CIS 700 – Machine Learning with Graphs

Sahil Gupta

Project Report

Edge Sampling in GNNs

Introduction

Graph neural networks (GNNs) have become standard tools for learning tasks on graphs. By iteratively aggregating information from neighborhoods, GNNs embed each node from its k-hop neighborhood and provides a significant improvement over traditional methods in node classification and link prediction tasks. However, most GNN models need a complete underlying network structure, which is often unavailable in real-world settings. Frequently it is the case that only a portion of the underlying network structure is observed, which can be considered as the result of graph sampling [1].

We consider the observed incomplete network structure as one random sampling instance from a complete graph, then we address the fundamental problem of GNN performance under graph sampling. [1]

1. Can we use GNNs if only a portion of the network structure is observed?
2. Which graph sampling methods and GNN models should we choose?

Experimental Setup

Models

We train 3 different GNNs for node classification tasks. These include GCN, GAT and TAGCN[2].

GCN has 2 Graph Convolution Layers. It has a input feature size equal to the number of features in the dataset, a hidden layer size of 16, and an output layer of size equal to the number of classes in the dataset. It also has Relu activation and dropout enabled in layer 1.

GAT has 2 Graph Attention Layers [3]. It has a input feature size equal to the number of features in the dataset, a hidden layer size of 16, and an output layer of size equal to the number of classes in the dataset. It has elu activation enabled in layer 1 and dropout enabled in both layer 1 and layer 2. We use the value of multi-headed attention heads = 8.

TAGCN has 2 Graph Convolution Layers. It has a input feature size equal to the number of features in the dataset, a hidden layer size of 16, and an output layer of size equal to the number of classes in the dataset. It also has Relu activation and dropout enabled in layer 1.

The exact implementation of these GNNs with all the parameters can be found with the accompanying code.

Datasets

The citation network datasets "Cora", "CiteSeer" and "PubMed" from “Revisiting Semi-Supervised Learning with Graph Embeddings”[4] paper. Nodes represent documents and edges represent citation links. Training, validation and test splits are given by binary masks.[5]

Name	#nodes	#edges	#features	#classes
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3

Edge Sampling

Graph sampling is a technique to pick a subset of nodes and/or edges from an original graph. The commonly studied sampling methods are node sampling, edge sampling, and traversal-based sampling.

Sampling methods in our experiment– Random Edge (implemented using the function dropout_edge in PyG), Random Walk Edge (implemented using the function dropout_path in PyG) [6][7].

Results

We conducted 5 experiments and averaged the accuracy values across these experiments.

Figure 1 represents the absolute accuracies of various edge sampling methods on different GNNs and on different datasets. Figure 2 compares the percentage drop in test accuracy wrt the maximum test accuracy for different edge sampling ratios. Figure 3 compares performance of different GNNs for the same dataset with the same edge sampling method.

The results can be summarized as follows:

1. Test accuracy on node classification tasks increases as the sampling ratio increases. This is intuitive as the more edges we have for training, the better the neighborhood estimate we have and better the accuracy.
2. Even at low values of sampling ratio, GNNs work with not a major drop in accuracy. From Fig 2 we can see that we can reduce the size of edges by 60% with only a 10% drop in accuracy for Cora dataset, a ~7.5% drop in accuracy for the CiteSeer dataset and about 5% drop in accuracy for PubMed dataset.
3. In our experiments, we see that the 3 models have a very comparable performance across datasets. They deliver accuracy within a 5% range of each other for all datasets. In particular GAT seems to work the best for Cora dataset, GCN works slightly better for CiteSeer dataset and TAGCN works the best for PubMed dataset.

4. RE and RW give comparable performance in almost all cases. In Cora and CiteSeer dataset, RW seems to give a very slight increase in performance whereas for PubMed dataset, RE gives a slightly better performance.

