iPool—Information-Based Pooling in Hierarchical Graph Neural Networks

Xing Gao, Wenrui Dai, *Member, IEEE*, Chenglin Li, *Member, IEEE*, Hongkai Xiong, *Senior Member, IEEE*, and Pascal Frossard, *Fellow, IEEE*

***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 article, 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 on arbitrary topologies. The proposed strategy achieves superior or competitive performance in graph classification on a collection of public graph benchmark data sets and superpixel-induced image graph data sets.**

***Index Terms*— Graph classification, graph neural networks (GNNs), graph pooling, hierarchical representation.**

I. 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 [1]–[16] have been generalizing convolution neural networks to irregular structures, including graph and point cloud data. For instance, convolution operator is extended for graph signals in the spectral domain on the basis of the spectral graph theory, to build the so-called spectral graph convolution neural networks [2]–[5], [10]. On the other hand, several methods, such as [2], [9], [11], [15], generalize the convolution operator in the spatial domain to address the basis-dependent problem where a spectral graph neural network (GNN) trained on one graph structure fails to transfer properly to other graph structures. 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 GNNs to distill effective hierarchical representations.

Hierarchical representations of network data necessitate a careful design for all elements of the learning architecture. In tasks such as graph classification, a global representation is required in addition to local features in order to predict the label for an entire graph. For example, all amino acids (nodes in graph) and their bonds (edges in graph) are considered to accurately classify a protein (global graph). The pooling operator is an important component in the construction of such hierarchical architectures. In the example of image representation, the pooling operator downsamples data by flipping the predefined local receptive field and aggregating information in each receptive field, from left to right and top to bottom, to take advantage of the inherent spatial order in lattice structures. However, the design of pooling operation becomes challenging for the general case of networks with diverse and irregular topologies and no spatial order of the nodes. 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 is formulated as an NP-hard max-cut problem [17]. In addition, it is still a challenge to construct a coarsened graph with the pooled features. In other words, two major problems to generalize the pooling operation to graphs include: 1) selection and aggregation of information to characterize the signals residing on graph and 2) coarsening the structure of graphs in the higher levels of representations. Graph theory may provide several tools to generalize the pooling operator to networks with the help of graph clustering or graph coarsening algorithms [18]–[20]. However, these methods typically involve high computational complexity led

Manuscript received 3 January 2020; revised 6 August 2020 and 1 December 2020; accepted 12 March 2021. Date of publication 31 March 2021; date of current version 2 September 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61932022, Grant 61931023, Grant 61971285, Grant 61871267, Grant 61972256, Grant 61720106001, Grant 91838303, and Grant 61831018; and in part by the Program of Shanghai Science and Technology Innovation Project under Grant 20511100100. *(Corresponding author: Wenrui Dai.)*

Xing Gao, Chenglin Li, and Hongkai Xiong are with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: william-g@sjtu.edu.cn; lcl1985@sjtu.edu.cn; xionghongkai@sjtu.edu.cn).

Wenrui Dai is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: daiwenrui@sjtu.edu.cn).

Pascal Frossard is with the Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland (e-mail: pascal.frossard@epfl.ch).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3067441.

Digital Object Identifier 10.1109/TNNLS.2021.3067441

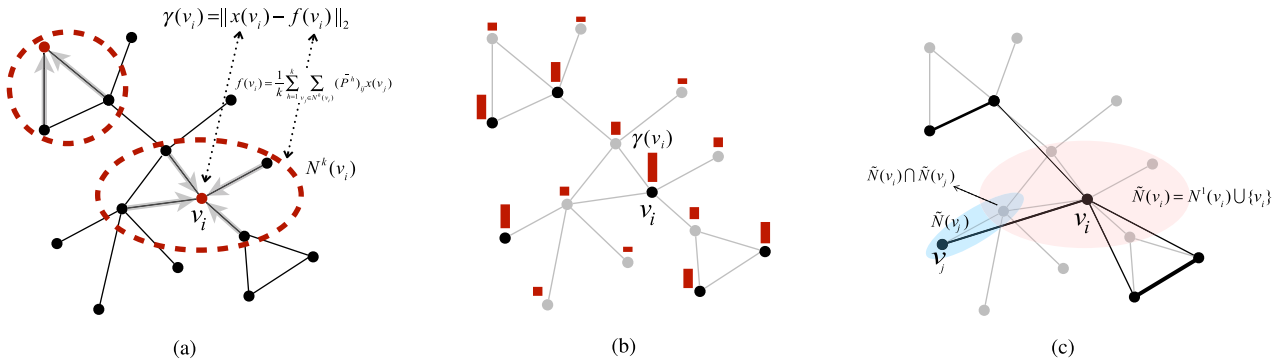

Fig. 1. Illustration of iPool. (a) Proposed neighborhood information gain criterion $\gamma(\cdot)$ and the prediction function $f(\cdot)$. (b) and (c) Node selection in accordance with neighborhood information gain criterion to perform graph downsampling and the subsequent coarsened graph construction from selected nodes, respectively. (c) Two selected nodes are randomly chosen to demonstrate the edge weight computation, with the thickness of lines indicating the scale of edge weights. Specifically, $x(v_i)$ indicates the signal or feature on the node v_i, $N^h(v_i)$ represents the h-hop neighborhood of the node v_i, and P_l^h implies an off-diagonal version of the hth power of transmission matrix.

by iterative solutions. This unfortunately prevents GNNs from simultaneous processing of a batch of graph data on diverse topologies, which is commonly required in applications such as graph classification.

In this article, we propose a flexible pooling operator, called iPool, which can be easily interleaved with diverse graph convolution operators and can deal with data that come with arbitrary graphs. iPool is designed to generate a faithful representation of signals supported on the graph. Therefore, it first uses a criterion to evaluate the amount of information carried by each node given observations of its neighbors and then constructs coarsened graphs accordingly. The benefits of iPool are summarized as follows.

1) iPool is a parameter-free and stackable building block to be paired with diverse GNNs.
2) The proposed neighborhood information gain criterion is interpretable in terms of entropy and it guarantees a faithful representation of the original graph signal in the downsampled graph.
3) Greedy and local strategies are developed for iPool based on local computation to implement a computation-friendly pooling operator.

In more detail, we propose a pooling algorithm to form hierarchical graph representation with node selection guided by a neighborhood information gain criterion and structure-preserved construction of coarsened graphs, as shown in Fig. 1. In particular, the node signals are predicted from the neighbor signal values to determine the most informative nodes in the graph. A coarsened graph is constructed from the most informative nodes with the topology that maintains consistency with the original graph. Furthermore, we develop two kinds of pooling schemes, namely greedy and local strategies, which permit to trade off preservation of graph structure and information aggregation. We discuss the different aspects of the iPool operator in detail and demonstrate theoretically under mild assumptions that the iPool operator plays an equivalent role as coarsening graphs in terms of maximum entropy.

We resort to graph classification tasks to evaluate the proposed iPool algorithm. It is shown to outperform the state-of-the-art graph pooling methods on a collection of public benchmark data sets and superpixel-induced image graph data sets. The proposed pooling operator is also promising for extension to discriminative tasks like graph segmentation and applications into non-Euclidean data, such as point clouds.

The remainder of this article is organized as follows. Section II briefly overviews the related work on graph convolutional neural networks with a specific emphasis on pooling operations. Section III elaborates the proposed iPool algorithm, and Section IV discusses its properties of maximum entropy and invariance to graph isomorphism. Experimental results are presented in Section V to validate the proposed pooling strategies in graph classification tasks. Finally, Section VI draws the conclusions.

II. Related Work

We begin with a brief review of graph representation learning, especially traditional graph coarsening schemes and recent attempts on graph pooling, which are relevant to the main problem addressed in this work.

A. Graph Convolutional Networks (GCNs)

GNNs can be traced back to [21]–[23]. With the rise of CNNs, GCNs have been developed to extend CNNs to graphs. GCNs are commonly realized in spectral or spatial domains. Spectral GCNs define convolutions for graphs on the basis of spectral filtering of graph signals [2]–[5], [10]. For instance, smoothed spectral graph convolution networks [2] are proposed to achieve constant learning complexity, and localized spectral graph convolution networks [3], [5], [24] are designed to further attain linear computational complexity and localized filters as CNNs with diverse approximations to the convolution operation, such as the Chebyshev expansion. On the other hand, spatial GCNs develop the convolution operation in the spatial domain [2], [9], [11], [25] to address the basis-dependent problem that a spectral GNN trained on one graph structure cannot be properly transferred to other graph structures. Most spatial GCNs generalize the convolution operation by defining a "message aggregation" scheme to aggregate information in the neighborhood. In architectures for spatial GCNs, these convolution layers can be integrated with

different pooling operators for hierarchical representations of graph data.

