CommPOOL: An Interpretable Graph Pooling Framework for Hierarchical Graph Representation Learning

Haoteng Tang[a,1], Guixiang Ma[b,2], Lifang He[c,3], Heng Huang[a,1], Liang Zhan[a,1,*]

[a] *3700 O'Hara St, Pittsburgh, PA, USA, 15260*
[b] *2111 NE 25th Ave, Hillsboro, OR, USA, 97124*
[c] *113 Research Dr, Bethlehem, PA, USA, 18015*

Abstract

Recent years have witnessed the emergence and flourishing of hierarchical graph pooling neural networks (HGPNNs) which are effective graph representation learning approaches for graph level tasks such as graph classification. However, current HGPNNs do not take full advantage of the graph's intrinsic structures (e.g., community structure). Moreover, the pooling operations in existing HGPNNs are difficult to be interpreted. In this paper, we propose a new interpretable graph pooling framework — CommPOOL, that can capture and preserve the hierarchical community structure of graphs in the graph representation learning process. Specifically, the proposed community pooling mechanism in CommPOOL utilizes an unsupervised approach for capturing the inherent community structure of graphs in an interpretable manner. CommPOOL is a general and flexible framework for hierarchical graph representation learning that can further facilitate various graph-level tasks. Evaluations on five public benchmark datasets and one synthetic dataset demonstrate the superior performance of CommPOOL in graph representation learning for graph classification compared to the state-of-the-art baseline methods, and its effectiveness in cap-

*Corresponding author
[1]Department of Electrical and Computer Engineering,University of Pittsburgh
[2]Intel Labs
[3]Department of Computer Science and Engineering, Lehigh University

turing and preserving the community structure of graphs.

Keywords: Graph Representation Learning, Hierarchical Graph Pooling Neural Network, Community Structure, Graph Classification

1. Introduction

In recent years, graph neural network (GNN) has emerged and been broadly used as a generalized deep learning architecture for graph representation learning in many fields, such as social network analysis [1, 2] and chemical molecule studies [3, 4, 5]. Generally, GNN models learn node embeddings by passing, transforming and aggregating node features across the graph. The generated node representations can then be forwarded to further layers for specific learning tasks, i.e., node classification [6, 7] and link prediction [8].

Most of the existing GNN models (e.g., GCN [6], GAT [7], GraphSage [9]) focus on node-level representation learning and only propagate information across edges of the graph in a flat way. When applying these GNNs for graph-level tasks such as graph classifications, existing works usually apply simple global pooling strategies (i.e., a summation over the learned node representations) to obtain the graph-level embedding and use it for graph label prediction [10, 11, 12]. One main drawback in these GNNs is that the hierarchical structure, often existing in graphs, is ignored during the global pooling process, which makes the models less effective for graph-level tasks. Hierarchical structure is a very important structure for many graphs in various domains. For example, the hierarchical community structure shown in **Figure 1** is a typical pattern that often appears in social networks [13, 14], chemical molecule networks [15] and brain networks [16, 17]. Therefore, preserving these community structures is critical for better understanding and analyzing these graphs.

Some recent works proposed hierarchical graph pooling neural networks (HGPNNs) to address the hierarchical structure representation issue by introducing the hierarchical pooling operations [18, 19, 20, 21]. Generally, these HGPNNs consist of two components: the GNN backbone which is used to embed

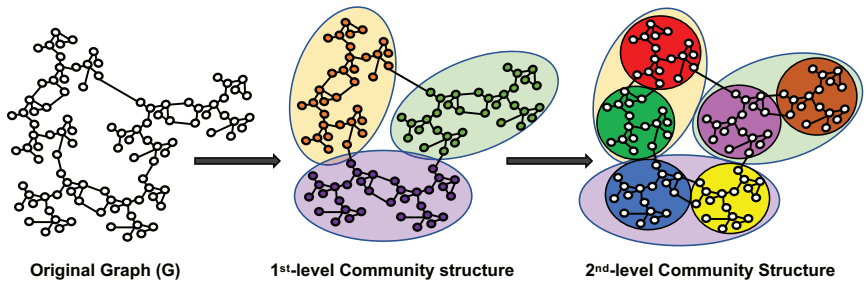


Figure 1: An example of hierarchical community structures in graphs

the graph nodes and local structures, and the pooling operation which represents graph structure in a hierarchical way. These HGPNNs have demonstrated the necessity of adding hierarchical pooling operations in GNNs to better preserve the graph hierarchical structure.

However, a critical limitation of the existing hierarchical graph pooling (HGP) strategies is that few of the pooling operations in the models are interpretable. In many real applications, it is desirable to have an interpretable model, where human can understand the cause of a decision made by the model [22, 23]. Moreover, an interpretable model is more robust under adversarial attacks [24, 25, 26, 27]. A few of recent works [28, 29, 30] interpreted the node feature embedding via GNN as a neighborhood aggregation scheme. Particularly, they stated that the GNN embed the local feature of each node v within two steps: (1) neighbor node features aggregation and (2) node feature transformation. However, the interpretability of pooling operations is still not well solved (Details are discussed in the **Related Work**.). In order to make the HGP operation interpretable, three questions should be considered:

Q1: How to capture the graph hierarchical structures in an interpretable way?

Q2: How to scale down the graph representation while preserving the structures via an interpretable process?

Q3: What do we obtain after the pooling operation?