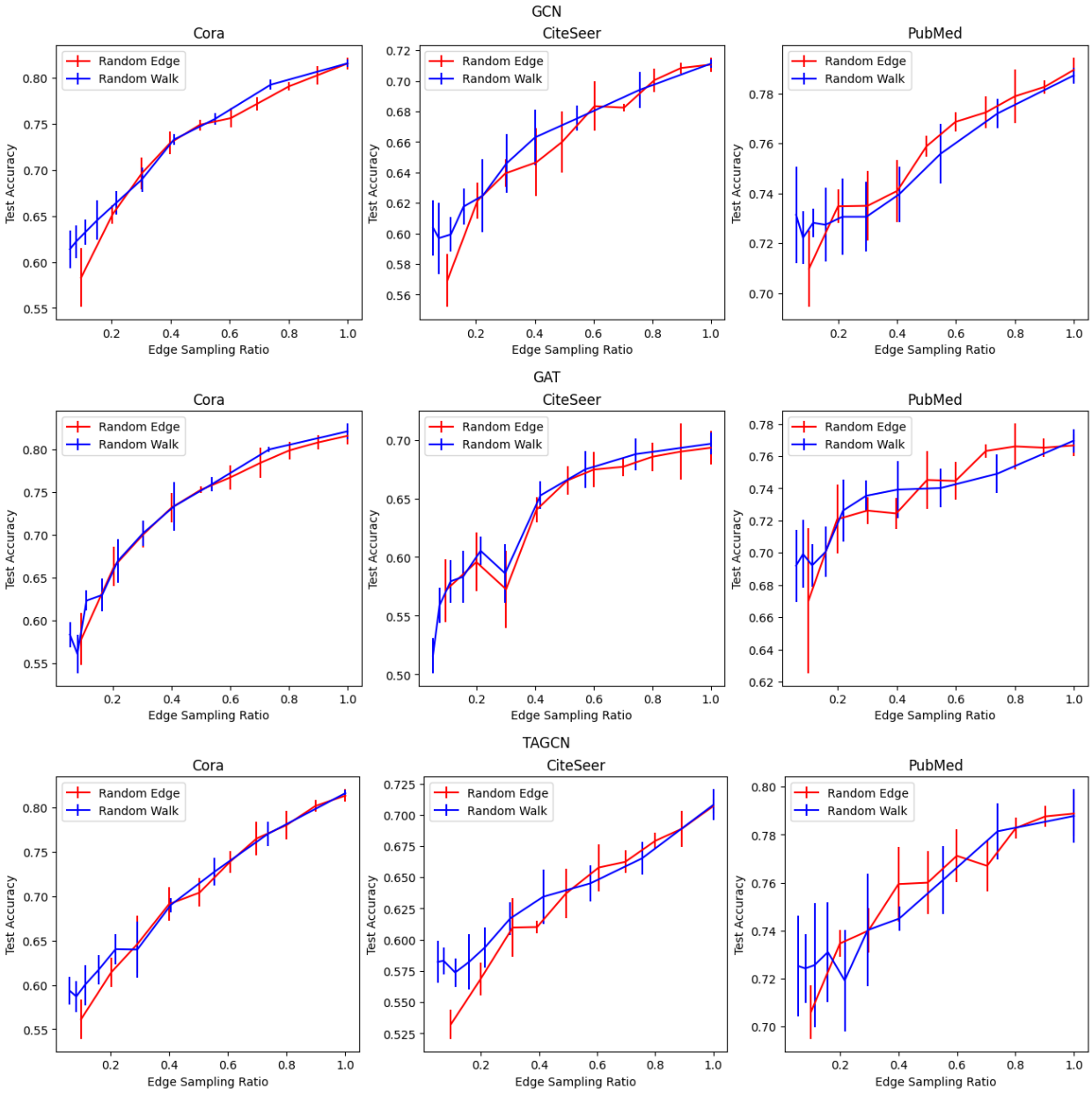

Figure 1: Comparing absolute accuracies of various edge sampling methods on different GNNs and on different datasets.

