Optimizing Communication in Parallel Deep Learning via Parameter Pruning

Siddharth Singh[1], Abhinav Bhatele[1]

[1]Department of Computer Science, University of Maryland

Abstract

1. Parallel training of neural networks at scale is challenging due to significant overheads arising from communication

2. Recently, parameter pruning algorithms have been proposed that can prune (set to zero) 80-90% of the parameters of a neural network without affecting it's test accuracy.

3. In this work, we propose a novel method that exploits the sparsity of these pruned networks to optimize communication in parallel training of large models.

4. We integrate our method in AxoNN [1] and improve the performance of a 2.7B parameter model on 384 GPUs by 17%.

Method

We integrate our method in AxoNN[1] and describe its working below:

1. We use the "Early Bird Ticket" algorithm [2] to identify an accuracy preserving sparse subnetwork after a few iterations of training.

2. We then convert the weight matrices of this sparse subnetwork to a 1D Sparse COOrdinate (COO) format.

3. For computation, we convert a sparse matrix into an ephemeral dense matrix and delete the dense matrix immediately after it has been used.

4. Since our computation is in dense, we utilize the extremely performant CuBLAS library. We thus match the performance of the unpruned network while saving memory

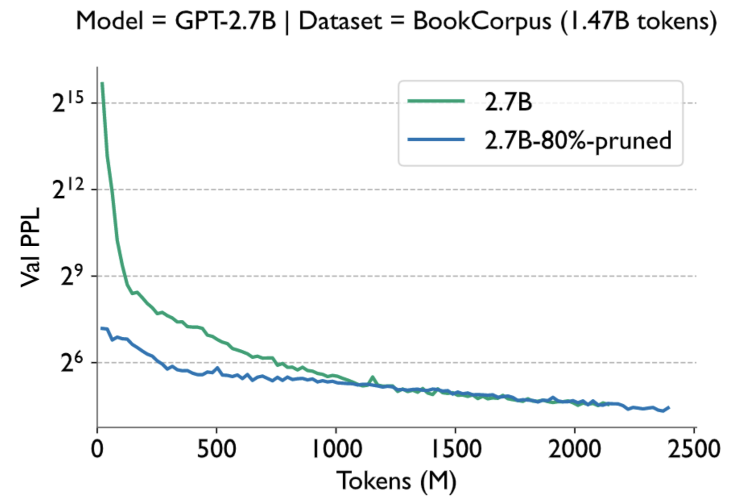

Fig 1: An 80% pruned network obtained via [2] matches the perplexity of the original model

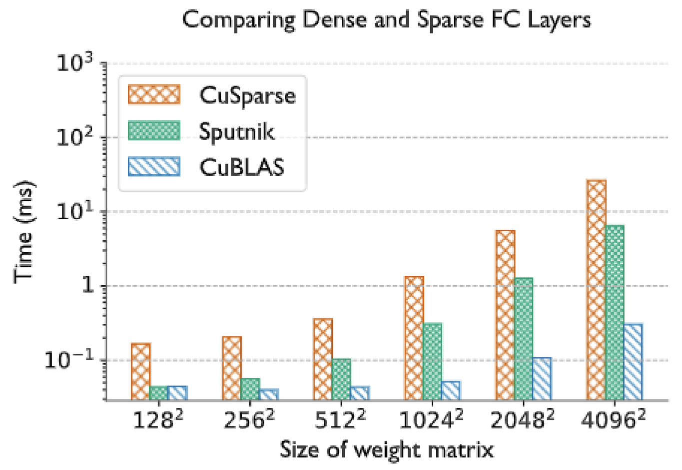

Fig 2: CuBLAS is 6-22x faster than sparse matrix computation libraries even at 90% sparsity

Benefits

1. Storing parameter matrices in sparse makes the neural networks extremely memory efficient. Our method can thus be used for inference or transfer learning in resource constrained environments.

