Hardware for Machine Learning: Challenges and Opportunities

Vivienne Sze, Yu-Hsin Chen, Joel Emer, Amr Suleiman, Zhengdong Zhang

Massachusetts Institute of Technology

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

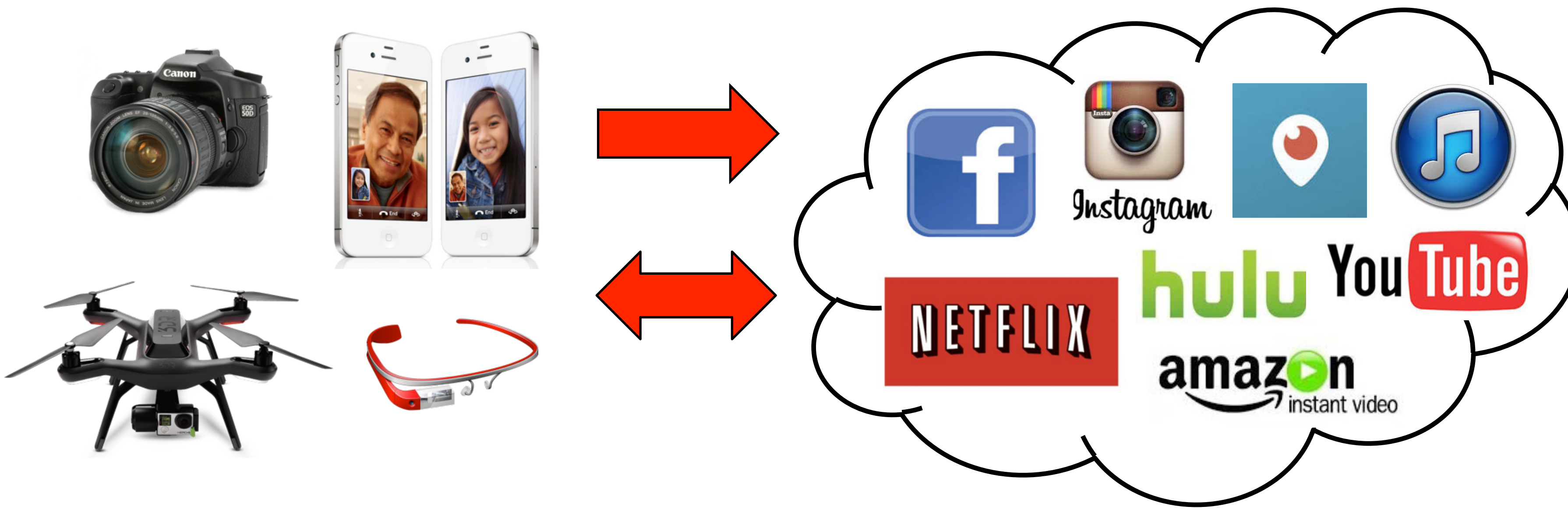
Follow @eems_mit

Video is the Biggest Big Data

Over 70% of today's Internet traffic is video

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

Over 500 million hours of video surveillance collected **every day**



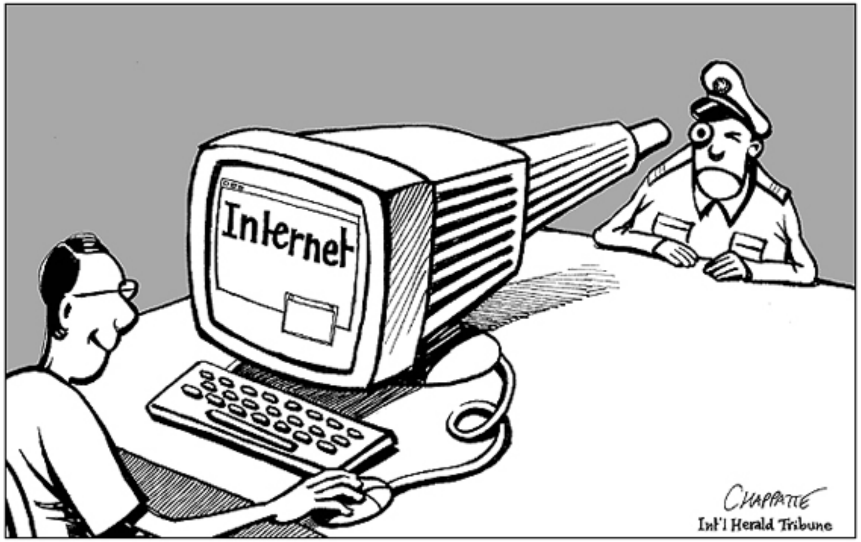
Energy limited due to battery capacity

Power limited due to heat dissipation

Need energy-efficient pixel processing!

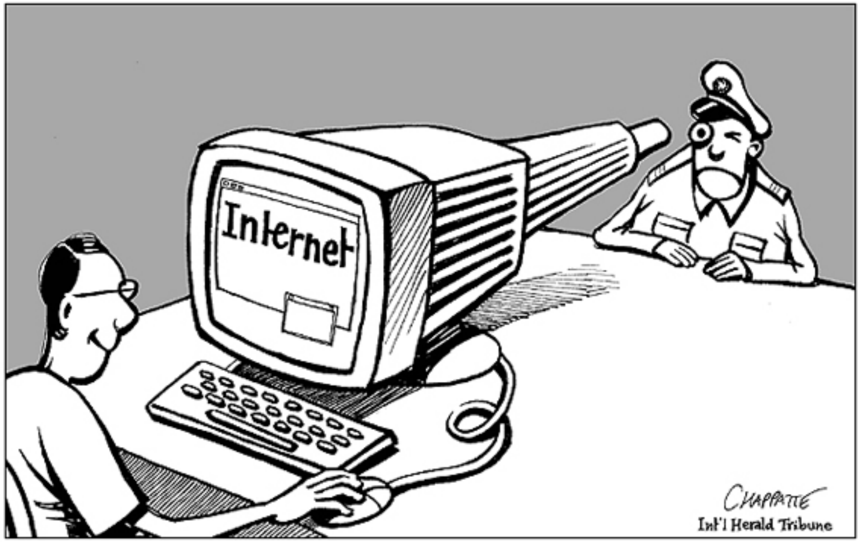
Processing at "Edge" instead of the "Cloud"

Privacy



Processing at "Edge" instead of the "Cloud"

Privacy



Latency

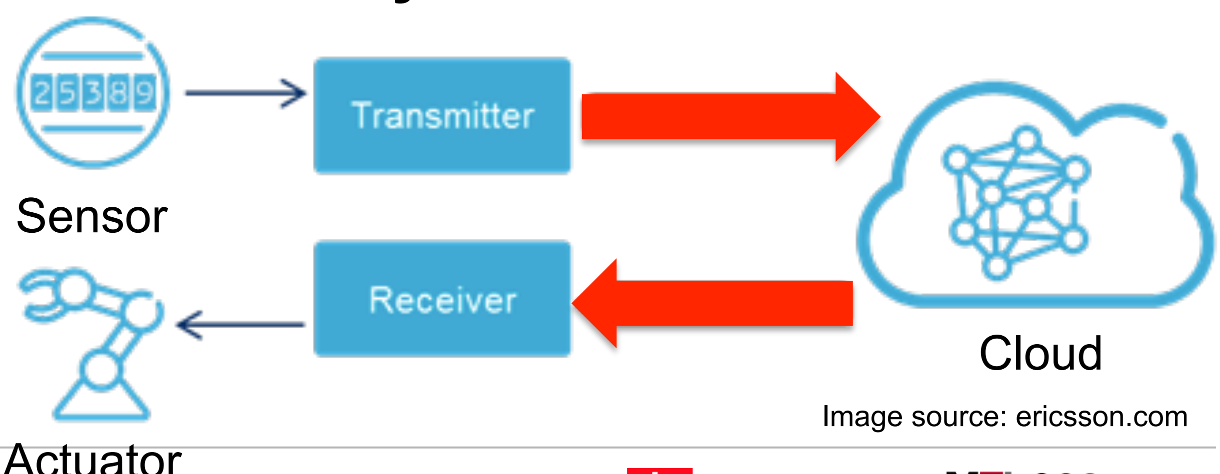
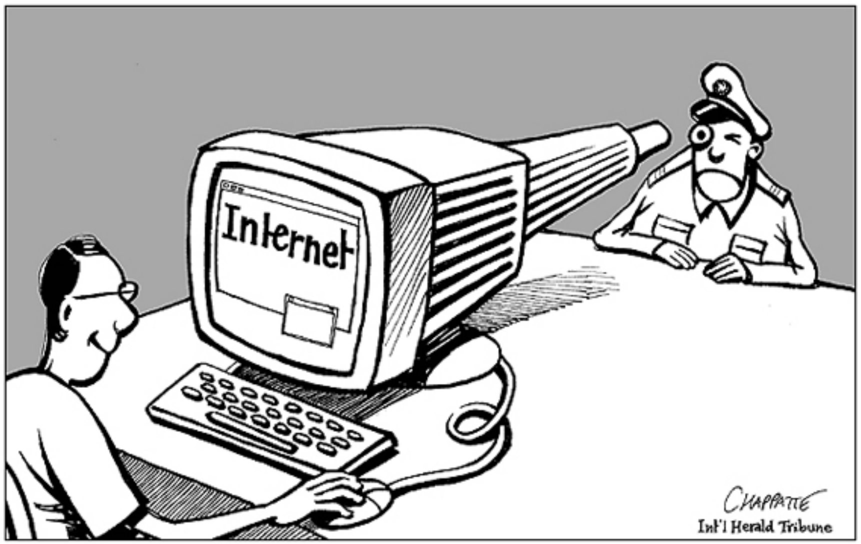


Image source: ericsson.com

Processing at "Edge" instead of the "Cloud"

Privacy



Communication

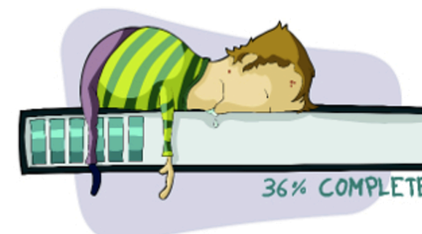


Image source: www.theregister.co.uk

Latency

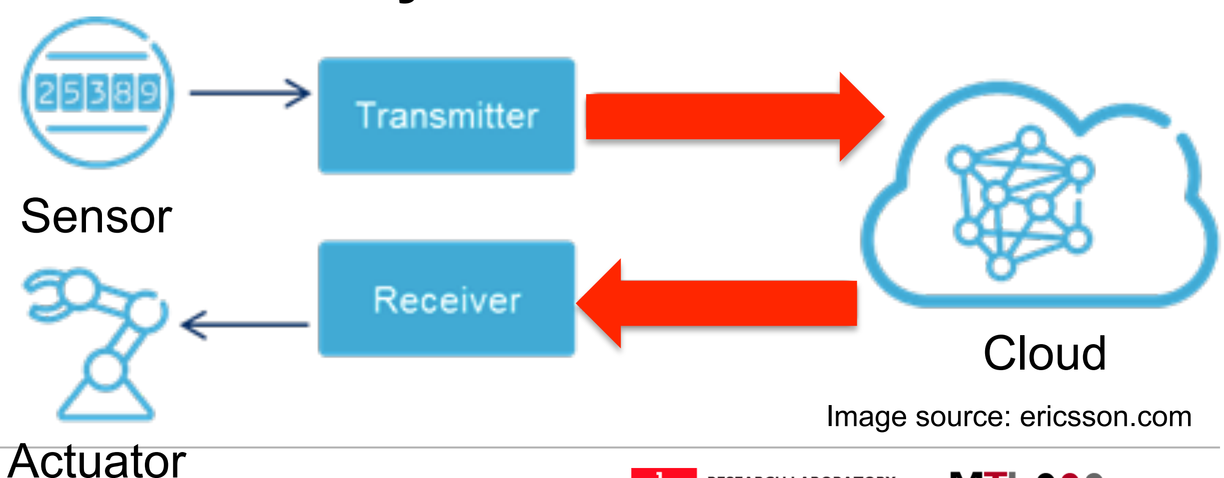
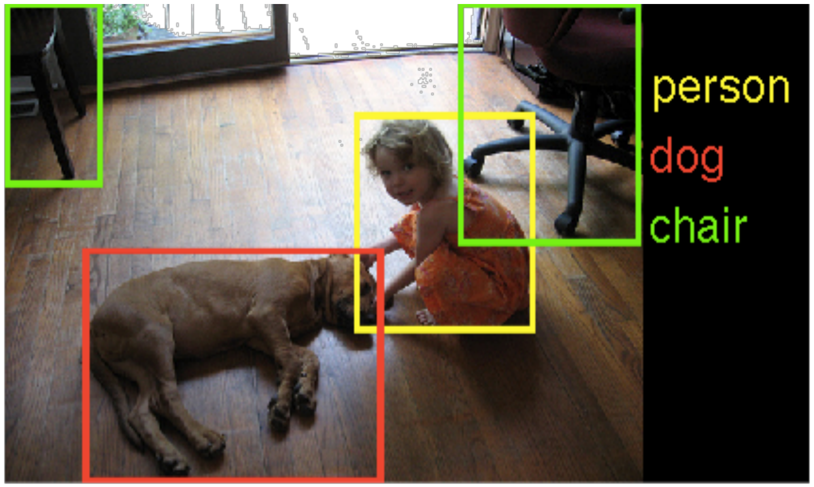


Image source: ericsson.com

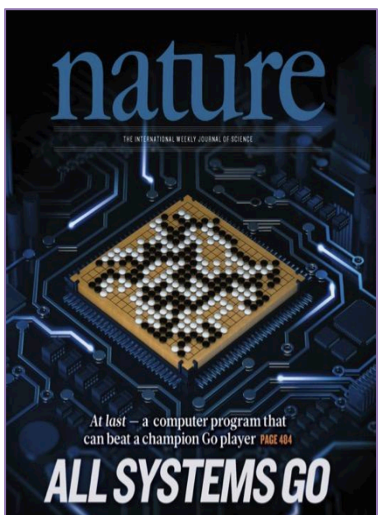
Example Applications of Machine Learning

Computer Vision

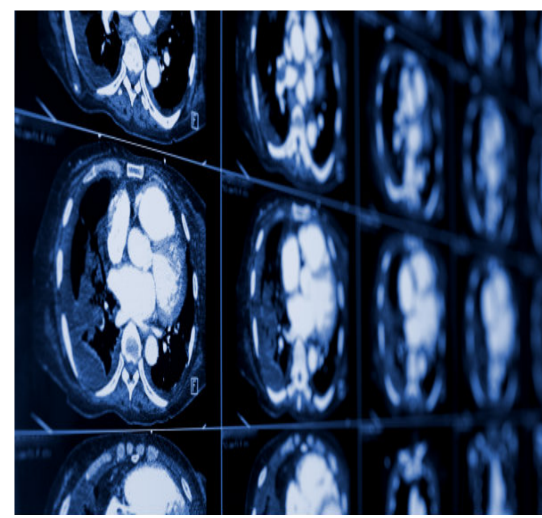


Speech Recognition

Game Play



Medical

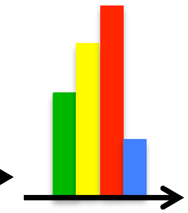


Machine Learning Pipeline (Inference)

Image

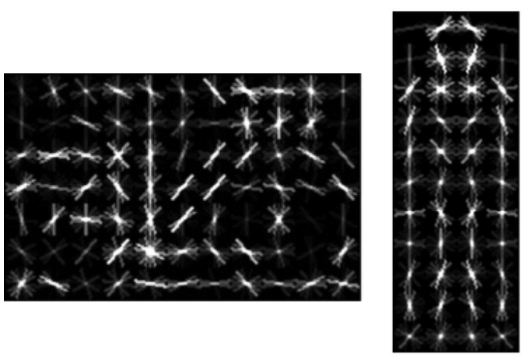
pixels → **Feature Extraction** → Features (**x**) → **Classification** ***(w^Tx)*** → Scores

Trained weights (**w**) → **Classification**



Scores per class
(select class based on max or threshold)

Handcrafted Features (e.g. HOG)



Learned Features (e.g. DNN)

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

Main Computation: Dot Product of Features (x) and Weights (w)

What is Deep Learning?

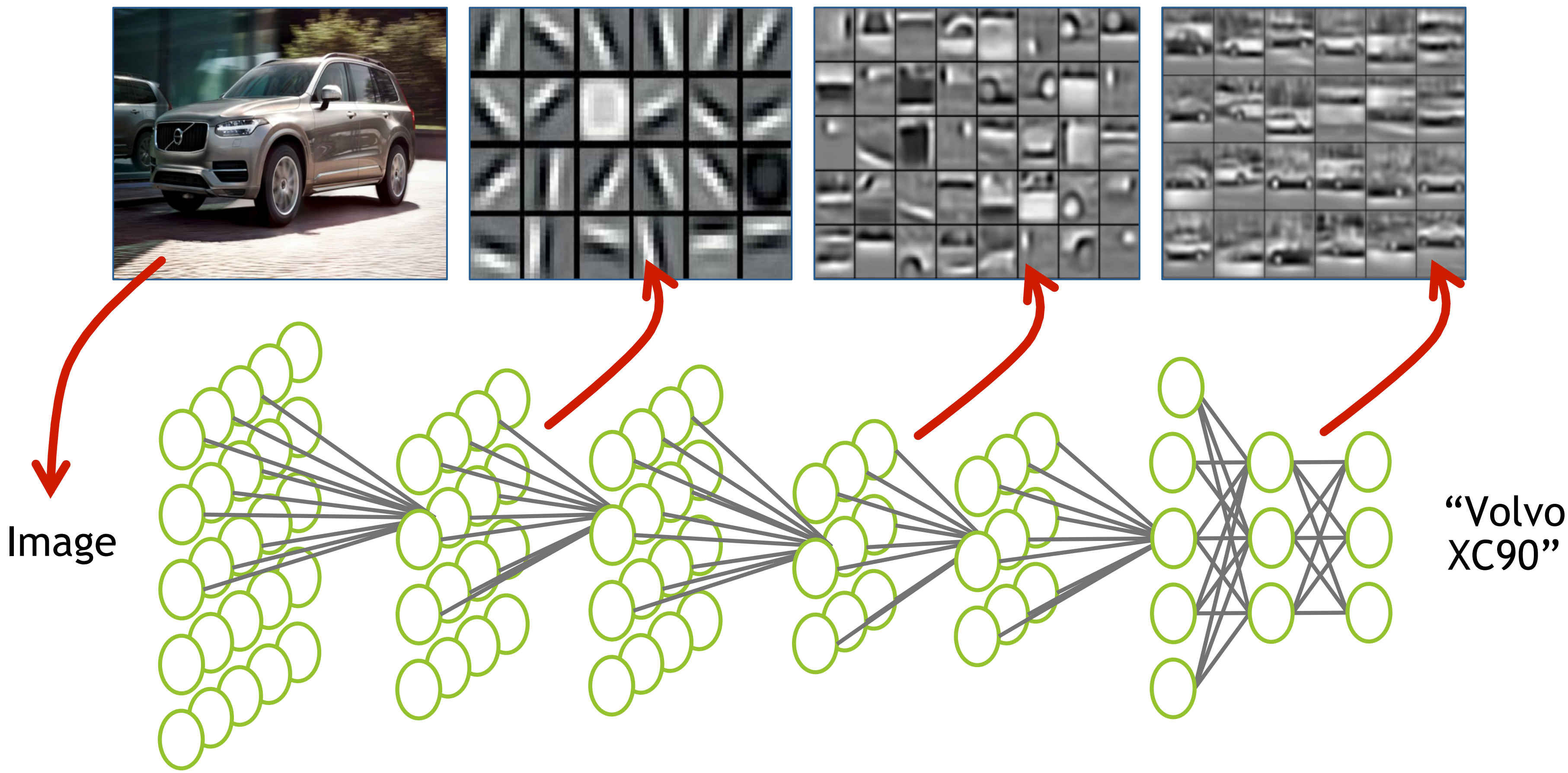


Image Source: [Lee et al., Comm. ACM 2011]

Weighted Sums

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

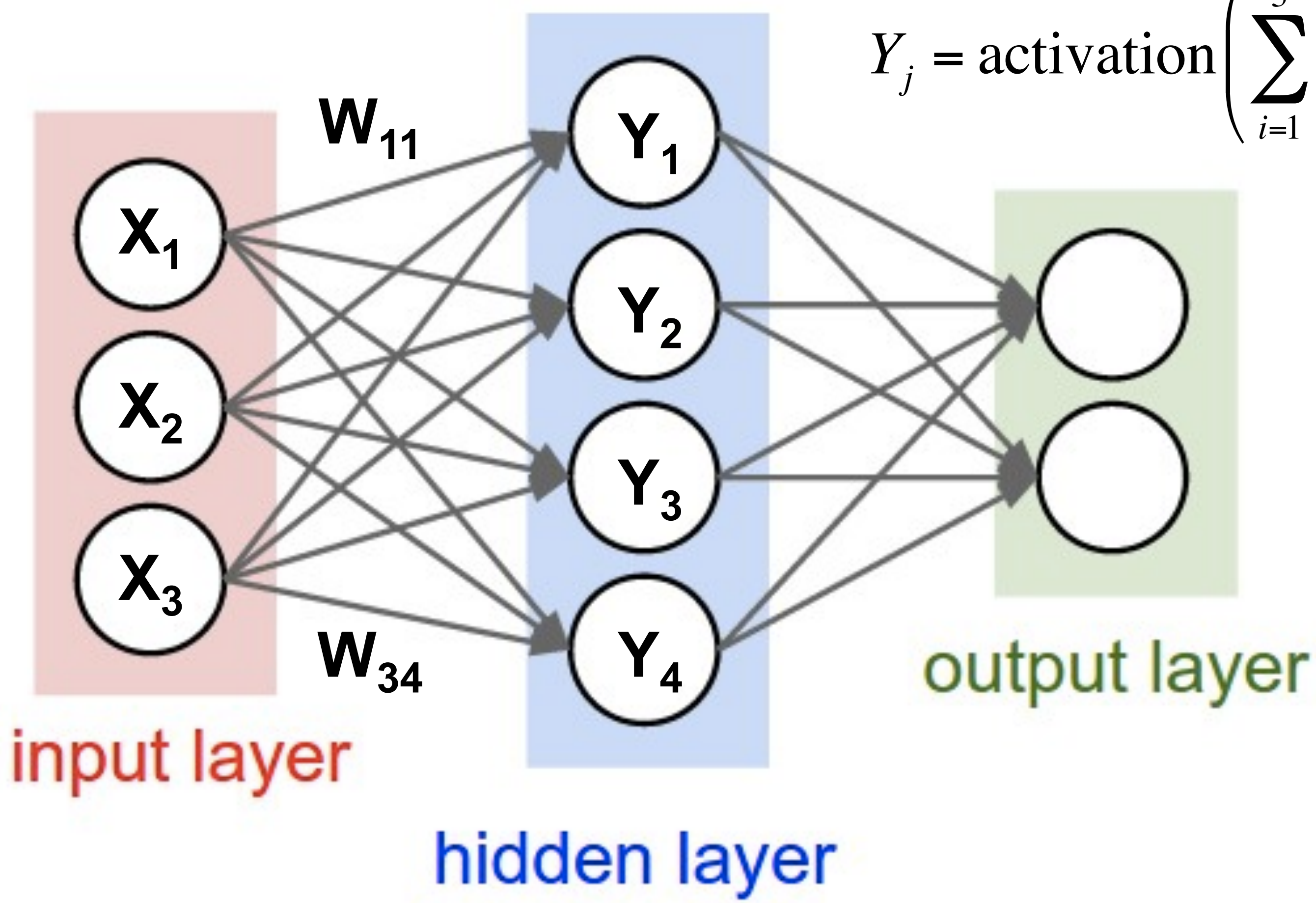


Image Source: Stanford

Why is Deep Learning Hot Now?

