Accelerated COVID-19 CT Image Enhancement via Sparse Tensor Cores

Ayush Chaturvedi
Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, Virginia
ayushchatur@vt.edu

Wu-Chun Feng
Department Computer Science
Virginia Tech
Blacksburg, Virginia
feng@cs.vt.edu

***Abstract*—Many deep learning model architecture are an inspiration of how human brain works however their implementation in computer programming deviates in the sense that these networks over time become dense or are intentionally designed in such a way to achieve better generalization and accuracy whereas neural architecture in brain is highly sparse. In this work we target a similar deep learning model designed to enhance CT images of Covid-19 chest scans namely DD-Net (short for Dense Net and Deconvolution Network) from prior work of ComputeCovid19+. The model follows an auto encoder decoder architecture in the deep learning paradigm and has high dimensionality due to presence of stack convolution layers and deconvolution layers and thus takes many compute hours of training. We propose a set of techniques which target these two aspects of model - dimensionality and training time. We implement structured sparsity along with a hybrid training schedule. By pruning neurons we make the model sparse and thus reduce the effective dimensionality and then retrain this sparsified model with minimal additional overhead of re-training. We also apply set of techniques tailored with respect to underlying hardware in order to better utilize the existing components of hardware (such as tensor core) and thus reduce the overall time and associated computational cost required to train this model with the new hybrid training schedule.**

***Index Terms*—deep learning, sparse, optimizations, covid-19, tensor core**

I. Introduction

Deep learning architectures show tremendous results in complex tasks but their success is often associated with huge compute resources and cost in terms of power and iterative process involved in tuning the model thus it is critical to extensively and efficiently use available resources. Many optimization strategies target to save computations in the network via evolving the deep neural network architecture to be as near capable as human brain which uses mere 20W but is capable to performs exaflops of computation per second compared to state of the art supercomputers which consumes thousands of Kilo Watts of power to achieve similar exaflop compute capabilities. Not only does the brain have limited resources but it also does not require all of biological neurons at all times. This has inspired scientists to follow other strategies that try to reduce the dimensionality of models as an indirect attempt to reduce total computation required via pruning neurons that contribute insignificantly to model. Other attempts which try to save costs in terms of power involve custom hardware architectures and accelerators dedicated and tailored to cater to deep learning workloads which are essentially matrix multiplications. In this work we target one such deep learning model namely DDnet [3], short for DenseNet and Deconvolution based network designed to enhance low dose covid-19 chest scan images to high quality images.The architecture of the model is show in Figure 1. DDnet is an auto encoder decoder model which has key components such as skip connections, concatenate connections and repeated convolutions and deconvolutions. In an attempt to evolve DDnet to ”brain like” architecture yielding better accuracy with reduced dimesionality and thus computation costs we will exploring pruning strategies, more specifically we will explore What to prune? When to prune ? and How to prune?. Additionally in doing so we will then apply architecture aware optimizations in order to efficiently use the underlying hardware capabilities on Nvidia Volta and Ampere GPUs in order to accelerate sparse deep learning training.

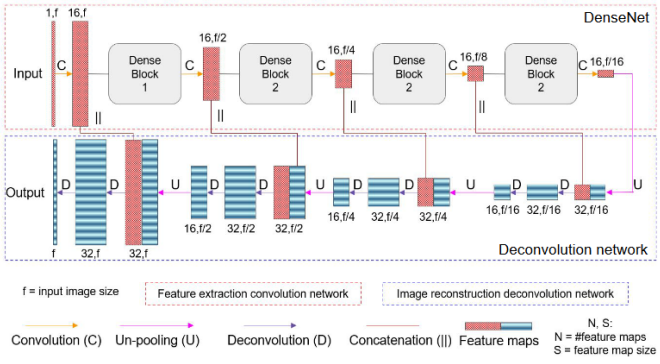

Fig. 1. DDnet Architecture.

