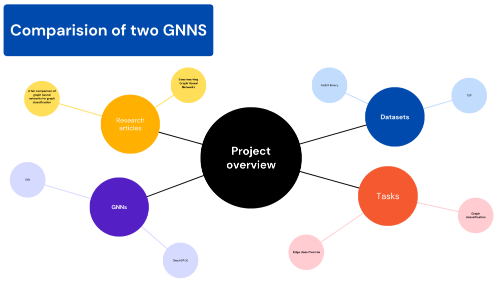
Final report: Comparison of two GNNs

Imane Elbacha
MVA 22-23 Graphical Models

Abstract

This project compares the performances of two popular graph neural networks, Graph Isomorphism Network (GIN) and GraphSAGE, on two distinct graph-related tasks: graph classification and edge classification. The paper draws from two recent contributions to the field of GNNs, Dwivedi et al.'s [1] benchmarking framework and Errica et al.'s [2] focus on experimental reproducibility and replicability. Specifically, the paper uses Dwivedi et al.'s benchmarking framework to compare GIN and GraphSAGE's performance on graph classification, and Errica et al.'s guidance to compare their performance on edge classification. The results provide insight into the relative strengths and weaknesses of the two GNNs, and highlight the importance of standardized benchmarking and reproducibility practices in the field of GNN research. Overall, this research contributes to the ongoing development of GNNs as a powerful tool for analyzing and learning from graph data.



1. Introduction:

Graph Neural Networks (GNNs) have become increasingly popular in recent years as a powerful tool for analyzing and learning from data on graphs. They have been applied successfully in a wide range of fields, including computer science, mathematics and biology. As GNNs continue to gain popularity, there is a growing need to develop benchmarks that can help quantify their progress and recognize the difference in their performances.

Two papers that have made significant contributions to the development of the GNN field which are "Benchmarking Graph Neural Networks" by Dwivedi et al. [1] and "A Fair Comparison of Graph Neural Networks for Graph Classification" by Errica et al. [2]. This project constitutes essentially a comparision of two GNNs but also a light confrontation of the frameworks proposed by both papers.

In their paper, Dwivedi et al. [1] proposed a benchmark framework that enables a fair comparison of GNNs with the same parameter budget. This framework includes a diverse collection of mathematical and real-world graphs, an open-source and reproducible code infrastructure, and is flexible enough for researchers to experiment with new theoretical ideas.

Errica et al.'s [2] paper focused on the critical topics of experimental reproducibility and replicability in the field of machine learning. They highlighted the lack of rigorousness and reproducibility in experimental procedures and provided an overview of common practices that should be avoided to fairly compare GNN models with the state of the art.

In this project, we compare two GNNs Graph isomorphism network (GIN) and GraphSAGE on two tasks graph classification and edge classification, fo that purpose we use Errica et al. and Dwivedi et al. on each task respectively.

2. Project elements :

2.1. Models presentation:

2.1.1 Graph isomorphism network GIN:

Graph Isomorphism Networks (GIN) are powerful Graph Neural Networks (GNN) that capture complex structural information from graphs using a simple message-passing scheme. Their high representational power allows them to almost perfectly fit training data, making them an important area of study for real-world applications. GINs are also computationally efficient and can handle large-scale graphs. In this project, we implemented a GIN network following the architecture proposed by Xu et al. (2018) [3] and compared its performance with GraphSAGE on two graph-related tasks. Our results showed that GIN networks outperformed GraphSAGE on some tasks, highlighting the effectiveness of GIN's permutation-invariant aggregation function.

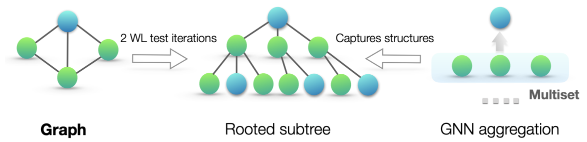
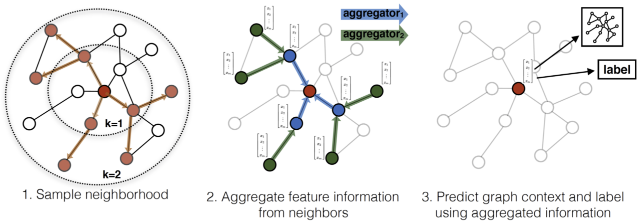


Figure 1: **An overview of our theoretical framework.** Middle panel: rooted subtree structures (at the blue node) that the WL test uses to distinguish different graphs. Right panel: if a GNN's aggregation function captures the *full multiset* of node neighbors, the GNN can capture the rooted subtrees in a recursive manner and be as powerful as the WL test.

