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

Chuang Liu[1*], **Yibing Zhan**[2], **Chang Li**[2], **Bo Du**[1], **Jia Wu**[3], **Wenbin Hu**[1†], **Tongliang Liu**[4] and **Dacheng Tao**[2]
[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]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 with a notable improvement. Among these tasks, graph pooling is an essential component of graph neural network architectures for obtaining a holistic graph-level representation of the entire 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 methods. 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 on graph pooling. Specifically, 1) we first propose a taxonomy of existing graph pooling methods and provide a mathematical summary for each category; 2) next, 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) then, we further outline in brief the applications that incorporate the idea of graph pooling in a number of domains; 4) and finally, we discuss some critical challenges faced by the current studies and share our insights on potential directions for improving graph pooling in the future.

1 Introduction

Graph Neural Networks (GNNs) have achieved a substantial improvement on 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 for graph-level tasks have been successfully applied to an extensive 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].

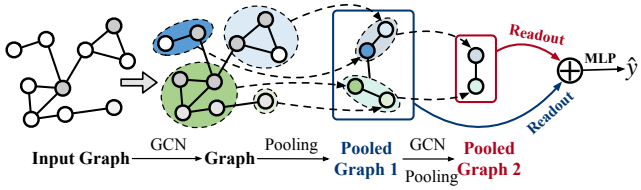


Figure 1: An illustrative example of graph pooling.

Different from node-level tasks, which mainly leverage the graph convolutional network (GCN) [Kipf and Welling, 2017; Hamilton *et al.*, 2017] to generate node representations for downstream tasks, graph-level tasks require holistic graph-level representations for graph-structured inputs of which the 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 learned by GCN into a smaller sized graph or a single vector, as shown in Figure 1.

To obtain an effective and reasonable graph representation, many designs of graph pooling have been proposed, which could be roughly divided into **flat pooling** and **hierarchical pooling**. 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 the graph gradually into a smaller sized graph, mainly through two means: **node clustering pooling** [Ying *et al.*, 2018] and **node drop pooling** [Gao and Ji, 2019]. Specifically, node clustering pooling generates new nodes to construct the coarsened graph by clustering nodes into clusters, which is time-and space-consuming [Bianchi *et al.*, 2020a]. In contrast, node drop pooling selects a subset of nodes from the original graph to construct the coarsened graph, which is more efficient and more fit for large-scale graphs [Lee *et al.*, 2019] but suffers from inevitable information loss [Gao *et al.*, 2021b].

[*]This work have been done when Chuang Liu worked as an intern at JD Explore Academy.
[†]Corresponding Author

Although such state-of-the-art graph pooling methods have been proposed, only a few recent works have attempted to systematically evaluate the effect of graph pooling [Mesquita *et al.*, 2020; Grattarola *et al.*, 2021], and a systematic review of the progress and the challenges faced in this emerging area is still missing. To this end, we comprehensively survey graph pooling in this paper, including proposing a taxonomy and formulating relevant frameworks; overviewing libraries; outlining applications; and discussing future directions. To the best of our knowledge, this paper is the first attempt to present a systematic and comprehensive review of recent progress in graph pooling.

2 Problem Formulation

Notions. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. The node features are denoted as $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the dimension of node features. The adjacency matrix is defined as $\boldsymbol{A} \in \{0,1\}^{n \times n}$.

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

$$\mathcal{G}' = \mathrm{POOL}(\mathcal{G}), \tag{1}$$

where $|\mathcal{V}'| < |\mathcal{V}|$ [1]. The generic 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 could be roughly divided into flat pooling and hierarchical pooling according to their roles in graph-level representation learning. The former 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

The flat pooling, also known as graph readout operation, directly generates a graph-level representation $\boldsymbol{h}_G$ in one step. Thus, Eq. 1 in the case of flat pooling can be expressed as:

$$\boldsymbol{h}_G = \mathrm{POOL}_{\text{flat}}(\mathcal{G}), \tag{2}$$

where $\mathrm{POOL}_{\text{flat}}$ denotes the graph pooling function, which must: **1)** output the 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.

Following 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 [Duvenaud *et al.*, 2015; Xu *et al.*, 2019]. Following the above methods, some methods [Navarin *et al.*, 2019; Chen *et al.*, 2019; Papp *et al.*, 2021] perform an additional nonlinearity 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 the node representations, SOPool, proposed by Wang et al. [2020], considers the important second-order statistics. Furthermore, DKEPool [Chen *et al.*, 2021b] takes into consideration the entire node distribution of a graph. Due to the space limit, some other flat pooling methods such as Set2set [Vinyals *et al.*, 2016], DEMO-Net [Wu *et al.*, 2019], SSRead [Lee *et al.*, 2021], and GMT [Baek *et al.*, 2021], are not presented here in detail.

However, most of the above flat pooling methods perform operations on node representations to obtain graph-level representations without consideration of the hierarchical structures of graphs, which causes information loss and downgrades the performance of graph representations.

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 of a smaller size. The hierarchical pooling can be roughly classified into node clustering pooling, node drop pooling, and other pooling according to the manners in which they coarsen the graph. The main difference between the former two types of methods is that node clustering pooling generates new nodes for the coarsened graph while node drop pooling retains nodes from the original graph. Note that hierarchical pooling methods technically still employ flat pooling methods (readout) 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. Then, the clusters are treated as new nodes of the new coarsened graph. To get a better understanding of node clustering pooling, we propose a universal and modularized framework to describe the process of node clustering pooling. Specifically, we deconstruct node clustering pooling with 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 adjacency matrix. The process can be formulated as follows:

$$\begin{aligned} &\underbrace{\boldsymbol{C}^{(l)} = \mathrm{CAM}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)})}_{\text{CAM Generator}}; \\ &\underbrace{\boldsymbol{X}^{(l+1)}, \boldsymbol{A}^{(l+1)} = \mathrm{COARSEN}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)}, \boldsymbol{C}^{(l)})}_{\text{Graph Coarsening}}, \end{aligned} \tag{3}$$

where functions CAM and COARSEN are specially designed by each method, respectively. $\boldsymbol{C}^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ indi-

[1] In some very specific cases, there exits $|\mathcal{V}'| \geq |\mathcal{V}|$, causing the graph to be upscaled by pooling.

Table 1: Summary of representative node clustering pooling methods in our framework.

