K-HOP GRAPH NEURAL NETWORKS

Author: Nikolentzos et al,
Presenter: Amr Abouelkhair,
CS 886

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Introduction

- We process an increasing amount of graph structured data
- GNNs have been a very useful technique for processing such data
- Specifically, GNNs have been the state of the art when it comes to node classification and link prediction
- Many works already attempt and evaluate the expressive power of GNNs
- A lot of these works find that GNNs and WL test are closely related in their expressive power

What is the problem?

- The goal is be able to identify fundamental graph properties like connectivity, bipartiteness and triangle freeness
- This is a step closer to being able to better identify isomorphic graphs
- It was recently shown that the WL subtree Kernel has insufficient expressive power to identify these fundamental graph properties [22]
- This work presents a proof that similar to the WL approach, GNNs have insufficient power to identify graph properties like connectivity, bipartiteness, and triangle freeness

Why is this problem important?

- Learning representations from graph data has many real-world applications
- It's important to understand the limits of GNNs and be aware of which tasks allows them to perform best
- Specifically identifying isomorphic graphs is important in chemo-informatics, biology and engineering
- For example, it's important to identify equivalent electronic circuits which is essentially a graph isomorphism problem

How do current GNNs work?

- Mostly apply a message passing scheme
- Each node updates its hidden state by aggregating the states of its neighbors
- After k iterations each node should essentially have the structural information within its k-hop neighborhood

$$a_v^{(t)} = \text{AGGREGATE}^{(t)}\left(\left\{h_u^{(t-1)} | u \in \mathcal{N}_1(v)\right\}\right)$$

$$h_v^{(t)} = \text{MERGE}^{(t)}\left(h_v^{(t-1)}, a_v^{(t)}\right)$$

Why don't previous methods work on that problem?

- We're still in the process of discovering the limitations of GNNs
- However, with WL kernel unable to identify the graph properties we're interested in, it's likely that GNNs won't either
- We present the following definitions and a proof that GNNs fail to identify fundamental graph properties

Definition 1

Definition 1. *Let $\mathcal{P}$ be a graph property. If for each $n \in \mathbb{N}$, a GNN produces different representations for every $G_1 \in \mathcal{P}_n$ and $G_2 \notin \mathcal{P}_n$, i.e., it holds that $h_{G_1} \neq h_{G_2}$, then we say that $\mathcal{P}$ can be identified by the GNN.*

Lemma 1

Lemma 1. *The standard GNN maps the nodes of two regular graphs of the same size and degree to the same feature vector.*

Lemma 1 Example

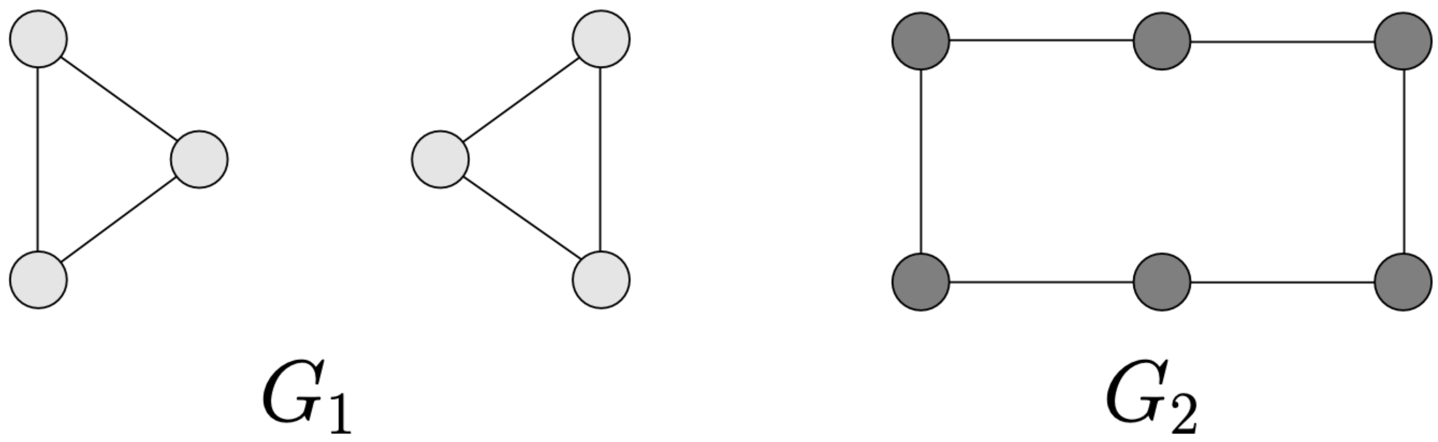


Figure 1: Two 2-regular graphs on 6 vertices. The two graphs serve as a counterexample for the proof of Theorem 1.

