Efficient Management of Scratch-Pad Memories in Deep Learning Accelerators

Subhankar Pal* Swagath Venkataramani† Viji Srinivasan† Kailash Gopalakrishnan†
*University of Michigan, Ann Arbor, MI †IBM TJ Watson Research Center, Yorktown Heights, NY
*subh@umich.edu †swagath.venkataramani@ibm.com {viji,kailash}@us.ibm.com

***Abstract*—A prevalent challenge for Deep Learning (DL) accelerators is how they are programmed to sustain utilization without impacting end-user productivity. Little prior effort has been devoted to the effective management of their on-chip Scratch-Pad Memory (SPM) across the DL operations of a Deep Neural Network (DNN). This is especially critical due to trends in complex network topologies and the emergence of eager execution. This work demonstrates that there exists up to a 5.2× performance gap in DL inference to be bridged using SPM management, on a set of image, object and language networks.**

We propose OnSRAM, a novel SPM management framework integrated with a DL accelerator runtime. OnSRAM has two variants, *viz.* OnSRAM-Static, which works on static graphs to identify data structures that should be held on-chip based on their properties, and OnSRAM-Eager, which targets an eager execution model (no graph) and uses a speculative scheme to hold/discard data structures. On a prototypical DL accelerator, OnSRAM-Static and OnSRAM-Eager achieve reductions in inference latency (batch size of 1) of 1.02-4.8× and 1.02-3.1×, respectively, over a baseline with no SPM management.

I. Introduction

The rapid success of Deep Neural Networks (DNNs) across the computing spectrum has triggered a revolution in computer architecture. Software-managed Scratch-Pad Memories (SPMs) are a common design choice in many computing platforms, ranging from GPUs to fixed-function and reconfigurable accelerators [1], [2], due to the ability to explicitly manage data with better energy efficiency. Recent years have witnessed specialized accelerators for DNNs [3]–[17], many of which use SPM hierarchies. The key to the success of such accelerators lies in how they are *programmed*, to sustain high utilization with minimal loss in end-user productivity.

An optimization that has received relatively little attention is the *effective management of on-chip SPM across the nodes of a DNN*, *i.e.* retaining the activations produced by a given layer on the on-chip SPM so as to eliminate writes/reads to/from external memory upon their reuse in subsequent layers. Two recent trends in modern DNN research make this a challenge.

- **Complex Topologies.** Recent DNN topologies such as ResNets [18], Inception [19], DenseNet [20], Multi-Head Attention [21], Neural Architecture Search (NAS) [22] and others rely on complex connections across layers (*e.g.* reconvergent paths) to achieve superior accuracy.
- **Eager Execution Model.** Recently, many popular Deep Learning (DL) frameworks (*e.g.* TensorFlow [23], [24], PyTorch [25]) have adopted an *eager execution* model which evaluates operations immediately, without building graphs.

II. Performance Impact of SPM Management

We study the performance impact of SPM management using a cycle-accurate performance simulator. Table I shows the speedups of 12 ConvNets, LSTM and Transformer DNNs [18], [19], [21], [26]–[33] compared to the case when there is no SPM management, *i.e.* all activations are double-buffered. The evaluation is done with a batch size of 1 on a simulated DL accelerator (specifications in Section IV). The infinite-sized SPM study illustrates a scope of 1.02-5.17× improvement in inference latency, thus underscoring the potential for hardware-aware SPM management.

TABLE I
Performance improvement using inter-node SPM management.

	AlexNet	*VGG 16*	*GoogLeNet*	*Inception-v3*	*Inception-v4*	*ResNet-50*	*SSD 300*	*ResNeXt*	*MobileNetV1*	*SqueezeNet*	*PTB*	*Multi-Head Attn*	***Geo Mean***
∞ SPM	1.04	1.19	1.94	1.64	1.58	1.75	1.31	3.86	5.17	2.84	1.02	1.06	**1.76**
1-Step	1.04	1.03	1.01	1.10	1.11	1.33	1.18	1.40	2.84	1.57	1.01	1.02	**1.24**

We first evaluate a simple scheme, *1-Step*, wherein we hold in SPM (if capacity permits) each activation whose reuse ends with the next scheduled node. This scheme is clearly optimal for sequential DNNs. However, for more recent networks like ResNet-50, InceptionV3, *etc.*, there is still a 1.45-2.7× gap to be bridged, which we address in this work.