Models	CAM Generator	Graph Coarsening	Notations
DiffPool [2018]	$\boldsymbol{C} = \mathrm{softmax}(\mathrm{GNN}_{\mathrm{pool}}(\boldsymbol{X}, \boldsymbol{A}))$	$\begin{cases} \boldsymbol{X}' = \boldsymbol{C}^T \cdot \mathrm{GNN}_{\mathrm{emb}}(\boldsymbol{X}, \boldsymbol{A}) \\ \boldsymbol{A}' = \boldsymbol{C}^T \boldsymbol{A} \boldsymbol{C} \end{cases}$	Auxiliary Loss[1]
NMF [2019]	$\begin{cases} \boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V} \\ \boldsymbol{C} = \boldsymbol{V}^T \end{cases}$	$\boldsymbol{X}' = \boldsymbol{C}^T \boldsymbol{X}; \boldsymbol{A}' = \boldsymbol{C}^T \boldsymbol{A} \boldsymbol{C}$	Matrix Decomposition
LaPool [2019]	$\begin{cases} \boldsymbol{S} = \lVert \boldsymbol{L}\boldsymbol{X} \rVert_2 \\ \mathcal{V}_c = \{v_i \in \mathcal{V} \mid \forall v_j, s_i - \boldsymbol{A}_{ij} s_j > 0\} \\ \boldsymbol{C} = \mathrm{sparsemax}\left(\mathbf{p} \frac{\boldsymbol{X}\boldsymbol{X}_c^T}{\lVert \boldsymbol{X} \rVert \lVert \boldsymbol{X}_c \rVert}\right) \end{cases}$	$\begin{cases} \boldsymbol{X}' = \mathrm{MLP}(\boldsymbol{C}^T \boldsymbol{X}) \\ \boldsymbol{A}' = \boldsymbol{C}^T \boldsymbol{A} \boldsymbol{C} \end{cases}$	–
MinCutPool [2020a]	$\boldsymbol{C} = \mathrm{MLP}(\boldsymbol{X})$	$\begin{cases} \boldsymbol{X}' = \boldsymbol{C}^T \boldsymbol{X}; \widehat{\boldsymbol{A}} = \boldsymbol{C}^T \widetilde{\boldsymbol{A}} \boldsymbol{C} \\ \boldsymbol{A}' = \widehat{\boldsymbol{A}} - \boldsymbol{I}\mathrm{diag}(\widehat{\boldsymbol{A}}) \end{cases}$	MinCut Loss[2]
StructPool [2020]	$\boldsymbol{C}: \text{Minimiz } E(\boldsymbol{C})$[3]	$\boldsymbol{X}' = \boldsymbol{C}^T \boldsymbol{X}; \boldsymbol{A}' = \boldsymbol{C}^T \boldsymbol{A} \boldsymbol{C}$	–
MemPool [2020]	$\begin{cases} \boldsymbol{C}_{i,j} = \frac{\left(1 + \lVert \mathbf{x}_i - \mathbf{k}_j \rVert^2 / \tau\right)^{-\frac{\tau+1}{2}}}{\sum_{j'} \left(1 + \lVert \mathbf{x}_i - \mathbf{k}_{j'} \rVert^2 / \tau\right)^{-\frac{\tau+1}{2}}} \\ \boldsymbol{C} = \mathrm{softmax}\left(\Gamma\left(\Vert_{t=0}^{\lvert m \rvert} \boldsymbol{C}_t\right)\right) \end{cases}$	$\boldsymbol{X}' = \mathrm{MLP}(\boldsymbol{C}^T \boldsymbol{X})$	Auxiliary Loss[4]
HAP [2021a]	$\begin{cases} \boldsymbol{T} = \mathrm{GCont}(\boldsymbol{X}) \\ \boldsymbol{C}_{ij} = \sigma\left(\mathbf{p}^T [\boldsymbol{T}_{\mathrm{Row}_i} \Vert \boldsymbol{T}_{\mathrm{Col}_j}]\right) \\ \boldsymbol{C} = \mathrm{softmax}(\boldsymbol{C}) \end{cases}$	$\begin{cases} \boldsymbol{X}' = \mathrm{MLP}(\boldsymbol{C}^T \boldsymbol{X}), \widehat{\boldsymbol{A}} = \boldsymbol{C}^T \boldsymbol{A} \boldsymbol{C} \\ \boldsymbol{A}' = \mathrm{Gumbel\text{-}SoftMax}(\widehat{\boldsymbol{A}}) \end{cases}$	–

[1] Auxiliary loss consists of the link prediction objective and entropy regularization. [2] MinCut loss consists of cut loss, which approximates the *mincut* problem, and orthogonality loss, which encourages the assignments to be orthogonal. [3] $E(\boldsymbol{C})$ is the Gibbs energy, which consists of unary energy and pairwise energy. [4] Auxiliary loss is an unsupervised clustering loss, which encourages the model to learn clustering-friendly embeddings.

Notations: $\boldsymbol{X}' \in \mathbb{R}^{c \times d}$ and $\boldsymbol{A}' \in \mathbb{R}^{c \times c}$ are the adjacency matrix and feature matrix for the new graph, respectively; $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$ is the adjacency matrix with self loop; $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ is the graph Laplacian; $\boldsymbol{I}_k \in \mathbb{R}^{c \times c}$ is the identity matrix; $\mathbf{k} \in \mathbb{R}^d$ is a memory key vector; $\mathbf{p} \in \mathbb{R}^{1 \times 2d}$ is a trainable vector; τ is the degree of freedom of the Student's t-distribution, i.e., temperature; c is the number of clusters; $|m|$ is the number of heads; Γ is an $[1 \times 1]$ convolutional operator; $\Vert$ is the concatenation operator; GCont is an auto-learned global graph content; and $\mathrm{Gumbel\text{-}SoftMax}$ achieves soft sampling for neighborhood relationships to decrease the edge density.

Table 2: Summary of representative node drop pooling methods in our framework.

