Graph Isomorphism Network

Jerrin John, Lokesh Gautham, Vishva Saravanan, Sanjai Kumaran

Abstract

Graph Neural Networks (GNNs) are an effective framework for representation learning of graphs. GNNs broadly follow a recursive neighborhood aggregation (or message passing) scheme, where each node aggregates feature vectors of its neighbors to compute its new feature vector. Many GNN variants have been proposed and have achieved state-of-the-art results for many tasks such as node classification, link prediction, and graph classification. The authors of HOW POWERFUL ARE GRAPH NEURAL NETWORKS? show that these popular variants of GNNs cannot learn to distinguish certain simple graph structures. They've also developed a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the WeisfeilerLehman graph isomorphism test. Our objective in this project is to understand the proposed architecture. We've implemented the architecture and compared its performance with benchmark GNNs across benchmark datasets. We've also collected other social network based datasets to verify the performance of the proposed architecture.

Link to repository: GitHub

I. Primer on GNNs and their variants

Feature vectors of a set of neighboring nodes form a multiset (a set with possibly repeating elements) because the same element can appear multiple times since different nodes can have identical feature vectors. To have strong representational power, a GNN must be able to aggregate different multisets into different representations. Discriminative power is characterized by how well different aggregation functions can distinguish different multisets. The more discriminative the multiset function is, the more powerful the representational power of the underlying GNN. Intuitively, the most powerful GNN should map two nodes to the same embedding only when their neighborhoods are the same multiset, i.e. the mapping is injective. The design of new GNNs is mostly based on empirical intuition, heuristics, and experimental trial-and-error. There is very little theoretical understanding of the properties and limitations of GNNs, and formal analysis of GNNs' representational capacity is limited.

II. Theoretical understanding of GNNs

Let $G = (V, E)$ denote a graph with node feature vectors X_v for $v \epsilon V$. There are 2 tasks of interest: Node classification: Where each node $v \epsilon V$ has an associated label y_v and the goal is to learn a representation vector h_v such that label for every v can be predicted as $y_v = f(h_v)$. Graph classification: Given a set of graphs $\{G_1, G_2, ..., G_N\} \subseteq G$ and their associated labels $\{y_1, y_2, ..., y_N\} \subseteq Y$ we aim to learn a representation vector h_G that predicts the label of the graph as $y_G = g(h_G)$. GNN follows a neighborhood aggregation strategy, where we iteratively update the representation of a node by aggregating representations of its neighbors. The K-th layer of GNN is formulated as $a_v^{(k)} = AGGREGATE^{(k)}(\{h_u^{(k-1)} : u \epsilon N(v)\})$ $h_v^k = COMBINE^{(k)}(h_v^{(k-1)}, a_v^{(k)})$ where $h_V^{(k)}$ is the feature vector of the node v at the k-th iteration/layer, h_v^0 is set to be the original node features X_v , $N(v)$ is the set of one-hop neighbors of v. For node classification the node representation $h_V^{(k)}$ of the final iteration is used for prediction. For graph classification, the $READOUT$ function aggregates node features from $h_v^{(k)}$ to obtain the graph representation h_G as $h_G = READOUT(\{h_v^{(K)} | v \forall G\})$ For GraphSage, the $AGGREGATE$ function is formulated as: $a_v^{(k)} = MAX(ReLU(W.h_u^{(k-1)}), \forall u \epsilon N(v))$ The $COMBINE$ step: $W.[h_v^{(k-1)}, a_v^{(k)}]$
For Graph Convolutional Network(GCN), the$AGGREGATE$ and $COMBINE$ steps are integrated as follows: $h_v^{(k)} = ReLU(W.MEAN\{h_u^{k-1}, \forall u \epsilon N(u) \cup \{v\}\})$

III. Primer on WeisfeilerLehman (WL) Test

The WeisfeilerLehman (WL) test decides if two graphs are isomorphic, i.e. topologically identical, or not. The WL test iteratively aggregates the labels of nodes and their neighbourhoods and hases the aggregated labels into unique new labels.The algorithm decides that two graphs are non-isomorphic if at some iteration the labels of the nodes between the two graphs differ. What makes the WL test so powerful is its injective aggregation update that maps different node neighborhoods to different feature vectors.

