How Powerful are K-hop Message Passing Graph Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

The most popular design paradigm for Graph Neural Networks (GNNs) is 1-hop message passing—aggregating features from 1-hop neighbors repeatedly. However, the expressive power of 1-hop message passing is bounded by the Weisfeiler-Lehman (1-WL) test. Recently, researchers extended 1-hop message passing to K-hop message passing by aggregating information from K-hop neighbors of nodes simultaneously. However, there is no work on analyzing the expressive power of K-hop message passing. In this work, we theoretically characterize the expressive power of K-hop message passing. Specifically, we first formally differentiate two kinds of kernels of K-hop message passing which are often misused in previous works. We then characterize the expressive power of K-hop message passing by showing that it is more powerful than 1-hop message passing. Despite the higher expressive power, we show that K-hop message passing still cannot distinguish some simple regular graphs. To further enhance its expressive power, we introduce a KP-GNN framework, which improves K-hop message passing by leveraging the peripheral subgraph information in each hop. We prove that KP-GNN can distinguish almost all regular graphs including some distance regular graphs which could not be distinguished by previous distance encoding methods. Experimental results verify the expressive power and effectiveness of KP-GNN. KP-GNN achieves competitive results across all benchmark datasets.

1 Introduction

Currently, most existing graph neural networks (GNNs) follow the *message passing* framework, which iteratively aggregates information from the neighbors and updates the representations of nodes. It has shown superior performance on graph-related tasks [1, 2, 3, 4, 5, 6, 7] comparing to traditional graph embedding techniques [8, 9]. However, as the procedure of message passing is similar to the 1-dimensional Weisfeiler-Lehman(1-WL) test [10], the expressive power of message passing GNNs is also bounded by the 1-WL test [7]. Namely, GNNs cannot distinguish two non-isomorphic graph structures if the 1-WL test would fail.

In normal message passing GNNs, the node representation is updated by the direct neighbors of node, which are called 1-hop neighbors. Recently, some works extend the notion of message passing into K-hop message passing [11, 12, 13, 14, 15]. **K-hop message passing** is a type of message passing where the node representation is updated by aggregating information from not only 1-hop, but all the neighbors within K hops of the node. However, there is no work on theoretically characterizing the expressive power of GNNs with K-hop message passing, e.g., whether it can improve the 1-hop message passing or not, and to what extent it can.

In this work, we theoretically characterize the expressive power of K-hop message passing GNNs. Specifically, 1) we formally distinguish two different kernels of the K-hop neighbors, which are

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

often misused in previous works. One is based on graph diffusion and the other is based on shortest path distance. We show that different kernels of K-hop neighbors will result in **different expressive power** of K-hop message passing. 2) We theoretically characterize the expressive power of K-hop message passing GNNs and generalize the proposed theorem to most existing K-hop models. 3) We show that **K-hop message passing is strictly more powerful than 1-hop message passing but bounded by 3-WL** and it failed in distinguishing some simple regular graphs, no matter which kernel is used. This motivates us to further improve K-hop message passing.

Specifically, we introduce **KP-GNN**, a new GNN framework with K-hop message passing, which significantly improves the expressive power of standard K-hop message passing GNNs. In particular, during the aggregation of neighbors in each hop, KP-GNN not only aggregates neighboring nodes in that hop but also aggregates the **peripheral subgraph** (subgraph induced by the neighbors in that hop). This additional information helps the GNN to learn more expressive local structural features around the node. We further prove that KP-GNN is able to distinguish almost all regular graphs and even some distance regular graphs. The proposed KP-GNN has several additional advantages. First, it can be applied to most existing message passing GNNs with little modification. Second, it adds little computational complexity to standard K-hop message passing. We demonstrate the effectiveness of the KP-GNN framework through extensive experiments on both classification and regression tasks.

2 K-hop message passing and its representation power

2.1 Notations

Denote a graph as $G = (V, E)$, where $V = \{1, 2, ..., n\}$ is the node set and $E \subseteq V \times V$ is the edge set. Meanwhile, denote $A \in \{0, 1\}^{n\times n}$ as the adjacency matrix of graph G. Denote x_v as the feature vector of node v and denote e_{uv} as the feature vector of the edge from u to v. Finally, we denote $Q^1_{v,G}$ as the set of 1-hop neighbors of node v in graph G and $\mathcal{N}^1_{v,G}$=$Q^1_{v,G} \cup \{v\}$. Note that when we say K-hop neighbors of node v, we mean **all** the neighbors that have distance from node v less than or equal to K. In contrast, k-th hop neighbors mean the neighbors with **exactly** distance k from node v. The definition of distance will be discussed in section 2.3.

