Efficient Processing of Deep Neural Networks

Vivienne Sze

Massachusetts Institute of Technology

In collaboration with Yu-Hsin Chen, Joel Emer, Tushar Krishna, Amr Suleiman, Tien-Ju Yang, Zhengdong Zhang

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

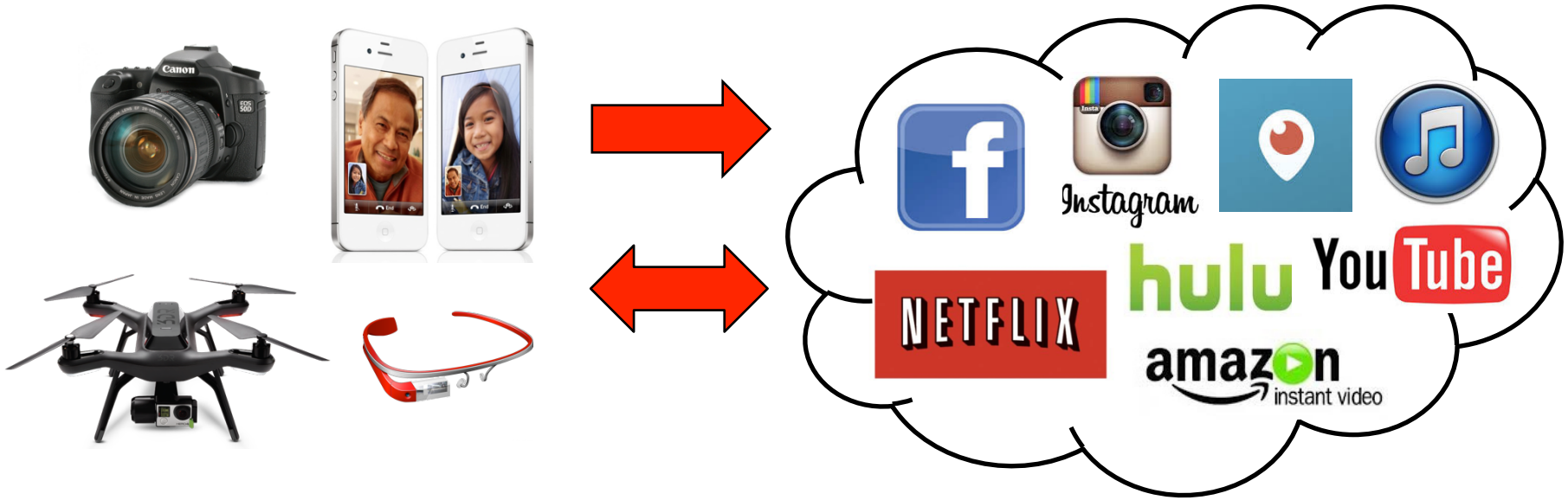
Follow @eems_mit

RESEARCH LABORATORY OF ELECTRONICS AT MIT

MTL microsystems technology laboratories massachusetts institute of technology

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**



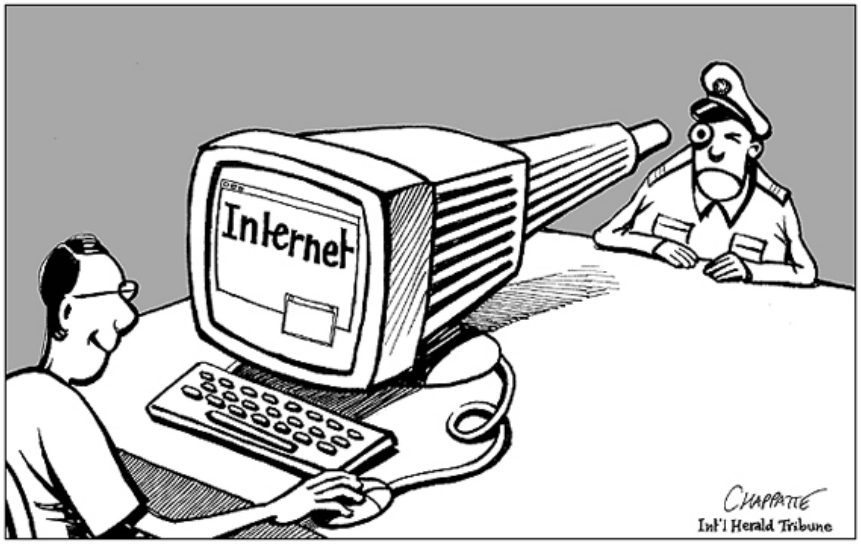
Energy limited due to battery capacity

Power limited due to heat dissipation

Need energy-efficient pixel processing!

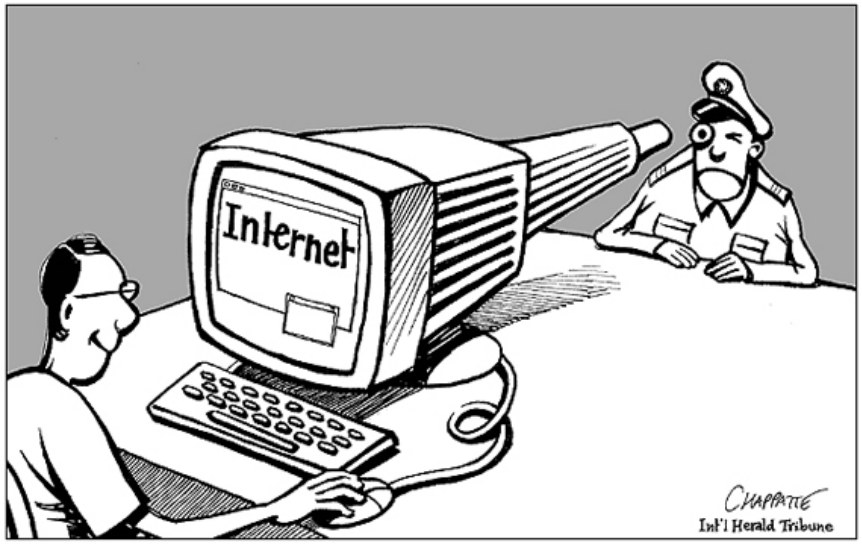
Processing at “Edge” instead of the “Cloud”

Privacy



Processing at “Edge” instead of the “Cloud”

Privacy



Latency

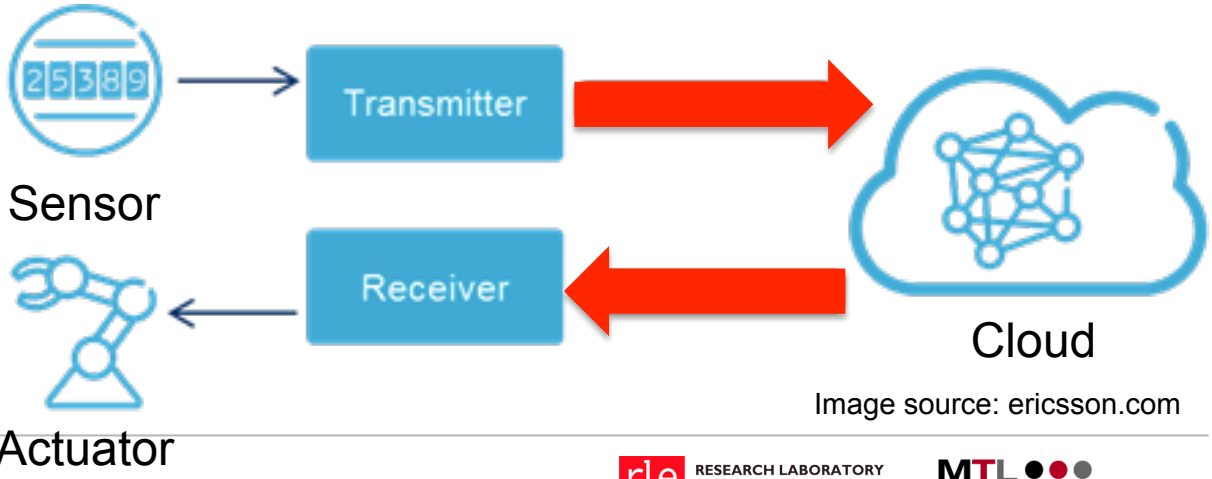
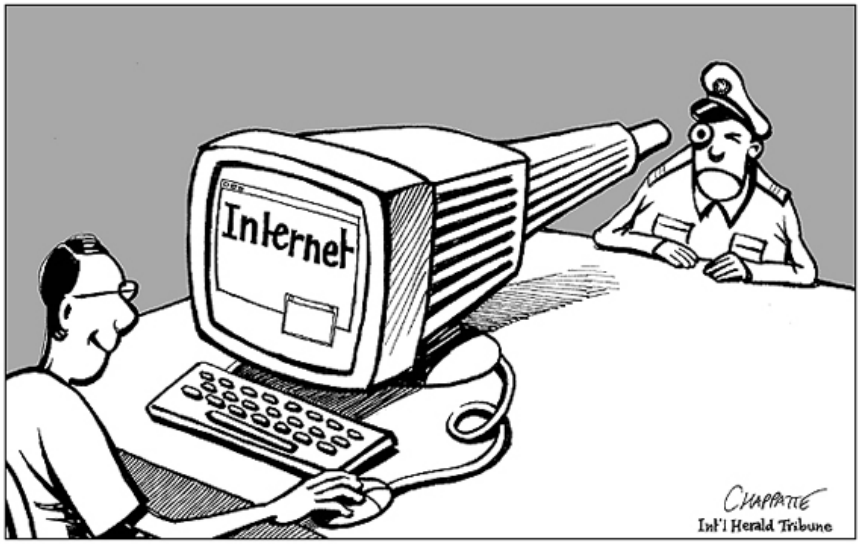


Image source: ericsson.com

Processing at "Edge" instead of the "Cloud"

Privacy



Communication

Image source:
www.theregister.co.uk

Latency

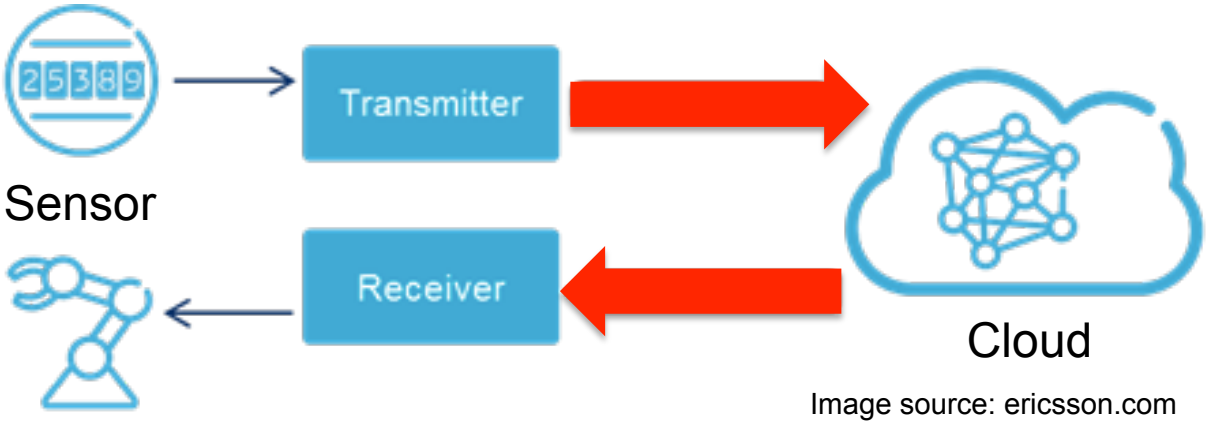
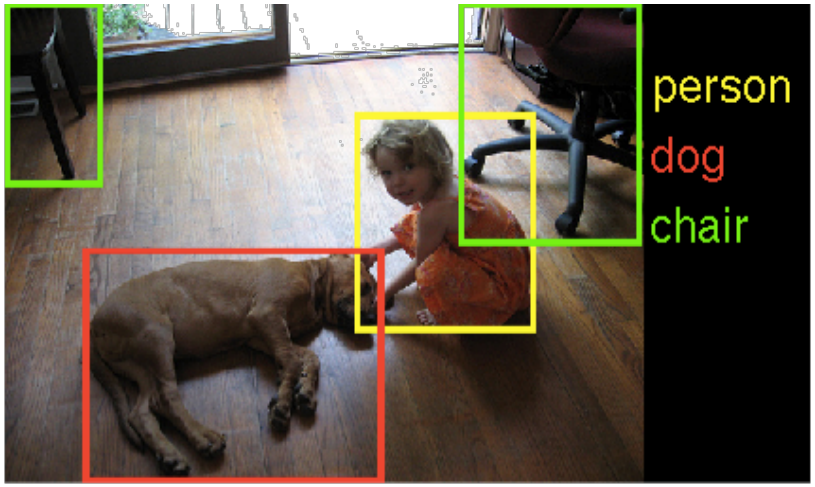


Image source: ericsson.com

Example Applications of Machine Learning

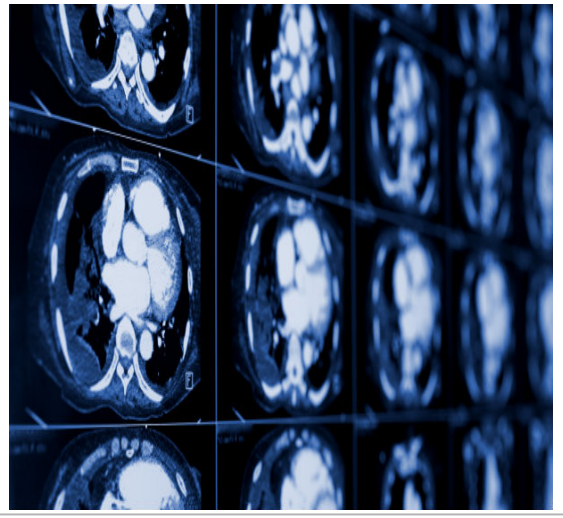
Computer Vision



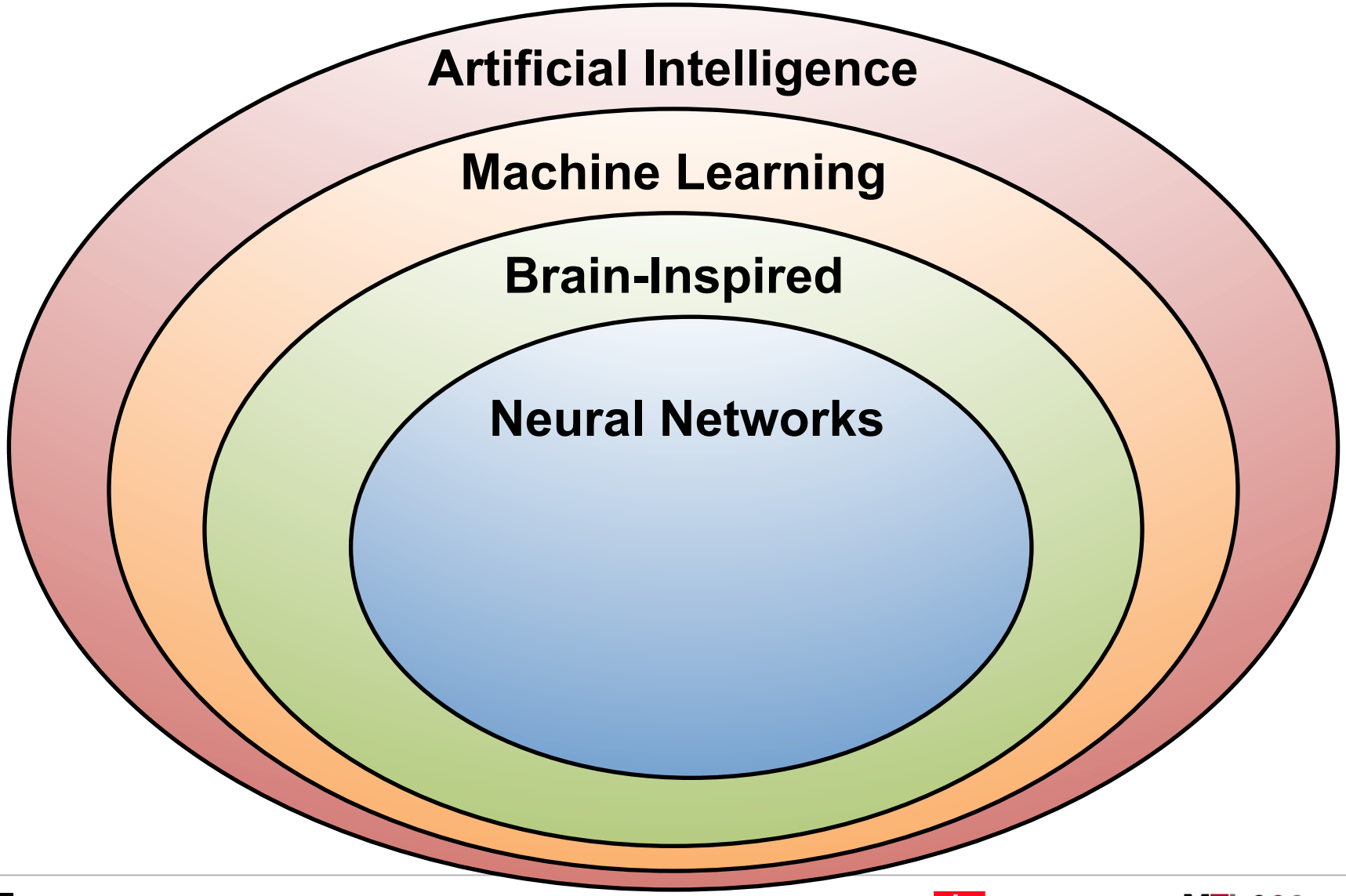
Speech Recognition

Game Play

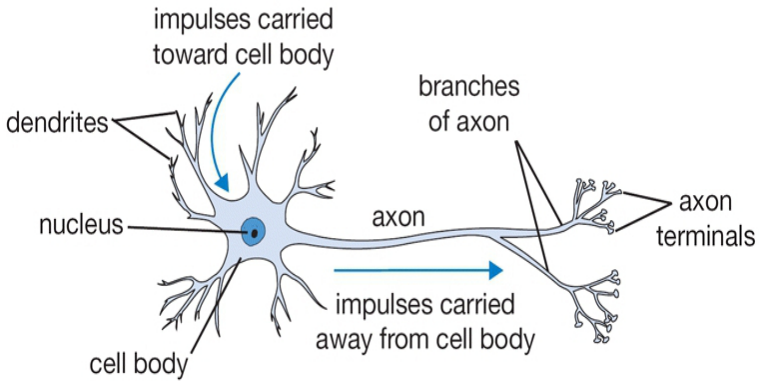
Medical



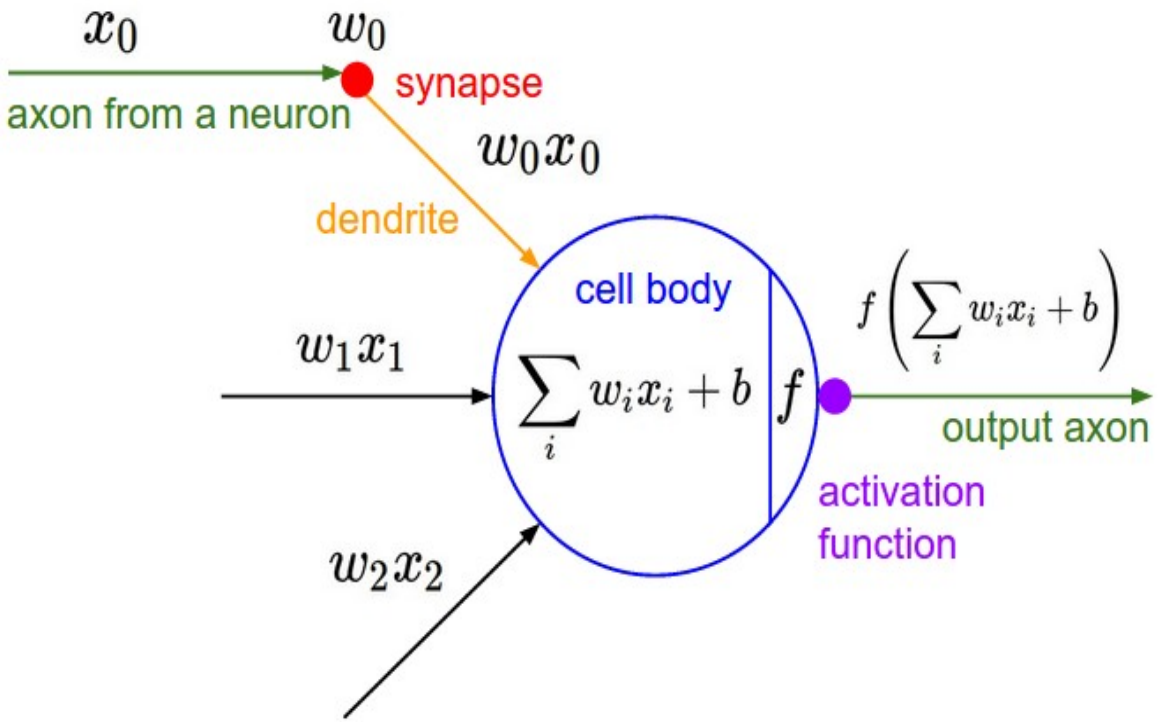
Machine Learning and Neural Networks



Neural Networks: Weighted Sum



The brain contains
~10^{11} neurons connected with
~10^{14} – 10^{15} synapses



Many Weighted Sums

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

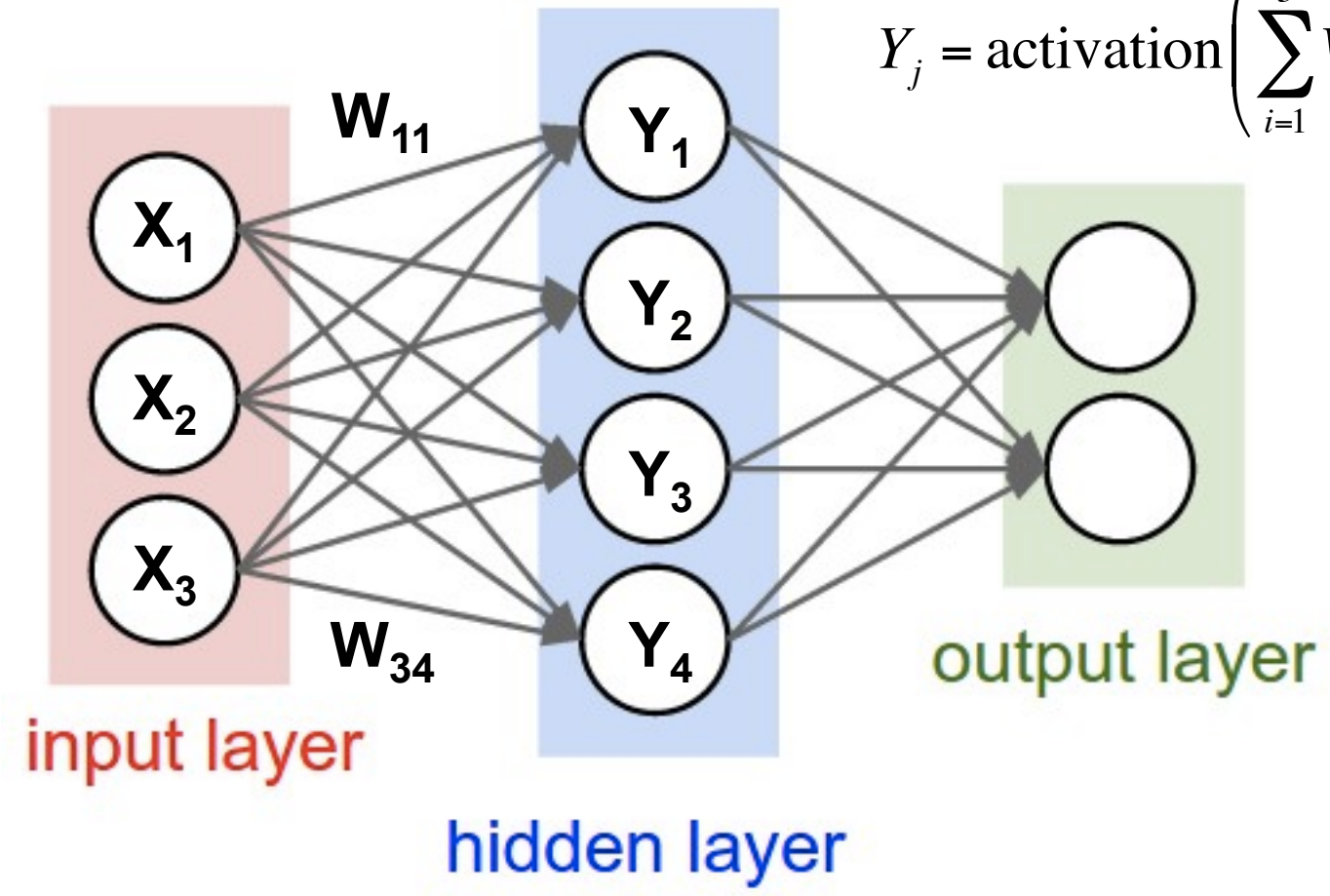
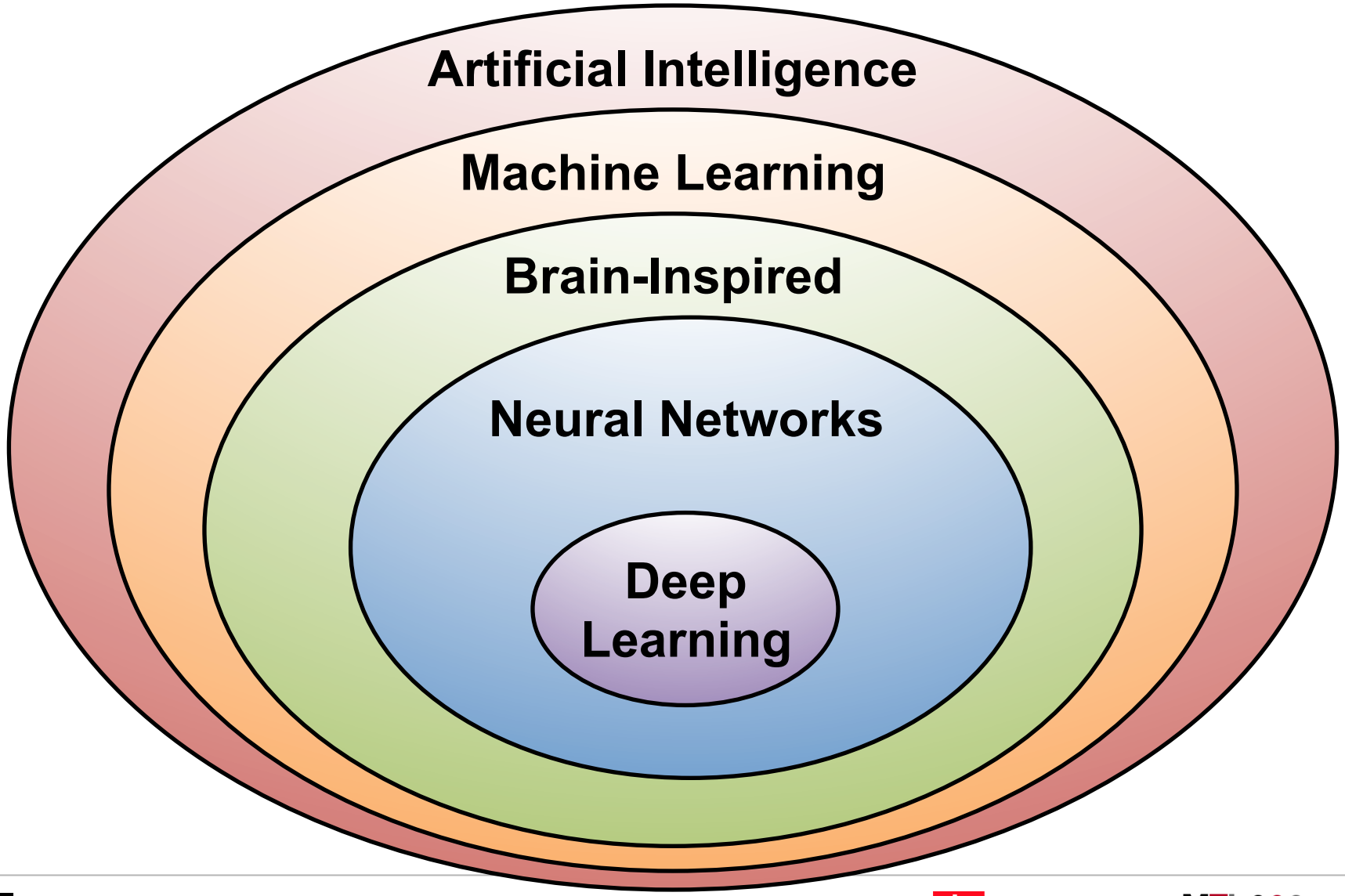


Image Source: Stanford

Deep Learning



What is Deep Learning?

