Graph Neural Network Compression for Edge Devices

Mustafa Munir
University of Texas at Austin
mmunir@utexas.edu

Geffen Cooper
University of Texas at Austin
geffen@utexas.edu

Abstract

Graph neural networks (GNNs) have shown high performance, but also high memory consumption. To decrease the memory consumption of GNNs we propose an iterative process for GNN compression using graph partitioning, subgraph training, and model compression to enable deployment on edge devices. Our results indicate that for citation networks (Cora, CiteSeer, ogbn-arxiv) and GNNs with few layers (two), there is a small decrease in the node classification accuracy (in the transductive setting) when training on partitioned graphs compared to the full graph.

1. Introduction and Motivation

Artificial intelligence (AI) and machine learning (ML) have had explosive growth in the past decade. Their rapid growth has lead to widespread research in convolutional neural networks, transformers for natural language processing and computer vision, and now recently graph neural networks (GNNs). GNNs have become ubiquitous for learning on non-Euclidean data due to their success on node classification, link prediction, and graph prediction tasks for complex networks. GNNs have shown success in learning in various fields including computational biology, drug discovery, computer vision, fraud detection, and social network modeling. In these domains the graphs can grow to huge sizes, containing many millions of nodes and edges [13]. This coupled with the high memory consumption and computation costs of GNNs due to the large size of graph data and the recursive nature of graph convolution make graph neural networks difficult to deal with outside of high performance computing applications [10, 8]. This makes deployment of GNNs in resource-constrained environments for edge artificial intelligence (edge AI) applications difficult [11]. Our objective is to use graph partitioning to generate subgraphs then use knowledge distillation to generate small subgraph student GNNs that perform node classification. This will address whether coupling graph partitioning and model compression can improve GNN compression for enabling deployment on edge devices.

We specifically use graph partitioning to reduce the subgraph diameters and therefore the required number of GNN layers and the number of parameters of each GNN. As a result, this can limit the receptive field of each node in the GNN to circumvent the following challenges [12]:

- degraded expressivity due to too much information in the receptive field of each node.
- extensive computation due to neighborhood explosion.

2. Prior Work

There have been several works on GNN compression. [10] proposed subgraph-level training via resource-aware graph partitioning. They first partition the graph into a set of disjoint subgraphs and perform local training at the subgraph-level. [11] proposed an approach for KD from a pre-trained graph convolutional network (GCN). [6] proposed the first teacher-free knowledge distillation method for GNNs called GNN Self-Distillation (GNN-SD), which serves as a drop-in replacement for the standard training process. [8] proposed applying Product Quantization (PQ) on GNNs to achieve memory capacity reduction. [14] proposed compression using pruning of the dimensions in each layer, called channel pruning, to accelerate GNN inference. [5] generalized the lottery ticket hypothesis to GNNs by defining a graph lottery ticket as a pair of core sub-dataset and sparse sub-network that can be identified from the original GNN and the full dense graph by iteratively applying unified graph sparsification (UGS). UGS simultaneously prunes the graph adjacency matrix and the model weights thereby effectively making the GNN model compressed.

Other works focused on machine learning aware partitioning rather than GNN compression specifically. [7] reduces the granularity of spatial datasets by combining fine-grained adjacent cells into coarser cells, then trains the machine learning model. This re-partitioning keeps informa-

tion loss within a defined threshold and leads to minimal accuracy degradation of the model.

Our approach builds upon prior work through implementing graph partitioning to generate subgraphs, then applying model compression on the generated subgraphs through KD. Our goal is that once our compressed subgraph GNNs are generated we can then perform distributed inference thereby achieving better efficiency while maintaining competitive accuracy in comparison to prior work.

3. Approach

The task we use as an evaluation metric is node classification for the **Cora** [3] (2,708 nodes and 5,429 links), **CiteSeer** [1] (3,312 nodes and 4,732 links), and **ogbn-arxiv** [2] (169,343 nodes and 1,166,243 links) datasets. These datasets each form a citation network where the nodes are research papers labeled with their individual field and the links are the citations between papers. The main question explored is how to partition a graph and compress its resulting GNNs for distributed inference while retaining high node classification accuracy.