Models	Score Generator	Node Selector	Graph Coarsening
TopKPool [2019]	$\boldsymbol{S} = \boldsymbol{X}\mathbf{p} / \lVert \mathbf{p} \rVert_2$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\boldsymbol{X}' = \boldsymbol{X}_{\mathrm{idx}} \odot \sigma(\boldsymbol{S}_{\mathrm{idx}}); \quad \boldsymbol{A}' = \boldsymbol{A}_{\mathrm{idx},\mathrm{idx}}$
SAGPool [2019]	$\boldsymbol{S} = \mathrm{GNN}(\boldsymbol{X}, \boldsymbol{A})$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\boldsymbol{X}' = \boldsymbol{X}_{\mathrm{idx}} \odot \boldsymbol{S}_{\mathrm{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\mathrm{idx},\mathrm{idx}}$
AttPool [2019]	$\boldsymbol{S} = \mathrm{softmax}(\boldsymbol{X}\boldsymbol{W})$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\boldsymbol{X}' = \boldsymbol{A}_{\mathrm{idx}}(\boldsymbol{X} \odot \boldsymbol{S}); \quad \boldsymbol{A}' = \boldsymbol{A}_{\mathrm{idx}} \boldsymbol{A}_{\mathrm{idx}}^T$
HGP-SL [2020b]	$\boldsymbol{S} = \left\lVert \left(\boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}\right) \mathbf{X} \right\rVert_1$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\begin{cases} \boldsymbol{X}' = \boldsymbol{X}_{\mathrm{idx}} \odot \boldsymbol{S}_{\mathrm{idx}}; \quad \widehat{\boldsymbol{A}} = \boldsymbol{A}_{\mathrm{idx},\mathrm{idx}} \\ \boldsymbol{A}'_{ij} = \mathrm{sparsemax}(\sigma(\overrightarrow{\mathbf{a}}[\boldsymbol{X}'(i,:) \Vert \boldsymbol{X}'(j,:)]^\top) + \lambda \cdot \widehat{\boldsymbol{A}}_{ij} \end{cases}$
RepPool [2020a]	$\boldsymbol{S} = \sigma\left(\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{X}\mathbf{p} / \lVert \mathbf{p} \rVert_2\right)$	$\mathrm{idx} = \mathrm{SEL}_k(\boldsymbol{S})$	$\begin{cases} \boldsymbol{B} = \boldsymbol{X}\boldsymbol{W}_b(\boldsymbol{X}_{\mathrm{idx}})^T \\ \boldsymbol{X}' = (\mathrm{softmax}(\boldsymbol{B} \odot \boldsymbol{M}))^T (\boldsymbol{X} \odot \boldsymbol{S}) \\ \boldsymbol{A}' = (\mathrm{softmax}(\boldsymbol{B} \odot \boldsymbol{M}))^T \boldsymbol{A} (\mathrm{softmax}(\boldsymbol{B} \odot \boldsymbol{M})) \end{cases}$
GSAPool [2020a]	$\begin{cases} \boldsymbol{S}_l = \mathrm{GNN}(\boldsymbol{X}, \boldsymbol{A}) \\ \boldsymbol{S}_2 = \sigma(\mathrm{MLP}(\boldsymbol{X})) \\ \boldsymbol{S} = \alpha \boldsymbol{S}_1 + (1 - \alpha) \boldsymbol{S}_2 \end{cases}$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\boldsymbol{X}' = (\boldsymbol{A}\boldsymbol{X}\boldsymbol{W})_{\mathrm{idx}} \odot \boldsymbol{S}_{\mathrm{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\mathrm{idx},\mathrm{idx}}$
CGIPool [2021]	$\begin{cases} \boldsymbol{S}_r = \mathrm{GNN}_r(\boldsymbol{X}, \boldsymbol{A}) \\ \boldsymbol{S}_f = \mathrm{GNN}_f(\boldsymbol{X}, \boldsymbol{A}) \\ \boldsymbol{S} = \sigma(\boldsymbol{S}_r - \boldsymbol{S}_f) \end{cases}$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\boldsymbol{X}' = \boldsymbol{X}_{\mathrm{idx}} \odot \boldsymbol{S}_{\mathrm{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\mathrm{idx},\mathrm{idx}}$
TAPool [2021a]	$\begin{cases} \boldsymbol{S}_l = \mathrm{softmax}\left(\frac{1}{n}((\boldsymbol{X}\boldsymbol{X}^T) \odot (\widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}}))\mathbf{1}_n\right) \\ \boldsymbol{S}_g = \mathrm{softmax}\left(\widetilde{\boldsymbol{D}}^{-1}\widetilde{\boldsymbol{A}}\boldsymbol{X}\mathbf{p}\right) \\ \boldsymbol{S} = \boldsymbol{S}_l + \boldsymbol{S}_g \end{cases}$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\boldsymbol{X}' = \boldsymbol{X}_{\mathrm{idx}} \odot \boldsymbol{S}_{\mathrm{idx}}; \quad \boldsymbol{A}' = \boldsymbol{A}_{\mathrm{idx},\mathrm{idx}}$
IPool [2021b][1]	$\boldsymbol{S} = \left\lVert \left(\boldsymbol{I} - \frac{1}{t}\sum_{h=1}^{t} (\bar{\boldsymbol{D}}^h)^{-1} \bar{\boldsymbol{A}}^h\right) \boldsymbol{X} \right\rVert_2$	$\mathrm{idx} = \mathrm{TOP}_k(\boldsymbol{S})$	$\begin{cases} \boldsymbol{X}' = \boldsymbol{X}_{\mathrm{idx}}, \\ \boldsymbol{A}'_{ij} = \lambda (\boldsymbol{A} + \boldsymbol{I})_{\mathrm{idx}[i],\mathrm{idx}[j]} + (1 - \lambda) \boldsymbol{O}_{ij} \end{cases}$

[1] This manuscript only presents the greedy IPool strategy in this table, and there is also a local IPool strategy introduced in [Gao *et al.*, 2021b].

Notations: $\boldsymbol{X}' \in \mathbb{R}^{k \times d}$ and $\boldsymbol{A}' \in \{0, 1\}^{k \times k}$ are the adjacency matrix and feature matrix for the new graph, respectively; $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is the degree matrix of $\boldsymbol{A}$; $\bar{\boldsymbol{A}}^h \in \mathbb{R}^{n \times n}$ is the matrix where diagonal values corresponding to the h-hop circles have been removed; $\bar{\boldsymbol{D}}^h \in \mathbb{R}^{n \times n}$ is the corresponding degree matrix of $\bar{\boldsymbol{A}}^h$; $\boldsymbol{W} \in \mathbb{R}^{d \times 1}$; and $\boldsymbol{W}_b \in \mathbb{R}^{d \times d}$ are the learnable weight matrices; $\mathbf{p} \in \mathbb{R}^d$ and $\mathbf{a} \in \mathbb{R}^{1 \times 2d}$ are the trainable projection vectors; $\boldsymbol{I}$ is the identity matrix; λ is a trade-off parameter; α is a user-defined hyperparameter; $\mathbf{1}_n \in \mathbb{R}^n$ is a vector with all elements being 1; $\boldsymbol{M} \in \mathbb{R}^{n \times k}$ is a masking matrix; $\boldsymbol{O}$ is the matrix to measure the overlap of neighbors between nodes; σ is the activation function (e.g. $\tanh$); $\odot$ is the broadcasted elementwise product; MLP is a multi-layer perceptron; SEL_k is the algorithm to select nodes one by one.

cates 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 can observe that these methods, having the same coarsening module choice, mainly differ in the way CAM is generated. **1) CAM Generator.** Node clustering methods generate CAM in different views. 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; and MemPool [Khasahmadi *et al.*, 2020] imposes 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, $\boldsymbol{X}^{(l+1)} = {\boldsymbol{C}^{(l)}}^T \boldsymbol{X}^{(l)} \in \mathbb{R}^{n_{l+1} \times d}$, are obtained by the sum of representations among the nodes in each cluster, weighted by the cluster assignment scores; and the coarsened adjacency matrix, $\boldsymbol{A}^{(l+1)} = {\boldsymbol{C}^{(l)}}^T \boldsymbol{A}^{(l)} \boldsymbol{C}^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}$, which indicates the connectivity strength between different clusters, is obtained by the weighted sum of edges between clusters.

