DEMO-Net: Degree-specific Graph Neural Networks for Node and Graph Classification

Jun Wu
Arizona State University
junwu6@asu.edu

Jingrui He
Arizona State University
Jingrui.he@asu.edu

Jiejun Xu
HRL Laboratories, LLC
jxu@hrl.com

ABSTRACT

Graph data widely exist in many high-impact applications. Inspired by the success of deep learning in grid-structured data, graph neural network models have been proposed to learn powerful node-level or graph-level representation. However, most of the existing graph neural networks suffer from the following limitations: (1) there is limited analysis regarding the graph convolution properties, such as *seed-oriented*, *degree-aware* and *order-free*; (2) the node's degree-specific graph structure is not explicitly expressed in graph convolution for distinguishing structure-aware node neighborhoods; (3) the theoretical explanation regarding the graph-level pooling schemes is unclear.

To address these problems, we propose a generic degree-specific graph neural network named *DEMO-Net* motivated by Weisfeiler-Lehman graph isomorphism test that recursively identifies 1-hop neighborhood structures. In order to explicitly capture the graph topology integrated with node attributes, we argue that graph convolution should have three properties: *seed-oriented*, *degree-aware*, *order-free*. To this end, we propose multi-task graph convolution where each task represents node representation learning for nodes with a specific degree value, thus leading to preserving the degree-specific graph structure. In particular, we design two multi-task learning methods: degree-specific weight and hashing functions for graph convolution. In addition, we propose a novel graph-level pooling/readout scheme for learning graph representation provably lying in a degree-specific Hilbert kernel space. The experimental results on several node and graph classification benchmark data sets demonstrate the effectiveness and efficiency of our proposed *DEMO-Net* over state-of-the-art graph neural network models.

KEYWORDS

Graph Neural Network, Degree-specific Convolution, Multi-task Learning, Graph Isomorphism Test

ACM Reference Format:
Jun Wu, Jingrui He, and Jiejun Xu. 2019. *DEMO-Net*: Degree-specific Graph Neural Networks for Node and Graph Classification. In *Proceedings of ACM Conference (Conference'17),* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Nowadays, graph data is being generated across multiple high-impact application domains, ranging from bioinformatics [4] to financial fraud detection [27, 28], from genome-wide association study [21] to social network analysis [5]. In order to leverage the rich information in graph-structured data, it is of great importance to learn effective node or graph representation from both node/edge attributes and the graph topological structure. To this end, numerous graph neural network models have been proposed recently inspired by the success of deep learning architectures on grid-structured data (e.g., images, videos, languages, etc.). One intuition behind this line of approaches is that the topological structure as well as node attributes could be integrated by recursively aggregating and compressing the continuous feature vectors from local neighborhoods in an end-to-end training architecture.

One key component of graph neural networks [4, 6] is the graph convolution (or feature aggregation function) that aggregates and transforms the feature vectors from a node's local neighborhood. By integrating the node attributes with the graph structure information using Laplacian smoothing [9, 12] or advanced attention mechanism [18], graph neural networks learn the node representation in a low-dimensional feature space where nearby nodes in the graph would share a similar representation. Moreover, in order to learn the representation for the entire graph, researchers have proposed the graph-level pooling schemes [1] that compress the nodes' representation into a global feature vector. The node or graph representation learned by graph neural networks has achieved state-of-the-art performance in many downstream graph mining tasks, such as node classification [26], graph classification [22], etc.

However, most of the existing graph neural networks suffer from the following limitations. **(L1)** There is limited analysis on graph convolution properties that could guide the design of graph neural networks when learning node representation. **(L2)** In order to preserve the node proximity, the graph convolution applies a special form of Laplacian smoothing [12], which simply mixes the attributes from node's neighborhood. This leads to the loss of degree-specific graph structure information for the learned representation. An illustrative example is shown in Figure 1: although nodes 4 and 5 are structurally different, they would be mapped to similar representation due to first-order node proximity using existing methods. Moreover, the neighborhood sub-sampling methods used to improve model efficiency [5] significantly degraded the discrimination of degree-specific graph structure. **(L3)** The theoretical explanation regarding the graph-level pooling schemes is largely missing.

To address the above problems, in this paper, we propose a generic graph neural network model *DEMO-Net* that considers the

degree-specific graph structure in learning both node and graph representation. Inspired by Weisfeiler-Lehman graph isomorphism test [20], the graph convolution of graph neural networks should have three properties: *seed-oriented*, *degree-aware*, *order-free*, in order to map different neighborhoods to different feature representation. As shown in Figure 1, nodes with identical degree value typically share similar subtree (root node followed by its 1-hop neighbors) structures. As a result, the representation of nodes 2 and 8 should be close in the feature space due to the similar subtree structure. On the other hand, nodes 4 and 5 have different subtree structures (i.e., number of subtree leaves), and they indicate different roles in the network, e.g., leader vs. deputy in a covert group. Therefore, they should not be mapped closely in the feature space.

To the best of our knowledge, very little effort on graph neural networks is devoted to learning the degree-specific representation for each node or the entire graph. To bridge the gap, we present a degree-specific graph convolution by assuming that nodes with the same degree value would share the same graph convolution. It can be formulated as a multi-task feature learning problem where each task represents the node representation learning for nodes with specific degree values.

In addition, we introduce a degree-specific graph-level pooling scheme to learn the graph representation. We theoretically show that the graph representation learned by our model lies in a Reproducing Kernel Hilbert space (RKHS) induced by a degree-specific Weisfeiler-Lehman graph kernel. The most similar work to us is Graph Isomorphism Network (GIN) [22] which used the sum-aggregator associated with multi-layer perceptrons as the neighborhood-injective graph convolution that mapped different node neighborhood to different features. However, one issue of GIN is that the degree-aware structures are implicitly expressed in its graph convolution relying on the universal approximation capacity of multi-layer perceptrons.

The main contributions of this paper are summarized as follows:

(1) We provide theoretical analysis for graph neural networks from the perspective of Weisfeiler-Lehman graph isomorphism test, which motivates us to design the graph convolution based on the following properties: seed-oriented, degree-aware and order-free.
(2) we propose a generic graph neural network framework named *DEMO-Net* by assuming that nodes with the same degree value would share the same graph convolution. A degree-specific multi-task graph convolution function is presented to learn the node representation. Furthermore, a novel graph-level pooling scheme is introduced for learning the graph representation provably lying in a degree-specific Hilbert kernel space.
(3) The experimental results on several node and graph classification benchmark data sets demonstrate the effectiveness and efficiency of our proposed *DEMO-Net* model.

The rest of the paper is organized as follows. We review the related work in Section 2, followed by the problem definition and background introduction in Section 3. Section 4 presents our proposed *DEMO-Net* framework for node and graph representation learning. The extensive experiments and discussion are provided in Section 5. Finally, we conclude the paper in Section 6.

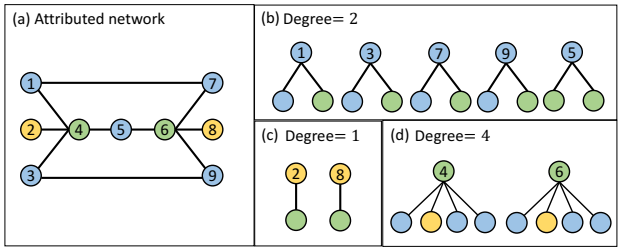

Figure 1: Nodes with the same degree value are structurally similar. For example, nodes 1, 3, 5, 7 and 9 in (b), nodes 2 and 8 in (c), nodes 4 and 6 in (d) share similar 1-hop neighborhood structure. Using the proposed model, the learned node representation integrates the degree-specific graph structure and node attributes such that the structurally similar nodes have similar representation.

