Comparative Study of Graph Neural Networks Acceleration Techniques

Aditya Gupta
Indian Institute of Science
Bengaluru, India
adityapg@iisc.ac.in

Armaan Khetarpaul
Indian Institute of Science
Bengaluru, India
armaank@iisc.ac.in

Shankaradithyaa Venkateswaran
Indian Institute of Science
Bengaluru, India
shankaradith@iisc.ac.in

Umang Majumder
Indian Institute of Science
Bengaluru, India
umangm@iisc.ac.in

***Abstract*—Training of Machine Learning models on Graph datasets tend to suffer from scalability and over-smoothing issues as they grow in size. We will evaluate the performance of 4 different GNN Acceleration techniques on key performance metrics. Our study will provide insights into how different architectural choices and sampling techniques impact model effectiveness, guiding the selection of optimal GNN frameworks for large-scale graph learning tasks.**

***Index Terms*—Graph Neural Networks, GCN, DropEdge, GraphSAINT, GraphSAGE, NeuralSparse, Scalability, Graph Sampling, Sparse Learning, Node Classification, Large-Scale Graphs**

I. Introduction

Graphs are one of the most versatile data structures, with the capability to model any relationship, even if it is not always the most efficient way to do it in practice. For example, while pixels in an image can be represented as a network of connected nodes, representing the image as an array is more convenient for CNNs. For complex data however, representation in the form of graphs becomes important. There are multiple real life use cases of representing data in the form of graphs:

- Representing relationships in social models
- Representing the structure of molecules and proteins
- Recreating relationships between objects in an image

A Graph Neural Network (GNN) is a class of Machine Learning models that predict properties on a dataset of graphs. The property to be predicted may be a graph-level, node-level or an edge-level classification or regression task.

Two popular ways to represent graphs are adjacency matrices and adjacency lists. Since most of the graphs we will be dealing with are sparse, adjacency lists are the better storage method.

II. Past Work

The most basic algorithm to perform Graph Level Predictions is the GCN. [1]

GCNs or Graph Convolutional Networks generalize convolution to graphs by aggregating and transforming information from a node's local neighborhood.

In real life, graph networks can be extremely large and scale very fast in complexity as the number of nodes and edges increase, so optimisation of their training becomes important

There are two major ways to perform GNN Acceleration.

A. Sampling

GCNs are executed in a full batch manner. Model weights are updated once per epoch and slows down training convergence. Graph Sampling methods select nodes from a target node's neighbourhood to acquire subgraphs for subsequent training

- **Node Wise Sampling:** Focuses on each node and its neighbours in a training graph. Considers all neighbours a fixed number of hops away. Eg. GraphSAGE [2], VR-GCN [3]
- **Layer wise Sampling:** Samples a fixed number of nodes in each layer based on a precomputed probability. Eg. FastGCN [4], AS-GCN [5]
- **Subgraph based sampling methods:** Generates subgraphs for training in a two-step manner: sampling nodes and constucting edges. Eg. GraphSAINT [6]
- **Heterogenous sampling methods:** Designed to accelerate training and handle graph heterogeneity. Eg. HetGNN [7]

B. Sparsification

Graph Sparsification typically remove edges from the graph either randomly or through a specific optimisation goal. Sparifying graphs before they are fed into a GNN makes computation efficient and reduces memory access for model training.

- **Heuristic Sparsification:** DropEdge [8] randomly removes edges in each training epoch, which aims to solve the oversmoothing issue in deep GNN training
- **Learnable Sparsification:** NeuralSparse [9] uses a DNN to learn a sparsification strategy. SGCN [10] casts it as an optimisation problem and resolves it via an alternating direction method of approach.

III. Plan of Action

We have identified two datasets that we will be running the experiments on.