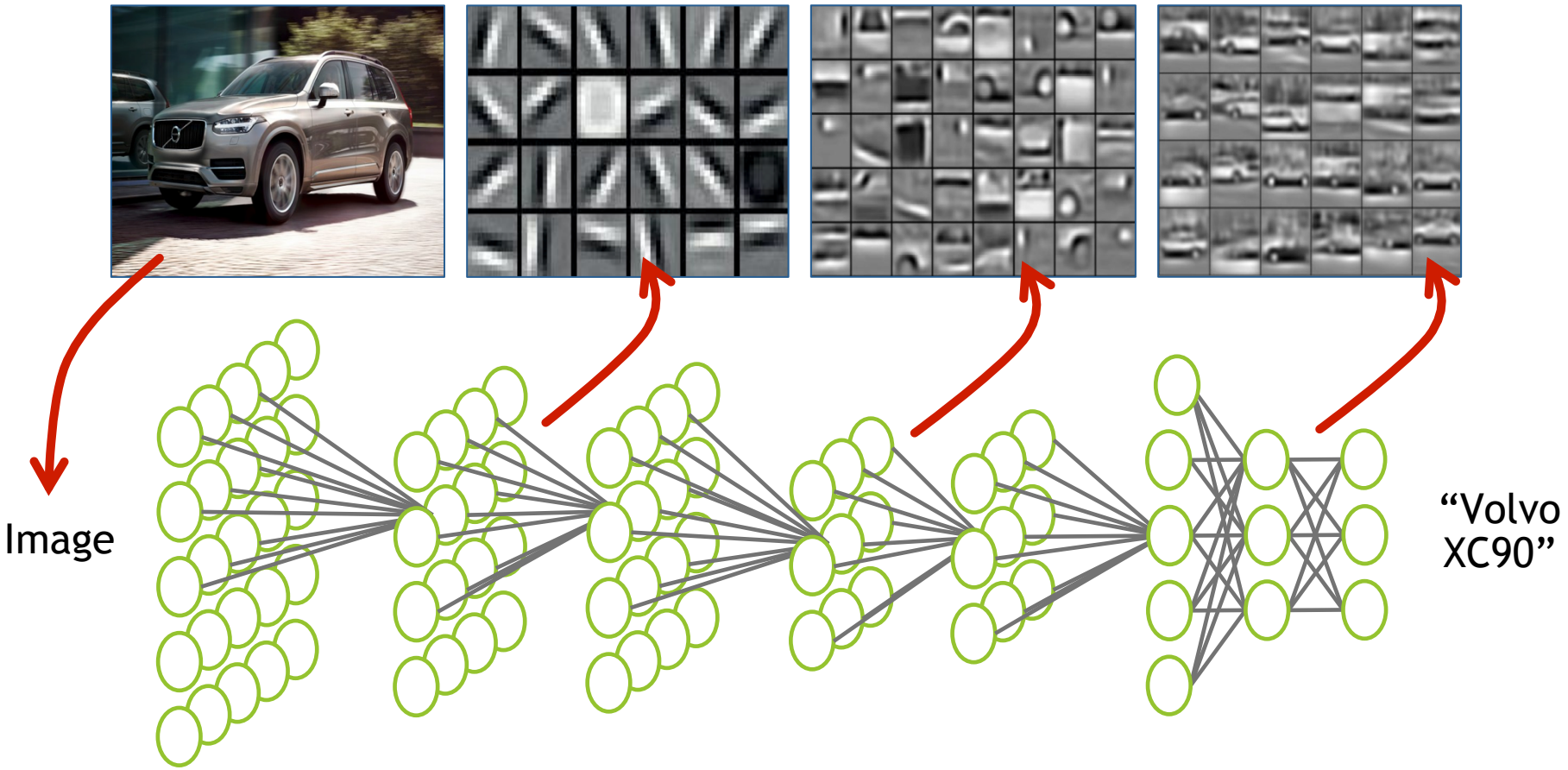


Image Source: [Lee et al., Comm. ACM 2011]

Why is Deep Learning Hot Now?

Big Data Availability

GPU Acceleration

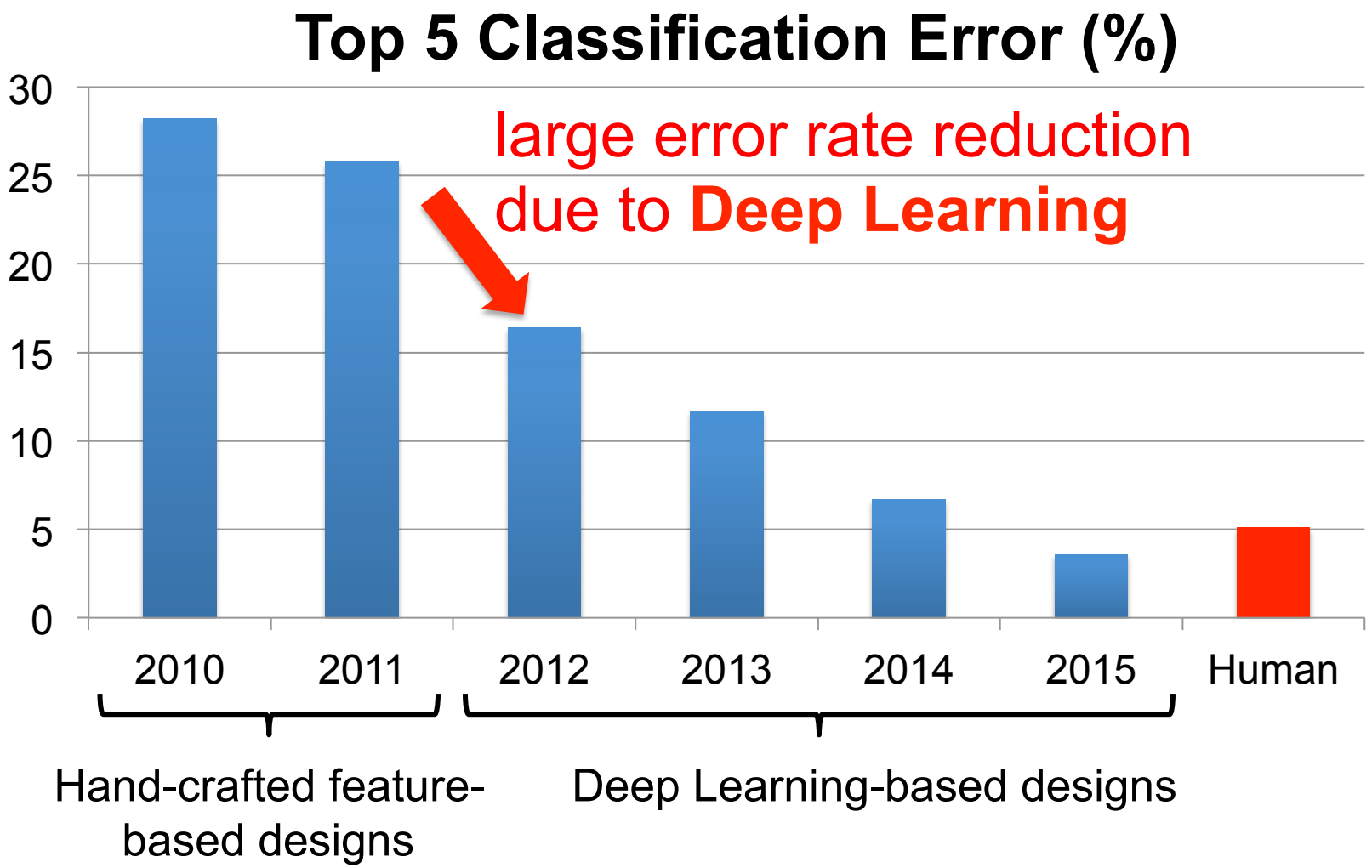
New ML Techniques

facebook **350M** images uploaded per day

Walmart **2.5 Petabytes** of customer data hourly

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

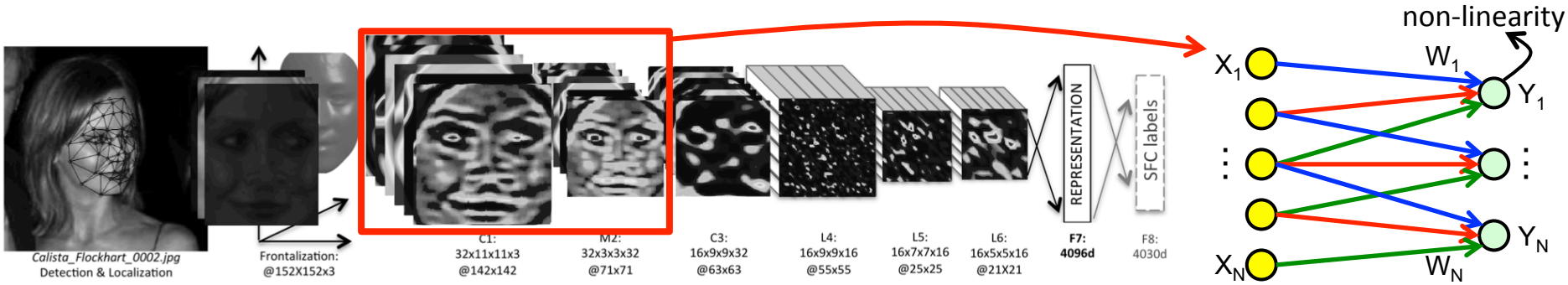
ImageNet: Image Classification Task



[Russakovsky et al., IJCV 2015]

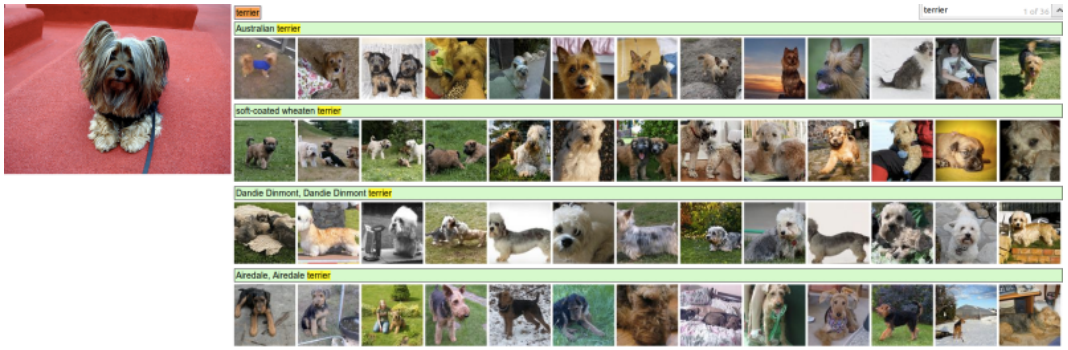
Human or *Superhuman* Accuracy Level

- 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



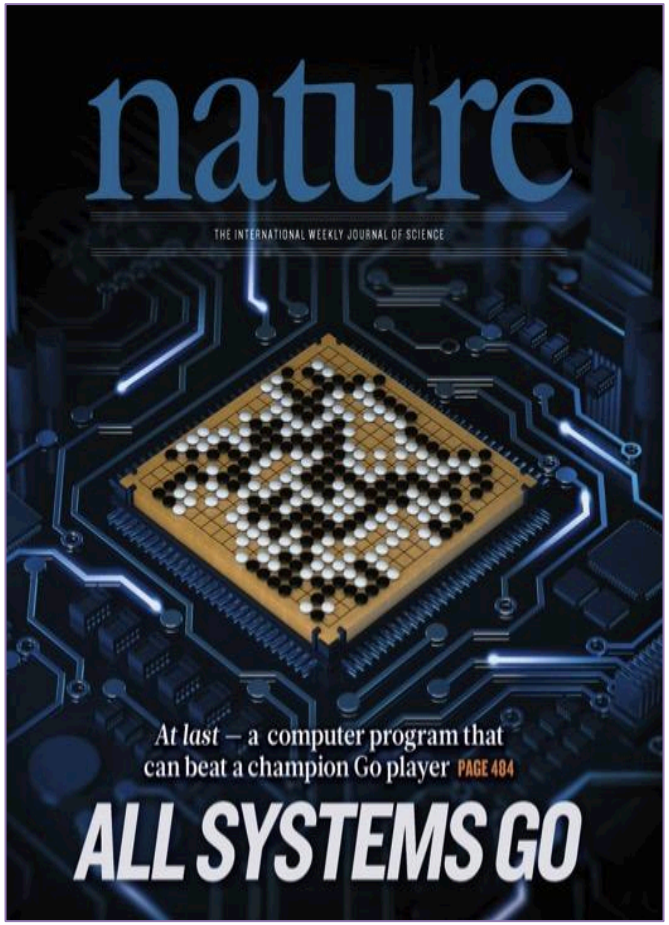
120 species of dogs

[O. Russakovsky et al., IJCV 2015]

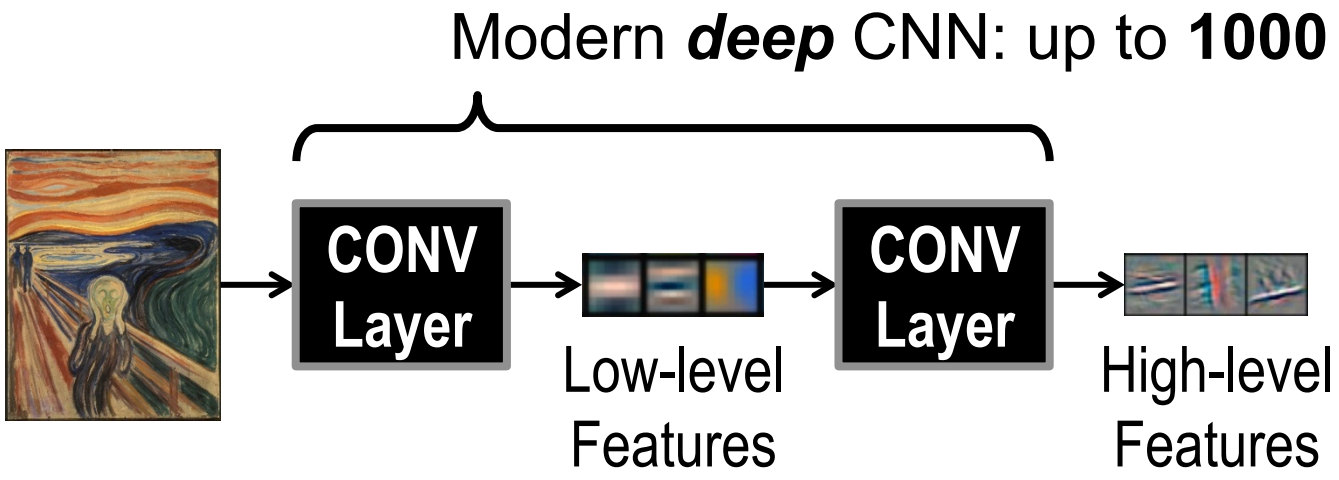
Deep Learning on Games

Google DeepMind AlphaGo

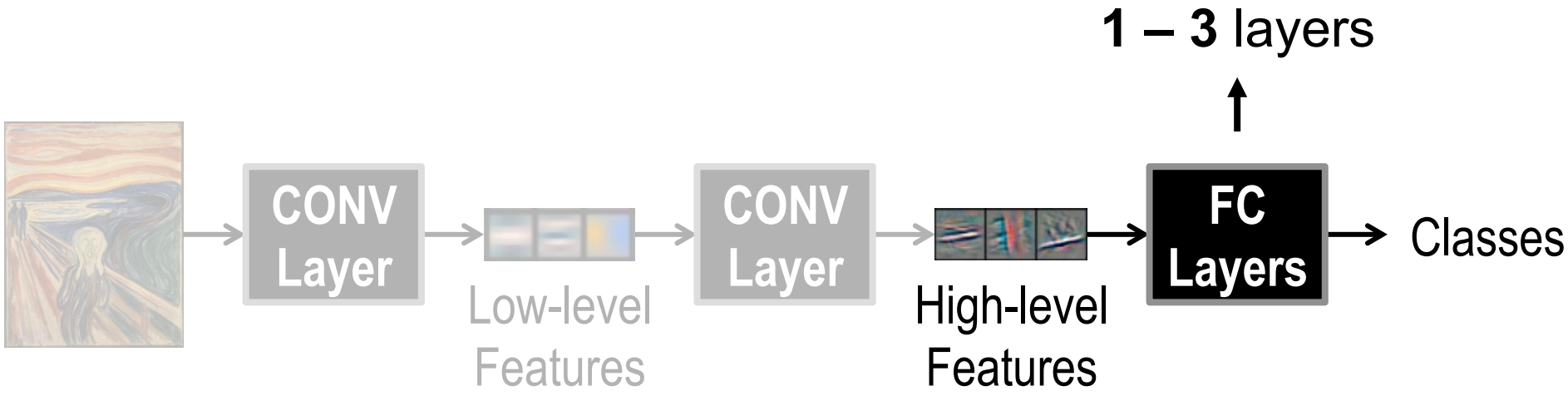
Go is exponentially more complex than chess (10^{170} legal positions)



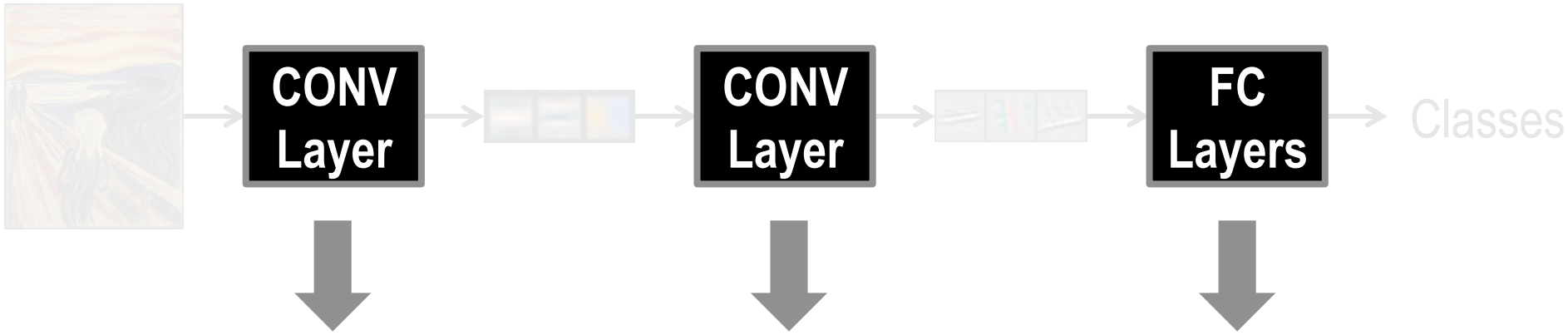
Deep Convolutional Neural Networks

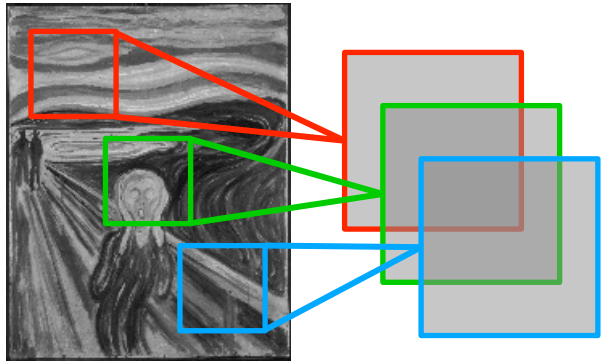


Deep Convolutional Neural Networks



Deep Convolutional Neural Networks



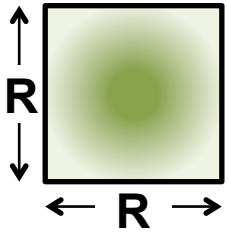


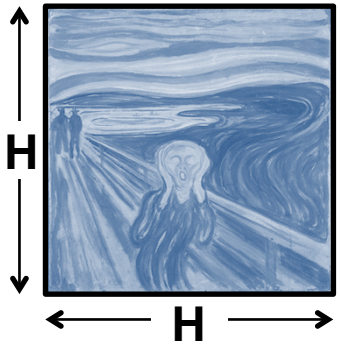
Convolutions account for more than 90% of overall computation, dominating **runtime** and **energy consumption**

High-Dimensional CNN Convolution

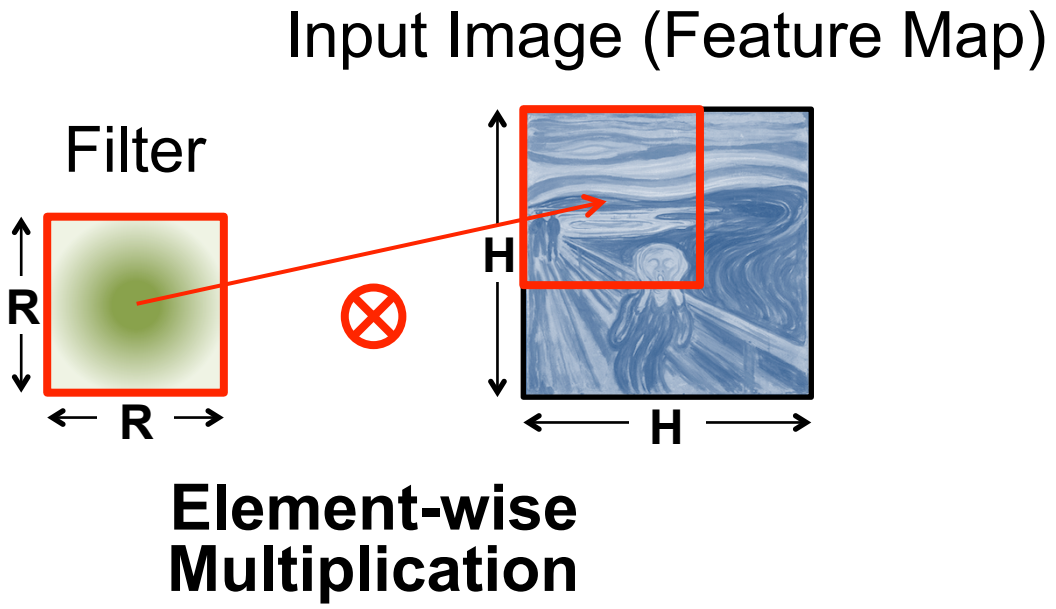
Input Image (Feature Map)

