iPool - Information-based Pooling in Hierarchical Graph Neural Networks

Xing Gao [*]
Shanghai Jiao Tong University
Department of Electronic Engineering
william-g@sjtu.edu.cn

Hongkai Xiong
Shanghai Jiao Tong University
Department of Electronic Engineering
xionghongkai@sjtu.edu.cn

Pascal Frossard
École Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory (LTS4)
pascal.frossard@epfl.ch

Abstract

With the advent of data science, the analysis of network or graph data has become a very timely research problem. A variety of recent works have been proposed to generalize neural networks to graphs, either from a spectral graph theory or a spatial perspective. The majority of these works however focus on adapting the convolution operator to graph representation. At the same time, the pooling operator also plays an important role in distilling multiscale and hierarchical representations but it has been mostly overlooked so far. In this paper, we propose a parameter-free pooling operator, called iPool, that permits to retain the most informative features in arbitrary graphs. With the argument that informative nodes dominantly characterize graph signals, we propose a criterion to evaluate the amount of information of each node given its neighbors, and theoretically demonstrate its relationship to neighborhood conditional entropy. This new criterion determines how nodes are selected and coarsened graphs are constructed in the pooling layer. The resulting hierarchical structure yields an effective isomorphism-invariant representation of networked data in arbitrary topologies. The proposed strategy is evaluated in terms of graph classification on a collection of public graph datasets, including bioinformatics and social networks, and achieves state-of-the-art performance on most of the datasets.

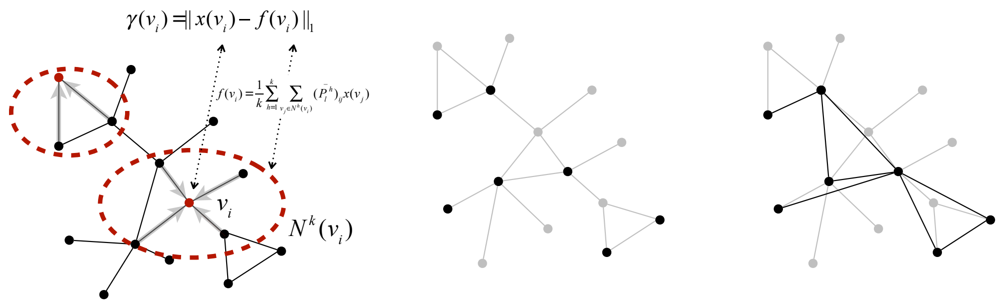
1 Introduction

Convolution neural networks (CNNs) are efficient to extract hierarchical representations of signals residing on regular grids, such as audios and images. With the convolution and pooling operations, CNNs have achieved state-of-the-art performance in a variety of applications. With the increasing availability of various forms of network data, recent pioneer works [2, 5, 9, 10, 26] have been generalizing convolution neural networks to irregular structures, including graph and point cloud data. Most of the current attempts for designing neural network representations of graph data however focus on the convolution operator. The pooling operator is mostly overlooked, yet it carries an important part of the ability of graph neural networks to distill effective hierarchical representations.

Hierarchical network data representations necessitate a careful design of all elements of the learning architecture. In tasks like graph classification for instance, a global representation is required in

[*]Work performed at EPFL-LTS4, visiting Prof. Pascal Frossard.

Preprint. Under review.


$\gamma(v_i) = \| x(v_i) - f(v_i) \|_1$
$f(v_i) = \frac{1}{k}\sum_{h=1}^{k}\sum_{v_j \in N^h(v_i)} (\bar{P_l}^h)_{ij} x(v_j)$
v_i
$N^k(v_i)$

(a) The neighborhood information gain of each node with a neighborhood aggregation prediction function (k=1).

(b) Node selection: the top m nodes with the largest $\gamma(\overline{\gamma})$ are preserved.

(c) Coarsened graph construction: the induced graph of the s-order graph (s=2).

Figure 1: Illustration of the iPool. The proposed neighborhood information gain criterion $\gamma(\cdot)$ and prediction function $f(\cdot)$ are demonstrated in (a). The node selection strategy on the basis of the γ ($\overline{\gamma}$) and coarsened graph construction are presented in (b) and (c), respectively. Specifically, $x(v_i)$ indicates the signal or feature on the node v_i, $N^k(v_i)$ represents the k-hop neighbors of the node v_i, and $\bar{P_l}^h$ implies an off-diagonal version of the h-th power of transmission matrix.

addition to local features in order to predict the label for an entire graph. The pooling operator is an important component in the construction of such hierarchical architectures. In the case of image representation, the pooling operator downsamples data simply by flipping the predefined local receptive field and aggregating information in each receptive field, from left to right and top to bottom taking advantage of the inherent spatial order in lattice structures. However, in the general case of networks with diverse irregular connections between vertices and no spatial order of the nodes, the pooling operation becomes more challenging. In particular, it is not appropriate to directly generalize the pooling operator from images to graphs. Even for the simple pooling operation that downsamples image signals with stride 2, its counterpart on graphs that can be formulated as a max-cut problem, becomes a NP-hard problem [3]. In addition, it is still a challenge to construct a coarsened graph with the pooled features. In other words, one has to solve the following two problems to generalize the pooling operation to graphs: (1) How to choose and aggregate information to characterize the signals residing on graph? (2) How to coarsen the structure of graphs in the higher levels of representation? Graph theory may provide several tools to generalize the pooling operator to networks with help of graph clustering or graph coarsening algorithms [25, 21, 8]. However, although these methods perform well on fixed graph structures, they fail to easily generalize to networks with different topologies. Furthermore, their high computational complexity might become a limitation in graph convolution neural networks.

In this paper, we propose a pooling operator, called iPool, for data that comes with arbitrary graphs. This new operator is designed to generate a faithful representation of signals supported on the graph in terms of their information. Therefore, it first uses a criterion to evaluate the amount of information carried by each node given observations of its neighbors. Then, as informative nodes play an important role in characterizing graphs, we propose a strategy to construct coarsened graphs by aggregating information in accordance with the proposed information criterion, such as the intrinsic structure of the original graph is preserved. In particular, the node signals are predicted from the neighbour signals values to determine the most informative nodes, as shown in Fig. 1. A coarsened graph is constructed on these most informative nodes with a topology that maintains consistency with the original graph. The proposed iPool scheme offers several advantages: 1) it is a parameter-free building block, and it can be paired with diverse graph convolution operators in order to deal with arbitrary graphs; 2) it is invariant to graph isomorphism, which means that coarsened (isomorphic) graphs are identical through the pooling operation; 3) it exploits signals residing on graph in addition to the structure of graph; 4) it could provide two kinds of pooling schemes, namely global and local pooling which permit to trade off preservation of graph structure and information aggregation. Finally, we resort to graph classification tasks in order to evaluate the proposed method and we show that it outperforms state-of-the-art graph neural networks based methods on a collection of public benchmark datasets. The proposed pooling operator also has a promising perspective to be applied to other discriminative tasks and generalized easily to other non-Euclidean data, such as point clouds.

