Benchmarking GCN and GraphSAGE

Krishanu Das Baksi
School of Information Technology
Indian Institute of Technology, Delhi
krishanudb@cse.iitd.ac.in

Anshul Yadav
Department of Electrical Engineering
Indian Institute of Technology, Delhi
ee1170565@iitd.ac.in

ABSTRACT

Over the last few years, there has been a steady increase in the application of neural networks to solve several problems with graphical data. The earliest works in this field pertain to graph convolutions and their application and mostly revolve around the ideas of Graph Convolutional Networks (GCN) [2] and its generalization GraphSAGE [1].
In this exercise, we try and evaluate the performance of GCN and GraphSAGE models in a variety of tasks viz., multi-label node classification, pairwise node classification and edge prediction. We further analyse the results and provide possible explanations for them. This study also aims to find out the favourable settings for both GCNs and GraphSAGE.

1. INTRODUCTION

In the real world, several important datasets come in the form of graphs or networks, for e.g., the world wide web, biological networks like protein protein interaction network (PPI), social networks like Facebook etc. to name a few. In recent years, graph neural networks have been demonstrated to outperform traditional methods on several prediction tasks on graphical data including but not limited to node classification and link/edge prediction.
GCNs and GraphSAGE uses the idea of graph convolutions to learn feature embeddings of nodes from their neighbourhood's features. These types of models have performed very well in several tasks.

1.1 GCN

Graph Convolution Neural Networks (GCNs) were introduced by Kipf and Welling [?] in 2016. It draws inspiration from spectral graph theory and exploit message passing to extract high-level features from a node as well as its neighbourhood using mean aggregation function. GCNs are designed for semi-supervised learning in a transductive setting, and requires the full graph Laplacian during training.

1.2 GraphSAGE

GraphSAGE was introduced by Hamilton et al. [1] in 2017. It generalizes the idea of message passing architectures and proposes concatenation of node features along with the mean / max / add pooled neighbourhood information in inductive framework. It samples the node's neighbourhood and aggregates features from it's local neighborhood.

2. DATASETS

2.1 PROTEIN

The protein dataset contains a total of around 1100 graphs, wherein each node denotes a protein and each edge denotes the interaction between proteins. In addition, each node has a 29 dimensional feature set, and may belong to one of 3 classes. For uniformity and ease of analysis, we discarded graphs which were very small (less than 20 nodes) or very big (greater than 1000 nodes). Thus all the analysis were done on the resulting 739 graphs.
A few properties of the graphs are listed below:
Mean Number of Nodes: 52
Mean Number of Edges: 197
Mean Network Diameter: 15.3
Mean Average Shortest Path: 5.99
There were 39 graphs which were not fully connected.

2.2 Brightkite

Brightkite is a large undirected network with 58228 nodes and 214,078 edges. It was collected from Brightkite which was a location-based social networking service provider where users shared their locations by checking-in. Nodes doesn't have feature vectors. It consists of 4 connected components among which one is very large. A few properties of the graphs are listed below:
Nodes in largest SCC: 56739 (0.974)
Edges in largest SCC: 212945 (0.995)
Average clustering coefficient: 0.1723
Diameter (longest shortest path): 16

We sampled 10k nodes from the dataset preserving 79402 edges due to limited compute power. All the analysis and experiments in this study refers to this sampled version of brightkite henceforth.

2.3 PPI

The PPI Dataset consists of 24 networks of protein interactions. Each node of the graphs denote a protein and edges denote interactions among proteins. Every node additionally has 50 dimensional feature vector, which is sparse, and 127 labels, wherein each node may belong to more than one classes.
A few properties of the graphs are listed below:
Mean Number of Nodes: 2245
Mean Number of Edges: 61318
Therefore, these graphs are in general much larger in size than the PROTEINS dataset graphs.

3. EXPERIMENTAL FRAMEWORK

The code for all the experiments performed can be found on Github at anshul3899/gcn-graphsage-benchmarking. We used Python for convenience which slightly limited us in terms of experiments due to its slow performance. The link prediction task was not able to scale up on the entire brightkite dataset and we had to use node sampling to make it work.

3.1 Pair-wise node classification

In pairwise node classification, the task is to predict whether a pair of nodes belong to the same class. The PROTEINS dataset and the pytorch/ pytorch-geometric implementations of GCN and GraphSAGE were used for the purpose. Ablation studies were conducted under different configurations of the model, wherein different values for epochs, learning rates, number of layers of the network etc were tried. The graphs were randomly divided into a training and testing set, in a 80:20 ratio. Cross Entropy Loss was used as the objective function. The ROC AUC scores of the model on the unseen test set are reported in the results section.