Due to the space limit, there are also many other node clustering pooling methods [Ma *et al.*, 2019; Zhou *et al.*, 2020; Bodnar *et al.*, 2020; Khasahmadi *et al.*, 2020; Wang *et al.*, 2020; Roy *et al.*, 2021; Yang *et al.*, 2021; Liang *et al.*, 2020; Su *et al.*, 2021] not presented in Table 1. Despite substantial improvements achieved on several graph-level tasks, the above methods suffer from a limitation of time and storage complexity caused by the computation of the dense cluster assignment matrix [Baek *et al.*, 2021]. Besides, as discussed in the recent work [Mesquita *et al.*, 2020], clustering-enforcing regularization is usually innocuous.

Node Drop Pooling

Node drop pooling exploits learnable scoring functions to delete nodes with comparatively lower significance scores. To give a thorough analysis of node drop pooling, we propose a universal and modularized framework, which consists of three disjoint modules: **1) Score Generator.** Given an input graph, the score generator calculates significance scores for each node. **2) Node Selector.** Node selector selects the nodes with top-k significance scores. **3) Graph Coarsening.** With the selected nodes, a new graph coarsened from the original one is obtained by learning a new feature matrix and adjacency matrix. The process can be formulated as follows:

$$\begin{gathered} \underbrace{\boldsymbol{S}^{(l)} = \mathrm{SCORE}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)})}_{\text{Score Generator}}; \quad \underbrace{\mathrm{idx}^{(l+1)} = \mathrm{TOP}_k(\boldsymbol{S}^{(l)})}_{\text{Node Selector}}; \\ \underbrace{\boldsymbol{X}^{(l+1)}, \boldsymbol{A}^{(l+1)} = \mathrm{COARSEN}(\boldsymbol{X}^{(l)}, \boldsymbol{A}^{(l)}, \boldsymbol{S}^{(l)}, \mathrm{idx}^{(l+1)})}_{\text{Graph Coarsening}}, \end{gathered} \tag{4}$$

where functions SCORE, TOP_k, and $\mathrm{COARSEN}$ are specially designed by each method for score generator, node selector, and graph coarsening, respectively. $\boldsymbol{S}^{(l)} \in \mathbb{R}^{n \times 1}$ indicates the significance scores; TOP_k ranks values and returns the indices of the largest k values in $\boldsymbol{S}^{(l)}$; and $\mathrm{idx}^{(l+1)}$ indicates the reserved node indexes for the new graph.

Accordingly, we present how the nine typical node drop pooling methods can fit into our proposed framework in Table 2. Intuitively, methods tend to design more sophisti-

Table 3: A list of commonly used and publicly accessible datasets

Collections	Datasets	Category	# graphs	# classes	Avg.$\lvert\mathcal{V}\rvert$	Avg.$\lvert\mathcal{E}\rvert$	Task
TUDataset[1]	D&D	Protein	1,178	2	284.32	715.66	Cla.
	PROTEINS	Protein	1,113	2	39.06	72.82	Cla.
	ENZYMES	Protein	600	6	32.63	124.20	Cla.
	NCI1	Molecule	4,110	2	29.87	32.30	Cla.
	NCI109	Molecule	4,127	2	29.68	32.13	Cla.
	MUTAG	Molecule	188	2	17.93	19.79	Cla.
	MUTAGENICITY	Molecule	4,337	2	30.32	30.77	Cla.
	PTC_MR	Molecule	344	2	14.30	14.69	Cla.
	FRANKENSTEIN	Molecule	4,337	2	16.90	17.88	Cla.
	REDDIT-BINARY	Social	2,000	2	429.63	497.75	Cla.
	RDT-M5K	Social	4,999	5	508.52	594.87	Cla.
	RDT-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.
	ZINC	Molecule	249,456	–	23.14	24.91	Rec.
OGB[2]	HIV	Molecule	41,127	2	25.51	27.52	Cla.
	TOX21	Molecule	7,831	12	18.57	19.3	Cla.
	TOXCAST	Molecule	8,576	617	18.78	19.3	Cla.
	BBBP	Molecule	2,039	2	24.06	26.0	Cla.
Synthetic[3]	COLORS-3	Synthetic	5,500	5	61.31	91.03	Cla.
	TRIANGLES	Synthetic	45,000	10	20.85	32.74	Cla.
CV[4]	MNIST	Image	60,000	10	70	91.03	Cla.
	CIFAR10	Image	70,000	10	117	32.74	Cla.

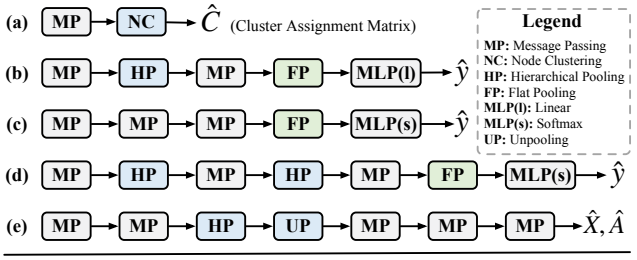
Cla., Reg., and Rec. refer to graph classification, graph regression, and graph reconstruction, respectively. [1] https://chrsmrrs.github.io/datasets/docs/datasets/; [2] https://ogb.stanford.edu/docs/graphprop/; [3] https://github.com/bknyaz/graph_attention_pool/tree/master/data; [4] https://github.com/graphdeeplearning/benchmarking-gnns

cated score generators and more reasonable graph coarsenings to select more representative nodes and to retain 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, GSAPool [Zhang *et al.*, 2020a] and TAPool [Gao *et al.*, 2021a] generate scores from two views, i.e., local and global. **2) Node Selector.** Most methods simply adopt 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], GSAPool [Zhang *et al.*, 2020a], and IPool [Gao *et al.*, 2021b] utilize both the selected nodes and the non-selected nodes to maintain more structural and feature information.

Due to the space limit, many other node drop pooling methods [Cangea *et al.*, 2018; Knyazev *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] are not presented in Table 2. Though more efficient and more applicable to large-scale networks [Cangea *et al.*, 2018] than node clustering pooling, node drop pooling suffers from inevitable information loss [Gao *et al.*, 2021b].