Lemma 1 Proof Sketch: Assumptions

- Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be 2 regular non-isomorphic graphs of the same degree and size
- Let $v_1 \in V_1$ and $v_2 \in V_2$
- All nodes have the same initial representation $h_{v1}^{(0)} = h_{v2}^{(0)}$
- We show that for an arbitrary iteration $t \geq 1$, $h_{v1}^{(t)} = h_{v2}^{(t)}$

Lemma 1 Proof Sketch: Induction

- Assume for induction that $h_{v1}^{(t-1)} = h_{v2}^{(t-1)}$
- Given the multisets M_{v1} and M_{v2} of neighbors of v_1 and v_2 respectively
- Since v_1 and v_2 have the same degree and by induction hypothesis $M_{v1} = M_{v2}$
- Therefore, no matter what AGGREGATE and MERGE functions we choose we get $h_{v1}^{(t)} = h_{v2}^{(t)}$ since we provide those functions with the same input

Theorem 1

Theorem 1. *The standard GNN cannot identify connectivity, bipartiteness or triangle freeness.*

Connectivity Contradiction

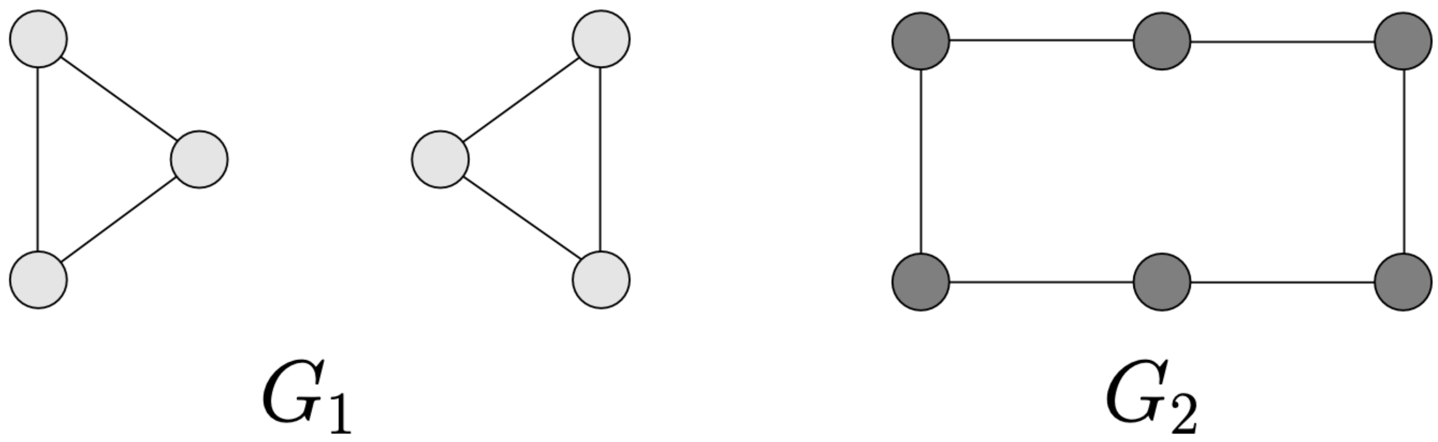


Figure 1: Two 2-regular graphs on 6 vertices. The two graphs serve as a counterexample for the proof of Theorem 1.

