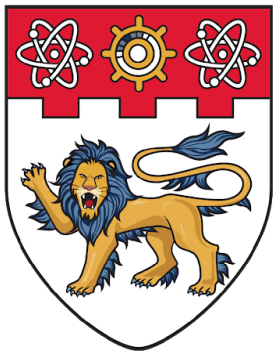

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4071: NETWORK SCIENCE PROJECT 2

HOW POWERFUL IS THE "*HOW POWERFUL ARE GRAPH NEURAL NETWORKS?*" PAPER?

TAN KIAT HWE (U1820766K)

FONG HAO WEI (U1821396H)

HUANG TIANCHENG (U1820154D)

KOH TAT YOU @ ARTHUR KOH (U1821604B)

Contents

1. Problem studied in the paper

1.1. Nascency & lack of formal analytical basis

In this ICLR 2019 paper, a group of Stanford and MIT researchers have attempted to address an issue that plagues the nascent field of Graph Neural Networks (GNNs): *the lack of a formal analytical basis for GNNs*. This lack of a formal basis is the reason why most GNNs have thus far been largely designed using a combination of lacklustre methods, including *empirical intuition*, *heuristics* and *experimental trial-and-error*. These three methods further highlight a key underpinning issue: *the lack of a theoretical understanding about the capabilities and limitations of GNNs*.

1.2. Graph Neural Networks

Graph Neural Networks (GNNs) refer to a multitude of deep learning methods that perform graph-based data inference. GNNs leverage on neural networks that are applied directly onto graphs to perform predictive tasks. These neural network structures are composed of multiple GNN layers. Each GNN layer can be formulated with two formulations as follow:

$$a_v^{(k)} = AGGREGATE^{(k)}(\{h_u^{(k-1)} : u \in N(v)\}) \qquad h_v^{(k)} = COMBINE^{(k)}(\{h_v^{(k-1)}, a_v^{(k)}\})$$

$h_v{}^{(k)}$ is a feature vector of node at the k-th iteration. $h_v{}^{(0)}$ is the original node features X_v , and $N(v)$ is the set of one-hop neighbours. In GNNs, the choice of functions are pivotal in determining the *discriminative power* of GNNs. This point about discriminative power is linked to the concept of injectivity, which will be further elaborated in Section 1.5. As stated before, the choosing of functions within a GNN is not an exact science due to the lack of a theoretical understanding.

1.3. Measuring representational capability: Aggregation Scheme

To tackle the issue of this theoretical understanding gap, the paper introduces a term for measuring representational capability in GNNs: *aggregation scheme*. A GNN's aggregation scheme is defined and abstracted as a class of functions over multisets that are representable by said GNN's neuronal network structure. This is an important metric that will be later used in this paper's most significant contribution: a theoretical framework, which will be later explained in Section 2.

1.4. Aggregation Scheme: Expressiveness & IFM Capability

A GNN's *aggregation scheme* can be further defined in terms of two other sub-characteristics, *Expressiveness* and *Injective Function Modelling (IFM) Capability.* Both terms are detailed as follow:

Expressiveness is a term used to represent the capability of a GNN's *aggregation scheme* to accurately capture graph structures of interest. A graph that is known to be maximally expressive

IFM Capability is a term used to describe the capability of a GNN's *aggregation scheme* to prevent mappings between two different neighborhoods to the same representation. This

is thus reasoned to be one that is able to generalize well.

capability to prevent such mappings is known as *injectivity*.

The above-mentioned two characteristics are crucial components for the evaluation of a robust and powerful GNN. A *maximally* powerful GNN is known as one that has an *aggregation scheme* that enables it to balance between and perform two seemingly divergent tasks at the same time: 1) to generalize well through expressiveness, 2) yet prevent unwanted mappings through injectivity.

1.5. Injectivity and the weaknesses of most known GNN variants

As initially mentioned, injectivity is an important measure of a GNN's discriminative power. As such, this paper also focuses on using injectivity in the evaluation of other known popular GNN variants. For instance, through this paper's theoretical framework, the aggregation scheme used by GCN and GraphSAGE have been determined to be non-injective through ablation studies. These studies have also shown that these non-injective GNN variants can suffer from being cripplingly confused by surprisingly simple graph cases.