2 RELATED WORK

In this section, we briefly review the related work on graph neural networks for node and graph classification.

2.1 Node Classification

Most of the existing graph neural networks [6] learn the node representation by recursively aggregating the continuous feature vectors from local neighborhoods in an end-to-end fashion. They could be fitted into the Message Passing Neural Networks (MPNNs) [4] which explained the feature aggregation of graph neural networks as message passing in local neighborhoods. Generally, they focus on extracting the spatial topological information by operating the convolutions in the node domain [26], which differs from some spectral approaches [2, 9] considering a node representation in the spectral domain. Graph Convolutional Network (GCN) [9] defined the convolution operation via a neighborhood aggregation function. Following the same intuition, many graph neural network models have been proposed with different aggregation functions, e.g., attention mechanism [18], mean and max functions [5], etc.

However, most of the graph neural network architectures are motivated by the success of deep learning on grid-like data, thus leading to little theoretical analysis for explaining the high performance and guiding the novel methodologies. Up till now, some work have been proposed to explain why graph neural networks work. The convolution of GCN was a special form of Laplacian smoothing on graph [12], which explained the over-smoothing phenomena brought by many convolution layers. Lei et al. [10] showed that the graph representation generated by graph neural networks lies in the Reproducing Kernel Hilbert Space (RKHS) of some popular graph kernels. Moreover, it shows that 1-dimensional aggregation-based graph neural networks are at most as powerful as the Weisfeiler-Lehman (WL) isomorphism test [20] in distinguishing graphs [22]. Compared with the existing work on graph neural networks, in this paper, we design a degree-specific graph convolution that captures the node neighborhood structures inspired by WL isomorphism test. This is in sharp contrast to the existing work which focused on preserving the node proximity in the feature space, thus leading to the loss of local graph structures.

2.2 Graph Classification

The graph-level pooling/readout schemes aim to learn a representation of the entire graph from its node representations for graph-level classification tasks. Mean/max/sum functions are commonly used due to its computational efficiency and effectiveness [1, 22]. One challenge for graph-level pooling is to maintain the invariance to node order. PATCHY-SAN [13] first adopted the external software to obtain a global node order for the entire graph, which is very time-consuming. More recently, a number of graph neural network models have been proposed [22, 24, 25], which formulated the node representation learning and graph-level pooling into a unified framework. Different from graph kernel approaches [16, 23] that intuitively extract the graph feature or define the graph similarity using ad-hoc knowledge or random walk properties, graph neural networks would automatically learn the graph representation to integrate node attributes with its topological information via an end-to-end training architecture.

Nevertheless, very little effort has been devoted to explicitly considering the degree-specific graph structures for graph representation learning. Our proposed degree-specific graph-level pooling method is designed to address this issue by compressing the learned node representation according to degree values.

3 PRELIMINARIES

In this section, we introduce the notation and problem definition, as well as some background information on graph neural networks.

3.1 Notation

Suppose that a graph is represented as $G = (V, E)$, where $V = \{v_1, \cdots, v_n\}$ is the set of n nodes and $E \subseteq V \times V$ is the edge set. Let $X \in \mathbb{R}^{n \times D}$ denote the attribute matrix where each row x_v is the D-dimensional attribute vector for node v. The graph G can also be represented by an adjacency matrix $A \in \mathbb{R}^{n \times n}$, where A_{ij} represents the similarity between v_i and v_j on the graph. For each node $v \in V$, its 1-hop neighborhood is denoted as $N(v)$. Let $\mathcal{G} = \{G_1, \cdots, G_t\}$ denote a set of graphs. In this paper, we focus on undirected attributed networks, although our model can be naturally generalized to other types of networks. The main notation used in this paper is summarized in Table 1.

3.2 Problem Definition

In this paper, we focus on two problems: node-level and graph-level representation learning by formulating a novel degree-specific graph neural network model. Furthermore, we analyze the proposed model from various aspects, and empirically demonstrate its superior performance on both node and graph classification.

Formally, the node- and graph-level representation learning problems can be defined below.

Definition 3.1. (***Node-level Representation Learning***)
Input: (i) An attributed graph $G = (V, E)$ with adjacency matrix $A \in \mathbb{R}^{n \times n}$ and node attributes $X \in \mathbb{R}^{n \times D}$; (ii) Labeled training nodes $\{x_v, y_v\}_{v \in \mathcal{I}_G}$.

Output: A vector representation $e_v \in \mathbb{R}^d$ for each node $v \in V$ on the d-dimensional embedding space where nodes would be well separated if their local neighborhoods are structurally different.

Table 1: Notation

Notation	Definition
$\mathcal{G} = \{G_i\}_{i=1}^t$	A set of graphs
$G = (V, E)$	A graph G with node set V and edge set E
X	Attribute matrix
A	Adjacency matrix
n	Number of nodes in the graph
d	Dimensionality of the node or graph representation
$N(v)$	1-hop neighborhood of node v
$\mathcal{I}_G$	Indices of labeled nodes' for node classification
$\mathcal{I}_\mathcal{G}$	Indices of labeled graphs' for graph classification
y_v	Label of node v
$\hat{y}_i$	Label of graph G_i
h_v^k	Node v's representation at the k^{th} iteration
$h_{N(v)}$	Feature set within node v's neighborhood
$\mathcal{T}$	A set of subtrees
$degree(G)$	A set of the degree values in graph G

Definition 3.2. (***Graph-level Representation Learning***)
Input: (i) A set of attributed graphs $\mathcal{G} = \{G_i\}_{i=1}^t$ with adjacency matrix $A_i \in \mathbb{R}^{n_i \times n_i}$ and node attributes $X_i \in \mathbb{R}^{n_i \times D}$; (ii) Labeled training graphs $\{G_i, \hat{y}_i\}_{i \in \mathcal{I}_\mathcal{G}}$.

Output: A vector representation $g_i \in \mathbb{R}^d$ for each graph G_i on the d-dimensional embedding space where graphs would be well separated if they have different graph topological structure.

3.3 Graph Neural Networks

It has been observed that a broad class of graph neural network (GNN) architectures followed the 1-dimensional Weisfeiler-Lehman (WL) graph isomorphism test [20]. From the perspective of WL isomorphism test, they mainly consist of the following crucial steps at each iteration of feature aggregation:

- Feature initialization (label[1] initialization): The node features are initialized by original attribute vectors.
- Neighborhood detection (multiset-label determination): It decides the local neighborhood in which node gathers the information from neighbors. More specifically, a seed[2] followed by its neighbors generates a subtree pattern.
- Neighbors sorting (multiset-label sorting): The neighbors are sorted in the ascending or descending order of degree values. The subtrees with permutation order of neighbors are recognized as the same one.
- Feature aggregation (label compression): The node feature is updated by compressing the feature vectors of the aggregated neighbors including itself.
- Graph-level pooling (graph representation): It summarizes all the node features to form a global graph representation.

Next, we briefly go over some existing graph neural network models, which follow the aforementioned steps of the 1-dimensional

[1] Here, label is an identifier of nodes. In order not to be confused with a class label, we will use node attribute to represent it in this paper.
[2] The seed denotes the root node to be learned in the graph. For example, node v_1 in Figure 2 is a seed when updating its feature at each iteration.