- The PROTEINS dataset [12] (Protein function prediction via graph kernels), a graph level binary prediction task.
- The ogbm-proteins dataset [13], a node level multilabel classification task

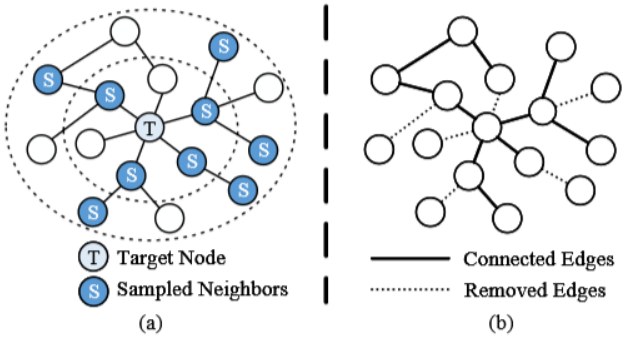

Fig. 1. Illustrations of graph-level improvements: (a) a graph sampling method that samples 2-hop neighbors; (b) a graph sparsification method that removes useless edges [11]

There are also other datasets as part of the Open Graph Benchmark [13] that we may visit in the future if time permits.

We will be implementing 4 different kinds of acceleration techniques and analysing how they compare to the basic GCN in terms of memory, operations, accuracy, EDP Score and other relevant metrics.

For now, our decision is to implement a node-level sampling technique, a subgraph-level sampling technique, a heuristic sparsification technique and a learnable sparsification technique.

We are going to implement GraphSage, GraphSAINT, DropEdge and NeuralSparse. The work will be evenly divided with each of the four members handling one task.

We will use Python to implement these frameworks, on the PyTorch and the PyTorch Geometric libraries

IV. Final Goal

In a GCN, each layer aggregates the information present in the nodes (weighted sum of neighbour features), as we keep going further, the information reaches the far-away nodes. However, if the layers go deep enough, the aggregation may result in all the nodes converging on a common value. This is known as the oversmoothing problem.

DropEdge randomly removes edges in each epoch. This is equivalent to image augmentation in CNNs that has been shown to reduce overfitting. It is also functionally similar to the Dropout technique utilised in DNNs that slow down overfitting. It can be done in conjunction with other techniques like GraphSAGE.

First, we want to see how oversmoothing and overfitting can affect the training of our models. Then, we want to measure the improvement in efficiency due to the 4 methods as well as how well they alleviate the oversmoothing and overfitting issues. Some are node-based while some are edge-based. For our benchmark dataset, we have a graph-level prediction task and a node-level prediction task. We want to compare these 4 methods based on their impact on memory, floating point operations and accuracy and identify potential trade-offs.

We also want to investigate when and why specific GNN acceleration techniques should be preferred over others based on dataset characteristics.

References

[1] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
[2] William L. Hamilton, Rex Ying, Jure Leskovec, "Inductive Representation Learning on Large Graphs", https://doi.org/10.48550/arXiv.1706.02216
[3] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. 2019. "A vectorized relational graph convolutional network for multi-relational network alignment." In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI'19). AAAI Press, 4135–4141.
[4] Chen, Jie, Tengfei Ma, and Cao Xiao. "Fastgcn: fast learning with graph convolutional networks via importance sampling." arXiv preprint arXiv:1801.10247 (2018).
[5] Li, Maosen, et al. "Actional-structural graph convolutional networks for skeleton-based action recognition." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.
[6] Zeng, Hanqing, et al. "Graphsaint: Graph sampling based inductive learning method." arXiv preprint arXiv:1907.04931 (2019).
[7] Shao, Zezhi, et al. "Heterogeneous graph neural network with multi-view representation learning." IEEE Transactions on Knowledge and Data Engineering 35.11 (2022): 11476-11488.
[8] Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." arXiv preprint arXiv:1907.10903 (2019).
[9] Zheng, Cheng, et al. "Robust graph representation learning via neural sparsification." International Conference on Machine Learning. PMLR, 2020.
[10] Shi, Liushuai, et al. "SGCN: Sparse graph convolution network for pedestrian trajectory prediction." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.
[11] Liu, Xin, et al. "Survey on graph neural network acceleration: An algorithmic perspective." arXiv preprint arXiv:2202.04822 (2022).
[12] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, Hans-Peter Kriegel, "Protein function prediction via graph kernels", Bioinformatics, Volume 21, Issue suppl1, June 2005, Pages i47–i56, https://doi.org/10.1093/bioinformatics/bti1007
[13] Hu, Weihua, et al. "Open graph benchmark: Datasets for machine learning on graphs." Advances in neural information processing systems 33 (2020): 22118-22133.