Big Data Availability

GPU Acceleration

New ML Techniques

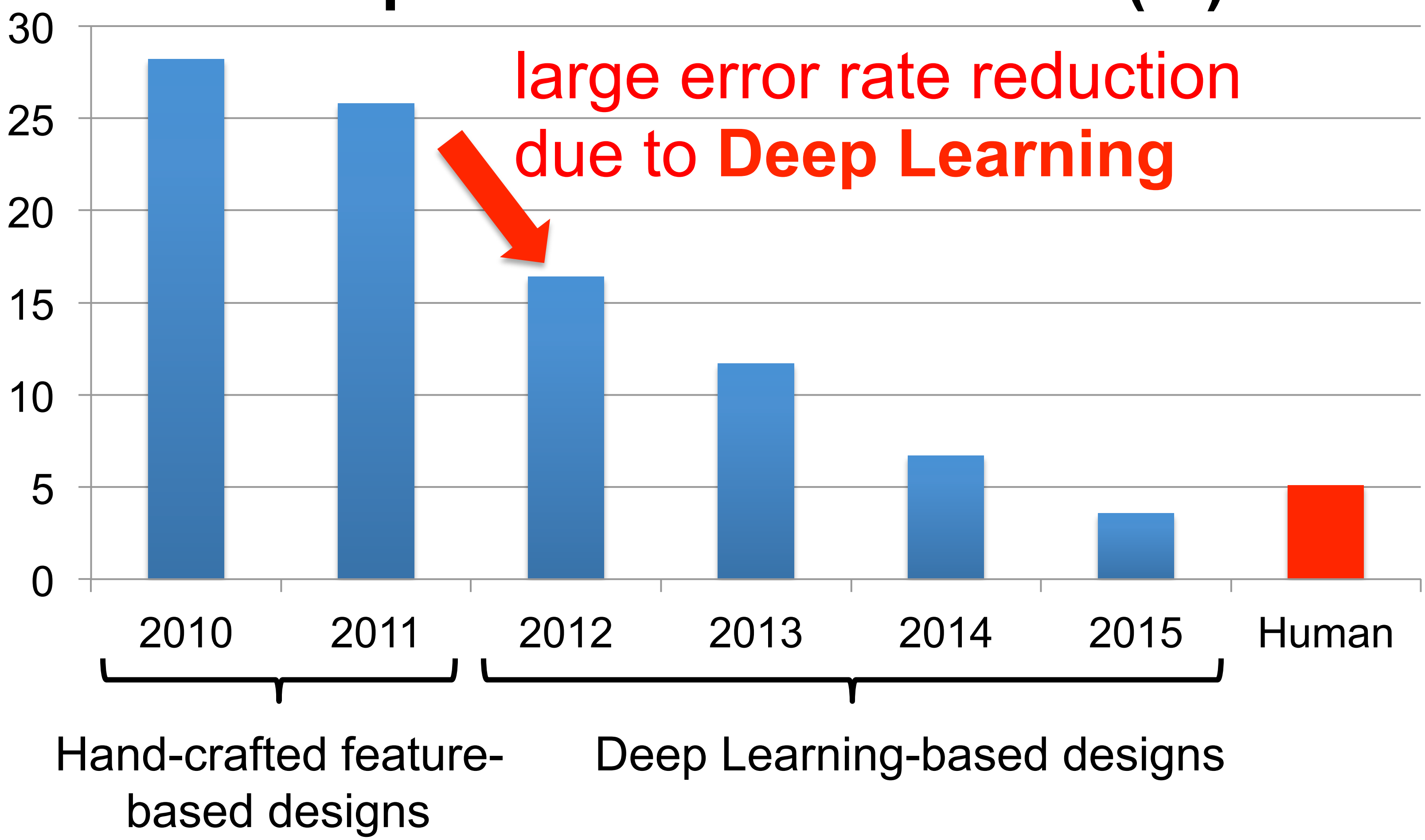
facebook — **350M** images uploaded per day

Walmart — **2.5 Petabytes** of customer data hourly

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

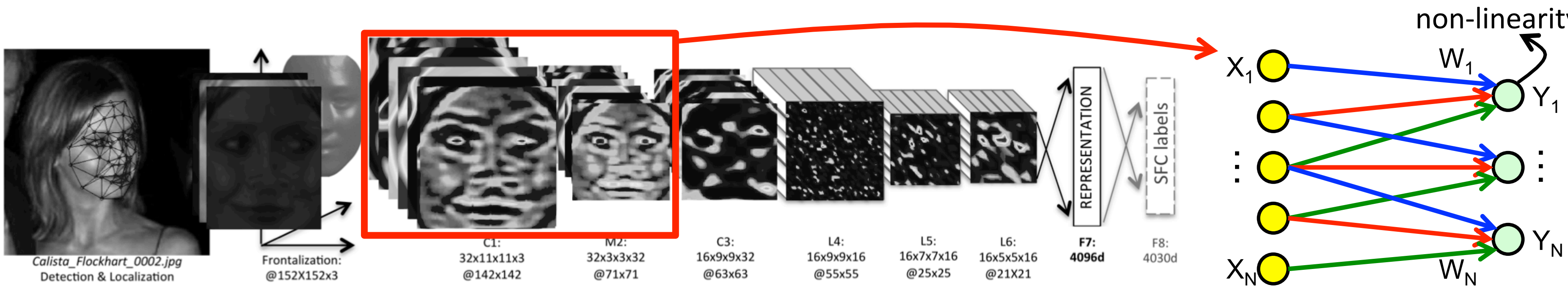
ImageNet: Image Classification Task

Top 5 Classification Error (%)



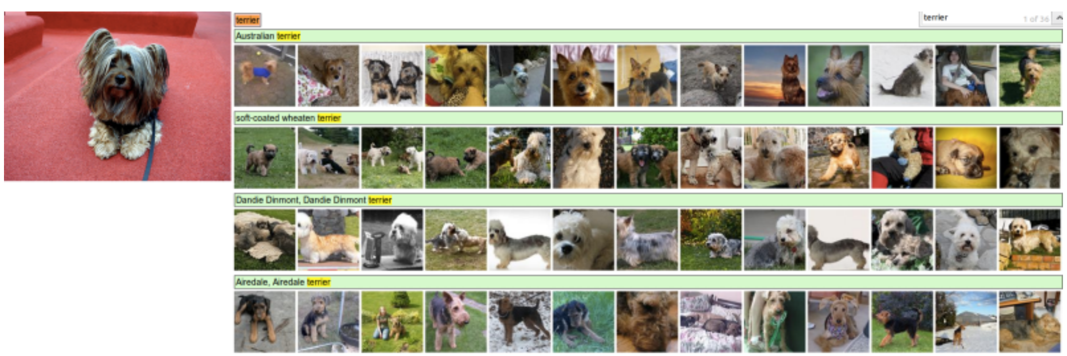
Human or *Superhuman* Accuracy Level

- Face recognition
  - Deep learning accuracy (97.25%) vs. Human accuracy (97.53%)



[Yaniv et al., CVPR 2014]

- Fine grained category recognition (e.g. dogs, monkeys, snakes, birds)
  - Deep learning errors: 7 vs. Human errors: 28



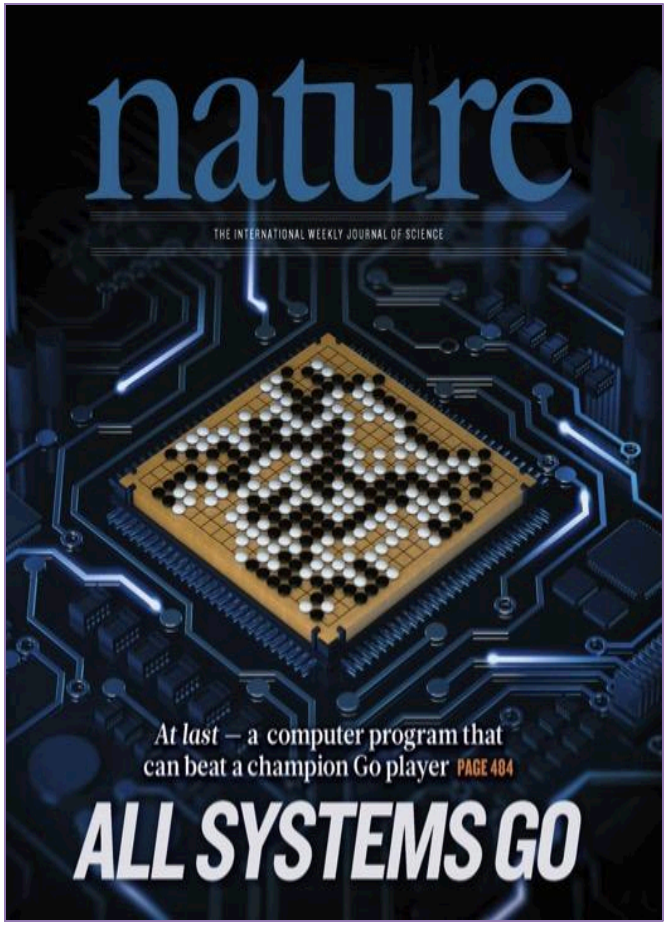
120 species of dogs

[O. Russakovsky et al., IJCV 2015]

Deep Learning on Games

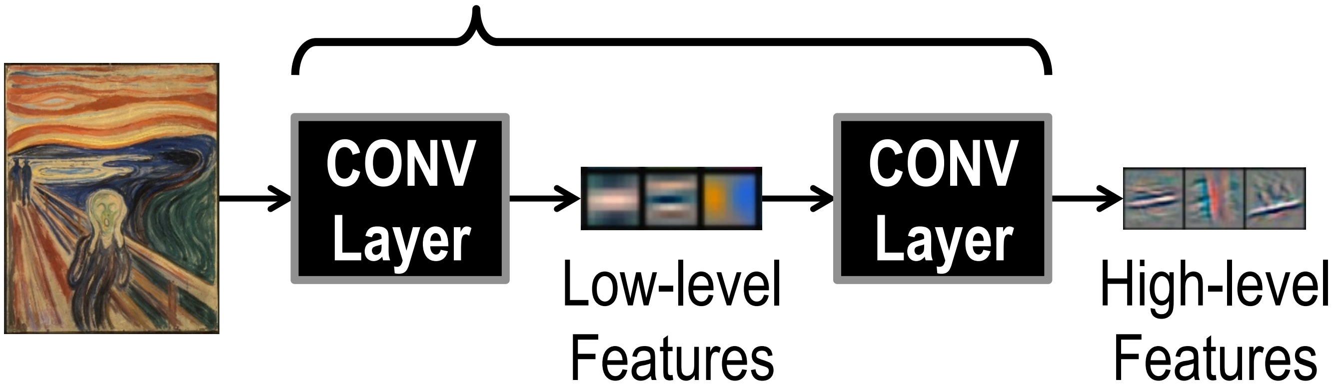
Google DeepMind AlphaGo

Go is exponentially more complex than chess (10^{170} legal positions)

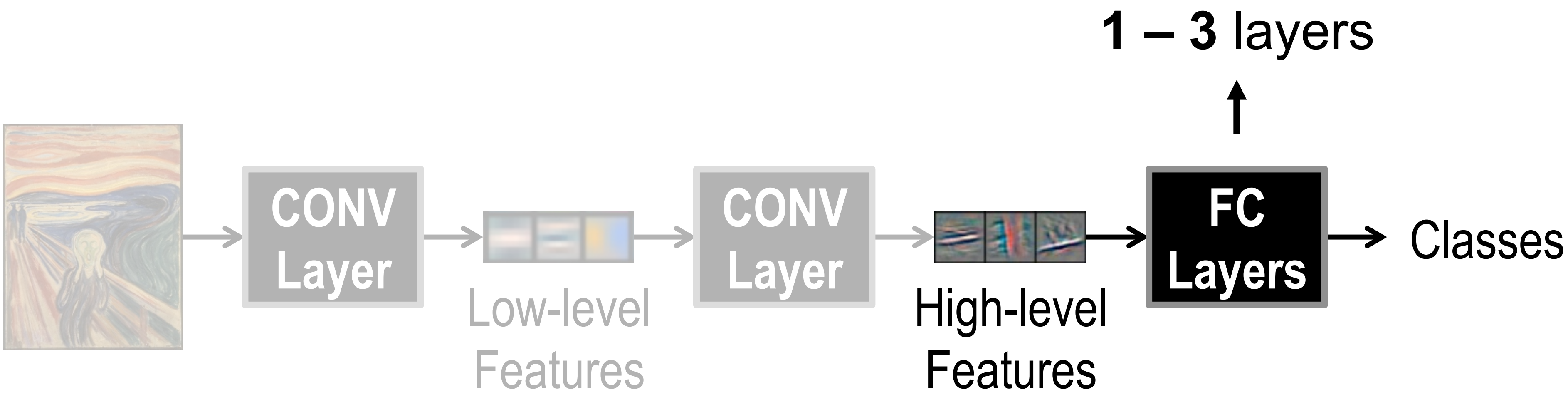


Deep Convolutional Neural Networks

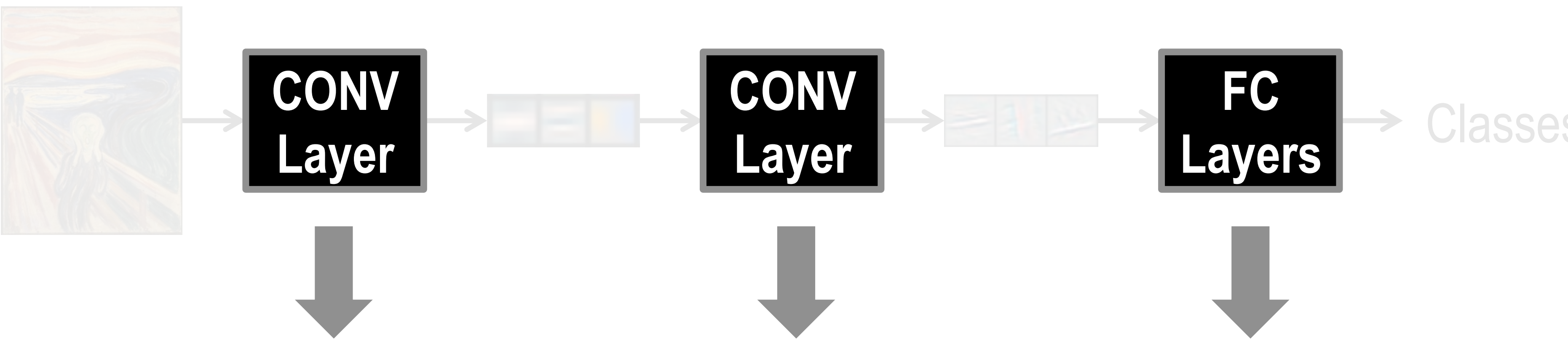
Modern ***deep*** CNN: up to **1000** CONV layers

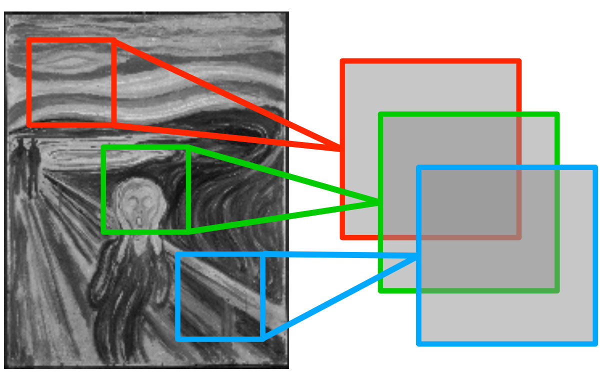


Deep Convolutional Neural Networks



Deep Convolutional Neural Networks



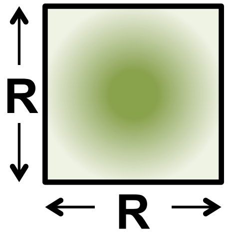


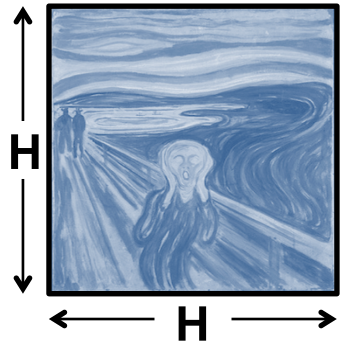
Convolutions account for more than 90% of overall computation, dominating **runtime** and **energy consumption**

High-Dimensional CNN Convolution

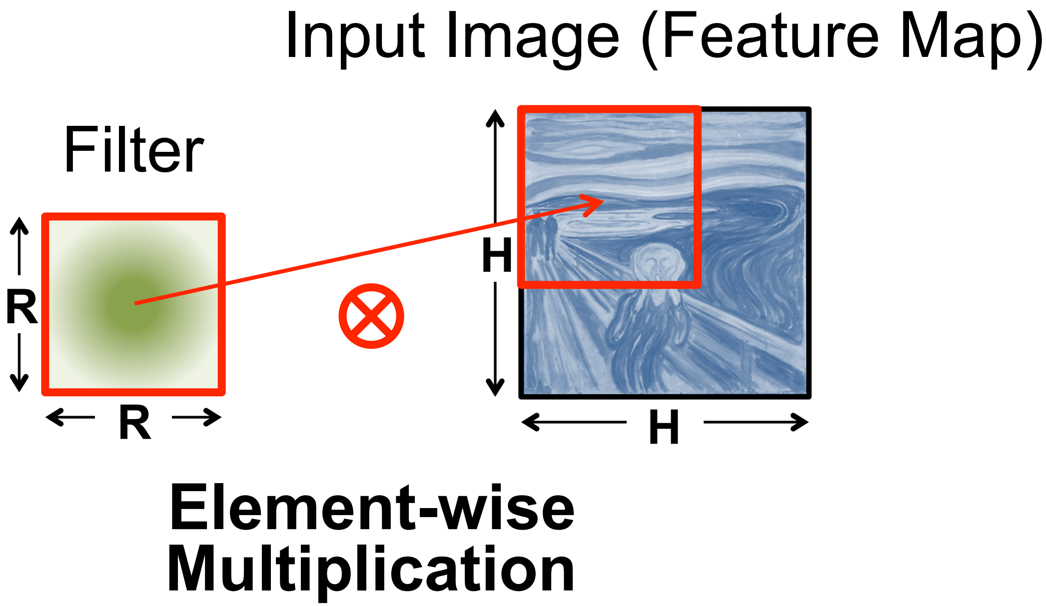
Input Image (Feature Map)

Filter

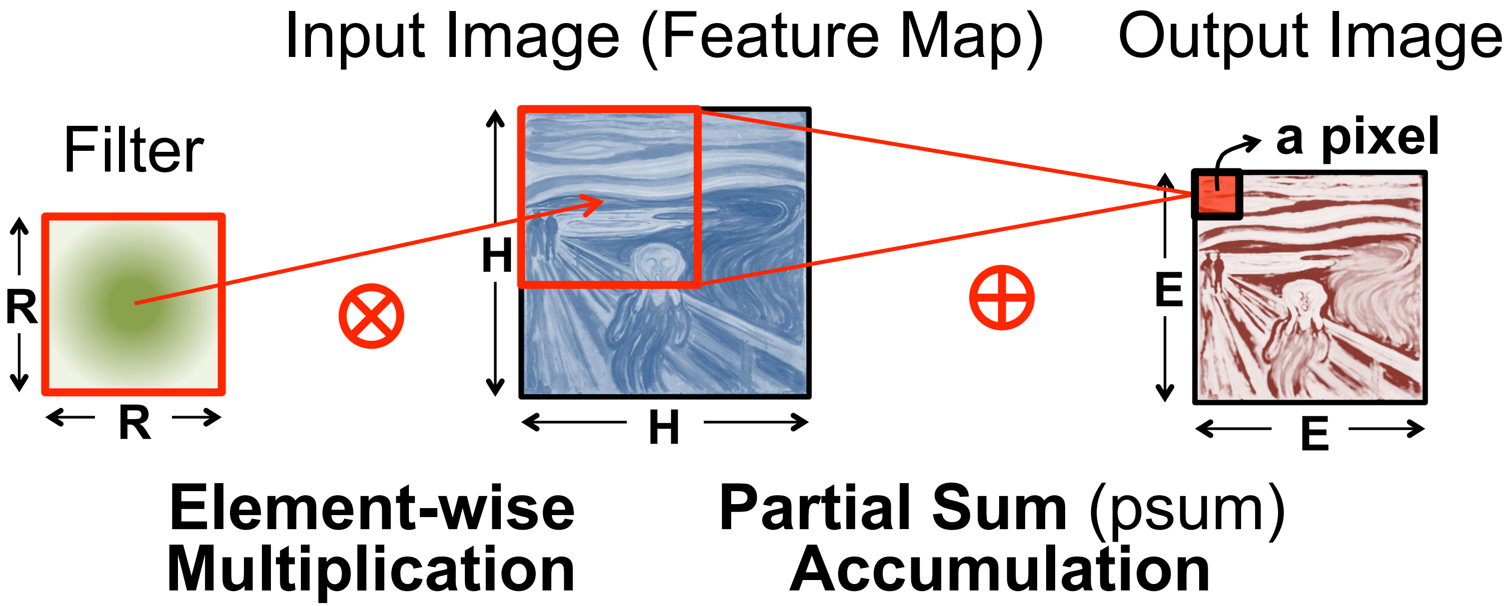




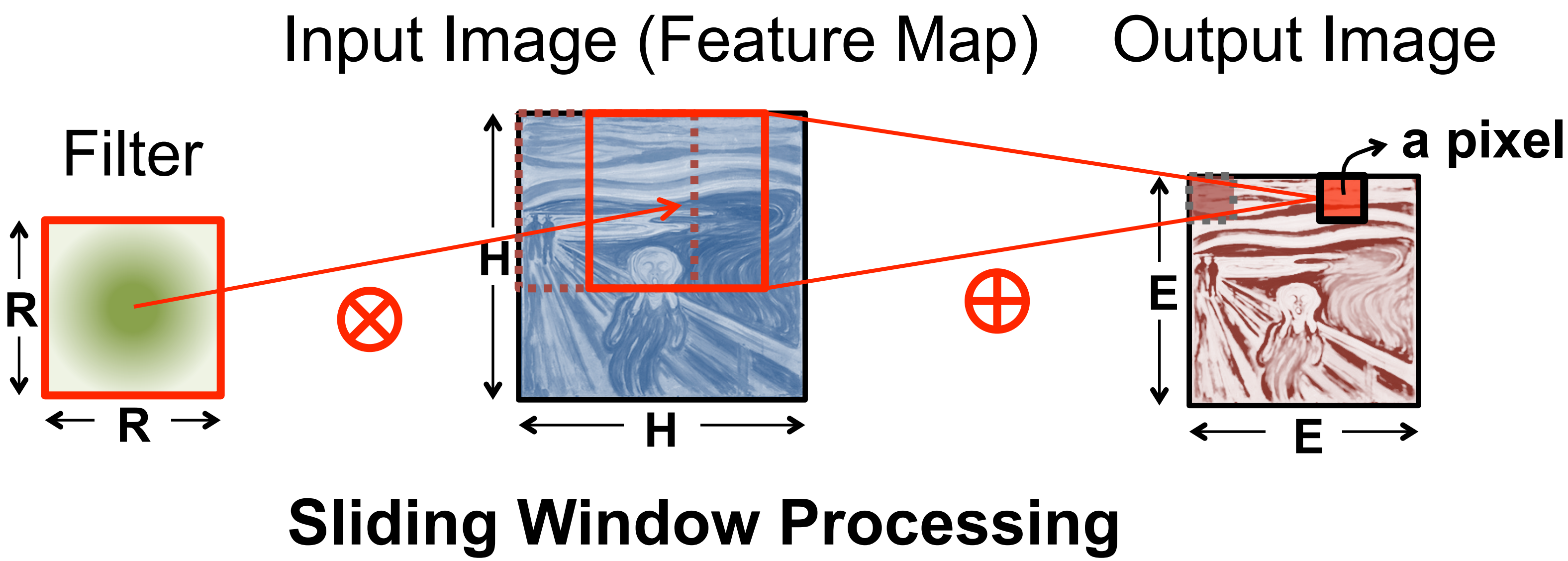
High-Dimensional CNN Convolution



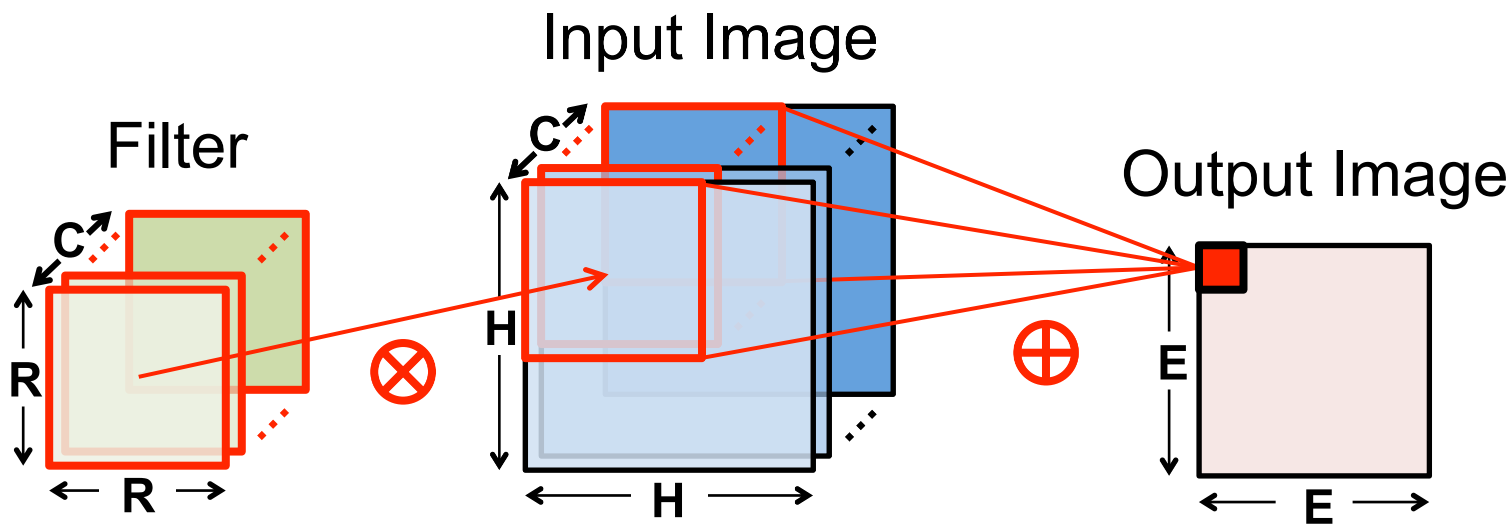
High-Dimensional CNN Convolution



High-Dimensional CNN Convolution



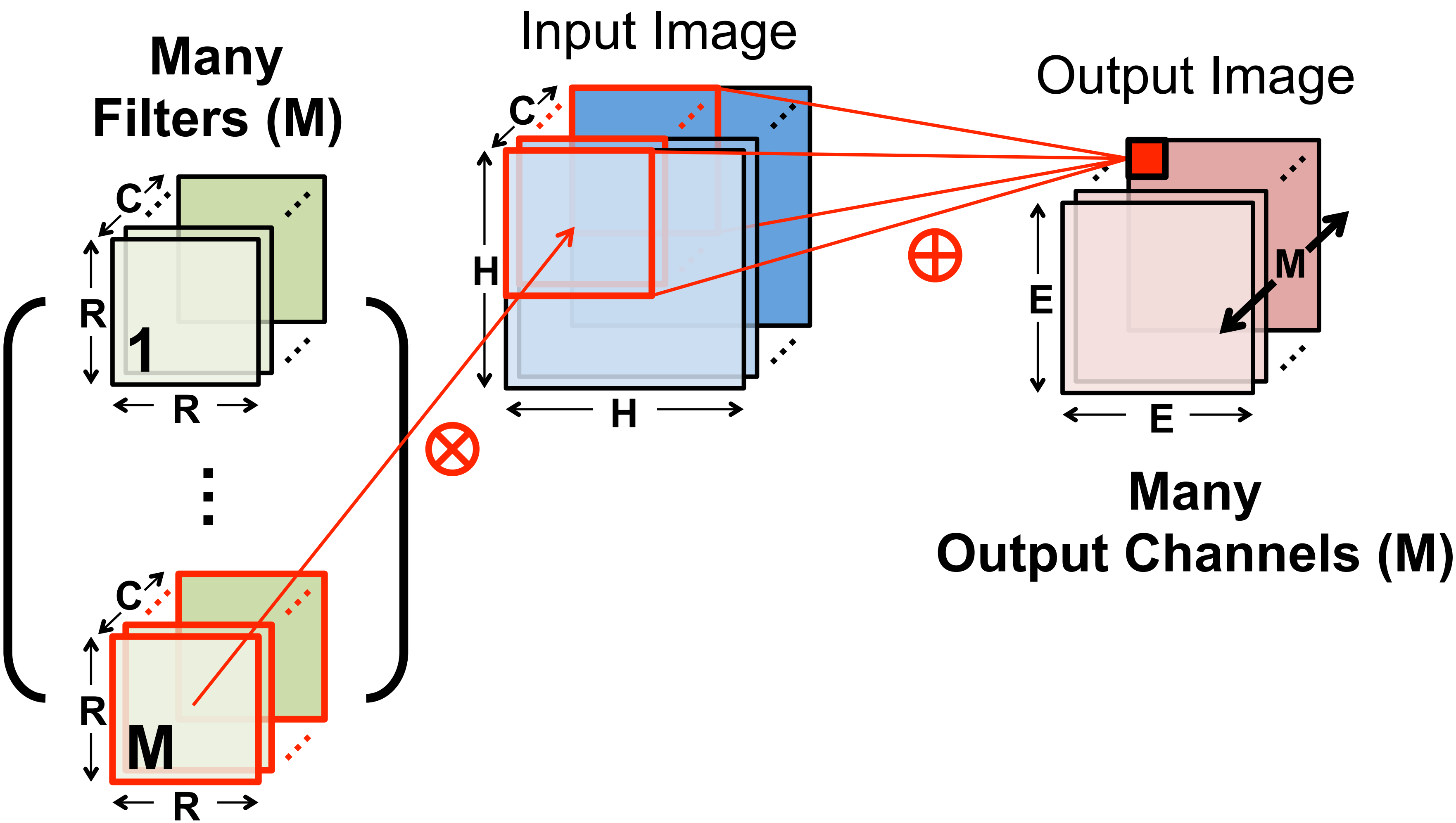
High-Dimensional CNN Convolution



Many Input Channels (C)