Weisfeiler-Lehman Graph Kernels proposed the WL subtree kernel that measures the similarity between graphs. The kernel uses the counts of node labels at different iterations of the WL test as the feature vector of a graph and has the same discriminative power as the WL testIntuitively, a node's label at the k-th iteration of WL test represents a subtree structure of height k rooted at the node. Thus, the graph features considered by the WL subtree kernel are essentially counts of different rooted subtrees in the graph.

GNNs are at most as powerful as the WL test in distinguishing graph structures. A GNN can have as large discriminative power as the WL test if the GNN's aggregation, combine and readout schemes are all injective on multisets.

IV. Study of GNNs

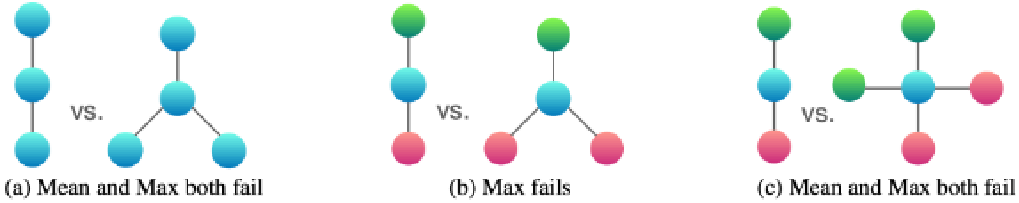
Studies on different aggregation functions (non injective) show that they can't support isomorphism. While GCN performs well on node classification, GNNs like GCN and GraphSAGE are less powerful than WL test.

A. 1-Layer Perceptrons are not sufficient

1-Layer perceptrons are not sufficient to attain isomorphism. This can be proved with a counterexample that two different multisets are possible to get mapped to the same feature vector, thereby not satisfying the isomorphism property.

B. Structures that Confuse Max and Mean Pooling

Counter examples can be provided to show that two different multisets get mapped to the same feature vector.



(a) Mean and Max both fail

(b) Max fails

(c) Mean and Max both fail

Figure 1: Visualising the cases where max and mean pooling fail

In the example above, the same color nodes have the same feature vector. In each of the three cases, though the pair of multisets are clearly distinct, max and mean pooling fail in several cases, thereby contradicting isomorphism.

V. Graph Isomorphism Network (GIN)

Popular GNN variants, such as Graph Convolutional Network (GCN) and GraphSAGE cannot distinguish some graph structures. Graph Isomorphism Network (GIN) is a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the WeisfeilerLehman graph isomorphism test. Choosing the following gives integration of an injective $AGGREGATE$ and $COMBINE$ functions. The $READOUT$ simply considers the node features of all levels.

$$h_v^{(k)} = MLP^{(k)}((1 + e^{(k)}).h_v^{(k-1)} + \sum_{u \epsilon N(v)} h_u^{(k-1)})$$

$$h_G = CONCAT(READOUT(\{h_v^{(k)} | v \epsilon G\}) | k = 0, 1, .., K)$$

VI. Experiments

We have implemented the GIN architecture based on our understanding from the paper. To evaluate the performance of the model, we compare its train set performance with that of other less powerful GNN variants, like Graph Convolutional Networks and GraphSAGE. We also use the test set performance to generalise the ability of the architecture.

A. Datasets

Train, validate and test are implemented using the 9 graph classification benchmarks: 4 bioinformatics datasets (MUTAG, PTC, NCI1, PROTEINS) and 5 social network datasets (COLLAB, IMDB-BINARY, IMDB-MULTI, REDDITBINARY and REDDIT-MULTI5K). The architecture is such that it primarily learns from the network structure rather than the input node features. The bioinformatics datasets contain categorical input node features, whereas social network datasets don't contain input node features. The input node features of the social networking datasets are initialised to zero, which means these features don't add up to any information.

B. Models and configuration

The implemented GIN network learns a learn-able parameter ϵ which distinguishes center nodes from neighboring nodes. The ϵ is learned by gradient descent. We have implemented neighbor nodes pooling and graph pooling using a **sum pool**.

The train-test split is $9:1$ on the entire dataset and the train-validation split is $9:1$ on the train set. The testing is done using several combinations of hyperparameters for tuning them. The number of GNN layers and the number of MLP layers remain 5 and 2 respectively. We have used Adam optimizer and batch norm for every hidden layer to normalize and prepare input data for the layers. The learning rate is fixed as 0.01 because a scheduler didn't bring much of a change in terms of performance. Combinations of hyperparameters like (1) number of hidden layers, (2) batch size, (3) dropout probability and (4) number of epochs were tried out and the finalised hyperparameters are 5 GNN layers (including the input layer), hidden units of size 64, minibatch of size 128, and 0.5 dropout ratio for 350 epochs.

