FINAL REPORT: Team 30

A MITTAL[1], A CHAUDHURI[2], P MALHOTRA[3] & Z AHMED[4]

How Powerful are Graph Neural Networks?

CONTENTS

ABSTRACT

In this report, we continued theoretical foundations for reasoning about the expressive power of GNNs[1], and proved tight bounds on the representational capacity of popular GNN variants. We also designed a provably maximally powerful GNN under the neighborhood aggregation framework.

We present the code and framework(s) used as a part of the project and talk about the models and algorithms used, in general. At the end of the report we present our results and describe our justifications for the same.

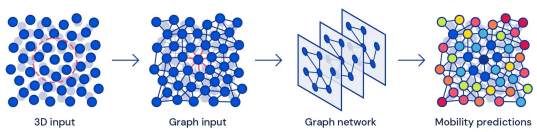


Figure 1: Graph ML

[1] *Abhishek Mittal, SPCRC, IIITH*
[2] *Alapan Chaudhuri, CSTAR & CQST, IIITH*
[3] *Pulak Malhotra, PRECOG, IIITH*
[4] *Zeeshan Ahmed, SPCRC, IIITH*

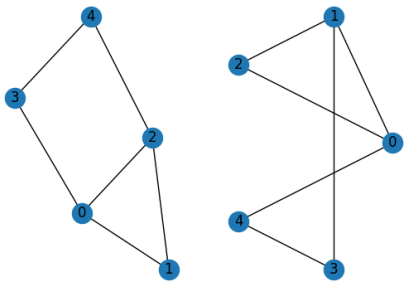


Figure 2: Isomorphic Graphs

1 INTRODUCTION

A lot of the present-day data can be naturally represented as graphs due to its inter-dependencies in data. Fields such as chemistry and biology model problems in terms of graphs. This means there is a need for creating algorithms that are able to interpret large graphs and provide embedding that represent the same in lower dimensions. Constructing lower dimensions allows the possibility of solving problems like graph classification using neural networks[2].

2 WEISFEILER-LEHMAN TEST

The power of a GNN can be quantified by its ability to map different graph structures to different representations in the embedding space. This is similar to the graph isomorphism problem. WL graph isomorphism test[3] is a powerful heuristic and is known to work for almost all the graphs with regular graphs as exceptions.

Below is a lemma to show that the power of WL Test is an upper bound to the power of a GNN.

Lemma *Let* G_1 *and* G_2 *be any two non-isomorphic graphs. If a graph neural network* $\mathcal{A} : \mathcal{G} \to R^d$ *maps* G_1 *and* G_2 *to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides* G_1 *and* G_2 *are not isomorphic.*

Proof. We start off with an assumption that the GNN is able to distinguish G_1 and G_2 at the k^{th} iteration whereas the WL Test fails.

3 GRAPH NEURAL NETWORKS

Graph Neural Networks[2] are iterative models used for graph classification. They are inspired from Convolutional neural networks and work on the similar basis of extracting local information across multiple patches to create a lower dimensional embed for the entire graph. These lower level embeds are created with the help of pooling functions that create an embed for each node based on it's neighbouring nodes. These node level embeds are constructed on the basis of pooling functions.

3.1 Aggregate

Aggregate functions[1] is a function take takes in the embeds of neighbouring nodes and gives out a new embed for the node. Due to the iterative nature of GNNs,

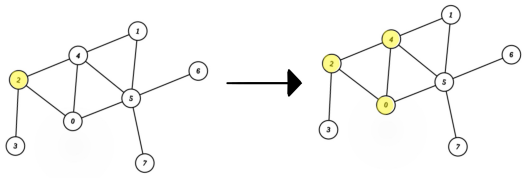


Figure 3: Increase in node influenced for node 3

the embeds are influenced by further nodes after each iteration. The increase in influence on the embed of node 3 is shown in 3

$$a_v = \text{AGGREGATE}(\{h_u \ : \ u \in N(v)\})$$

3.2 Combine

The function AGGREGATE does not account the node's own embeds, so a combine function combines the embed based on the neighbours with the features of the node itself.

$$a_v = \text{COMBINE}(a_v, h_v)$$

3.3 Readout

Since we are dealing with graph-classification, we require a embed for the entire graph. The above functions have only given us embeds for individual nodes, Readout functions takes the embeds of individual nodes and constructs an embed for the entire graph.

Most common choice of READOUT is sum of the node embeds.

$$h_G = \text{READOUT}(h_v | v \in G)$$

4 IMPLEMENTATION DETAILS

The original paper[1] compares variants of GNN on multiple datasets, the classification benchmark was set by WL test. Graph Neural Network is a class of models which can be tweaked base on heuristics. Since the base concept is same, a base class for a GNN and a GNN layer has been implemented.