Filter

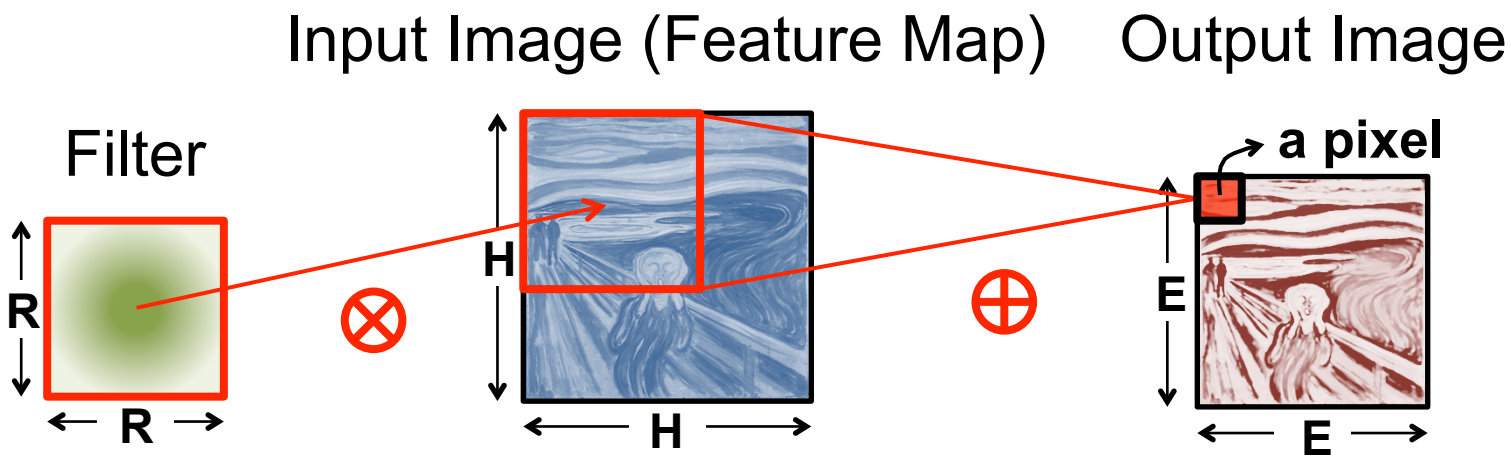




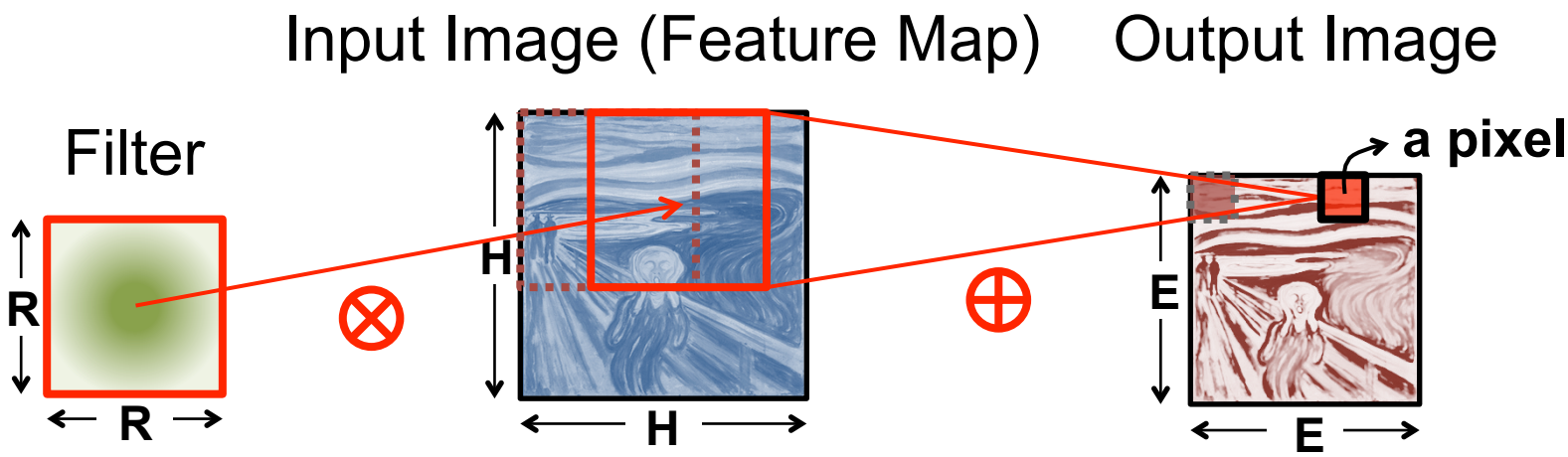
High-Dimensional CNN Convolution



High-Dimensional CNN Convolution

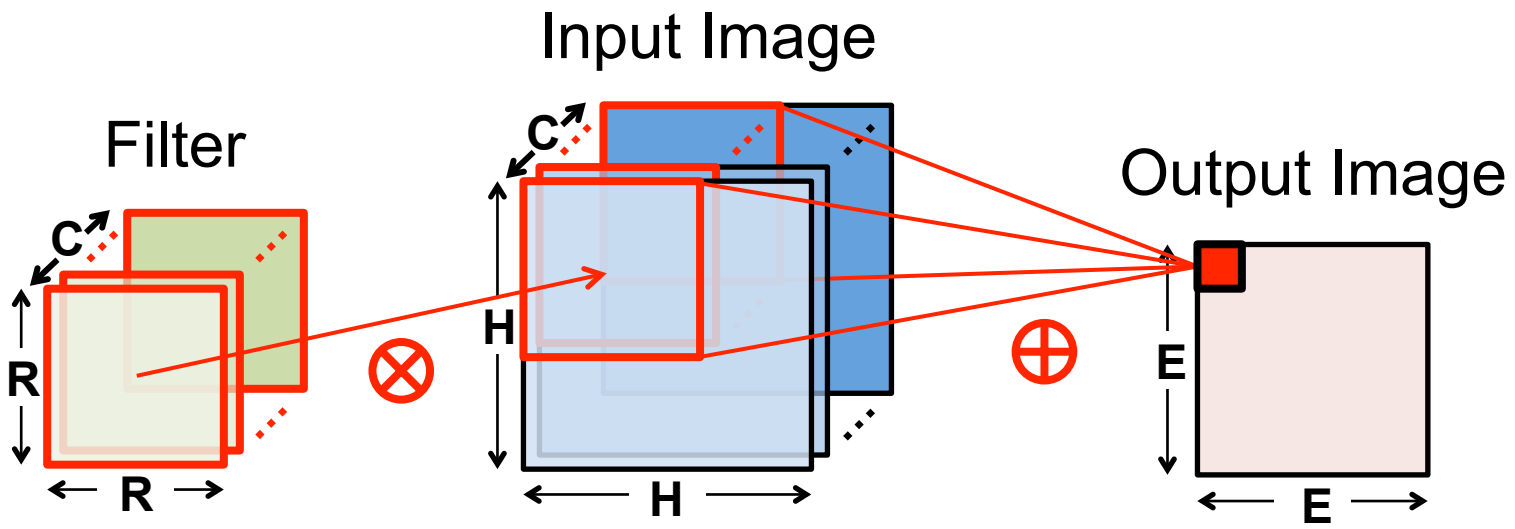


High-Dimensional CNN Convolution



Sliding Window Processing

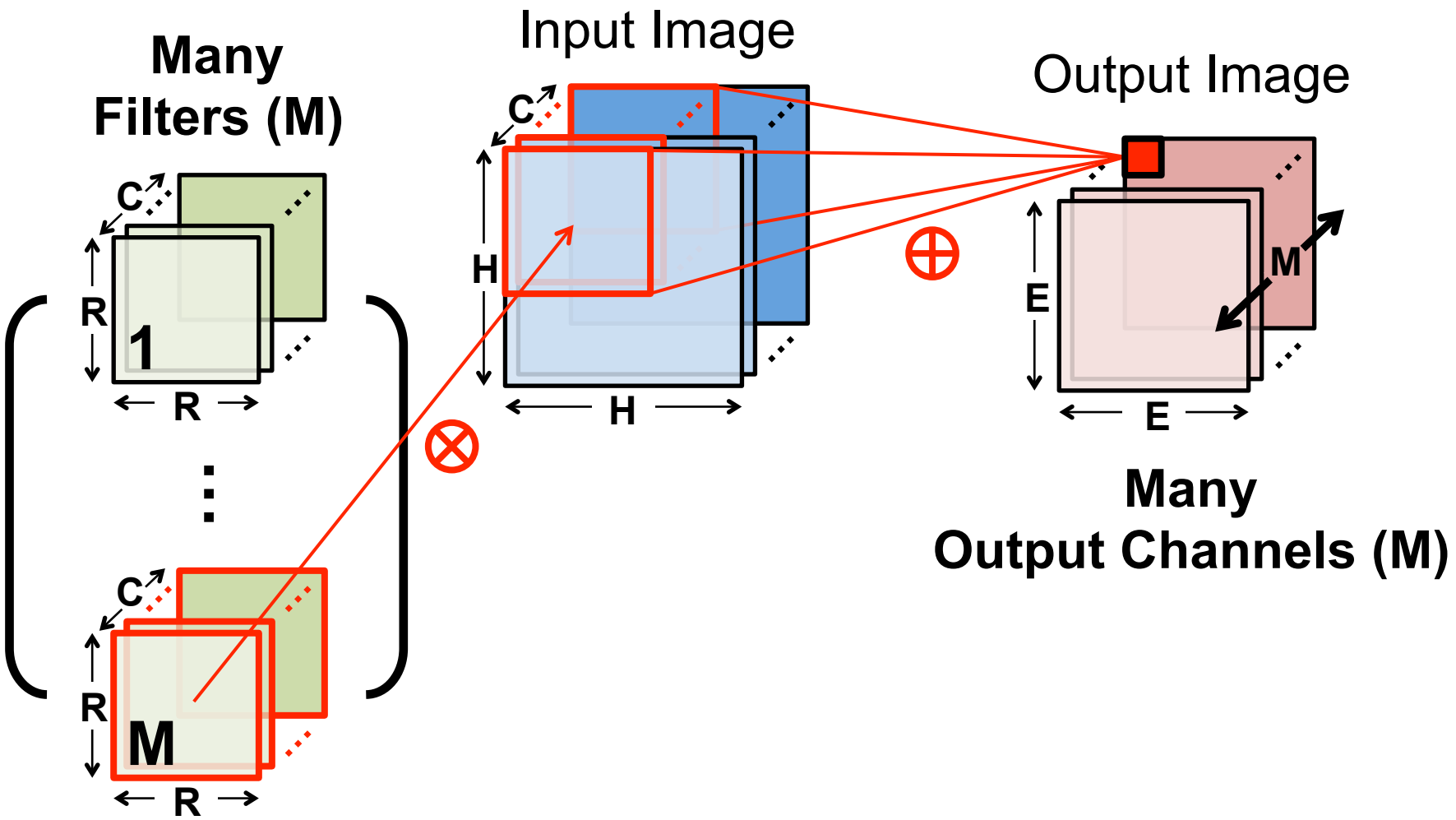
High-Dimensional CNN Convolution



Many Input Channels (C)

AlexNet: 3 – 192 Channels (C)

High-Dimensional CNN Convolution



AlexNet: 96 – 384 Filters (M)

High-Dimensional CNN Convolution

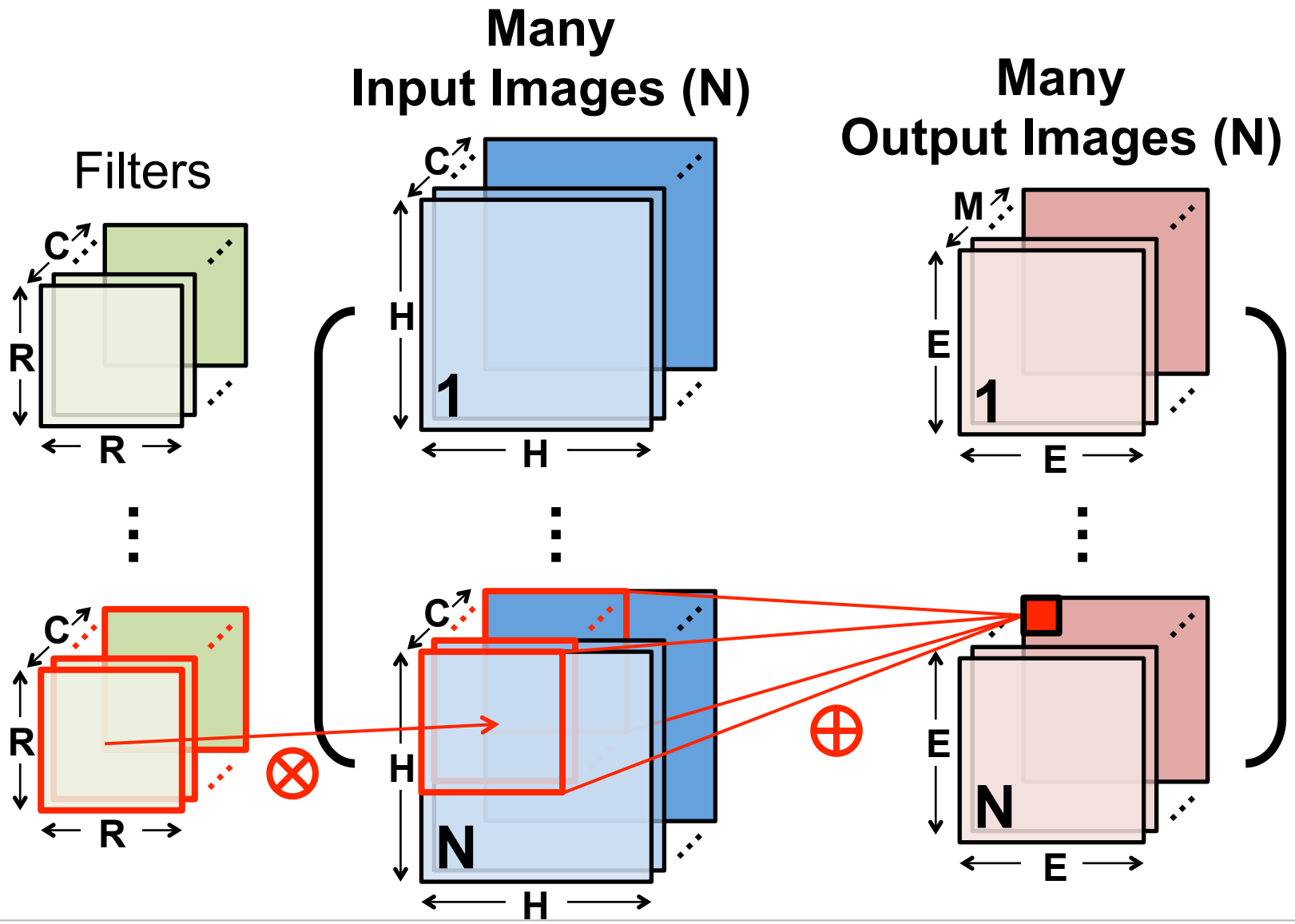


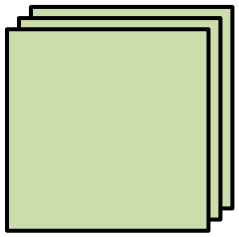
Image batch size: 1 – 256 (N)

Large Sizes with Varying Shapes

AlexNet[1] Convolutional Layer Configurations

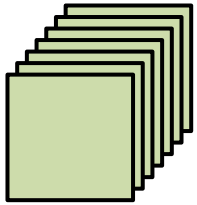
Layer	Filter Size (R)	# Filters (M)	# Channels (C)	Stride
1	11x11	96	3	4
2	5x5	256	48	1
3	3x3	384	256	1
4	3x3	384	192	1
5	3x3	256	192	1

Layer 1



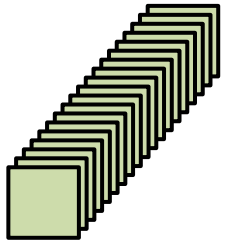
34k Params
105M MACs

Layer 2



307k Params
224M MACs

Layer 3

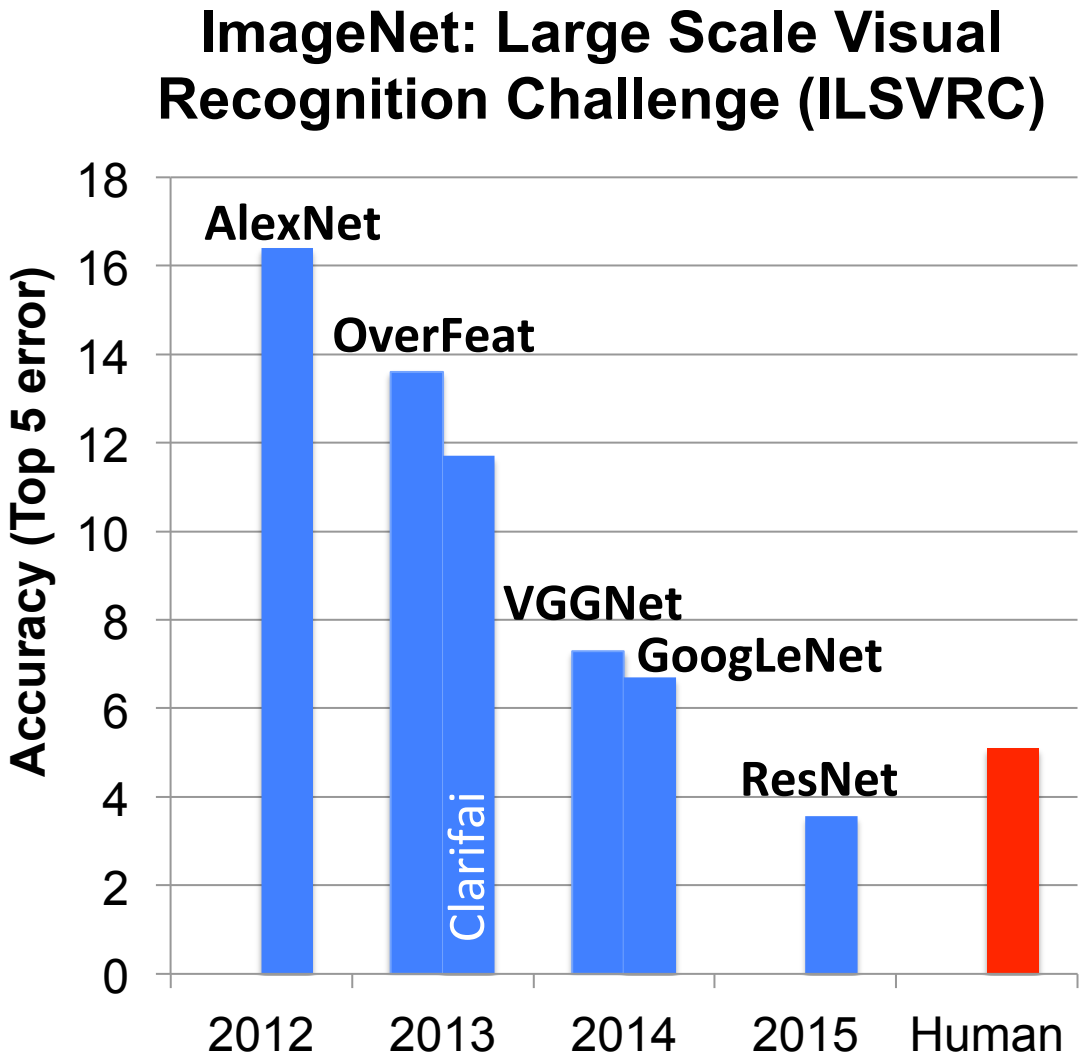


885k Params
150M MACs

1. [Krizhevsky, NIPS 2012]

Popular 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**

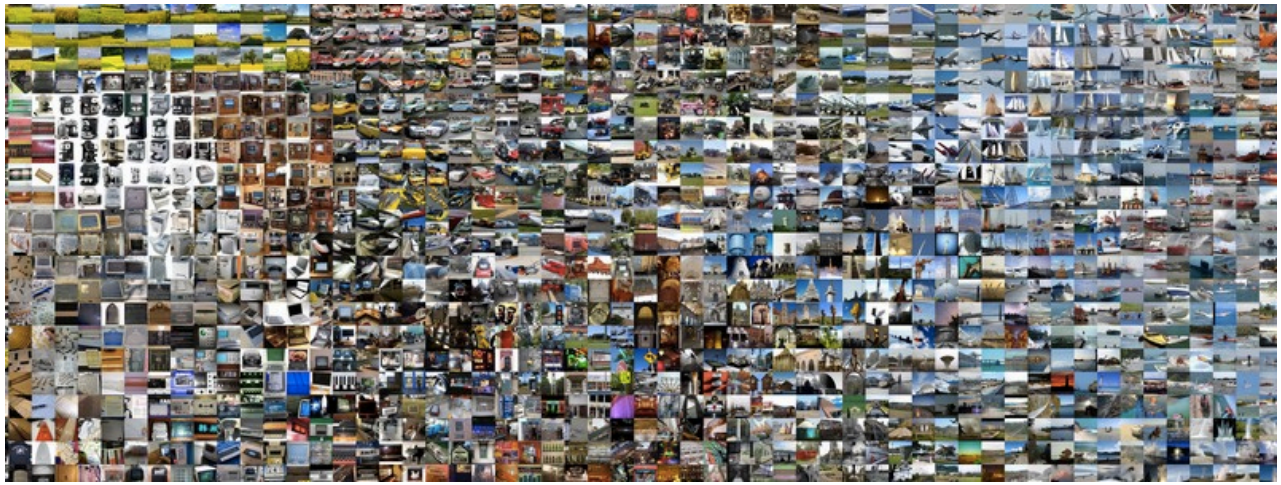
CONV Layers increasingly important!

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)

Weights

Inference
(use weights)

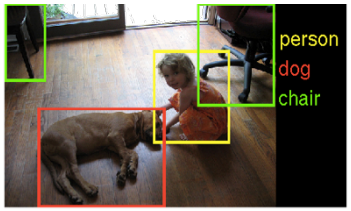
Challenges

Key Metrics

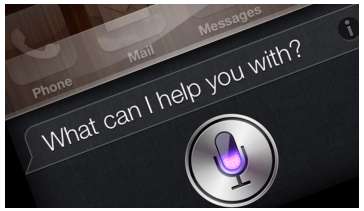
- **Accuracy**
  - Measured on a publicly available dataset
  - Popular DNN Models
- **Programmability**
  - Support multiple applications
  - Different weights
- **Energy/Power**
  - Energy per operation
  - DRAM Bandwidth
- **Throughput/Latency**
  - GOPS, frame rate, delay
- **Cost**
  - Area (memory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

ImageNet

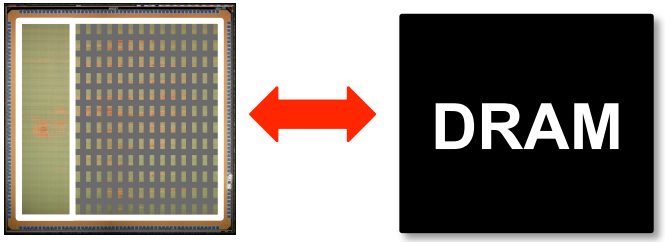
Computer Vision



Speech Recognition

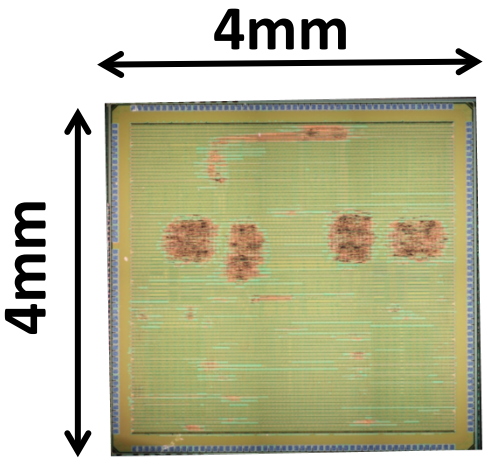


Chip

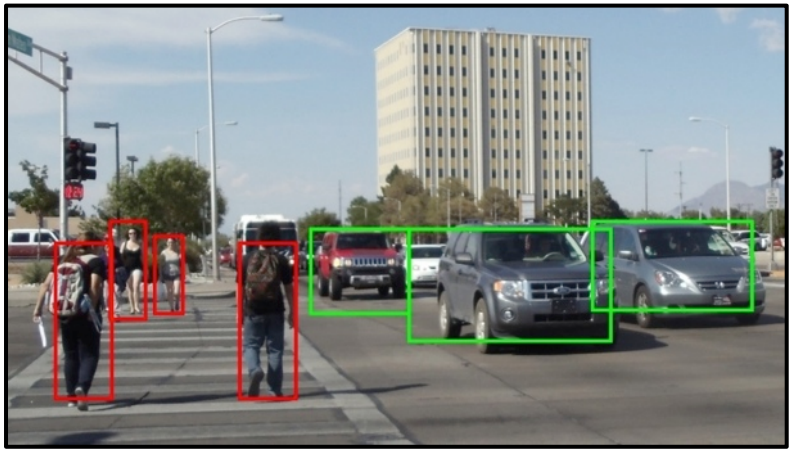


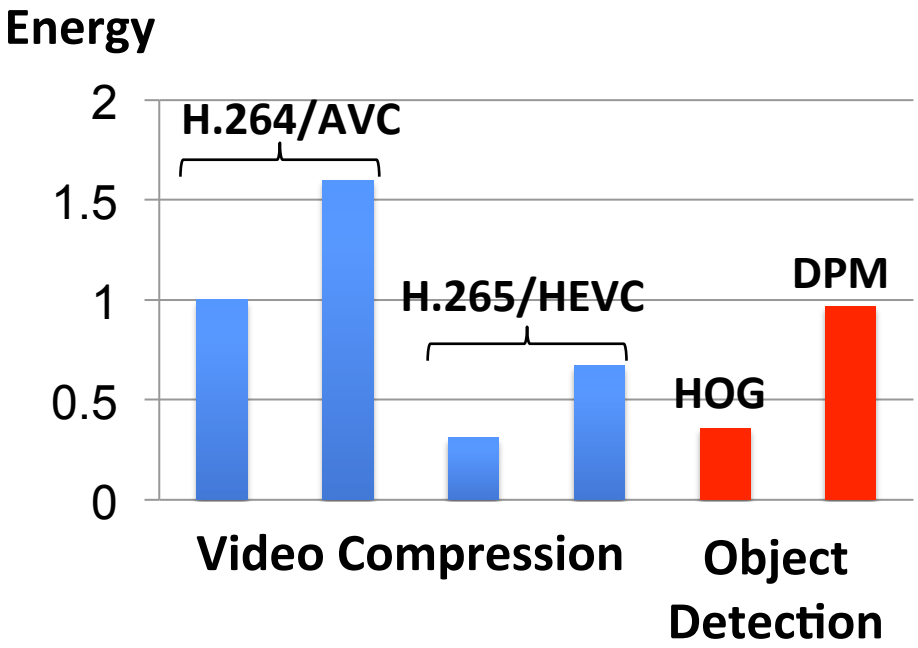
Typical Constraints on Video Coding

- **Area cost**
  - **Memory Size 100-500kB**
- **Power budget**
  - **< 1W for smartphones**
- **Throughput**
  - **Real-time 30 fps**
- **Energy**
  - **~1nJ/pixel**



MIT Object Detection Chip [VLSI 2016]





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)

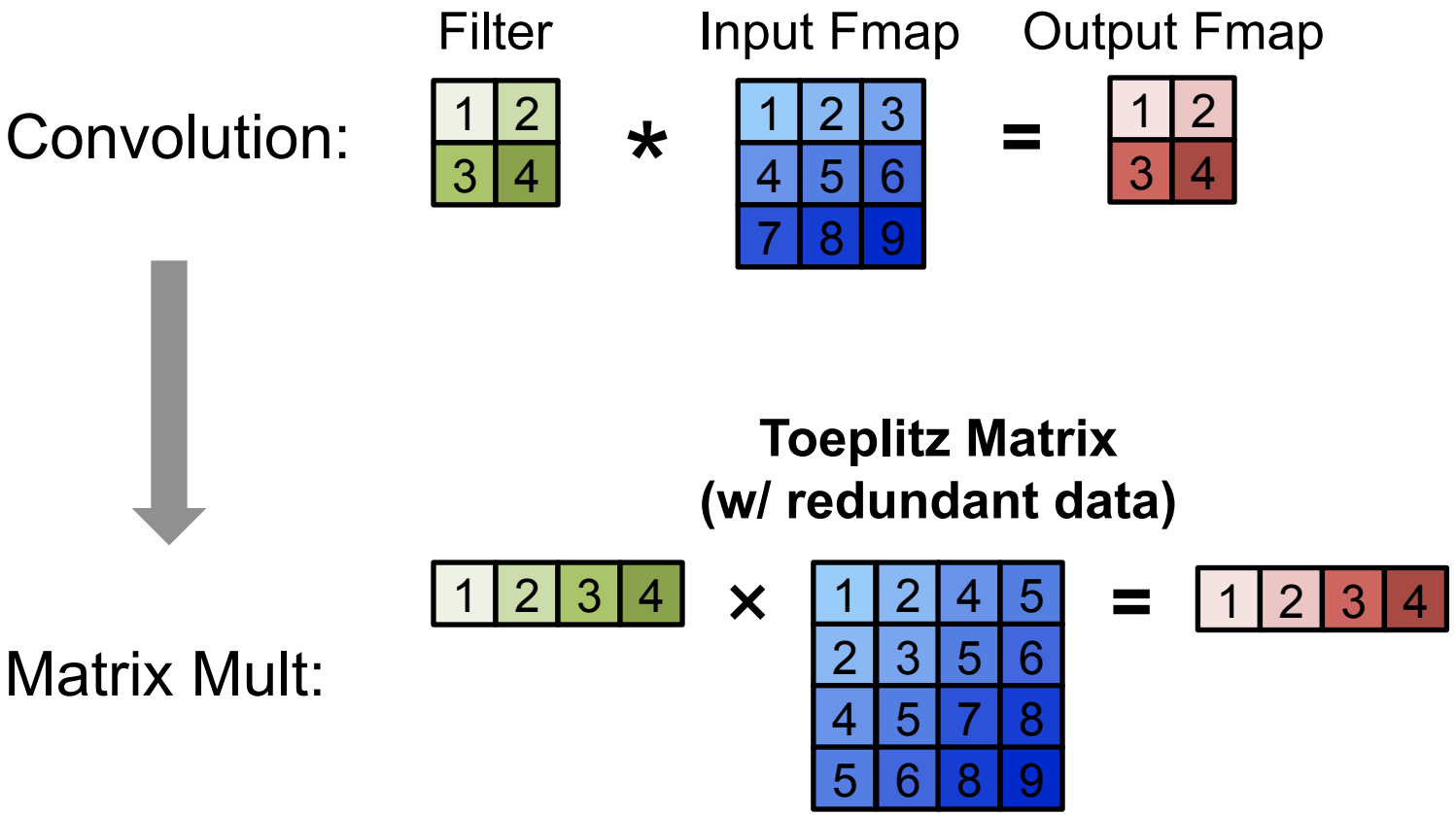
Use **matrix multiplication libraries** on CPUs and GPUs

Accelerate Matrix Multiplication

- Implementation: **Matrix Multiplication (GEMM)**
  - **CPU:** OpenBLAS, Intel MKL, etc
  - **GPU:** cuBLAS, cuDNN, etc
- Optimized by tiling to storage hierarchy

