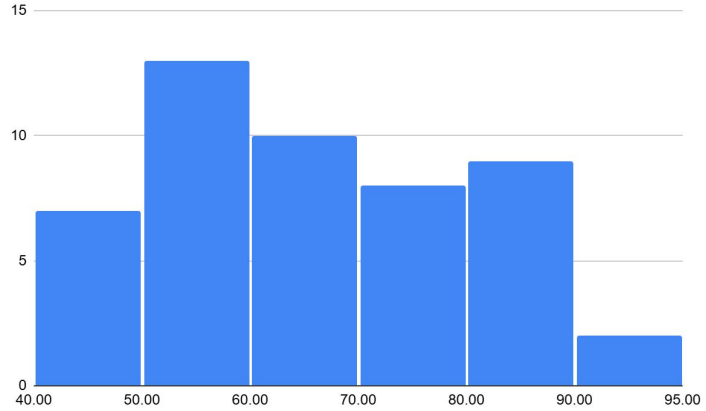
Lecture 11:
CNN & Transformer Accelerators

Some Notes

- There will be a around of project meeting on April 24. https://docs.google.com/spreadsheets/d/1WvGBRjQGjdfo3M7JE29L-Z4fmzmhly5xYN5Ynw5hud4/edit?usp=sharing

- No Quiz today.

Midterm



- Average: 68 (After)
- Median: 67
- Average: 55 (Before)

NYU SAI LAB

Recap

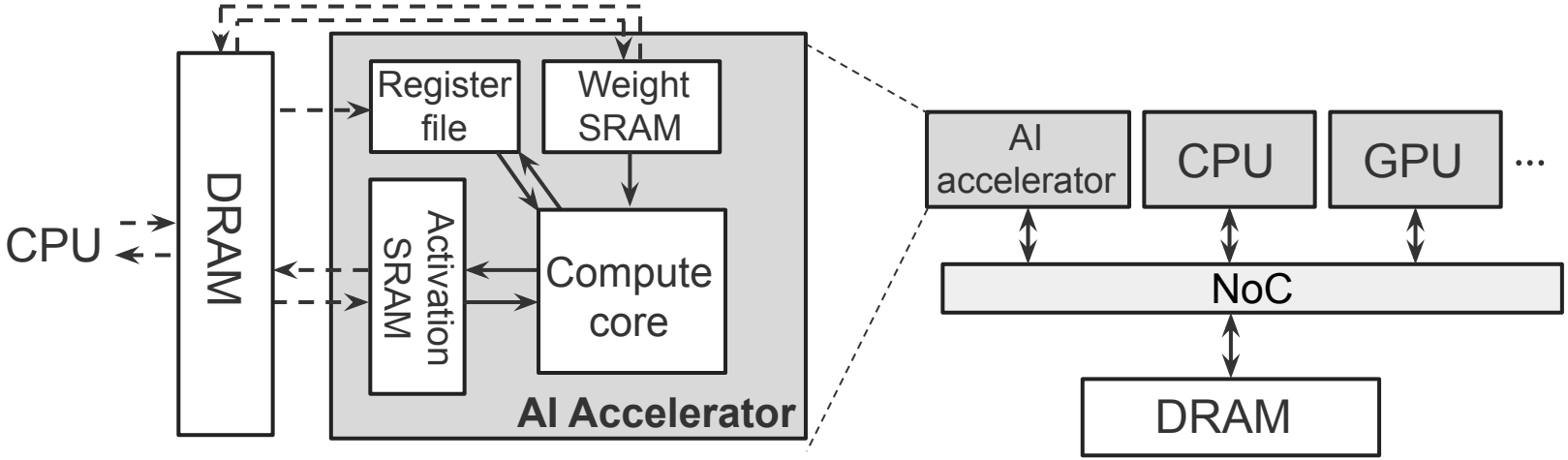
- Machine Learning Compiler
- Introduction to AI Accelerator
- CNN Accelerator

Topics

- Systolic Array
- Popular CNN Accelerator Design
- Transformer Accelerator

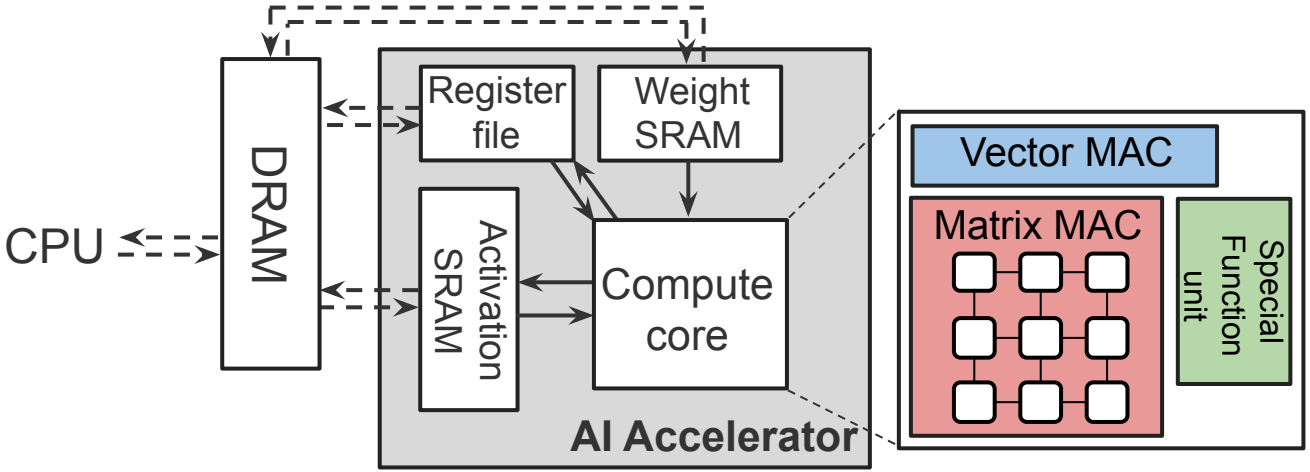
AI Accelerator

- The AI accelerator can execute part of the machine code that is related to the AI workload.



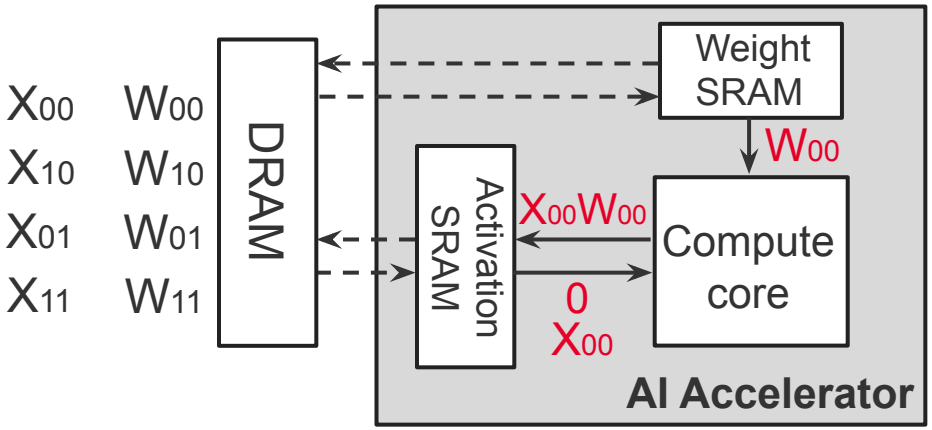
NYU SAI LAB

AI Accelerator



- The compute core consists of Multiply and accumulator (MAC) engine for 2D matrix multiplication.
- It also contains vector multiplier MAC as well as special function unit.

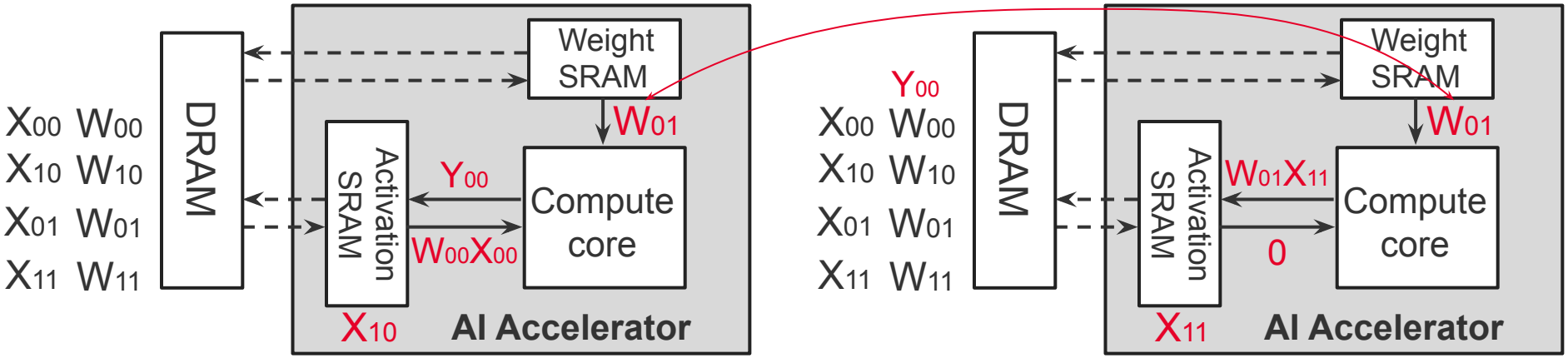
AI Accelerator



$$\begin{bmatrix} W_{00} & W_{01} \\ W_{10} & W_{11} \end{bmatrix} \times \begin{bmatrix} X_{00} & X_{01} \\ X_{10} & X_{11} \end{bmatrix} = \begin{bmatrix} W_{00}X_{00}+W_{01}X_{10} & W_{00}X_{01}+W_{01}X_{11} \\ W_{10}X_{00}+W_{11}X_{10} & W_{10}X_{01}+W_{11}X_{11} \end{bmatrix} = \begin{bmatrix} Y_{00} & Y_{01} \\ Y_{10} & Y_{11} \end{bmatrix}$$

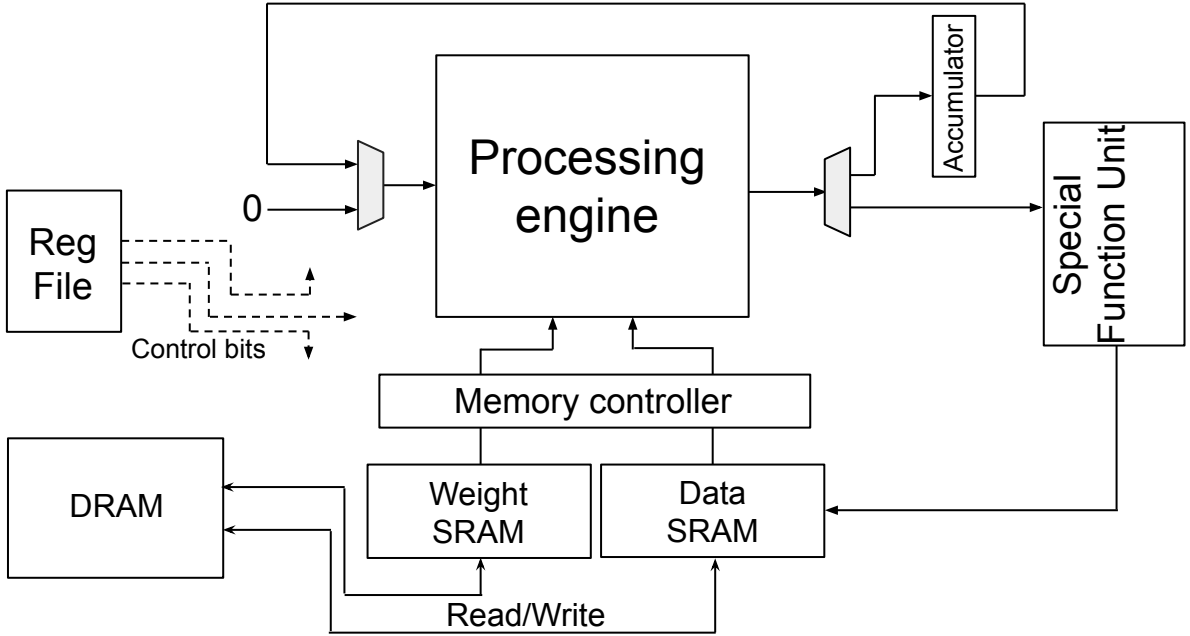
NYU SAI LAB

Memory Access Reduction



- The computation and memory access pattern can be changed to minimize the computational cost without impacting the results.