2.1.2 GraphSAGE:

GraphSAGE is a framework that generates node embeddings for previously unseen data by leveraging node feature information, making it an efficient and powerful tool for a range of applications such as social network analysis, recommendation systems, and bioinformatics. Unlike previous methods, GraphSAGE learns a function that generalizes to unseen nodes by sampling and aggregating features from a node's local neighborhood, and it can handle feature-rich graphs while making use of structural features present in all graphs.

To generate embeddings, GraphSAGE samples a fixed number of neighbors for each node, aggregates feature information using a customizable neural network architecture, and generates an embedding for each node based on this information. GraphSAGE is highly efficient, as it only samples a fixed number of neighbors for each node and aggregates their features using stochastic gradient descent (SGD), allowing it to generate embeddings for large graphs in a reasonable amount of time. The efficiency and ability to handle feature-rich graphs were the motivation behind choosing GraphSAGE for the comparison project.



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

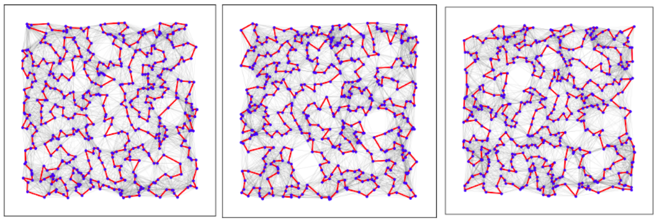
2.2. Dataset presentation:

2.2.1 Reddit binary for graph classification:

The dataset used for comparing GraphSAGE and GIN on the graph classification task is REDDIT-BINARY, which was introduced by Yanardag et al. It consists of graphs representing online discussions on Reddit, where nodes correspond to users and edges represent interactions between them. The dataset includes four subreddits, IAmA, AskReddit, TrollXChromosomes, and atheism, with the first two being question/answer-based and the latter two being discussion-based. Each graph in the dataset is labeled based on whether it belongs to a question/answer-based or discussion-based community. The REDDIT-BINARY dataset is commonly used to evaluate the performance of graph neural networks, making it an ideal choice for comparing GraphSAGE and GIN.

2.2.2 TSP dataset for Edge Classification:

The TSP dataset used for the comparison of GIN and GraphSAGE on the edge classification task was introduced by Pouya et al. as a benchmark set for the Traveling salesman problem (TSP) with small instances that are challenging for state-of-the-art TSP algorithms. The instances are based on difficult instances of Hamiltonian cycle problem (HCP), including those from literature, modified randomly generated instances, and instances converted from other difficult problems to HCP.



2.3. Evaluation framework:

2.3.1 A Fair Comparison of Graph Neural Networks for Graph Classification

The evaluation framework used in ”A Fair Comparison of Graph Neural Networks for Graph Classification” [2] (figure 2) involves a three-step process. First, the dataset is split into k equally sized folds and the model is trained and evaluated k times using external k-out-fold cross-validation. This step helps to estimate the model's generalization performance. Second, a hold-out set is created to select the best hyperparameters for the model using the hold-out technique. Finally, an inner k-inn-fold cross-validation is applied within each fold of the external cross-validation loop to select hyperparameters for the model. While this process can be computationally expensive, it ensures that the model is evaluated rigorously and reproducibly for graph classification tasks.

2.3.2 Benchmarking graph neural networks:

The evaluation framework presented in this paper [1] (figure 1) is designed to be modular, easy-to-use, and open-source. It includes a set of graph datasets, model parameters, and a standard codebase with data, training, and evaluation pipelines. This enables researchers to compare the performance of different GNN models on the same datasets under controlled conditions. The framework also provides a set of metrics for evaluating the performance of GNN models on various tasks such as node classification, link prediction, and graph classification.