First, we explore how graph partitioning affects node classification accuracy. In general, graph partitioning algorithms search for similarly sized subgraphs which have minimal inter-partition links to balance partition workloads while minimizing communication. However, it is unclear if these optimization objectives for generating partitions maintain node classification accuracy. Our initial assumption was that each disjoint partition may not contain enough information for each subgraph GNN to generate high quality node embeddings and may result in overfitting. Specifically, nodes on the boundaries of partitions are particularly vulnerable to information loss as they get cut-off from a subset of their first order neighbors during partitioning.

To ensure each GNN does not overfit to its subgraph, we propose investigating various aggregation schemes such as weighted aggregation over each subgraph's GNN as shown in figure 1. We believe that training and aggregating multiple smaller GNNs over subgraphs can result in better efficiency while maintaining similarly performing node embeddings in comparison to full graph training. Specifically, smaller subgraph diameters circumvent neighborhood explosion which can limit the over-squashing effect where the compression of information from the exponentially growing receptive field of each node into a fixed-size vector causes information loss [4]. Thus, each subgraph GNN can learn local information while retaining long range information through aggregation of each subgraph GNN.

Our approach is designed to be scalable through the potential of parallelizing the training of the subgraph GNNs. Without aggregation, the subgraphs and their GNNs are disjoint allowing for distributed training and inference at the cost of accuracy degradation caused by missing information from the other partitions. With aggregation, the subgraph GNNs can exchange parameters (or node embeddings if a subgraph gets broadcasted to multiple GNNs) at the cost of increased communication and computation. Therefore, our method is flexible in order to balance the tradeoff of accuracy with communication and computation.

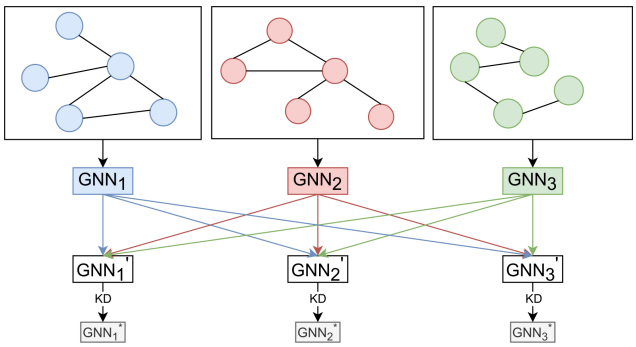

Figure 1. GNN Weighted Aggregation

4. Experimental Setup and Results

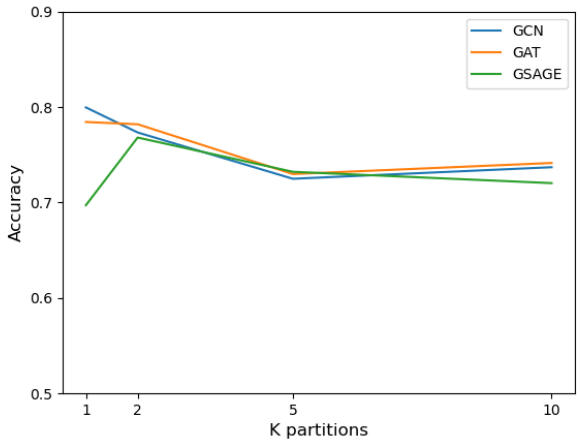

Figure 2. Baseline Accuracy over number of partitions for the Cora dataset

We first set up baseline experiments to empirically analyze the effect of graph partitioning on node classification accuracy. We train three GNN architectures, GCN, Graph Attention Network (GAT), and GraphSAGE, on the Cora and CiteSeer datasets over four partitioning sizes $k = 1, 2, 5, 10$. Each GNN has two layers with a ReLU activation. For graph partitioning we use METIS [9], a widespread tool for efficiently generating high quality partitions. In each experiment, we collect the predictions of each

subgraph to calculate the validation accuracy over the entire graph. The results are shown in figures 2 and 3. These experiments do not employ aggregation over subgraph GNNs, KD of the subgraph GNNs, or any hyperparameter tuning and serve as baseline results for comparison.