To address these challenges, we propose a Community-Based HGP framework, **COMMUNITY-POOL** or **CommPOOL**. We aim to encode the hierarchical community structure in graphs, which is a natural structure in many

graphs, where nodes within each community are more densely connected than the nodes across different communities. Specifically, we propose a community-based hierarchical pooling operation which aggregates and synthesizes the node features based on the detected communities, such that the community structure of graphs can be preserved during the pooling process. Moreover, we introduce a GNN-based framework with the proposed community-based hierarchical pooling operation for learning latent graph representations, where both local node features and the hierarchical community-structure information are encoded and preserved. Our contributions can be summarized as:

- We propose a community-based HGP framework (CommPOOL) for learning graph representation in a hierarchical way that can preserve both the local node features and the hierarchical community structure of graphs.
- The proposed hierarchical community pooling strategy relies on the community structure which is explicitly detected from the graphs, therefore the pooling operation can capture the intrinsic community-level latent representation of graphs and the pooling process is inherently interpretable.
- We evaluate our CommPOOL framework for the whole graph classification task on multiple public benchmark datasets. The results demonstrate the superior performance of our model compared to several state-of-the-art graph pooling neural networks.
- Evaluations on synthetic graphs with community ground-truth labels show that our proposed CommPOOL can capture and preserve the intrinsic community structure of graphs during the learning process.

2. Related Work

2.1. Graph Pooling

Graph pooling operation is a strategy aiming to scale down the size of input graphs. It can not only help to avoid model overfitting and reduce the

computational cost but also generate graph-level representations [31]. In the early works [32, 33, 34, 35, 11], the graph pooling methods simply compute the $mean/max/sum$ of all graph node features as the representation of the whole graph. Such a primitive pooling strategy is named as global pooling. Later on, a few advanced techniques (e.g. attention mechanisms [10, 5, 36], feature sorted [12]) are proposed to improve the performance and efficiency of the global pooling. However, the global pooling methods do not learn the hierarchical representations, which are crucial for capturing the structural information of graphs. Therefore, HGPNNs are proposed [21, 18, 37, 20, 38, 39, 40].

2.2. Interpretability of HGPNNs

Most HGP operations in the current HGPNNs [21, 18, 37, 20, 38, 39, 40] show little interpretability and are difficult to be understood by the users. Moreover, very few studies present the interpretabiliy of their HGP operations in the paper, which may be accounted for by the following two issues: (1). Hardly any clear definition or analysis can be found to explain what is the captured graph structure. Therefore, the model users are lack of heuristic knowledge to understand the pooling operation. (2). Although some studies [18] present the visualization of hierarchical clusters captured by the model, no quantitative analysis is provided to examine whether the captured clusters of nodes are aligned with the intrinsic clusters in the original graph. Apart from these, most recent studies unfold the HGP operation as a neural network layer with trainable parameters. The black-box nature of neural networks may also raise extra difficulties to interpret the models in a way.

3. Preliminaries

3.1. Graph Notation

We consider the graph classification problem on attributed graphs with different numbers of nodes. Let $G = (A, H)$ be any of the attributed graph with N nodes, where $A \in \{0, 1\}^{N \times N}$ is the graph adjacency matrix and $H \in \mathcal{R}^{N \times d}$

is the node feature matrix assuming that each node has d features. Also, $Z = [Z_1, ..., Z_N]^T$ is defined as the node latent feature matrix where Z_i is the latent feature vector for the node i. Given a set of labeled data $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), (G_3, y_3), ...\}$ where $y_i \in \mathcal{Y}$ is the classification label to the corresponding graph $G_i \in \mathcal{G}$. The graph classification task can be formulated as learning a mapping, f: $\mathcal{G} \rightarrow \mathcal{Y}$.

3.2. Graph Neural Network

Graph Neural Network (GNN) is an effective message-passing architecture for embedding the graph nodes and their local structures. Generally, GNN can be formulated as:

$$Z^{(k)} = F(A^{(k-1)}, Z^{(k-1)}; \theta^{(k)}), \tag{1}$$

where k denotes the layer k of GNN. $A^{(k-1)}$ is the graph adjacency matrix computed by layer $(k-1)$ of the GNN. $\theta^{(k)}$ is the trainable parameters in the layer k of the GNN. Particularly, $Z^0 = H$.

$F(\cdot)$ is the forward function to combine and transform the messages across the nodes. Many different versions of forward functions $F(\cdot)$ are proposed in the previous studies [5, 9] such as Graph Convolutional Neural Network (GCN) [6] and Graph Attention Network (GAT) [7]. The GCN linearly combines the neighborhoods as the node the representation. And the GAT computes node representations in entire neighborhoods based on attention mechanisms [41].

4. The Proposed Framework

4.1. Model Architecture

Our goal is to provide a general graph pooling framework that can capture and preserve the hierarchical community structure of graphs in the representation learning process of GNNs. The framework should be interpretable and it should be able to facilitate further graph-level learning tasks, for example, graph classification. To achieve this goal, we propose a community-based hierarchical

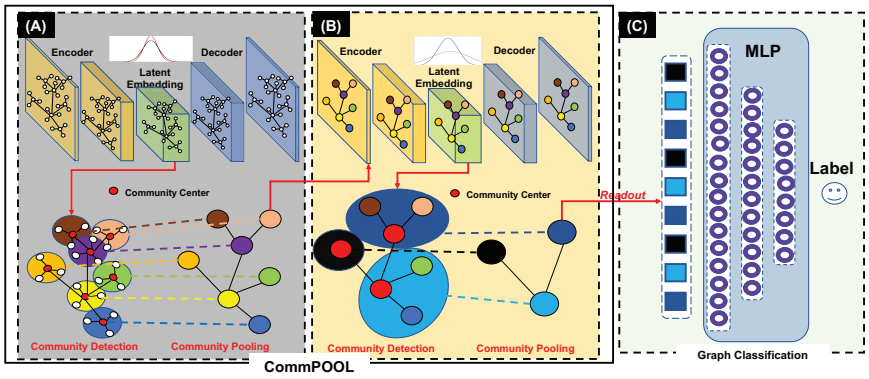


Figure 2: Framework of the CommPOOL for graph classification. **(A)** is the 1^{st} Embedding-Pooling (EP) module and **(B)** is the 2^{nd} EP module. In each module, we embed the graph into the latent space by using VGAE. In the latent space, we scale down the graph representation based on the detected communities. **(C)** is the MLP for graph classification.