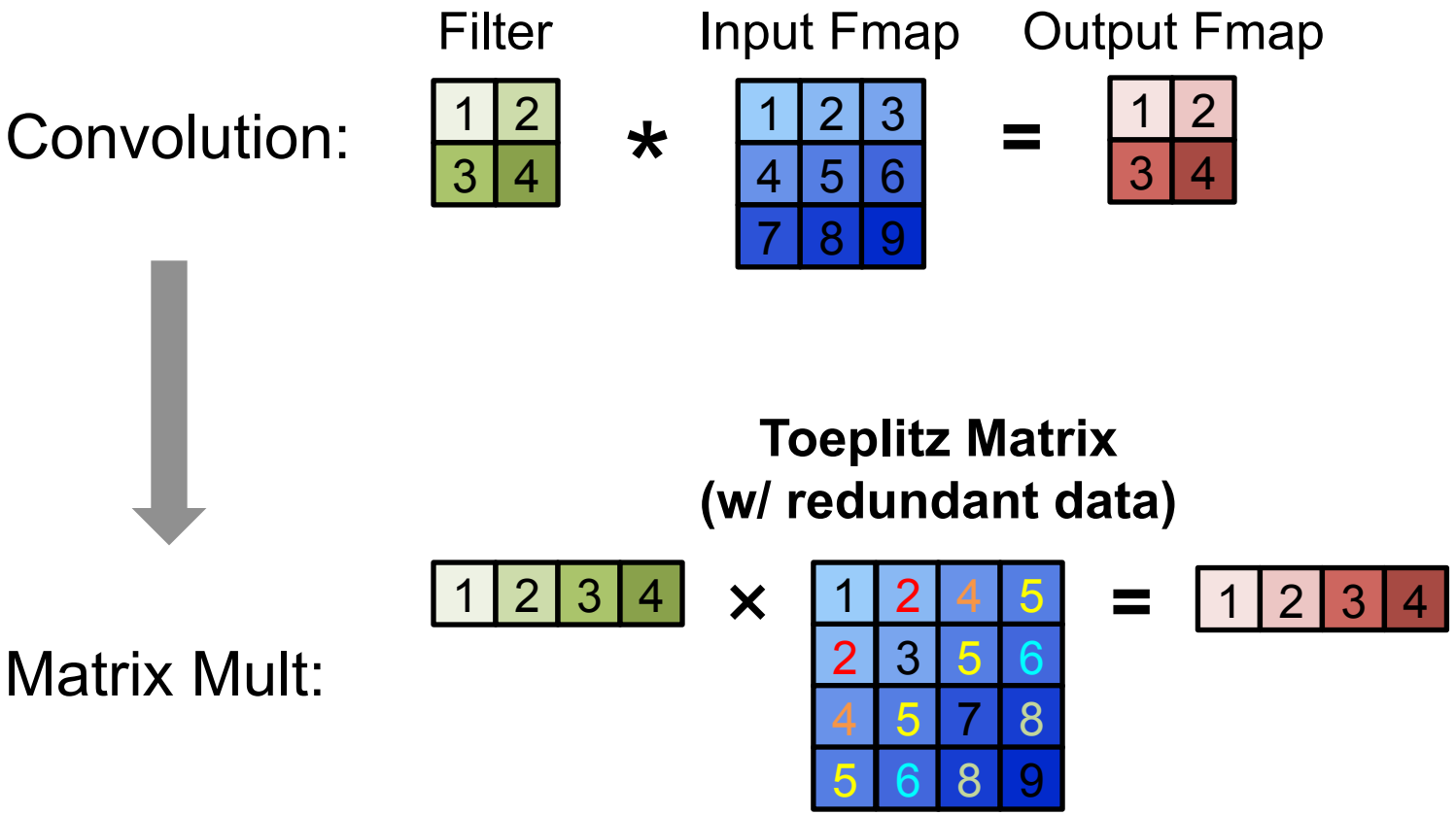
Map DNN to a Matrix Multiplication

- Convert to matrix mult. using the **Toeplitz Matrix**



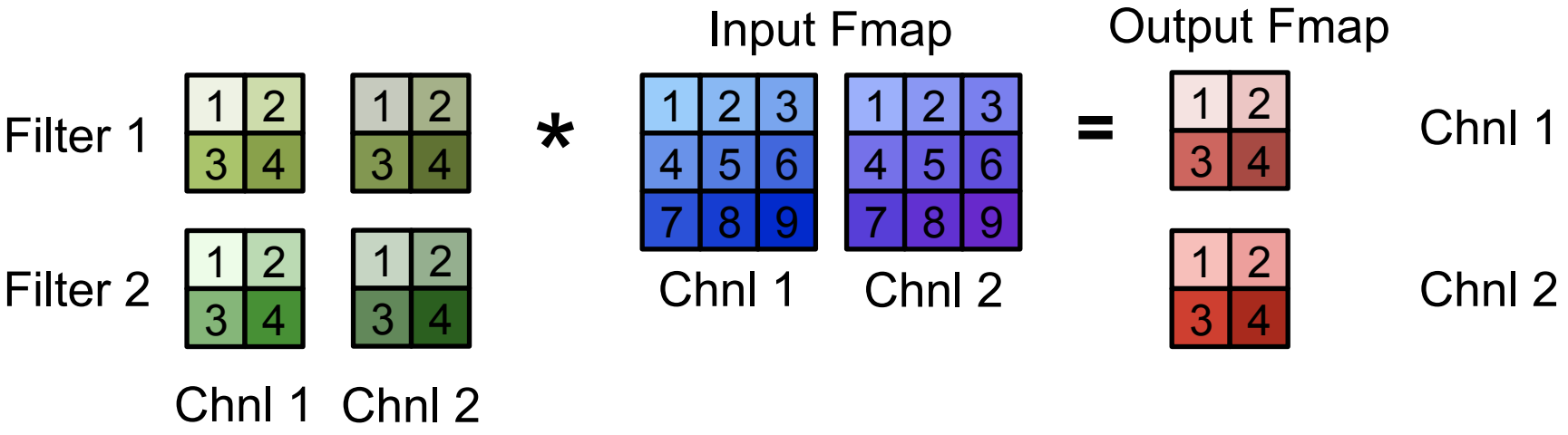
Map DNN to a Matrix Multiplication

- Convert to matrix mult. using the **Toeplitz Matrix**



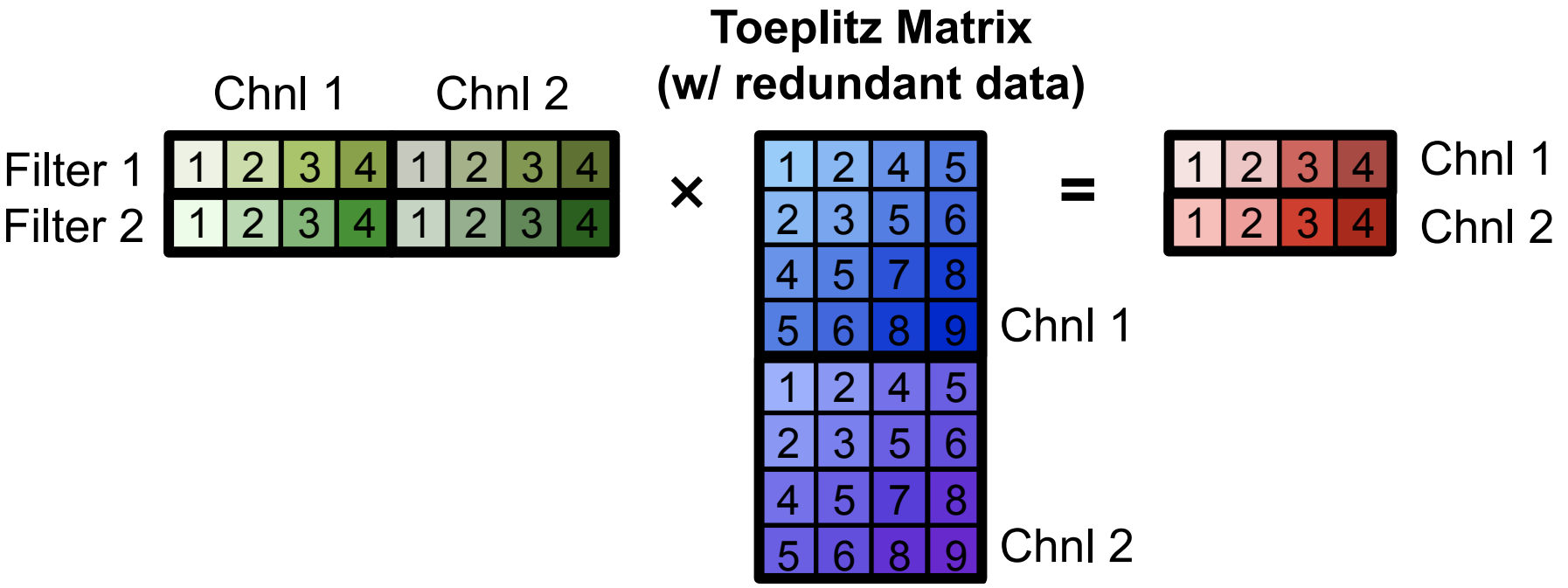
Map DNN to a Matrix Multiplication

- Multiple Channels and Filters



Map DNN to a Matrix Multiplication

- Multiple Channels and Filters



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

Computation Transformations

- **Goal: Bitwise same result, but reduce number of operations**
- **Focuses mostly on compute**

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

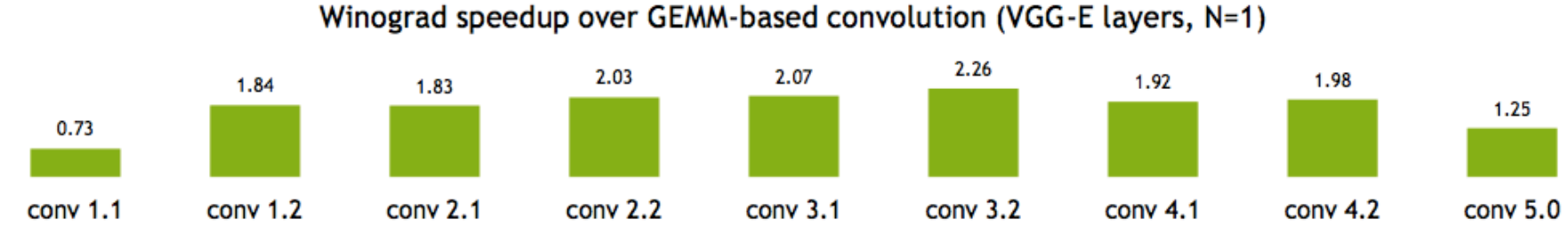
Reduce number of multiplications, but **increase** number of additions

Reduce Operations in Matrix Multiplication

- **Winograd** [Lavin, CVPR 2016]
  - **Pro:** 2.25x speed up for 3x3 filter
  - **Con:** Specialized processing depending on filter size
- **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 Performance Varies

Optimal convolution algorithm depends on convolution layer dimensions



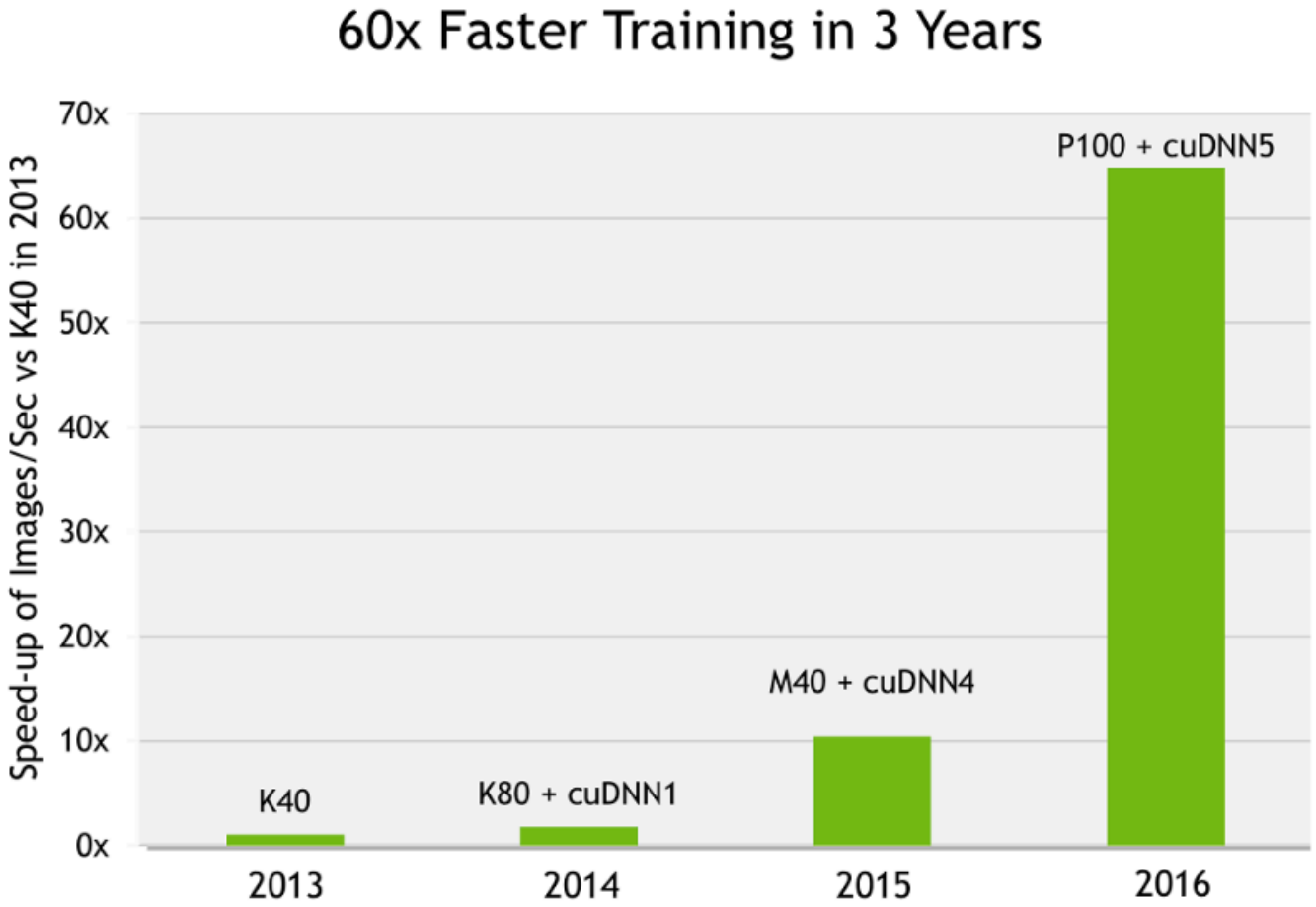
Meta-parameters (data layouts, texture memory) afford higher performance

Using texture memory for convolutions: 13% inference speedup

(GoogLeNet, batch size 1)

Source: Nvidia

cuDNN: Speed up with Transformations



Source: Nvidia

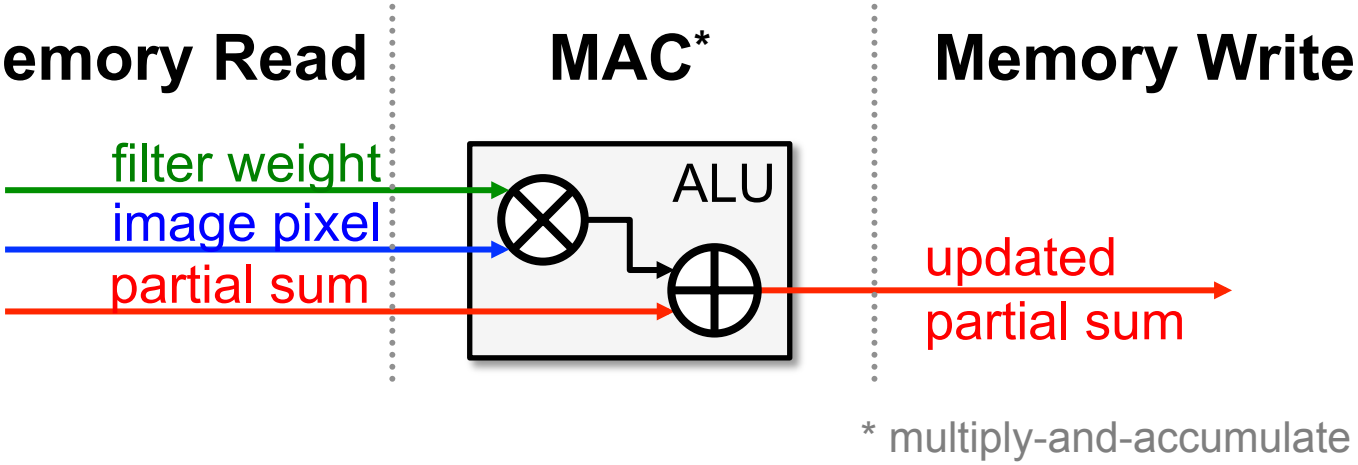
Specialized Hardware (Accelerators)

Properties We Can Leverage

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

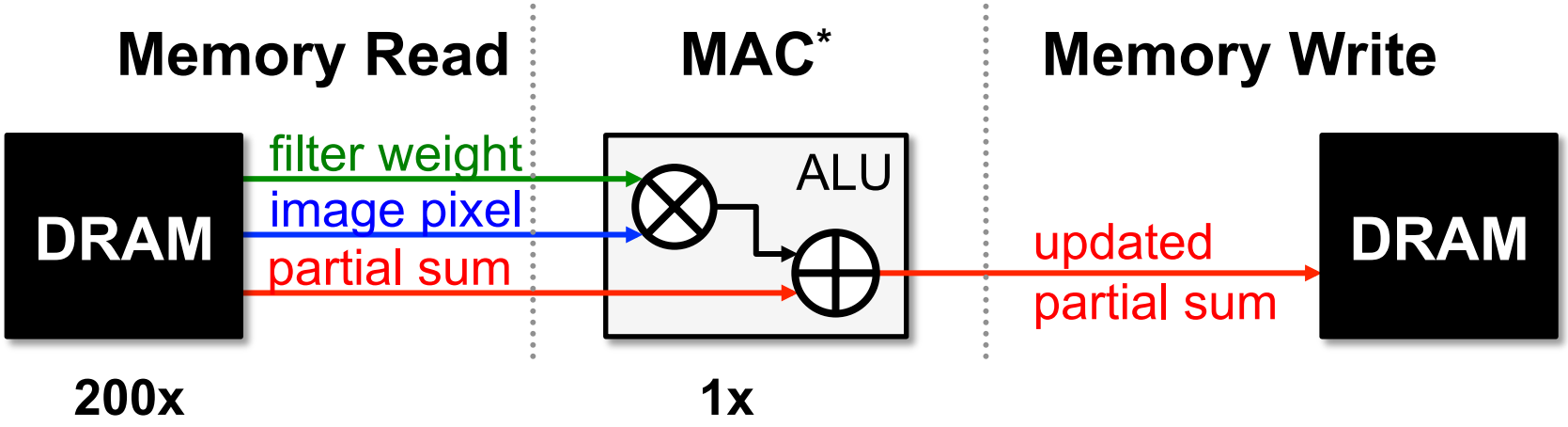
Properties We Can Leverage

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



Properties We Can Leverage

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

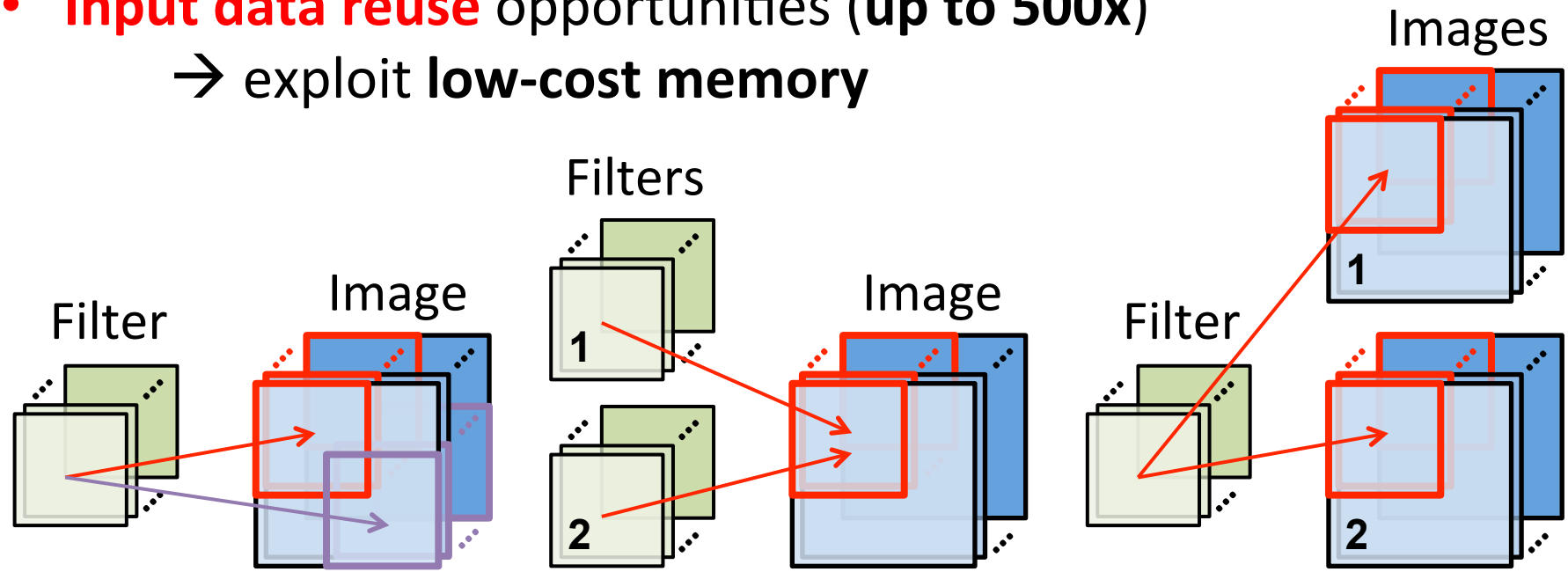


Worst Case: all memory R/W are **DRAM** accesses

- 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**



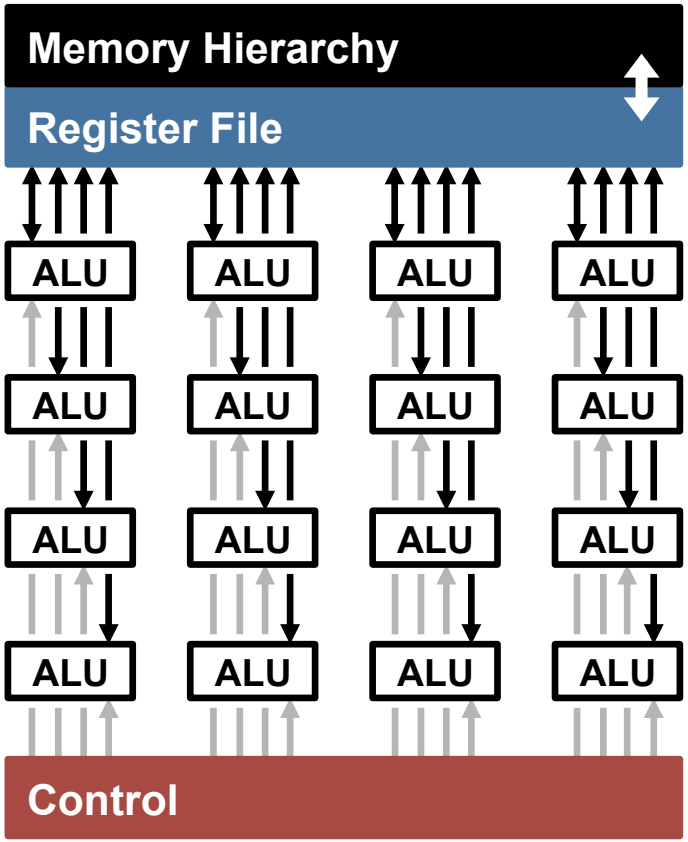
Convolutional Reuse (pixels, weights)

Image Reuse (pixels)

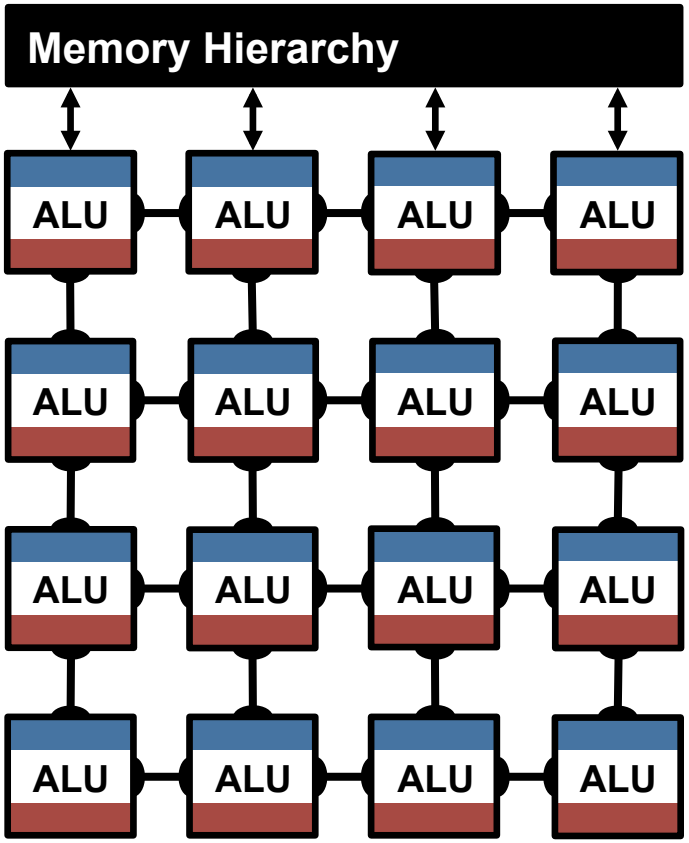
Filter Reuse (weights)

Highly-Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)



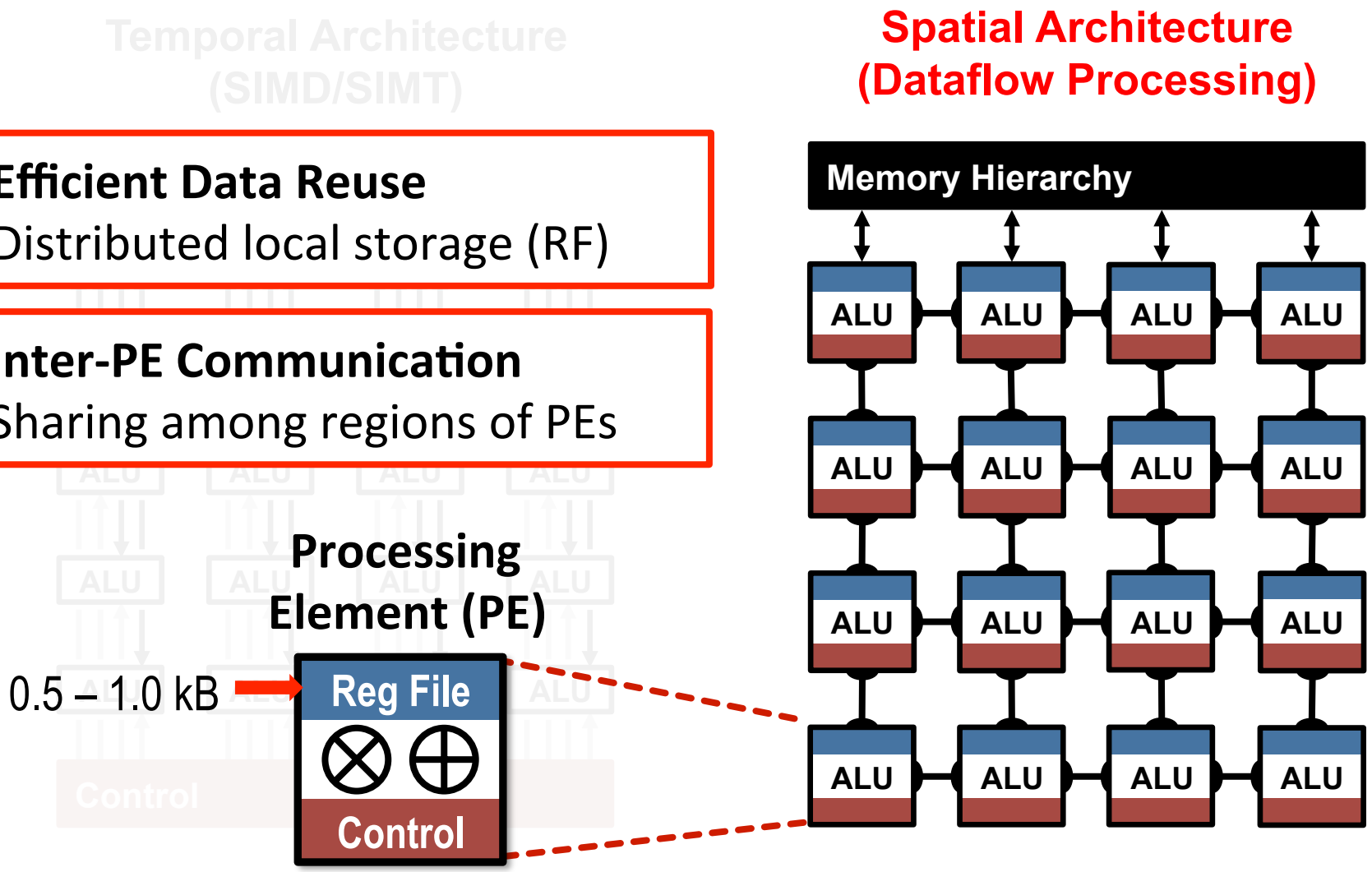
Spatial Architecture (Dataflow Processing)



Advantages of Spatial Architecture

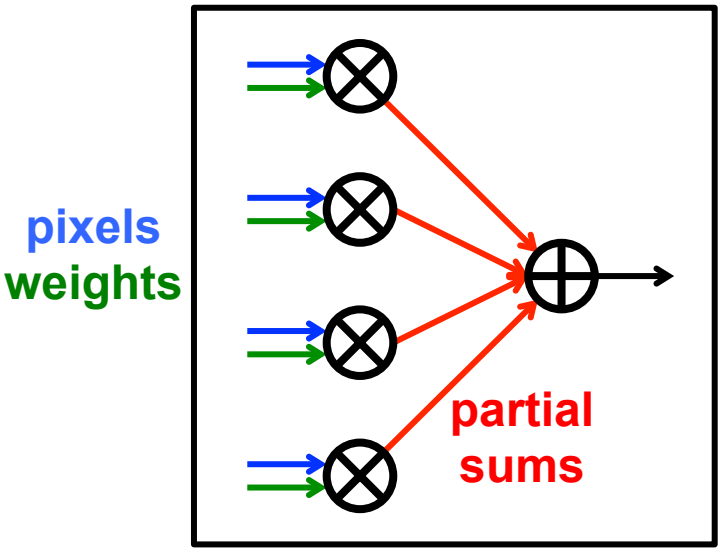
Efficient Data Reuse
Distributed local storage (RF)

Inter-PE Communication
Sharing among regions of PEs



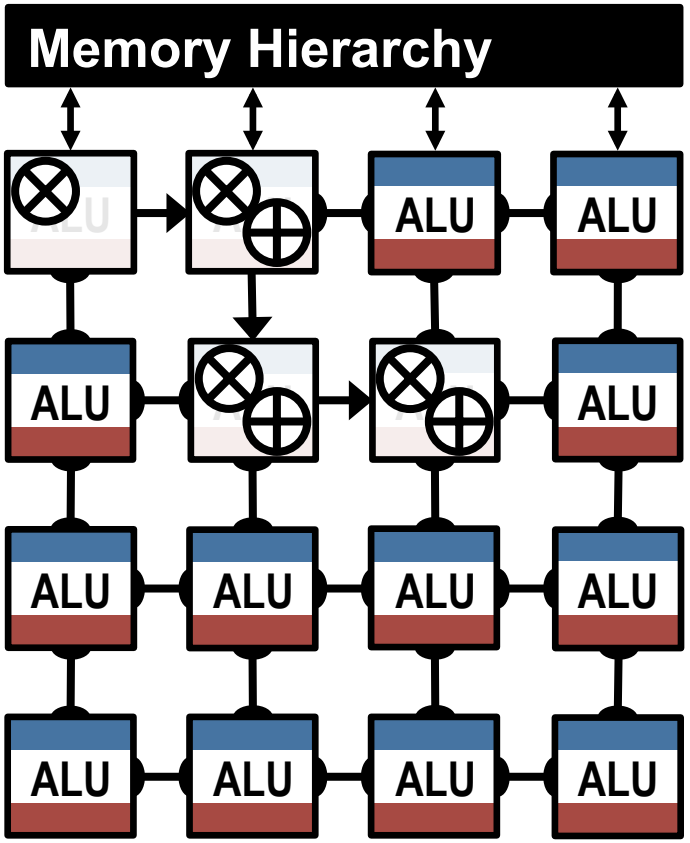
How to Map the Dataflow?