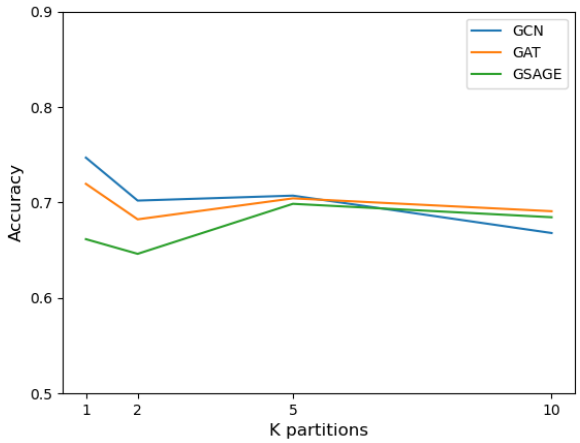

Figure 3. Baseline Accuracy over number of partitions for the CiteSeer dataset

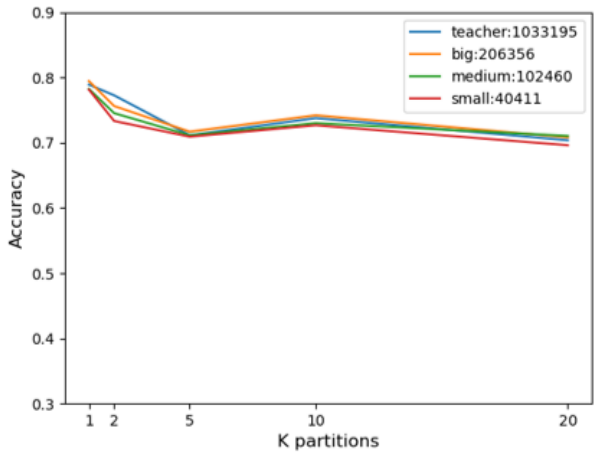

Figure 4. Accuracy over number of partitions for the distilled subgraph GCNs on the Cora dataset

Using the same method as our baseline results, we analyzed the effect of coupling graph partitioning with KD using the GCN architecture on the Cora, CiteSeer, and ogbn-arxiv datasets over five partitioning sizes of $k = 1, 2, 5, 10, 20$. The teacher GNN architectures are described as follows:

- GCN: two graph convolutional layers, ReLU activation function, and a dropout layer.
- GAT: two graph attention layers, ReLU activation function, and a dropout layer.
- GraphSAGE: two SAGEConv layers (From GraphSAGE that use mean aggregation of neighbor nodes), ReLU activation function, and a dropout layer.

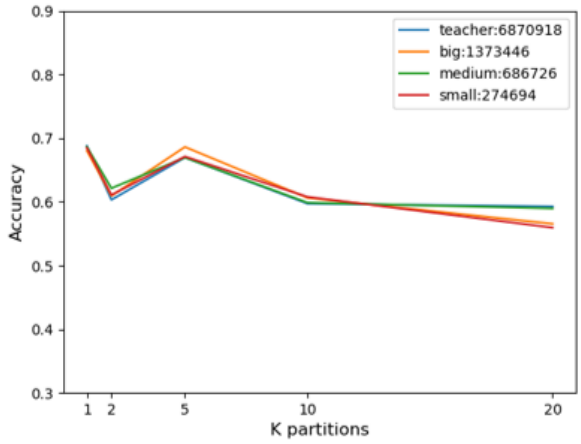

Figure 5. Accuracy over number of partitions for the distilled subgraph GCNs on the CiteSeer dataset

