SISCA: An <u>S</u>RAM <u>I</u>n-<u>S</u>itu <u>C</u>omputation <u>A</u>ccelerator

Proposed Architecture

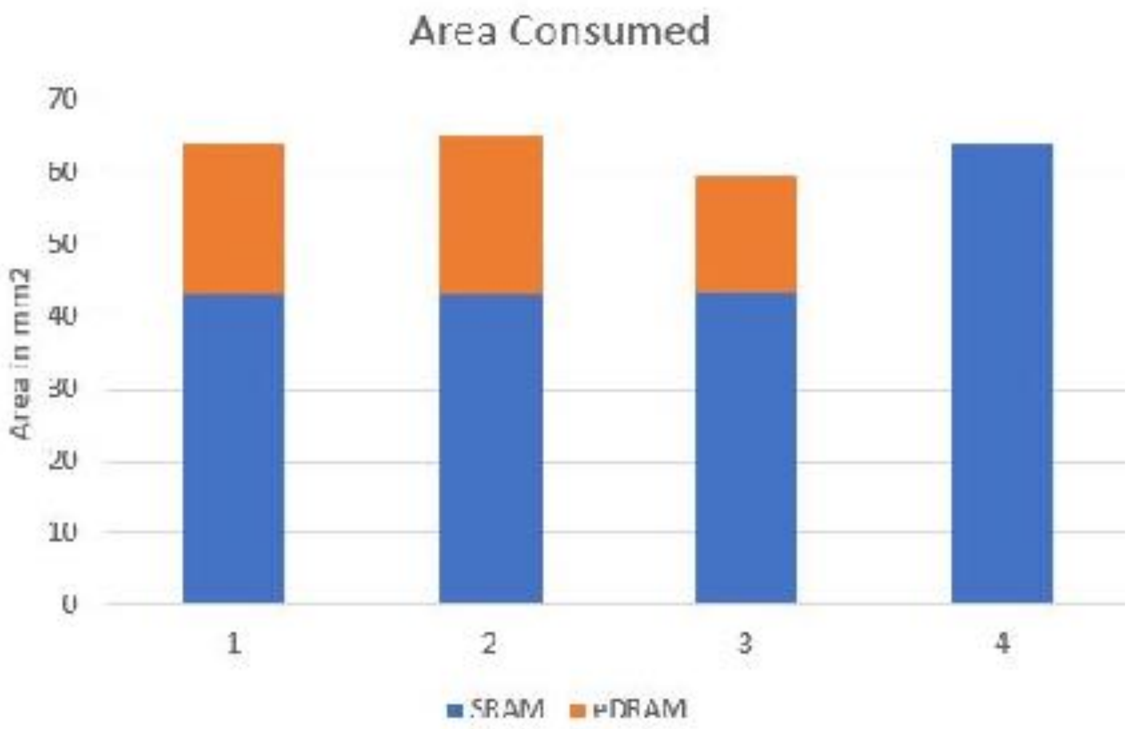
- SRAM Computing: Limited by memory capacity.
- Twist in the story: A hybrid architecture with SRAM and eDRAM.
- eDRAM is placed to prefetch the weights necessary for future computations.

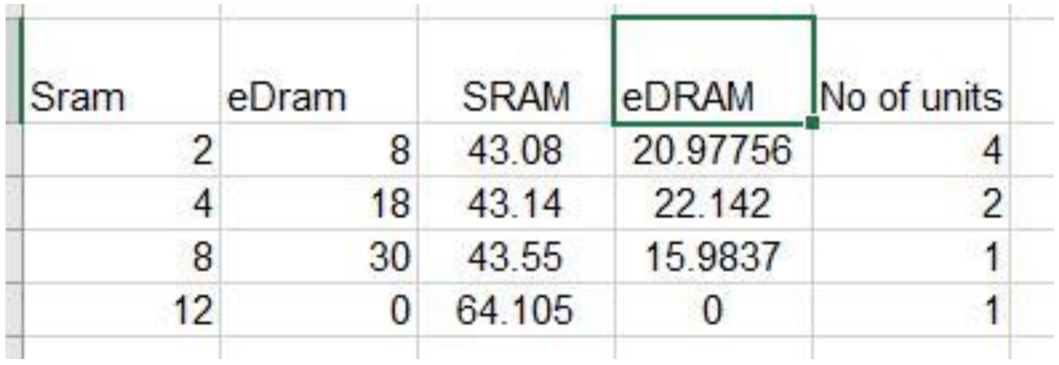
Design space exploration

- We tried different combinations of SRAM and eDRAM to find a sweetspot between in-situ computations and memory storage.