II. Related Work

A. Sparse Training

Sparse training tries to mimic how the brain works, as an example [9] uses a training schedule that is inspired by the development of the human brain [10] [24]. Many works propose techniques to sparsify the deep neural network via pruning weights and neurons in a based on statistical formulation of neuron activation rate or rate of change of neuron over the

course of learning. This reduces simple parameters of the model, can shrink it substantially, and results in a new model that is essentially dense (i.e., can efficiently be executed on the same hardware as the original model) [21] [20] [14] [16]. Other works simply try to prune random weights, however this adds overheads for maintaining index structures and leads to less efficient execution on hardware that is optimized for dense computations [15] [18] [19].

III. Sparse optimization

In this section we describe our approach we carry out to efficiently reduce model dimensionality via selection of existing strategies that sparsify the existing model and enable sparse training without loosing much model accuracy with a little overhead in terms of computation and training time. Sections **??**–III-A discuss and formulate the recipes in deep learning training as to the model pruning strategies and schedule. Sections III-B discusses how latest Nvidia GPU architectures have dedicated on chip cores specialized for sprase matrix multiplication operations and how our optimizations will leverage these cores and will reduce the training time required for our sprasified model. We will compare our results in terms of training time and number of operation with the baseline implementation of Enhancement-AI (DDnet) from DL-FACT framework [4]. For test environment we will validate our results on ARC resources with Nvidia A100.

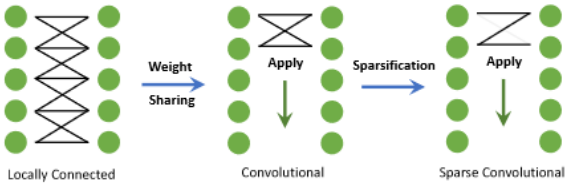

Fig. 2. Sparification in convolution layers.

A. Structured Sparsity

Unstructured sparsity does not guarantee the purning of entire neurons as there may be associated weights and biases for a connection of a neuron that have a constant high value throughout the training process. Thus in order to surely guarantee the pruning of entire neuron structured sparsity is the best way to go. Structured sparsity also aid the acceleration via hardware as entire blocks of weight and bias matrix are dropped thus reducing a significant portion of matmul operations. Fig 2. describes on high level how structured sparsity prune all the connection of a single or more than one neurons. Fig 3. Shows how structured sparsity and unstructured sparsity prunes the weight matrix stored for actual matrix operations. Channel and filters are the dimensions of the weight matrix which basically represent the number of connection from-to a particular neuron. We implement structured sparsity in DDNet with varying the amount of sparsity in the model from 10-90%. In order to select which neurons to prune we maintain a normal distribution of neuron weights and biases and prune those neurons that lie in the lower distribution. Post pruning we also retrain the model to calculate the additional training overhead required with varying sparsity.

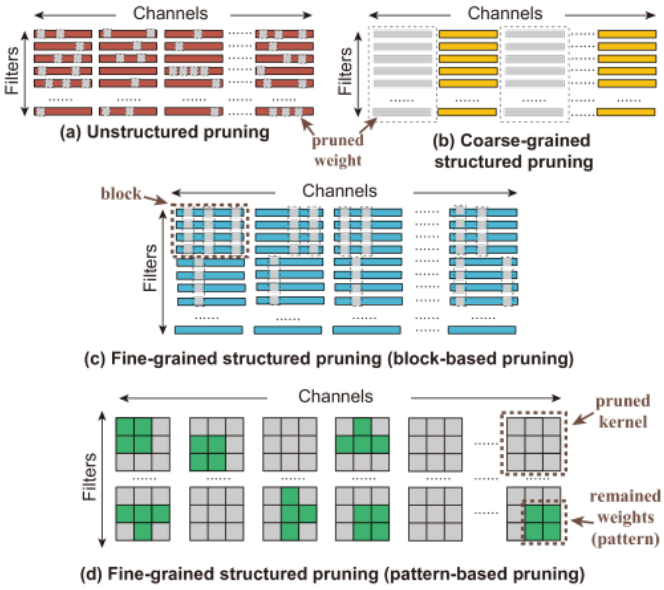

Fig. 3. Structured Vs Unstructured Pruning.