Hardware Architectures for DNN Processing

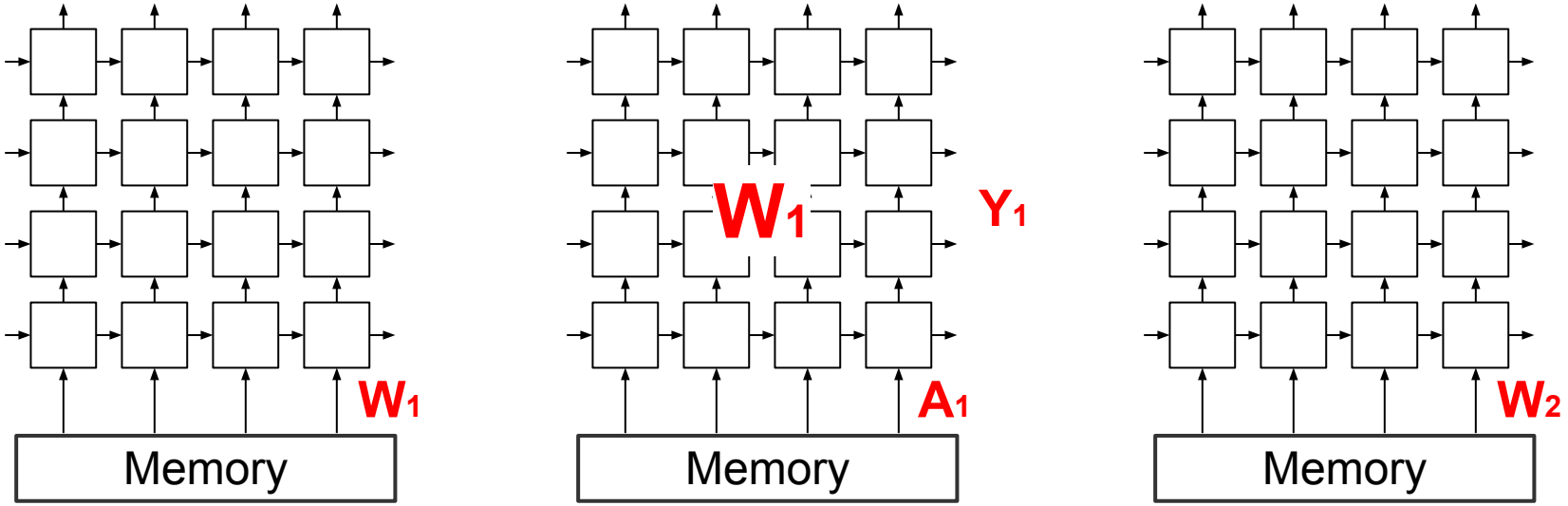


Major building blocks:

- Processing engine
- Accumulator
- Reg file
- Special function unit
- Memory subsystem
  - Weight SRAM
  - Data SRAM
  - DRAM

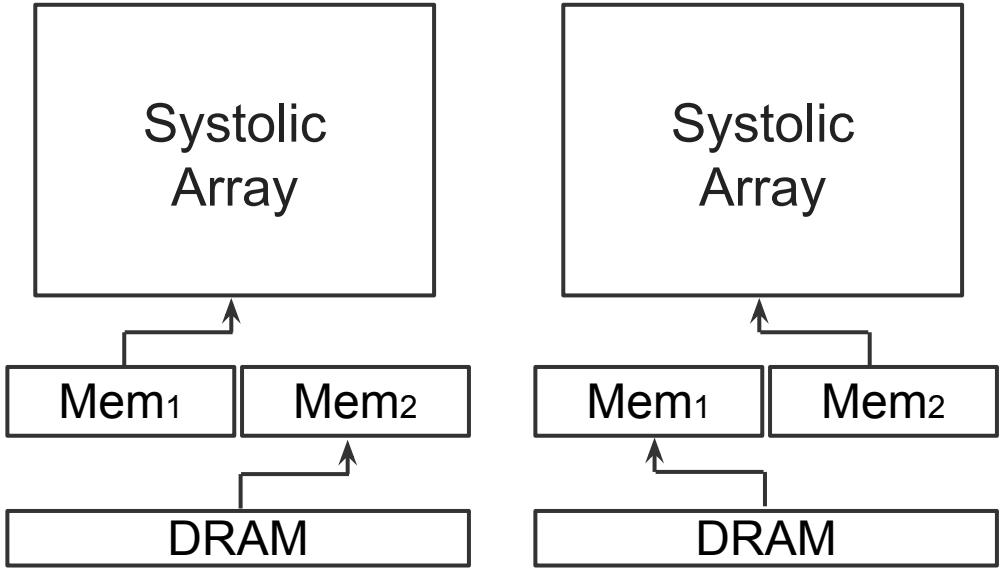
NYU SAI LAB

Computing Paradigms



- Spatial architecture can achieve great reuse of the extracted content, leading to a reduced memory access cost.

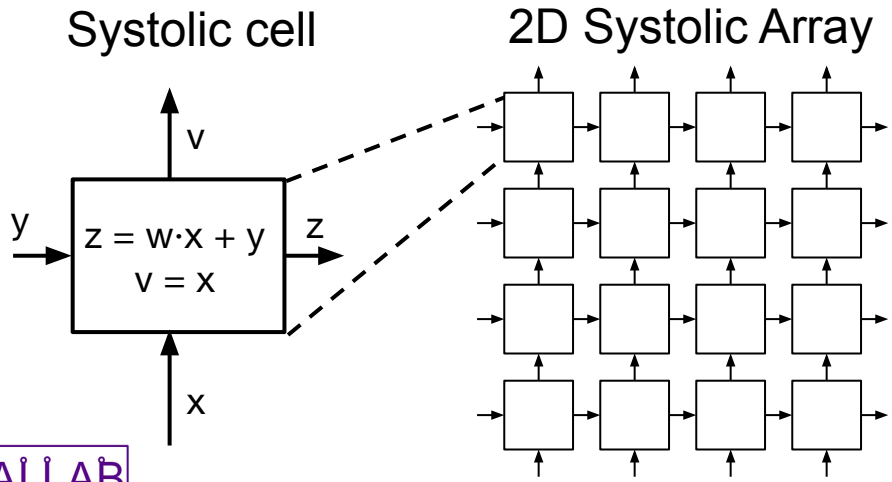
Double Buffering



- Double buffering in hardware design is a technique used to improve the efficiency and performance of data processing, especially in systems that require smooth and continuous data transfer.
- The idea is to overlap the data production and consumption processes to avoid delays.

Systolic Array (Weight Stationary Version)

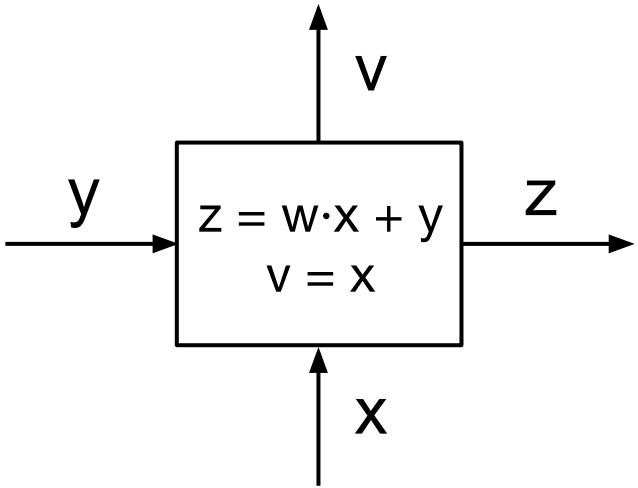
- Kung and Leiserson, "Systolic Arrays for VLSI," 1978 and Kung, "Why systolic architectures?' 1982
- 2D grid of multiplier-accumulators (MACs) for matrix multiplication
- Used by Google TPU for deep learning (2017), etc



TPU (Google)

NYU SAI LAB

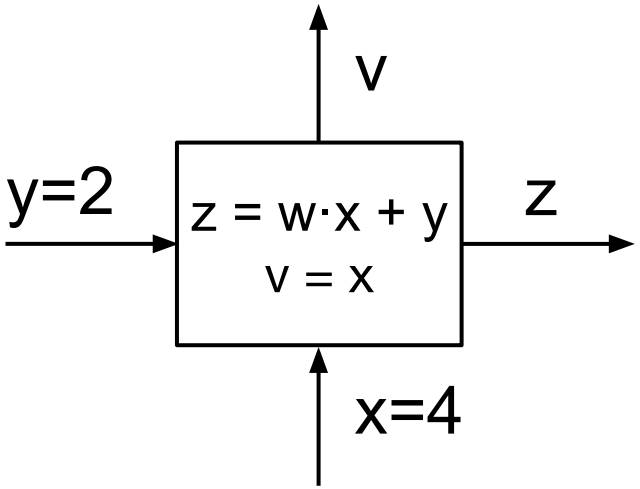
Systolic Cell



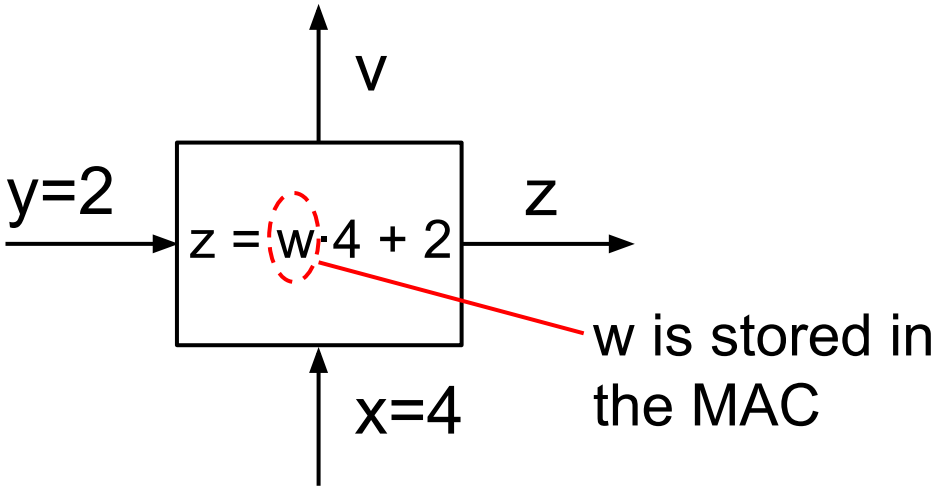
- Takes data (x and y) as input
- w stays in the systolic cell
- Performs a multiply-accumulate operation