III. Proposed OnSRAM Framework

We implement OnSRAM as an extension to a DL framework runtime in order to manage the accelerator's on-chip memory. We propose two variants, *viz.* OnSRAM-Static for static DNN graphs and OnSRAM-Eager for eager execution.

SPM Management for Static DNN Graphs. OnSRAM-Static receives an optimized DNN graph from the DL framework, and the key parameters of the AI accelerator (SPM capacity, external bandwidth, *etc.*) as its inputs. It derives (i) a graph execution plan that maximizes data reuse in the SPM, and (ii) an SPM management plan for each data structure as a tuple comprising of (a) whether the data structure is pinned in SPM or external memory ($PinLoc$), and (b) allocation and deallocation timesteps ($StartTS$, $EndTS$).

The node execution order plays a key role in determining the *liveness* of a data structure in SPM. We observe that by choosing an order in which activation-bound operations are placed as close as possible to their producers, their input activations will most likely be present in SPM. We realize such a schedule through a hybrid Breadth-First Search (BFS) - Depth-First Search (DFS) approach. Given this node execution schedule, OnSRAM-Static determines $PinLoc$, $StartTS$ and $EndTS$ using a heuristic-driven, low-overhead approach based on the following key characteristics of a data structure (D):

1) **Unused Liveness (UL)**, or the duration for which D is retained unused, in the SPM.
2) **Memory-Boundedness**, or how memory-bound D is, considering all operations in which it is used.
3) **Impact**, or how costly the operations that use D are.

*This work was done when the author was an intern at IBM TJ Watson Research Center, Yorktown Heights, NY.

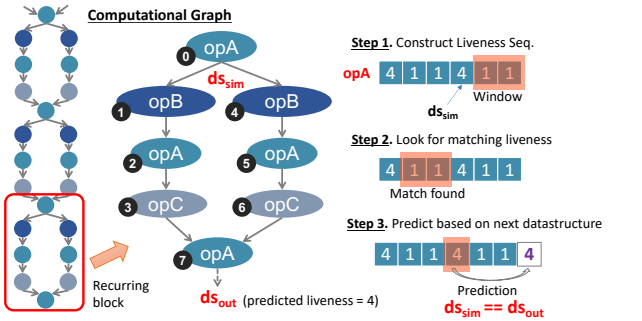

Fig. 1. Illustration of history-buffer based liveness prediction. The dark circles represent the timestep when the node is scheduled for execution.

So, *data structures with small UL, large memory-boundness and high impact are the best candidates to pin in SPM*. We compose these factors into a single Figure-of-Merit (FoM).

OnSRAM-Static iterates over data structures in FoM order to determine where each one is pinned. For each data structure, it is determined whether one could overwrite it onto any existing data structure in SPM. Once the overwrite-able candidates are identified, it is determined whether the SPM has enough capacity to hold the data structure *for all the timesteps that it is alive*. If so, OnSRAM-Static pins the data structure to SPM, else it is pinned to external memory.

SPM Management with Eager Execution. In eager execution, the DL framework does not build a graph, rather offloads operations to the accelerator in a user-defined sequence.

As the criticalities of data structures are unknown during offload, we propose to *predict* the data structure's characteristics based on the *history* of patterns observed. This is motivated by the observation that although operations arrive one-by-one, *most DNNs have recurring patterns*, *e.g.* the residual block in ResNet [18]. Thus, we propose to learn the reuse, liveness and criticality of data structures for the first few times the pattern repeats, and then use it during the rest of execution. Inspired by efforts in computer architecture on prefetching and value prediction [34]–[39], we employ a *history-buffer based prediction* scheme in OnSRAM-Eager. Its simplicity enables a small overhead in the critical path of an inference operation.

OnSRAM-Eager predicts, at offload-time, whether the operation's output(s) are to be held in the SPM, and if so, for what duration. It maintains two key state entities, *viz.* a *history buffer* and a *data structure table*. Each entry in the buffer is linked to one or more entries in the data structure table, corresponding to the output(s) produced by the operation. For each data structure, we maintain: (i) static information *viz.* size and the timestep at which it was entered into the table ($StartTs$), (ii) current information *viz.* timestep when it was last used ($LastUsedTs$) and its status, *i.e.* in SPM, in external memory, or evicted, and (iii) information predicted by OnSRAM-Eager *viz.* liveness, FoM, and a binary field ($isPinCand$) denoting if we should attempt to pin a data structure with similar characteristics in SPM.

