Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities

Chuang Liu\(^{1*}\), Yibing Zhan\(^{2}\), Jia Wu\(^{3}\), Chang Li\(^{2}\), Bo Du\(^{4}\), Wenbin Hu\(^{1\dagger}\), Tongliang Liu\(^{5}\) and Dacheng Tao\(^{5}\)

\(^{1}\)School of Computer Science, Wuhan University, Wuhan, China  
\(^{2}\)JD Explore Academy, JD.com, China  
\(^{3}\)School of Computing, Macquarie University, Sydney, Australia  
\(^{4}\)National Engineering Research Center for Multimedia Software, Wuhan University, Wuhan, China  
\(^{5}\)School of Computer Science, University of Sydney, Sydney, Australia

{chuangliu, dubo, hwb}@whu.edu.cn, \{zhanyibing, lichang93\}@jd.com, jia.wu@mq.edu.au  
tongliang.liu@sydney.edu.au, dacheng.tao@gmail.com

Abstract

Graph neural networks have emerged as a leading architecture for many graph-level tasks, such as graph classification and graph generation. As an essential component of the architecture, graph pooling is indispensable for obtaining a holistic graph-level representation of the whole graph. Although a great variety of methods have been proposed in this promising and fast-developing research field, to the best of our knowledge, little effort has been made to systematically summarize these works. To set the stage for the development of future works, in this paper, we attempt to fill this gap by providing a broad review of recent methods for graph pooling.

Specifically, 1) we first propose a taxonomy of existing graph pooling methods with a mathematical summary for each category; 2) then, we provide an overview of the libraries related to graph pooling, including the commonly used datasets, model architectures for downstream tasks, and open-source implementations; 3) next, we further outline the applications that incorporate the idea of graph pooling in a variety of domains; 4) finally, we discuss certain critical challenges facing current studies and share our insights on future potential directions for research on the improvement of graph pooling.

1 Introduction

Graph Neural Networks (GNNs) have achieved a substantial improvement over many graph-level tasks such as graph classification [Errica et al., 2020], graph regression [Bianchi et al., 2020a], and graph generation [Baek et al., 2021]. Specifically, GNNs have been successfully applied to graph-level tasks across a broad range of areas such as chemistry and biology [Baek et al., 2021], social network [Ma et al., 2021], computer vision [Fey et al., 2018], natural language processing [Gao et al., 2019], and recommendation [Wu et al., 2021].

Different from node-level tasks, which mainly use the graph convolutional network (GCN) [Kipf and Welling, 2017] to generate node representations for downstream tasks, graph-level tasks require holistic graph-level representations for graph-structured inputs whose size and topology are varying. Therefore, for graph-level tasks, the pooling mechanism is an essential component, which condenses the input graph with node representations generated by GCN into a smaller sized graph or a single vector, as shown in Figure 1.

In order to obtain an effective and reasonable graph representation, many designs of graph pooling have been proposed, which could be roughly divided into Flat Pooling (Section 3.1) and Hierarchical Pooling (Section 3.2). The former directly generates a graph-level representation in one step, mostly taking the average or sum over all node embeddings as the graph representation [Duvenaud et al., 2015], while the latter coarsens a graph gradually into a smaller sized graph by two main means: Node Clustering Pooling and Node Drop Pooling. Specifically, node clustering pooling [Ying et al., 2018] groups nodes into clusters as a coarsened graph, which is time-and space-consuming [Bianchi et al., 2020a]. In contrast, node drop pooling [Gao and Ji, 2019] selects a subset of nodes from the original graph to construct a coarsened graph, which is more efficient and more suitable for large-scale graphs [Lee et al., 2019] but suffers from inevitable information loss [Gao et al., 2021b].

Although such state-of-the-art graph pooling methods have been proposed, only a few recent works have attempted to comprehensively evaluate the effects of graph pooling [Mesquita et al., 2020; Grattarola et al., 2021], and a systematic review of the progress of and challenges facing this emerging area is still lacking. To fill the gaps, we comprehensively survey graph pooling in this paper, including proposing a taxonomy and formulating relevant frameworks (Section 3), overviewing libraries (Section 4), outlining applications (Section 5), and discussing future research directions (Section 6). To the best of our knowledge, our paper is the first attempt to present a systematic and comprehensive
review of recent progress on graph pooling. The purpose of this paper is to provide new practitioners with a comprehensive understanding of graph pooling and to keep researchers informed about the latest advancements in this field.

2 Problem Formulation

Notions. Let \( G = (\mathcal{V}, \mathcal{E}) \) denote a graph with node set \( \mathcal{V} \) and edge set \( \mathcal{E} \). Node features are denoted as \( X \in \mathbb{R}^{n \times d} \), where \( n \) is the number of nodes, and \( d \) is the dimension of node features. The adjacency matrix is defined as \( A \in \{0, 1\}^{n \times n} \). \( A[i, j] = 1 \) if there exists an edge between node \( v_i \) and node \( v_j \), otherwise, \( A[i, j] = 0 \).

