Efficient Deployment of Very Wide and Very Deep Hypersparse FFNs on FPGA

Paramdeep Singh
Computer Science and Engineering
Santa Clara University
Santa Clara, CA, USA
psingh7@scu.edu

David C. Anastasiu
Computer Science and Engineering
Santa Clara University
Santa Clara, CA, USA
danastasiu@scu.edu

Abstract

Model compression techniques such as quantization and pruning have shown great promise in drastically reducing model size without degrading model effectiveness. Quantization of model parameters when combined with parameter pruning results in a significantly reduced model size. However, such sparse neural networks have irregular structures. As such the forward pass (inference step) of such networks cannot be executed efficiently by processing hardware like GPUs. FPGA's offer a flexible platform to process irregular sparse networks. However, in order to fully realize the efficiency gains promised by the FPGA architecture, it is essential to minimize or completely eliminate off-chip memory accesses. Accommodating a large model completely on the FPGA fabric is restricted by the scarcity of available high-speed on-chip RAM, forcing a fraction of model weights to be stored in off-chip DRAM. We propose a method to accommodate very wide and very deep hypersparse feed forward networks (FFNs) completely on the FPGA fabric by compressing data structures in addition to quantizing the network parameters. Our method makes it possible to fit large FFNs completely on the FPGA fabric, resulting in inference performance almost 1000x higher than that of the state-of-the-art.

1 Introduction

In this project, we overcome challenges that inhibit the realization of all the benefits of the FPGA architecture to enable highly efficient inference at the edge. Specifically, we focus on eliminating off-chip memory accesses, optimizing usage of scarce Block Ram (BRAM) and parallelizing the custom processing logic. FPGAs are primarily used as custom processors, relying on off-chip DRAM for data. We alleviate this bottleneck by completely eliminating DRAM accesses. Our optimizations enable the deployment of very wide and very deep hypersparse FFNs completely on the FPGA fabric. FFNs are integral part of modern cognitive networks such as CNN's [1, 2] and Transformer-based LLMs [3, 4]. Our work results in inference performance that is orders of magnitude higher than the baseline.

2 Background

Compressed Sparse Row (CSR) is a widely used data structure for storing weight matrices in globally sparse networks. However, for extremely wide and deep quantized sparse FFNs, the memory demand for storing the indices of non-zeros (NNZs) can surpass that required for storing quantized weights. Table 1 shows the proportion of total memory occupied by indices arrays at different quantization

Preprint. Under review.

Table 1: Proportion of memory occupied by indices arrays

	Network Width			
Quantization (bits)	***1024***	***2048***	***4096***	***8192***
16	38.46%	40.74%	42.86%	44.83%
8	55.56%	57.89%	60.00%	61.90%
4	71.43%	73.33%	75.00%	76.47%

levels for very wide FFNs. In this work, we present an algorithm to reduce the number of bits required to store indices information, enabling the accommodation of very large FFNs completely on the FPGA fabric

3 Method

Our method reduces the total memory footprint of very wide and very deep sparse FFNs represented using the CSR data structure. Although applicable to any sparse FFN, our method focuses on pre-pruned [5, 6, 7] FFNs which are pruned using RadiX-Net pre-pruning [8]. RadiX-Net pre-pruning results in FFNs as expressive as their dense counterparts [9], while facilitating the encoding of the indices arrays into a compressed format. We develop novel decoding and encoding algorithms to compress and decompress the indices arrays.

3.1 Encoding Algorithm

The encoding algorithm is a one-time pre-processing step. It transforms the array of indices from the CSR format into a more memory-efficient representation. The method assumes that the FFN's width (M) is a power of 2 (e.g., 1024, 2048, 4096, ...) and considers the weight matrix as a single block with structured sparsity in the form $N{:}M$, where $N \ll M$. This sparsity pattern, referred to as quasi-unstructured sparsity, allows for a compact representation of indices.

3.2 Decoding Algorithm

The decoding algorithm is implemented as custom processing logic within the Processing Elemnt (PE) of each layer, and is invoked at each inference pass. It reconstructs the absolute values of indices from the compressed formats created by the created by the encoding algorithm. The decoding algorithm operates concurrently for all layers, ensuring efficient and high-throughput execution.

4 Results

We compared our method against that of Huang et al. [10]. Our results, shown in Table 2 are composed of two parts. The first part of the results compares the efficiency of our method with that of the baseline, both in terms of inference time and resource utilization. The second part of the results demonstrate that model effectiveness is retained after the optimizations performed by our method.

Table 2: Efficiency Results

Method	Efficiency (sec)		Resource Usage (%)			
	latency	Throughput	LUTs	FFs	BRAMs	DSPs
Baseline (Matlab)	124.07	N/A*	N/A*	N/A*	N/A*	N/A*
Baseline (Huang et. al.)	251.31	N/A*	48.43	26.86	55.44	4.17
Ours	**0.247**	**0.0020**	**10.76**	**3.62**	**94.35**	**3.03**

*N/A = not reported

5 Conclusion

To the best of our knowledge, our work is the very first attempt to deploy a very large pre-pruned FFN, which is pruned using Radix-Net pre-pruning, completely on the FPGA fabric. By combining the compression of sparse data structures and quantizing model parameters, we were able to accommodate a very wide and very deep sparse FFN completely on the FPGA fabric. This led to performance gains that were orders of magnitude greater compared to the SOTA.

Acknowledgments

GPU hardware for our research was provided by NVIDIA and Supermicro. Computing resources were also made possible by the Santa Clara University HPC Center.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[4] Lochan Basyal and Mihir Sanghvi. Text summarization using large language models: A comparative study of mpt-7b-instruct, falcon-7b-instruct, and openai chat-gpt models, 2023.

[5] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow, 2020.

[6] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow, 2020.

[7] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019.

[8] Jeremy Kepner and Ryan Robinett. Radix-net: Structured sparse matrices for deep neural networks. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, page 268–274. IEEE, May 2019.

[9] Kevin Kwak, Zack West, Hayden Jananthan, and Jeremy Kepner. Testing radix-nets: Advances in viable sparse topologies, 2023.

[10] Sitao Huang, Carl Pearson, Rakesh Nagi, Jinjun Xiong, Deming Chen, and Wen-mei Hwu. Accelerating sparse deep neural networks on fpgas. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7, 2019.