AlexNet: 3 – 192 Channels (C)

High-Dimensional CNN Convolution



AlexNet: 96 – 384 Filters (M)

High-Dimensional CNN Convolution

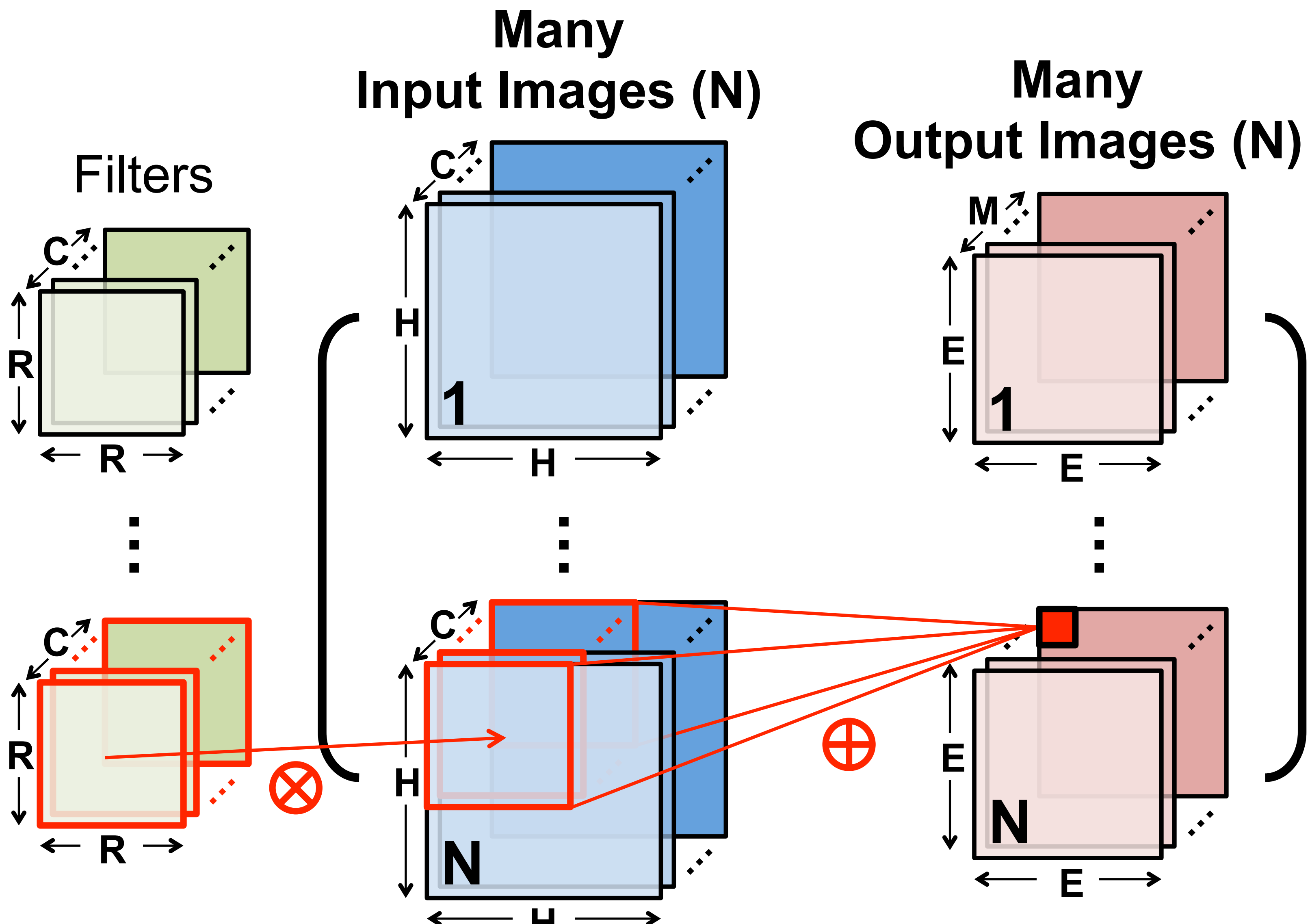
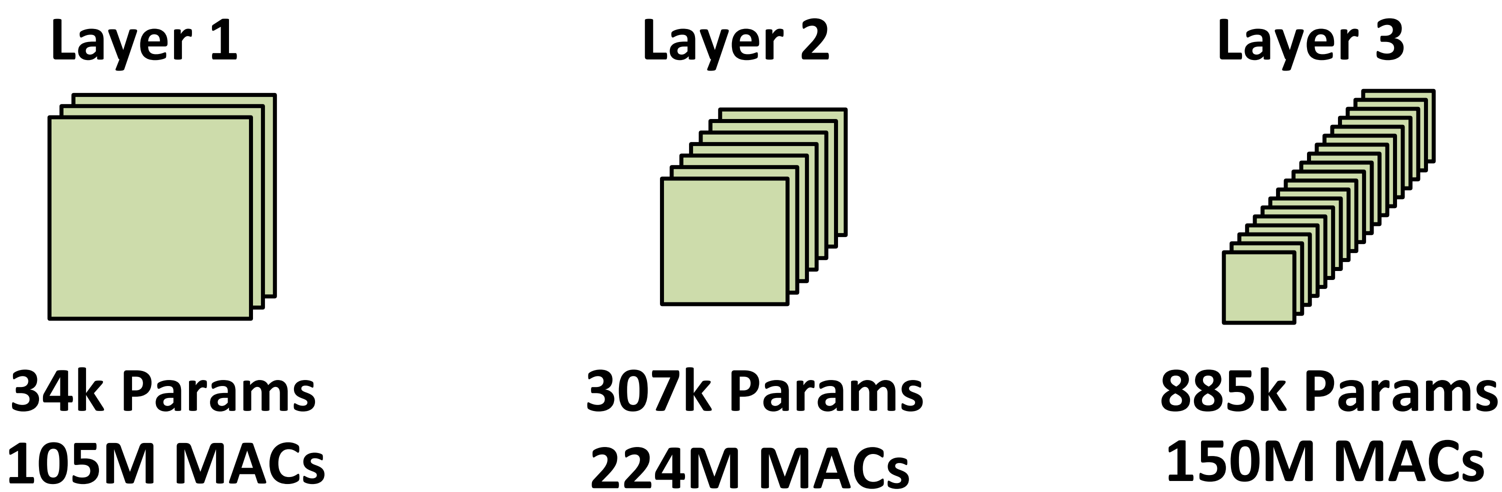


Image batch size: 1 – 256 (N)

Large Sizes with Varying Shapes

AlexNet[1] Convolutional Layer Configurations

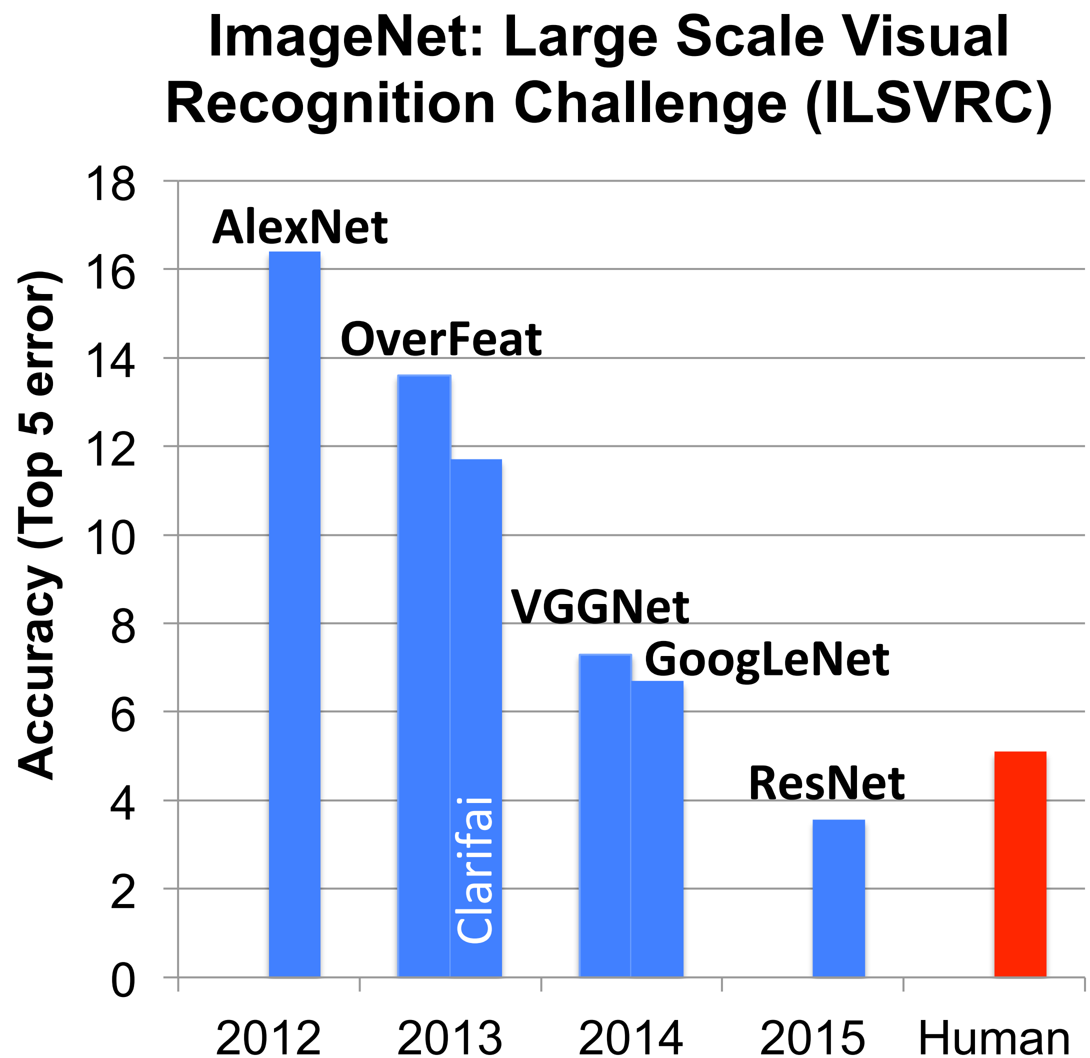
Layer	Filter Size (R)	# Filters (M)	# Channels (C)	Stride
1	11x11	96	3	4
2	5x5	256	48	1
3	3x3	384	256	1
4	3x3	384	192	1
5	3x3	256	192	1



1. [Krizhevsky, NIPS 2012]

Popular DNNs

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



[O. Russakovsky et al., IJCV 2015]

Summary of Popular DNNs

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**

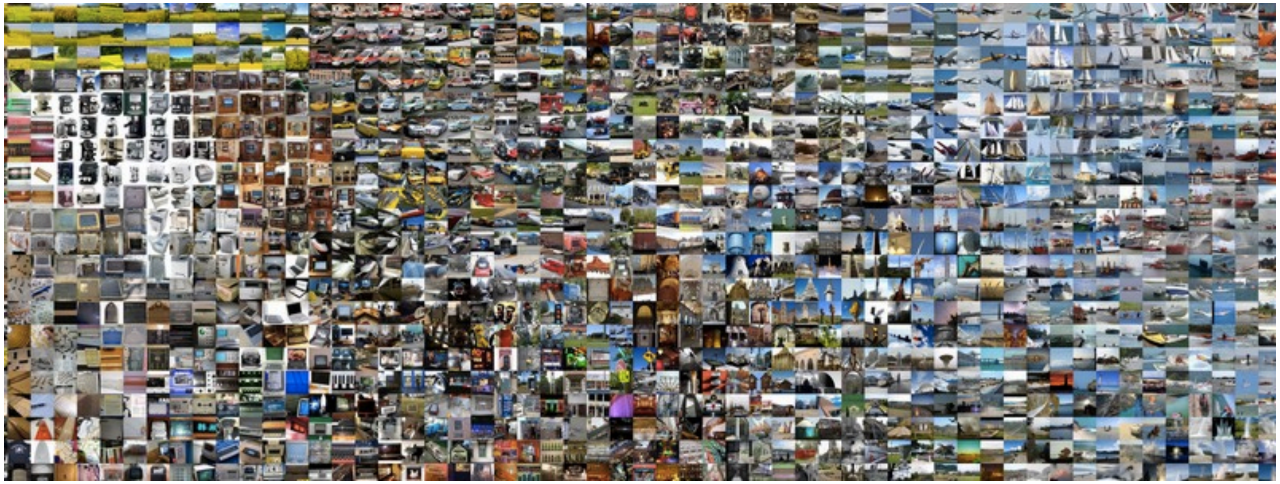
CONV Layers increasingly important!

Complexity versus Difficulty of Task

- **Evaluate hardware using the appropriate DNN model and dataset**
  - **Difficult tasks typically require larger models**
  - **Different datasets for different tasks**

MNIST

ImageNet



Training vs. Inference

Training (determine weights)

Inference (use weights)

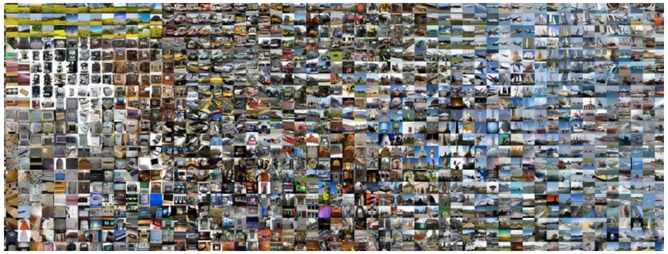
Weights

Challenges

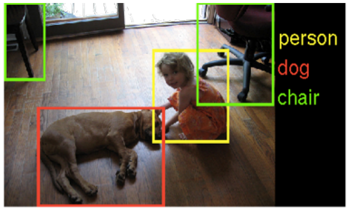
Key Metrics

- **Accuracy**
  - Measured on a publicly available dataset
  - Popular DNN Models
- **Programmability**
  - Support multiple applications
  - Different weights
- **Energy/Power**
  - Energy per operation
  - DRAM Bandwidth
- **Throughput/Latency**
  - GOPS, frame rate, delay
- **Cost**
  - Area (memory and logic size)

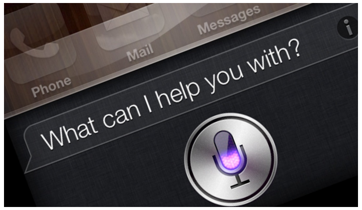
ImageNet



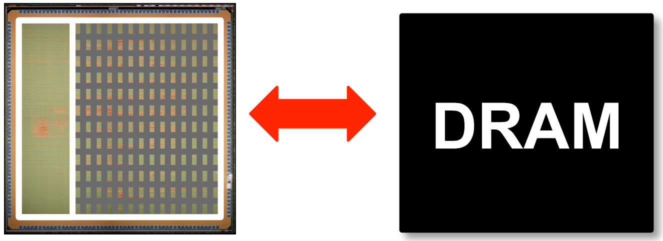
Computer Vision



Speech Recognition



Chip



Website to Summarize DNN Results

- http://eyeriss.mit.edu/benchmarking.html
- **Send results or feedback to: eyeriss@mit.edu**

ASIC Specs	Input
Process Technology	65nm LP TSMC (1.0V)
Core area (mm^2) / multiplier	0.073
On-Chip memory (kB) / multiplier	1.14
Measured or Simulated	Measured
If Simulated, Syn or PnR? Which corner?	n/a

Metric	Units	Input
Name of CNN	Text	AlexNet
# of Images Tested	#	100
Bits per operand	#	16
Batch Size	#	4
# of Non Zero MACs	#	409M
Runtime	ms	115.3
Power	mW	278
Energy/non-zero MACs	**pJ/MAC**	**21.7**
DRAM access/non-zero MACs	**operands /MAC**	**0.005**

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

Filter, Input Fmap, Output Fmap

Convolution:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Toeplitz Matrix (w/ redundant data)

Matrix Mult:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 4 & 5 \\ 2 & 3 & 5 & 6 \\ 4 & 5 & 7 & 8 \\ 5 & 6 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

Data is repeated

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

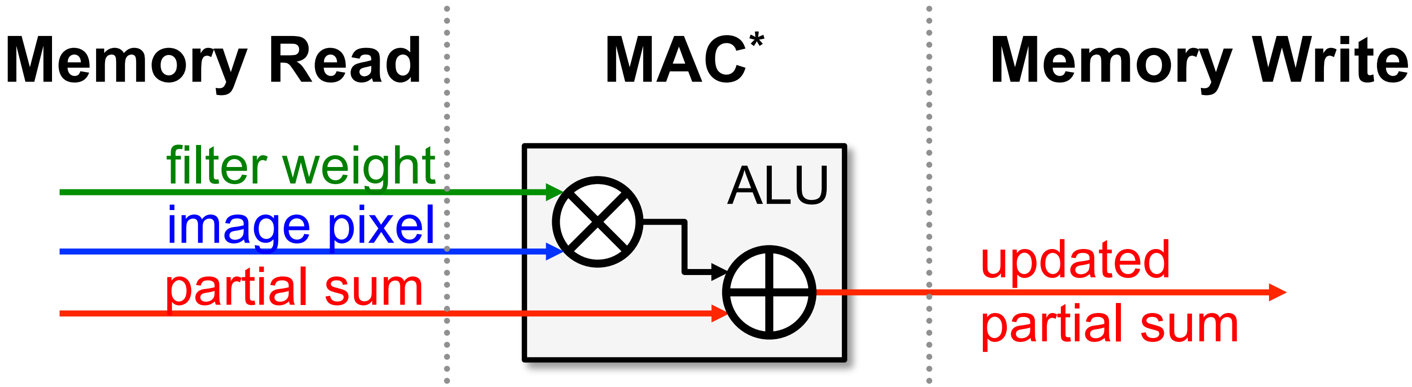
Accelerators

Properties We Can Leverage

- Operations exhibit **high parallelism**
  → **high throughput** possible

Properties We Can Leverage

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

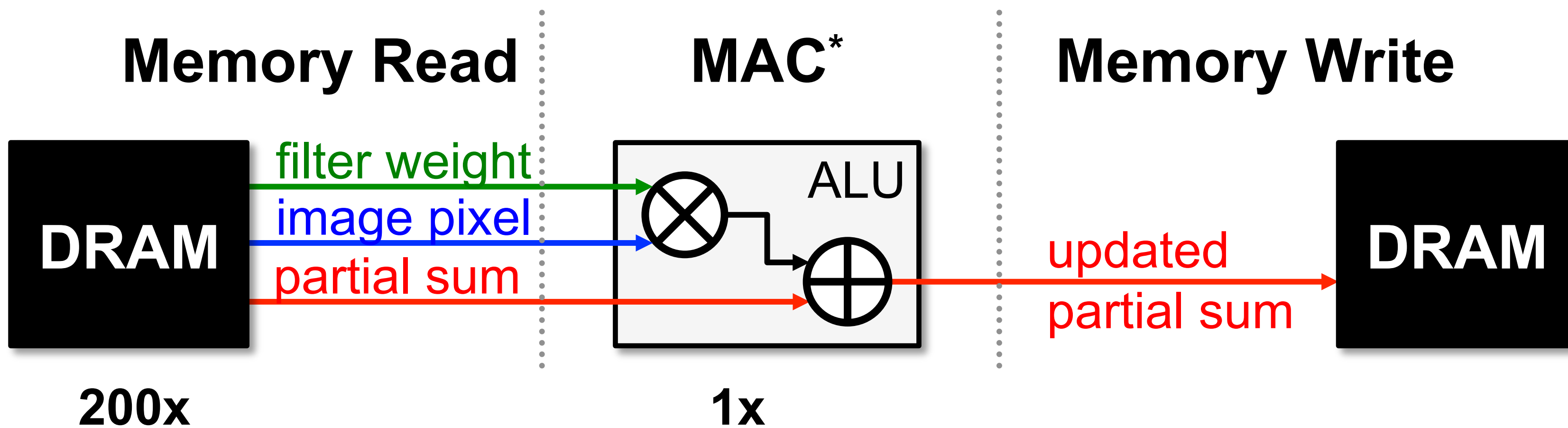


* multiply-and-accumulate

Properties We Can Leverage

- Operations exhibit **high parallelism**
  → **high throughput** possible

- Memory Access is the Bottleneck



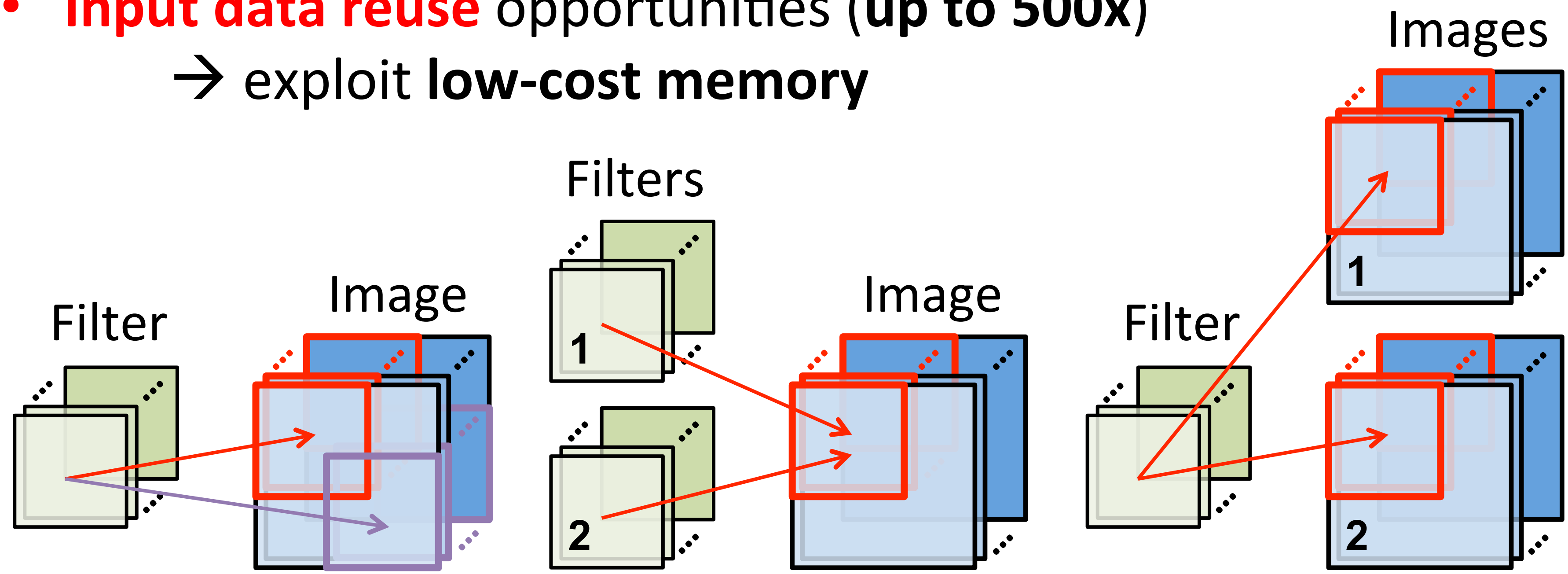
Worst Case: all memory R/W are **DRAM** accesses