graph pooling (HGP) framework: **CommPOOL**, which is composed of k cascaded Embedding-Pooling (EP) modules to learn the graph representation in a hierarchical way. Each EP module consists of (1) an Embedding stage, where a GNN model is employed to get the latent node representations (i.e., node embeddings) of the input graph, and (2) a Pooling stage, where a newly proposed community pooling mechanism is conducted on the node embeddings to detect communities from the graph and obtain a scaled-down graph-level representation that encodes both the local node features and the community structure of the graph. The output of the last EP module will be the final graph-level representation that preserves the overall hierarchical community structure of the graph. **Figure 2(A, B)** shows an instance of the proposed framework with two cascaded EP modules. In real applications of our framework, the choice of value for k is flexible and it can be decided based on the practical needs or domain knowledge for the specific application (e.g., domain evidence about how many community hierarchies exist in the graphs). In this paper, we set $k = 2$ and use the architecture given in **Figure 2(A, B)** for illustrating our framework and we use the MLP shown in **Figure 2(C)** for evaluating the CommPOOL in graph representation learning for facilitating graph classification task.

In the following subsections, we introduce the two main parts in the proposed EP module for CommPOOL: (1) the GNN-based Graph Node **Embedding**, and (2) the Community **Pooling** Operation.

4.2. GNN-based Node Embedding

We aim at a general GNN-based model to embed the graph nodes into the latent feature space $\mathcal{Z}$ that well preserves the inherent graph structures. On the one hand, the desired node latent features should well encode the node information and the information between the node and its neighbors. On the other hand, the latent features should preserve the intrinsic structures of the graphs without task-specific influences or supervised information. On account of the above considerations, we choose the Variational Graph Auto-Encoders (VGAE) [8] for this node embedding stage. VGAE is a robust unsupervised method that can embed nodes into the latent space by reconstructing the graph itself. This can help preserve the intrinsic structures of the graph without involving any task-specific information or supervision.

4.2.1. Encoder

In the VGAE, we need to learn a Gaussian distribution $q(Z|H, A) = \mathcal{N}(Z|\mu, \sigma^2)$ which is used to approximate the Gaussian prior $p(Z) = \mathcal{N}(Z|0, I)$. Particularly, we utilize two GNN layers to compute the μ and σ^2 parameters of q. In the first layer, μ and σ^2 share the same GNN encoder. And in the second layer, two separate GNNs are used to generate μ and σ^2 respectively. The approximation can be achieved by maximizing the Kullback–Leibler (KL) loss between p and q:

$$\mathcal{L}_{KL} = KL(q(Z|H, A)||p(Z)) \tag{2}$$

The latent features Z can be obtained by resampling from the optimal $q(Z|H, A)$.

4.2.2. Decoder

After we obtain the latent features Z, we reconstruct the original graph adjacency matrix by:

$$\hat{A} = sigmoid(ZZ^T). \tag{3}$$

We define $+$ and $-$ as the edges and non-edges position index in A. So we reconstruct the adjacency matrix by minimizing the $\mathcal{L}_A$:

$$\begin{aligned} \mathcal{L}_A &= \mathcal{L}^+ + \mathcal{L}^- \\ &= -\frac{1}{E_1}\Sigma(log(\hat{A}^+)) - \frac{1}{E_2}\Sigma(log(1-\hat{A}^-)) \end{aligned} \tag{4}$$

where E_1, E_2 is the number of edges and non-edges. The overall objective function of VGAE is:

$$\underset{Z \in \mathcal{Z}}{\text{minimize}} \quad \mathcal{L}_A - \mathcal{L}_{KL} \tag{5}$$

In our CommPOOL, we use GCN [6] to build up the basic encoder layers and use GAT [7] and Higher-Order Graph Neural Network (HO-GNN) [42] as the encoder variations.

4.3. Community Pooling

After embedding the graph nodes into the latent space $\mathcal{Z}$, the latent features represent the node attributes and topology structures among nodes in the latent space. The distance between two latent features measures the node attributes similarity and structure similarity between the two nodes. The proposed CommPOOL assigns the similar nodes into a community which is further used to scale down the graph.

4.3.1. Community Capturing

We adopt an unsupervised clustering method Partitioning Around Medoids (PAM) [43, 19] on the node latent feature vectors to group the graph nodes into L different communities, where L is a parameter denoting the number of communities in the graph. Our community partition problem can be defined as: given

all the N nodes in graph G with their latent feature vectors set $V = \{Z_1, ..., Z_N\}$, find L different nodes with their latent features $Z_C = \{Z_{C_1}, ..., Z_{C_L}\} \subset V$ from the N nodes as the optimal community centers, and assign the other nodes into these L communities based on the distances between their latent feature vectors ($O = V \setminus Z_C$) and Z_C. PAM realizes the community partition problem via the following four steps.

- **Step 1.** Initialization: Randomly select L nodes with their features Z_C as the community medoid nodes.
- **Step 2.** Clustering: Compute the L_1 distances between the medoid nodes and the rest nodes based on their feature vectors, and assign each non-medoid node to its closest community; Calculate the value for the below cost function, which computes the total distance between the non-medoid node feature vectors $O_j \in O$ and their community medoid feature vectors by:

$$Cost = \Sigma_{j=1}^{N-L} |O_j - Z_{C_x}|_{L_1}, \tag{6}$$

  where $Z_{C_x} \in Z_C$ is the corresponding medoid of O_j.
- **Step 3.** Adjusting: Swap each medoid node by all other non-medoids and calculate the total cost for current configuration referring to Step 2; Compare the cost of current and previous configuration and keep the configuration with the smaller total cost.
- **Step 4.** Optimization: Repeat Step 2 and 3 until the configuration does not change.

