Energy-Efficient Deep Learning: Challenges and Opportunities

Vivienne Sze

Massachusetts Institute of Technology

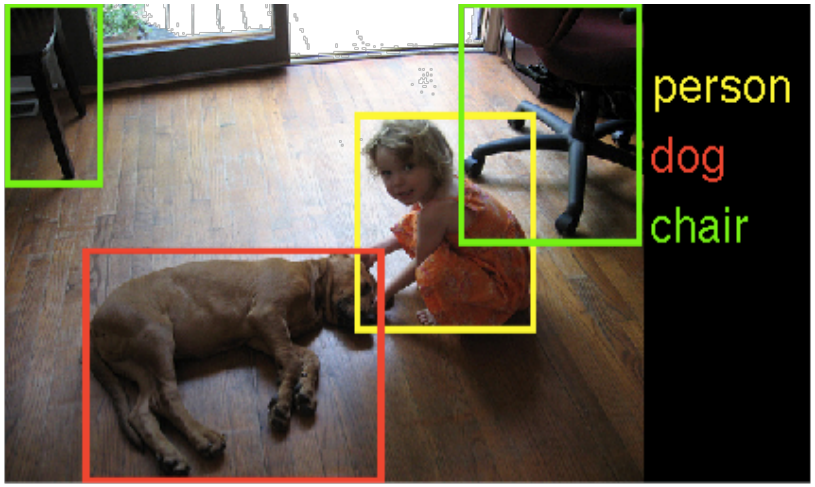
In collaboration with

Yu-Hsin Chen, Joel Emer, Tien-Ju Yang

Contact Info
email: sze@mit.edu
website: www.rle.mit.edu/eems

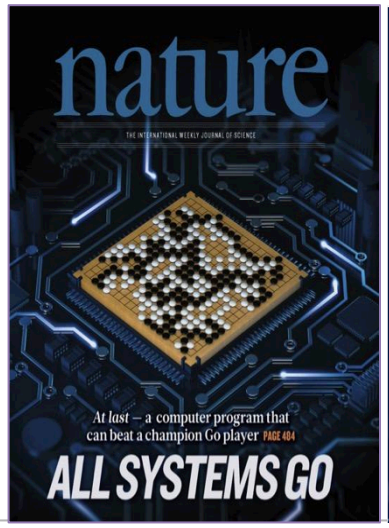
Example Applications of Deep Learning

Computer Vision

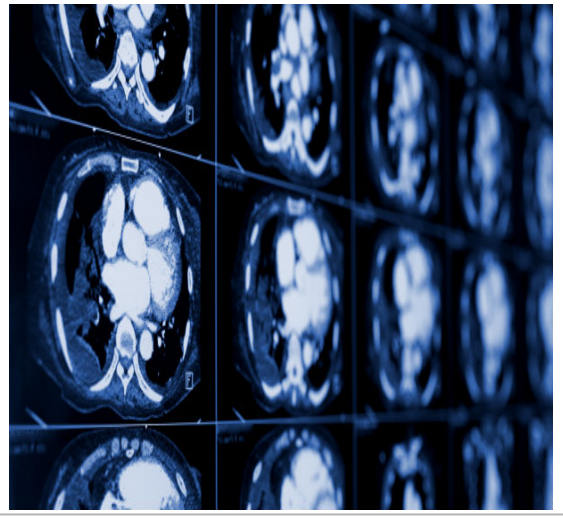


Speech Recognition

Game Play



Medical



What is Deep Learning?

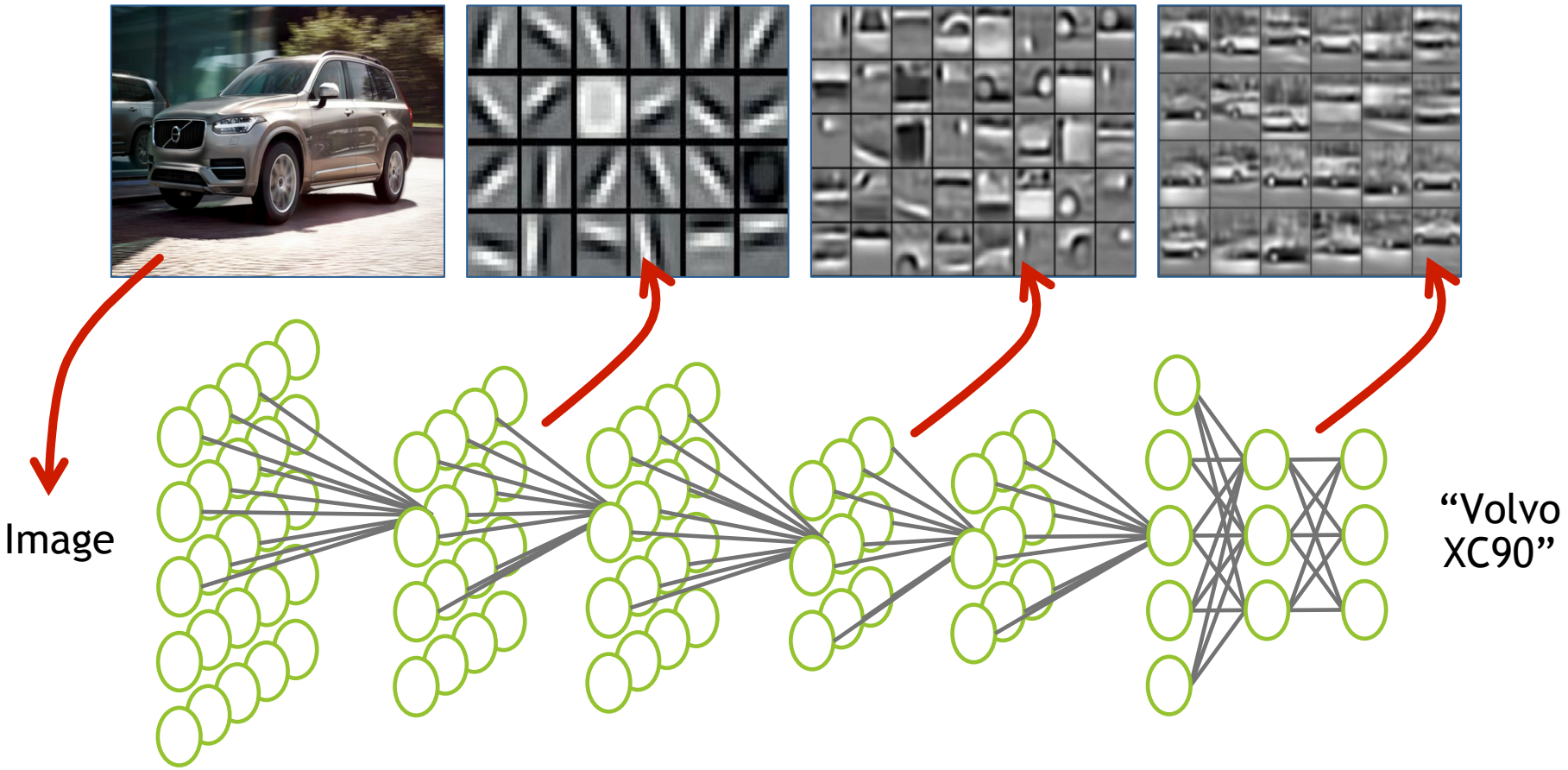
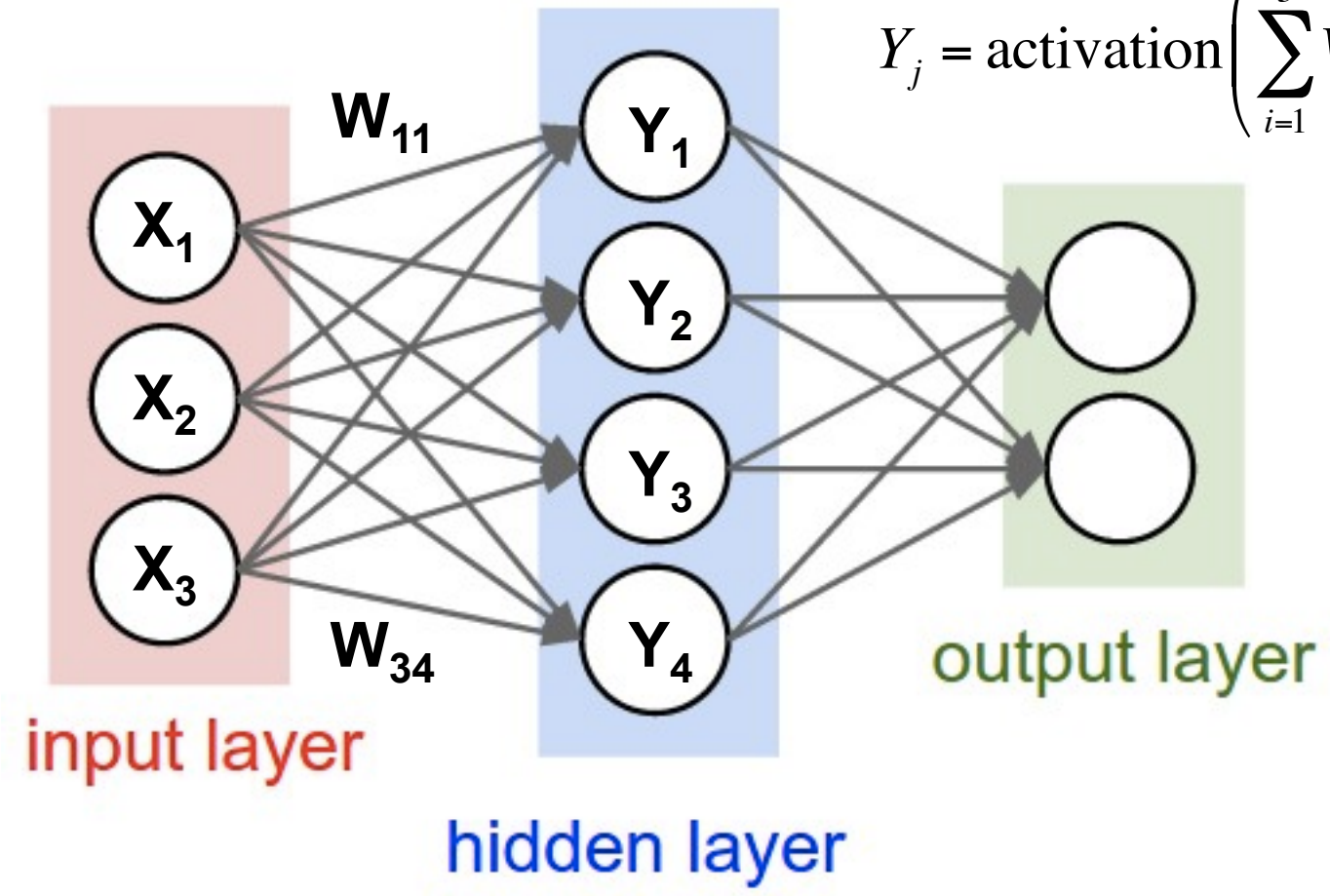


Image Source: [Lee et al., Comm. ACM 2011]

Weighted Sums

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



Why is Deep Learning Hot Now?

Big Data Availability

GPU Acceleration

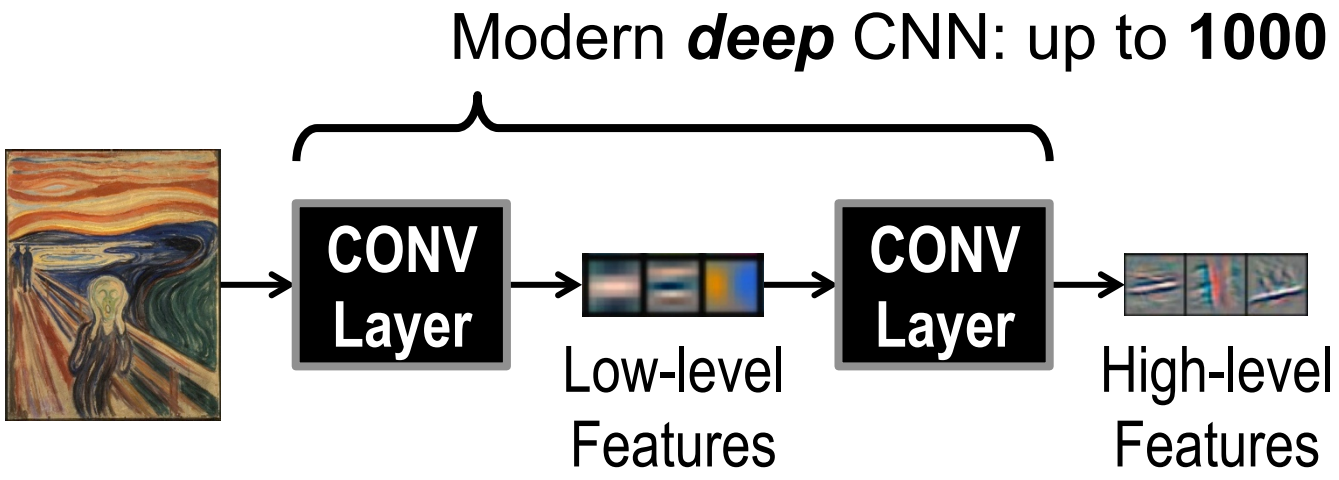
New ML Techniques

facebook **350M** images uploaded per day

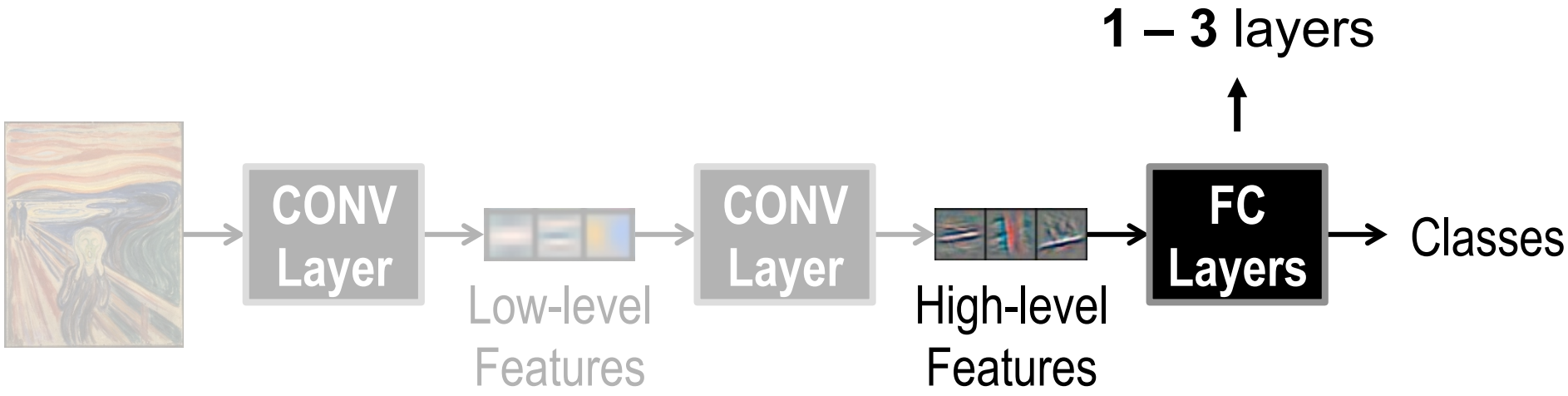
Walmart **2.5 Petabytes** of customer data hourly

YouTube **300 hours** of video uploaded every minute

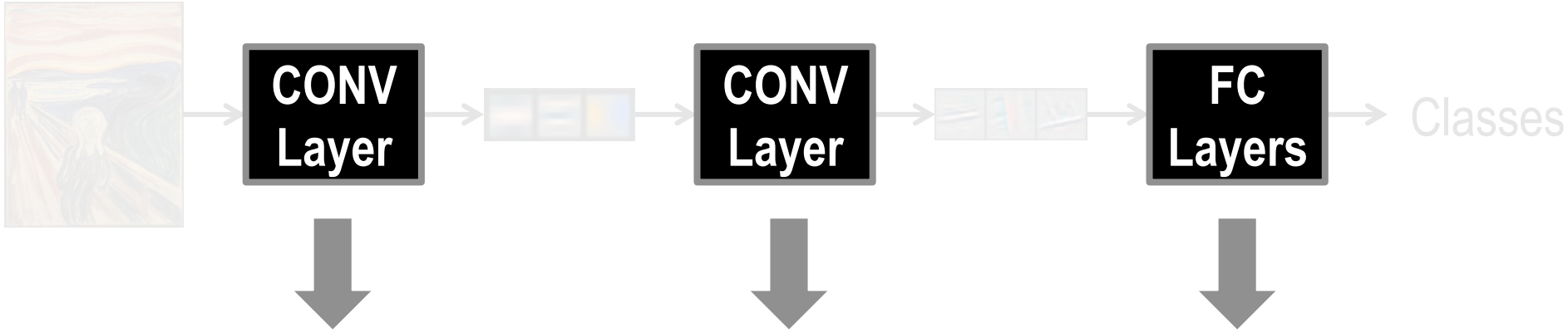
Deep Convolutional Neural Networks

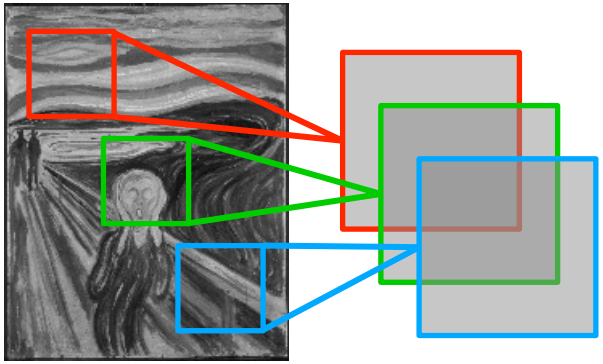


Deep Convolutional Neural Networks



Deep Convolutional Neural Networks

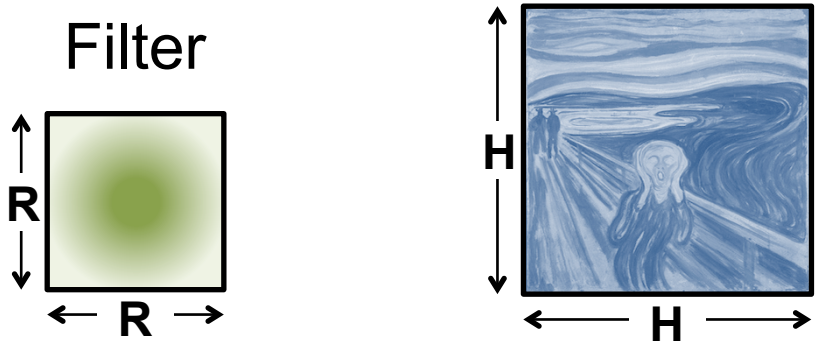




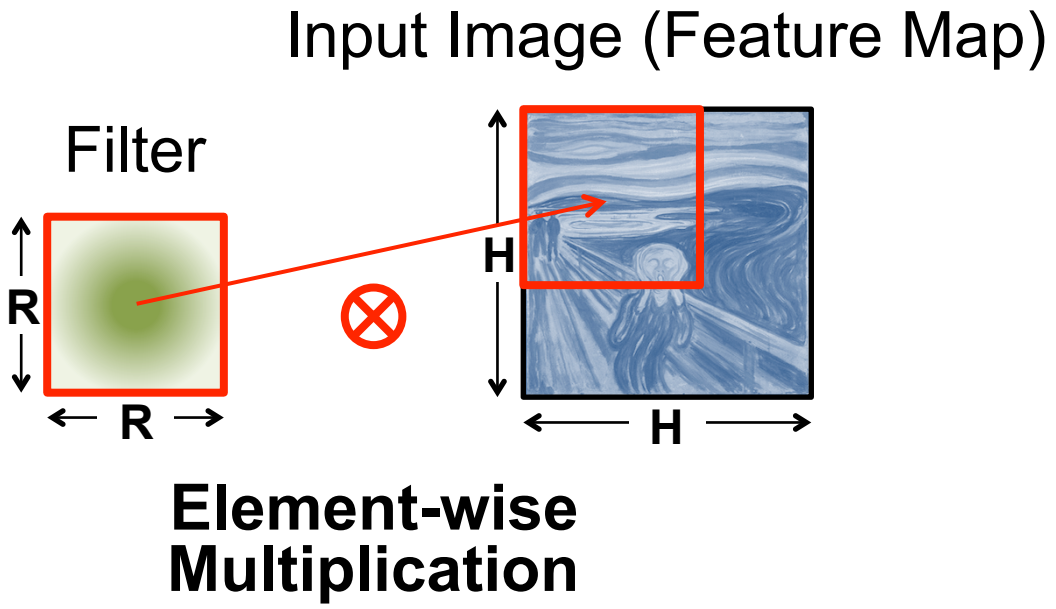
Convolutions account for more than 90% of overall computation, dominating **runtime** and **energy consumption**

High-Dimensional CNN Convolution

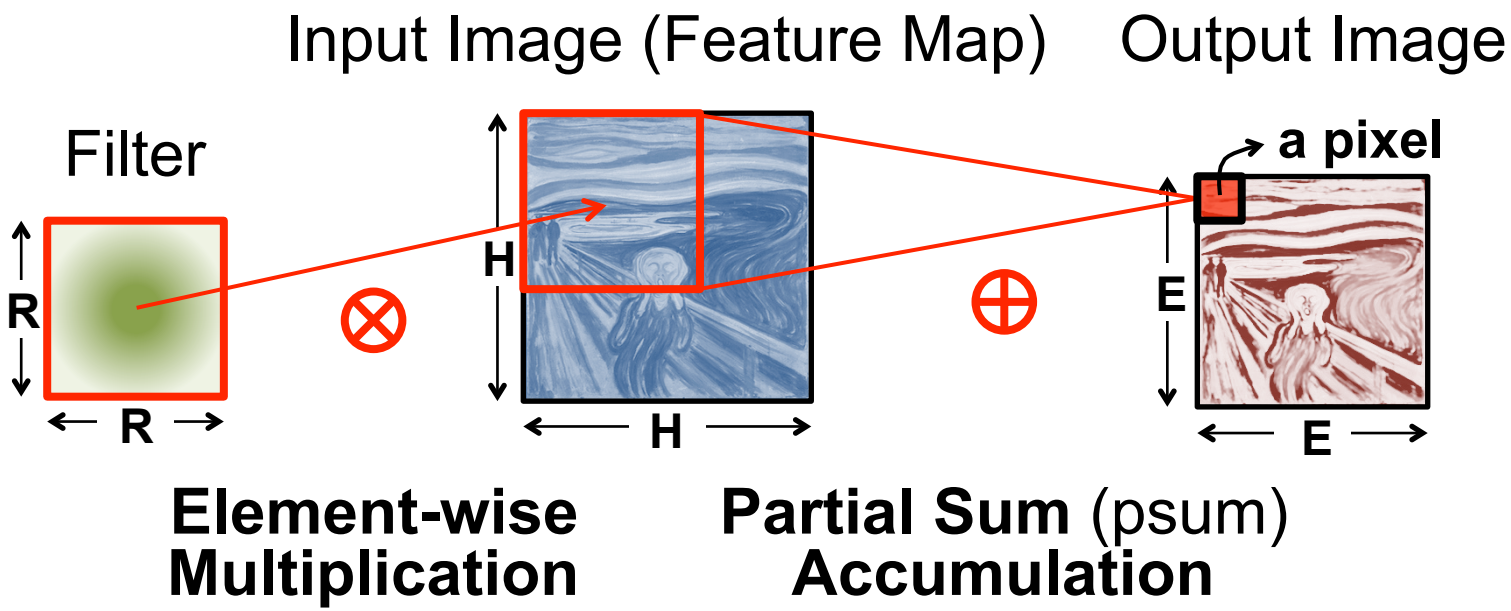
Input Image (Feature Map)