2.2 1-hop message passing framework

Currently, most existing GNNs are designed based on 1-hop message passing framework [16]. Denote h^l_v as the output representation of node v at layer l and $h^0_v = x_v$. Briefly, given a graph G and a 1-hop massage passing GNN, at layer l of the GNN, h^l_v is computed by h^{l-1}_v and $\{\!\{h^{l-1}_u \mid u \in \mathcal{Q}^1_{v,G}\}\!\}$:

$$m^l_v = \text{MES}^l(\{\!\{(h^{l-1}_u, e_{uv})|u \in Q^1_{v,G}\}\!\}), \quad h^l_v = \text{UPD}^l(m^l_v, h^{l-1}_v), \tag{1}$$

where m^l_v is the message to node v at layer l, MES^l and UPD^l are message and update functions at layer l respectively. After L layers of message passing, h^L_v is used as the final node representation of node v. Such a representation can be used to conduct node-level tasks like node classification and node regression. To get the graph representation, a readout function is used:

$$h_G = \text{READOUT}(\{\!\{h^L_v|v \in V\}\!\}), \tag{2}$$

where READOUT is the readout function for computing final graph representation. Then h_G can be used to conduct graph-level tasks like graph classification and graph regression.

2.3 K-hop message passing framework

The 1-hop message passing framework can be directly generalized to K-hop message passing, as it shares the same message and update mechanism. The difference is that an independent message and update function can be employed for each hop. Meanwhile, a combine function is needed to combine the results from different hops into the final node representation at this layer. First, we differentiate two different kernels of K-hop neighbors, which are interchanged and misused in previous research.

The first kernel of K-hop neighbors is *shortest path distance (spd) kernel*. Namely, the k-th hop neighbors of node v in graph G is the set of nodes that have shortest path distance of k from v.

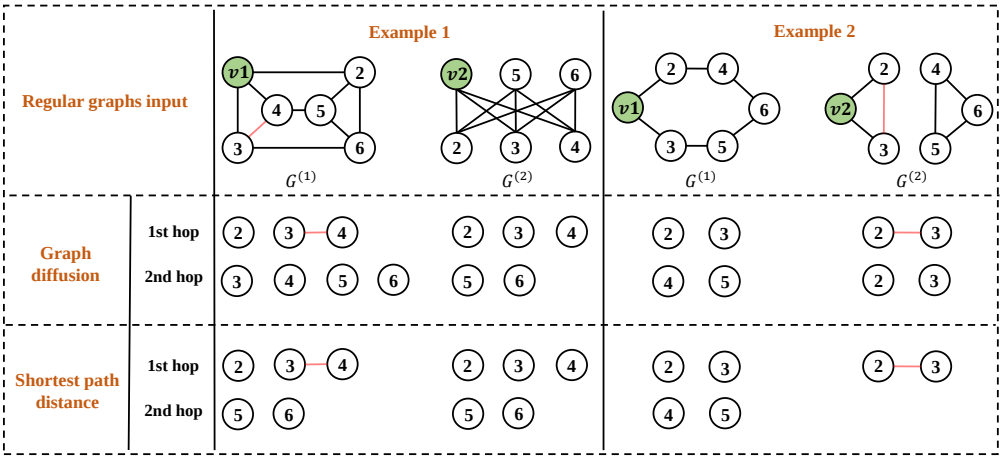

Figure 1: Here are two pairs of non-isomorphic regular graphs. With 2-hop message passing, example 1 can be distinguished by graph diffusion kernel and example 2 can be distinguished by shortest path distance kernel. However, it is indistinguishable if we switch the kernel. Finally, both two examples can be distinguished by adding peripheral edge information.

Definition 1. *For a node v in graph G, the K-hop neighbors $\mathcal{N}_{v,G}^{K,spd}$ of v based on shortest path distance kernel is the set of nodes that have the shortest path distance from node v* ***less than or equal to*** *K. We further denote $Q_{v,G}^{k,spd}$ as the set of nodes in G that are* ***exactly*** *the k-th hop neighbors (with shortest path distance of exactly k) and $\mathcal{N}_{v,G}^{0,spd} = Q_{v,G}^{0,spd} = \{v\}$ is the node itself.*

The second kernel of the K-hop neighbors is based on *graph diffusion (gd)*.

Definition 2. *For a node v in graph G, the K-hop neighbors $\mathcal{N}_{v,G}^{K,gd}$ of v based on graph diffusion kernel is the set of nodes that can diffuse information to node v* ***within the number of random walk diffusion steps*** *K with the diffusion kernel A. We further denote $Q_{v,G}^{k,gd}$ as the set of nodes in G that are* ***exactly*** *the k-th hop neighbors (nodes that can diffuse information to node v with k diffusion steps) and $\mathcal{N}_{v,G}^{0,gd} = Q_{v,G}^{0,gd} = \{v\}$ is the node itself.*

Note that a node can be a k-th hop neighbor of v for multiple k based on the graph diffusion kernel. We include more discussions of K-hop kernels in Appendix A. Next, we formally define the K-hop message passing framework as follows:

$$\begin{gathered} m_v^{l,k} = \text{MES}_k^l(\{\!\{(h_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}\!\}), \quad h_v^{l,k} = \text{UPD}_k^l(m_v^{l,k}, h_v^{l-1}), \\ h_v^l = \text{COMBINE}^l(\{\!\{h_v^{l,k} | k = 1, 2, ..., K\}\!\}), \end{gathered} \tag{3}$$

where $t = \{spd, gd\}$ indicates the kernel of K-hop neighbors. Here, for each hop, we can apply unique MES and UPD functions. Note that for $k > 1$, there may not exist the edge feature e_{uv} as edges are not directly connected. But we leave it here since we can use another type of feature to replace it, which is described in Appendix H. Compared to the 1-hop message passing framework described in Equation (1), the COMBINE function is introduced to combine the representations of node v at different hops. **It is easy to see that the L layer 1-WL GNNs are actually L layer K-hop message passing GNNs with $K = 1$.** We include more discussion of K-hop message passing GNNs in Appendix A.

