Hardware for Deep Learning

Muhammad Husnain Mubarik

3rd year PhD candidate @ ECE UIUC

Advised by: Prof. Rakesh Kumar

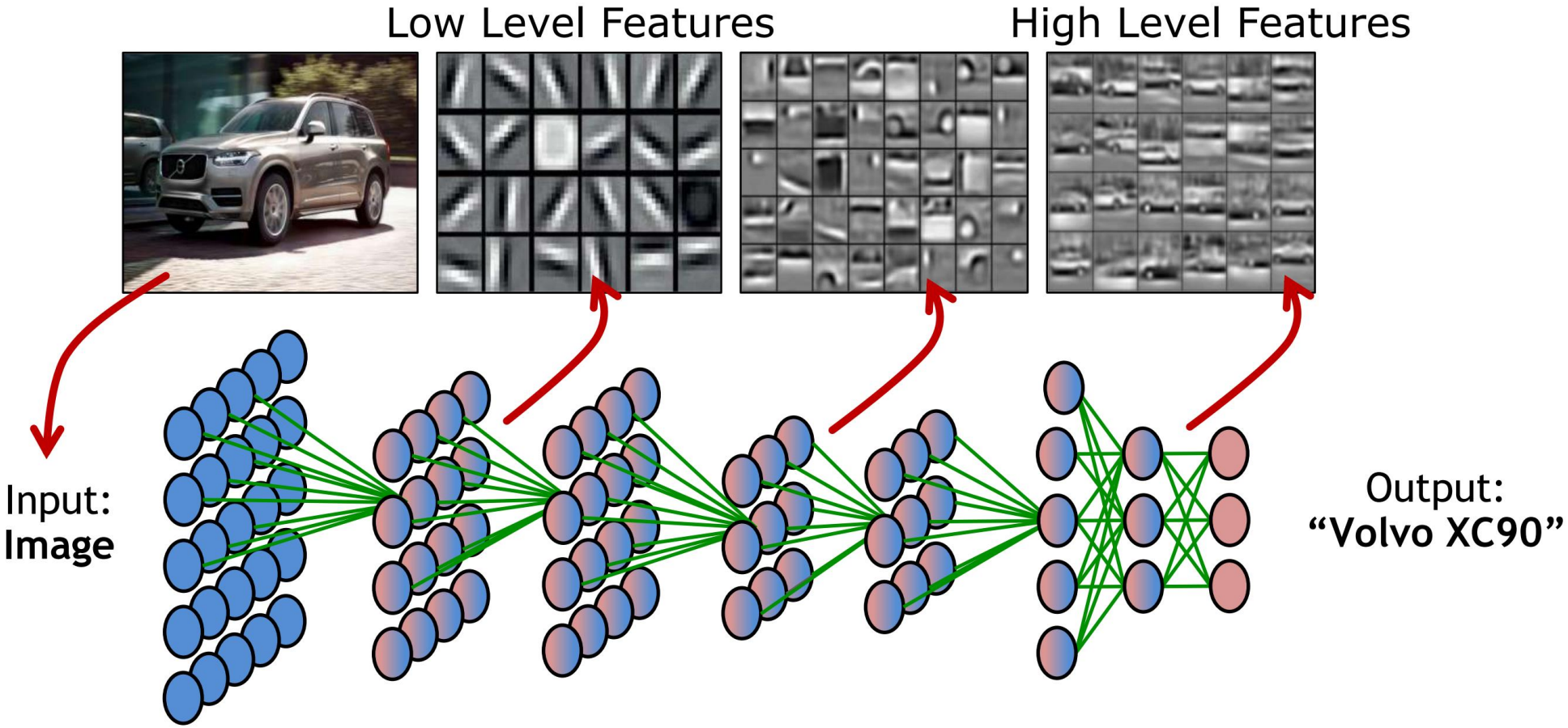
Goals of this Talk

- Many approaches for efficient processing of DNNs. Too many to cover!
- We will focus on how to evaluate approaches for efficient processing of DNNs
  - What are the key questions to ask? When you read ML/HW papers.
- Specifically, we will discuss
  - What are the **key metrics** that should be measured and compared?
  - What are the **challenges** towards achieving these metrics?
  - What are the **design considerations** and tradeoffs?
- We will focus on inference, but many concepts covered also apply to training

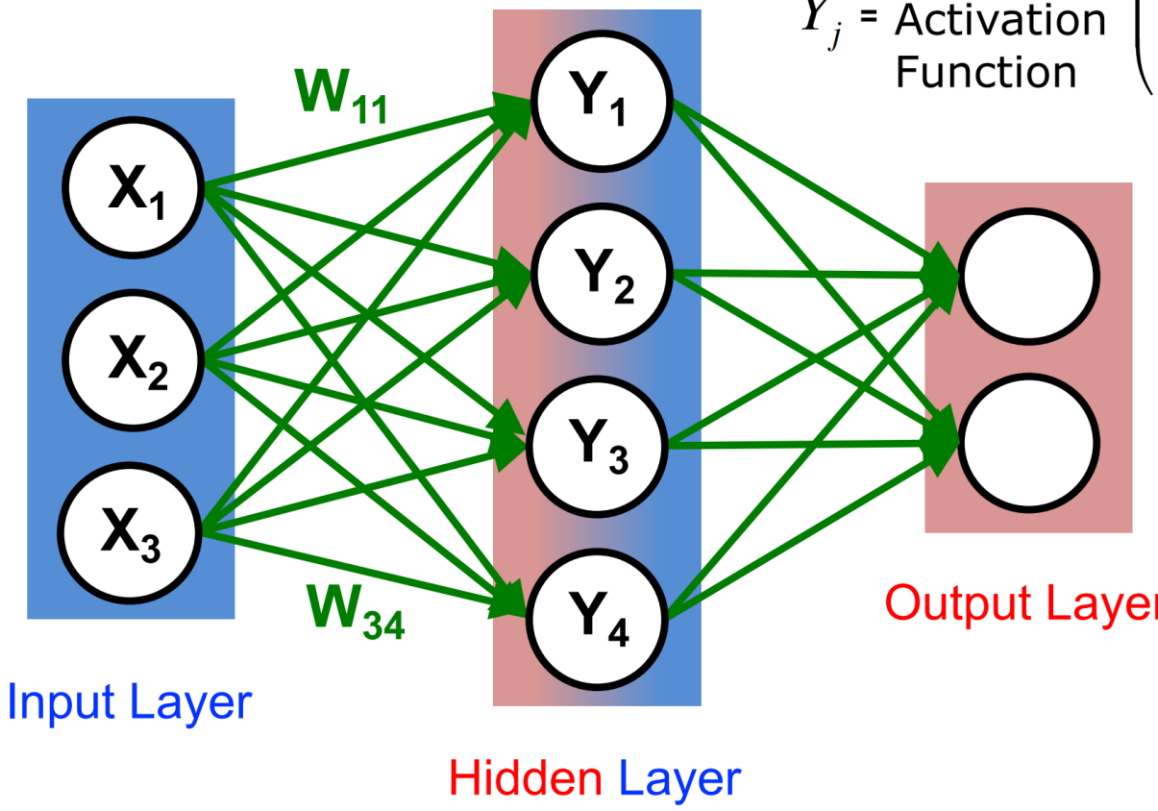
Overview

- Deep Neural Networks Overview (Terminology)
- Key Metrics and Design Objectives
- Design Considerations
  - CPU and GPU Platforms
  - Specialized / Domain-Specific Hardware (ASICs)
  - Algorithm (DNN Model) and Hardware Co-Design
  - Other Platforms
- Optimizations – Quantization, Sparsity, Pruning
- Summary

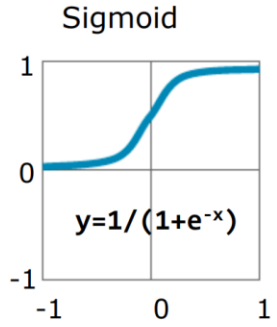
What are Deep Neural Networks?



