2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) | 978-1-7281-8643-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISPASS51385.2021.00043

Sparseloop: An Analytical, Energy-Focused Design Space Exploration Methodology for Sparse Tensor Accelerators

Yannan Nellie Wu
MIT
Cambridge, US
nelliewu@mit.edu

Po-An Tsai
NVIDIA
Westford, US
poant@nvidia.com

Angshuman Parashar
NVIDIA
Westford, US
aparashar@nvidia.com

Vivienne Sze
MIT
Cambridge, US
sze@mit.edu

Joel S. Emer
MIT / NVIDIA
Cambridge, US
emer@csail.mit.edu

***Abstract*—This paper presents Sparseloop, the first infrastructure that implements an analytical design space exploration methodology for sparse tensor accelerators. Sparseloop comprehends a wide set of architecture specifications including various sparse optimization features such as compressed tensor storage. Using these specifications, Sparseloop can calculate a design's energy efficiency while accounting for both optimization savings and metadata overhead at each storage and compute level of the architecture using stochastic tensor density models. We validate Sparseloop on a well-known accelerator design and achieve ∼99% accuracy in terms of runtime activities (*e.g.*, compressed memory accesses). We also present a case study that highlights the key factors (*e.g.*, uncompressed traffic, data density) that affect sparse optimization features' impact on energy efficiency. Tool available at: *https://github.com/NVlabs/timeloop*.**

***Index Terms*—analytical modeling, sparse tensor accelerators**

I. Introduction

Many popular applications (*e.g.*, deep neural networks [1], graph algorithms [2]) involve tensor computations (*e.g.*, cross products) whose operand and result tensors can have sparsity (*i.e.*, fraction of zeroes) ranging from 10^{-6}% to 100% [3]. Due to the nature of multiplication, zero multiplicands always result in zero products. Such computations (which are called as *ineffectual*) can be exploited by hardware *sparse optimization features* to improve energy efficiency and throughput. We classify these sparse optimization features into three categories: zero-gating, zero-skipping, and zero-compression. Zero-gating improves energy efficiency by keeping the associated hardware components idle for ineffectual computations. Zero-skipping further improves throughput by skipping cycles where ineffectual computations would have taken place. Zero-compression reduces required storage by only storing nonzero values.

In recent years, a variety of sparse tensor accelerators [3]–[12] have been proposed. Based on the designer's intuitions, each design applies variations of the aforementioned sparse optimization features differently to the storage and compute levels of the architecture. However, these specific designs are just points in a large and diverse space of sparse tensor accelerators. A fast, flexible, and accurate modeling framework would enable architects to perform early design space exploration in the complete space instead of picking specific points based on intuition. Existing tensor accelerator models are either very detailed and design-specific [3]–[13], leading to slow and limited design space exploration, or fast and flexible but unable to systematically evaluate the impact of sparse optimization features [14]–[21], resulting in inaccurate modeling. In this work, we propose Sparseloop, an analytical modeling infrastructure for performing fast design space exploration of sparse accelerators that vary in both (1) properties associated with sparsity (*e.g.*, compression formats, ineffectual operations' gating/skipping, and workload attributes), and (2) architecture properties (*e.g.*, organization of the storage hierarchy). To the authors' knowledge, Sparseloop is the first analytical model that allows systematic evaluation of sparse tensor accelerators.

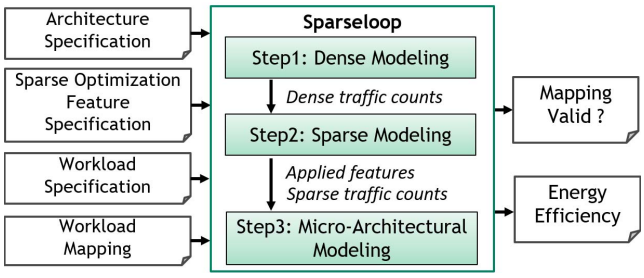


Fig. 1. Sparseloop High-Level Framework

II. Sparseloop High-Level Framework

Fig. 1 shows an overview of the Sparseloop framework. Sparseloop needs the following inputs: (1) an architecture specification that describes the levels in the architecture (*e.g.*, there are two storage levels and one compute level), the components in each level (*e.g.*, there are four ALUs in the compute level), and the components' hardware attributes (*e.g.*, each ALU has a data-width of 16 bits), (2) a workload specification that provides the shape and density for operand and result tensors (*e.g.*, a DNN's weight operand tensor has a shape of 3×7×7 and a density of 0.3), (3) a list of sparse optimization features supported by the architecture (*e.g.*, compute level supports *zero-gating*), and (4) a workload mapping (*i.e.*, the scheduling of data movement and compute in space and time).