Due to the similarity of the results, we only run and report the experiments using the GCN architecture in the remainder of the report. We implement a *big*, *medium*, and *small* student GNN architecture by simply reducing the dimension of the input to the second GNN layer to 50%, 20%, and 10% of the original. The results are shown in figures 4, 5, and 6. Over all the student models, the results for partitioning on Cora and CiteSeer are consistent with our initial teacher GNN baseline as there is an initial small drop in accuracy that is then recovered slightly or flat depending on the number of partitions. For ogbn-arxiv our results were initially quite unexpected as the accuracy actually increased when we used more partitions for the distilled models (plot not shown). We hypothesized that when the partitions are too large, the small student GNNs do not have enough parameters to capture sufficient information for producing high quality node embeddings since ogbn-arxiv has more nodes, links, and output classes than Cora and CiteSeer. However, when the number of partitions increase, the student GNNs are better able to capture the information in the smaller subgraphs resulting in an overall increase in accuracy. We were able to verify this hypothesis by extending the ogbn-arxiv experiment to larger student models and found that accuracy degradation was relatively minimal as shown in figure 6.

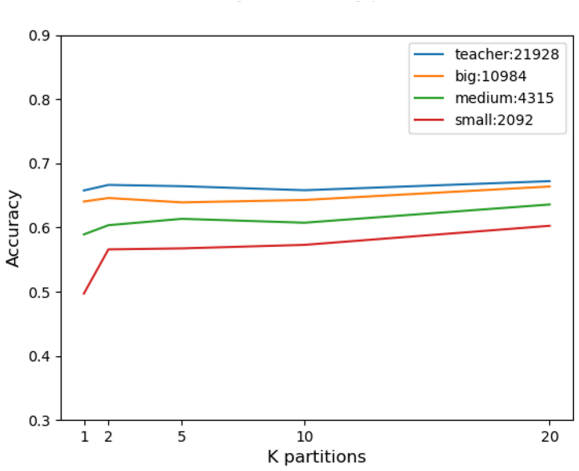

Figure 6. Accuracy over number of partitions for the distilled subgraph GCNs on the Arxiv dataset

Contrary to our initial assumption, it seems that graph partitioning has a small effect on node classification accuracy. In line with our predictions in M1 it was possible to further compress each GNN into a student GNN using knowledge distillation. In general, this result is significant because it suggests that local information within a subgraph partition is sufficient for node classification on the Cora, CiteSeer, and ogbn-arxiv datasets.

To understand why accuracy remains relatively stable, even at 20 partitions, we visualized the different subgraphs in Gephi. In general, we found that the classification accuracy for each partition seems to correlate with how *homogeneous* the partition is. Specifically, a partition where the majority of the nodes are from few classes, where nodes from each class are tightly clustered, gets higher accuracy than a partition where the nodes come from several classes and all the nodes cluster evenly. This results in a high variance of accuracies where some partitions get greater than 90% accuracy and others get less than 50% accuracy. This suggests that the accuracy for each subgraph is highly dependent on how METIS selects each partition. Interestingly, these 'good' and 'bad' partitions seem to average out so that the overall accuracy remains stable with the increase in the number of partitions. We hypothesize that aggregation will help account for highly heterogeneous partitions that have suffered from increased information loss.

Our preliminary experiments on weighted subgraph aggregation were more challenging than initially expected. We first attempted to do a weighted average of the subgraph GNN parameters every few epochs to enable the spread of long range information. However this aggregation of model parameters increased the training time significantly since models were constantly being loaded and saved into memory. Furthermore, the accuracy of the aggregated models almost always ended up being lower than the original subgraph GNN models. Based on our analysis of the individual subgraphs (prior paragraph), we hypothesize that the individual subgraphs have a structure that is either too similar (when the number of partitions is small) or too different (when the number of partitions is large). As a result, our current aggregation scheme seems to have little to no accuracy improvement at the cost of increased training time.

Given that the accuracy degradation was minimal after graph partitioning, we analyzed the memory usage during training and inference to see the benefits of our approach. Figure 7 shows the memory usage when training the teacher model on ogbn-arxiv and subsequently doing knowledge distillation. As shown in the figure, this requires over a gigabyte of memory which may not be suitable for edge devices. Figure 8 shows the memory usage for graph partitioning then subgraph training and model distillation on one out of ten partitions of ogbn-arxiv. There is an initial memory spike due to the copying of the graph data for METIS. After this spike, the memory usage stabilizes to around 0.5 gigabytes which is more suitable for a Raspberry Pi which has 1 gigabyte of RAM. Also note that the speed of training is much faster in comparison to full graph training since all partitions are trained in parallel across 10 CPU cores.

