Data Independent GCN Pruning via Coresets

Anmol Anand
Department of Computer Science
Texas A&M University
aanand@tamu.edu

Abstract

This paper extends Coresets methodology for Data Independent Neural Pruning to Graph Convolutional Networks (GCNs). Leveraging Coresets, this approach compresses GCNs while maintaining a trade-off between compression rate and training latency for graph-based data inputs. Our method employs layer-wise pruning aimed at reducing model size while preserving predictive accuracy and reducing latency.

1 Introduction

Prior research endeavors in data independent neural pruning via Coresets were predominantly centered on neural nets with fully connected layers, overlooking the complexities inherent in graph datasets that are accounted for by using Graph Neural Networks. This oversight has resulted in a limited adaptation of Coresets, thereby necessitating a novel direction in research to specifically accommodate and cater to the unique attributes and complexities posed by these specialized graph based neural networks.

1.1 Previous Work: Data Independent Neural Pruning via Coresets

Coresets are a technique used in discrete structures, computational geometry, and machine learning to efficiently summarize large datasets while maintaining their essential characteristics. They are representative subsets of the original dataset that capture the data's main structure or properties. The primary goal of creating a coreset is to significantly reduce the size of the dataset while ensuring that algorithms or computations performed on the coreset yield similar results as if they were executed on the entire dataset.

In the context of machine learning, particularly neural network compression, coresets are utilized to reduce the size of neural networks while preserving their predictive capabilities [2]. The concept involves identifying a smaller subset of critical elements (such as neurons in a neural network) that can represent the entire network's functionality adequately. By constructing these coresets intelligently, it becomes possible to significantly reduce the network's size without sacrificing its overall performance. The idea is to maintain a smaller, condensed representation of the network that can approximate the behavior of the original, larger network effectively. This reduction in size is crucial for deploying neural networks in resource-constrained environments.

1.2 Previous Work: Graph Convolutional Networks

Graph Convolutional Networks (GCNs) have emerged as a powerful approach for semi-supervised learning on graph-structured data. The focus lies in classifying nodes/edges within graphs, or entire graphs. Traditional approaches for such tasks relied on explicit graph-based regularization techniques, leveraging methods like graph Laplacian regularization to propagate label information through connected nodes in the graph. However, these methods often faced limitations in modeling

capacity, assuming node similarity based on graph edges, which may not consistently encode such relationships.

Addressing these limitations, a pioneering approach by Kipf and Weiling [1] introduced Graph Convolutional Networks (GCNs) as an effective and scalable solution for semi-supervised learning tasks on graph data. In a GCN architecture, each graph convolutional layer operates akin to a fully connected layer. Initially, the features of individual nodes are independently processed through this layer. Subsequently, an aggregator operator is applied to the resulting features of each node. This aggregator operator performs a convolutional operation, combining the features of neighboring nodes to generate the updated features for each node in the graph. Notably, the aggregator operator can be as simple as the summation of the identity matrix with the adjacency matrix of the graph, represented as $A + I$. This process allows for the incorporation of graph structure information, enhancing the network's ability to learn and generalize across interconnected nodes within the graph.

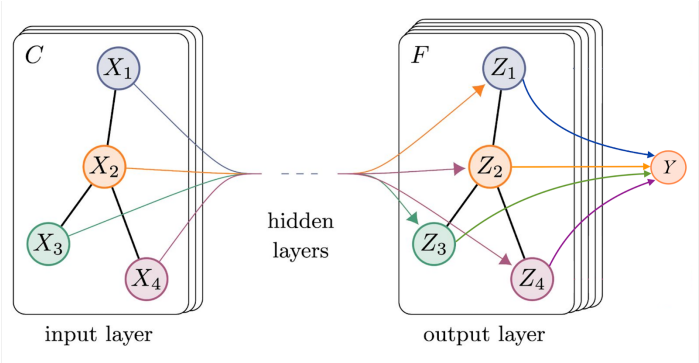

Figure 1. Aggregating the features of neighboring nodes to get the updated features of each node after individually propagating each node's features through a Graph Convolution Layer

