# HOW POWERFUL ARE GRAPH NEURAL NETWORKS? / ICLR 2019 REPRODUCIBILITY CHALLENGE

**Jaroslaw Dzikowski** [*]
Institute of Informatics
University of Wroclaw
Wroclaw, Poland
273233@uwr.edu.pl

## ABSTRACT

The main principle of Graph Neural Networks (GNN) is representing each node as a feature vector computed by recursive aggregation and transformation of neighbouring nodes' feature vectors. This paper is an attempt at reproduction of *How Powerful are Graph Neural Networks?* paper submitted for ICLR 2019 conference that studies the discriminative power of various aggregation schemes applied so far and provides Graph Isomorphism Network (GIN), a scheme with the most discriminative power according to the reproduced paper. We compare the results obtained by our implementation of GIN with the results provided in the reproduced paper.

## 1 INTRODUCTION

Graph neural networks perform aggregation of neighbouring node features for each node $v$ in graph $G$. The aggregation is performed iteratively $K$ times resulting in each node collecting features of nodes $u$ in $K$ edges radius. After $k$ iterations of aggregation, a nodes representation captures the structural information within its $k$-hop network neighborhood. Formally, the $k$-th layer $h^{(k)}$ of a GNN is

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right), \quad h_v^{(k)} = \text{COMBINE}\left(h_v^{(k-1)}, a_v^{(k)}\right).$$

The aggregated node features from the last, $K$-th, layer can be used for node classification tasks. For graph classification problems, a readout operation is performed on all node feature vectors $h^{(K)}$ in the graph $G$ to obtain the entire graph's representation $h_G$.

$$h_G = \text{READOUT}\left(\left\{h_v^{(K)} | v \in G\right\}\right)$$

The main distinctive features of various GNNs are the choices of initial node features $h^{(0)}$, aggregation scheme and readout scheme. The article Xu et al. (2019) provides a theoretical framework for studying expressiveness of various aggregation schemes and proposes an aggregation scheme, Graph Isomorphism Network (GIN), that is optimal according to the framework. GIN sums up neighbouring nodes' feature vectors and adds them to node's weighted current feature. The weight $\epsilon$ is a learnable parameter. Formally GIN is defined as

$$h_v^{(k)} = \text{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right), \tag{1}$$

where MLP is a multilayer perceptron. The final representation of graph $h_G$ is the concatenation of READOUT performed on each of $K$ aggregation layers:

$$h_G = \text{CONCAT}\left(\text{READOUT}\left(\left\{h_v^{(k)} | v \in G\right\}\right) \mid k = 0, 1, \ldots, K\right). \tag{2}$$

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies. Funding acknowledgements go at the end of the paper.

The READOUT operation can be implemented by a simple sum or mean of nodes' final feature vectors. Aside from GIN aggregation scheme that sums up neighbouring nodes' feature vectors, there exist mean and max aggregation schemes that, according to the reprodced paper's theoretical framework, lack discriminative power compared to GIN and, therefore, are unable to distinguish certain types of graphs as shown in figures 2 and 3 of Xu et al. (2019).

## 2 REPRODUCTION DETAILS

### 2.1 IMPLEMENTATION

Since the authors of Xu et al. (2019) did not provide any source code, we had to implement the GIN and other GNN variants ourselves. We implemented our code in Python 3 using PyTorch framework for neural networks. We leveraged existing implementations of feature vector aggregations and graph level readouts from `pytorch-geometric`[1] library (Fey et al. (2018)). While the details of aggregation layers and readout operations have been provided, there is little information about classification layers that predict the class of a graph from its representation $h_G$. The authors of the reproduced paper mentioned they used cross validation using LIB-SVM library (Which gives a hypothesis that the authors used an SVM as the classification layer), but since we were unable to use it, we implemented cross validation from scratch and used an MLP as the classification layer. Our implementation is publicly available on Github[2].

