Lecture 11:
AI Accelerator Introduction and CNN Accelerators

Notes

- First round of team meeting on Dec 1 and Dec 2.
- Extra-credit quiz today.

Recap

- Federated Learning
- Machine Learning Compiler
- Machine Learning System

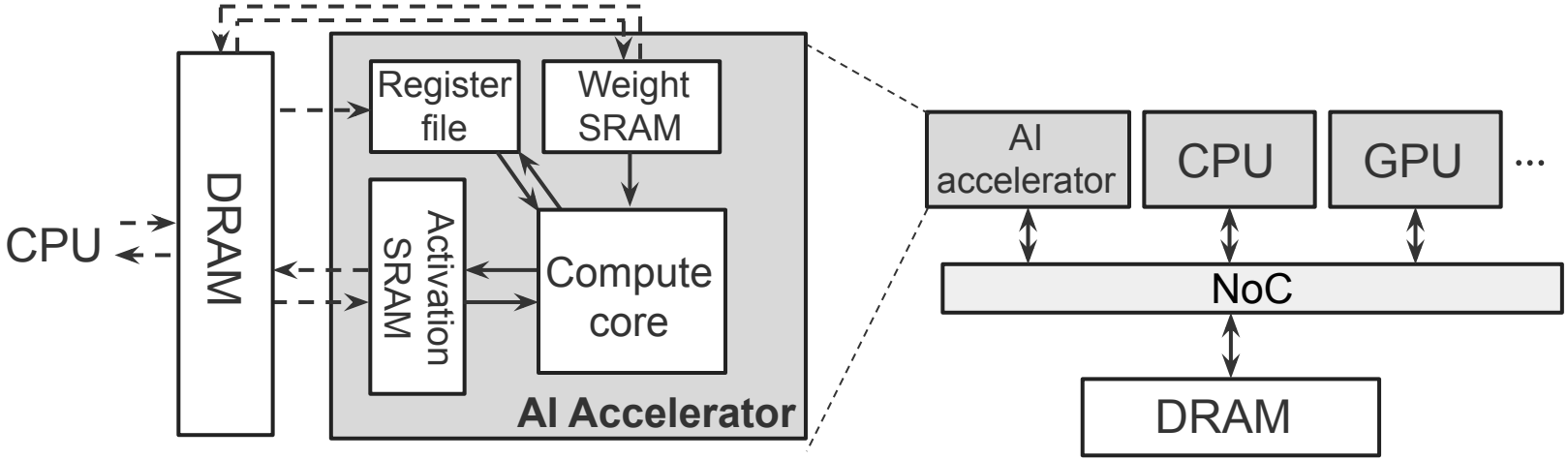
NYU SAI LAB

Topics

- Hardware accelerator: Overview
- Convolutional operation conversion
- Systolic array
- Convolutional Neural Network System
- Popular accelerator design

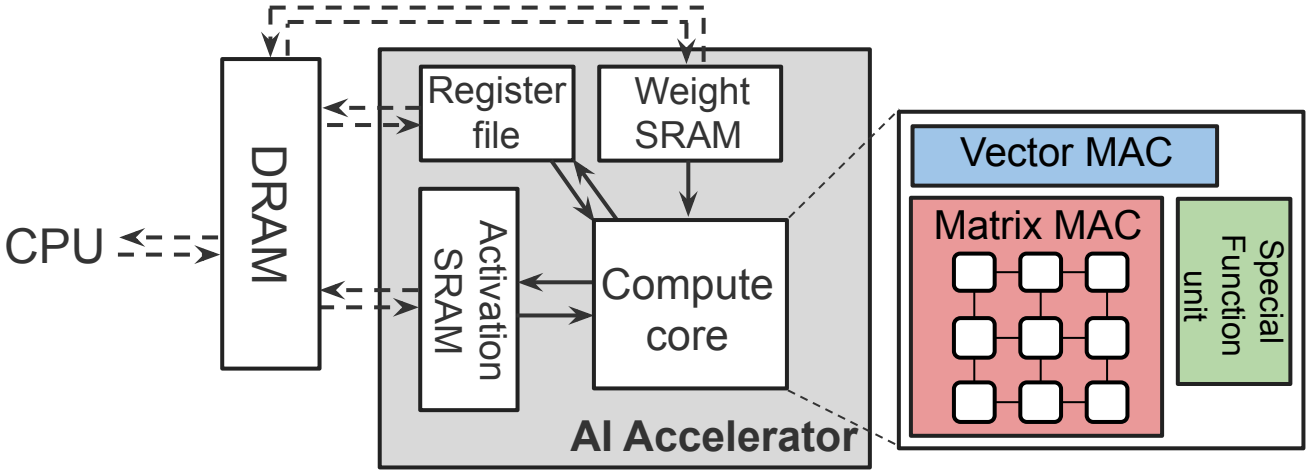
AI Accelerator

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



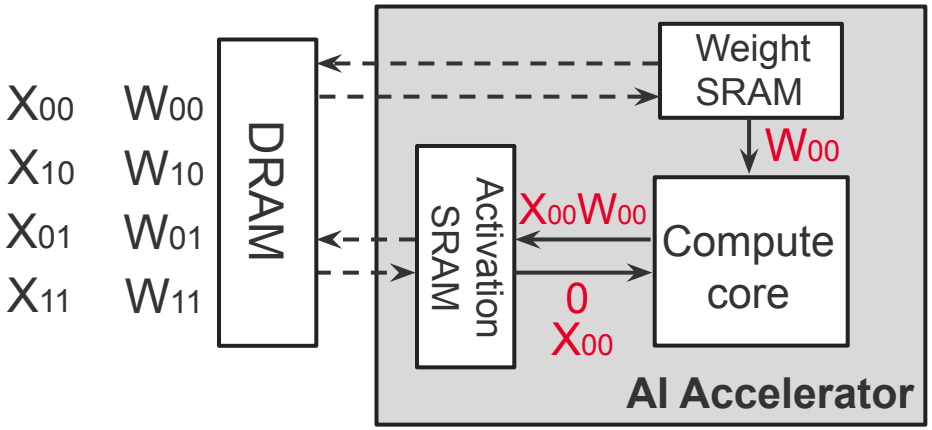
NYU SAI LAB

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.

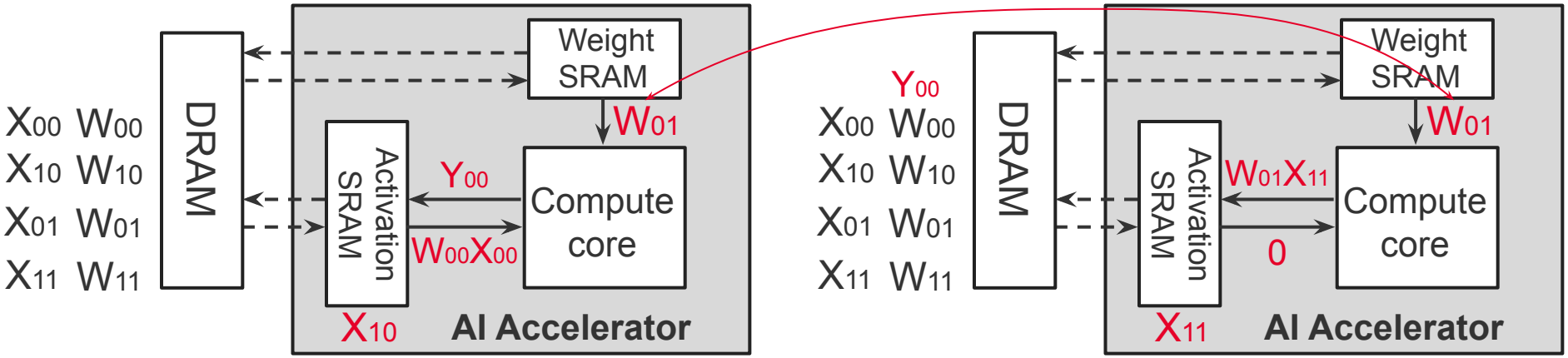
AI Accelerator



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

NYU SAI LAB

Memory Access Reduction

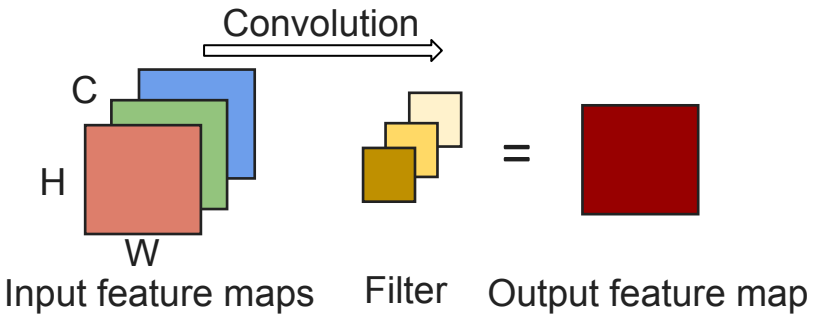


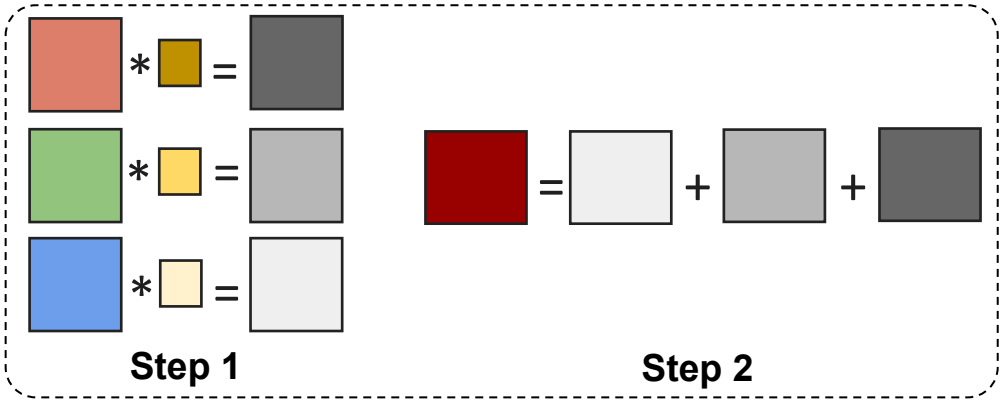
- The computation and memory access pattern can be changed to minimize the computational cost without impacting the results.

Topics

- Hardware accelerator: Overview
- Convolutional operation conversion
- Systolic array
- Convolutional Neural Network System
- Popular accelerator design

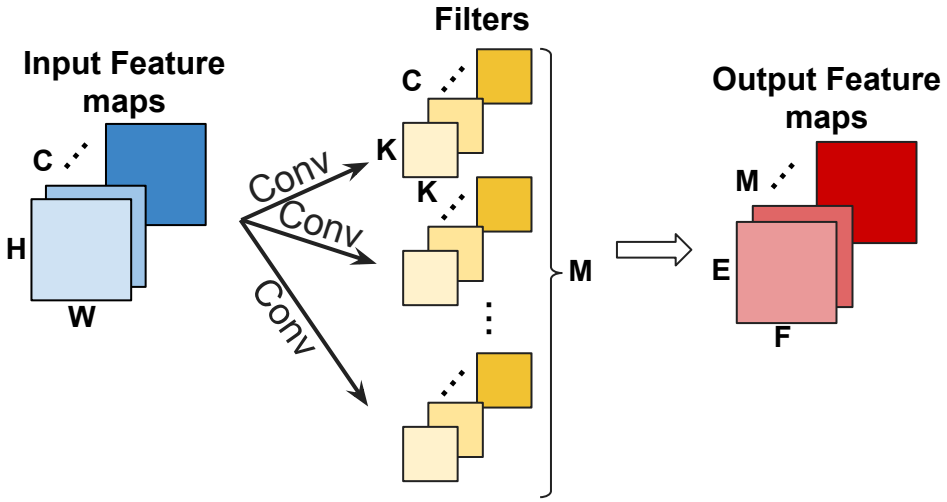
Convolutional Layers





- Core building block of a CNN, it is also the most computational intensive layer.

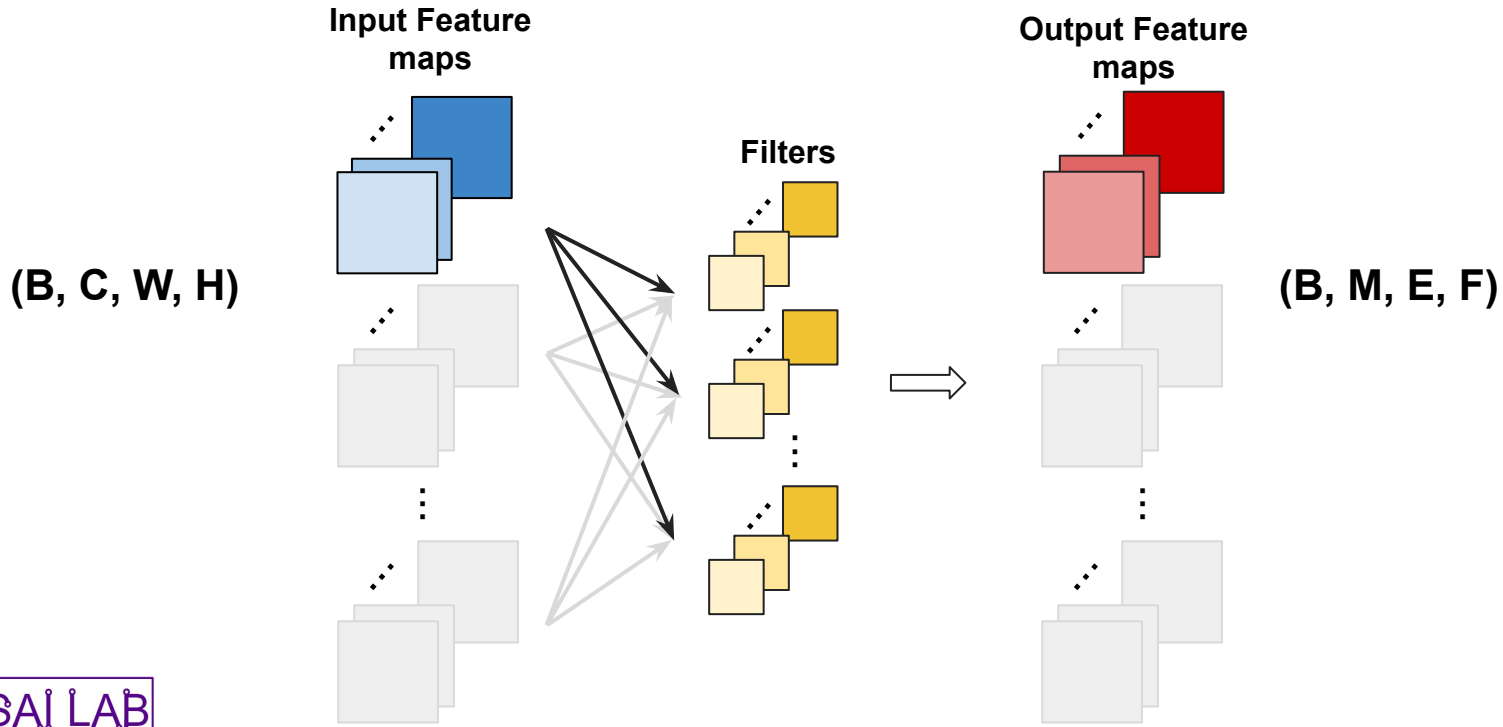
Convolution



- Number of MACs: M×K×K×C×E×F
- Storage cost: 32×(M×C×K×K+C×H×W+M×E×F)