WL algorithms. We would like to point out that graph neural networks would learn the node or graph representation using continuous node attributes, whereas WL algorithms update the node attributes by directly compressing the augmented discrete attributes.

Taking 1-hop neighborhood $N(v) = \{u|(v,u) \in E\}$ into consideration at each iteration, the following node-level graph neural network variants have the same feature initialization and neighborhood detection on learning node representation. And when element-wise average or max operations are used for feature aggregation, graph neural networks would be invariant to the order of neighbors. We summarize the feature aggregation functions (graph convolution) of those graph neural networks as follows.

- Graph Convolutional Network (GCN) [9]:

$$h_v^k = \sigma\Big(\sum\nolimits_{u\in\{v\}\cup N(v)} \widehat{a}_{vu} W^k h_u^{k-1}\Big) \tag{1}$$

  where $\widehat{A} = (\widehat{a}_{vu}) \in \mathbb{R}^{n\times n}$ is the re-normalization of the adjacency matrix A with added self-loops, and W^k is the trainable matrix at k^{th} layer. It is essentially a weighted feature aggregation from node neighborhood.
- Graph Attention Network (GAT) [18]:

$$h_v^k = \sigma\Big(\sum\nolimits_{u\in\{v\}\cup N(v)} \alpha_{vu} W^k h_u^{k-1}\Big) \tag{2}$$

  where α_{vu} is a self-attention score indicating the importance of node u to node v on feature aggregation. It is obvious that GCN can be considered as a special case of GAT when the attention score α_{vu} is defined as $\widehat{a}_{vu}$.
- GraphSAGE [5]:

$$\begin{aligned} h_{N(v)}^k &= AGGREGATE_k\Big(\{h_u^{k-1}|u \in N(v)\}\Big) \\ h_v^k &= \sigma\Big(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^k)\Big) \end{aligned} \tag{3}$$

  where mean-, max- and LSTM-aggregator are presented for feature aggregation. Though LSTM considers node neighbors as an ordered sequence, the LSTM aggregator is adapted on an unordered neighbors with random permutation.

There are some observations from these GNN variants: (i) Their feature aggregation schemes are invariant to the order of the neighbors except for GraphSAGE with LSTM-aggregator; (ii) The output feature at k-layer neural network can be seen as the representation of a subtree around the seed; (iii) The node representation become closer and indistinguishable when the neural layers are going deeper, because the subtrees would share more common elements. However, little work theoretically discusses the reasons behind these observations to guide the design of graph neural networks: how is the node representation affected by node degree and order of neighbors? what kind of graph convolution is required to learn the subtree structures? Inspired by WL graph isomorphism test, we present a degree-specific graph neural network model named *DEMO-Net* in Section 4 to discuss those problems.

Additionally, the neighborhood aggregation schemes of graph neural networks, such as mean-aggregator in GraphSAGE [5], self-attention in GAT [18], can be regarded as the relabeling step in WL isomorphism test. Figure 2 provides an example to illustrate the essence of feature aggregation on graph neural networks. The node feature is actually a special representation of subtree consisting of the seed followed by its neighbors. For example, node 1's feature

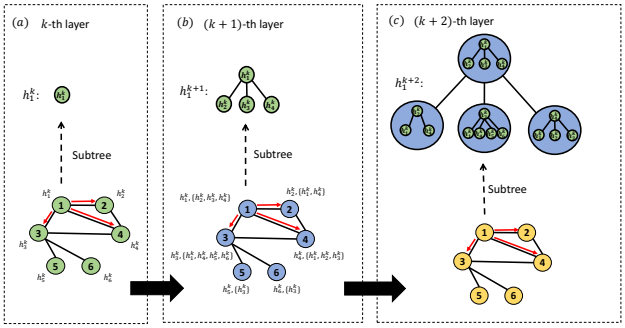

Figure 2: Feature aggregation of the graph neural networks: For node 1, its feature is (a) h_1^k at k^{th} layer; (b) h_1^{k+1} at $(k+1)^{\text{th}}$ layer compressed from a subtree $(h_1^k; \{h_2^k, h_3^k, h_4^k\})$; (c) h_1^{k+2} at $(k+2)^{\text{th}}$ layer learned from a subtree $(h_1^{k+1}; \{h_2^{k+1}, h_3^{k+1}, h_4^{k+1}\})$.

h_1^{k+1} represents the subtree $(h_1^k; \{h_2^k, h_3^k, h_4^k\})$ collected from previous layer. As a result, graph neural networks with k layers learn the representation of subtree with depth k rooted at the seed. That provides us an intuition to design a graph convolution for explicitly preserving the degree-specific subtree structures.

4 PROPOSED MODEL: *DEMO-NET*

In this section, we propose a generic degree-specific graph neural network named *DEMO-Net*. Key to our algorithm is the degree-specific graph convolution for feature aggregation which can map different subtrees to different feature vectors. Figure 4 provides an overview of the proposed *DEMO-Net* framework on learning node and graph representation, which will be described in detail below.

4.1 Node Representation Learning

Let $h_{N(v)}$ denote the feature set $\{h_u|u \in N(v)\}$ within node v's neighborhood. Let $\mathcal{T} = \big\{\big(h_v, h_{N(v)}\big)\big\}$ be the set of subtrees consisting of the features of seed v and its 1-hop neighbors $N(v)$. To formalize our analysis, we first give the definition of structurally identical subtree below.

Definition 4.1. (Structurally Identical Subtree) Any two subtrees in $\mathcal{T}$ are structurally identical if the only possible difference between them is the order of neighbors.

The following lemma shows that graph neural networks could distinguish the local graph structures as well as the WL graph isomorphism test when **graph convolution is an injective function** that maps two subtrees in $\mathcal{T}$ to different features if they are not structurally identical.

LEMMA 4.2. *Let $G = \{V, E\}$ be a graph and $u, v \in V$ be two nodes in the graph. When the mapping function $f : \mathcal{T} \to \mathbb{R}^d$ in graph neural networks is injective, the learned features of v and u will be different if and only if the WL graph isomorphism test determines that they are not structurally identical.*

The feature aggregation of graph neural networks can be simply summarized as follows.

$$h_v^k = f(\{h_v^{k-1}, h_u^{k-1}|u \in N(v)\}) \tag{4}$$

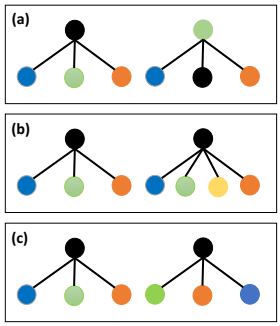

Figure 3: Examples of subtree in $\mathcal{T}$ with: (a) different seeds' attributes; (b) different seeds' degree values; (c) different neighbors' order. In such cases, two subtrees in (a) and (b) are mapped to different feature vectors, respectively. Two subtrees in (c) will be mapped to the same feature vector. (Best seen in color. Colors denote the node attributes.)

Obviously, most of the existing graph neural networks [9, 18] did not consider the injective aggregation function when learning node representation. From the perspective of WL isomorphism test, an injective graph convolution has the following properties.