4.1 Data Format

The data is parsed into networkx objects. For the graphs with no node features, the features are set to one-hot zero vector of labels if any.

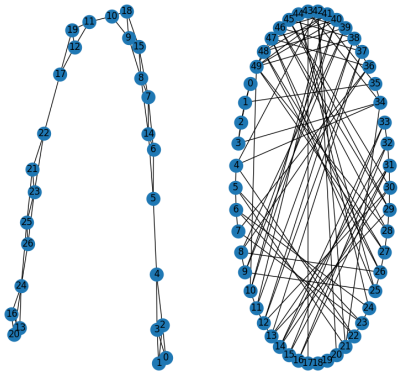


Figure 4: Graph structure of the proteins

4.2 Graph Neural Networks

A graph neural network's primary features are it's AGGREGATE, COMBINE and READOUT functions. Each model follows the iterative method of creating the graph embedding with varying key functions listed above.

4.2.1 Model Layers

A graph embedding is generated at each iteration with each iteration capturing features of more nodes. GNNs do a k-hop at every iteration to get better results in lesser iterations. To increase modularity, a general class for layers has been implemented.

```
class GNNLayer(nn.Module):
    def __init__():
        """
        Class for a general layer of a graph neural network that is based on the
        Graph class defined. Serves as a base class for different types of GNN
        layers.
        """


    def aggregate(self, H, N):
        """
        AGGREGATE function that is overloaded.
        """


    def combine(self, H, a):
        """
        COMBINE function that is overloaded.
        """
```

[H]

Aggregate: The base class uses element-wise MAX on the neighbouring embeds as it's AGGREGATE function. Other variants are SUM and AVERAGE that are used in the original paper.

Combine: The base class uses no combine function, this was inspired from GraphSAGE.

4.2.2 Model

The model encapsulates all the layers along with the READOUT function and the simple linear classifier.

```
class GNN(nn.Module):
    def __init__():
        """
        General class for a graph neural network that is based on the Graph
        class defined. Serves as a base class for different types of GNNs.
        """


    def readout(self, H, graph_cumulative):
        """
        Creates embedding for the graph by summing all the embeds of the graph.
        """


    def classify(self, graph_embeds):
        """
        Takes a list of graph embeddings and runs a simple linear
        classifier on it.
        """
```

Readout: The function used in the base class is the SUM of all node embeds.

4.3 WL test

```
class WLSubtreeKernel:
    """
    The Subtree kernel variant of the WL Graph Isomorphism test
    """


    def __init__(self,G1:Graph,G2:Graph,num_iter):
    """
    Takes 2 graph objects and runs the WL test on them num_iter number of times
    """


    def multiset_label_determination(self):
    """
    For each node we will find the its neighbours and group all their labels into
    a multiset
    """


    def sorting(self):
    """
    We define a function which converts the multiset labels into a string and
    then apply another function to map each function to a single multiset label
    """


    def label_compression(self):
    """
    compress all the multiset labels and the current node's label into a new label
```

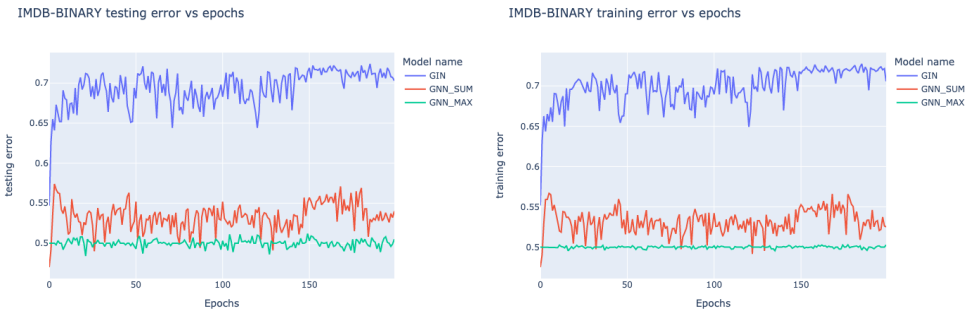
```
    for that node
    """

    def kernel(self):
    """
    It is a function that takes the labels of the nodes of 2 graphs and outputs
    the similarity between the 2 nodes
    """
```

Dataset	IMDB	MUTAG	PROTEINS	PTC_MR
GNN MAX	50.29 ±0.69	95.51 ±2.39	75.42 ±1.23	90.38 ±3.75
GNN SUM	56.67 ±7.07	93.91 ±3.08	75.46 ±1.18	82.53 ±6.12
GIN	72.73 ±0.67	95.74 ±2.52	77.08 ±0.63	77.29 ±6.63
WL	5.5	5.5	74	5.5