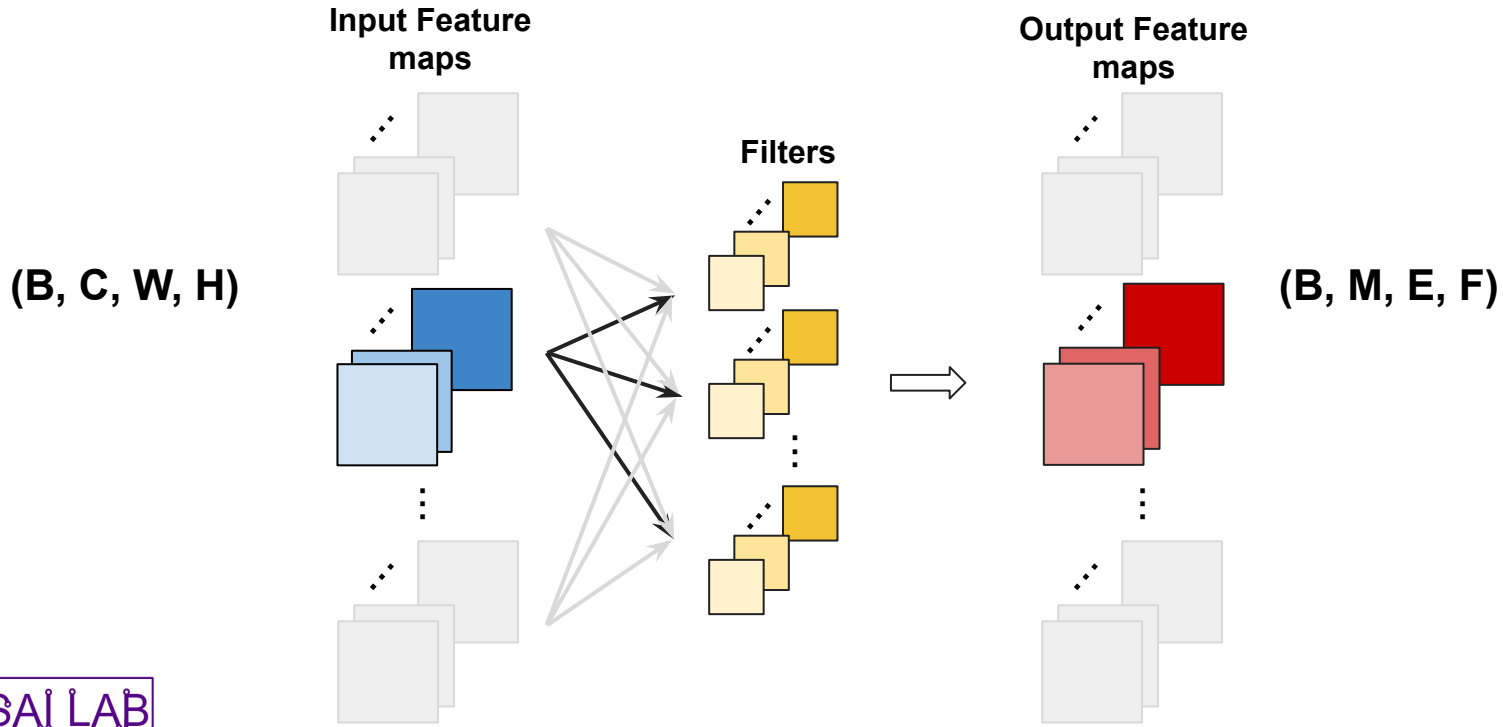
C: number of input channels
H,W: size of the input feature maps
M: number of weight filters
K: weight kernel size
E,F: size of the output feature maps

Convolution



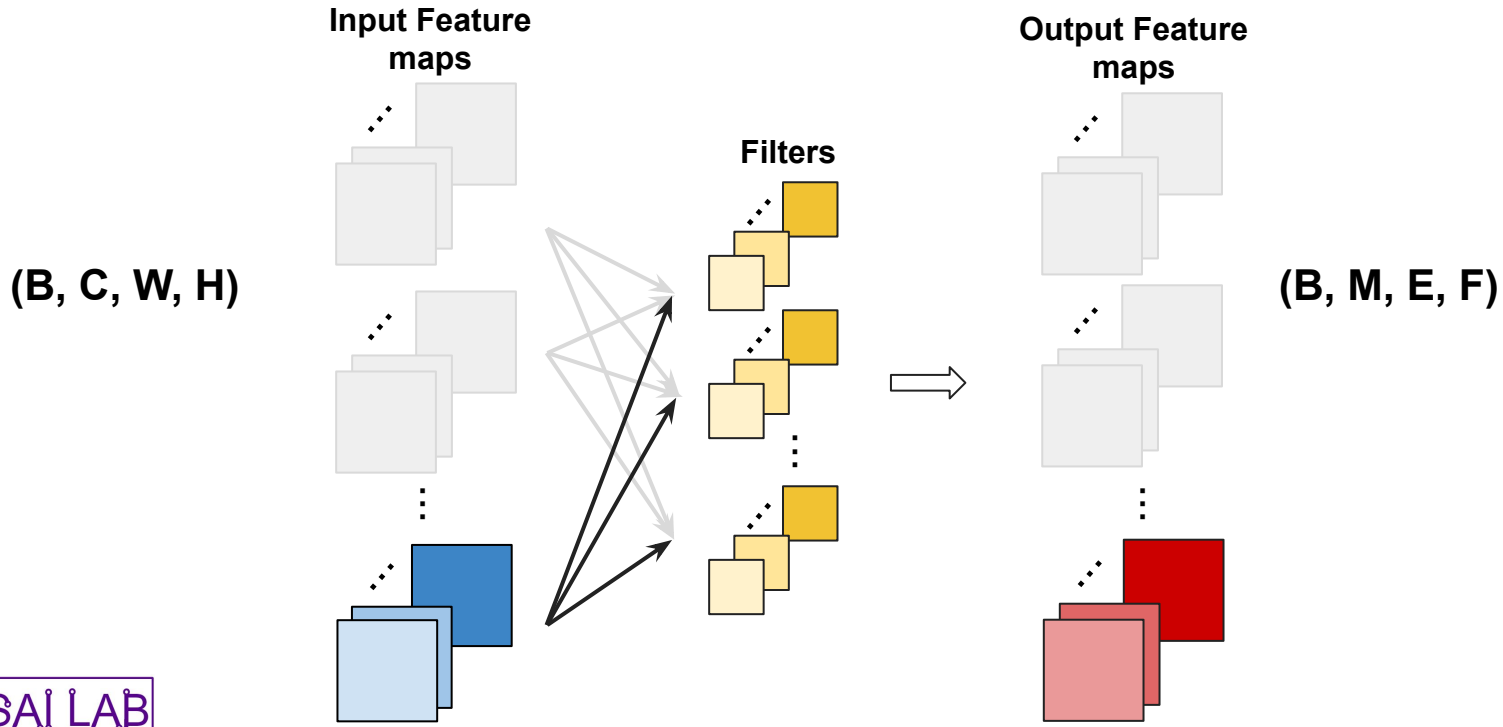
NYU SAI LAB

Convolution

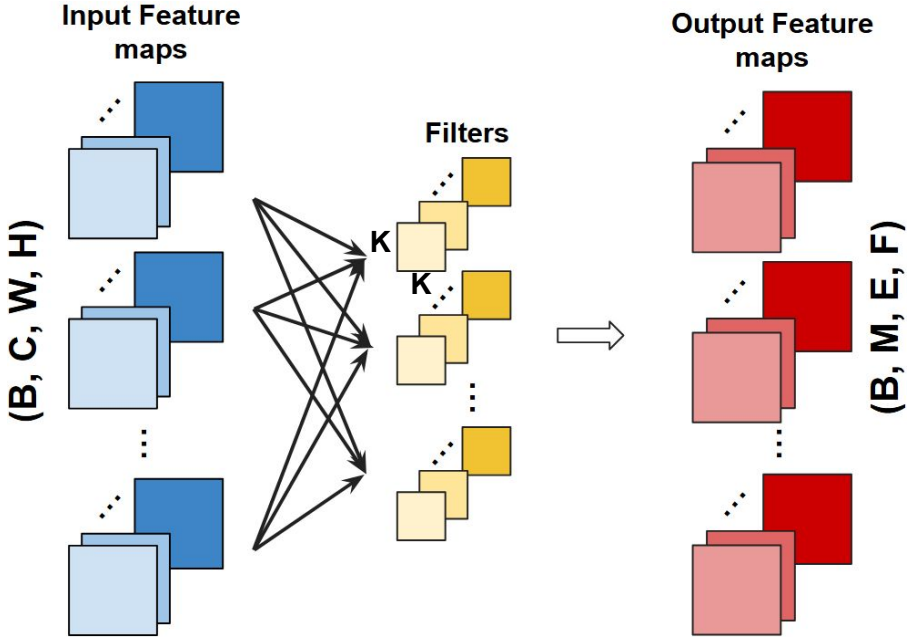


NYU SAI LAB

Convolution



Computational Cost: Standard Convolution



- Number of MACs: B×M×K×K×C×E×F
- Storage cost: 32×(M×C×K×K+B×C×H×W+B×M×E×F)

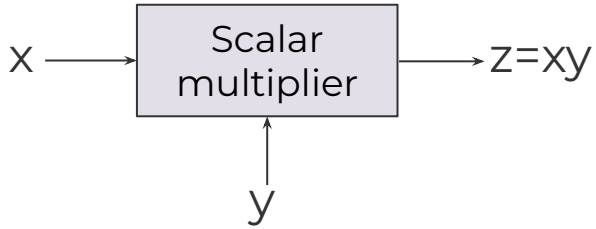
  B: batch size
  C: number of input channels
  H,W: size of the input feature maps
  M: number of weight filters
  K: weight kernel size
  E,F: size of the output feature maps

- We need to iterate over seven dimensions:
  - B, M, C, E, F, K(kernel width), K (kernel height)

NYU SAI LAB

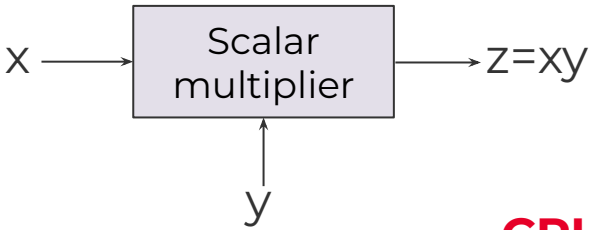
Computational Dataflow for CNN

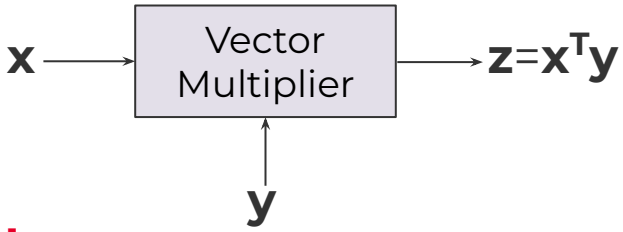
```
for b = 1 to B
   for m = 1 to M
      for c = 1 to C
         for w = 1 to E
            for h = 1 to F
               for k1 = 1 to K
                  for k2 = 1 to K
                     out[b][m][e][f] += in[b][c][e+k1-(K+1)/2][f+k2-(K+1)/2] * filter[m][c][k1][k2];
```



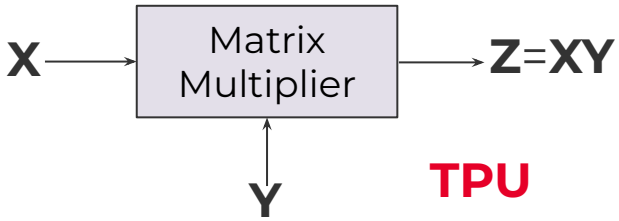
- This simple loop nest can be transformed in numerous ways to capture different reuse patterns of the activations and weights and to map the computation to a hardware accelerator implementation.
- A CNN's dataflow defines how the loops are ordered, partitioned, and parallelized
- We can use the scaler machine to compute the results of CNN using this for loop

