RECURSIVE NEIGHBORHOOD POOLING FOR GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

While massage passing based Graph Neural Networks (GNNs) have become increasingly popular architectures for learning with graphs, recent works have revealed important shortcomings in their expressive power. In response, several higher-order GNNs have been proposed, which substantially increase the expressive power, but at a large computational cost. Motivated by this gap, we introduce and analyze a new recursive pooling technique of local neighborhoods that allows different tradeoffs of computational cost and expressive power. First, we show that this model can count subgraphs of size k, and thereby overcomes a known limitation of low-order GNNs. Second, we prove that, in several cases, RNP-GNNs can greatly reduce computational complexity compared to the existing higher-order k-GNN and Local Relational Pooling (LRP) networks.

1 INTRODUCTION

Graph Neural Networks (GNNs) are powerful tools for graph representation learning (Scarselli et al., 2008; Kipf & Welling, 2017; Hamilton et al., 2017), and have been successfully used in applications such as encoding molecules, simulating physics, social network analysis, knowledge graphs, and many others (Duvenaud et al., 2015; Defferrard et al., 2016; Battaglia et al., 2016; Jin et al., 2018). An important class of GNNs is the set of Message Passing Graph Neural Networks (MPNNs) (Gilmer et al., 2017; Kipf & Welling, 2017; Hamilton et al., 2017; Xu et al., 2018; Scarselli et al., 2008), which follow an iterative message passing scheme to compute a graph representation.

Despite the empirical success of MPNNs, their expressive power has been shown to be limited. For example, their discriminative power, at best, corresponds to the one-dimensional Weisfeiler-Leman (1-WL) graph isomorphism test (Xu et al., 2018; Morris et al., 2019), so they cannot, e.g., distinguish regular graphs. Moreover, they also cannot count any induced subgraph with at least three vertices (Chen et al., 2020)), or learn graph parameters such as clique information, diameter, or shortest cycle (Garg et al., 2020). Still, in several applications e.g. in computational chemistry, materials design or pharmacy (Elton et al., 2019; Sun et al., 2020), we aim to learn functions that depend on the presence or count of specific substructures.

To strengthen the expressive power of GNNs, higher-order representations such as $k-$GNNs (Morris et al., 2019) and $k-$Invariant Graph Networks ($k-$IGNs) (Maron et al., 2019) have been proposed. $k-$GNNs are inspired by the k-dimensional WL ($k-$WL) graph isomorphism test, a message passing algorithm on $k-$tuples of nodes, and $k-$IGNs are based on equivariant linear layers of a feed-forward neural networks applied on the input graph as a matrix, and are at least as powerful as $k-$GNN. These models are provably more powerful than MPNNs and can, e.g., count any induced substructure with at most k vertices. But, this power comes at the computational cost of $O(n^k)$ operations for n nodes. The necessary tradeoffs between expressive power and computational complexity are still an open question.

The expressive power of a GNN is often measured in terms of a hierarchy of graph isomorphism tests, i.e., by comparing it to a $k-$WL test. Yet, there is limited knowledge about how the expressive power of higher order graph isomorphism tests relates to various functions of interest (Arvind et al., 2020). A different approach is to take the perspective of specific functions that are of practical interest, and quantify a GNN's expressive power via those. Here, we focus on counting induced substructures to measure the power of a GNN, as proposed in (Chen et al., 2020). In particular, we

study whether it is possible to count given substructures with a GNN whose complexity is between that of MPNNs and the existing higher-order GNNs.

To this end, we study the scheme of many higher-order GNNs (Morris et al., 2019; Chen et al., 2020): select a collection of subgraphs of the input graph, encode these, and (possibly iteratively) compute a learned function on this collection. First, we propose a new such class of GNNs, Recursive Neighborhood Pooling Graph Neural Networks (RNP-GNNs). Specifically, RNP-GNNs represent each node by a representation of its neighborhood of a specific radius. Importantly, this neighborhood representation is computed *recursively* from its subgraphs. As we show, RNP-GNNs can count any induced substructure with at most k vertices. Moreover, for any set of substructures with at most k vertices, there is a specifiable RNP-GNN that can count them. This flexibility allows to design a GNN that is adapted to the power needed for the task of interest, in terms of counting (induced) substructures.