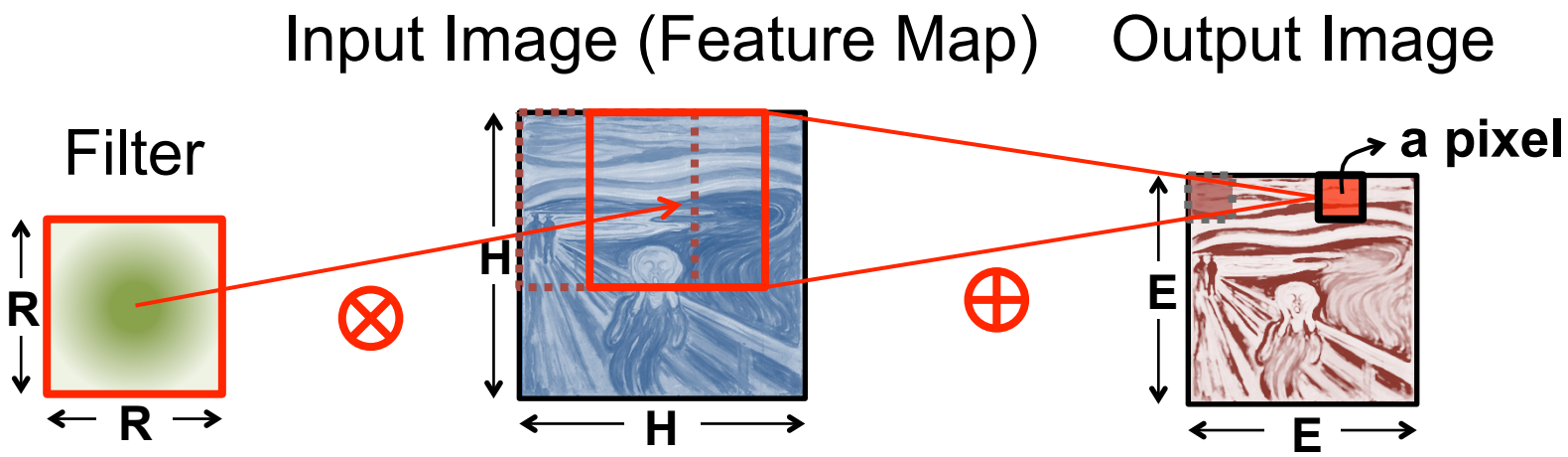
High-Dimensional CNN Convolution



High-Dimensional CNN Convolution

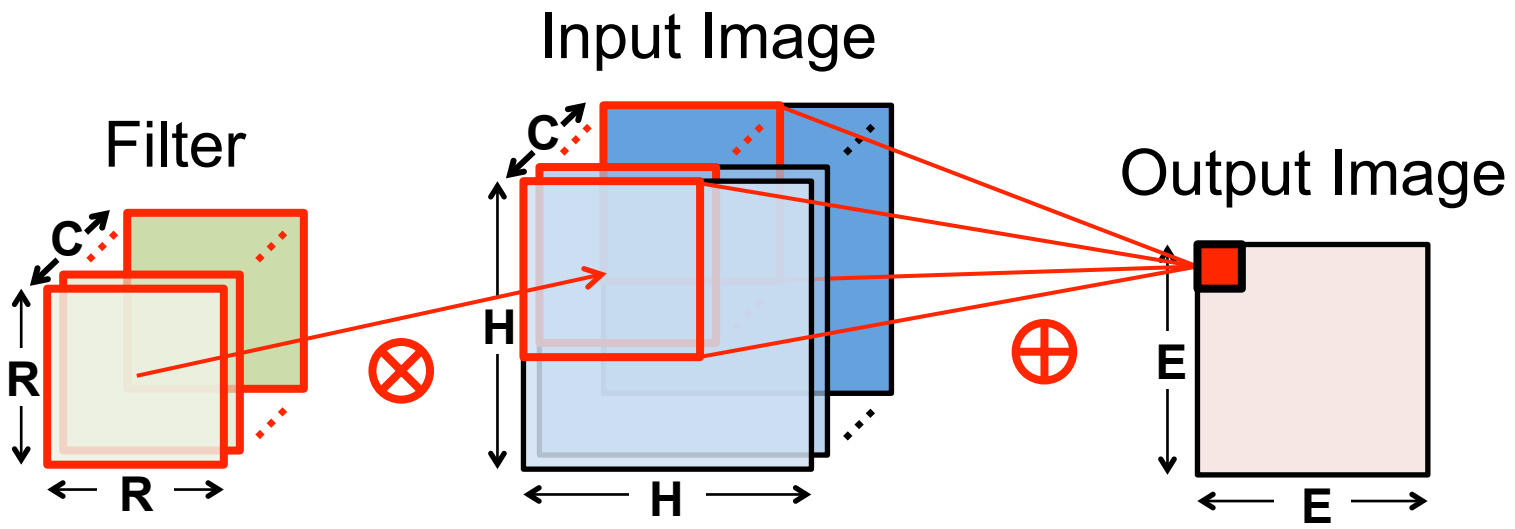


High-Dimensional CNN Convolution



Sliding Window Processing

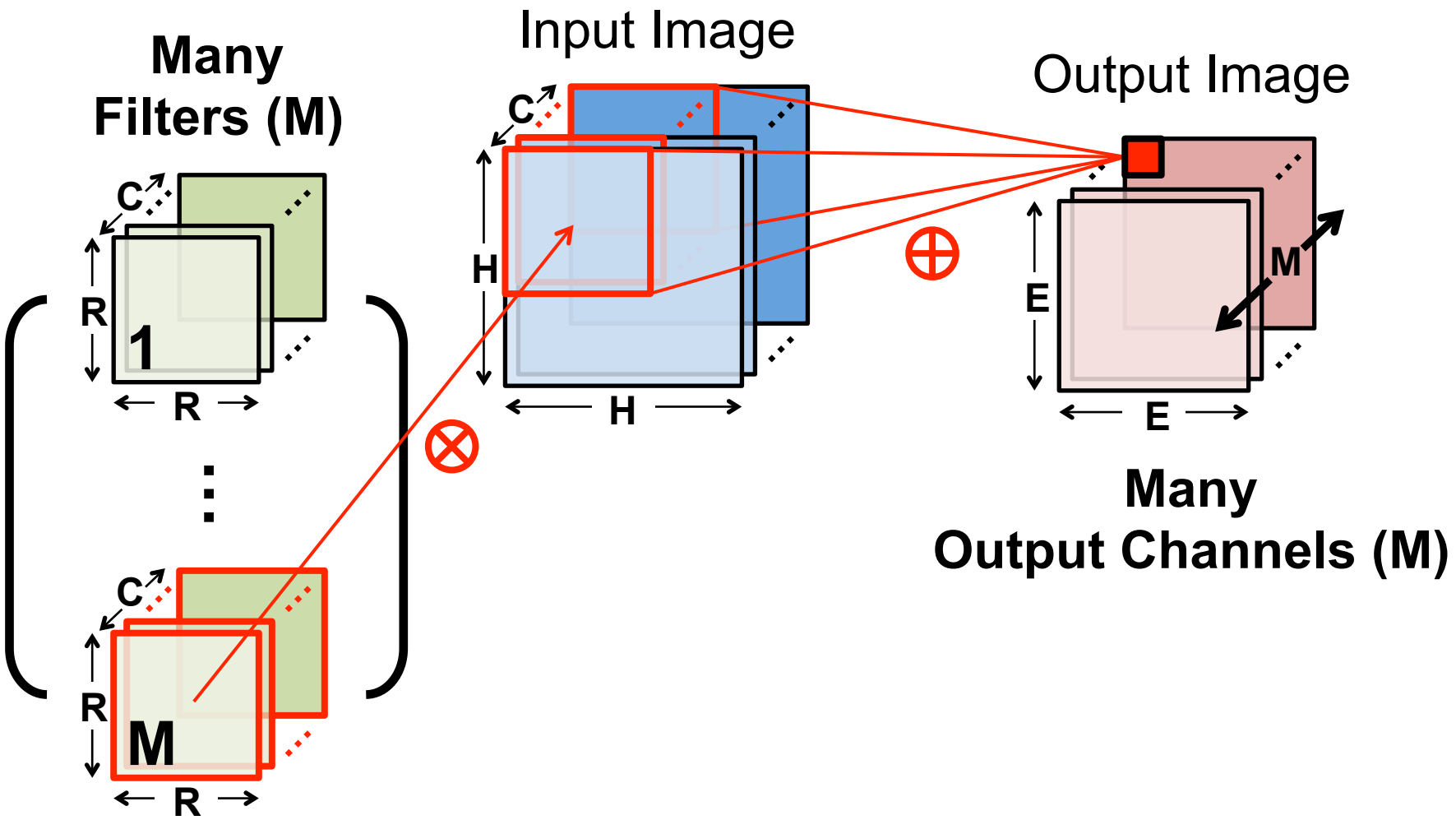
High-Dimensional CNN Convolution



Many Input Channels (C)

AlexNet: 3 – 192 Channels (C)

High-Dimensional CNN Convolution



AlexNet: 96 – 384 Filters (M)

High-Dimensional CNN Convolution

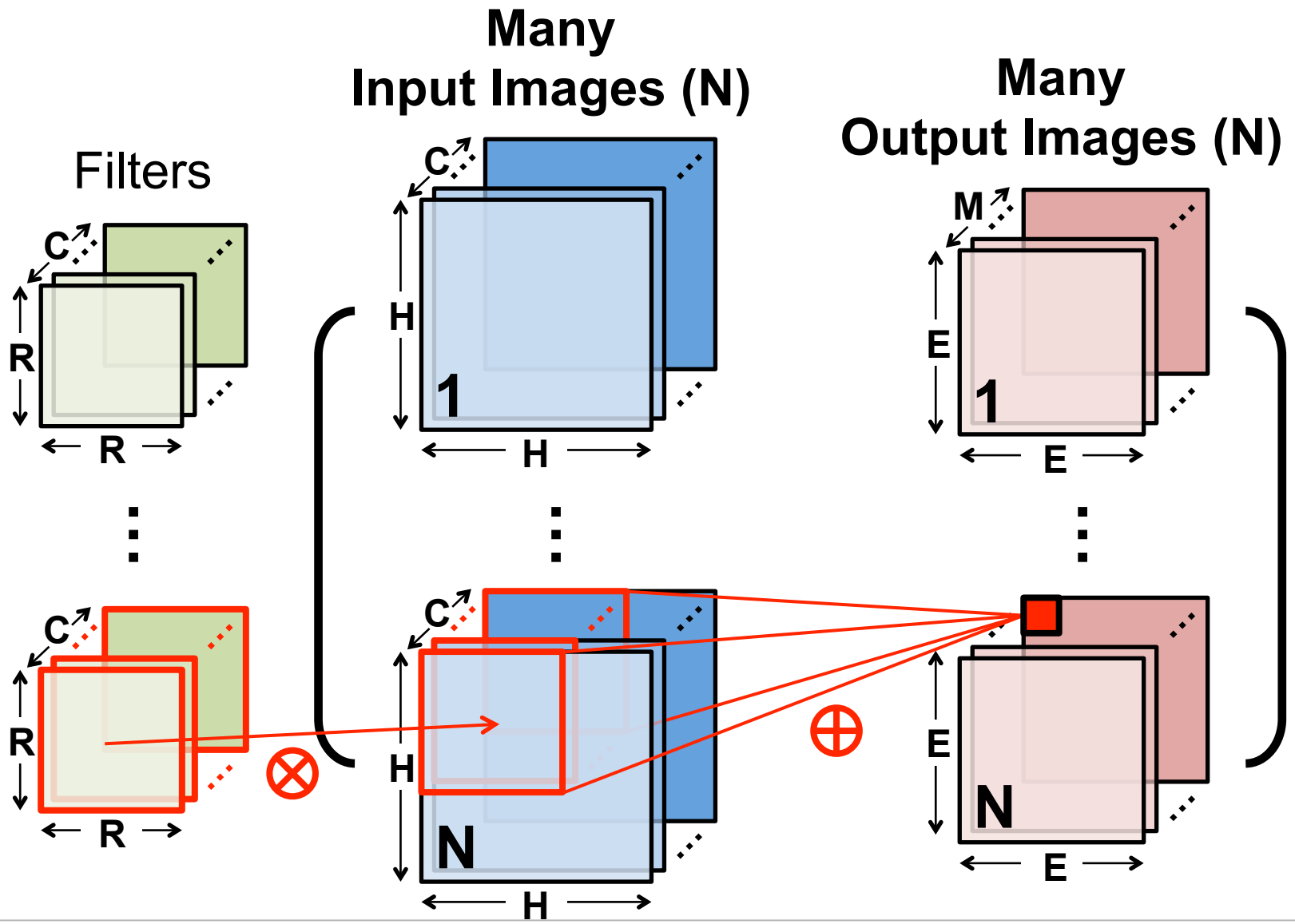


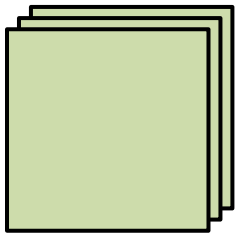
Image batch size: 1 – 256 (N)

Large Sizes with Varying Shapes

AlexNet[1] Convolutional Layer Configurations

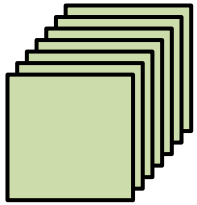
Layer	Filter Size (R)	# Filters (M)	# Channels (C)	Stride
1	11x11	96	3	4
2	5x5	256	48	1
3	3x3	384	256	1
4	3x3	384	192	1
5	3x3	256	192	1

Layer 1



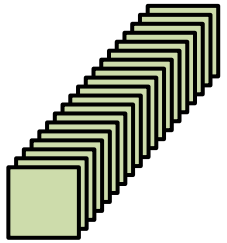
34k Params
105M MACs

Layer 2



307k Params
224M MACs

Layer 3

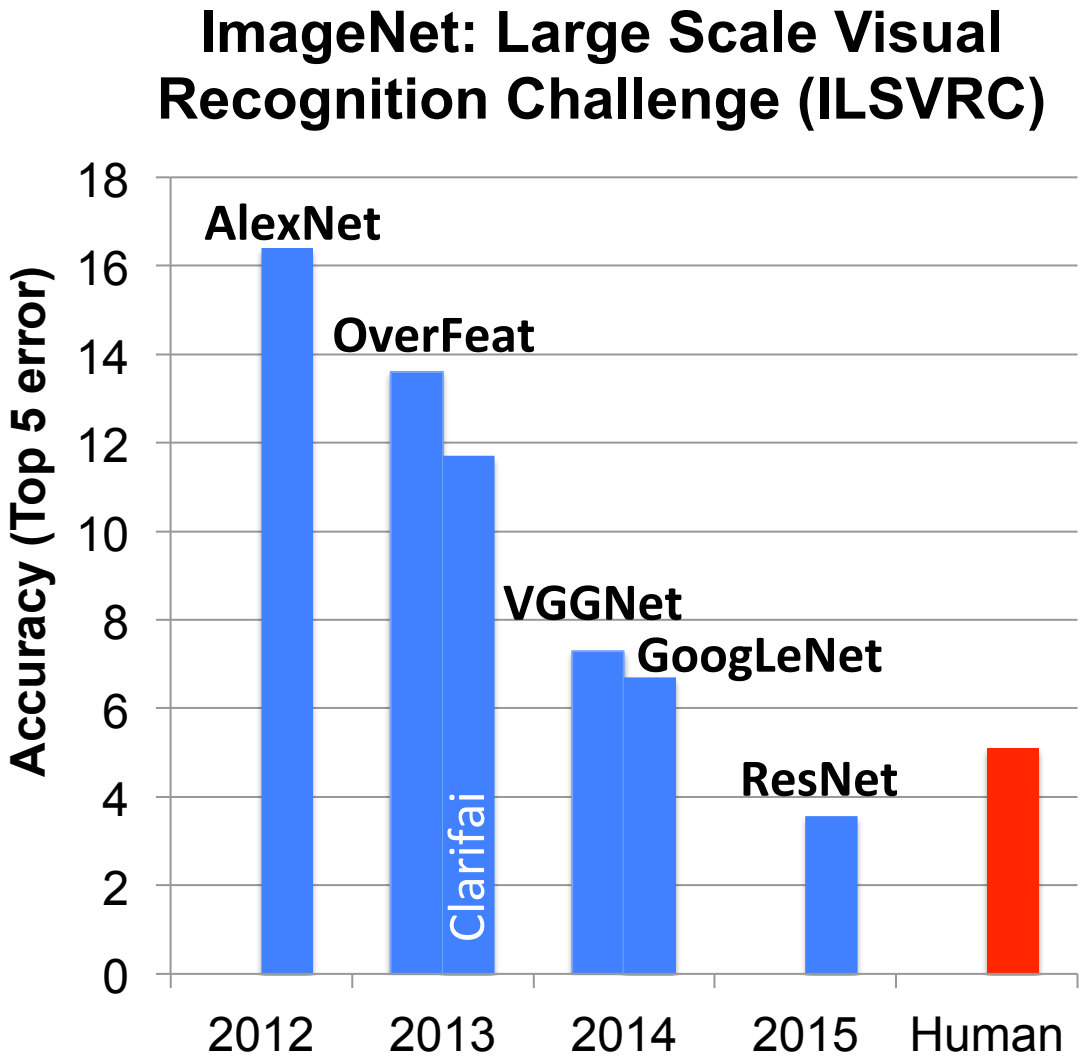


885k Params
150M MACs

1. [Krizhevsky, NIPS 2012]

Popular CNNs

- LeNet (1998)
- AlexNet (2012)
- OverFeat (2013)
- VGGNet (2014)
- GoogleNet (2014)
- ResNet (2015)



[O. Russakovsky et al., IJCV 2015]

Summary of Popular CNNs

Metrics	LeNet-5	AlexNet	VGG-16	GoogLeNet (v1)	ResNet-50
Top-5 error	n/a	16.4	7.4	6.7	5.3
Input Size	28x28	227x227	224x224	224x224	224x224
# of CONV Layers	**2**	**5**	**16**	**21 (depth)**	**49**
Filter Sizes	5	3, 5,11	3	1, 3 , 5, 7	1, 3, 7
# of Channels	1, 6	3 - 256	3 - 512	3 - 1024	3 - 2048
# of Filters	6, 16	96 - 384	64 - 512	64 - 384	64 - 2048
Stride	1	1, 4	1	1, 2	1, 2
# of Weights	2.6k	2.3M	14.7M	6.0M	23.5M
# of MACs	283k	666M	15.3G	1.43G	3.86G
# of FC layers	**2**	**3**	**3**	**1**	**1**
# of Weights	58k	58.6M	124M	1M	2M
# of MACs	58k	58.6M	124M	1M	2M
Total Weights	**60k**	**61M**	**138M**	**7M**	**25.5M**
Total MACs	**341k**	**724M**	**15.5G**	**1.43G**	**3.9G**

CONV Layers increasingly important!

Training vs. Inference

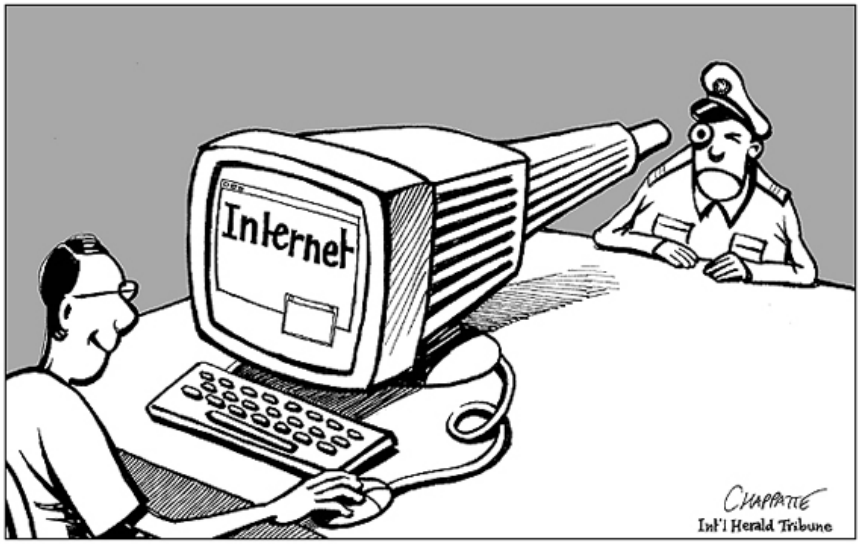
Training
(determine weights)

Weights

Inference
(use weights)

Processing at "Edge" instead of the "Cloud"

Privacy



Communication

Image source:
www.theregister.co.uk

Latency

Sensor

Actuator

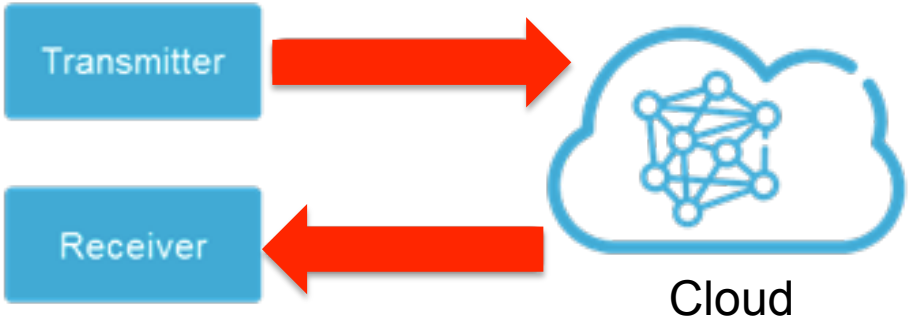


Image source: ericsson.com

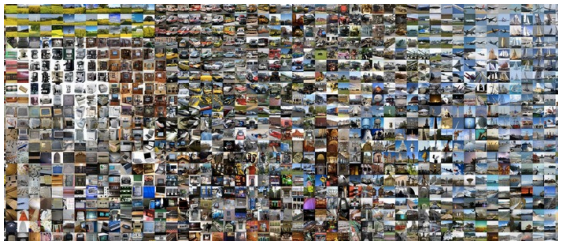
Challenges

Key Metrics

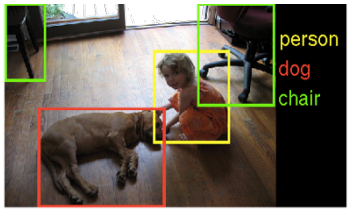
- **Accuracy**
  - Evaluate hardware using the appropriate DNN model and dataset
- **Programmability**
  - Support multiple applications
  - Different weights
- **Energy/Power**
  - Energy per operation
  - DRAM Bandwidth
- **Throughput/Latency**
  - GOPS, frame rate, delay
- **Cost**
  - Area (size of memory and # of cores)

MNIST

ImageNet

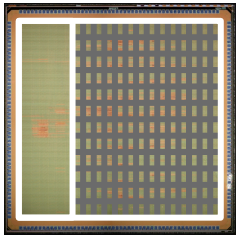


Computer Vision



Speech Recognition

Chip



[Sze et al., CICC 2017]

Opportunities in Architecture

GPUs and CPUs Targeting Deep Learning

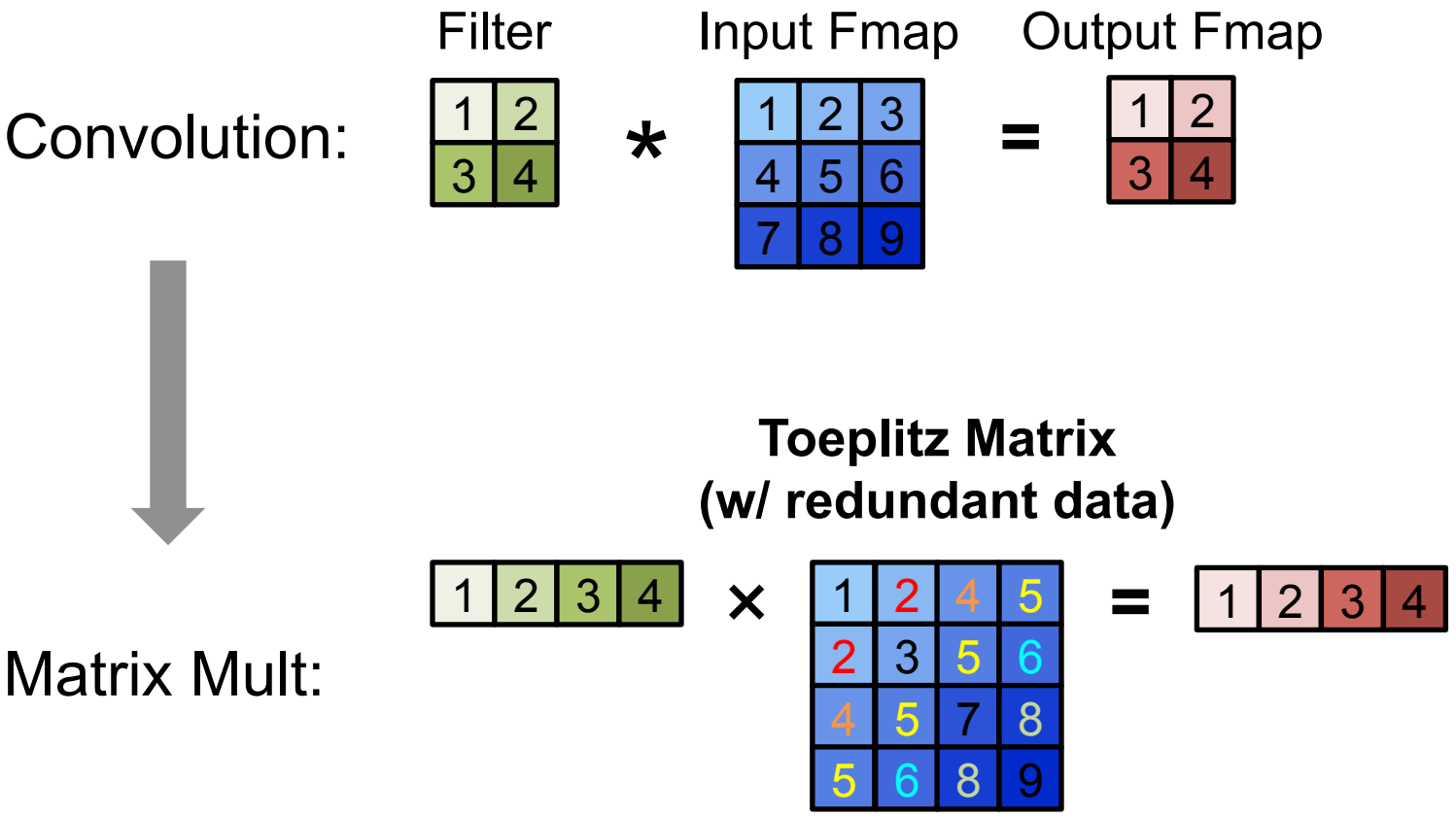
Intel Knights Landing (2016)

Knights Mill: next gen Xeon Phi "optimized for deep learning"

Nvidia PASCAL GP100 (2016)

Use **matrix multiplication libraries** on CPUs and GPUs

Map DNN to a Matrix Multiplication



Goal: Reduced number of operations to **increase throughput**

Reduce Operations in Matrix Multiplication

- **Fast Fourier Transform** [Mathieu, ICLR 2014]
  - **Pro:** Direct convolution $O(N_o^2 N_f^2)$ to $O(N_o^2 \log_2 N_o)$
  - **Con:** Increase storage requirements
- **Strassen** [Cong, ICANN 2014]
  - **Pro:** $O(N^3)$ to $(N^{2.807})$
  - **Con:** Numerical stability
- **Winograd** [Lavin, CVPR 2016]
  - **Pro:** 2.25x speed up for 3x3 filter
  - **Con:** Specialized processing depending on filter size

Analogy: Gauss's Multiplication Algorithm

$$(a+bi)(c+di) = (ac-bd) + (bc+ad)i.$$

4 multiplications + 3 additions

$$k_1 = c \cdot (a+b)$$

$$k_2 = a \cdot (d-c)$$

$$k_3 = b \cdot (c+d)$$

$$\text{Real part} = k_1 - k_3$$

$$\text{Imaginary part} = k_1 + k_2.$$

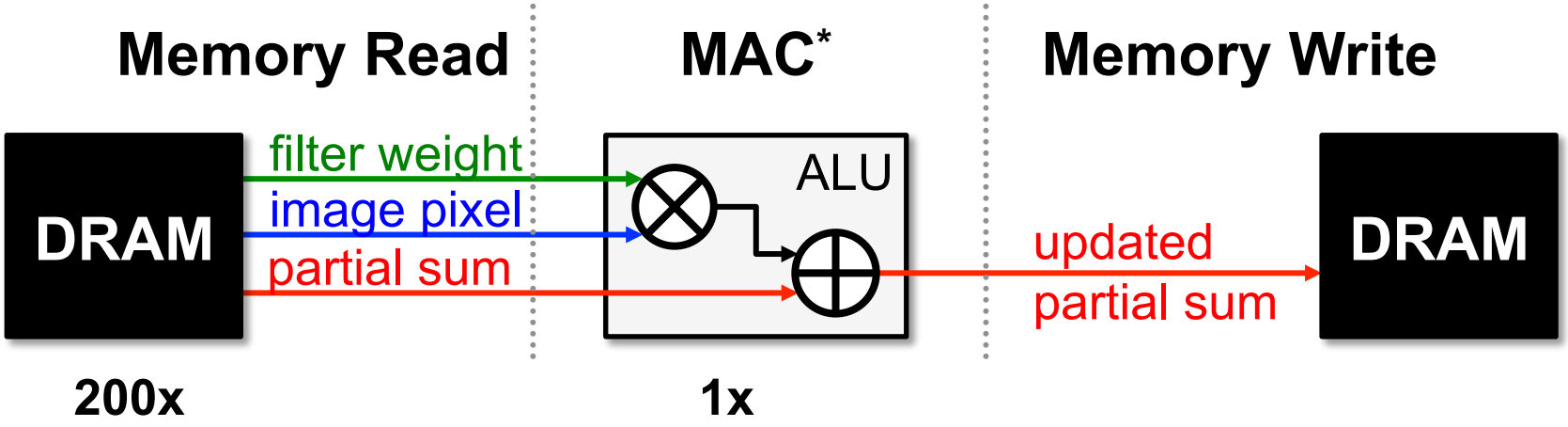
3 multiplications + 5 additions

Reduce number of multiplications, but **increase** number of additions

Specialized Hardware (Accelerators)