B. Graph Coarsening and Graph Pooling

Hierarchical representations of graph data can be achieved with variants of graph coarsening algorithms and different forms of graph pooling operations. Due to high computational complexity for general graphs, graph coarsening is commonly achieved with approximate methods, including multiscale methods [19], [20], [26], kernel K-means [27], and spectral clustering algorithms [18]. Yet, computational complexity stays high, so that these methods are mostly suitable for processing fixed structure graphs off-line, but have clear limitations in providing an online building block to cope with graphs of various structures.

Pooling probably provides the most constructive alternative to develop multiscale representation in GNNs. To begin with, several global pooling operators are designed to produce an embedding of an arbitrary graph. For example, SET2SET [28] computes a global representation by aggregating node information through LSTMs. SortPooling [25] sorts the nodes based on the value of last feature map in a descending manner and preserves its first k nodes to represent the graph. Furthermore, several hierarchical pooling operators are proposed to consider coarsened graph construction for multiscale representation learning. The gPool algorithm [29] globally selects nodes according to their footprints obtained from projecting node features onto a trainable projection vector. SAGPool [30] resorts to self-attention of nodes to select nodes. They take the induced subgraph of the original graph or its second power graph as a coarsened graph. However, gPool and SAGPool fail to preserve diverse representative nodes in different neighborhoods, on the contrary to their counterparts that work on grid-like data [31], [32]. Specifically, the selected nodes of SAGPool concentrate in several specific neighborhoods since the attention scores of nodes that are calculated with a graph convolution operation are similar in the same neighborhood. The nodes preferred by gPool share a specific pattern, represented by the trainable projection vector. Thereby, information of the original graph cannot be faithfully represented by the coarsened graphs.

Inspired by clustering methods, DiffPool [33] softly assigns nodes with a learnable cluster assignment matrix. EigenPooling [34] employs spectral clustering to partition a graph into subgraphs. In each subgraph, information is aggregated by projecting the signals that it supports onto its first k eigenvectors. However, they are faced with high computational complexity or storage complexity. For example, DiffPool requires an additional branch of GNNs to learn projection matrices, and the number of parameters is related to the number of vertices of graph data, which limits its application to large-scale graphs. For EigenPooling, the eigendecomposition utilized in spectral clustering and computing eigenvectors of subgraphs also involves high computational complexity.

Contrary to existing methods, the proposed graph pooling operator is interpretable in the view of information theory. iPool leverages the proposed neighborhood information gain criterion to select informative nodes in each neighborhood, in order to represent the original graph faithfully. Furthermore, it is parameter-free and "plug and play" and thereby leads to a fast implementation at both training and testing phases.

III. iPool Algorithm

We present in this section the main elements of the proposed iPool algorithm. We first introduce the general framework of GNN architectures and define the role of the pooling algorithm. We later introduce a neighborhood information gain criterion that drives our iPool algorithm. Finally, we show how our iPool algorithm can be used to construct hierarchical representations with both greedy and local strategies.

A. Framework

In this article, we consider learning hierarchical representations for network data with the 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 article, 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 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of the set of vertices $\mathcal{V}$ and the set of edges $\mathcal{E}$. The adjacency matrix A is defined to represent the topology of the network, which has a nonzero value at position (i, j) (i.e., $A_{ij} \neq 0$) only when there is an edge in $\mathcal{E}$ that connects vertices v_i and v_j in $\mathcal{V}$. Here, A is composed of unitary values or actual edge weights corresponding to unweighted or weighted graphs. Let us denote D the diagonal degree matrix with its element $D_{ii} = \sum_j A_{ij}$ and $P = D^{-1}A$ the transmission matrix of $\mathcal{G}$ representing the transmission probability of each pair of nodes. Furthermore, each vertex of the graph $v_i \in \mathcal{V}$ might further be attributed a signal value or feature, which is denoted by $\mathbf{x}_i = x(v_i)$ and $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^{\mathrm{T}}$ for the whole graph. Moreover, for hierarchical representation of graph data, we use the subscript l to indicate features or parameters belonging to the lth layer of the neural networks. For example, for the graph $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ with $n_l = |\mathcal{V}_l|$ vertices in the lth layer of the neural network, the ith vertex of the graph is defined as $v_{l,i} \in \mathcal{V}_l$, and $\mathbf{x}_{l,i} = x(v_{l,i}) \in \mathbb{R}^{d_l}$ is the d_l-dimensional signal or feature residing on the node $v_{l,i}$.

Most of the existing GCNs stack graph convolution layers to learn a representation or embedding of graph data. The graph convolution layers are usually designed to follow a neighborhood message aggregation scheme:

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

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 $\xi(\cdot)$ and $\omega(\cdot)$, respectively, denote a data aggregation function and a nonlinear activation

function, and the parameters W_l are learned during the training of the neural network. The aggregation function $\xi(\cdot)$ varies in different GCN architectures and is usually selected as (weighted) summation [33], mean [11], [25], or multilayer perceptron (MLP) [15].

A key element of these architectures is graph pooling or coarsening operator, which aims at selecting a subset of data and obtains 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 predefine the sampling structure or the local receptive fields on graphs, as it can be done in image representation learning [17]. The graph pooling operator has thus to be defined adaptively, yet in a generic way so that it can accommodate different network structures.

For $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$, the graph pooling operator takes its adjacency matrix $A_l \in \mathbb{R}^{n_l \times n_l}$ as well as graph signals $X_l \in \mathbb{R}^{n_l \times d_l}$ as inputs and produces $A_{l+1} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}$ and $X_{l+1} \in \mathbb{R}^{n_{l+1} \times d_{l+1}}$ of the coarsened graph $\mathcal{G}_{l+1}$

$$A_{l+1},\ X_{l+1} = \phi(A_l, X_l) \tag{2}$$

with $\phi(\cdot)$ indicating a specific function corresponding to different pooling strategies.

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

B. 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, 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 Euclidean distance between the observed signal $\mathbf{x}_{l,i} = x(v_{l,i})$ and the one predicted from observations at the neighbor nodes.[1] We choose the Euclidean distance as it represents a common similarity measure that is especially convenient for high-dimensional vectors that might be present in some graph data sets. Specifically, with a prediction function $f(\cdot)$ using information from the neighborhood, the neighborhood information gain of each node could be formulated as

$$\gamma\left(v_{l,i}\right) = \left\|x\left(v_{l,i}\right) - f\left(v_{l,i}\right)\right\|_2. \tag{3}$$

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 hth power of transmission matrix P_l^h is an effective measure of the level of connection or dependence between any pair of nodes reachable with h hops, we adopt the elements of $P_l^h (h = 1, 2, \ldots, k)$ as the weights in our prediction function $f(\cdot)$ in order to give more confidence to nodes that have stronger connections. We, however, modify P_l^h into an off-diagonal transition matrix $\bar{P}_l^h$ so that the signal supported on itself is not considered in the signal prediction of each node. Finally, the prediction function can be formulated as

$$f\left(v_{l,i}\right) = \frac{1}{k}\sum_{h=1}^{k}\sum_{v_{l,j}\in N^h\left(v_{l,i}\right)}\left(\bar{P}_l^h\right)_{ij} x\left(v_{l,j}\right) \tag{4}$$

with

$$\bar{P}_l^h = \left(\bar{D}_l^h\right)^{-1}\bar{A}_l^h, \quad \bar{A}_l^h = A_l^h - \mathrm{diag}\left(A_l{}^h\right) \tag{5}$$

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 [diag$(\cdot)$ returns a diagonal matrix with the diagonal elements of the input matrix], and $\bar{D}_l^h$ is the corresponding degree matrix. In this way, (3) could be further formulated as

$$\gamma\left(v_{l,i}\right) = \left\|x\left(v_{l,i}\right) - \frac{1}{k}\sum_{h=1}^{k}\sum_{v_{l,j}\in N^h\left(v_{l,i}\right)}\left(\bar{P}_l^h\right)_{ij} \times x\left(v_{l,j}\right)\right\|_2 \tag{6}$$

and for the whole graph

$$\Gamma(\mathcal{G}_l) = \left|\left(I - \frac{1}{k}\sum_{h=1}^{k}\bar{P}_l^h\right)X_l\right|_2 \tag{7}$$

where I denotes an identity matrix and $|\cdot|_2$ indicates the l_2 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 is more informative and should be preserved in the pooling operation. In accordance with the proposed neighborhood information gain criterion, iPool can then effectively downsample signals and is robust to noisy signals, given that the previous convolution layers based on message aggregation [see (1)] perform a series of

[1]For graph without signals, graph structure is taken as signals, such as node degree. For isolated nodes, we assign their neighborhood information gain criteria as the lowest score, i.e., -1, given that their information can be extracted by previous graph convolution layers.

equivalent low-pass filtering of the graph information. Note that the definition of the information criterion is local, which is very important toward low-complexity and possible distributed implementation.