C. Baseline

We compare our test set results with several state-of-art baselines for graph classification. We use (1) WL test as our ideal baseline. Our state-of-art deep neural network architectures include (2) DCNN, (3) PATCHYSAN, (4) DGCNN and (5) AWL.

VII. Experimenting on different datasets

A. Datasets

In addition to the standard benchmark datasets, we have trained and tested the GIN model on 3 different disjoint graph classification datasets:

1) Deezer Ego-nets

A database of ego-nets of Eastern European users collected from the music streaming service Deezer in February 2020. The nodes represent users and edges represent mutual follower relationships. The task is to classify nodes into either of 2 genders.

2) Github Stargazers
The social networks of developers who starred popular ML and WebDev repositories having at least 10 stars as of August 2019. The nodes represent users and edges represent follower relationships. The task is to classify nodes into being a ML developer or a web developer.

3) Twitch Ego-Nets
The ego-nets of Twitch users who were members of the Partnership Program in April 2018. The nodes represent users and edges represent friendships. The binary classification task is to predict whether a node plays singleplayer or multiplayer games.

VIII. Results

A. *Train set performance*

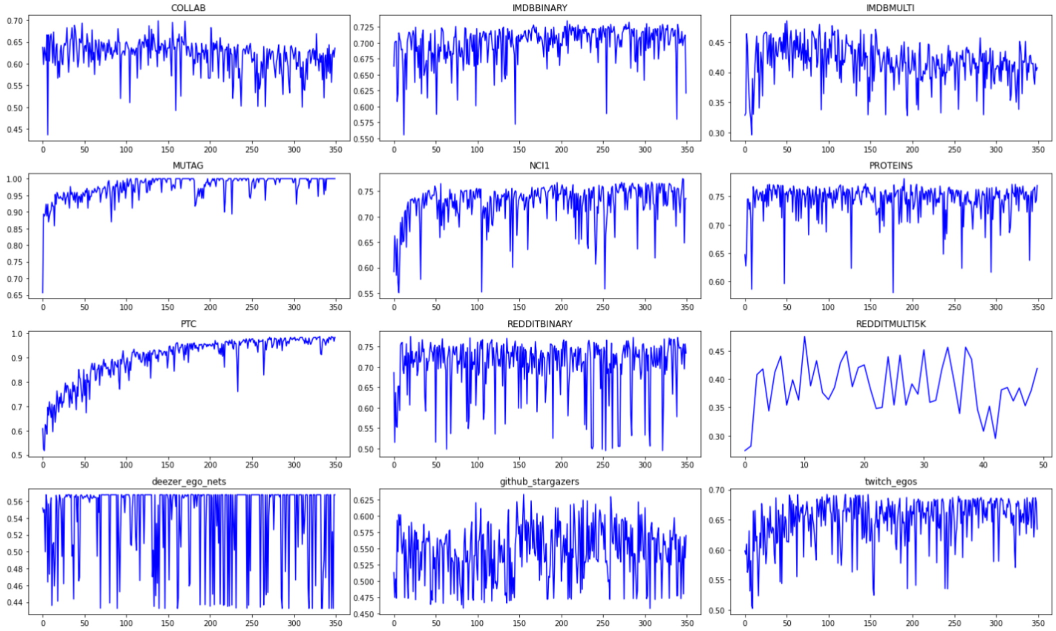


Figure 2: Training set performance of the GIN network

Networks with high representational capability will produce better accuracy. Our theoretical understanding is validated by the results. The implemented GIN boasts a better accuracy, thereby validating its better representational power.