2 Related Work

We give a brief review below on a collection of works on graph representation learning. We focus on traditional graph coarsening schemes as well as on recent attempts in graph pooling, which are relevant to the main problem addressed in this work.

Graph neural networks (GNNs) are first proposed in [6, 17, 18] and some recent works have attempted to extend convolution neural networks to graphs, termed graph convolution networks (GCNs). One category of methods define convolutions for graphs on the basis of spectral filtering of graph signals, which leads to spectral GCNs [2, 5, 9, 10]. Another line of works focus on defining the convolution operation in the spatial domain [2, 7, 26, 23, 29, 27] to address the basis-dependent problem where a spectral graph neural network trained on one graph structure fails to transfer properly to other graph structures. Most of them generalize the convolution operation by defining a "message aggregation" scheme, where information in the neighborhood is aggregated. These convolution layers in these architectures might be combined with different pooling operators to generate hierarchical representations of graph data.

When it comes to building multiscale representations of graph data, previous works rely on variants of graph coarsening algorithms, or differnet forms of graph pooling. For general graphs, unfortunately, this problem becomes a complex problem and a collection of approximate methods have been proposed in the literature, such as multiscale methods [21, 8]. Yet, computational complextiy stays high, so that these methods are mostly suitable for processing fixed structure graphs off-line, but have clear limitations in providing an on-line building block able to cope with graphs of various structures. An alternative to graph coarsening consists in implementing spectral clustering which split graphs into parts that can yield a hierarchical representation [25]. Spectral clustering is however not ideal in building multiscale representations of graph data.

Pooling probably provides the most constructive alternative to develop multiscale representation in graph neural networks, and different methods have been proposed for generalizing poling ideas to graphs. For example Zhang et al. [29] propose SortPooling that sorts the nodes based on the value of the last feature map in a descending manner, and preserves the first k nodes of this list. On the other hand, DIFFPOOL [28] follows clustering methods and assign nodes softly with generating the cluster assignment matrix using neural networks. SET2SET [24] then implements an equivalent global pooling by aggregating information through LSTMs. However, these two recent methods have limitations: DIFFPOOL requires another graph neural network with the same capacity as the network for main task to learn projection matrices and the SET2SET depends on extra LSTM units which are time-consuming to train, in order to gather information.

In contrary to previous methods, we propose a graph pooling scheme in accordance with the information of graph signals to enhance the capability of GNNs to construct hierarchical representations of graphs with arbitrary topologies. Similarly to the pooling operation for images, the proposed graph pooling operator is parameter-free and "plug and play", and thereby leads to fast implementations in both training and testing phases.

3 The iPool algorithm

We present in this section the main elements of our new iPool algorithm. We first introduce the general framework of graph neural network architectures as well as the notation used in the paper. We later define the role of the pooling algorithm and introduce an information gain criterion that drives our iPool algorithm. We finally show how our pooling algorithm can be used to construct hierarchical representations with both global and local graph pooling strategies.

3.1 Framework

In this paper, we consider the learning of hierarchical representations for network data with help of a neural network architecture. Such an architecture is typically built on the concatenation of several layers, composed of graph convolution blocks and pooling operators. We focus here on the pooling operator, which is a key element for effective learning of hierarchical representations.

We define the notation used in this paper, where we employ capital letters and bold lowercase letters to indicate matrices and vectors, respectively. The network data at the input of the learning architecture, is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ and $\mathcal{E}$ respectively the set of vertices, and the set of edges. The adjacency matrix A represents the topology of the network, and presents a non-zero value at position (i, j) (i.e., $A_{ij} \neq 0$) only if there is an edge in $\mathcal{E}$ that connects vertices i and j. We consider both unweighted and weighted graphs, which leads to A consisting of respectively unitary values or actual edge weights. We further define D and P as the degree matrix and respectively the transmission matrix of $\mathcal{G}$. The degree matrix D is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$ and the transmission matrix P defined as $P = D^{-1}A$ denotes the transmission probability of each pair of nodes. Furthermore, each vertex of the graph might further be attributed a signal value or feature, which is denoted by $\mathbf{x}_i = x(v_i) \in \mathbb{R}^d$, and $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^T$. Finally, as we construct a hierarchical representation of graph data, the subscript l is used to denote features or parameters belonging to the l-th layer of the neural networks. For example, for the graph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ in the l-th neural network representation layer, the i-th vertex of the graph is written as $v_{l,i} \in \mathcal{V}_l$, and $\mathbf{x}_{l,i} = x(v_{l,i})$ represents the signal or feature residing on the node $v_{l,i}$.

Most of current graph convolution networks utilize a stack of graph convolution layers to learn a representation or embedding of the graph data. The graph convolution layers are usually designed to follow a neighborhood message aggregation scheme, which can be written as:

$$X_{l+1} = \sigma(g(S_l, X_l, W_l)), \tag{1}$$

where $S_l \in \mathbb{R}^{n_l \times n_l}$ can be any graph shift operator that has nonzero values only in the positions corresponding to edges in the graph and in its diagonal, including but not limited to A_l and P_l. The functions $g(\cdot)$ and $\sigma(\cdot)$ respectively denote a data aggregation function and a non-linear activation function, and the parameters W_l are learned during the training of the neural network. The aggregation function $g(\cdot)$ varies in different graph convolution network architectures and is usually defined as (weighted) summation [28], mean [7, 29], or even a Multilayer Perceptron (MLP) [27].

A key element in these architectures is the graph pooling or coarsening operator, which aims at selecting a subset of data and obtain a version with a reduced dimension that still represents well the original graph data. These operators equip the neural networks with the ability to construct a sort of multiscale representations of network data. However, due to the diverse structures of graphs and the absence of a regular grid-like topology in general, it is not possible to pre-define the sampling structure or the local receptive fields on graphs, as it can be done in image representation learning [3]. The graph pooling operator has thus to be defined adaptively, yet in a generic way so that it can accommodate different network structures.