Other Pooling

Apart from node drop and node clustering pooling methods, there also exist some other graph pooling methods. For example, EdgePool [Diehl, 2019], Co-Pooling [Zhou *et al.*, 2021] and HyperDrop [Jo *et al.*, 2021] pool the input graph from an edge view; MuchPool [Du *et al.*, 2021] combines node clustering pooling and node drop pooling to capture different characteristics of a graph; and PAS [Wei *et al.*, 2021] proposes to search for adaptive pooling architectures by neural architecture search.



(a): Node Clustering **(b)**: Graph Regression **(c)**: Flat Graph Classification
(d): Hiearchical Graph Classification **(e)**: Graph Reconstruction / Node Classification

Figure 2: Illustration of model architectures for different tasks.

4 Libraries for Graph Pooling

Benchmark Datasets. Table 3 provides the statistics of common datasets for graph-level tasks, 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 are constructed if the distance between two nodes is less than 6 Angstroms. **3) Molecule Graphs.** The commonly used molecular datasets include NCI1 and MUTAG, in which nodes refer to atoms, and edges are of the bonds. **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 by using super-pixels.

Model Architectures. The graph pooling methods are generally evaluated on two levels of graph tasks: **1) Node-level tasks** include node classification and node clustering, 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, which are usually tested on multiple graphs in the form of inductive learning. The model architectures for the commonly used tasks are summarized in Figure 2.

Codes. To facilitate the access for empirical analysis, we summarize the source codes of representative graph pooling methods in Table 4. Due to the space limit, we provide a more complete summary (more than 130 papers reviewed) in our GitHub repository [2]. 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

[2]https://github.com/LiuChuang0059/graph-pooling-papers

Table 4: A list of twenty-five representative graph pooling methods.

Method	Type	Task	Dataset	Venue	Code Link
SortPool [Zhang *et al.*, 2018]♠♣	FP	Graph Classification	D&D, PROTEINS, NCI1, MUTAG, PTC, COLLAB, IMDB-B, IMDB-M	AAAI'2018	https://github.com/muhanzhang/pytorch_DGCNN
DiffPool [Ying *et al.*, 2018]♠♣	NC	Graph Classification	D&D, PROTEINS, ENZYMES, COLLAB, RDT-M12K	NIPS'2018	https://github.com/RexYing/diffpool
TopKPool [Gao and Ji, 2019]♠♣	ND	Graph Classification Node Classification	D&D, PROTEINS, COLLAB Cora, Citeseer, Pubmed	ICML'2019	https://github.com/HongyangGao/Graph-U-Nets
SAGPool [Lee *et al.*, 2019]♠♣	ND	Graph Classification	D&D, PROTEINS, NCI1, NCI109, FRANK.	ICML'2019	https://github.com/inyeoplee77/SAGPool
AttPool [Huang *et al.*, 2019]♠	ND	Graph Classification	D&D, PROTEINS,NCI1, COLLAB, RDT-B,RDT-M12K	ICCV'2019	https://github.com/hjjpku/Attention_in_Graph
EigenPool [Ma *et al.*, 2019]	NC	Graph Classification	D&D, PROTEINS,ENZYMES, NCI1, NCI109, MUTAG	KDD'2019	https://github.com/alge24/eigenpooling
EdgePool [Diehl, 2019]♠	OT	Graph Classification	PROTEINS, COLLAB, RDT-B, RDT-M12K	ICML-W'2019	https://github.com/pyg-team/pytorch_geometric
ASAP [Ranjan *et al.*, 2020]♠	ND	Graph Classification	D&D, PROTEINS, ENZYMES, NCI1, NCI109, MUTAGEN.	AAAI'2020	https://github.com/malllabiisc/ASAP
HGP-SL [Zhang *et al.*, 2020b]	ND	Graph Classification	D&D, PROTEINS, NCI1, NCI109, FRANK.	AAAI'2020	https://github.com/cszhangzhen/HGP-SL
VIPool [Li *et al.*, 2020b]	ND	Graph Classification Node Classification	D&D, PROTEINS, ENZYMES, IMDB-B, IMDB-M, COLLAB Cora, Citeseer, Pubmed	NIPS'2020	https://github.com/limaosen0/GXN
RepPool [Li *et al.*, 2020a]	ND	Graph Classification	D&D, PROTEINS, NCI1, NCI109, MUTAG, PTC, IMDB-B	ICDM'2020	https://github.com/Juanhui28/RepPool
GSAPool [Zhang *et al.*, 2020a]	ND	Graph Classification	D&D, NCI1, NCI109, MUTAGEN.	WWW'2020	https://github.com/psp3dcg/GSAPool
MinCutPool [Bianchi *et al.*, 2020a]♠♣	NC	Graph Classification Node Clustering Graph Regression	D&D, PROTEINS, ENZYMES, IMDB-B, IMDB-M, COLLAB Cora, Citeseer, Pubmed QM9	ICML'2020	https://github.com/FilippoMB
HaarPool [Wang *et al.*, 2020]	NC	Graph Classification Graph Regression	PROTEINS, NCI1, NCI109, MUTAG, MUTAGEN., TRIANGLES QM9	ICML'2020	https://github.com/limaosen0/GXN
MemPool [Khasahmadi *et al.*, 2020]	NC	Graph Classification Graph Regression	D&D, PROTEINS, ENZYMES, COLLAB, RDT-B, ESOL, Lipophilicity	ICLR'2020	https://github.com/amirkhas/GraphMemoryNet
StructPool [Yuan and Ji, 2020]	NC	Graph Classification	PROTEINS, ENZYMES,MUTAG, PTC, IMDB-B, IMDB-M,COLLAB	ICLR'2020	https://github.com/Nate1874/StructPool
PANPool [Ma *et al.*, 2020]♠	ND	Graph Classification	PROTEINS, PROTEINS-FULL, NCI1, MUTAGEN.,AIDS	NIPS'2020	https://github.com/YuGuangWang/PAN
SOPool [Wang and Ji, 2020]	FP	Graph Classification	PROTEINS, NCI1, MUTAG, PTC,IMDB-B, IMDB-M, COLLAB, RDT-B,RDT-M5K	TPAMI'2020	–
CGIPool [Pang *et al.*, 2021]	ND	Graph Classification	PROTEINS, NCI1, NCI109, MUTAGEN., IMDB-B, IMDB-M, COLLAB	SIGIR'2021	https://github.com/PangYunsheng8/CGIPool
GMT [Baek *et al.*, 2021]♠	FP	Graph Classification Graph Reconstruction Graph Generation	D&D, PROTEINS, MUTAG, IMDB-B, IMDB-M, COLLAB, HIV, TOX21, TOXC., BBBP ZINC QM9	ICLR'2021	https://github.com/JinheonBaek/GMT
PAS [Wei *et al.*, 2021]	OT	Graph Classification	D&D, PROTEINS, NCI109, IMDB-B, IMDB-M,COX-2	CIKM'2021	https://github.com/AutoML-Research/PAS
MuchPool [Du *et al.*, 2021]	OT	Graph Classification	D&D, PROTEINS,ENZYMES, NCI1, NCI109, COLLAB	IJCAI'2021	https://github.com/kingloon/MultiChannelPooling
TAPool [Gao *et al.*, 2021a]	ND	Graph Classification	D&D, PROTEINS, MUTAG, PTC	TPAMI'2021	–
IPool [Gao *et al.*, 2021b]	ND	Graph Classification	D&D, PROTEINS, ENZYMES, NCI1, NCI109, MNIST, CIFAR10	TNNLS'2021	–
MVPool [Zhang *et al.*, 2021]	ND	Graph Classification Node Classification Node Clustering Graph Clustering	D&D, PROTEINS,ENZYMES, NCI1, NCI109, MUTAGEN. IMDB-B, RDT-M12K Cora, Citeseer, Pubmed, Coauthor-CS, Coauthor-Phy Cora, Citeseer, Pubmed PROTEINS, NCI109, MUTAGEN.	TKDE'2021	https://github.com/cszhangzhen/MVPool