NYU SAI LAB

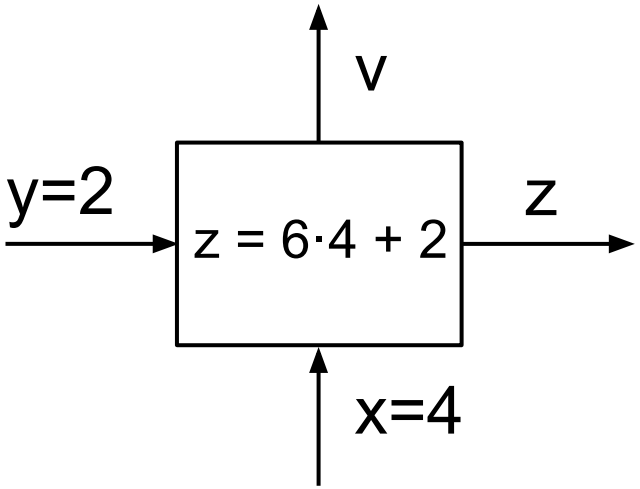
Systolic Cell



Systolic Cell

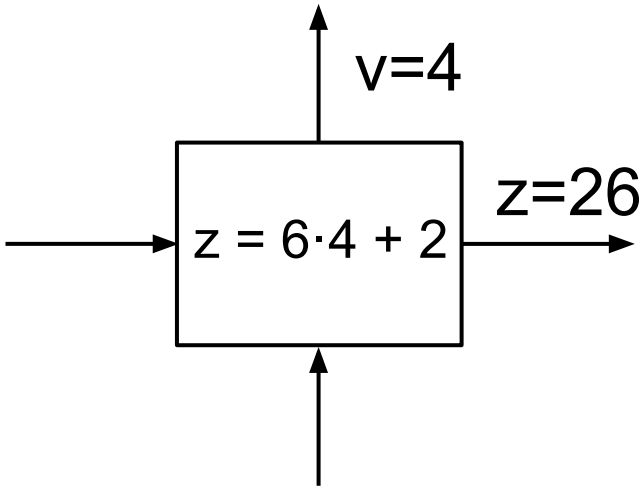


Systolic Cell



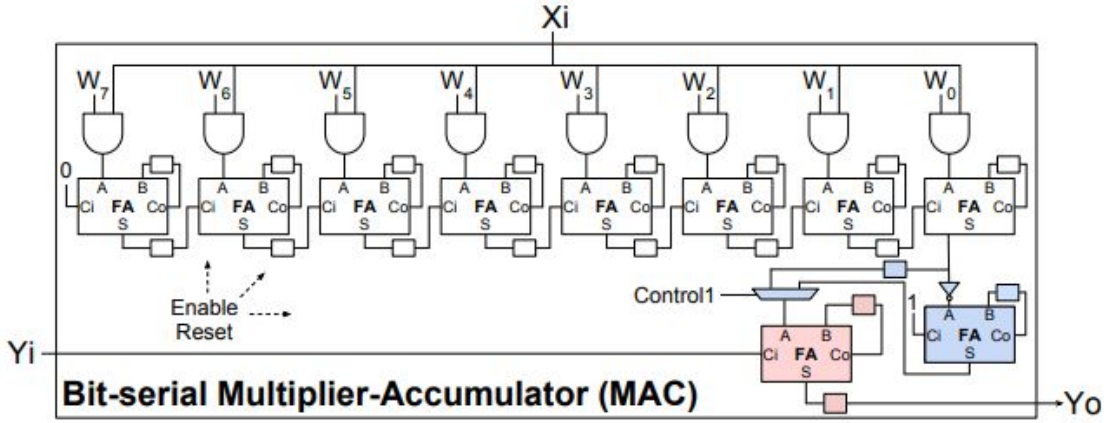
NYU SAI LAB

Systolic Cell



NYU SAI LAB

Bit-serial Multiplier Design



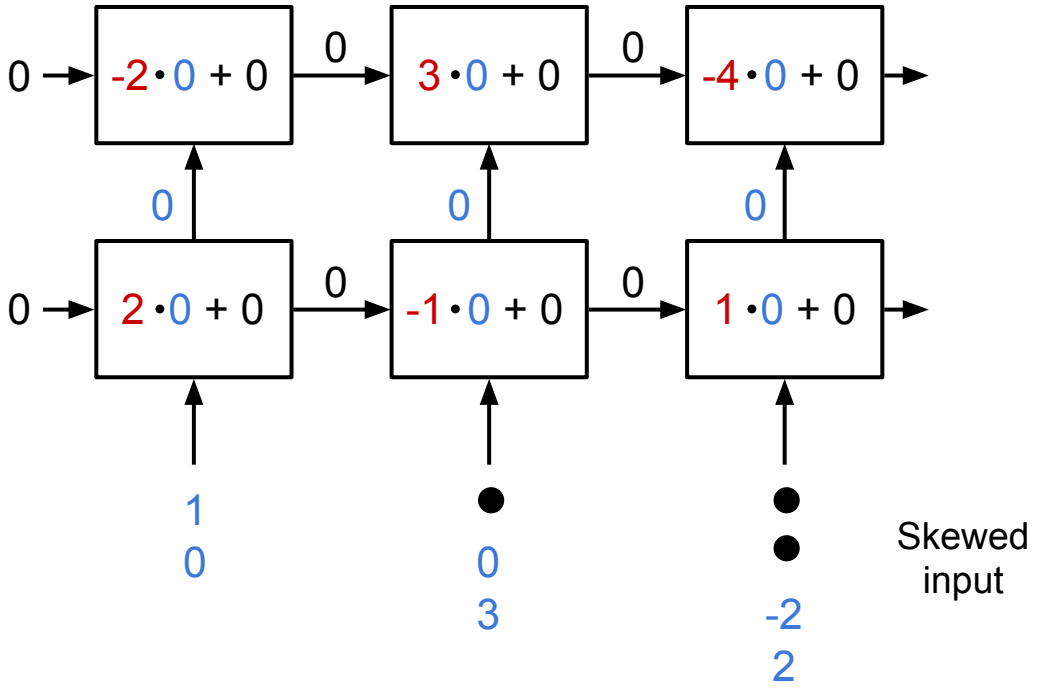
NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

$$\begin{bmatrix} 2 & -1 & 1 \\ -2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 1 \end{bmatrix}$$

Weights in red are preloaded into the systolic array

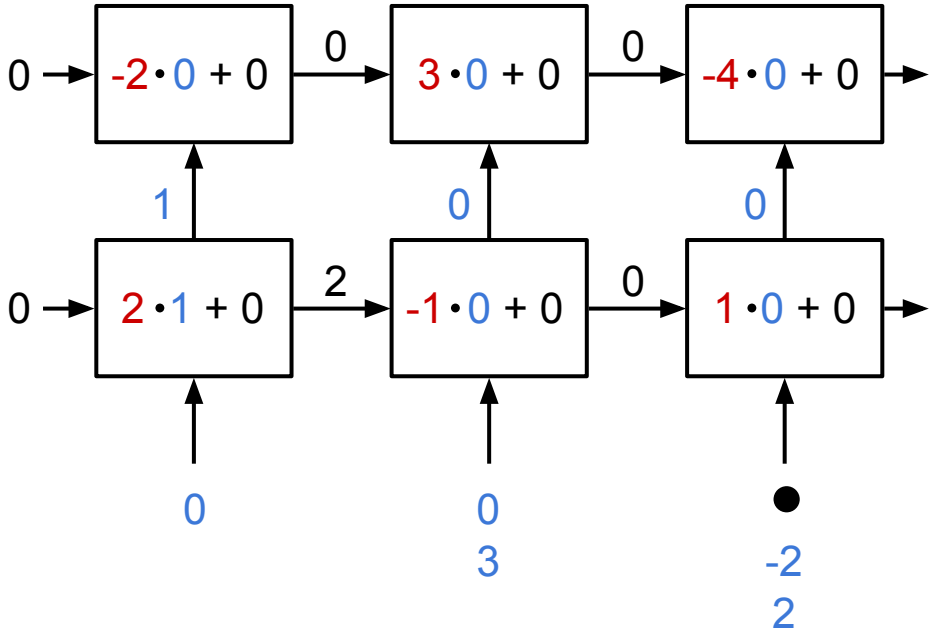


Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

$$\begin{bmatrix} 2 & -1 & 1 \\ -2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 1 \end{bmatrix}$$

Weights in red are preloaded into the systolic array

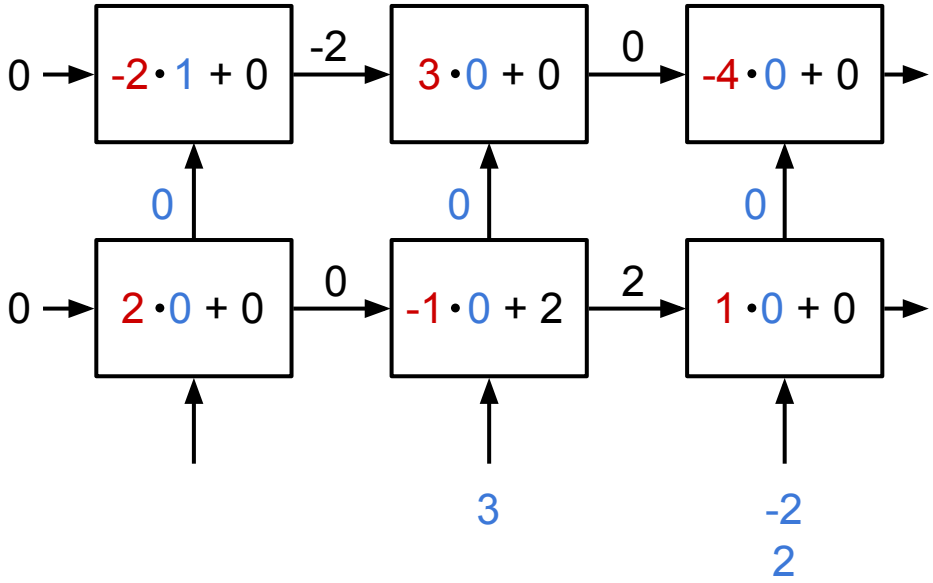


NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

$$\begin{bmatrix} 2 & -1 & 1 \\ -2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 1 \end{bmatrix}$$



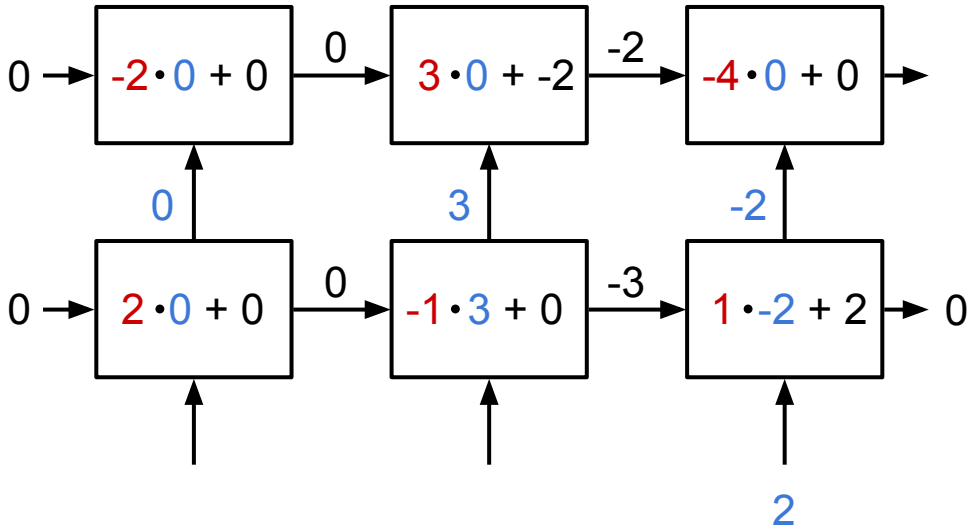
Weights in red are preloaded into the systolic array

NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

$$\begin{bmatrix} 2 & -1 & 1 \\ -2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 1 \end{bmatrix}$$



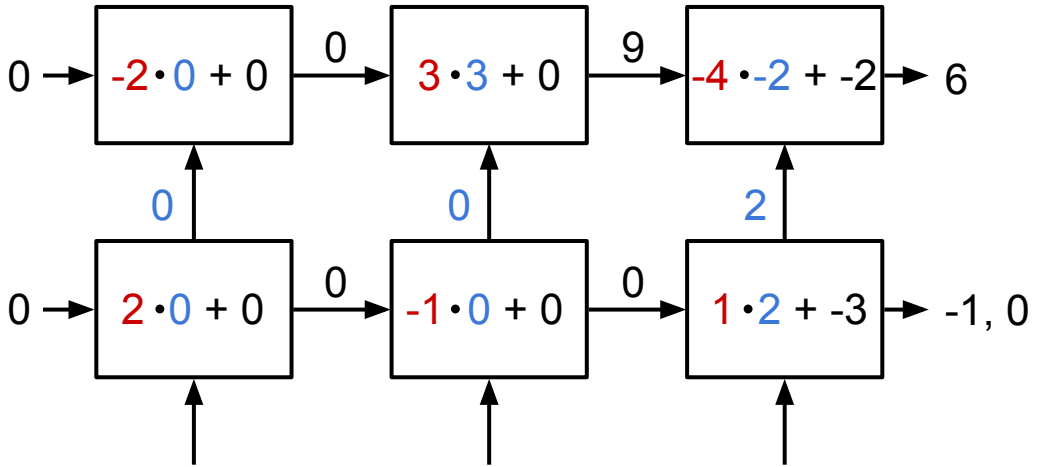
Weights in red are preloaded into the systolic array

NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

$$\begin{bmatrix} 2 & -1 & 1 \\ -2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 1 \end{bmatrix}$$



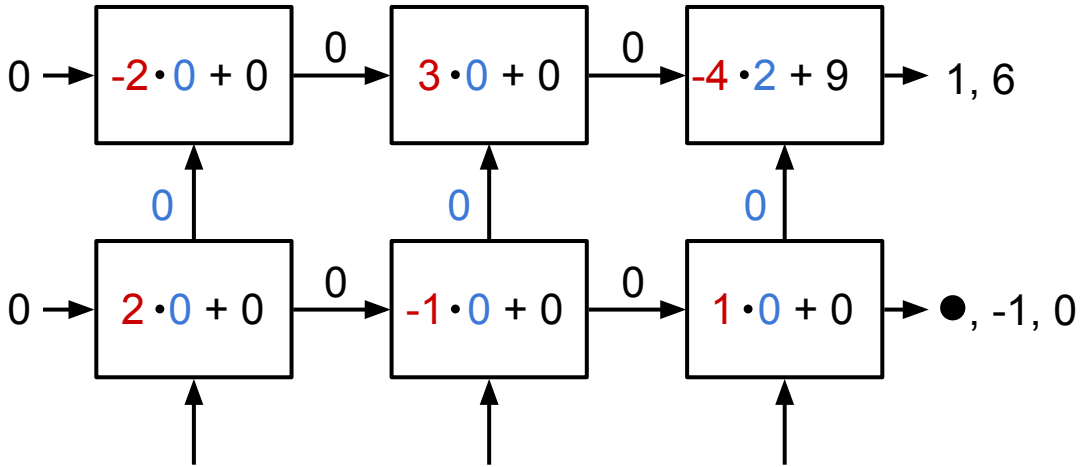
Weights in red are preloaded into the systolic array

NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

$$\begin{bmatrix} 2 & -1 & 1 \\ -2 & 3 & -4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 6 & 1 \end{bmatrix}$$



Weights in red are preloaded into the systolic array

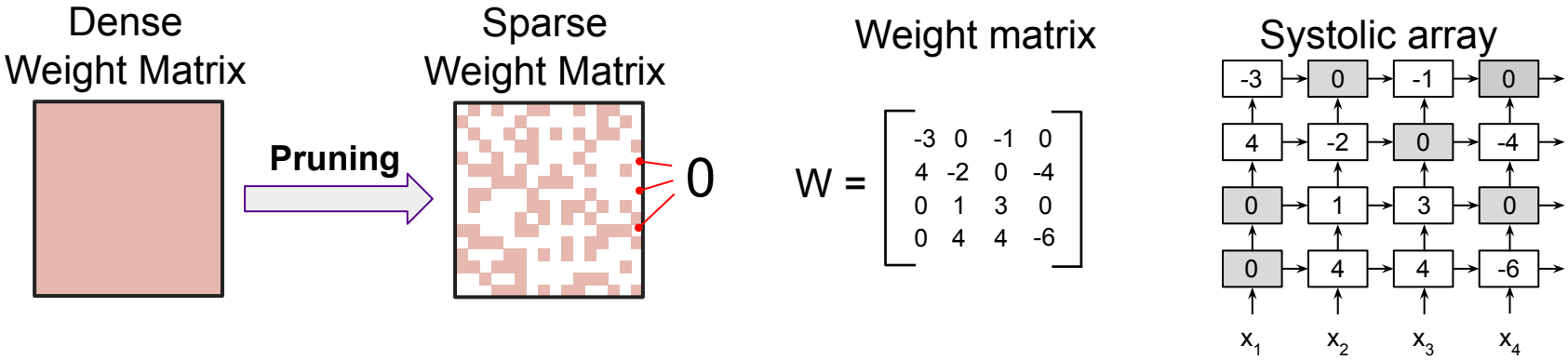
NYU SAI LAB

Topics

- Systolic Array
- Popular CNN Accelerator Design
- Transformer Accelerator

Column Combining

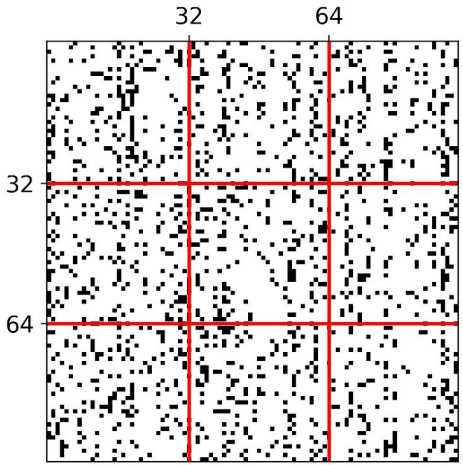
- Unimportant (i.e., small weights) are set to 0
- However, it is hard to leverage these zero weights to reduce the hardware cost



Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

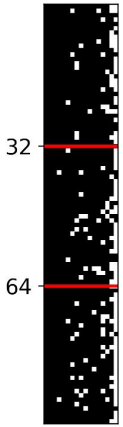
Column Combining

Sparse
Weight Matrix



Column Combining
8x reduction in size

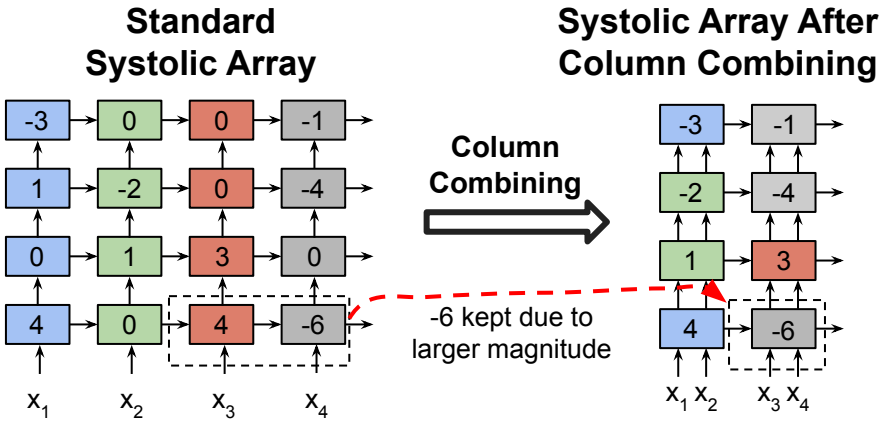
Packed Format in
Systolic Array

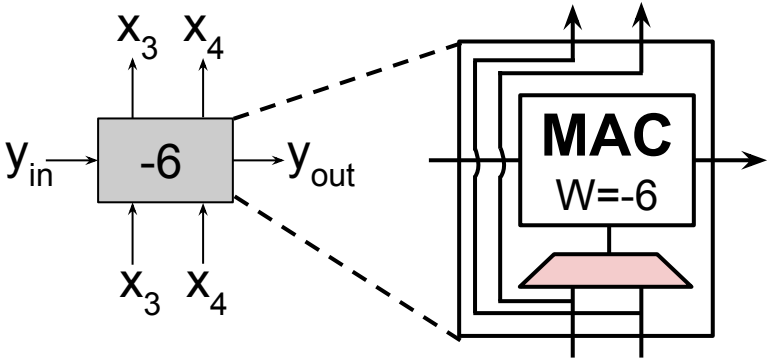


NYU SAI LAB

Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Column Combining

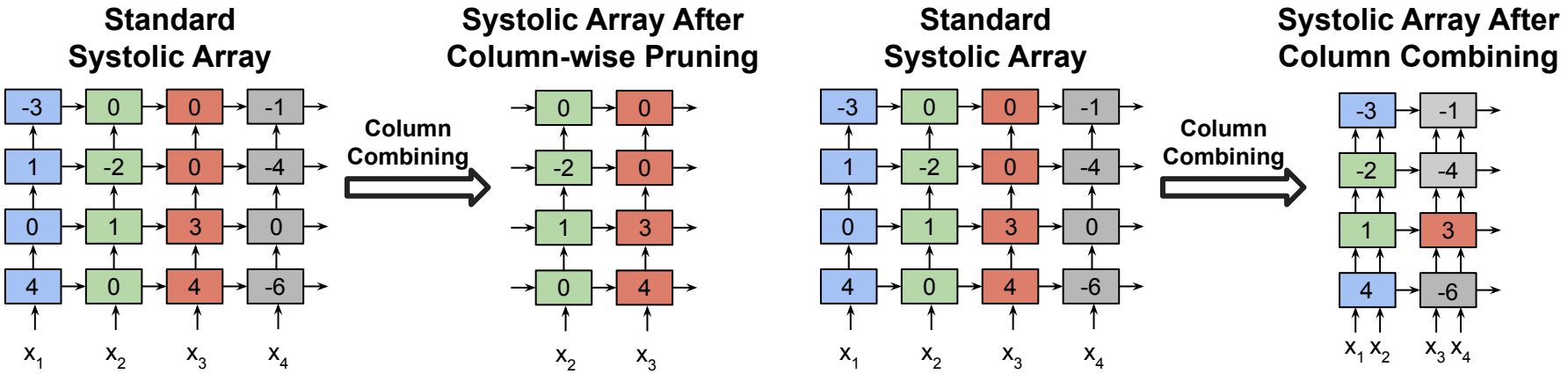




- Column combining can greatly increase the utilization of the systolic array.

Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Column Combining

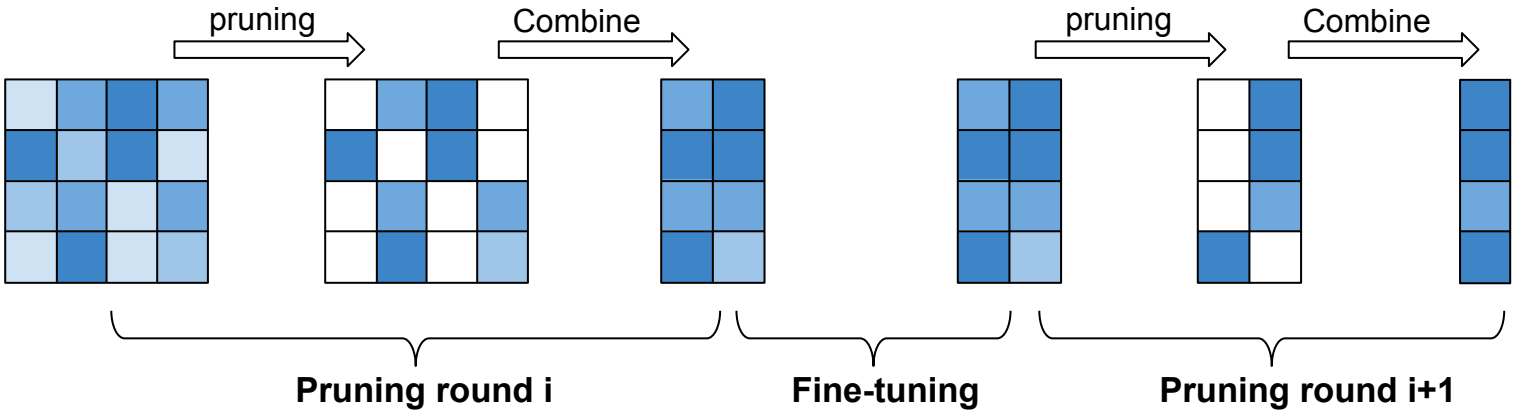


- Compared with column-wise Pruning (filterwise pruning), Column Combining allows for a much more flexible pruning pattern
- We can also apply Column Combining pruning on the input of each layer.