2.4 Expressive power of K-hop message passing framework

In this section, we theoretically analyze the expressive power of K-hop message passing. We assume there is no edge feature and all nodes in the graph have the same feature, which means that GNNs can only distinguish two nodes using the local structure of nodes. Note that including node features only increases the expressive power of GNNs as nodes/graphs are more easily discriminated. It has been proved that the expressive power of 1-hop message passing is bounded by the 1-WL test on discriminating non-isomorphic graphs [7, 17]. In this section, We show that the K-hop message

passing is strictly more powerful than 1-WL test when $K > 1$. Across the analysis, we utilize regular graphs as examples to illustrate our theorems since they cannot be distinguished using either 1-hop message passing or the 1-WL test. Note that our analysis is not limited to regular graphs but is capable of describing any graphs.

To begin the analysis, we first define *proper K-hop message passing GNNs*.

Definition 3. *A proper K-hop message passing GNN is a class of GNN models where the message, update and combine functions are all injective given the input from a countable space.*

A proper K-hop message passing GNN is easy to find due to the universal approximation theorem [18] of neural network and the Deep Set for set operation [19]. In the latter sections, by default all mentioned K-hop message passing GNNs are proper. Next, we introduce node configuration:

Definition 4. *The node configuration of node v in graph G within k hops under t kernel is a list $A^{k,t}_{v,G} = (a^{1,t}_{v,G}, a^{2,t}_{v,G}, ..., a^{k,t}_{v,G})$, where $a^{i,t}_{v,G} = |Q^{i,t}_{v,G}|$ is the number of i-th hop neighbors of node v.*

When we say two node configurations $A^{k,t}_{v_1,G^{(1)}}$ and $A^{k,t}_{v_2,G^{(2)}}$ are equal, we mean that these two lists are component-wise equal to each other. Now, we state the first proposition:

Proposition 1. *A proper K-hop message passing GNN is strictly more powerful than 1-hop message passing GNNs when $K > 1$.*

To see why this is true, we first discuss how node configuration is related to the first layer of message passing. In the first layer of K-hop message passing, each node aggregates neighbors from up to K hops. As each node has identical label, an injective message function can only know how many neighbors at each hop, which is exactly the node configuration. In other words, **The first layer of K-hop message passing is equivalent to inject node configuration for each node label**. Next, we can characterize the expressive power of 1-hop message passing GNNs using node configuration. When $K = 1$, the node configuration of v_1 and v_2 are $d_{v_1,G^{(1)}}$ and $d_{v_2,G^{(2)}}$, where $d_{v,G}$ is the node degree of v. After L layers, GNN can get node configurations of each node within L hops. Given the statement, it is straightforward to see why these GNNs cannot distinguish any n-sized r-regular graph, as each node in the regular graph have same degree. From another prospective, the expressive power of 1-hop message passing GNN is limited because it only has the degree information of each node in the graph within the receptive field of GNN.

Next, when $K > 1$, the K-hop message passing is at least equally powerful as 1-hop message passing since node configuration up to k hops includes all the information the 1 hop has. To see why it is more powerful, we use two examples to illustrate it. The first example is shown in the left part of Figure 1. Suppose here we use graph diffusion kernel and we want to learn the representation of node v_1 and node v_2 in the two graphs, we know that the 1-hop message passing framework produces the same representation for two nodes as they are both nodes in 6-sized 3-regular graphs. However, it is easy to see that v_1 and v_2 have different local structures and should have different representations. Instead, if we use the 2-hop message passing with the graph diffusion kernel, we can easily distinguish the two nodes by checking the 2nd hop neighbors of the node, as node v_1 has four 2nd hop neighbors but node v_2 only has two 2nd hop neighbors. The second example is shown in the right part of Figure 1. Two graphs in the example are still regular graphs and the 1-hop message passing continue to fail in distinguishing node v_1 and node v_2. In contrast, suppose here we use shortest path distance kernel, node v_1 and v_2 have different numbers of 2nd hop neighbors thus will have different representations by performing 2-hop message passing. These two examples convincingly demonstrate that the K-hop message passing with $K > 1$ can have better expressive power than $K = 1$.

The above discussion can be applied to any existing K-hop GNNs using shortest path distance kernel like GINE [15], Graphormer [20]. However, for K-hop GNNs using graph diffusion kernel like MixHop [11], GPR-GNN [14], MAGNA [13], we find they actually use a weak version of K-hop than the definition of us. We leave the detailed discussion in Appendix B. Further, Distance Encoding [21] also uses shortest path distance information to augment the 1-hop message passing, which is similar to K-hop GNNs with shortest path distance kernel. However, we find the expressive power of the two frameworks are actually different from each other. We leave the detailed discussion in Appendix C.