978-1-7281-8643-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPASS51385.2021.00043

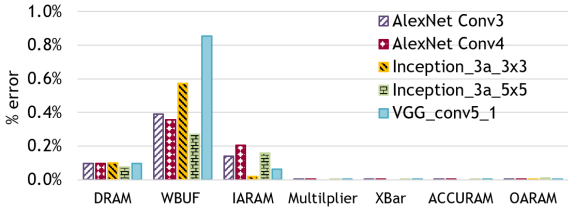
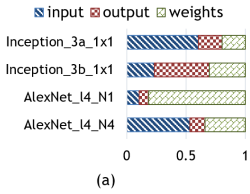


Fig. 2. Component runtime activity validation on SCNN architecture [6]. *DRAM*: off-chip storage level, *WBUF, IARAM, ACCURAM, & OARAM*: on-chip storage levels, *Multipliers*: compute level.



Layer #	Inputs	Outputs	Weights
Inception_3a_1x1	0.71	0.66	0.37
Inception_3b_1x1	0.53	0.33	0.41
Alexnet_conv4_N1/N4	0.39	0.43	0.37

(b)

Fig. 3. (a) Uncompressed DRAM traffic breakdown for different layers. (b) Layer densities N* stands for the batch size of the workload.

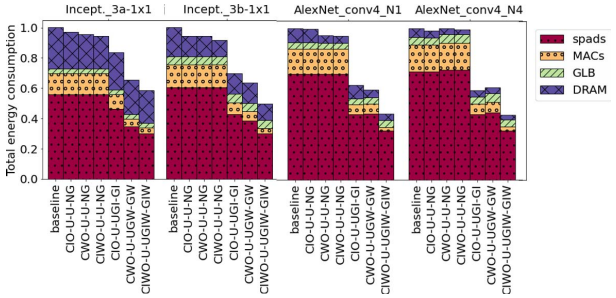


Fig. 4. Various optimizations applied to Eyeriss-based [4] topology. *baseline* refers to a dense architecture. Other architectures are named based on the applied sparse optimizations at four levels separated by -: DRAM-GLB-spads-MACs. Tensors: *I*: input, *W*: weight, *O*: output. Sparse optimizations: *U*: uncompressed, *C*: compressed, *NG*: no gating, *G*: zero-gating

Given these inputs, Sparseloop evaluates the validity of the mapping and (for valid mappings) its associated energy consumption. Since comprehensive modeling of a sparse architecture is a complex problem, Sparseloop carefully decomposes the original intertwined problem into three more tractable modeling steps: *dense modeling*, *sparse modeling*, and *micro-architectural modeling*, as shown in Fig. 1. This clean separation significantly reduces the complexity at each step and enables easy integration of extensions of different types. Dense modeling does not consider any sparse optimization features and analyzes the dense dataflow described by the user-specified mapping to derive *dense traffic counts*, which include the number of accesses to storage and dense arithmetic operations. The sparse modeling step adjusts the dense traffic counts to reflect the impact, *i.e.*, the savings and overhead, of sparse optimization features. To ensure fast modeling speed required by analytical modeling, statistical tile density models are used to derive *sparse traffic counts*, which include statistical characteristics of accesses and computes. Finally, the micro-architectural modeling step refines the sparse traffic counts based on the micro-architectural details (*e.g.*, bank access width) and the design's sparse optimization features. The final energy consumption is also evaluated at this step.

III. Experimental Results

We integrated our proposed sparse tensor accelerator modeling methodology to enhance an existing dense tensor analytical modeling framework, Timeloop [14]. To accurately capture the energy consumption of the fine-grained actions introduced by the sparse optimization features, we used Accelergy [22] as the energy estimation back-end. We validate Sparseloop on SCNN [6], a well-known sparse DNN accelerator. We select several convolutional layers from three DNN networks [23]–[25] as validation workloads and compare the runtime activities of the components (*e.g.*, the number of reads and writes to the storage levels), with the ground truth data generated by a custom SCNN simulator. Fig. 2 shows that Sparseloop's results are 99% accurate for all components in the architecture.