Lemma 4.3. *(Properties) Let $f : \mathcal{T} \to \mathbb{R}^d$ be the aggregation function. If it is an injective function that maps any different subtrees in $\mathcal{T}$ to different feature vectors, then it has the following properties:*

(i) Seed-oriented: $f\big(h_i, \{h_u|u \in N(i)\}\big) \neq f\big(h_j, \{h_w|w \in N(j)\}\big)$ if the seeds' attributes are different, i.e., $h_i \neq h_j$.
(ii) Degree-aware: $f\big(h_i, \{h_u|u \in N(i)\}\big) \neq f\big(h_j, \{h_w|w \in N(j)\}\big)$ if the seeds' degree values are different, i.e., $deg(i) \neq deg(j)$.
(iii) Order-free: $f\big(h_i, \{h_u|u \in N(i)\}\big) = f\big(h_j, \{h_w|w \in N(j)\}\big)$ if $h_i = h_j$ and the only possible difference between $\{h_u|u \in N(i)\}$ and $\{h_w|w \in N(j)\}$ is the order of neighbors.

Figure 3 lists some examples to illustrate those properties. The injective function f maps the subtrees in Figure 3(a) to different features due to the distinctive seeds' attributes. Here, we hold that the subtree's structure properties are guided by seed node. Thus they are not structurally identical though both subtrees share the same leaf elements. Seeds' degree values also decide the subtree structure (shown in Figure 3(b)) because it is obvious that nodes with identical degree value share the similar structure. Figure 3(c) shows that neighbors' order will not change the subtree structure.

These properties will guide us to build a structure-specific graph neural network model. Based on properties (i) and (ii), the feature aggregation function in Eq. (4) can be expressed as follow.

$$h_v^k = f_s(h_v^{k-1}) \circ f_{deg(v)}\big(\{h_u^{k-1}|u \in N(v)\}\big) \tag{5}$$

where f_s and $f_{deg(v)}$ are seed-related and degree-specific mapping functions, respectively, and $deg(v)$ denotes the degree value of node v. All the nodes share one seed-oriented mapping function f_s, but have a degree-specific function for compressing node neighborhoods. Here, $\circ$ denotes the vector concatenation which combines the mapped features to form a single vector. If f_s and $f_{deg(v)}$ are injective, it will have the first two properties in Lemma 4.2 that subtrees with different seeds' features or degree values would be mapped differently. Additionally, the degree-specific mapping function $f_{deg(v)}$ should be symmetric[3] that is invariant to the order of neighbors. And we have the following theorem (proven in Appendix 7.1) to show the existence of mapping functions f_s and f_{deg}.

Theorem 4.4. *(Existence Theorem) Assume $\mathcal{T}$ is countable, there exist mapping functions f_s and $\{f_{deg}|deg \in degree(G)\}$ such that for any two subtrees in $\mathcal{T}$, the function $f : \mathcal{T} \to \mathbb{R}^d$ defined in Eq. (5) maps them to different features if they are not structurally identical.*

Next, we present our graph neural network model where the injective aggregation function could be approximated by multi-layer neural network due to its exceptional expression power. For seed-related mapping function $f_s(\cdot)$ in Eq. (5), we use a simple one-layer fully-connected neural network as follows.

$$f_s(h_v^{k-1}) = \sigma(W_0^k h_v^{k-1}) \tag{6}$$

where the trainable matrix W_0^k is shared by all the seeds at k^{th} hidden layer. Here $\sigma(\cdot)$ is a nonlinear activation function.

For degree-specific neighborhood aggregation on $h_{N(v)}$, it can be formulated as a multi-task feature learning problem (shown in Figure 4(b)(c)) in which each task represents node representation learning for nodes with a specific degree value, thus leading to preserving the degree-specific graph structure. Here, we present two schemes for this multi-task learning problem.

Degree-specific weight function: The degree-specific aggregation function can be expressed as follow.

$$f_{deg(v)}(h_{N(v)}^{k-1}) = \sigma\Big(\sum\nolimits_{u \in N(v)} (W_g^k + W_{deg(v)}^k) h_u^{k-1}\Big) \tag{7}$$

where $W_{deg(v)}^k$ is a degree-specific trainable matrix at k^{th} layer and W_g^k is a global trainable matrix shared by all the seeds.

Hashing function: Since the number of degree values on graphs could be very large, a critical challenge is how to perform multi-task learning efficiently. To address this challenge, hash kernel [19] (also called feature hashing or hash trick) is applied for our multi-task neighborhood learning problem. Given two vectors x and x', the hash map ϕ and the corresponding kernel $K_\phi(\cdot, \cdot)$ are defined:

$$\phi_i^{\xi_1, \xi_2}(x) = \sum\nolimits_{j:\xi_1(j)=i} \xi_2(j) x_j \tag{8}$$

$$K_\phi(x, x') = \langle x, x' \rangle_\phi = \Big\langle \phi^{\xi_1, \xi_2}(x), \phi^{\xi_1, \xi_2}(x') \Big\rangle \tag{9}$$

where ξ_1 and ξ_2 denote two hash functions such that $\xi_1 : \mathbb{N} \to \{1, \cdots, m\}$ and $\xi_2 : \mathbb{N} \to \{1, -1\}$. Notice that hash kernel is unbiased, i.e., $E_\phi[\langle x, x' \rangle_\phi] = \langle x, x' \rangle$ for any pair of input feature vectors. Let $w_{deg(v)}^k$ denote one of the row vectors in $W_{deg(v)}^k$, then we have $E_\phi\Big[\Big\langle w_{deg(v)}^k, h_u^{k-1} \Big\rangle_\phi\Big] = \Big\langle w_{deg(v)}^k, h_u^{k-1} \Big\rangle$. In this way, the multi-task feature aggregation function $f_{deg(v)}$ can be expressed as:

$$f_{deg(v)}(h_{N(v)}^{k-1}) = \sigma\Big(\sum_{u \in N(v)} W^k \big(\phi_g(h_u^{k-1}) + \phi_{deg}(h_u^{k-1})\big)\Big) \tag{10}$$

[3] A symmetric function of n variables is one whose value given n arguments is the same no matter the order of the arguments. For example, $f(x_1, x_2) = f(x_2, x_1)$ for any pair (x_1, x_2).

where $W^k = \phi_g(W_g^k) + \sum_{deg} \phi_{deg}(W_{deg}^k)$ is the trainable matrix shared by all the nodes, and $\phi_g(\cdot)$ and $\phi_{deg}(\cdot)$ are global and degree-specific hash maps, respectively.

One common assumption in multi-task learning is that all the tasks are related with some shared knowledge, and meanwhile have their own task-specific knowledge. As shown in Figure 3(b), two subtrees are structurally different, but they share some common leaves for neighborhood aggregation. By adopting both common (global) and task-specific (local) weight/hash functions, it allows learning the shared sub-structures and degree-specific neighborhood structures simultaneously.

There might be many different node degrees in real networks. One intuitive idea is that we could partition the degree values into several buckets to reduce the number of tasks. This heuristic solution might improve our model robustness to noisy graph structure or labeled nodes on source networks brought by human annotations [29, 30]. We leave this as our future work because hashing kernel [19] used in *DEMO-Net* is efficient to tackle large-scale multi-task learning problem.

4.2 Graph Representation Learning

The goal of graph representation learning is to use a compact feature vector to represent the entire graph. To this end, we provide a degree-specific graph-level pooling scheme.

When graph neural networks are going deeper, node representation actually captures the higher-order topological information within its local neighborhood. By mapping the original graph to a sequence of graphs $\{G_0, G_1, \cdots, G_K\}$ where G_0 denotes the original graph and $G_k (1 \le k \le K)$ represents the graph after the k^{th} layer of feature aggregation (as shown in Figure 4(e)-(g)), the k^{th} graph representation can be expressed as follow.

$$h_{G_k} = CONCAT\Big(\big\{\sum\nolimits_{v\in V} h_v^k \cdot \delta(deg(v), d)\big\} \big| d \in degree(G)\Big) \quad (11)$$

where $degree(G)$ denotes the set of degree values in graph G, and $\delta(\cdot,\cdot)$ is 1 when its two arguments are equal and 0 otherwise.