2.5 Limitation of K-hop message passing framework

Although we show that K-hop message passing with $K > 1$ is better at distinguishing non-isomorphic structures than 1-hop message passing, there are still limitations. In this section, we discuss the limitation of K-hop message passing. Specifically, we show that **the choice of the kernel can affect the power of K-hop message passing**. Furthermore, even with K-hop message passing, we **cannot distinguish some simple non-isomorphic structures and the expressive power of K-hop message passing is bounded by 3-WL.**

Continue looking at the provided examples. In example 1, we know that node v_1 and v_2 have different numbers of 2nd hop neighbors with the graph diffusion kernel. However, if we use the shortest path distance kernel, the two nodes have the same number of neighbors in the 2nd hop, which means that we cannot distinguish two nodes using 2-hop message passing with the shortest path distance kernel. Similarly, in example 2, two nodes have the same number of neighbors in both 1st hop and 2nd hops with graph diffusion kernel. These results highlight that the choice of kernel can affect the expressive power of K-hop message passing. Furthermore, **none of them can distinguish both two examples** with 2-hop message passing.

Recently, Frasca et al. [22] show that any subgraph-based GNNs can be implemented by 3-IGN [23, 24] and thus their power is bounded by 3-WL. Here, we show that the K-hop message passing can also be implemented by 3-IGN and thus:

Theorem 1. *The expressive power of a proper K-hop message passing GNN of any kernel is bounded by 3-WL.*

We include the proof in Appendix D. Given all these observations, we may wonder if there is a way to further improve the expressive power of K-hop message passing?

3 KP-GNN: improving the power of K-hop message passing by peripheral subgraph

In this section, we describe how to improve the expressive power of K-hop message passing by adding additional features to message passing. Specifically, by adding the *peripheral subgraph* information, we can improve the representation power of K-hop message passing by a large margin.

3.1 Peripheral edge and peripheral subgraph

First, we define *peripheral edge* and *peripheral subgraph*.

Definition 5. *The peripheral edges $E(Q^{k,t}_{v,G})$ are defined as the set of edges that connect nodes within set $Q^{k,t}_{v,G}$. We further denote $|E(Q^{k,t}_{v,G})|$ as the number of peripheral edges. The peripheral subgraph $G^{k,t}_{v,G} = (Q^{k,t}_{v,G}, E(Q^{k,t}_{v,G}))$ is defined as the subgraph induced by $Q^{k,t}_{v,G}$ from the whole graph G.*

Briefly speaking, the peripheral edges $E(Q^{k,t}_{v,G})$ record all the edges whose two ends are both from $Q^{k,t}_{v,G}$ and the peripheral subgraph is a graph constituted by peripheral edges. It is easy to see that the peripheral subgraph $G^{k,t}_{v,G}$ automatically contains all the information of peripheral edges $E(Q^{k,t}_{v,G})$. Next, we show that the power of K-hop message passing can be improved by leveraging the information of peripheral edges and peripheral subgraph. We again refer to the examples in Figure 1. Here we only consider the peripheral edge information. In example 1, we notice that at the 1st hop, there is an edge between node 3 and node 4 in the left graph. More specifically, $E(Q^{1,t}_{v_1,G^{(1)}}) = \{(3,4)\}$. In contrast, we have $E(Q^{1,t}_{v_2,G^{(2)}}) = \{\}$ in the right graph, which means there is no edge between the 1st hop neighbors of v_2. Therefore, by adding this information to the message passing, we can successfully distinguish these two nodes. Similarly, in example 2, there is one edge between the 1st hop neighbors of node v_2 but no such edge exists for node v_1. By leveraging peripheral edge information, we can distinguish the two nodes as well. The above examples demonstrate the effectiveness of the peripheral edge and peripheral subgraph information.

3.2 K-hop peripheral-subgraph-enhanced graph neural network

In this section, we propose the **K**-hop **P**eripheral-subgraph-enhanced **G**raph **N**eural **N**etwork (KP-GNN), which equipes the K-hop message passing with peripheral subgraph information for more powerful GNN design. Recall the K-hop message passing defined in Equation (3). The only difference of KP-GNN is that we revise the message function as follows:

$$\hat{h}_v^{l,k} = \text{MES}_k^l(\{\!\{(h_u^{l-1},\ e_{uv}) | u \in Q_{v,G}^{k,t}\}\!\},\ G_{v,G}^{k,t}). \tag{4}$$

Briefly speaking, in the message step at the k-th hop, we not only aggregate information of the neighbors but also the peripheral subgraph at the k-th hop. The implementation of KP-GNN can be very flexible as any graph encoding function can be used. To maximize the information the model can encode while keeping it simple, we implement the message function as:

$$\text{MES}_k^l = \text{MES}_k^{l,normal}(\{\!\{(h_u^{l-1}, e_{uv}) | u \in Q_{v,G}^{k,t}\}\!\}) + \sum_{c \in C} \frac{1}{|C|} \sum_{(i,j) \in E(Q_{v,G}^{k,t})_c} e_{ij}\ , \tag{5}$$

where $\text{MES}_k^{l,normal}$ denotes the message function in the original GNN model, C is the set of connected components in $G_{v,G}^{k,t}$, $E(Q_{v,G}^{k,t})_c$ is the edge set of the c-th connected component in $G_{v,G}^{k,t}$. Such implementation helps the KP-GNN to not only encode the $E(Q_{v,G}^{k,t})$ but also partial information of $G_{v,G}^{k,t}$ (number of components). With this implementation, any GNN model can be incorporated into and be enhanced by the KP-GNN framework by replacing $\text{MES}_k^{l,normal}$ and UPD_k^l with the corresponding functions for each hop k. We leave the detailed implementation in Appendix H.