- 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**



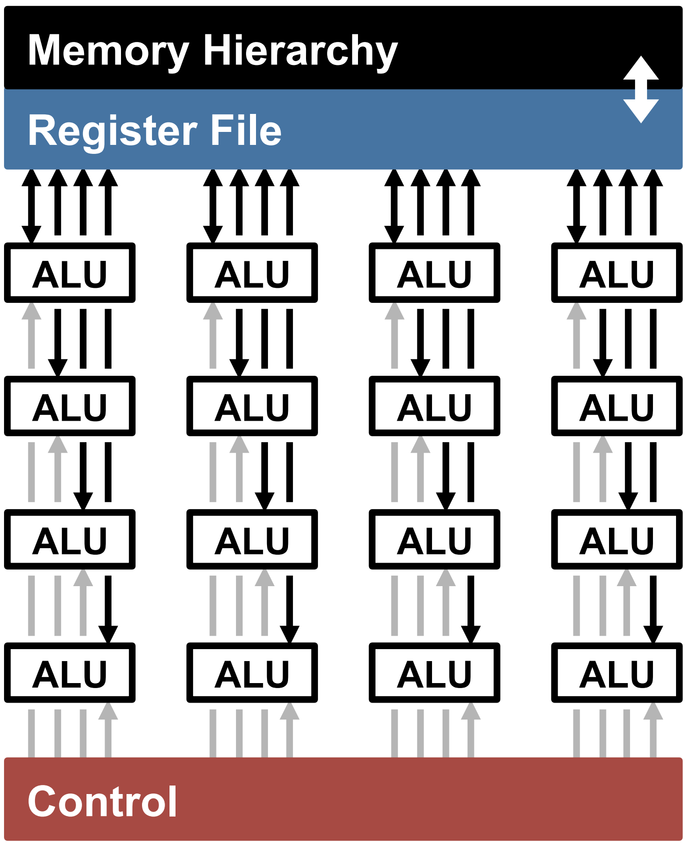
Convolutional Reuse
(pixels, weights)

Image Reuse
(pixels)

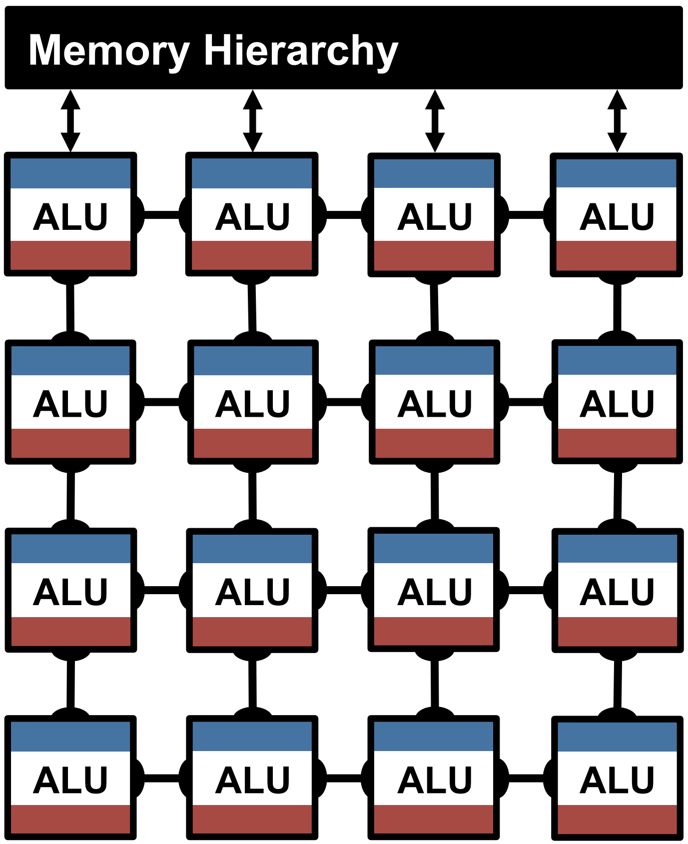
Filter Reuse
(weights)

Highly-Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)



Spatial Architecture (Dataflow Processing)



Advantages of Spatial Architecture

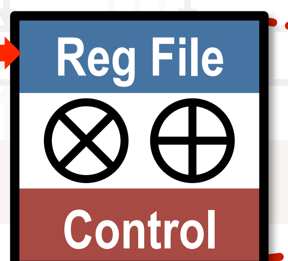
Temporal Architecture
(SIMD/SIMT)

Efficient Data Reuse
Distributed local storage (RF)

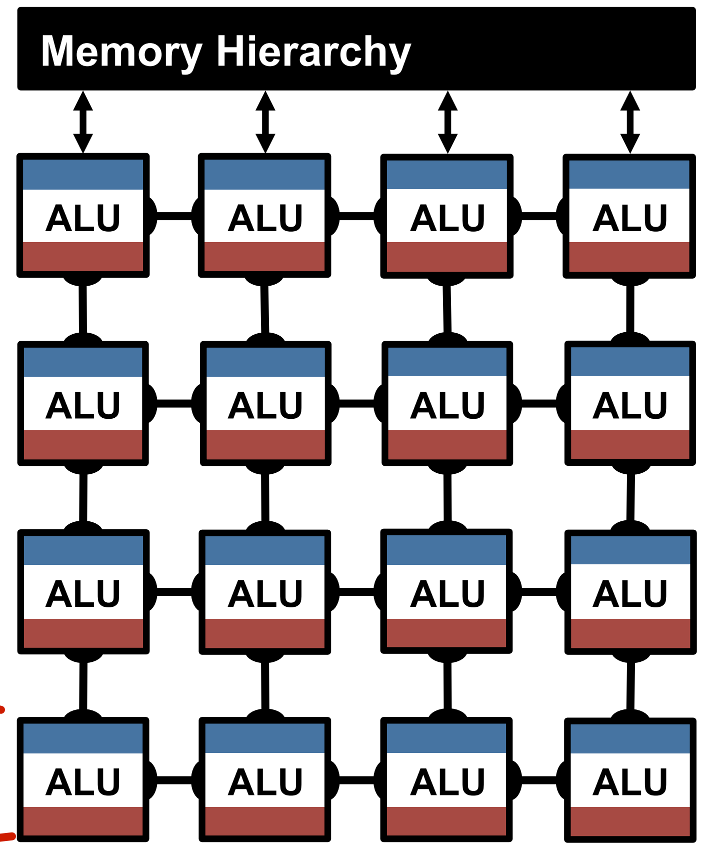
Inter-PE Communication
Sharing among regions of PEs

Processing Element (PE)

0.5 – 1.0 kB

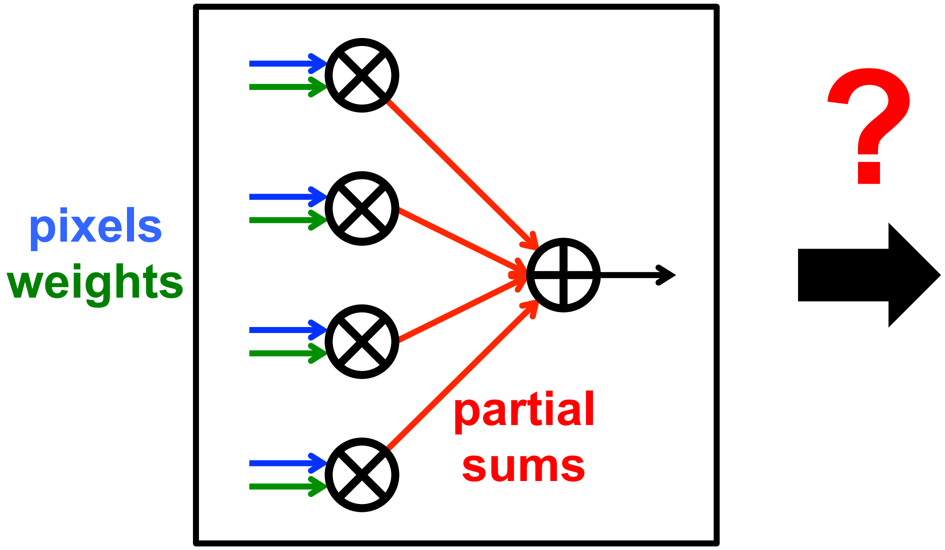


Spatial Architecture (Dataflow Processing)

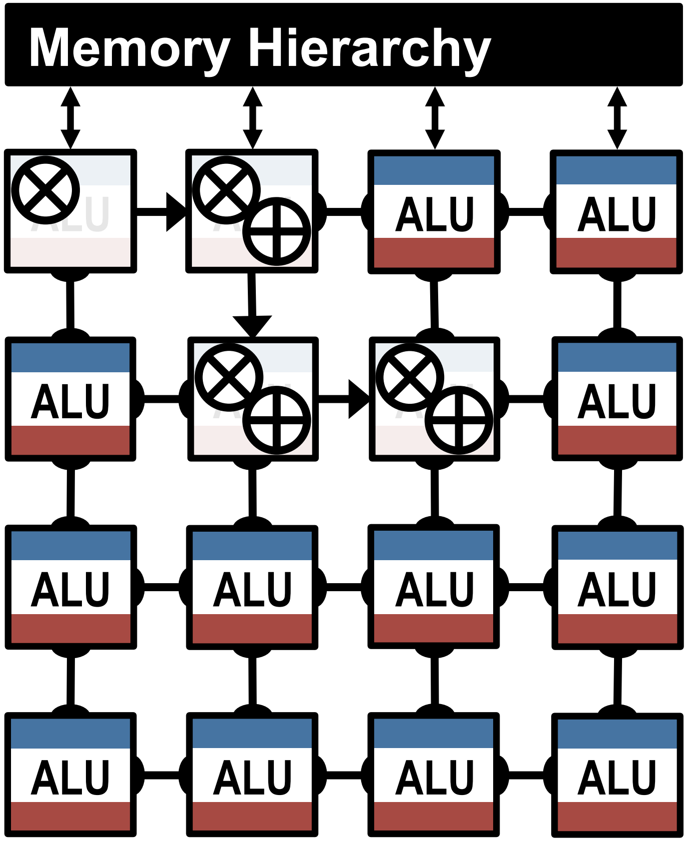


How to Map the Dataflow?

CNN Convolution



Spatial Architecture
(Dataflow Processing)



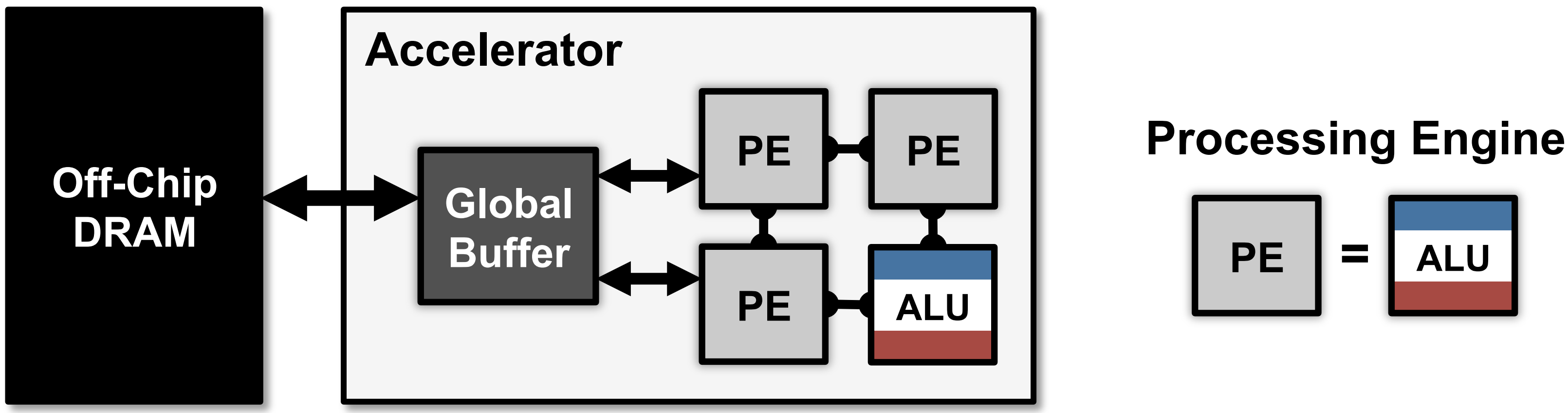
Goal: Increase reuse of input data (**weights** and **pixels**) and local **partial sums** accumulation

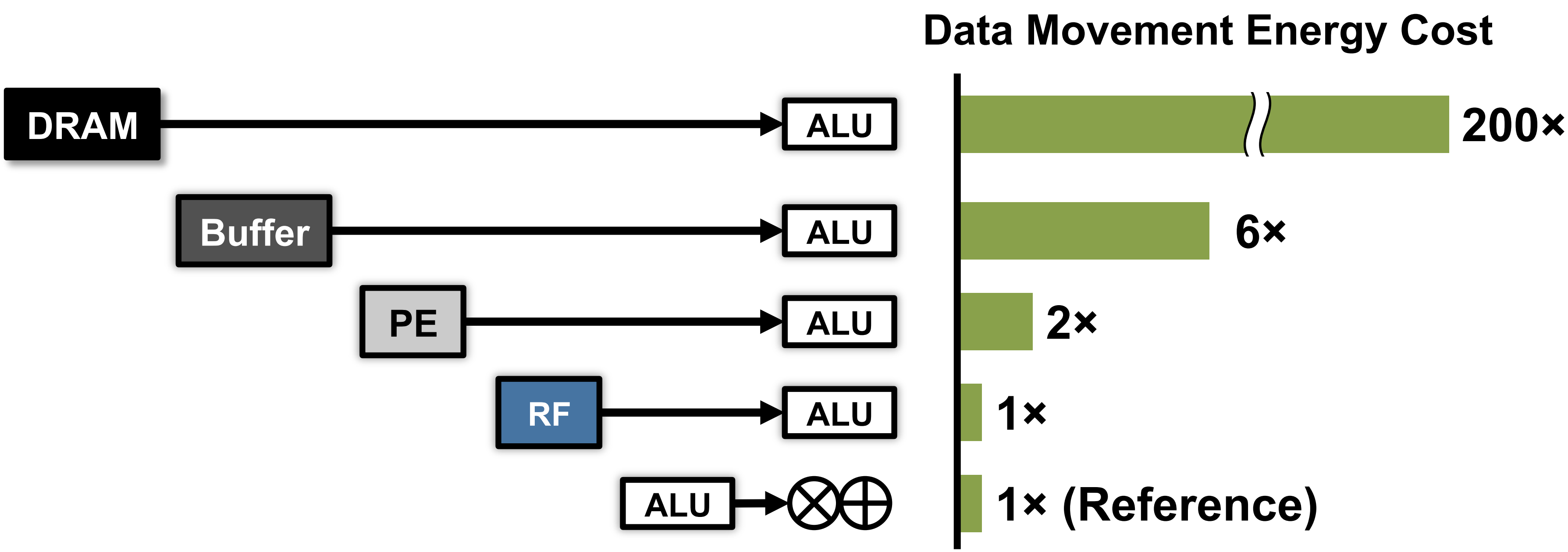
Energy-Efficient Dataflow

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

Maximize data reuse and accumulation at RF

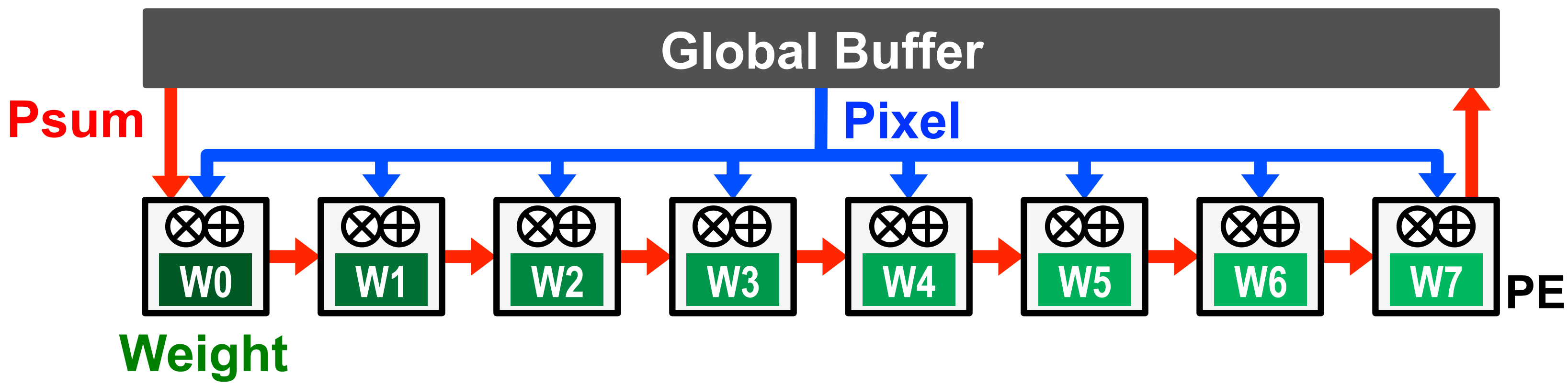
Data Movement is Expensive





Maximize data reuse at lower levels of hierarchy

Weight Stationary (WS)

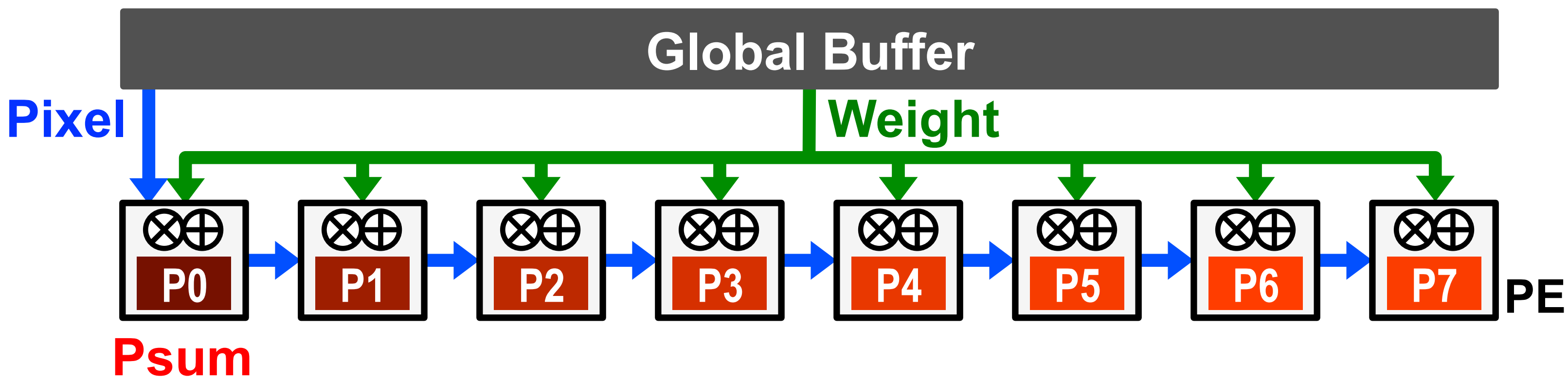


- **Minimize weight** read energy consumption
  - maximize convolutional and filter reuse of weights

- **Examples:**

  [**Chakradhar**, *ISCA* 2010] [**nn-X (NeuFlow)**, *CVPRW* 2014]
  [**Park**, *ISSCC* 2015] [**Origami**, *GLSVLSI* 2015]

Output Stationary (OS)



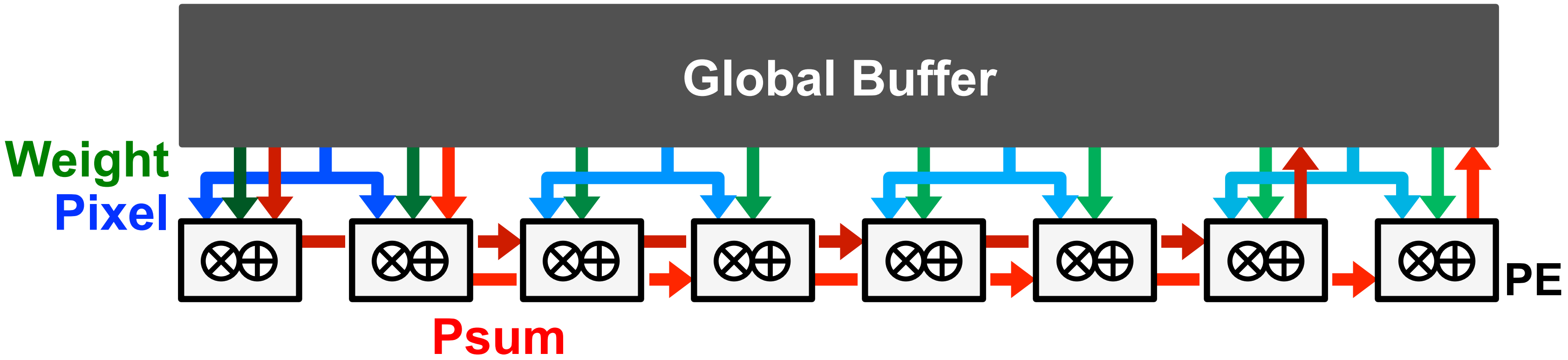
- **Minimize** **partial sum** R/W energy consumption
  - maximize local accumulation

- **Examples:**

  [**Gupta**, *ICML* 2015] [**ShiDianNao**, *ISCA* 2015]

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

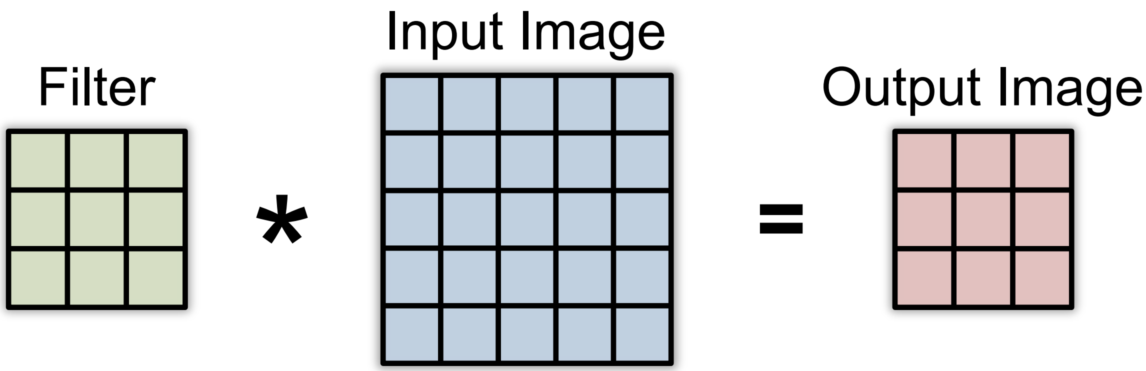
No Local Reuse (NLR)