B. Leveraging Tensor Cores

Matrix-Matrix multiplication (GEMM) operations are at the core of neural network training, inference as these operations are used for multiplication of large matrices of input data and weights in the connected layers of the network. Starting from the Volta generation of GPUs Nvidia introduced tensor cores in the GPU cards. Tensor Cores and their associated data paths are custom-designed to dramatically increase floating-point compute throughput with high energy efficiency [11]. Ampere architecture builds on top of volta architecture and introduces hardware optimized sparse GEMM operations giving a significant reduction in number of clock cycles needed to compute FMA (Fused Multiply Add) operations [12]. [12] also shows the how sparse matrix multiplications are accelerated via hardware level optimizations on ampere architecture via CUDA APIs. In addition to hardware based optimized FMA operations, sparse tensor cores also support mixed precision training meaning for a fused multiply add operation on three matrices A, B and D, A and B can be of different precision scale than that of D. This further reduces the time (in terms of clock cycle), compute and storage required for FMA operations as lower precision data type use lesser number of bits for their representation and the results that overflow from the multiplication are stored with a little overhead in an accumulator of higher precision. This precision reduction makes our model susceptible to accuracy loss, but that is within our tolerable range. We implement this mixed precision training in specifically in sparsified convolution and deconvolution layers and discuss our results in section of the poster.

IV. Directions to Run

Please refer to the attached text file and to github repo: https://github.com/ayushchatur/2dnet for directions to run

References

[1] Tan, M. amp; Le, Q.. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research 97:6105-6114 Available from https://proceedings.mlr.press/v97/tan19a.html.

[2] C. Gong, Z. Jiang, D. Wang, Y. Lin, Q. Liu and D. Z. Pan, "Mixed Precision Neural Architecture Search for Energy Efficient Deep Learning," 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019, pp. 1-7, doi: 10.1109/ICCAD45719.2019.8942147.

[3] Garvit Goel, Atharva Gondhalekar, Jingyuan Qi, Zhicheng Zhang, Guohua Cao, and Wu Feng. 2021. ComputeCOVID19+: Accelerating COVID-19 Diagnosis and Monitoring via High-Performance Deep Learning on CT Images. In 50th International Conference on Parallel Processing (ICPP '21), August 9–12, 2021, Lemont, IL, USA. ACM, New York, NY, USA 11 Pages. https://doi.org/10.1145/3472456.3473523.

[4] Donald O. Hebb. 1949. The organization of behavior: A neuropsychological theory. Wiley, New York.

[5] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng. 2010. Tiled convolutional neural networks. In Advances in Neural Information Processing Systems 23. 1279–1287.

[6] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2015. Sparsifying Neural Network Connections for Face Recognition. (2015). arXiv:cs.CV/1512.01891.

[7] Hoefler, Torsten and Alistarh, Dan and Ben-Nun, Tal and Dryden, Nikoli and Peste, Alexandra. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. (2021) arXiv:10.48550/ARXIV.2102.00554.

[8] Pramod L. Narasimha, Walter H. Delashmit, Michael T. Manry, Jiang Li, and Francisco Maldonado. 2008. An integrated growing-pruning method for feedforward network training. Neurocomputing 71, 13 (2008), 2831 – 2847. https://doi.org/

[9] Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. 2018a. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. (2018). arXiv:cs.NE/1711.02017

[10] J. Hawkins. 2017. Special report : Can we copy the brain? - What intelligent machines need to learn from the Neocortex. IEEE Spectrum 54, 6 (2017), 34–71. https://doi.org/10.1109/MSPEC.2017.7934229

[11] Nvidia Volta Architecture White Paper https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[12] Nvidia Ampere Architecture White Paper https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

[13] Nvidia Apex Library for pytorch for optimizing deep learning training https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html