As discussed before, the node representation in G_k captures the topological information within k-hop neighborhood. In order to consider all the subtrees' information, we concatenate the representation h_{G_k} from all graphs $\{G_0, G_1, \cdots, G_K\}$:

$$h_G = CONCAT\big(\, h_{G_k} \,|k = 0, 1, \cdots, K\big) \quad (12)$$

Next, we compare the degree-specific pooling scheme with existing graph-level pooling methods [1, 22] and Weisfeiler-Lehman (WL) subtree kernel [16]. We define a degree-specific WL kernel:

$$\begin{aligned}\mathcal{K}_{DWL}(G_k, G_k') &= \big\langle \phi_{DWL}(G_k), \phi_{DWL}(G_k') \big\rangle \\ &= \sum_{v\in V}\sum_{v'\in V'} \delta(deg(v), deg(v')) \cdot \left\langle h_v^k, h_{v'}^k \right\rangle \end{aligned} \quad (13)$$

The corresponding mapping function is defined as:

$$\phi_{DWL}(G_k) = \big[c(G_k, d_1) \circ \cdots \circ c(G_k, d_{|degree(G_k)|})\big] \quad (14)$$

where d_i is the degree of G_k, and

$$c(G_k, d_i) = \sum\nolimits_{v\in V} h_v^k \cdot \delta(d_i, deg(v)) \quad (15)$$

As shown in [10], the non-linear activation function $\sigma(\cdot)$ has a mapping function $\phi_\sigma(\cdot)$ such that $\sigma(\mathbf{w}^T\mathbf{x}) = \big\langle \phi_{\sigma(\mathbf{x})}, \psi(\mathbf{w}) \big\rangle$ for some mapping $\psi(\mathbf{w})$ constructed from $\mathbf{w}$. By the following theorem (proven in Appendix 7.1), we show that our graph representation lies in a degree-specific Hilbert kernel space.

Table 2: Comparison of graph neural networks

Properties	*seed-oriented*	*degree-aware*	*order-free*
GCN [9]	×	×	✓
GAT [18]	✓	×	✓
GraphSAGE [5]	✓	×	–
DCNN [1]	×	×	✓
DEMO-Net	✓	✓	✓

Theorem 4.5. *For a degree-specific Weisfeiler-Lehman kernel, the graph representation h_G in Eq. (12) belongs to the Reproducing Kernel Hilbert Space (RKHS) of kernel $\mathcal{K}_{\sigma,DWL}(\cdot,\cdot)$ where*

$$\mathcal{K}_{\sigma,DWL}(G_k, G_k') = \big\langle \phi_\sigma(\phi_{DWL}(G_k)), \phi_\sigma(\phi_{DWL}(G_k')) \big\rangle \quad (16)$$

The sum/mean based graph-level pooling approaches make the learned graph representation lie in the kernel as follow.

$$\mathcal{K}_{MWL}(G_k, G_k') = \sum\nolimits_{v\in V}\sum\nolimits_{v'\in V'} \left\langle h_v^k, h_{v'}^k \right\rangle \quad (17)$$

And WL subtree Kernel [16] can be expressed as:

$$\mathcal{K}_{WLsubtree}(G_k, G_k') = \sum\nolimits_{v\in V}\sum\nolimits_{v'\in V'} \delta(h_v^k, h_{v'}^k) \quad (18)$$

It is easy to see that: (1) WL subtree kernel cannot be applied to measure the graph similarity when nodes have the continuous attribute vectors. (2) Our graph-level representation lies in a degree-specific kernel space comparing Eq. (13) with (17), thus leading to explicitly preserving the degree-specific graph structure.

4.3 Discussion

We compare the proposed *DEMO-Net* with some existing graph neural networks regarding the properties of graph convolution.

Lemma 4.3 shows that an injective aggregation function has three properties: *seed-oriented*, *degree-aware*, *order-free*. We summarize the properties of graph convolution of GCN [9], GAT [18], GraphSAGE [5], and DCNN [1] in Table 2. It can be seen that: (1) The existing graph neural networks do not have all the three properties. More importantly, none of them capture the degree-specific graph structures. (2) For graphSAGE, it is *order-free* when using mean or max aggregator. But graphSAGE with LTSM-aggregator is not *order-free* because it considers the node neighborhood as an ordered sequence. (3) Our proposed *DEMO-Net* considers all the properties, and the *degree-aware* property in particular allows our model to explicitly preserve the neighborhood structures for node and graph representation learning. In addition, the time complexity of graph convolution of *DEMO-Net* is linear with respect to the number of nodes and edges.

5 EXPERIMENTAL RESULTS

In this section, we present the experimental results on real networks. In particular, we focus on answering the following questions:
Q1: Is the proposed *DEMO-Net* algorithm effective on node classification compared to the state-of-the-art graph neural networks?
Q2: How does the proposed *DEMO-Net* perform on identifying graph structure compared to structure-aware embedding approaches?

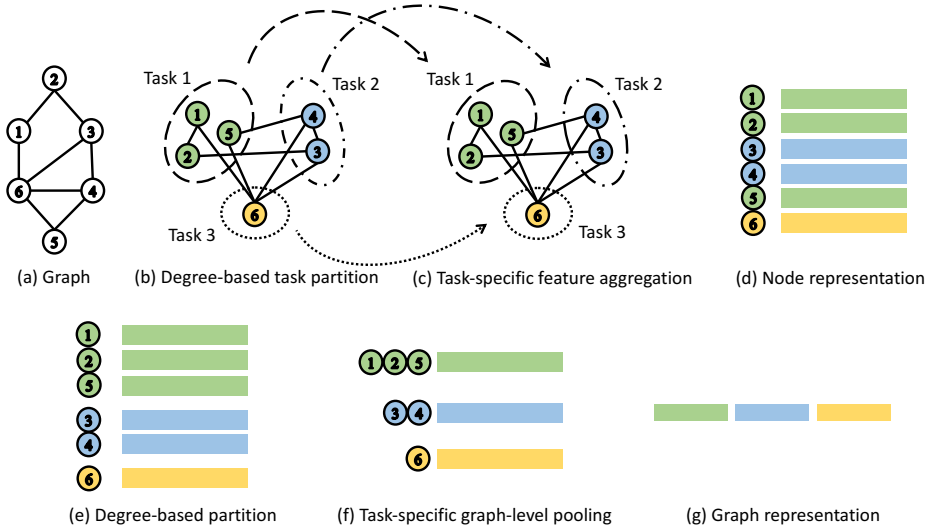

Figure 4: Overview of our proposed *DEMO-Net* framework (best seen in color). (b) and (c) represent the multi-task feature aggregation. The node representation in (d) can be used for node-level classification. For learning graph embedding, (e)-(g) provide the graph-level pooling method based on node degree distribution for learning graph representation.

Q3: How does the proposed *DEMO-Net* with degree-specific graph-level pooling perform on graph classification task?
Q4: Is the proposed degree-specific graph convolution of *DEMO-Net* efficient on learning node representation?