Properties We Can Leverage

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

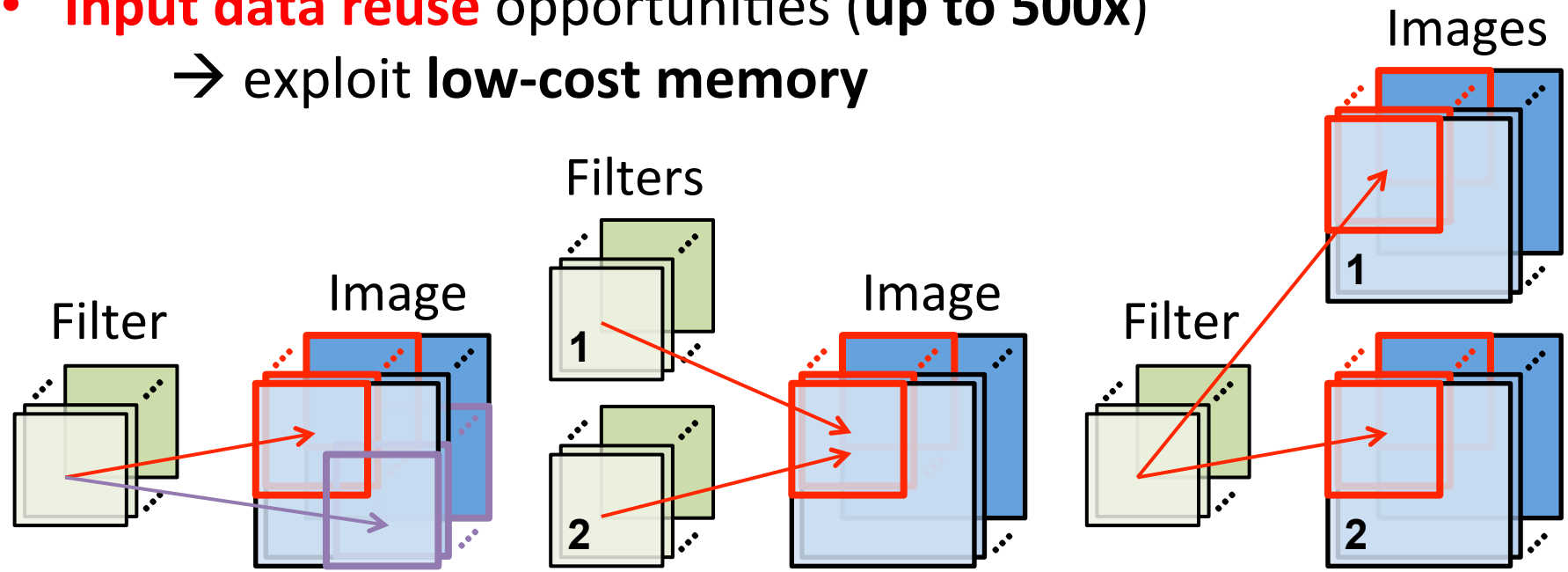


Worst Case: all memory R/W are **DRAM** accesses

- Example: AlexNet [NIPS 2012] has **724M** MACs
  → **2896M** DRAM accesses required

Properties We Can Leverage

- Operations exhibit **high parallelism**
  → **high throughput** possible
- **Input data reuse** opportunities (**up to 500x**)
  → exploit **low-cost memory**



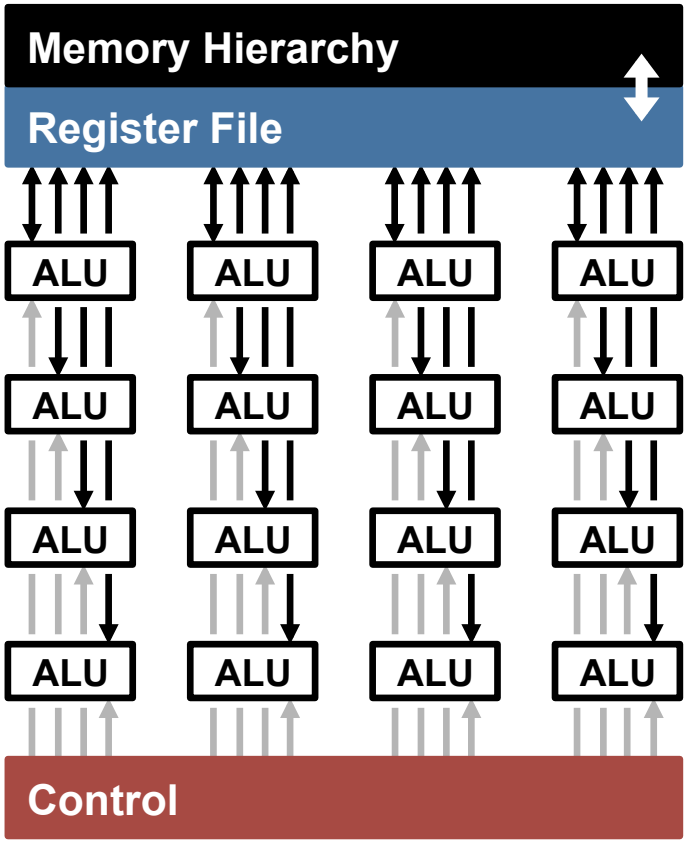
Convolutional Reuse (pixels, weights)

Image Reuse (pixels)

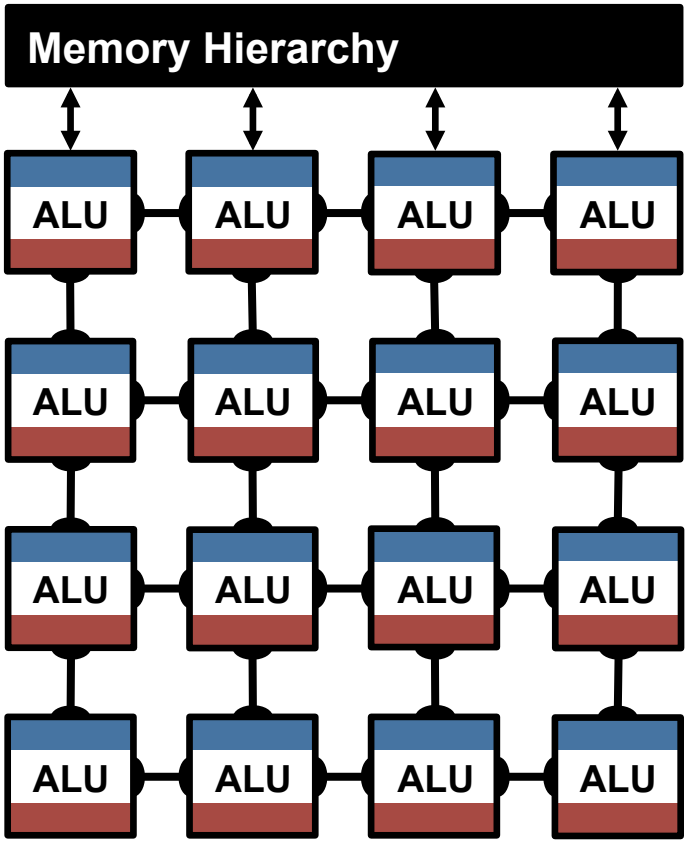
Filter Reuse (weights)

Highly-Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)



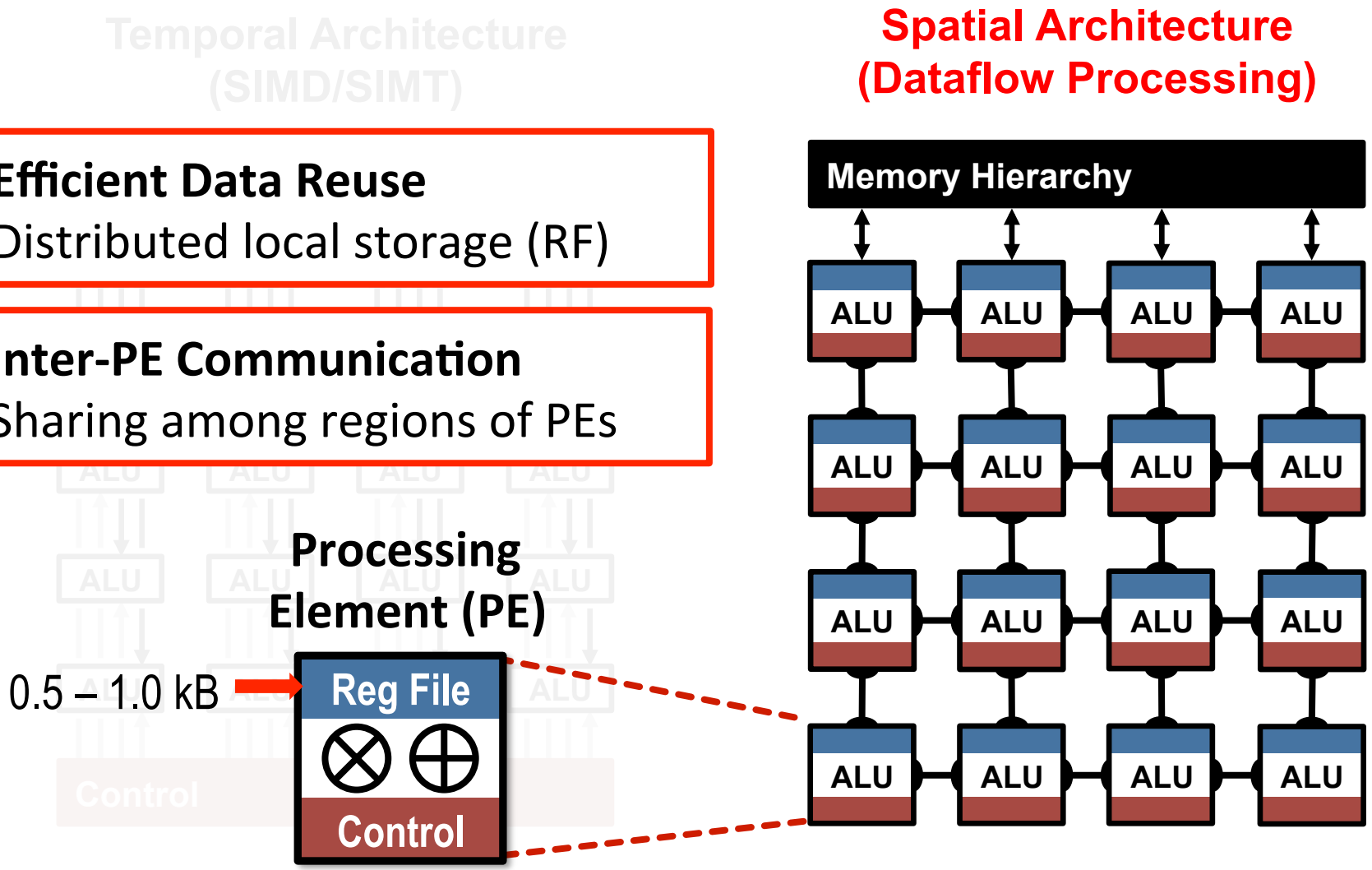
Spatial Architecture (Dataflow Processing)



Advantages of Spatial Architecture

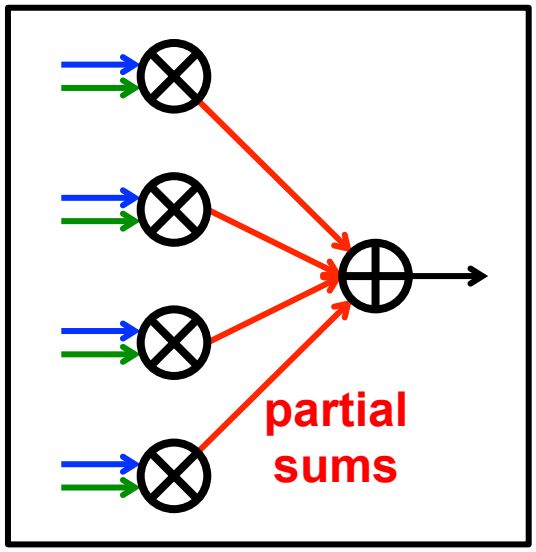
Efficient Data Reuse
Distributed local storage (RF)

Inter-PE Communication
Sharing among regions of PEs



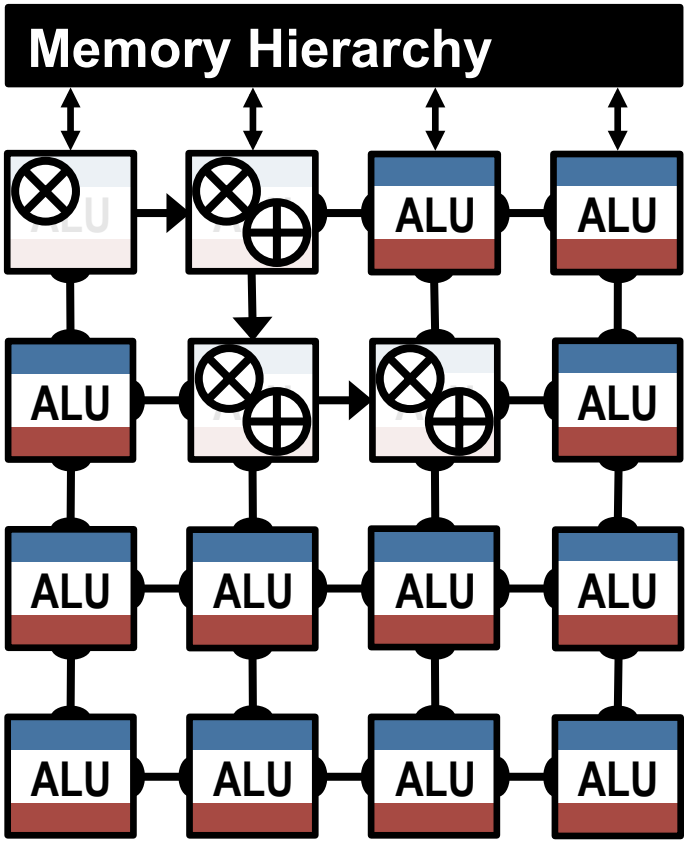
How to Map the Dataflow?

CNN Convolution



?

Spatial Architecture (Dataflow Processing)



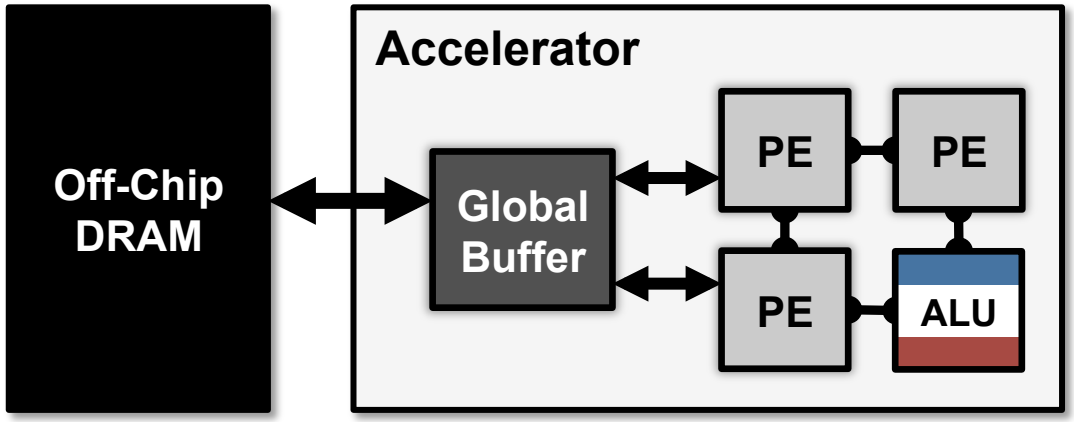
Goal: Increase reuse of input data (**weights** and **pixels**) and local **partial sums** accumulation

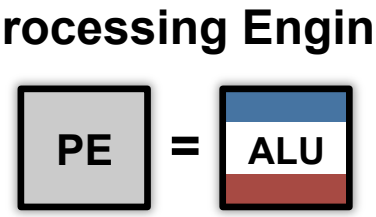
Energy-Efficient Dataflow

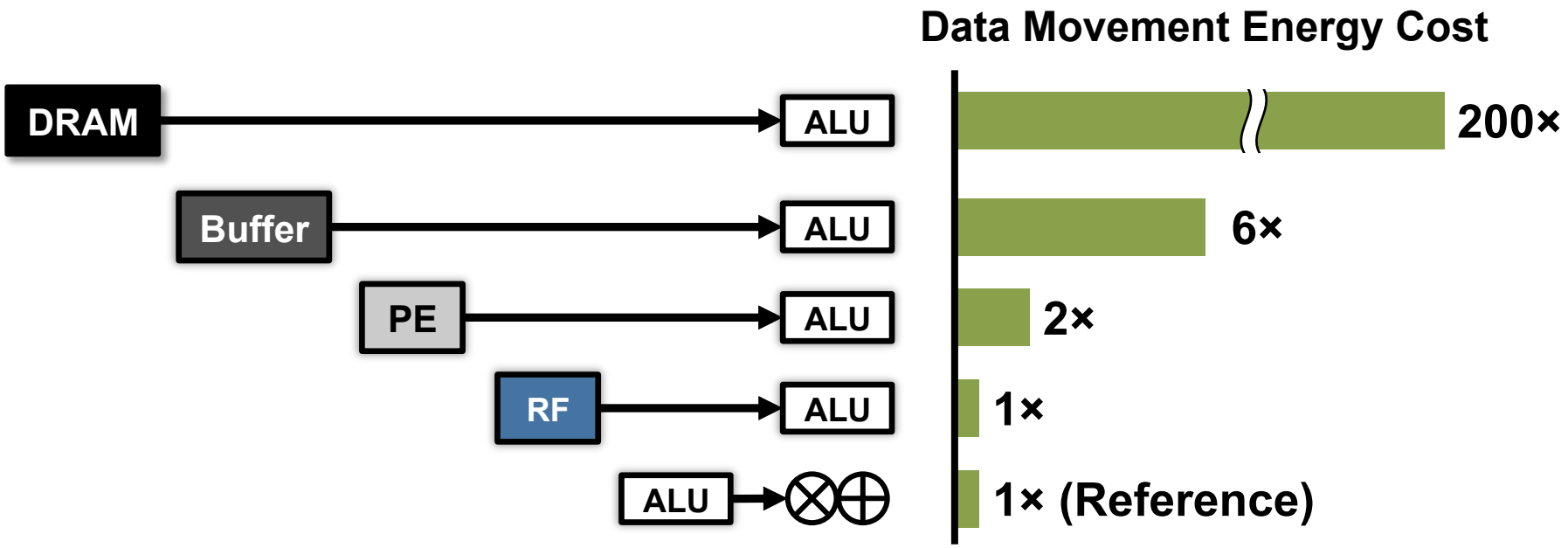
Yu-Hsin Chen, Joel Emer, Vivienne Sze, ISCA 2016

Maximize data reuse and accumulation at RF

Data Movement is Expensive

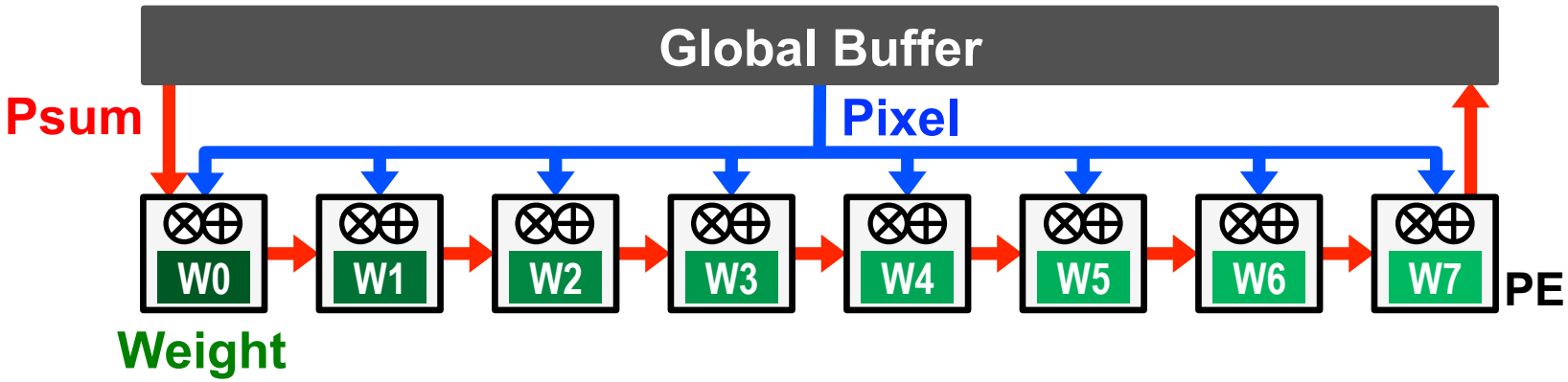






Maximize data reuse at lower levels of hierarchy

Weight Stationary (WS)

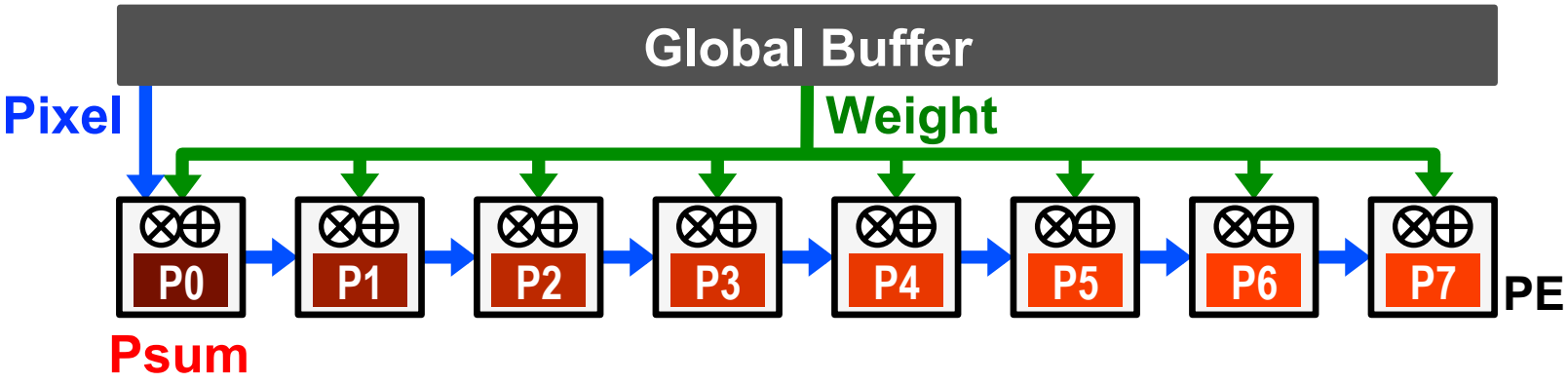


- **Minimize weight** read energy consumption
  - maximize convolutional and filter reuse of weights
- **Examples:**

[**Chakradhar**, *ISCA* 2010] [**nn-X (NeuFlow)**, *CVPRW* 2014]

[**Park**, *ISSCC* 2015] [**Origami**, *GLSVLSI* 2015]

Output Stationary (OS)



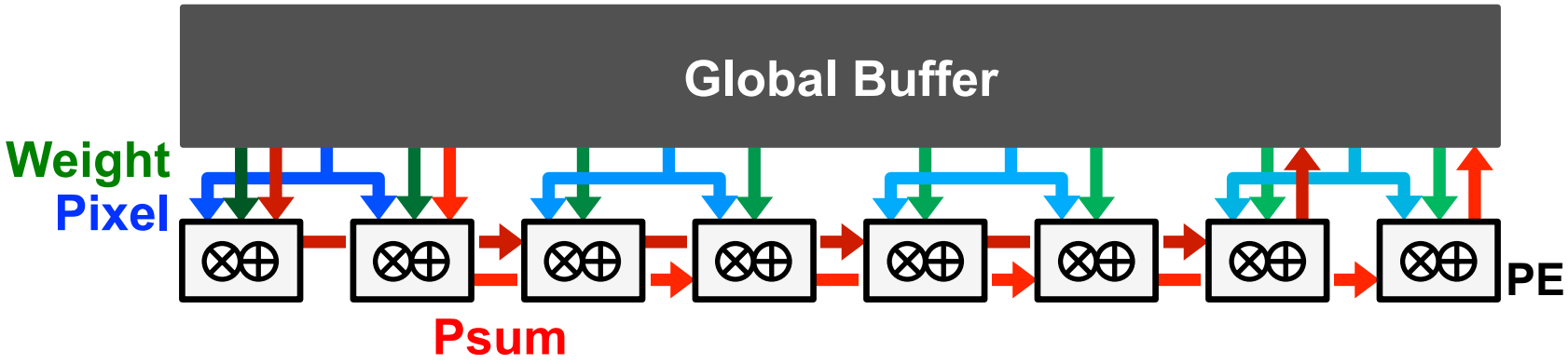
- **Minimize partial sum** R/W energy consumption
  - maximize local accumulation
- **Examples:**

  [**Gupta**, *ICML* 2015]

  [**ShiDianNao**, *ISCA* 2015]

  [**Peemen**, *ICCD* 2013]

No Local Reuse (NLR)

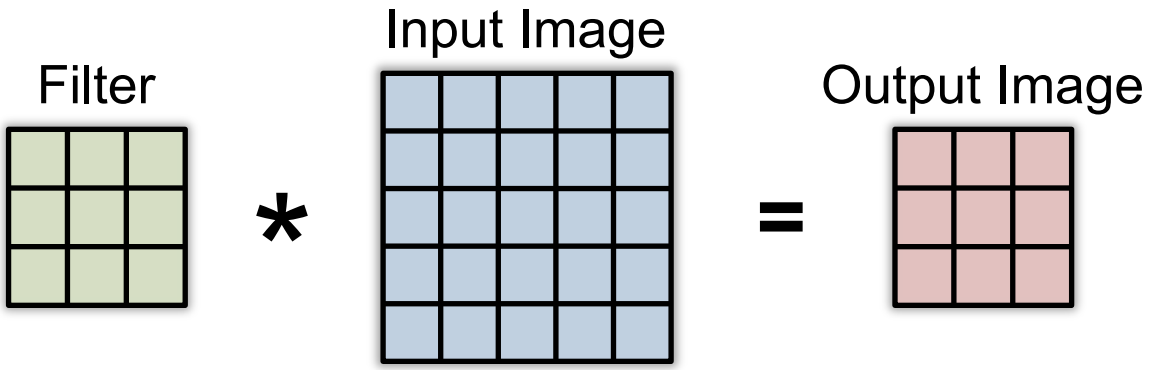


- Use a **large global buffer** as shared storage
  - Reduce **DRAM** access energy consumption
- **Examples:**

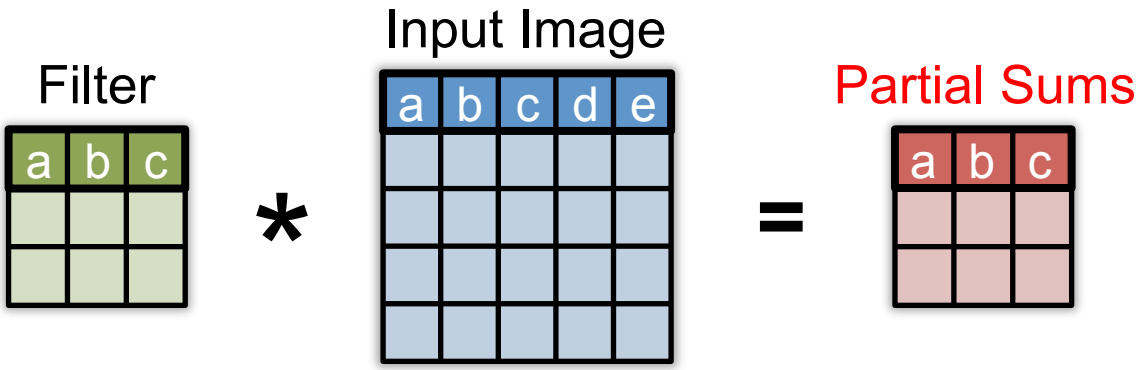
  [**DianNao**, *ASPLOS* 2014] [**DaDianNao**, *MICRO* 2014]