Bipartiteness and triangle freeness contradiction

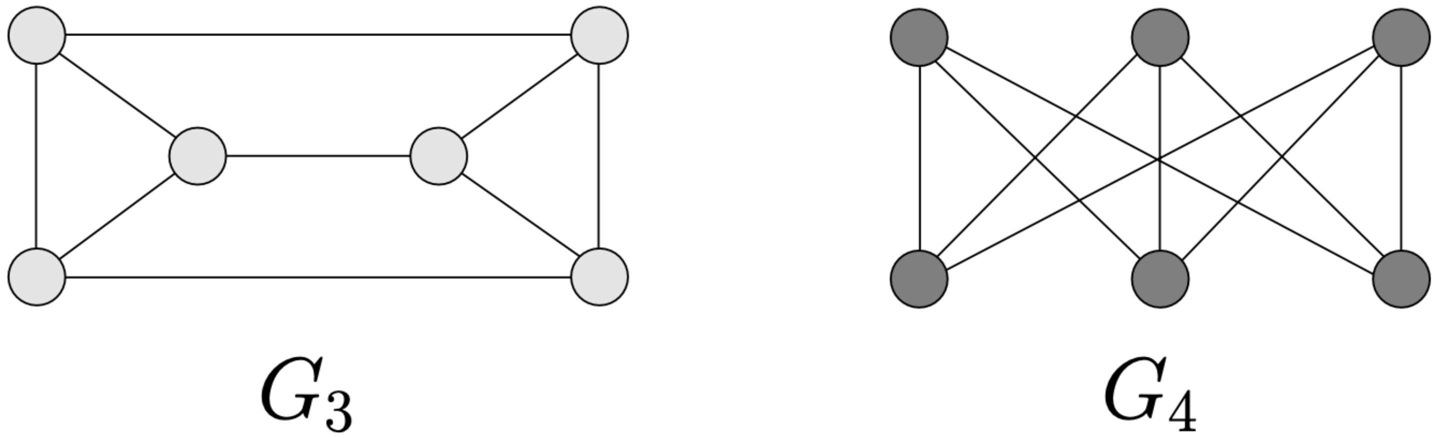


Figure 2: Two 3-regular graphs on 6 vertices. The two graphs serve as a counterexample for the proof of Theorem 1.

What is the solution to the problem the authors propose?

- The authors propose a generalization of GNNs noted k-hop Graph Neural Network
  - Instead of aggregating only the 1-hop neighbors for a node this approach attempts to aggregate the k-hop neighbors to update the hidden state for each node

$$a_v^{(t)} = \text{AGGREGATE}^{(t)}\left(\left\{h_u^{(t-1)} | u \in \mathcal{N}_k(v)\right\}\right)$$

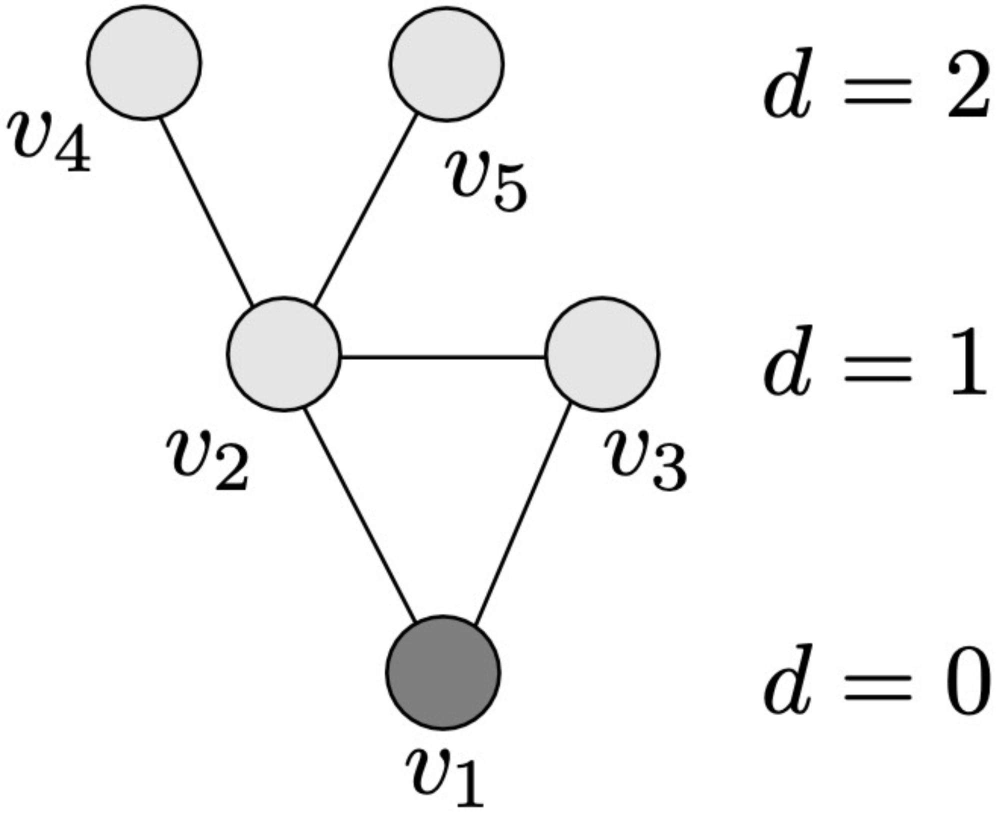
$$h_v^{(t)} = \text{MERGE}^{(t)}\left(h_v^{(t-1)}, a_v^{(t)}\right)$$