Generally, for an undirected graph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$, the graph pooling process could be formulated as:

$$A_{l+1} = C_l \; A_l \; {C_l}^T, \tag{2}$$

where $A_{l+1} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}$ and $A_l \in \mathbb{R}^{n_l \times n_l}$ are the adjacency matrices of the coarse and fine graphs and $C_l \in \mathbb{R}^{n_{l+1} \times n_l}$ is the coarsening matrix. Correspondingly, the signal X_l residing on the graph is downsampled by the coarsening matrix:

$$X_{l+1} = C_l X_l. \tag{3}$$

Notably, Eq.(2) and Eq.(3) are general formations that are suitable for both traditional graph coarsening as well as clustering methods [13, 4, 12] and state-of-the-art graph pooling methods [28].

The design of effective hierarchical representations of graph data largely relies on the proper choice of the graph pooling operator, hence on a proper choice of the operator C. We present below a novel pooling strategy using a neighborhood information gain criterion.

3.2 The neighborhood information gain criterion

In order to design a proper pooling operator, one first needs to define a criterion that governs the selection of the most important nodes in the graph. The objective is to coarsen the graph representation while keeping a faithful representation of the original one, with the coarsening matrix or equivalently the pooling operator generated on the basis of the structure of graph and the signals that it supports. In general, if a signal residing on one particular node of the graph could be well predicted from signals supported by other nodes, this node can probably be removed in the coarsened graph, with negligible information loss. If we further consider the typical localization and smoothness properties of most signals, it is reasonable to limit the node signal prediction process within the node neighborhood. Therefore, we can relate the amount of information carried by a graph node, to the difficulty of predicting the signal value from nodes in its neighborhood. We therefore introduce below a measure, namely the neighborhood information gain criterion, in order to quantitatively evaluate the uncertainty or information of node signals given observations of the neighbors. We later use this measure to design a new pooling operator that eventually preserves the most important nodes in the graph.

The neighborhood information gain criterion is defined as the "Manhattan" distance between the observed signal $\mathbf{x}_{l,i} = x(v_{l,i})$ and the one predicted from observations at the neighbor nodes [2]. We choose the "Manhattan" distance as it represents a common similarity measure that is especially convenient for high dimensional vectors that might be present in some graph datasets. Specifically, with a prediction function $f(\cdot)$ using information from the neighborhood, the neighborhood information gain of each node could be formulated as:

$$\gamma(v_{l,i}) = \parallel x(v_{l,i}) - f(v_{l,i}) \parallel_1 . \tag{4}$$

With this definition, the neighborhood information gain criterion gives the same importance to differences along each dimension; using an ℓ_1 norm prevents specific dimensions with large deviations from dominating the other ones as it typically happens with ℓ_p norms with $p > 1$. We now have different options for choosing the prediction function $f(\cdot)$. Among them, neighborhood aggregating functions are promising, in particular when considering the typical localization and smoothness properties of graph signals. We therefore choose to predict the node signal as the weighted average of signals supported on nodes within its k-hop neighborhood. Given that the h-th power of transmission matrix ${P_l}^h$ is a effective measure of the level of connection or dependence between any pair of nodes reachable with h hops, we adopt the elements of ${P_l}^h (h = 1, 2, \dots, k)$ as the weights in our prediction function f in order to give more confidence to nodes that have stronger connections. We however modify ${P_l}^h$ into an off-diagonal transition matrix $\bar{P_l}^h$ corresponding to graph without h-hop circles. Finally, the prediction function can be formulated as

$$f(v_{l,i}) = \frac{1}{k} \sum_{h=1}^{k} \sum_{v_{l,j} \in N^h(v_{l,i})} (\bar{P_l}^h)_{ij} x(v_{l,j}), \tag{5}$$

with

$$\bar{P_l}^h = (\bar{D_l}^h)^{-1} \bar{A_l}^h, \quad \bar{A_l}^h = {A_l}^h - \mathrm{diag}({A_l}^h), \tag{6}$$

where N^h is the h-hop neighborhood, $\bar{A_l}^h$ is the adjacency matrix where diagonal values corresponding to the h-hop circles have been removed, and $\bar{D_l}^h$ is the corresponding degree matrix. In this way, Eq. (4) could be further formulated as:

$$\gamma(v_{l,i}) = \parallel x(v_{l,i}) - \frac{1}{k} \sum_{h=1}^{k} \sum_{v_{l,j} \in N^h(v_{l,i})} (\bar{P_l}^h)_{ij} \times x(v_{l,j}) \parallel_1, \tag{7}$$

$$\Gamma(\mathcal{G}_l) = \mid (I - \frac{1}{k} \sum_{h=1}^{k} \bar{P_l}^h) X_l \mid, \tag{8}$$

where we use $\mid \cdot \mid$ to indicate the l_1 norm of each row of a matrix. The value of γ will be high when a node signal is very different from the ones in its neighborhood, which means that the node has high information and should be preserved in the pooling operator. Note that the definition of the information criterion is local, which is very important towards low complexity and possible distributed implementation.

3.3 Graph Coarsening

With the neighborhood information gain criterion defined above, we can now identify nodes that should be preserved by the pooling operator. The convolution layers based on message aggregation (see Eq. (1)) performs a series of equivalent low-pass filtering of the graph information. The pooling then identifies the nodes that have the higher information gain and permits to adaptively downsample the graph while preserving the local characteristics of graph signals. We describe below a local and

[2]For graph without signals, graph structure is taken as signals by using features after convolution layers with constant features as input (i.e., $\mathbf{1}$ for all nodes) implicitly.

global version of the pooling algorithm, and then show how we construct coarsened graphs after pooling in order to build a multiscale representation of network data.

Similarly to pooling strategies developed for image representations, a global pooling scheme can aggregate information of the whole graph globally at the possibly expense of losing the graph structure information, while local pooling strategy can preserve the general structure of graph with however a limited receptive field to collect information. With different applications having diverse requirements, it is worthwhile to explore both of global pooling and local pooling for graphs. Luckily, with the neighborhood information criterion defined above, we can derive both global pooling as well as local pooling strategies with only minor adjustments.

Global iPool strategy. On the basis of the neighborhood information gain, nodes are assigned different priorities to construct coarse graph globally. In order to approximate the information of the graph, the pooling should preserve the nodes that can not be well represented by their neighbors. In other words, the nodes with relative high neighborhood information gain have to be preserved in the construction of a coarsened graph. Specifically, the graph nodes are re-ordered based on the value of their neighborhood information gain. The global pooling strategy then simply select the top $n_{l+1} = \rho \times |\mathcal{V}_l|$ nodes that have the highest information gain, as:

$$\mathbf{idx} = \mathrm{rank}(\Gamma(\mathcal{G}_l), n_{l+1}), \tag{9}$$

where ρ is the pooling ratio and rank represents the global ranking operator.