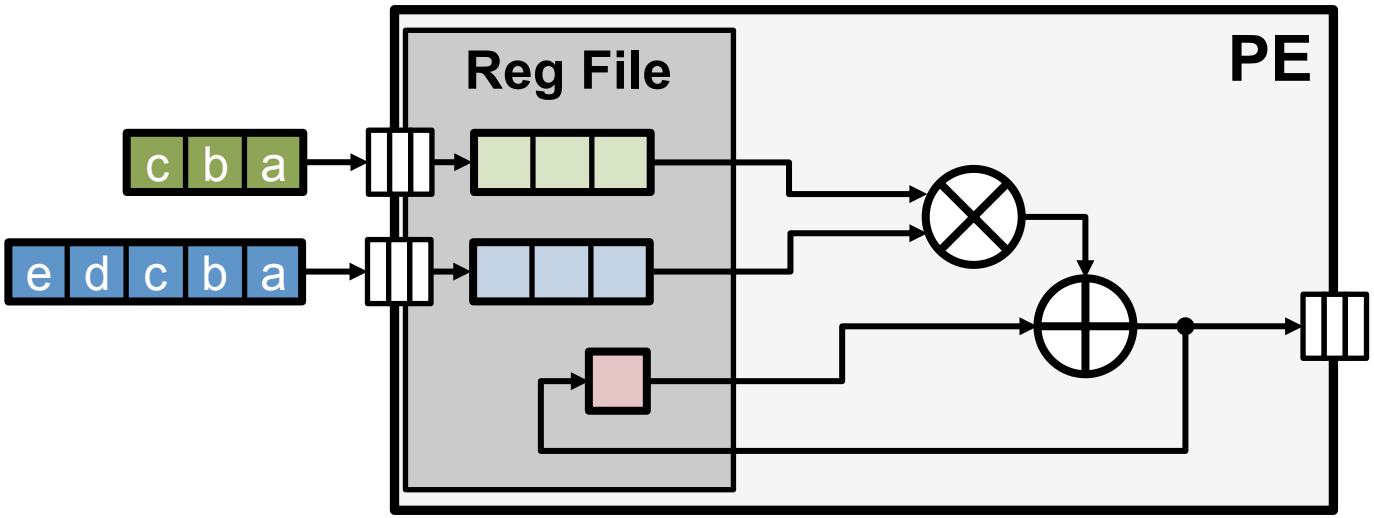
  [**Zhang**, *FPGA* 2015]

Row Stationary: Energy-efficient Dataflow

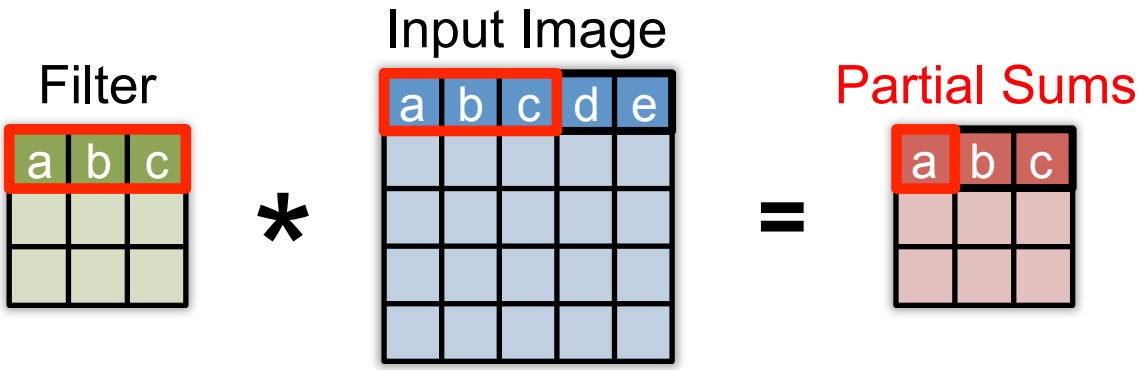


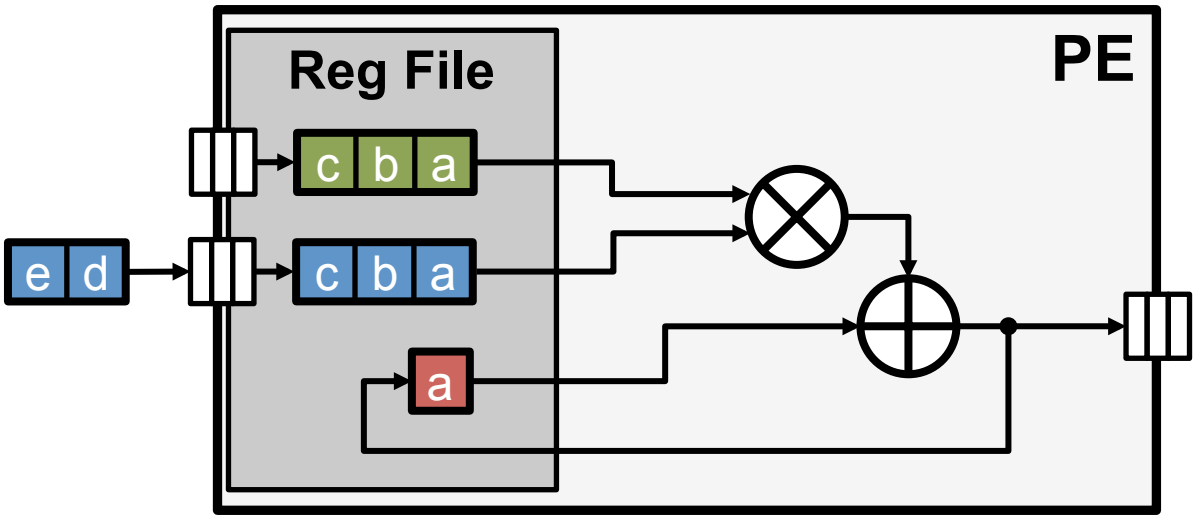
1D Row Convolution in PE



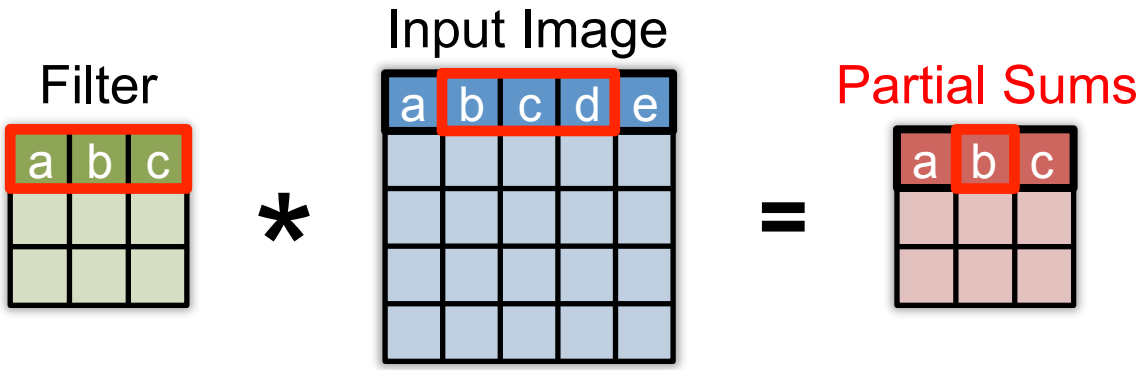


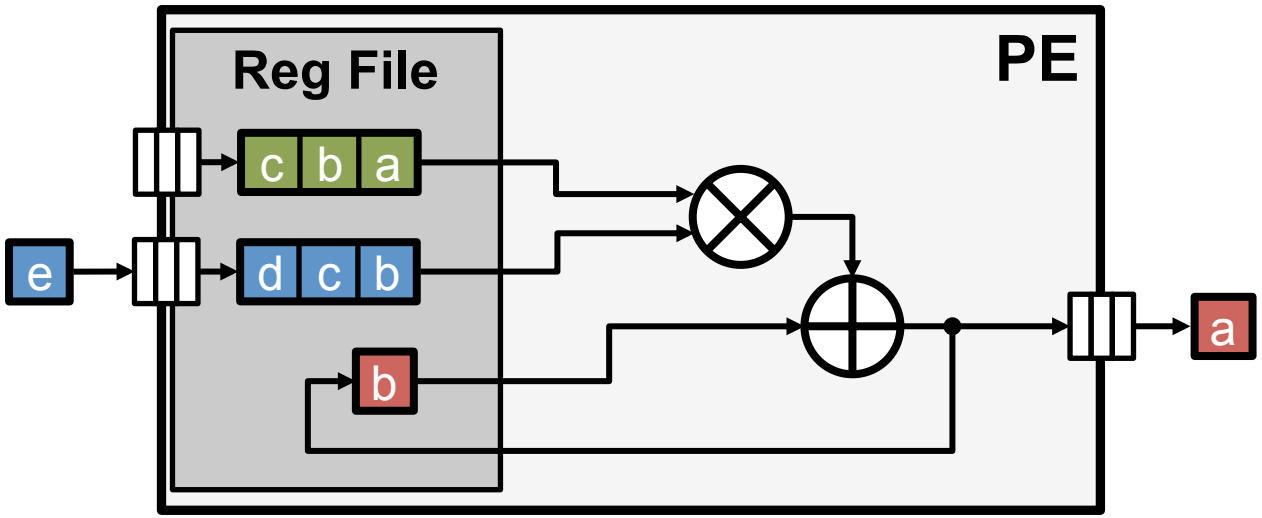
1D Row Convolution in PE



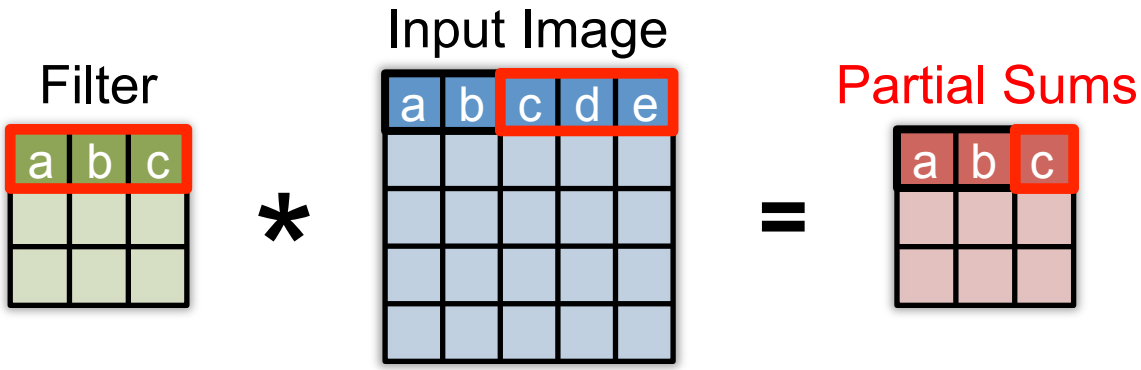


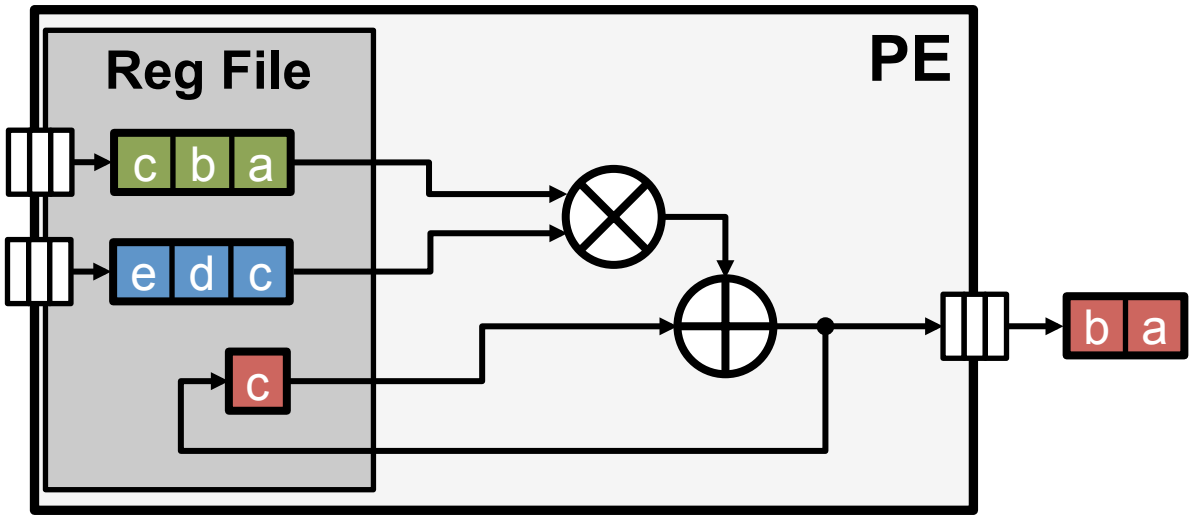
1D Row Convolution in PE





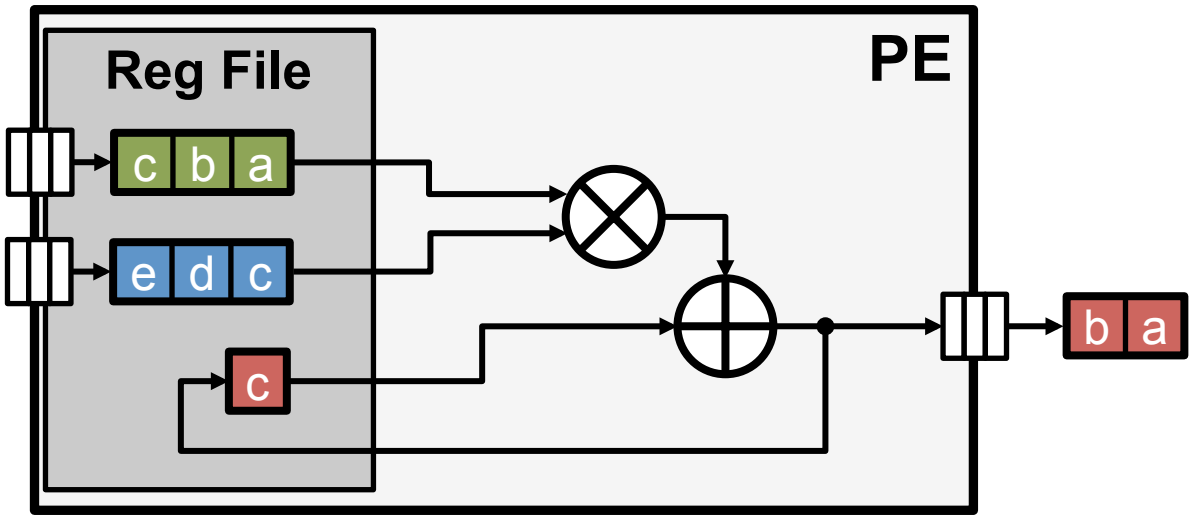
1D Row Convolution in PE



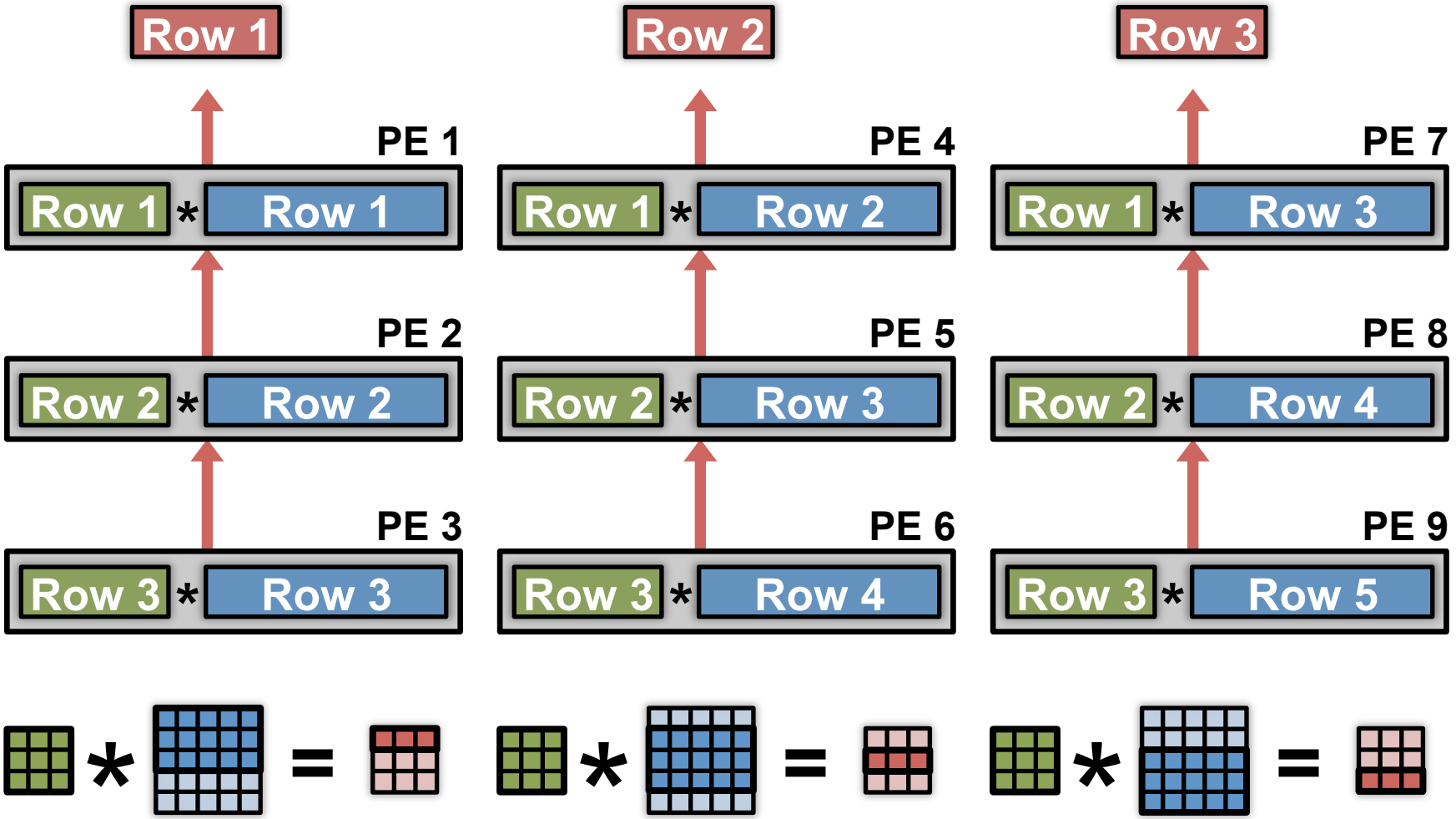


1D Row Convolution in PE

- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **image** sliding window in RF
- Maximize row **psum accumulation** in RF



Row Stationary Dataflow



Optimize for **overall energy efficiency** instead for only a certain data type

Evaluate Reuse in Different Dataflows

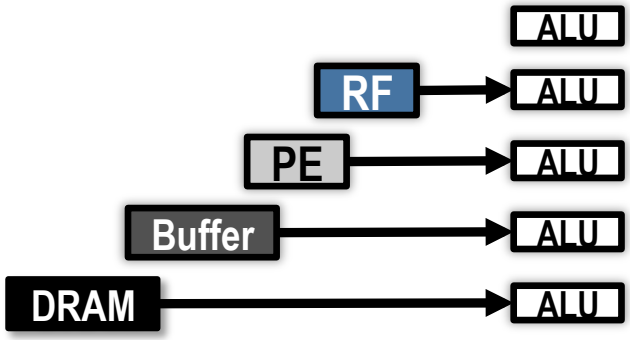
- **Weight Stationary**
  - Minimize movement of filter weights
- **Output Stationary**
  - Minimize movement of partial sums
- **No Local Reuse**
  - Don't use any local PE storage. Maximize global buffer size.
- **Row Stationary**

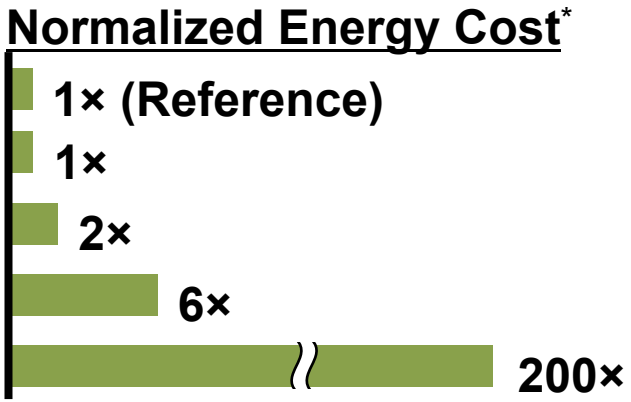
Evaluate Reuse in Different Dataflows

- **Weight Stationary**
  - Minimize movement of filter weights
- **Output Stationary**
  - Minimize movement of partial sums
- **No Local Reuse**
  - Don't use any local PE storage. Maximize global buffer size.
- **Row Stationary**

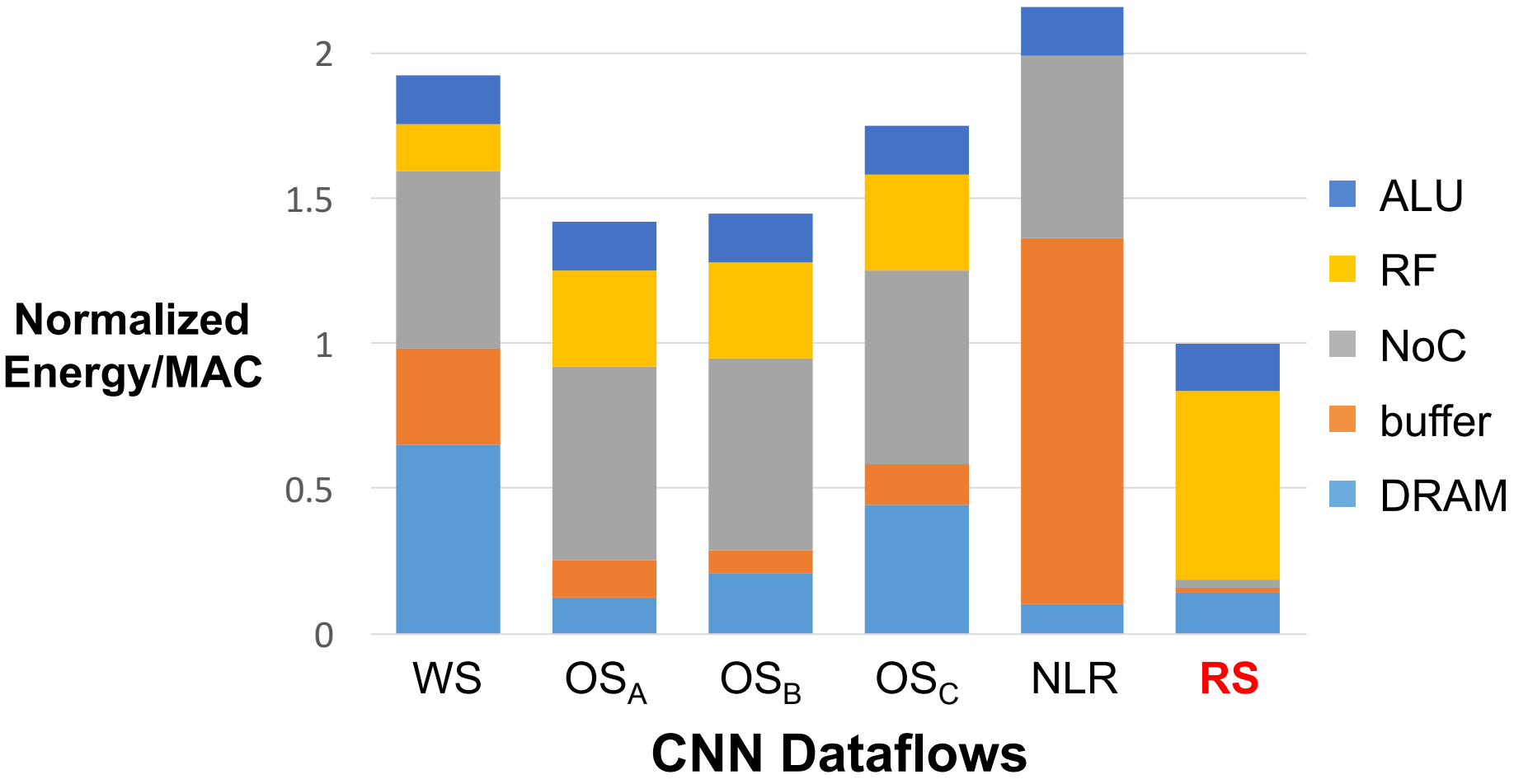
Evaluation Setup

- Same Total Area
- AlexNet
- 256 PEs
- Batch size = 16





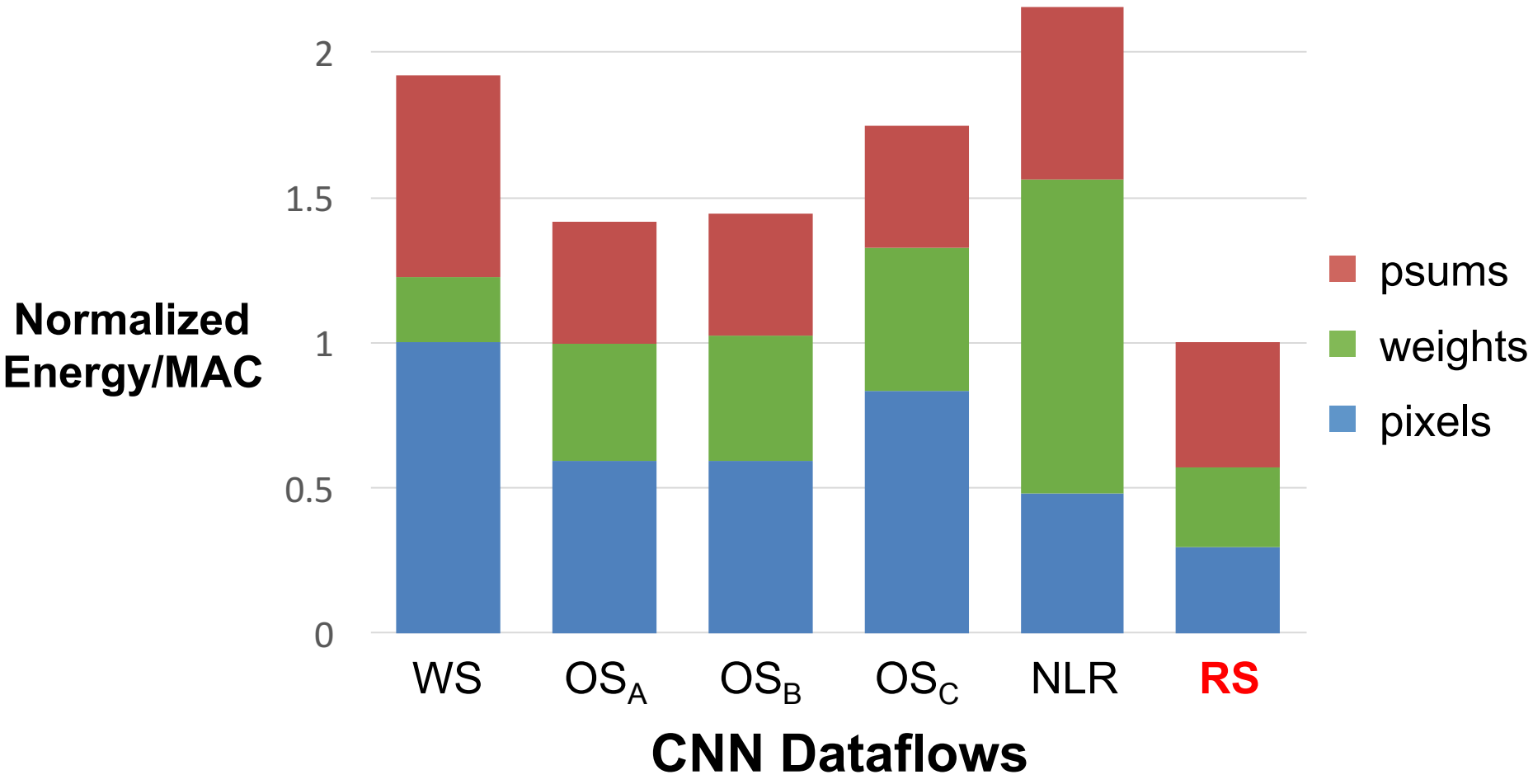
Dataflow Comparison: CONV Layers



RS uses **1.4× – 2.5× lower** energy than other dataflows

[**Chen**, *ISCA* 2016]