Table 1: Training accuracies for all the models on the various datasets

Dataset	IMDB	MUTAG	PROTEINS	PTC_MR
GNN MAX	51.20 ±2.75	86.20 ±6.46	74.92 ±4.09	62.14 ±8.63
GNN SUM	57.4 ±8.56	86.23 ±5.29	75.29 ±3.74	61.94 ±5.70
GIN	72.40 ±3.20	88.33 ±3.88	74.61 ±1.74	60.99 ±7.80
WL	5.5	5.5	74	5.5

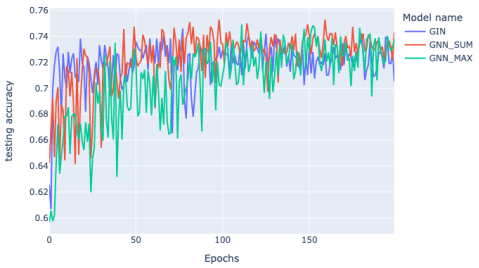
Table 2: Testing accuracies for all the models on the various datasets



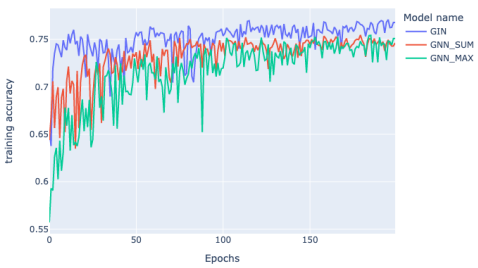
5 RESULTS

Each model was tested on 4 Datasets. The results have been tabulated. As proven in the paper, WL test comprises the upper limit of the capacity of aggregation-based GNNs. The obtained results of WL test outperforming both GNN and GIN stay consistent with the theorem.

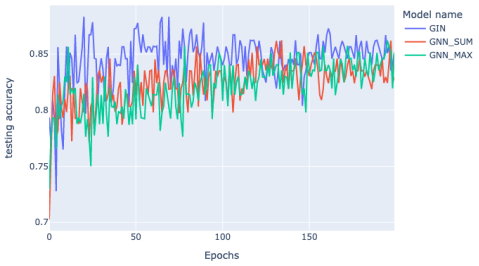
PROTEINS testing accuracy vs epochs



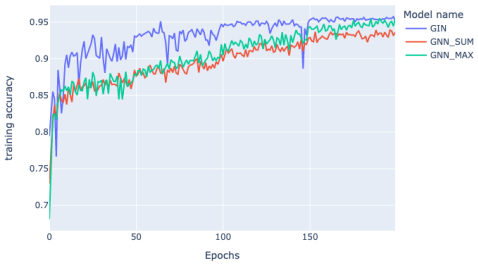
PROTEINS training accuracy vs epochs



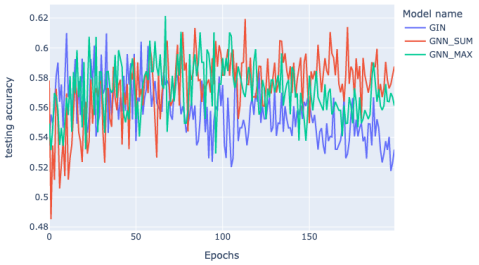
MUTAG testing accuracy vs epochs



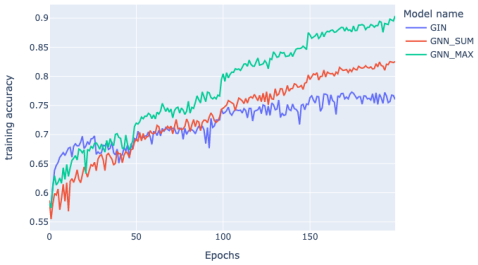
MUTAG training accuracy vs epochs



PTC_MR testing accuracy vs epochs



PTC_MR training accuracy vs epochs



6 INFERENCES

Based on the results, some basic inferences that have been supported theoretically can be seen.

6.1 Max vs Sum

As stated in the paper, SUM AGGREGATE captures more details about the neighbouring nodes than MAX AGGREGATE, this can be seen in testing report.

6.2 WL test and GIN

GIN performance reaches the closest to the WL test performance. The best performance is observed by GIN on MUTAG set.

7 CONCLUSION

In conslusion, through this project we studied Graph Neural Networks and how good they are in terms of the problem of graph classification. We programmed a Graph Insomorphism Network and tested its accuracy on a multitude of datasets over the Weisfeiler-Lehman test.

We also studied how a GIN can be provably a maximally powerful GNN under the neighborhood aggregation framework. Also, as mentioned in the paper itself, possible future work can be looking into realms beyond neighborhood aggregation or message passing.

REFERENCES

[1] Xu et. al. How Powerful are Graph Neural Networks? 2019.

[2] 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, 2009.

[3] B. L. Douglas. The Weisfeiler-Lehman Method and Graph Isomorphism Testing, 2011.