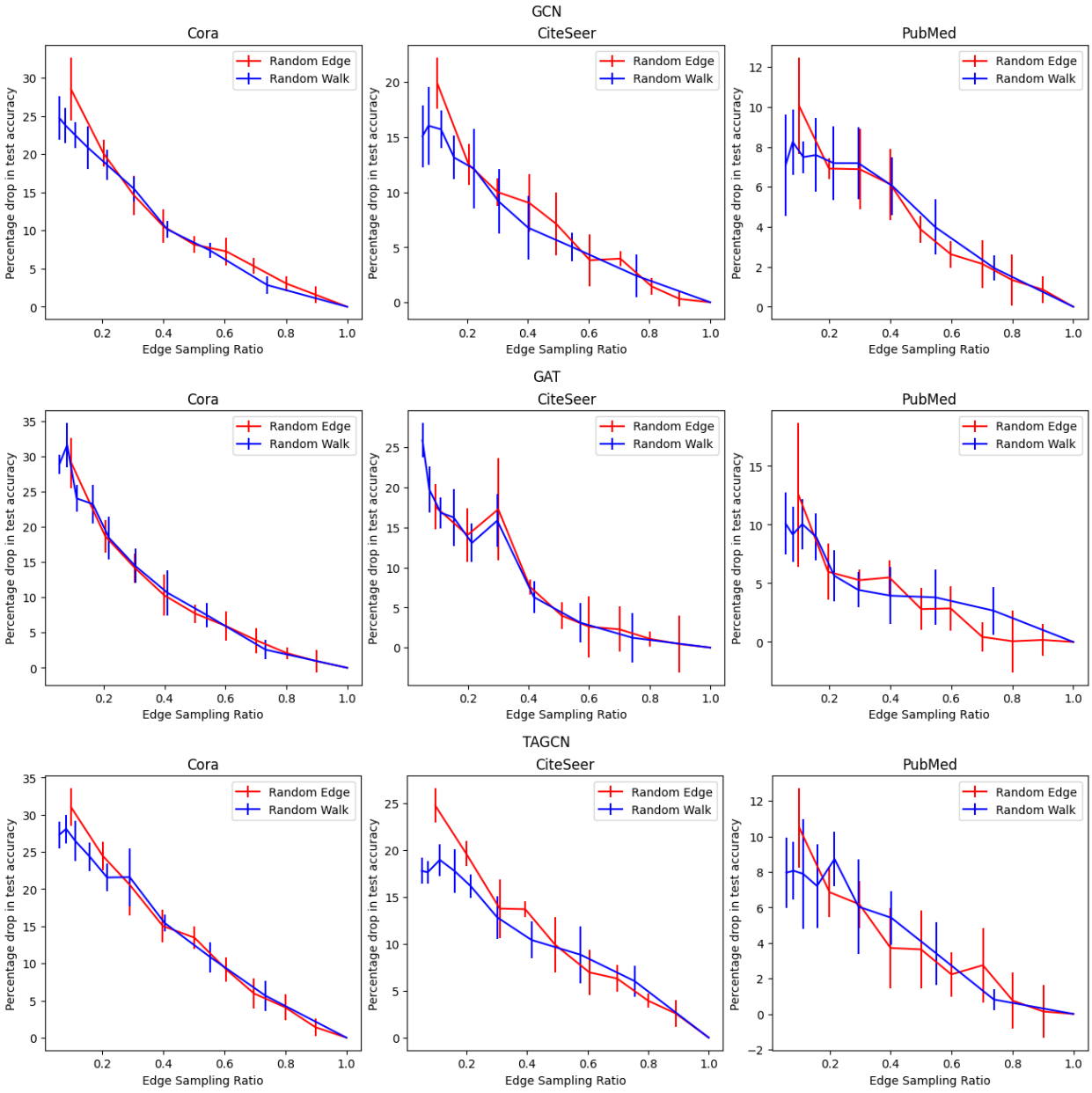

Figure 2: Comparing the percentage drop in test accuracy wrt the maximum test accuracy for different edge sampling ratios