4.3.2. Community Pooling

In order to preserve the captured community structure during the pooling process for the entire-graph representation learning, we propose a new pooling mechanism called "community pooling", which summarizes the learned node representations based on the detected community structure. Suppose

$Z_{M_i} = \{Z^1_{M_i}, ..., Z^W_{M_i}\}$ is the set consisting of the latent feature vectors of all W community member nodes except for the community center nodes $Z_{C_i} \in Z_C$ in the community$-i$. Our community pooling problem can be defined as: given a community center feature $Z_{C_i} \in Z_C$, and the corresponding W community member features Z_{M_i}, compute the community representation Z_{Comm_i}. The community pooling operation computes the community$-i$'s representation by:

$$Z_{Comm_i} = Z_{C_i} + \Sigma^W_{w=1}\ Sim(Z^w_{M_i}, Z_{C_i}) Z^w_{M_i}, \tag{7}$$

where $Sim(\cdot)$ is a function to measure the normalized similarity ($\in$(0,1]) between each member $Z^w_{M_i}$ and the community center Z_{C_i}. In our model, we mainly define $Sim(\cdot)$ based on L_1 distance:

$$Sim(Z^w_{M_i}, Z_{C_i}) = \frac{1}{\|Z^w_{M_i} - Z_{C_i}\|_{L1}} \tag{8}$$

When each community representation Z_{Comm_i} is computed, we replace the center node feature Z_{Ci} by Z_{Comm_i} and remove other community member nodes. As for the graph topology structure, the preserved center nodes are connected if and only if they are connected in the original graph. To sum up, during the pooling, the community structure information and the node features are preserved onto the community center nodes. And the graph structures among the communities are presented as the topology structure of down-scaled graph with $M < N$ nodes.

4.4. CommPOOL for Graph Classification

When the community representations $Z^{(K)}_{Comm} = [Z^{(K)}_{Comm_1}, ..., Z^{(K)}_{Comm_L}]^T$ are obtained from the last Embedding-Pooling module ($k = K$), a global readout operation is used to generate the whole graph representation Z_{graph} by averaging $Z^{(K)}_{Comm}$. Finally, an Multilayer Perceptron (MLP) utilizes Z_{graph} to make predictions for graph classification. The training procedure of CommPOOL for the graph classification task is summarized in **Algorithm 1**.

Algorithm 1: Training Procedure

Input : graph: $G = (A, H)$, classification label: y, K

Output: prediction: $\hat{y}$

for $k = 1, 2, ..., K$ **do**

Step 1: Use G to train the VGAE

Step 2: Obtain the latent feature using trained VGAE

Step 3: Community Pooling on latent features and generate down-scaled graph $G^{(k)} = (A^{(k)}, Z^{(k)}_{Comm})$. Set $G = G^{(k)}$.

end

Step 4: $Z_{graph} = GlobalReadout(Z^{(K)}_{Comm})$

Step 5: Train MLP to generate $\hat{y} = MLP(Z_{graph})$

5. Experiment

In this section, we evaluate our CommPOOL framework using graph classification tasks. We present our experiment results in the following four subsections: (1) We introduce the dataset used in the experiments. (2) We compare the graph classification performance between CommPOOL and several competing HGPNN models. (3) We provide some variations of the CommPOOL. (4) We test our model on the simulation data to evaluate whether CommPOOL can accurately preserve the community structures in the graph.

5.1. Dataset

Five graph dataset are selected from the public benchmark graph data collection [44]. **Table 1** summarizes the statistics of all dataset.

PROTEINS and **Synthie** [45, 46, 47] are two sets of graphs representing the protein structure. The nodes are some amino acid features such as secondary structure content and amino acid propensities. Nodes are linked by edges if the amino acid is an amino acid sequence. **FRANKENSTEIN** [48] is a set of graphs representing the molecules with or without mutagenicity. The nodes represent different chemical atoms and the edges are the chemical bonds type. **BZR** [49] is a set of graphs representing the ligands for the benzodiazepine receptor. And **AIDS** [50] is set of graphs representing molecular compounds

with activity against HIV or not. The molecules are converted into graphs by representing chemical atoms as nodes and the bonds as edges.

Table 1: Dataset Statistics: $\mathcal{V}$ and the $\mathcal{E}$ represent the nodes and edges in graph G. c represents graph classes.

Dataset	$\#\|G\|$	Ave.$\|\mathcal{V}\|$	Ave.$\|\mathcal{E}\|$	$\#\|c\|$
BZR	405	35.75	38.36	2
Synthie	400	95.00	172.93	4
FRANKENSTEIN	4337	16.90	17.88	2
PROTEINS	1113	39.06	72.82	2
AIDS	2000	15.69	16.20	2

5.2. Graph Classification

5.2.1. Baseline Methods

Our baseline methods include: two graph global pooling models (**Set2Set** [11] and **SortPool** [12]), and three HGP models (**DIFFPOOL** [18], **SAGPOOL** [19] and **HGP-SL** [21]). For fair comparisons, we set two embedding-pooling modules for all HGP models including three baseline HGPs and our CommPOOL. For the baselines, we follow the original hyperparameter search strategies provided in the related papers.

5.2.2. Experiments Setting

Following previous works [51, 18, 21], we randomly split the whole dataset into training (80%) set, validation (10%) set and testing (10%) set. We repeat this randomly splitting process 10 times, and the average test performance with standard derivation is reported in **Table 2**. We optimize the model via Pytorch Adam optimizer. For the VGAE in the first module, the learning rate (lr) and the weight decay (wd) are searched in $\{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1\}$. The dimension of two latent GNN layers are 32 and 16. For the VGAE in the second module, the lr and wd are searched in $\{0.0001, 0.001, 0.005, 0.01\}$ and the dimension of two latent GNN layers are 64 and 32. In the community pooling operation, the number of communities (L) is searched in $\{40\%, 50\%, 60\%\}$ of the

number of graph nodes (N). The MLP consists of two fully connected layers with 64 and 32 neurons and a $softmax$ output layer. The lr for training the MLP is searched in $\{0.001, 0.005, 0.01\}$. We stop training if the validation loss does not decrease for 50 epochs. Our experiments are deployed on NVIDIA Tesla P100 GPUs supported by the Bridges system of Pittsburgh Supercomputing Center (PSC) [52, 53]. We implement all baselines and CommPOOL using PyTorch [54] and the torch geometric library [55].