Graph Pooling. Let a graph pooling operator be defined as any function \( \text{POOL} \) that maps a graph \( G \) to a new pooled graph \( G' = (\mathcal{V}', \mathcal{E}') \):

\[
G' = \text{POOL}(G),
\]

where \(|\mathcal{V}'| < |\mathcal{V}| \).

The primary goal of graph pooling is to reduce the number of nodes in a graph while preserving the semantic information of the graph.

3 Approaches for Graph Pooling

Graph pooling can be roughly divided into flat pooling and hierarchical pooling according to its role in graph-level representation learning. The former directly generates graph-level representations in a single step (\(|\mathcal{V}'| = 1\)), while the latter coarsens the graph gradually into a smaller sized graph (\(|\mathcal{V}'| > 1\)).

3.1 Flat Pooling

Flat pooling, also known as graph readout operation, directly generates a graph-level representation \( h_G \) in one step. Thus, Eq. 1 in the case of flat pooling can be denoted as:

\[
h_G = \text{POOL}_{\text{flat}}(G),
\]

where \( \text{POOL}_{\text{flat}} \) denotes the graph pooling function, which must: 1) output fixed-sized graph representations when input graphs are of different sizes; 2) output the same representation when the order of nodes of an input graph changes.

In light of the above discussions, several designs of flat pooling layers have been proposed. The most commonly used method is the sum-pool or mean-pool, which performs averaging or summing operations over all node representations [Duvanau et al., 2015; Xu et al., 2019]. Some methods [Navarin et al., 2019; Chen et al., 2019; Papp et al., 2021] perform an additional non-linearity transformation to improve the expressive power of pooling methods. Moreover, some methods [Li et al., 2016; Atwood and Towsley, 2016; Bai et al., 2019; Fan et al., 2020; Itoh et al., 2022] introduce the soft attention mechanism to determine the weight of each vertex in the final graph-level representation. Besides, some methods [Zhang et al., 2018; Bai et al., 2021] apply convolutional neural networks to sorted node representations. Different from the above methods, which collect the first-order statistic information of node representations, SOPool, proposed by Wang et al. [2020], considers the important second-order statistics, which refers to functions that utilize the second power of node features. Furthermore, DKEPool [Chen et al., 2022] takes into consideration the entire node distribution of a graph. Due to space limitation, some other flat pooling methods such as Set2set [Vinyals et al., 2016], DEMO-Net [Wu et al., 2019], SSRead [Lee et al., 2022], and GMT [Baek et al., 2021] are not presented here.

Most of the above flat pooling methods perform operations on node representations to obtain graph-level representations without consideration of the intrinsic hierarchical structures of graphs, which causes information loss and degrades the performance of graph representations [Knyazev et al., 2019; Bianchi and Lachi, 2023].

3.2 Hierarchical Pooling

Hierarchical pooling methods aim to preserve the hierarchical graph’s structural information by iteratively coarsening the graph into a new graph in smaller size. The hierarchical pooling can be roughly classified into node clustering pooling, node drop pooling, and other pooling according to the manner in which it coarsens a graph. The main difference between the first two types of methods is that node clustering pooling generates new nodes for the coarsened graph, whereas node drop pooling retains nodes from the original graph. Note that hierarchical pooling methods still technically employ flat pooling methods (readout in Figure 1) to obtain the graph-level representation of the coarsened graph.

Node Clustering Pooling

Node clustering pooling considers graph pooling as a node clustering problem, which maps the nodes into a set of clusters. After that, the clusters are treated as new nodes of the new coarsened graph. To gain a better insight into node
clustering pooling, we propose a universal and modularized framework to describe the process of node clustering pooling. Specifically, we deconstruct node clustering pooling into two disjoint modules: 1) Cluster Assignment Matrix (CAM) Generator. Given an input graph, the CAM generator predicts the soft / hard assignment for each node. 2) Graph Coarsening. With the assignment matrix, a new graph coarsened from the original one is obtained by learning a new feature matrix and an adjacency matrix. The process can be formulated as follows:

\[ C^{(l)} = \text{CAM}(X^{(l)}, A^{(l)}); \]

\[ X^{(l+1)}, A^{(l+1)} = \text{COARSEN}(X^{(l)}, A^{(l)}, C^{(l)}), \]  \( l \) \( \in \mathbb{N} \)

(3)

where functions CAM and COARSEN are specially designed by each method, respectively. \( C^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}} \) indicates the learned cluster assignment matrix; \( n_l \) is the number of nodes (or clusters) at layer \( l \).