[1] FP refers to flat pooling; NC refers to node clustering pooling; ND refers to node drop pooling; OT refers to other pooling methods; and "–" indicates that no open-source code is found.
[2] Models with ♠ have another implementation by Pytorch available in Pytorch Geometric: https://pytorch-geometric.readthedocs.io/.
[3] Models with ♣ have another implementation by TensorFlow available in Spektral: https://graphneural.network/.

can be divided into two classes according to the types 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 [Gaudelet *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 need to be 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 [Gao *et al.*, 2019], computer vision [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].

As discussed above, graph pooling methods have achieved a substantial improvement on diverse applications, including structural scenarios and non-structural scenarios, which demonstrates the applicability of graph pooling.

6 Challenges and Opportunities

I: Different Tasks and Multi-tasks

Challenge. As shown in Table 4, most existing graph pooling methods focus on graph-level tasks, while few methods tend to deal with node-level tasks, such as node classification. Actually, graph pooling methods are promising for the reduction of the number of parameters, which helps reduce the time complexity and makes models more resilient to over-fitting, and the large receptive fields, which helps 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 is still a key challenge.

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 opera-

tion 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 as 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) designing a new learning structure to integrate pooling operation more appropriately; and 3) developing a unified graph pooling approach that can simultaneously handle both node-level and graph-level tasks [Holtz *et al.*, 2019; Zeng *et al.*, 2021], where different tasks may benefit from each other.

II: Different Types of Graphs

Challenge. The graph pooling methods introduced in Section 3 are mainly designed to deal with regular 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 other types of graphs, since different types of graphs have different characteristics. Therefore, designing specific graph pooling operators for handling different types of graphs largely remains open in the literature.

Opportunity. Recently, several pooling works have attempted to handle the seldom studied graphs, which are also commonly used in real applications, such as heterogenous 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 missing. Therefore, we suggest two promising research opportunities: 1) extending the existing graph pooling methods or designing new pooling methods to deal with specific graphs by considering 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; and 2) devising a general pooling method, which can handle different types of graphs in a universal manner.

III: Interpretability

Challenge. Although the existing graph pooling methods have achieved excellent results on various graph-based tasks, most of them show little interpretability. However, the interpretability is desirable and critical for pooling methods on graphs, especially in terms of drug or disease related problems, since it enables humans to understand the cause of a decision made by these methods. However, despite some progress made in the interpretation of node representation via GNNs [Yuan *et al.*, 2020], the interpretability of pooling methods remains under explored.

Opportunity. With the aim of explaining exactly what has been learned from graph pooling operations, some studies [Ying *et al.*, 2018; Khasahmadi *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. And, the interpretability of pooling operations is still not well explored. Thus, generalizing the studies on the interpretation of GNNs [Yuan *et al.*, 2020] into graph pooling may be a promising future research direction.

IV: 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; Chen *et al.*, 2021a; Liu *et al.*, 2022], most graph pooling methods fail to distinguish the noisy information from the input graph, thus significantly degrading their performance when the input graph is perturbed in terms of features or topology.

Opportunity. Despite some initial studies on the robustness of graph machine learning, few studies have been conducted on 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 before we develop a robust graph pooling method for actual usages, such as: 1) building up a more complete adversarial attack framework which contains different types of attacks; and 2) designing adversarial defense graph pooling methods. One possible solution is to generalize the techniques proposed on the robustness of graph machine learning into graph pooling.

V: 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 evaluate the effectiveness of graph pooling in reducing time and space complexities. Another important issue is the efficiency of pooling methods since the high cost of time or space hinders their applicability to large-scale datasets. For example, node clustering pooling methods suffer from high time and storage complexities caused by the computation of the dense assignment matrix.

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; and 2) designing more efficient graph pooling methods and making them more practical in real-world scenarios.

VI: Expressive Power

Challenge. Most existing graph pooling methods are designed by intuition, and their performance gains are evaluated by empirical experiments. The lack of characterizing the expressive power of graph pooling operators hinders the design of more powerful graph pooling models.

Opportunity. Only a few works [Murphy *et al.*, 2019; Chen *et al.*, 2020b; Baek *et al.*, 2021] tend to analyze the expressive power of their models in terms of the 1-Weisfeiler-Lehman (WL) test. Therefore, exploring how the expressive power of graph pooling methods can be matched with the 1-WL test, and the creation of more powerful graph pooling methods are promising and important future directions.

References

[Adnan *et al.*, 2020] Mohammed Adnan, Shivam Kalra, and Hamid R Tizhoosh. Representation learning of histopathology images using graph neural networks. In *CVPR Workshop*, 2020.

[Atwood and Towsley, 2016] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *NeurIPS*, 2016.

[Bacciu and Sotto, 2019] Davide Bacciu and Luigi Di Sotto. A non-negative factorization approach to node pooling in graph convolutional neural networks. In *AIIA*, 2019.

[Baek *et al.*, 2021] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *ICLR*, 2021.

[Bai *et al.*, 2019] Yunsheng Bai, et al. Unsupervised inductive graph-level representation learning via graph-graph proximity. In *IJCAI*, 2019.

[Bai *et al.*, 2021] Lu Bai, et al. Learning graph convolutional networks based on quantum vertex information propagation. *IEEE TKDE*, 2021.

[Bianchi *et al.*, 2020a] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *ICML*, 2020.