2. Most significant contributions

2.1. A theoretical framework to close the evaluative gap

In an attempt to address this theoretical understanding gap, the paper's most significant contribution is its attempt to introduce a formal basis of analysis for GNNs through a *theoretical framework*. Taking inspiration from a test; the *Weisfeiler-Lehman (WL) Graph Isomorphism Test*, this paper's proposed theoretical framework works by analyzing the previously-mentioned *aggregation scheme*.

2.1.1. What is the *Weisfeiler-Lehman* Test?

The *Weisfeiler-Lehman* (WL) Graph Isomorphism Test is a test that is known to be an effective and computationally efficient substitute for performing graph isomorphism (barring edge cases). This graph isomorphism process is necessary for graph distinguishment, which is performed by the WL test through node label comparison. Prior to this comparison, two iterative steps must be done, namely: 1) node label and node neighborhood aggregation and 2) hashing of aggregated labels into unique new labels. Non-isomorphism is determined if and only if the labels of the nodes between any two graphs are different.

2.1.2. Why the need for the WL Test over Graph Isomorphism?

Graph isomorphism is an equivalence relation that seeks to identify whether any two graphs are topologically identical. In theory, graph isomorphism would thus be suitable for the task of graph distinguishment, except for two glaring issues: it is currently not NP-complete, and it currently lacks a polynomial time solution. To perform graph distinguishment would require the problem of Graph Isomorphism to be solvable within reasonable time, which is not possible due to the

absence of a current solution. As such, the WL test serves to fulfil a vast majority of what Graph Isomorphism is able to achieve within a computationally reasonable time period.

2.1.3. Crux of the WL Test

The WL Test's crux is its node label comparison implementation, which enables *injective aggregation update*. This is a powerful mapping characteristic that allows the WL test to map different node neighborhoods to different feature vectors. This powerful mapping characteristic is known as the previously mentioned *aggregation scheme* and it is used as a necessary evaluative measure in determining whether or not a GNN has the same discriminative powers as that of the WL test. That is to say, a GNN that matches the WL Test must map two nodes to the same embeddings only when their neighbourhood is of the same *multiset*.

The theoretical framework proposed in this paper is its most important contribution, as it attempts to formalize the evaluation of different GNN variants through an evaluative metric of comparison. This evaluative metric of comparison comes in the form of an *aggregation scheme,* which can in turn be decomposed into two sub-characteristics, namely: *Expressiveness* and *IFM Capability*. In particular, this framework tries to show how the non-injectivity of these GNN variants' aggregation schemes are the reason for why certain GNN variants are less powerful than the WL test.

2.2. Beyond theoretical evaluative analysis

Rather than just stopping at proposing a theoretical framework for evaluating GNNs through their aggregation schemes, this paper further attempts to illustrate the usefulness of their framework by using the extrapolated key characteristics from their framework's analysis output in an attempt to formalize and realize a simple, yet maximally powerful GNN architecture: *Graph Isomorphism Network* (GIN).

2.3. Graph Isomorphism Network

The Graph Isomorphism Network (GIN) is this paper's attempt at proposing a network architecture discovered through a formal framework used for evaluating GNNs. The architecture of GIN, in comparison with two other popular GNNs are diagrammed below as follow:

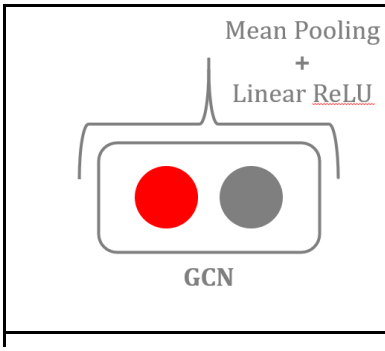

Figure 1.1. Simplified GCN Architecture

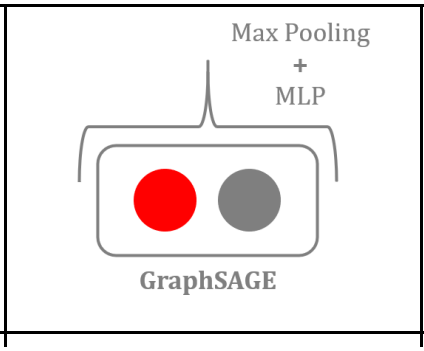

Figure 1.2. Simplified GraphSAGE Architecture

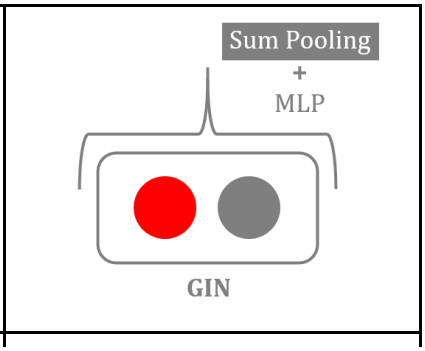

Figure 1.3. Simplified GIN Architecture

2.3.1 Fulfilling Expressiveness

We can see from the diagramed figures above that the GIN architecture is not radically different from its GCN and GraphSAGE counterparts. The GIN architecture borrows the usage of a multi-layer perceptron final layer from GraphSAGE's architecture, as attributed in the paper in *Lemma 5*, due to the Universal Approximation Theorem.

The reason for this is due to *Expressiveness,* as MLPs are considered to be a better approximation of multi-set functions. These multi-set functions can in turn be parameterized into unique embeddings which are better able to capture structural similarities of graphs. In contrast, GNNs such as GCN use Linear ReLU, which behave more similarly to 1-layer perceptron linear mappings that sum neighbourhood features in the final output layer. Due to lack of dimensionality in the output layer, non MLP-based GNNs are unable to adequately capture structural information from graphs. As GINs use MLPs, they are thus able to fulfil the criterion of Expressiveness by addressing the failings of non MLP-based GNNs.

2.3.2. Fulfilling IFM Capability (Injectivity)

However, it is noted that the chosen variant for the GIN architecture's pooling function is distinctive from its GCN and GraphSAGE counterparts. Namely, the *Sum* Pooling function was chosen instead of the GCN's *Mean* and GraphSAGE's *Max* pooling function variants. This differing design choice for the GIN architecture's pooling function was not made through lacklustre methods like empirical intuition, heuristics and experimental trial-and-error. Instead, this specific design choice of choosing *Sum* Pooling was attributed to this paper's proposed theoretical framework.

The choice of using *Sum* Pooling means that the proposed GIN architecture would be able to better capture unique structural features of graphs whilst simultaneously retaining an increased density of structural information. This is because *Sum* Pooling allows for every node feature to be treated as unique, which allows for *characteristically unique values* to be returned when the *Sum* Pooling function performs value aggregation of graph features. These characteristically unique values are *more representative* of a graph, and are akin to the detail that is captured in *Graph Isomorphism*. Thus, this ability to accurately characterize a graph's distinctiveness is what allows GIN to theoretically supersede Mean/Max Pooling reliant GNNs, and thus fulfil the criterion of *IFM Capability.*

2.3.3. Why does GIN generalize the WL Test?

To answer this question, we must first look into the operational details behind GIN and the WL Test. The proposed GIN architecture updates node features by aggregating the features of its neighbours through a process known as *Injective Neighbour Aggregation*. This aggregation is achieved through combining the injective aggregate with Sum Pooling, and it also means that GIN leverages upon the usage of *multiset injective functions*.

This point about multiset injective functions is important, as correspondingly, the WL Test also has an operational equivalent. In particular, the WL Test uses an iterative graph recolouring algorithm. In this algorithm, the node labels (colours) and their neighbourhoods are first aggregated as multisets. Then, the aggregate labels are hashed into unique labels. Because of this algorithmic process, it can be seen that only multiset injective functions are generalizable to the WL Test.

Noting what has been explained about GIN and the WL Test, it can thus be seen that GIN and the WL Test are both operationally equivalent. This means that GIN should theoretically be able to distinguish graphs that the WL Test can distinguish, and vice-versa. Henceforth, we can conclude that for a given discriminative power index, the GIN is indeed able to generalize the WL Test.