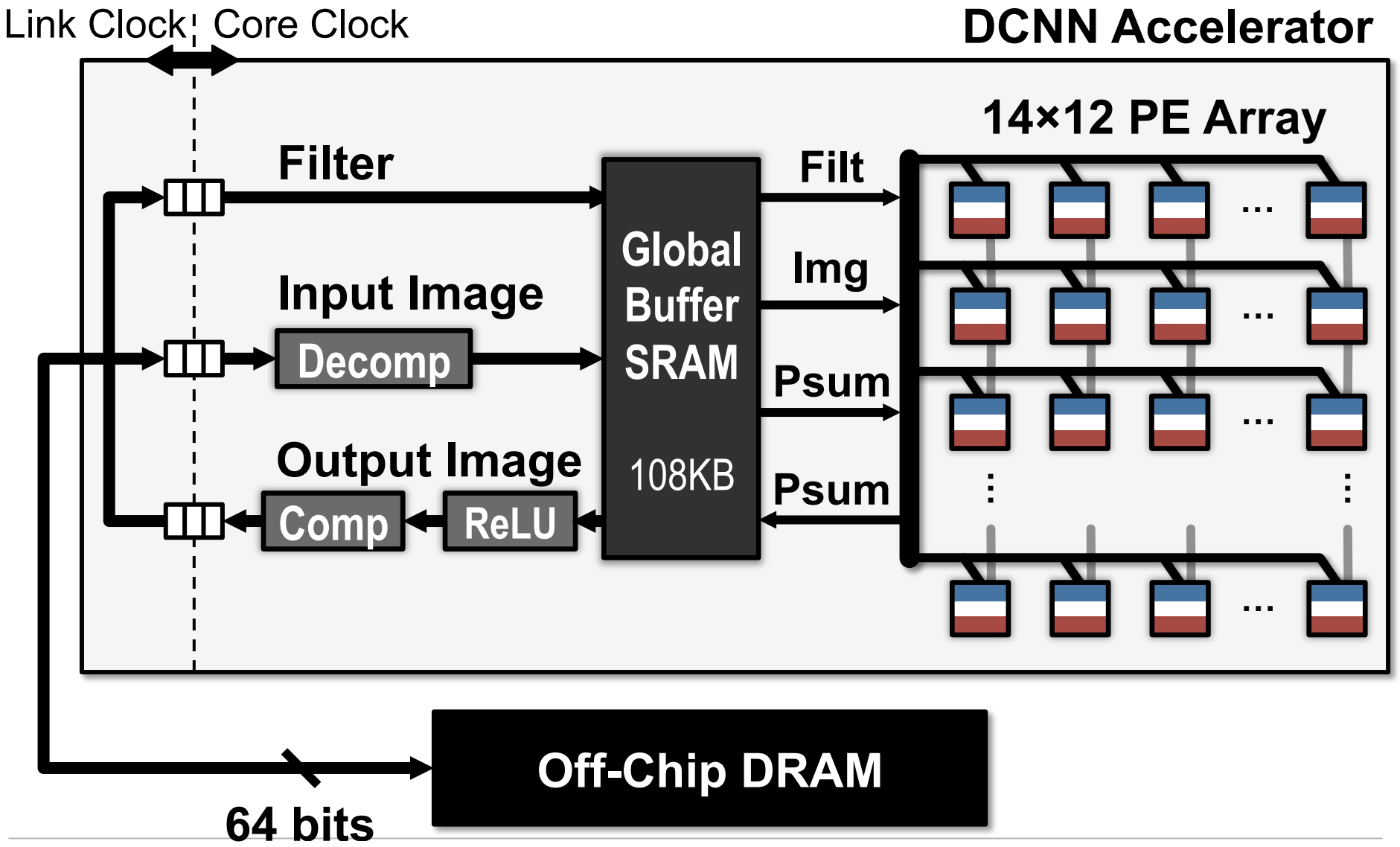
Dataflow Comparison: CONV Layers



RS optimizes for the best **overall** energy efficiency

[**Chen**, *ISCA* 2016]

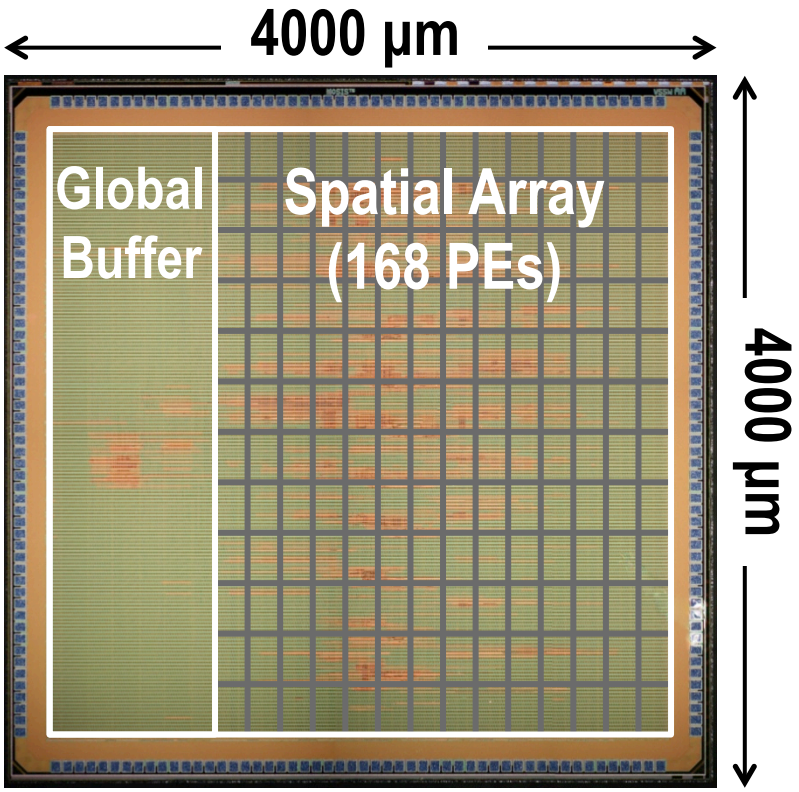
Eyeriss Deep CNN Accelerator



[Chen et al., ISSCC 2016]

Eyeriss Chip Spec & Measurement Results

Technology	TSMC 65nm LP 1P9M
On-Chip Buffer	108 KB
# of PEs	168
Scratch Pad / PE	0.5 KB
Core Frequency	100 – 250 MHz
Peak Performance	33.6 – 84.0 GOPS
Word Bit-width	16-bit Fixed-Point
Natively Supported CNN Shapes	Filter Width: 1 – 32 Filter Height: 1 – 12 Num. Filters: 1 – 1024 Num. Channels: 1 – 1024 Horz. Stride: 1–12 Vert. Stride: 1, 2, 4



AlexNet: For 2.66 GMACs [8 billion 16-bit inputs (**16GB**) and 2.7 billion outputs (**5.4GB**)], only requires **208.5MB** (buffer) and **15.4MB** (DRAM)

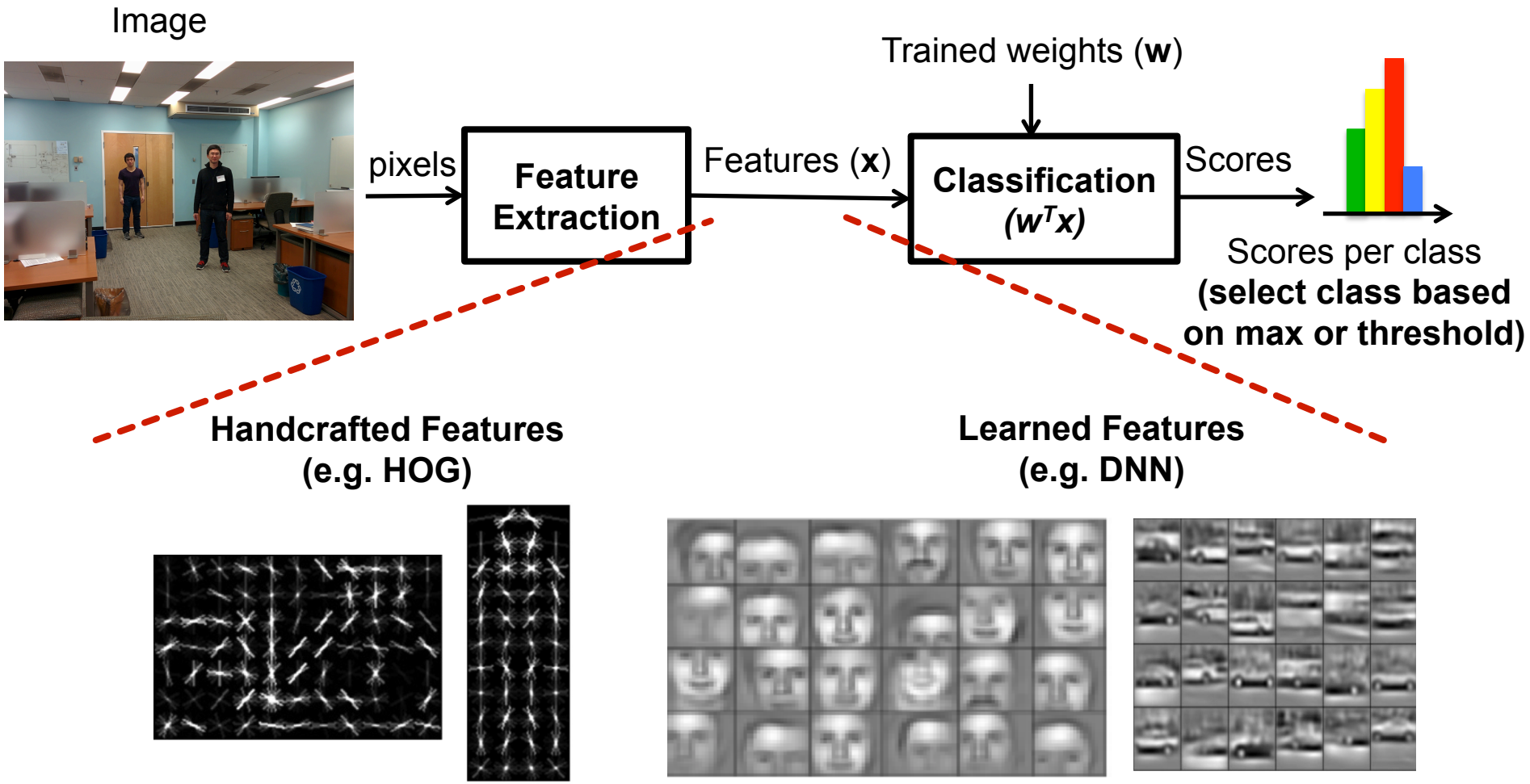
Comparison with GPU

	Eyeriss	NVIDIA TK1 (Jetson Kit)
Technology	65nm	28nm
Clock Rate	200MHz	852MHz
# Multipliers	168	192
On-Chip Storage	Buffer: 108KB Spad: 75.3KB	Shared Mem: 64KB Reg File: 256KB
Word Bit-Width	16b Fixed	32b Float
Throughput[1]	34.7 fps	68 fps
Measured Power	278 mW	Idle/Active[2]: 3.7W/10.2W
DRAM Bandwidth	127 MB/s	1120 MB/s [3]

1. AlexNet Convolutional Layers Only
2. Board Power
3. Modeled from [Tan, SC11]

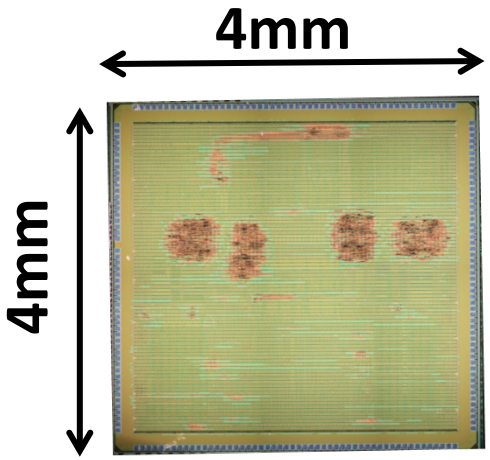
http://eyeriss.mit.edu

Machine Learning Pipeline (Inference)

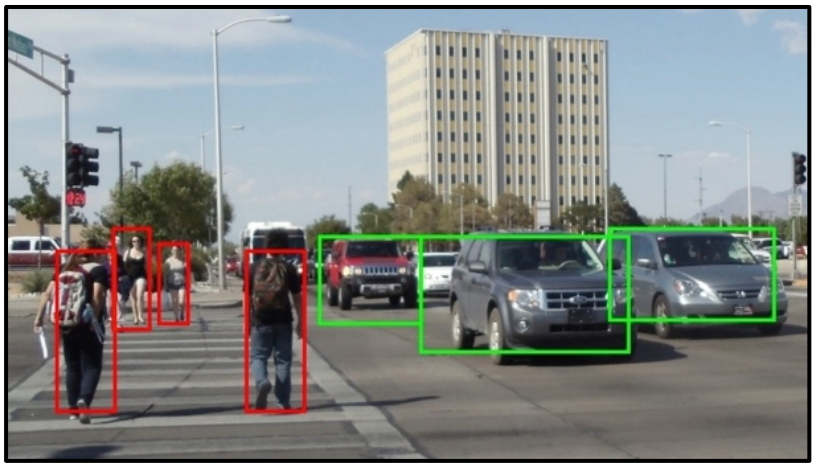


$$\text{Score} = \Sigma_n \, x_i \, w_i$$

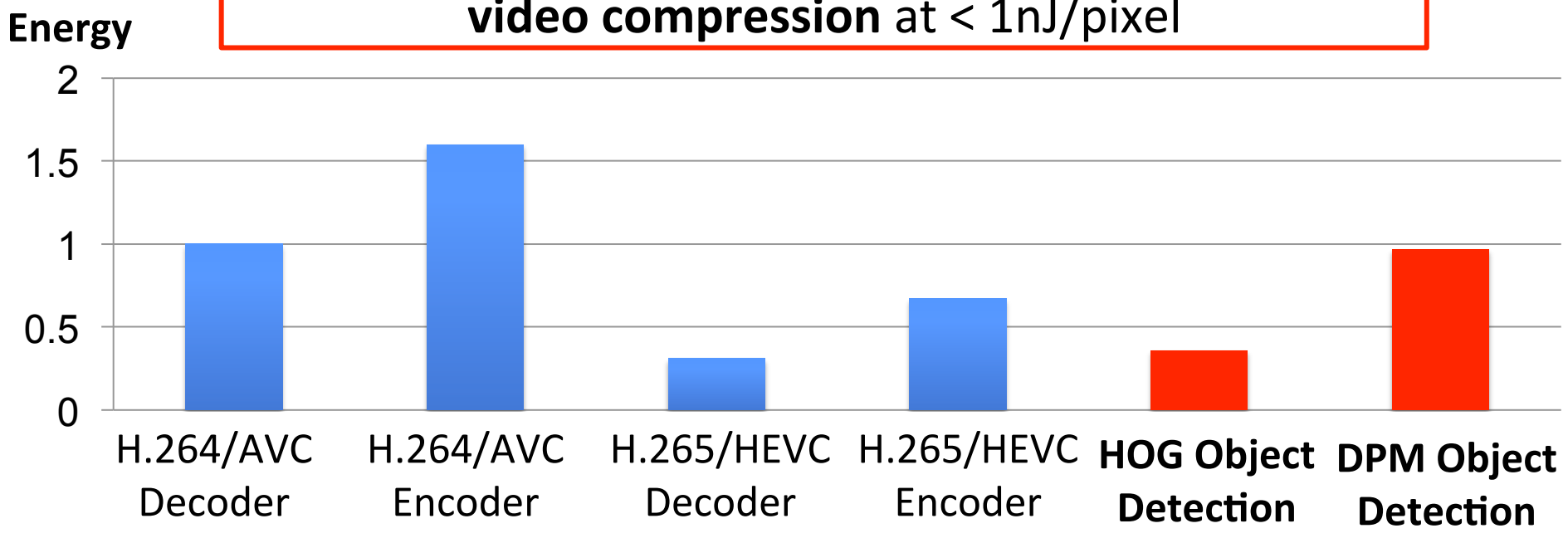
Energy-Efficient Object Detection



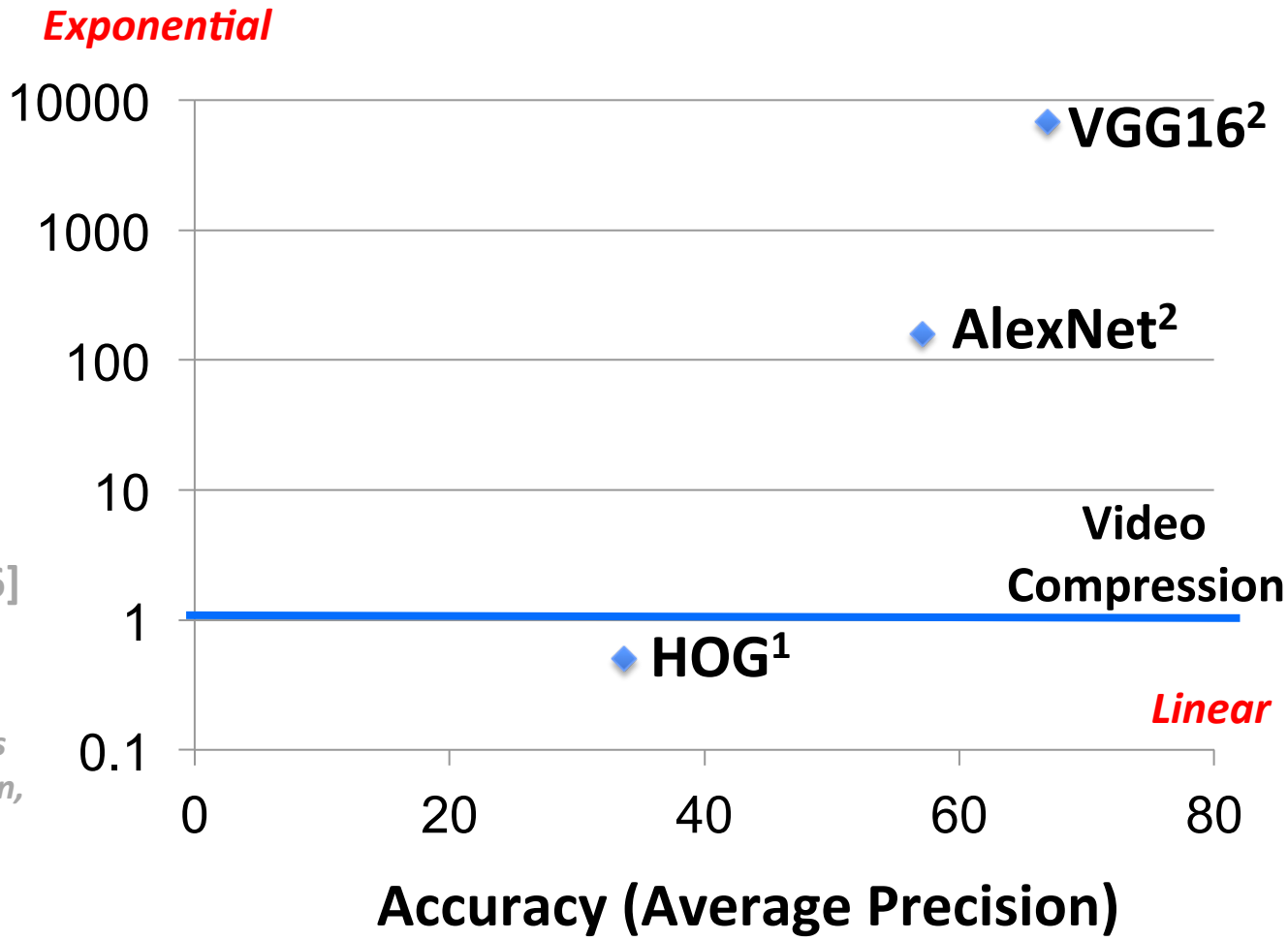
[Suleiman et al., VLSI 2016]



Enable **object detection** to be as **energy-efficient** as **video compression** at < 1nJ/pixel



Features: Energy vs. Accuracy



*Measured in 65nm**

1. [Suleiman, VLSI 2016]
2. [Chen, ISSCC 2016]

** Only feature extraction. Does not include data, augmentation, ensemble and classification energy, etc.*

Measured in on VOC 2007 Dataset

1. DPM v5 [Girshick, 2012]
2. Fast R-CNN [Girshick, CVPR 2015]

Opportunities in Joint Algorithm Hardware Design

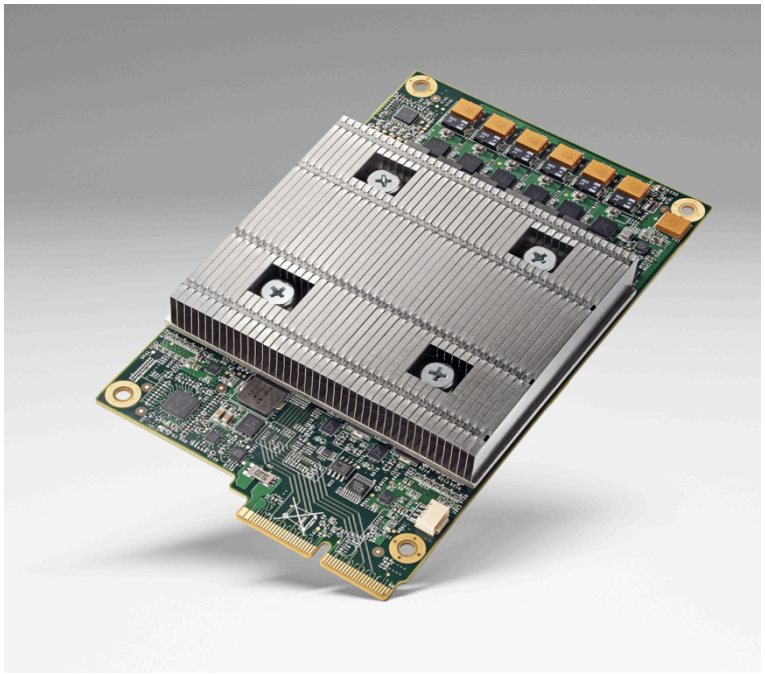
Approaches

- **Reduce size of operands for storage/compute**
  - Floating point → Fixed point
  - Bit-width reduction
  - Non-linear quantization

- **Reduce number of operations for storage/compute**
  - Exploit Activation Statistics (Compression)
  - Network Pruning
  - Compact Network Architectures

Commercial Products using 8-bit Integer

Nvidia's Pascal (2016)

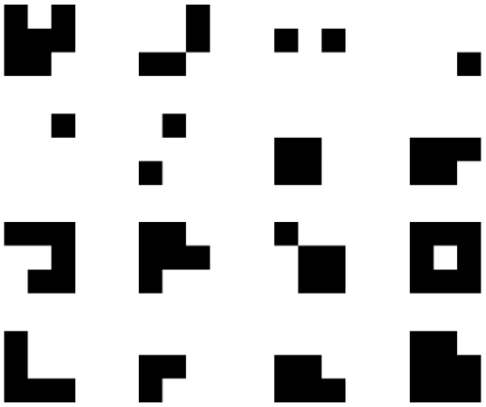


Google's TPU (2016)

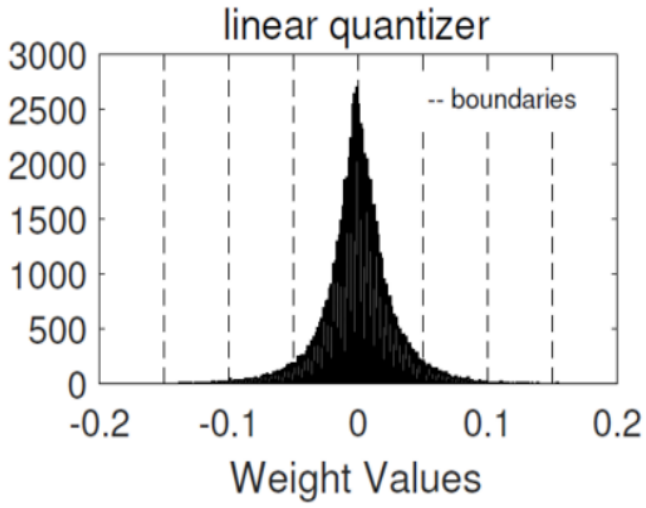
Reduced Precision in Research

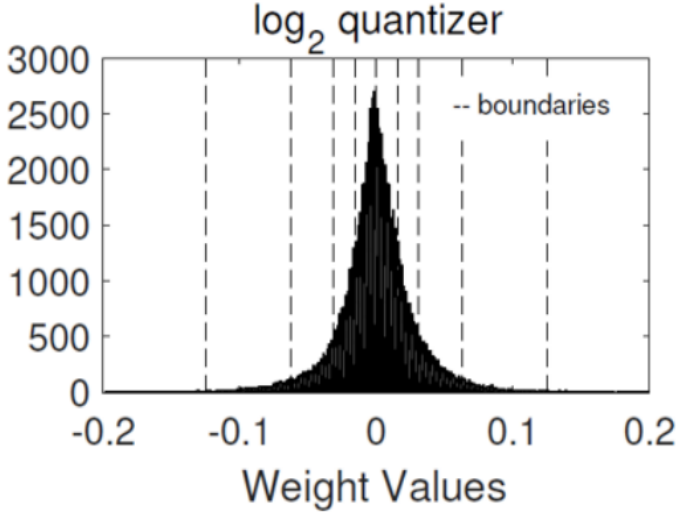
- **Reduce number of bits**
  - Binary Nets [Courbariaux, NIPS 2015]
- **Reduce number of unique weights**
  - Ternary Weight Nets [Li, arXiv 2016]
  - XNOR-Net [Rategari, ECCV 2016]
- **Non-Linear Quantization**
  - LogNet [Lee, ICASSP 2017]