The Local Relational Pooling (LRP) architecture was also introduced with the goal of counting substructures (Chen et al., 2020). While it can do so, it is polynomial-time only if the encoded neighborhoods are of size $o(\log(n))$. In contrast, RNP-GNNs use almost linear operations, i.e., $n^{1+o(1)}$, if the size of encoded neighborhoods is $n^{o(1)}$. This is an exponential theoretical improvement in the tolerable size of neighborhoods, and a significant improvement over the complexity of $O(n^k)$ in $k-$GNN and $k-$IGN).

Finally, we take a broader perspective and provide an information-theoretic lower bound on the complexity of a general class of GNNs that can provably count substructures with at most k vertices. This class includes GNNs that represent a given graph by aggregating a number of encoded graphs, where the encoded graphs are related to the given graph with an arbitrary function.

In short, in this paper, we make the following contributions:

1. We introduce Recursive Neighborhood Pooling Graph Neural Networks (RNP-GNNs), a flexible class of higher-order graph neural networks, that provably allow to design graph representation networks with any expressive power of interest, in terms of counting (induced) substructures.
2. We show that RNP-GNNs offer computational gains over existing models that count substructures: an exponential improvement in terms of the "tolerable" size of the encoded neighborhoods compared to LRP networks, and much less complexity in sparse graphs compared to $k-$GNN and $k-$IGN.
3. We provide an information-theoretic lower bound on the complexity of a general class of GNN that can count (induced) substructures with at most k vertices.

2 BACKGROUND

Message Passing Graph Neural Networks. Let $G = (\mathcal{V}, \mathcal{E}, X)$ be a labeled graph with $|\mathcal{V}| = n$ nodes. Here, $X_v \in \mathcal{X}$ denotes the initial label of $v \in \mathcal{V}$, where $\mathcal{X} \subseteq \mathbb{N}$ is a (countable) domain.

A typical message passing Graph Neural Network (MPNN) first computes a representation of each node, and then aggregates the node representations via a readout function into a representation of the entire graph G. The representation $h_v^{(i)}$ of each node $v \in \mathcal{V}$ is computed iteratively by aggregating the representations $h_u^{(i-1)}$ of the neighboring nodes u:

$$m_v^{(i)} = \text{AGGREGATE}^{(i)}\Big(\{\{h_u^{(i-1)} : u \in \mathcal{N}_v\}\}\Big), \tag{1}$$

$$h_v^{(i)} = \text{COMBINE}^{(i)}\Big(h_v^{(i-1)}, m_v^{(i)}\Big), \tag{2}$$

for any $v \in \mathcal{V}$, for k iterations, and with $h_v^{(0)} = X_v$. The AGGREGATE/COMBINE functions are parametrized, learnable functions, and $\{\{.\}\}$ denotes a multi-set, i.e., a set with (possibly) repeating elements. A graph-level representation can be computed as

$$h_G = \text{READOUT}\Big(h_v^{(k)} : v \in \mathcal{V}\Big), \tag{3}$$

where READOUT is a (learnable) function. For representational power, it is important that the learnable functions above are injective, which can be achieved, e.g., if the AGGREGATE function is a summation and COMBINE is a weighted sum concatenated with an MLP (Xu et al. (2018)).

Higher-Order GNNs. To increase the representational power of GNNs, several higher-order GNNs have been proposed. In a $k-$*GNN*, a message passing algorithm is applied to the $k-$tuples of nodes, in a similar fashion as GNNs do on nodes (Morris et al., 2019). At initialization, each $k-$tuple is labeled with its type, that is, two $k-$tuples are labeled differently if their induced subgraphs are not isomorphic. As a result, $k-$GNNs can count induced substructures with at most k vertices even at initialization. Another class of higher-order networks are $k-$*IGNs*, which are constructed with linear invariant/equivariant feed-forward layers, whose inputs consider graphs via adjacency matrices (Maron et al., 2019). The $k-$IGNs are at least as powerful as $k-$GNNs, and hence they too can count substructures with at most k vertices. However, both methods need $O(n^k)$ operations.