C. iPool

With the neighborhood information gain criterion defined above, the pooling operator identifies the nodes to be preserved that have the higher information gain and permits to adaptively downsample the graph while preserving the local characteristics of graph signals. We describe below greedy and local versions of the pooling algorithm and then show how we construct coarsened graphs after pooling, in order to build a multiscale representation of network data.

1) Greedy iPool Strategy: On the basis of the neighborhood information gain, nodes are assigned different priorities to construct coarse graph globally. We adopt the downsampling method mentioned in [25], [29] that a part of nodes in the original graph are selected. In order to approximate the information of the graph, the pooling should preserve the nodes that cannot 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 reordered based on the value of their neighborhood information gain. The greedy strategy then generalizes the k-max pooling and greedily selects the top $n_{l+1} = \rho \times |\mathcal{V}_l|$ nodes that have the highest information gain

$$\mathbf{idx} = \text{rank}(\Gamma(\mathcal{G}_l), n_{l+1}) \tag{8}$$

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

2) Local iPool Strategy: Pooling can also be implemented locally, and nodes can be selected within each receptive field, similar to what is done for images. However, the receptive field is hardly predefined for graph data due to their diverse topology. Considering the inherent neighborhood of each node in graphs, we take it as the equivalent receptive field of the node. The local iPool strategy leads to a relatively even distribution of the selected nodes over the original graph based on a normalized information gain in each neighborhood. Specifically, the neighborhood information gain of each node is first normalized by the average neighborhood information gain of its neighbors

$$\overline{\gamma}\left(v_{l,i}\right) = \frac{\gamma\left(v_{l,i}\right)}{\sum_{v_{l,j}\in N\left(v_{l,i}\right)}\left(\bar{P}_l\right)_{ij}\gamma\left(v_{l,j}\right)}. \tag{9}$$

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

$$\mathbf{idx} = \text{rank}\left(\overline{\Gamma}(\mathcal{G}_l), n_{l+1}\right). \tag{10}$$

In this way, we ignore the information of nodes that can be well predicted in each neighborhood and preserve their relatively informative neighbors in the coarsened graph. This solution takes into consideration the amount of information of nodes in their respective neighborhoods and avoids a computationally intensive iterative selection process.

3) 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.

Instead of directly taking the induced graph formed by selected nodes, such as SAGPool [30], we exploit the relationship between the neighborhood of selected nodes to build their connections in order to keep consistency with the structure of original graphs. An induced graph directly deletes edges connected to nonselected nodes and loses some important structural information. For instance, among direct neighbors of an arbitrary node $v_{l,i}$, some nodes together with $v_{l,i}$ belong to a well-connected cluster with many common neighbors, while other nodes belong to other clusters with few links with $v_{l,i}$. Without connections to nonselected nodes, such structural information cannot be effectively characterized. To address this problem, for a pair of selected nodes, their connection in the coarsened graph is built based on their edge weight in the original graph and the overlap of their neighborhoods

$$\begin{aligned}(A_{l+1})_{ij} = {} & \lambda(A_l + I)_{\mathbf{idx}[i],\mathbf{idx}[j]} \\ & + (1-\lambda)\frac{2|\tilde{N}\left(v_{l,\mathbf{idx}[i]}\right) \cap \tilde{N}\left(v_{l,\mathbf{idx}[j]}\right)|}{|\tilde{N}\left(v_{l,\mathbf{idx}[i]}\right)| + |\tilde{N}\left(v_{l,\mathbf{idx}[j]}\right)|}\end{aligned} \tag{11}$$

where $\mathbf{idx}[i]$ indicates the index of the ith selected nodes in $\mathcal{G}_l$, $\tilde{N}(v_{l,i})$ represents the expanded one-hop neighborhood of $v_{l,i}$ composed of $v_{l,i}$ and its direct neighbors, and $|\cdot|$ is the cardinality of a set. The hyperparameter λ is introduced to adjust the contribution of two parts. In this way, two nodes tend to have a large edge weight in the coarsened graph, if their neighborhoods have dense interconnections in the original graph.

For the graph signal, it takes the signal supported on selected nodes in the original graph

$$X_{l+1} = X_l[\mathbf{idx}, :]. \tag{12}$$

Like the max-pooling operator for grid-like data, the gradient of the output of iPool X_{l+1} directly propagates back to its previous layer for the selected nodes and zeros for other nodes. Mathematically, with δ indicating the gradient, we have

$$(\delta X_l)_i = \begin{cases} (\delta X_{l+1})_m, & i \in \mathbf{idx} \text{ and } i = \mathbf{idx}[m] \\ 0, & \text{others.} \end{cases} \tag{13}$$

Note that the iPool operator takes as inputs the adjacency matrix and node features of the graphs and produces the adjacency matrix and node features of coarsened graphs in the forward propagation, as it is the common case for hierarchical graph models. Then, it back-propagates gradients according to (13) during the training phase of the neural network to enable the end-to-end learning of the networks. It is generic enough to be integrated in diverse architectures.

IV. Properties of iPool

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

Ideally, in order to maximally preserve the information of signals residing on the original graph, the n_{l+1} nodes from

$\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$ with $|\mathcal{V}_l| = n_l$ ($n_l > n_{l+1}$) should be selected by optimizing the following objective:

$$\max_{\pi \in \Pi} H\left(\mathbf{x}_{l,\pi_1}, \mathbf{x}_{l,\pi_2}, \ldots, \mathbf{x}_{l,\pi_{n_{l+1}}}\right) \tag{14}$$

where π represents a specific permutation of nodes in $\mathcal{V}_l$ and π_i indicates the ith node in the permutation.

However, there are several challenges to directly compute the optimal solution. First, due to the diversity of signals on different graphs, it is intractable to model the joint distribution of node signals of arbitrary graphs with a universe distribution and thereby is difficult to design a general pooling strategy. Furthermore, to obtain the optimal solution, one has to solve an NP-hard problem. In contrast with the intractable joint distribution, it is much easier to model the conditional distribution of a node signal given signals supported on its neighbor nodes, especially for the locality and smoothness of the most graph signals, with a bell-shaped function centered on the mean of its neighbor signals such as Gaussian distributions and Laplacian distributions. Consequently, we discuss the two strategies of iPool as two kinds of constructive approximate solutions to this optimization problem.

We first introduce neighborhood conditional entropy of node signals for graph data and further discuss its relationship with the proposed neighborhood information gain.

A. Neighborhood Conditional Entropy

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 a k-hop neighborhood

$$H\left(\mathbf{x}_{l,i}|\left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,i})}\right) = H\left(\mathbf{x}_{l,i}\right) - I\left(\mathbf{x}_{l,i}; \left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,i})}\right) \tag{15}$$

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

Proposition 1: Let us assume that the components of neighborhood conditional distribution of each node are independent and that each component satisfies a Gaussian distribution $p(x_{l,i,z}|\{\mathbf{x}_{l,j}\}_{N^k(v_{l,i})}) \sim \mathcal{N}(\mu_{l,i,z}, \sigma_{l,i}^2)$ or a Laplace distribution $p(x_{l,i,z}|\{\mathbf{x}_{l,j}\}_{N^k(v_{l,i})}) \sim \text{Laplace}(\mu_{l,i,z}, b_{l,i})$, with mean $\mu_{l,i} = [\mu_{l,i,1}, \mu_{l,i,2}, \ldots, \mu_{l,i,d_l}] = f(v_{l,i})$ and variation $\sigma_{l,i}^2$ (or scale parameter $b_{l,i}$), and the neighborhood information gain $\gamma(v_{l,i})$ of each node is an approximate empirical estimation of its neighborhood conditional entropy.

Proof:

1) If the conditional distribution is a Gaussian distribution, the neighborhood conditional entropy is

$$\begin{aligned} &H\left(\mathbf{x}_{l,i}|\left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,i})}\right) \\ &\quad = \mathbb{E}\left[-\log p\left(\mathbf{x}_{l,i}|\left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,i})}\right)\right] \\ &\quad = \mathbb{E}\left[-\log \Pi_{z=1}^{d_l} p\left(x_{l,i,z}|\left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,i})}\right)\right] \\ &\quad = d_l\left(\ln\left(\sqrt{2\pi}\sigma_{l,i}\right) + \frac{1}{2}\right). \end{aligned} \tag{16}$$

Since $x_{l,i,z}|\{\mathbf{x}_{l,j}\}_{N^k(v_{l,i})} \sim \mathcal{N}(\mu_{l,i,z}, \sigma_{l,i}^2)$, we have $x_{l,i,z} - f(v_{l,i,z})|\{\mathbf{x}_{l,j}\}_{N^k(v_{l,i})} \sim \mathcal{N}(0, \sigma_{l,i}^2)$, for $z = 1, 2, 3, \ldots, d_l$. Thereby, $\sigma_{l,i}$ can be estimated as

$$\hat{\sigma}_{l,i} = \sqrt{\frac{1}{d_l}\sum_{z=1}^{d_l}\left(x_{l,i,z} - f\left(v_{l,i,z}\right)\right)^2} = \frac{1}{\sqrt{d_l}}\gamma\left(v_{l,i}\right). \tag{17}$$

Correspondingly, the empirical estimation of the neighborhood conditional entropy is $d_l(\ln((2\pi/d_l)^{1/2} \cdot \gamma(v_{l,i})) + 1/2)$.