Figure 9 shows the detailed memory usage overtime during full graph inference with the teacher model on ogbn-arxiv. From this figure it can be seen that the required libraries (green), graph data (purple), and forward pass (yellow) dominate the memory usage. Figure 10 shows the memory usage over time for subgraph inference with the student model on one of the ten partitions of ogbn-arxiv. In comparison to figure 9, we see that the memory usage for loading the graph data (purple) and forward pass (yellow) have been significantly reduced so that inference only requires around 350 megabytes which is dominated by the imported libraries (green). Note that the time axis does not represent inference latency. Delays were added in order to visualize the sections of memory usage more clearly.

Overall, we see that the reduction in memory usage during training and inference is significant when we partition the graph. While not shown in the current figures, knowledge distillation further reduces the inference time memory usage significantly as the dimensionality of the intermediate outputs are reduced significantly. However, we see that graph partitioning still requires significant memory and time. Based off these results, we think it is feasible to distribute the training and inference on edge devices assuming that a powerful host computer is capable of storing and partitioning the original graph.

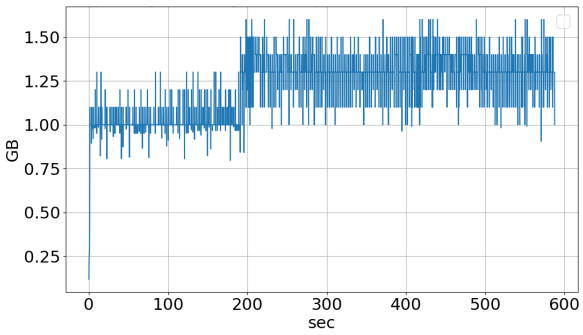

Figure 7. Memory usage for full graph training with the teacher model on ogbn-arxiv

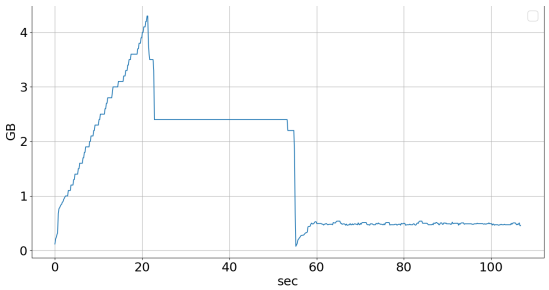

Figure 8. Memory usage for subgraph training with small student model on 1 out of 10 partitions of ogbn-arxiv

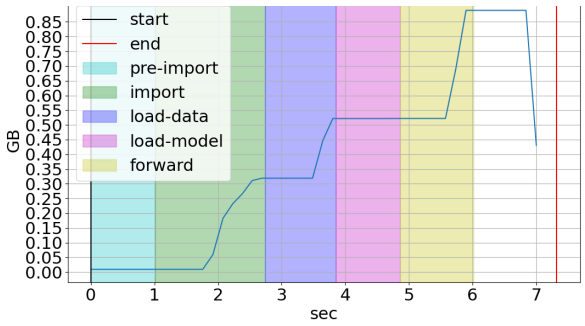

Figure 9. Memory usage for full graph inference with teacher model on ogbn-arxiv

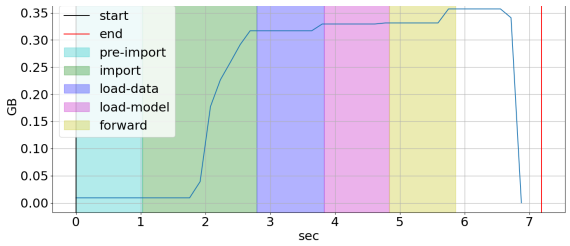

Figure 10. Memory usage for subgraph inference with small student model on 1 out of 10 partitions of ogbn-arxiv

5. Conclusion and Future Work

Our future plans are to continue developing an aggregation scheme to see if accuracy can be gained by sharing information across each subgraph GNN without major communication costs. We can also plan to further investigate why accuracy seems to increase when we increase the number of partitions after an initial drop in accuracy. For example, in the results for CiteSeer the accuracy decreases from 1 to 2 partitions, then increases from 2 to 5 partitions. This is the opposite of our hypothesized behavior and we believe it may be due to how METIS performs the partitioning leading to partitions that are highly homogeneous when we create more partitions up until the number of classes in the dataset. Future work includes extending our research to larger datasets as our parallel training does not work on ogbn products and only sequential training does. Future work also includes incorporating some sort of subgraph aggregation scheme to recover any lost information from the partitioning.