Specifically for counting substructures, Chen et al. (2020) propose *Local Relational Pooling (LRP)* networks. LRPs apply Relational Pooling (RP) networks (Murphy et al., 2019) on the neighborhoods around each node. RP networks use permutation-variant functions and convert them to a permutation-invariant function by summing over all permutations. This summation is computationally expensive.

3 Other Related Works

Expressive power. Several other works have studied the expressive power of GNNs (Azizian & Lelarge, 2020). Scarselli et al. (2009) extended universal approximation from feedforward networks to GNNs, using the notion of *unfolding equivalence*. Chen et al. (2019) establish an equivalence between the graph isomorphism problem and the power to approximate permutation invariant functions on graphs. Maron et al. (2019) and Keriven & Peyré (2019) propose higher-order, tensor-based GNN models that provably achieve universal approximation of permutation-invariant functions on graphs, Loukas (2019) studies expressive power under depth and width restrictions. Studying GNNs from the perspective of local algorithms, Sato et al. (2019) show that GNNs can approximate solutions to certain combinatorial optimization problems.

Neighborhoods/Subgraphs in GNNs. Several ideas for extending/using GNNs are made based on exploring subgraphs around each node/link. For example, for link prediction applications based on GNNs, we also need to explore subgraphs around a link as suggested in (Zhang & Chen (2018)). The idea to explore subgraphs around each node to strengthen the GNNs is defined in its most general form in (Li et al. (2019)), where the authors suggest to aggregate on local neighborhoods with arbitrary radius. However, no specific way is introduced in that work to how perform such an aggregation, unless some known results on based on paths/random walks. A novel method based on applying GNN, then clustering, and applying another GNN on subgraphs is also proposed (Ying et al. (2018)). For graph comparison (i.e., testing whether a given possibly large subgraph exists in the given model), it is also suggested to compare the GNN outputs constructed based on their embedded subgraphs with small radii (Ying et al. (2020)).

To take local information into consideration, it is suggested to count, and add them to the network as features (Bouritsas et al. (2020)). However, this kernel-based method simply fails in practice since we need extensive search even for small subgraphs. Another related work is (Vignac et al. (2020)), where a method is proposed to boost MPNNs by introducing the local context matrix around each node, and trying to learn it during the procedure.

4 Recursive Neighborhood Pooling

Next, we construct Recursive Neighborhood Pooling Graph Neural Networks (RNP-GNNs), GNNs that can count any set of induced substructures of interest, with lower complexity than previous models. We represent each node by a representation of its radius r_1-neighborhood, and then combine these representations. The key question is how to encode these local neighborhoods in a vector representation. To do so, we introduce a new idea: we view local neighborhoods as small subgraphs, and recursively apply our model to encode these neighborhood subgraphs. When encoding the local subgraphs, we use a different radius r_2, and, recursively, a sequence of radii $(r_1, r_2, \ldots, r_t) \in \mathbb{N}^t$ to obtain the final representation $h_v^{(t)}$ of nodes after t recursion steps.

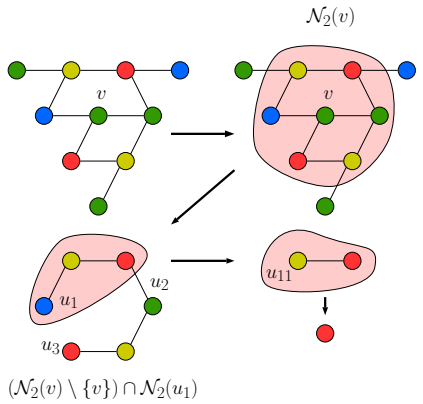

Figure 1. Illustration of a Recursive Neighborhood Pooling GNN (RNP-GNN) with recursion parameters $(2, 2, 1)$. To compute the representation of node v in the given graph (depicted in the top left), we first recurse on $G(\mathcal{N}_2(v) \setminus \{v\})$ (top right). To do so, we find the representation of each node $u \in G(\mathcal{N}_2(v) \setminus \{v\})$. For instance, to compute the representation of u_1, we apply an RNP-GNN with recursion parameters $(2, 1)$ and aggregate on $G((\mathcal{N}_2(v) \setminus \{v\}) \cap (\mathcal{N}_2(u_1) \setminus \{u_1\}))$, shown in the bottom left. To do so, we recursively apply an RNP-GNN with recursion parameters (1) on $G((\mathcal{N}_2(v) \setminus \{v\}) \cap (\mathcal{N}_2(u_1) \setminus \{u_1\}) \cap (\mathcal{N}_1(u_{11}) \setminus \{u_{11}\}))$, in the bottom right.