Table 2: Average graph classification test accuracy ± standard deviation (%).

Models	BZR	Synthie	FRANKENSTEIN	PROTEINS	AIDS
Set2Set	80.50 ± 1.03	22.50 ± 0.86	60.62 ± 0.27	68.08 ± 0.56	88.80 ± 0.45
SortPool	77.00 ± 1.24	32.50 ± 1.24	59.86 ± 1.22	70.11 ± 0.04	86.00 ± 2.42
DIFFPOOL	80.50 ± 1.48	57.00 ± 2.62	60.60 ± 1.62	72.43 ± 0.26	93.50 ± 1.00
SAG-POOL	82.00 ± 2.13	45.00 ± 4.21	61.73 ± 0.76	71.86 ± 0.97	93.50 ± 1.00
HGP-SL	83.00 ± 4.30	54.00 ± 0.04	59.51 ± 1.50	**84.91 ± 1.62**	95.50 ± 1.00
CommPOOL	**86.00 ± 1.23**	**66.50 ± 0.38**	**62.15 ± 0.37**	74.74 ± 0.06	**98.50 ± 0.05**

5.2.3. Summary of Results

Table 2 summarizes the classification performances of six models on five public datasets. Our CommPOOL outperforms all baselines in the graph classification task on almost all datasets, especially on the four-class data **Synthie**. For example, our CommPOOL shows about 5.11% improvement in the classification accuracy comparing to all baselines on **BZR** data. This superiority of CommPOOL may be credited to its advanced mechanism for capturing and preserving the community structure in the pooling operation. Also, these results indicate that the community is a crucial hierarchical structure for learning the whole graph representation.

Moreover, **Table 2** shows that hierarchical pooling methods generally perform better than global pooling methods, which verifies that the hierarchical pooling can better capture the graph global representations. Among all baseline models, HGP-SL relatively performs better than others, which may be attributed to the structure learning (SL) operations in the model. On PROTEINS, HGP-SL performs the best among all baseline methods and even better

than ours. As discussed in [21], the structure learning in the HGP-SL is superior to preserve more topology structure of the original protein graph after pooling several times. The topology structure preserved by HGP-SL may play an important role on PROTEINS data classification, which improve its classification performance on this data. However, our proposed CommPOOL is a general approach that can capture the inherent community structure existing in different kinds of graphs, therefore, it overall outperforms the baselines across multiple datasets.

5.3. Ablation Studies

5.3.1. Model Variations

To show the flexibility of CommPool, we compare several variations of CommPOOL on PROTEINS and FRANKENSTEIN data. As noted in **The Proposed Framework** section, GAT [7] and HO-GNN [42] are used to replace GCN as VGAE encoder variations. Moreover, instead of using the reciprocal of L_1 distance, we adopt the cosine-similarity as $Sim(\cdot)$ to measure the similarity between community members Z_{M_i} and the corresponding community center Z_{C_i} in community$-i$:

$$Sim(Z^w_{M_i}, Z_{C_i}) = \frac{Z^w_{M_i} Z_{C_i}}{\|Z^w_{M_i}\| \|Z_{C_i}\|} \tag{9}$$

The performance of CommPool with different encoders and similarity measures are listed in **Table 3**, which indicates that **GAT**, compared to **GCN**, has a better performance as the encoder in CommPOOL to embed the graph nodes. In addition, **Table 3** shows that L_1 distance is better than *cosine* distance when measuring the similarity between the latent features of community member nodes and the community center nodes. A possible explanation is that L_1 distance is used in the PAM clustering. Therefore, it may be better to use the same distance metric in the community partition process.

5.3.2. Parameters Analysis

To investigate how the hyperparameters impact the performance of the proposed CommPOOL, we show the graph classification results achieved by our

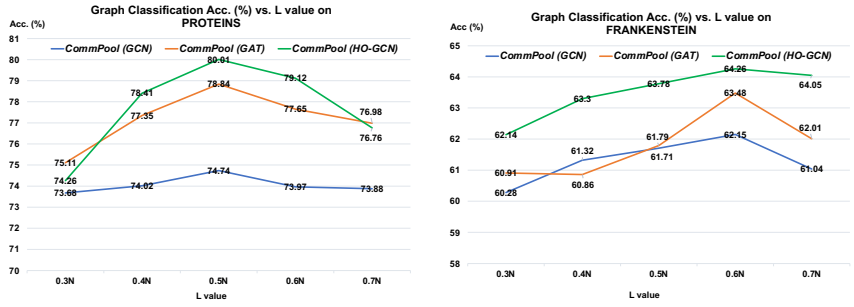
Table 3: Performance (%) of CommPOOL with different encoder settings and different similarity measures

CommPOOL		**PROTEINS**	**FRANKENSTEIN**
GCN	L_1	74.74 ± 0.06	62.15 ± 0.37
	cosine	73.84 ± 0.13	60.18 ± 0.42
GAT	L_1	78.84 ± 0.02	63.48 ± 0.52
	cosine	76.01 ± 0.21	62.32 ± 0.39
HO-GCN	L_1	$80.01. \pm 0.13$	64.26 ± 0.20
	cosine	77.09 ± 0.07	63.01 ± 0.11

model under different number of communities (L) and different latent feature dimensions on PROTEINS and FRANKENSTEIN data. As shown in the **Figure 3**, the performance of models is consistent with different L values. When increasing the L value from $0.3N$ to $0.7N$, the classification results tend to incline and decline. The best results appear when $L = 0.5N$ on PROTEINS data whereas appear when $L = 0.6N$ on FRANKENSTEIN data. This may be attributed to the different intrinsic community patterns in these two datasets. We show the model performance under different latent feature dimensions in **Table 4**. As shown in table **Table 4**, four different feature dimension combinations are set as the dimensions of latent feature generated by layers of EP module 1 and EP module 2 (e.g. $[32, 16], [64, 32]$ indicates that the dimensions of latent features generated by the two layers in the EP module 1 are 32 and 16, while the dimensions of latent features generated by the two layers in the EP module 2 are 64 and 32). It shows that, generally, the feature dimensions have slight impact on the classification results. Empirically, it indicates that lower feature dimensions should be set in EP module 1 whereas higher dimensions should be set in EP module 2.