3.2 Link prediction

In link prediction, the task is to find out whether an edge exists between two nodes. Two nodes are generally more likely to form a link, if they are close together in the graph. The ppi and brightkite datasets and the pytorch-geometric implementations of GCN and GraphSAGE were used for this purpose. Binary cross entropy was used as loss function with Adam optimizer. We tried varying the aggregation scheme, the depth of layers and the activation functions to perform the ablation analysis. The graphs were split into a 80:20 train - test split. The train data was further split into 80:20 train-val split. Various metrics such as ROC-AUC, F1 score, Precision, Recall have been considered. We also plot the variations of metrics with training time for better insights into performance. The models were trained on 500 epochs for ppi dataset and 2000 epochs for brightkite dataset with learning rate 0.01.

3.3 Multi-class node classification

In multiclass node classification, the task was to predict all the class labels of every node. Ablation studies were conducted under different configurations of the model, wherein different values for epochs, learning rates, number of layers of the network etc were tried. The graphs were randomly divided into a training, validation and testing set, in a 20:2:2 ratio. Since each node could belong to none, one or more classes, this task is a multi-label classification task. Cross Entropy loss was minimized and the ROC AUC scores of the model on unseen test are reported in the results section.

4. OBSERVATIONS AND ANALYSIS

4.1 Pair-wise node classification

As a baseline model, we simply ran a sigmoid on the dot products of pairs of nodes' features. Using this simple non graph model, we got an ROC AUC value of 0.5, which is a fairly good baseline.

4.1.1 GCN

The summary of the different configurations and their results are shown in Table 1.

Table 1: Results on pair-wise node classification task for GCNs on Protein dataset

Layers	Learning Rate	Epochs	ROC AUC
2	0.001	500	0.63
2	0.001	800	0.63
3	0.01	500	0.53
3	0.001	500	0.53

It was observed that having 2 layers gave the optimum results. However, increasing the number of layers to 3 causes the testing ROC AUC to drop. This might be due to the small size of the networks in the proteins dataset, which have an average shortest path of around 6. Therefore, having large layer size causes the model to gather information from a much larger part of the network which is detrimental for the model. Also, it was seen that a learning rate of 0.001 with an Adam optimizer led to the best results. Furthermore, it was empirically observed that the using a hidden layer dimension of 256 gave the best results. Hence we used 256 dimensional hidden representations for all our models. Overall only one GCN model (with two layers and learning rate of 0.001) could significantly outperform the baseline.

4.1.2 GraphSAGE

The summary of the different configurations and their results are shown in Table 2.
GraphSAGE was used with two different configurations, using the mean pool and the max pool aggregators. As observed in the GCN, for GraphSAGE as well, using 2 layers gave the best results, strengthening our view that this is due to small size of the networks. It was seen that neither mean pool nor max pool consistently outperform the other. However the best results were obtained using max pool aggregator.

In general, the performance of GraphSAGE and GCN were more or less comparable, although GraphSAGE marginally outperformed GCN, which can be due to the presence of one more neural nonlinear layer in GraphSAGE. One more point to note is that it was seen that dropout or any other regularizations were not used in both the models types as the models did not significantly overfit the training data.

4.2 Link prediction

4.2.1 GCN

The summary of all the results of GCNs on ppi dataset is shown in Table 3 and on brightkite dataset is shown in Table 4. We used three different aggregation choices viz. mean, max and add pooling for GCNs but found out that the aggregation choice is not affecting the performance much. We also tried varying the activation functions but found no significant improvement. The performance somewhat hurts by increasing the number of layers. However Max pool with Sigmoid activation and 3 layers provided the maximum AUC-ROC score. On brightkite also, we observe maximum performance on mean pool with 3 layers with no comparable difference. The variation of metrics with training time is shown below.

Table 2: Results on pairwise node classification task for GraphSAGE on Protein dataset

Aggregator	**Layers**	**Learning Rate**	**Epochs**	**ROC AUC**
Mean Pool	2	0.01	500	0.64
Mean Pool	2	0.001	800	0.63
Mean Pool	2	0.01	500	0.59
Mean Pool	2	0.001	800	0.60
Mean Pool	3	0.01	500	0.53
Mean Pool	3	0.001	500	0.56
Max Pool	2	0.01	500	0.59
Max Pool	2	0.001	800	0.59
Max Pool	2	0.01	500	0.60
Max Pool	2	0.001	800	0.65
Max Pool	3	0.01	500	0.55
Max Pool	3	0.001	500	0.58