Computational Dataflow for CNN

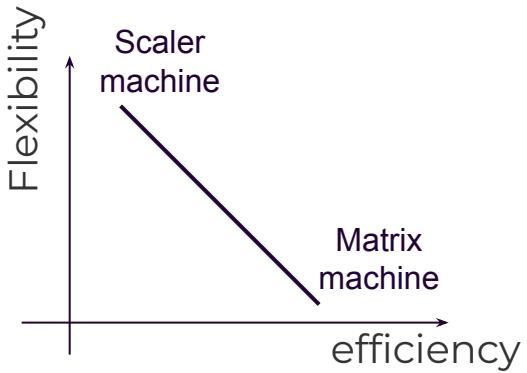




CPU & GPU

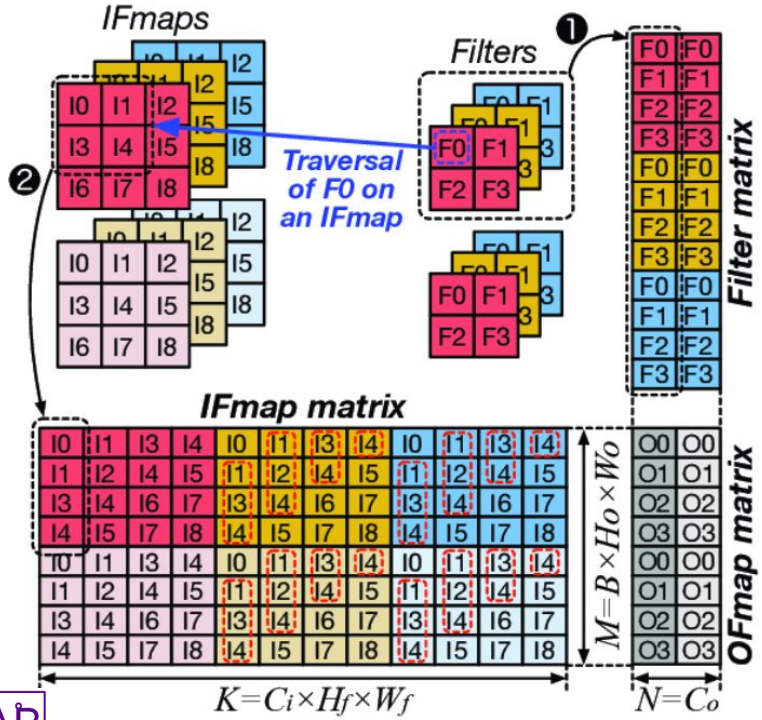


TPU



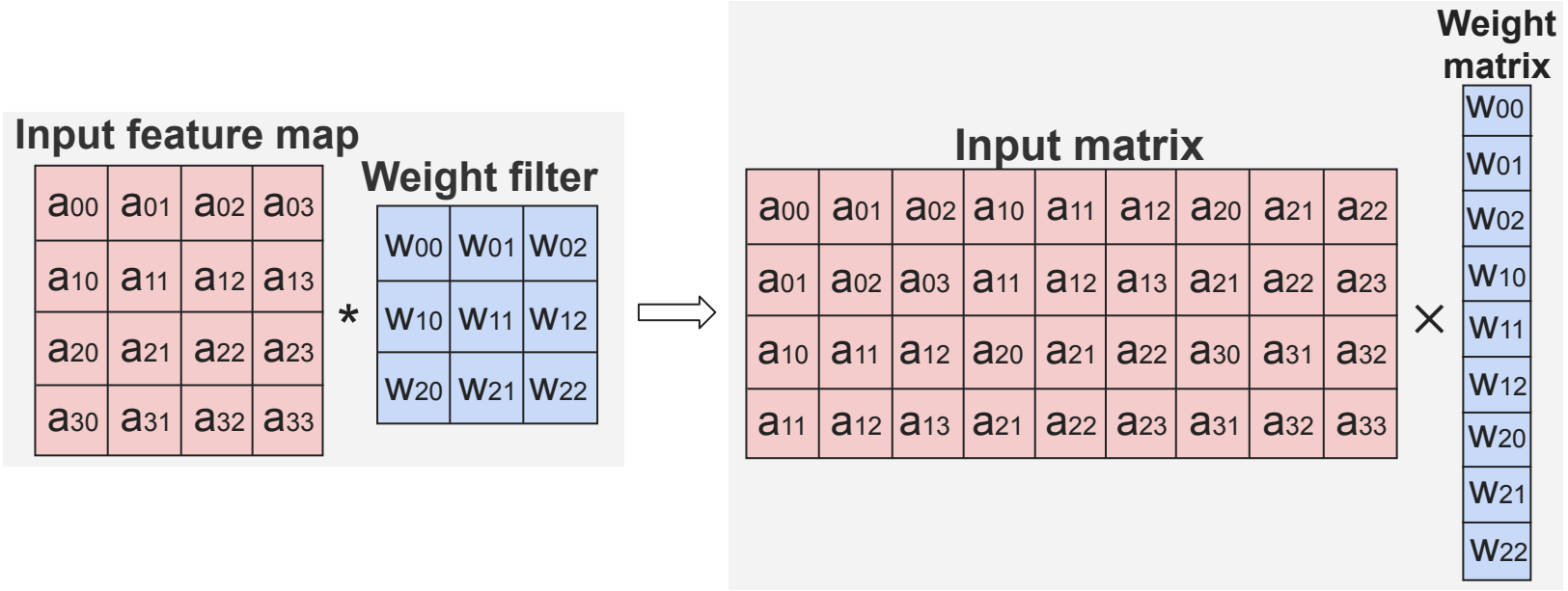
NYU SAI LAB

How to Convert to Matrix Multiplication?

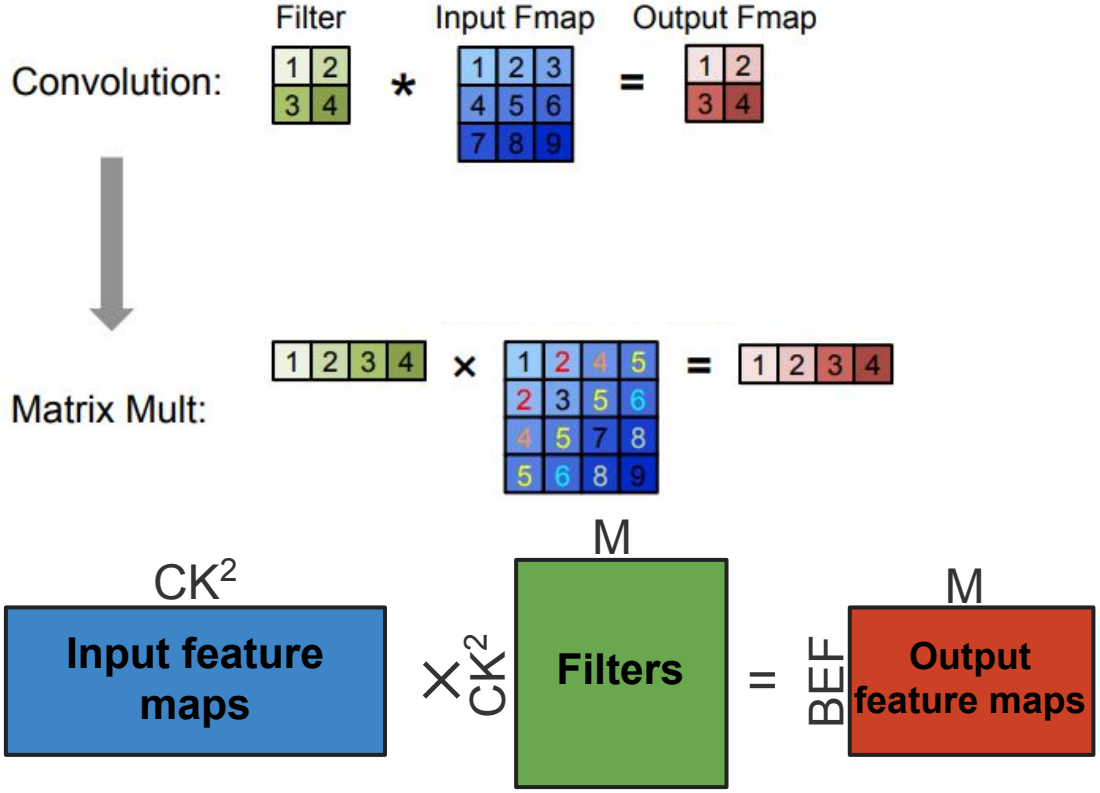


- A standard Convolutional operation can be converted to 2D matrix multiplication using **Im2Col** operations.

How to Convert to Matrix Multiplication?



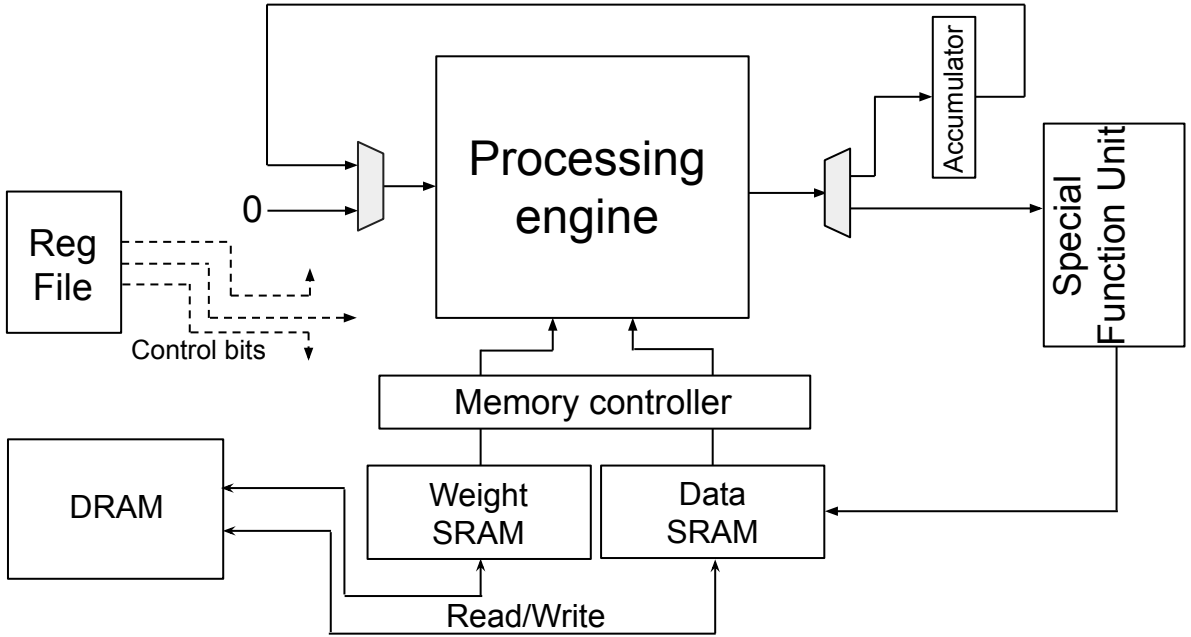
How to Convert to Matrix Multiplication?



Topics

- Hardware accelerator: Overview
- Convolutional operation conversion
- Systolic array
- Convolutional Neural Network System
- Popular accelerator design

Hardware Architectures for DNN Processing

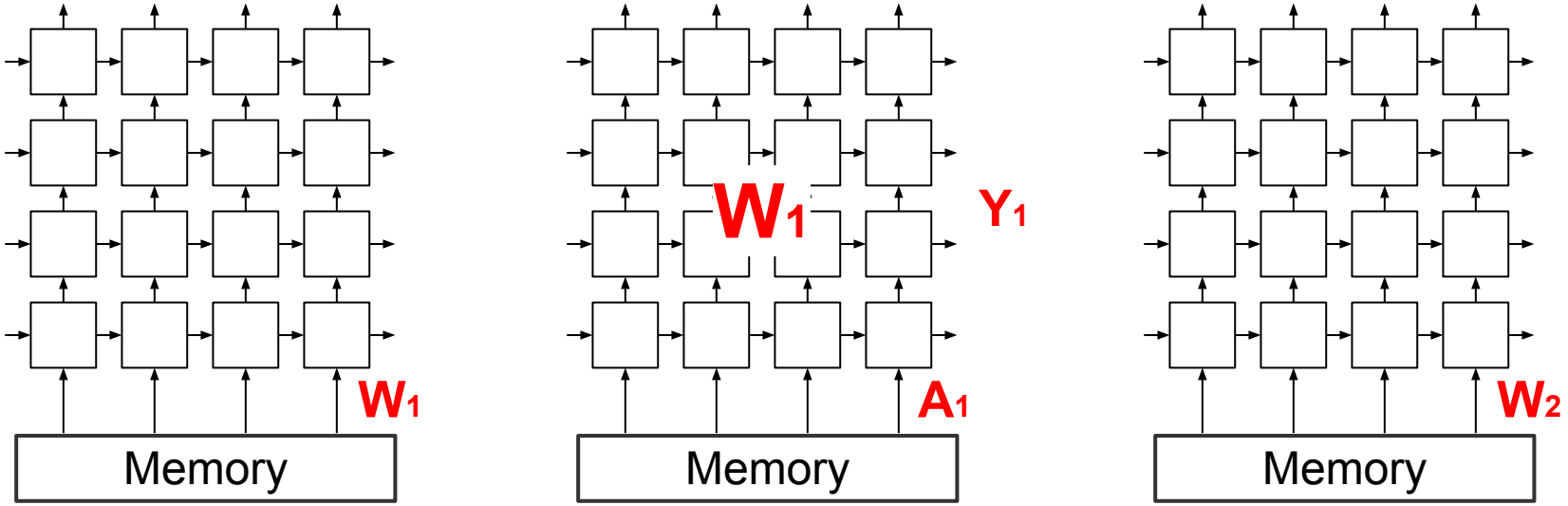


Major building blocks:

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

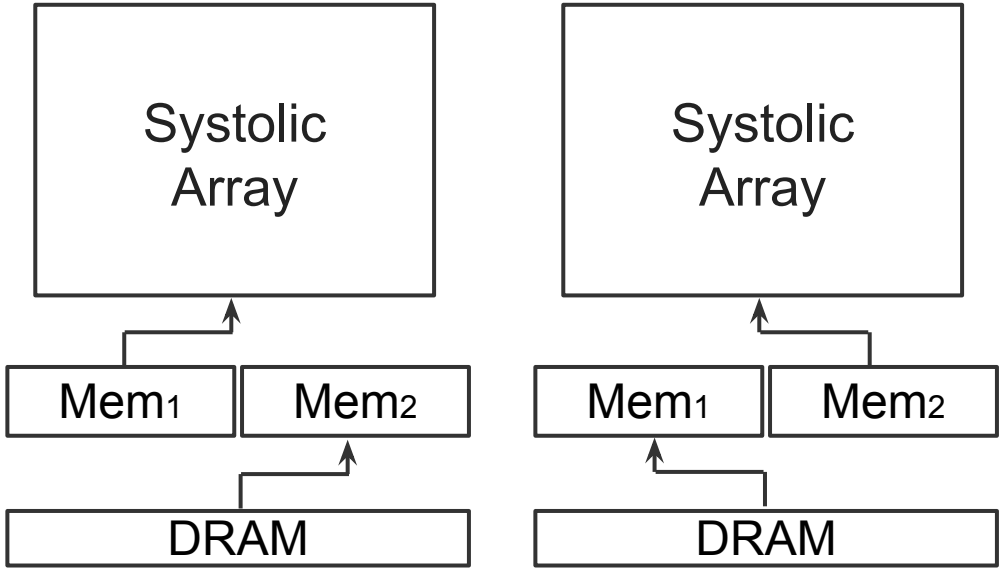
NYU SAI LAB

Computing Paradigms



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

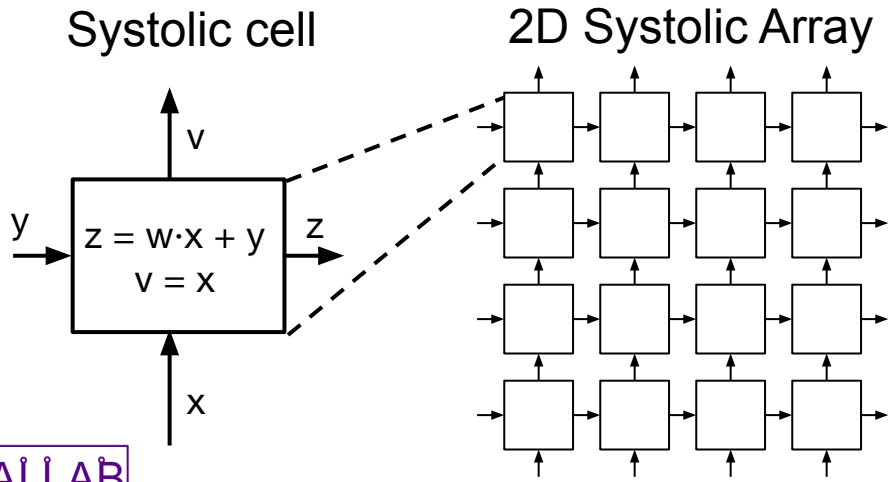
Double Buffering



- 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)