Weighted Sums



$$Y_j = \text{Nonlinear Activation Function}\left(\sum_{i=1}^{3} W_{ij} \times X_i\right)$$



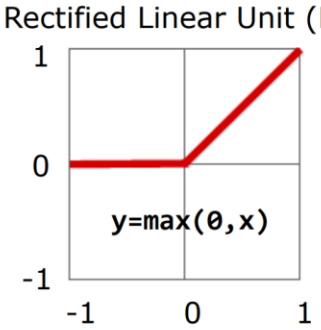


Image source: Caffe tutorial

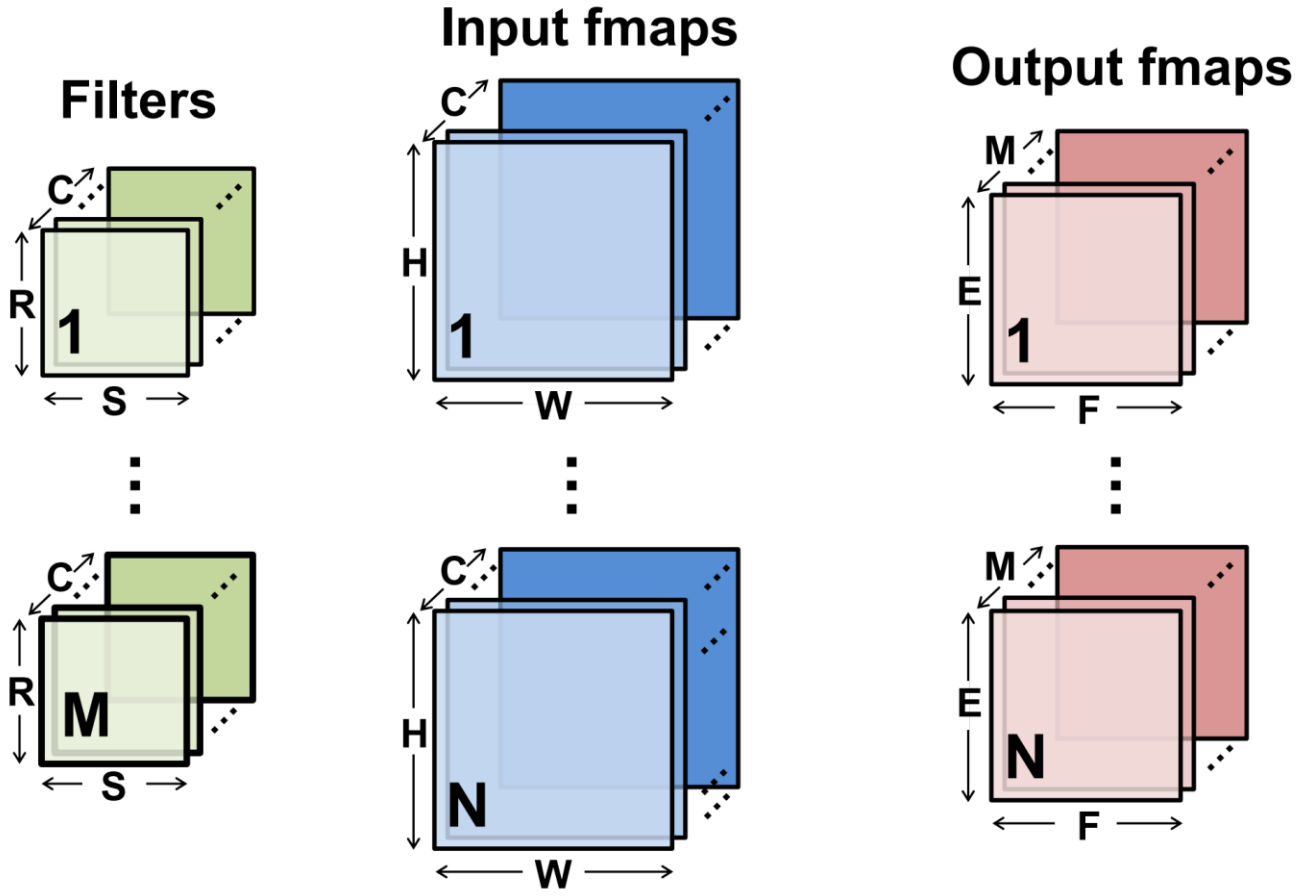
Key operation is
multiply and accumulate (MAC)
Accounts for > 90% of computation

Popular Types of Layers in DNNs

- Fully Connected Layer
  - Feed forward, dense layers
  - Multilayer Perceptron (MLP)
- Convolutional Layer
  - Feed forward, sparsely-connected w/ weight sharing
  - Convolutional Neural Network (CNN)
  - Typically used for images
- Recurrent Layer
  - Feedback
  - Recurrent Neural Network (RNN)
  - Typically used for sequential data (e.g., speech, language)
- Attention Layer/Mechanism
  - Attention (matrix multiply) + feed forward, fully connected
  - Transformer [Vaswani, NeurIPS 2017]

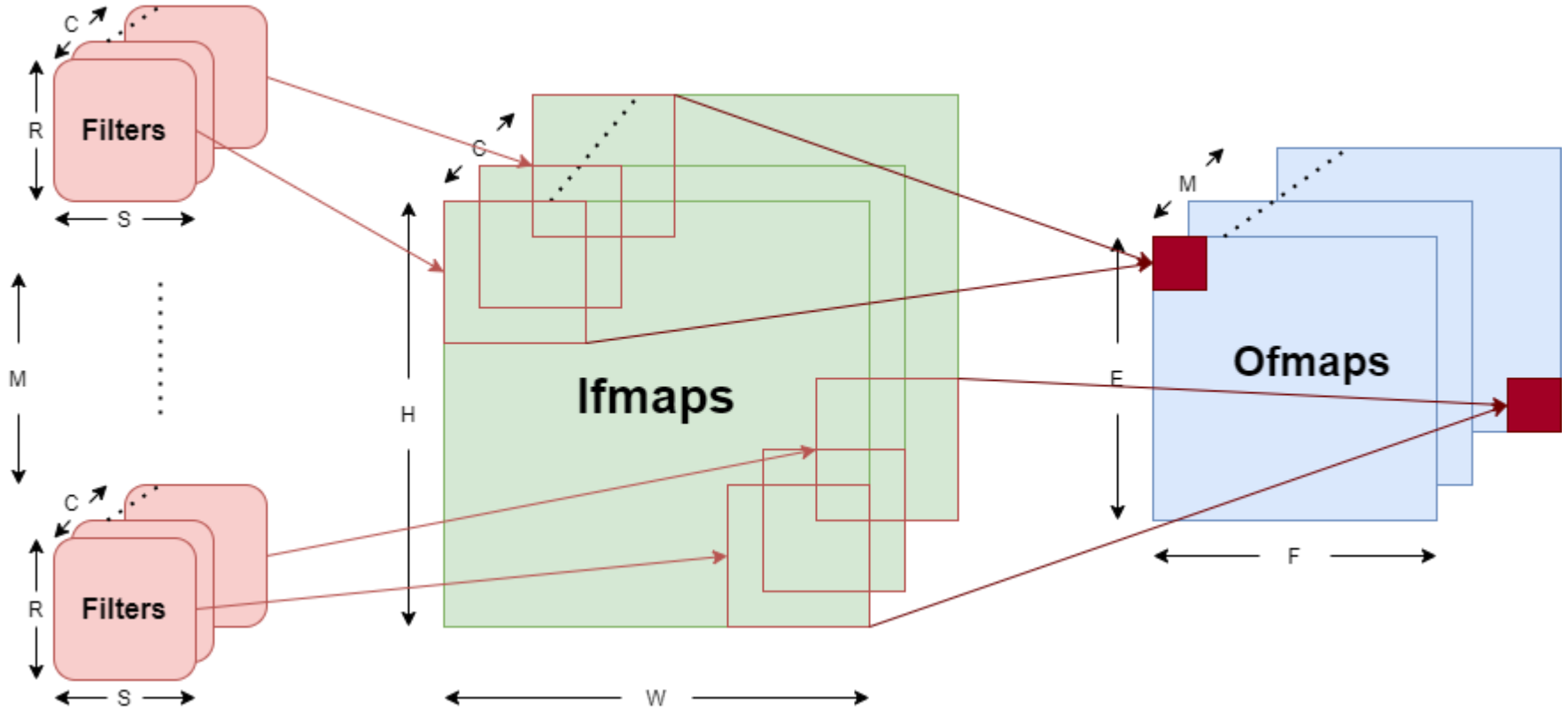
Convolution in CNNs

H – Height of input fmap (activations)
W – Width of input fmap (activations)
C – Number of 2-D input fmaps /filters (channels)
R – Height of 2-D filter (weights)
S – Width of 2-D filter (weights)
M – Number of 2-D output fmaps (channels)
E – Height of output fmap (activations)
F – Width of output fmap (activations)
N – Number of input fmaps/output fmaps (batch size)



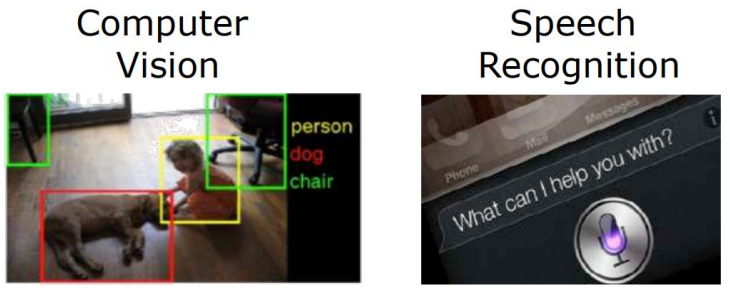
Convolution in CNNs

- Element-wise multiplication
- Partial sum accumulation
- Sliding window processing



Key Metrics: Much more than OPS/W!