[Bianchi *et al.*, 2020b] Filippo Maria Bianchi, et al. Hierarchical representation learning in graph neural networks with node decimation pooling. *IEEE TNNLS*, 2020.

[Bodnar *et al.*, 2020] Cristian Bodnar, Cătălina Cangea, and Pietro Liò. Deep graph mapper: Seeing graphs through the neural lens. In *NeurIPS Workshop*, 2020.

[Cangea *et al.*, 2018] Cătălina Cangea, et al. Towards sparse hierarchical graph classifiers. *arXiv:1811.01287*, 2018.

[Chang *et al.*, 2021] Jianxin Chang, et al. Sequential recommendation with graph neural networks. In *SIGIR*, 2021.

[Chen *et al.*, 2019] Ting Chen, Song Bian, and Yizhou Sun. Are powerful graph neural nets necessary? a dissection on graph classification. *arXiv:1905.04579*, 2019.

[Chen *et al.*, 2020a] Chaofan Chen, et al. Hapgn: Hierarchical attentive pooling graph network for point cloud segmentation. *IEEE TMM*, 2020.

[Chen *et al.*, 2020b] Zhengdao Chen, et al. Can graph neural networks count substructures? In *NeurIPS*, 2020.

[Chen *et al.*, 2021a] Jinyin Chen, et al. Egc2: Enhanced graph classification with easy graph compression. *arXiv:2107.07737*, 2021.

[Chen *et al.*, 2021b] Kaixuan Chen, et al. Distribution knowledge embedding for graph pooling. *arXiv:2109.14333*, 2021.

[Demir *et al.*, 2021] Andac Demir, et al. Eeg-gnn: Graph neural networks for classification of electroencephalogram (eeg) signals. *arXiv:2106.09135*, 2021.

[Diehl, 2019] Frederik Diehl. Edge contraction pooling for graph neural networks. *arXiv:1905.10990*, 2019.

[Du *et al.*, 2021] Jinlong Du, et al. Multi-channel pooling graph neural networks. In *IJCAI*, 2021.

[Duvenaud *et al.*, 2015] David Duvenaud, et al. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015.

[Dwivedi *et al.*, 2020] Vijay Prakash Dwivedi, et al. Benchmarking graph neural networks. *arXiv:2003.00982*, 2020.

[Errica *et al.*, 2020] Federico Errica, et al. A fair comparison of graph neural networks for graph classification. In *ICLR*, 2020.

[Fan *et al.*, 2020] Xiaolong Fan, et al. Structured self-attention architecture for graph-level representation learning. *Pattern Recognition*, 2020.

[Fey *et al.*, 2018] Matthias Fey, et al. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *CVPR*, 2018.

[Gao and Ji, 2019] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *ICML*, 2019.

[Gao *et al.*, 2019] Hongyang Gao, Yongjun Chen, and Shuiwang Ji. Learning graph pooling and hybrid convolutional operations for text representations. In *WWW*, 2019.

[Gao *et al.*, 2020] Zhangyang Gao, et al. Lookhops: light multi-order convolution and pooling for graph classification. *arXiv:2012.15741*, 2020.

[Gao *et al.*, 2021a] H. Gao, Y. Liu, and S. Ji. Topology-aware graph pooling networks. *IEEE TPAMI*, 2021.

[Gao *et al.*, 2021b] Xing Gao, et al. ipool–information-based pooling in hierarchical graph neural networks. *IEEE TNNLS*, 2021.

[Gaudelet *et al.*, 2021] Thomas Gaudelet, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 2021.

[Gopinath *et al.*, 2022] Karthik Gopinath, Christian Desrosiers, and Herve Lombaert. Learnable pooling in graph convolutional networks for brain surface analysis. *IEEE TPAMI*, 2022.

[Grattarola *et al.*, 2021] Daniele Grattarola, et al. Understanding pooling in graph neural networks. *arXiv:2110.05292*, 2021.

[Gu, 2021] Jindong Gu. Interpretable graph capsule networks for object recognition. *AAAI*, 2021.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, et al., editors, *NeurIPS*, 2017.

[Hermosilla *et al.*, 2021] Pedro Hermosilla, et al. Intrinsic-extrinsic convolution and pooling for learning on 3d protein structures. In *ICLR*, 2021.

[Holtz *et al.*, 2019] Chester Holtz, et al. Multi-task learning on graphs with node and graph level labels. In *NeurIPS Workshop*, 2019.

[Hu *et al.*, 2020] Weihua Hu, et al. Open graph benchmark: Datasets for machine learning on graphs. *arXiv:2005.00687*, 2020.

[Huang *et al.*, 2019] Jingjia Huang, et al. Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism. In *ICCV*, 2019.

[Isufi and Mazzola, 2021] Elvin Isufi and Gabriele Maz-

zola. Graph-time convolutional neural networks. *arXiv:2103.01730*, 2021.

[Itoh *et al.*, 2022] Takeshi D. Itoh, Takatomi Kubo, and Kazushi Ikeda. Multi-level attention pooling for graph neural networks: Unifying graph representations with multiple localities. *Neural Networks*, 2022.

[Jo *et al.*, 2021] Jaehyeong Jo, et al. Edge representation learning with hypergraphs. In *NeurIPS*, 2021.

[Khasahmadi *et al.*, 2020] Amir Hosein Khasahmadi, et al. Memory-based graph networks. In *ICLR*, 2020.

[Kim *et al.*, 2021] Byung-Hoon Kim, Jong Chul Ye, and Jae-Jin Kim. Learning dynamic graph representation of brain connectome with spatio-temporal attention. In *NeurIPS*, 2021.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Knyazev *et al.*, 2019] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. In *NeurIPS*, 2019.

[Lee *et al.*, 2019] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, 2019.

[Lee *et al.*, 2021] Dongha Lee, et al. Learnable structural semantic readout for graph classification. In *ICDM*, 2021.

[Li *et al.*, 2016] Yujia Li, et al. Gated graph sequence neural networks. In *ICLR*, 2016.

[Li *et al.*, 2020a] Juanhui Li, et al. Graph pooling with representativeness. In *ICDM*, 2020.

[Li *et al.*, 2020b] Maosen Li, et al. Graph cross networks with vertex infomax pooling. In *NeurIPS*, 2020.

[Li *et al.*, 2020c] Xiaoxiao Li, et al. Pooling regularized graph neural network for fmri biomarker analysis. In *MICCAI*, 2020.

[Li *et al.*, 2021a] Jia Li, et al. Deconvolutional networks on graph data. *NeurIPS*, 2021.

[Li *et al.*, 2021b] Shuangli Li, et al. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In *KDD*, 2021.

[Li *et al.*, 2021c] Xiaoxiao Li, et al. Braingnn: Interpretable brain graph neural network for fmri analysis. *Medical Image Analysis*, 2021.