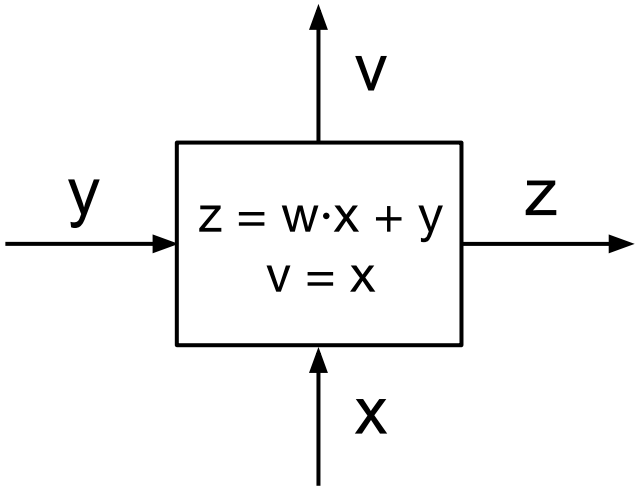
- Kung and Leiserson, "Systolic Arrays for VLSI," 1978 and Kung, "Why systolic architectures?' 1982
- 2D grid of multiplier-accumulators (MACs) for matrix multiplication
- Used by Google TPU for deep learning (2017), etc



TPU (Google)

NYU SAI LAB

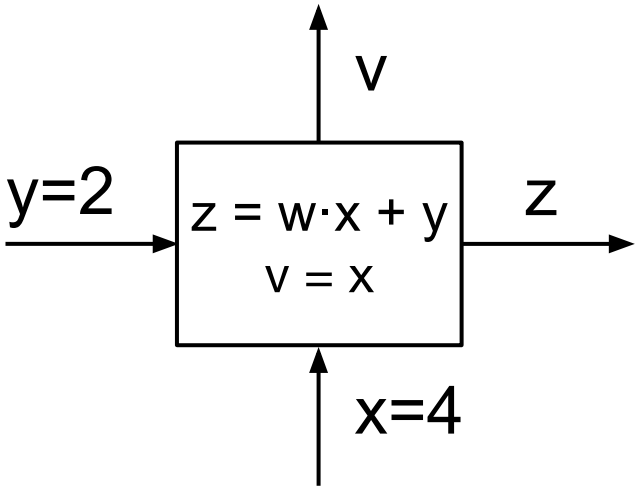
Systolic Cell



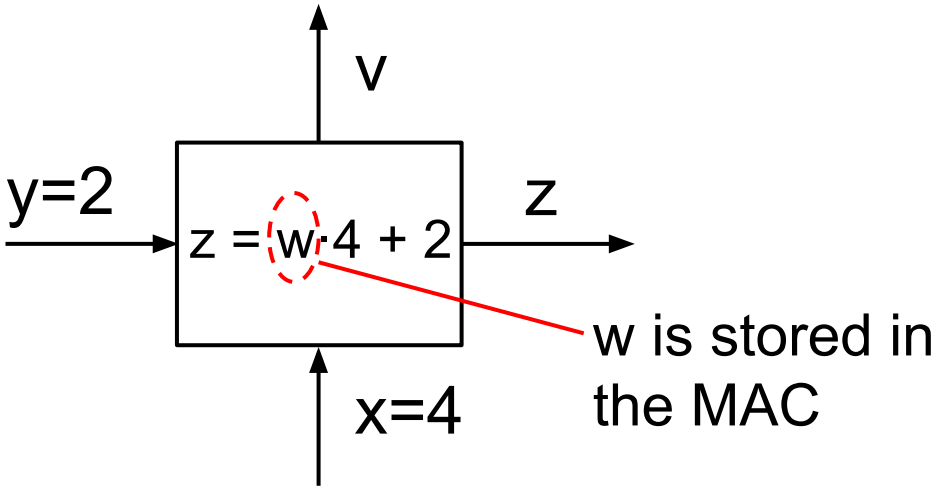
- Takes data (x and y) as input
- w stays in the systolic cell
- Performs a multiply-accumulate operation

NYU SAI LAB

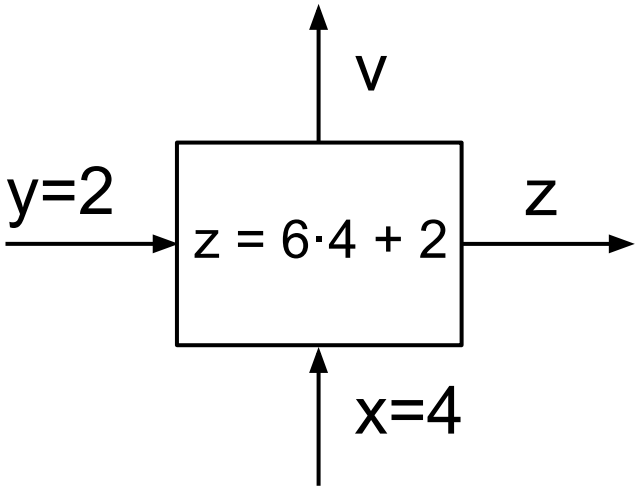
Systolic Cell



Systolic Cell

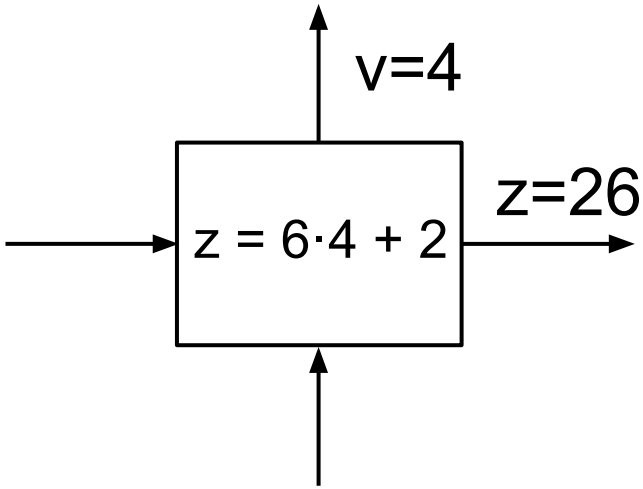


Systolic Cell

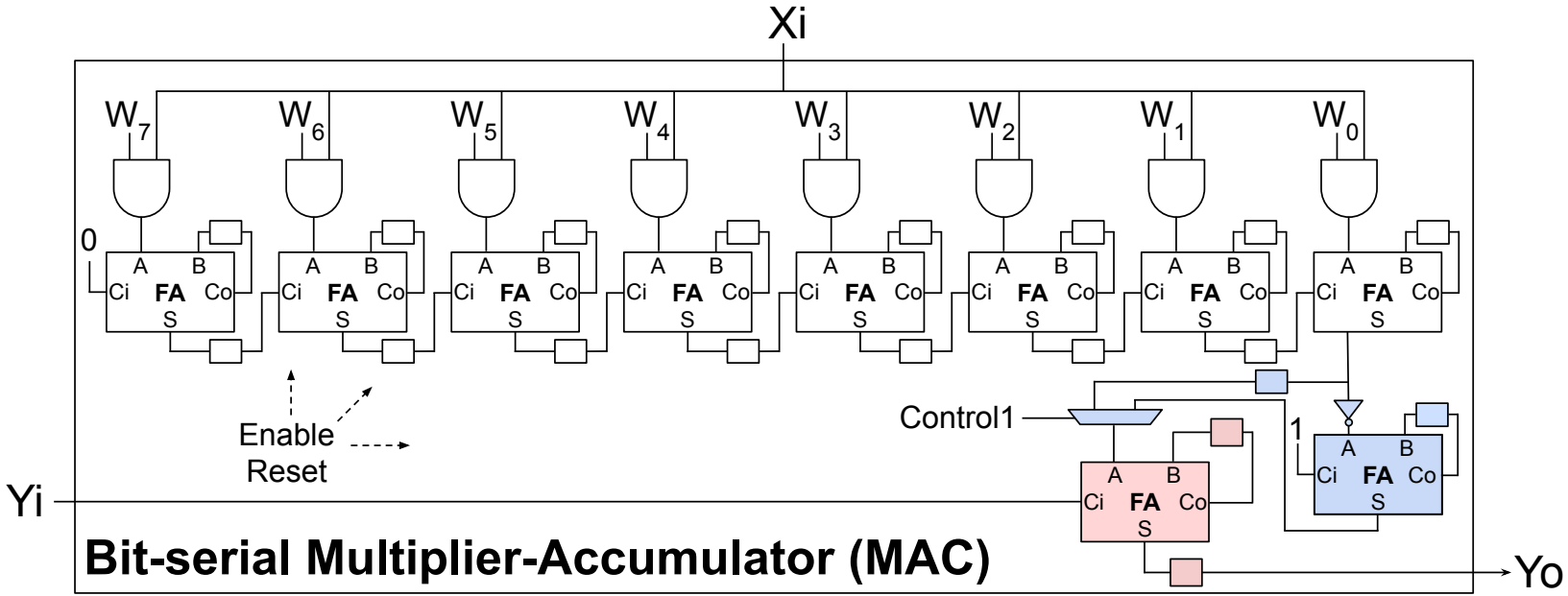


NYU SAI LAB

Systolic Cell



Multiplier Accumulator



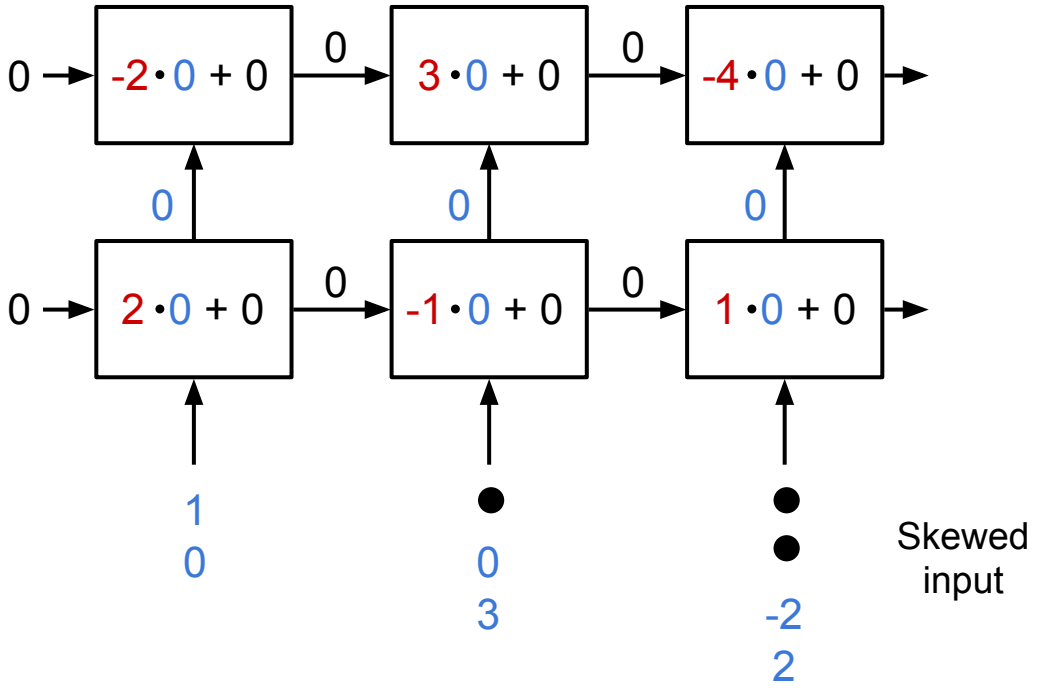
NYU SAI LAB

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

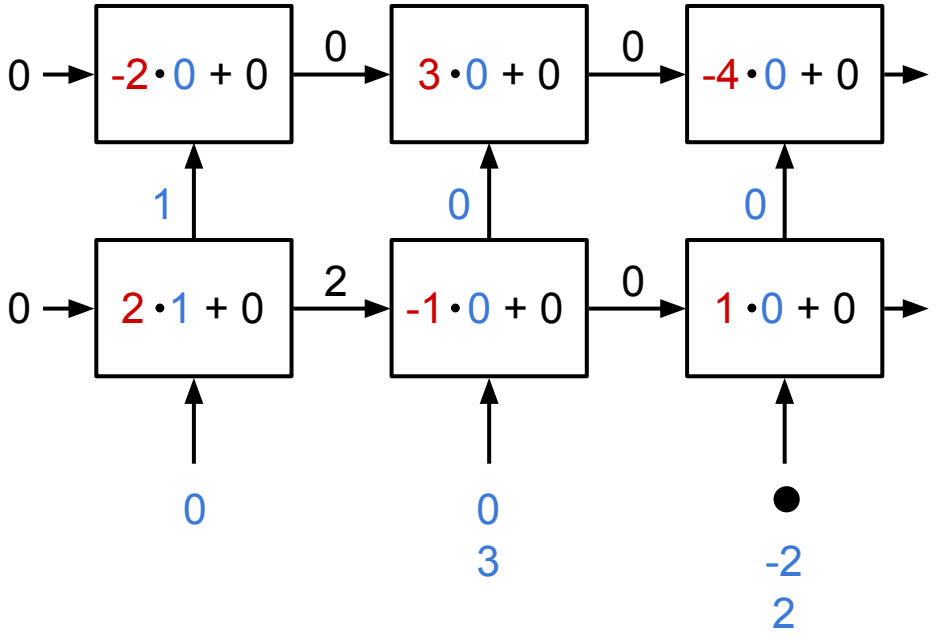


Visualizing Systolic Array Multiplication

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

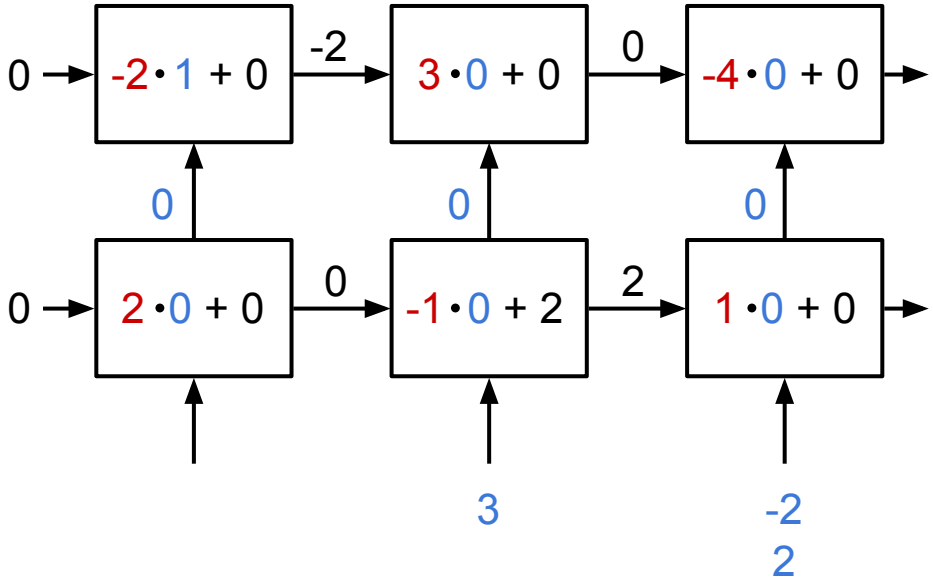


NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

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



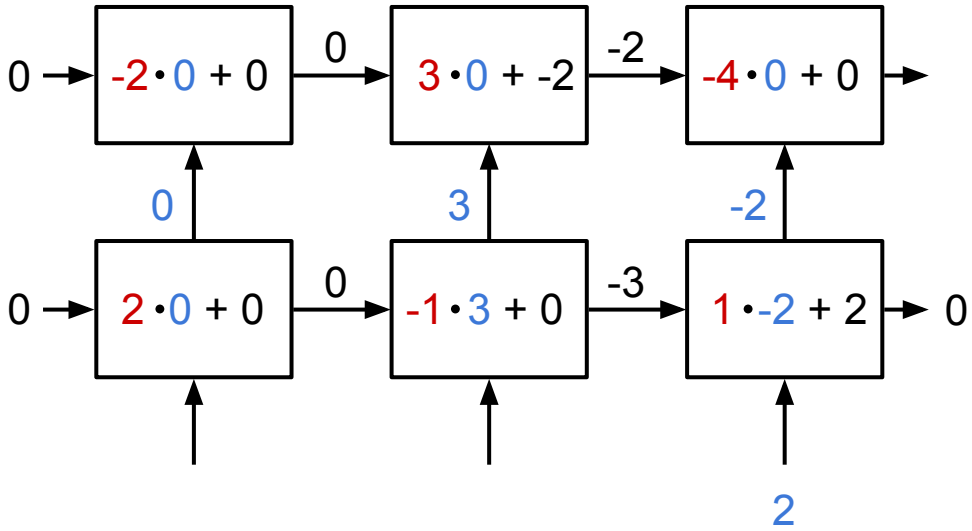
Weights in red are preloaded into the systolic array

NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

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



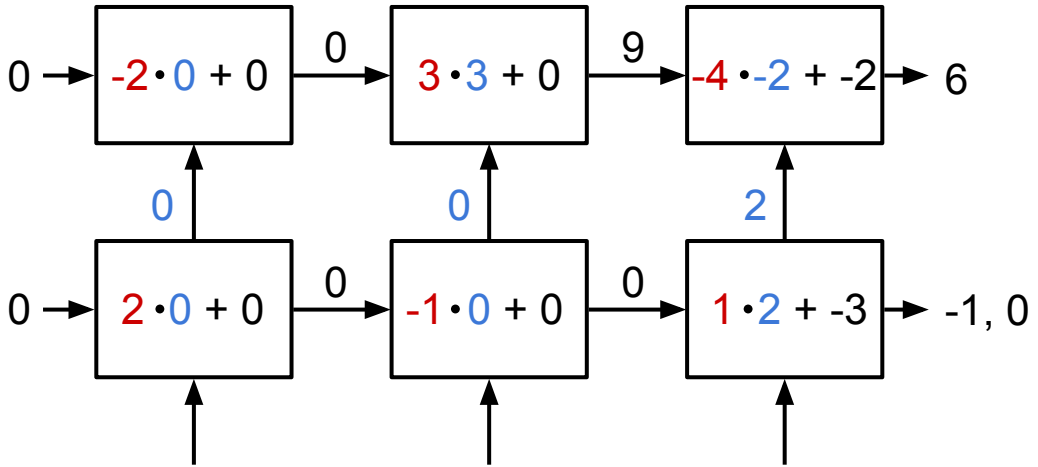
Weights in red are preloaded into the systolic array

NYU SAI LAB

Visualizing Systolic Array Multiplication

Weight Matrix × Data Matrix = Result Matrix

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



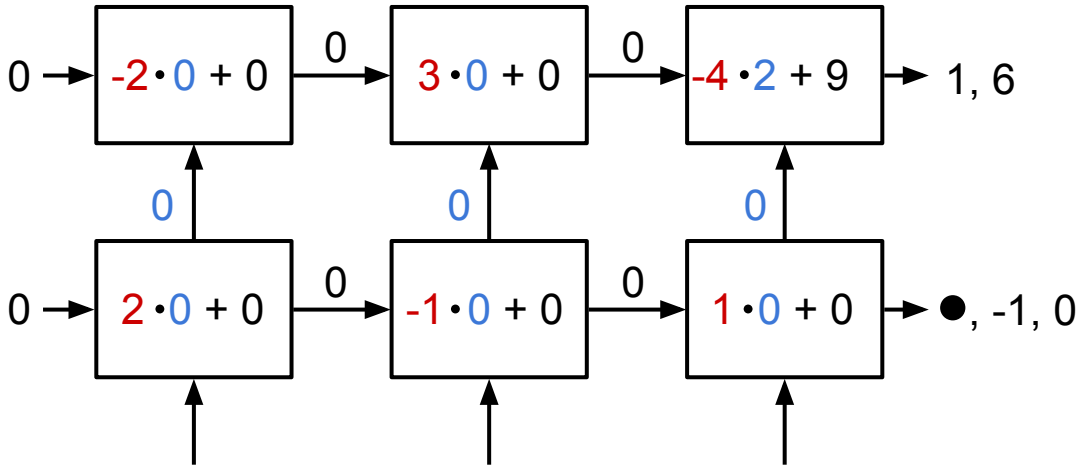
Weights in red are preloaded into the systolic array

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

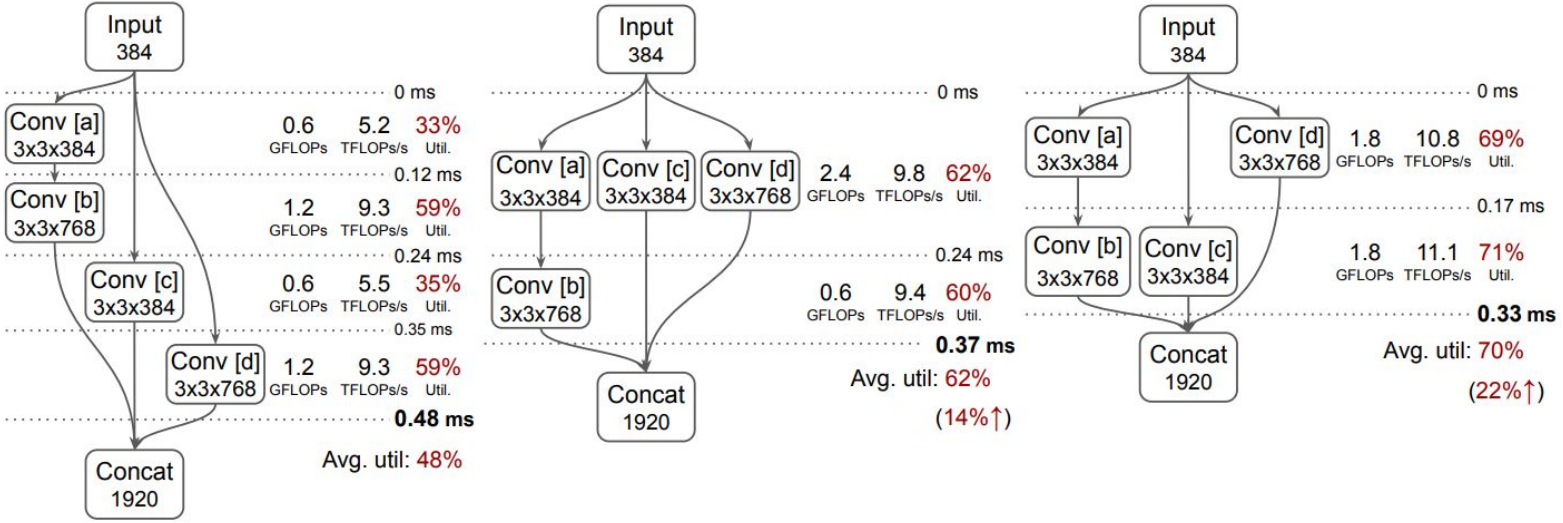
NYU SAI LAB

Topics

- Hardware accelerator: Overview
- Convolutional operation conversion
- Systolic array
- Convolutional Neural Network System
- Popular accelerator design

Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." *IEEE journal of solid-state circuits* 52.1 (2016): 127-138.

Computational Scheduling



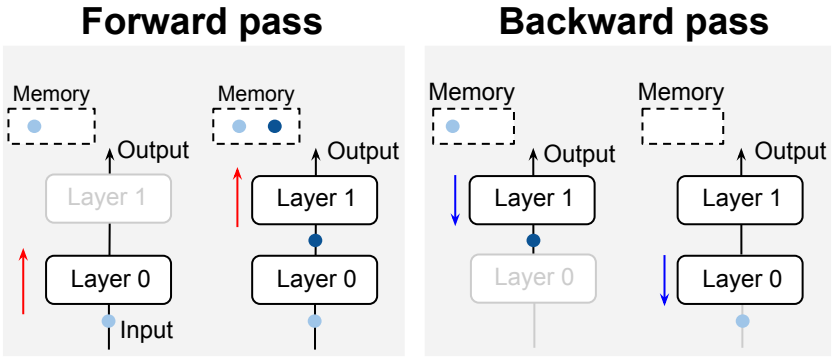
(1) Sequential Schedule (2) Greedy Schedule (3) Our Schedule (IOS)

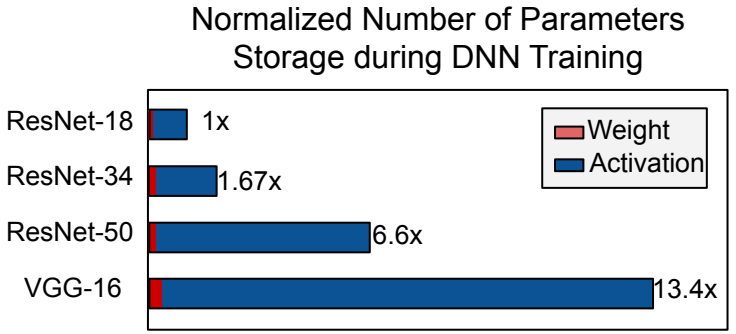
- The branchy CNN can be scheduling and computed in a much higher efficiency.
- Two convolutional operations can be combined to achieve less memory cost.

NYU SAI LAB

Ding, Yaoyao, et al. "Ios: Inter-operator scheduler for cnn acceleration." *Proceedings of Machine Learning and Systems* 3 (2021): 167-180.

Memory Efficient Neural Network Training





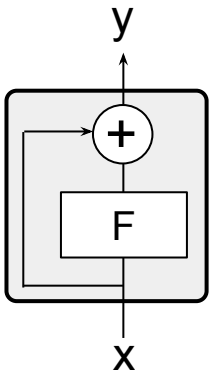
- The memory footprint grows proportional with the layer depth.
- On top of this, small edge devices typically have limited on-chip storage, leading to frequent and costly accesses to off-chip memories.

NYU SAI LAB

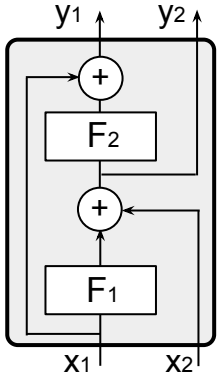
Zhang, Sai Qian, et al. "CAMEL: Co-Designing AI Models and eDRAMs for Efficient On-Device Learning." *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024.

Memory Efficient Neural Network Training

Residual Architecture



Reversible Architecture



Forward pass:

$y_2 = F_1(x_1) + x_2$

$y_1 = F_2(y_2) + x_1$

Backward Pass:

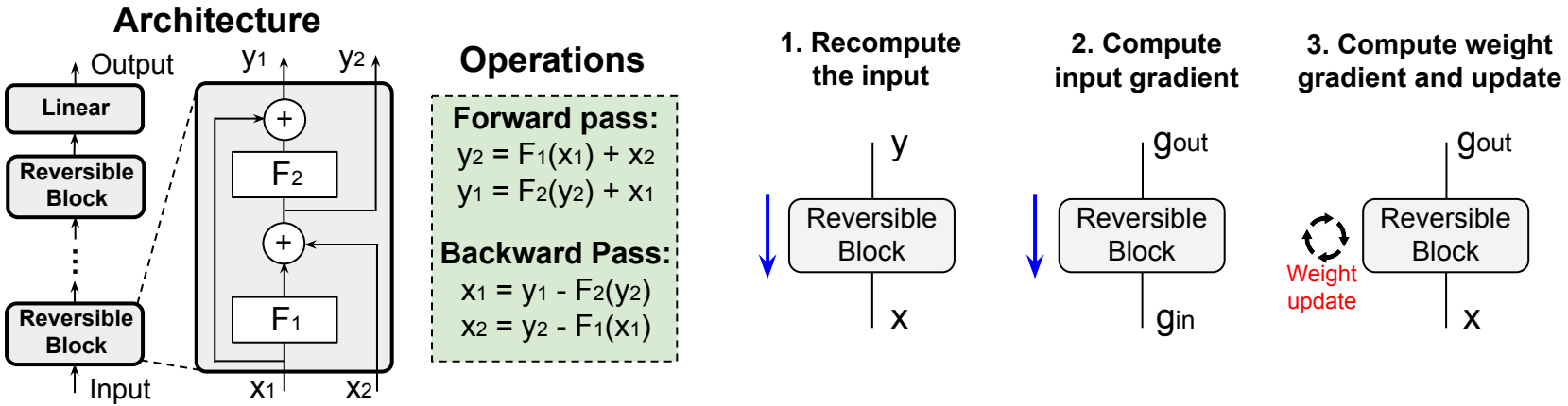
$x_1 = y_1 - F_2(y_2)$

$x_2 = y_2 - F_1(x_1)$

- A reversible residual network (RevNet) is a variant of the canonical residual neural network (ResNet).

Zhang, Sai Qian, et al. "CAMEL: Co-Designing AI Models and eDRAMs for Efficient On-Device Learning." *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024.