2) Similarly, for Laplacian distribution as the conditional distribution

$$\begin{aligned} H\left(\mathbf{x}_{l,i}|\left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,i})}\right) &= d_l\left(\ln\left(2b_{l,i}\right) + 1\right) \\ &= d_l\left(\ln\left(\sqrt{2}\sigma_{l,i}\right) + 1\right) \end{aligned} \tag{18}$$

with the relationship between standard variation and scale parameter $\sigma_{l,i} = \sqrt{2}b_{l,i}$. $\sigma_{l,i}$ can be estimated in the same way as the Gaussian distribution, with $\hat{\sigma}_{l,i} = (1)/(d_l)^{1/2}\gamma(v_{l,i})$, and thereby, the neighborhood conditional entropy $d_l(\ln((2/d_l)^{1/2} \cdot \gamma(v_{l,i})) + 1)$.

For both cases, the empirical estimation of the neighborhood conditional entropy can be uniformly represented as $d_l \ln \gamma(v_{l,i}) + c_l$, with c_l denoting a constant. ■

Proposition 1 implies that, under the moderate assumption that the components of the conditional distribution are independent and from the same type (not strictly identical) distribution, Gaussian distributions or Laplacian distributions, the neighborhood information gain criterion used in iPool is actually an estimation of the neighborhood conditional entropy to preserve the graph information during pooling. This observation motivates the choice of the formulation of the neighborhood information gain criterion.

B. Maximum Entropy

With the relationship between neighborhood information gain and neighborhood conditional entropy, we further demonstrate that the greedy strategy and local strategy of iPool provide two constructive approximations to coarsen graphs in accordance with the maximum entropy strategy.

Based on the chain rule, the objective function [see (14)] of the ideal maximum entropy strategy can be unfolded as

$$\begin{aligned} &H\left(\mathbf{x}_{l,\pi_1}, \mathbf{x}_{l,\pi_2}, \ldots, \mathbf{x}_{l,\pi_{n_{l+1}}}\right) \\ &\quad = H\left(\mathbf{x}_{l,\pi_1}, \mathbf{x}_{l,\pi_2}, \ldots, \mathbf{x}_{l,\pi_{n_{l+1}}}, \ldots, \mathbf{x}_{l,\pi_{n_l}}\right) \\ &\qquad - H\left(\mathbf{x}_{l,\pi_{n_{l+1}+1}}, \ldots, \mathbf{x}_{l,\pi_{n_l}}|\mathbf{x}_{l,\pi_1}, \mathbf{x}_{l,\pi_2}, \ldots, \mathbf{x}_{l,\pi_{n_{l+1}}}\right). \end{aligned} \tag{19}$$

Since the entropy $H(\mathbf{x}_{l,\pi_1}, \mathbf{x}_{l,\pi_2}, \ldots, \mathbf{x}_{l,\pi_{n_l}})$ is constant for a given graph, we can then express the target [see (14)] as

$$\min_{\pi \in \Pi} H\left(\mathbf{x}_{l,\pi_{n_{l+1}+1}}, \ldots, \mathbf{x}_{l,\pi_{n_l}} | \mathbf{x}_{l,\pi_1}, \mathbf{x}_{l,\pi_2}, \ldots, \mathbf{x}_{l,\pi_{n_{l+1}}}\right). \quad (20)$$

The greedy iPool strategy resorts to a greedy algorithm to solve it. Again, based on the locality of signals, if the Markov blanket of each variable is its k-hop neighbors, the greedy iPool strategy actually coarsens graphs according to a lower approximation of the objective function [see (20)]

$$H\left(\mathbf{x}_{l,\pi_{n_{l+1}+1}}, \ldots, \mathbf{x}_{l,\pi_{n_l}} | \left\{\mathbf{x}_{l,j}\right\}_{j=\pi_1}^{\pi_{n_{l+1}}}\right)$$

$$= \sum_{i=n_{l+1}+1}^{n_l} H\left(\mathbf{x}_{l,\pi_i} | \left\{\mathbf{x}_{l,j}\right\}_{j=\pi_1}^{\pi_{i-1}}\right) \quad (21.1)$$

$$\geq \sum_{i=n_{l+1}+1}^{n_l} H\left(\mathbf{x}_{l,\pi_i} | \left\{\mathbf{x}_{l,j}\right\}_{j=\pi_1}^{\pi_{i-1}} \bigcup \left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,\pi_i})}\right) \quad (21.2)$$

$$= \sum_{i=n_{l+1}+1}^{n_l} H\left(\mathbf{x}_{l,\pi_i} | \left\{\mathbf{x}_{l,j}\right\}_{N^k(v_{l,\pi_i})}\right) \quad (21.3)$$

where (21.1) is the chain rule unfolding and (21.3) is on the basis of the Markov blanket assumption. We can unfold (21.1) in such order that a node whose neighbors are mostly contained in $\{\mathbf{x}_{l,j}\}_{j=\pi_1}^{\pi_{i-1}}$ is first unfolded. With the expansion of chain rule, more nodes are in $\{\mathbf{x}_{l,j}\}_{j=\pi_1}^{\pi_{i-1}}$ and the neighbors of most nodes are contained. Thereby, (21.2) is a reasonable approximation to (21.1). Selecting the top n_{l+1} nodes with the largest neighborhood information gain globally, the greedy iPool strategy minimizes (21.3) the approximation of the objective function [see (20)].

Alternatively, the local iPool strategy directly optimizes (20) with a heuristic algorithm. Intuitively, the deleted nodes $(\mathbf{x}_{l,\pi_{n_{l+1}+1}}, \ldots, \mathbf{x}_{l,\pi_{n_l}})$ should be ones that are best predicted, i.e., the nodes with the smallest neighborhood information gain γ, in different neighborhoods. The local iPool strategy implements a fast procedure to globally select nodes according to the normalized neighborhood information gain. This is because, through normalization within each neighborhood, γ of the relatively uninformative and the rather informative nodes within each neighborhood will be rescaled to less than 1 and greater than 1, respectively. It avoids the iterative node selection and neighborhood information criterion update that is computational-intensive and time-consuming.

C. Invariance to Graph Isomorphism

We now further show that iPool is invariant to isomorphic graphs so that the GNNs consisting of iPool combined with other components that are also invariant to graph isomorphism will produce invariant representations for isomorphic graphs.

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.

Proof: Please refer to Appendix. ■

As demonstrated in Proposition 2, the iPool algorithm is invariant under graph isomorphism. It is the foundation for a pooling operator to be used in graph classification tasks since it guarantees that isomorphic graphs will be classified into the same category.

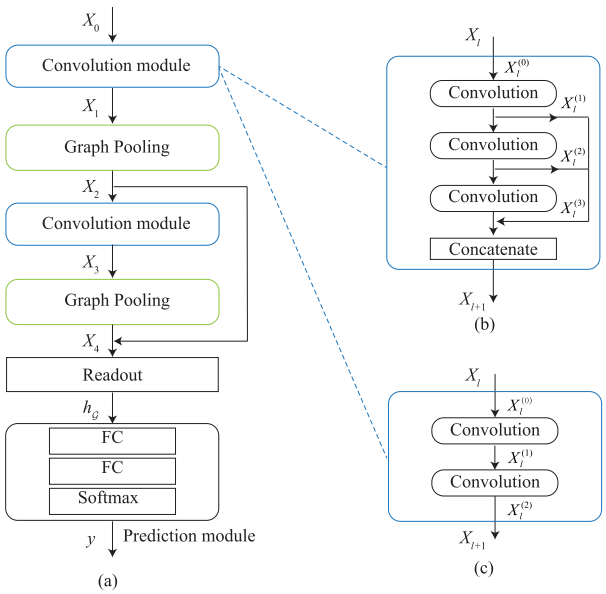

Fig. 2. Hierarchical graph convolution network architectures. The network is composed of a stack of convolution modules, graph pooling layers, a readout module, as well as a prediction module. (a) Network architecture. (b) Convolution module I. (c) Convolution module II.

V. Experiments

In this section, we evaluate the effectiveness of the proposed pooling schemes in graph classification tasks and compare with a collection of graph pooling operators. Although classification tasks are well studied on grid-like data, graph classification is more challenging with diverse graph signals and arbitrary irregular graph topologies that graph signals reside on. To correctly classify a graph, an effective representation considering jointly the topology and signal information of the graph is necessary. We evaluate the proposed pooling operator in multiscale representations of graph data and show its potentiality in graph classification tasks.