- Accuracy
  - Quality of result
- Throughput
  - Analytics on high volume data
  - Real-time performance (e.g., video at 30 fps)
- Latency
  - For interactive applications (e.g., autonomous navigation)
- Energy and Power
  - Embedded devices have limited battery capacity
  - Data centers have a power ceiling due to cooling cost
- Hardware Cost
  - $$$
- Flexibility
  - Range of DNN models and tasks
- Scalability
  - Scaling of performance with amount of resources

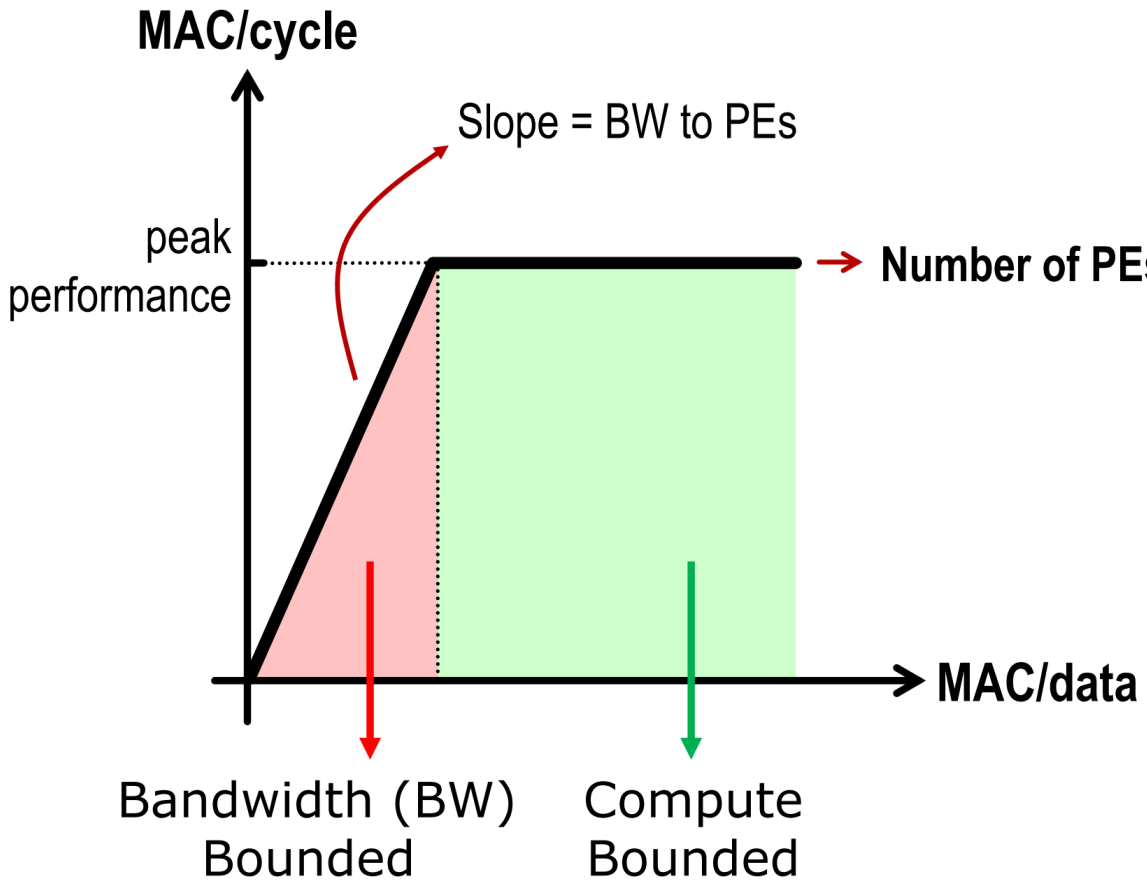


[**Sze**, *CICC* 2017]

Key Design Objectives of DNN Processor

- **Increase Throughput and Reduce Latency**
  - Reduce time per MAC
    - Reduce critical path -> increase clock frequency
    - Reduce instruction overhead
  - Avoid unnecessary MACs (save cycles)
  - Increase number of processing elements (PE) -> more MACs in parallel
    - Increase area density of PE or area cost of system
  - Increase PE utilization -> keep PEs busy
    - Distribute workload to as many PEs as possible
    - Balance the workload across PEs
    - Sufficient memory bandwidth to deliver workload to PEs (reduce idle cycles)

Eyexam: Performance Evaluation Framework



Based on Roofline Model

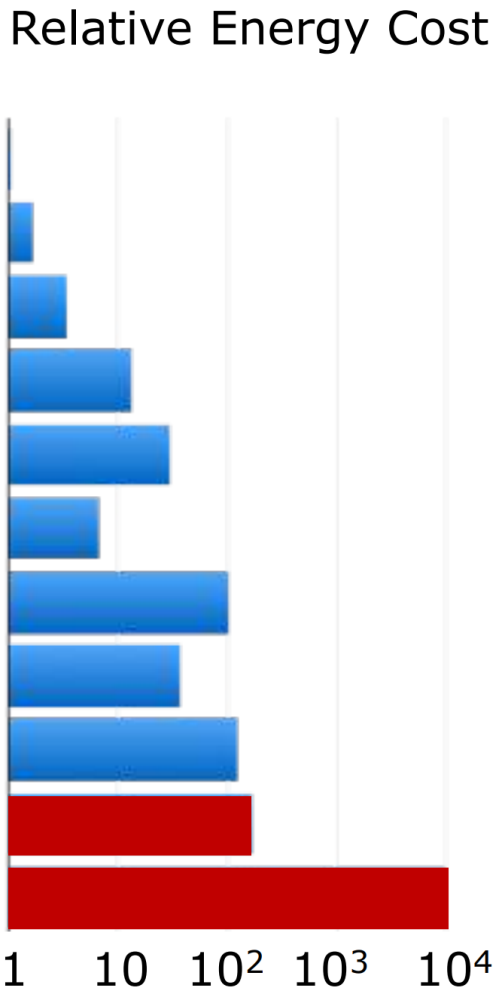
[**Williams**, *CACM* 2009]

Key Design Objectives of DNN Processor

- Reduce Energy and Power Consumption
  - Reduce data movement as it dominates energy consumption
    - Exploit data reuse
  - Reduce energy per MAC
    - Reduce switching activity and/or capacitance
    - Reduce instruction overhead
  - Avoid unnecessary MACs
- Power consumption is limited by heat dissipation, which limits the maximum # of MACs in parallel (i.e., throughput)

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

[**Horowitz**, *ISSCC* 2014]

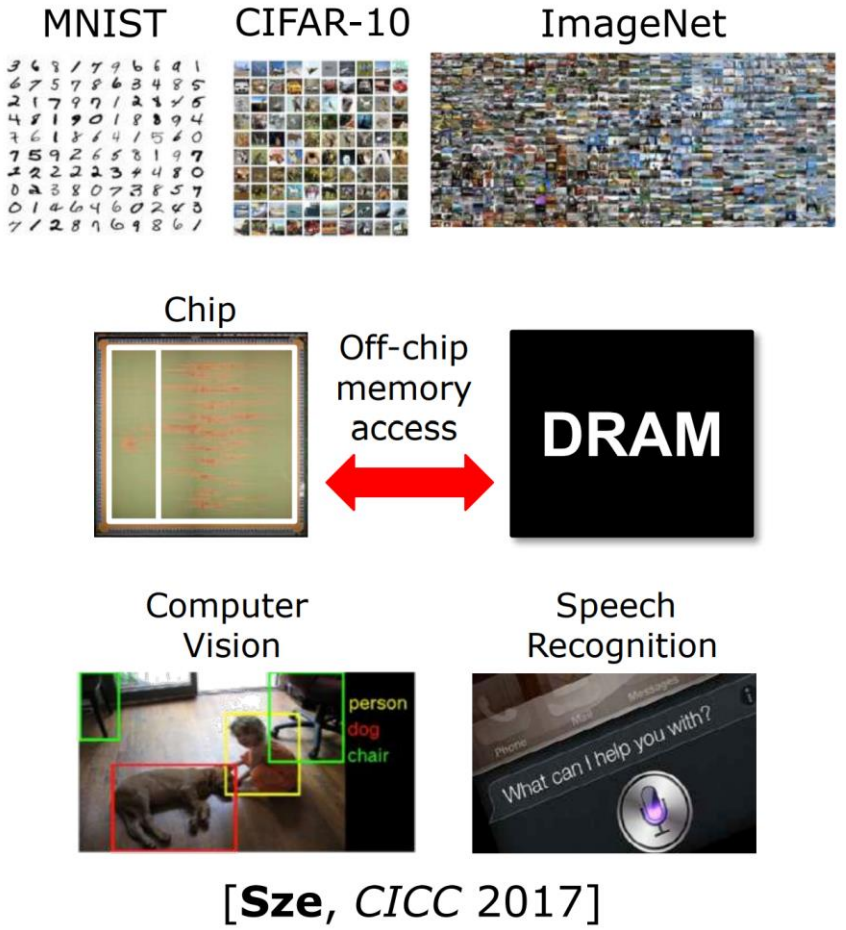


Key Design Objectives of DNN Processor

- Flexibility
  - Reduce overhead of supporting flexibility
  - Maintain efficiency across wide range of DNN models
    - Different layer shapes impact the amount of
      - Required storage and compute
      - Available data reuse that can be exploited
    - Different precision across layers & data types (weight, activation, partial sum)
    - Different degrees of sparsity (number of zeros in weights or activations)
    - Types of DNN layers and computation beyond MACs (e.g., activation functions)
- Scalability
  - How does performance (i.e., throughput, latency, energy, power) scales with increase in amount of resources (e.g., number of PEs, amount of memory, etc.)