- Optimal Tile size:

    - SRAM: 2MB, eDRAM: 8MB





Sram	eDram	SRAM	eDRAM	No of units
2	8	43.08	20.97756	4
4	18	43.14	22.142	2
8	30	43.55	15.9837	1
12	0	64.105	0	1

Design space exploration

- The proposed tile can perform 16K logical operations in one clock cycle.

- Overhead: We need to perform multiplication operations. But in-situ support only logical and shift operations.

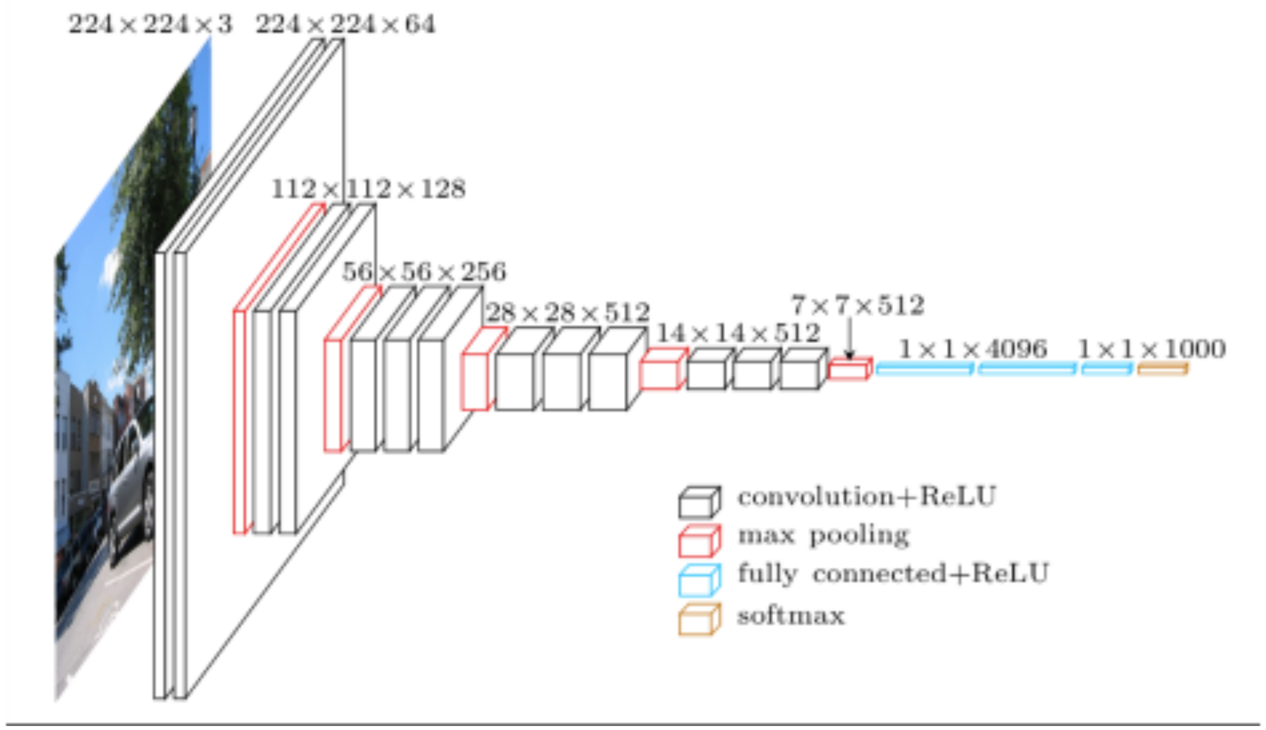
- Need to run each computation 16 iterations by shifting and accumulating the partial sums.

- This overhead reduces as the weights are pruned.

- Binary weights can perform computations is 1 iteration only.