[14] Aditya Sharma, Nikolas Wolfe, and Bhiksha Raj. 2017. The Incredible Shrinking Neural Network: New Perspectives on Learning Representations Through The Lens of Pruning. (2017). arXiv:cs.NE/1701.04465

[15] Lutz Prechelt. 1997. Connection pruning with static and adaptive pruning schedules. Neurocomputing 16, 1 (1997), 49 – 61. https://doi.org/10.1016/S0925-2312(96)00054-9

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014b. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J. Mach. Learn. Res. 15, 1 (Jan. 2014), 1929–1958

[17] Subutai Ahmad and Luiz Scheinkman. 2019. How Can We Be So Dense? The Benefits of Using Highly Sparse Representations. (2019). arXiv:cs.LG/1903.11257

[18] Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. 2017. The Power of Sparsity in Convolutional Neural Networks. (2017). arXiv:cs.CV/1702.06257

[19] Deepak Mittal, Shweta Bhardwaj, Mitesh M. Khapra, and Balaraman Ravindran. 2018. Recovering from Random Pruning: On the Plasticity of Deep Convolutional Neural Networks. (2018). arXiv:cs.CV/1801.10447

[20] Maximilian Golub, Guy Lemieux, and Mieszko Lis. 2019. Full deep neural network training on a pruned weight budget. (2019). arXiv:cs.LG/1806.06949

[21] E. D. Karnin. 1990. A simple procedure for pruning back-propagation trained neural networks. IEEE Transactions on Neural Networks 1, 2 (1990), 239–242. https://doi.org/10.1109/72.80236

[22] Maxwell D. Collins and Pushmeet Kohli. 2014. Memory Bounded Deep Convolutional Networks. CoRR abs/1412.1442 (2014). arXiv:1412.1442 http://arxiv.org/abs/1412.1442

[23] A. P. Engelbrecht. 2001. A new pruning heuristic based on variance analysis of sensitivity information. IEEE Transactions on Neural Networks 12, 6 (2001), 1386–1399. https://doi.org/10.1109/72.963775

[24] Richard F Betzel, John D Medaglia, Lia Papadopoulos, Graham L Baum, Ruben Gur, Raquel Gur, David Roalf, Theodore D Satterthwaite, and Danielle S Bassett. 2017. The modular organization of human anatomical brain networks: Accounting for the cost of wiring. Network Neuroscience 1, 1 (2017), 42–68

[25] D. Abts et al., "Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), 2020, pp. 145-158, doi: 10.1109/ISCA45697.2020.00023.

[26] S. Lie, "Multi-Million Core, Multi-Wafer AI Cluster," 2021 IEEE Hot Chips 33 Symposium (HCS), 2021, pp. 1-41, doi: 10.1109/HCS52781.2021.9567153.

[27] X. Zhou, Z. Du, Q. Guo, S. Liu, C. Liu, C. Wang, X. Zhou, L. Li, T. Chen, and Y. Chen. 2018. Cambricon-S: Addressing Irregularity in Sparse Neural Networks through A Cooperative Software/Hardware Approach. In 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 15–28. https://doi.org/10.1109/MICRO.2018.00011

[28] Jingyang Zhu, Jingbo Jiang, Xizi Chen, and Chi-Ying Tsui. 2017. SparseNN: An Energy-Efficient Neural Network Accelerator Exploiting Input and Output Sparsity. (2017). arXiv:cs.LG/1711.01263

[29] Ashish Gondimalla, Noah Chesnut, Mithuna Thottethodi, and T. N. Vijaykumar. 2019. SparTen: A Sparse Tensor Accelerator for Convolutional Neural Networks. In Proceedings of the 52nd Annual IEEE/ACM International Sym- posium on Microarchitecture (MICRO '52). Association for Computing Machinery, New York, NY, USA, 151–165. https://doi.org/10.1145/3352460.3358291

[30] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna. 2020. SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training. In 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). 58–70. https://doi.org/10.1109/HPCA47549.2020.00015