5.1 Experiment Setup

Data Sets: We use seven node classification data sets, including four social networks and three air-traffic networks. Facebook, Wiki-Vote [11], BlogCatalog and Flickr[4] are social networks. The posted keywords or tags in BlogCatalog and Flickr networks are used as node attribute information. There are three air-traffic networks [14]: Brazil, Europe and USA, where each node corresponds to an airport and edge indicates the existence of commercial flights between the airports. Their class labels are assigned based on the level of activity measured by flights or people that passed the airports. Data statistics are summarized in Table 3. For those networks without node attributes, we use the one-hot encoding of node degrees. In BlogCatalog, Flickr and other air-traffic networks, node class labels are available. In Facebook and Wiki-Vote, we use the degree-induced class labels by labeling the node according to its degree value.

In addition, we use four bioinformatics networks to evaluate the model performance on graph classification, including MUTAG, PTC, PROTEINS and ENZYMES[5] where the nodes are associated with categorical input features. The detailed statistics for these bioinformatics networks are summarized in Table 4.

Model Configuration: We adopt two hidden layers followed by the softmax activation layer in *DEMO-Net*, where the proposed multi-task feature learning schemes in Eq. (7) and (10) are applied to each hidden layer for neighborhood aggregation (termed as *DEMO-Net*(weight) and *DEMO-Net*(hash), respectively). In addition, we apply Adam optimizer [8] with the learning rate 0.005 on the cross-entropy loss to train our models. To prevent our models from over-fitting, we adopt the dropout [17] with $p = 0.6$ and L_2 regularization with $\lambda = 0.0005$. The hidden layer size of neural units is set as 64. An early stopping strategy with a patience of 100 epochs on validation set is applied in our experiments.

Table 3: Data sets for node classification

Data sets	# nodes	# edges	# classes	# attributes
Facebook	4039	88234	4	-
Wiki-Vote	7115	103689	4	-
BlogCatalog	5196	171743	6	8189
Flickr	7575	239738	9	12047
Brazil	131	1038	4	-
Europe	399	5995	4	-
USA	1190	13599	4	-

Baseline Methods: The baseline methods used in our experiments are given below: (1) node-level graph neural networks: GCN [9], GCN_cheby [9], GraphSAGE (mean aggregator) [5], Union [12], Intersection [12] and GAT [18]; (2) node-level structure-aware embedding approaches: RolX [7], struc2vec [14] and GraphWAVE [3]; (3) graph-level graph neural networks: DCNN [1], PATCHY-SAN [13] and DIFFPOOL [24]; (4) deep graph kernel: DeepWL [23]. In our experiments, all the baseline models used the default hyperparameters suggested in the original papers.

All our experiments are performed on a Windows machine with four 3.60GHz Intel Cores and 32GB RAM. The source code will be available at https://github.com/jwu4sml/DEMO-Net.

5.2 Node Classification

For a fair comparison of different architectures [15], we use different train/validation/test splits of the networks on node classification. For social networks, we randomly choose 10% and 20% of the graph

[4]http://people.tamu.edu/~xhuang/Code.html
[5]https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

Table 4: Data sets for graph classification

Data sets	# graphs	# classes	Avg # nodes	# attributes
MUTAG	188	2	17.9	7
PTC	344	2	25.5	19
PROTEINS	1113	2	39.1	3
ENZYMES	600	6	32.6	3

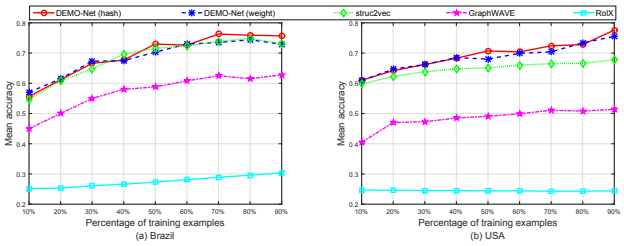

Figure 5: Node-level classification accuracy on the Brazil and USA air-traffic networks using different train/test splits

nodes as the training and validation set, respectively, and the rest as the test set. For air-traffic networks, the training, validation and test sets are randomly assigned with equal number of nodes. We run 10 times and report the mean accuracy with the standard variance for performance comparison. As shown in Table 5, we report the classification results on the real networks where the best results are indicated in bold. It can be observed that the proposed *DEMO-Net*models significantly outperform other graph neural networks (answering **Q1**). In particular, our *DEMO-Net* models are at least 10% higher on mean accuracy over baseline methods. One explanation is that baseline methods focus on preserving the node proximity by roughly mixing a node with its neighbors, whereas our proposed *DEMO-Net* models capture the degree-specific structure to distinguish the structural roles of nodes in the networks.

We also evaluate the performance of our models against three structure-aware embedding approaches: RolX, struc2vec and GraphWAVE. All of them are unsupervised embedding approaches identifying the structural roles of nodes in the networks. Following [14], we use the one-vs-rest logistic regression with L2 regularization to train a classifier for node representations learned by baseline methods. Here we consider using different train-test splits where the percentage of training nodes ranges from 10% to 90% and the rest is used for testing. The experimental results on the Brazil and USA air-traffic networks are provided in Figure 5. We observe that our proposed *DEMO-Net* models outperform the comparison methods across all the data sets (answering **Q2**). Besides, the structure roles identified by those baselines only represent the local graph structure without considering node attributes. Instead, both topological information and node attributes are captured in our *DEMO-Net* models when learning node representation.

5.3 Graph Classification

We use four public graph classification benchmarks to evaluate the proposed *DEMO-Net* models with the degree-specific graph-level pooling scheme. DCNN [1], PATCHY-SAN [13] and DIFFPOOL [24] adopted the end-to-end training architectures for supervised graph classification. For unsupervised graph kernel method DeepWL [23], we use the one-vs-rest logistic regression with L2 regularization

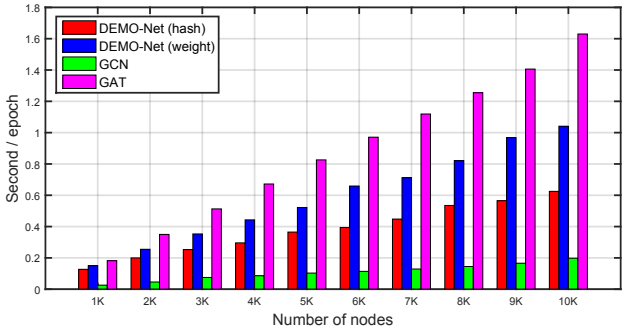

Figure 6: Running time per epoch (best seen in color)

to train a supervised classifier for graph classification. We also consider our model variants (denoted as *DEMO-Net*_m(hash) and *DEMO-Net*_m(weight) respectively) which replace the proposed degree-specific graph-level pooling with mean-pooling scheme [1]. The input graphs are randomly assigned to the training, validation, or test set where each set has the same number of nodes.

The graph classification results are shown in Table 6 where the best results are indicated in bold. It is observed that (1) compared to the existing mean-pooling method, the proposed degree-specific pooling method improves the model performance in most cases, which is consistent with our analysis in Section 4.2; (2) the classification results of our *DEMO-Net* models are comparable to other graph neural networks and graph kernel method (answering **Q3**). Moreover, on MUTAG and ENZYMES data sets, our proposed *DEMO-Net*(weight) outperforms the baseline methods. One explanation might be that the graph representation generated by *DEMO-Net* explicitly preserves the degree-specific graph structure information.