Benchmarks

- We used VGG16 as a benchmark to evaluate performance and energy consumption of the proposed architecture.



source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Benchmarks

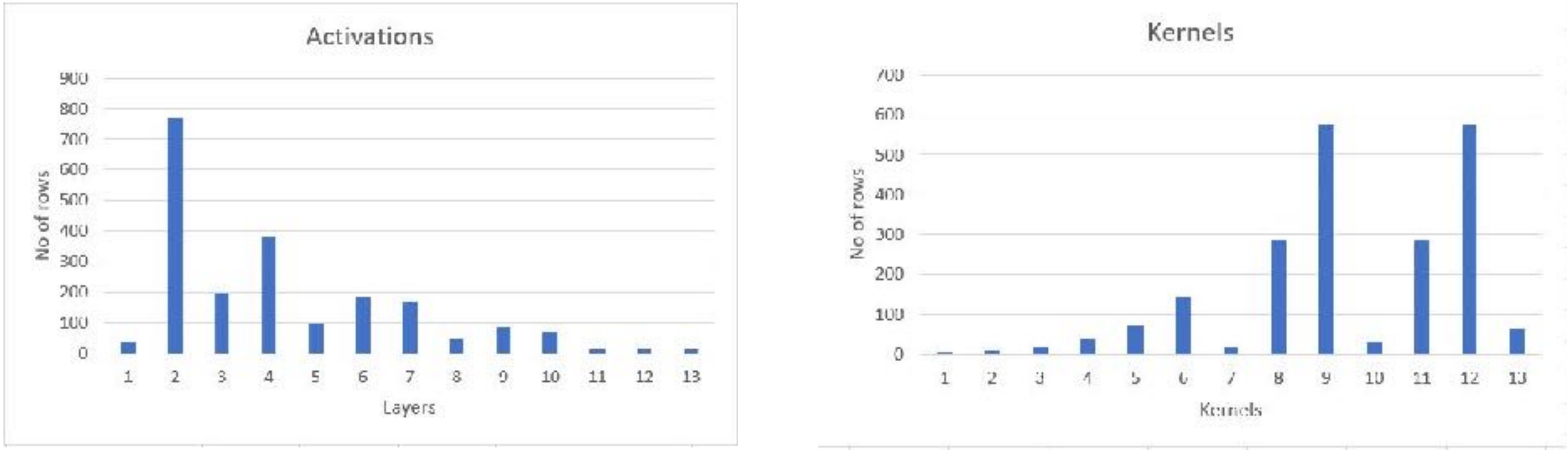
- Assumption: For the convolutional layer we have assumed that all the neurons in the IF map will have a partial product with all the values in the kernel of the layer. Ideally only around 70% of IF map values have partial products with all the kernels.

- We tried to manually map VGG16 to both DaDianNao and SISCA architecture.

Results

- To map VGG16 over DaDianNao (DDN), we need 8 DDN nodes to accomodate all the weights.

- For an analogous area, SISCA would need 32 tiles.



Distribution of Activations and Kernels over the proposed architecture. X-axis represents the layers and Y-axis represents the number of rows taken in one SISCA tile