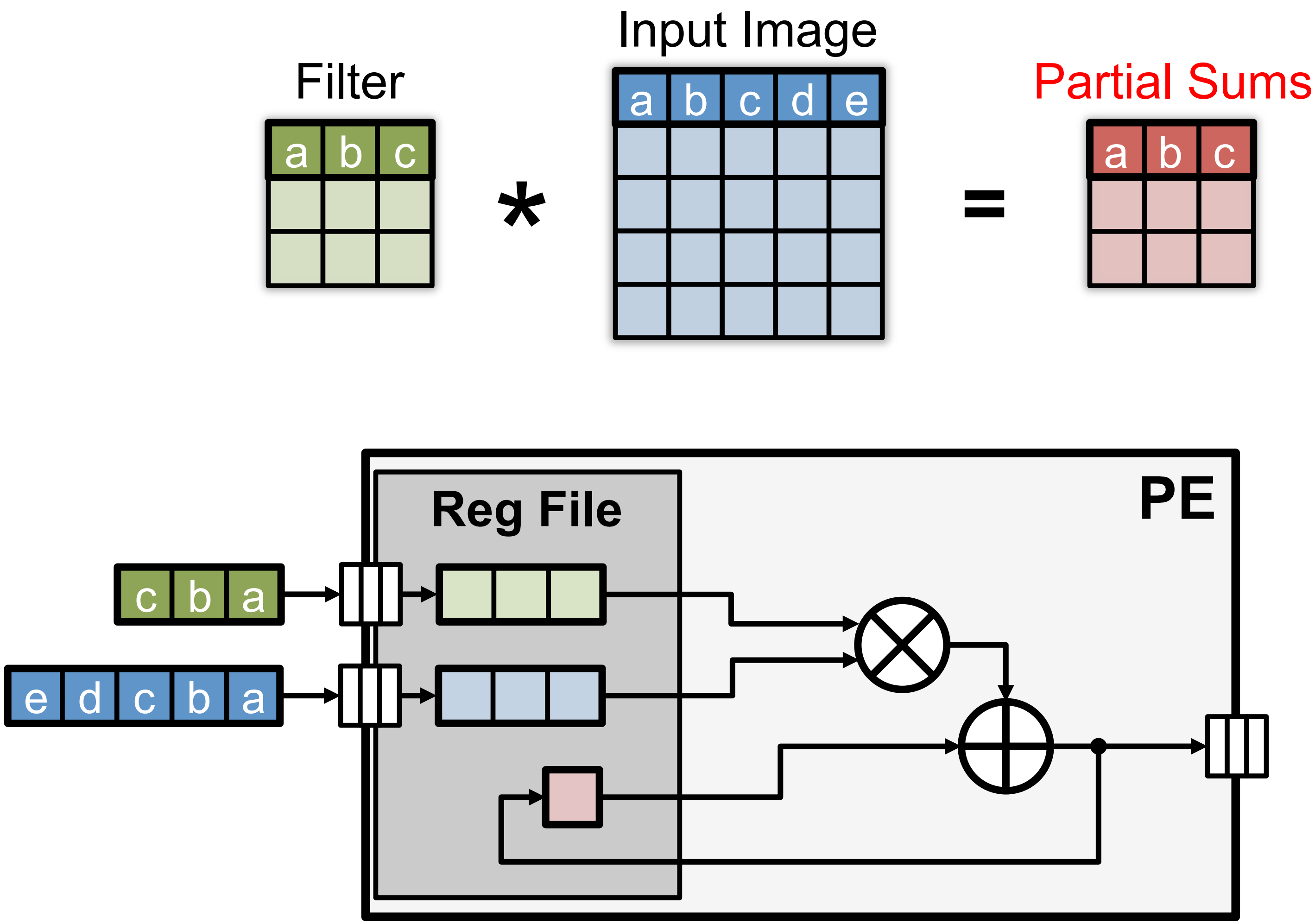
- Use a **large global buffer** as shared storage
  - Reduce **DRAM** access energy consumption
- **Examples:**

  [**DianNao**, *ASPLOS* 2014] [**DaDianNao**, *MICRO* 2014]
  [**Zhang**, *FPGA* 2015]

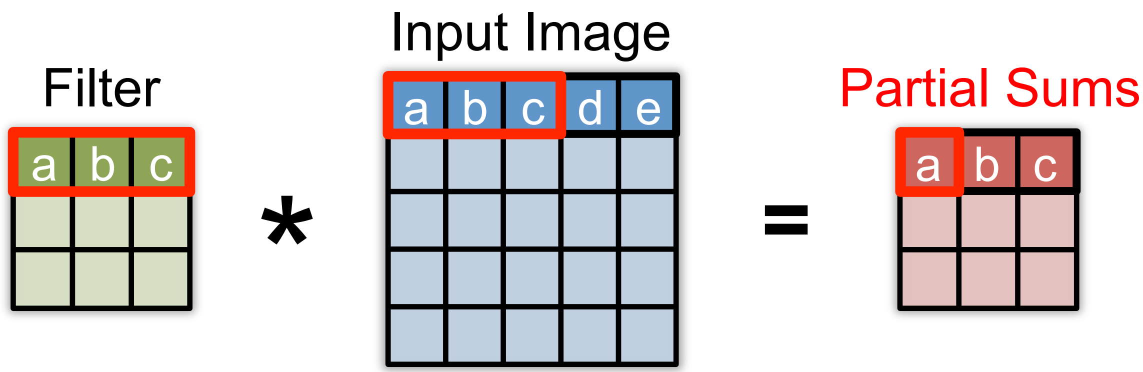
Row Stationary: Energy-efficient Dataflow

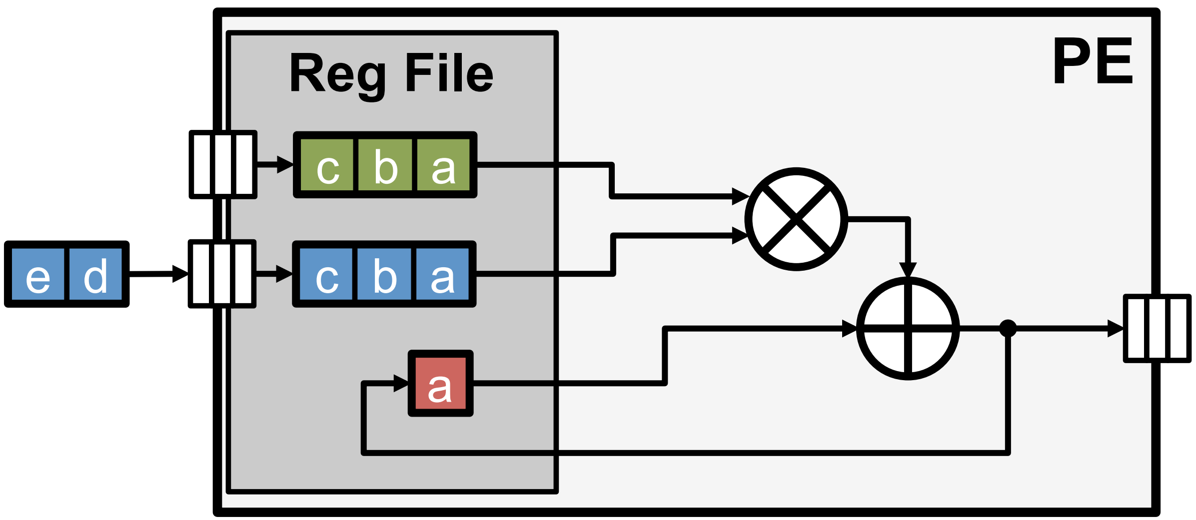


1D Row Convolution in PE

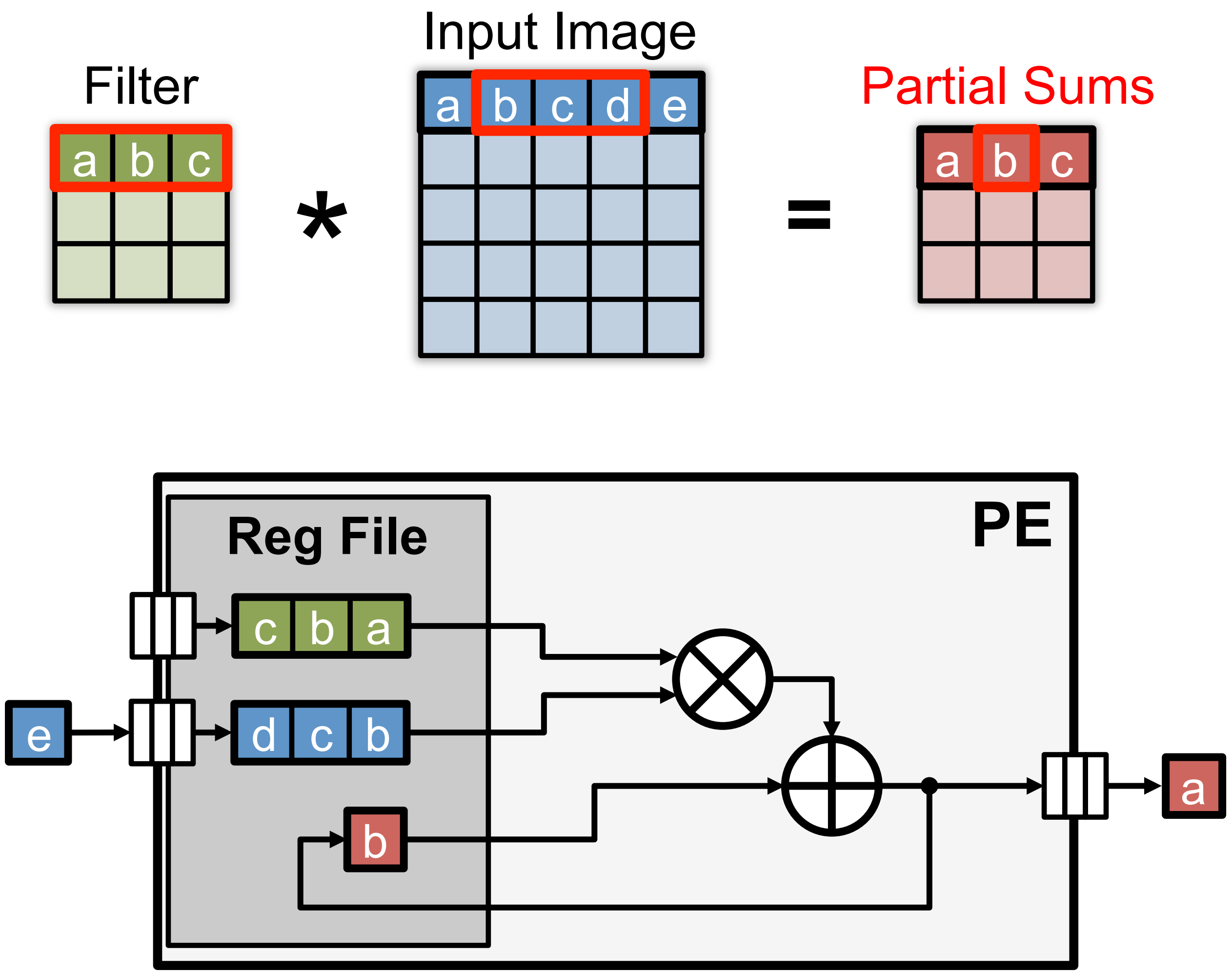


1D Row Convolution in PE

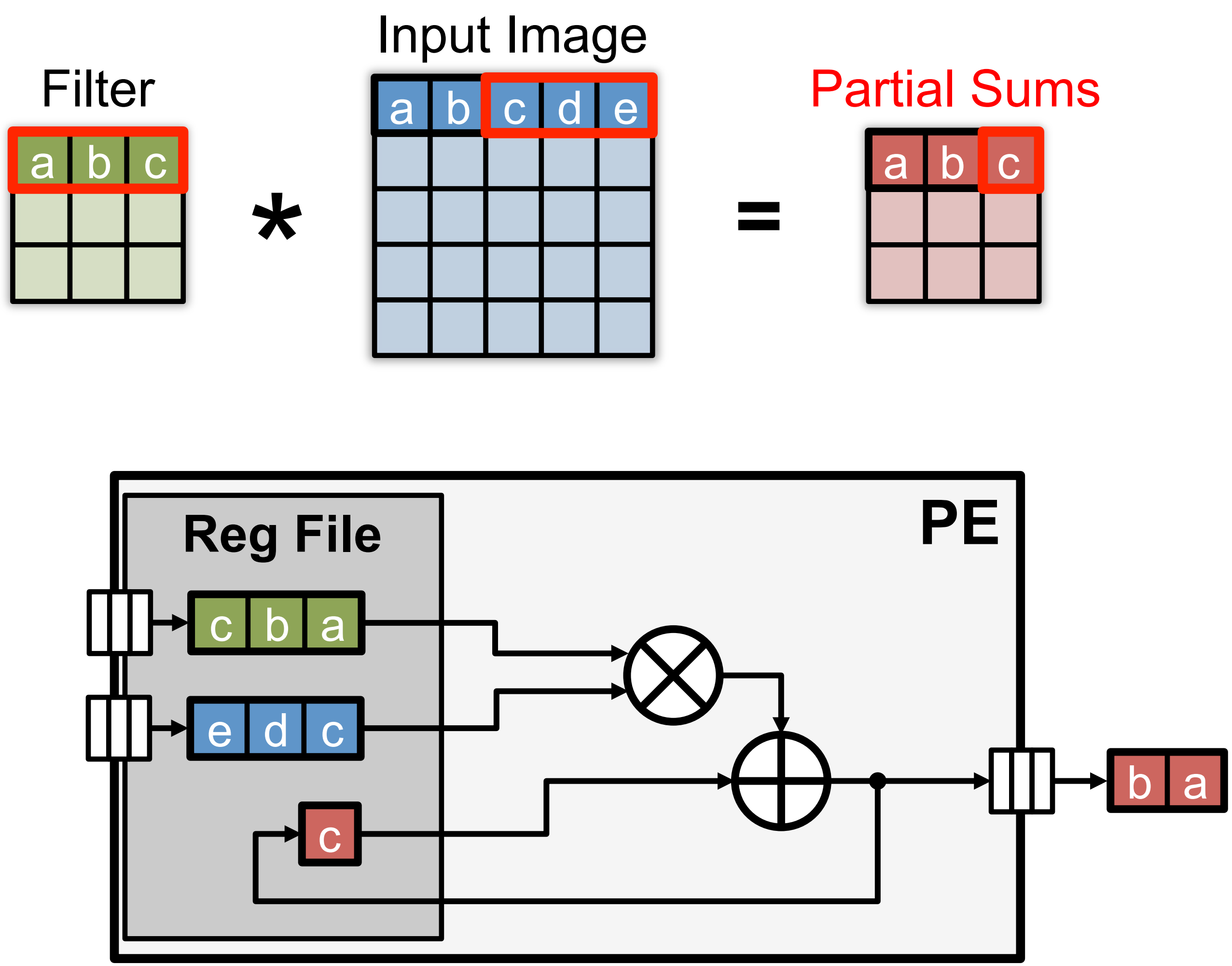




1D Row Convolution in PE

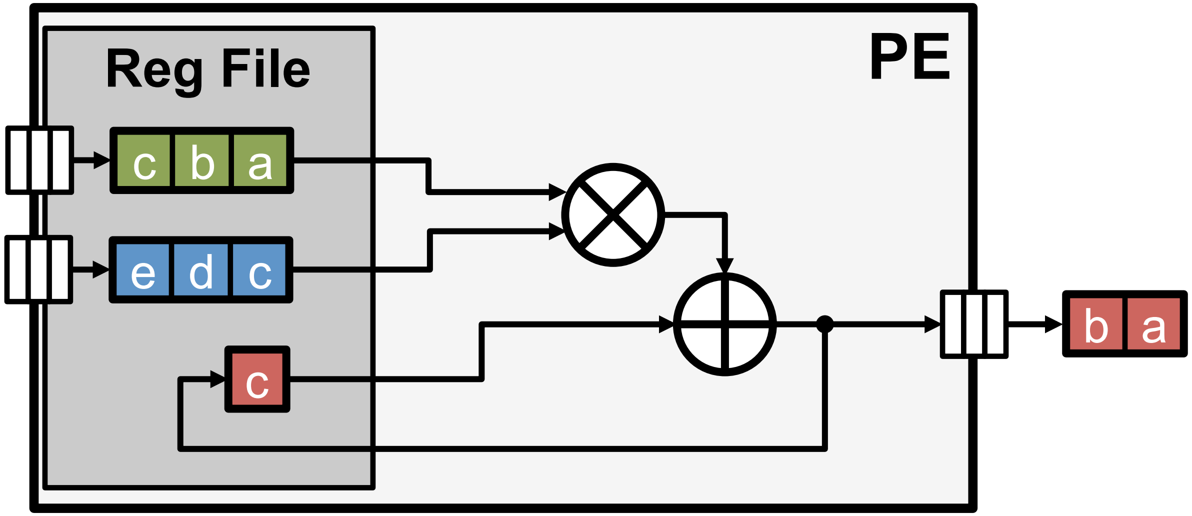


1D Row Convolution in PE

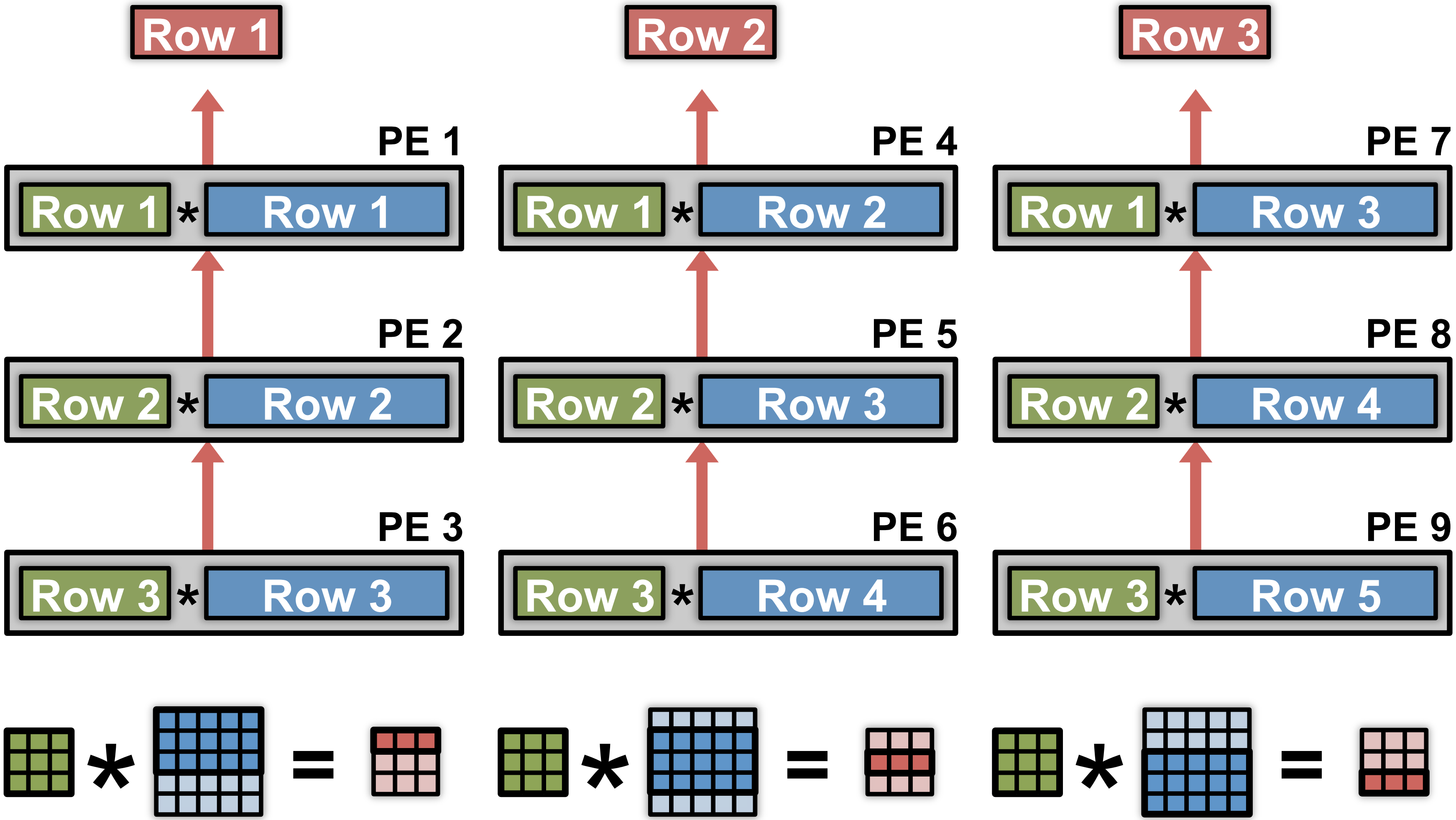


1D Row Convolution in PE

- 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

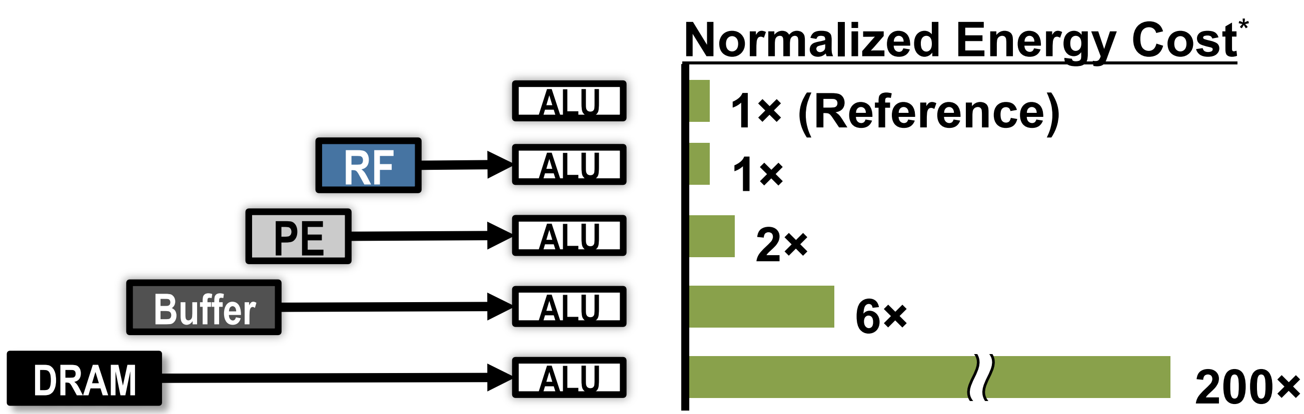
- **Weight Stationary**
  - Minimize movement of filter weights
- **Output Stationary**
  - Minimize movement of partial sums
- **No Local Reuse**
  - Don't use any local PE storage. Maximize global buffer size.
- **Row Stationary**

Evaluate Reuse in Different Dataflows

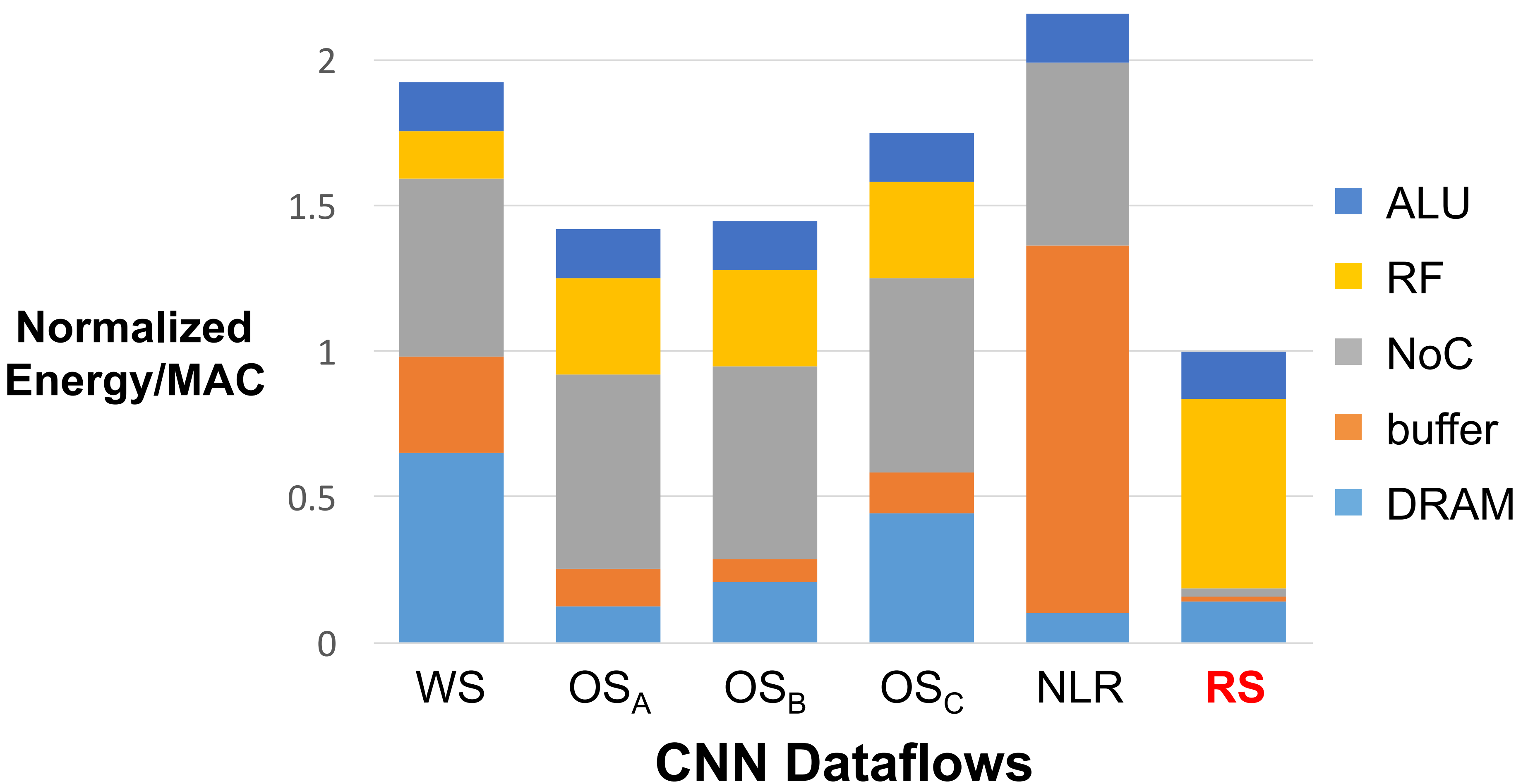
- **Weight Stationary**
  - Minimize movement of filter weights
- **Output Stationary**
  - Minimize movement of partial sums
- **No Local Reuse**
  - Don't use any local PE storage. Maximize global buffer size.
- **Row Stationary**