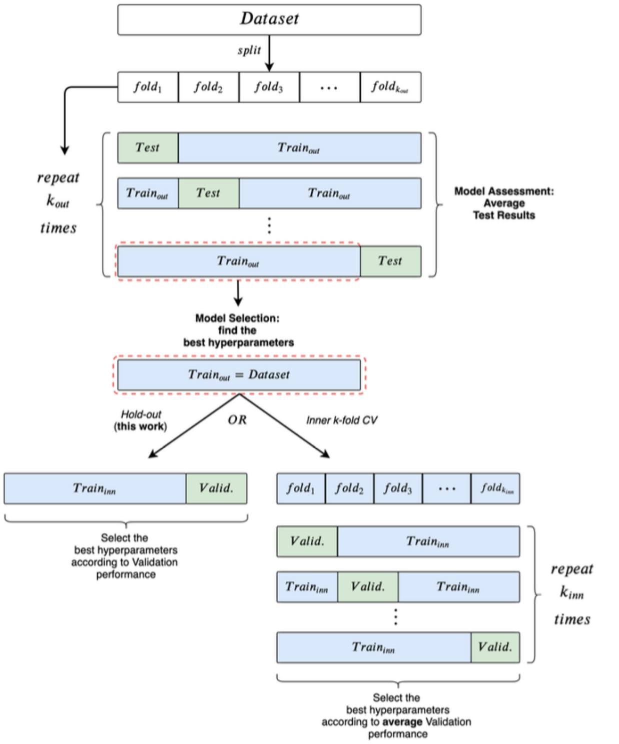

Dataset
split
$fold_1$ $fold_2$ $fold_3$ ... $fold_{k_{out}}$
repeat k_{out} times
Test
$Train_{out}$
$Train_{out}$ Test $Train_{out}$
$Train_{out}$ Test
Model Assessment: Average Test Results
Model Selection: find the best hyperparameters
$Train_{out}$ = Dataset
Hold-out (this work)
OR
Inner k-fold CV
$Train_{inn}$ Valid.
Select the best hyperparameters according to Validation performance
$fold_1$ $fold_2$ $fold_3$ ... $fold_{k_{inn}}$
Valid. $Train_{inn}$
$Train_{inn}$ Valid. $Train_{inn}$
$Train_{inn}$ Valid.
repeat k_{inn} times
Select the best hyperparameters according to average Validation performance

Figure 2: We give a visual representation of the evaluation framework. We apply an external k_{out}-fold CV to get an estimate of the generalization performance of a model, and we use an hold-out technique (bottom-left) to select the best hyper-parametres. For completeness, we show that it is also possible to apply an inner k_{inn}-fold CV (implementing a complete *Nested Cross Validation*), which obviously amounts to multiplying the computational costs of model selection by a factor k_{inn}.

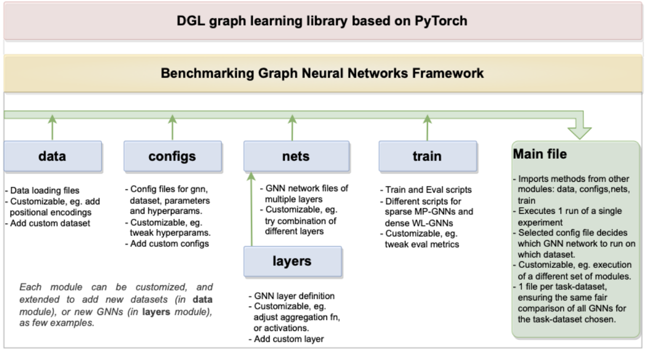

DGL graph learning library based on PyTorch
Benchmarking Graph Neural Networks Framework
data
configs
nets
train
Main file
- Data loading files
- Customizable, eg. add positional encodings
- Add custom dataset
- Config files for gnn, dataset, parameters and hyperparams.
- Customizable, eg. tweak hyperparams.
- Add custom configs
- GNN network files of multiple layers
- Customizable, eg. try combination of different layers
- Train and Eval scripts
- Different scripts for sparse MP-GNNs and dense WL-GNNs
- Customizable, eg. tweak eval metrics
- Imports methods from other modules: data, configs,nets, train
- Executes 1 run of a single experiment
- Selected config file decides which GNN network to run on which dataset.
- Customizable, eg. execution of a different set of modules.
- 1 file per task-dataset, ensuring the same fair comparison of all GNNs for the task-dataset chosen.
layers
- GNN layer definition
- Customizable, eg. adjust aggregation fn, or activations.
- Add custom layer
Each module can be customized, and extended to add new datasets (in data module), or new GNNs (in layers module), as few examples.

Figure 1: Overview sketch of the proposed GNN benchmarking framework with different modular components. This benchmark is built upon DGL and PyTorch libraries.