### 2.2 ENVIRONMENT

Our code was executed on a Google Cloud Platform VM instance of size n1-standard-4 (4 vCPUs backed by Intel Xeon E5 (Sandy Bridge), 15 GB RAM) with NVIDIA Tesla K80 GPU (12 GB VRAM). Detailed parameters of the VM instance's CPU can be found in GCP documentation[3]. The VM instance was deployed with Debian based Deep Learning Image coming with Python 3.7.1, PyTorch 1.0.0, fastai m15 and CUDA 10.0.

### 2.3 DATASETS

There were 9 datasets we measured performance of GIN and other aggregation schemes on 5 social graph datasets and 4 bioinformatics datasets. We downloaded the datasets from TU Dortmund's Benchmark Data Sets for Graph Kernels site[4] containing our 9 datasets and a lot of other graph classification datasets.

#### 2.3.1 SOCIAL DATASETS

The 5 social graph datasets are IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI5K and COLLAB.

- IMDB-BINARY and IMDB-MULTI are movie collaboration datasets. Each graph corresponds to an ego-network for each actor/actress, where nodes correspond to actors/actresses and an edge is drawn betwen two actors/actresses if they appear in the same movie. Each graph is derived from a pre-specified genre of movies, and the task is to classify the genre graph it is derived from.
- REDDIT-BINARY and REDDIT-MULTI5K are balanced datasets where each graph corresponds to an online discussion thread and nodes correspond to users. An edge was drawn between two nodes if at least one of them responded to anothers comment. The task is to classify each graph to a community or a subreddit it belongs to.
- COLLAB is a scientific collaboration dataset, derived from 3 public collaboration datasets, namely, High Energy Physics, Condensed Matter Physics and Astro Physics. Each graph

[1] `https://github.com/rusty1s/pytorch_geometric`
[2] `https://github.com/jdzikowski/iclr2019`
[3] `https://cloud.google.com/compute/docs/cpu-platforms`
[4] `https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets`

corresponds to an ego-network of different researchers from each field. The task is to classify each graph to a field the corresponding researcher belongs to.

Since the nodes in the social datasets do not have any labels or feature vectors, we have to create them ourselves. Same as in the reproduced paper, we set the same label for all nodes in REDDIT datasets (Making node features uninformative). Formally, each node's initial feature vector is a one-hot vector $h_v^{(0)} \in \mathbb{R}^1$ with 1 in its only dimension. For IMDB and COLLAB datasets, we set the initial node feature vectors to one-hot encodings of node degrees.

### 2.3.2 Bioinformatics datasets

The 4 bioinformatics graph datasets are MUTAG, PROTEINS, PTC and NCI1. Each node has a categorical feature

- MUTAG is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds where each node has one of 7 discrete labels. The compounds are divided into two classes according to their mutagenic effect on a bacterium.
- PROTEINS is a dataset where nodes are secondary structure elements (SSEs) and there is an edge between two nodes if they are neighbors in the amino-acid sequence or in 3D space. Every node has one of 29 discrete labels and the task is to classify the proteins into two classes.
- PTC is a dataset of chemical compounds that reports the carcinogenicity for male and female rats and mice. There are actually 4 different PTC datasets, one for each combination of gender and rodent species. In Xu et al. (2019), the authors describe the PTC_MR dataset consisting of 334 chemical compounds reporting carcinogenicity for only male rats. Each node has one of 19 discrete labels and the task is to classify the compounds into two carcinogenicity classes. From now on when referring to PTC dataset we will mean the PTC_MR dataset used by the authors of the reproduced paper.
- NCI1 is a dataset made publicly available by the National Cancer Institute (NCI) and is a subset of balanced datasets of chemical compounds screened for ability to suppress or inhibit the growth of a panel of human tumor cell lines, where each node has one of 37 discrete labels and the task is to classify the compounds into two carcinogenicity classes.

One of anonymous commenters under the OpenReview page[5] for the reproduced paper suggested that the authors should try measuring performance of GIN and other studied aggregation schemes on a larger dataset such as REDDIT-MULTI12K consisting of 12000 graphs with 11 classes. We attempted to measure performance of various GNNs on the dataset, but cross validating performance for just one aggregation scheme took us over a week and we eventually decided to abandon the dataset.