While MPNNs also encode a representation of a local subgraph of certain radius, the recursive representations differ as they essentially take into account intersections of subgraphs. As a result, as we will see in Section 5.1, they retain more structural information and are more expressive. Models such as k-GNN and LRP also compute encodings of subgraphs, and then update the resulting representations via message passing. We can do the same with the neighborhood representations computed by RNP-GNNs to encode more global information, although our representation results in Section 5.1 hold even without that. In Section 6, we will compare the computational complexity of RNP-GNN and these other models.

Formally, an RNP-GNN is a parametrized learnable function $f(.;\theta) : \mathbb{G}_n \to \mathbb{R}^d$, where $\mathbb{G}_n$ is the set of all labeled graphs on n vertices. Let $G = (\mathcal{V}, \mathcal{E}, X)$ be a labeled graph with $|\mathcal{V}| = n$ nodes, and let $h_v^{(0)} = X_v$ be the initial representation of each node v. Let $\mathcal{N}_r(v)$ denote the neighborhood of radius r of node v, and let $G_v^{(t-1)}\Big(\mathcal{N}_{r_1}(v) \setminus \{v\})\Big)$ denote the induced subgraph of G on the set of vertices $\mathcal{N}_{r_1}(v) \setminus \{v\}$, with augmented node label $X_u^{(t-1)} = (h_u^{(t-1)}, \mathbb{1}[(u, v) \in \mathcal{E}])$ for any $u \in \mathcal{N}_{r_1}(v) \setminus \{v\}$. This means we add information about whether nodes are direct neighbors (with distance one) of v. Given a recursion sequence $(r_1, r_2, \ldots, r_t)$ or radii, the representations are updated as

$$m_v^{(t)} = \text{RNN-GNN}^{(t-1)}\Big(G_v^{(t-1)}(\mathcal{N}_{r_1}(v) \setminus \{v\})\Big), \tag{4}$$

$$h_v^{(t)} = \text{COMBINE}^{(t)}\Big(h_v^{(t-1)}, m_v^{(t)}\Big), \tag{5}$$

for any $v \in \mathcal{V}$, and

$$h_G = \text{READOUT}\Big(\{\{h_v^{(t)} : v \in \mathcal{V}\}\}\Big), \tag{6}$$

Different from MPNNs, the recursive update (4) is in general applied on a subgraph, and not a multi-set of node representations. RNN-GNN$^{(t-1)}$ is an RNP-GNN with recursion parameters $(r_2, \ldots, r_t) \in \mathbb{N}^{t-1}$. The final READOUT is an injective, permutation-invariant learnable multi-set function.

If $t = 1$, then

$$m_v^{(t)} = \text{AGGREGATE}^{(t)}\Big(\{\{(h_u^{(t-1)}, \mathbb{1}(u, v) \in \mathcal{E}) : u \in \mathcal{N}_{r_1}(v)\}\}\Big) \tag{7}$$

is a permutation-invariant aggregation function as used in MPNNs, only over a potentially larger neighborhood. For $r_1 = 1$ and $t = 1$, RNP-GNN reduces to MPNN.

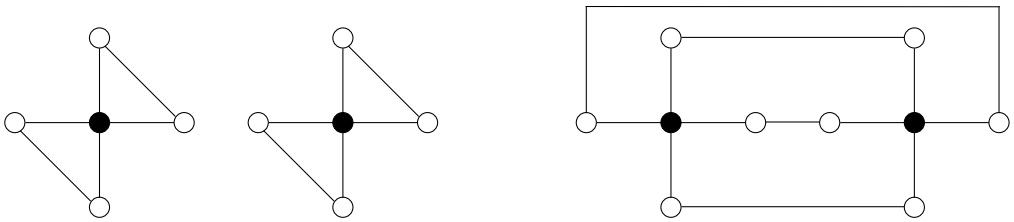

Figure 2. MPNNs cannot count substructures with three nodes or more (Chen et al., 2020). For example, the graph with black center node on the left cannot be counted, since the two graphs on the left result in the same node representations as the graph on the right.