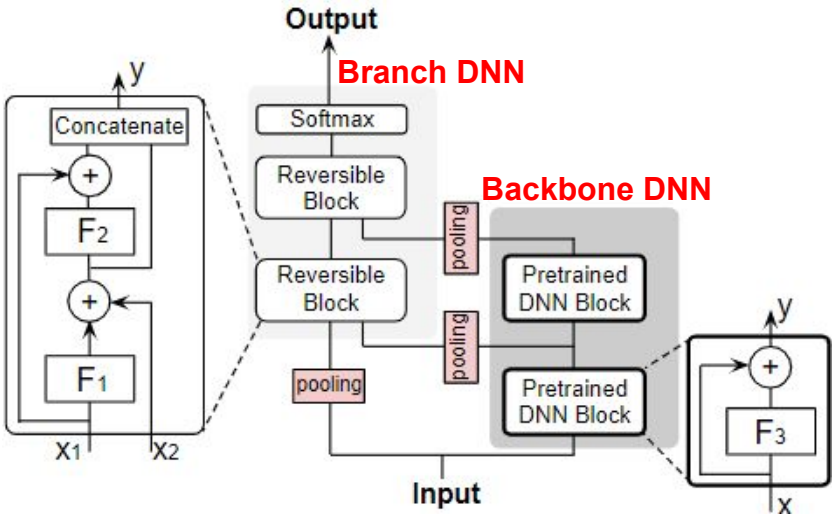
Memory Efficient Neural Network Training



- The reversible architecture enables the backward pass computations to be performed without the need to store the input activations.
- Given the output y, the input activations are first recomputed. Afterwards, the input and weight gradients are computed with standard backward pass operations.

Zhang, Sai Qian, et al. "CAMEL: Co-Designing AI Models and eDRAMs for Efficient On-Device Learning." *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024.

Memory Efficient Neural Network Training



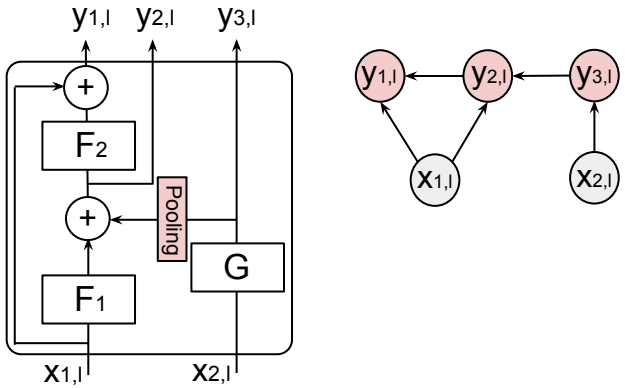
Duplex DNN

- This approach in turn imposes higher compute demands.
- We propose to judiciously train a subset of the model parameters to minimize training.
- The backbone DNN is frozen during the backward pass of the DNN.
- The normalization layers are removed from the branch DNN to facilitate the training process.

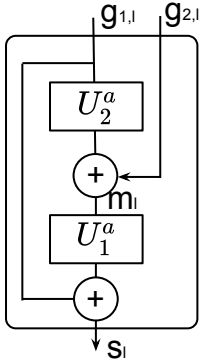
Zhang, Sai Qian, et al. "CAMEL: Co-Designing AI Models and eDRAMs for Efficient On-Device Learning." *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024.

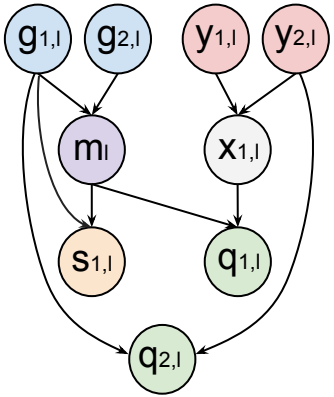
Memory Efficient Neural Network Training

Forward pass



Backward pass





- We carefully schedule the computations of DuDNN during both forward and backward passes to achieve optimal system performance.
- We propose the optimal compute pattern to minimize memory usage and lifetime.

NYU SAI LAB

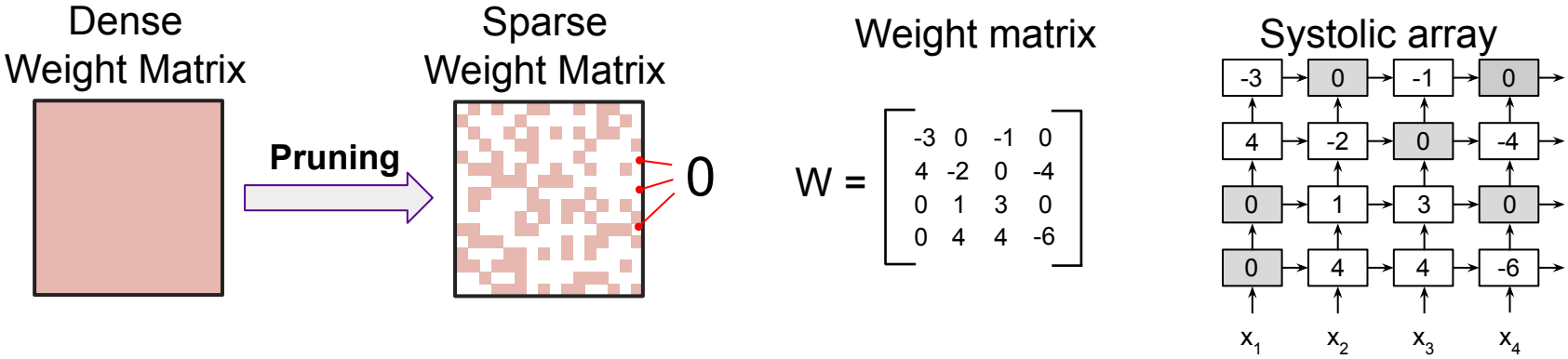
Zhang, Sai Qian, et al. "CAMEL: Co-Designing AI Models and eDRAMs for Efficient On-Device Learning." *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024.

Topics

- Hardware accelerator: Overview
- Convolutional operation conversion
- Systolic array
- Convolutional Neural Network System
- Popular accelerator design

Column Combining

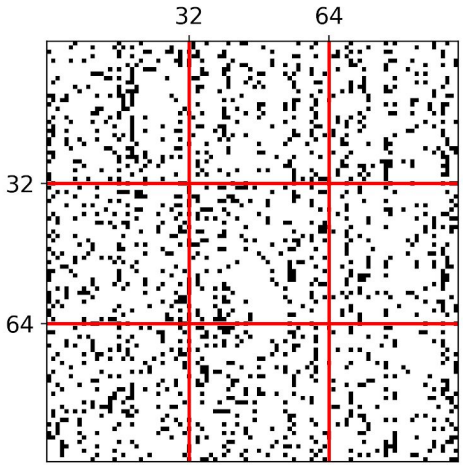
- Unimportant (i.e., small weights) are set to 0
- However, it is hard to leverage these zero weights to reduce the hardware cost



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.

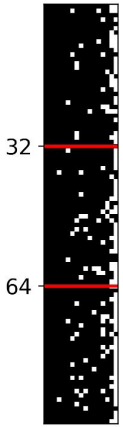
Column Combining

Sparse
Weight Matrix



Column Combining
8x reduction in size

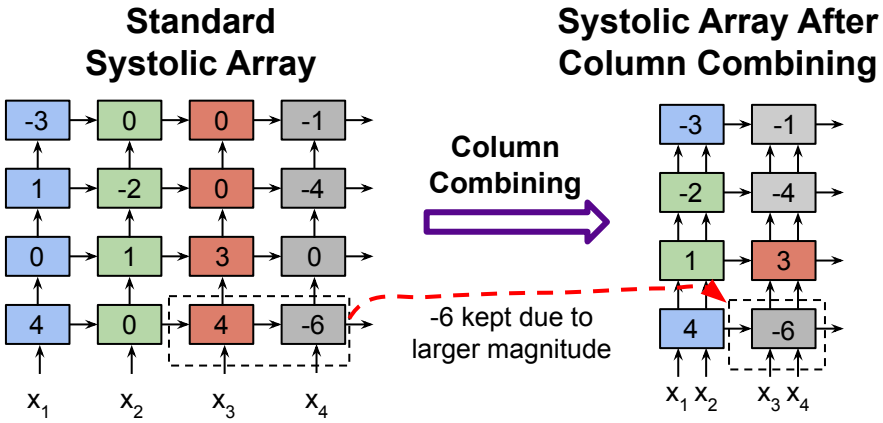
Packed Format in
Systolic Array

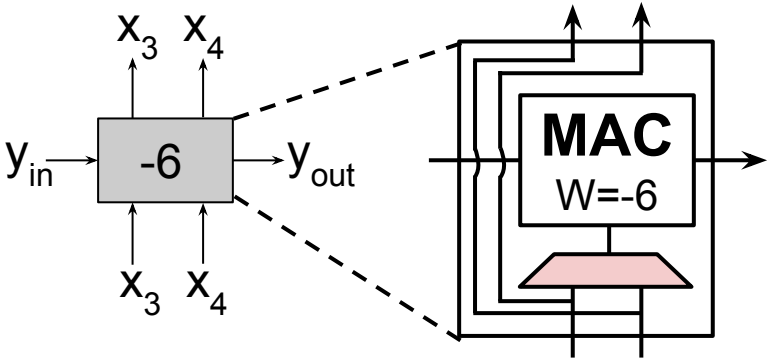


NYU SAI LAB

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



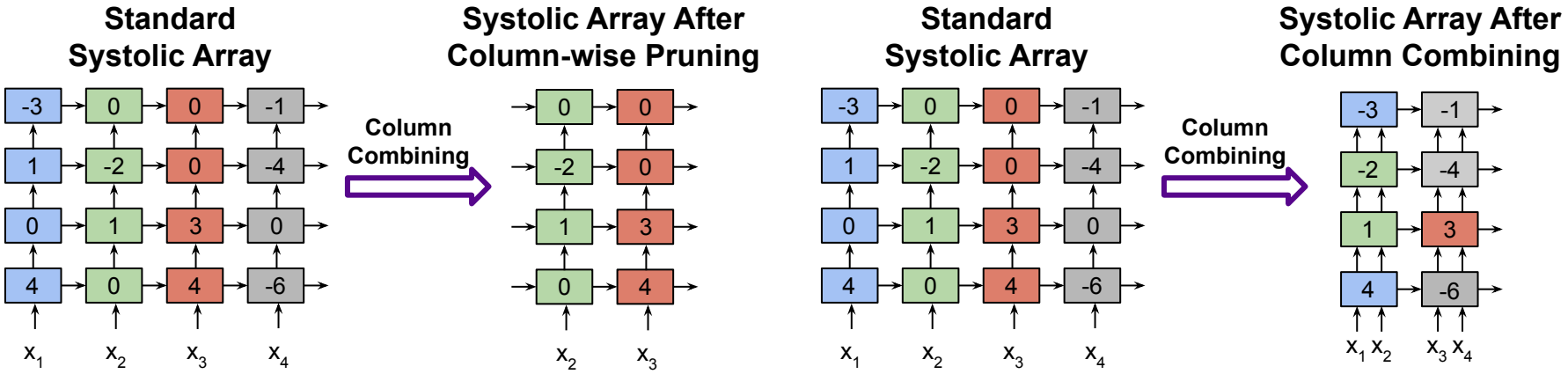


- Column combining can greatly increase the utilization of the systolic array.

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

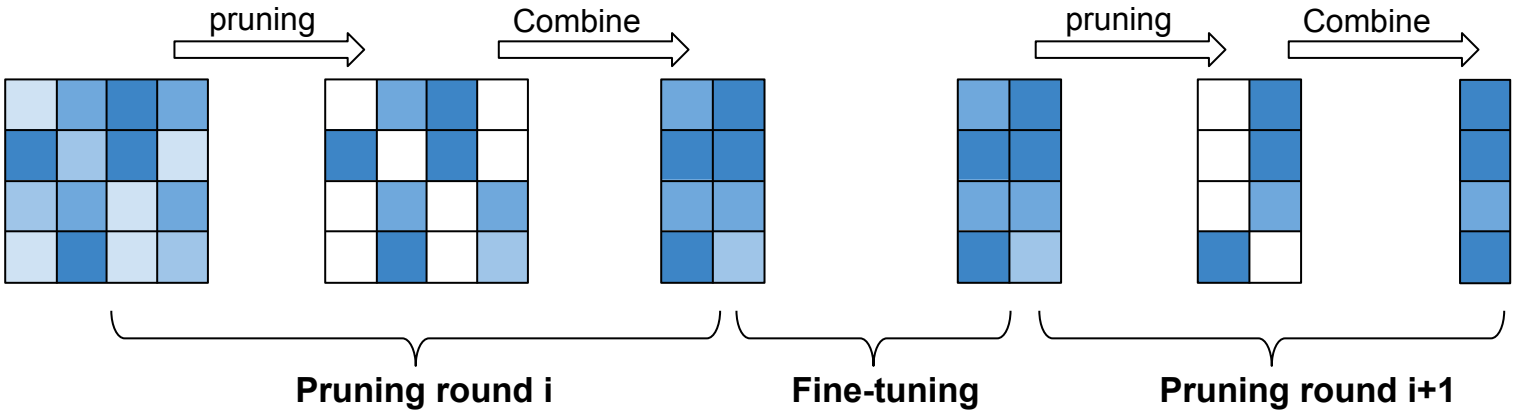


- Compared with column-wise Pruning (filterwise pruning), Column Combining allows for a much more flexible pruning pattern
- We can also apply Column Combining pruning on the input of each layer.