While the propagation rule for a fully connected layer is shown in equation 1, the propagation rule for a graph convolutional layer is shown in equation 2:

$$X^{(l+1)} = \sigma\left(X^{(l)}W^{(l)}\right) \tag{1}$$

$$X^{(l+1)} = \sigma\left((A+I)X^{(l)}W^{(l)}\right) \tag{2}$$

where

$X^{(l)}$	is the output of layer l
$X^{(0)}$	are the input features of nodes in the training batch
σ	is the activation function
A	is the adjacency matrix of the sample graph
$W^{(l)}$	is the trainable weight matrix of layer l

2 Architecture

This architecture comprises two Graph Convolutional Layers (GCN) followed by two Fully Connected Layers. The GCN layers each consist of 512 neurons, while the subsequent fully connected layer contains 1024 neurons. The final fully connected layer contains 2 or 3 neurons based on the number of classes in the dataset used.

Layer	Original number of neurons	Number of neurons after compression	Compression factor
Graph Convolution Layer 1	512	512	
Graph Convolution Layer 2	512	256	2
Fully Connected Layer 1	1024	512	2
Fully Connected Layer 2	Num of classes	Num of classes	

Table 1: Pruned GCN architecture

3 Results

3.1 IMDb Graph Dataset Overview

The IMDb datasets utilized for graph-based machine learning tasks comprise ego-network graphs, where each node represents an individual associated with the film industry. Nodes correspond to actors and encompass diverse features, including age, the total number of movies they've participated in, accolades or awards received, the count of interviews they've appeared in, and a popularity score indicative of their influence or recognition within the industry. Edges in these graphs connect actors who have shared screen space by appearing together in the same movies.

3.1.1 IMDb Binary Dataset

The IMDb-Binary dataset is designed for binary classification tasks on IMDb ego-network graphs. The objective here is to categorize nodes (actors) into one of two genres: Action or Romance.

3.1.2 IMDb Multi Dataset

In contrast, the IMDb-Multi dataset extends the classification task to multiple genres within IMDb ego-network graphs. This dataset challenges machine learning models to categorize nodes into one of three genres: Action, Romance, or Sci-Fi.

3.2 Results

Multiple iterations of training were conducted for each dataset, comparing the results obtained from the original unpruned network, the uniformly pruned network, and the network pruned using the coreset technique.

3.2.1 Results for IMDb-BINARY dataset

The figures in Appendix A illustrate the Gaussian distribution of latency for the IMDb-Binary dataset across three distinct scenarios: the unpruned network, the uniformly pruned network, and the coreset-based pruned network. These results are summarized in Table 2. Coreset Pruning required 35% less time compared to the Uncompressed network, while Uniform Pruning reduced the time by 28% in contrast to the Uncompressed network.

Compression type	**Expected latency (seconds) to reach saturation accuracy of 75%**
None	58.67
Uniform Pruning	42.53
Coreset Pruning	38.21

Table 2: Results for IMDb-Binary

3.2.2 Results for IMDb-MULTI dataset

The following in Appendix B illustrate the Gaussian distribution of latency for the IMDb-Multi dataset across three distinct scenarios: the unpruned network, the uniformly pruned network, and the coreset-based pruned network. These results are summarized in Table 3. Coreset Pruning required 13% less time compared to the Uncompressed network. Meanwhile, Uniform Pruning reduced the time by 15% in comparison to the Uncompressed network.