Accordingly, we show how existing node clustering pooling methods fit into our proposed framework, and select seven typical methods presented in Table 1. We observe that these methods, with the same coarsening module choice, mainly differ in the way CAM is generated. 1) CAM Generator. Node clustering methods generate CAM from different perspectives. Specifically, DiffPool [Ying et al., 2018] directly employs GNN models, and StructPool [Yuan and Ji, 2020] extends DiffPool by explicitly capturing high-order structural relationships; LaPool [Noutahi et al., 2019] and MinCutPool [Bianchi et al., 2020a] both design the generator from the perspective of spectral clustering; MemPool [Kasahmadi et al., 2020] introduces a clustering-friendly distribution to generate the cluster matrix. 2) Graph Coarsening. Most node clustering methods adopt nearly the same coarsening strategy: the pooled node representations, \( X^{(l+1)} \in \mathbb{R}^{m_{l+1} \times d} \), obtained by the sum of representations of the nodes in each cluster and weighted by the cluster assignment scores; the coarsened adjacency matrix, \( A^{(l+1)} = C^{(l)T} A C^{(l)} \in \mathbb{R}^{m_{l+1} \times m_{l+1}} \), which indicates the connectivity strength between different clusters, is obtained by the weighted sum of edges between clusters.

Due to space limitation, many other node clustering pool-
<table>
<thead>
<tr>
<th>Models</th>
<th>Score Generator</th>
<th>Node Selector</th>
<th>Graph Coarsening</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopKPool[1]</td>
<td>$S = Xp /</td>
<td></td>
<td>p</td>
</tr>
<tr>
<td>SAGPool[2]</td>
<td>$S = \text{GNN}(X, A)$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>AttnPool[3]</td>
<td>$S = \text{softmax}(WX)$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = A_{\text{sh}}(X \odot S)$; $A' = A_{\text{sh}, \text{sh}}A'_{\text{sh}}$</td>
</tr>
<tr>
<td>ASAP[4]</td>
<td>$S = L.E.Cwire(X^*, A)$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>HGP-SL[5]</td>
<td>$S = | (I - D^{-1}A) X |_1$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>VIPool[6]</td>
<td>${ p = \frac{1}{2} \sum_{i,j} (D^{-1}A^{-1}X)^T W^\text{MLP}(X) } \in \text{TOP}_K(S)$</td>
<td>${ p = \frac{1}{2} \sum_{i,j} (D^{-1}A^{-1}X)^T W^\text{MLP}(X) } \in \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>RepPool[7]</td>
<td>$S = \sigma(D^{-1}AXp /</td>
<td></td>
<td>p</td>
</tr>
<tr>
<td>GSAPool[8]</td>
<td>${ S_1 = \text{GNN}(X, A)$, $S_2 = \sigma(AMLP(X))$, $S = S_1 \odot (1 - \alpha) S_2 }$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = (AXW_{\text{sh}} \odot S_{\text{sh}})$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>PANPool[9]</td>
<td>${ Z = \sum_{a=0}^{N} \sum_{j=1}^{N} g(i, j, n)$, $M = Z^{-1} \sum_{j=1}^{N} \sum_{i=0}^{N} A^{i,j}$, $S = Xp + \beta (\text{diag}(M)) }$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>CGPool[10]</td>
<td>${ S_1 = \text{GNN}(X, A)$, $S_2 = \sigma(S_1 - S_2)$, $S = S_1 \odot S_2 }$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>TAPool[11]</td>
<td>${ S_1 = \text{softmax}(\Gamma_0(X^T \odot (D^{-1}A)))$, $S_2 = \text{softmax}(\Gamma_0(D^{-1}AXp))$, $S = S_1 + S_2$ }</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}} \odot S_{\text{sh}}$; $A' = A_{\text{sh}, \text{sh}}$</td>
</tr>
<tr>
<td>IPool[12]</td>
<td>$S = | (I - \sum_{i=1}^N (D^{-1}A^T) X)_i |^2$</td>
<td>$\text{idx} = \text{TOP}_K(S)$</td>
<td>$X' = X_{\text{sh}}$, $A' = \lambda (A + I_{\text{sh}, \text{sh}})/(\lambda + 1)$</td>
</tr>
</tbody>
</table>

**Notations:** $X \in \mathbb{R}^{n \times d}$ and $A' \in \{0, 1\}^{n \times k}$ are the adjacency matrix and feature matrix for the new graph, respectively; $D \in \mathbb{R}^{n \times n}$ is the degree matrix of $A$; $A^\text{h}$ is the matrix where diagonal values correspond to the $h$-hop circles have been removed; $D^\text{h} \in \mathbb{R}^{n \times n}$ is the corresponding degree matrix of $A^\text{h}$; $W \in \mathbb{R}^{k \times d}$ and $W_{\text{sh}} \in \mathbb{R}^{k \times d}$ are the learnable weight matrices; $p \in \mathbb{R}^d$ and $a \in \mathbb{R}^n$ are the trainable projection vectors; $I$ is the identity matrix; $\lambda$ is a trade-off parameter; $\alpha$ is a user-defined hyperparameter; $L \in \mathbb{R}^{n \times n}$ is a vector with all elements being 1; $M \in \mathbb{R}^{n \times k}$ is a masking matrix; $O$ is the matrix used for measuring the overlap between node neighbors; $\sigma$ is the activation function ($\sigma$, tanh); $\odot$ is the broadcasted elementwise product; MLP is a multi-layer perceptron; SEL$_a$ is the algorithm used for selecting nodes one by one.