In Figure 1, we illustrate an RNP-GNN with recursion parameters $(2, 2, 1)$ as an example. We also provide pseudocode for RNP-GNNs in Appendix **??**.

5 Expressive Power

In this section, we analyze the expressive power of RNP-GNNs.

5.1 Counting (Induced) Substructures

In contrast to MPNNs, which, in general, cannot count substructures of three vertices or more, in this section we prove that for any set of substructures, there is an RNP-GNN that provably counts them. We begin with a few definitions.

Definition 1. *Let G, H be arbitrary labeled simple graphs, where $\mathcal{V}$ is the set of vertices in G. Also, for any $\mathcal{S} \subseteq \mathcal{V}$, let $G(\mathcal{S})$ denote the subgraph of G induced by $\mathcal{S}$. The* induced-subgraph count function *is defined as*

$$C(G; H) := \sum_{\mathcal{S} \subseteq \mathcal{V}} \mathbb{1}\{G(\mathcal{S}) \cong H\}, \tag{8}$$

i.e., the number of subgraphs of G isomorphic to H. For unlabeled H, the function is defined analogously.

We also need to define a notion of *covering* for graphs. Our definition uses distances on graphs.

Definition 2. *Let $H = (\mathcal{V}_H, \mathcal{E}_H)$ be a (possibly labeled) simple connected graph. For any $\mathcal{S} \subseteq \mathcal{V}_H$ and $v \in \mathcal{V}_H$, define*

$$\bar{d}_H(v; \mathcal{S}) := \max_{u \in \mathcal{S}} d(u, v), \tag{9}$$

where $d(.,.)$ is the shortest-path distance in H.

Definition 3. *Let H be a (possibly labeled) simple connected graph on $t+1$ vertices. A permutation of vertices, such as $(v_1, v_2, \ldots, v_{t+1})$, is called a* vertex covering sequence*, with respect to a sequence $\mathbf{r} = (r_1, r_2, \ldots, r_t) \in \mathbb{N}^t$, called* covering sequence*, if and only if*

$$\bar{d}_{H'_i}(v_i; \mathcal{S}_i) \leq r_i, \tag{10}$$

for any $i \in [t+1] = \{1, 2, \ldots, t+1\}$, where $\mathcal{S}_i = \{v_i, v_{i+1} \ldots, v_{t+1}\}$ and $H'_i = H(\mathcal{S}_i)$ is the subgraph of H induced by the set of vertices $\mathcal{S}_i$. We also say that H admits *the covering sequence $\mathbf{r} = (r_1, r_2, \ldots, r_t) \in \mathbb{N}^t$ if there is a vertex covering sequence for H with respect to $\mathbf{r}$.*

In particular, in a covering sequence we first consider the whole graph as a local neighborhood of one of its nodes with radius r_1. Then, we remove that node and compute the covering sequence of the remaining graph. Figure 3 shows an example of a covering sequence computation. An important property, which holds by definition, is that if $\mathbf{r}$ is a covering sequence for H, then any $\mathbf{r}' \geq \mathbf{r}$ (in a point-wise sense) is also a covering sequence for H.

Note that any connected graph on k vertices admits at least one covering sequence, which is $(k-1, k-2, \ldots, 1)$. To observe this fact, note that in a connected graph, there is at least one node that

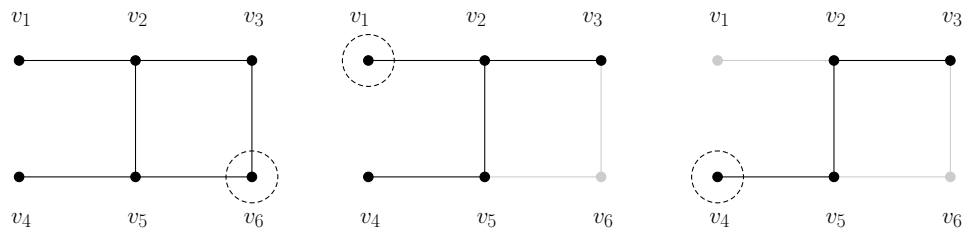

Figure 3. Example of a covering sequence computation for the left graph $G = (\mathcal{V}, \mathcal{E})$. For this graph, $(v_6, v_1, v_4, v_5, v_3, v_2)$ is a vertex covering sequence with respect to the covering sequence $(3, 3, 3, 2, 1)$. The first two computations to obtain this covering sequence are depicted in the middle and on the right.