Compression type	**Expected latency (seconds) to reach saturation accuracy of 75%**
None	16.31
Uniform Pruning	13.92
Coreset Pruning	14.19

Table 3: Results for IMDb-Multi

4 Observations

In the context of our experimentation with Graph Convolutional Networks (GCNs) and pruning techniques on IMDb datasets, several notable observations emerged:

- **Coreset Pruning Enhanced Performance**: Coreset-based pruning exhibited superior performance compared to the unpruned network. It notably showcased faster convergence to saturation accuracy, indicating more rapid learning and efficiency gains.
- **Performance comparison between Coreset and Uniform Pruning**: While Coreset pruning outperformed Uniform pruning for the IMDb-Binary dataset, it did not outperform Uniform pruning for the IMDb-Multi dataset. Possible reasons why Coreset based pruning did not consistently outperform Uniform pruning:
  - **Algorithm Suitability for GCN Layers**: The use of the same coreset computation algorithm for Graph Convolutional (GCN) layers and Fully Connected (FC) layers might not be ideal. GCN-specific intricacies might necessitate tailored or specialized coreset approaches for optimal pruning.
  - **Identical Network Size for Pruned Networks**: Both Uniform and Coreset pruned networks retained the same network size, differing only in their initial network weights. This indicates that the initial coreset weights might not inherently lead to faster training or improved performance.

References

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[2] Ben Mussay, Samson Zhou, Vladimir Braverman, and Dan Feldman. On activation function coresets for network pruning. *CoRR*, abs/1907.04018, 2019.

Appendix A

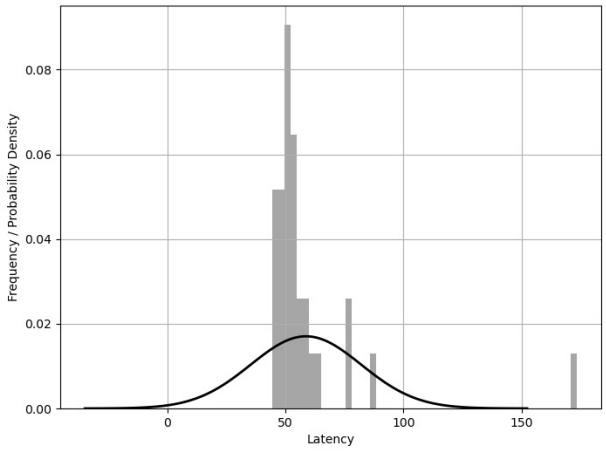

Figure A.1 Average latency to reach saturation accuracy of 75%: 58.67 seconds
Dateset: IMDB-BINARY || Compression Type: None

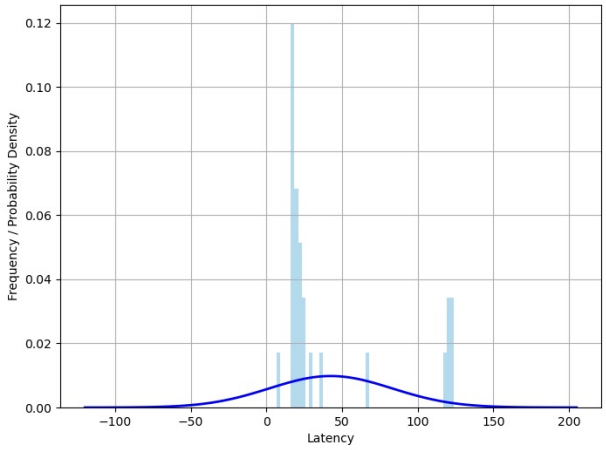

Figure A.2 Average latency to reach saturation accuracy of 75%: 42.53 seconds
Dateset: IMDB-BINARY || Compression Type: Uniform