Table 2: Summary of representative node drop pooling methods in our framework.
lector, and graph coarsening, respectively. $S^{(i)} \in \mathbb{R}^{n \times 1}$ indicates the significance scores; $\text{TOP}_k$ ranks values and returns the indices of the largest $k$ values in $S^{(i)}$; $\text{idx}^{(i+1)}$ indicates the reserved node indexes for the new graph.

Accordingly, we present how the nine typical node drop pooling methods fit into our proposed framework in Table 2. Intuitively, methods tend to design more sophisticated score generators and more reasonable graph coarsenings to select more representative nodes and return more important structural information, respectively, thus alleviating the problem of information loss. 1) **Score Generator.** Different from TopKPool [Gao and Ji, 2019], SAGPool [Lee et al., 2019], and HGP-SL [Zhang et al., 2020b], which predict scores from a single view, GSPPool [Zhang et al., 2020a] and TAPool [Gao et al., 2021a] generate scores from two different views, i.e., local and global views. 2) **Node Selector.** Most methods simply adopt $\text{TOP}_k$ as a selector, and only a few works [Li et al., 2020a; Qin et al., 2020] design different selectors. 3) **Graph Coarsening.** Instead of directly obtaining the coarsened graph formed by the selected nodes, such as TopKPool [Gao and Ji, 2019], SAGPool [Lee et al., 2019], and TAPool [Gao et al., 2021a], RepPool [Li et al., 2020a], GSPPool [Zhang et al., 2020a], and IPool [Gao et al., 2021b] utilize both the selected nodes and non-selected nodes to maintain more structural and feature information in a graph.

Due to space limitation, many other node drop pooling methods [Cangea et al., 2018; Kniazev et al., 2019; Ranjan et al., 2020; Ma et al., 2020; Bianchi et al., 2020b; Gao et al., 2020; Li et al., 2020b; Zhang et al., 2021; Tang et al., 2021; Liu et al., 2022; Liu et al., 2023] are not presented in Table 2. Though more efficient and more applicable to large-scale graph datasets [Cangea et al., 2018] than node clustering pooling methods, node drop pooling methods suffer from inevitable information loss [Gao et al., 2021b; Back et al., 2021; Liu et al., 2022a].

**Other Pooling**

Apart from node drop and node clustering pooling methods, there also exist some other graph pooling methods. For example, EdgePool [Diehl, 2019] and HyperDrop [Jo et al., 2021] pool the input graph from the edge view, which maintains the connectivity of the input graph through edge contractions based on an adaptive edge scoring design. MuchPool [Du et al., 2021] combines node clustering pooling and node drop pooling to capture different characteristics of a graph; PAS [Wei et al., 2021] proposes to search for adaptive pooling architectures by neural architecture search.

## 4 Libraries for Graph Pooling

**Benchmark Datasets.** Table 3 provides the statistics of commonly used datasets for evaluating graph pooling methods, which mainly come from two widely used repositories: TU dataset [Morris et al., 2020], which contains over 130 datasets varying in content domains and dataset sizes, and Open Graph Benchmark (OGB) dataset [Hu et al., 2020], which contains many large-scale benchmark datasets. The above datasets can be classified into four categories: 1) Social Networks. The social networks consider entities as nodes, and their social interactions as edges. 2) Protein Networks. The commonly used datasets include PROTEINS and D&D, where nodes correspond to amino acids, and edges