3. Connections made with CZ4071: Network Science

3.1. Similarities between GNNs and the PageRank Algorithm

In general, a GNN aggregates the feature vectors of each node's neighbours. This draws parallels to the PageRank (PR) algorithm, specifically with regard to PR's key update steps as follow:

$$PR_{k+1}(u) = \sum_{v \in B_u} \frac{PR_k(v)}{|F_v|}$$

F_v is the set of out neighbours of v

B_u is the set of in neighbours of u

$PR_k(u)$ is the PageRank score of node u at iteration k

From the above equation, we see that PR scores are an aggregation of a node's neighbours' features. This means that a node's PR score after *k* iterations is a score that has taken into account of the node's *k*-hop neighbourhood's PR score.

The *k*-iteration PR scores of nodes can then be passed to a READOUT function to obtain an overall representation. This overall representation is also known as a *graph fingerprint*. This graph fingerprint may not be unique, meaning that it cannot determine if two graphs are topologically identical. However, it is noted that two graphs that are non-identical would not share the same fingerprint.

3.2. Using Subgraph Isomorphism to improve Discriminative Power

GNNs like GIN have achieved remarkable results in terms of discriminative power for Graph Isomorphism testing. However, the discriminative power for GIN is still bounded by the WL Graph Isomorphism Test, which is limited by its inability to detect and count graph substructures.

From our understanding in CZ4071, we propose the utilization of subgraph isomorphism to improve on the architectures that form the backbone for existing GNNs and GNN-based variants (such as GCN,

GrapheSAGE, GIN, etc). Given that substructures are often informative for downstream tasks, we could leverage substructural encodings to incorporate domain specific knowledge. The incorporation of this domain knowledge allows for the improvement of discriminative power in GNNs and their associated variants. Using additional structural descriptors associated with Subgraph-Isomorphism-counting-constructed subgraph nodes, we are able to positionally encode the nodes and obtain a local graph structure. We can also then use graph partitioning learnt in CZ4071 to partition the nodes of different graphs into equivalence classes. These equivalence classes are necessary to accurately reflect the topological characteristics across various graph structures. Hence, during message passing within a GNN, we theoretically are able to achieve more discriminative power by evaluating the topological relationship between endpoint nodes.

3.3. Using the WL Test in Graph Mining

In CZ4071, we learnt that Graph Mining is reliant on Subgraph Isomorphism and Graph Isomorphism. The reliance on Graph Isomorphism stems from the need to check whether or not two graph patterns are identical. Because Graph Isomorphism is an open problem that is not known to be P or NP-Hard, we have also learnt that there were many algorithmic approaches, such as Apriori Approaches and Patten Growth Approaches employed in order to attempt to workaround the exponential search space issue that plague the usage of Graph Isomorphism.

While this paper primarily focuses on an evaluative framework that can be used to evaluate and realize GNNs, this paper also introduced us to a possible substitute that can be used for Graph Mining, specifically: the WL Test. The WL Test is known to be able to essentially perform Graph Isomorphism, barring a few specific edge cases. As such, it might be theoretically possible to augment a conventional Graph Mining heuristic approach with the WL Test for checking graph pattern identicality between two graphs. That is to say, in non-edge cases, a Graph Mining approach could be defined in a manner to favour the WL Test approach, whilst falling back on Graph Isomorphism in cases where the WL Test is unable to effectively perform.

The edge cases where the WL Test fails feature graphs with high degree of symmetry. This symmetry could come in the form of chains, complete graphs, tori and stars. Thus, accounting for these specific cases could still allow for a Graph Mining approach to leverage on a maximal amount of benefit from the use of the WL Test when possible, which could be plausibly computationally faster and efficient in comparison to adopting Graph Isomorphism solely for checking identicality between two graph structures.