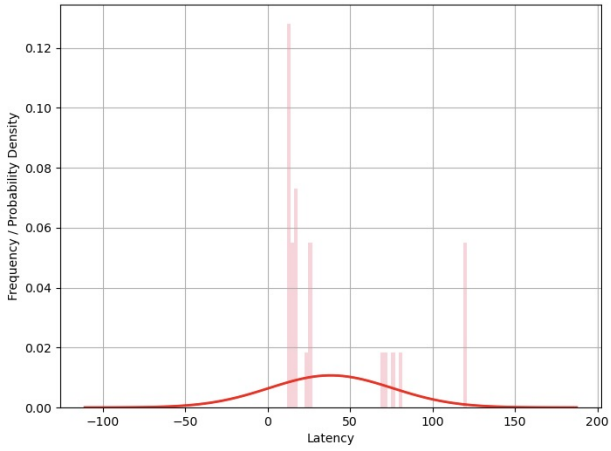

Figure A.3 Average latency to reach saturation accuracy of 75%: 38.21 seconds
Dateset: IMDB-BINARY || Compression Type: Coreset

Appendix B

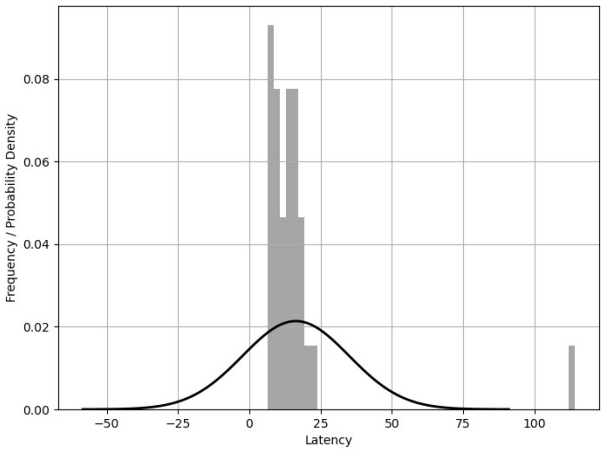

Figure B.1 Average latency to reach saturation accuracy of 56%: 16.31 seconds
Dateset: IMDB-MULTI || Compression Type: None

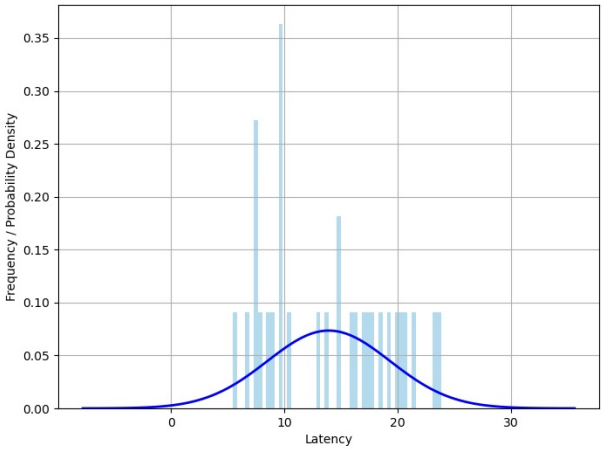

Figure B.2 Average latency to reach saturation accuracy of 56%: 13.92 seconds
Dateset: IMDB-MULTI || Compression Type: Uniform

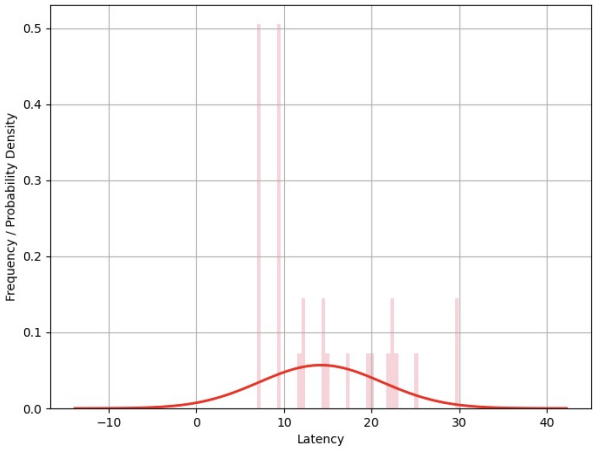

Figure B.3 Average latency to reach saturation accuracy of 56%: 14.19 seconds
Dateset: IMDB-MULTI || Compression Type: Coreset