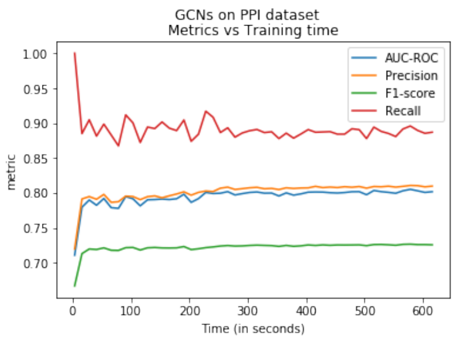

Figure 1: Variation of metrics for GCN on ppi

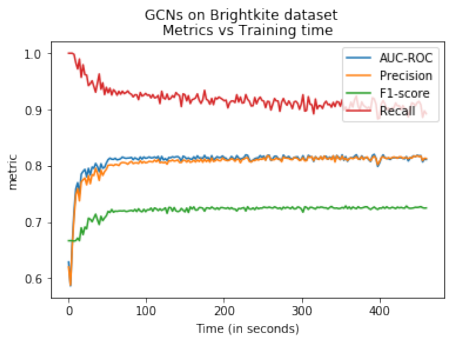

Figure 2: Variation of metrics for GCN on brightkite

4.2.2 GraphSAGE

The summary of all the results of GraphSAGE on ppi dataset is shown in Table 5 and on brightkite dataset is shown in Table 6. It is rather interesting to observe that GraphSAGE which outperformed GCNs on node classification tasks and supposed to generate better embeddings fails with significant margin against GCNs on link prediction task. Another interesting observation was that the AUC-ROC performance is greatly hurt with increase in the number of layers roughly by 0.2 for the ppi dataset. On the brightkite dataset, the performance of GraphSAGE is even worse.
These observations can be attributed to the fact that GraphSAGE maybe fails in capturing the closeness in graph while generating the embeddings. Clearly for link prediction, GCNs outperforms GraphSAGE algorithm by a huge margin. The variation of metrics with training time is shown below.

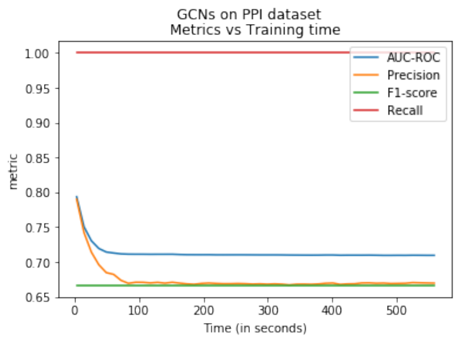

Figure 3: Variation of metrics for GraphSAGE on ppi

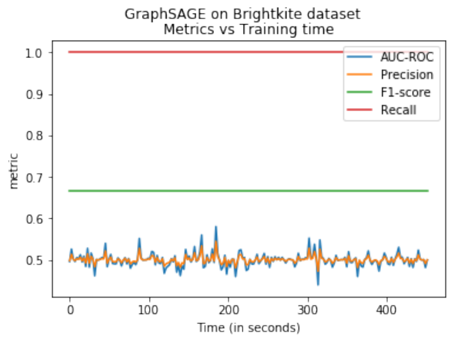

Figure 4: Variation of metrics for GraphSAGE on brightkite

4.3 Multi-class node classification

4.3.1 GCN

The summary for different results obtained using a GCN model are shown in Table 7.

Table 3: Results on link prediction task for GCNs on ppi dataset

Aggregation	**Layers**	**Activation**	**ROC-AUC**	**F1-score**	**Precision**	**Recall**
Mean pool	3	Sigmoid	0.792	0.712	0.795	0.901
Mean pool	3	ReLU	0.788	0.725	0.788	0.887
Mean pool	5	Sigmoid	0.788	0.723	0.790	0.892
Mean pool	5	ReLU	0.785	0.726	0.782	0.881
Max pool	3	Sigmoid	0.794	0.721	0.802	0.890
Max pool	3	ReLU	0.790	0.725	0.788	0.887
Max pool	5	Sigmoid	0.789	0.723	0.791	0.893
Max pool	5	ReLU	0.788	0.726	0.787	0.876
Add	3	Sigmoid	0.794	0.721	0.801	0.891
Add	3	ReLU	0.791	0.726	0.792	0.888
Add	5	Sigmoid	0.789	0.722	0.790	0.894
Add	5	ReLU	0.787	0.727	0.785	0.882