NYU SAI LAB

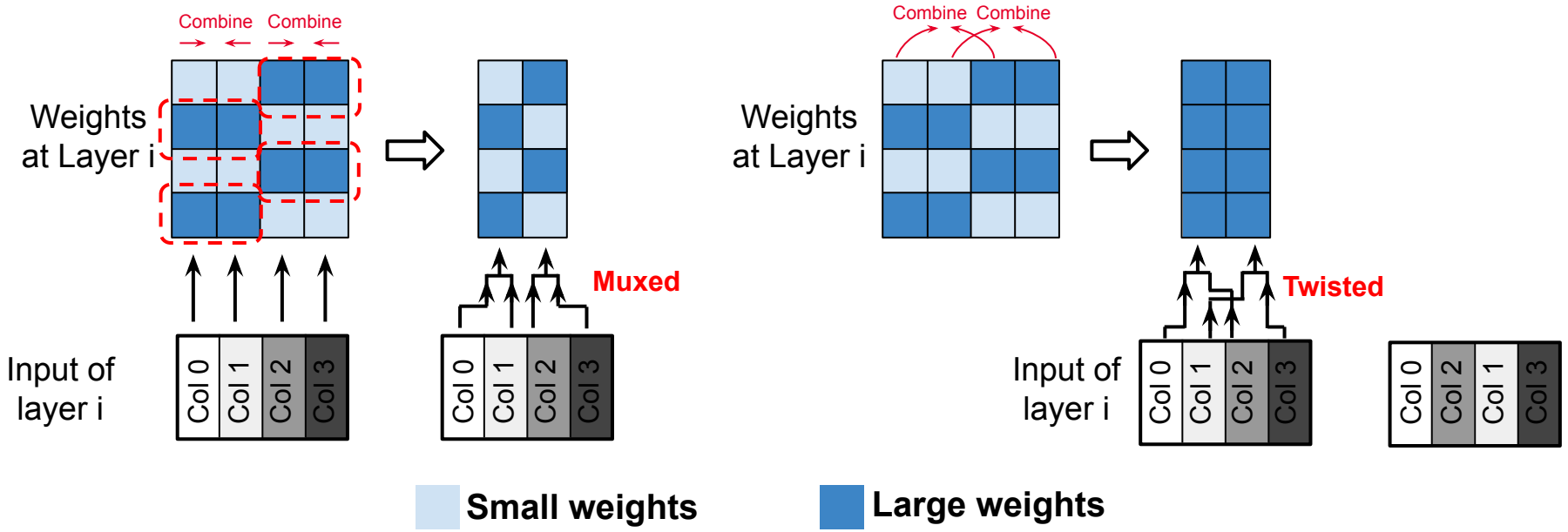
Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating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

Column-Combining Pruning



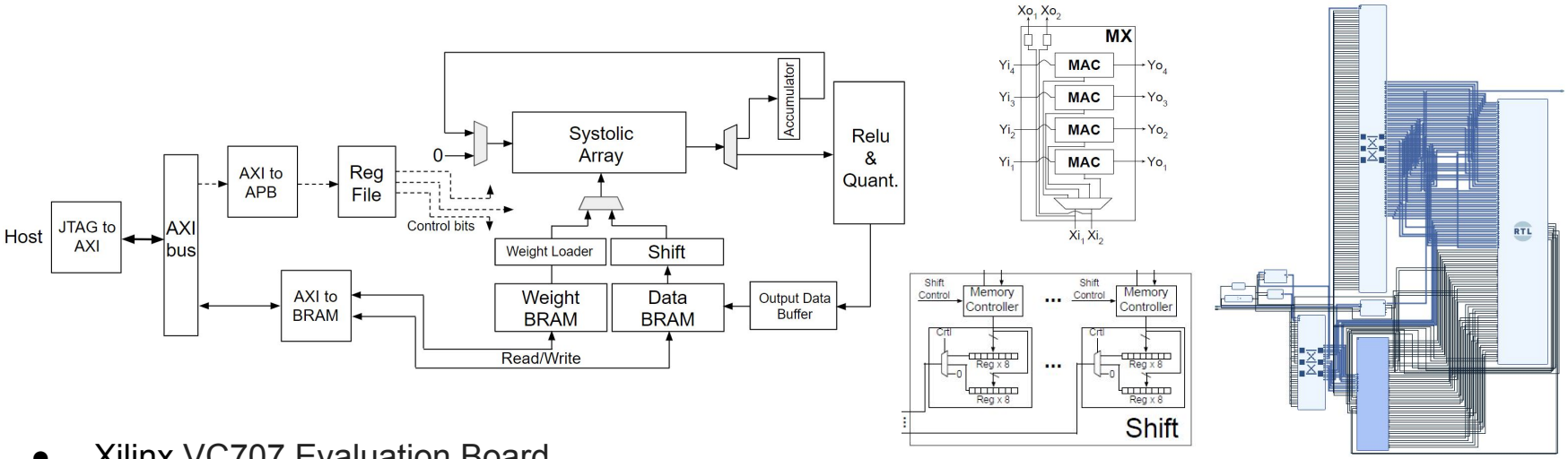
Kung, H. T., Bradley McDanel, and Sai Qian Zhang. "Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating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

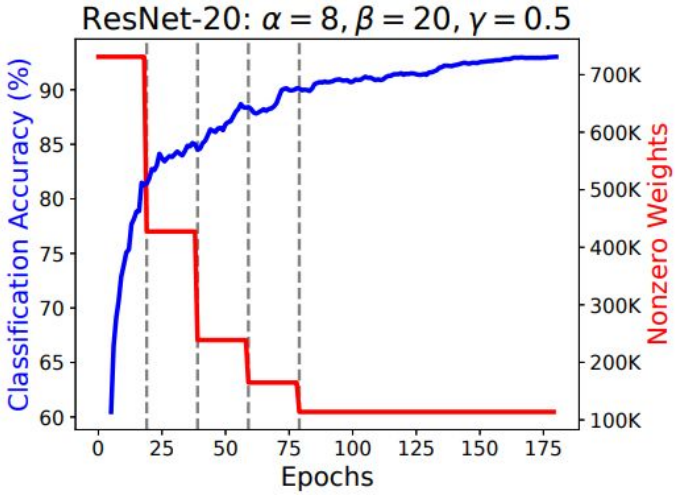


- Xilinx VC707 Evaluation Board
- Total hardware available: Lookup Table (303600), Flip-Flops (607200), BRAM (1030, each 36Kb)
- More than 15K lines of verilog code
- 128 by 64 systolic array

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%

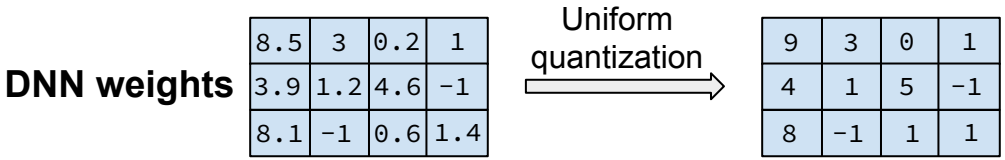
VGG-19 on ImageNet (87.5% sparsity)

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

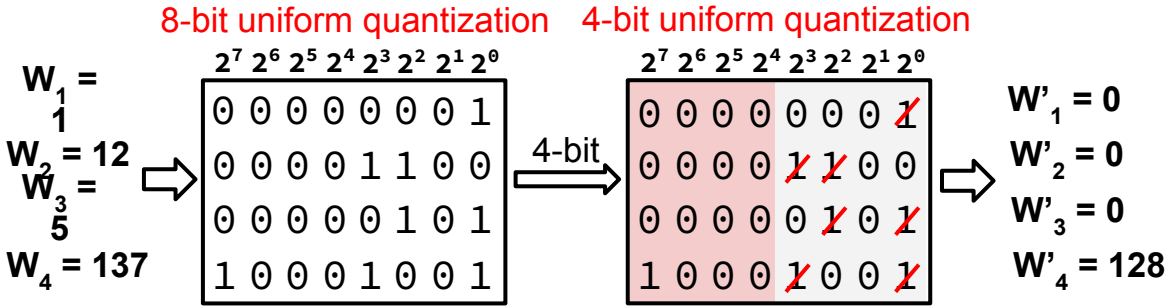
- Filter weights and activations can be quantized with low precision to accelerate the inference and reduce the model size.



- 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.

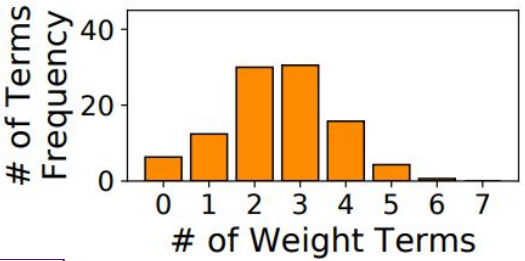
Representing Values in Power-of-two Terms

- 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	$2^7+2^3+2^0$

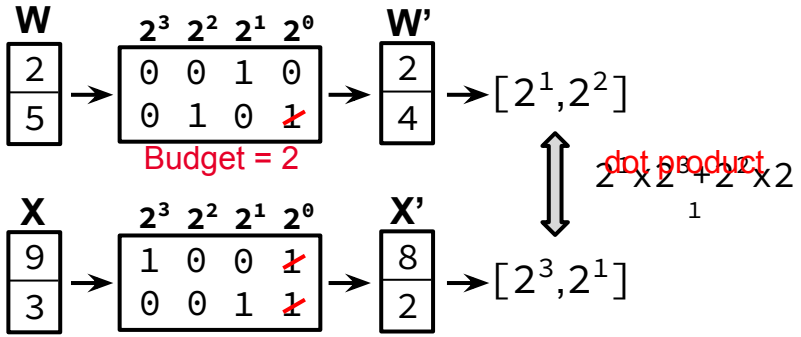
$$W_2X_2 = 12 \times 2 = (2^3+2^2) \times 2^1 = 2^3 \times 2^1 + 2^2 \times 2^1$$



- 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.

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.

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

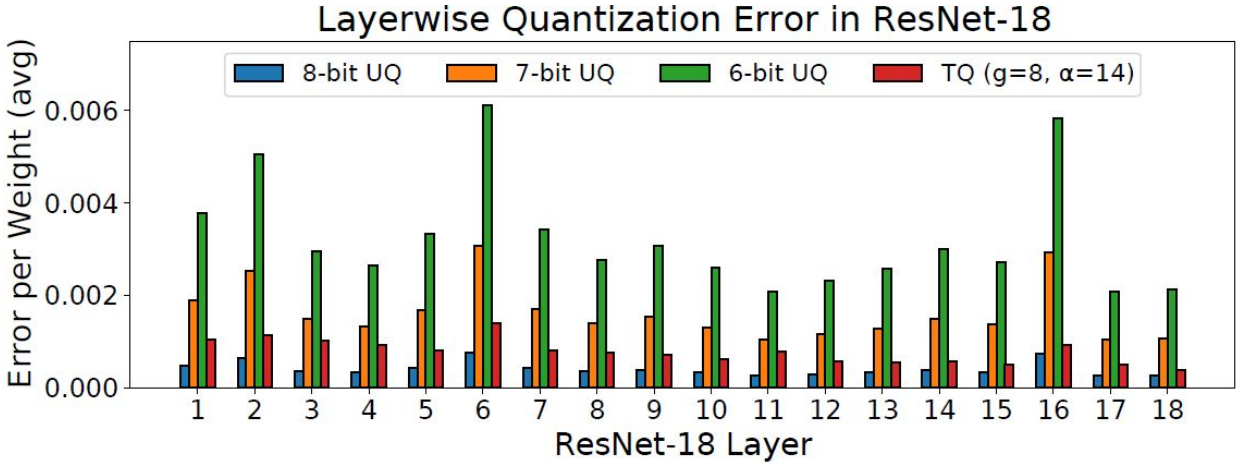
Binary expression	$31 = 2^4+2^3+2^2+2^1+2^0$	5 terms
	$27 = 2^4+2^3+2^1+2^0$	4 terms

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



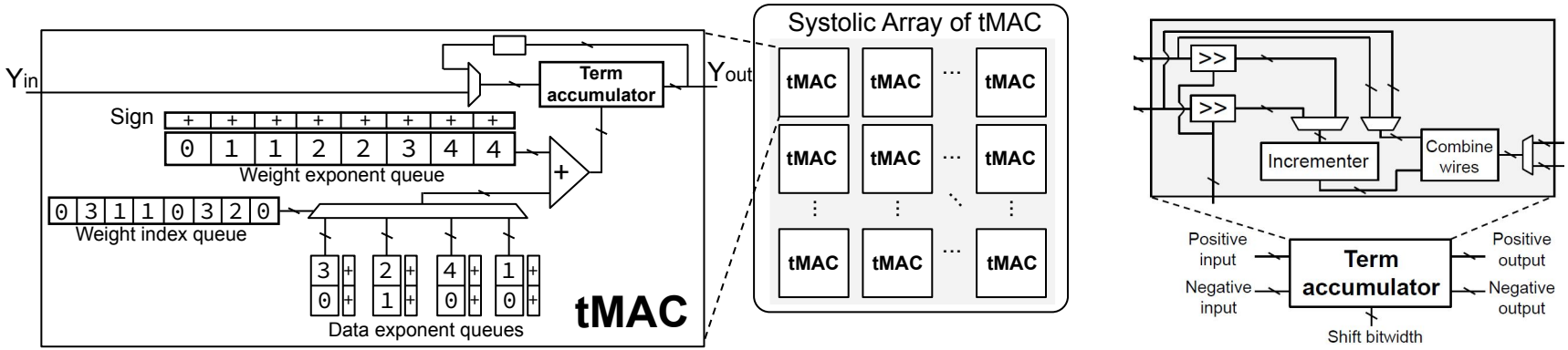
- We represent a group of 8 weights with 14 terms, each weight only requires 1.75 terms on average
- Term Quantization (TQ) introduces a small amount of quantization error over 8-bit uniform quantization (UQ)
- TQ achieves a much lower quantization error than 7-bit and 6-bit uniform quantization
- TQ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

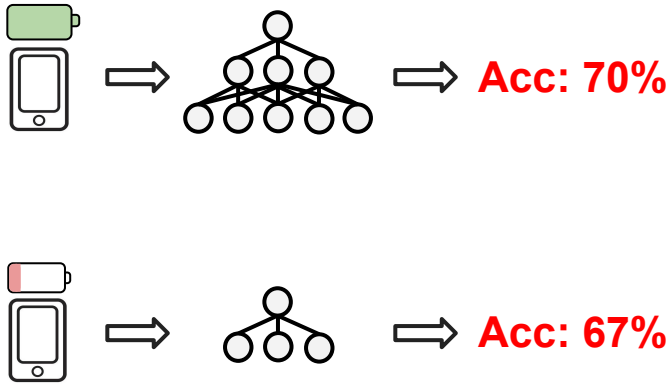
- We propose the term MAC (tMAC) for the efficient implementation of TQ.
- A tMAC processes all term-pair multiplications across a group of weight and data values.

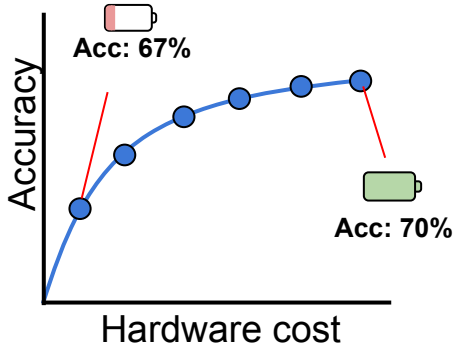


- Each term is represented by their corresponding exponent (2-3 bits).
- The term accumulation can be implemented using half adders.

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

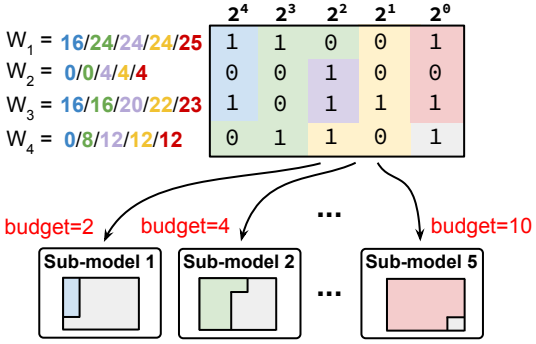




- DNN is expected to run at different resolution to achieve a good trade-off between hardware cost and accuracy.