Results

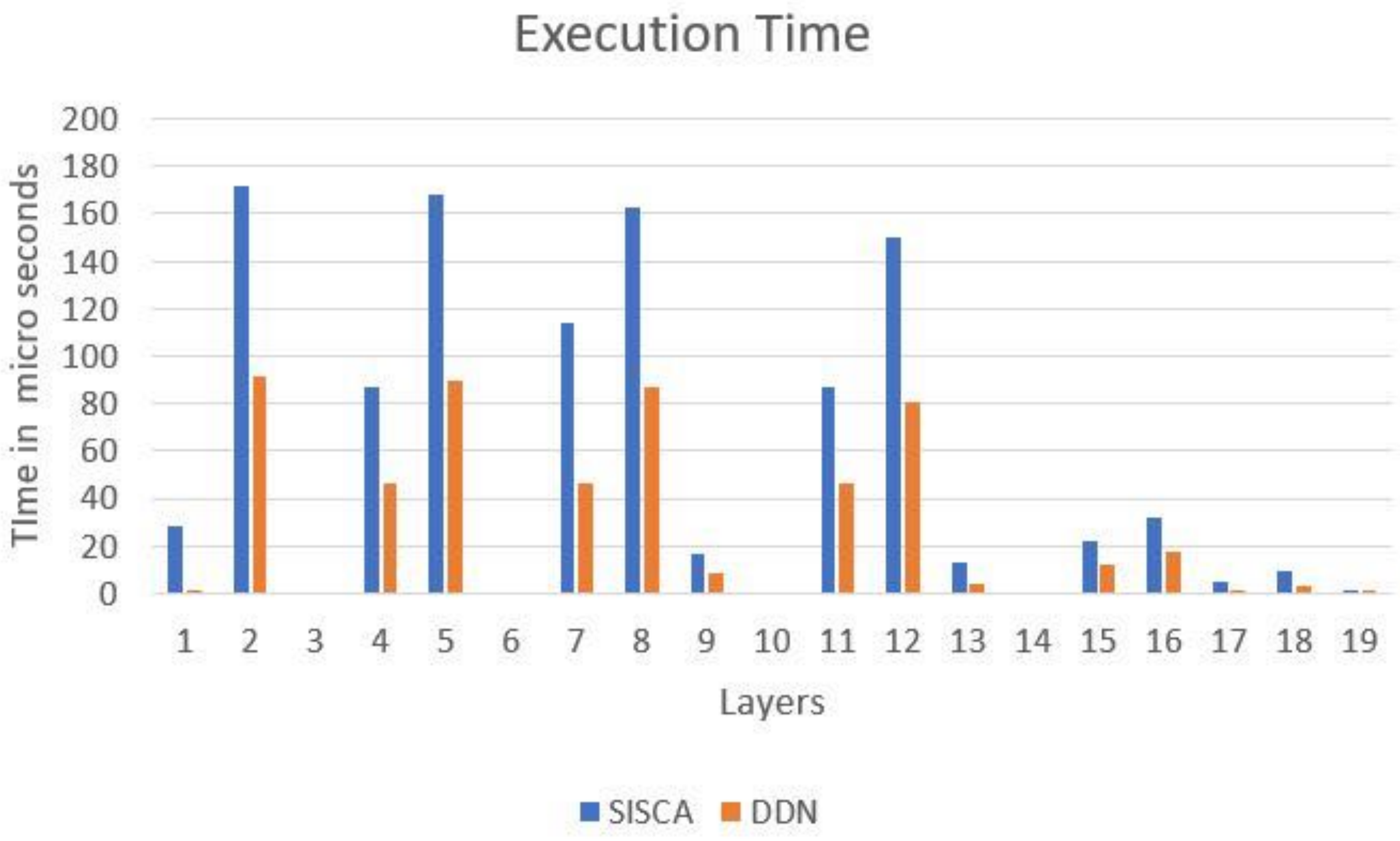
- The number of rows allocated in a tile to activation and weights depend on their distribution.

- Obsevation: Initial convolutional layers are dominated by activations and later are dominated by weights.

- Intial layer execution uses more tile rows for activations, where as final layers uses most of the tile rows to store weights.

- This enabled us to get maximum computations out of the available hardware resources with minimal data moment.

Results



Note: The numbers do not include the HT link communication overhead

Observations

- To increase the number of computations in SISCA, we increased the number of sub-arrays per node. We observe that 8 times increase in the number of sub-arrays increased the total area by 3.2 times for the same memory.

- SRAM in-situ computation architecture is limited by the area, in turn limiting us with the number of computational units and memory storage per node. DRISA on the other hand has dense memory, thereby allowing more computational units and memory.

- If SISCA uses 1 bit weights and 8 bit activations as DRISA, there is a good chance that it will out perform DaDianNao in performance and energy consumption. Although it may not be enough to outperform DRISA.

References

- S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, “A 28 nm configurable memory (tcam/bcam/sram) using push-rule 6t bit cell enabling logic-in-memory,” IEEE Journal of Solid-State Circuits, 2016.

- Shuangchen Li, Dimin Niu, Krishna T. Malladi, Hongzhong Zheng, Bob Brennan, and Yuan Xie. 2017. DRISA: a DRAM-based Reconfigurable In-Situ Accelerator. In Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50 '17). ACM, New York, NY, USA, 288-301.

- S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw and R. Das, "Compute Caches," 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, TX, 2017

- A. Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, 2016

- Y. Chen et al., "DaDianNao: A Machine-Learning Supercomputer," 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, 2014, pp. 609-622.