Evaluation Setup

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



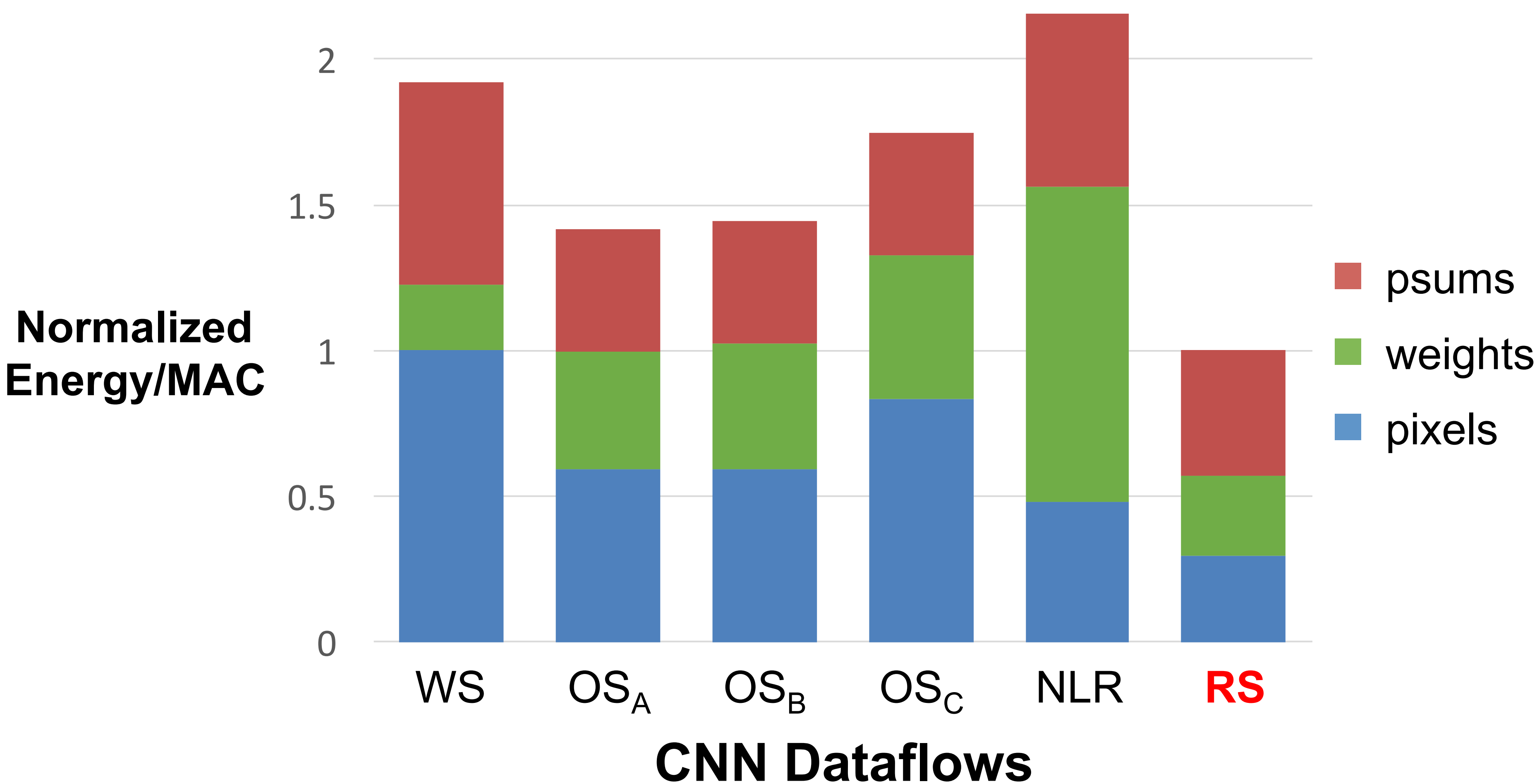
Dataflow Comparison: CONV Layers



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

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

Dataflow Comparison: CONV Layers

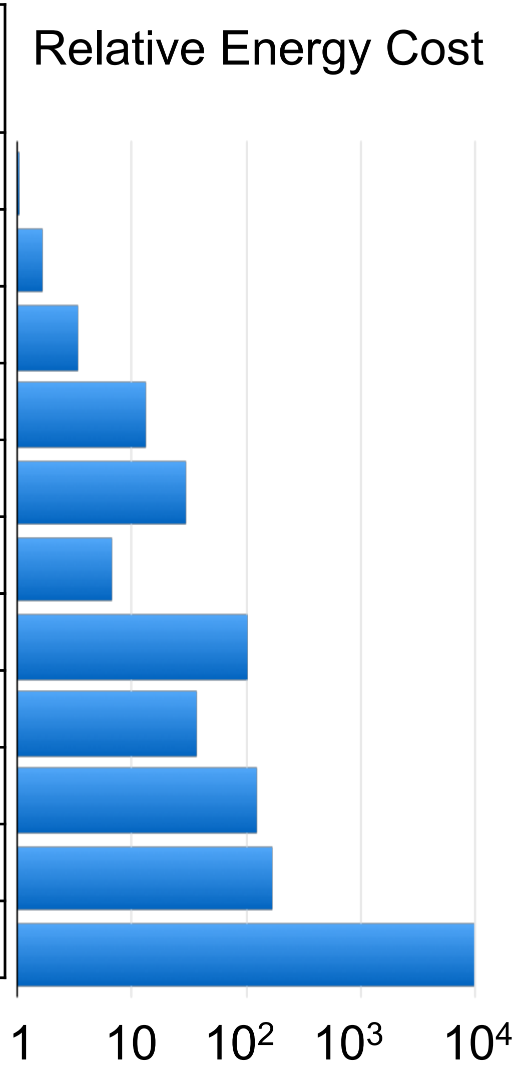


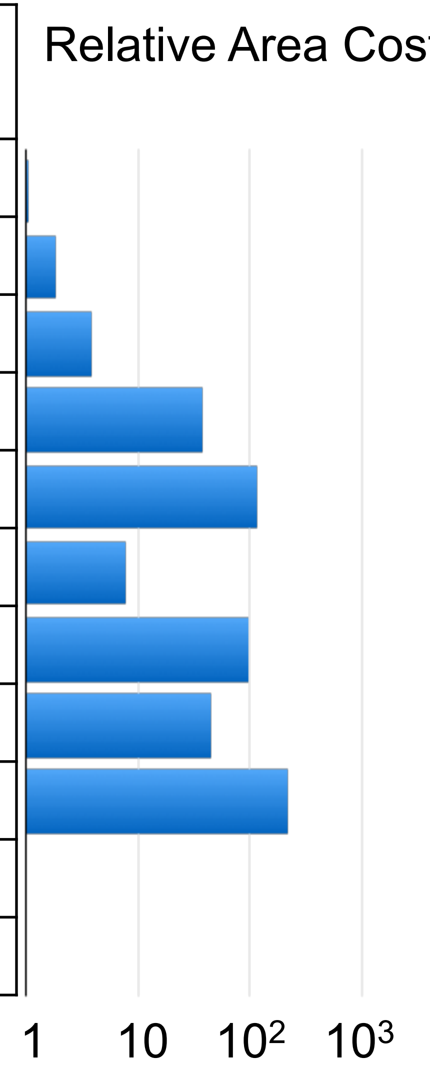
RS optimizes for the best **overall** energy efficiency

Opportunities in Joint Algorithm Hardware Design

Cost of Operations

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

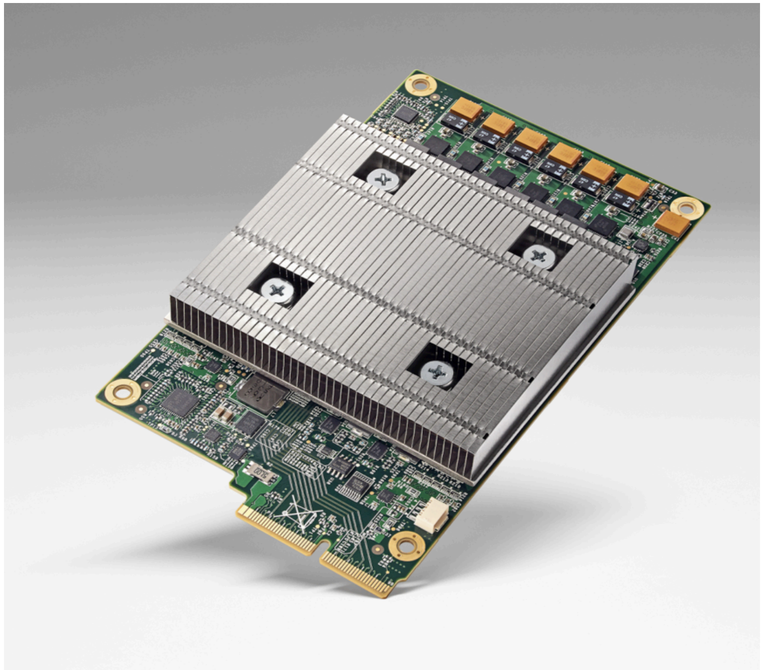




[Horowitz, “Computing’s Energy Problem (and what we can do about it)”, ISSCC 2014]

Commercial Products using 8-bit Integer

Nvidia's Pascal (2016)

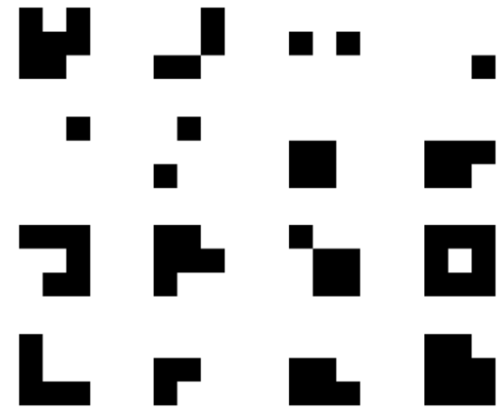


Google's TPU (2016)

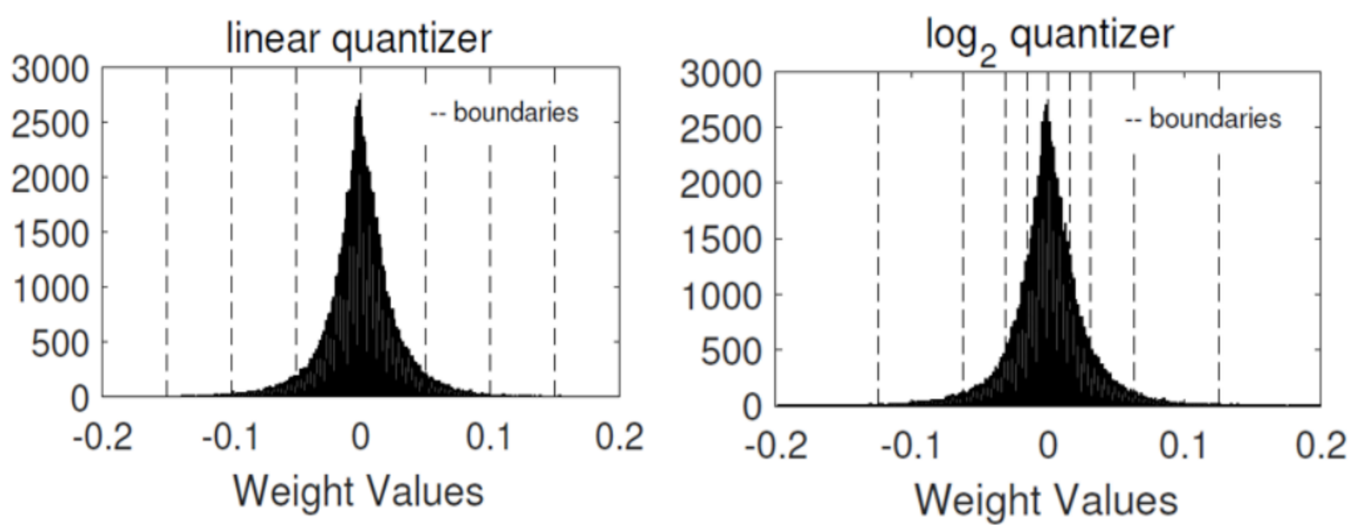
Reduced Precision in Research

- **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

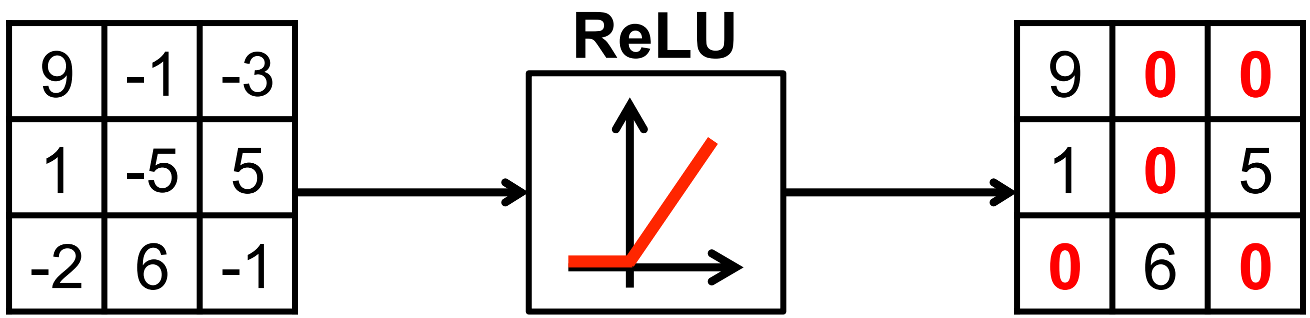


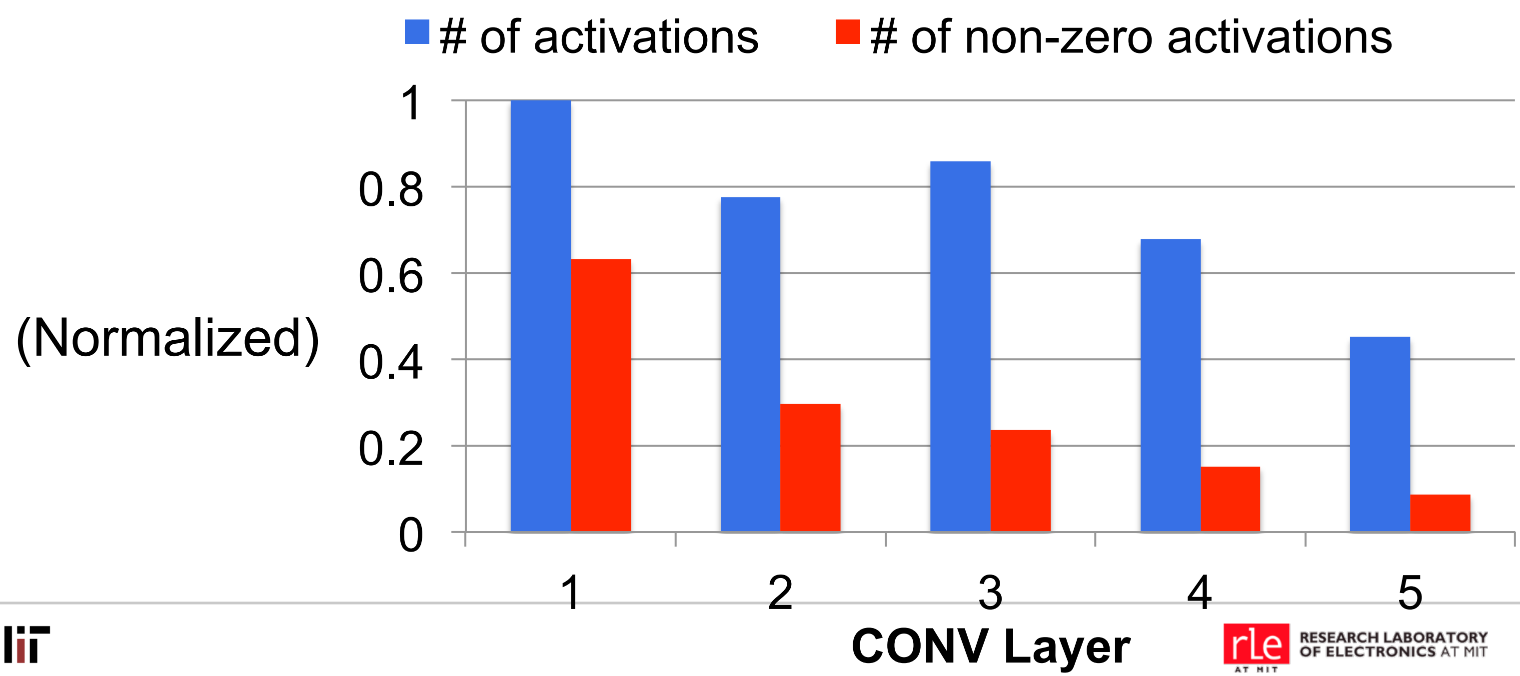
Log Domain Quantization



Sparsity in Data

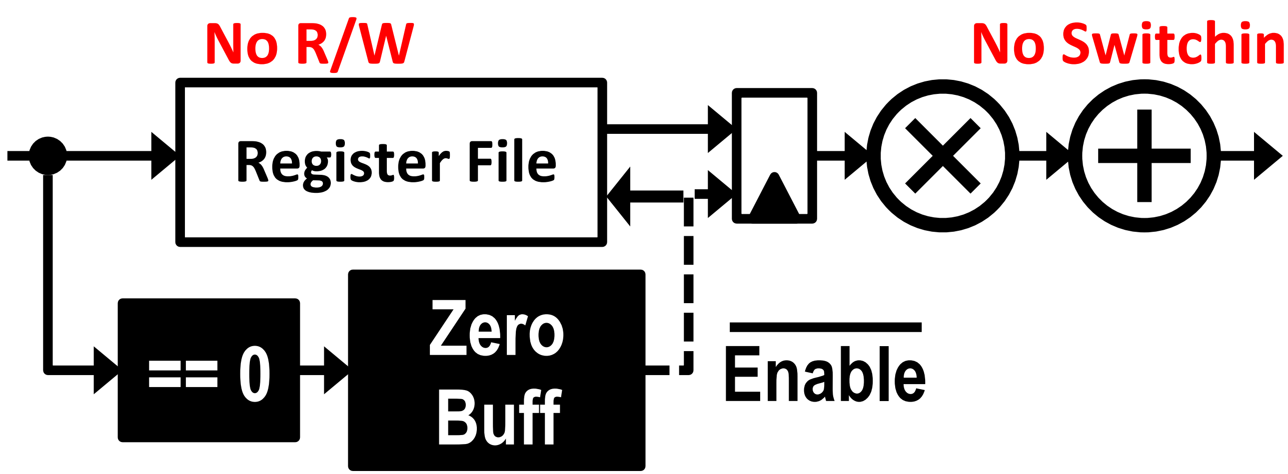
Many **zeros** in **output fmaps** after **ReLU**





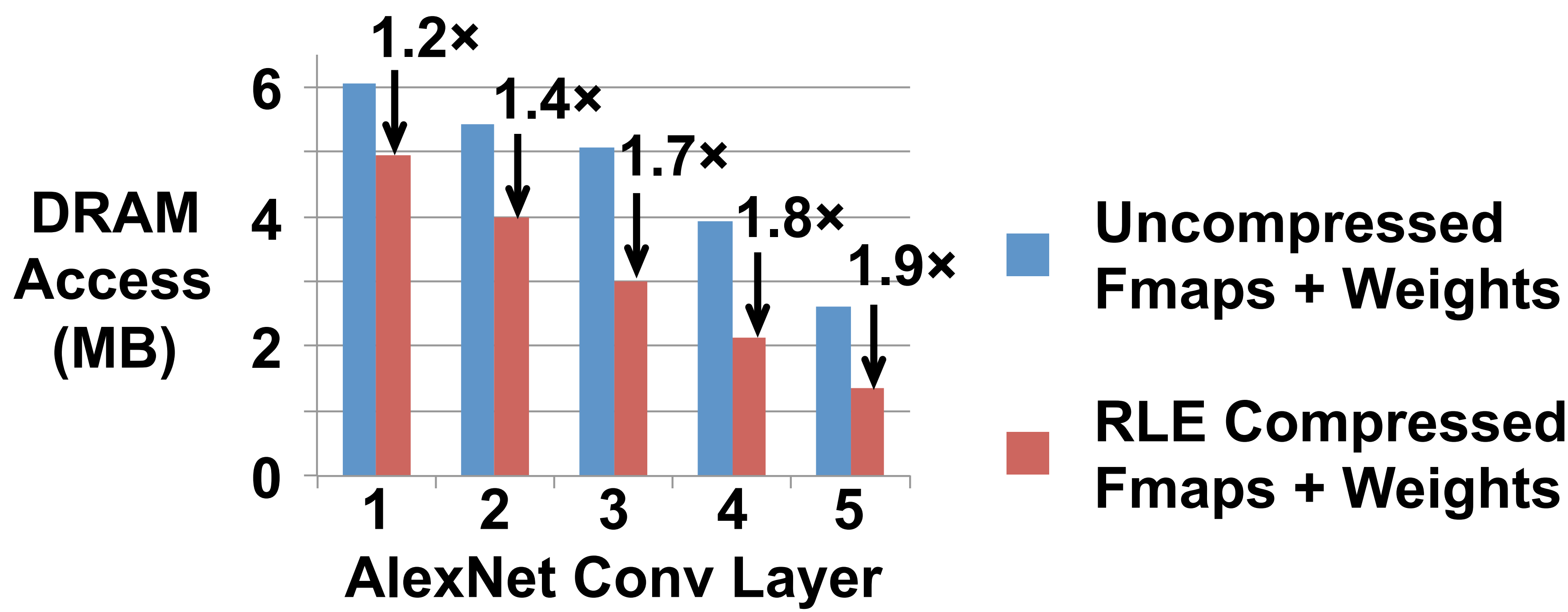
Zero Data Processing Gating

- Skip PE local **memory access**
- Skip MAC **computation**
- Save PE processing power by 45%



[Chen et al., ISSCC 2016]

Compression Reduces DRAM BW



Simple RLC within 5% - 10% of theoretical entropy limit

[Chen et al., ISSCC 2016]

Sparsity with Basis Projection

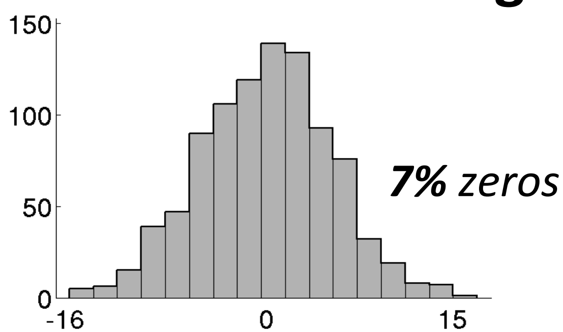
Reduce the number of multiplications by projecting onto a basis that increases sparsity (>1.8x power reduction)

Basis Projection Equation

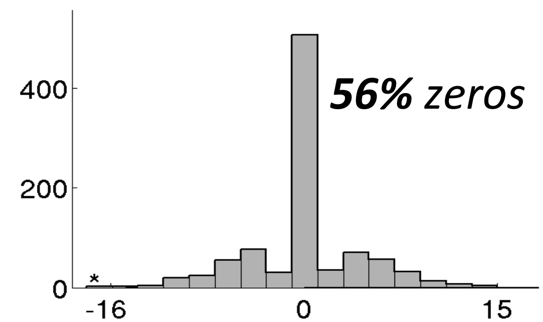
$$\langle H, W \rangle = \left\langle H, \sum_d S_d \alpha_d \right\rangle = \sum_d \langle H, S_d \rangle \alpha_d = \sum_d P_d \alpha_d$$

Features Weights Basis Projected Features Projected Weights

Histogram of Weights



Weights (W)