NYU SAI LAB

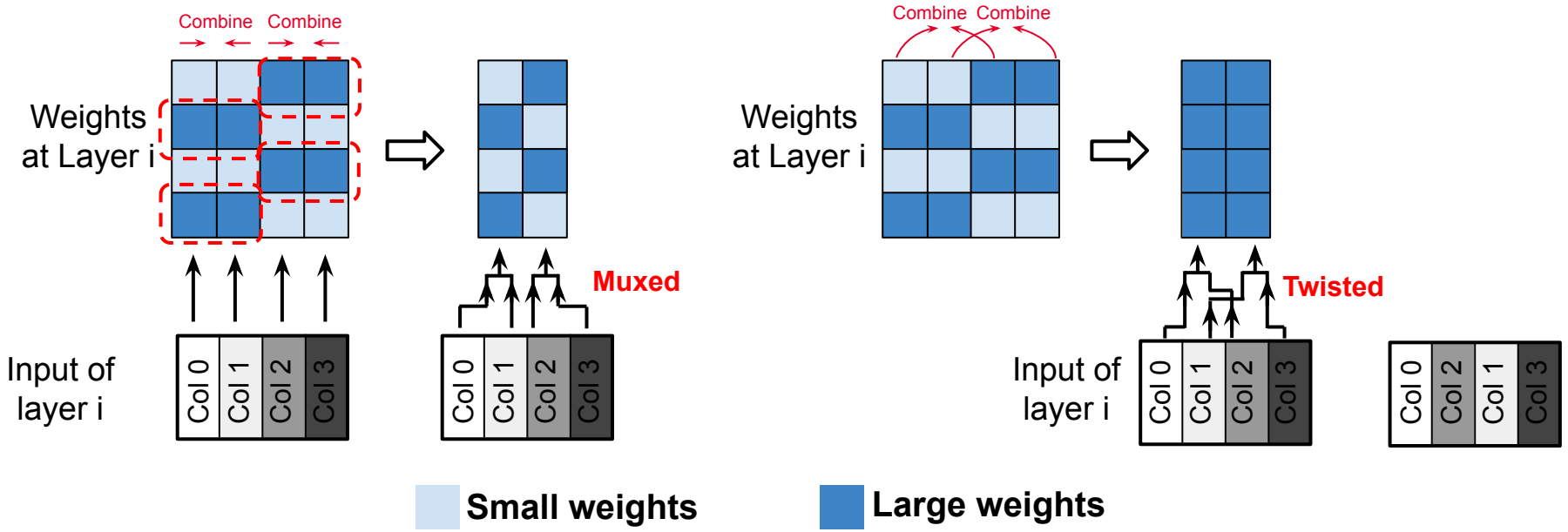
Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Column-Combining Pruning



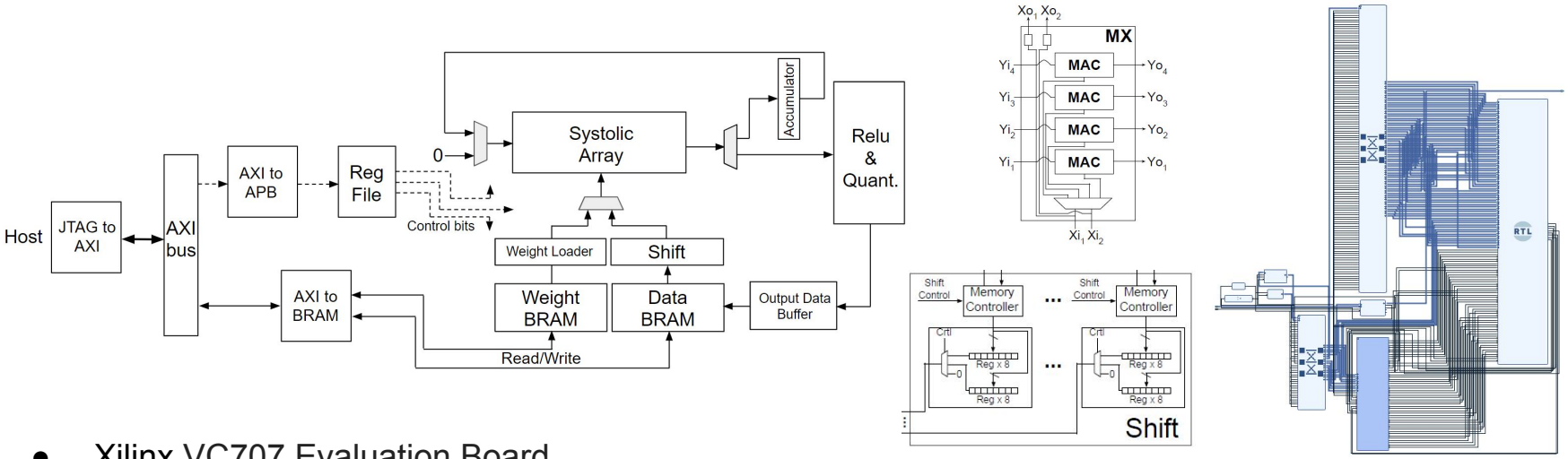
Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Column-Combining Pruning: Row Permutation



Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Hardware Implementation

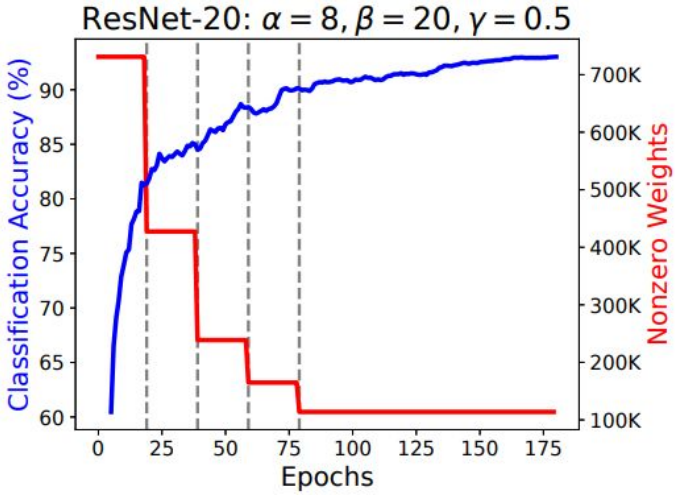


- Xilinx VC707 Evaluation Board
- Total hardware available: Lookup Table (303600), Flip-Flops (607200), BRAM (1030, each 36Kb)
- More than 15K lines of verilog code
- 128 by 64 systolic array

NYU SAI LAB

Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Accuracy Evaluation Results



ResNet-20 on CIFAR-10

	Accuracy
Original DNN	72.08%
Structured Filterwise Pruning	69.0%
Column Combining	71.81%

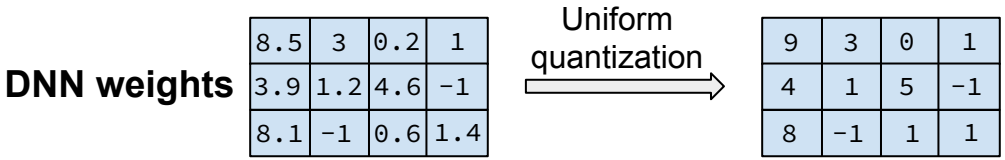
VGG-19 on ImageNet (87.5% sparsity)

NYU SAI LAB

Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

Prior Work: Efficient DNN Data Types

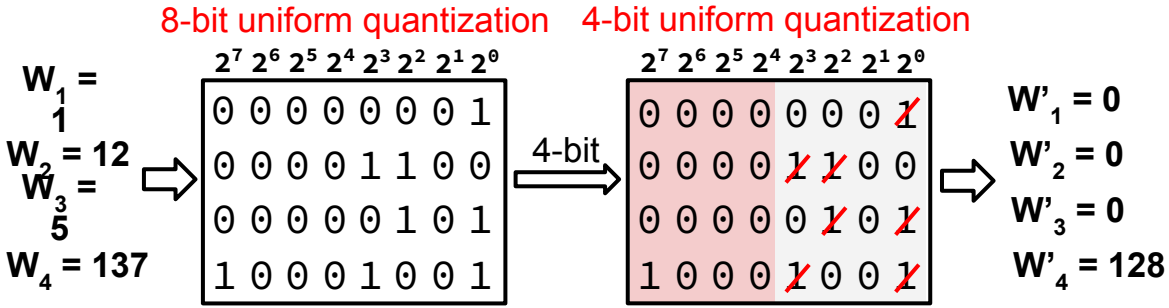
- Filter weights and activations can be quantized with low precision to accelerate the inference and reduce the model size.



- Low-precision quantization leads to large accuracy loss.

Li, Fengfu, Bo Zhang, and Bin Liu. "Ternary weight networks."
Rastegari, Mohammad, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks."
Zhou, Aojun, et al. "Incremental network quantization."

Problem on Low Precision Uniform Quantization



- Low-precision quantization leads to significant quantization error.
- Both weights and input activation are highly biased in values.

Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Term quantization: Furthering quantization at run time." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

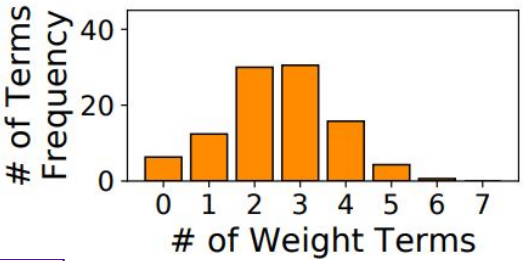
Representing Values in Power-of-two Terms

- An integer value can be represented as a summation of power-of-two term(s).

8-bit uniform quantization

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
$W_1 = 1$	0	0	0	0	0	0	0	1	2^0
$W_2 = 12$	0	0	0	0	1	1	0	0	2^3+2^2
$W_3 = 5$	0	0	0	0	0	1	0	1	2^2+2^0
$W_4 = 137$	1	0	0	0	1	0	0	1	$2^7+2^3+2^0$

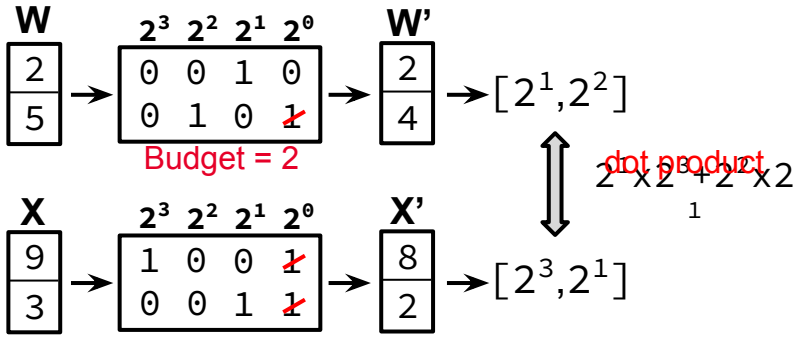
$$W_2X_2 = 12 \times 2 = (2^3+2^2) \times 2^1 = 2^3 \times 2^1 + 2^2 \times 2^1$$



- Most quantized weight and data values can be represented with 2 or 3 power-of-two terms.

Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Term quantization: Furthering quantization at run time." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

Term Quantization



4-bit uniform quantization

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

0	0	0	0	0	0	0	1
0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	1
1	0	0	0	1	0	0	1

$W'_1 = 0$

$W'_2 = 0$

$W'_3 = 0$

$W'_4 = 128$

TQ with a budget = 4

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

0	0	0	0	0	0	0	1
0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	1
1	0	0	0	1	0	0	1

$W'_1 = 0$

$W'_2 = 12$

$W'_3 = 0$

$W'_4 = 136$

- We can control the term-level computations by setting a **group term budget.**
- For a group of values, we rank and remove the small terms based on this budget.

Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Term quantization: Furthering quantization at run time." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

Hybrid Encoding for Shortened Expressions (HESE)

- To minimize the number of power-of-two terms in the binary input, we propose Hybrid Encoding for Shortened Expression (HESE)
- HESE offers:
  - Signed power-of-two expression with minimum length
  - Much less term-pair multiplications

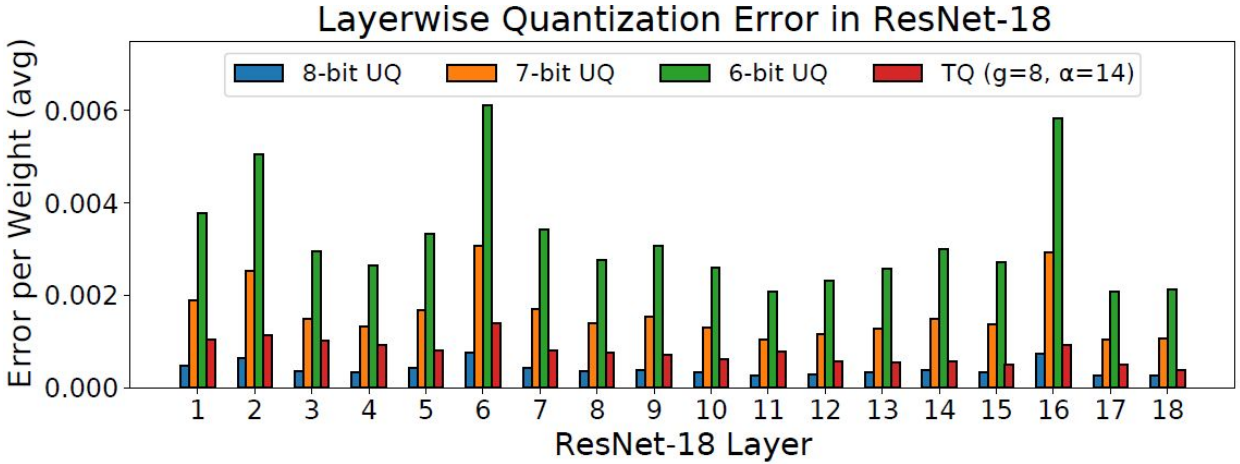
Binary expression	$31 = 2^4+2^3+2^2+2^1+2^0$	5 terms
	$27 = 2^4+2^3+2^1+2^0$	4 terms

HESE	$31 = 2^5-2^0$	2 terms
	$27 = 2^5-2^2-2^0$	3 terms

NYU SAI LAB

Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Term quantization: Furthering quantization at run time." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

Quantization Error Analysis



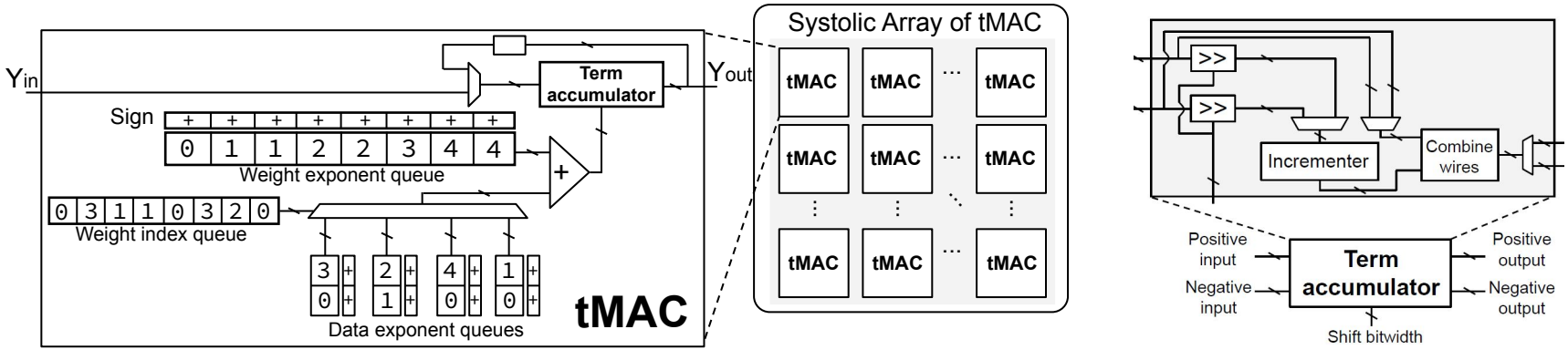
- We represent a group of 8 weights with 14 terms, each weight only requires 1.75 terms on average
- Term Quantization (TQ) introduces a small amount of quantization error over 8-bit uniform quantization (UQ)
- TQ achieves a much lower quantization error than 7-bit and 6-bit uniform quantization
- TQ with 1.75 term per value achieves a similar quantization error as 5-bit UQ

NYU SAI LAB

Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Term quantization: Furthering quantization at run time." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

Multiplier-accumulator Design

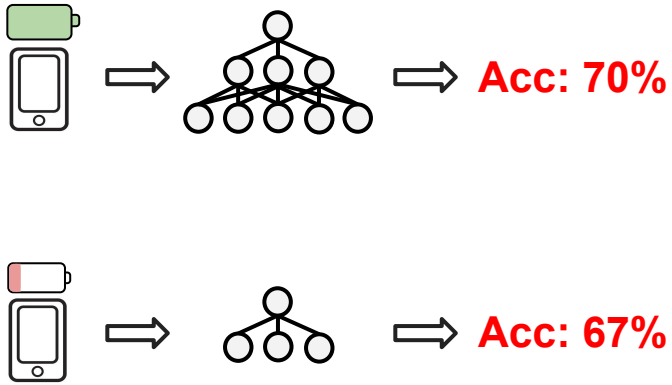
- We propose the term MAC (tMAC) for the efficient implementation of TQ.
- A tMAC processes all term-pair multiplications across a group of weight and data values.

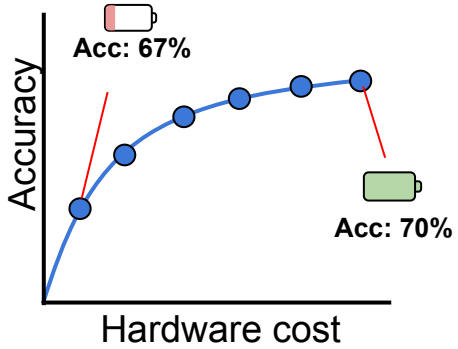


- Each term is represented by their corresponding exponent (2-3 bits).
- The term accumulation can be implemented using half adders.

Kung, Hsiang-Tsung, Bradley McDanel, and Sai Qian Zhang. "Term quantization: Furthering quantization at run time." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.

Multi-resolution DNN Inference with TQ

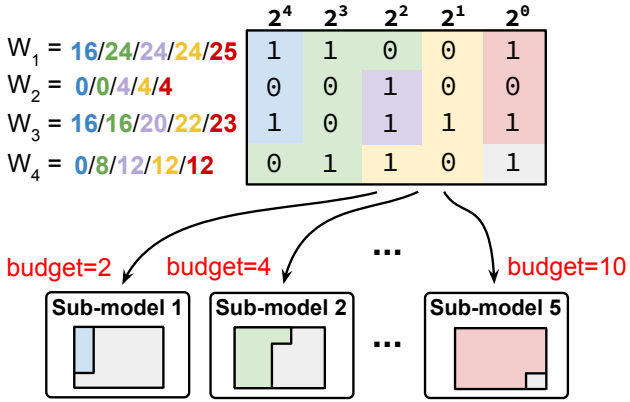




- DNN is expected to run at different resolution to achieve a good trade-off between hardware cost and accuracy.

Zhang, Sai Qian, et al. "Training for multi-resolution inference using reusable quantization terms." *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021.

Multi-resolution DNN Inference with TQ

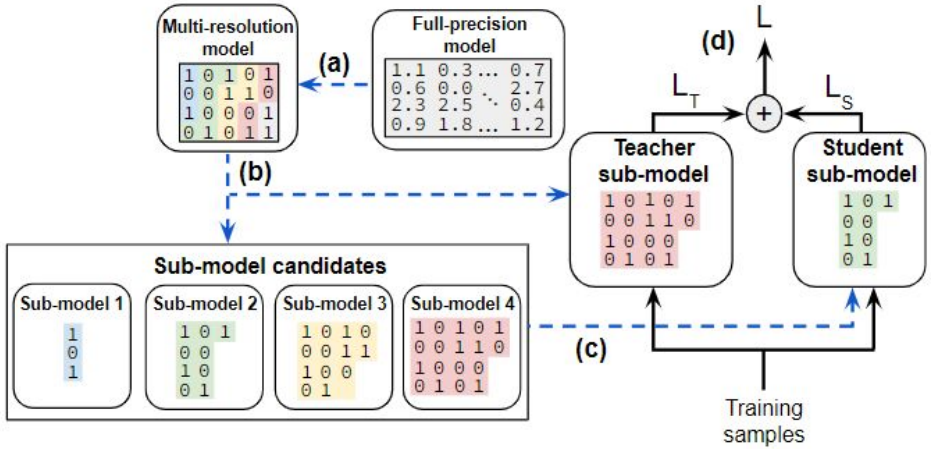


Multi-resolution DNN model

- A meta multi-resolution DNN model which can work under different term budget needs to be trained.

Zhang, Sai Qian, et al. "Training for multi-resolution inference using reusable quantization terms." *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021.

Multi-resolution DNN Training

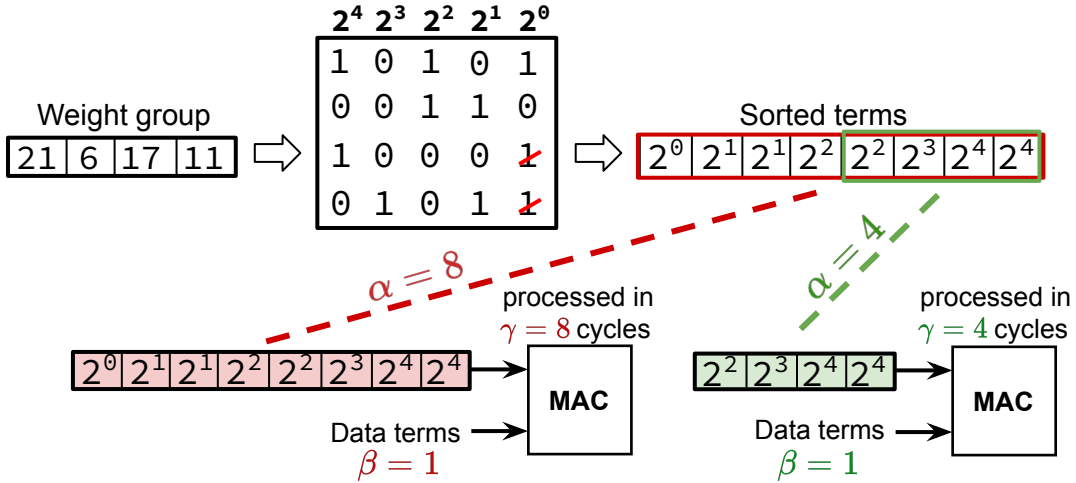


- We develop a multi-resolution training scheme to jointly train multiple DNN models under different term budgets.
- Instead of jointly train all the sub-models together, we apply the knowledge distillation framework to jointly train two sub-models per iteration.

NYU SAI LAB

Zhang, Sai Qian, et al. "Training for multi-resolution inference using reusable quantization terms." *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021.