A. Graph Classification on Benchmark Data Sets

We conduct experiments to classify graphs on five widely used public benchmark graph data sets (ENZYMES, D&D, PROTEINS, NCI1, and NCI109).[2] Statistics and properties of these data sets are presented in Table I. In the experiments, node categorical features are adopted as graph signals.

1) Network Architecture: We evaluate the proposed pooling algorithms in the context of deep graph convolution networks. Specifically, the hierarchical graph convolution networks used in the experiments consist of two convolution modules, two graph pooling layers, a readout module, as well as a prediction module, as shown in Fig. 2(a). The adopted graph convolution layer follows the common message propagation and aggregation schemes with the specific formation:

$$X_l^{(m+1)} = \mathrm{ReLU}\left(\psi\left(A_l X_l^{(m)} W_l\right)\right), \quad X_l^{(0)} = X_l \quad (22)$$

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

TABLE I

GRAPH CLASSIFICATION ACCURACIES WITH TENFOLD CROSS VALIDATION. RESULTS OF BASELINE METHODS WITH "†" ARE CITED FROM THEIR ORIGINAL PUBLICATIONS

	Method	ENZYMES	D&D	PROTEINS	NCI1	NCI109
Datasets	Avg $\|\mathcal{V}\|$	32.63	284.32	39.06	29.87	29.68
	Avg $\|\mathcal{E}\|$	62.14	715.66	72.82	32.30	32.13
	#Classes	6	2	2	2	2
	#Graphs	600	1,178	1,113	4,110	4,127
Baselines	PSCN†	-	76.27 ± 2.64	75.00 ± 2.51	76.34 ± 1.68	-
	ECC†	53.50	74.10	72.65	76.82	75.03
	SET2SET	55.50 ± 6.99	78.93 ± 3.63	76.91 ± 4.04	81.19 ± 2.16	79.52 ± 1.81
	SortPooling†	57.12	79.37	76.26	74.44	-
	DiffPool	**62.17 ± 3.66**	**80.57 ± 2.91**	76.82 ± 3.86	81.17 ± 1.33	80.37 ± 1.28
	gPool	56.17 ± 3.58	76.81 ± 2.77	76.73 ± 4.14	**82.00 ± 1.25**	**80.96 ± 1.75**
	SAGPool	54.17 ± 4.67	78.94 ± 2.89	77.19 ± 3.95	80.61 ± 1.57	80.37 ± 2.04
Ablation	iPool-local	58.17 ± 3.11	**79.45 ± 2.40**	**77.36 ± 4.78**	**82.41 ± 2.29**	**80.93 ± 2.13**
	iPool-local, l_1	**59.50 ± 5.97**	79.11 ± 1.43	**77.63 ± 4.78**	81.95 ± 0.77	80.86 ± 2.06
	iPool-greedy	55.83 ± 6.47	77.91 ± 2.23	76.82 ± 3.66	80.97 ± 1.38	79.28 ± 1.52
	iPool-greedy, l_1	55.50 ± 7.03	78.16 ± 2.27	**77.36 ± 4.47**	80.97 ± 1.35	79.62 ± 2.31

where ReLU denotes the rectified linear unit activation function and $\psi(\cdot)$ indicates an l_2 normalization function to stabilize and accelerate the training process. Three such graph convolution layers are concatenated to form a convolution module in order to produce node features on the basis of information within three-hop neighborhoods, as shown in Fig. 2(b)

$$X_{l+1} = \text{Concat}\left(X_l^{(1)}, X_l^{(2)}, X_l^{(3)}\right). \tag{23}$$

A pooling layer follows a convolution module and coarsens graphs in accordance with the iPool operators, as introduced in Section III. 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

$$h_{\mathcal{G}} = \text{Concat}(\eta(X_l)|l = 2, 4) \tag{24}$$

where $\eta(\cdot)$ indicates an elementwise operator to aggregate information of all nodes along each dimension of features. Specifically, an elementwise sum operator and an elementwise max operator are adopted on all the data sets. A prediction module is finally added to the architecture for graph classification. It consists of two fully connected (FC) layers and a soft-max layer to make the prediction of the graph category based on the graph representation $h_{\mathcal{G}}$. No batch normalization layer is adopted in the neural networks.

2) Experimental Settings: In the experiments, we adopt the same network architecture on all the data sets, where each convolution layer consists of 64 hidden neurons. The pooling ratio is set as 0.5, i.e., 50% of nodes per graph deleted after a pooling layer. These networks are optimized with the Adam optimizer [35], with the following hyperparameters tuned through grid search for each data set: learning rate $\in \{1e{-}3, 1e{-}4, 1e{-}5\}$ and weight decay $\in \{0, 3e{-}5, 1e{-}4\}$. For all the data sets, one-hop neighborhood is adopted in computing the neighborhood information gain, i.e., $k = 1$ in (6), and λ is set as 0.6 in calculating edge weight [see (11)]. We implement the proposed model in Pytorch [36]. Like [33], we evaluate the models with tenfold cross validation (stratified sampling) on all of the data sets and report the best average accuracy.

3) Baseline Models: We compare the proposed methods with a collection of graph representation learning methods, especially state-of-the-art graph pooling operators for deep GNNs. Several GNNs are first taken into consideration, including **PSCN** [9] and **ECC** [6]. Then, two recent global pooling schemes are compared: **SET2SET** [28] exploits LSTMs to globally aggregate node information and produces a fixed-length graph embedding and **SortPooling** [25] performs global pooling by selecting a fixed number of nodes per graph according to the value of last channel of feature maps. Both of them do not construct the coarsened graphs. Hierarchical pooling methods are further considered: **gPool** [29] and **SAGPool** [30] resort to heuristic strategies, node footprint and self-attention, to select nodes, and these heuristic measures need to be optimized together with GCNs. Specifically, a node footprint is obtained by protecting features of nodes to a trainable vector, and the self-attention score of the node is computed with a graph convolution operator. Both of them take the induced subgraph of the original graph or its second power graph as the coarsened graph. Furthermore, **DiffPool** [33] learns an assignment matrix with an extra GCN to cluster nodes and produces coarsened graphs with a fixed size that is proportional to the size of the largest graph in the data set rather than the size of each graph. In order to make the average size of coarsened graphs consistent with those produced by the other methods on most data sets, we set the pooling ratio as 0.25 for DiffPool. Results of SET2SET, DiffPool, gPool, and SAGPool are reimplemented and compared under the same network architecture used for iPool with their respective hyperparameters (learning rate and weight decay) obtained through grid search, and the results of other baseline models are cited from their respective publications.

B. Graph Classification on MNIST and CIFAR-10 Data Sets

In addition to graph benchmark data sets, we further consider graphs from real-world data sets that are more