Our empirical results show that graph partitioning and knowledge distillation may have a limited effect on node classification accuracy. Furthermore, we find that node classification accuracy does initially decrease with partitioning, but with more partitions the accuracy recovers making it decrease only slightly when compared to baseline results for full graph training. This demonstrates the potential of our approach in compressing each subgraph GNN. Our proposed method will be the first GNN model compression method for subgraphs, allowing for decreased memory utilization while maintaining accuracy.

6. Contributions and Lessons Learned

During the course of the project Geffen Cooper programmed the training, evaluation, memory profiling, and graph visualization scripts while Mustafa Munir programmed the graph partitioning, GNN architectures, parallelization, and KD for GNN model compression scripts. Geffen Cooper and Mustafa Munir both investigated why accuracy increases with the increasing number of partitions to find theoretical justification and compared the total GNN memory and computation reduction achieved through the methods employed.

The lessons learned throughout the course of the project can be summarized as node classification of citation networks requires information only a few layers out from the

node being classified, thus graph partitioning does not cause a large decrease in accuracy when implemented. The subgraph GNNs generated lose some information from the edges lost when partitioning, but most information is maintained. Furthermore, knowledge distillation can also be implemented on these subgraph GNNs resulting in only a minimal decrease in accuracy, but a large decrease in memory allowing the distilled subgraph GNNs to possibly fit on edge devices. In our testing we did not test on edge devices, but we did test on a 16-core CPU and assigned a partition to each core for the subgraph GNN training and knowledge distillation. Overall, this work suggests that certain classes of graphs (i.e. citation networks) may not require or benefit from long range information so that expensive full graph training can be replaced by distributed subgraph training.

References

[1] https://linqs-data.soe.ucsc.edu/public/lbc/.

[2] https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv.

[3] https://relational.fit.cvut.cz/dataset/cora.

[4] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications, 2020.

[5] T. Chen, Y. Sui, X. Chen, A. Zhang, and Z. Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning*, pages 1695–1706. PMLR, 2021.

[6] Y. Chen, Y. Bian, X. Xiao, Y. Rong, T. Xu, and J. Huang. On self-distilling graph neural network. *arXiv preprint arXiv:2011.02255*, 2020.

[7] K. Chowdhury, V. V. Meduri, and M. Sarwat. A machine learning-aware data re-partitioning framework for spatial datasets. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2426–2439, 2022.

[8] L. Huang, Z. Zhang, Z. Du, S. Li, H. Zheng, Y. Xie, and N. Tan. Epquant: A graph neural network compression approach based on product quantization. *Neurocomputing*, 503:49–61, 2022.

[9] G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20, No. 1:359—392, 1999.

[10] Z. Xue, Y. Yang, M. Yang, and R. Marculescu. SUGAR: efficient subgraph-level training via resource-aware graph partitioning. *CoRR*, abs/2202.00075, 2022.

[11] Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang. Distilling knowledge from graph convolutional networks. *CoRR*, abs/2003.10477, 2020.

[12] H. Zeng, M. Zhang, Y. Xia, A. Srivastava, A. Malevich, R. Kannan, V. Prasanna, L. Jin, and R. Chen. Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, 34:19665–19679, 2021.

[13] D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang, and G. Karypis. Distdgl: distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*, pages 36–44. IEEE, 2020.

[14] H. Zhou, A. Srivastava, H. Zeng, R. Kannan, and V. Prasanna. Accelerating large scale real-time gnn inference using channel pruning. *arXiv preprint arXiv:2105.04528*, 2021.