Local iPool strategy. Pooling can also be implemented locally, and nodes can be selected within each receptive field, similarly to what is done for images. Receptive fields may however overlap, and we propose to normalize the information gain in each neighborhood before local pooling. This reduces the probability that selected nodes mainly come from dominant neighborhoods and permits to have a better distribution of the pooled nodes over the original graph. Mathematically, the neighborhood information gain of each node is normalized by the average neighborhood information gain of its neighborhood:

$$\overline{\gamma}(v_{l,i}) = \frac{\gamma(v_{l,i})}{\sum\limits_{v_{l,j} \in N(v_{l,i})} (P_l)_{ij}\gamma(v_{l,j})} \tag{10}$$

Then, the nodes are ordered and selected globally in terms of the normalized neighborhood information gain:

$$\mathbf{idx} = \mathrm{rank}(\overline{\Gamma}(\mathcal{G}_l), n_{l+1}). \tag{11}$$

Coarsened graph construction. The above iPool versions select the nodes to be preserved and we can now construct a coarsened graph with the selected nodes. First, the coarsening matrix of Eq. (2) and Eq. (3) can be obtained directly from the indices of the nodes selected by iPool. Concretely, with the indices of the pooled nodes, the coarsening matrix $C_l \in \mathbb{R}^{n_{l+1} \times |\mathcal{V}_l|}$ is constructed according to the following formulation, for $\forall i \in \{1, 2, \ldots, n_{l+1}\}$:

$$(C_l)_{ij} = \begin{cases} 1 & j = \mathbf{idx}(i) \\ 0 & \text{others.} \end{cases} \tag{12}$$

The coarsening matrix is sparse; the cardinality of each row is 1, and the cardinality of each column is no more than 1.

The coarsened graph is constructed as follow. To facilitate propagation of signals on the graph, we first alter the connection between nodes in the original graph. Specifically, there is an edge connecting two nodes if and only if there is a path consisting of s edges between these two nodes in the original graph. The weights of the edges are further set either as the corresponding elements of the s-th power of the adjacency matrix for weighted graphs or as 1 for unweighted graphs. Namely, we set

$$(\tilde{A}_l)_{ij} = 1 \iff ({A_l}^s)_{ij} > 0, \quad i \neq j, \tag{13}$$

for unweighted graphs. For weighted graphs, we set

$$\tilde{A}_l = {A_l}^s - \mathrm{diag}({A_l}^s), \tag{14}$$

where the value of s is set to be either 1 or 2 depending on the properties of the datasets and applications. Notably, the hyper-parameter s plays a similar role as the hyper-parameter stride in image pooling. The coarsening matrix in Eq. (12) is applied to the adjacency matrix of the expanded

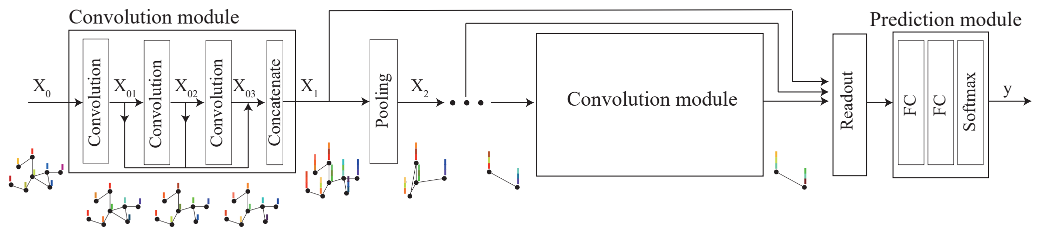


Figure 2: Hierarchical graph convolution network architecture. The network is composed of a stack of convolution modules, iPool layers, a readout module as well as a prediction module.

graph $\tilde{A}_l$ to obtain the coarsened graph. The graph and features at the next layer of the representation thus become:

$$A_{l+1} = C_l \tilde{A}_l {C_l}^T, \tag{15}$$

$$X_{l+1} = C_l X_l. \tag{16}$$

Note that, during the training phase of the neural network architecture, the iPool operator as any general end-to-end differentiable layer, takes as inputs the adjacency matrix and node features of graphs and produces the adjacency matrix and node features of coarsened graphs. Furthermore, the operator is generic enough to be integrated in diverse architectures.

4 Properties of iPool

In this section, we will discuss the relationship between the neighborhood information gain and neighborhood conditional entropy and derive some properties of iPool.

The neighborhood information gain is defined as the deviation between the observed signal and the predicted signal based on signals in the neighborhood. Empirically, this deviation reflects the uncertainty of one signal value given other values in its neighborhood. In information theory, the entropy is designed to quantify the uncertainty, and the conditional entropy is specifically utilized to measure the amount of information of one variable given values of the other variable(s). Generalized to graph signals, we arrive at the neighborhood conditional entropy by considering variables as signals supported on nodes in neighborhood:

$$H(\mathbf{x}_{l,i} | \{\mathbf{x}_{l,j}\}_{N(v_{l,i})}) = H(\mathbf{x}_{l,i}) - I(\mathbf{x}_{l,i}; \{\mathbf{x}_{l,j}\}_{N(v_{l,i})}), \tag{17}$$

where $H(\cdot)$ is the entropy and $I(\cdot)$ represents the mutual information. Actually, under a certain assumption on the conditional distribution, the proposed neighborhood information gain has a close relationship with the neighborhood conditional entropy.

Proposition 1. *Let assume that the components of neighborhood conditional distribution of each node are independent and that each component satisfies a Laplace distribution $p(x_{l,i,z} | \{\mathbf{x}_{l,j}\}_{N(v_{l,i})})$= Laplace($\mu_{l,i,z}, b_{l,i}$) with $\mu_{l,i} = [\mu_{l,i,1}, \mu_{l,i,2}, \ldots, \mu_{l,i,d}] = f(v_{l,i})$, the neighborhood information gain $\gamma(v_{l,i})$ of each node is an approximate empirical estimation of its neighborhood conditional entropy.*

With Prop. 1, the proposed iPool algorithm is a constructive way to coarsen graphs in accordance with *the maximum neighborhood conditional entropy strategy*. Specifically, the global iPool preserves nodes with the maximum neighborhood conditional entropy, while the local iPool selects nodes with relatively large neighborhood conditional entropy. Furthermore, the global iPool strategy implies that nodes share the same or similar variations of neighborhood conditional distribution, while the local iPool strategy relaxes this restriction to nodes in the same neighborhood having similar variations in terms of neighborhood conditional distribution. The proof is presented in Appendix.