can be removed and the remaining graph still remains connected. Therefore, we may take this node as the first element of a vertex covering sequence, and inductively find the other elements. Since the diameter of a connected graph with k vertices is always bounded by $k-1$, we achieve the desired result. However, we will see in the next section that, when using covering sequences to identify sufficiently powerful RNP-GNNs, it is desirable to have covering sequences with low r_1, sincethe complexity of the resulting RNP-GNN depends on r_1.

More generally, if H_1 and H_2 are (possibly labeled) simple graphs on k vertices and $H_1 \Subset H_2$, i.e., H_1 is a subgraph of H_2 (not necessarily induced subgraph), then, it follows from the definition that any covering sequence for H_1 is also a covering sequence for H_2. As a side remark, as illustrated in Figure 4, covering sequences need not always be decreasing.

Using covering sequences, we can show the following result.

Theorem 1. *Consider a set of (labeled or unlabeled) graphs $\mathcal{H}$ with $t+1$ vertices, such that any $H \in \mathcal{H}$ admits the covering sequence $(r_1, r_2, \ldots, r_t)$. Then, there is an RNP-GNN with recursion parameters $(r_1, r_2, \ldots, r_t)$ that can count any $H \in \mathcal{H}$. In other words, if there exists $H \in \mathcal{H}$ such that $C(G_1; H) \neq C(G_2; H)$, then $f(G_1; \theta) \neq f(G_2; \theta)$. The same result also holds for the non-induced subgraph count function.*

Theorem 1 states that, with appropriate recursion parameters, any set of (labeled or unlabeled) substructures can be counted by an RNP-GNN. Interestingly, induced and non-induced subgraphs can be both counted in RNP-GNNs[1].

For large substructures, the sequence $(r_1, r_2, \ldots, r_t)$ can be long or includes large numbers, and this will affect the computational complexity of RNP-GNNs. For small, e.g., constant-size substructures, the recursion parameters are also small (i.e., $r_i = O(1)$ for all i), raising the hope to count these structures efficiently. In particular, r_1 is an important parameter. In Section 6, we analyze the complexity in more detail.

5.2 A Universal Approximation Result for Local Functions

Theorem 1 shows that RNP-GNNs can count substructures if their recursion parameters are chosen carefully. Next, we provide a universal approximation result, which shows that they can learn any function related to the local neighborhoods or small subgraphs in a graph.

First, we recall that for a graph G, $G(\mathcal{S})$ denotes the subgraph of G induced by the set of vertices $\mathcal{S}$.

Definition 4. *A function $\ell : \mathbb{G}_n \to \mathbb{R}^d$ is called an $r-$local graph function if*

$$\ell(G) = \phi(\{\{\psi(G(S)) : \mathcal{S} \subseteq \mathcal{V}, |\mathcal{S}| \leq r\}\}), \tag{11}$$

where $\psi : \mathbb{G}_r \to \mathbb{R}^{d'}$ is a function on graphs and ϕ is a multi-set function.

In other words, local functions only depend on small substructures.

[1]For simplicity, we assume that $\mathcal{H}$ only contains $t+1$ vertex graphs. If $\mathcal{H}$ includes graphs with strictly less than $t+1$ vertices, we can simply add a sufficient number of zeros to the end of their covering sequences.

Theorem 2. *For any finite $r-$local graph function $\ell(.)$, there exists an RNP-GNN $f(.;\theta)$ with recursion parameters $(r-1, r-2, \ldots, 1)$ such that $f(G;\theta) = \ell(G)$ for any $G \in \mathbb{G}_n$.*

As a result, we can provably learn all the local information in a graph with an appropriate RNP-GNN. Note that we still need recursions, because the function $\psi(.)$ may be an arbitrarily difficult graph function. However, to achieve the full generality of such a universal approximation result, we need to consider large recursion parameters ($r_1 = r-1$) and injective aggregations in the RNP-GNN network. For universal approximation, We may also need high dimensions if feedforward network layers are used for for aggregation (see the proof of the theorem for more details).