| Datasets       | Category   | # graphs | # classes | Avg. $|V|$ | Avg. $|E|$ | Task      |
|---------------|------------|----------|-----------|--------|---------|-----------|
| MUTAG         | Molecule   | 1,882    | 2         | 17.93  | 19.79   | Cla.      |
| PROTEINS      | Protein    | 600      | 6         | 32.63  | 124.20  | Cla.      |
| ENZYMES       | Protein    | 1,113    | 2         | 39.06  | 72.82   | Cla.      |
| NCI1          | Molecule   | 4,110    | 2         | 29.87  | 32.30   | Cla.      |
| NCI109        | Molecule   | 4,127    | 2         | 29.68  | 32.13   | Cla.      |
| MUTAG         | Molecule   | 182      | 2         | 17.93  | 19.79   | Cla.      |
| PTO_M         | Molecule   | 344      | 2         | 14.30  | 14.69   | Cla.      |
| MUTAGENCYITIZE| Molecule   | 4,337    | 2         | 30.32  | 30.77   | Cla.      |
| FRANKENSTEIN  | Molecule   | 4,337    | 2         | 16.90  | 17.88   | Cla.      |
| REDDIT-BINARY | Social     | 2,000    | 2         | 429.63 | 497.75  | Cla.      |
| REDDIT-M5K    | Social     | 4,999    | 5         | 508.52 | 594.87  | Cla.      |
| REDDIT-M12K   | Social     | 11,929   | 11        | 391.41 | 456.89  | Cla.      |
| IMDB-BINARY   | Social     | 1,000    | 2         | 19.77  | 96.53   | Cla.      |
| IMDB-MULTI    | Social     | 1,500    | 3         | 13.00  | 65.94   | Cla.      |
| COLLAB        | Social     | 5,000    | 3         | 74.49  | 2457.78 | Cla.      |
| QM9           | Molecule   | 133,885  | –         | 18.03  | 18.63   | Reg.      |
| MoleculeNet(1) | Molecule   | 7,165    | 1         | 61.31  | 91.03   | Reg.      |
| QM8           | Molecule   | 21,786   | 12        | 61.31  | 91.03   | Reg.      |
| ESOG          | Molecule   | 1,128    | 1         | 61.31  | 91.03   | Reg.      |
| FREESOLV      | Molecule   | 643      | 1         | 20.85  | 32.74   | Reg.      |
| LIPOPHILICITY | Molecule   | 4,200    | 1         | 20.85  | 32.74   | Reg.      |
| COLORS-3      | Synthetic  | 5,500    | 5         | 61.31  | 91.03   | Cla.      |
| TRIANGLES     | Synthetic  | 45,000   | 10        | 20.85  | 32.74   | Cla.      |
| MNIST         | Image      | 60,000   | 10        | 70     | 91.03   | Cla.      |
| CIFAR10       | Image      | 70,000   | 10        | 117    | 32.74   | Cla.      |

| Task          | | | | | | | |
|---------------|------------|----------|-----------|--------|---------|-----------|

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{MP}$</td>
<td>$\text{NC}$</td>
<td>$\tilde{C}$</td>
<td>$\text{MP}$</td>
<td>$\text{FP}$</td>
</tr>
<tr>
<td>$\text{MP}$</td>
<td>$\text{HP}$</td>
<td>$\text{MP}$</td>
<td>$\text{FP}$</td>
<td>$\text{MLP}(\cdot)$</td>
</tr>
<tr>
<td>$\text{MP}$</td>
<td>$\text{MP}$</td>
<td>$\text{FP}$</td>
<td>$\text{MLP}(\cdot)$</td>
<td>$\tilde{y}$</td>
</tr>
<tr>
<td>$\text{MP}$</td>
<td>$\text{MP}$</td>
<td>$\text{HP}$</td>
<td>$\text{MP}$</td>
<td>$\text{FP}$</td>
</tr>
<tr>
<td>$\text{MP}$</td>
<td>$\text{MP}$</td>
<td>$\text{HP}$</td>
<td>$\text{MP}$</td>
<td>$\text{MP}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Clustering</td>
<td>Graph Regression</td>
<td>(c) Flat Graph Classification</td>
<td>Hierarchical Graph Classification</td>
<td>Graph Reconstruction / Node Classification</td>
</tr>
</tbody>
</table>

Figure 2: Illustration of model architectures for different tasks.
Graphs are constructed if the distance between two nodes is below 6 Angstroms. 3) Molecule Graphs. The commonly used molecular datasets include NCI1 and MUTAG, in which nodes and edges refer to atoms and bonds, respectively. 4) Others. Besides the above three types of datasets, there are also some less commonly used datasets, such as synthetic datasets introduced by Knyazev et al. [2019], and image datasets [Dwivedi et al., 2020], which are converted into graphs using super-pixels. The former is used to evaluate the generalization capability of pooling methods, and the latter is usually used to visualize the preserved information via graph visualization. The former is used to evaluate the interpretability of pooling methods.

Table 4: A list of representative graph pooling methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Task</th>
<th>Dataset</th>
<th>Venue</th>
<th>Code Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>SortPool</td>
<td>Graph Classification</td>
<td>D&amp;D, PROTEINS, NCI1, MUTAG, PTC, COLLAB, IMDB-B (M)</td>
<td>AAAI'2018</td>
<td><a href="https://shorturl.at/j0A9">https://shorturl.at/j0A9</a></td>
</tr>
<tr>
<td>SOPool</td>
<td>Graph Classification</td>
<td>PROTEINS, NCI1, MUTAG, PTC, IMDB-B (M), COLLAB, RDT-B, RDT-M5K</td>
<td>TPAMI'2020</td>
<td>--</td>
</tr>
<tr>
<td>GMT</td>
<td>Graph Classification</td>
<td>D&amp;D, PROTEINS, MUTAG, IMDB-B (M), COLLAB, HIV, TOX21, TOXIC, BBBP</td>
<td>ICLR'2021</td>
<td><a href="https://shorturl.at/iorN3">https://shorturl.at/iorN3</a></td>
</tr>
<tr>
<td>DKEPool</td>
<td>Graph Classification</td>
<td>PROTEINS, NCI1, MUTAG, PTC, IMDB-B (M), HIV, BBBP</td>
<td>TKDE'2022</td>
<td><a href="https://shorturl.at/drz12">https://shorturl.at/drz12</a></td>
</tr>
</tbody>
</table>