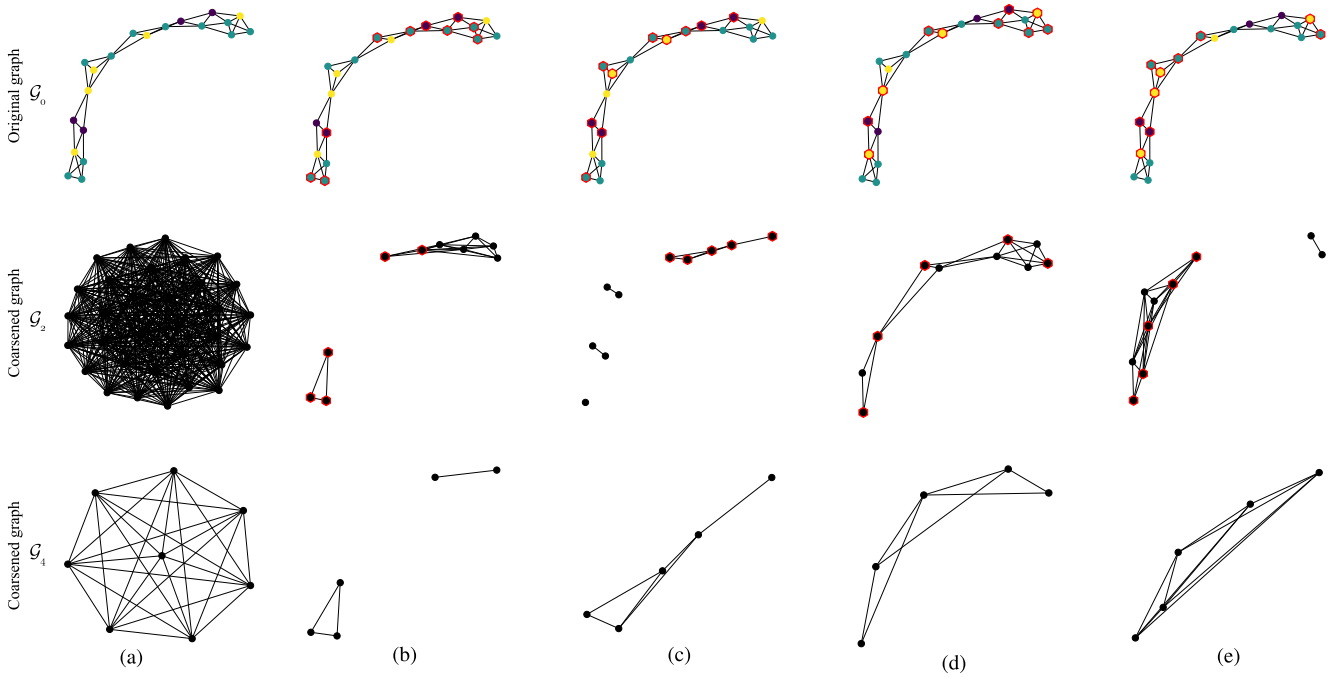

Fig. 3. Illustration of selected nodes and corresponding coarsened graphs produced by different pooling methods on a graph of the ENZYMES data set. Graph signals with node categories are indicated by different node colors, and selected nodes are indicated with red hexagons. (a) DiffPool. (b) gPool. (c) SAGPool. (d) iPool-local. (e) iPool-greedy.

challenging in terms of signal variation and graph category. We choose two popular image data sets, MNIST [37] and CIFAR-10 [38],[3] which contain 70 000 28×28 images and 60 000 32×32 images, respectively.

We construct a graph from the superpixels of each image with a procedure similar to the one used in [39] and [40]. Specifically, for each image, we adopt the superpixels computed in [39] with the SLIC algorithm [41]. Each superpixel is represented as a node, and the intensity and coordinates of superpixels, which are, respectively, the average intensity and coordinates of pixels covered by each superpixel, form the node signal. From these nodes, a k-nearest neighbor graph is subsequently constructed that v_i and v_j is connected with an edge if v_i is one of the k-nearest neighbors of v_j or v_j is one of the k-nearest neighbors of v_i in terms of the Euclidean distance of their coordinates, with $k = 8$. The edge weight is computed as the Gaussian kernel distance

$$(A)_{i,j} = \begin{cases} e^{-\frac{\|c(v_i)-c(v_j)\|_2^2}{\sigma_i \sigma_j}}, & v_i \in N(v_j) \text{ or } v_j \in N(v_i) \\ 0, & \text{others} \end{cases} \tag{25}$$

with

$$\sigma_i = \frac{1}{k} \sum_{v_j \in N(v_i)} \|c(v_i) - c(v_j)\|_2 \tag{26}$$

where $N(v_i)$ represent the neighborhood of node v_i composed of its k-nearest neighbors and $c(v_i)$ indicates its coordinate. In this way, we construct a weighted graph with a symmetric and nonnegative adjacency matrix for each image and correspondingly obtain a graph-version MNIST data set with 40–75 nodes per graph and a graph-version CIFAR-10 data set with graph size in 85–150.

1) Network Architecture: To evaluate the versatility of the proposed pooling operators, we adopt two widely used network architectures with different graph convolution operators. The first network is the same as the one used on benchmark data sets, except that the number of hidden neurons is increased to 128 per convolution layer in consideration of the difficulty on these data sets. Besides, another network is built on GraphSage [11] that has achieved significant performance on node-level tasks. Specifically, a convolution module is composed of two graph convolution layers as commonly used on node-level tasks, as shown in Fig. 2(c), each of which is adopted as a mean aggregator of GraphSage

$$g^{(m)}(v_{l,i}) = \text{MEAN}\{x^{(m)}(v_{l,j}), v_{l,j} \in N(v_{l,i})\} \tag{27}$$

$$x^{(m+1)}(v_{l+1,i}) = \text{ReLU}(\psi(W_{l,s} x^{(m)}(v_{l,i}) || W_{l,n} g^{(m)}(v_{l,i}))) \tag{28}$$

where $W_{l,s}$ and $W_{l,n}$ are learnable parameters, "||" indicates concatenation along the channel dimension, and ReLU as well as $\psi(\cdot)$ denote the rectified linear unit activation function and the l_2 normalization function, respectively. We use $X_l^{(m)} = [x^{(m)}(v_{l,1}), x^{(m)}(v_{l,2}), \ldots, x^{(m)}(v_{l,n_l})]^T$ to represent the graph features produced by the mth graph convolution layer in the lth graph convolution module, with $X_l^{(0)} = X_l$ and $X_{l+1} = X_l^{(2)}$.

2) Experimental Settings: Due to the large scale of these data sets, we are able to evaluate the generalization of models by splitting the data sets into training set, validation set, and testing set following the common procedure with the ratio 55 000:5000:10 000 and 45 000:5000:10 000 on MNIST and CIFAR-10, respectively. The model achieved the best

[3]These two data sets have been chosen to address the lack of large-scale graph data sets, rather than to improve on the state-of-the-art computer vision algorithms with GNNs.

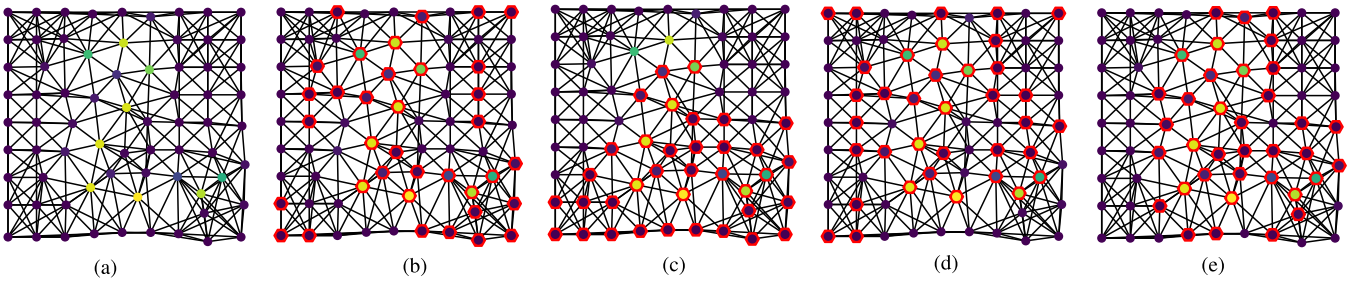

Fig. 4. Illustration of selected nodes produced by different pooling schemes under GraphSage network architecture on a graph of the MNIST data set. The color of the node indicates the intensity of node (superpixel), and nodes with red hexagons are the selected ones to form a coarsened graph. (a) Original graph. (b) gPool. (c) SAGPool. (d) iPool-local. (e) iPool-greedy.

TABLE II
GRAPH CLASSIFICATION RESULTS ON THE MNIST AND CIFAR-10 DATA SETS

Dataset	MNIST				CIFAR-10			
Avg $\|\mathcal{V}\|$	$70\ (40-75)$				$117\ (85-150)$			
#Classes	10				10			
#Graphs	$70,000$				$60,000$			
#Train: #val: #test	$55,000:5,000:10,000$				$45,000:5,000:10,000$			
Network	GCN		GraphSage		GCN		GraphSage	
Method/Measure	acc	micro-F1	acc	micro-F1	acc	micro-F1	acc	micro-F1
gPool	94.23 ± 0.05	94.18 ± 0.05	96.01 ± 0.39	95.97 ± 0.40	48.82 ± 0.05	48.46 ± 0.06	60.50 ± 0.96	60.29 ± 1.05
SAGPool	93.54 ± 0.87	93.46 ± 0.87	94.82 ± 1.46	94.76 ± 1.48	49.17 ± 0.22	48.82 ± 0.13	54.33 ± 1.62	54.19 ± 1.69
DiffPool	93.68 ± 0.07	93.62 ± 0.08	94.60 ± 0.67	94.55 ± 0.68	51.84 ± 0.11	51.47 ± 0.09	58.95 ± 0.59	58.86 ± 0.56
iPool-local	94.51 ± 0.10	94.45 ± 0.10	96.48 ± 0.05	96.45 ± 0.05	$\mathbf{51.94 \pm 0.11}$	$\mathbf{51.57 \pm 0.14}$	$\mathbf{63.43 \pm 0.25}$	$\mathbf{63.29 \pm 0.29}$
iPool-greedy	$\mathbf{94.63 \pm 0.17}$	$\mathbf{94.58 \pm 0.17}$	$\mathbf{96.85 \pm 0.17}$	$\mathbf{96.82 \pm 0.18}$	50.47 ± 0.23	50.24 ± 0.37	63.03 ± 0.32	62.89 ± 0.26

performance on the validation set is evaluated on the testing set. To eliminate the impact of network initialization, we repeat the evaluation procedure three times with different random seeds. The mean and standard variation of the test results in terms of accuracy (acc) and micro-F1 are reported.