As a case study, we use Sparseloop to explore other potential designs by applying sparse optimization features based on the density of different workload tensor(s). We first study the effect of applying DRAM compression. Shown in Fig. 3(a), for different the batch sizes (*i.e.*, *N*) of AlexNet conv4, the relative input accesses are much smaller when *N=1*. Thus, Fig. 4 shows that *CIO-U-U-NG*, *i.e.*, compressing DRAM based on input and output tensor density, performs worse for *AlexNet_conv4_N1*. However, compressing the tensors that dominate the uncompressed DRAM traffic does not necessarily introduce the most savings, *e.g.*, for layer *inception_3a_1x1*, although inputs have almost 2x more uncompressed traffic, compressing weights introduces more savings as weights tensor density (46%) is much lower (Fig. 3(b)). We then study the various ways of applying different zero gating to scratchpads (*spads*) and MACs. AlexNet conv4's results indicate that considering both total uncompressed traffic and the tensor density is still not sufficient, as larger weight metadata storage size results in a much larger energy-per-access value for each weight metadata access. Thus, to comprehensively compare sparse optimization features, we should consider uncompressed traffic breakdown across tensors, tensor density, and architectural details.

IV. Conclusion

In this paper, we presented Sparseloop, an infrastructure that implements a generally applicable methodology for analytical evaluation of sparse tensor accelerators' energy efficiency. To reduce the complexity of comprehensive modeling, we present a modularized three-step evaluation process for analyzing a sparse tensor accelerator running a given workload, with sparsity being modeled statistically. We demonstrate that Sparseloop can achieve high estimation accuracy while retaining the high speed of analytical modeling, and help designers to understand the critical factors involved in sparse tensor architecture evaluation and design.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016.

[2] T. Mattson, D. Bader, J. Berry, A. Buluc, J. Dongarra, C. Faloutsos, J. Feo, J. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. Leiserson, A. Lumsdaine, D. Padua, S. Poole, S. Reinhardt, M. Stonebraker, S. Wallach, and A. Yoo, "Standards for graph algorithm primitives," in *HPEC*, 2013.

[3] K. Hegde, H. Asghari-Moghaddam, M. Pellauer, N. Crago, A. Jaleel, E. Solomonik, J. S. Emer, and C. Fletcher, "ExTensor: An accelerator for sparse tensor algebra," in *MICRO*, 2019.

[4] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *ISSCC*, 2016.

[5] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.

[6] A. Parashar, M. Rhu, A. M. asnd A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *ISCA*, 2017.

[7] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-X: An accelerator for sparse neural networks," in *MICRO*, 2016.

[8] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *ISCA*, 2016.

[9] Z. Zhang, H. Wang, S. Han, and W. J. Dally, "Sparch: Efficient architecture for sparse matrix multiplication," in *26th IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020.

[10] N. Srivastava, H. Jin, J. Liu, D. Albonesi, and Z. Zhang, "Matraptor: A sparse-sparse matrix multiplication accelerator based on row-wise product," in *MICRO*, 2020.

[11] S. Pal, J. Beaumont, D. Park, A. Amarnath, S. Feng, C. Chakrabarti, H. Kim, D. Blaauw, T. Mudge, and R. Dreslinski, "Outerspace: An outer product based sparse matrix multiplication accelerator," in *HPCA*, 2018.

[12] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *HPCA*, 2020, pp. 58–70.

[13] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *Asilomar*, 2017.

[14] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *ISPASS*, 2019.

[15] Y. N. Wu, V. Sze, and J. S. Emer, "An architecture-level energy and area estimator for processing-in-memory accelerator designs," in *ISPASS*, 2020.

[16] A. Samajdar, J. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," in *ISPASS*, 2020.

[17] H. Kwon, P. Chatarasi, V. Sarkar, T. Krishna, M. Pellauer, and A. Parashar, "Maestro: A data-centric approach to understand reuse, performance, and hardware cost of dnn mappings," *IEEE Micro*, vol. 40, no. 3, pp. 20–29, 2020.

[18] L. Ke, X. He, and X. Zhang, "Nnest: Early-stage design space exploration tool for neural network inference accelerators," in *ISLPED*, 2018.

[19] Y. Zhao, C. Li, Y. Wang, P. Xu, Y. Zhang, and Y. Lin, "DNN-Chip Predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures," in *ICASSP*, 2020.

[20] D. Yang, A. Ghasemazar, X. Ren, M. Golub, G. Lemieux, and M. Lis, "Procrustes: a dataflow and accelerator for sparse deep neural network training," in *MICRO*, 2020.

[21] N. Corp., "NVIDIA A100 Tensor Core GPU Architecture," Tech. Rep., 2020.

[22] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *ICCAD*, 2019.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.