## 2.4 Experiments

Following the same steps as in the reproduced paper, we measure training and test accuracy of GIN and other, less powerful according to Xu et al. (2019), GNNs. Under the GIN framework, we consider two variants: (1) GIN-$\epsilon$ in which $\epsilon$ from equation 1 is a learnable parameter which will be optimized by neural network's backpropagation, and (2) GIN-0, a simpler and less powerful GIN in which the $\epsilon$ is fixed to 0. For the less powerful GNN variants we consider ones that use mean or max-pooling aggregation as opposed to GIN's summation of feature vectors and GNNs that use 1-layer perceptrons (Linear layer followed by ReLU) instead of MLPs in equation 1. In table 3, each GNN is named after aggregation scheme and perceptron it uses. We have total of 7 different GNNs. Same as in the reproduced paper, we only consider graph classification task and apply the same graph level readout (READOUT from equation 2) for GINs and other GNN variants, specifically, sum readout on bioinformatics datasets and mean readout on social datasets.

Same as in Xu et al. (2019), we perform 10-fold cross validation, however, in the reproduced paper the authors used LIB-SVM library for cross validation and classification of graph feature vectors, while we use our own implementation of cross validation and use a two layer MLP with dropout

[5]`https://openreview.net/forum?id=ryGs6iA5Km&noteId=S1evjSRPc7`

layer in between (Linear, ReLU, Dropout, Linear, ReLU) for classification. (((((We report the average and standard deviation of validation accuracies across the 10 folds within the cross-validation: for each epoch we compute mean and standard deviation of accuracies from each of the 10 folds and then we return the maximum mean accuracy across all epochs along with the standard deviation for the best epoch. Following the reproduced paper, for all configurations, we apply 5 aggregation layers and all the MLPs have 2 layers (Linear, Dropout, ReLU, BatchNorm, Linear, Dropout, ReLU, BatchNorm). Same as in the reproduced paper, we use Adam optimizer with initial learning rate 0.01 and decay it by 0.5 every 50 epochs.

There are 4 hyperparameters the authors tuned:

1. The number of hidden units in MLP layers $\in \{16, 32\}$ for bioinformatics datasets and 64 for social datasets.
2. Batch size $\in \{32, 128\}$.
3. Dropout ratio after the dense layer $\in \{0, 0.5\}$.
4. The number of epochs.

The authors of Xu et al. (2019) did not provide the optimal parameters for each GNN variant and dataset, therefore, we had to find them ourselves. Our implementation is capable of hyperparameter tuning, however, due to time constraints and learning proving to be time consuming, we had to give up on finding optimal hyperparameter sets for each configuration and we used the same configuration for all datasets:

1. The number of hidden units in MLP layers was 32 for bioinformatics datasets and 64 for social datasets.
2. Batch size was 128.
3. Dropout ratio after the dense layer was 0.5.
4. We trained each GNN for 350 epochs.
5. For the classification MLP hidden layers had $K \cdot 32$ hidden units for bioinformatics datasets and $K \cdot 64$ for social datasets where $K = 5$ is the number of aggregation layers.

## 3 RESULTS

Table 3 compares the classification accuracies and standard deviations obtained by the authors of Xu et al. (2019) (top) with the results obtained in our reproduction attempt (bottom). There are missing results in both tables: the authors of the reproduced paper were unable to obtain results for max aggregation GNN variants for the three largest datasets, REDDIT-BINARY, REDDIT-MULTI5K and COLLAB, due to GPU memory constraints. In our reproduction attempt, training on the largest two datasets, REDDIT-MULTI5K and COLLAB, proved to be too time consuming and we were unable to obtain results due to time constraints.