The pooling ratio is 0.5 for all the models, including DiffPool. Since original graphs are weighted graphs, we set λ in calculating edge weight of the coarsened graph [see (11)] as 0.8 to preserve information encoded in the edge weight of the original graph. The learning rate is initialized as $1e{-}3$ and decays with scale 0.1 per 50 epochs. The other settings are the same as those in Section V-A.

C. Experimental Results and Analysis

We evaluate the classification performance of various baseline models on five graph benchmark data sets and two superpixel-induced data sets (MNIST and CIFAR-10). Even without any learning parameters, the proposed iPool algorithms still achieve competitive or superior performance compared with several GNNs as well a collection of graph pooling operators on the graph benchmark data sets, as presented in Table I. Furthermore, under two kinds of network architectures, iPool achieves superior performance over the other hierarchical graph pooling operators on the more complex MNIST and CIFAR-10 data sets (composed of graphs from ten categories) in terms of accuracy and micro-F1 scores, as shown in Table II.

The local iPool strategy outperforms its greedy counterpart on all the data sets, except for MNIST. Preserving informative nodes within each neighborhood, the local iPool strategy preserves better the structure of the graph while maintaining information of graph signals. As shown in Fig. 3, the local strategy prefers informative nodes from different neighborhoods, and thereby, the structure of coarsened graphs is more consistent with that of the original graphs. Since the background nodes are uninformative on MNIST, the greedy iPool strategy globally ignores them and thereby performs better. As an ablation study, we substitute the l_1 norm in computing neighborhood information gain for the l_2 norm. It achieves competitive performance but degrades a bit on most data sets, as presented in Table I.

Compared with node-selection-based pooling schemes including gPool and SAGPool, the local iPool strategy obtains the best performance on almost all the data sets. In accordance with the proposed neighborhood information gain, iPool performs better in selecting informative nodes. For example, compared with the selection of informative nodes by iPool, gPool prefers the pattern composed of green and navy blue nodes but loses information of yellow nodes, as shown in Fig. 3, for its selection guided by the inner product of node signal and a trainable vector that determines the specific pattern. In addition, SAGPool ignores several informative nodes to discriminate the number but preserves numerous noninformative background nodes on the MNIST data set, as shown in Fig. 4. In contrast, iPool strategies select all of the informative nodes that make up the number, especially the greedy iPool that effectively selects informative nodes as well as their surrounding nodes. Furthermore, as it directly takes the induced graph formed by the selected nodes, SAGPool loses part of structural information. Considering the overlap of neighborhood, iPool, espe-

TABLE III
NUMBER OF PARAMETERS OF DIFFERENT GRAPH POOLING OPERATORS

iPool	DiffPool	SET2SET	SAGPool	gPool
0	$O(d_l \times n_{l+1})$	$O(d_l^2)$	$O(d_l)$	$O(d_l)$

TABLE IV
GRAPH CLASSIFICATION RESULTS ON THE REDUCED MNIST AND CIFAR-10 DATA SETS UNDER GRAPHSAGE NETWORK ARCHITECTURE

Dataset	Method/Measure	acc	micro-F1	Degrade
MNIST (5000)	gPool	89.75 ± 0.38	89.65 ± 0.39	-6.6%
	SAGPool	88.61 ± 4.39	88.46 ± 4.48	-6.6%
	DiffPool	86.49 ± 0.34	86.34 ± 0.32	-8.6%
	iPool-local	91.37 ± 0.37	91.28 ± 0.38	-5.3%
	iPool-greedy	**92.02 ± 0.17**	**91.94 ± 0.16**	-5.0%
CIFAR-10 (5000)	gPool	45.84 ± 0.57	45.47 ± 0.63	-24.4%
	SAGPool	40.03 ± 1.42	39.64 ± 1.35	-26.6%
	DiffPool	43.35 ± 0.36	42.96 ± 0.28	-26.7%
	iPool-local	**49.42 ± 0.23**	**49.13 ± 0.22**	-22.2%
	iPool-greedy	49.29 ± 0.07	49.02 ± 0.10	-21.9%

cially the local iPool strategy, preserves better the structural information.

With only basic elements of graphs, iPool does not involve any learnable parameter, which makes training easy and reduces storage complexity. As listed in Table III, the number of parameters in DiffPool is dependent on both the number of nodes n_{l+1} and the dimension of features d_l. While the number of parameters in SET2SET is quadratic to the dimension of features, gPool and SAGPool reduce it to be linear to the feature dimension. The large number of parameters, such as $O(d_l \times n_{l+1})$ in DiffPool, relies on a large number of labeled data to train. As presented in Table IV, with only 500 graphs per category as reduced training sets, iPool still achieves the best performance on reduced MNIST and CIFAR-10 data sets in terms of both metrics on the same validation and test sets. Furthermore, compared with the results achieved with the full training set, the performance degradation of iPool is the least among these hierarchical graph pooling operators. These suggest that the proposed parameter-free iPool operator alleviates the need for training data and leads to better generalization.

Furthermore, iPool is also competitive in terms of computational complexity. Specifically, the complexity of computing neighborhood information gain is $O(|\mathcal{V}_l|^2 d_l)$ (d_l denotes the feature dimension) with dense implementation or $O(|\mathcal{E}_l| d_l)$ with spare implementation, for $k = 1$ used in the experiments. The computational complexity of coarsened graph construction is $O(|\mathcal{V}_{l+1}|^2 u_l)$, where u_l is the maximum node degree. Overall, the computational complexity of forward propagation is $O(|\mathcal{V}_l|^2 d_l + |\mathcal{V}_{l+1}|^2 u_l) = O(|\mathcal{V}_l|^2 (d_l + \rho^2\, u_l))$ with the pooling ratio $0 < \rho < 1$. At the backward propagation, it directly backpropagates the gradient without any computation and thereby with a constant complexity $O(1)$. In contrast, the computational complexity of DiffPool is $O(|\mathcal{V}_l|^3 \rho + |\mathcal{V}_l|^3 \rho^2 + |\mathcal{V}_l|^2 \rho d_l)$ for both forward and backward propagations. For gPool, the computation is dominated by coarsened graph construction $O(|\mathcal{V}_l|^3 \rho^2)$ at forward propagation and $O(|\mathcal{V}_l| d_l)$ at backward propagation. SAGPool achieves a relatively low computational complexity $O(|\mathcal{V}_l|^2)$ for both forward and backward propagations by directly taking the induced graph as coarsened graphs at the expense of structural information loss. Thus, for large-scale graphs (usually $|\mathcal{V}_l| \gg d_l$), iPool will be faster than DiffPool and competitive with gPool and SAGPool at forward propagation. At backward propagation during the training phase, iPool is the fastest one. In addition, we further compare the practical training time (including forward and backward propagation) of the networks with the same architecture except for adopting different pooling layers on a workstation (GPU: GeForce GTX 1080 Ti and CPU: Intel Xeon E5-1620 V4). As listed in Table V, the running time of iPool is competitive with gPool and SAGPool and is less than DiffPool and SET2SET. We finally note that the running time may vary a lot with the implementation and the computational settings. Our relative comparison, however, confirms the computational complexity analysis provided earlier, where iPool is shown to be among the low-complexity methods.

TABLE V
RUNNING TIME (SEC/EPOCH) OF NETWORKS WITH DIFFERENT GRAPH POOLING OPERATORS AT THE TRAINING PHASE (g/L INDICATES THE GREEDY/LOCAL STRATEGY OF IPOOL)

	iPool (g/l)	DiffPool	SET2SET	SAGPool	gPool
NCI1	3.28/3.31	4.52	17.40	3.26	3.26
DD	4.39/4.36	5.05	58.95	4.31	4.38
MNIST	19.24/18.94	25.83	-	18.02	17.94

VI. CONCLUSION

We have proposed in this article a parameter-free, low complexity, and interpretable graph pooling operator for GNNs, which improves their capability of distilling hierarchical representations of graph and network data. The new operator has interesting properties in practice, in that it is mostly based on local computations, it leads to invariance properties under graph isomorphism, and it produces coarsened graphs that faithfully represent the original graph. The proposed iPool solution further permits to achieve superior or competitive performance on several graph classification data sets. 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 [15], and to apply it to other tasks, including link prediction.