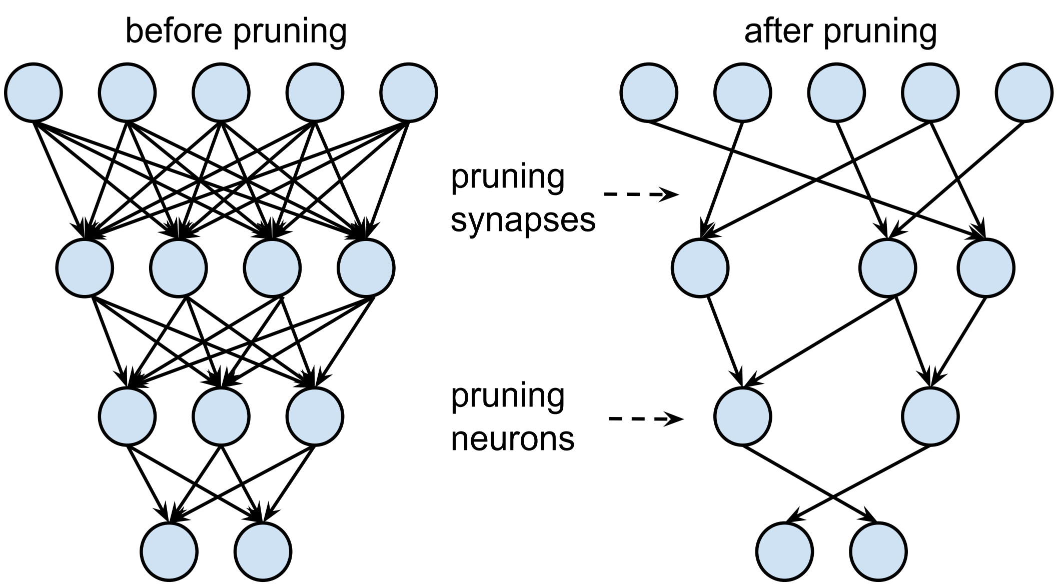


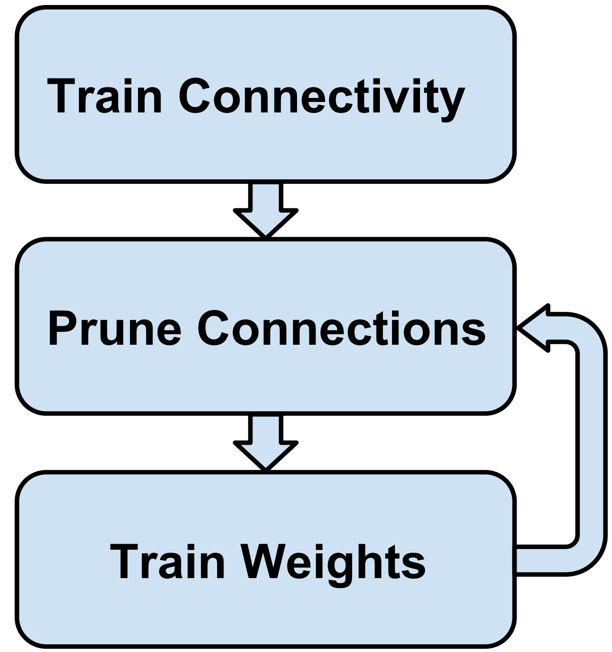
Projected Weights (α_d)

[Suleiman et al., VLSI 2016]

Pruning – Make Weights Sparse

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



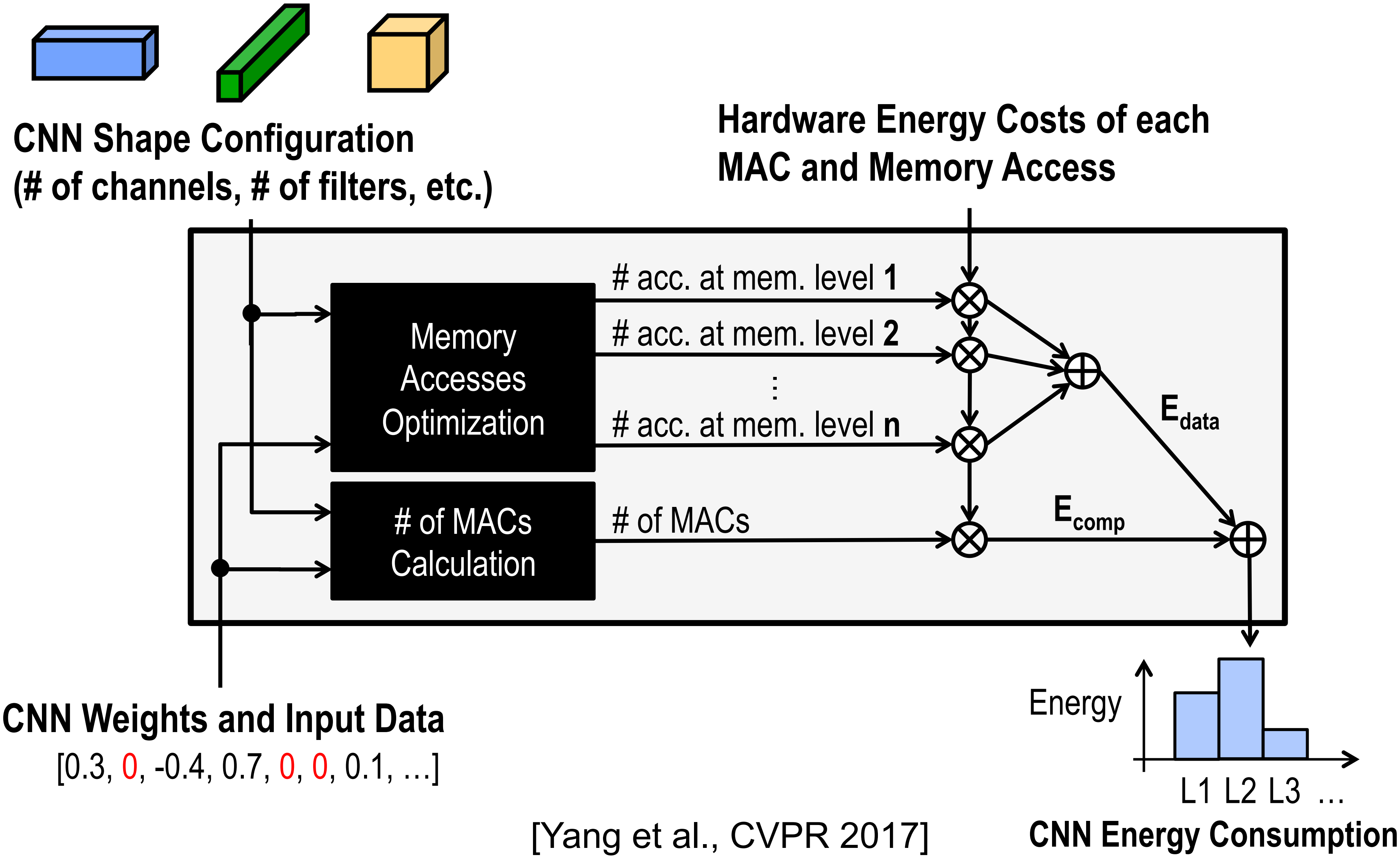


Example: *AlexNet*
Weight Reduction: *CONV layers 2.7x, FC layers 9.9x*
(Most reduction on fully connected layers)
Overall: *9x weight reduction, 3x MAC reduction*

Key Metrics for Embedded DNN

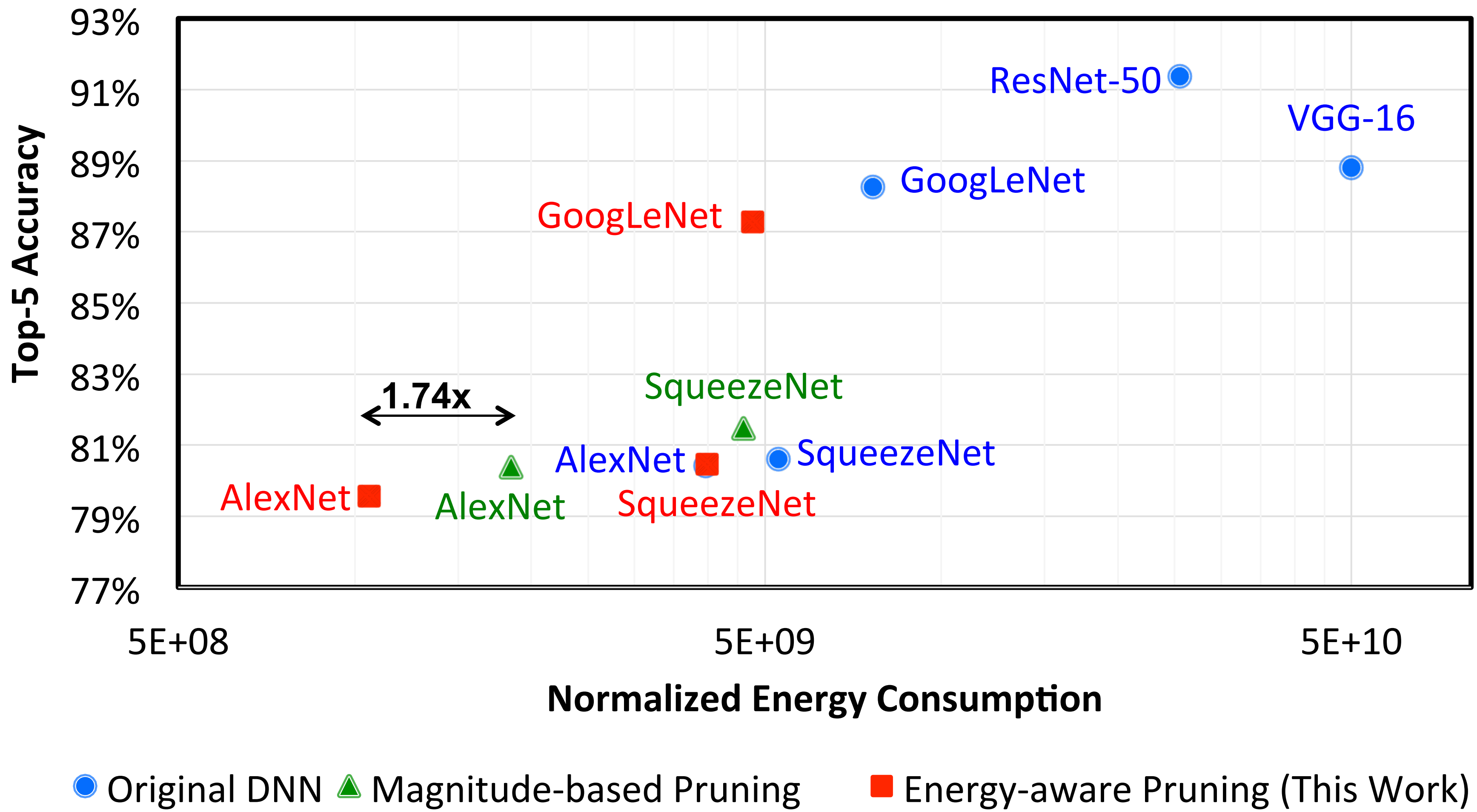
- **Accuracy → Measured on Dataset**
- **Speed → Number of MACs**
- **Storage Footprint → Number of Weights**
- **Energy → ?**

Energy-Evaluation Methodology



[Yang et al., CVPR 2017]

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]

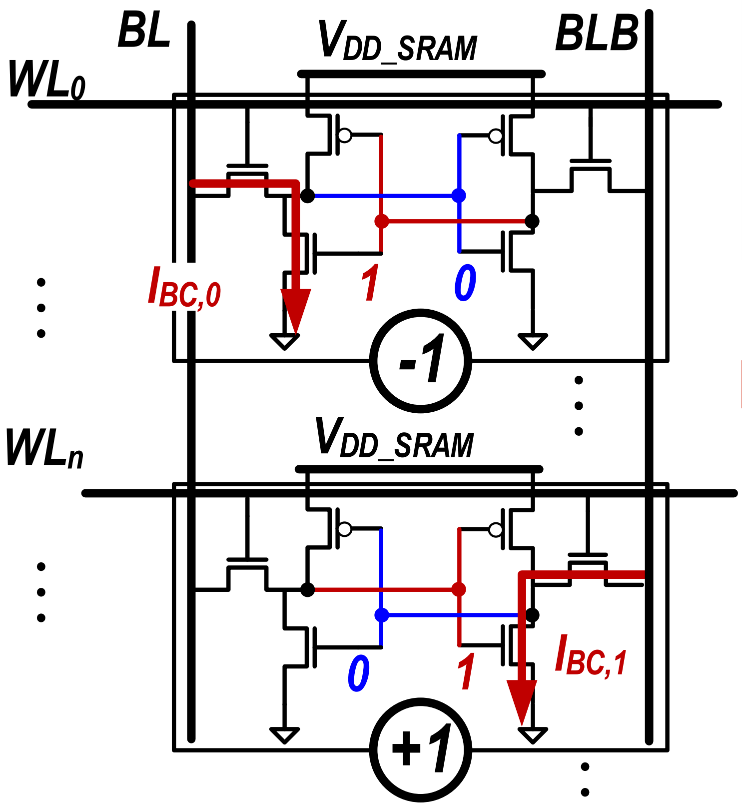
Opportunities in Mixed Signal Circuits

Reduce data movement by embedding computation into memory and sensor

Mixed-Signal Circuit Processing

- **Primarily target dot product**
  - **Reduced precision (e.g., binary weights)**
- **Challenges**
  - **Need ADC and DAC conversion**
    - **Weights trained in digital domain**
  - **More sensitive to variations and nonlinearity**
- **Reduce data movement from memory and sensor**

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:

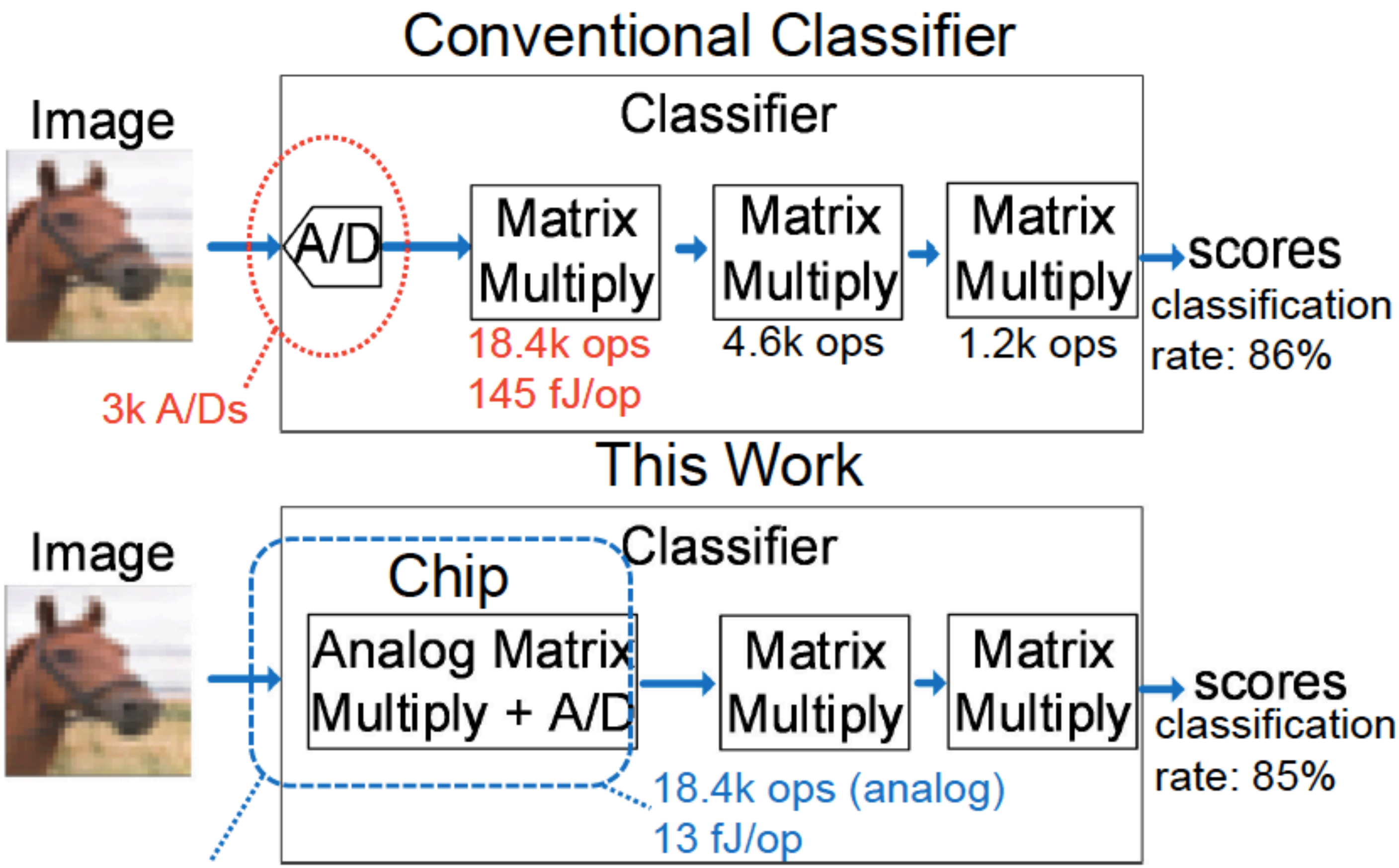
1. **Weights restricted to be +/-1**
2. **Bit-cell discharge subject to variation, nonlinearity**

[Zhang et al., VLSI 2016]

Switched Cap MAC for Classification

Reduce ADC conversions by 21x

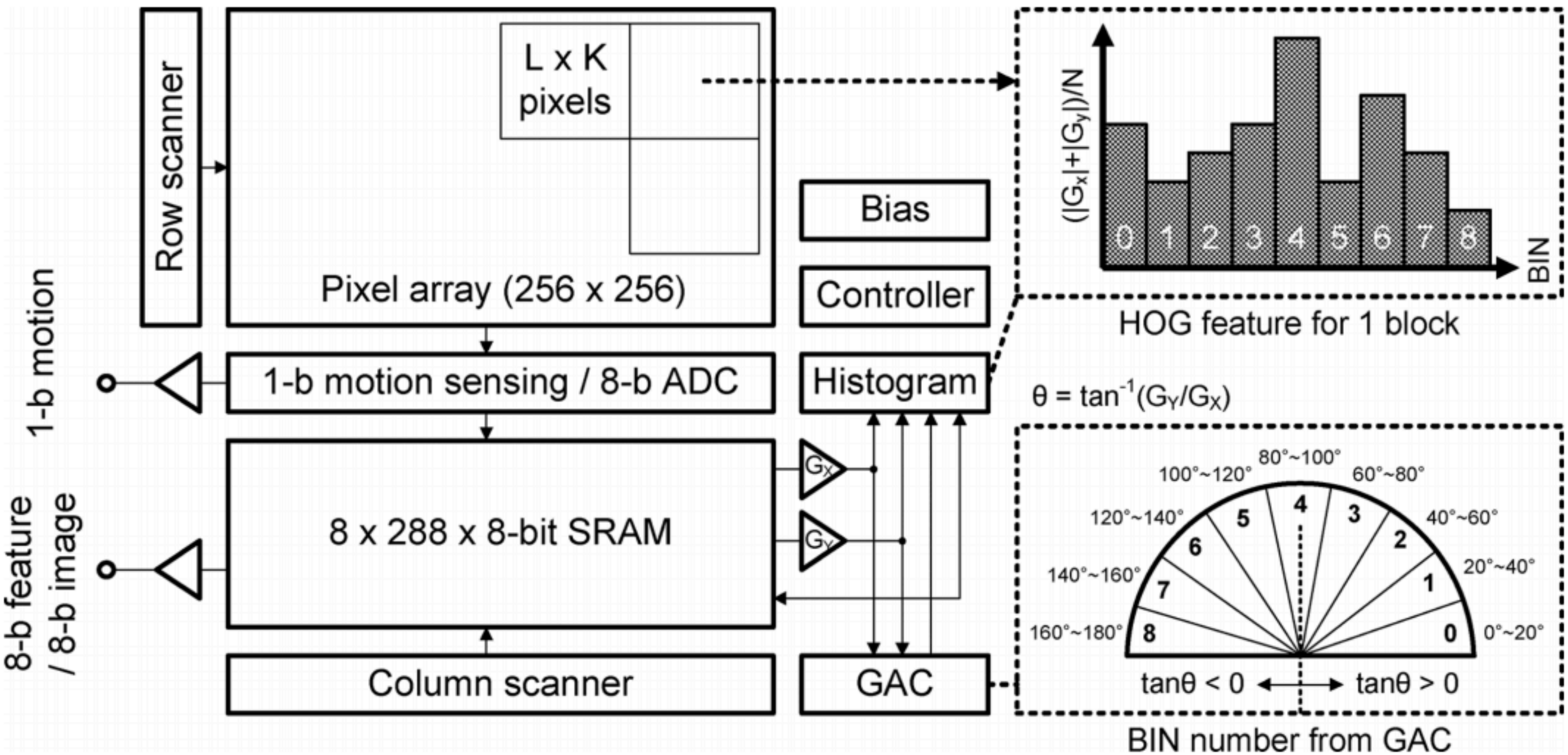
Input: 32x32x3 (6b) → Output: 4x4x9 (6b); Weight 3b



Embedded Feature Extraction in Sensor

Compute the HOG feature in Image Sensor

- Reduce bandwidth by 96.5% (vs. 8b output)
- Mixed-signal computation of gradient angle



[Choi et al., ISSCC 2013]

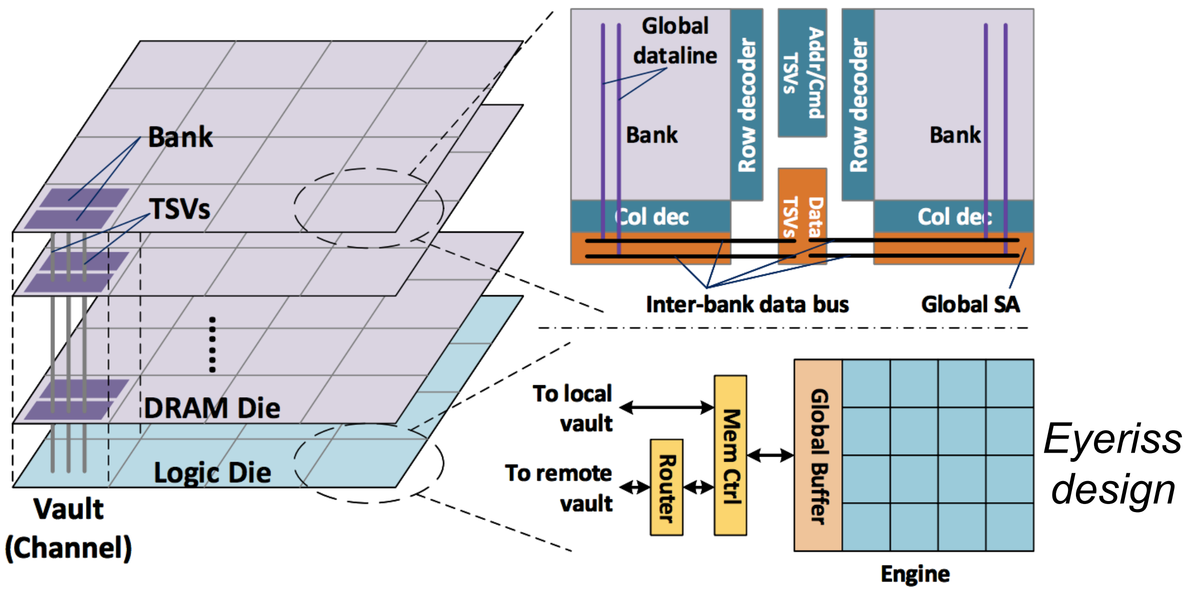
Opportunities in Advanced Technologies

Reduce data movement by embedding computation into memory and sensor

Advanced Memory Technologies

Many new memories and devices explored to reduce data movement

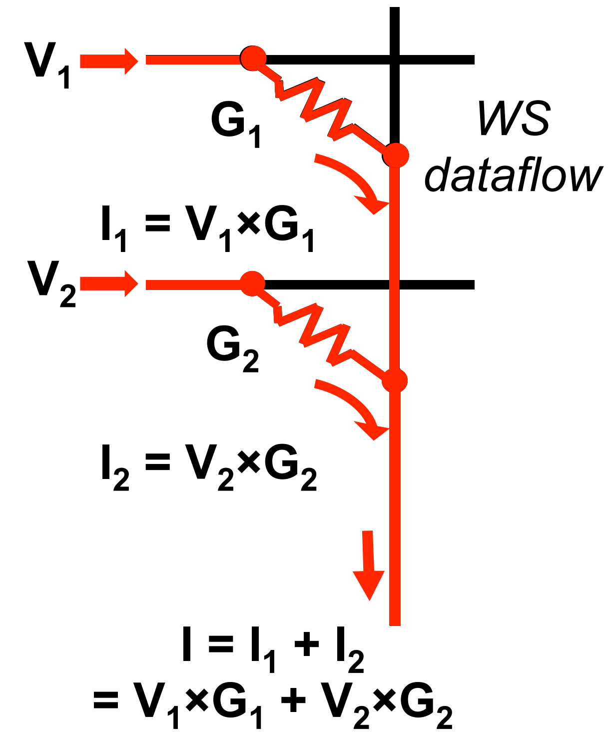
Stacked DRAM



[Gao et al., Tetris, ASPLOS 2017]

[Kim et al., NeuroCube, ISCA 2016]

Non-Volatile Resistive Memories



[Shafiee et al., ISCA 2016]

[Chi et al., PRIME, ISCA 2016]