Specifications to Evaluate Metrics

- Accuracy
  - Difficulty of dataset and/or task should be considered
  - Difficult tasks typically require more complex DNN models
- Throughput
  - Number of PEs with utilization (not just peak performance)
  - Runtime for running specific DNN models
- Latency
  - Batch size used in evaluation
- Energy and Power
  - Power consumption for running specific DNN models
  - Off-chip memory access (e.g., DRAM)
- Hardware Cost
  - On-chip storage, # of PEs, chip area + process technology
- Flexibility
  - Report performance across a wide range of DNN models
  - Define range of DNN models that are efficiently supported



[**Sze**, *CICC* 2017]

Comprehensive Coverage for Evaluation

- All metrics should be reported for fair evaluation of design tradeoffs
- Examples of what can happen if a certain metric is omitted:
  - Without the accuracy given for a specific dataset and task, one could run a simple DNN and claim low power, high throughput, and low cost – however, the processor might not be usable for a meaningful task
  - Without reporting the off-chip memory access, one could build a processor with only MACs and claim low cost, high throughput, high accuracy, and low chip power – however, when evaluating system power, the off-chip memory access would be substantial
- Are results measured or simulated? On what test data?

Example Evaluation Process

- The evaluation process for whether a DNN processor is a viable solution for a given application might go as follows:
  - Accuracy determines if it can perform the given task
  - Latency and throughput determine if it can run fast enough and in real-time
  - Energy and power consumption will primarily dictate the form factor of the device where the processing can operate
  - Cost, which is primarily dictated by the chip area, determines how much one would pay for this solution
  - Flexibility determines the range of tasks it can support

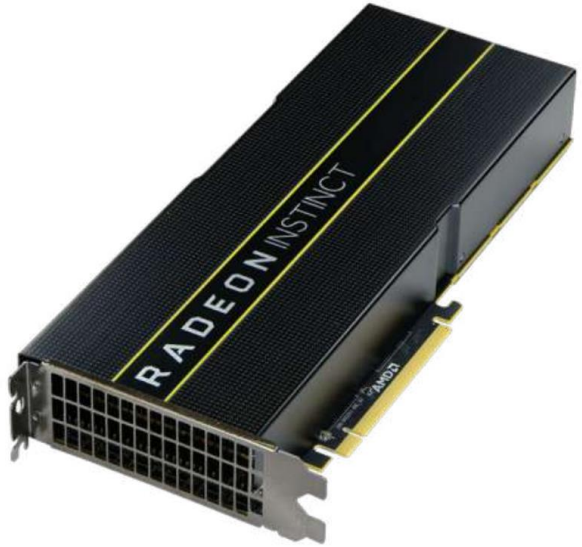
CPUs and GPUs Targeting DNNs

- Use matrix multiplication libraries on CPUs and GPUs

Intel Xeon (Cascade Lake)

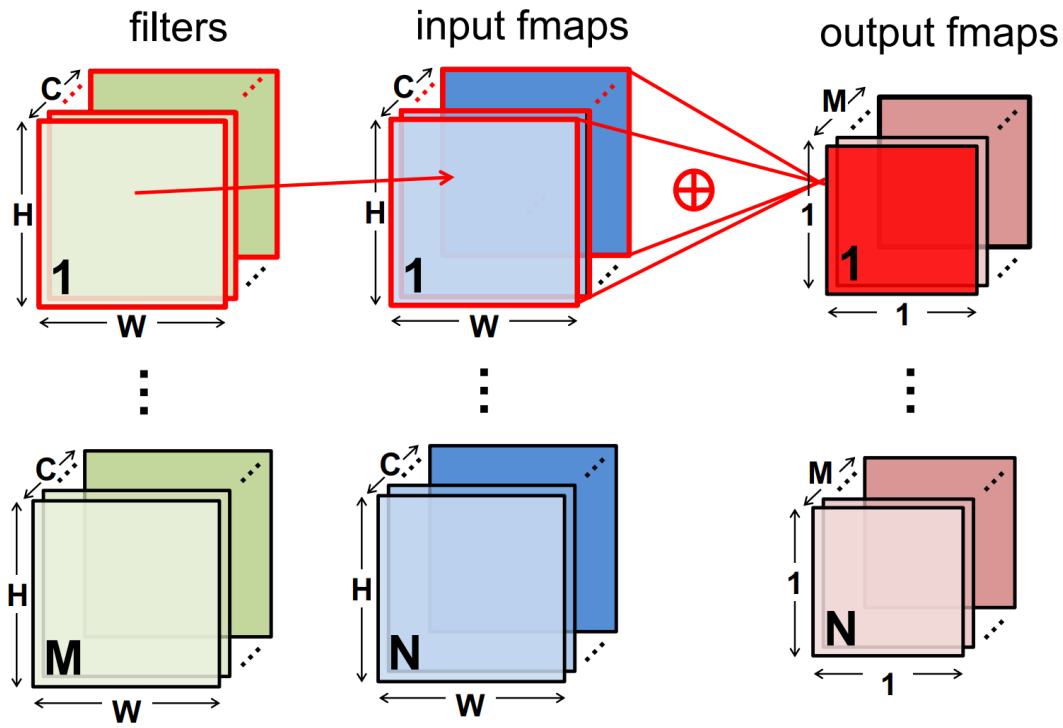
Nvidia Tesla (Volta)

AMD Radeon (Instinct)

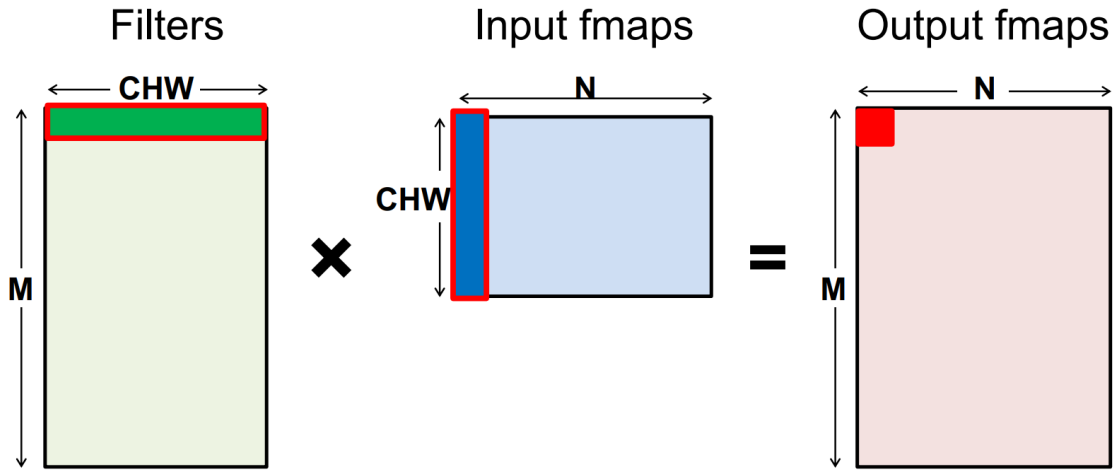


Map DNN to a Matrix Multiplication

- Fully connected layer can be directly represented as matrix multiplication



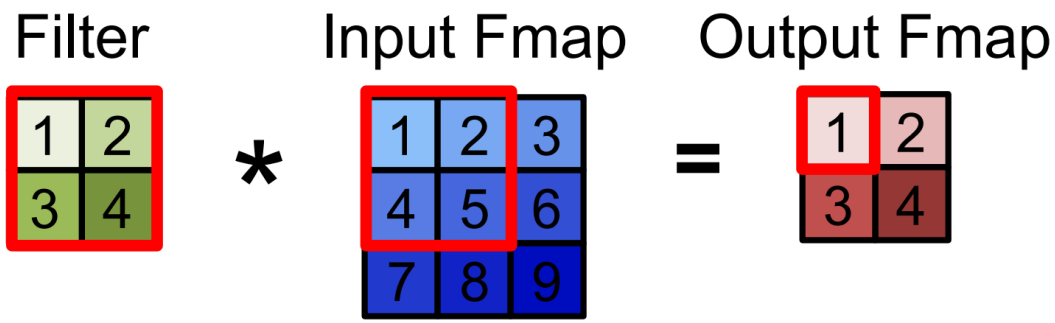
In fully connected layer, filter size (R, S) same as input size (H, W)



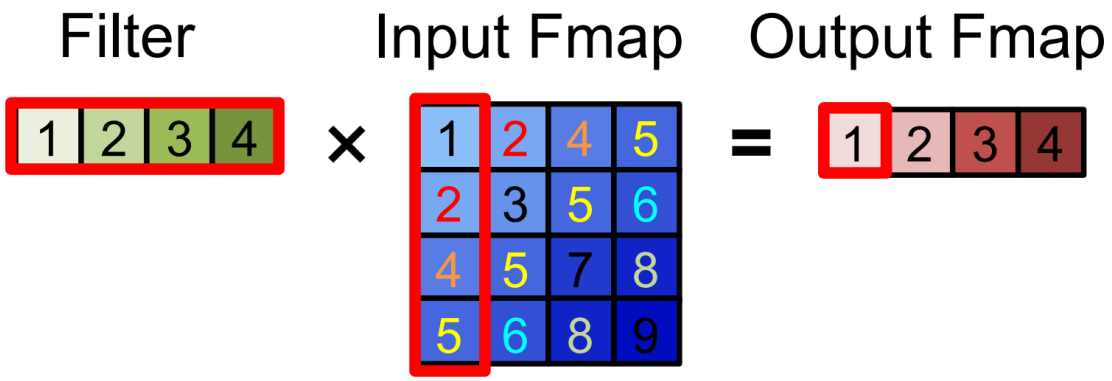
Note: Matrix multiplication also heavily used by recurrent and attention layers