3.3 The expressive power of KP-GNN

In this section, we theoretically characterize the expressive power of KP-GNN and compare it with the original K-hop message passing framework. The key insight is that, according to Equation (4), the message function at the k-th hop additionally encodes $G_{v,G}^{k,t}$ compared to normal K-hop message passing. Next, we show that even with very simple information, the KP-GNN can become powerful enough for distinguishing regular graphs.

Theorem 2. *Consider all pair of n-sized r-regular graphs, where $3 \leq r < \sqrt{2logn}$. For any small constant $\epsilon > 0$, there exists a KP-GNN using shortest path distance as kernel and only peripheral edge information with at most $K = \lceil (\frac{1}{2} + \epsilon \frac{logn}{log(r-1-\epsilon)}) \rceil$, which distinguishes almost all $(1 - o(1))$ such pair of graphs with only 1-layer message passing.*

We include the proof in Appendix E. The above theorem proves that a simple implementation of KP-GNN leveraging only peripheral edge information can distinguish almost all regular graphs with some K and 1 layer.

Moreover, K-hop message passing with shortest path distance kernel cannot distinguish any distance regular graphs with the same intersection array according to the Theorem 3.7 in Distance Encoding [21]. Here we show that KP-GNN is more powerful than Distance Encoding on **distinguishing distance regular graphs**.

Theorem 3. *For two non-isomorphic distance regular graphs $G^{(1)} = (V^{(1)}, E^{(1)})$ and $G^{(2)} = (V^{(2)}, E^{(2)})$ with the same intersection array $(b_0, b_1, ..., b_{d-1}; c_1, c_2, ..., c_d)$, we pick two nodes v_1 and v_2 from two graphs respectively. Given a proper 1-layer K-hop KP-GNN with message functions defined in Equation (5), it can distinguish v_1 and v_2 if $b_0 - b_j - c_j = 2$ for some $j \leq K$ and $G_{v_1,G^{(1)}}^{j,t}$ and $G_{v_2,G^{(2)}}^{j,t}$ are non-isomorphic.*

We include the proof in Appendix F. Theorem 3 shows that the KP-GNN with a simple implementation can distinguish some distance regular graphs, which further distinguishes KP-GNN from Distance Encoding [21] and subgraph-based GNNs [25, 26, 27, 28] as they cannot distinguish distance regular graphs. We include further discussion in Appendix F. However, with the current implementation, KP-GNN cannot distinguish all distance regular graphs.

3.4 Time, space complexity, and limitation

In this section, we discuss the time and space complexity of K-hop message passing GNN and KP-GNN. Suppose a graph has n nodes and m edges. Then, the K-hop message passing and KP-GNN have both the space complexity of $O(n)$ and time complexity of $O(n^2)$ for shortest path distance kernel. Note that the complexity of graph diffusion is no less than the shortest path distance kernel. We can see that KP-GNN only require same space complexity as vanilla GNNs and much less time complexity than $O(nm)$ of subgraph-based GNNs. We leave the detailed discussion on complexity and limitation of KP-GNN in Appendix G

4 Related Work

Expressive power of GNN. Analyzing the expressive power of GNN is a crucial problem as it can serve as a guidance on how to improve GNNs. Xu et al. [7] and Morris et al. [17] first proved that the power of 1-hop message passing is bounded by the 1-WL test. In other words, 1-hop message passing cannot distinguish any non-isomorphic graphs that the 1-WL test fails to. In recent years, many efforts have been put into increasing the expressive power of 1-hop messaging passing. The first line of research tries to mimic the higher-order WL tests, like 1-2-3 GNN [17], PPGN [23], ring-GNN [29]. However, they require exponentially increasing space and time complexity w.r.t. node number and cannot be generalized to large-scale graphs. The second line of research tries to enhance the rooted subtree of 1-WL with additional features. Some works [30, 31, 32] add one-hot or random features into nodes. Although they achieve good results in some setting, they deteriorate the generalization ability as such features produce different representations for nodes even with the same local graph structure. Instead, ID-GNN [33] enhances the rooted subtree by labeling the root node with a different color. However, the additional expressive power of ID-GNN only comes from the rooted node's labeling, which may not be powerful enough to distinguish more complex structures. Some works like Distance Encoding [21], SEAL [34], labeling trick [35] and GLASS [36] introduce node labeling based on either distance or distinguishing target node set. Specifically, Distance Encoding can be regarded as K-hop message passing with the shortest path distance kernel. On the other hand, GraphSNN [37] introduces a hierarchy of local isomorphism and proposes structural coefficients as additional features to identify such local isomorphism. However, the function designed to approximate the structural coefficient cannot fully achieve its theoretical power. The third line of research attends to revising the rooted subtree. Specifically, NGNN [25] encodes a rooted subgraph instead of a rooted subtree thus achieving superior expressive power on distinguishing regular graphs. However, it needs to run an inner GNN on every node of the graph thus introducing much more computation overhead. GNN-AK [26] applies a similar idea as NGNN. The only difference lies in how to compute the node representation from the local subgraph.