Table 4: Results on link prediction task for GCNs on brightkite dataset

Aggregation	**Layers**	**Activation**	**ROC-AUC**	**F1-score**	**Precision**	**Recall**
Mean pool	3	Sigmoid	0.803	0.721	0.799	0.903
Mean pool	5	Sigmoid	0.804	0.701	0.790	0.965
Max pool	3	Sigmoid	0.801	0.714	0.801	0.936
Max pool	5	Sigmoid	0.804	0.708	0.787	0.957
Add	3	Sigmoid	0.801	0.720	0.804	0.919
Add	5	Sigmoid	0.796	0.701	0.782	0.968

Table 5: Results on link prediction task for GraphSAGE on ppi dataset

Aggregation	**Layers**	**Activation**	**ROC-AUC**	**F1-score**	**Precision**	**Recall**
Mean pool	3	Sigmoid	0.712	0.667	0.671	1.0
Mean pool	3	ReLU	0.712	0.667	0.677	1.0
Mean pool	5	Sigmoid	0.501	0.667	0.505	1.0
Mean pool	5	ReLU	0.497	0.667	0.501	1.0
Max pool	3	Sigmoid	0.711	0.667	0.671	1.0
Max pool	3	ReLU	0.712	0.667	0.677	1.0
Max pool	5	Sigmoid	0.501	0.667	0.502	1.0
Max pool	5	ReLU	0.513	0.667	0.513	1.0
Add	3	Sigmoid	0.713	0.667	0.675	1.0
Add	3	ReLU	0.712	0.667	0.676	1.0
Add	5	Sigmoid	0.507	0.667	0.505	1.0
Add	5	ReLU	0.520	0.667	0.512	1.0

Table 6: Results on link prediction task for GraphSAGE on brightkite dataset

Aggregation	**Layers**	**Activation**	**ROC-AUC**	**F1-score**	**Precision**	**Recall**
Mean pool	3	Sigmoid	0.501	0.667	0.501	1.0
Mean pool	5	Sigmoid	0.497	0.667	0.4988	1.0
Max pool	3	Sigmoid	0.501	0.667	0.501	1.0
Max pool	5	Sigmoid	0.501	0.667	0.501	1.0
Add	3	Sigmoid	0.499	0.667	0.500	1.0
Add	5	Sigmoid	0.500	0.667	0.500	1.0

Table 7: Results on mulit-class node classification task for GCN on ppi dataset

Layers	Learning Rate	Epochs	ROC AUC
2	0.01	300	0.87
2	0.001	300	0.876
3	0.01	300	0.88
3	0.001	300	0.89

Using GCN for multiclass node classification, it was seen that having 3 layers made the model perform slightly better than having 2 layers. This is most likely because the PPI networks are larger as compared to PROTEIN networks, therefore aggregating more neighbour information improves the model. It was observed that after around 300 epochs training did not improve the validation performance of the model. The interesting observation is that using different learning rates did not significantly change the model performance.

4.3.2 GraphSAGE

The summary of the results obtained using GraphSAGE are shown in Table 8.

Table 8: Results on mulit-class node classification task for GraphSAGE on ppi dataset

Aggregator	Layer	LR	Epochs	ROC-AUC
Mean Pool	2	0.001	300	0.858
Mean Pool	3	0.001	300	0.869
Max Pool	2	0.001	300	0.876
Max Pool	3	0.001	300	0.914

In this task, the performance of the GraphSAGE model with Max pool aggragator was found to be significantly better than that using Mean Pool. This may be because the features of the nodes were mostly discrete features, which would lose information if mean pool is used. On the other hand max pool retains the important features' structures. Training the model beyond 300 epochs did not improve test set performance. One more important observation was that the 3 layer model performed better than 2 layer model as was observed in GCNs. This further consolidates our view that for larger networks, with higher average diameter, higher number of layers improves model performance.

Overall it was observed that GraphSAGE with max pooling aggregator outperformed all other combinations we tried.

5. CONCLUSIONS

This study benchmarked GCN and GraphSAGE on pairwise node classification, link prediction and multi-class node classification using appropriate evaluation metrics. Several design choices and parameter variation were also studied. The experiments provided with deeper understanding of GCNs and GraphSAGE and clarified which of the two should be used for a task while considering their pros and cons.

6. ACKNOWLEDGEMENTS

We thank Prof. Sayan Ranu for providing us with this benchmarking opportunity. We are highly thankful to the P-GNN's [3] official implementation from which our majority of code is borrowed for the link prediction task. We sincerely acknowledge the Python, PyTorch, networkx documentations and Stackoverflow's support. We would like to thank Google for providing us with Colab for GPU compute power.

7. REFERENCES

[1] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[2] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[3] J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. *CoRR*, abs/1906.04817, 2019.