Map DNN to a Matrix Multiplication

- Convolutional layer can be converted to Toeplitz Matrix



Convolution



Matrix Multiply (by Toeplitz Matrix)

Data is repeated

CPU, GPU Libraries for Matrix Multiplication

- Implementation: Matrix Multiplication (GEMM)
  - CPU: OpenBLAS, Intel MKL, etc
  - GPU: cuBLAS, cuDNN, etc
- Library will note shape of the matrix multiply and select implementation optimized for that shape
- Optimization usually involves proper tiling to memory hierarchy

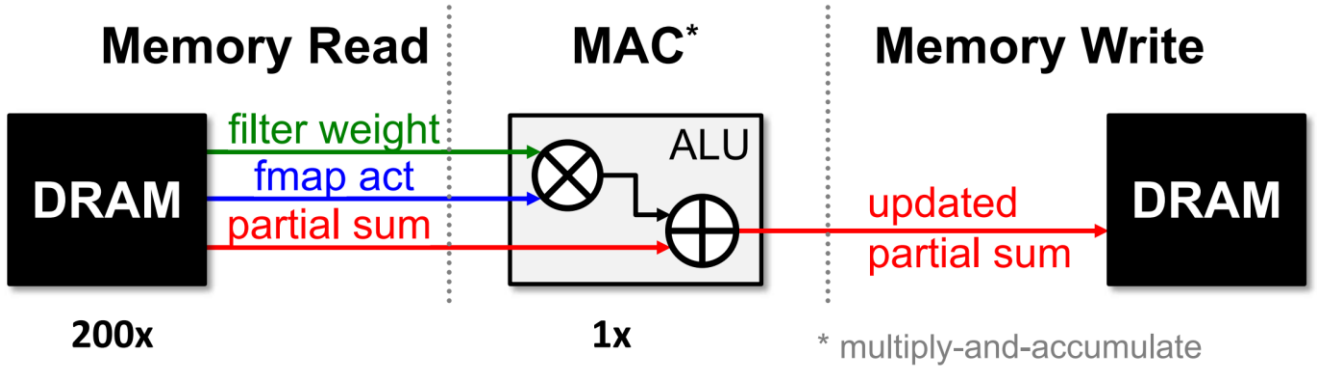
Design Considerations for CPU and GPU

- Software (compiler)
  - Reduce unnecessary MACs: Apply transforms
  - Increase PE utilization: Schedule loop order and tile data to increase data reuse in memory hierarchy
- Hardware
  - Reduce time per MAC
    - Increase speed of PEs
    - Increase MACs per instruction using large aggregate instructions (e.g., SIMD, tensor core) -> requires additional hardware
  - Increase number of parallel MACs
    - Increase number of PEs on chip -> area cost
    - Support reduced precision in PEs
  - Increase PE utilization
    - Increase on-chip storage -> area cost
    - External memory BW -> system cost

Specialized / Domain-Specific Hardware

Properties We Can Leverage

- Operations exhibit high parallelism
  - high throughput possible
- Memory Access is the Bottleneck

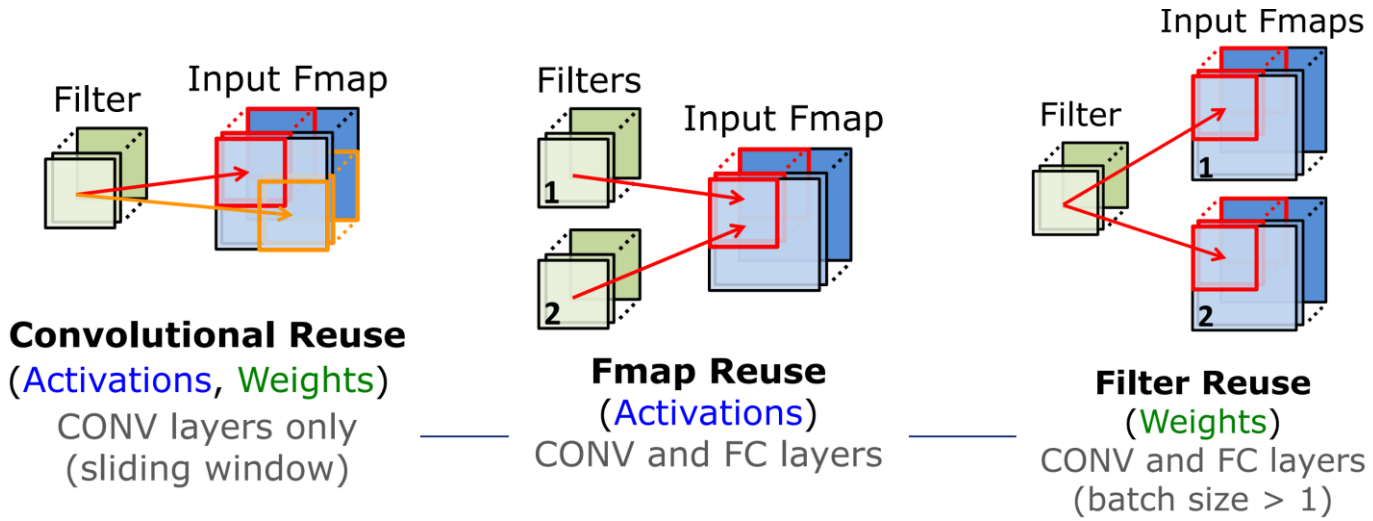


Worst Case: all memory R/W are **DRAM** accesses

Example: AlexNet has **724M** MACs → **2896M** DRAM accesses required

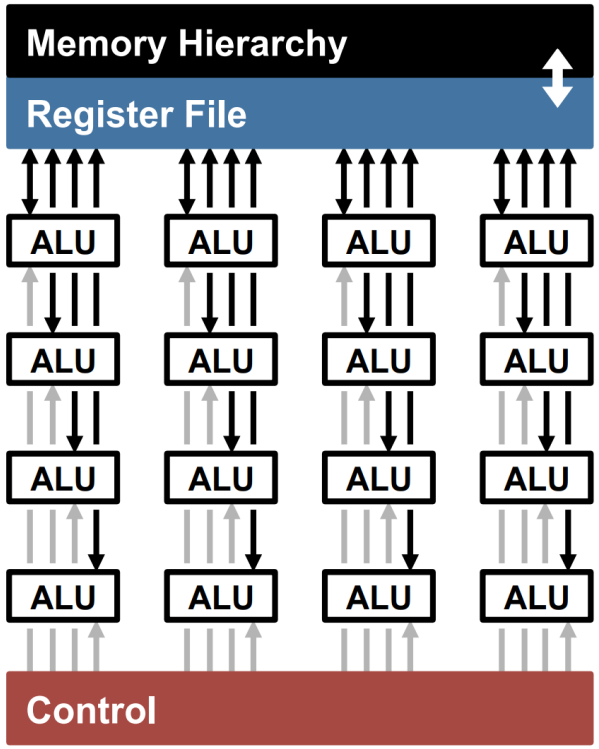
Properties We Can Leverage

- Operations exhibit high parallelism
  - high throughput possible
- Input data reuse opportunities (e.g., up to 500x for AlexNet)
  - exploit low-cost memory