[Liang *et al.*, 2020] Yanyan Liang, et al. Mxpool: Multiplex pooling for hierarchical graph representation learning. *arXiv:2004.06846*, 2020.

[Liu *et al.*, 2020] Fanzhen Liu, et al. Deep learning for community detection: progress, challenges and opportunities. In *IJCAI*, 2020.

[Liu *et al.*, 2021a] Ning Liu, et al. Hierarchical adaptive pooling by capturing high-order dependency for graph representation learning. *IEEE TKDE*, 2021.

[Liu *et al.*, 2021b] Y. Liu, et al. Learning hierarchical review graph representations for recommendation. *IEEE TKDE*, 2021.

[Liu *et al.*, 2022] Chuang Liu, et al. On exploring node-feature and graph-structure diversities for node drop graph pooling, 2022.

[Ma *et al.*, 2019] Yao Ma, et al. Graph convolutional networks with eigenpooling. In *KDD*, 2019.

[Ma *et al.*, 2020] Zheng Ma, et al. Path integral based convolution and pooling for graph neural networks. In *NeurIPS*, 2020.

[Ma *et al.*, 2021] Xiaoxiao Ma, et al. A comprehensive survey on graph anomaly detection with deep learning. *IEEE TKDE*, 2021.

[Mai *et al.*, 2020] Sijie Mai, et al. Analyzing unaligned multimodal sequence via graph convolution and graph pooling fusion. *arXiv:2011.13572*, 2020.

[Mesquita *et al.*, 2020] Diego Mesquita, Amauri Souza, and Samuel Kaski. Rethinking pooling in graph neural networks. In *NeurIPS*, 2020.

[Morris *et al.*, 2020] Christopher Morris, et al. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv:2007.08663*, 2020.

[Murphy *et al.*, 2019] Ryan Murphy, et al. Relational pooling for graph representations. In *ICML*, 2019.

[Nadgeri *et al.*, 2021] Abhishek Nadgeri, et al. KGPool: Dynamic knowledge graph context selection for relation extraction. In *ACL*, 2021.

[Navarin *et al.*, 2019] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Universal readout for graph convolutional neural networks. In *IJCNN*, 2019.

[Noutahi *et al.*, 2019] Emmanuel Noutahi, et al. Towards interpretable sparse graph representation learning with laplacian pooling. *arXiv:1905.11577*, 2019.

[Pan *et al.*, 2021] Li Pan, et al. Identifying autism spectrum disorder based on individual-aware down-sampling and multi-modal learning. *arXiv:2109.09129*, 2021.

[Pang *et al.*, 2021] Yunsheng Pang, Yunxiang Zhao, and Dongsheng Li. Graph pooling via coarsened graph infomax. In *SIGIR*, 2021.

[Papp *et al.*, 2021] Pál András Papp, et al. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *NeurIPS*, 2021.

[Qin *et al.*, 2020] Jian Qin, et al. Uniform pooling for graph networks. *Appl. Sci.*, 2020.

[Ranjan *et al.*, 2020] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *AAAI*, 2020.

[Rhee *et al.*, 2018] Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. In *IJCAI*, 2018.

[Roy *et al.*, 2021] Kashob Kumar Roy, et al. Structure-aware hierarchical graph pooling using information bottleneck. In *IJCNN*, 2021.

[Shen *et al.*, 2018] Y. Shen, et al. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 2018.

[Simonovsky and Komodakis, 2017] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017.

[Su *et al.*, 2021] Zidong Su, Zehui Hu, and Yangding Li. Hierarchical graph representation learning with local capsule pooling. In *MMAsia*, 2021.

[Tak *et al.*, 2021] Hemlata Tak, et al. End-to-end spectro-temporal graph attention networks for speaker veri-

fication anti-spoofing and speech deepfake detection. *arXiv:2107.12710*, 2021.

[Tang *et al.*, 2020] Haoteng Tang, et al. Adversarial attack on hierarchical graph pooling neural networks. *arXiv:2005.11560*, 2020.

[Tang *et al.*, 2021] Haoteng Tang, et al. Commpool: An interpretable graph pooling framework for hierarchical graph representation learning. *Neural Networks*, 2021.

[Vinyals *et al.*, 2016] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.

[Wang and Ji, 2020] Zhengyang Wang and Shuiwang Ji. Second-order pooling for graph neural networks. *IEEE TPAMI*, 2020.

[Wang *et al.*, 2020] Yu Guang Wang, et al. Haar graph pooling. In *ICML*, 2020.

[Wang *et al.*, 2021] Jie Wang, et al. Papooling: Graph-based position adaptive aggregation of local geometry in point clouds. *arXiv:2111.14067*, 2021.

[Wei *et al.*, 2021] Lanning Wei, et al. Pooling architecture search for graph classification. In *CIKM*, 2021.

[Wu *et al.*, 2019] Jun Wu, Jingrui He, and Jiejun Xu. Demonet: Degree-specific graph neural networks for node and graph classification. In *KDD*, 2019.

[Wu *et al.*, 2021] Chuhan Wu, et al. User-as-graph: User modeling with heterogeneous graph pooling for news recommendation. In *IJCAI*, 2021.

[Xu *et al.*, 2019] Keyulu Xu, et al. How powerful are graph neural networks? In *ICLR*, 2019.

[Yang *et al.*, 2021] Jinyu Yang, et al. Hierarchical graph capsule network. In *AAAI*, 2021.

[Ying *et al.*, 2018] Zhitao Ying, et al. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

[Yuan and Ji, 2020] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *ICLR*, 2020.

[Yuan *et al.*, 2020] Hao Yuan, et al. Explainability in graph neural networks: A taxonomic survey. *arXiv:2012.15445*, 2020.

[Zeng *et al.*, 2021] Hanqing Zeng, et al. Decoupling the depth and scope of graph neural networks. In *NeurIPS*, 2021.

[Zhang *et al.*, 2018] Muhan Zhang, et al. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018.

[Zhang *et al.*, 2020a] Liang Zhang, et al. Structure-feature based graph self-adaptive pooling. In *WWW*, 2020.

[Zhang *et al.*, 2020b] Zhen Zhang, et al. Hierarchical graph pooling with structure learning. In *AAAI*, 2020.

[Zhang *et al.*, 2021] Zhen Zhang, et al. Hierarchical multi-view graph pooling with structure learning. *IEEE TKDE*, 2021.

[Zhou *et al.*, 2020] Kaixiong Zhou, et al. Multi-channel graph neural networks. In *IJCAI*, 2020.

[Zhou *et al.*, 2021] Xiaowei Zhou, Jie Yin, and Ivor W Tsang. Edge but not least: Cross-view graph pooling. *arXiv:2109.11796*, 2021.