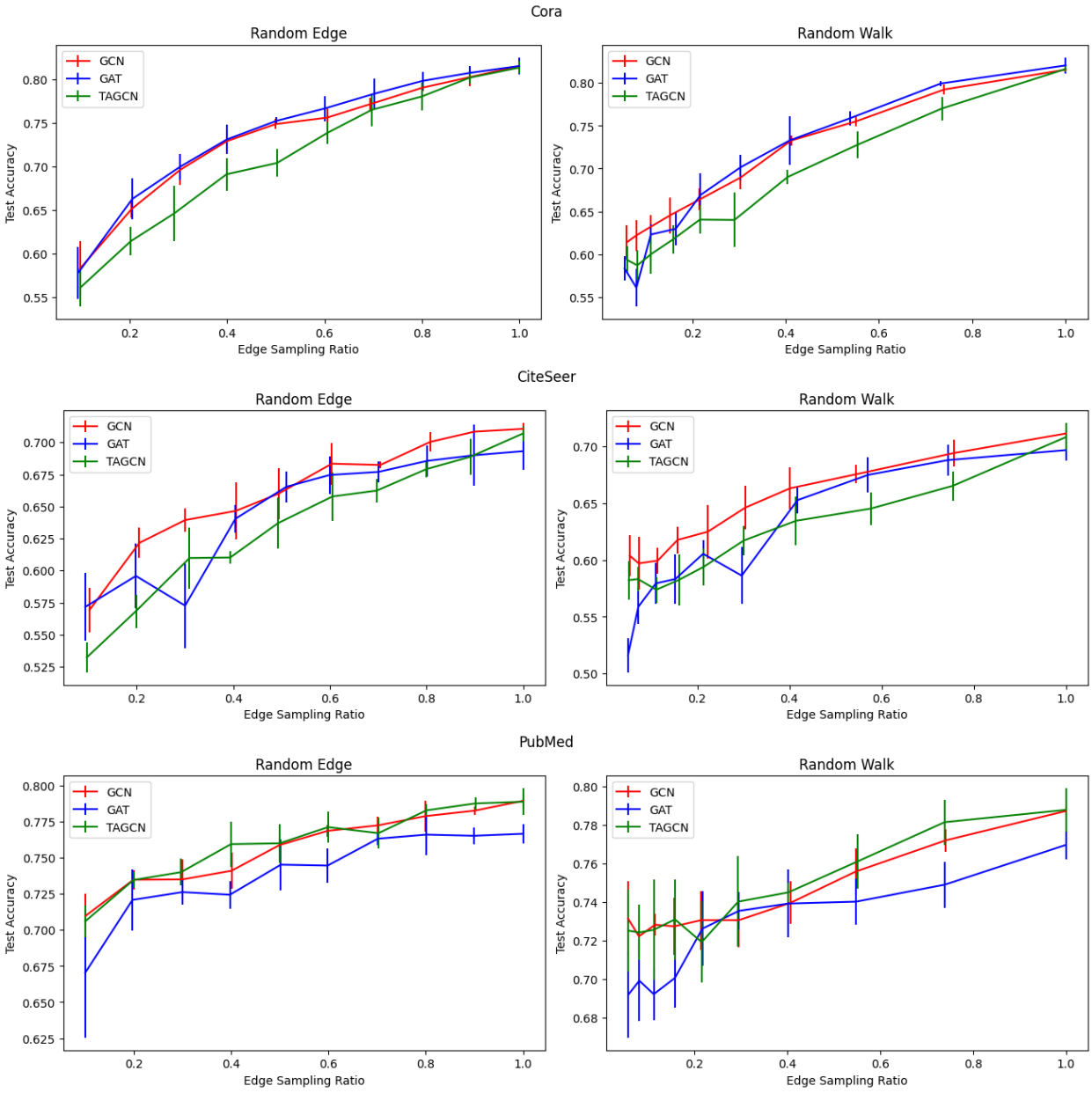

Figure 3:Comparing performance of different GNNs for the same dataset with the same edge sampling method.

References:

1. Wei Q, Hu G. Evaluating graph neural networks under graph sampling scenarios. PeerJ Comput Sci. 2022 Mar 4;8:e901. doi: 10.7717/peerj-cs.901. PMID: 35494843; PMCID: PMC9044246.
2. Jian Du, Shanghang Zhang, Guanhang Wu, José M. F. Moura, & Soummya Kar. (2018). Topology Adaptive Graph Convolutional Networks.
3. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, & Yoshua Bengio. (2018). Graph Attention Networks.

4. https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.Planetoid.html#torch_geometric.datasets.Planetoid
5. Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16). JMLR.org, 40–48.
6. Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06). Association for Computing Machinery, New York, NY, USA, 631–636. https://doi.org/10.1145/1150402.1150479
7. https://pytorch-geometric.readthedocs.io/en/latest/modules/utils.html#torch_geometric.utils.dropout_path