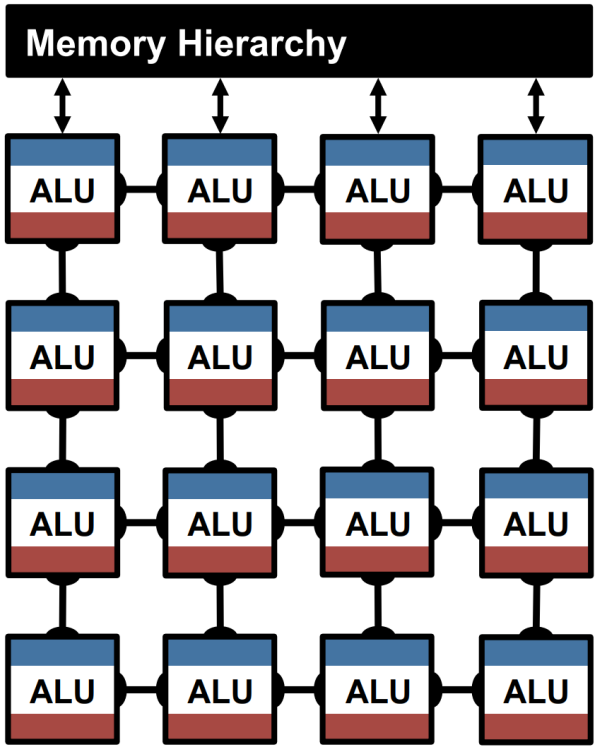


Highly-Parallel Compute Paradigms

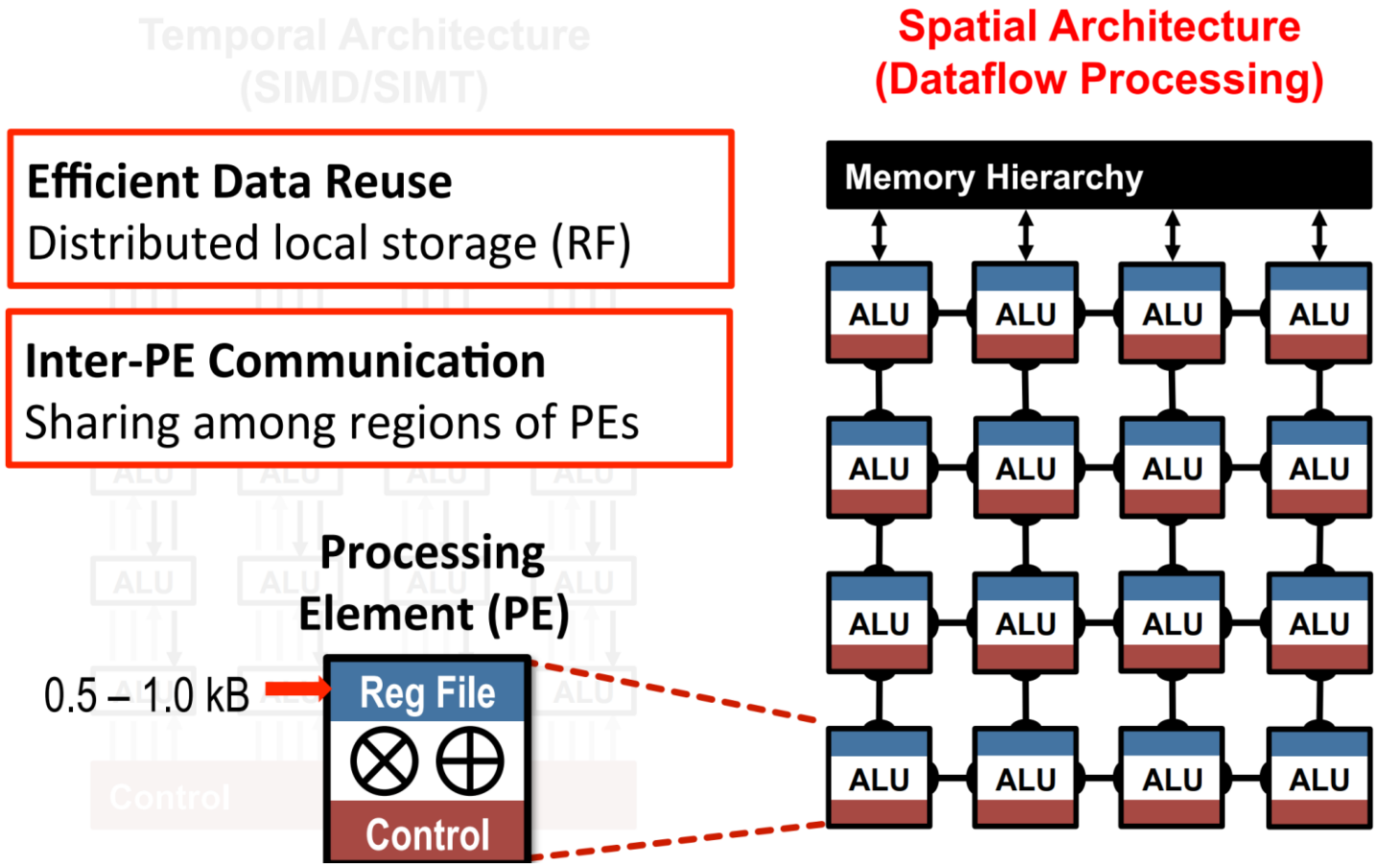
Temporal Architecture (SIMD/SIMT)



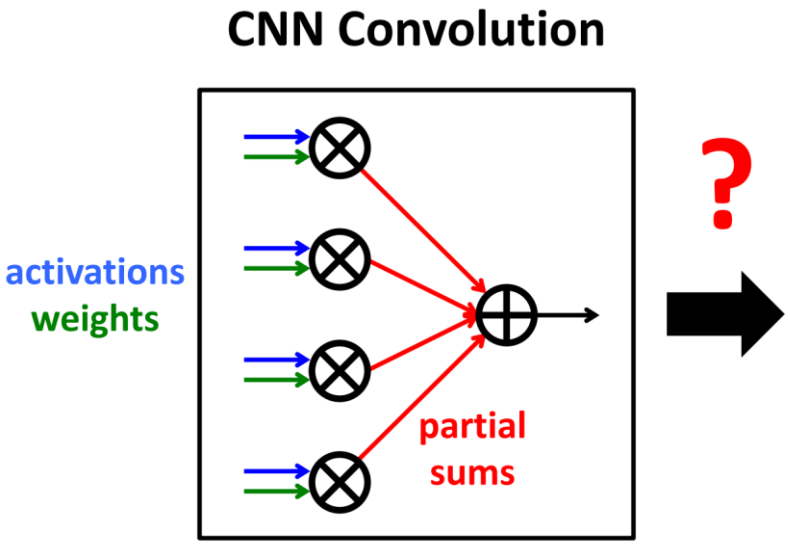
Spatial Architecture (Dataflow Processing)



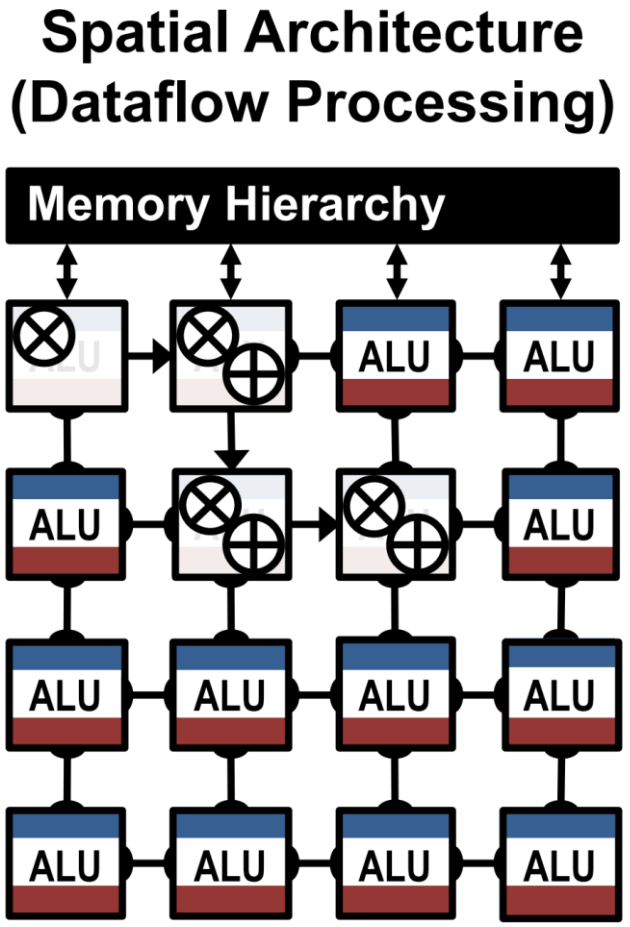
Advantages of Spatial Architecture



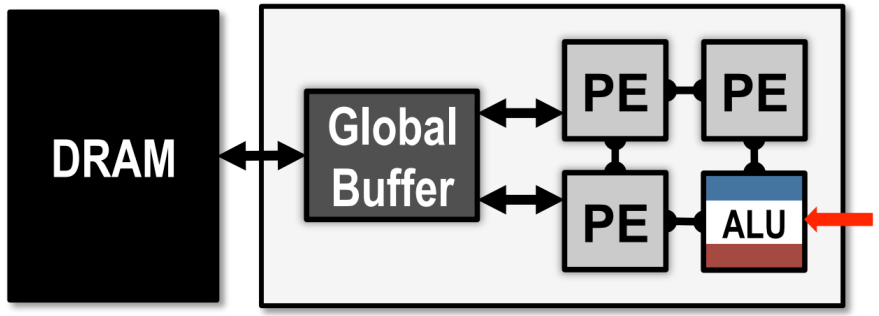
How to Map the Dataflow?