Given a node, the history buffer and the data structure table are consulted to identify data structure(s) that are computationally similar to the output(s) of the current node. Based on their attributes, a prediction is made for $PinLoc$ and liveness (*i.e.* $StartTS$ and $EndTS$). We illustrate the history buffer lookup scheme for a simple computational graph in Figure 1.

We further propose a variant called *OnSRAM Eager Spec* that speculatively writes back only certain data structures which it pins in SPM, thus saving memory write-back bandwidth at the risk of mis-speculation. To recover from mis-speculation, the parent nodes that produced the data structure are re-executed by backtracking in the execution schedule.

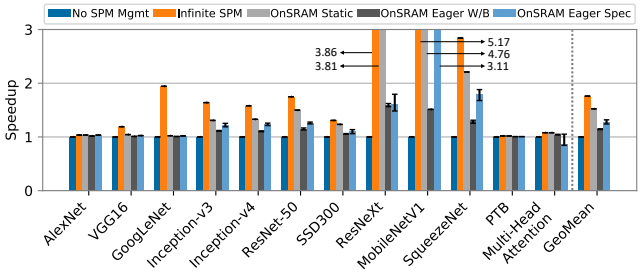

Fig. 2. Inference speedup with different SPM management policies.

IV. Evaluation

Our baseline accelerator, based on [11] and [1], has a 3 TFLOP dense 2D-systolic array and a 375 GFLOP SIMD special function unit array, supported by a 2 MB on-chip SPM with 384 GBps bandwidth, and a 32 GBps external memory. We present the inference speedup with the SPM management policies in Figure 2, for the DNNs in Table I. Weights are held in external memory and double-buffered during computation. For sequential DNNs like AlexNet, VGG and PTB, there is little gap to be bridged between *No SPM Mgmt* and *Infinite SPM*, as they are bottlenecked by the time to fetch weights. However, there exists a gap of 1.02-5.17× for the other DNNs.

OnSRAM Static exploits liveness/criticality-aware node execution ordering and data structure pinning in SPM to achieve a speedup of 1.02-4.8× over *No SPM Mgmt*. This improvement is already 90% of that achievable by *Infinite SPM*, while the remainder primarily stems from SPM capacity limitations. *OnSRAM Static* achieves pronounced improvements for topologies that exhibit far-flung activation reuse, such GoogLeNet and ResNeXt. Finally, it outperforms cache-based (LRU/FIFO) policies by up to 2.8× and *1-Step* by up to 2.7× (not shown).

We study the benefits with OnSRAM-Eager (6-entry history buffer, window size of 3) across 50 possible, valid execution orders for each DNN, illustrated using error bars in Figure 2. *OnSRAM Eager W/B* shows a limited speedup (average 1.15×) because every data structure is written back to external memory anticipating a reuse beyond its predicted lifetime in SPM. *OnSRAM Eager Spec*, in contrast, speculatively writes back only selected data structures to external memory. *OnSRAM Eager Spec* gives an overall speedup of 1.02-3.1× compared to *No SPM Mgmt* and is within 15% of *OnSRAM Static*.

V. Conclusion

Effective management of Scratch-Pad Memory (SPM) across the operations of a DNN is critical to bridge the compute-memory gap in AI accelerators. Two trends make this a challenging problem: a) evolution of topologies with complex inter-layer connections, and b) the eager execution model adopted in many DL frameworks. To address this, we propose OnSRAM, a SPM management framework with two variants, OnSRAM-Static and OnSRAM-Eager, that use hardware constraints and work within the compiler runtime of DNN accelerators. Our evaluation of OnSRAM-Static and OnSRAM-Eager on 12 DNNs with a prototypical DL accelerator yield 1.02-4.8× and 1.02-3.1× speedup for DNN inference.

REFERENCES

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden *et al.*, "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, Jun. 2017. [Online]. Available: https://doi-org.proxy.lib.umich.edu/10.1145/3140659.3080246

[2] S. Pal, S. Feng, D.-h. Park, S. Kim, A. Amarnath, C.-S. Yang, X. He, J. Beaumont, K. May *et al.*, "Transmuter: Bridging the efficiency gap using memory and dataflow reconfiguration," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, 2020, pp. 175–190.

[3] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 247–257, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1815993

[4] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *CVPR 2011 WORKSHOPS*, June 2011, pp. 109–116.

[5] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '14. New York, NY, USA: ACM, 2014, pp. 269–284. [Online]. Available: http://doi.acm.org/10.1145/2541940.2541967

[6] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 367–379. [Online]. Available: https://doi.org/10.1109/ISCA.2016.40

[7] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 243–254.