5.4. Community Evaluation

In order to evaluate if CommPool can capture the community structures, we simulate a set of graphs with the known community ground-truth and evaluate



(a) (b)

Figure 3: Graph classification accuracy (%) performed by CommPool with there VGAE encoders (e.g. GCN, GAT and HO-GCN) on (**a**) PROTEINS data and (**b**) FRANKENSTEIN data under different number of communities (NC). The NC values range from 0.3N to 0.7N with a step as 0.1N, where N is the number of node in each graph.

Table 4: Average graph classification accuracy $\pm$ standard deviation (%) under different feature dimensions.

Feature Dimensions	PROTEINS	FRANKENSTEIN
$[32, 16], [64, 32]$	74.74 ± 0.06	62.15 ± 0.37
$[64, 32], [128, 64]$	73.61 ± 0.13	62.02 ± 0.14
$[64, 32], [32, 16]$	73.89 ± 0.25	61.29 ± 1.04
$[128, 64], [64, 32]$	74.26 ± 0.31	61.90 ± 0.81

how CommPOOL preserves the intrinsic community structures on these simulation graphs. Meanwhile, on the real-data, we use PROTEINS data with the node labels as the community ground-truth to evaluate the community structures captured by the CommPOOL.

5.4.1. Simulation Graphs

We create 3 classes of simulation graphs using different graph generating methods, including the Random Partition Graphs, the Relax Caveman Graphs, and the Gaussian Random Community Graphs [56, 57]. Each class contains 300 graphs and each graph has 4 communities with the average size of 6 nodes. A community label is assigned to each graph node. Meanwhile, we randomly sample from the normal distribution $\mathcal{N}(0, I)$ as node features. We evaluate

CommPOOL on the simulation graphs to predict their class labels. **Table 5** compares the graph classification performance of CommPool with the baseline models. The results show that, on the simulation data, the CommPOOL can also outperform all the baseline models. **N/A** in **Table 5** indicates the SAG-POOL cannot achieve an optimal point in reachable epochs.

Table 5: Average graph classification accuracy ± standard deviation (%) on the simulation data.

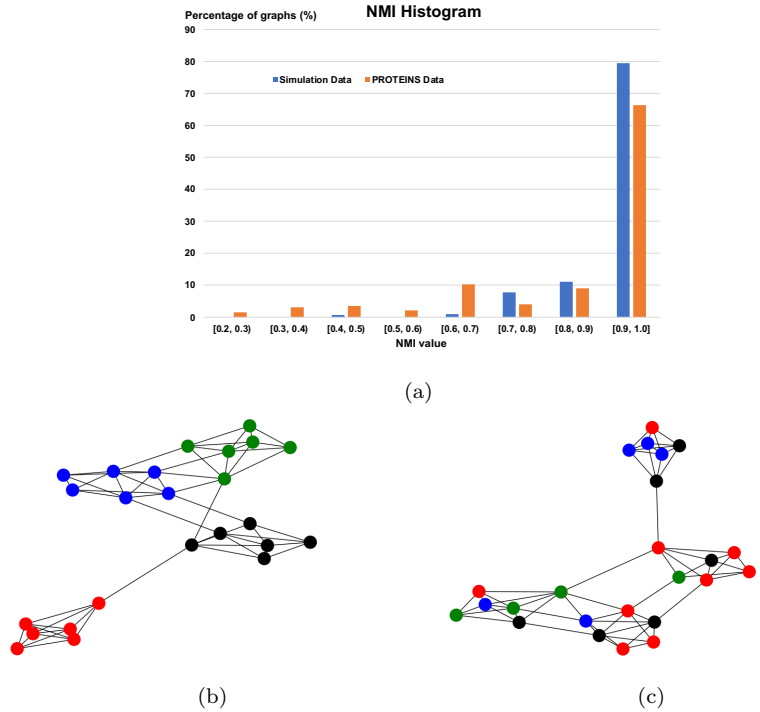
Models	Classification Accuracy
Set2Set	46.54 ± 3.85
SortPOOL	51.29 ± 0.61
DIFF-POOL	67.14 ± 2.16
SAG-POOL	N/A
HGP-SL	72.70 ± 1.95
CommPOOL	80.14 ± 2.15

5.4.2. Evaluation of Community Detection

In order to evaluate if CommPool can capture the community structures, we compare the node community label assigned by PAM clustering in the 1^{st} EP module to the community ground-truth labels. Specifically, we compute the **Normalized Mutual Information (NMI)** [58] between distribution of community labels predicted by the model and given by the ground-truth for each graph. **Figure 4a** is a histogram presenting the distribution of NMI scores for all 900 simulation graphs and 1113 PROTEINS data. Statistically, on the simulation data, 79.44% graphs have an NMI score larger than 0.9 and the mean NMI score is 0.9516 ± 0.098. On the PROTEINS data, 66.32% graphs have an NMI score larger than 0.9 and the mean NMI score is 0.8065 ± 0.032.

6. Evaluation and Discussion

In this section, we firstly discuss the interpretibility of our proposed community pooling operation. And then we analyze the importance of community



(a)

(b) (c)

Figure 4: (**a**) is the NMI histogram between the distributions of community labels of ground-truth and CommPOOL prediction. **(b)** is a positive example of community structure captured by CommPOOL leading to a correct classification. **(c)** is a negative example which leads to a graph misclassification. Different colors represent different communities.

structure to the graph classification task.