As a remark, for $r = n$, achieving universal approximation on graphs implies solving the graph isomorphism problem. But, in this extreme case, the computational complexity of the model that in this case ($r_1 = \Omega(n)$) the complexity of the model in general not polynomial in n.

6 Computational Complexity

The computational complexity of RNP-GNNs is graph-dependent. For instance, we need to compute the set of local neighborhoods, which is cheaper for sparse graphs. A complexity measure existing in the literature is the tensor order. For higher-order networks, e.g., $k-$IGNs, we need to consider tensors in $\mathbb{R}^{n^k}$. The space complexity is then $O(n^k)$ and the time complexity can be even more, dependent on the algorithm used to process tensors. In general, for a message passing algorithm on the graph, the complexity of the model depends linearly on the number of vertices (if the graph is sparse). Therefore, to bound the complexity of the method, we need to bound the number of node representation updates, which we do in the following theorem.

Theorem 3. *Let $f(.;\theta) : \mathbb{G}_n \to \mathbb{R}^d$ be an RNP-GNN with recursion parameters $(r_1, r_2, \ldots, r_t)$. Assume the observed graphs $G_1, G_2, \ldots$, whose representations we compute, satisfy the following property:*

$$\max_{v \in [n]} |\mathcal{N}_{r_1}(v)| \leq c, \tag{12}$$

where c is a graph independent constant. Then, the number of node updates in the RNP-GNN is $O(nc^t)$.

In other words, if $c = n^{o(1)}$ and $t = O(1)$, then RNP-GNN requires relatively few updates (that is $n^{1+o(1)}$), compared to the higher-order networks ($O(n^{t+1})$). Also, in this case, finding neighborhoods is not difficult, since neighborhoods are small ($n^{o(1)}$). Note that if the maximum degree of the given graphs is Δ, then $c = O(r_1 \Delta^{r_1})$. Therefore, similarly, if $\Delta = n^{o(1)}$ then we can count with at most $n^{1+o(1)}$ updates.

The above results show that by using RNP-GNN, in sparse graphs, we are able to learn functions of the substructures with k vertices without needing to $k-$order tensors $k \geq 2$. In LRP, we also encode neighborhoods of distance r_1 around nodes. In particular, all $c!$ permutations of the nodes in neighborhoods are considered to obtain the representation. As a result, LRPs are more expensive, since $\log(c!) = \Theta(c \log(c)$ which is much greater than $\log(n^k) = k \log(n)$ if $c = \Omega(\log(n))$. As a result, LRP networks only have polynomial complexity if $c = o(\log(n))$. Thus, RNP-GNNs provide an exponential improvement in terms of the tolerable size of neighborhoods with distance r_1 in the graph.

Also, the theorem suggests to take small r_1. Note that the other r_i's are not involved in the upper bound on the complexity, and hence as long as r_1 is small we have low complexity. Note that the covering sequence for graphs may not be increasing. For example, see Figure 4.

7 An Information-Theoretic Lower Bound

In this section, we provide a general information-theoretic lower bound on graph representations which satisfy two conditions. First, they encode a given graph G by representing a number of (possibly small) graphs $G_1, G_2, \ldots, G_t$, and then aggregate the resulting representations. The sequence of graphs $G_1, G_2, \ldots, G_t$ can be obtained in an arbitrary way from G. For example, in a MPNN,

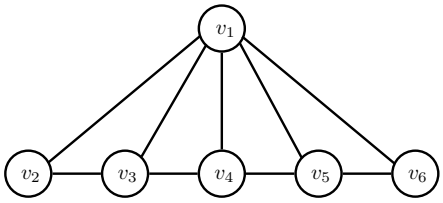

Figure 4. For the above graph, $(v_1, v_2, \ldots, v_6)$ is a vertex covering sequence, and the corresponding covering sequence is $(1, 4, 3, 2, 1)$, which is not decreasing.

G_i can be the computation tree (rooted tree) at node i. As another example, in LRP, G_i can be considered as the local neighborhood around node i.