Goal: Increase reuse of input data (weights and activations) and local partial sums accumulation

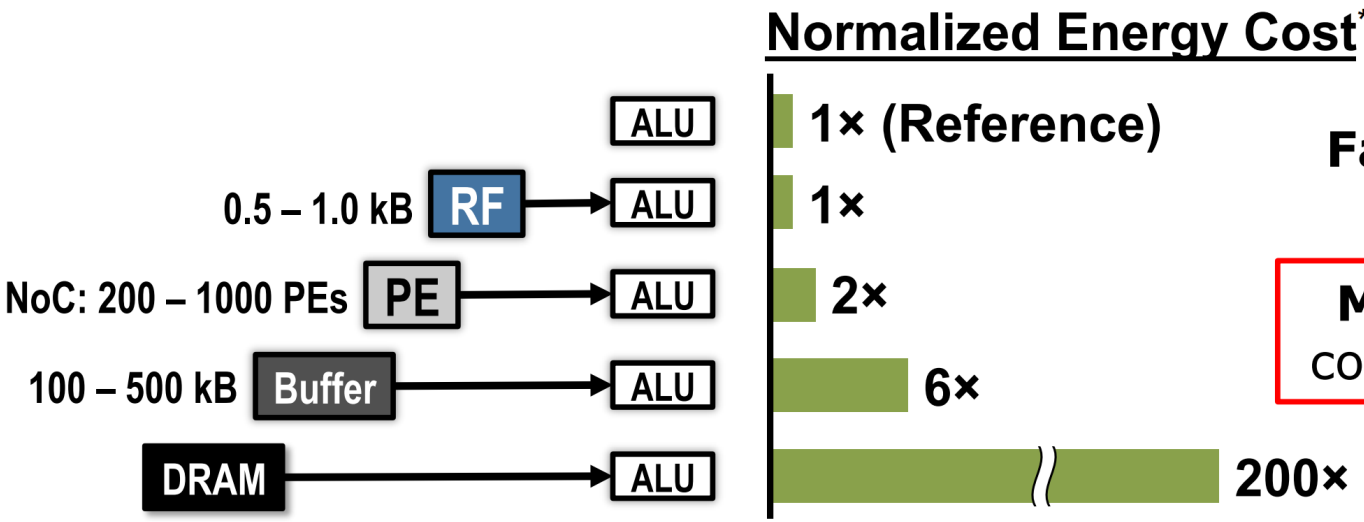


Data Movement is Expensive



Specialized hardware with small (< 1kB) low cost memory near compute

fetch data to run a MAC here

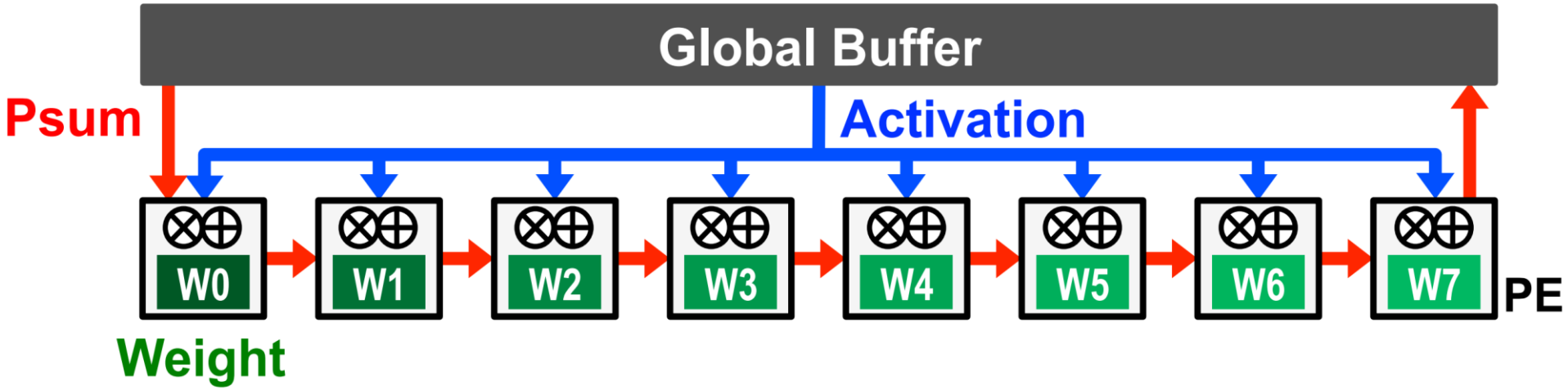


Farther and **larger** memories consume more power

Maximize data reuse at low cost levels of memory hierarchy

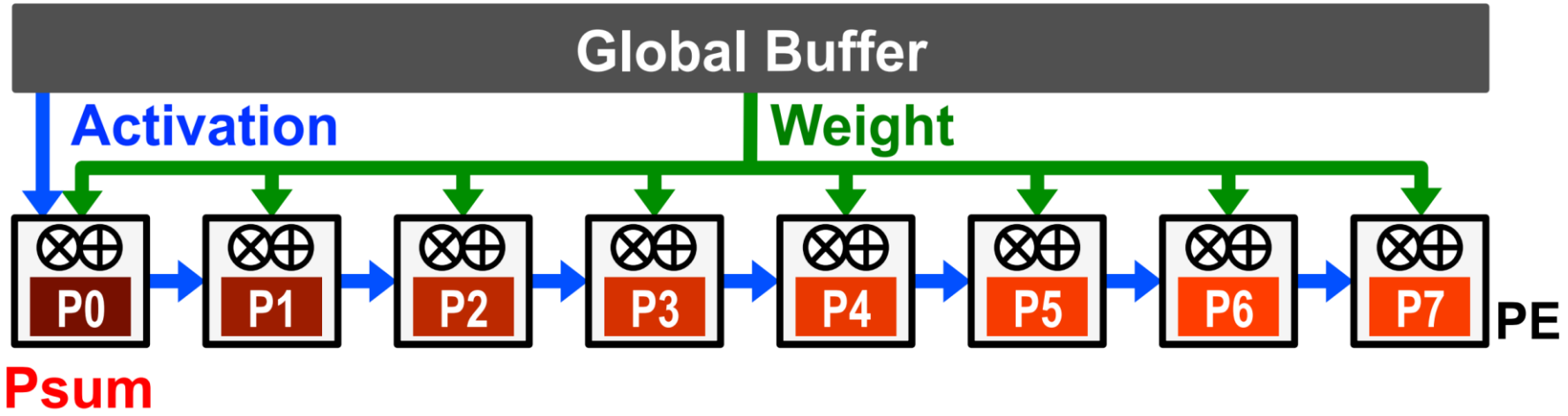
Weight Stationary (WS)

- Minimize weight read energy consumption
  - maximize convolutional and filter reuse of weights
- Broadcast activations and accumulate partial sums spatially across the PE array
- Examples: TPU [Jouppi, ISCA 2017], NVDLA



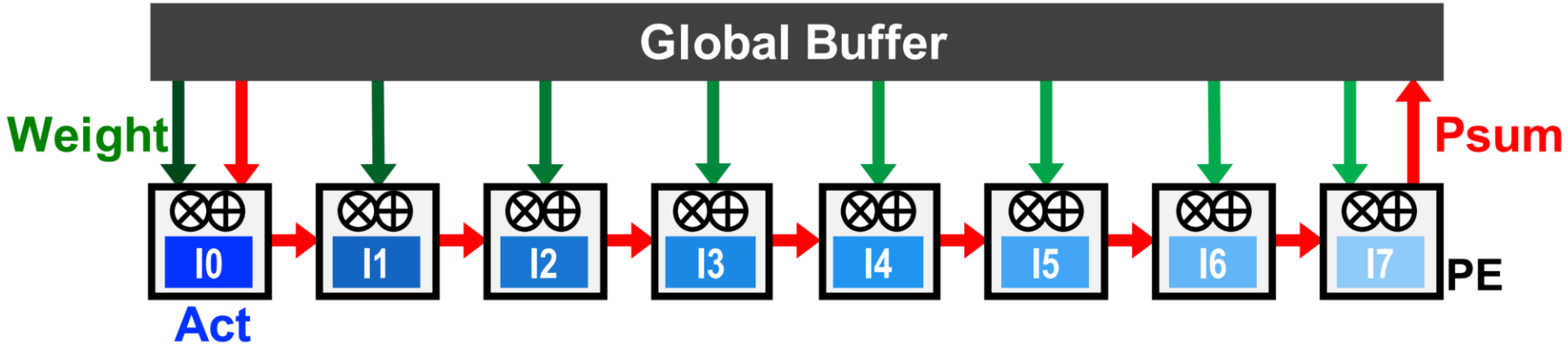
Output Stationary (OS)

- Minimize partial sum R/W energy consumption
  - maximize local accumulation
- Broadcast/Multicast filter weights and reuse activations spatially across the PE array
- Examples: [Moons, VLSI 2016], [Thinker, VLSI 2017]