K-hop message passing GNN. There are some existing works that instantiate the K-hop message passing framework. For example, MixHop [11] performs messing passing on each hop with graph diffusion kernel and concatenates the representation on each hop as the final representation. K-hop [12] sequentially performs the message passing from hop K to hop 1 to compute the representation of the center node. However, it is not parallelizable due to its computational procedure. MAGNA [13] introduces an attention mechanism to K-hop message passing. GPR-GNN [14] use graph diffusion kernel to perform graph convolution on K-hop and aggregate them with learnable parameters. However, none of them give a formal definition of K-hop message passing and theoretically analyze its representation power and limitations.

5 Experiments

In this section, we conduct extensive experiments to evaluate the performance of KP-GNN. Specifically, we 1) empirically verify the expressive power of KP-GNN on 3 simulation datasets and demonstrate the benefits of KP-GNN compared to normal K-hop message passing GNNs and existing models; 2) show that the KP-GNN can achieve state-of-the-art performance on 5 TU datasets; 3) demonstrate that the KP-GNN achieves comparable performance on 3 molecular prediction datasets; 4) analyze the running time of KP-GNN. The detail of each variant of KP-GNN is described in Appendix H and the detailed experimental setting is described in Appendix I.

Table 1: Simulation dataset result. The top two are highlighted by **First**, **Second** . (*: with encoding)

Method	EXP(ACC)	Node Properties ($\log_{10}$(MAE))			Graph Properties ($\log_{10}$(MAE))			Counting Substructures (MAE)			
		SSSP	Ecc.	Lap.	Connect.	Diameter	Radius	Tri.	Tailed Tri.	Star	4-Cycle
GIN	50	-2.0000	-1.9000	-1.6000	-1.9239	-3.3079	-4.7584	0.3569	0.2373	0.0224	0.2185
PNA	50	**-2.8900**	**-2.8900**	-3.7700	-1.9395	3.4382	-4.9470	0.3532	0.2648	0.1278	0.2430
PPGN	100	-	-	-	-1.9804	-3.6147	-5.0878	**0.0089**	**0.0096**	0.0148	**0.0090**
GIN-AK+	100	-	-	-	-2.2268	-3.7585	-5.1044	0.0885	0.0696	0.0162	0.0668
K-GIN+	100	**-2.7651**	-2.6159	**-4.4309**	-2.0725	-3.9732	**-5.3113**	0.1180	0.0747	**0.0009**	0.0840
KP-GIN+	100	**-2.7651**	**-2.6193**	**-4.6107**	**-4.1803**	**-3.9952**	-5.2206	0.0377	0.0314	**0.0024**	0.0258
GIN-AK+*	100	-	-	-	-2.7513	-3.9687	5.1846	0.0123	0.0112	0.0150	0.0126
KP-GIN+*	100	-2.7628	-2.6164	-4.3540	**-3.7229**	**-4.0160**	**-5.2357**	**0.0029**	**0.0027**	0.0029	**0.0082**

Datasets:To evaluate the expressive power of KP-GNN, we choose: 1) EXP dataset [38], which contains 600 pairs of 1-WL-indistinguishable but non-isomorphic graphs. 2) Graph property regression (connectedness, diameter, radius) and node property regression (single source shortest path, eccentricity, Laplacian feature) task on graph random dataset [39]. 3) Graph substructure counting (triangle, tailed triangle, star and 4-cycle) tasks on random graph dataset [40]. For TU datasets evaluation, we choose MUTAG [41],D&D [42],PROTEINS [42],PTC-MR [43], and IMDB-B [44] from TU database. For molecule prediction datasets, we pick QM9 [45, 46], ZINC [47], and MolHIV [48]. The detailed statistics of the datasets are described in Appendix J.Without further highlight, all error bars in the result tables are standard deviation of multiple runs (more details are in Appendix I).

Table 2: Ablation study on EXP

kernel	K	K-GIN+	KP-GIN+
GD	K=1	50	50
	K=2	50	100
	K=3	66.17	100
	K=4	100	100
SPD	K=1	50	50
	K=2	50	100
	K=3	100	100
	K=4	100	100

Empirical verification of the expressive power: To evaluate the power of KP-GNN, we compare it with several existing models. For the baseline model, we use GIN [7], which has the same expressive power as 1-WL test. For more powerful baselines, we use GIN-AK+ [26], PNA [39] and PPGN [23]. For KP-GNN, we implement the KP-GIN+. To evaluate the effectiveness of peripheral subgraph, we also implement the normal K-hop version of GINE+, denote as K-GIN+. The results are shown in Table 1. Baseline results are taken from [26] and [39]. We also compare results of GIN-AK+ for both with and without D2C encoding and KP-GIN+ both with and without path encoding. We can see both K-GIN+ and KP-GIN+ achieve perfect performance on EXP dataset. Further, we conduct an ablation study on KP-GIN+ and K-GIN+ using EXP dataset. Table 2 presents the results. We can see that KP-GNN achieves perfect results with only $K \geq 2$ for both two kernels. However, K-GIN requires $K \geq 3$ and $K \geq 4$ to get perfect results for shortest path distance kernel and graph diffusion kernel respectively. Results on various property regression tasks further demonstrate the advantage of KP-GNN over existing models and normal K-hop message passing GNNs.