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

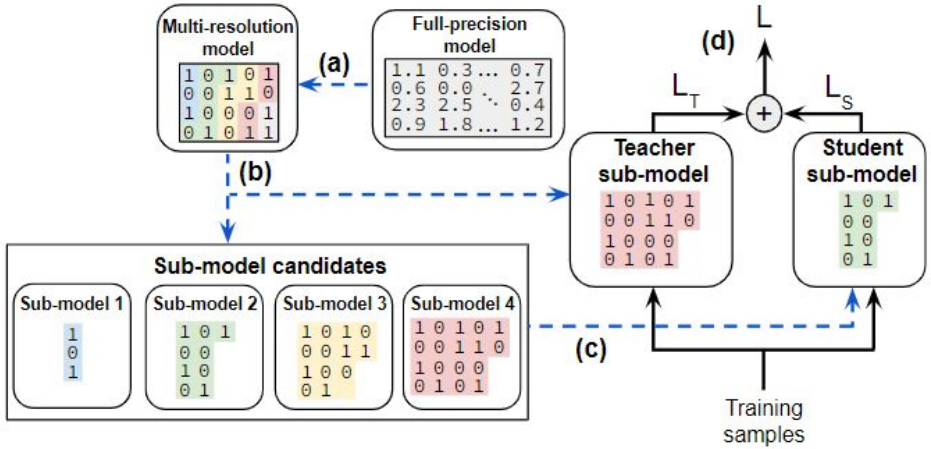


Multi-resolution DNN model

- 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

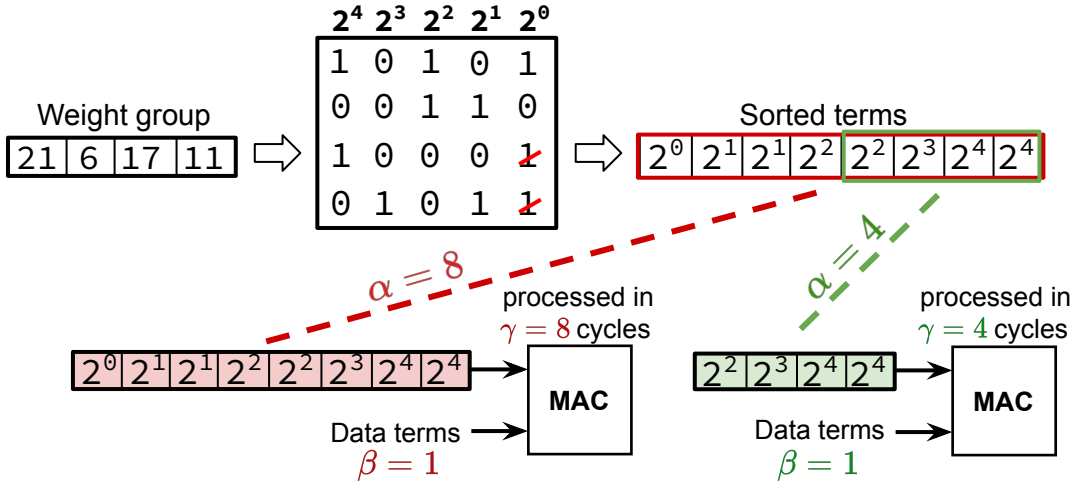


- We develop a multi-resolution training scheme to jointly train multiple DNN models under different term budgets.
- Instead of jointly train all the sub-models together, we apply the knowledge distillation framework to jointly train two sub-models per iteration.

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.

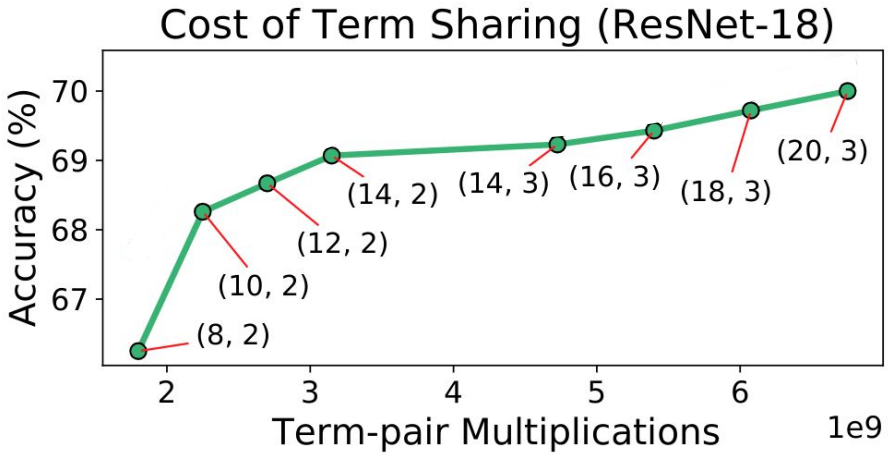
Multi-resolution DNN System

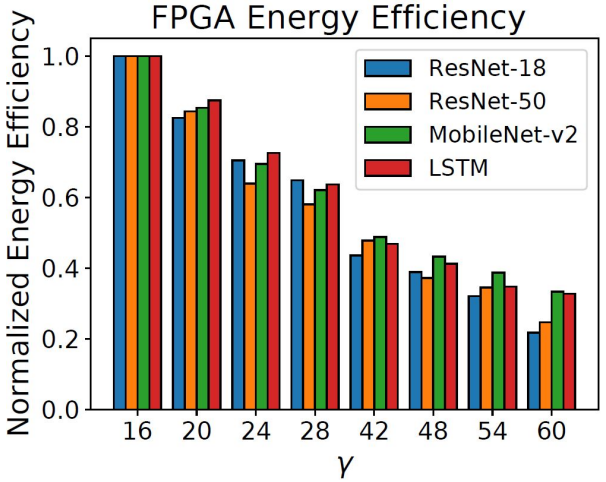


- The energy consumption will scale with the term budgets of the weight and data.

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

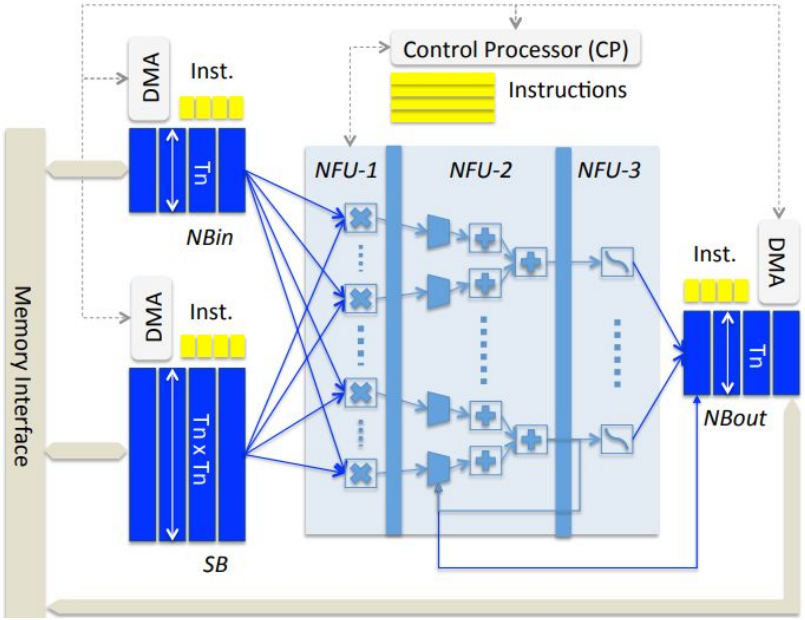




- The multi-resolution DNN incurs 0.4%-3.8% degradation compared to the original floating-point DNN (70.2%).
- The energy efficiency grows (3.25x on average), as term budget reduces from 60 to 16.

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

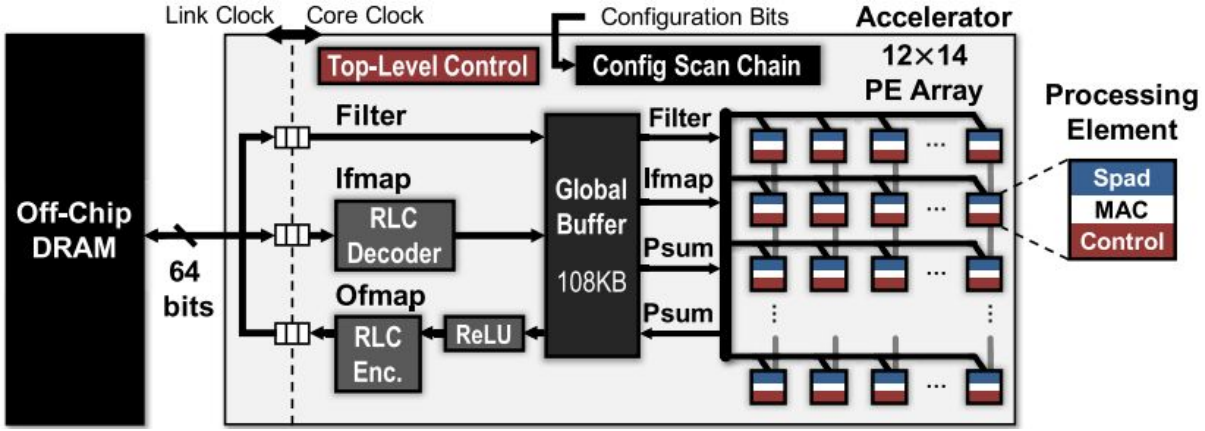


- The first popular end-to-end DNN (CNN) accelerator.
- Diannao is synthesized with 65nm using Synopsys tools, achieving a throughput of 482 GOP/s.
- NFU consists of three stages:
  - Multiplier units
  - Adder tree
  - Nonlinear unit

NYU SAI LAB

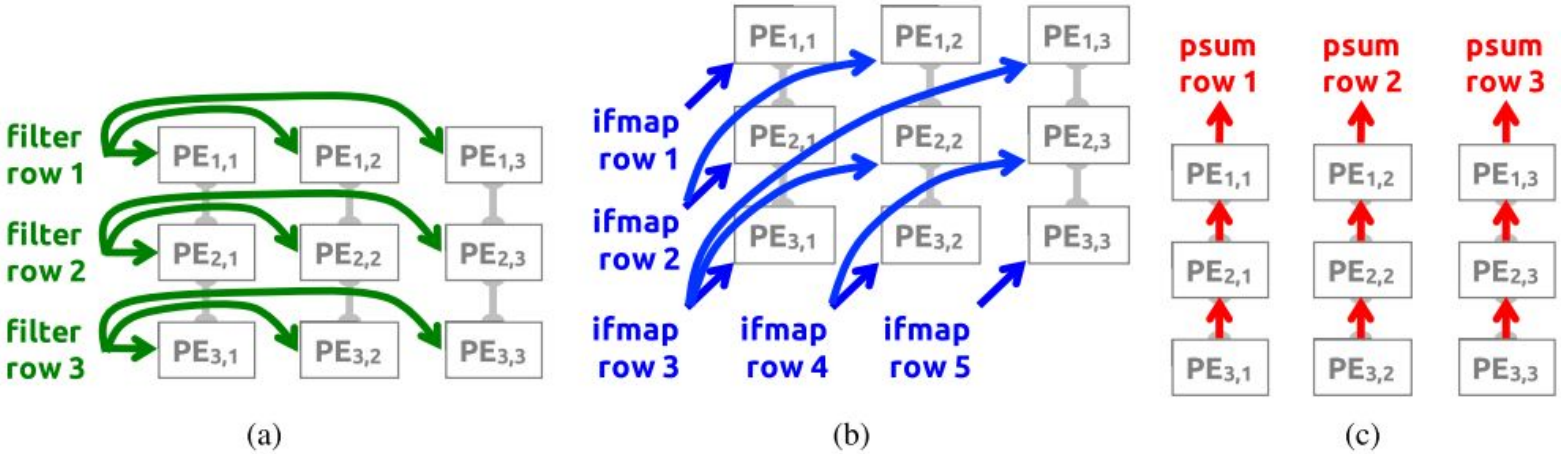
Chen, Tianshi, et al. "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning." *ACM SIGARCH Computer Architecture News* 42.1 (2014): 269-284.

Eyeriss



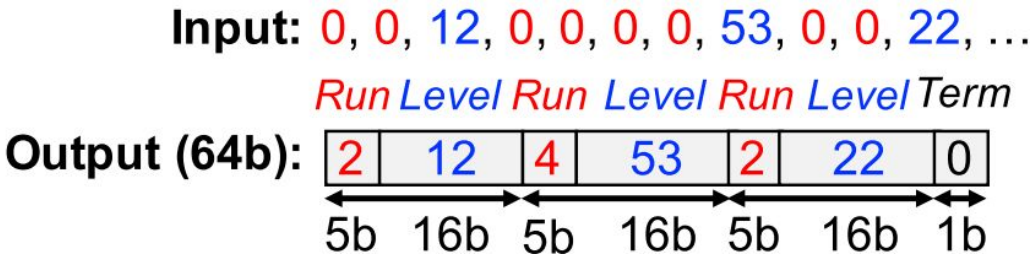
- Eyeriss optimizes for the energy efficiency of the entire system, including the accelerator chip and off-chip DRAM, for various CNN shapes by reconfiguring the architecture.
- The core clock domain consists of a spatial array of 168 PEs organized as a 12 × 14 rectangle, a 108-kB GLB, an RLC CODEC,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

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.

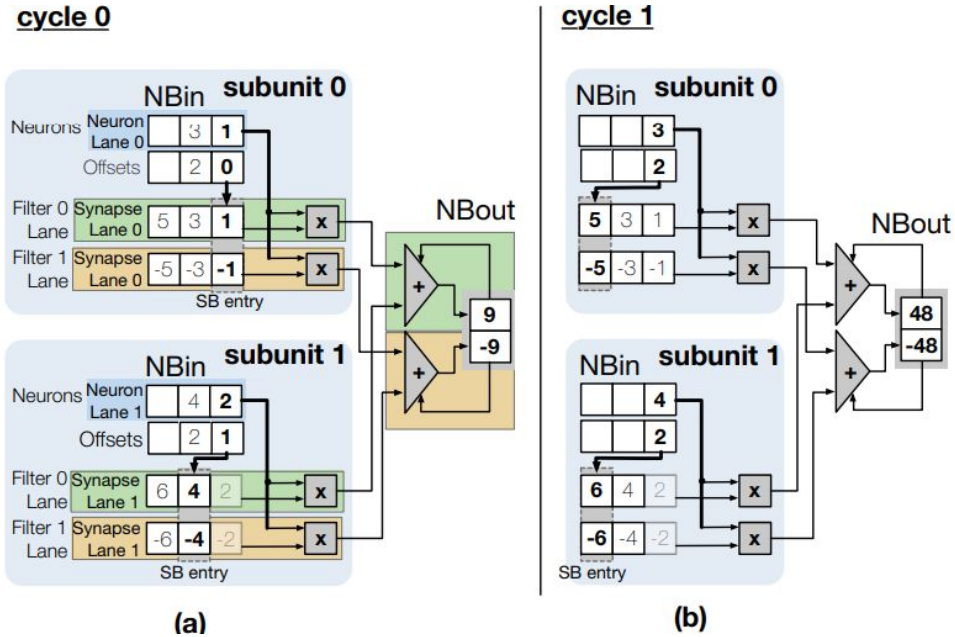
Rerun-length encoding



- 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.