4. Discussion of Experiment Results

4.1. Training Accuracy Results

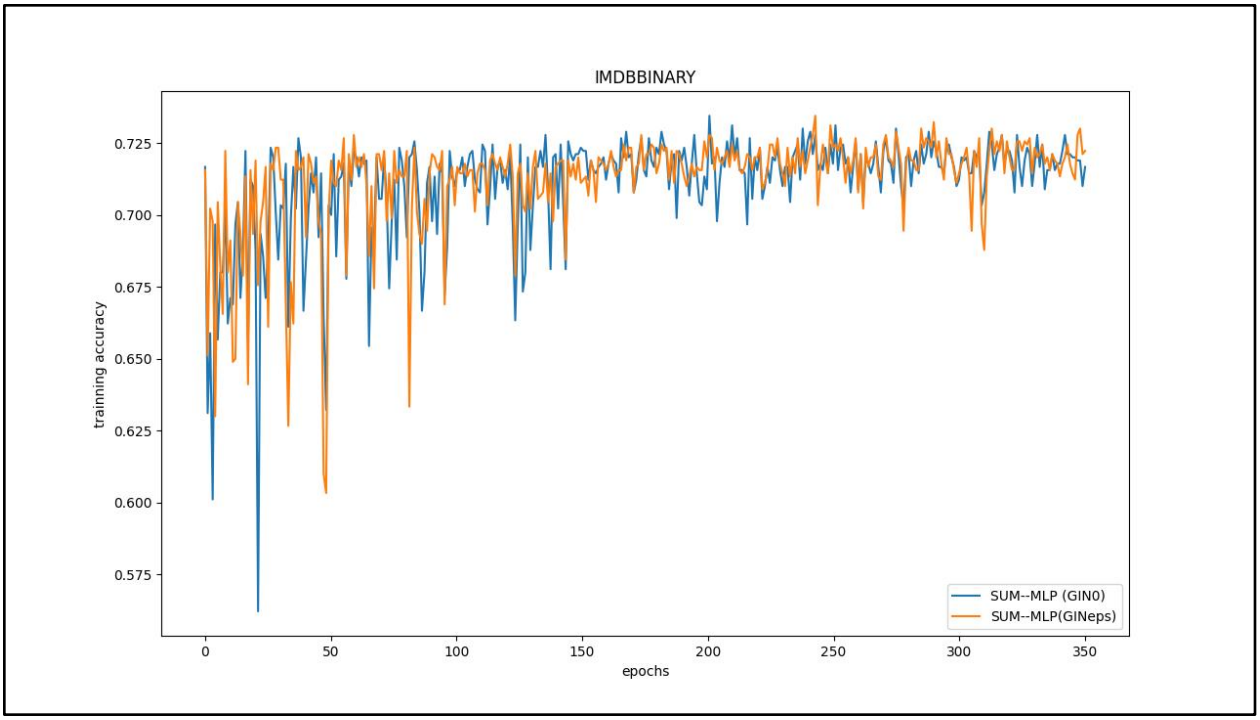

Figure 2.1. Attempted Reproduction of IMDBBINARY results in Figure 4 from the original paper

We attempted to reproduce the experiment results in *Figure 2.1*, which is a singular plot from Figure 4 of the original paper. The results are pertinent to models GIN-ε and GIN-0 from the IMDB-Binary datasets, since these two models are described in greater detail in the original paper.

In our attempted reproduction of their experiment, both models failed to converge to nearly 100% training accuracy. This is different from the experiment results described in the paper. This could be the result of hardware differences, or perhaps a difference in hyper-parameter choice. It is noted that in our experiment, we used the default hyper-parameters provided in the original code for this paper.

As we can see from Figure 2.1, the MLP-GIN-0 seems to have higher variance values than the GIN-ε's variance values. One possible reason for this variance could be because GIN-ε is able to use gradient descent to learn the value of ε, while GIN-0 does not attempt to learn the value of ε, as it fixes the value of ε at 0.

Despite these differing variance values, both models achieve roughly the same training accuracy, of 0.725. Our reproduced code is based on the source code provided by the original paper and we made some modifications. The code and instructions to run it is provided with the report.