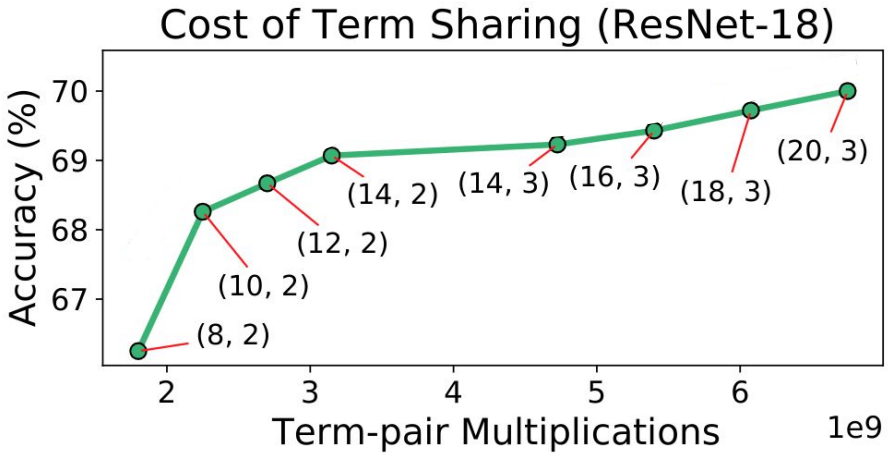
Multi-resolution DNN System

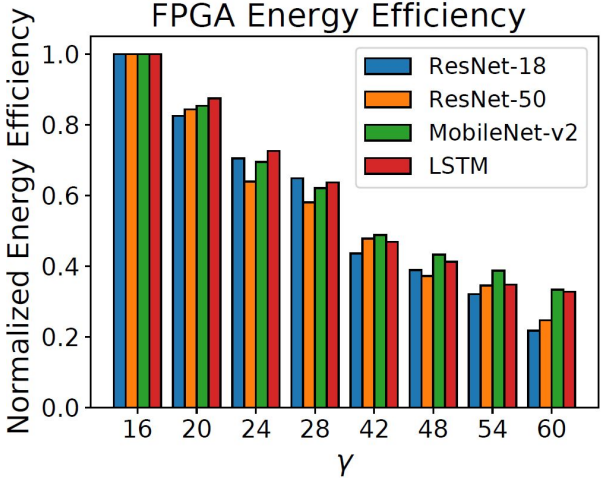


- The energy consumption will scale with the term budgets of the weight and data.

Zhang, Sai Qian, et al. "Training for multi-resolution inference using reusable quantization terms." *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021.

Evaluation: Accuracy Performance



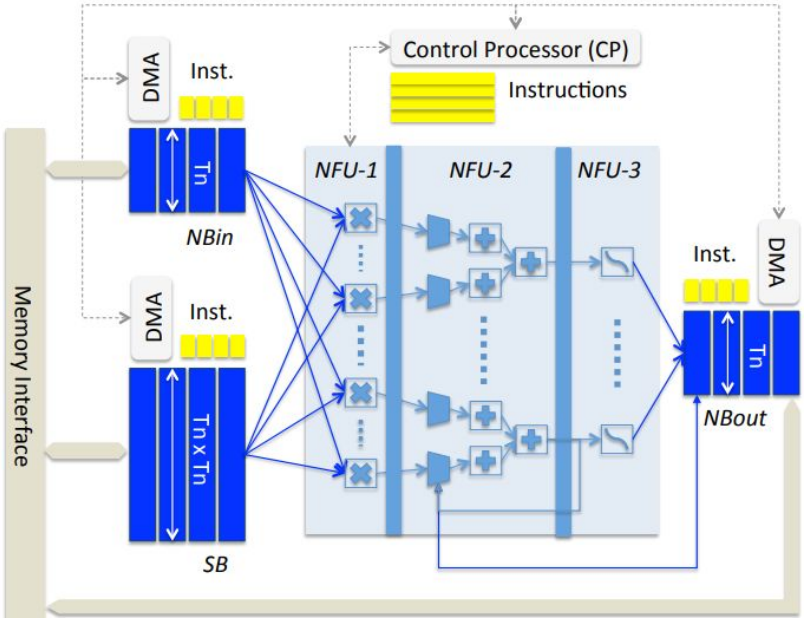


- The multi-resolution DNN incurs 0.4%-3.8% degradation compared to the original floating-point DNN (70.2%).
- The energy efficiency grows (3.25x on average), as term budget reduces from 60 to 16.

NYU SAI LAB

Zhang, Sai Qian, et al. "Training for multi-resolution inference using reusable quantization terms." *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021.

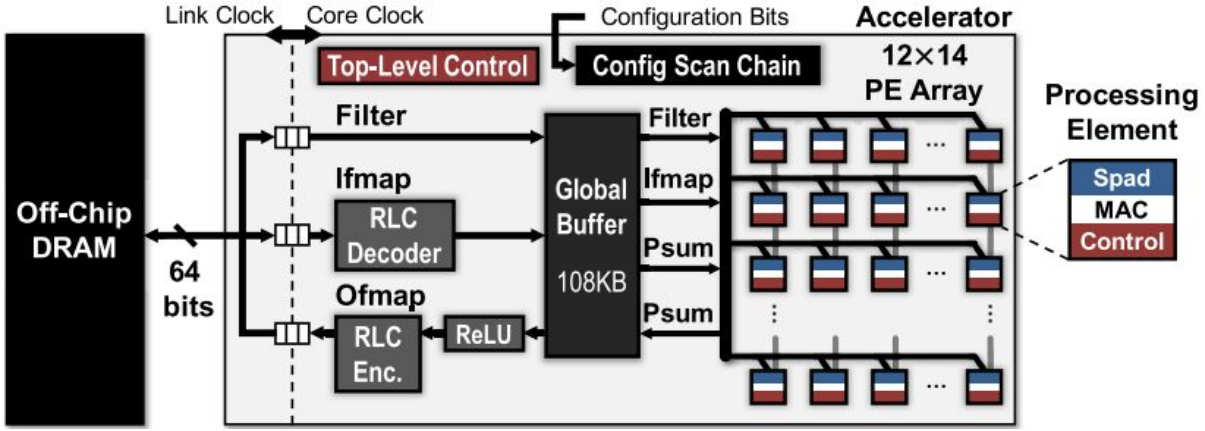
Diannao



- The first popular end-to-end DNN (CNN) accelerator.
- Diannao is synthesized with 65nm using Synopsys tools, achieving a throughput of 482 GOP/s.
- NFU consists of three stages:
  - Multiplier units
  - Adder tree
  - Nonlinear unit

Chen, Tianshi, et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning." *ACM SIGARCH Computer Architecture News* 42.1 (2014): 269-284.

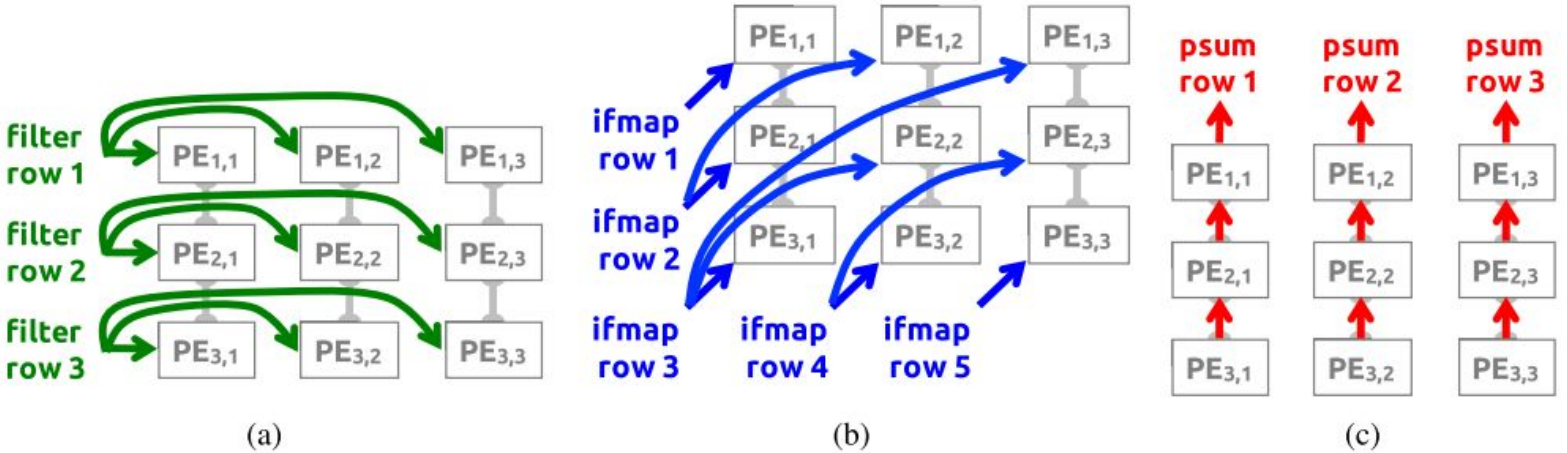
Eyeriss



- Eyeriss optimizes for the energy efficiency of the entire system, including the accelerator chip and off-chip DRAM, for various CNN shapes by reconfiguring the architecture.
- The core clock domain consists of a spatial array of 168 PEs organized as a 12 × 14 rectangle, a 108-kB GLB, an RLC CODEC, and an ReLU module.

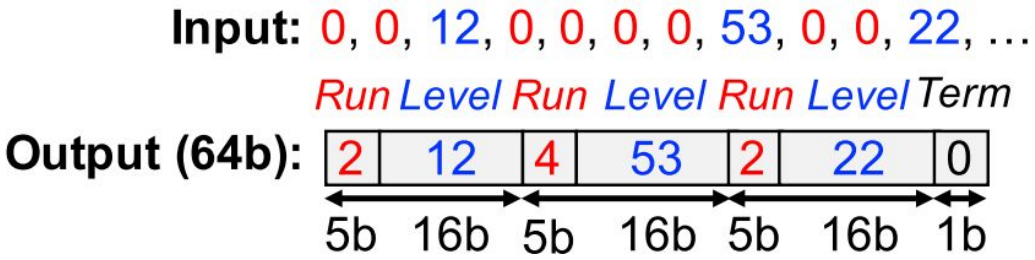
NYU SAI LAB

Data Reuse for Memory Access Reduction



- Reuse and accumulation of data within a PE set reduce accesses to the GLB and DRAM, saving data movement energy cost.

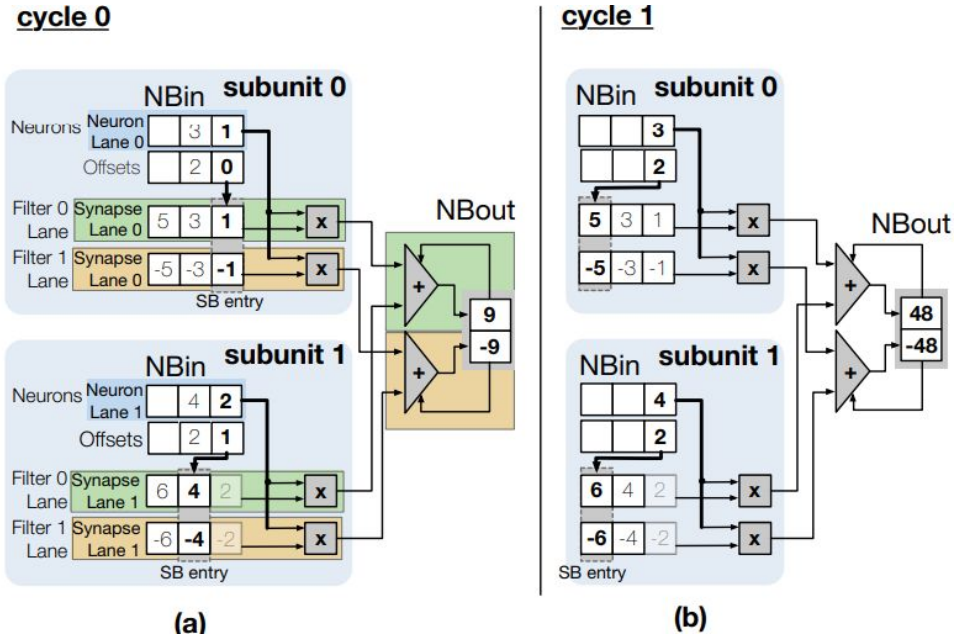
Rerun-length encoding



- RLC is used for compressing the input activation.