CNN Convolution



?

Spatial Architecture (Dataflow Processing)



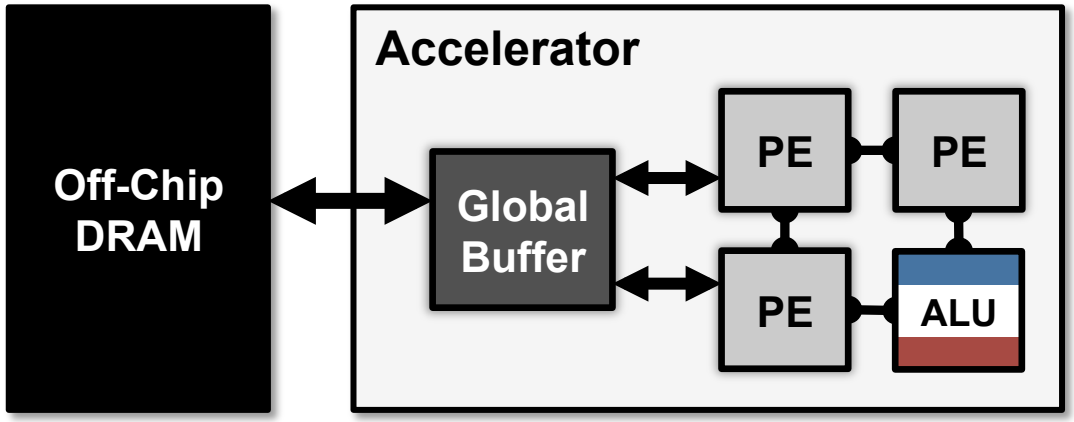
Goal: Increase reuse of input data (**weights** and **pixels**) and local **partial sums** accumulation

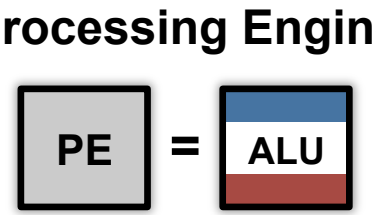
Energy-Efficient Dataflow

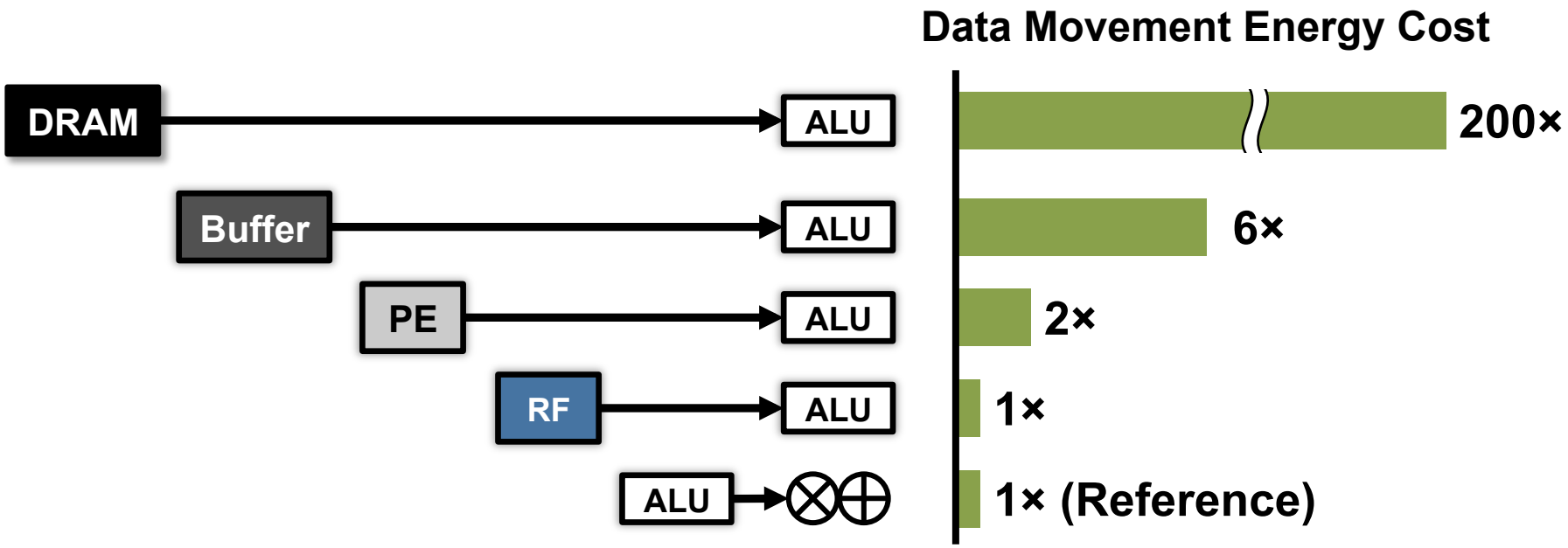
Yu-Hsin Chen, Joel Emer, Vivienne Sze, ISCA 2016

Maximize data reuse and accumulation at RF

Data Movement is Expensive

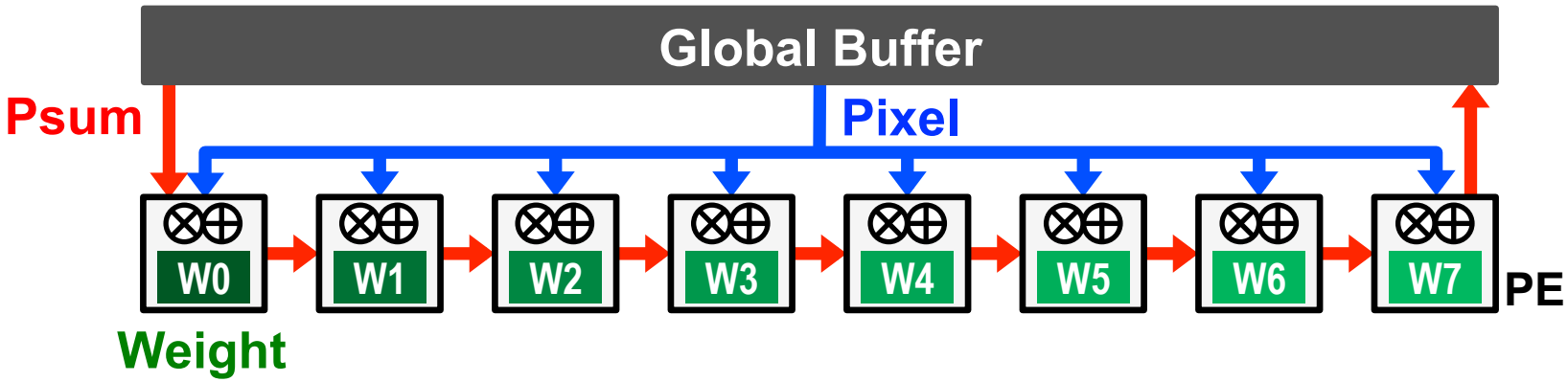






Maximize data reuse at lower levels of hierarchy

Weight Stationary (WS)

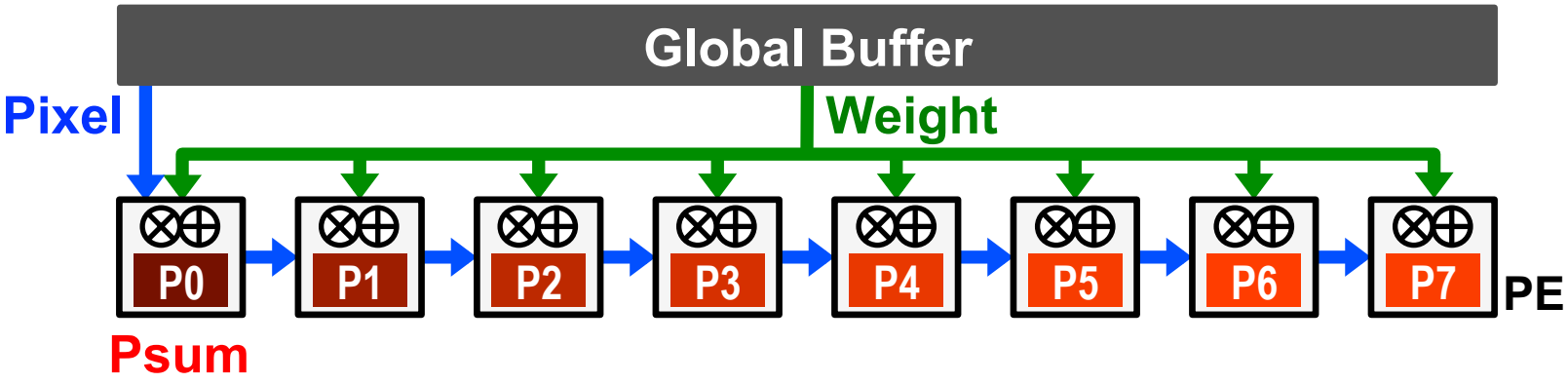


- **Minimize weight** read energy consumption
  - maximize convolutional and filter reuse of weights
- **Examples:**

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

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

Output Stationary (OS)



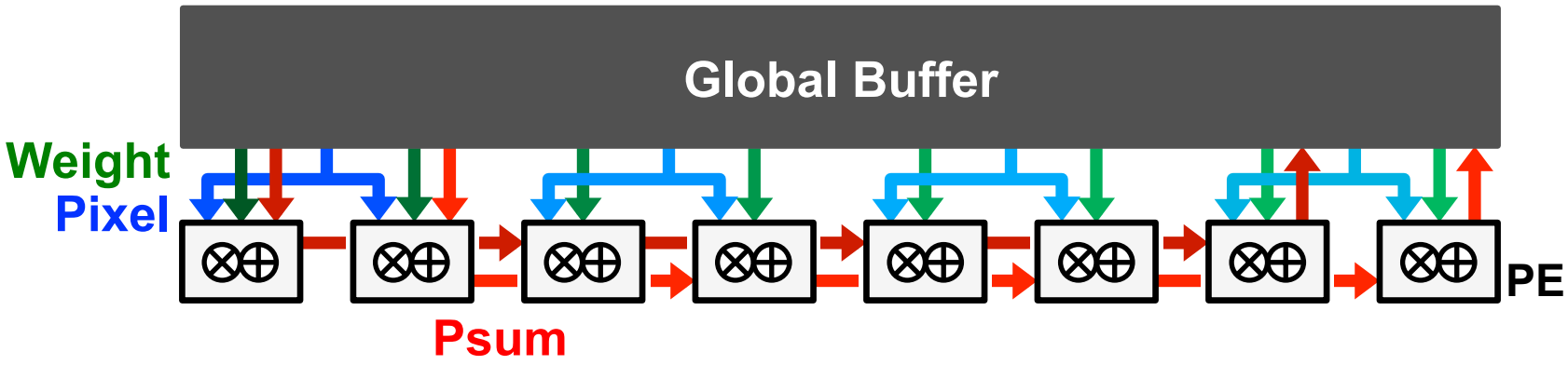
- **Minimize partial sum** R/W energy consumption
  - maximize local accumulation
- **Examples:**

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

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

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

No Local Reuse (NLR)

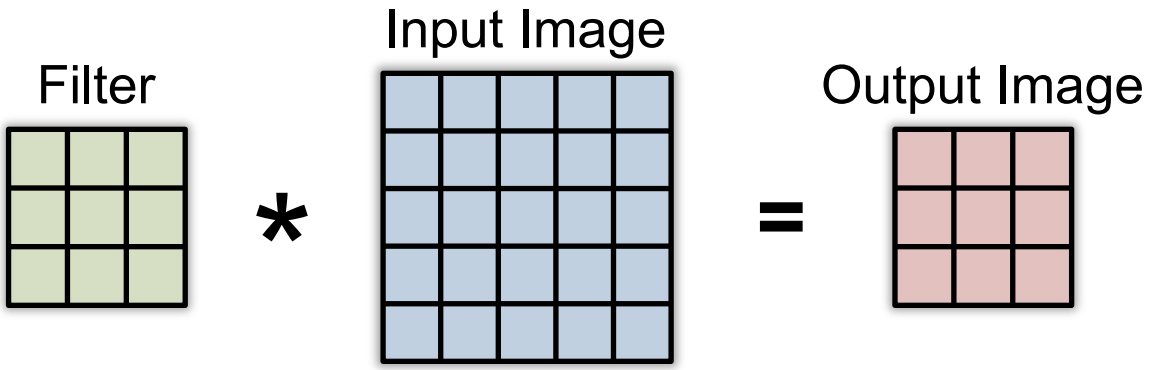


- Use a **large global buffer** as shared storage
  - Reduce **DRAM** access energy consumption
- **Examples:**

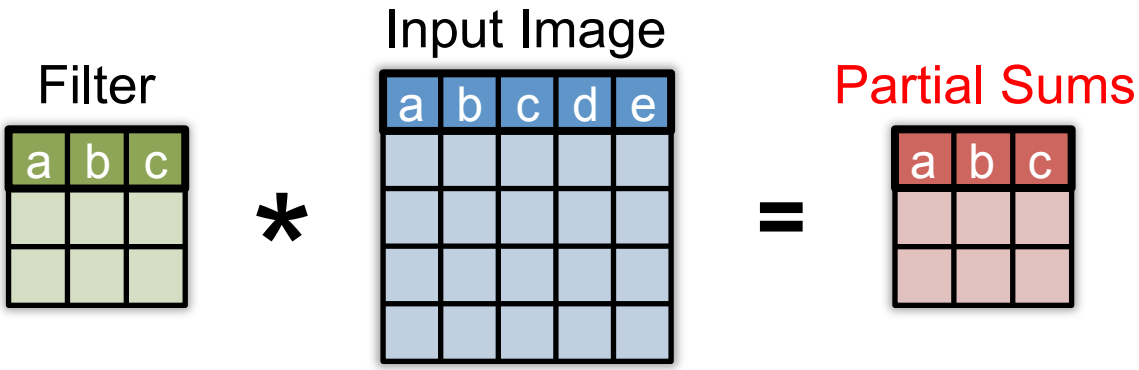
  [**DianNao**, *ASPLOS* 2014] [**DaDianNao**, *MICRO* 2014]

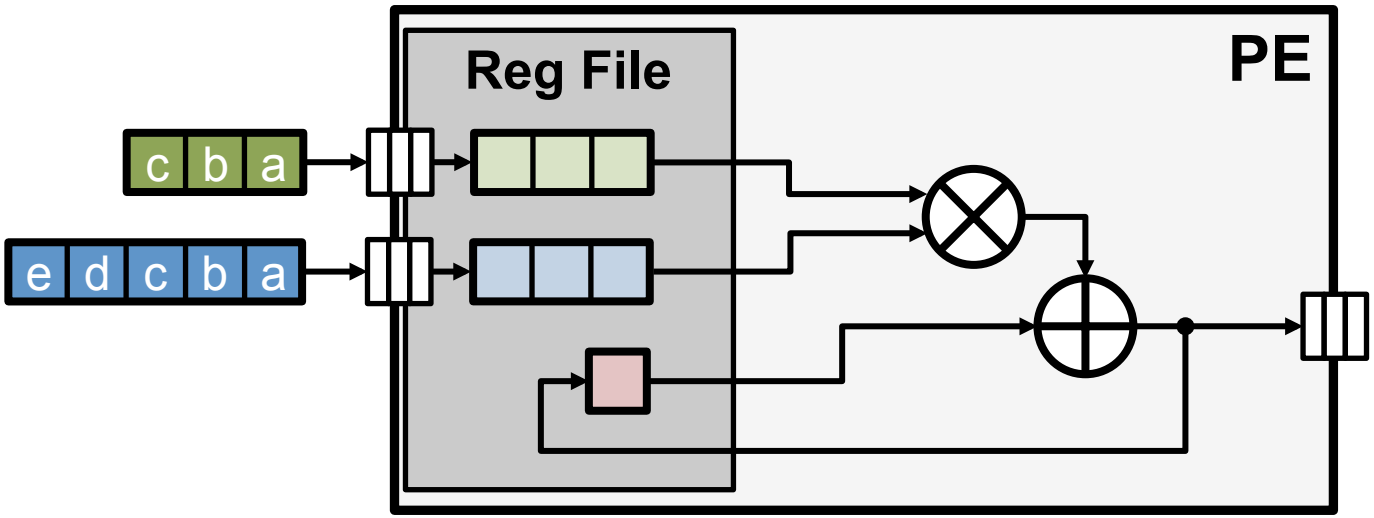
  [**Zhang**, *FPGA* 2015]