| DiffPool     | Graph Classification | D&D, PROTEINS, ENZYMES, COLLAB, RDT-M12K                              | NeurIPS'18| https://shorturl.at/afB3       |
| MinCutPool   | Node Clustering      | D&D, PROTEINS, ENZYMES, IMDB-B (M), COLLAB                            | ICML'2020 | https://shorturl.at/joy9       |
| MemPool      | Graph Regression     | D&D, PROTEINS, ENZYMES, COLLAB, RDT-B                                | ICLR'2020 | https://shorturl.at/dqyY4      |
| StructPool   | Graph Classification | PROTEINS, ENZYMES, MUTAG, PTC, IMDB-B (M), COLLAB                     | ICLR'2020 | https://shorturl.at/ijJ01      |
| HocPool      | Node Clustering      | D&D, PROTEINS, NCI1, MUTAGEN., RDT-B, COX2-MD, ER-MD                  | CIKM'2022 | https://shorturl.at/abR26      |
| SEP          | Graph Classification | D&D, PROTEINS, MUTAG, NCI1, IMDB-B (M), COLLAB                       | ICML'2022 | https://shorturl.at/tovDL      |

| TopKPool     | Graph Classification | D&D, PROTEINS, COLLAB                                                | ICML'2019 | https://shorturl.at/cjnr       |
| SAGPool      | Graph Classification | D&D, PROTEINS, NCI1, NCI109, FRANK.                                  | ICML'2019 | https://shorturl.at/bEJNQ      |
| ASAP         | Graph Classification | D&D, PROTEINS, ENZYMES, NCI1, NCI109, MUTAGEN.                      | AAAI'2020 | https://shorturl.at/depuz      |
| VIPool       | Node Clustering      | D&D, PROTEINS, ENZYMES, IMDB-B (M), COLLAB                           | NeurIPS'20| https://shorturl.at/luvz1      |
| GSAPool      | Graph Classification | D&D, NCI1, NCI109, MUTAGEN.                                         | WWW'2020  | https://shorturl.at/ciS02      |
| PANPool      | Graph Classification | PROTEINS, PROTEINS-FULL, NCI1, MUTAGEN., AIDS                        | NeurIPS'20| https://shorturl.at/OP015       |
| TAPOOL       | Graph Classification | D&D, PROTEINS, MUTAG, PTC                                            | TPAMI'2021| --                            |
| IPool        | Graph Classification | D&D, PROTEINS, ENZYMES, NCI1, NCI109, MNIST, CIFAR10                | TNNLS'2021| --                            |
| AdamGNN      | Node Classification | D&D, PROTEINS, MUTAG, NCI109, MUTAGEN.                               | TKDE'2022 | https://shorturl/atQWz         |

[1][Zhang et al., 2018]; [2][Wang and Ji, 2020]; [3][Baek et al., 2021]; [4][Chen et al., 2022]; [5][Ying et al., 2018]; [6][Bianchi et al., 2020a]; [7][Khasahmadi et al., 2020]; [8][Yuan and Ji, 2020]; [9][Dwivedi and Malliaros, 2022]; [10][Wu and others, 2022]; [11][Gao and Ji, 2019]; [12][Lee et al., 2019]; [13][Ranjian et al., 2020]; [14][Li et al., 2020b]; [15][Zhang et al., 2020a]; [16][Ma et al., 2020]; [17][Gao et al., 2021a]; [18][Gao et al., 2021b]; [19][Zhang et al., 2021]; [20][Zhong et al., 2022].

Models with ♦ have another implementation by TensorFlow available in Spectral: https://graphneural.network/.
**Model Architectures.** The graph pooling methods are generally evaluated by two levels of graph tasks: 1) **Node-level tasks** include node classification and node clustering tasks, which are generally tested on a single graph in the form of transductive learning; 2) **Graph-level tasks** include graph classification, graph regression, graph reconstruction, and graph generation tasks, which are usually tested on multiple graphs in the form of inductive learning. Figure 2 summarizes the model architectures commonly used for various tasks. The abbreviations used in the figure are MP for message passing, NC for node clustering, HP for hierarchical pooling, FP for flat pooling, and UP for unpooling. In addition to the above task-oriented evaluations, Daniele et al. [2021] provided another two evaluation criteria, including preserving the information content of the node attributes and preserving the topological structure, which help to comprehensively quantify the ability of graph pooling.