Cnvlutin

Input: [1, 0, 3] → [1, 3] (input) [0, 2] (offset)
Weight: [1, 3, 5]



- A large fraction of the computations performed by CNNs are intrinsically ineffectual as they involve a multiplication where one of the inputs is zero.
- Cnvlutin is a value-based approach to hardware acceleration that eliminates most of these ineffectual operations, improving performance and energy over a state-of-the-art accelerator with no accuracy loss.

NYU SAI LAB

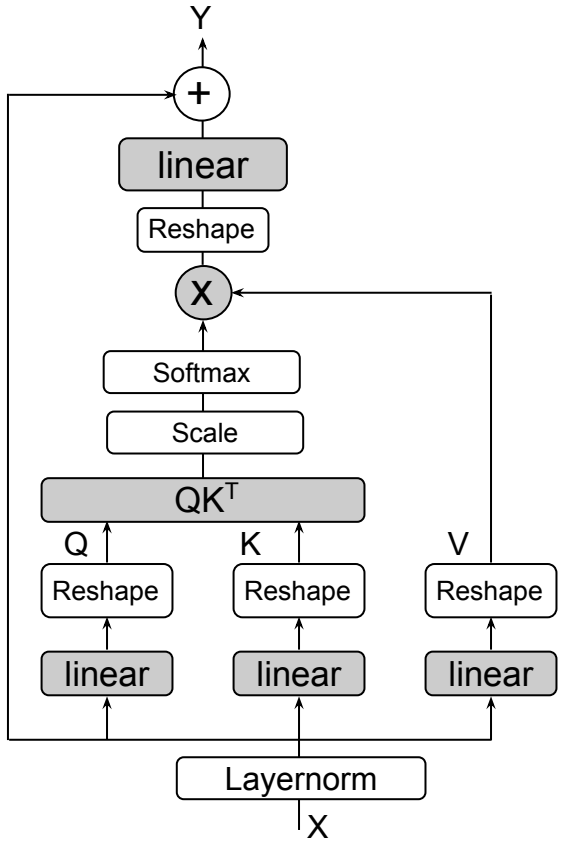
Albericio, Jorge, et al. "Cnvlutin: Ineffectual-neuron-free deep neural network computing." *ACM SIGARCH Computer Architecture News* 44.3 (2016): 1-13.

Topics

- Systolic Array
- Popular CNN Accelerator Design
- Transformer Accelerator

Self-Attention Block

- Given input x, the first step in calculating self-attention is to create three vectors from each of the input x', denoted as: Query (Q), Key (K), Value (V).
  - $(B,L,E) \times (E \times E) \rightarrow (B \times L \times E)$
- The second step in calculating self-attention. This will compute the attention score between each pair of input tokens.
  - $QK^T \rightarrow (B, L \times E) \times (B,E \times L) \rightarrow (B, L \times L)$
- Scale and normalize the score using softmax.
  - $\text{Softmax}(QK^T) \rightarrow (B, L \times L)$
- Multiply each value vector by the softmax score.
  - $\text{Softmax}(QK^T) \times V$
  - $(B, L \times L) \times (B, L \times E) \rightarrow (B, L \times E)$
- Pass the result to the linear layer, sum with the input.

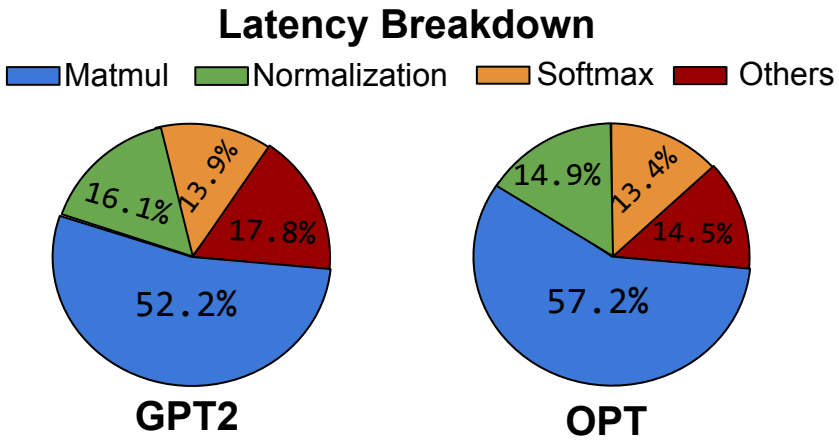


NYU SAI LAB

Operations Other than Multiplications

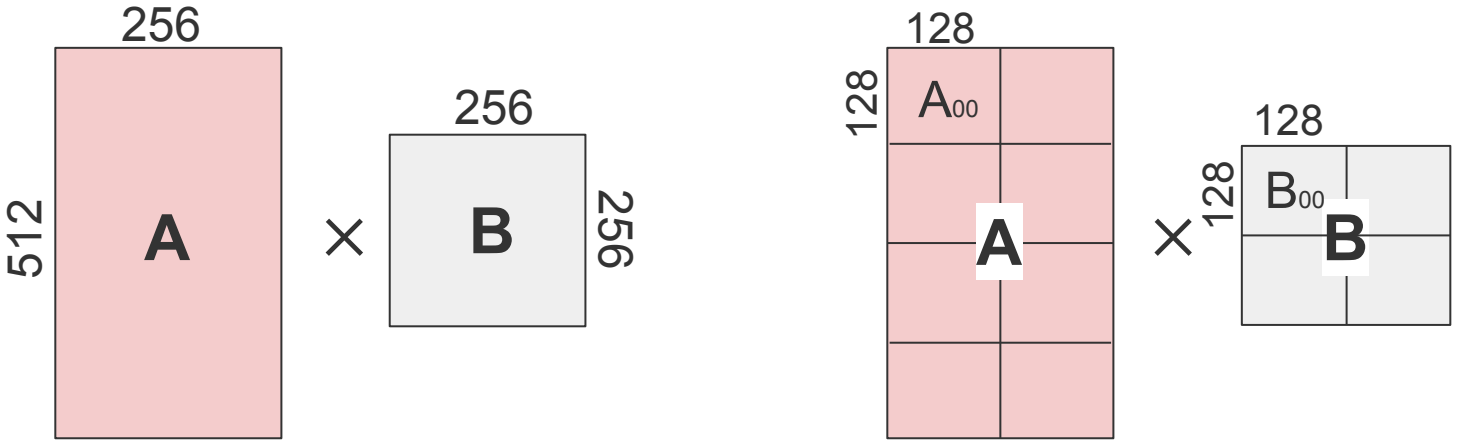
- Transposition
- Nonlinear operations
  - Softmax
  - LayerNorm
  - GeLU

Breakdown on Computational Cost



- Matmul still contributes to majority of the overall latency.
- Nonlinear operations are not negligible.
- Also other operations (e.g., transposition) also contributes to a great portion of the overall latency.

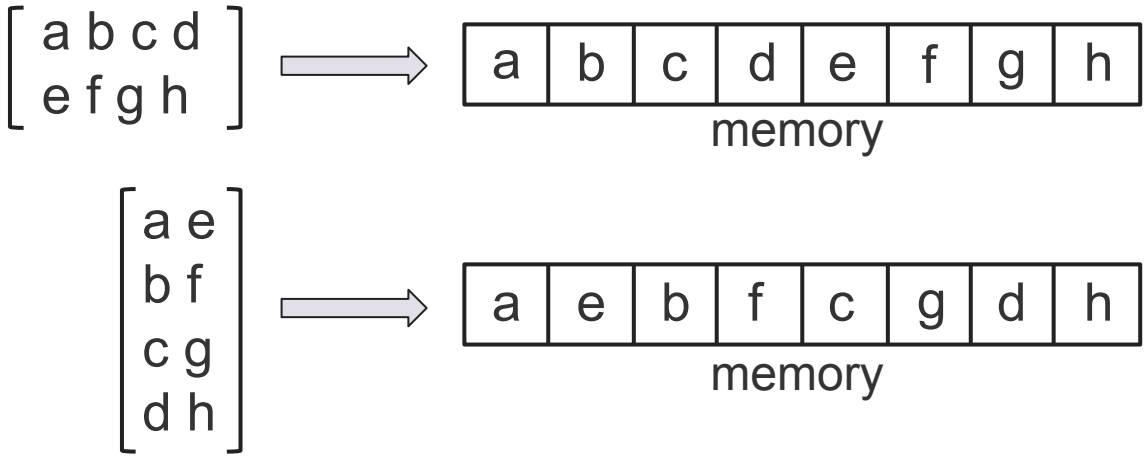
Matrix Multiplication



- The large matrix operands are first partitioned into tiles that can fit the size of the compute core.

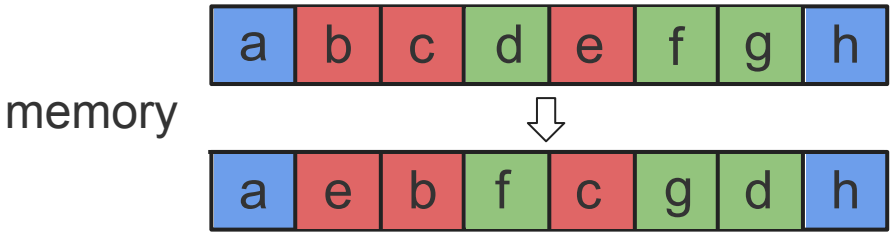
In-Place Matrix Transposition

- In-place matrix transposition refers to the process of transposing a matrix directly within its existing memory space, requiring only a minimal amount of extra storage.

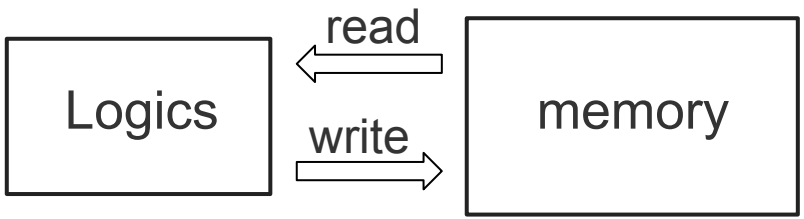


NYU SAI LAB

In-Place Matrix Transposition

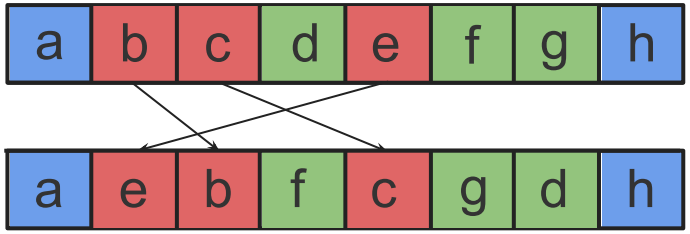


(b,c,e) → (e,b,c)
(d,f,g) → (f,g,d)

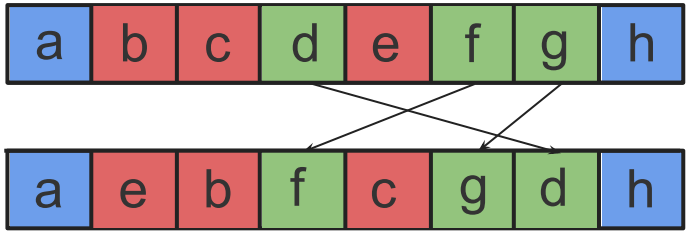


- Need to read multiple entries from the memory, permute them and write them back.
- This operation should be performed efficiently with minimal memory access cost.

In-Place Matrix Transposition



Step 1



Step 2

- The search for optimal swapping patterns that minimize permutations is a well-established problem in mathematics.

Norman Brenner, "Algorithm 467: matrix transposition in place," ACM Transactions on Mathematical Software 16 (11), p. 692-694 (1973). doi:10.1145/355611.362542

TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning
Ruturaj, Zihan, Canyu

SCNN : An Accelerator for Compressed-sparse Convolutional Neural Networks
Julio, Jiajing, Mohammed