The first apparent difference in the results can be observed for the IMDB datasets. Our implementation obtained lower accuracies than the authors'. While accuracies for GINs are not that far from the authors', the biggest contrast is in the results for mean and max-pooling aggregation schemes - these are much worse than the ones reported in the reproduced paper. In fact, they are the same as for random guessing. Notice that mean and max aggregation schemes don't work for REDDIT-BINARY as well. This shows that the discriminative power of mean and max-pooling aggregation schemes is lower than sum aggregation's when node features are either the same or are one-hot encodings of node degrees. With the same feature vectors, the mean and max of neighbouring feature vectors is always going to be the same and will not be informative, the perceptrons we apply feature vectors to during each aggregation are most likely not able to learn anything useful. We do not have any hypothesis explaining why node degrees do not work with mean and max aggregations.

As for bioinformatics datasets, where each node has a categorical label, it can be observed that the results do not diverge that much between different aggregation schemes. The accuracies are also similar to the results obtained by the authors of the reproduced paper. For all the bioinformatics datasets sum aggregation GNN variants yielded the best results. Despite using a simple one layer

| Datasets | IMDB-B | IMDB-M | RDT-B | RDT-M5K | COLLAB | MUTAG | PROTEINS | PTC | NCI1 |
|---|---|---|---|---|---|---|---|---|---|
| # graphs | 1000 | 1500 | 2000 | 5000 | 5000 | 188 | 1113 | 334 | 4110 |
| # classes | 2 | 3 | 2 | 5 | 3 | 2 | 2 | 2 | 2 |
| Avg # nodes | 19.8 | 13.0 | 429.6 | 508.5 | 74.5 | 17.9 | 39.1 | 25.5 | 29.8 |
| GIN-$\epsilon$ (SUM-MLP) | **74.3 ± 5.1** | **52.1 ± 3.6** | **92.2 ± 2.3** | **57.0 ± 1.7** | **80.1 ± 1.9** | **89.0 ± 6.0** | **75.9 ± 3.8** | 63.7 ± 8.2 | **82.7 ± 1.6** |
| GIN-0 (SUM-MLP) | **75.1 ± 5.1** | **52.3 ± 2.8** | **92.4 ± 2.5** | **57.5 ± 1.5** | **80.2 ± 1.9** | **89.4 ± 5.6** | **76.2 ± 2.8** | **64.6 ± 7.0** | **82.7 ± 1.7** |
| SUM-1-LAYER | 74.1 ± 5.0 | **52.2 ± 2.4** | 90.0 ± 2.7 | 55.1 ± 1.6 | **80.6 ± 1.0** | **90.0 ± 8.8** | **76.2 ± 2.6** | 63.1 ± 5.7 | 82.0 ± 1.5 |
| MEAN-MLP | 73.7 ± 3.7 | **52.3 ± 3.1** | 50.0 ± 0.0 | 20.0 ± 0.0 | 79.2 ± 2.3 | 83.5 ± 6.3 | 75.5 ± 3.4 | **66.6 ± 6.9** | 80.9 ± 1.8 |
| MEAN-1-LAYER | 74.0 ± 3.4 | 51.9 ± 3.8 | 50.0 ± 0.0 | 20.0 ± 0.0 | 79.0 ± 1.8 | 85.6 ± 6.3 | 76.0 ± 3.2 | 64.2 ± 4.3 | 80.2 ± 2.0 |
| MAX-MLP | 73.2 ± 5.8 | 51.1 ± 3.6 | - | - | - | 84.0 ± 6.1 | 76.0 ± 3.2 | 64.6 ± 10.2 | 77.8 ± 1.3 |
| MAX-1-LAYER | 72.3 ± 5.3 | 50.9 ± 2.2 | - | - | - | 85.1 ± 7.6 | 75.9 ± 3.2 | 63.9 ± 7.7 | 77.7 ± 1.5 |