We now further show that iPool is invariant to isomorphic graphs so that the graph neural networks consisting of iPool combined with other components that are also invariant to graph isomorphism, will produce invariant representations for isomorphic graphs. This property is important for a wide collection of discriminative tasks, such as graph classification. We prove this proposition in Appendix.

Table 1: Graph classification accuracies with 10-fold cross-validation with the large networks. Results of baseline methods with '*' and '†' are respectively cited from [28] and the respective publications.

Method	ENZY	D&D	RED-M-12K	COLL	PROT
GRAPHLET*	41.03	74.85	21.73	64.66	72.91
SP*	42.32	78.86	36.93	59.10	76.43
WL*	53.43	74.02	39.03	78.61	73.76
WL-OA*	**60.13**	79.04	44.38	**80.74**	75.26
PSCN†	-	76.27 ± 2.64	41.32 ± 0.42	72.60 ± 2.15	75.00 ± 2.51
GRAPHSAGE	54.00 ± 4.36	78.08 ± 3.52	41.19 ± 7.19	76.28 ± 1.67	76.55 ± 4.20
ECC†	53.50	74.10	41.73	67.79	72.65
SET2SET	55.50 ± 6.99	78.51 ± 3.59	> 7days	75.46 ± 1.40	76.73 ± 3.97
SORTPOOL†	57.12	79.37	41.82	73.76	75.54
DIFFPOOL	58.67 ± 5.37	79.19 ± 4.17	47.37 ± 1.02	76.38 ± 1.93	77.00 ± 2.93
Proposed global	56.00 ± 7.72	78.76 ± 3.45	47.02 ± 1.01	76.86 ± 1.67	76.46 ± 3.22
Local ($k=1$)	$\mathbf{59.00\pm5.73}$	$\mathbf{79.45\pm2.78}$	47.24 ± 1.10	$\mathbf{77.28\pm2.17}$	$\mathbf{77.36\pm3.27}$
Local ($k=2$)	57.50 ± 6.20	78.93 ± 4.07	$\mathbf{47.64\pm1.56}$[3]	77.20 ± 1.76	76.72 ± 3.06

Table 2: Dataset statistics and properties.

Method	ENZY	D&D	RED-M-12K	COLL	PROT	NCI1	NCI109	MUTAG	IMDB-B	IMDB-M
Avg $\|\mathcal{V}\|$	32.63	284.32	391.41	74.49	39.06	29.87	29.68	17.93	19.77	13.00
Avg $\|\mathcal{E}\|$	62.14	715.66	456.89	2457.78	72.82	32.30	32.13	19.79	96.53	65.94
Node labels	V	V	X	X	V	V	V	V	X	X
#Classes	6	2	11	3	2	2	2	2	2	3
#Graphs	600	1178	11929	5000	1113	4110	4127	188	1000	1500

Proposition 2. *For any isomorphic graphs $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l, X_l)$ and $\mathcal{G}_l^{'} = (\mathcal{V}_l^{'}, \mathcal{E}_l^{'}, X_l^{'})$, the iPool will produce the same coarsened graphs.*

In summary, the information gain criterion used in iPool is actually a constructive approximation of the maximum neighborhood conditional entropy, which further validates the choice of this measure to preserve the graph information. Furthermore, the iPool algorithm is invariant under graph isomorphism, which is very important in numerous tasks.

5 Experiments

In this section, we evaluate the effectiveness of the proposed pooling scheme in graph classification tasks, and compare with a variety of kernel based methods and neural network based methods.

5.1 Experimental settings

We evaluate the proposed pooling algorithm in the context of deep graph convolution networks. The hierarchical graph convolution networks used in the experiments following the general framework of GraphSAGE [7] consist of a stack of convolution modules, iPool layers, a readout module as well as a prediction module, as presented in Fig. 2.

The convolution module consists of three graph convolution layers. For the graph convolution layer, we follow the design of common architectures and choose the general message propagation and aggregation formation. Specifically, we adopt the following convolution module:

$$X_{l+1} = \mathrm{ReLU}(\phi(A_l X_l W_l)), \tag{18}$$

where ReLU denotes the rectified linear unit activation function and $\phi(\cdot)$ indicates an l_2 normalization function to stablise and accelerate the training process. In more details, this convolution operation

[3]Better performance will be achieved with other configurations, and detailed information is presented in Ablation studies.

Table 3: Graph classification accuracies with 10-fold cross-validation with the small networks. Results of baseline methods with '†' are cited from original publications.

Method	NCI1	NCI109	MUTAG	IMDB-B	IMDB-M
PSCN[†]	76.34±1.68	-	88.95±4.37	71.00±2.29	45.23±2.84
GRAPHSAGE	78.76±1.73	77.92±2.47	86.78±8.88	72.10±2.66	49.20±3.94
ECC[†]	76.82	75.03	76.11	-	-
SORTPOLL[†]	74.44	-	85.83	70.03	47.83
k-GNNs[†]	76.2	-	86.1	**74.2**	49.5
SET2SET	80.80±1.72	79.09±1.96	86.78±7.33	71.00±7.54	49.73±4.19
DIFFPOOL	80.46±1.22	79.04±1.25	88.87±6.75	73.00±3.22	49.60±3.87
Proposed global	80.46±1.66	78.80±2.62	89.42±5.68	72.90±3.08	50.73±3.68
Local ($k = 1$)	81.41±1.53	80.01±2.32	87.84±6.12	73.10±2.98	50.53±2.71
Local ($k = 2$)	**81.58±1.46**	**80.03±2.05**	**90.42±4.68**	**73.30±2.72**	**51.27±3.44**