Proposed Architecture Notation: Prelims

- Let $G = (V, E)$, let's focus on one node $v \in V$ which would be the root of out k-hop neighborhood subgraph G^k_v
- For a given iteration t, we define an inner representation x_u for node
- We initialize $x_u = h_u^{(t-1)}$

Proposed Architecture Notation: Rings

- We also denote the ring at distance d from v to be $R_d(v)$
- Finally, we note that the neighbors of any node $u \in R_d(v)$ all belong to either $R_{d-1}(v)$, $R_d(v)$, $R_{d+1}(v)$



The UPDATE module

- We also define an UPDATE module
- Given node w and set on Nodes S

$$\mathrm{UPDATE}(w, S) = \mathrm{MLP}\Big(\mathrm{MLP}_1(x_w) + \sum_{u \in S} \mathrm{MLP}_2(x_u)\Big)$$

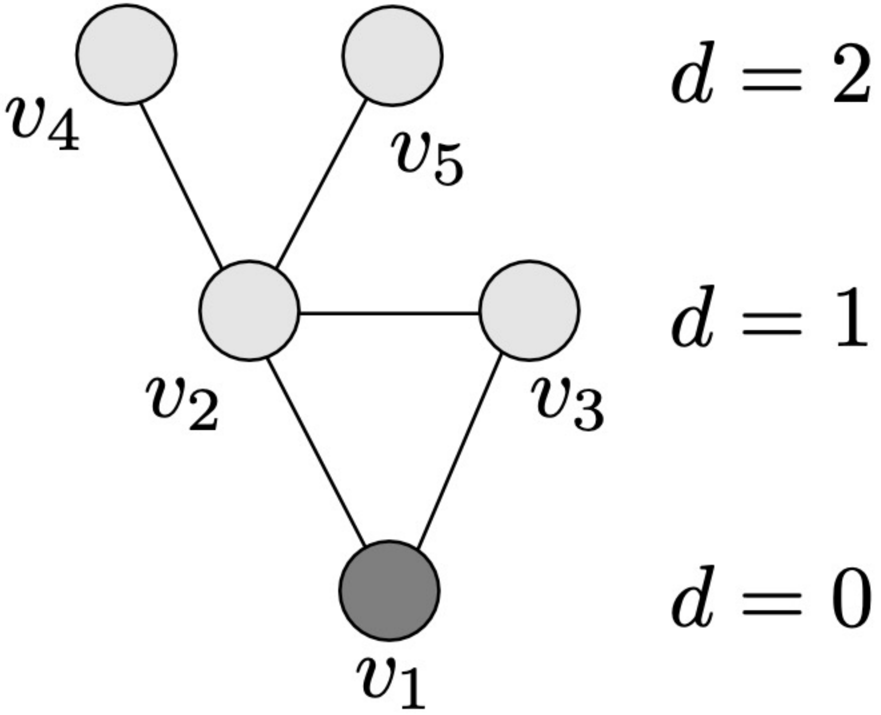
The UPDATE procedure

- We start from the outside in (ie. d = k, d = k-1, …., d = 0)
  - Neighbors of v can either be updated **across** rings or **within** rings
  - For a given node $u \in R_d(v)$
    - If it has neighbors in $R_{d+1}(v)$ then we update x_u across
    - If it has neighbors in $R_d(v)$ then we update x_u within
    - Else we do not update it and $x_u = h^{t-1}{}_u$
- We iteratively decrease d within one iteration until $h^{(t)}{}_v$ is updated