| Datasets | IMDB-B | IMDB-M | RDT-B | RDT-M5K | COLLAB | MUTAG | PROTEINS | PTC | NCI1 |
|---|---|---|---|---|---|---|---|---|---|
| # graphs | 1000 | 1500 | 2000 | 5000 | 5000 | 188 | 1113 | 334 | 4110 |
| # classes | 2 | 3 | 2 | 5 | 3 | 2 | 2 | 2 | 2 |
| Avg # nodes | 19.8 | 13.0 | 429.6 | 508.5 | 74.5 | 17.9 | 39.1 | 25.5 | 29.8 |
| GIN-$\epsilon$ (SUM-MLP) | **69.1 ± 7.5** | **44.2 ± 4.6** | **92.8 ± 1.6** | - | - | **88.0 ± 6.0** | 73.7 ± 5.0 | **65.5 ± 5.3** | **82.5 ± 2.1** |
| GIN-0 (SUM-MLP) | **69.6 ± 4.6** | **46.2 ± 5.0** | **92.7 ± 1.7** | - | - | **90.6 ± 5.4** | **74.8 ± 3.6** | 61.6 ± 5.4 | 82.1 ± 1.6 |
| SUM-1-LAYER | **69.1 ± 4.7** | **47.5 ± 5.0** | **93.0 ± 1.7** | - | - | 87.3 ± 8.2 | **75.7 ± 4.0** | **65.8 ± 6.7** | **83.0 ± 2.7** |
| MEAN-MLP | 52.4 ± 2.1 | 35.6 ± 2.6 | 52.1 ± 1.8 | - | - | 82.3 ± 6.8 | 73.1 ± 2.9 | 64.5 ± 7.3 | 81.1 ± 2.3 |
| MEAN-1-LAYER | 52.6 ± 2.7 | 33.9 ± 2.7 | 51.7 ± 3.0 | - | - | 85.8 ± 8.8 | 74.3 ± 4.5 | **65.8 ± 11.5** | 80.9 ± 1.2 |
| MAX-MLP | 53.4 ± 4.9 | 36.5 ± 3.1 | 53.2 ± 1.6 | - | - | 81.5 ± 8.4 | 70.6 ± 6.4 | 62.4 ± 7.9 | 79.1 ± 2.2 |
| MAX-1-LAYER | 53.4 ± 4.4 | 34.4 ± 3.3 | 51.6 ± 3.7 | - | - | 79.1 ± 7.3 | 72.6 ± 4.1 | 62.9 ± 8.2 | 78.1 ± 1.8 |

Table 1: Top: classification accuracies (%) and standard deviations for GNN variants as shown in Table 1 of Xu et al. (2019). Bottom: reproduced classification accuracies (%) and standard deviations for GNN variants.

perceptron instead of an MLP, SUM-1-LAYER achieved the best results for PROTEINS, PTC and NCI1 datasets. It is also worth noting that MEAN-1-LAYER was tied with SUM-1-LAYER in the results for the PTC dataset.

For some datasets, such as the two IMDBs, PROTEINS and NCI1, the accuracies yielded by GINs were lower that the ones reported in the reproduced paper, however, we cannot determine whether GIN really performs worse. In our reproduction attempt, we tested each GNN variant on each dataset only once, with only one configuration. With hyperparameter tuning we could obtain better results for each combination of GNN variant and dataset. Addtionally, we do not know what kind of classification layer the authors used in their implementation. We used a simple 2 layer MLP for classification and did not tune its hyperparameters. Therefore, the difference in the results could also be caused by applying different classifiers.

## 4 Conclusion

In our reproduction attempt of Xu et al. (2019), the proposed GNN aggregation scheme yielded worse results for certain datasets than what was reported in the original paper. However, without taking more time to test each combination of hyperparameters as the authors of the reproduced paper did, we cannot determine whether GINs perform better or worse than baselines and other GNN variants. Furthermore, not all parts of authors' implementation are clear, such as classification layers of their network, which further compounds the divergence of obtained results. One matter that raises questions is why our implementation obtained clearly different results for mean and max-pooling GNNs on IMDB datasets, while obtaining similar to authors' results on bioinformatics datasets. Finally, it would also be helpful to take more time and compute the results on larger datsets such as REDDIT-MULTI5K, COLLAB and the large dataset, REDDIT-MULTI12K, requested by one of anonymous commenters at the review page of the ICLR 2019 submission for the reproduced paper.

## REFERENCES

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.