Formally, consider a graph representation $f(.;\theta) : \mathbb{G}_n \to \mathbb{R}^d$ as

$$f(G;\theta) = \Phi(\{\{\psi(G_i) : i \in [t]\}\}), \quad [t] = \{1, \ldots, t\} \tag{13}$$

for any $G \in \mathbb{G}_n$, where Φ is a multi-set function, $(G_1, G_2, \ldots, G_t) = \Xi(G)$ where $\Xi(.) : \mathbb{G}_n \to \left(\bigcup_{m=1}^{\infty} \mathbb{G}_m\right)^t$ is a function from one graph to t graphs, and also, $\psi : \bigcup_{m=1}^{\infty} \mathbb{G}_m \to [s]$ is a function on graphs. We call this graph representation function an $(s,t)-$good graph representation.

Theorem 4. *Consider a parametrized class of $(s,t)-$good representations $f(.;\theta) : \mathbb{G}_n \to \mathbb{R}^d$ which is able to count any (not necessarily induced[2]) substructure with k vertices. More precisely, for any graph H with k vertices there exists $f(.;\theta)$ such that if $C(G_1;H) \neq C(G_2;H)$, then $f(G_1;\theta) \neq f(G_2;\theta)$. Then[3], $t = \tilde{\Omega}(n^{\frac{k}{s-1}})$.*

In particular, for any $(s,t)-$good graph representation with $s = 2$, i.e., binary encoding functions, we need $\tilde{\Omega}(n^k)$ encoded graphs, which shows that in this case the trivial manual counting is near optimal. Also, if $s \leq k$, then $t = \Omega(n)$ would not suffices for counting.

More interestingly, if $k, s = O(1)$, then it is impossible to do the substructure counting task with $t = O(\log(n))$. As a result, in this case, considering n encoded graphs (such as what is considered in GNNs or LRP networks) cannot be exponentially improved.

The lower bound in this section is information theoretic and hence applies to any algorithm. It may be possible to strengthen it by considering computational complexity, too. For binary encoding (i.e., $s = 2$); however, we know that the bound cannot be improved since manual counting of subgraphs matches the lower bound.

8 Conclusion

In this paper, we studied the theoretical possibility of counting substructures (induced-subgraphs) by a graph representation network. We proposed an architecture, called RNP-GNN, and we proved that for reasonably sparse graphs we can efficiently count substructures. We believe considering the expressive power of GNNs via the set of functions they can learn on substructures can be useful for future developments of new architectures. In the end, we proved a general lower bound for any graph representation which counts subgraphs and works by aggregating representations of a collection of graphs derived from the graph.

[2]Since the theorem holds for non-induced subgraphs, it will also hold for induced subgraphs. Also we consider unlabeled graphs in the theorem for simplicity, but the result also holds when we are given labeled graphs and we need counting labeled substructures.

[3]$\tilde{\Omega}(m)$ is $\Omega(m)$ up to poly-log factors.

REFERENCES

Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 2020.

Waïss Azizian and Marc Lelarge. Characterizing the expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv:2006.15646*, 2020.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems*, pp. 15894–15902, 2019.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Daniel C Elton, Zois Boukouvalas, Mark D Fuge, and Peter W Chung. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design & Engineering*, 4(4):828–849, 2019.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. In *Int. Conference on Machine Learning (ICML)*, pp. 5204–5215. 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7092–7101, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Michael Lingzhi Li, Meng Dong, Jiawei Zhou, and Alexander M. Rush. A hierarchy of graph neural networks based on learnable local features. *arXiv preprint arXiv:1911.05256*, 2019.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2019.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International Conference on Machine Learning*, pp. 4363–4371, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

R Murphy, B Srinivasan, V Rao, and B Riberio. Relational pooling for graph representations. In *International Conference on Machine Learning (ICML 2019)*, 2019.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. In *Advances in Neural Information Processing Systems*, pp. 4081–4090, 2019.

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

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Mengying Sun, Sendong Zhao, Coryandar Gilvary, Olivier Elemento, Jiayu Zhou, and Fei Wang. Graph convolutional networks for computational drug development and discovery. *Briefings in bioinformatics*, 21(3):919–935, 2020.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with message-passing. *arXiv preprint arXiv:2006.15107*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Rex Ying, Zhaoyu Lou, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pp. 4800–4810, 2018.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.