Binary Filters



Log Domain Quantization



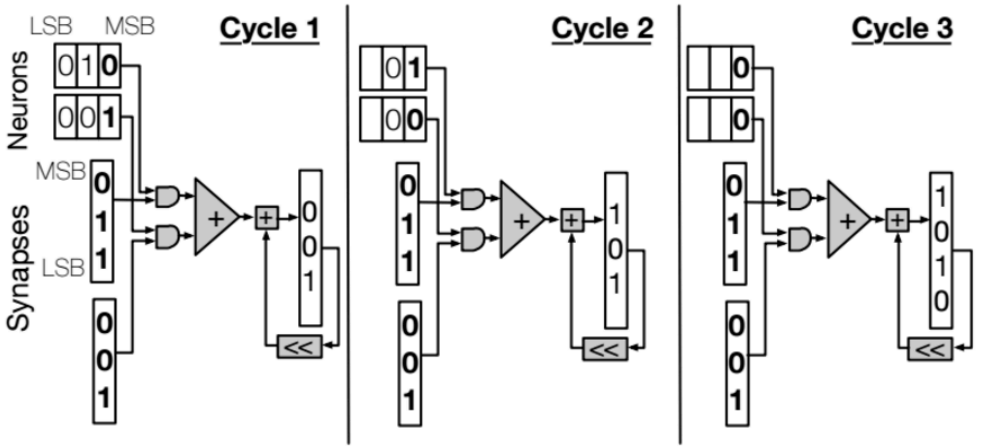


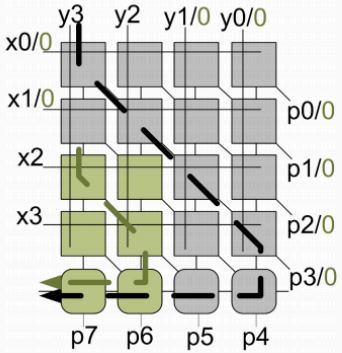
Reduced Precision Hardware

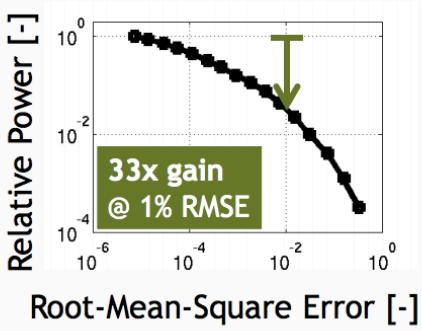
Stripes

[Judd et al., MICRO 2016]

Bit-serial processing for speed







$$P_{precise} = \alpha C f V^2 \quad \Rightarrow \quad P_{imprecise} = \frac{\alpha}{k_1} C f (\frac{V}{k_2})^2$$

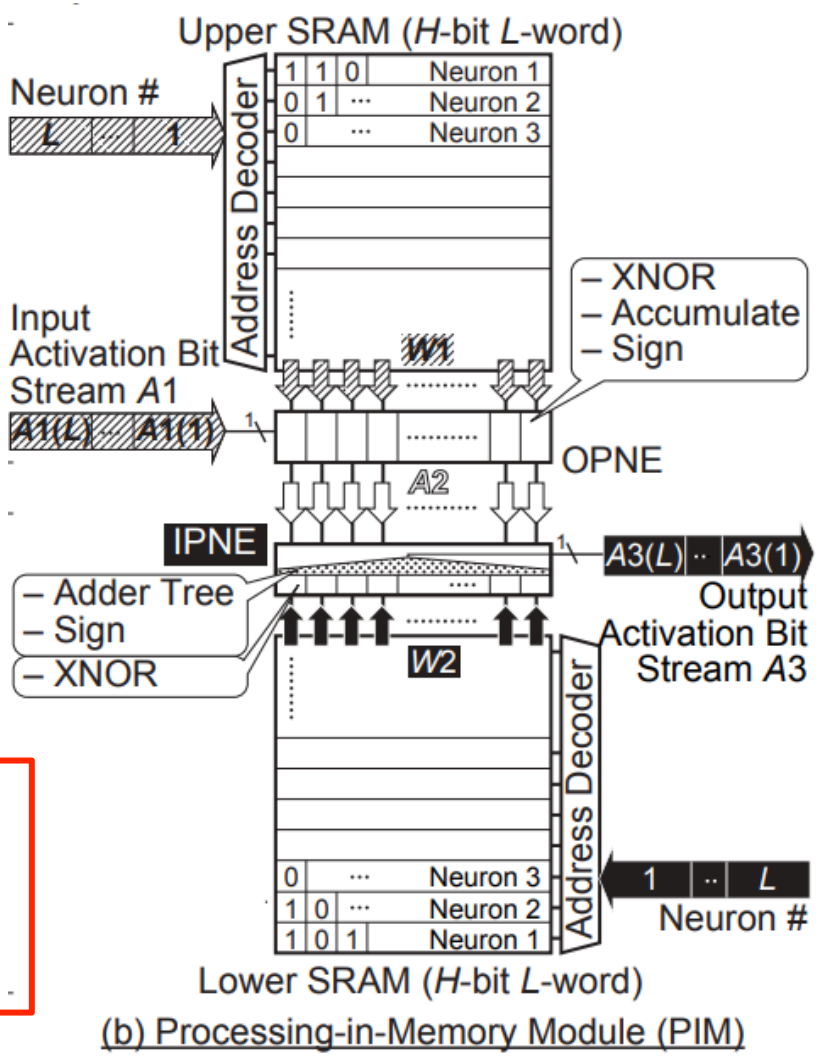
KU Leuven

[Moons et al., VLSI 2016]

Voltage scaling for energy savings

Binary/Ternary Net Hardware

- Examples
  - **YodaNN** (binary weights)
  - **BRein** (binary weights and activations)
  - **TrueNorth** (ternary weights and binary activations)

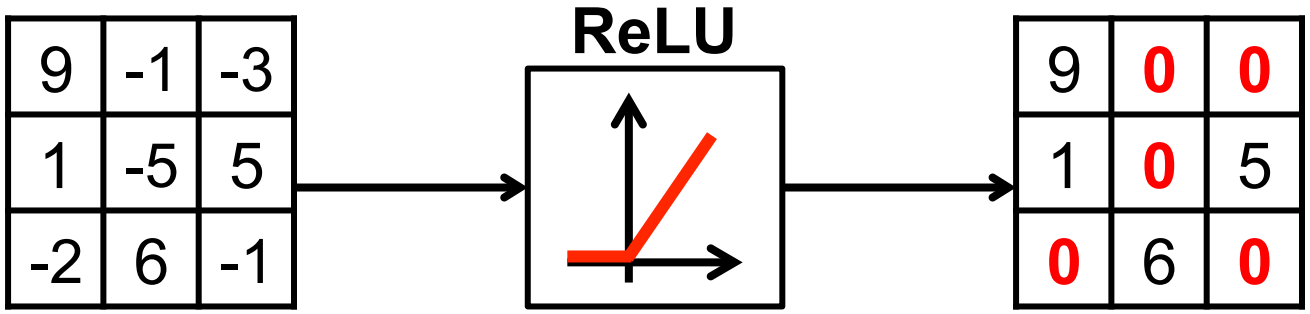


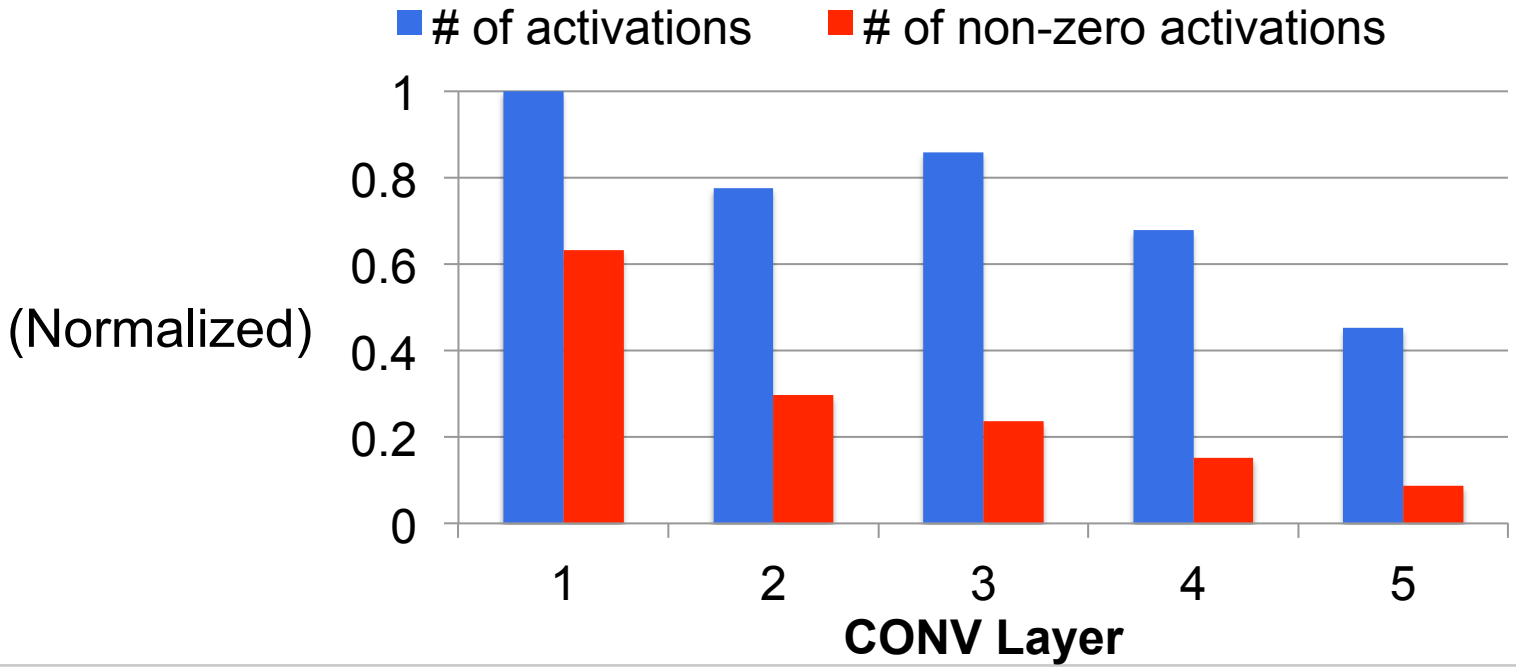
[BRein, VLSI 2017]

These designs tend not to support state-of-the-art DNN models (except YodaNN)

Sparsity in Feature Maps

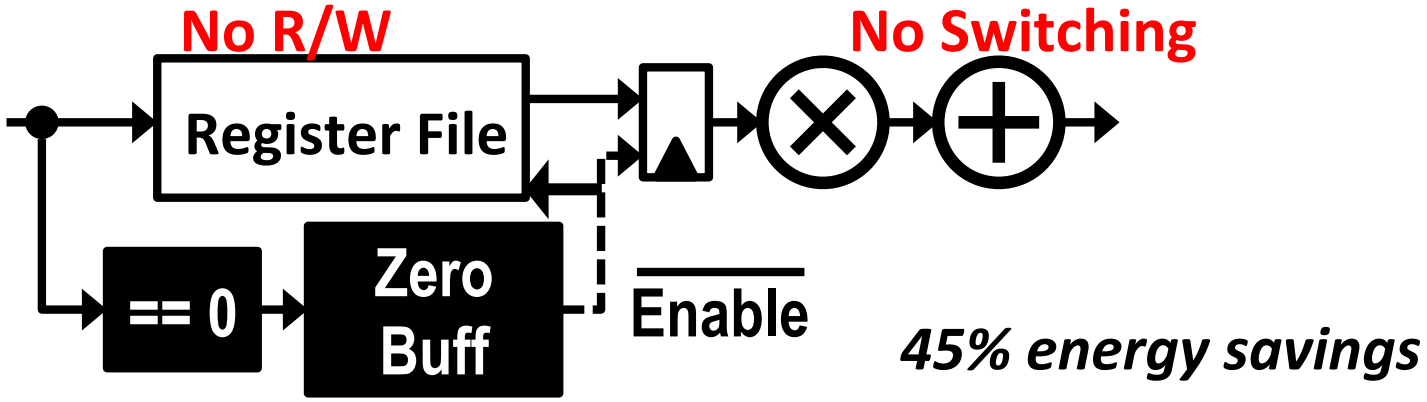
Many **zeros** in **output fmaps** after **ReLU**





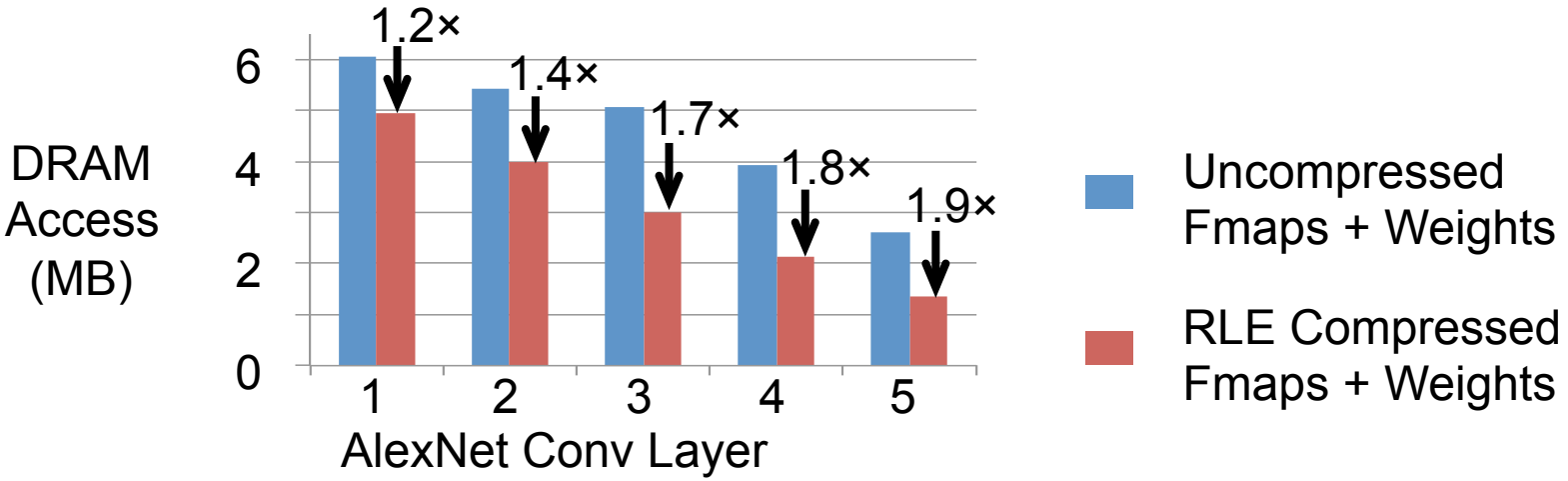
Exploit Sparsity

Method 1: Skip memory access and computation



45% energy savings

Method 2: Compress data to reduce storage and data movement

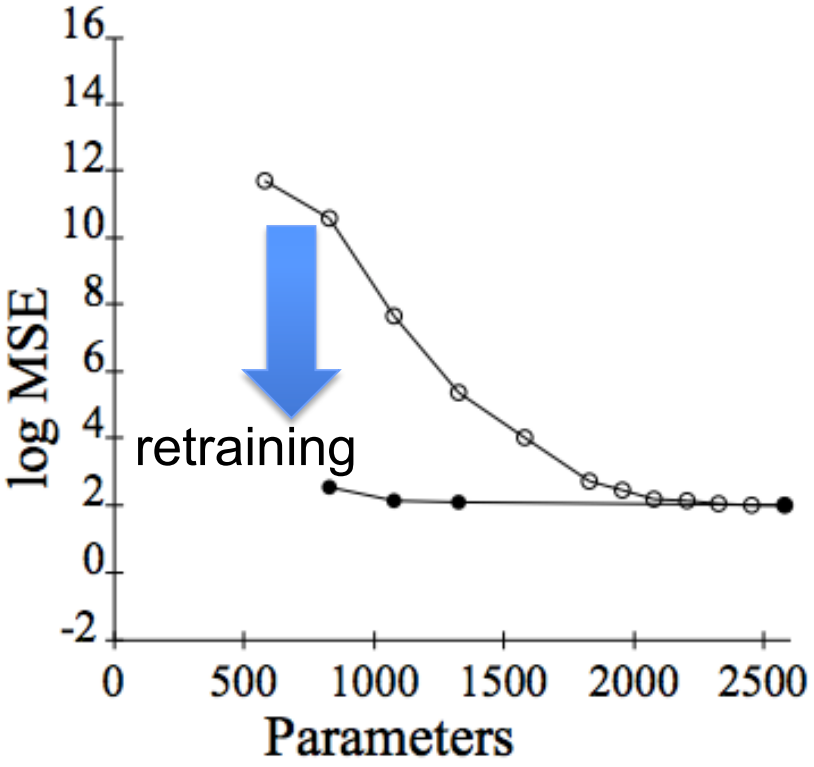


[Chen et al., ISSCC 2016]

Pruning – Make Weights Sparse

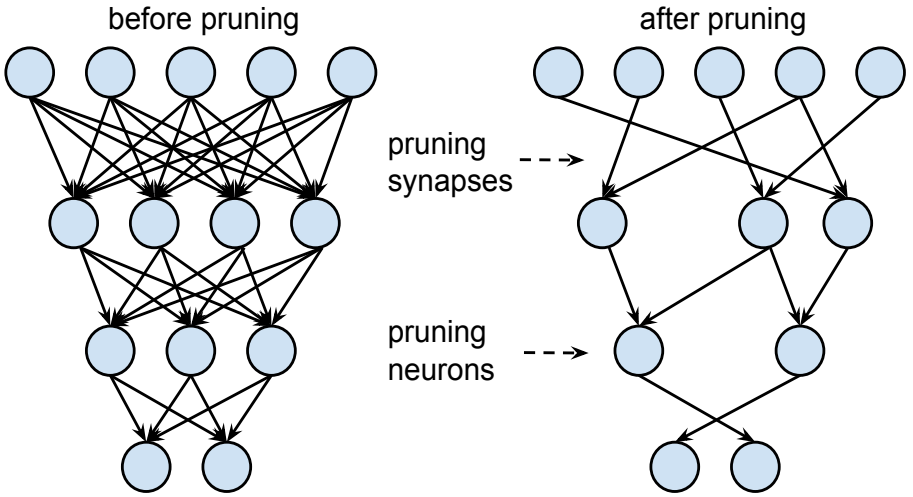
Optimal Brain Damage

[Lecun et al., NIPS 1989]



Prune DNN based on ***magnitude*** of weights

[Han et al., NIPS 2015]



Example: *AlexNet*

Weight Reduction:

CONV layers 2.7x, ***FC layers 9.9x***

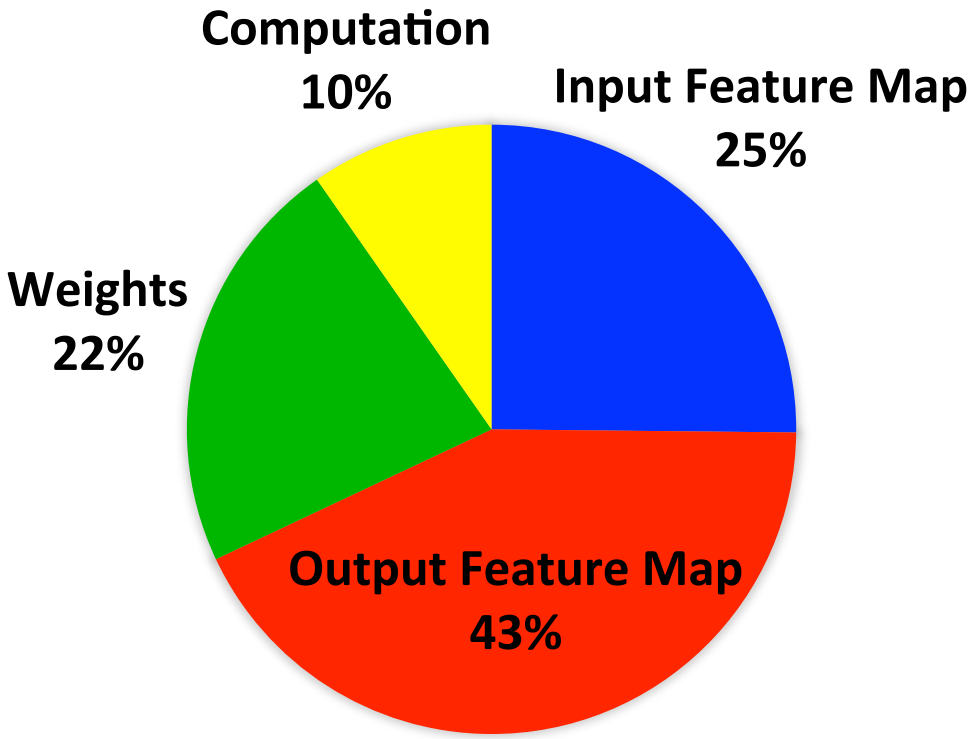
Overall Reduction:

Weights 9x, MACs 3x

Key Observations

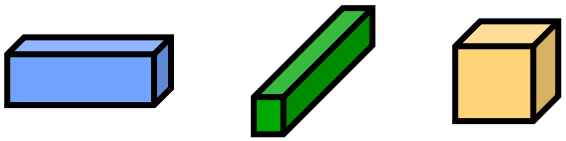
- Number of weights ***alone*** is not a good metric for energy
- **All data types** should be considered

Energy Consumption of GoogLeNet



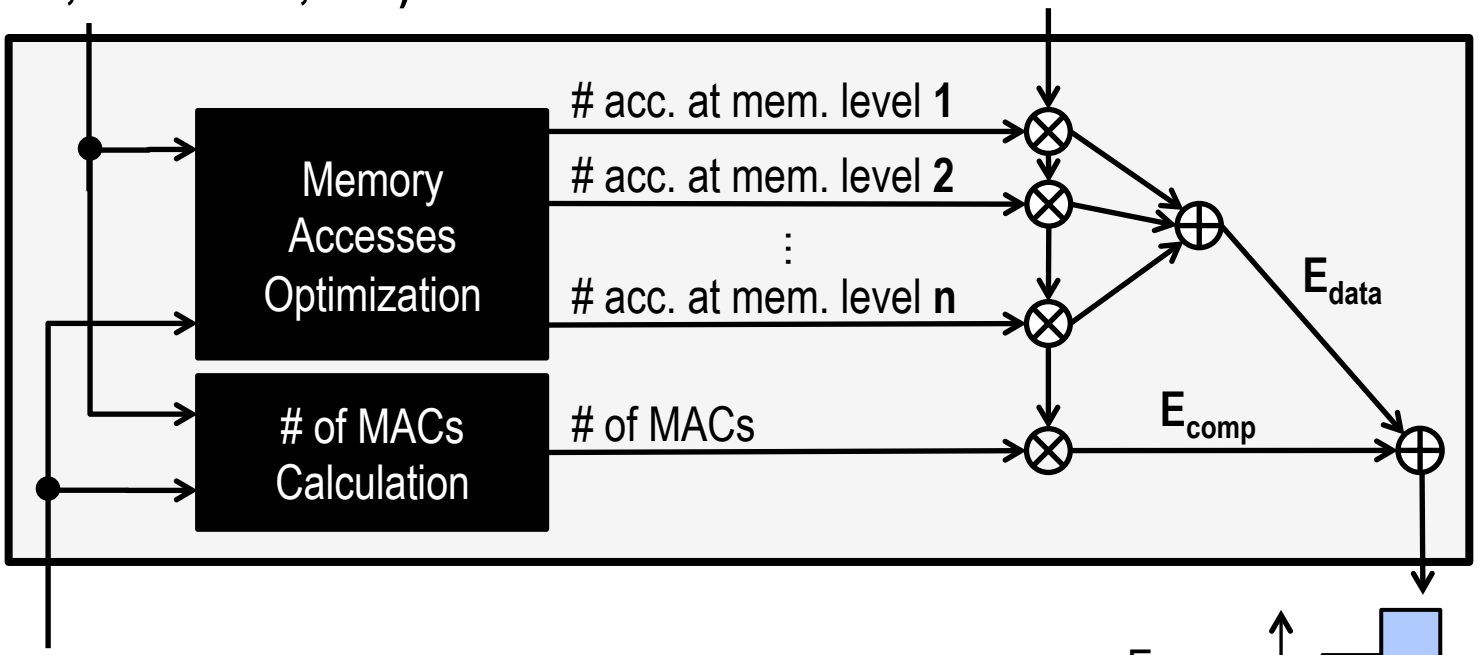
[Yang et al., CVPR 2017]

Energy-Evaluation Methodology



CNN Shape Configuration
(# of channels, # of filters, etc.)