Example of 2-hop GNN in action

$$x_{v_2} = \text{UPDATE}^{(t)}_{1,across}(v_2, \{v_4, v_5\})$$

$$x_{v_2} = \text{UPDATE}^{(t)}_{1,within}(v_2, \{v_3\})$$



$$x_{v_3} = \text{UPDATE}^{(t)}_{1,within}(v_3, \{v_2\})$$

$$h^{(t)}_{v_1} = \text{UPDATE}^{(t)}_{0,across}(v_1, \{v_2, v_3\})$$

Figure 3: The 2-hop neighborhood graph $G^2_{v_1}$ of a node v_1 of graph G.

Expressive power of k-hop GNNs

Theorem 2. *For the k-hop GNN, there exists a sequence of modules $UPDATE^{(0)}_{0,across}$, $UPDATE^{(0)}_{1,within}$, $UPDATE^{(0)}_{1,across}, \ldots\ldots, UPDATE^{(T)}_{k-1,across}$, $UPDATE^{(T)}_{k,within}$ such that*

1. *it can identify triangle-freeness for $k \geq 1$*
2. *connectivity for $k > \delta_{min}$ where δ_{min} is the minimum of the diameters of the connected components*
3. *bipartiteness for $k \geq \frac{l-1}{2}$ where l is the length of the smallest odd cycle in the graph (if any)*

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Experimental Evaluation

- The authors evaluate the proposed model on both
  - Node classification
    - Synthetic dataset
    - Real-world dataset
  - Graph classification
    - Synthetic dataset
    - Real-world dataset

Node classification synthetic dataset

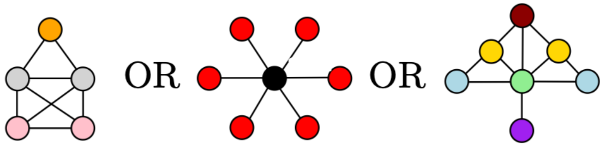
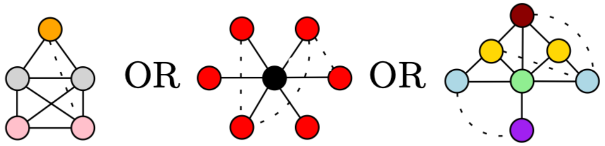
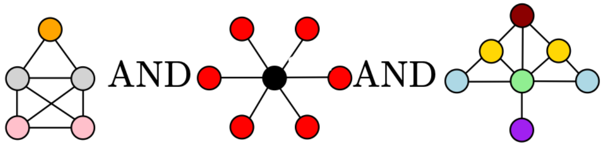
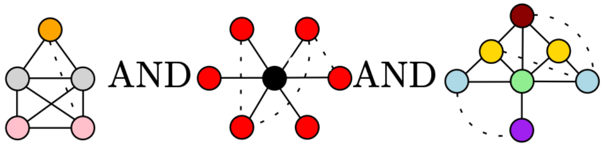
Configuration	Shapes placed along a cycle graph
BASIC	
BASIC PERTURBED	
VARIED	
VARIED PERTURBED	

Table 1: Example of synthetically generated structures for each configuration. The different colors denote structurally equivalent nodes. Dashed lines denote perturbed graphs (obtained by randomly adding edges).