3. Experiments:

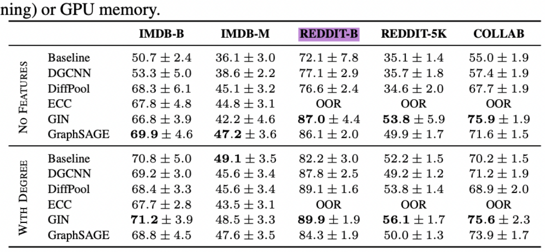
3.1. Task 1: Graph classification

In this project, we replicated the evaluation of Errica et al. [2] by implementing the GIN model and comparing it to the provided GraphSAGE and baseline models for graph classification. However, we encountered limited computational resources, which forced us to restrict the hyperparameters for the grid searches. To assess the impact of additional features, we performed training on the Reddit binary dataset both with and without features. The decision to test the models with and without features is important since featureless graphs are common in many real-world applications, and models should be able to handle them efficiently. Our results provide valuable insights into the performance of these models under different settings and highlight the importance of replicating and testing existing models to ensure that they are reliable and effective in various contexts. Furthermore, our findings show that GIN outperformed GraphSAGE on graphs with features, demonstrating the importance of selecting the appropriate model for a given dataset and task.

Table 1. Results on training with and without features

Dataset	Model	Train acc	Val acc
Reddit binary	Baseline	74.81	74.44
	GIN	81.39	78.4
	GraphSAGE	85.1	79.12
Reddit binary with features	Baseline2	75.0	73.8
	GIN	82.6	80.0
	GraphSAGE2	86.7	80.1

Table 4: Results on social datasets with mean accuracy and standard deviation are reported. Best performances are highlighted in bold. OOR means Out of Resources, either time (> 72 hours for a single training) or GPU memory.



		IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	COLLAB
NO FEATURES	Baseline	50.7 ± 2.4	36.1 ± 3.0	72.1 ± 7.8	35.1 ± 1.4	55.0 ± 1.9
	DGCNN	53.3 ± 5.0	38.6 ± 2.2	77.1 ± 2.9	35.7 ± 1.8	57.4 ± 1.9
	DiffPool	68.3 ± 6.1	45.1 ± 3.2	76.6 ± 2.4	34.6 ± 2.0	67.7 ± 1.9
	ECC	67.8 ± 4.8	44.8 ± 3.1	OOR	OOR	OOR
	GIN	66.8 ± 3.9	42.2 ± 4.6	$\mathbf{87.0} \pm 4.4$	$\mathbf{53.8} \pm 5.9$	$\mathbf{75.9} \pm 1.9$
	GraphSAGE	$\mathbf{69.9} \pm 4.6$	$\mathbf{47.2} \pm 3.6$	86.1 ± 2.0	49.9 ± 1.7	71.6 ± 1.5
WITH DEGREE	Baseline	70.8 ± 5.0	$\mathbf{49.1} \pm 3.5$	82.2 ± 3.0	52.2 ± 1.5	70.2 ± 1.5
	DGCNN	69.2 ± 3.0	45.6 ± 3.4	87.8 ± 2.5	49.2 ± 1.2	71.2 ± 1.9
	DiffPool	68.4 ± 3.3	45.6 ± 3.4	89.1 ± 1.6	53.8 ± 1.4	68.9 ± 2.0
	ECC	67.7 ± 2.8	43.5 ± 3.1	OOR	OOR	OOR
	GIN	$\mathbf{71.2} \pm 3.9$	48.5 ± 3.3	$\mathbf{89.9} \pm 1.9$	$\mathbf{56.1} \pm 1.7$	$\mathbf{75.6} \pm 2.3$
	GraphSAGE	68.8 ± 4.5	47.6 ± 3.5	84.3 ± 1.9	50.0 ± 1.3	73.9 ± 1.7

3.2. Task 2: Edge classification

For edge classification, I utilized the evaluation framework proposed by Dwivedi et al. [1] in their research paper. To ensure a fair comparison, I employed a comparable set of hyperparameters on three different models: GIN, GraphSage, and a baseline model which is a simple MLP. The TSP dataset was used for the evaluation, and the training time for each model was approximately two hours, which was proportional to the available resources.

The evaluation results indicated that GNNs, specifically GIN and GraphSage, outperformed the baseline MLP model. Additionally, GraphSage slightly outperformed GIN, but the results were very close to those presented in the research paper. Overall, the evaluation framework of Dwivedi et al. proved to be an effective and reliable method for evaluating the performance of different models in edge classification tasks.

