, on the two *plain graph* datasets (IMDB-B and REDDIT-M), keeping all edges renders the highest classification accuracy. As shown in Fig. 5 (e) and (f), Co-Pooling achieves the best performance on both graphs when keeping all edges ($\gamma = 1.0$). This is what we have expected, because when graphs have no node attributes, preserving all graph structures would best benefit graph classification.

4.5 Graph Regression

Lastly, we carry out experiments to evaluate the efficacy of our method on the graph regression task. We compare Co-Pooling with the same state-of-the-art pooling methods on the following two graph datasets:

- **ZINC** [12] contains 250,000 molecules. The task is to regress the properties of molecules. We focus on predicting one specific graph property, contained solubility. Following the setting in [4], we use 10,000 graphs from ZINC for training, 1,000 graphs for validation, and 1,000 graphs for testing.
- **QM9** [12] is a graph dataset consisting of 13,000 molecules with 19 regression targets. We focus on regressing dipole moment μ, one of 19 molecular properties. All 13,000 molecules are randomly divided into training-validation-test sets using a 80%-10%-10% split.

For training a regression model on each dataset, we use the same GNN architecture and training strategy as in Section 4.1. Following [4], we use L1 loss to train each model. The initial learning rate and weight decay are set as 0.001 and 0.0001, respectively. We train regression models with four different random seeds and report the average mean absolute error (MAE) on the test set.

Table 4: MAE results of graph regression on ZINC and QM9. Lower is better.

Methods	ZINC	QM9
GCN+SAGPool	0.378±0.031	0.545±0.010
GCN+ASAP	0.372±0.026	0.500±0.017
GCN+DiffPool	1.641±0.026	1.331±0.014
GCN+HGPSL	1.326±0.096	1.035±0.049
GCN+EdgePool	0.382±0.030	0.489±0.022
GCN+Co-Pooling (ours)	0.340±0.036	0.439±0.009

We compare our Co-Pooling method with SAGPool, ASAP, DiffPool, HGPSL, and EdgePool on the two datasets. As shown in Table 4, Co-Pooling consistently yields lower error than other baseline methods. Particularly, Co-Pooling outperforms DiffPool and HGPSL by a large margin on both datasets. These results indicate that our method effectively learns a better graph-level representation by fusing edge-view pooling and node-view pooling, leading to competitive performance on graph regression tasks as well.

5 Conclusion

We proposed a new graph pooling method (Co-Pooling) for learning graph-level representations. We argued that most of current graph pooling methods are highly node-centric and fail to leverage crucial graph substructures, which are beneficial to various prediction tasks. Our proposed Co-Pooling method fuses the pooled graph information from two views. From the edge view, generalized PageRank is used to aggregate valuable structural information from multi-hop neighbours. The proximity weights between node pairs are then calculated to prune less important edges. From the node view, the node importance scores are computed through the proximity matrix to select the top important nodes. Through cross-view interaction, edge-view pooling and node-view pooling complement each other to effectively learn informative graph representations. Extensive experiments on 16 graph datasets demonstrate the superior performance of Co-Pooling on both graph classification and regression tasks.

Acknowledgements Xiaowei Zhou is supported by a Data61 PhD Scholarship from CSIRO. Ivor W. Tsang is supported by the Center for Frontier AI research, A*STAR, and ARC under grants DP200101328. This work is partially supported by the USYD-Data61 Collaborative Research Project grant.

References

1. Chen, Z., Chen, L., Villar, S., Bruna, J.: Can graph neural networks count substructures? NeurIPS **33**, 10383–10395 (2020)

2. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. ICLR (2021)
3. Diehl, F.: Edge contraction pooling for graph neural networks. arXiv preprint arXiv:1905.10990 (2019)
4. Dwivedi, V.P., Joshi, C.K., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking graph neural networks. arXiv preprint arXiv:2003.00982 (2020)
5. Galland, A., marc lelarge: Graph pooling by edge cut (2021)
6. Gao, H., Ji, S.: Graph u-nets. ICML. pp. 2083–2092. PMLR (2019)
7. Gao, X., Dai, W., Li, C., Xiong, H. & Frossard, P. iPool–Information-Based Pooling in Hierarchical Graph Neural Networks. IEEE TNNLS. pp. 1-13 (2021)
8. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. NIPS. pp. 1025–1035 (2017)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. ICLR (2017)
10. Lee, J., Lee, I. & Kang, J. Self-attention graph pooling. ICML. pp. 3734-3743 (2019)
11. Liu, N., Jian, S., Li, D., Zhang, Y., Lai, Z. & Xu, H. Hierarchical Adaptive Pooling by Capturing High-order Dependency for Graph Representation Learning. IEEE TKDE. pp. 1-1 (2021)
12. Morris, C., Kriege, N., Bause, F., Kersting, K., Mutzel, P. & Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. ArXiv:2007.08663. (2020)
13. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. & Alon, U. Network motifs: simple building blocks of complex networks. Science. **298**, 824-827 (2002)
14. Orsini, F., Frasconi, P., De Raedt, L.: Graph invariant kernels. IJCAI pp. 3756-3762 (2015)
15. Ranjan, E., Sanyal, S., Talukdar, P.: Asap: Adaptive structure aware pooling for learning hierarchical graph representations. AAAI. pp. 5470–5477 (2020)
16. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. IAPR Workshops on SPR and SSPR. pp.287–297. Springer (2008)
17. Shang, J., Wang, Y., Chen, M., Dai, J., Zhou, X., Kuttner, J., Hilt, G., Shao, X., Gottfried, J.M., Wu, K.: Assembling molecular sierpiński triangle fractals. Nature chemistry **7**(5), 389–393 (2015)
18. Sun, Q., Li, J., Peng, H., Wu, J., Ning, Y., Yu, P.S., He, L.: Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. Proceedings of the Web Conference 2021. pp.2081–2091 (2021)
19. Sutherland, J.J., O'brien, L.A., Weaver, D.F.: Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships. Journal of chemical information and computer sciences **43**(6), 1906–1915 (2003)
20. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. ICLR (2018)
21. Wang, Y.G., Li, M., Ma, Z., Montufar, G., Zhuang, X., Fan, Y.: Haar graph pooling. ICML. pp. 9952–9962. (2020)
22. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? ICLR (2019)
23. Yanardag, P., Vishwanathan, S.: Deep graph kernels. SIGKDD pp. 1365-1374 (2015)

24. Ying, R., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. NIPS pp. 4805-4815 (2018)
25. Yuan, H., Ji, S.: Structpool: Structured graph pooling via conditional random fields. ICLR (2020)
26. Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., Wang, C.: Hierarchical graph pooling with structure learning. arXiv preprint arXiv:1911.05954 (2019)
27. Zhang, M., Cui, Z., Neumann, M. & Chen, Y. An end-to-end deep learning architecture for graph classification. AAAI pp. 4438-4445 (2018)