B. Test set performance

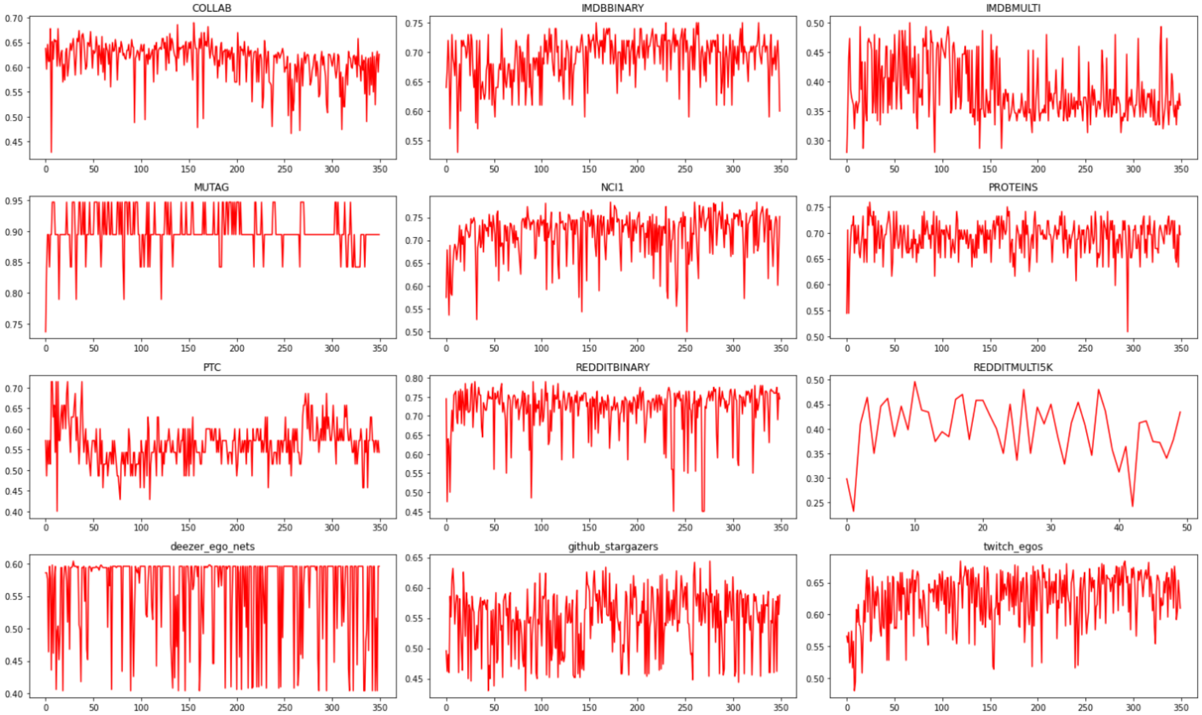
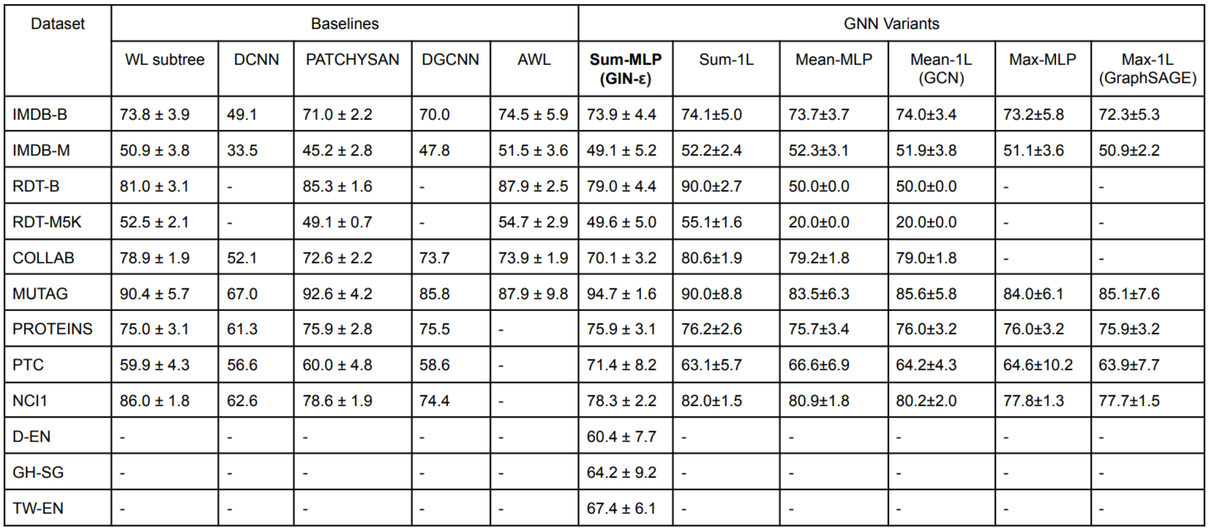