6.1. Interpretability of Community Pooling

CommPOOL is a hierarchical graph pooling framework with an interpretable pooling operation. The user can transparently understand the pooling results by monitoring the pooling operation. An interpretable pooling operation should be capable of clearly answering three questions mentioned in the **Introduction** section. Our CommPOOL provides the heuristic and knowledgeable answers for the questions in the following way:

- **Q1: How to capture the graph hierarchical structures in an interpretable way?**

The CommPOOL considers the communities as the basic graph hierarchical structure. In the community pooling operation, we adopt PAM to group graph nodes into different communities based on the similarities among their features. The goal of the pooling operations is integrating the similar features thereby coarsening the graphs. The CommPOOL coarsens the graphs based on the graph's inherent community structure captured by the PAM, which is a natural way that allows the further pooling operation to be able to integrate the features of similar nodes. In another word, the features of nodes assigned in the same community by the CommPOOL are similar and are supposed to be integrated into one node after the pooling operation. Therefore, the overall pooling procedure is intuitive and explainable. The community capture ability of our pooling operation has been shown in the previous **Community Evaluation on Simulation Data** section.

- **Q2: How to scale down the graph while preserving the structures via an interpretable process?**
  To scale down the graph, we choose the community medoid node as the representation of the whole community, which can be understood like the centroid can be used to represent the whole mass. Meanwhile, without loss of necessary graph structure information, an interpretable structure preservation process is introduced during downscaling the graph. The community pooling achieves the preservation via gathering the nodal and structure information of the community member nodes as the features of the community medoid node.

- **Q3: What do we obtain after the pooling operation?**
  From the graph topology view, the community pooling generates community-based sub-graphs of the original graph since the pooling operation does not generate any new graph nodes and edges. Each node in the sub-graph contains the corresponding community information.

6.2. Community Effect on Graph Classification

We design a further experiment named ***semi-random pooling*** to show that a solid community preservation is important to the graph classification. Instead of randomly partitioning the graph into multiple communities, we only randomly select the community center nodes. After determining the community center nodes, we assign each other node to the closest community based on the similarity of node features. Such a semi-random partition method can generate a few node cliques in graphs. These cliques, though are not the optimal communities, can still maintain the hierarchical information to some degree. We replace the PAM clustering by the semi-random partition in the pooling operation. **Table 6** indicates that the community pooling has significant improvements in the graph classification tasks comparing to the semi-random pooling, which demonstrates that the success of community capture and preservation is crucial to the graph classification. To visualize, we select two simulation graphs to show (1) a positive example of community structure captured by the CommPOOL (**Figure 4b**); and (2) a negative example of community structure captured by the CommPOOL, which eventually leads to the graph's misclassification (**Figure 4c**). In addition, the performance of semi-random pooling does not decrease a lot comparing with the community pooling, which is beyond our expectations in a way. A reasonable explanation is that although unable to preserve the optimal community structure, the semi-random pooling method can still capture some degree of graph hierarchical structure, which again justifies that the significance of the community structure in the graph.

Table 6: Graph Classification Accuracy of Semi-random Pooling vs. Community-based Pooling

Dataset	**semi-rand.**	**community-based**
PROTEINS	64.90 ± 2.45	74.74 ± 0.06
BZR	81.50 ± 2.82	86.00 ± 1.23
Synthie	59.00 ± 5.89	66.50 ± 0.38
Simulation	70.34 ± 1.26	80.14 ± 2.15

7. Conclusion

In this paper, we propose CommPOOL, a new interpretable hierarchical graph pooling framework. CommPOOL is designed for being able to capture and preserve the inherent hierarchical community structures in graphs during the graph representation learning and scaling-down process. Moreover, CommPOOL is a general graph representation learning framework that can facilitate various graph-level tasks. Experiments on both real-world graph datasets from different domains and synthetic graph data have shown that CommPOOL outperforms the state-of-the-art methods in graph representation learning for the graph classification task. In future work, we will explore leveraging CommPOOL for other graph-level tasks, such as graph regression.

8. Acknowledgement

This study was funded in part by NSF-IIS 1837956, NSF-IIS 1852606, NSF-IIS 2045848, and NSF-IIA 2040588. Also, it used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges system, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

References

[1] J. Chen, T. Ma, C. Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, arXiv preprint arXiv:1801.10247 (2018).

[2] W. Huang, T. Zhang, Y. Rong, J. Huang, Adaptive sampling towards fast graph representation learning, in: Advances in neural information processing systems, 2018, pp. 4558–4567.

[3] H. Dai, B. Dai, L. Song, Discriminative embeddings of latent variable models for structured data, in: International conference on machine learning, 2016, pp. 2702–2711.

[4] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R. P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in: Advances in neural information processing systems, 2015, pp. 2224–2232.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, arXiv preprint arXiv:1704.01212 (2017).

[6] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[8] T. N. Kipf, M. Welling, Variational graph auto-encoders, arXiv preprint arXiv:1611.07308 (2016).

[9] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in neural information processing systems, 2017, pp. 1024–1034.

[10] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, arXiv preprint arXiv:1511.05493 (2015).

[11] O. Vinyals, S. Bengio, M. Kudlur, Order matters: Sequence to sequence for sets, arXiv preprint arXiv:1511.06391 (2015).

[12] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An end-to-end deep learning architecture for graph classification, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[13] M. Girvan, M. E. Newman, Community structure in social and biological networks, Proceedings of the national academy of sciences 99 (12) (2002) 7821–7826.

[14] Q. Long, Y. Wang, L. Du, G. Song, Y. Jin, W. Lin, Hierarchical community structure preserving network embedding: A subspace approach, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 409–418.

[15] V. Spirin, L. A. Mirny, Protein complexes and functional modules in molecular networks, Proceedings of the national Academy of sciences 100 (21) (2003) 12123–12128.

[16] X. Kong, P. S. Yu, Brain network analysis: a data mining perspective, ACM SIGKDD Explorations Newsletter 15 (2) (2014) 30–38.

[17] D. Meunier, R. Lambiotte, A. Fornito, K. Ersche, E. T. Bullmore, Hierarchical modularity in human brain functional networks, Frontiers in neuroinformatics 3 (2009) 37.