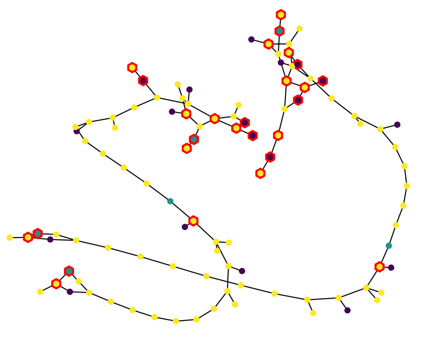
consists of three steps: (1) Neighborhood information aggregation: through $A_l X_l$, each node propagates information in neighborhood and make a weighted summation of information from neighbors utilizing the weights of shift matrix $(A_l)_{ij}$. It is analogous to weighted average filtering, a common low-pass filtering, and thereby enables the graph convolution networks to be robust to noisy signals to some degree. (2) Affine transformation: if we unfold the formation $\tilde{X}_l = [\tilde{\mathbf{x}}_{l,1}, \tilde{\mathbf{x}}_{l,2}, \cdots, \tilde{\mathbf{x}}_{l,n}]^T = A_l X_l = A_l[\mathbf{x}_{l,1}, \mathbf{x}_{l,2}, \cdots, \mathbf{x}_{l,n}]^T$, we will see that $\tilde{X}_l W_l$ imposes an affine transformation on each node after aggregation, because of $\tilde{X}_l W_l = ({W_l}^T {\tilde{X}_l}^T)^T = ({W_l}^T [\tilde{\mathbf{x}}_{l,1}, \tilde{\mathbf{x}}_{l,2}, \cdots, \tilde{\mathbf{x}}_{l,n}])^T$. (3) Non-linear mapping: the ReLU non-linear function is chosen to further perform point-wise non-linear transformation. Feature maps of convolution layers within a convolution module are further concatenated to form the outputs of the convolution module. The pooling module then follows the convolution module to coarsen graphs in accordance with the iPool operator, as introduced in Section 3. In addition to convolution and pooling modules, a readout module is utilized to attain graph embeddings of different coarsened versions and these graph embeddings are concatenated to produce the final graph representation:

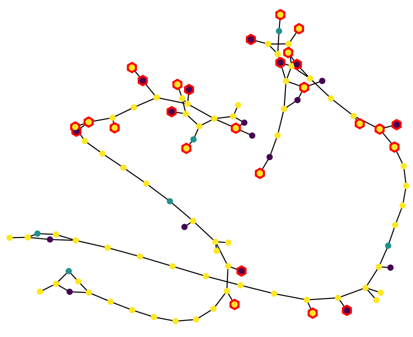
$$h_{\mathcal{G}} = \text{Concat}(\eta(X_l) | l = 1, 3, \ldots, K), \tag{19}$$

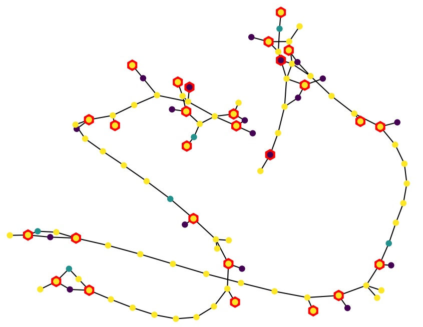
where $\eta(\cdot)$ indicates an element-wise operator to aggregate information of all nodes along each dimension of features. Specifically, an element-wise mean operator is used on biological datasets and an element-wise sum operator is adopted on social network datasets. A prediction module is finally added to the architecture for graph classification. It consists of two fully connected layers and a soft-max layer to make the prediction of the graph category based on the graph representation $h_{\mathcal{G}}$.

We implement the proposed model in Pytorch [16]. We conduct experiments to classify graphs on ten public benchmark graph datasets, including biological datasets (MUTAG, ENZYMES, DD, PROTEINS, NCI1, NCI109) and social network datasets (IMDB-BINARY, IMDB-MULTI, COLLAB, REDDIT-MULTI-12K) [4]. Statistics and properties of the datasets are presented in Table 2. Node categorical features are adopted in biological datasets, while constant features are utilized in social datasets. Following prior methods [28, 27], we perform 10-fold cross-validation on all of the datasets and report the best average accuracy. In the experiments, we employ two kinds of networks, a small one with the number of hidden neurons as 30 and a large one with 64 hidden neurons at each convolution layer. Also, one iPool layer is utilized for most datasets except that two iPool layers are employed for D&D and REDDIT-M-12K because of the large number of nodes per graph. The pooling ratio (ρ) is set as 0.1 for middle datasets, including ENZYMES, PROTEINS, and COLLAB, and 0.25 for the others. These networks are optimized by mini-batch gradient descent algorithm (batch size =20) with the Adam optimizer. The following hyper-parameters are tuned for each dataset: (1) learning rate $\in \{0.01, 0.001, 0.0001\}$; (2) the drop-out ratio $\in \{0, 0.5\}$ and weight decay $\in \{0, 3e-5, 1e-4\}$. Detailed information about hyper-parameters is presented in Appendix (Table A1). Finally, we compare the proposed method with a collection of state-of-the-art kernel-based solutions as well as GNN-based graph representation learning methods. For graph kernel methods, we compare with the Weisfeiler-Lehman subtree kernel (WL) [20], the Weisfeiler-Lehman optimal assignment kernel (WL-OA) [11], graphlet kernel [19], and shortest-path kernel

[4]Datasets could be downloaded from https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets


(a) Global pooling ($k = 2$).


(b) Local pooling ($k = 1$).


(c) Local pooling ($k = 2$).

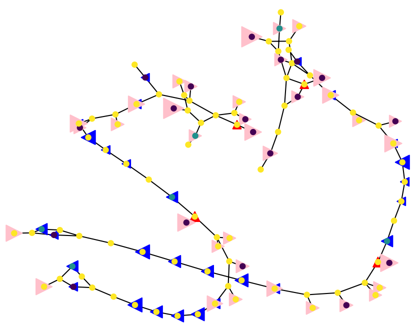

(d) DIFFPOOL.

Figure 3: Illustration of different pooling schemes with the color of the node indicating the node category on a graph of the NCI1 dataset. Results of global and local strategies of iPool are respectively shown in (a)∼(c), where nodes with red hexagons are ones transferred to the coarsened graph. The result of the DIFFPOOL is shown in (d), where only first three nodes in the coarsened graph are presented and only the nodes with assign ratio larger than 0.1 be absorbed to the new node are illustrated for clarity. The triangular nodes with the same color are softly assigned to the same node in the coarsened graph with the size of the triangular being proportional to the assign ratio.

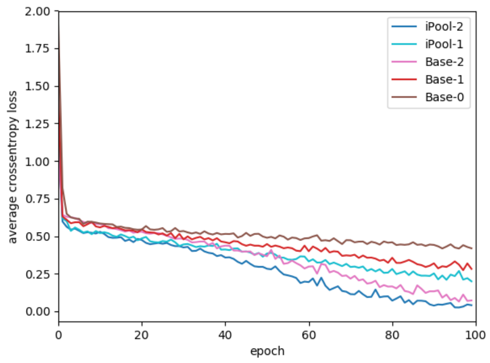
Table 4: Running time (s/epoch).