Figure 3: Test set accuracy graph for the GIN network



Dataset	Baselines					GNN Variants					
	WL subtree	DCNN	PATCHYSAN	DGCNN	AWL	**Sum-MLP (GIN-ε)**	Sum-1L	Mean-MLP	Mean-1L (GCN)	Max-MLP	Max-1L (GraphSAGE)
IMDB-B	73.8 ± 3.9	49.1	71.0 ± 2.2	70.0	74.5 ± 5.9	73.9 ± 4.4	74.1±5.0	73.7±3.7	74.0±3.4	73.2±5.8	72.3±5.3
IMDB-M	50.9 ± 3.8	33.5	45.2 ± 2.8	47.8	51.5 ± 3.6	49.1 ± 5.2	52.2±2.4	52.3±3.1	51.9±3.8	51.1±3.6	50.9±2.2
RDT-B	81.0 ± 3.1	-	85.3 ± 1.6	-	87.9 ± 2.5	79.0 ± 4.4	90.0±2.7	50.0±0.0	50.0±0.0	-	-
RDT-M5K	52.5 ± 2.1	-	49.1 ± 0.7	-	54.7 ± 2.9	49.6 ± 5.0	55.1±1.6	20.0±0.0	20.0±0.0	-	-
COLLAB	78.9 ± 1.9	52.1	72.6 ± 2.2	73.7	73.9 ± 1.9	70.1 ± 3.2	80.6±1.9	79.2±1.8	79.0±1.8	-	-
MUTAG	90.4 ± 5.7	67.0	92.6 ± 4.2	85.8	87.9 ± 9.8	94.7 ± 1.6	90.0±8.8	83.5±6.3	85.6±5.8	84.0±6.1	85.1±7.6
PROTEINS	75.0 ± 3.1	61.3	75.9 ± 2.8	75.5	-	75.9 ± 3.1	76.2±2.6	75.7±3.4	76.0±3.2	76.0±3.2	75.9±3.2
PTC	59.9 ± 4.3	56.6	60.0 ± 4.8	58.6	-	71.4 ± 8.2	63.1±5.7	66.6±6.9	64.2±4.3	64.6±10.2	63.9±7.7
NCI1	86.0 ± 1.8	62.6	78.6 ± 1.9	74.4	-	78.3 ± 2.2	82.0±1.5	80.9±1.8	80.2±2.0	77.8±1.3	77.7±1.5
D-EN	-	-	-	-	-	60.4 ± 7.7	-	-	-	-	-
GH-SG	-	-	-	-	-	64.2 ± 9.2	-	-	-	-	-
TW-EN	-	-	-	-	-	67.4 ± 6.1	-	-	-	-	-

Figure 4: Test set performance comparison across 9 benchmark datasets + 3 extra datasets

IX. Conclusion

We developed theoretical and hands-on understanding on the GIN model proposed. We realise that the design of new GNNs is mostly based on empirical intuition, heuristics, and experimental trial-and-error. One direction of future work would be in optimizing the aggregation by expanding beyond just the neighbors. It would be interesting if there is development in understanding the properties and limitations of GNNs, which would help in improving the representational power of GNNs.

X. Distribution of Work

A. Jerrin John

- Explored the bioinformatics and social network benchmark datasets.
- Parsing and pre-processing the datasets to a suitable format.
- Looking for application(s) of GINs and available datasets.
- Collecting the benchmark datasets and splitting them into train, validation and test datasets.

B. Lokesh Gautham

- Implementing parts of the Graph Isomorphism Network (GIN) architecture.
- Training the network by trying out combinations of hyperparameters and evaluating the best model.
- Computing the benchmark performance and accuracy.
- Analysing the benchmark performance and accuracy.
- Comparing with other state-of-the-art baselines for graph classification.

C. Vishva Saravanan

- Training the network by trying out combinations of hyperparameters and evaluating the best model.
- Implementing parts of the Graph Isomorphism Network (GIN) architecture.
- Computing the benchmark performance and accuracy.
- Analysing the benchmark performance and accuracy.
- Comparing with other state-of-the-art baselines for graph classification.

D. Sanjai Kumaran

- Explored the bioinformatics and social network benchmark datasets.
- Parsing and pre-processing the datasets to a suitable format.
- Looking for application(s) of GINs and available datasets.
- Collecting the benchmark datasets and splitting them into train, validation and test datasets.