**Codes.** To facilitate the access to empirical analysis, we summarize the open-source codes of representative graph pooling studies in Table 4. Meanwhile, we list the applied tasks and corresponding benchmark datasets of each method. Due to space limitation, a more complete summary (over 150 papers reviewed) is presented in our GitHub repository. Moreover, we will update the repository in real-time as more methods and their implementations become available.

**5 Applications**

We briefly review the recent studies that incorporate the idea of graph pooling on a wide range of applications, which can be divided into two classes according to the type of datasets: 1) **Structural datasets**, where the data have explicit relational structures, such as molecular property prediction [Wang et al., 2020; Khasahmadi et al., 2020], molecular generation [Baek et al., 2021], protein-ligand binding affinity prediction [Li et al., 2021b], 3D protein structure analysis [Hermosilla et al., 2021], drug discovery [Gaudete et al., 2021], recommendation [Wu et al., 2021; Chang et al., 2021; Liu et al., 2021b], community detection [Liu et al., 2020], and relation extraction of knowledge graph [Nadgeri et al., 2021].

2) **Non-structural datasets**, where the relational structures are implicit, and the graphs are built according to domain knowledge, such as cancer diagnosis [Rhee et al., 2018; Adnan et al., 2020], brain data analysis [Li et al., 2020c; Li et al., 2021c; Kim et al., 2021; Pan et al., 2021; Demir et al., 2021; Gopinath et al., 2022], anti-spoofing and speech deepfaked detection [Tak et al., 2021], natural language processing (NLP) [Gao et al., 2019; Mao et al., 2022], computer vision (CV) [Simonovsky and Komodakis, 2017; Fey et al., 2018; Gu, 2021], 3D point clouds [Shen et al., 2018; Chen et al., 2020a; Wang et al., 2021], and multimodal sentiment analysis [Mai et al., 2020].

The aforementioned studies have effectively decreased sample size [Nadgeri et al., 2021] and incorporated hierarchical information [Chen et al., 2020a] by utilizing either pre-existing or newly developed graph pooling techniques. This has resulted in notable enhancements across diverse applications, confirming the efficacy of graph pooling methods as demonstrated through experimental results in these works [Wu et al., 2021; Pan et al., 2021; Shirian et al., 2022; Gopinath et al., 2022]. Thus, the applicability of graph pooling has been established.

**6 Challenges and Opportunities**

**6.1 Different Tasks and Multi-tasks**

**Challenge.** Table 4 demonstrates that most existing graph pooling methods focus on graph-level tasks, with few addressing node-level tasks like node classification. Graph pooling methods show promise in reducing the number of parameters, which can decrease time complexity and increase model resistance to over-fitting. Additionally, these methods have large receptive fields that capture high-order information. However, designing efficient and effective pooling/unpooling operators and integrating them into graph convolution networks for better performance in node-level tasks (e.g., node classification) are still key challenges.

**Opportunity.** Some methods attempt to handle the node classification task with an encoder-decoder learning structure as shown in Figure 2 (e), where the unpooling operation is an important component. Several node drop pooling methods [Li et al., 2020b; Zhang et al., 2021] utilize the unpooling operation proposed in the Graph U-net [Gao and Ji, 2019]. Another important component is the convolution of decoder, termed deconvolution. Besides directly adopting the convolution of encoder as deconvolution, we can design a more reasonable deconvolution, inspired by DGN [Li et al., 2021a], which reconstructs graph signals from smoothed node representations from the perspective of spectral domain. Therefore, future research directions include: 1) handling node-level tasks with an encoder-decoder structure by designing efficient unpooling and deconvolutional operators; 2) developing a unified graph pooling approach, through which node-level and graph-level tasks can be simultaneously handled [Holtz et al., 2019; Zeng et al., 2021], which is beneficial since different tasks may benefit from each other.

**6.2 Different Types of Graphs**

**Challenge.** The graph pooling methods discussed in Section 3 are primarily intended for plain graphs. However, there are also many other types of graphs in real-world datasets. Obviously, it is not optimal to directly apply the above existing graph pooling methods to these different types of graphs because they possess distinct characteristics. Therefore, there is a significant gap in the literature regarding the development of specific graph pooling operators tailored to handle diverse types of graphs.

**Opportunity.** Recently, several graph pooling works have attempted to handle the rarely studied but commonly used graphs in real applications, such as heterogeneous graphs [Wu et al., 2021], spatio-temporal graphs [Isufi and Mazzola, 2021], and hypergraphs [Jo et al., 2021]. However, specific graph pooling operators for handling other types of graphs are still lacking. Therefore, we suggest two promising research opportunities: 1) extending the existing graph pooling methods or designing new pooling methods to deal with specific...
graphs in consideration of their properties, such as dynamic graphs, where graph structures and graph nodes dynamically change over time, directed graphs, where the edges have a direction, and signed networks, where signed edges represent positive or negative relationships between nodes; 2) devising a general pooling method, which can efficiently handle different types of graphs in a universal manner.