Row Stationary: Energy-efficient Dataflow

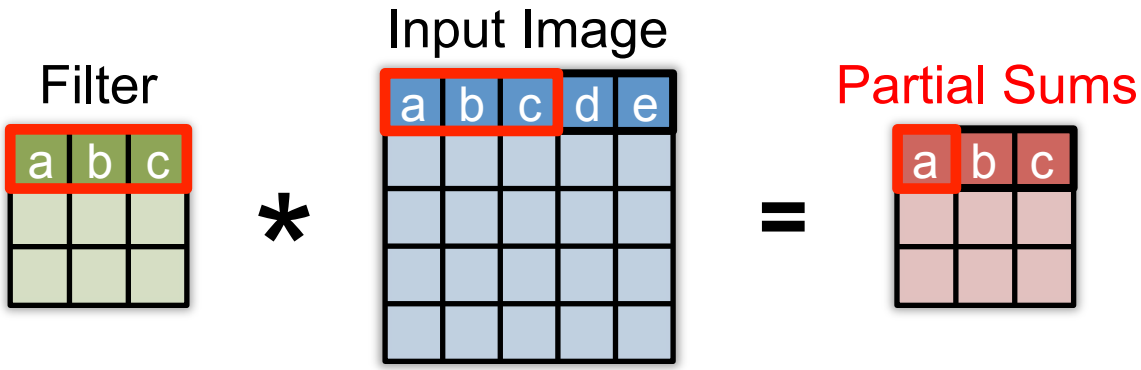


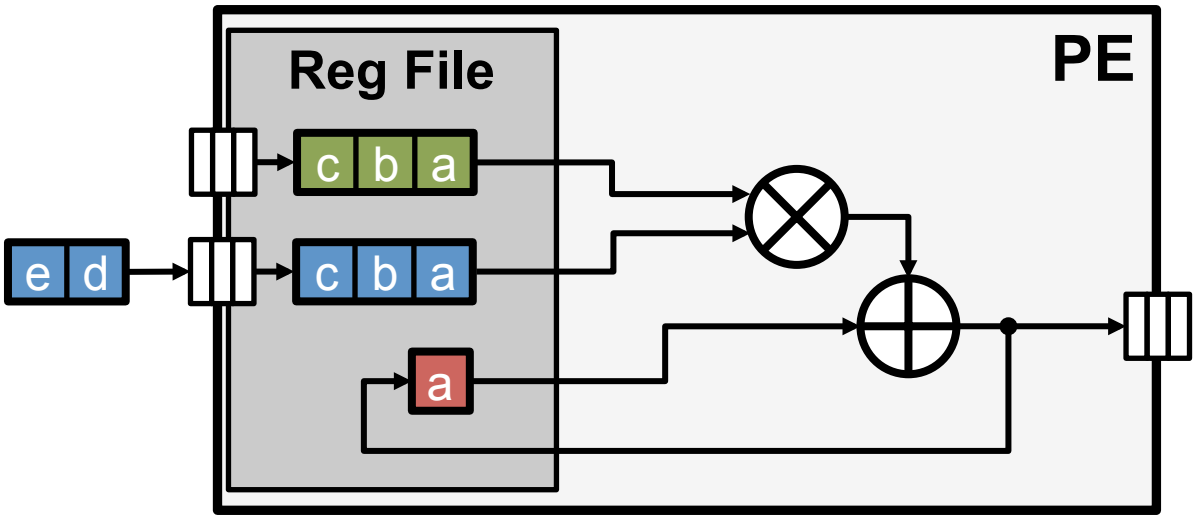
1D Row Convolution in PE



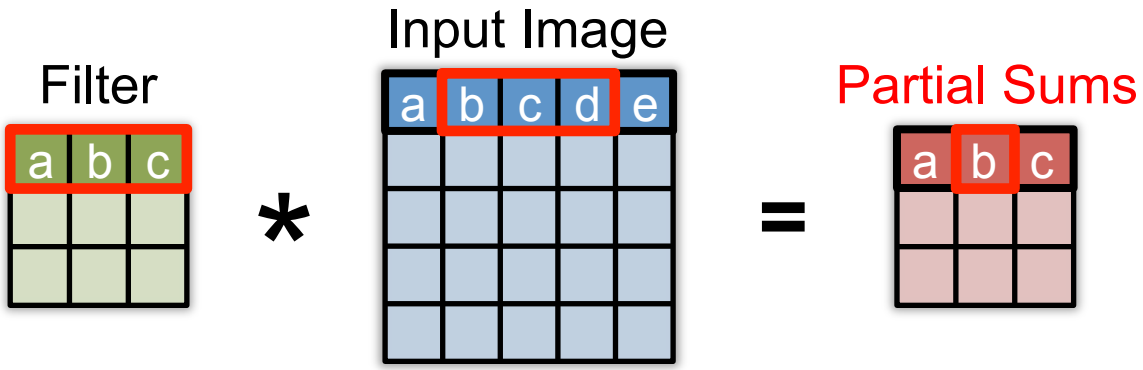


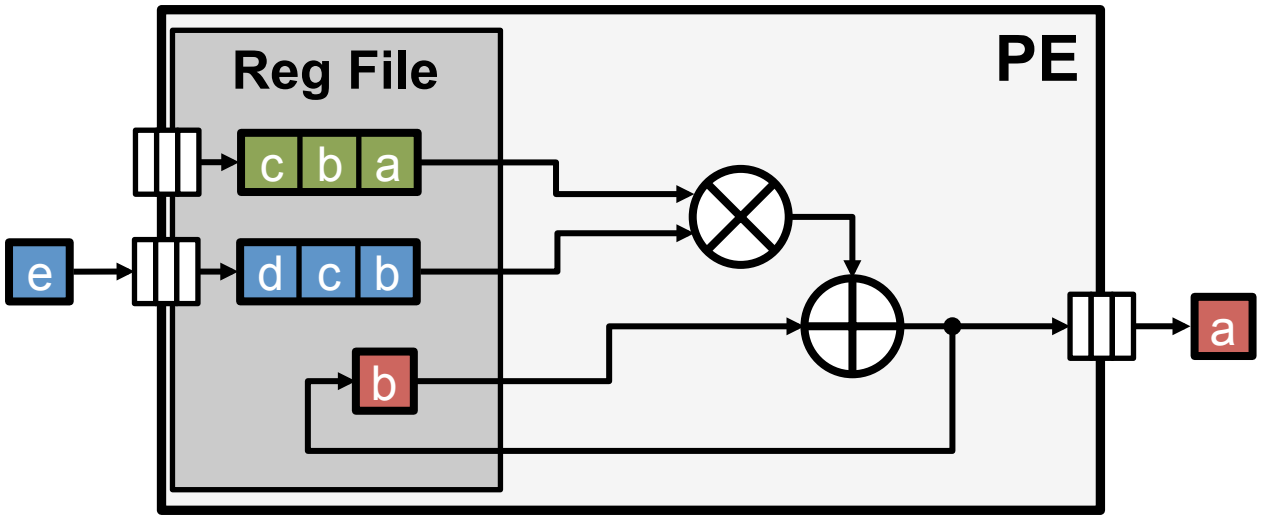
1D Row Convolution in PE



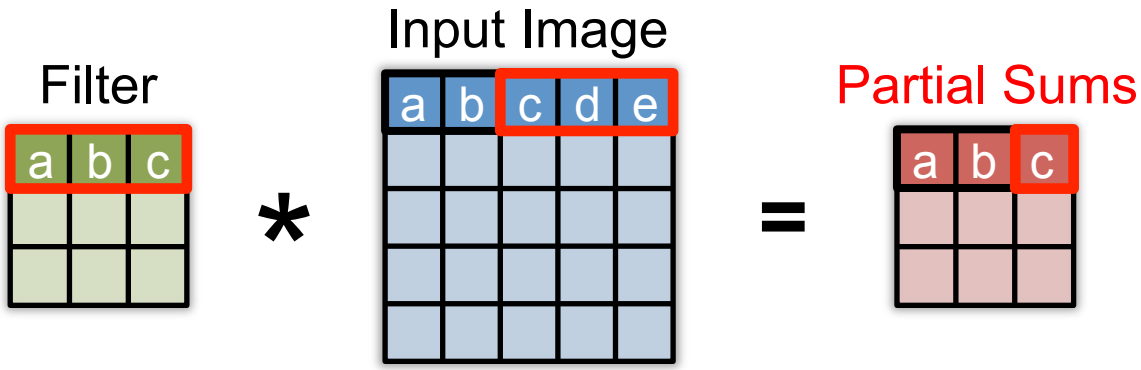


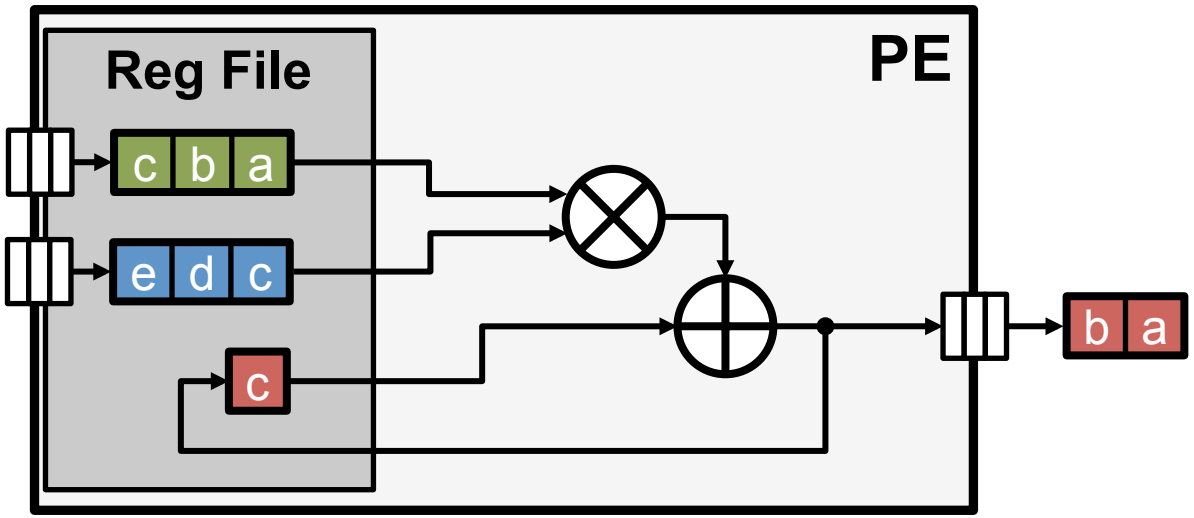
1D Row Convolution in PE





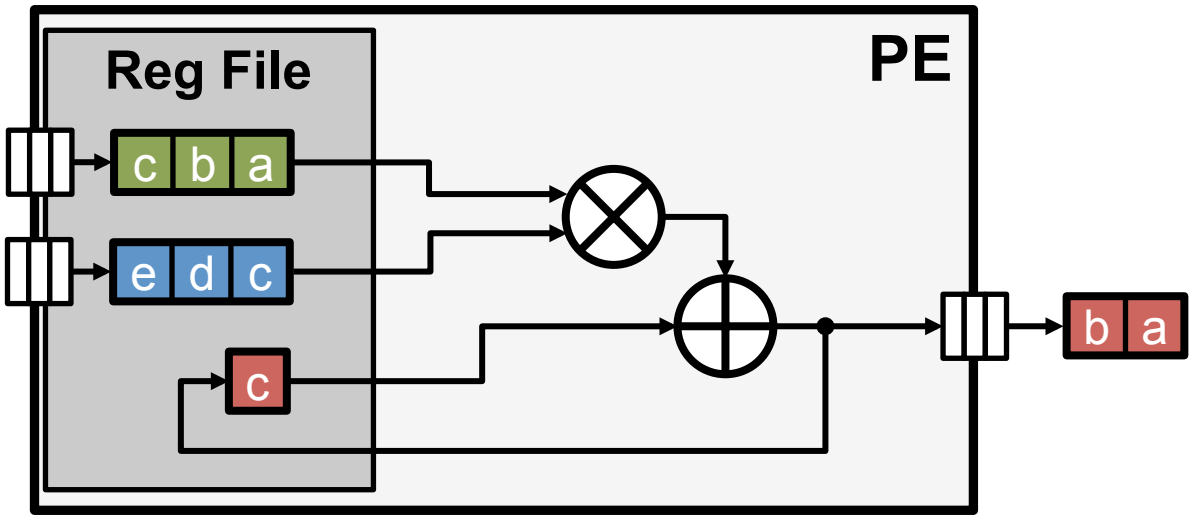
1D Row Convolution in PE



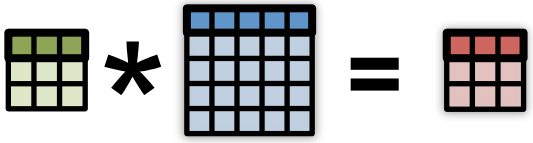


1D Row Convolution in PE

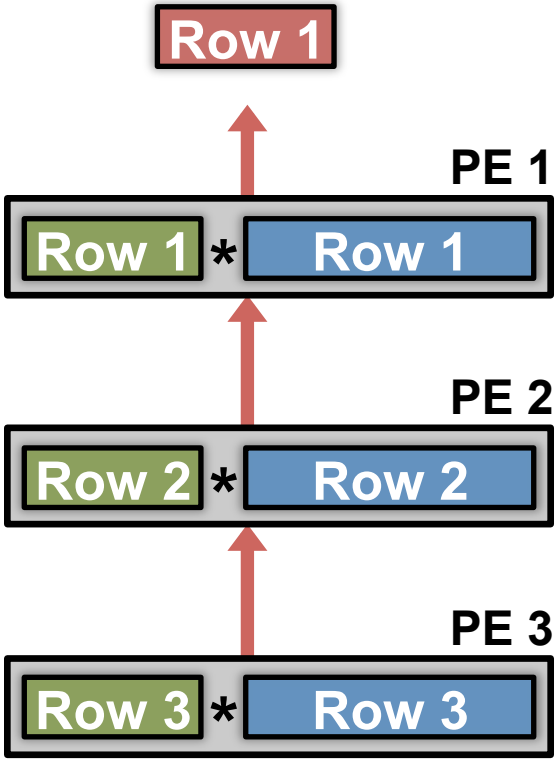
- Maximize row **convolutional reuse** in RF
  - Keep a **filter** row and **image** sliding window in RF
- Maximize row **psum accumulation** in RF

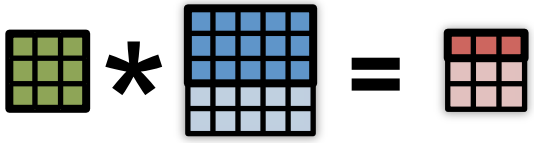


2D Convolution in PE Array

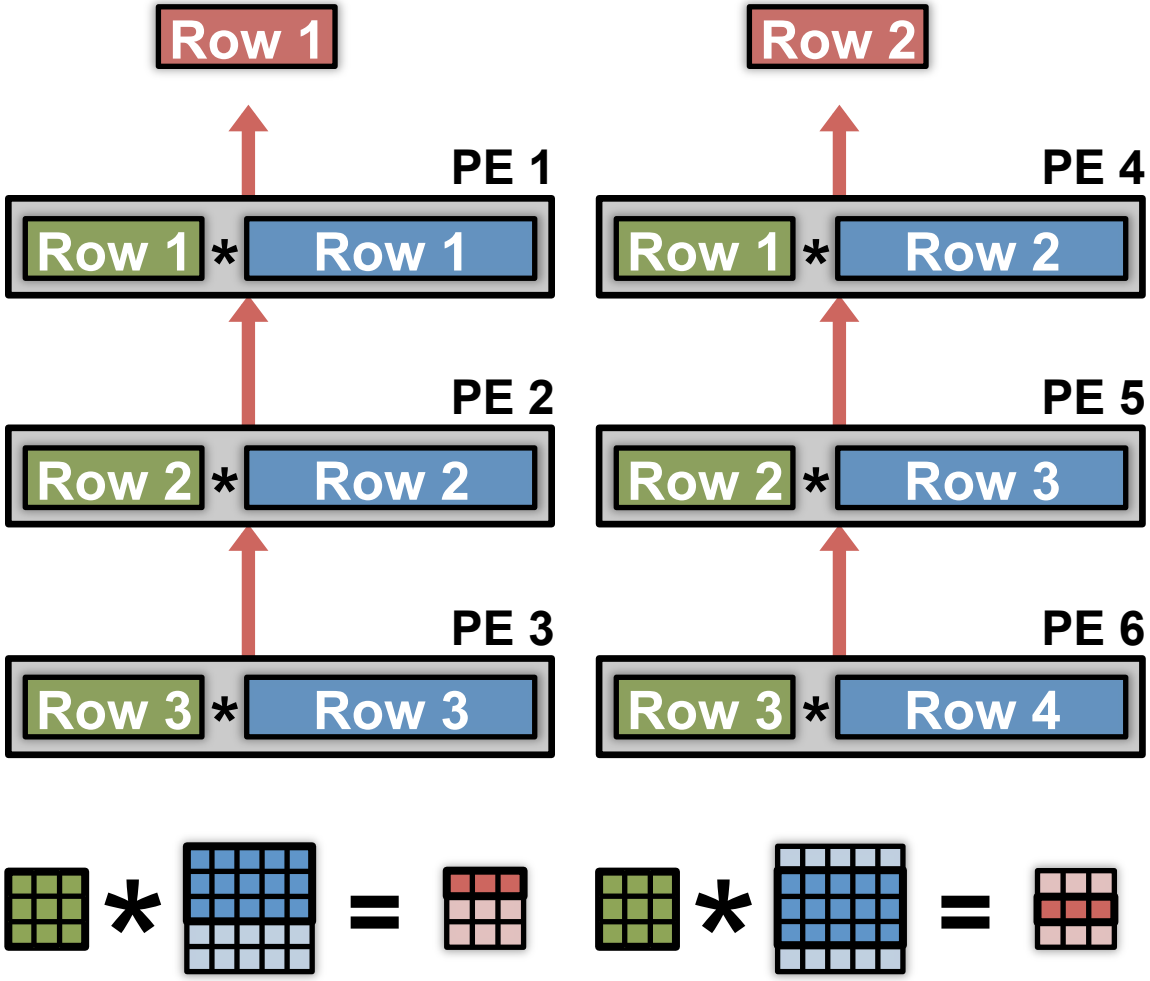


2D Convolution in PE Array

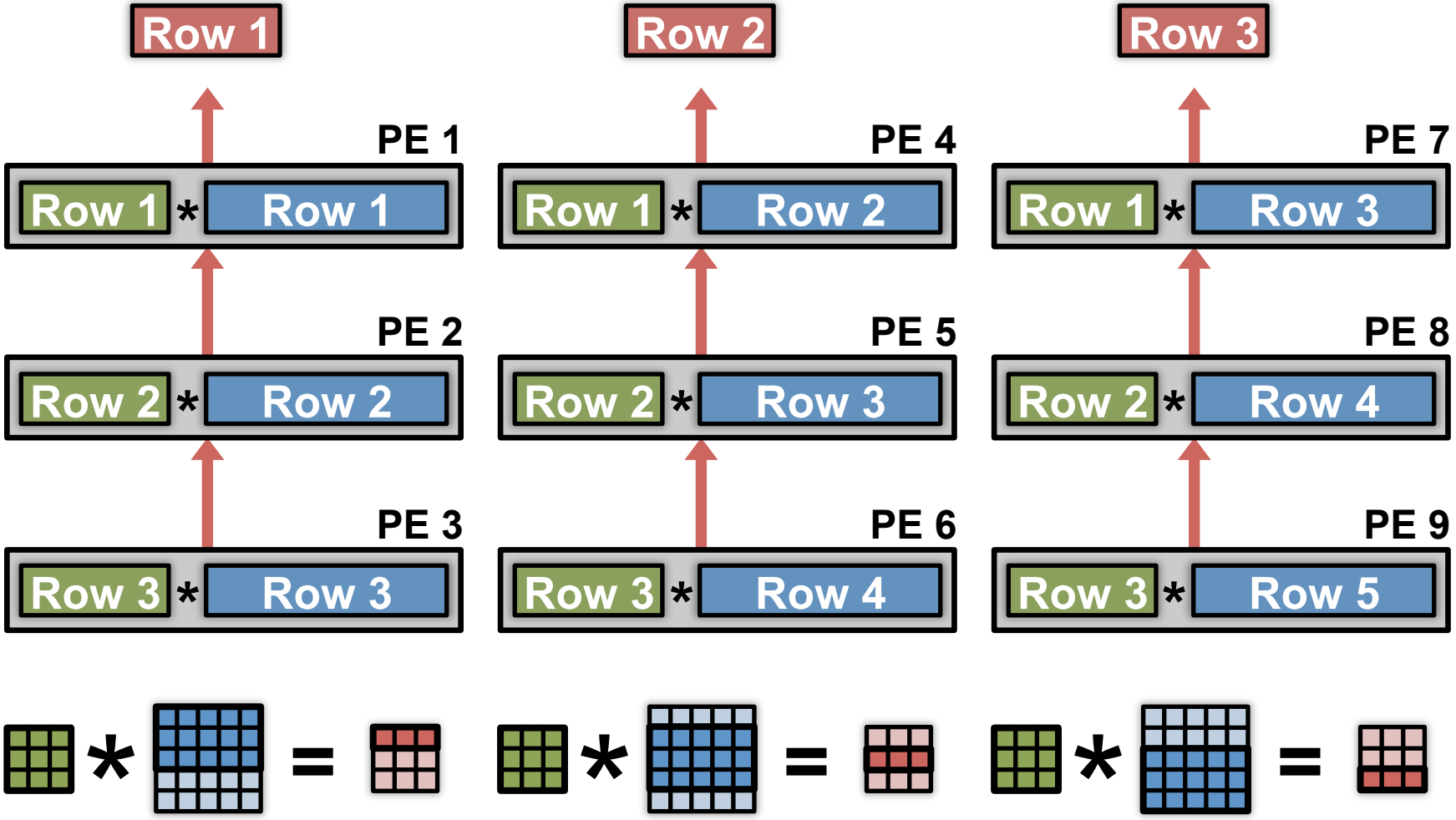




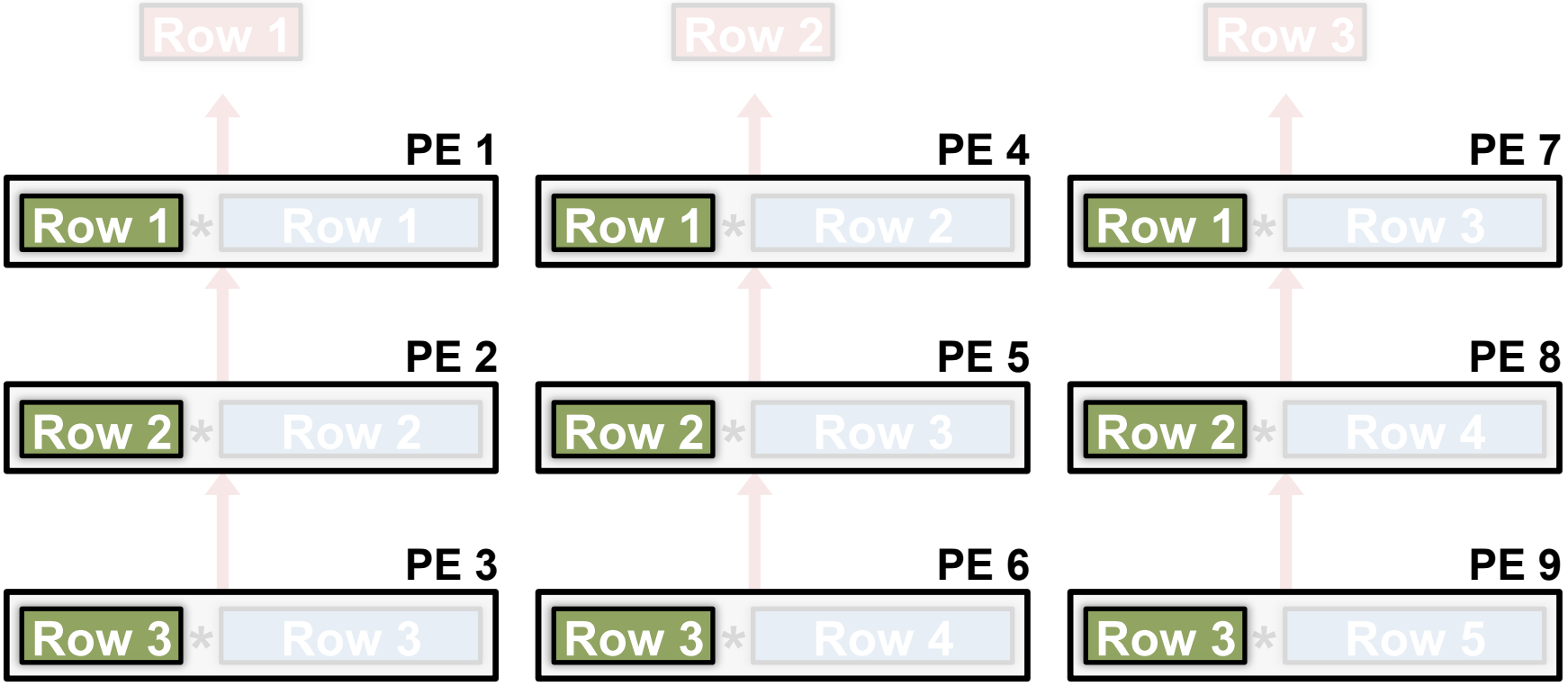
2D Convolution in PE Array



2D Convolution in PE Array



Convolutional Reuse Maximized



Filter rows are reused across PEs **horizontally**

Convolutional Reuse Maximized

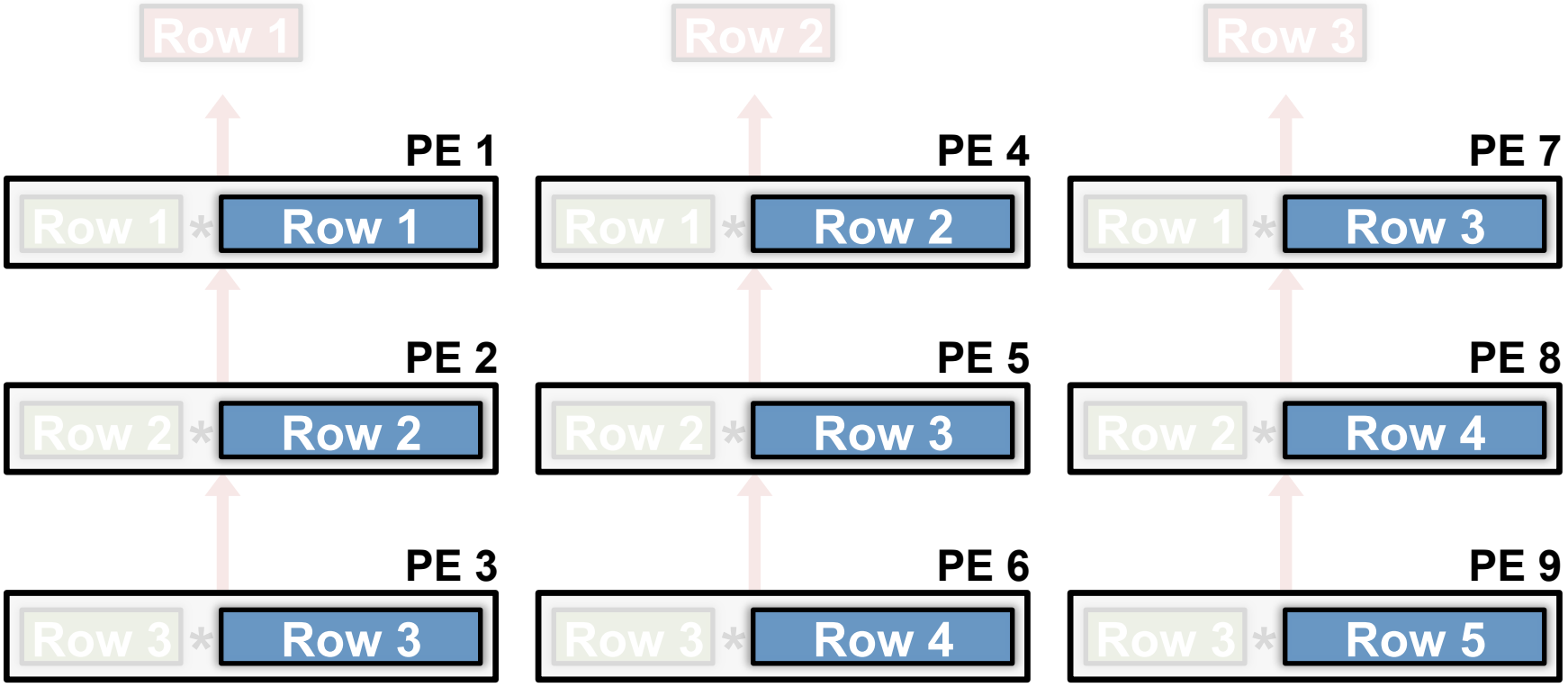
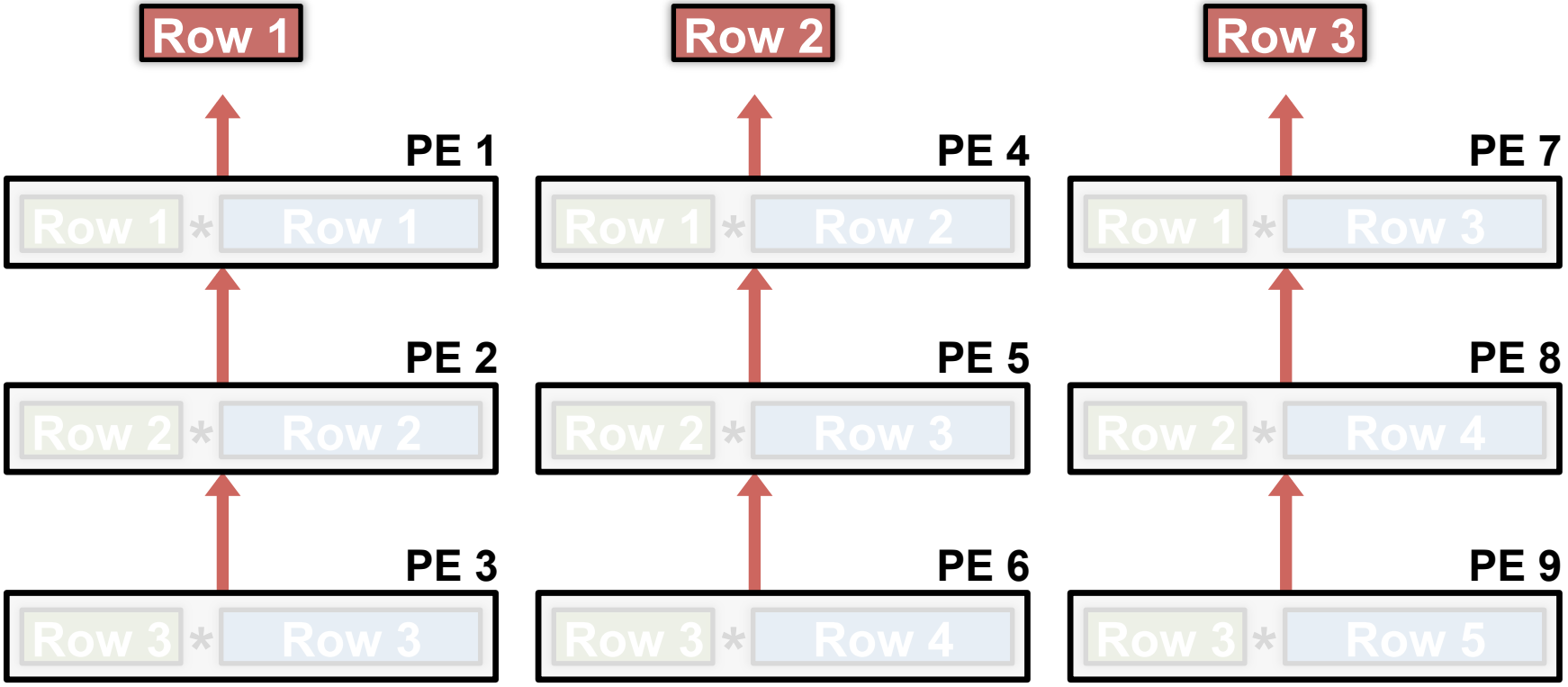


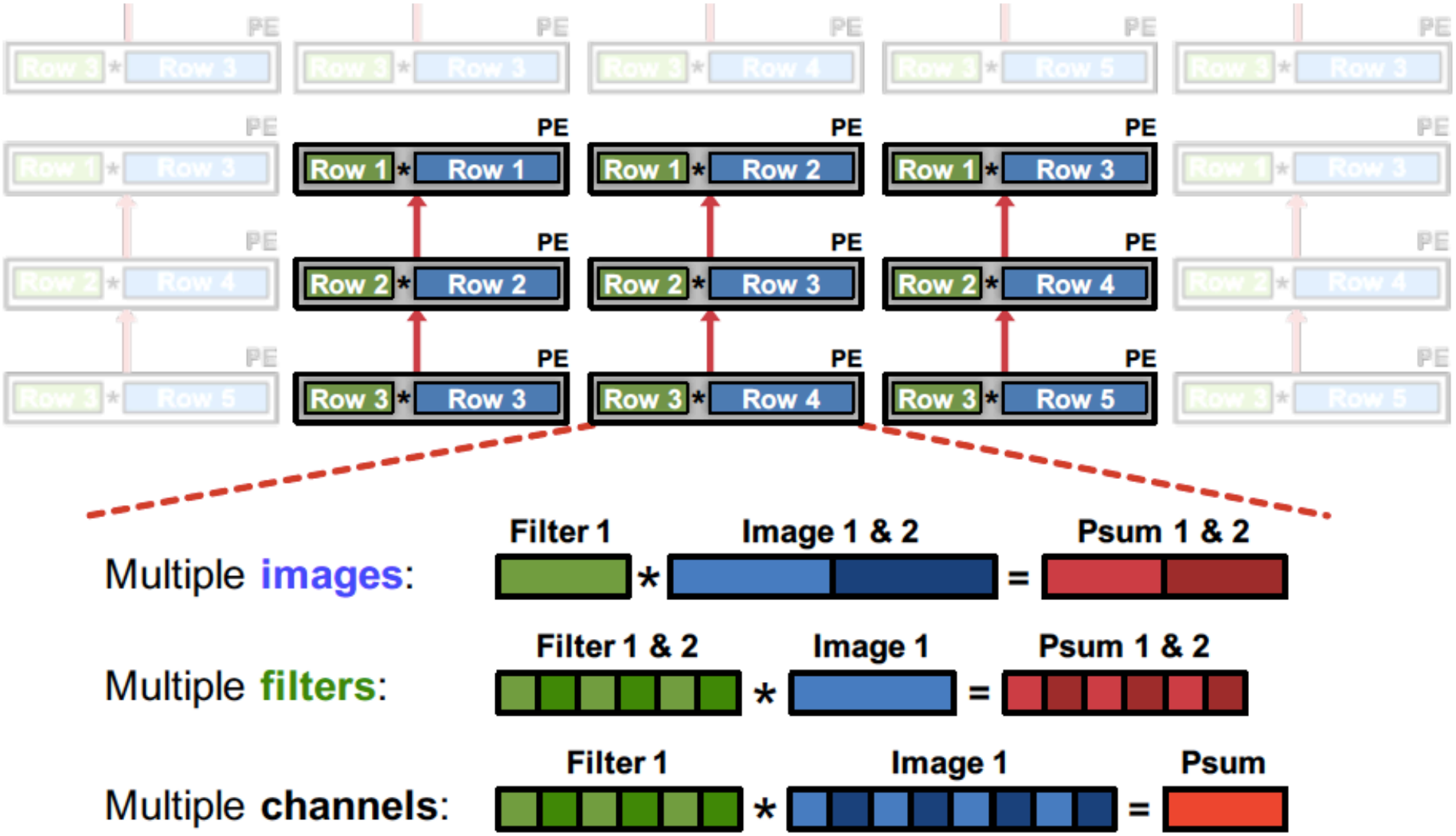
Image rows are reused across PEs **diagonally**

Maximize 2D Accumulation in PE Array



Partial sums accumulate across PEs **vertically**

CNN Convolution – The Full Picture



Map rows from **multiple images**, **filters** and **channels** to same PE to exploit other forms of reuse and local accumulation

Evaluate Reuse in Different Dataflows

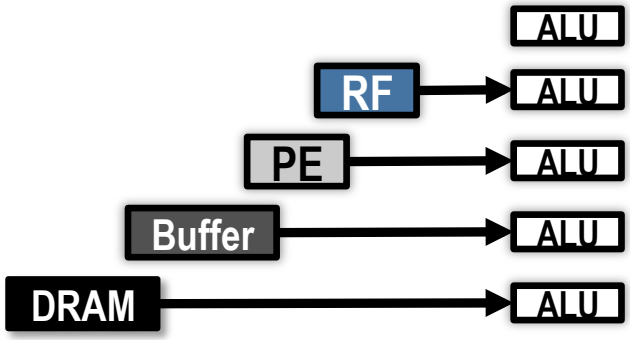
- **Weight Stationary**
  - Minimize movement of filter weights
- **Output Stationary**
  - Minimize movement of partial sums
- **No Local Reuse**
  - Don't use any local PE storage. Maximize global buffer size.
- **Row Stationary**