4.2. Testing Accuracy Results

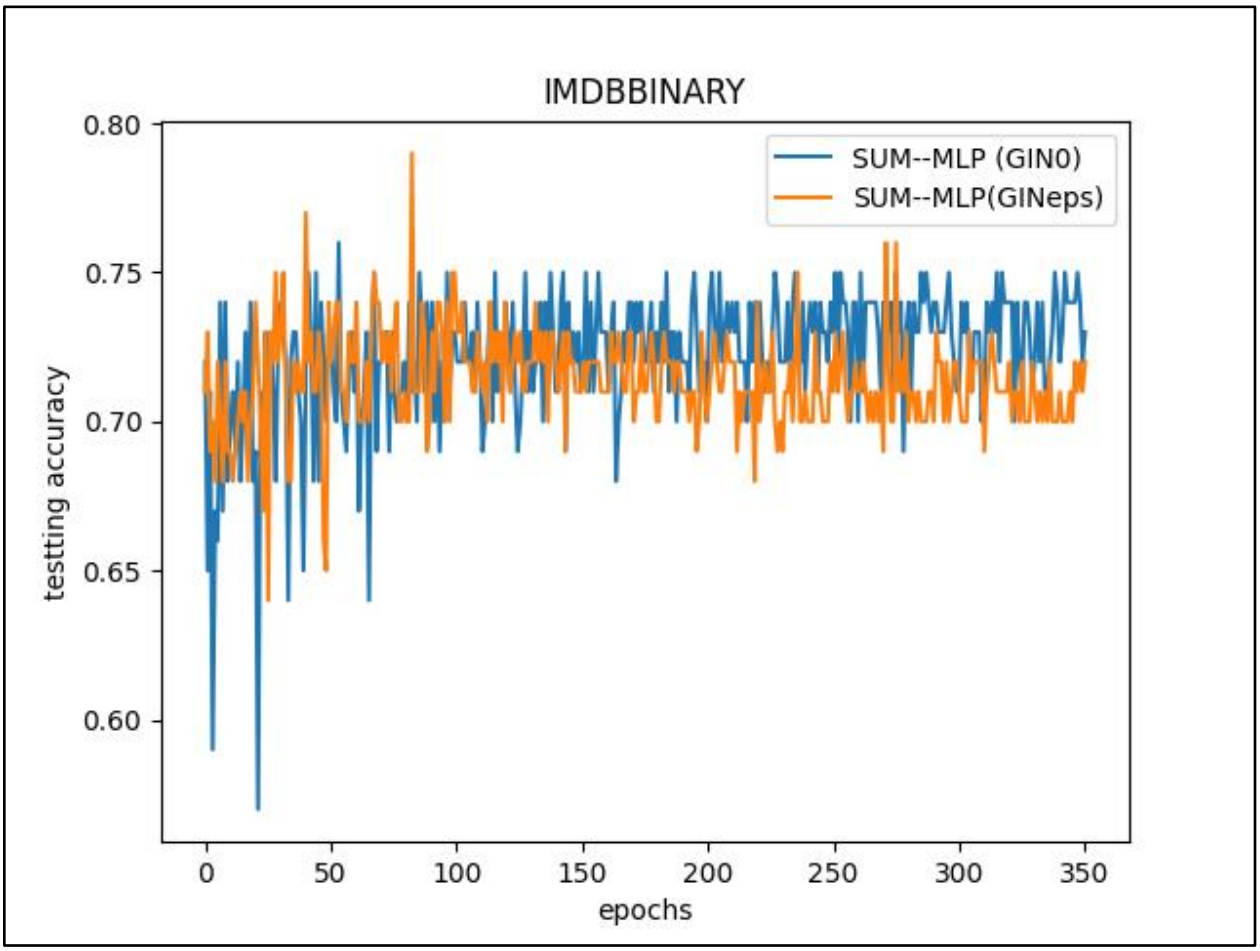

Figure 2.2. Testing Accuracy of IMDBBINARY results

Although the original paper does not contain the testing accuracy in Figure 4, we have decided to attach these results for a more rounded evaluation of our results.

As shown in Figure 2.2, GIN-0 is able to outperform GIN-ε after the results have converged. This point corroborates with what was mentioned in the original paper.

The converged testing accuracy for GIN-0 is approximately 0.75, while the converged testing accuracy for GIN-ε is approximately 0.70. Both are close to the results described in Table 1 of the original paper, where the testing accuracy for GIN-0 is 75.1 ± 5.1 and 74.3 ± 5.1 for GIN-ε.