Node classification synthetic dataset

- All graphs are cycle of length 40
- 10 instances of a shape per graph
- 20 Graphs for each setup

Node classification on synthetic dataset: Results

Method	Configuration							
	basic		basic perturbed		varied		varied perturbed	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
RolX	**1.000**	**1.000**	0.928	0.886	0.998	0.996	0.856	0.768
struc2vec	0.784	0.708	0.703	0.632	0.738	0.592	0.573	0.412
GraphWave	0.995	0.993	0.906	0.861	0.982	0.965	0.793	0.682
2-GNN	0.997	0.994	0.920	0.876	0.990	0.979	0.852	0.753
3-GNN	0.997	0.994	0.911	0.859	0.993	0.985	0.866	0.775
ChebNet (K=2)	0.988	0.979	0.866	0.787	0.852	0.732	0.624	0.471
ChebNet (K=3)	0.992	0.987	0.904	0.850	0.958	0.917	0.758	0.612
ARMA (T=2)	0.996	0.992	0.914	0.861	0.982	0.961	0.839	0.728
ARMA (T=3)	0.997	0.996	0.919	0.872	0.993	0.987	0.850	0.747
2-hop GNN	**1.000**	**1.000**	0.961	**0.934**	**0.999**	**0.999**	0.948	0.910
3-hop GNN	**1.000**	**1.000**	**0.962**	**0.934**	0.996	0.993	**0.952**	**0.916**

Table 2: Performance of the baselines and the proposed k-hop GNN models for learning structural embeddings averaged over 20 synthetically generated graphs for each configuration.

Node classification on real-world dataset

- Enron dataset
- Email network encoding communication between employees in a company
- Nodes are employees and edges are email communication between them
- 143 nodes and 2583 edges
- Goal is to predict the function of the employee from 7 possible options

Node classification on real-world dataset: Results

Method	Accuracy	F1-score
RolX	0.264	0.154
struc2vec	0.323	0.190
GraphWave	0.257	0.149
2-GNN	0.357	0.183
3-GNN	0.366	0.195
ChebNet (K=2)	0.342	0.179
ChebNet (K=3)	0.360	0.191
ARMA (T=2)	0.374	0.192
ARMA (T=3)	**0.376**	0.190
2-hop GNN	0.366	**0.198**
3-hop GNN	0.327	0.171

Table 3: Performance of the baselines and the proposed k-hop GNN models for learning structural embeddings on the Enron dataset.

Graph classification on synthetic dataset

- 3 datasets one for each property of
  - Connectivity
  - Bipartiteness
  - Triangle freeness
- Each of 800 4–regular graphs of 60 nodes
- Each graph is assigned a label indicating whether it has the property or not

Graph classification on synthetic dataset: Results

Method	Property		
	Connectivity	Bipartiteness	Triangle freeness
2-GNN	55.00 ± 5.30	53.78 ± 2.61	51.87 ± 7.43
3-GNN	56.20 ± 2.28	58.13 ± 2.10	55.90 ± 4.44
ChebNet (K=2)	56.37 ± 7.76	50.33 ± 1.20	53.12 ± 6.35
ChebNet (K=3)	57.62 ± 3.84	51.98 ± 3.56	54.75 ± 7.14
ARMA (T=2)	55.55 ± 5.59	54.50 ± 4.61	53.00 ± 3.18
ARMA (T=3)	55.63 ± 5.69	53.92 ± 3.22	54.25 ± 5.80
2-hop GNN	81.24 ± 5.22	84.69 ± 1.74	$\mathbf{84.06} \pm 2.12$
3-hop GNN	$\mathbf{94.77} \pm 3.41$	$\mathbf{91.12} \pm 2.76$	82.53 ± 5.33

Table 4: Average classification accuracy of the proposed k-hop GNN models and the baselines on the 3 synthetic datasets.

Graph classification on real-world dataset

- 3 datasets from bioinformatics and chemoinformatics
  - MUTAG: 188 mutagenic nitro compounds classified based on their effect of Salmonella
  - PROTEINS: 1113 proteins represented as graphs classified as enzymes and non-enzymes
  - NCI1: 4110 chemical compounds screened for activity against some types of cancer
- 2 Social interaction datasets
  - IMDB-BINARY: 1000 movie collaboration graphs
  - IMDB-MULTO: 1500 movie collaboration graphs
  - Goal: To predict the genre of the graph

Graph classification on real-world dataset

Dataset	Statistics					
	#Graphs	#Classes	Max Class Imbalance	Avg. #Nodes	Avg. #Edges	Labels (Num.)
MUTAG	188	2	1 : 1.98	17.93	19.79	+ (7)
PROTEINS	1,113	2	1 : 1.47	39.06	72.82	+ (3)
NCI1	4,110	2	1 : 1	29.87	32.30	+ (37)
IMDB-BINARY	1,000	2	1 : 1	19.77	96.53	–
IMDB-MULTI	1,500	3	1 : 1	13.00	65.94	–

Table 5: Summary of the 5 datasets used in our experiments. The "Max Class Imbalance" column indicates the ratio of the size of the smallest class of the dataset to the size of its largest class.