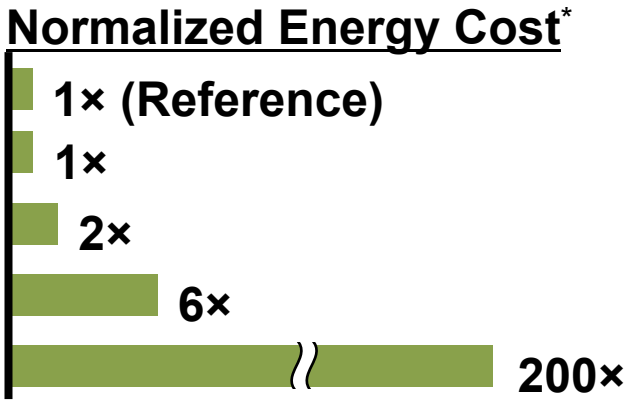
Evaluate Reuse in Different Dataflows

- **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**

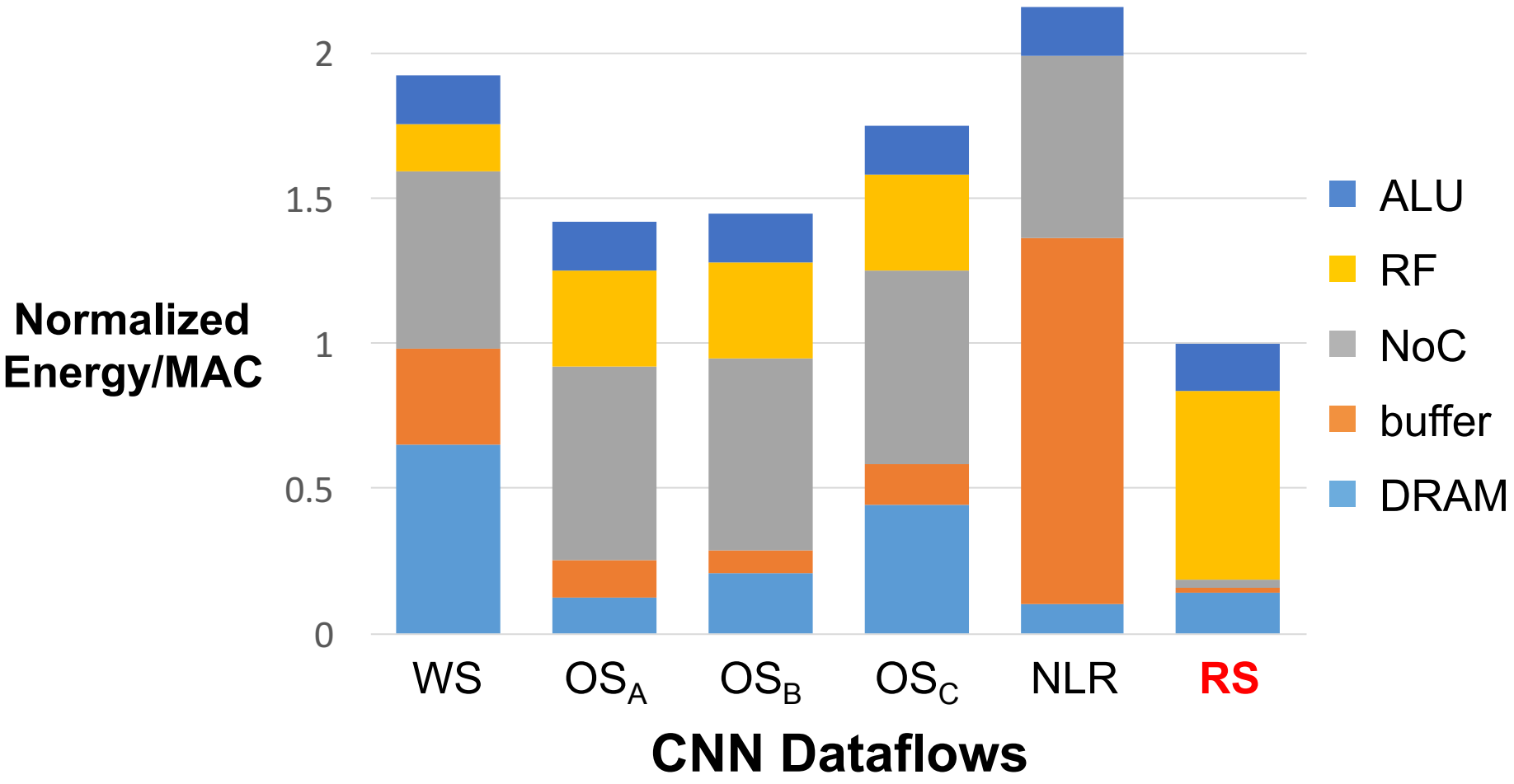
Evaluation Setup

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





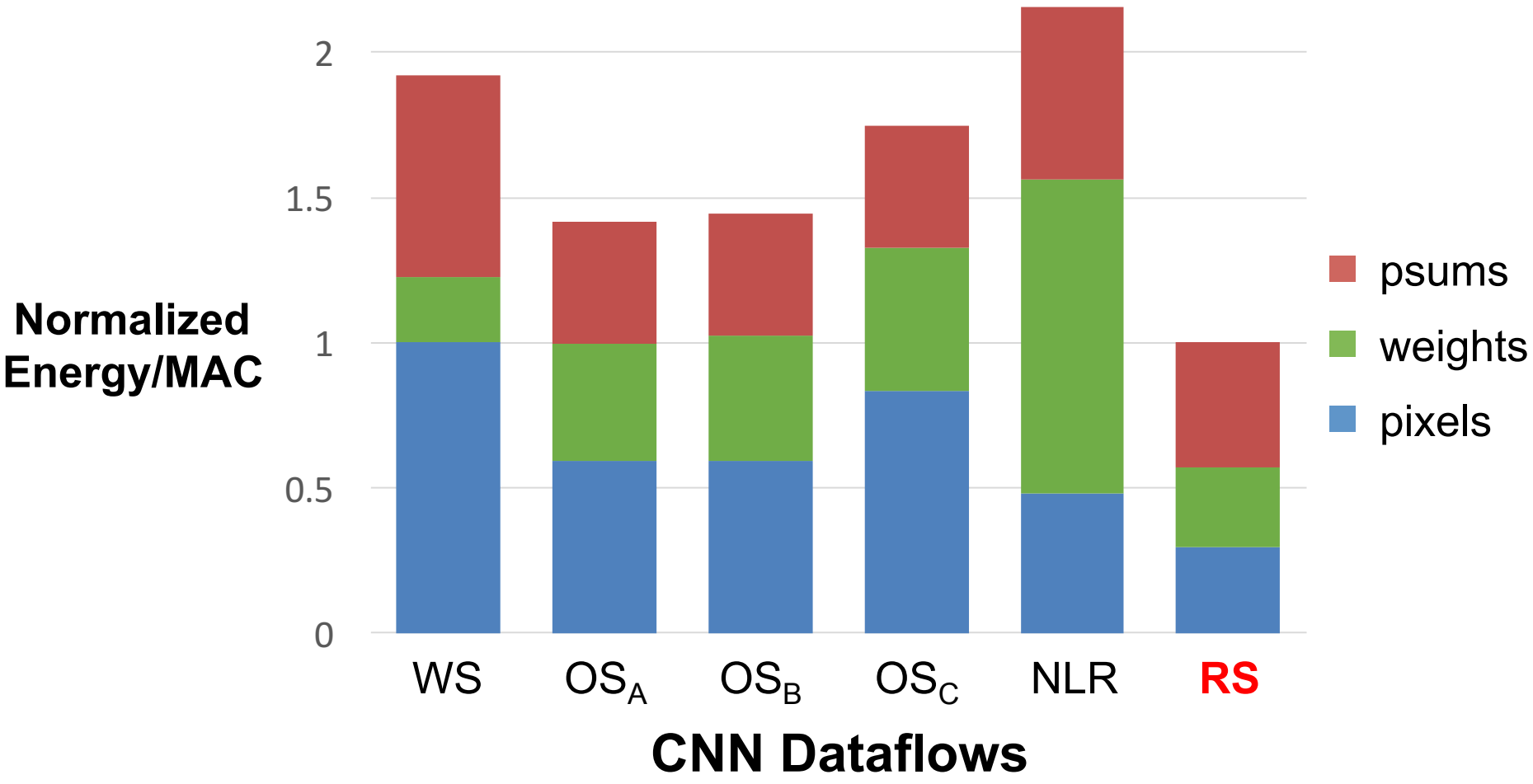
Dataflow Comparison: CONV Layers



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

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

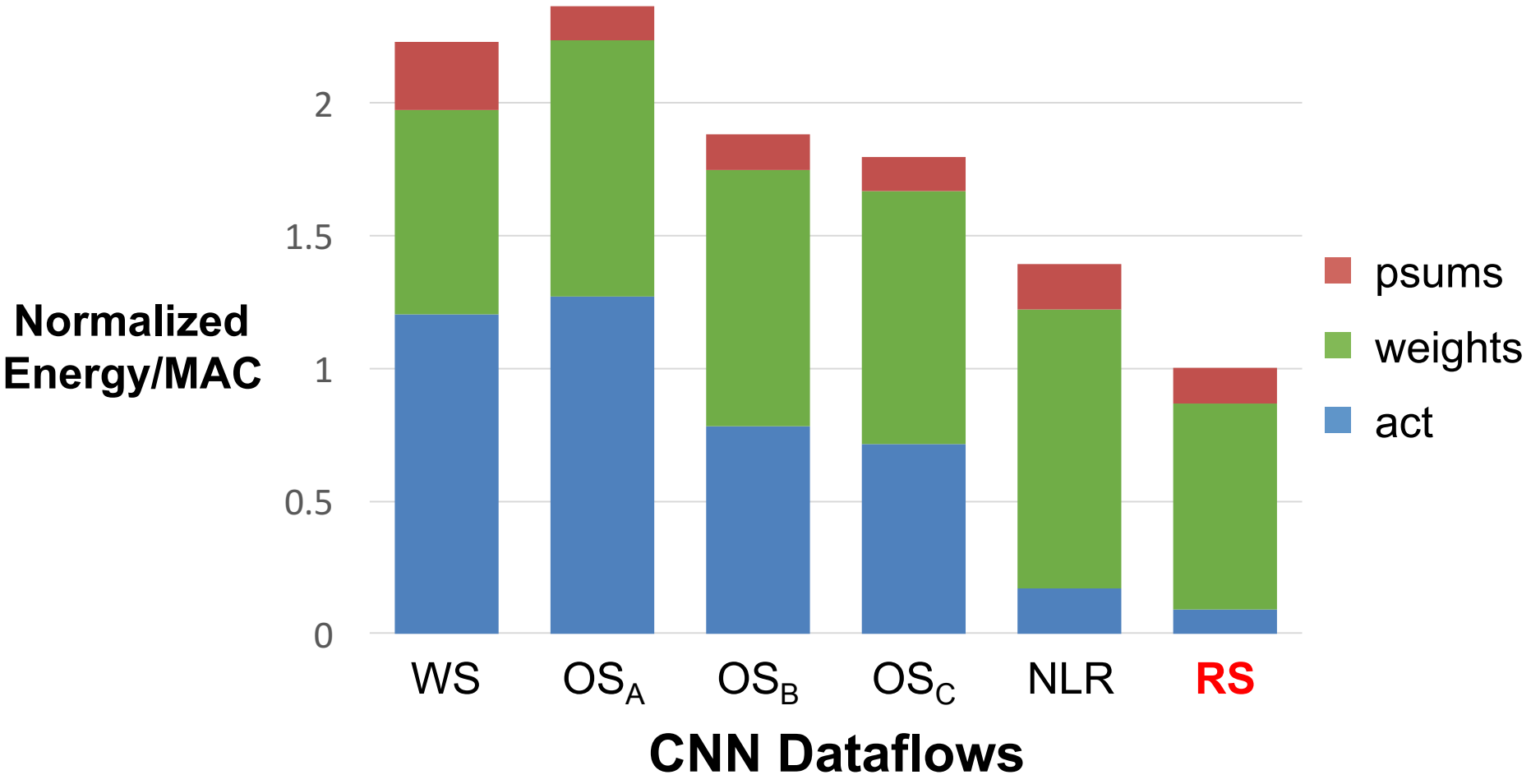
Dataflow Comparison: CONV Layers



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

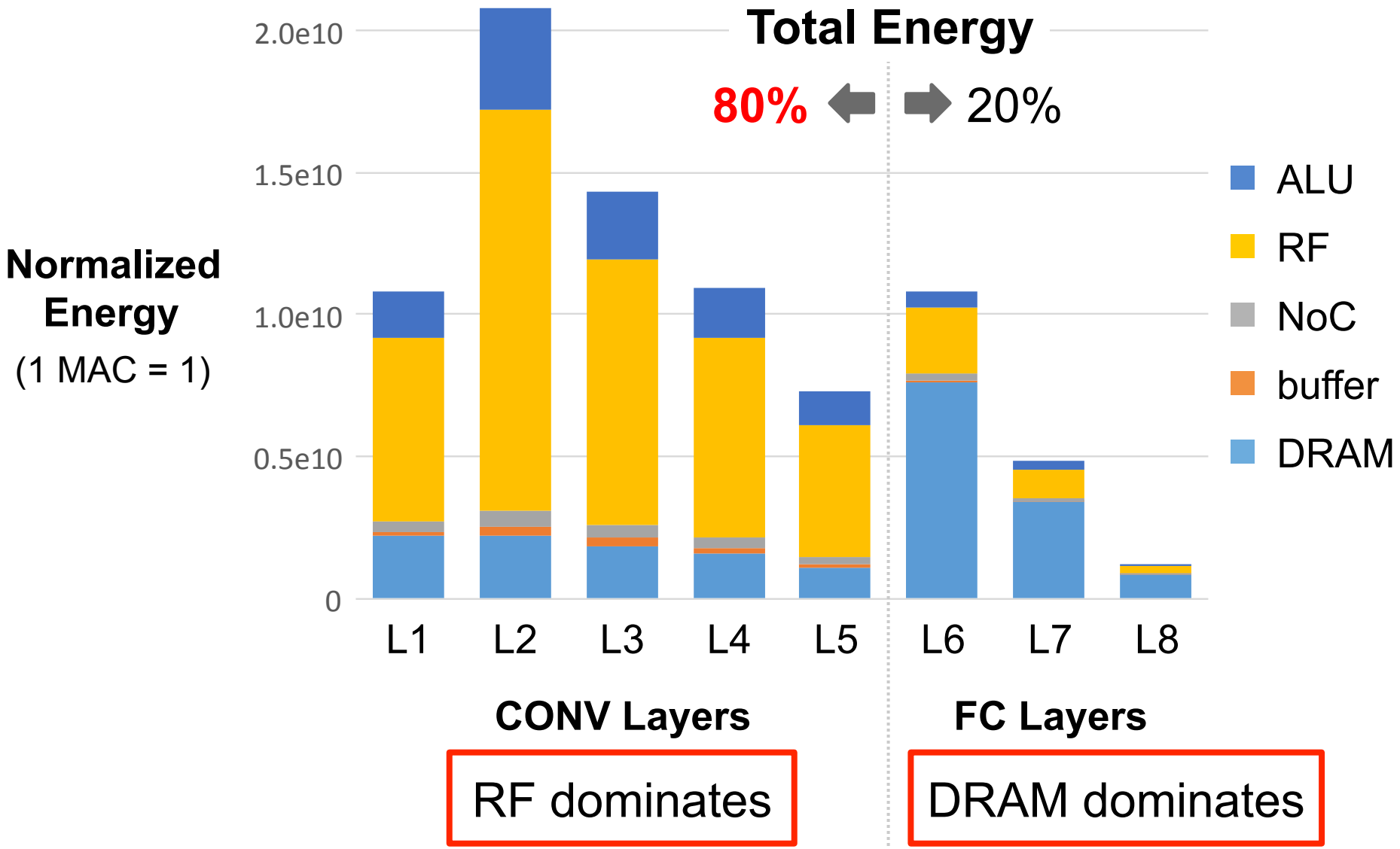
[**Chen**, *ISCA* 2016]

Dataflow Comparison: FC Layers

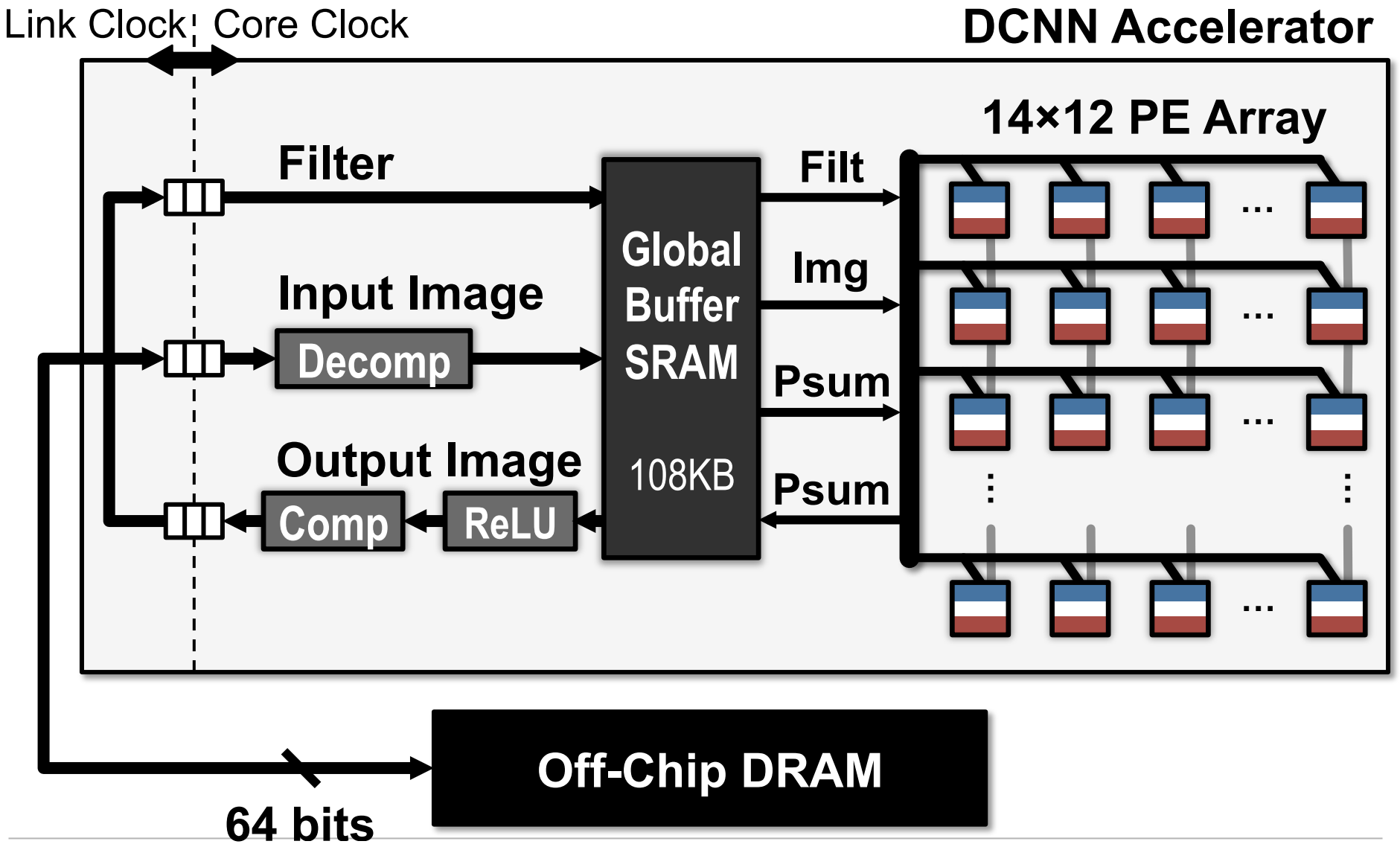


RS uses at least **1.3× lower** energy than other dataflows

Row Stationary: Layer Breakdown



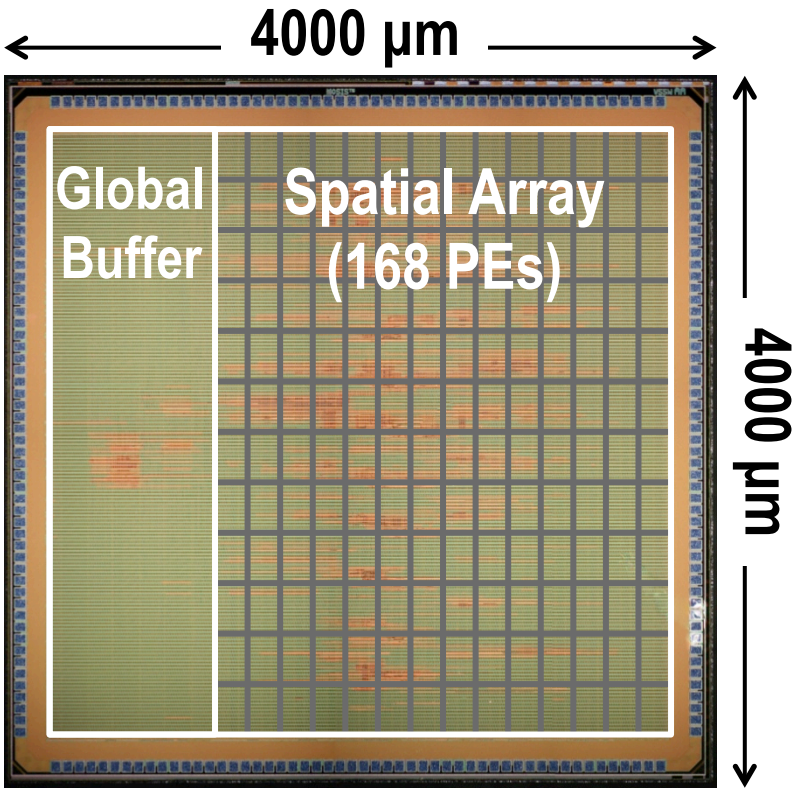
Eyeriss Deep CNN Accelerator



[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



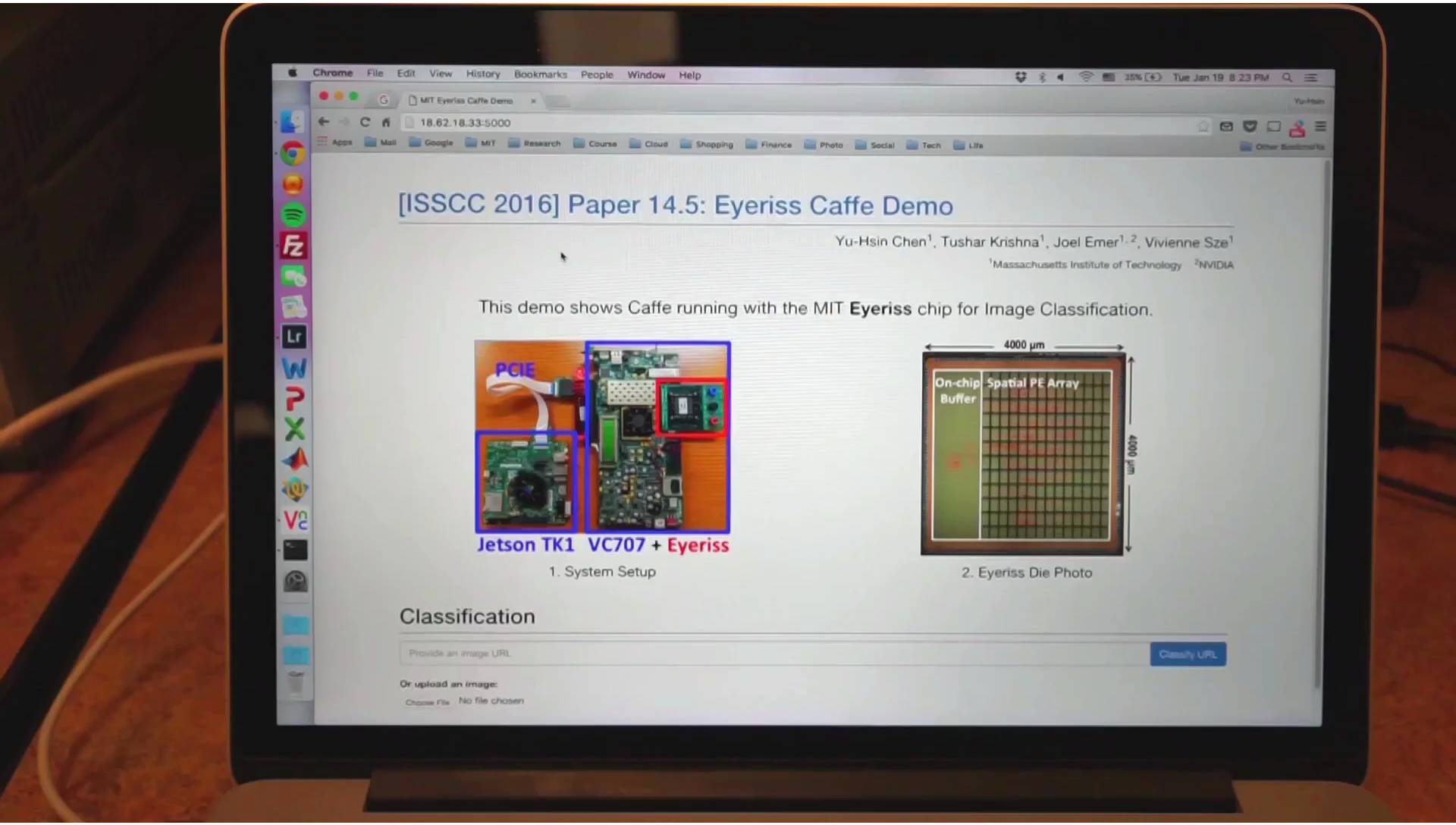
AlexNet: For 2.66 GMACs [8 billion 16-bit inputs (**16GB**) and 2.7 billion outputs (**5.4GB**)], only requires **208.5MB** (buffer) and **15.4MB** (DRAM)

Comparison with GPU

	This Work	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]

Demo of Image Classification on Eyeriss



https://vimeo.com/154012013

Integrated with BVLC Caffe DL Framework

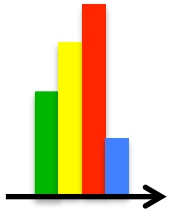
Hand-Crafted vs. Learned Features

Machine Learning Pipeline (Inference)

Image

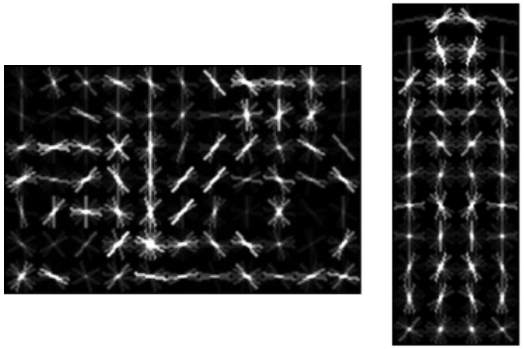
Trained weights (**w**)

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



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

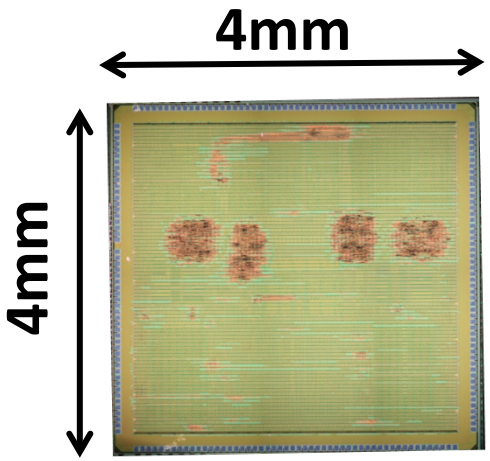
Handcrafted Features (e.g. HOG)



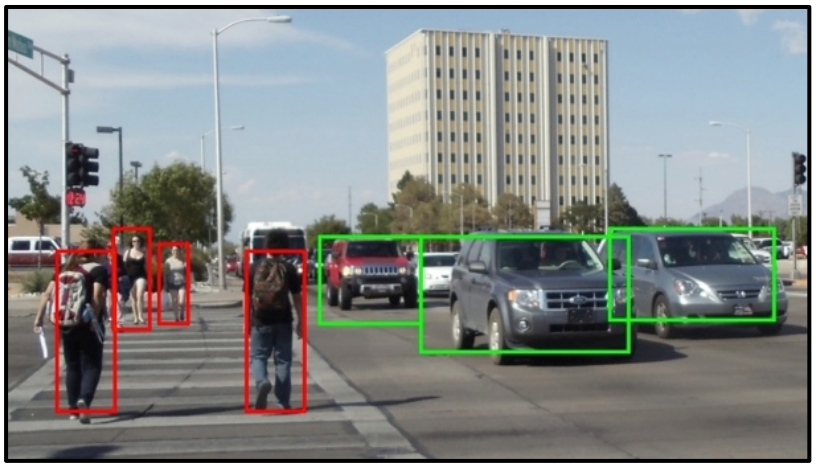
Learned Features (e.g. DNN)