Method	NCI1	NCI109	MUTAG	IMDB-B	IMDB-M	Multiple
SET2SET	24.86 (7.1)	26.08 (7.1)	0.45 (2.0)	7.71 (7.1)	8.25 (7.1)	×7.1
DIFFPOOL	4.03 (1.1)	4.05 (1.1)	0.25 (1.1)	1.25 (1.1)	1.30 (1.1)	×1.1
Proposed	3.51	3.66	0.22	1.09	1.16	1

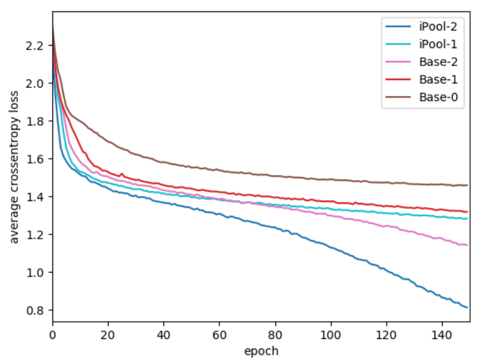
(SP) [1]. On the other hand, a series of graph neural network variants and different graph pooling schemes designed for deep graph neural networks are taken into consideration, including PSCN [15], GraphSAGE [7], ECC [22], k-GNNs [14], global pooling scheme with LSTMs (SET2SET) [24], sort pooling (SORTPOOL) scheme [29], and DIFFPOOL generating coarsening matrix with an extra GNN [28]. Results of the baseline models are cited from the original works except the results of GraphSAGE, SET2SET, and DIFFPOOL methods which are obtained using their public code with the same architecture used for iPool.

5.2 Experimental results and analysis

Table 1 and Table 3 respectively illustrate performance of two kinds of networks, a small one (30 hidden neurons) and a large one (64 hidden neurons). The proposed iPool strategies outperform other GNNs based methods on 9 of 10 datasets. Compared with the state-of-the-art kernel based methods, the iPool methods also achieve competitive performance. The limited number of training



(a) D&D



(b) Reddit-Multi-12k

Figure 4: Training convergence on the D&D and REDDIT-MULTI-12K datasets.

Table 5: Results of ablation experiments.

Dataset	Base-0	Base-1	Base-2	iPool-1	iPool-2
D&D	78.52±3.89	76.37±3.13	75.94±4.47	79.28±2.29	79.45±2.78
RED-M-12K	46.85±1.13	48.44±1.23	48.58±1.65	48.71±0.95	47.64±1.56

samples per category would cause the neural network overfitting, which probably explains the inferior performance of GNNs based methods on the ENZYMES dataset. With regards to the COLLAB dataset, the WL kernel and WL-OA kernel outperforms all GNN variants, which probably results from the large number of node degrees making it difficult for graph neural networks to effectively aggregate neighborhood information with convolution layers.

We also note that the local pooling strategy performs better than the global one on all of the datasets. The local strategy prefers nodes from different neighborhoods and thereby the coarsened graphs could better preserve the structure of the original graphs, as demonstrated in Fig. 3. In addition, the number of hops (i.e., k) utilized in the prediction function (Eq. (5)) has an important impact on the classification performance, and has a close relationship with the average node degree. Specifically, the sparser the connections between nodes of a graph, the larger the necessary number of hops (k) to achieve the best performance. This is consistent with intuition, since the prediction function with large k in graphs with dense connections, will aggregate global information of the whole graph rather than local information of each neighborhood.

Without the graph pooling layer, the GraphSAGE and SET2SET methods extract representations of graphs at the scale of the original signal. In contrast, DIFFPOOL and iPool obtain multiscale graph representations and achieve better results on most datasets. In addition, the iPool scheme further outperforms DIFFPOOL on all of the datasets. It probably better preserves the structure of the original graphs and the localization property of graph signals, as demonstrated in Fig. 3. In addition, the running speed of iPool is about 7 times faster than SET2SET and 1.1 times faster than DIFFPOOL, as illustrated in Table 4.

Ablation studies. We further explore the effectiveness of the iPool operator to deal with multiscale representations of graph data with a series of ablation experiments on the D&D and REDDIT-MULTI-12K datasets, the largest datasets of bioinformatics and social networks respectively in terms of average number of nodes per graph. Specifically, we conduct experiments on 3 network architectures, as presented in Fig. 2, to deal with single-scale (Base-0, 1 convolution module), double-scale (iPool-1, 2 convolution modules and 1 pooling module), and triple-scale (iPool-2, 3 convolution modules and 2 pooling modules) graph representations. To eliminate the impact of the depth of neural networks, we further extract single-scale graph representations with the same architecture as their multiscale counterparts except for the pooling layers, i.e., Base-1 corresponding to iPool-1 and Base-2 matching iPool-2. We utilize the same procedure as the one for Table. 1 and report 10-fold cross-validation results in terms of the mean and standard variation of classification accuracy in Table 5.

According to Fig. 4 and Table 5, we have the following findings. (1) The iPool operation could accelerate the convergence of representation learning in the training phase. Concretely, the iPool-2 and iPool-1 models converge faster than their single-scale counterparts Base-2 and Base-1, and are far faster than the single-scale shallow model Base-0 on both datasets. (2) The iPool operation could improve the classification performance of models due to its hierarchical representation. Specifically, models with iPool achieve the best performance on both datasets. Compared to iPool-1, the inferior performance of iPool-2 on the REDDIT-MULTI-12K dataset probably results from the overfitting of the model, given the simple constant node signals and the scaled-down graphs through two pooling operators.

6 Conclusion

We have proposed in this paper a low complexity and adaptive graph pooling operator for GNNs, which improves their capability of distilling hierarchical representations of graphs and network data. The new operator has interesting properties in practice, in that it is mostly based on local computations, and it leads to invariance properties under graph isomorphism. The proposed iPool solution further permits to achieve state-of-the-art performance on several graph classification datasets. An interesting future direction is to explore other neighborhood prediction functions to make the neighbor information gain in line with more general conditional distribution of nodes. It is also worthwhile to utilize the proposed pooling operation with other convolution schemes, such as GINs.

References

[1] K. M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[2] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *International Conference for Learning Representations*, 2014.

[3] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

[4] C. Chevalier and I. Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *International Conference on Learning and Intelligent Optimization*, pages 191–205. Springer, 2009.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[6] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[7] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[8] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[9] R. Khasanova and P. Frossard. Graph-based isometry invariant representation learning. *arXiv preprint arXiv:1703.00356*, 2017.