6.3 Interpretablity

**Challenge.** Although the existing graph pooling methods have achieved excellent results on various graph-based tasks, most of them show little interpretability [Knyazev et al., 2019]. This is particularly problematic for drug or disease-related problems where understanding the reasoning behind these methods is crucial. Although there has been some progress in interpreting node representation of GNNs [Yuan et al., 2020], the interpretability of pooling methods remains largely unexplored.

**Opportunity.** To exactly explain what has been learned from graph pooling operations, some studies [Ying et al., 2018; Khosahmadi et al., 2020; Tang et al., 2021] have attempted to present the visualization of hierarchical clusters or hierarchical community structures captured by pooling operations, but they do not provide quantitative analyses to assess the quality of clustering. Moreover, the interpretability of pooling operations is still not well explored. Therefore, there is a need for further research in this area. A promising direction would be to extend existing studies on interpreting GNNs [Yuan et al., 2020] to include graph pooling.

6.4 Robustness

**Challenge.** Since many applications of graph pooling methods are risk-sensitive, e.g., drug design and disease diagnosis, the robustness of methods is essential and indispensable for actual usages. However, according to the analysis in recent studies [Tang et al., 2020; Roy et al., 2021; Liu et al., 2022a], most graph pooling methods fail to distinguish the noise information from the input graph, thus dramatically degrading their performance when the input graph is perturbed in terms of features or topology.

**Opportunity.** Although some initial studies have been conducted on the robustness of graph machine learning, few studies have explored the robustness of graph pooling methods. So far, only an adversarial attack framework has been proposed by Tang et al. [2020] to evaluate the robustness of graph pooling methods. Therefore, there is much work to be done to develop a robust graph pooling method for practical applications, such as: 1) building up a comprehensive adversarial attack framework encompasses various types of attacks; 2) designing adversarial defense graph pooling methods. One potential solution is to extend the techniques used for enhancing the robustness of graph machine learning into improving the robustness of graph pooling methods.

6.5 Large-scale Data

**Challenge.** Most graph pooling methods are tested on small benchmark datasets, which may be insufficient for comparisons among different graph pooling models. For example, applying graph pooling methods to small node classification datasets, i.e., Cora, is insufficient to assess their effectiveness in reducing the time and space complexity. Only a few works have attempted to deal with relatively larger graph datasets [Zhong et al., 2022]. Additionally, the efficiency of pooling methods is crucial. The high cost of time or space limits their applicability to large-scale datasets. Node clustering pooling methods, for example, suffer from high storage complexities due to the computation of dense assignment matrices [Grattarola et al., 2021].

**Opportunity.** Hence, we suggest the following research objectives: 1) performing further verifications on large-scale graph datasets, e.g., on the Open Graph Benchmark [Hu et al., 2020], which is a recently proposed large-scale graph machine learning benchmark; 2) designing more efficient graph pooling methods and making them more practical with constrained resource in real-world scenarios.

6.6 Expressive Power

**Challenge.** Most existing graph pooling methods are designed by intuition, and their performance gains are evaluated by empirical experiments. The lack of the means by which we can characterize the expressive power of graph pooling operators hinders the creation of more powerful graph pooling models [Xu et al., 2019].

**Opportunity.** Only a limited number of studies [Murphy et al., 2019; Chen et al., 2020; Baek et al., 2021] have examined the expressive ability of their models in terms of the 1-Weisfeiler-Lehman (WL) test. Recently, Bianchi et al. [2023] conducted an comprehensive analysis on the expressive power of graph pooling techniques. Therefore, based on their theoretical findings, it is promising and significant to explore more powerful graph pooling methods as future research directions.

6.7 Generalize to Out-of-Distribution Data.

**Challenge.** Graph Neural Networks are proposed without considering the agnostic distribution shifts between training graphs and testing graphs, causing the degeneration of the generalization ability in out-of-distribution (OOD) settings. Boris et al. [Knyazev et al., 2019] have attempted to improve the generalization ability of GNNs with the help of graph pooling models. Moreover, Xu et al. [Xu et al., 2021] emphasized the significance of selecting appropriate pooling functions for enabling GNNs to generalize over graph data beyond the distribution of training data.

**Opportunity.** Recently, to improve the OOD generalization ability of GNNs has become an appealing and non-trivial task. According to the above introduction, utilizing graph pooling techniques can be an effective approach to improve the generalization capability of GNNs on OOD graph data.

**Acknowledgments**

This work was supported in part by the Natural Science Foundation of China (Nos. 61976162, 82174230, 62002090), Artificial Intelligence Innovation Project of Wuhan Science and Technology Bureau (No. 2022010702040070), and Science and Technology Major Project of Hubei Province (Next
References