5.4 Efficiency Analysis

It is easy to show that the time complexity of each layer in our proposed *DEMO-Net*(hash) model is $O(nFF' + TF + nHF' + mF')$ where n and m are the number of nodes and edges in the graph, respectively, F and F' are the dimensionalities of input and output features at each layer, respectively, T is the number of tasks (degree values) in the graph, and H is the hashing dimension. By observing that $T \leq n$ in the networks, its time complexity would be $O(nFF' + mF')$, which is on par with GCN and GAT models. Similarly, we can show that the time complexity of each layer in *DEMO-Net*(weight) is $O(T(nFF' + mF'))$. When $T \ll n$ and $T \ll m$, it also scales linearly with respect to the number of nodes and edges.

Following [9], we report the running time (measured in seconds wall-clock time) per epoch (including forward pass, cross-entropy calculation, backward pass) on a synthetic network assigning $2n$ edges uniformly at random. As shown in Figure 6, we observe that (answering **Q4**) (1) the wall-clock time of our proposed *DEMO-Net* model is linear with respect to the number of nodes; (2) our models are much more efficient than GAT on node classification task.

6 CONCLUSIONS

In this paper, we focus on building a degree-specific graph neural network for both node and graph classification. We start by

Table 5: Node-level classification accuracy (mean ± standard variance) on the social and air-traffic networks

	Social networks				Air-traffic networks		
	Facebook	Wiki-Vote	BlogCatalog	Flickr	Brazil	Europe	USA
GraphSAGE [5]	0.389 ±0.019	0.245 ±0.000	0.828 ±0.007	0.641 ±0.006	0.404 ±0.035	0.272 ±0.022	0.316 ±0.022
GCN [9]	0.575 ±0.013	0.329 ±0.029	0.720 ±0.013	0.546 ±0.019	0.432 ±0.064	0.371 ±0.046	0.432 ±0.022
GCN_cheby [9]	0.646 ±0.012	0.495 ±0.016	0.686 ±0.037	0.479 ±0.023	0.516 ±0.070	0.460 ±0.038	0.526 ±0.045
Union [12]	0.600 ±0.000	0.463 ±0.000	0.730 ±0.000	0.566 ±0.000	0.466 ±0.006	0.418 ±0.002	0.582 ±0.000
Intersection [12]	0.598 ±0.000	0.462 ±0.000	0.725 ±0.000	0.557 ±0.000	0.459 ±0.003	0.443 ±0.002	0.573 ±0.000
GAT [18]	0.570 ±0.036	0.594 ±0.070	0.663 ±0.000	0.359 ±0.000	0.382 ±0.126	0.424 ±0.073	0.585 ±0.021
DEMO-Net(hash)	0.887 ±0.020	0.997 ±0.000	0.849 ±0.006	**0.678 ±0.010**	**0.614 ±0.069**	**0.479 ±0.064**	**0.659 ±0.020**
DEMO-Net(weight)	**0.919 ±0.003**	**0.998 ±0.000**	**0.849 ±0.000**	0.656 ±0.000	0.543 ±0.034	0.459 ±0.025	0.647 ±0.021

Table 6: Graph-level classification on the real networks

	MUTAG	PTC	PROTEINS	ENZYMES
DeepWL [23]	0.733	0.537	0.680	0.210
DCNN [1]	0.670	0.572	0.579	0.160
PATCHY-SAN [13]	0.795	0.568	0.714	0.170
DIFFPOOL [24]	0.663	0.251	**0.733**	0.184
*DEMO-Net*_m(hash)	0.760	**0.586**	0.617	0.236
*DEMO-Net*_m(weight)	0.798	0.550	0.616	0.251
DEMO-Net(hash)	0.771	0.563	0.705	0.251
DEMO-Net(weight)	**0.814**	0.572	0.708	**0.272**

analyzing the limitations of the existing graph neural networks from the perspective of Weisfeiler-Lehman graph isomorphism test. Furthermore, it is observed that the graph convolution should have the following properties: *seed-oriented*, *degree-aware*, *order-free*. To this end, we propose a generic graph neural network model named *DEMO-Net* which formulates the feature aggregation into a multi-task learning problem according to nodes' degree values. In addition, we also present a novel graph-level pooling method for learning graph representations provably lying in a degree-specific Hilbert kernel space. The extensive experiments on real networks demonstrate the effectiveness of our *DEMO-Net* algorithm.

ACKNOWLEDGMENTS

This work is supported by the United States Air Force and DARPA under contract number FA8750-17-C-0153, National Science Foundation under Grant No. IIS-1552654, Grant No. IIS-1813464 and Grant No. CNS-1629888, the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-02-00, and an IBM Faculty Award. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *NIPS*.
[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.
[3] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *SIGKDD*.
[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.
[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.
[6] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).
[7] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RolX: structural role extraction & mining in large graphs. In *SIGKDD*.
[8] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[9] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[10] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2017. Deriving neural architectures from sequence and graph kernels. In *ICML*.
[11] Jure Leskovec and Andrej Krevl. 2014. SNAP datasets: Stanford large network dataset Collection. http://snap.stanford.edu/data.
[12] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.
[13] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.
[14] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *SIGKDD*.
[15] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. In *NIPS*.
[16] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research* (2011), 2539–2561.
[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
[19] Kilian Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alex Smola. 2009. Feature hashing for large scale multitask learning. In *ICML*.
[20] Boris Weisfeiler and AA Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia* 2, 9 (1968), 12–16.
[21] Mengmeng Wu, Wanwen Zeng, Wenqiang Liu, Hairong Lv, Ting Chen, and Rui Jiang. 2018. Leveraging multiple gene networks to prioritize GWAS candidate genes via network representation learning. *Methods* (2018).
[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *ICLR*.
[23] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *SIGKDD*.
[24] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NIPS*.
[25] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*.
[26] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2018. Graph convolutional networks: Algorithms, applications and open challenges. In *International Conference on Computational Social Networks*. Springer, 79–91.
[27] Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. 2018. SPARC: Self-paced network representation for few-shot rare category characterization. In *SIGKDD*.
[28] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A local algorithm for structure-preserving graph cut. In *SIGKDD*. 655–664.
[29] Yao Zhou and Jingrui He. 2016. Crowdsourcing via Tensor Augmentation and Completion.. In *IJCAI*. 2435–2441.
[30] Yao Zhou, Lei Ying, and Jingrui He. 2017. MultiC^2: an optimization framework for learning from task and worker dual heterogeneity. In *SDM*. 579–587.

7 APPENDIX FOR REPRODUCIBILITY

To better reproduce the experimental results, we provide additional details about the algorithms (Section 7.1) and experimental results (Section 7.2).

7.1 Algorithm Analysis

Proof of Theorem 4.4. Theorem 4.4 says that there exist mapping functions f_s and $\{f_{deg}|deg \in degree(G)\}$ such that for any two subtrees in $\mathcal{T}$, the function $f : \mathcal{T} \to \mathbb{R}^d$ defined in Eq. (5) maps them to different feature vectors if they are not structurally identical.

Proof. Let $\mathcal{T}_s = \{h_v|v \in V\}$ denote the seed set in $\mathcal{T}$ and $d_m = max\{deg\} + 1$ the maximum degree values plus one. Becuase $\mathcal{T}$ is countable, there exists an injective function $Z : \mathcal{T} \to \mathbb{N}$ that maps each subtree from $\mathcal{T}$ to an unique natural number. It can be observed that $\mathbb{N}$ can be divided into d_m disjoint sets: $\mathbb{N}_0 = \{i*d_m\}_{i=0}^{\infty}, \mathbb{N}_1 = \{i*d_m+1\}_{i=0}^{\infty}, \cdots, \mathbb{N}_{d_m-1} = \{i*d_m+d_m-1\}_{i=0}^{\infty}$.