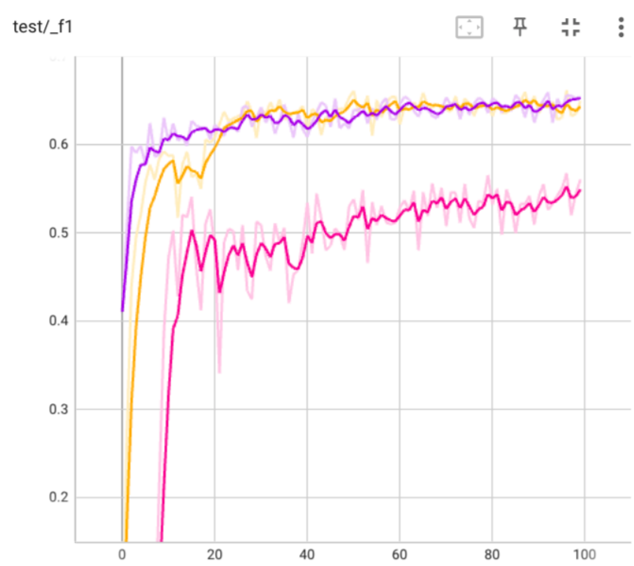
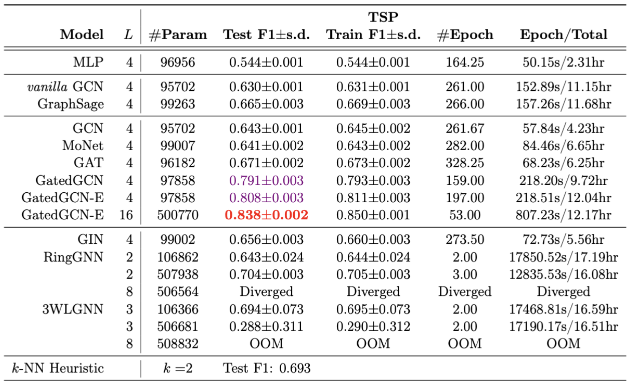


Figure 1. Results of test pink: baseline yellow:GIN and purple: GraphSAGE



Model	L	#Param	Test F1±s.d.	TSP Train F1±s.d.	#Epoch	Epoch/Total
MLP	4	96956	0.544±0.001	0.544±0.001	164.25	50.15s/2.31hr
vanilla GCN	4	95702	0.630±0.001	0.631±0.001	261.00	152.89s/11.15hr
GraphSage	4	99263	0.665±0.003	0.669±0.003	266.00	157.26s/11.68hr
GCN	4	95702	0.643±0.001	0.645±0.002	261.67	57.84s/4.23hr
MoNet	4	99007	0.641±0.002	0.643±0.002	282.00	84.46s/6.65hr
GAT	4	96182	0.671±0.002	0.673±0.002	328.25	68.23s/6.25hr
GatedGCN	4	97858	0.791±0.003	0.793±0.003	159.00	218.20s/9.72hr
GatedGCN-E	4	97858	0.808±0.003	0.811±0.003	197.00	218.51s/12.04hr
GatedGCN-E	16	500770	**0.838±0.002**	0.850±0.001	53.00	807.23s/12.17hr
GIN	4	99002	0.656±0.003	0.660±0.003	273.50	72.73s/5.56hr
RingGNN	2	106862	0.643±0.024	0.644±0.024	2.00	17850.52s/17.19hr
	2	507938	0.704±0.003	0.705±0.003	3.00	12835.53s/16.08hr
	8	506564	Diverged	Diverged	Diverged	Diverged
3WLGNN	3	106366	0.694±0.073	0.695±0.073	2.00	17468.81s/16.59hr
	3	506681	0.288±0.311	0.290±0.312	2.00	17190.17s/16.51hr
	8	508832	OOM	OOM	OOM	OOM
k-NN Heuristic		k =2	Test F1: 0.693			

Figure 2. Results of Dwivedi et al.

4. Conclusion:

In this project, we conducted an evaluation on two different GNNs for two tasks: edge classification and graph classification. For edge classification, we used the evaluation framework of Dwivedi et al. [1] and compared the performance of GraphSage and GIN to a baseline MLP model on the TSP dataset. The results showed that GNNs performed better than the MLP model, with GraphSage slightly outperforming GIN. These results were in line with the findings presented in the paper.

For graph classification, we used the evaluation framework of Errica et al. [2] and compared the performance of GraphSage and GIN to a baseline model on the Reddit binary dataset. The training was computationally expensive compared to the limited available resources.

Overall, this project was very interesting and helped further our understanding of GNNs and how to approach the task of choosing the right model and selecting the right hyperparameters. The use of different evaluation frameworks and comparison to baselines provided valuable insights into the strengths and weaknesses of different GNN models. Further research can build on this work by exploring other datasets and evaluating different GNN architectures.

References

[1] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020. 1, 2, 3, 4

[2] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 3, 4

[3] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 1

[4] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.