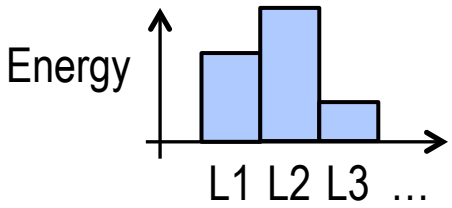
Hardware Energy Costs of each MAC and Memory Access



CNN Weights and Input Data

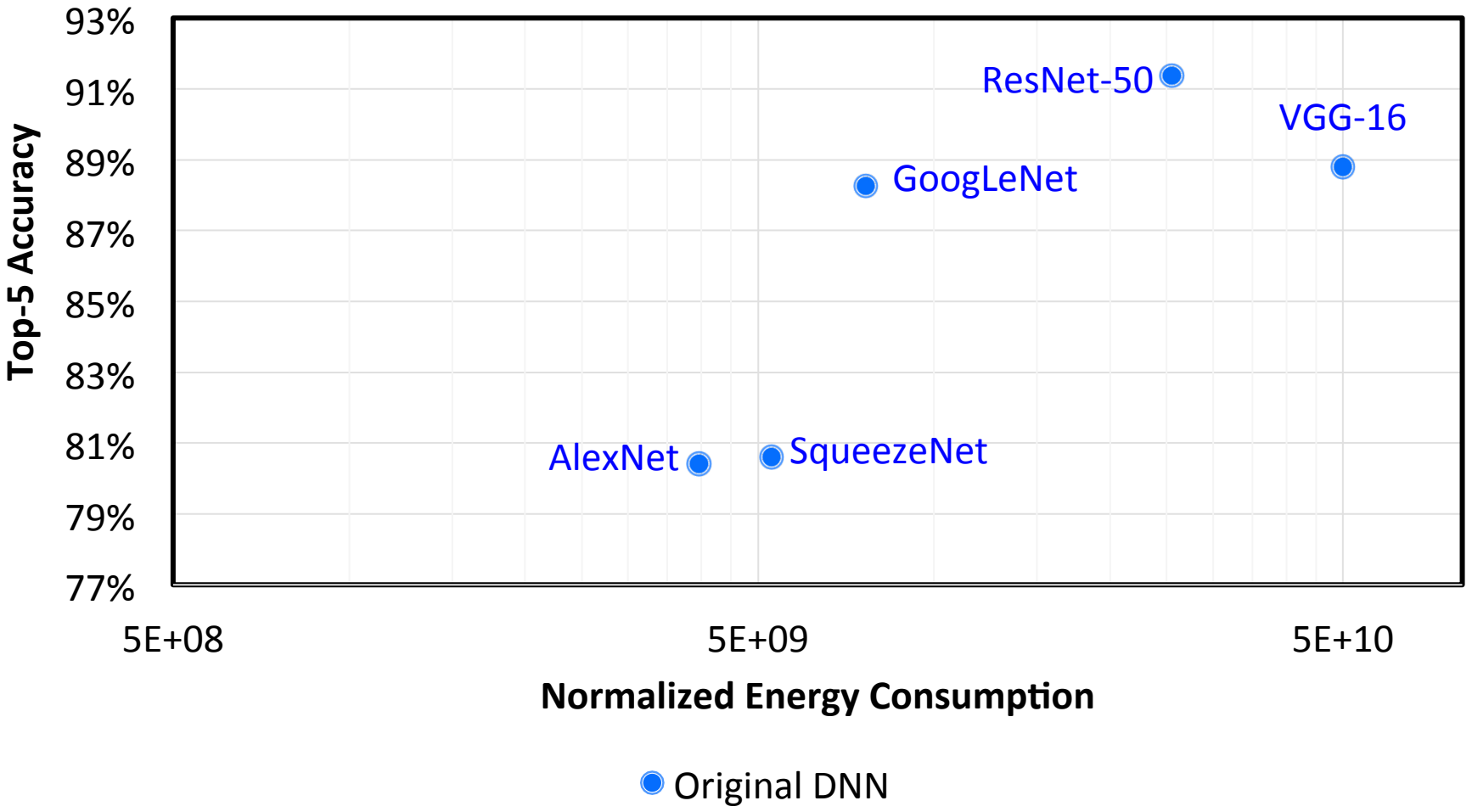
[0.3, 0, -0.4, 0.7, 0, 0, 0.1, …]

[Yang et al., CVPR 2017]



CNN Energy Consumption

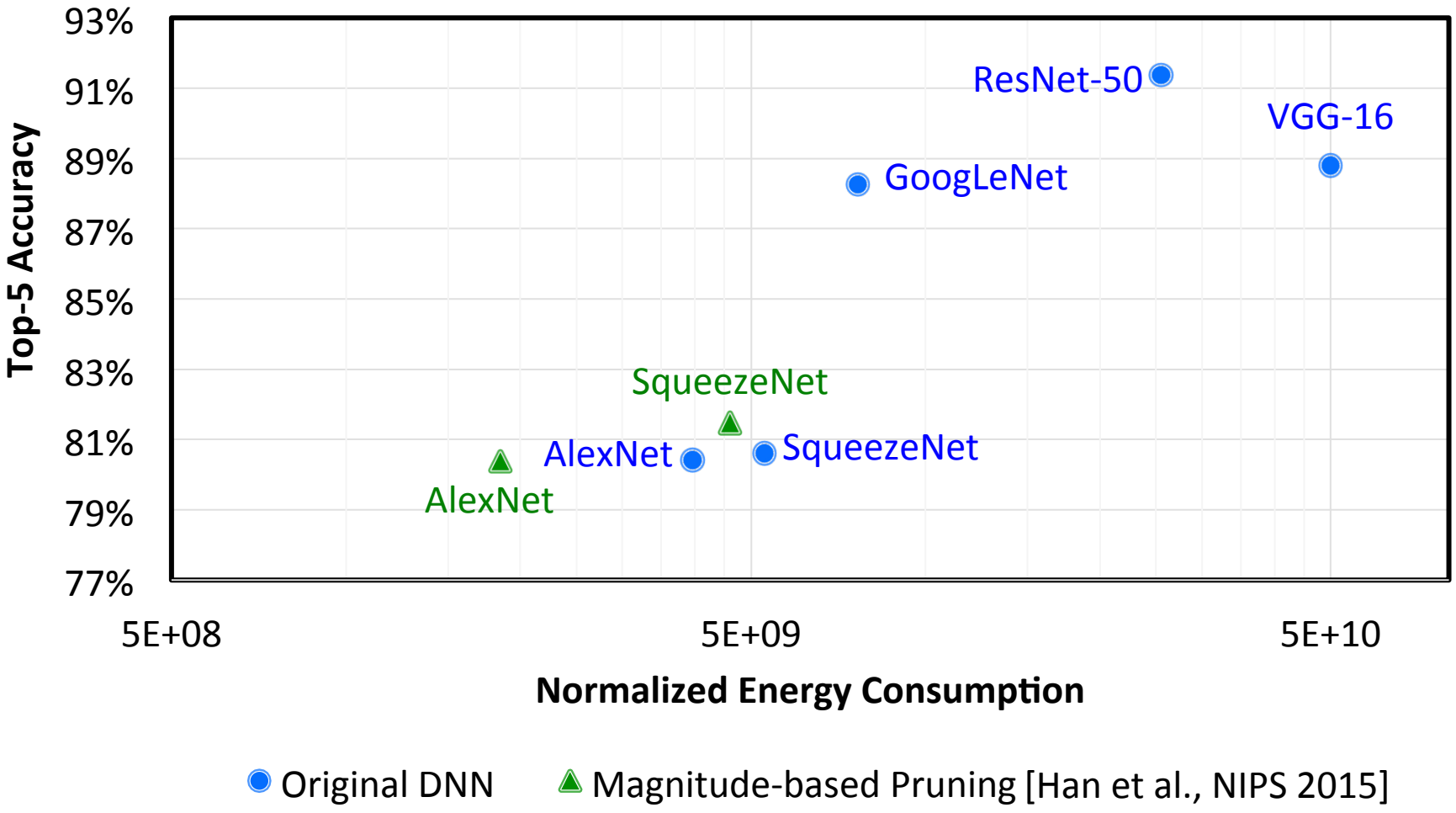
Energy Consumption of Existing DNNs



Deeper CNNs with fewer weights do not necessarily consume less energy than shallower CNNs with more weights

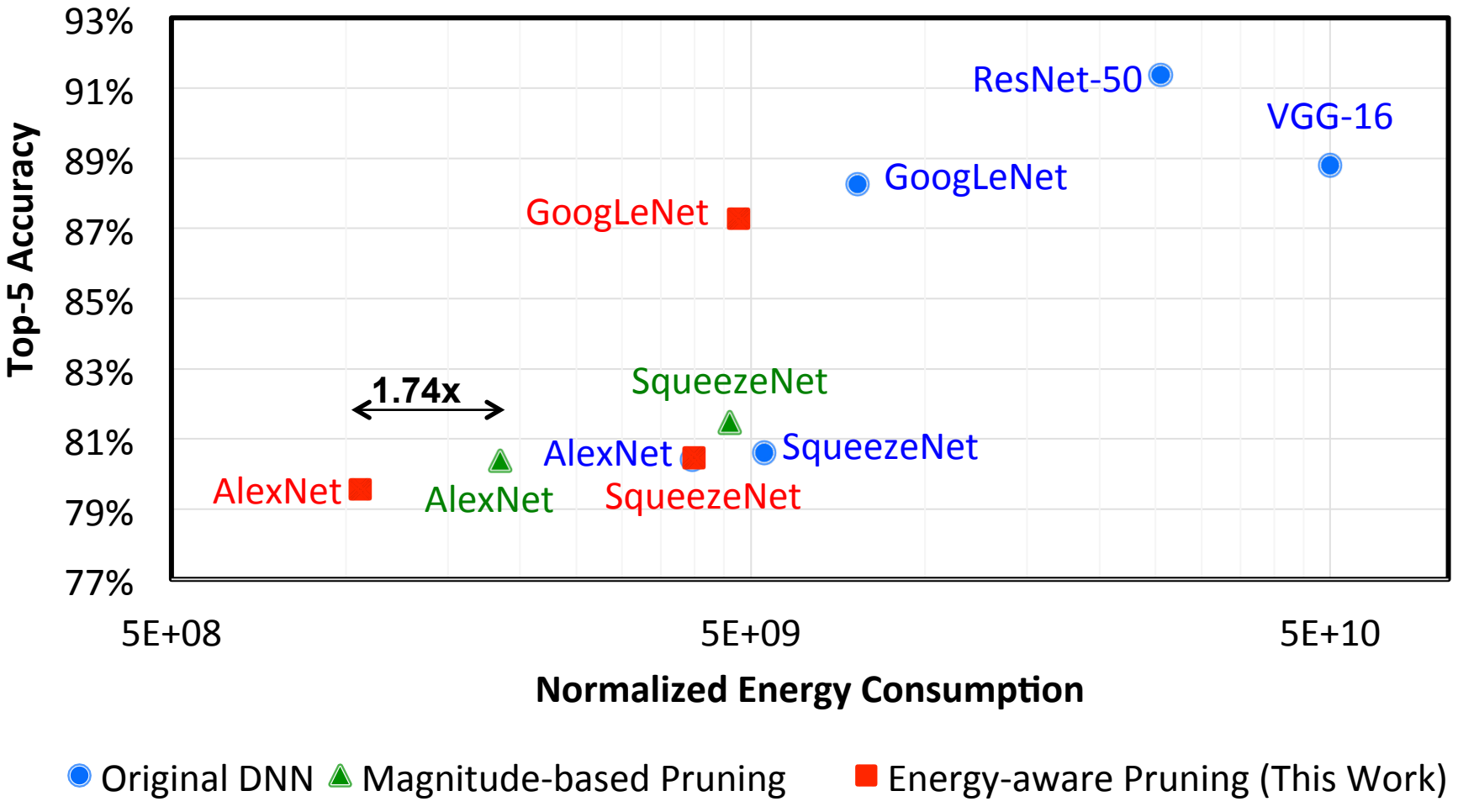
[Yang et al., CVPR 2017]

Magnitude-based Weight Pruning



Reduce number of weights by **removing small magnitude weights**

Energy-Aware Pruning



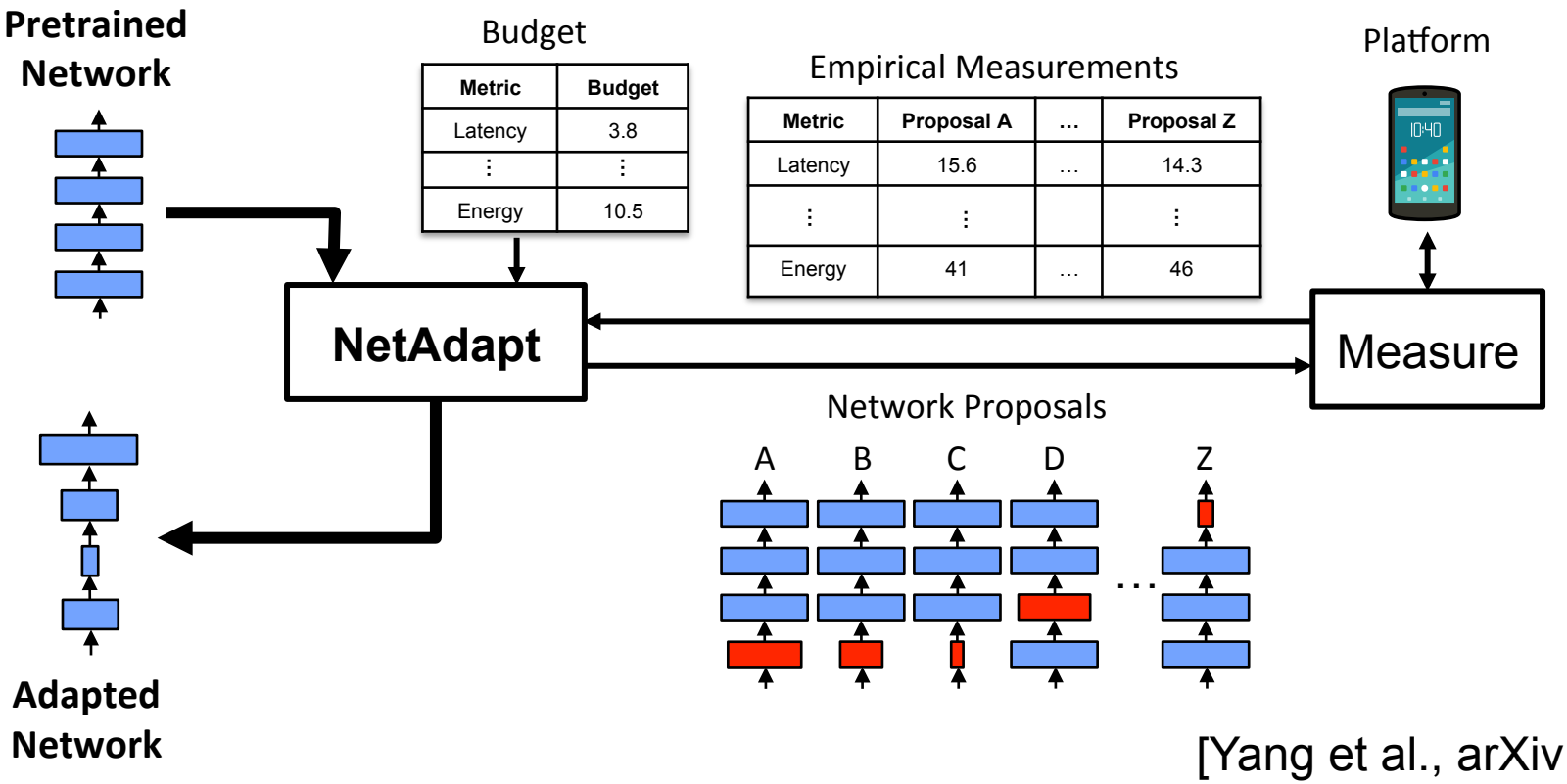
Remove weights from layers **in order of highest to lowest energy**

3.7x reduction in AlexNet / 1.6x reduction in GoogLeNet

[Yang et al., CVPR 2017]

NetAdapt: Platform-Aware DNN Adaptation

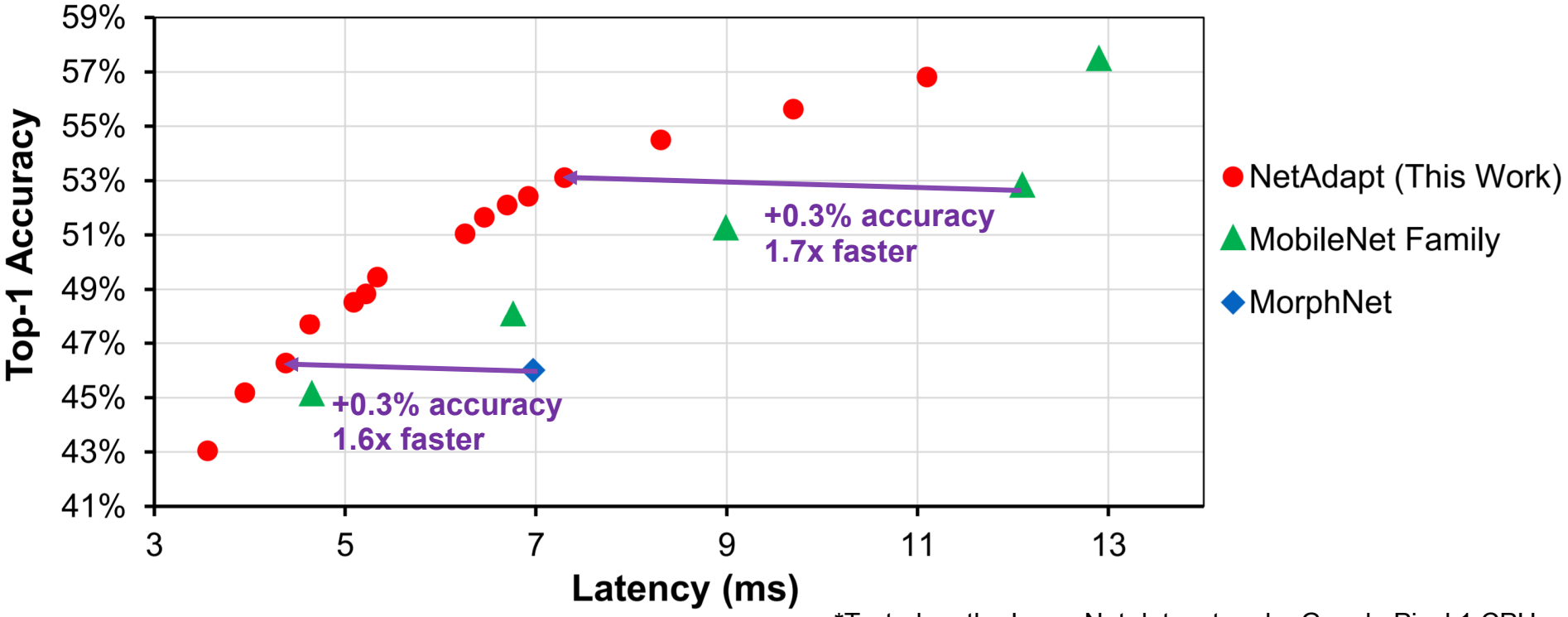
- **Automatically adapt DNN** to a mobile platform to reach a target latency or energy budget
- Use **empirical measurements** to guide optimization (avoid modeling of tool chain or platform architecture)



[Yang et al., arXiv 2018]

Improved Latency vs. Accuracy Tradeoff

- NetAdapt boosts **the real inference speed** of MobileNet by up to 1.7x with higher accuracy



*Tested on the ImageNet dataset and a Google Pixel 1 CPU

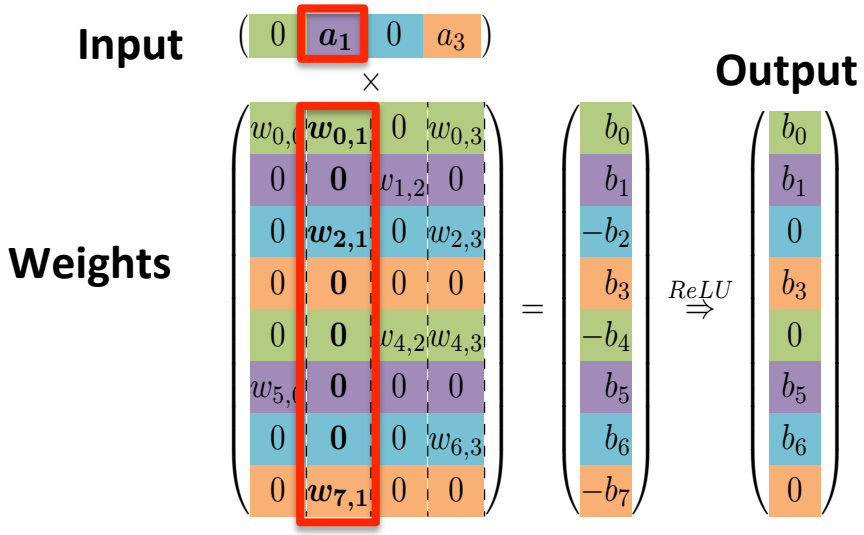
Reference:
MobileNet: Howard et al, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", arXiv 2017
MorphNet: Gordon et al., "Morphnet: Fast & simple resource-constrained structure learning of deep networks", CVPR 2018

Sparse Hardware

EIE

[Han et al., ISCA 2016]

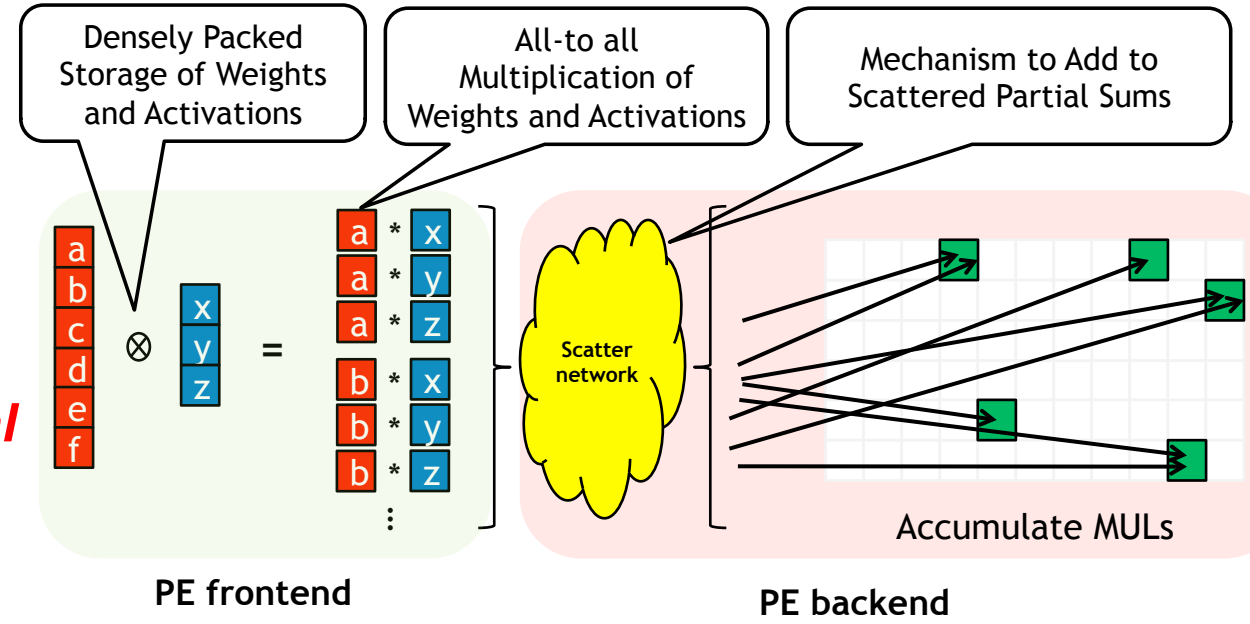
Supports Fully Connected Layers Only



SCNN

[Parashar et al., ISCA 2017]