[10] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference for Learning Representations*, 2017.

[11] N. M. Kriege, P.-L. Giscard, and R. Wilson. On valid optimal assignment kernels and applications to graph classification. In *Advances in Neural Information Processing Systems*, pages 1623–1631, 2016.

[12] D. Kushnir, M. Galun, and A. Brandt. Fast multiscale clustering and manifold identification. *Pattern Recognition*, 39(10):1876–1891, 2006.

[13] D. Kushnir, M. Galun, and A. Brandt. Efficient multilevel eigensolvers with applications to data analysis tasks. *IEEE transactions on pattern analysis and machine intelligence*, 32(8): 1377–1391, 2009.

[14] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of AAAI Conference on Artificial Inteligence*, 2019.

[15] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 2009.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[19] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495, 2009.

[20] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[21] D. I. Shuman, M. J. Faraji, and P. Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2016.

[22] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

[24] O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets. *International Conference on Learning Representations*, 2015.

[25] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[26] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

[28] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4805–4815, 2018.

[29] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI Conference on Artificial Inteligence*, 2018.

Appendix

A1 Proof of Proposition 1

Proof.

$$\begin{aligned}
&H(\mathbf{x}_{l,i}|\{\mathbf{x}_{l,j}\}_{N(v_{l,i})})\\
=&\mathbb{E}[-\log p(\mathbf{x}_{l,i}|\{\mathbf{x}_{l,j}\}_{N(v_{l,i})})]\\
=&\mathbb{E}[-\log \Pi_{z=1}^{d} p(x_{l,i,z}|\{\mathbf{x}_{l,j}\}_{N(v_{l,i})})] && \text{(A1.1)}\\
\approx&\frac{1}{m}\sum_{k=1}^{m} -\log \Pi_{z=1}^{d} p(x_{l,i,z}^{(k)}|\{\mathbf{x}_{l,j}^{(k)}\}_{N(v_{l,i})}) && \text{(A1.2)}\\
=&\frac{1}{mb_{l,i}}\sum_{k=1}^{m}\sum_{z=1}^{d}|x_{l,i,z}^{(k)}-\mu_{l,i,z}^{(k)}| + d\log 2b_{l,i}\\
=&\frac{1}{mb_{l,i}}\sum_{k=1}^{m}\parallel \mathbf{x}_{l,i}^{(k)}-\mu_{l,i}^{(k)}\parallel_1 + d\log 2b_{l,i}\\
\approx&\frac{1}{b_{l,i}}\parallel \mathbf{x}_{l,i}-\mu_{l,i}\parallel_1 + d\log 2b_{l,i} && \text{(A1.3)}\\
=&\frac{1}{b_{l,i}}\parallel \mathbf{x}_{l,i}-f(v_{l,i})\parallel_1 + d\log 2b_{l,i}\\
=&\frac{1}{b_{l,i}}\gamma(v_{l,i}) + d\log 2b_{l,i}, && \text{(A1.4)}
\end{aligned}$$

where Eq. (A1.2) utilizes an empirical probability to approximate Eq. (A1.1) and Eq. (A1.3) further approximates it with one sample estimation. □

A2 Proof of Proposition 2

Proof. Since $\mathcal{G}_l \simeq \mathcal{G}_l^{'}$, there exists an edge-preserving bijection:

$$t : \mathcal{V}_l \longrightarrow \mathcal{V}_l^{'}. \tag{A2}$$

For $\forall v_{l,i} \in \mathcal{V}_l$, there is a $v_{l,m}^{'} = t(v_{l,i}) \in \mathcal{V}_l^{'}$ and their neighborhood information gains are respectively:

$$\gamma(v_{l,i}) = \parallel x(v_{l,i}) - \frac{1}{k}\sum_{h=1}^{k}\sum_{v_{l,j}\in N^h(v_{l,i})}(\bar{P_l^h})_{ij}\times x(v_{l,j})\parallel_1 . \tag{A3}$$

$$\gamma(v_{l,m}^{'}) = \parallel x(v_{l,m}^{'}) - \frac{1}{k}\sum_{h=1}^{k}\sum_{v_{l,n}^{'}\in N^h(v_{l,m}^{'})}(\bar{P_l^{'h}})_{mn}\times x(v_{l,n}^{'})\parallel_1 . \tag{A4}$$

Since $t(\cdot)$ is edge-preserving, $N^h(v_{l,i}) = N^h(v_{l,m}^{'})$ and any edge $(v_{l,i}, v_{l,j}) \in \mathcal{E}_l$ shares the same weights with its counterpart $(v_{l,m}^{'}, v_{l,n}^{'}) \in \mathcal{E}_l^{'}$:

$$(A_l)_{ij} = (A_l^{'})_{mn}, \quad (A_l{}^h)_{ij} = (A_l^{'h})_{mn}, \quad (\bar{P_l^h})_{ij} = (\bar{P_l^{'h}})_{mn}. \tag{A5}$$

Therefore,

$$\gamma(v_{l,i}) = \gamma(v_{l,m}^{'}), \quad \forall v_{l,i} \in \mathcal{V}_l, \tag{A6}$$

and it holds true also for the normalized neighborhood information gain. Note that the ranking function takes only the value of (normalized) neighborhood information gain of nodes under consideration, and that $v_{l,i}$ has the same ranking as $t(v_{l,i})$. Then

$$\mathcal{V}_{l+1} = \mathcal{V}_{l+1}^{'}, \quad \mathcal{E}_{l+1} = \mathcal{E}_{l+1}^{'}, \quad X_{l+1} = X_{l+1}^{'}. \tag{A7}$$

Thus, the iPool operation is invariant to isomorphism. □

A3 More information about experiments

Table A1: Hyper-paramters used in the experiments. The 'lr' denotes learning rate and the 'k-global' indicates the number of hops used in the prediction function under the global iPool strategy.

Dataset	lr	s	k-global	drop-out	weight decay
ENZY	1e-3	2	1	0	1e-4
D&D	1e-4	2	1	0.5	0
RED-M-12K	1e-3	2	2	0	0
COLL	1e-3	1	1	0	0
PROT	1e-2	2	1	0	1e-4
NCI1	1e-3	2	2	0	1e-4
NCI109	1e-3	2	2	0.5	0
MUTAG	1e-2	2	2	0.5	3e-5
IMDB-B	1e-3	1	2	0	1e-4
IMDB-M	1e-3	1	2	0	0