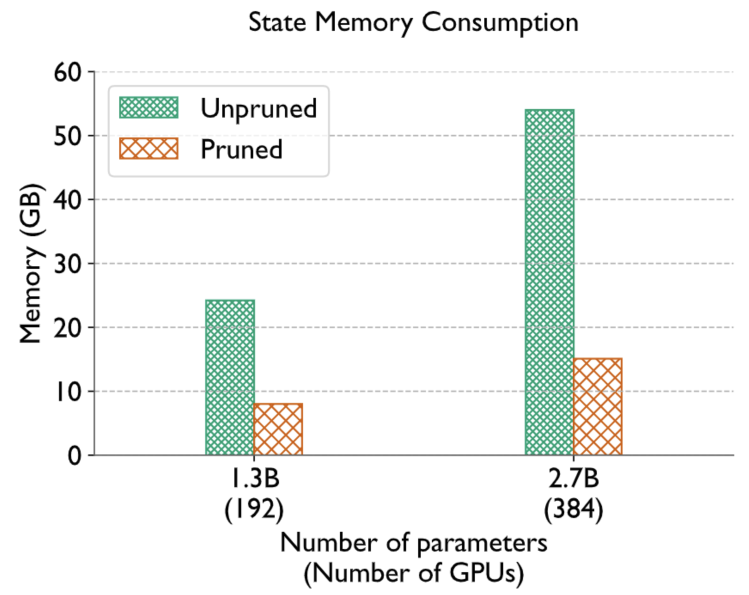

Fig 3: We save 2.8x and 3.5x memory for GPT-1.3B and GPT-2.7B

2. We reduce collective communication overhead by only communicating gradients for the unpruned parameters.

3. The saved memory also allows us to reduce the amount of GPU idle time that AxoNN spends in the "bubble" phase by decreasing the number of GPUs required to deploy a single model replica.

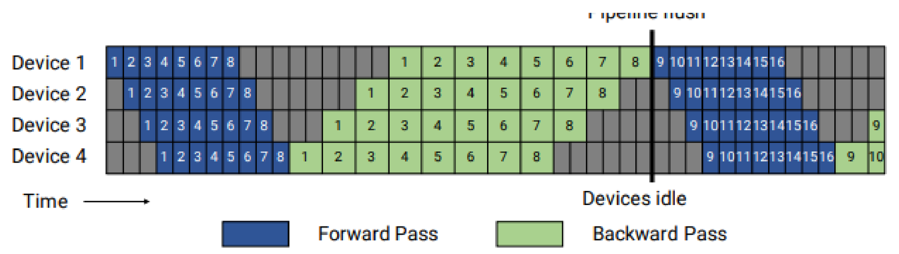

Fig 4: The idle time in AxoNN's pipelining is proportional to the pipeline depth (4 in the figure)

Results

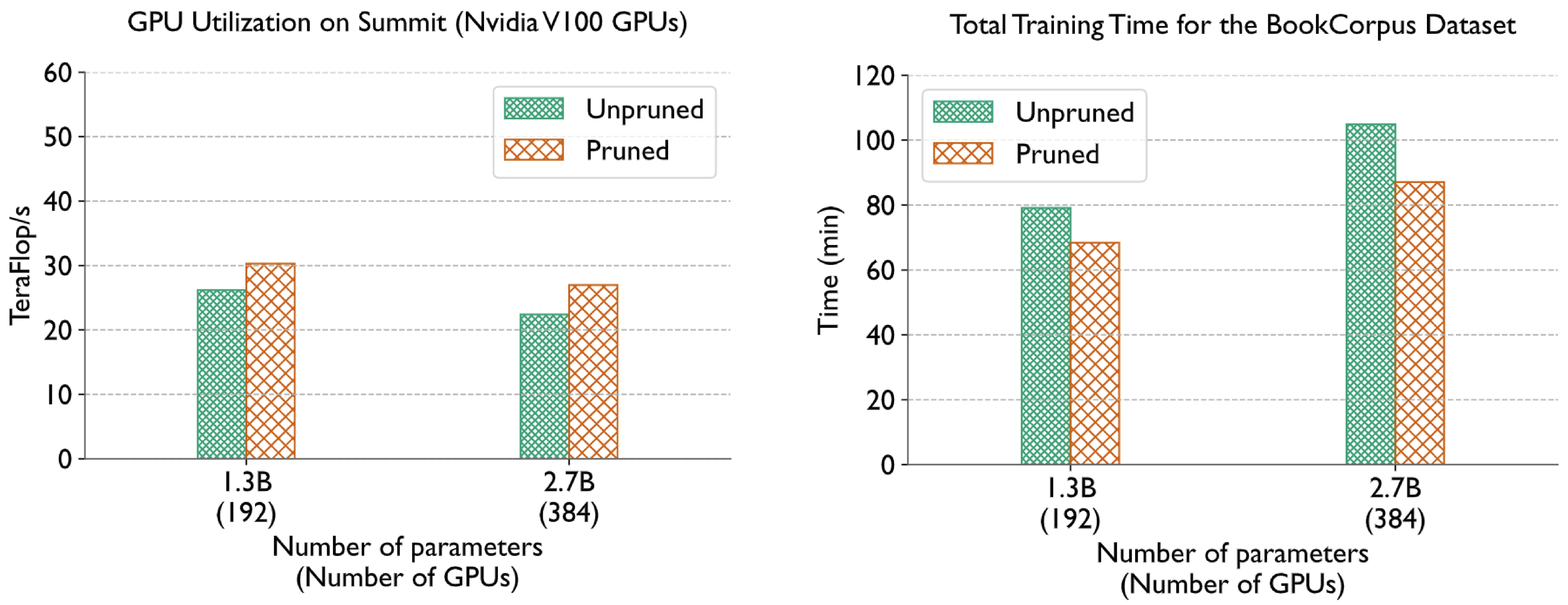

Fig 5: Comparison of teraflop/s (left) and total training time (right) for GPT-1.3B and GPT-2.7B on 192 and 384 GPUs respectively on Summit using AxoNN.[1]