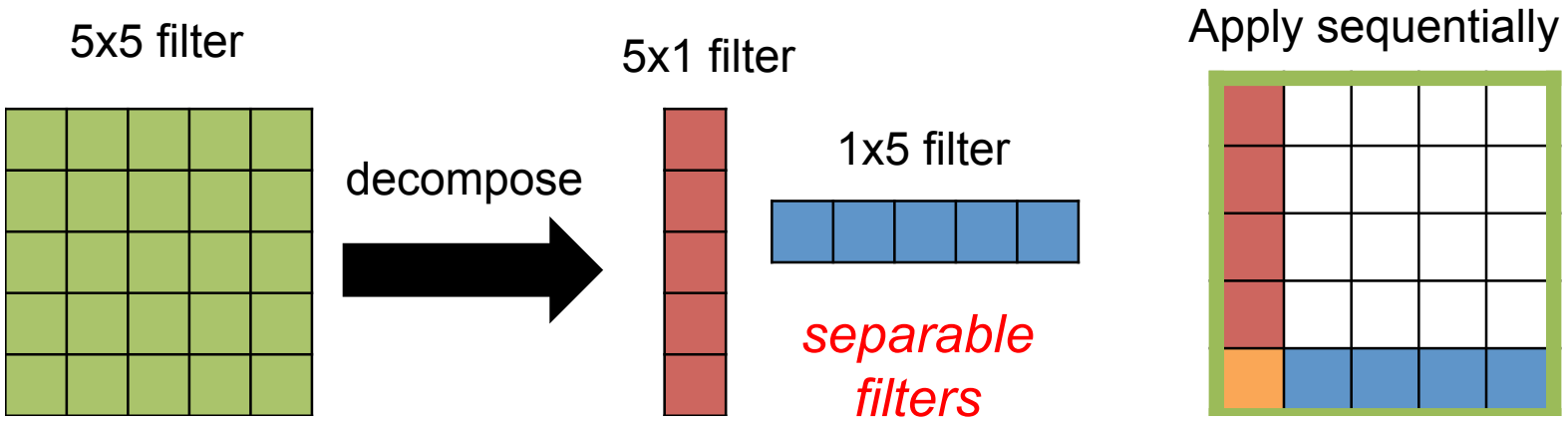
Supports Convolutional Layers Only



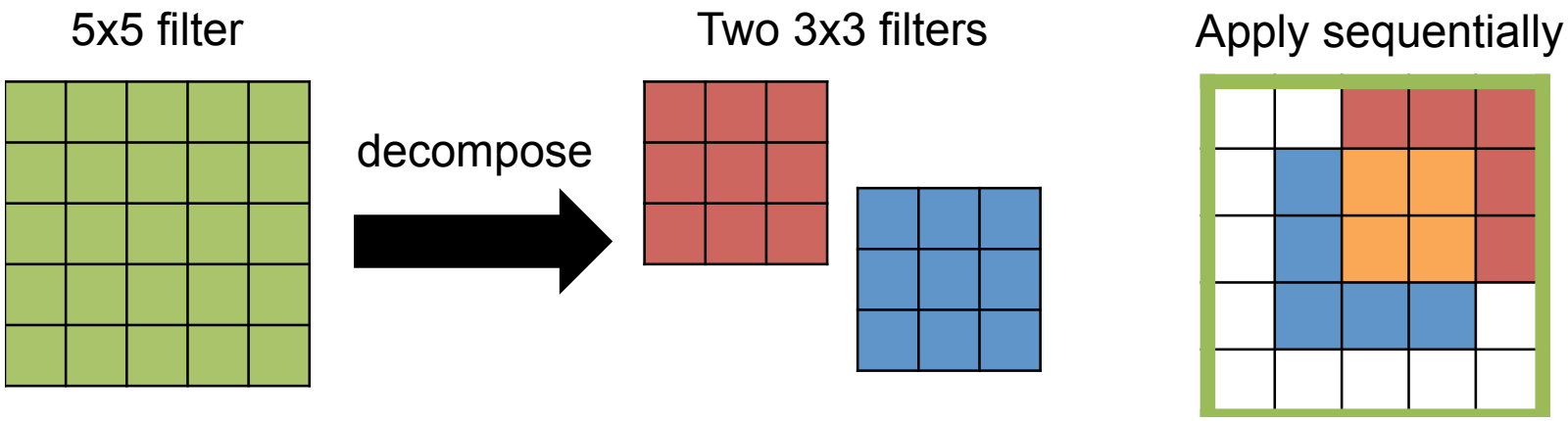
Network Architecture Design

Build Network with series of Small Filters

GoogleNet/Inception v3

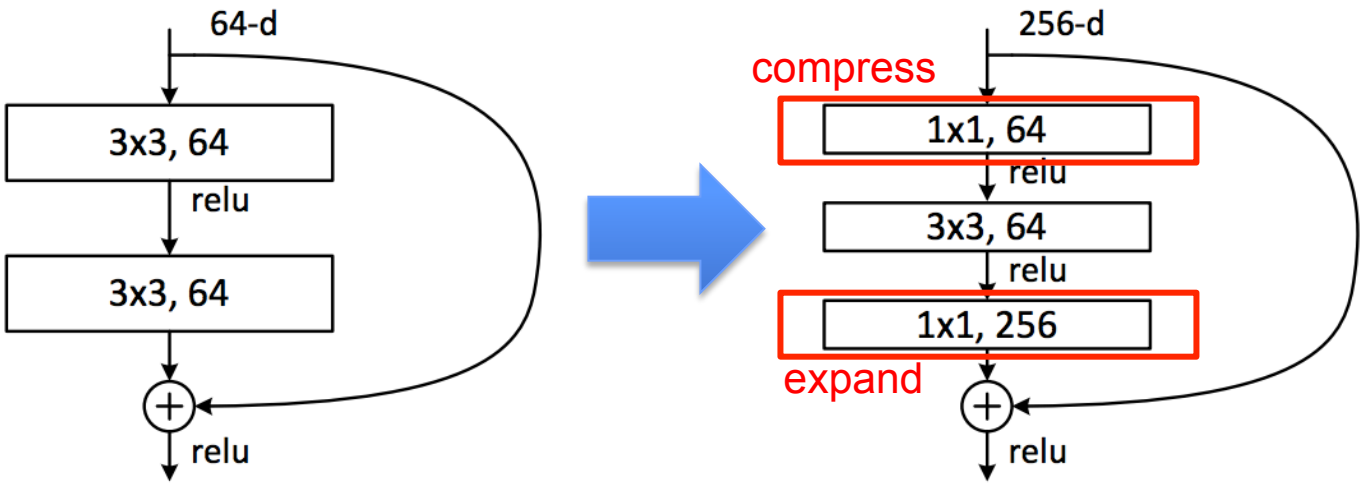


VGG-16

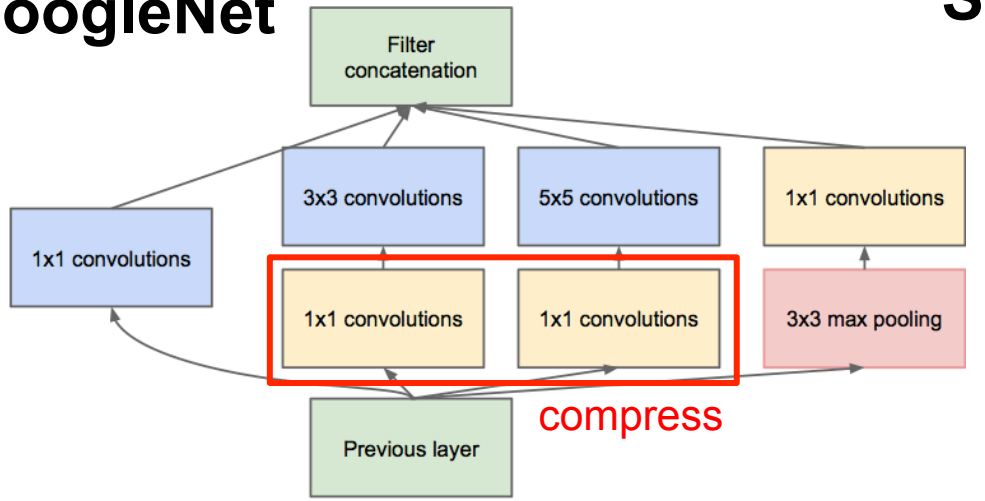


1x1 Bottleneck in Popular DNN models

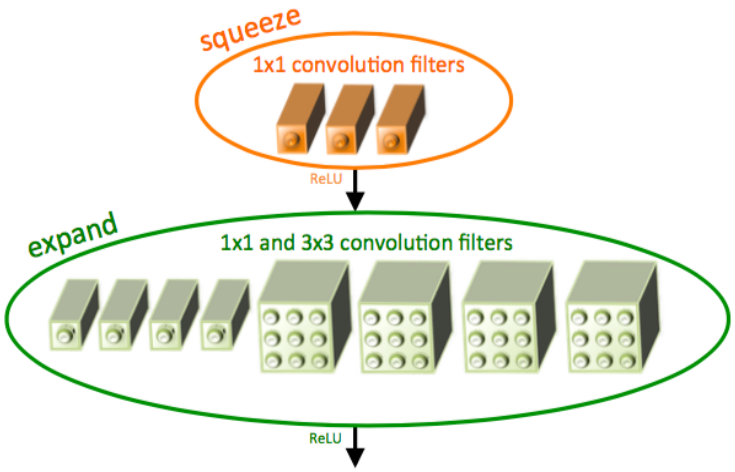
ResNet



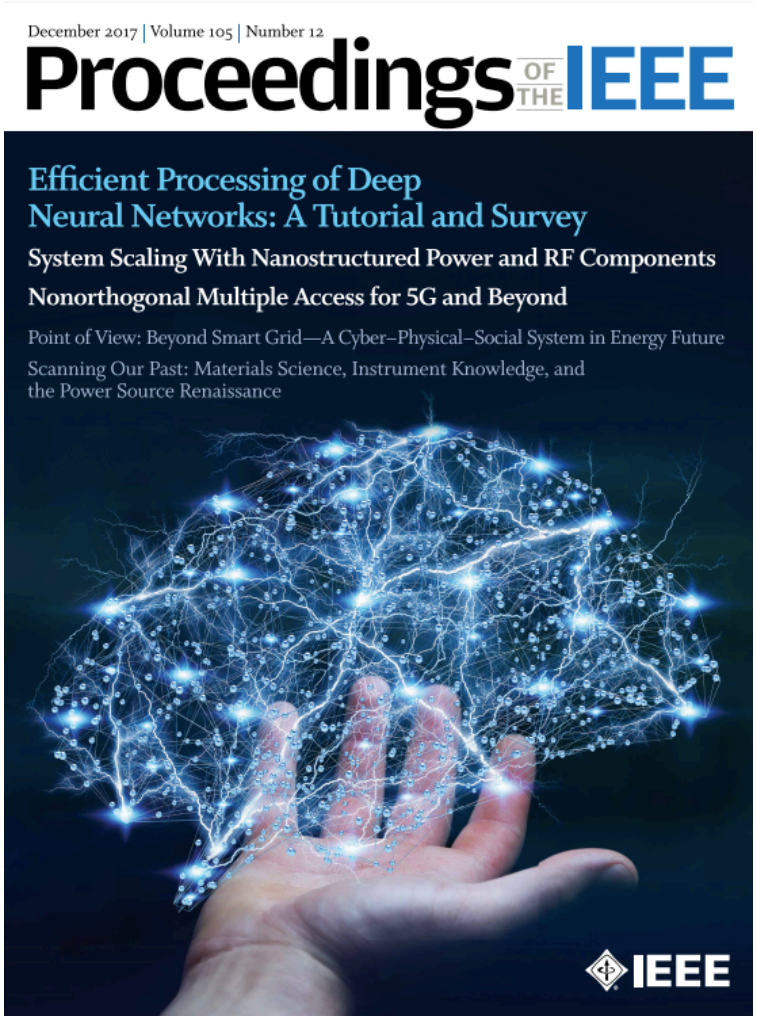
GoogleNet



SqueezeNet



Tutorial Material on Efficient DNNs



Tutorial on Hardware Architectures for Deep Neural Networks

MICRO-49 (Full Day: October 16, 2016)

Joel Emer — MIT, NVIDIA
Vivienne Sze — MIT
Yu-Hsin Chen — MIT

Email: eyeriss at mit dot edu

Updates

Follow @eems_mit or subscribe to our mailing list for updates on the Tutorial (e.g. notification of when slides will be posted)

Overview

Deep neural networks (DNNs) are currently widely used for many AI applications including computer vision, speech recognition, robotics, etc. While DNNs deliver state-of-the-art accuracy on many AI tasks, it comes at the cost of high computational complexity. Accordingly, designing efficient hardware architectures for deep neural networks is an important step towards enabling the wide deployment of DNNs in AI systems.

http://eyeriss.mit.edu/tutorial.html

V. Sze, Y.-H. Chen, T-J. Yang, J. Emer, "*Efficient Processing of Deep Neural Networks: A Tutorial and Survey,*" Proceedings of the IEEE, 2017

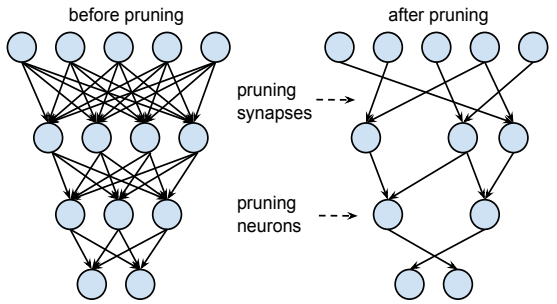
Need More Comprehensive Benchmarks

Processors should support a **diverse set of DNNs** that utilize different techniques

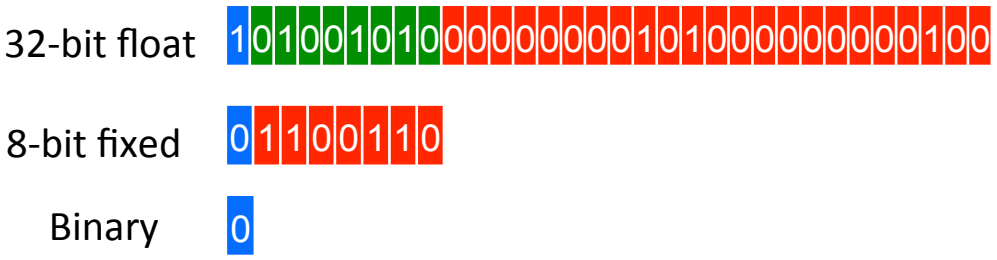
Example:

- Sparse **and** Dense
- Large **and** Compact network architectures
- Different Layers (e.g., CONV **and** FC)
- Variable Bit-width

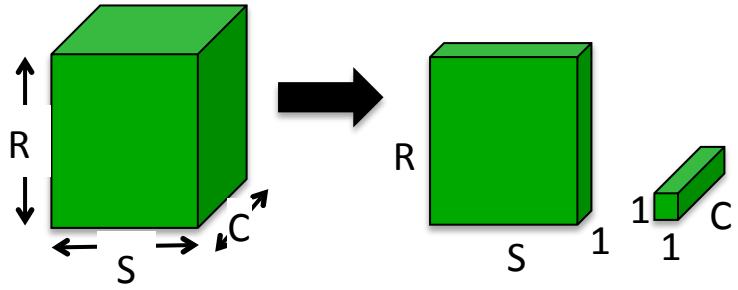
Network Pruning



Reduce Precision



Compact Network Architecture

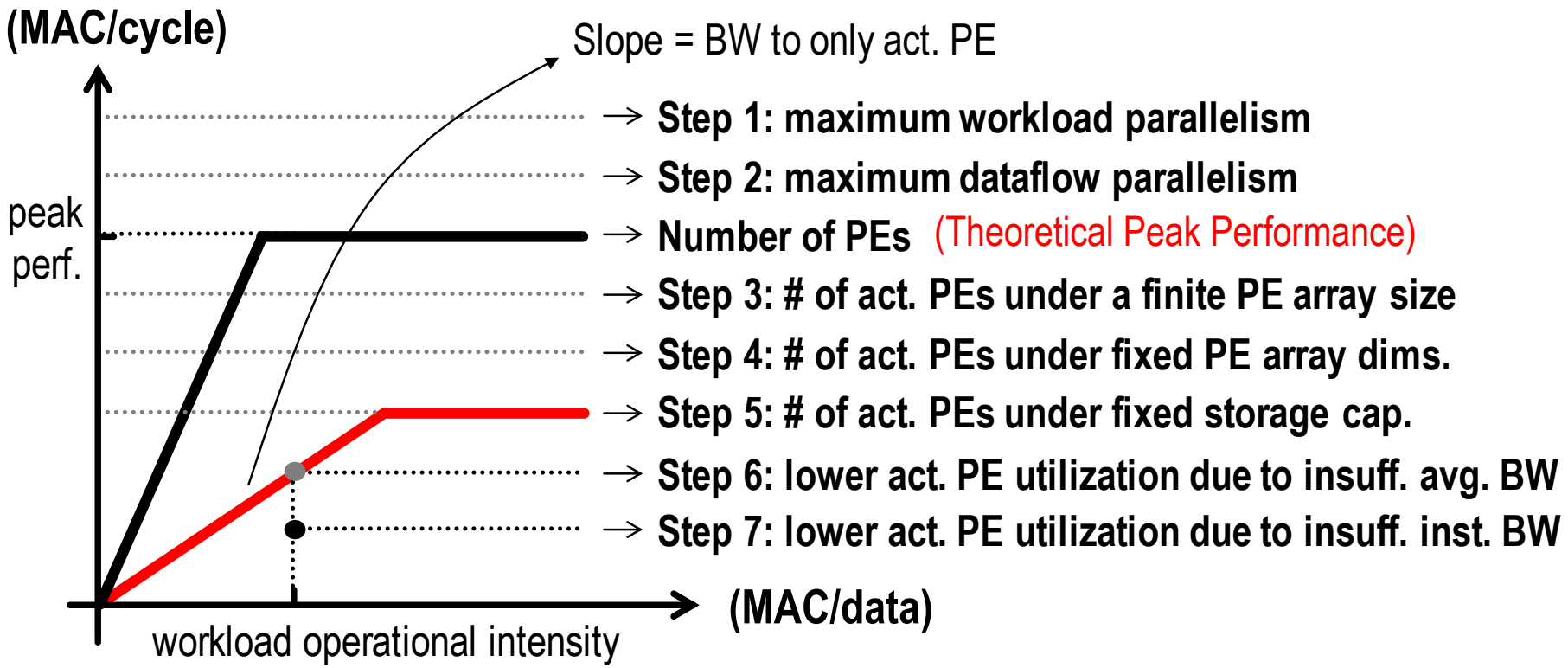


[Chen et al., SysML 2018]

Eyexam: Understanding Sources of Inefficiencies in DNN Accelerators

A systematic way to evaluate how each architectural decision affects performance (throughput) for a given DNN workload

Tightens the roofline model



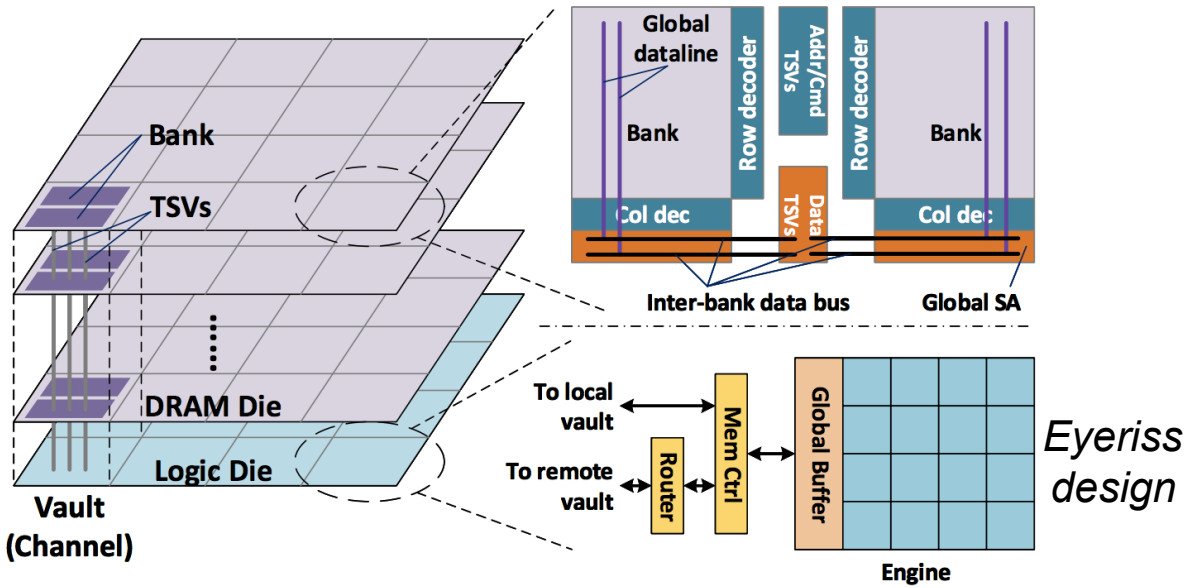
[Chen et al., In Submission]

Opportunities in Memories and Devices

Advanced Memory Technologies

Many new memories and devices explored to reduce data movement

Stacked DRAM



Eyeriss design

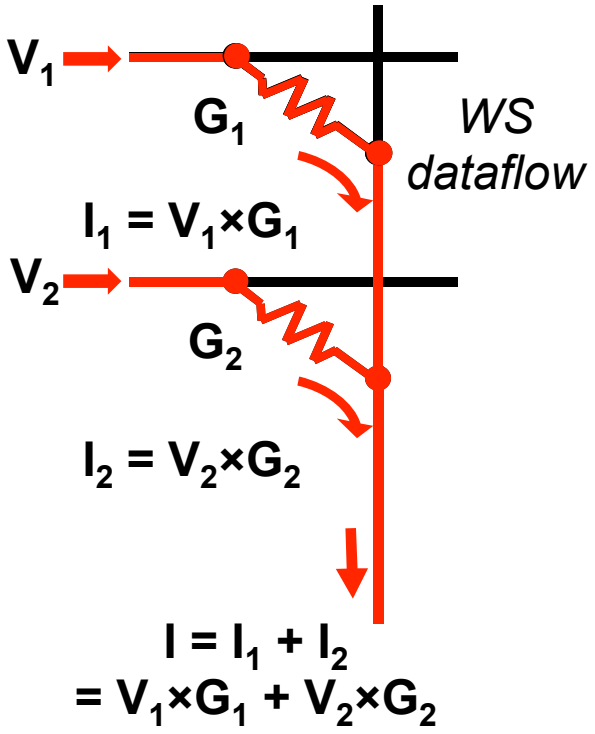
[Gao et al., Tetris, ASPLOS 2017]

[Kim et al., NeuroCube, ISCA 2016]

eDRAM

[Chen et al., DaDianNao, MICRO 2014]

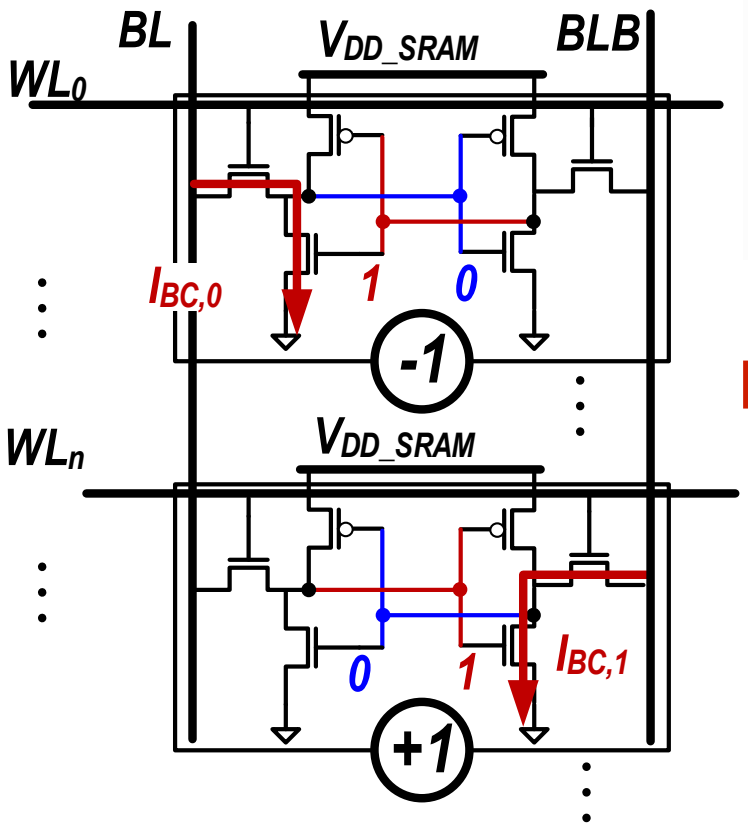
Non-Volatile Resistive Memories



[Shafiee et al., ISCA 2016]

[Chi et al., PRIME, ISCA 2016]

Binary Weight Classifier in SRAM



$$V_{BL} - V_{BLB} = \sum_{n=0}^{127} w_n \times \Delta V_{BL/B,n}$$

$$\approx \sum_{n=0}^{127} w_n \times V_{WL,n}$$

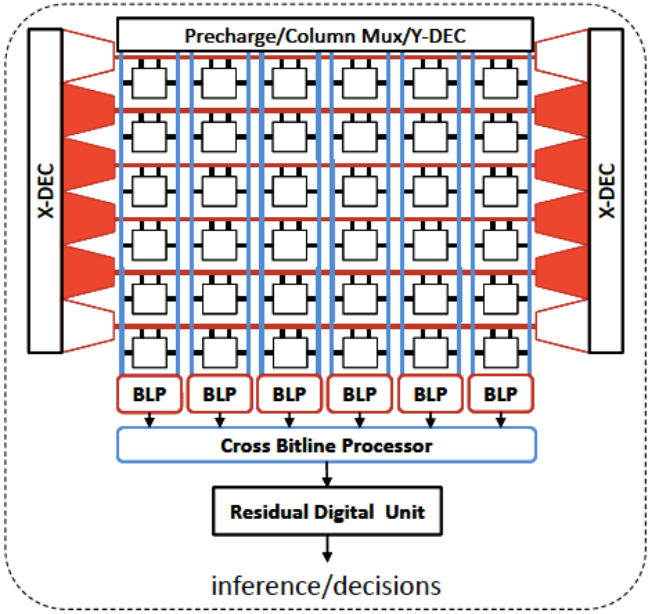
Weight restricted to ± 1

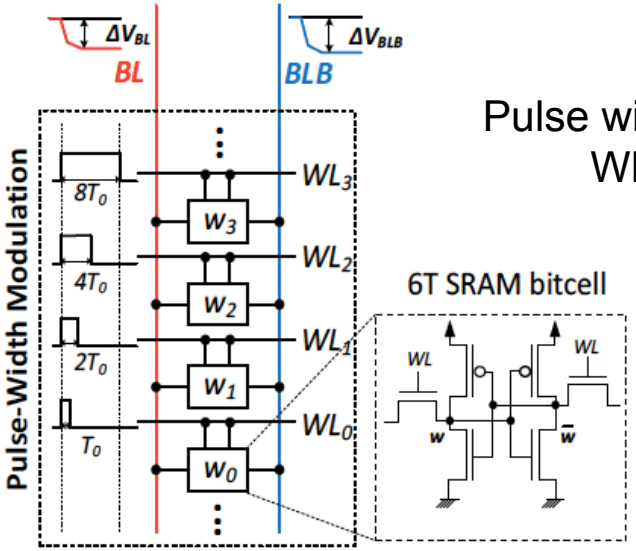
Weak because:

1. **Weights restricted to be +/-1**
2. **Bit-cell discharge subject to variation, nonlinearity**

[Zhang et al., VLSI 2016]

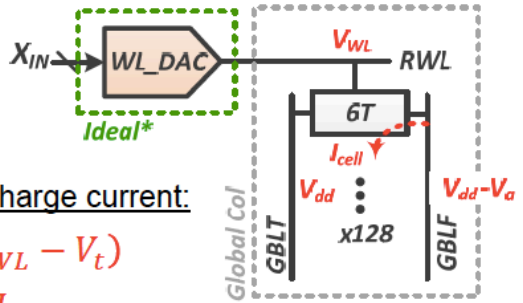
More Compute In Memory





Pulse width modulation on WL (activation)

[S. Gonugondla, ISSCC 2018]



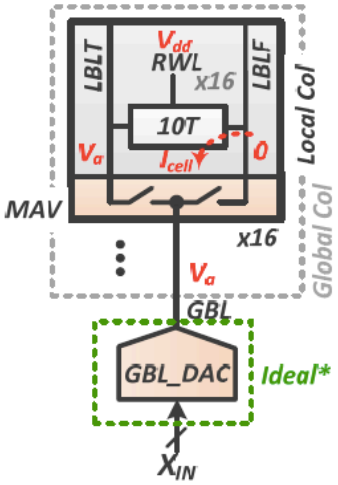
Bit-cell discharge current:

$I_{cell} \propto (V_{WL} - V_t)$

$V_a \propto I_{cell}$

***Conventional* [4]**

V_a varies widely due to I_{cell} variation



**Assumption:*
Both DACs are ideal

Directly apply V_a on GBL(LBL) by DAC

$V_a \not\propto I_{cell}$

Proposed

V_a has no variation due to I_{cell}

Apply Va (activation) to BL rather than WL

[A. Biswas, Conv-RAM, ISSCC 2018]

Benchmarking Metrics for DNN Hardware

How can we compare designs?

V. Sze, Y.-H. Chen, T-J. Yang, J. Emer,
"***Efficient Processing of Deep Neural Networks: A Tutorial and Survey***,"
Proceedings of the IEEE, Dec. 2017

Metrics for DNN Hardware

- **Accuracy**
  - Quality of result for a given task
- **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**
  - Edge and embedded devices have limited battery capacity
  - Data centers have stringent power ceilings due to cooling costs
- **Hardware Cost**
  - $$$

Specifications to Evaluate Metrics

- **Accuracy**
  - Difficulty of dataset and/or task should be considered
- **Throughput**
  - Number of cores (include utilization along with peak performance)
  - Runtime for running specific DNN models
- **Latency**
  - Include batch size used in evaluation
- **Energy and Power**
  - Power consumption for running specific DNN models
  - Include external memory access
- **Hardware Cost**
  - On-chip storage, number of cores, chip area + process technology

Example: Metrics of Eyeriss Chip

ASIC Specs	Input
Process Technology	65nm LP TSMC (1.0V)
Total Core Area (mm^2)	12.25
Total On-Chip Memory (kB)	192
Number of Multipliers	168
Clock Frequency (MHz)	200
Core area (mm^2) / multiplier	0.073
On-Chip memory (kB) / multiplier	1.14
Measured or Simulated	Measured

Metric	Units	Input
Name of CNN Model	Text	AlexNet
Top-5 error classification on ImageNet	#	19.8
Supported Layers		All CONV
Bits per weight	#	16
Bits per input activation	#	16
Batch Size	#	4
Runtime	ms	115.3
Power	mW	278
Off-chip Access per Image Inference	MBytes	3.85
Number of Images Tested	#	100

Comprehensive Coverage

- **All metrics** should be reported for fair evaluation of design tradeoffs
- Examples of what can happen if 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 bandwidth**, one could build a processor with only multipliers 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?

Evaluation Process

The evaluation process for whether a DNN system is a viable solution for a given application might go as follows:

1. **Accuracy** determines if it can perform the given task
2. **Latency and throughput** determine if it can run fast enough and in real-time
3. **Energy and power consumption** will primarily dictate the form factor of the device where the processing can operate
4. **Cost**, which is primarily dictated by the chip area, determines how much one would pay for this solution

Summary

- **Deep Learning is an important area of research**
  - Wide range of applications
- **Challenge is to balance the key metrics**
  - Accuracy, Energy, Throughput, Cost, etc.
- **Opportunities at various levels of hardware design**
  - Architecture, Joint Algorithm-Hardware, Mixed-Signal Circuits/Memories, Advanced Technologies
  - Important to consider interactions between levels to maximize impact

For updates on Eyerissv2, Eyexam, NetAdapt, etc.

Follow @eems_mit

or join EEMS news mailing list

References

Overview Paper

V. Sze, Y.-H. Chen, T-J. Yang, J. Emer, "*Efficient Processing of Deep Neural Networks: A Tutorial and Survey,*" **Proceedings of the IEEE**, December 2017

More info about **Eyeriss** and **Tutorial on DNN Architectures**

http://eyeriss.mit.edu

MIT Professional Education Course on

"Designing Efficient Deep Learning Systems"

July 23 – 24, 2018 on MIT Campus

http://professional-education.mit.edu/deeplearning

For updates Follow @eems_mit

http://mailman.mit.edu/mailman/listinfo/eems-news

Acknowledgements

Research conducted in the **MIT Energy-Efficient Multimedia Systems Group** would not be possible without the support of the following organizations:

AFOSR AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
NSF
ANALOG DEVICES
BROADCOM
Google
IBM
intel
NVIDIA
DARPA
SRC
3M
Qualcomm
SAMSUNG
Texas Instruments
tsmc