Table 3: TU dataset evaluation result.

Method	MUTAG	D&D	PTC-MR	PROTEINS	IMDB-B
WL	90.4±5.7	79.4±0.3	59.9±4.3	75.0±3.1	73.8±3.9
GIN	89.4±5.6	-	64.6±7.0	75.9±2.8	75.1±5.1
DGCNN	85.8±1.7	79.3 ±0.9	58.6 ±2.5	75.5±0.9	70.0±0.9
GraphSNN	91.24±2.5	82.46 ±2.7	66.96±3.5	76.51 ±2.5	76.93±3.3
GIN-AK+	91.30±7.0	-	68.20±5.6	77.10±5.7	75.60±3.7
KP-GCN	91.1±6.0	78.9±3.9	64.1±7.9	76.4±4.8	75.1±3.6
KP-GraphSAGE	91.1±3.9	78.2±3.7	65.9±7.6	76.1±4.4	74.7±3.7
KP-GIN	91.1±5.4	78.6±4.3	64.7±6.8	76.1±5.4	74.4±3.4
GIN-AK+*	95.0±6.1	00M	74.1±5.9	78.9±5.4	77.3±3.1
GraphSNN*	94.70±1.9	83.93±2.3	70.58±3.1	78.42±2.7	78.51±2.8
KP-GCN*	**96.1±4.6**	83.1±2.8	72.4±5.8	**80.0±3.8**	**79.0±2.7**
KP-GraphSAGE*	**96.1±4.6**	**84.0±3.4**	**74.4±6.5**	79.9±4.2	78.7±4.0
KP-GIN*	95.6±5.1	82.9±2.3	71.8±6.8	79.8±3.8	78.0±3.7

Evaluation on TU datasets: For baseline models, we select: 1) graph kernel based method WL subtree kernel [49]; 2) 1-hop message passing based GNN methods: GIN [7] and DGCNN [6]; 3) advanced GNN methods: GraphSNN [37] and GIN-AK+ [26]. For the proposed KP-GNN, we implement GCN [1], GraphSAGE [3], and GIN [7] using the KP-GNN framework, denoted as KP-

Table 4: QM9 results. The top two are highlighted by **First**, **Second**

Target	DTNN	MPNN	Deep LRP	PPGN	Nested 1-2-3-GNN	KP-GIN+
μ	**0.244**	0.358	0.364	**0.231**	0.433	0.365
α	0.95	0.89	0.298	0.382	**0.265**	**0.249**
$\varepsilon_{\text{HOMO}}$	0.00388	0.00541	**0.00254**	0.00276	0.00279	**0.00243**
$\varepsilon_{\text{LUMO}}$	0.00512	0.00623	0.00277	0.00287	**0.00276**	**0.00252**
$\Delta\varepsilon$	0.0112	0.0066	**0.00353**	0.00406	0.00390	**0.00346**
$\langle R^2 \rangle$	17.0	28.5	19.3	**16.7**	20.1	**16.64**
ZPVE	0.00172	0.00216	0.00055	0.00064	**0.00015**	**0.00017**
U_0	2.43	2.05	0.413	0.234	**0.205**	**0.06632**
U	2.43	2.00	0.413	0.234	**0.200**	**0.09447**
H	2.43	2.02	0.413	**0.229**	0.249	**0.06037**
G	2.43	2.02	0.413	0.238	**0.253**	**0.04752**
C_v	0.27	0.42	0.129	0.184	**0.0811**	**0.0977**

GCN, KP-GraphSAGE, and KP-GIN respectively. The results are shown in Table 3. For a detailed comparison, we report the results of two different settings. The first setting follows Xu et al. [7] and the second setting follows Wijesinghe and Wang [37]. We denote the second setting with * in the table. We can see that under the setting of Wijesinghe and Wang [37], KP-GNN achieves state-of-the-art performance across all datasets, which demonstrates the performance of KP-GNN on real-world datasets. Under the setting of Xu et al. [7], KP-GNN still achieves comparable performance.

Evaluation on molecular prediction tasks: For QM9 dataset, we report results of DTNN, MPNN from [46]. We further select Deep LRP [40], PPGN [23], and Nested 1-2-3-GNN [25]. For ZINC dataset, we report results of MPNN [16] and PNA [39] from [47]. We further pick Graphormer [20], GSN [50], GIN-AK+ [26], and CIN [51] . For MolHIV dataset, we report results of PNA [39], DeepLRP [40], GINE [15], NGNN [25], GIN-AK+[26] and GraphSNN [37]. The results of QM9 dataset are shown in Table 4. We can see KP-GNN achieves state-of-the-art performance on most of targets. The results of MolHIV and ZINC dataset are shown in Table 5 and Table 6. Although KP-GNN does not achieve the best result, it is still comparable to other methods.