Graph classification on real-world dataset: Results

Dataset / Method	MUTAG	PROTEINS	NCI1	IMDB BINARY	IMDB MULTI	Average Rank
GK	69.97 (± 2.22)	71.23 (± 0.38)	65.47 (± 0.14)	60.33 (± 0.25)	36.53 (± 0.93)	16.8
SP	84.03 (± 1.49)	75.36 (± 0.61)	72.85 (± 0.24)	60.21 (± 0.58)	39.62 (± 0.57)	12.8
WL	83.63 (± 1.57)	73.12 (± 0.52)	84.42 (± 0.25)	73.36 (± 0.38)	**51.06** (± 0.47)	6.8
WL-OA	86.63 (± 1.49)	75.35 (± 0.45)	**85.74** (± 0.37)	73.61 (± 0.60)	50.48 (± 0.33)	3.0
GS-SVM	83.57 (± 6.75)	74.11 (± 4.02)	79.14 (± 1.28)	71.20 (± 3.25)	48.73 (± 2.32)	10.8
2-GNN	85.92 (± 2.19)	75.24 (± 0.45)	76.32 (± 0.41)	71.40 (± 0.74)	47.73 (± 0.86)	8.8
3-GNN	85.74 (± 1.48)	74.59 (± 0.71)	79.62 (± 0.45)	71.60 (± 0.84)	47.33 (± 1.01)	9.4
ChebyNet (K=2)	85.33 (± 1.42)	74.72 (± 0.97)	78.97 (± 0.35)	71.08 (± 0.51)	47.08 (± 0.60)	10.6
ChebyNet (K=3)	82.49 (± 1.52)	74.81 (± 0.82)	81.01 (± 0.39)	70.90 (± 0.73)	46.66 (± 0.59)	10.8
ARMA (T=2)	82.98 (± 1.90)	74.84 (± 0.59)	80.83 (± 0.42)	70.62 (± 0.95)	46.10 (± 0.82)	11.2
ARMA (T=3)	81.52 (± 1.22)	74.74 (± 0.67)	81.34 (± 0.38)	70.52 (± 0.71)	46.12 (± 0.98)	11.6
PatchySan ($k = 10$)	**88.95** (± 4.37)	75.00 (± 2.51)	76.34 (± 1.68)	71.00 (± 2.29)	45.23 (± 2.84)	9.6
DGCNN	85.83 (± 1.66)	75.54 (± 0.94)	74.44 (± 0.47)	70.03 (± 0.86)	47.83 (± 0.85)	9.6
CapsGNN	86.67 (± 6.88)	**76.28** (± 3.63)	78.35 (± 1.55)	73.10 (± 4.83)	50.27 (± 2.65)	5.0
1-2-3-GNN	86.1	75.5	76.2	**74.2**	49.5	6.0
2-hop GNN	87.93 (± 1.22)	75.03 (± 0.42)	79.31 (± 0.57)	73.33 (± 0.30)	49.79 (± 0.25)	5.4
3-hop GNN	87.56 (± 0.72)	75.28 (± 0.36)	80.61 (± 0.34)	-	-	4.8

Table 6: Average classification accuracy (± standard deviation) of the baselines and the proposed k-hop GNN models on the 5 graph classification benchmark datasets. The "Average Rank" column illustrates the average rank of each method. The lower the average rank, the better the overall performance of the method.

Research Questions

- What are the restriction for k-hop GNNs to identify isomorphic graphs?
- What other fundamental graph properties can we evaluate k-hop GNNs against?
- What is the formal representation of the expressive power of k-hop GNNs?
- What is the practical use cases where k-hop GNNs provide real value?

UNIVERSITY OF WATERLOO

FACULTY OF MATHEMATICS

Thank you for your attention,
Questions?