There exists an injective function $Z_s : \mathcal{T}_s \to \mathbb{N}_0$ that maps each seed from $\mathcal{T}_s$ to an unique natural number in $\mathbb{N}_0$. Let $\mathcal{T}_i = \{h_{N(v)}|v \in V \text{ and } deg(v) = i\}$ denote the neighbor set consisting of the seeds' neighbors when their degree values are equal to i. Because $\mathcal{T}$ is countable, all the subsets $\mathcal{T}_i$ $(1 \le i \le max\{deg\})$ are countable. There exists the injective, symmetric function $Z_i : \mathcal{T}_i \to \mathbb{N}_i$ that maps each element from $\mathcal{T}_i$ to an unique real number in $\mathbb{N}_i$. Moreover, there is a function $Z_f : \mathcal{T} \to \mathbb{N}^2$ that maps each subtree from $\mathcal{T}$ to an unique feature vector in $\mathbb{N}^2$ when $Z_f(h_v, h_{N(v)}) = Z_s(h_v) \circ Z_{deg(v)}(h_{N(v)})$. Please note that the structurally identical subtrees would be considered the same one when the degree-specific function Z_i is symmetric.

It is easy to construct an injective function $g : \mathbb{N}^2 \to \mathbb{R}^d$. Based on the properties of injective function, $g(Z_f(\cdot))$ will be injective function that maps any two subtrees in $\mathcal{T}$ to different feature vectors in $\mathbb{R}^d$ if they are not structurally identical, which completes the proof. □

Proof of Theorem 4.5. Theorem 4.5 says that the graph representation h_G learned in Eq. (12) belongs to the Reproducing Kernel Hilbert Space (RKHS) of kernel $\mathcal{K}_{\sigma,DWL}(\cdot,\cdot)$.

Proof. Let $h_{G_k}[i] = \sum_{v \in V} h_v^k \cdot \delta(deg(v), d_i)$ denote the feature vector of graph G_k for nodes with degree value d_i. Let $h_G[k,i,j] = h_{G_k}[i][j]$ denote the j^{th} element of $h_{G_k}[i]$. Our graph convolution (feature aggregation) function can be written as:

$$h_v^k = \sigma(W_0^k h_v^{k-1} \circ \sum_{u \in N(v)} \hat{W}_{deg(v)}^k h_u^{k-1}) \tag{19}$$

where $\hat{W}_{deg(v)}^k$ represents the degree-specific parameters, and more specifically, $\hat{W}_{deg(v)} = W_g^k + W_{deg(v)}^k$ for degree-specific weight matrix in Eq. (7) and $\hat{W}_{deg(v)} = W^k(\phi_g(\cdot)+\phi_{deg}(\cdot))$. Because we use the concatenation operator $\circ$ to combine the learned features of seed and its neighborhood, it holds that $h_{G_k}[i][j]$ lies in **either** seed's feature $\sigma(W_0^k h_v^{k-1})$ **or** $\sigma(\sum_{u \in N(v)} \hat{W}_{deg(v)}^k h_u^{k-1})$, but not both.

Let w_{0j} denote the j^{th} row from W_0^k. To show our results, we construct a k-regular "reference graph" $G_{r_k} = (V_{r_k}, E_{r_k})$ which has the same nodes as the input graph G (i.e., $V_{r_k} = V$). Its degree value k is d_i and each node in "reference graph" is associated with the same feature vector w_{0j}/n. Then when $h_{G_k}[i][j]$ lies in the seed's feature $\sigma(W_0^k h_v^{k-1})$, we have:

$$\begin{aligned} &h_G[k,i,j] = h_{G_k}[i][j] \\ =&\sum_{v \in V} h_v^k[j] \cdot \delta(deg(v), d_i) \\ =&\sigma\Big(\sum_{v \in V} \delta(deg(v), d_i) \cdot \Big\langle h_v^{k-1}, w_{0j} \Big\rangle\Big) \\ =&\sigma\Big(\frac{1}{n}\sum_{v \in V}\sum_{r \in V_{r_k}} \delta(deg(v), deg(r)) \cdot \Big\langle h_v^{k-1}, w_{0j} \Big\rangle\Big) \\ =&\sigma\Big(\mathcal{K}_{DWL}(G_{k-1}, G_{r_k})\Big) \end{aligned} \tag{20}$$

The lemma 1 in [10] holds that for activation functions σ, there exists kernel functions $\mathcal{K}_\sigma(\cdot,\cdot)$ and the underlying mapping $\phi_\sigma(\cdot)$ such that $f(x) = \sigma(w^T x) = \langle \phi_\sigma(x), \psi(w) \rangle$ for some mapping function $\psi(w)$ constructed from w. Therefore, we have:

$$h_G[k,i,j] = \mathcal{K}_{\sigma,DWL}(G_{k-1}, G_{\sigma,r_k}) \tag{21}$$

where $\mathcal{K}_{\sigma,DWL}(\cdot,\cdot)$ is the composition of $\mathcal{K}_\sigma(\cdot,\cdot)$ and $\mathcal{K}_{DWL}(\cdot,\cdot)$, and $\mathcal{K}_{\sigma,DWL}(x,y) = \phi_\sigma(\phi_{DWL}(x))^T \phi_\sigma(\phi_{DWL}(y))$. And G_{σ,r_k} is the "reference graph" constructed from model parameters and activation function.

Let $\hat{w}_{d_i j}$ denote the j^{th} row from $\hat{W}_{deg(v)}^k$ with $deg(v) = d_i$. Similarly, we construct a k-regular "reference graph" $\hat{G}_{r_k} = (\hat{V}_{r_k}, \hat{E}_{r_k})$ which has the same nodes as the input graph G with degree value d_i. Each node in this "reference graph" is associated with the same feature vector $\hat{w}_{d_i j}/n$. when when $h_{G_k}[i][j]$ lies in the neighborhood's feature $\sigma(\sum_{u \in N(v)} \hat{W}_{deg(v)}^k h_u^{k-1})$, we have:

$$\begin{aligned} &h_G[k,i,j] = h_{G_k}[i][j] \\ =&\sum_{v \in V} h_v^k[j] \cdot \delta(deg(v), d_i) \\ =&\sigma\Big(\sum_{v \in V}\sum_{u \in N(v)} \delta(deg(v), d_i) \cdot \Big\langle h_u^{k-1}, \hat{w}_{d_i j} \Big\rangle\Big) \\ =&\sigma\Big(\frac{1}{n}\sum_{v \in V}\sum_{r \in \hat{V}_{r_k}} \delta(deg(v), deg(r)) \cdot \Big\langle \sum_{u \in N(v)} h_u^{k-1}, \hat{w}_{d_i j} \Big\rangle\Big) \\ =&\sigma\Big(\mathcal{K}_{DWL}(G_{k-1}, \hat{G}_{r_k})\Big) \end{aligned} \tag{22}$$

Please notice that in this case, node features are assumed to be the sum of neighborhood features. And moreover, it can be written as:

$$h_G[k,i,j] = \mathcal{K}_{\sigma,DWL}(G_{k-1}, \hat{G}_{\sigma,r_k}) \tag{23}$$

where $\hat{G}_{\sigma,r_k}$ is the "reference graph" constructed from model parameters and activation function. Therefore, the graph representation h_G belongs to the RKHS of kernel $\mathcal{K}_{\sigma,DWL}(\cdot,\cdot)$, which completes the proof. □