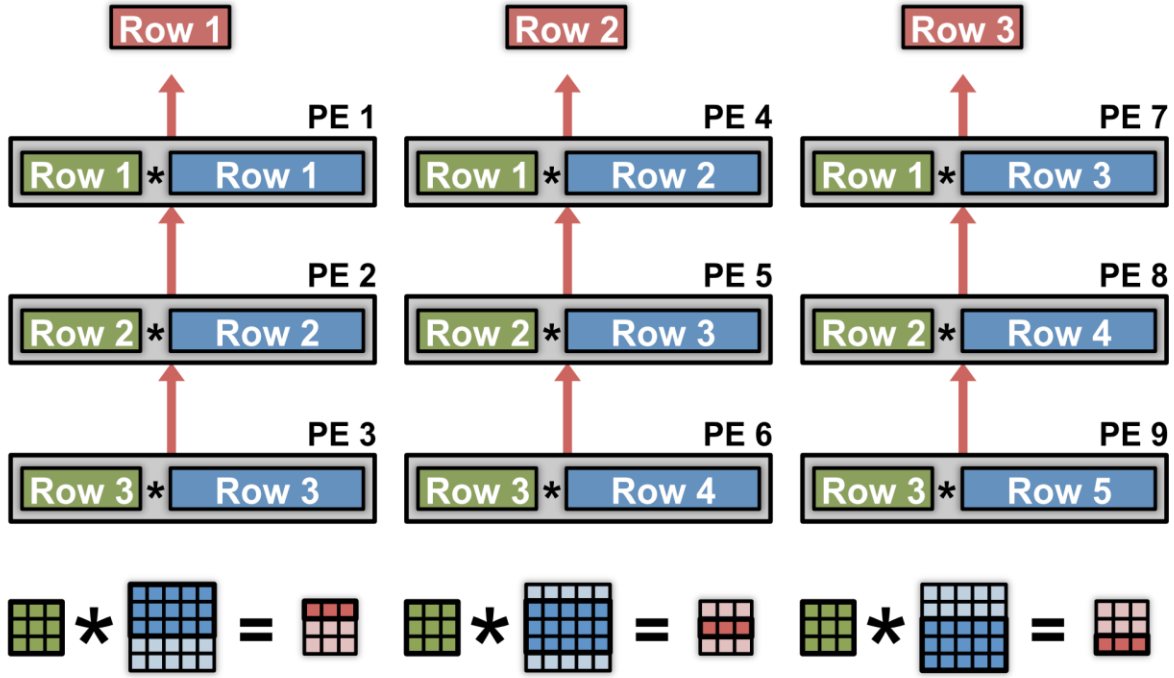
Input Stationary (IS)

- Minimize activation read energy consumption
  - maximize convolutional and fmap reuse of activations
- Unicast weights and accumulate partial sums spatially across the PE array
- Example: [SCNN, ISCA 2017]



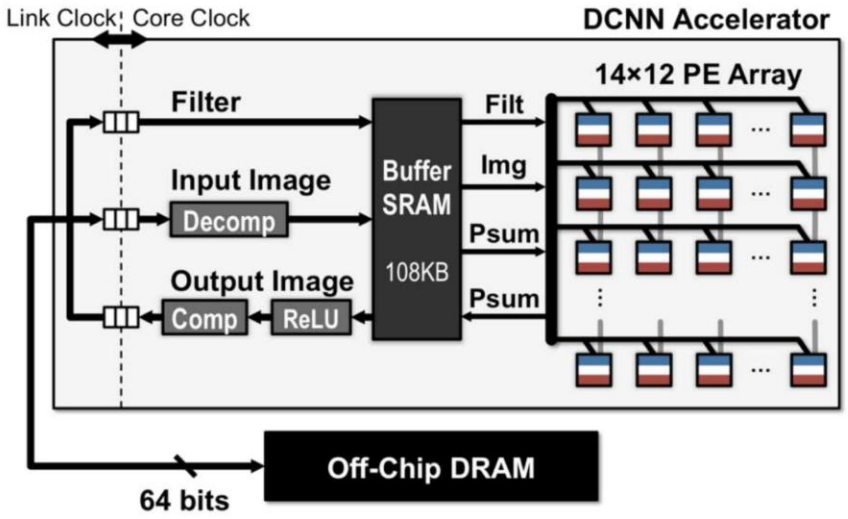
Row Stationary Dataflow

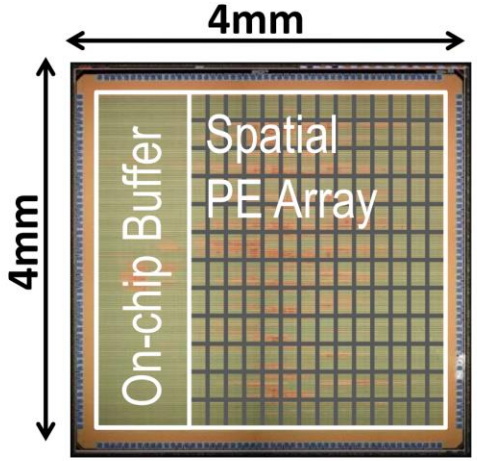
- Maximize row convolutional reuse in RF
  - Keep a filter row and fmap sliding window in RF
- Maximize row psum accumulation in RF
- Optimize for overall energy efficiency instead for only a certain data type



Eyeriss: Deep Neural Network Accelerator On

- Exploits data reuse for 100x reduction in memory accesses from global buffer and 1400x reduction in memory accesses from off-chip DRAM
- Overall >10x energy reduction compared to a mobile GPU (Nvidia TK1)





[**Chen**, *ISSCC* 2016]

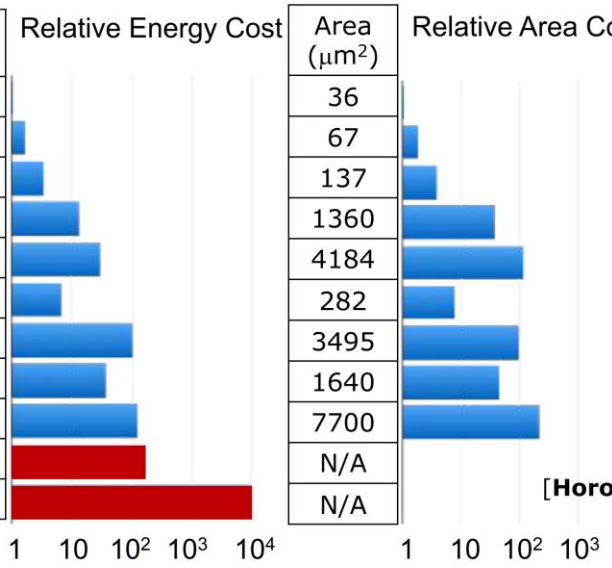
Algorithm & Hardware Co-Design

- Co-design algorithm + hardware -> better than what each could achieve alone
- Co-design approaches can be loosely grouped into two categories:
  - Reduce size of operands for storage/compute (Reduced Precision)
  - Reduce number of operations for storage/compute (Sparsity and Efficient Network Architecture)
- Hardware support required to increase savings in latency and energy
  - Ensure that overhead of hardware support does not exceed benefits
- Unlike previously discussed approaches, these approaches can affect accuracy!
  - Evaluate tradeoff between accuracy and other metrics

Optimizations – Reduced Precision

- Reduce data movement and storage cost for inputs and outputs of MAC
  - Smaller memory -> lower energy
- Reduce cost of MAC
  - Cost of multiply increases with bit width (n) -> energy and area by O(n2); delay by O(n)

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A



Sparsity / Pruning

- Reduce number of MACs
  - Anything multiplied by zero is zero -> avoid performing unnecessary MACs
  - Reduce energy consumption and latency
- Reduce data movement
  - If one of the inputs to MAC is zero, can avoid reading the other input
  - Compress data by only sending non-zero values
- Example: AlexNet weight reduction by pruning
  - CONV layers 2.7x; FC layers 9.9x
  - Overall Reduction: Weights 9x, MACs 3x

Design Considerations for Sparsity

- Impact on accuracy
  - Must consider difficulty of dataset, task, and DNN model
    - e.g., AlexNet and VGG known to be over parameterized and thus easy to prune weights; does method work on efficient DNN models?
- Does hardware cost exceed benefits?
  - Need extra hardware to identify sparsity
    - e.g., Additional logic to identify non-zeros and store non-zero locations
  - Accounting for sparsity in both weights and activations is challenging
    - Need to compute intersection of two data streams rather than find next non-zero in one
  - Compressed data will be variable length
    - Reduced flexibility in access order -> random access will have significant overhead

Design Considerations for Co-Design

- Impact on accuracy
  - Consider quality of baseline (initial) DNN model, difficulty of task and dataset
  - Sweep curve of accuracy versus latency/energy to see the full tradeoff
- Does hardware cost exceed benefits?
  - Need extra hardware to support variable precision and shapes or to identify sparsity
- Time required to perform co-design
  - e.g., Difficulty of tuning affected by
    - Number of hyperparameters
    - Uncertainty in relationship between hyperparameters and impact on performance
- How does the approach perform on different platforms?
  - Is the approach a general method, or applicable on specific hardware?

Design Considerations for ASIC

- Increase PE utilization
  - Flexible mapping and on-chip network for different DNN models -> requires additional hardware
- Reduce data movement
  - Custom memory hierarchy and dataflows that exploit data reuse
  - Apply compression to exploit redundancy in data -> requires additional hardware
- Reduce time and energy per MAC
  - Reduce precision -> if precision varies, requires additional hardware; impact on accuracy
- Reduce unnecessary MACs
  - Exploit sparsity -> requires additional hardware; impact on accuracy
  - Exploit redundant operations -> requires additional hardware

Summary

- DNNs are a critical component in the AI revolution, delivering record breaking accuracy on many important AI tasks for a wide range of applications; however, it comes at the cost of high computational complexity
- Efficient processing of DNNs is an important area of research with many promising opportunities for innovation at various levels of hardware design, including algorithm co-design
- When considering different DNN solutions it is important to evaluate with the appropriate workload in term of both input and model, and recognize that they are evolving rapidly
- It is important to consider a comprehensive set of metrics when evaluating different DNN solutions: accuracy, throughput, latency, power, energy, flexibility, scalability and cost