APPENDIX

PROOF OF PROPOSITION 2

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

$$t : \mathcal{V}_l \longrightarrow \mathcal{V}'_l. \tag{29}$$

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

$$\begin{aligned}\gamma\left(v_{l,i}\right) &= \| x\left(v_{l,i}\right) - f\left(v_{l,i}\right) \|_2 \\ &= \left\| x\left(v_{l,i}\right) - \frac{1}{k}\sum_{h=1}^{k} \sum_{v_{l,j}\in N^h\left(v_{l,i}\right)} \left(\bar{P}_l^h\right)_{ij} \times x\left(v_{l,j}\right) \right\|_2 \end{aligned} \tag{30}$$

$$\begin{aligned}\gamma\left(v'_{l,m}\right) &= \|x\left(v'_{l,m}\right) - f\left(v'_{l,m}\right)\|_2 \\ &= \left\| x\left(v'_{l,m}\right) - \frac{1}{k}\sum_{h=1}^{k} \sum_{v'_{l,n}\in N^h\left(v'_{l,m}\right)} \left(\bar{P}_l^{'h}\right)_{mn} \times x\left(v'_{l,n}\right) \right\|_2 . \end{aligned} \tag{31}$$

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 $(v'_{l,m}, v'_{l,n}) \in \mathcal{E}'_l$

$$\begin{aligned}(A_l)_{ij} &= \left(A'_l\right)_{mn}, \quad \left(A_l^h\right)_{ij} = \left(A_l^{'h}\right)_{mn} \\ \left(\bar{P}_l^h\right)_{ij} &= \left(\bar{P}_l^{'h}\right)_{mn}. \end{aligned} \tag{32}$$

Therefore,

$$f\left(v_{l,i}\right) = f\left(v'_{l,m}\right), \quad \gamma\left(v_{l,i}\right) = \gamma\left(v'_{l,m}\right) \quad \forall v_{l,i} \in \mathcal{V}_l \tag{33}$$

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

$$\mathcal{V}_{l+1} = \mathcal{V}'_{l+1}, \quad \mathcal{E}_{l+1} = \mathcal{E}'_{l+1}, \quad X_{l+1} = X'_{l+1}. \tag{34}$$

Thus, the iPool operator is invariant to isomorphism.

ACKNOWLEDGMENT

This work was partially performed at the Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, when Xing Gao visited Prof. Pascal Frossard under the support of China Scholarship Council (CSC).

REFERENCES

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[2] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent.*, Banff, AB, Canada, Apr. 2014.

[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 3844–3852.

[4] R. Khasanova and P. Frossard, "Graph-based isometry invariant representation learning," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Aug. 2017, pp. 1847–1856.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, Toulon, France, Apr. 2017.

[6] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 3693–3702.

[7] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, p. 146, Nov. 2019.

[8] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[9] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 2014–2023.

[10] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.

[11] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 1024–1034.

[12] K. Xu *et al.*, "Representation learning on graphs with jumping knowledge networks," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 5449–5458.

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Aug. 2017, pp. 1263–1272.

[14] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2015, pp. 2224–2232.

[15] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. 7th Int. Conf. Learn. Represent.*, New Orleans, LA, USA, May 2019.

[16] P. Veličković *et al.*, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, 2018.

[17] T. N. Bui and C. Jones, "Finding good approximate vertex and edge partitions is NP-hard," *Inf. Process. Lett.*, vol. 42, no. 3, pp. 153–159, May 1992.

[18] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[19] D. I. Shuman, M. J. Faraji, and P. Vandergheynst, "A multiscale pyramid transform for graph signals," *IEEE Trans. Signal Process.*, vol. 64, no. 8, pp. 2119–2134, Apr. 2016.

[20] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Jan. 1998.

[21] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Montreal, QC, Canada, Aug. 2005, pp. 729–734.

[22] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 81–102, Jan. 2009.

[23] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[24] R. Liao, Z. Zhao, R. Urtasun, and R. Zemel, "LanczosNet: Multi-scale deep graph convolutional networks," in *Proc. 7th Int. Conf. Learn. Represent.*, New Orleans, LA, USA, May 2019.

[25] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, Feb. 2018, pp. 4438–4445.

[26] D. Ron, I. Safro, and A. Brandt, "Relaxation-based coarsening and multiscale graph organization," *Multiscale Modeling Simul.*, vol. 9, no. 1, pp. 407–423, Mar. 2011.

[27] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[28] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015.

[29] H. Gao and S. Ji, "Graph U-Nets," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, Jun. 2019, pp. 2083–2092.

[30] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, Jun. 2019, pp. 3734–3743.

[31] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, Jun. 2010, pp. 111–118.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[33] Z. Ying *et al.*, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2018, pp. 4805–4815.
[34] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with EigenPooling," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, Jul. 2019, pp. 723–731.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015.
[36] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. Adv. Neural Inf. Process. Syst., Autodiff Workshop*, Long Beach, CA, USA, Dec. 2017. [Online]. Available: https://openreview.net/pdf?id=BJJsrmfCZ
[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. of Toronto, Toronto, ON, Canada, Tech. Rep., Apr. 2009.
[39] B. Knyazev, G. W. Taylor, and M. Amer, "Understanding attention and generalization in graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2019, pp. 4202–4212.
[40] V. Prakash Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," 2020, *arXiv:2003.00982*. [Online]. Available: http://arxiv.org/abs/2003.00982
[41] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

Xing Gao received the B.S. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Electronic Engineering.

From 2018 to 2019, he was a guest Ph.D. Student with the Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. His research focuses on developing novel algorithms on the basis of the intersection of deep learning and signal processing to learn representation of data, especially representation learning of graph data and images.

Wenrui Dai (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2005, 2008, and 2014, respectively.

He is currently an Associate Professor with the Department of Computer Science and Engineering, SJTU. Before joining SJTU, he was with the faculty of The University of Texas Health Science Center at Houston, Houston, TX, USA, from 2018 to 2019. He was a Post-Doctoral Scholar with the Department of Computer Science and Engineering, SJTU, from 2014 to 2015, and the Department of Biomedical Informatics, University of California at San Diego, San Diego, CA, USA, from 2015 to 2018. His research interests include learning-based image/video coding, image/signal processing, and predictive modeling.

Chenglin Li (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2007, 2009, and 2015, respectively.

From 2011 to 2013, he was a Visiting Ph.D. Student with the Electrical and Computer Engineering Department, University of Florida, Gainesville, FL, USA. From 2015 to 2017, he was a Post-Doctoral Research Fellow with the Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. From 2017 to 2018, he was a Senior Researcher with the Chair of Media Technology (LMT), Technical University of Munich (TUM), Munich, Germany, supported by the Hildegard Maier Research Fellowship of the Alexander von Humboldt Foundation for post-doctoral researchers. Since 2018, he has been an Associate Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. His main research interests include network-oriented image/video processing and communication, network-based optimization for video sources, adaptive multimedia communication systems, and caching and edge computing for multimedia delivery.

Dr. Li was awarded as the Microsoft Research Asia (MSRA) Fellow in 2011 and the Alexander von Humboldt Research Fellow in 2017. He received the IEEE International Conference on Visual Communications and Image Processing (VCIP) Best 10% Paper Award in 2016.

Hongkai Xiong (Senior Member, IEEE) received the Ph.D. degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2003.

He was an Associate Professor from 2005 to 2011 and an Assistant Professor from 2003 to 2005. From 2007 to 2008, he was a Research Scholar with the Department of Electrical and Computer Engineering, Carnegie Mellon University (CMU), Pittsburgh, PA, USA. From 2011 to 2012, he was the Scientist of the Division of Biomedical Informatics, University of California at San Diego (UCSD), San Diego, CA, USA. He is currently a Distinguished Professor with the Department of Electronic Engineering and the Department of Computer Science and Engineering, SJTU. Since then, he has been with the Department of Electronic Engineering, SJTU. He has published more than 200 refereed journal and conference papers. His research interests include multimedia signal processing, image and video coding, multimedia communication and networking, computer vision, biomedical informatics, and machine learning.

Dr. Xiong was a coauthor of the Best Paper in IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) 2013, the Best Student Paper in IEEE International Conference on Visual Communications and Image Processing (VCIP) 2014, the Top 10% Paper Award in VCIP 2016, and the Top 10% Paper Award in IEEE International Workshop on Multimedia Signal Processing (MMSP) 2011. He is an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT).

Pascal Frossard (Fellow, IEEE) was a member of the Research Staff at the International Business Machines Corporation (IBM) T. J. Watson Research Center, Yorktown Heights, NY, USA, from 2001 to 2003. He has been a Faculty Member with the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, since 2003, where he heads the Signal Processing Laboratory (LTS4). His research interests include network data analysis, image representation and understanding, and machine learning.

Dr. Frossard is a fellow of ELLIS. He received the Swiss National Science Foundation (SNSF) Professorship Award in 2003, the IBM Faculty Award in 2005, the IBM Exploratory Stream Analytics Innovation Award in 2008, the IEEE TRANSACTIONS ON MULTIMEDIA Best Paper Award in 2011, the *IEEE Signal Processing Magazine* Best Paper Award in 2016, and the Google Faculty Award 2017.