[18] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in: Advances in neural information processing systems, 2018, pp. 4800–4810.

[19] C.-F. Lee, J.-J. Shen, K.-L. Hou, F.-W. Hsu, A high-performance computing method for photographic mosaics upon the hadoop framework, Journal of Internet Technology 20 (5) (2019) 1343–1358.

[20] H. Gao, S. Ji, Graph u-nets, arXiv preprint arXiv:1905.05178 (2019).

[21] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, C. Wang, Hierarchical graph pooling with structure learning, arXiv preprint arXiv:1911.05954 (2019).

[22] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial Intelligence 267 (2019) 1–38.

[23] C. Molnar, Interpretable Machine Learning, Lulu. com, 2020.

[24] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song, Adversarial attack on graph structured data, arXiv preprint arXiv:1806.02371 (2018).

[25] D. Zügner, A. Akbarnejad, S. Günnemann, Adversarial attacks on neural networks for graph data, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2847–2856.

[26] D. Zügner, S. Günnemann, Certifiable robustness of graph convolutional networks under structure perturbations, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1656–1665.

[27] H. Tang, G. Ma, Y. Chen, L. Guo, W. Wang, B. Zeng, L. Zhan, Adversarial attack on hierarchical graph pooling neural networks, arXiv preprint arXiv:2005.11560 (2020).

[28] Z. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, in: Advances in neural information processing systems, 2019, pp. 9244–9255.

[29] Y. Hou, J. Zhang, J. Cheng, K. Ma, R. T. Ma, H. Chen, M.-C. Yang, Measuring and improving the use of graph information in graph neural networks, in: International Conference on Learning Representations, 2019.

[30] H. Yuan, J. Tang, X. Hu, S. Ji, Xgnn: Towards model-level explanations of graph neural networks, arXiv preprint arXiv:2006.02587 (2020).

[31] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. P. Yu, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems (2020).

[32] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data. arxiv 2015, arXiv preprint arXiv:1506.05163 (2015).

[33] R. Levie, F. Monti, X. Bresson, M. M. Bronstein, Cayleynets: Graph convolutional neural networks with complex rational spectral filters, IEEE Transactions on Signal Processing 67 (1) (2018) 97–109.

[34] H. Tang, L. Guo, E. Dennis, P. M. Thompson, H. Huang, O. Ajilore, A. D. Leow, L. Zhan, Classifying stages of mild cognitive impairment via augmented graph embedding, in: Multimodal Brain Image Analysis and Mathematical Foundations of Computational Anatomy, Springer, 2019, pp. 30–38.

[35] I. S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE transactions on pattern analysis and machine intelligence 29 (11) (2007) 1944–1957.

[36] D. V. Tran, N. Navarin, A. Sperduti, On filter size in graph convolutional networks, in: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2018, pp. 1534–1541.

[37] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, arXiv preprint arXiv:1904.08082 (2019).

[38] Z. T. Kefato, S. Girdzijauskas, Graph neighborhood attentive pooling, arXiv preprint arXiv:2001.10394 (2020).

[39] F. M. Bianchi, D. Grattarola, C. Alippi, Spectral clustering with graph neural networks for graph pooling., arXiv: Learning (2020).

[40] E. Ranjan, S. Sanyal, P. P. Talukdar, Asap: Adaptive structure aware pooling for learning hierarchical graph representations., in: AAAI, 2020, pp. 5470–5477.

[41] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arXiv preprint arXiv:1312.6203 (2013).

[42] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, M. Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4602–4609.

[43] L. Kaufmann, Clustering by means of medoids, in: Proc. Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, 1987, 1987, pp. 405–416.

[44] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, M. Neumann, Benchmark data sets for graph kernels (2016).
URL `http://graphkernels.cs.tu-dortmund.de`

[45] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, H.-P. Kriegel, Protein function prediction via graph kernels, Bioinformatics 21 (suppl_1) (2005) i47–i56.

[46] P. D. Dobson, A. J. Doig, Distinguishing enzyme structures from non-enzymes without alignments, Journal of molecular biology 330 (4) (2003) 771–783.

[47] C. Morris, N. M. Kriege, K. Kersting, P. Mutzel, Faster kernels for graphs with continuous attributes via hashing, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 1095–1100.

[48] F. Orsini, P. Frasconi, L. De Raedt, Graph invariant kernels, in: Proceedings of the twenty-fourth international joint conference on artificial intelligence, Vol. 2015, IJCAI-INT JOINT CONF ARTIF INTELL, 2015, pp. 3756–3762.

[49] J. J. Sutherland, L. A. O'brien, D. F. Weaver, Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships, Journal of chemical information and computer sciences 43 (6) (2003) 1906–1915.

[50] K. Riesen, H. Bunke, Iam graph database repository for graph based pattern recognition and machine learning, in: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2008, pp. 287–297.

[51] Y. Ma, S. Wang, C. C. Aggarwal, J. Tang, Graph convolutional networks with eigenpooling, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 723–731.

[52] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, et al., Xsede: Accelerating scientific discovery computing in science & engineering, 16 (5): 62–74, sep 2014, URL https://doi. org/10.1109/mcse (2014).

[53] N. A. Nystrom, M. J. Levine, R. Z. Roskies, J. R. Scott, Bridges: a uniquely flexible hpc resource for new communities and data analytics, in: Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure, 2015, pp. 1–8.

[54] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).

[55] M. Fey, J. E. Lenssen, Fast graph representation learning with pytorch geometric, arXiv preprint arXiv:1903.02428 (2019).

[56] U. Brandes, M. Gaertler, D. Wagner, Experiments on graph clustering algorithms, in: European Symposium on Algorithms, Springer, 2003, pp. 568–579.

[57] S. Fortunato, Community detection in graphs, Physics reports 486 (3-5) (2010) 75–174.

[58] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, Journal of machine learning research 3 (Dec) (2002) 583–617.