Analysis

1. We observe a massive improvement of 66% in the collective communication time!

2. We observe a minor speedup of 6.34 % in the pipeline bubble time.

3. There is an additional 10% overhead in the compute phase due to the conversion of sparse matrices to dense..

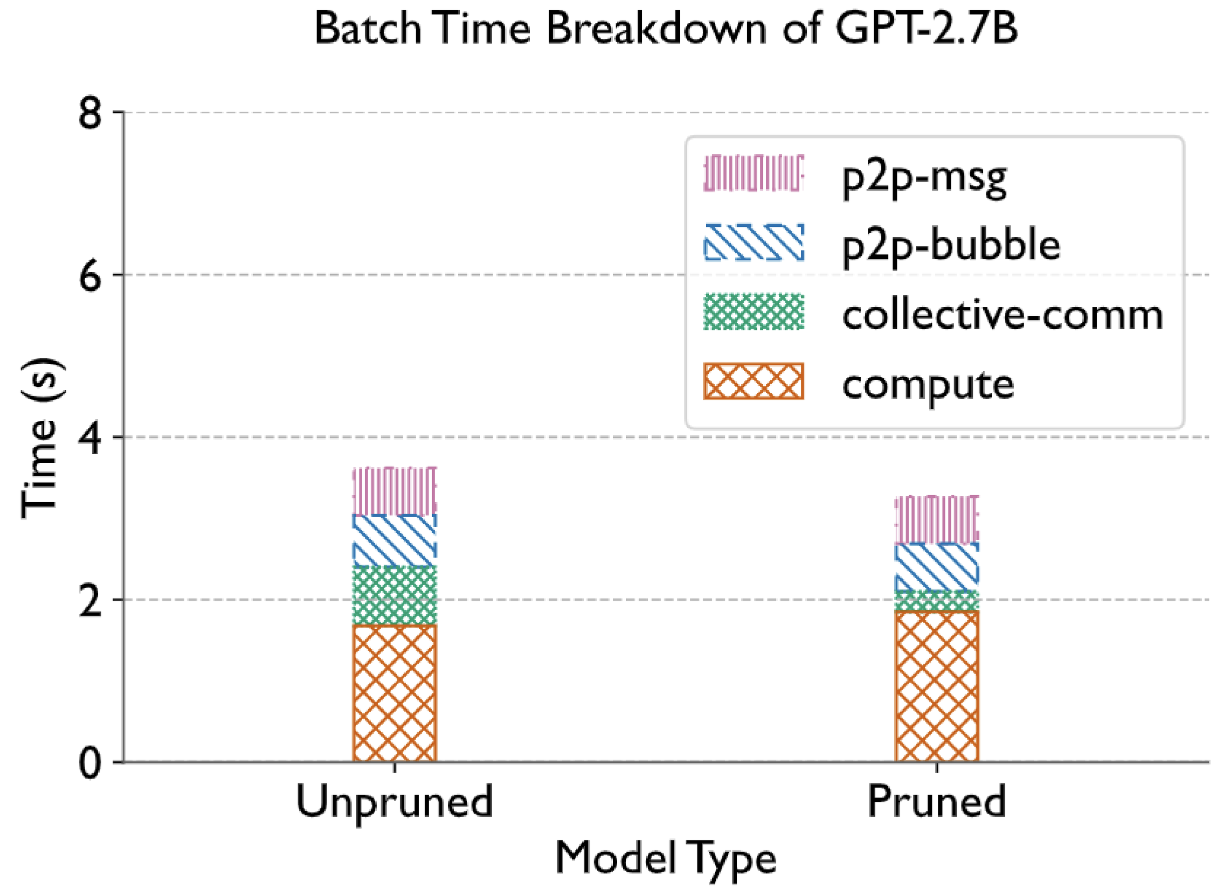

Fig 6: Comparison of communication and computation times for unpruned and pruned models

Conclusion

1. We introduced performance optimizations that exploit sparsity in pruned networks to reduce communication

2. We saved memory by storing sparse parameter matrices in a 1D SparseCOO format. For compute efficiency, we dynamically converted these to dense when needed.

3. We used the saved memory to reduce the pipeline depth and thus AxoNN's pipeline idle time. To optimize collective communication, we only communicated gradients for unpruned parameters

4. We improved the training times for GPT-1.3B by 14% and GPT-2.7B by 17% on 192 and 384 V100 GPUs of Summit respectively.

References

[1] Singh et al., AxoNN: An asynchronous, message-driven parallel framework for extreme-scale deep learning, IPDPS 2022

[2] You et al., Drawing early bird tickets: Toward more efficient training of deep neural networks, ICML 2020

This work was supported by funding provided by the University of Maryland College Park Foundation. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.