Table 5: OGB-MolHIV result.

Method	Test AUC
PNA	79.05 ±1.32
DeepLRP	77.19±1.40
GINE-VN	76.60±1.40
NGNN	78.34±1.86
GIN-AK+	79.61±1.19
GraphSNN-VN	**79.72±1.83**
KP-GIN+-VN	78.48±0.87

Table 6: ZINC result.

Method	# param.	test MAE
MPNN	480805	0.145±0.007
PNA	387155	0.142±0.010
Graphormer	489321	0.122±0.006
GSN	~500000	0.101±0.010
GIN-AK+	-	0.080±0.001
CIN	-	**0.079±0.006**
KP-GIN+	500790	0.119±0.002

Table 7: Running time (ms/epoch)

Method	D&D	ZINC	Graph property
GIN	1.15	5.00	1.80
K-GIN	6.55	11.00	3.77
KP-GIN	7.84	13.00	6.82
KP-GIN+	7.96	9.00	8.74

Running time comparison: In this section, we compare the running time of KP-GNN to 1-hop message passing GNN and K-hop message passing GNN. We use GIN [7] as the base model. We also include the KP-GIN+. All models use the same number of layers and hidden dimensions for a fair comparison. The results are shown in Table 7. For D&D, ZINC, Graph property dataset, we set $K = 5$, $K = 4$, $K = 6$ respectively. We can see the computational overhead grows nearly linearly with K, which is far less than the theoretical analysis.

6 Conclusion

In this paper, we theoretically characterize the power of K-hop message passing GNNs and propose the KP-GNN to improve the expressive power by leveraging the peripheral subgraph information at each hop. Theoretically, we prove that KP-GNN can distinguish almost all regular graphs including some distance regular graphs. Empirically, KP-GNN achieves competitive results across all simulation and real-world datasets.

References

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[2] 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*, pages 2224–2232, 2015.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

[4] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[5] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[6] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, pages 4438–4445, 2018.

[7] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[8] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[10] Boris Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

[11] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

[12] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020.

[13] Guangtao Wang, Zhitao Ying, Jing Huang, and Jure Leskovec. Multi-hop attention graph neural network, 2021. URL `https://openreview.net/forum?id=muppfCkU9H1`.

[14] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=n6jl7fLxrP`.

[15] Rémy Brossard, Oriel Frigo, and David Dehaene. Graph convolutions that can finally model local structure. *arXiv preprint arXiv:2011.15069*, 2020.

[16] 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*, pages 1263–1272. JMLR. org, 2017.

[17] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[18] George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[19] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

[20] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=OeWooOxFwDa`.

[21] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding–design provably more powerful gnns for structural representation learning. *arXiv preprint arXiv:2009.00142*, 2020.

[22] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *ArXiv*, abs/2206.11140, 2022.

[23] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pages 2156–2167, 2019.

[24] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

[25] Muhan Zhang and Pan Li. Nested graph neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=7_eLEvFjCi3`.

[26] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Mspk_WYKoEH`.

[27] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=ZKbZ4mebI9l`.

[28] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=dFbKQaRk15w`.

[29] 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*, pages 15894–15902, 2019.

[30] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

[31] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization, 2021. URL `https://openreview.net/forum?id=L7Irrt5sMQa`.

[32] Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

[33] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. *arXiv preprint arXiv:2101.10320*, 2021.

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

[35] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=Hcr9mgBG6ds`.

[36] Xiyuan Wang and Muhan Zhang. GLASS: GNN with labeling tricks for subgraph representation learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XLxhEjKNbXj.

[37] Asiri Wijesinghe and Qing Wang. A new perspective on "how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uxgg9o7bI_3.

[38] Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

[39] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

[40] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 2020.

[41] Asim Kumar Debnath, de Compadre RL Lopez, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[42] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[43] Hannu Toivonen, Ashwin Srinivasan, Ross D King, Stefan Kramer, and Christoph Helma. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics*, 19(10): 1183–1193, 2003.

[44] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

[45] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[46] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[47] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[48] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[49] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011.

[50] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting, 2021. URL https://openreview.net/forum?id=LT0KSFnQDWF.

[51] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montufar, and Michael M. Bronstein. Weisfeiler and lehman go cellular: CW networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=uVPZCMVtsSG.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[53] Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks, 2022. URL `https://arxiv.org/abs/2205.11172`.

[54] Floris Geerts. The expressive power of kth-order invariant graph networks. *ArXiv*, abs/2007.12035, 2020.

[55] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL `https://aclanthology.org/D15-1166`.

[56] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *CoRR*, abs/1806.03536, 2018. URL `http://arxiv.org/abs/1806.03536`.

[57] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The implementation of KP-GNN can be found at https://github.com/anoncodesub/KP-GNN.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] The license can be found in their github.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] All datasets we used are all open-sourced.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] There is no personal information in the datasets.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]