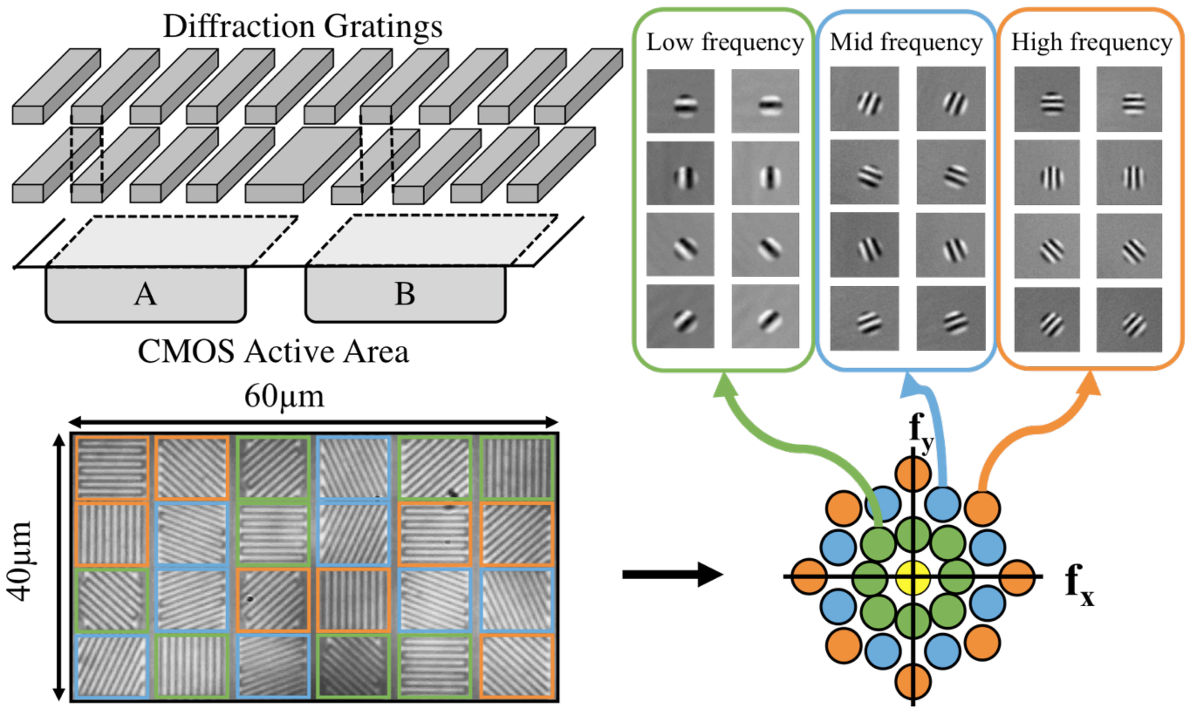
eDRAM

[Chen et al., DaDianNao, MICRO 2014]

ASP: Angle Sensitive Pixels

Extract gradients directly in the sensor

- Reduces read bandwidth by 10x
- Reduces ADC conversion by 10x



[Chen et al., CICC 2012]

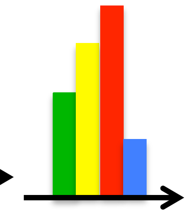
Hand-Crafted vs. Learned Features

Machine Learning Pipeline (Inference)

Image

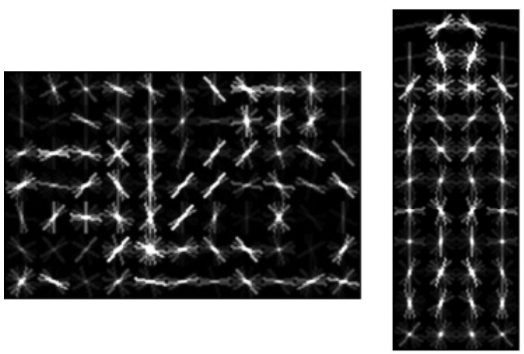
pixels → **Feature Extraction** → Features (**x**) → **Classification** ***(w^Tx)*** → Scores

Trained weights (**w**) → **Classification**



Scores per class
(select class based on max or threshold)

Handcrafted Features (e.g. HOG)



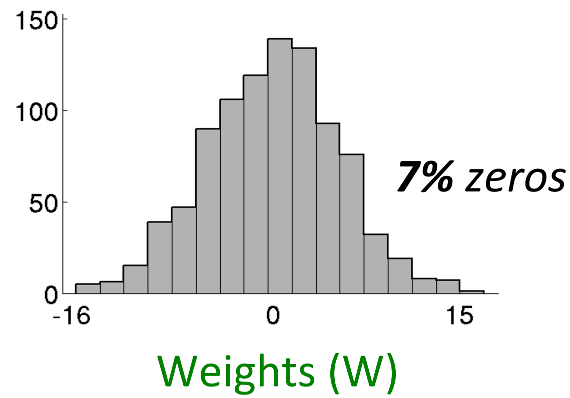
Learned Features (e.g. DNN)

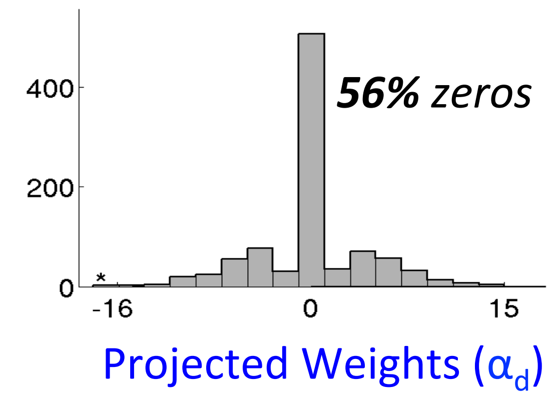
$$\textbf{Score} = \Sigma_n \, x_i \, w_i$$

Joint Algorithm Hardware Optimizations

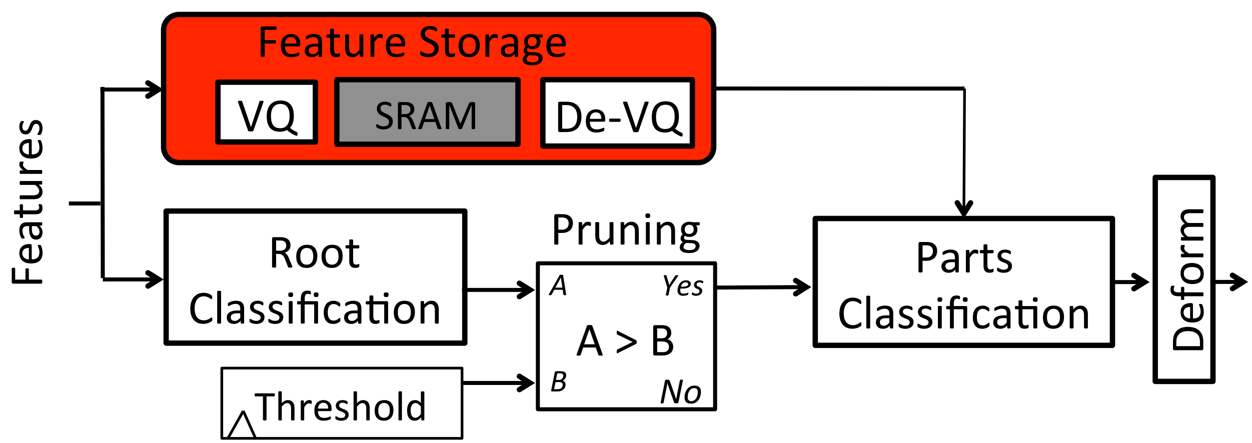
Histogram of Weights

Exploit Sparsity



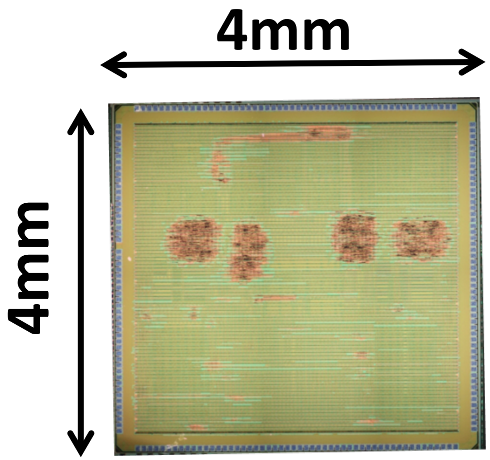


Exploit Compression

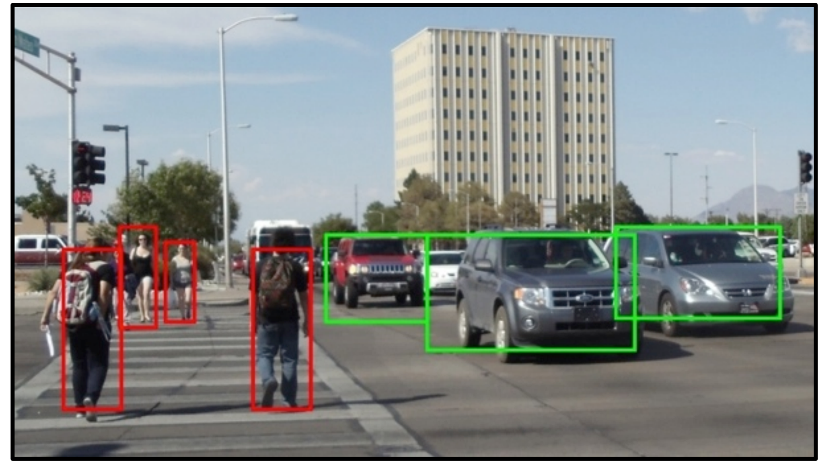


[Suleiman et al., VLSI 2016]

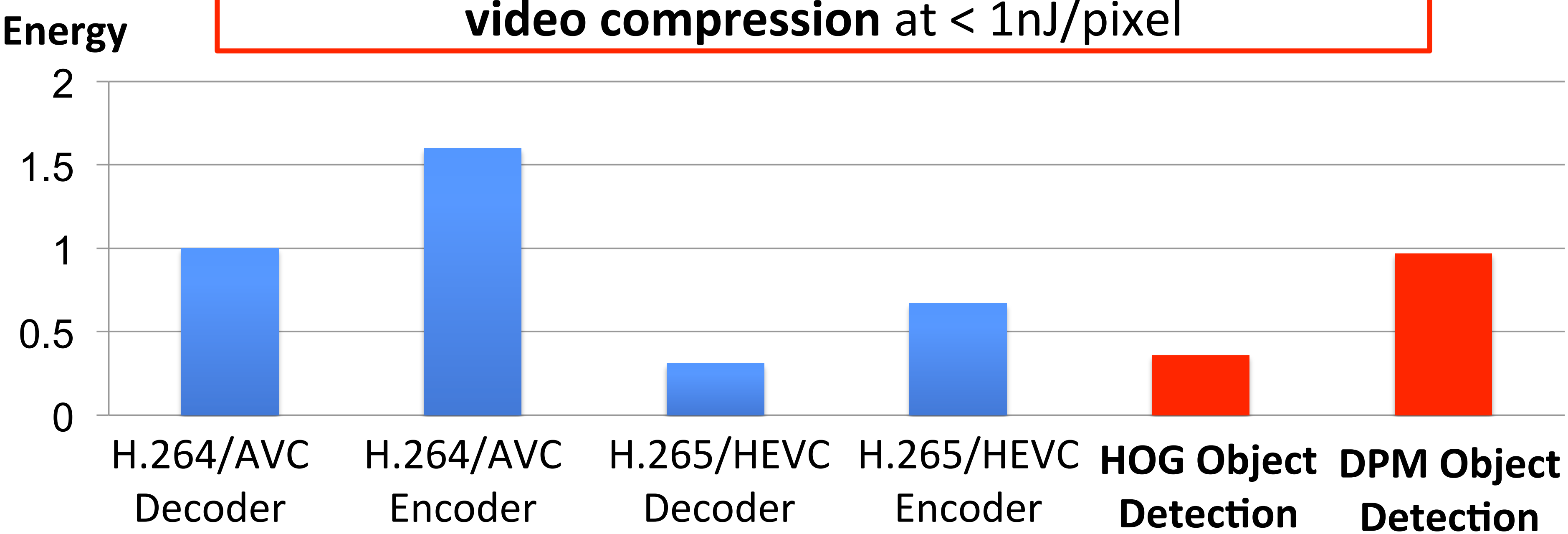
Energy-Efficient Object Detection



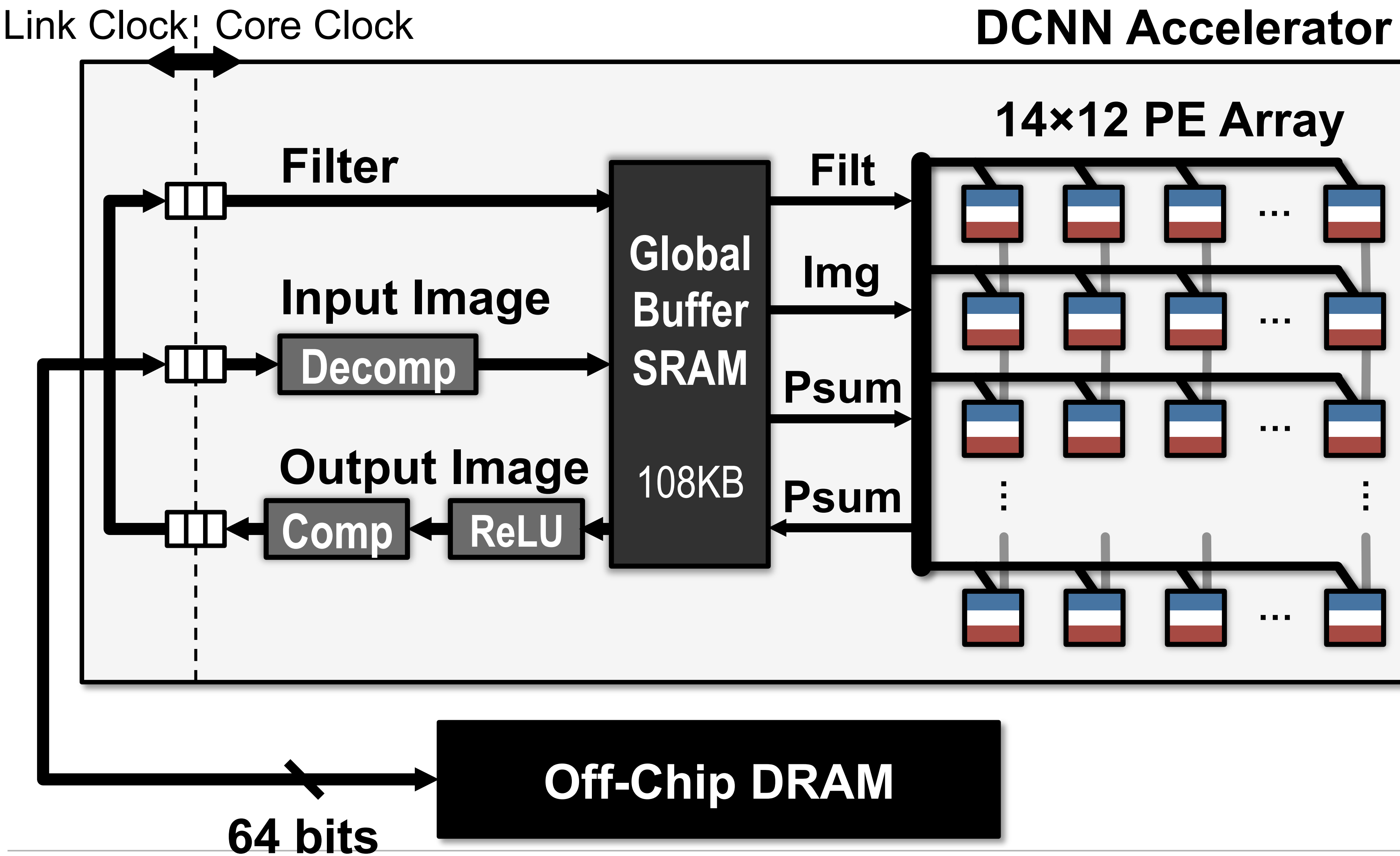
MIT Object
Detection Chip
[VLSI 2016]



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



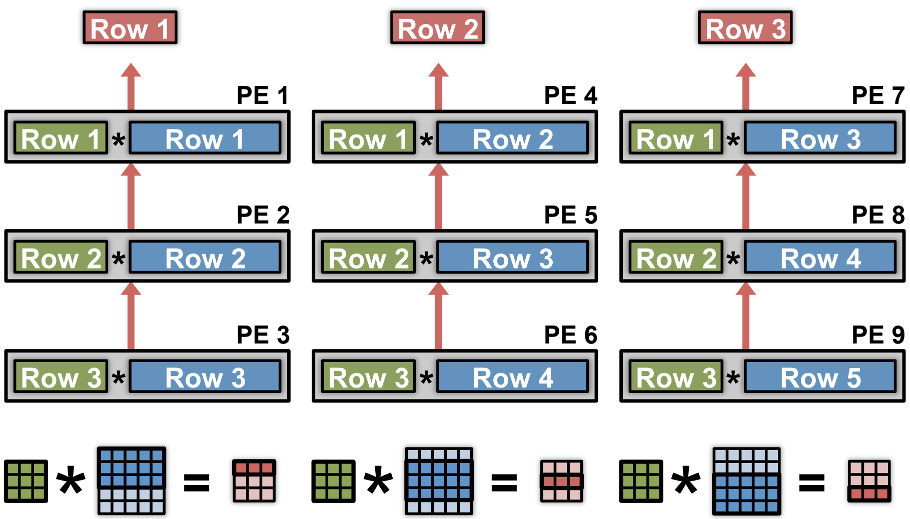
Eyeriss Deep CNN Accelerator



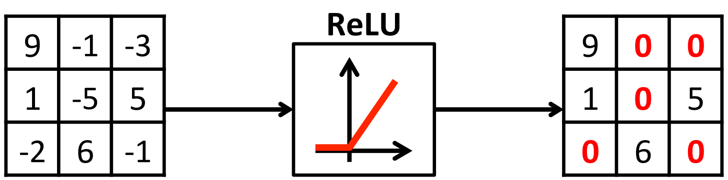
[Chen et al., ISSCC 2016]

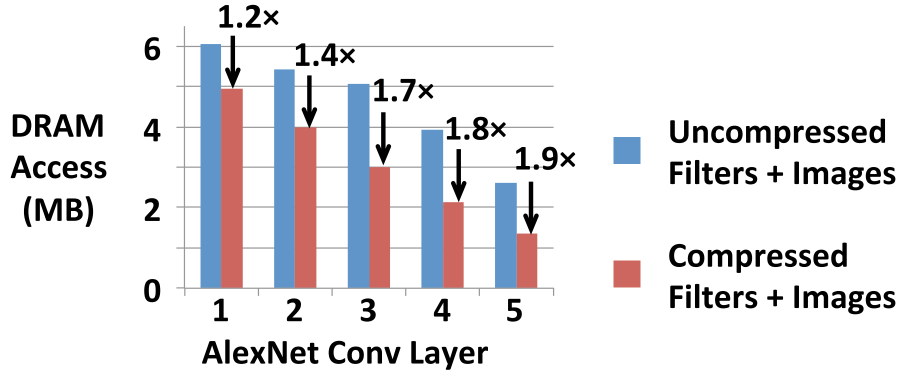
Optimization to Reduce Data Movement

- Energy-efficient **dataflow to reduce data movement**
- **Exploit data statistics** for high energy efficiency



Apply Non-Linearity (**ReLU**) on Filtered Image Data

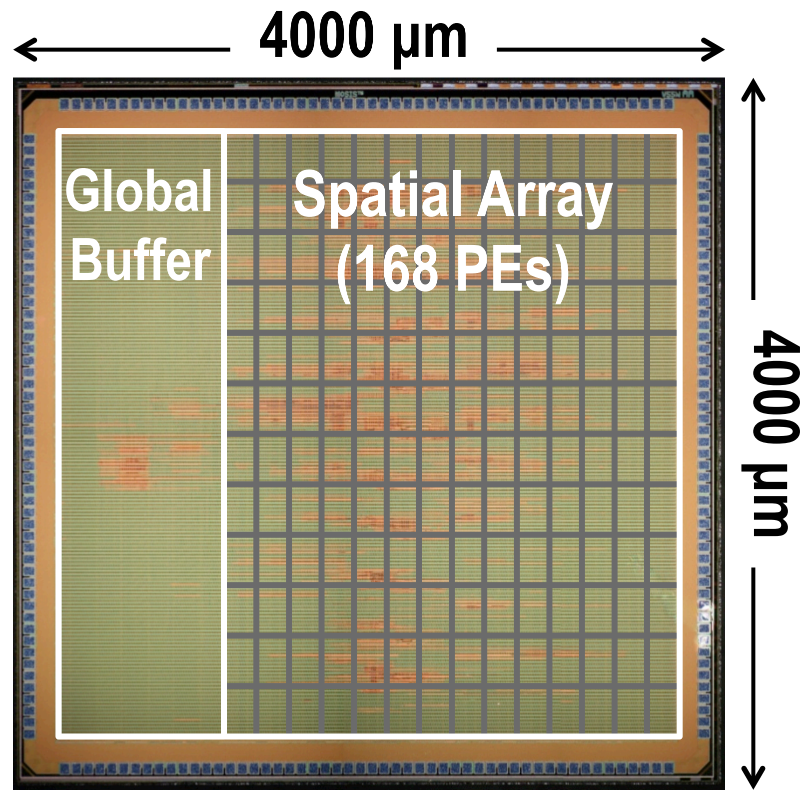




[Chen et al., ISCA 2016, ISSCC 2016]

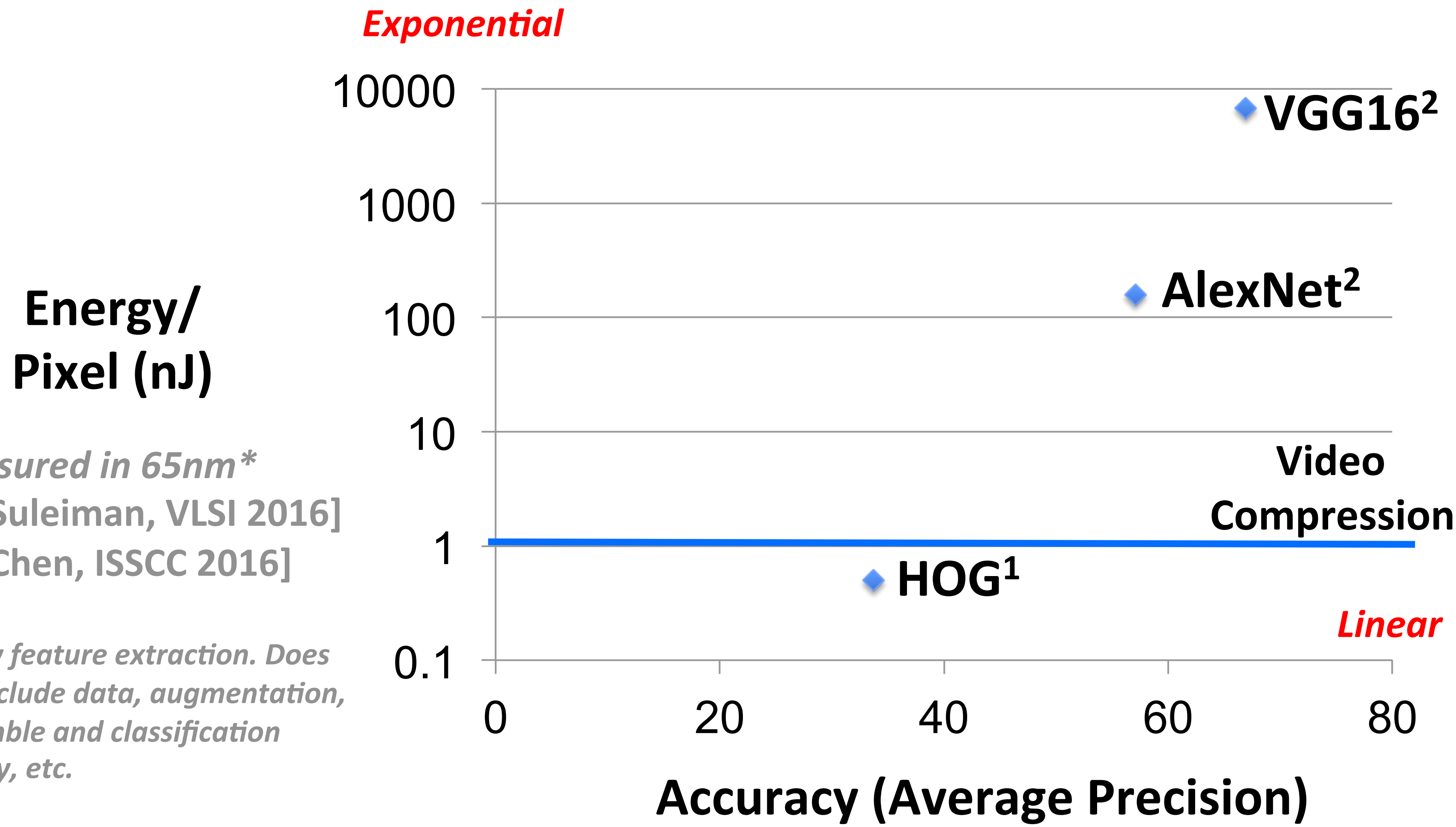
Eyeriss Chip Spec & Measurement Results

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)

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]

[Suleiman et al., ISCAS 2017]

Summary

- **Machine Learning is an important area of research**
  - Wide range of applications
  - Various methods to extract features (hand-crafted and learned)
- **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, Advanced Technologies
  - Important to consider interactions between levels to maximize impact

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 · 3M · ANALOG DEVICES · BROADCOM · intel · NVIDIA · DARPA · SRC · QUALCOMM · SAMSUNG · TEXAS INSTRUMENTS · tsmc

References

More info about **Eyeriss** and
Tutorial on DNN Architectures at
http://eyeriss.mit.edu

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

More info about research in the **Energy-Efficient Multimedia Systems Group @ MIT**
http://www.rle.mit.edu/eems

For updates Follow @eems_mit

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