[8] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 267–278. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2016.32

[9] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 1–13. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2016.11

[10] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 380–392.

[11] B. Fleischer, S. Shukla, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky, N. Cao, C.Y. Chen, P. Chuang, T. Fox, G. Gristede, M. Guillorn, H. Haynie, M. Klaiber, D. Lee, S.Lo, G. Maier, M. Scheuermann, S. Venkataramani, C. Vezyrtzis, N. Wang, F. Yee, C. Zhou, P. F. Lu, B. Curran, L. Chang, K. Gopalakrishnan, "A scalable multi-teraops deep learning processor core for ai training and inference," in *Proc. VLSI Symposium*, June 2018.

[12] A. Majumdar, S. Cadambi, M. Becchi, S. T. Chakradhar, and H. P. Graf, "A massively parallel, energy efficient programmable accelerator for learning and classification," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 1, pp. 6:1–6:30, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2133382.2133388

[13] S. Venkataramani, A. Ranjan, S. Banerjee, D. Das, S. Avancha, A. Jagannathan, A. Durg, D. Nagaraj, B. Kaul *et al.*, "Scaledeep: A scalable compute architecture for learning and evaluating deep networks," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, vol. 00, June 2017, pp. 13–26. [Online]. Available: doi.ieeecomputersociety.org/10.1145/3079856.3080244

[14] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Quality programmable vector processors for approximate computing," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-46. New York, NY, USA: ACM, 2013, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/2540708.2540710

[15] "Google supercharges machine learning tasks with tpu custom chip," *Google Research blog*, 2016.

[16] O. Wechsler, M. Behar, and B. Daga, "Spring hill (nnp-i 1000) intel's data center inference chip," in *2019 IEEE Hot Chips 31 Symposium (HCS)*, Aug 2019, pp. 1–12.

[17] "Nvidia tensor cores: https://www.nvidia.com/en-us/data-center/tensorcore/," *NVIDIA blog*, 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[19] C. S. et. al., "Going deeper with convolutions," in *Proc. CVPR*, 2015.

[20] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016. [Online]. Available: http://arxiv.org/abs/1608.06993

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. USA: Curran Associates Inc., 2017, pp. 6000–6010. [Online]. Available: http://dl.acm.org/citation.cfm?id=3295222.3295349

[22] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1611.01578

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.

[24] Google. (2017) Google ai blog: Eager execution: An imperative, define-by-run interface to tensorflow. [Online]. Available: http://ai.googleblog.com/2017/10/eager-execution-imperative-define-by.html

[25] B. Steiner, Z. Devito, S. Chintala, S. Gross, A. Paszke, F. Massa, A. Lerer, G. Chanan, Z. Lin *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS 2019*, 2019.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, p. 2012.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

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

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[30] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2

[31] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR*, vol. abs/1611.05431, 2016. [Online]. Available: http://arxiv.org/abs/1611.05431

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size," 2016.

[34] Tien-Fu Chen and Jean-Loup Baer, "Effective hardware-based data prefetching for high-performance processors," *IEEE Transactions on Computers*, vol. 44, no. 5, pp. 609–623, May 1995.

[35] K. Wang and M. Franklin, "Highly accurate data value prediction using hybrid predictors," in *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1997, pp. 281–290.

[36] T.-F. Chen and J.-L. Baer, "A performance study of software and hardware data prefetching schemes," in *Proceedings of the 21st Annual International Symposium on Computer Architecture*, ser. ISCA '94. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994, pp. 223–232. [Online]. Available: http://dx.doi.org/10.1145/191995.192030

[37] K. Wang and M. Franklin, "Highly accurate data value prediction using hybrid predictors," in *Proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture*, ser. MICRO 30. Washington, DC, USA: IEEE Computer Society, 1997, pp. 281–290. [Online]. Available: http://dl.acm.org/citation.cfm?id=266800.266827

[38] P. Marcuello, J. Tubella, and A. Gonzalez, "Value prediction for speculative multithreaded architectures," in *MICRO-32. Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture*, Nov 1999, pp. 230–236.

[39] B. Calder, G. Reinman, and D. M. Tullsen, "Selective value prediction," in *Proceedings of the 26th Annual International Symposium on Computer Architecture*, ser. ISCA '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 64–74. [Online]. Available: http://dx.doi.org/10.1145/300979.300985