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

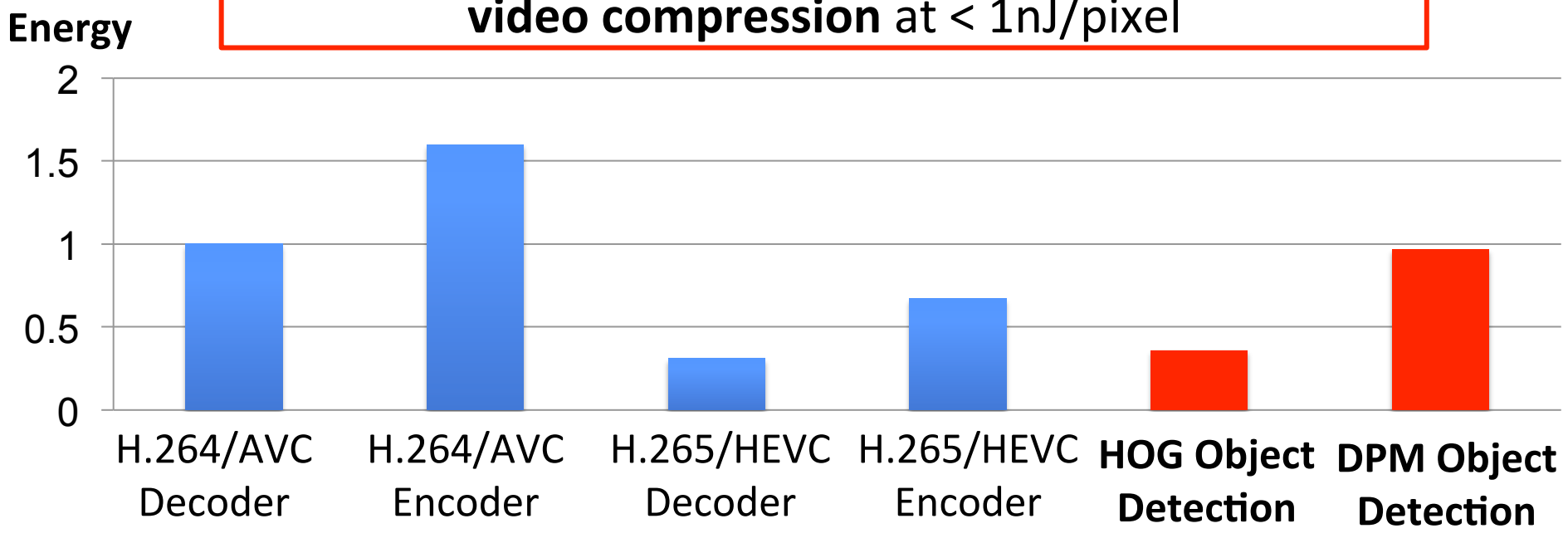
Energy-Efficient Object Detection



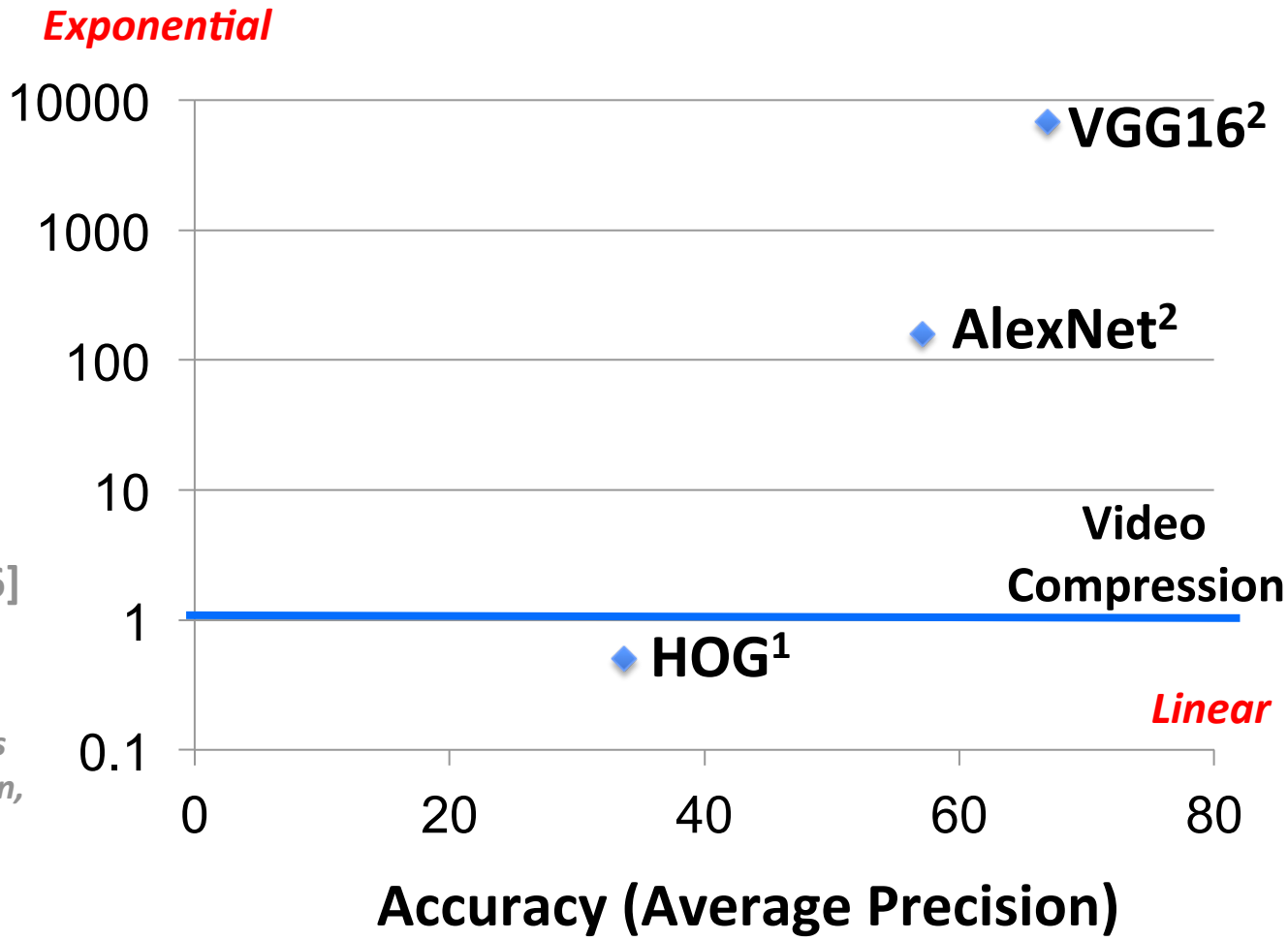
MIT Object Detection Chip [VLSI 2016]



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

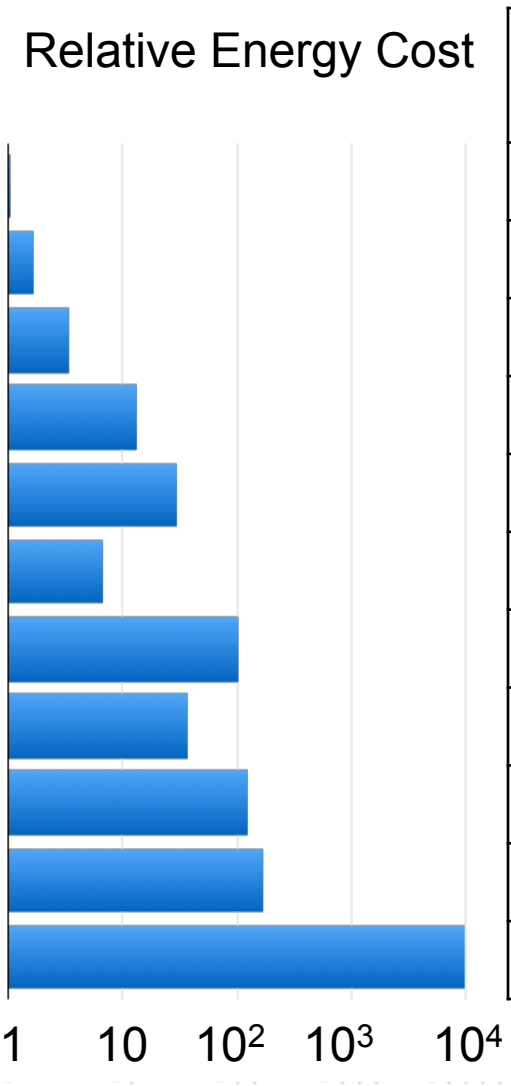
Approaches

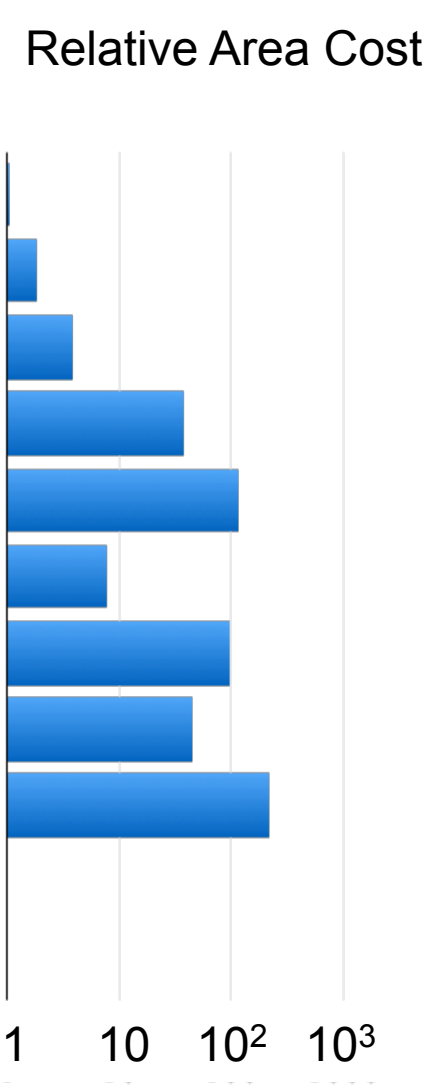
- Reduce size of operands for storage/compute
  - Floating point -> Fixed point
  - Bit-width reduction
  - Non-linear quantization

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

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

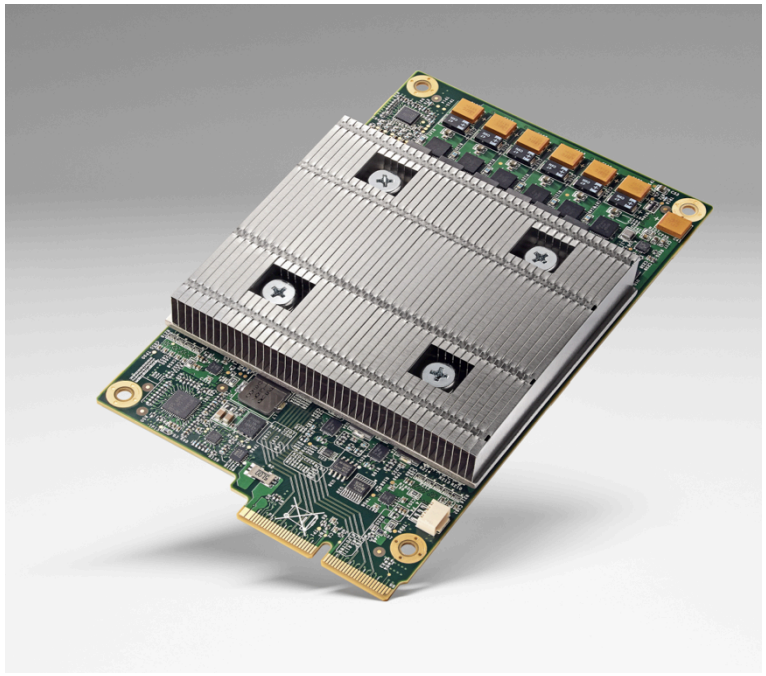




[Horowitz, "Computing's Energy Problem (and what we can do about it)", ISSCC 2014]

Commercial Products using 8-bit Integer

Nvidia's Pascal (2016)

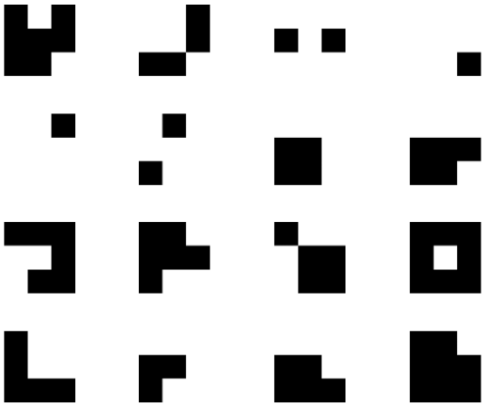


Google's TPU (2016)

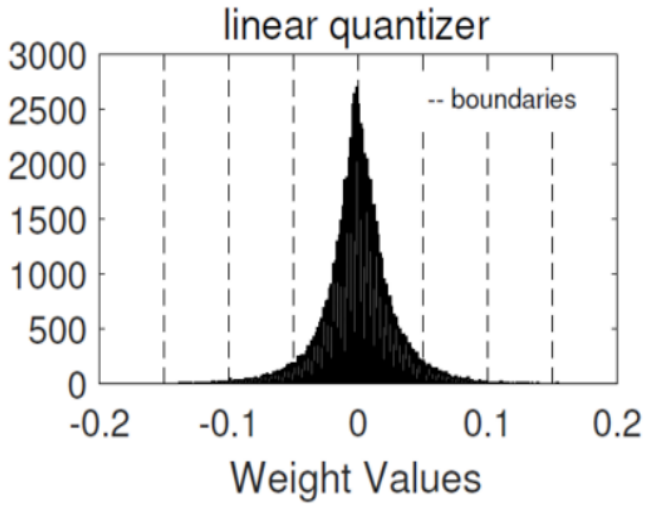
Reduced Precision in Research

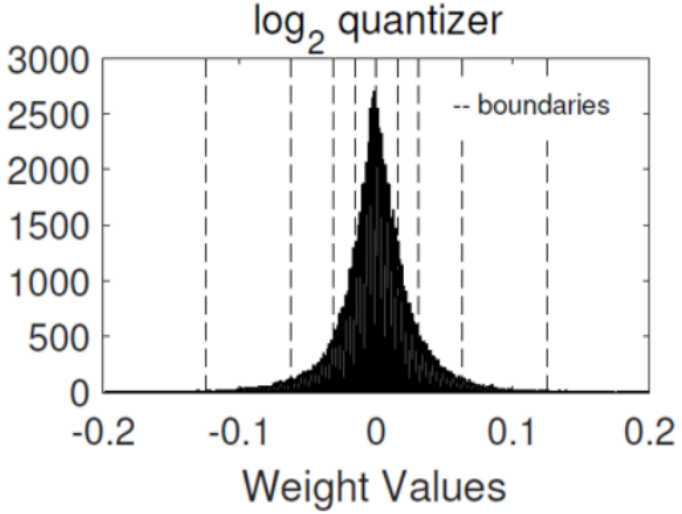
- **Reduce number of bits**
  - Binary Nets [Courbariaux, NIPS 2015]
- **Reduce number of unique weights**
  - Ternary Weight Nets [Li, arXiv 2016]
  - XNOR-Net [Rategari, ECCV 2016]
- **Non-Linear Quantization**
  - LogNet [Lee, ICASSP 2017]

Binary Filters



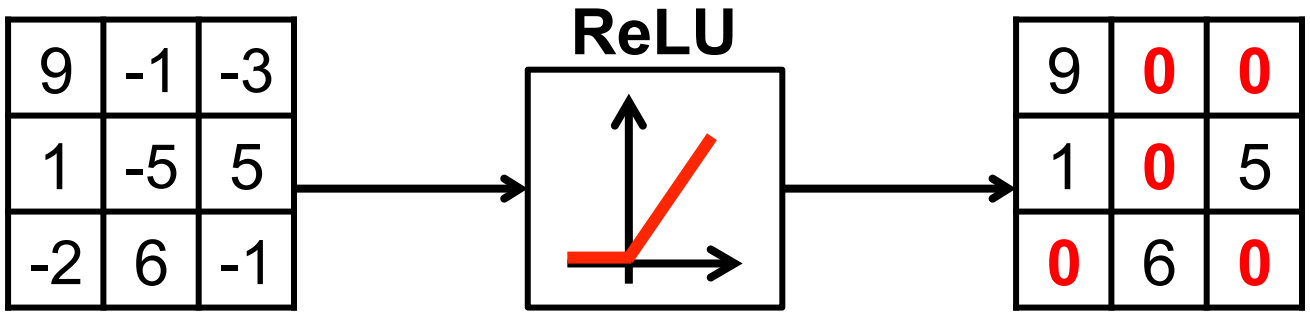
Log Domain Quantization

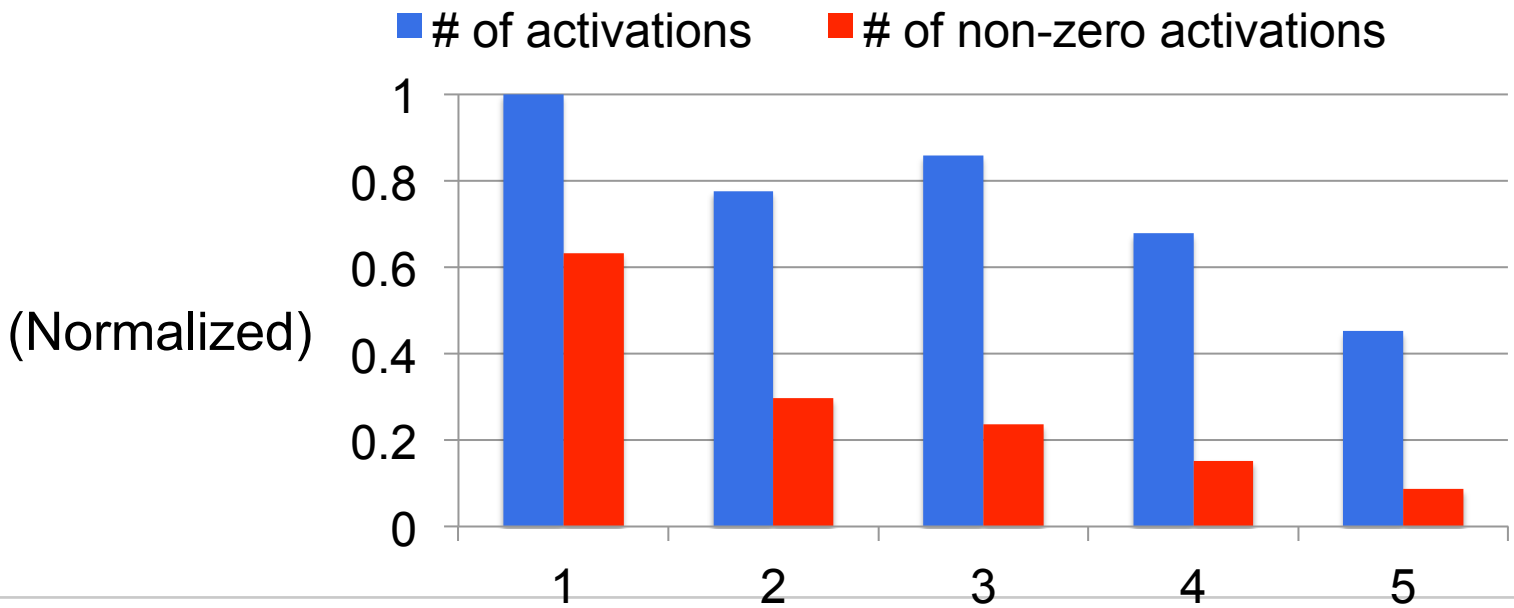




Sparsity in Data

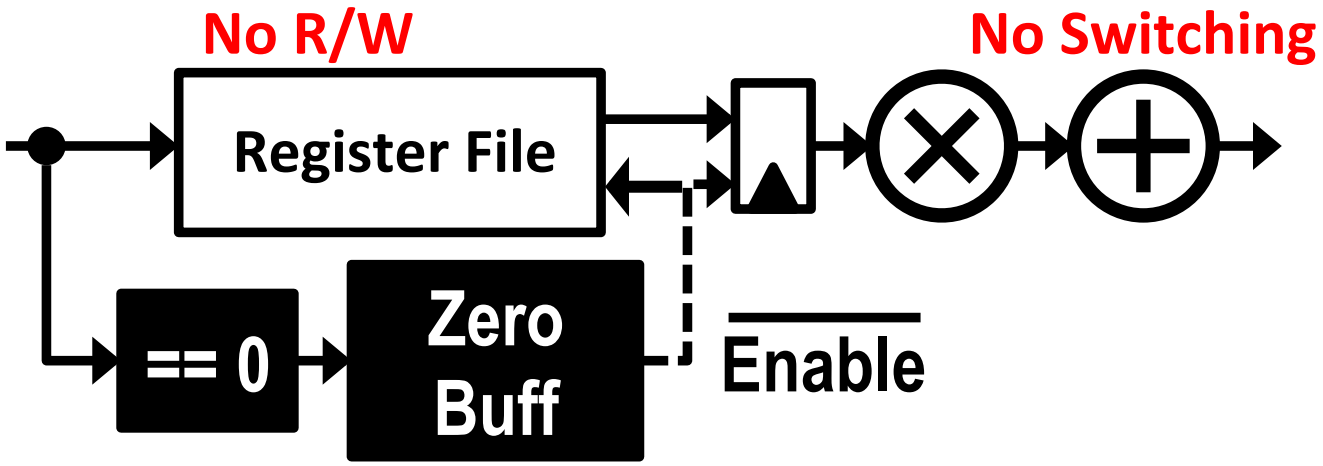
Many **zeros** in **output fmaps** after **ReLU**





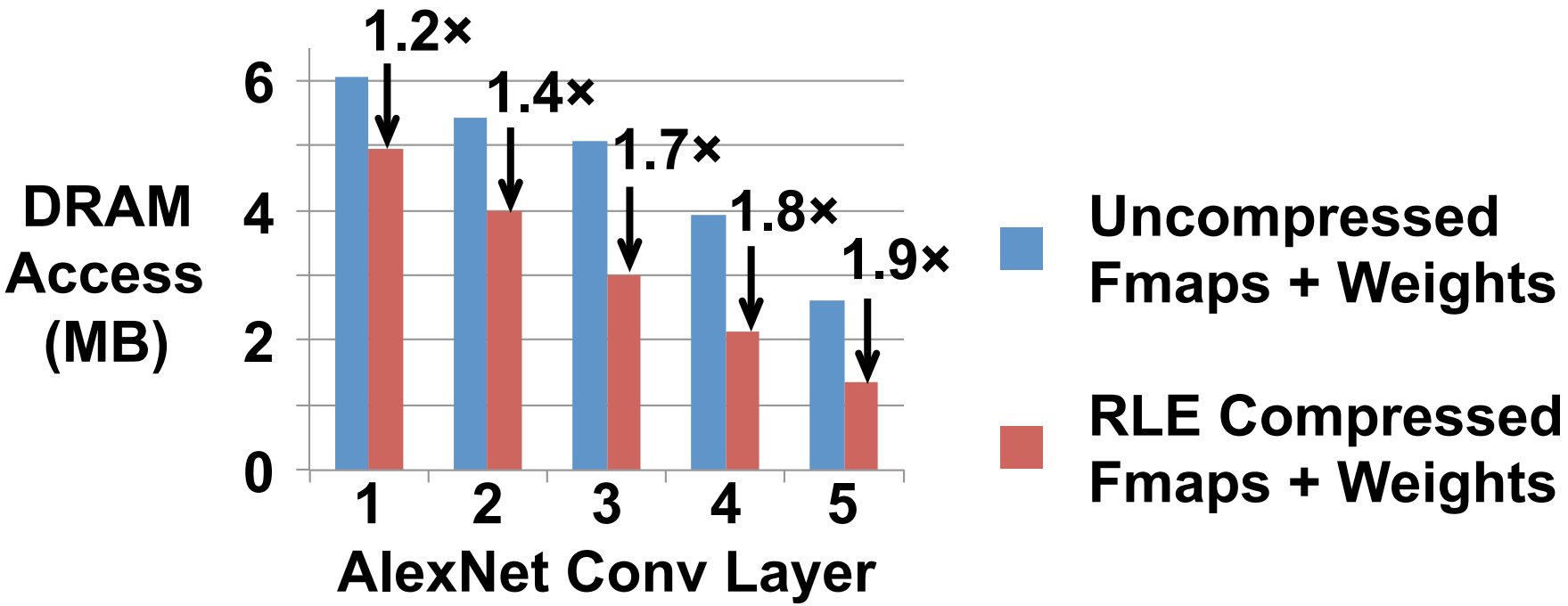
Zero Data Processing Gating

- 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

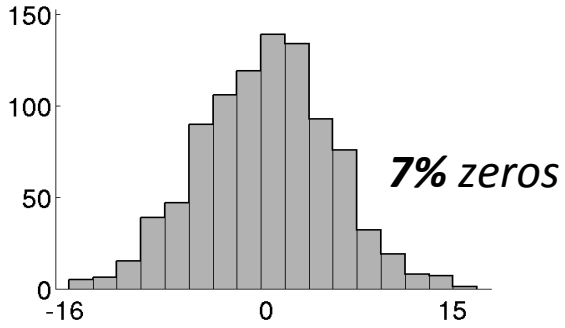
Reduce the number of multiplications by projecting onto a basis that increases sparsity (>1.8x power reduction)

Basis Projection Equation

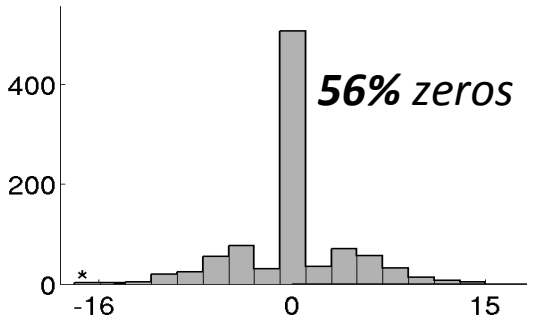
$$\langle H, W \rangle = \left\langle H, \sum_d S_d \alpha_d \right\rangle = \sum_d \langle H, S_d \rangle \alpha_d = \sum_d P_d \alpha_d$$

Features — Weights — Basis — Projected Features — Projected Weights

Histogram of Weights



Weights (W)

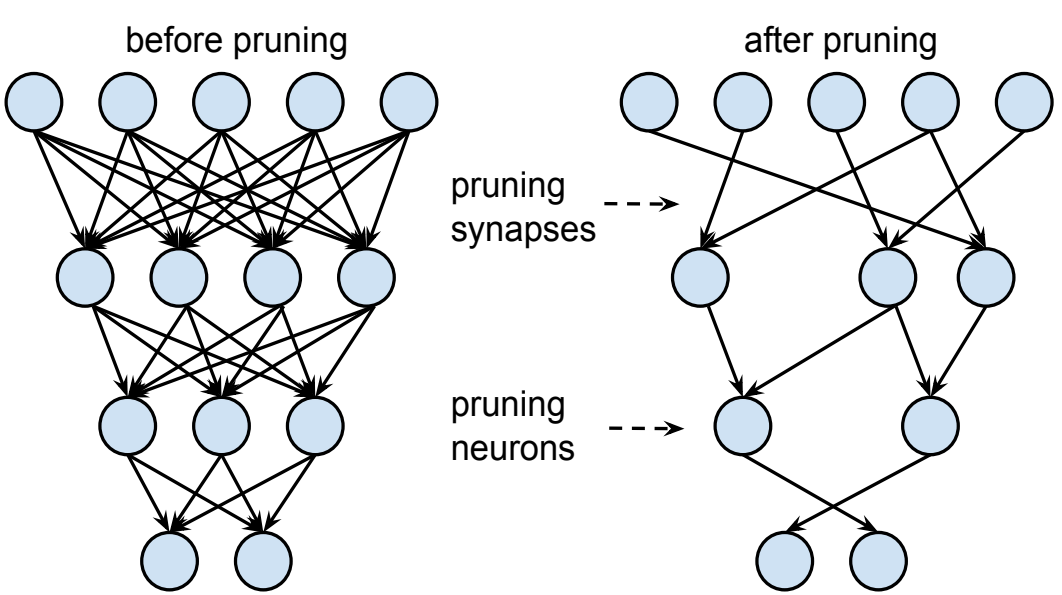


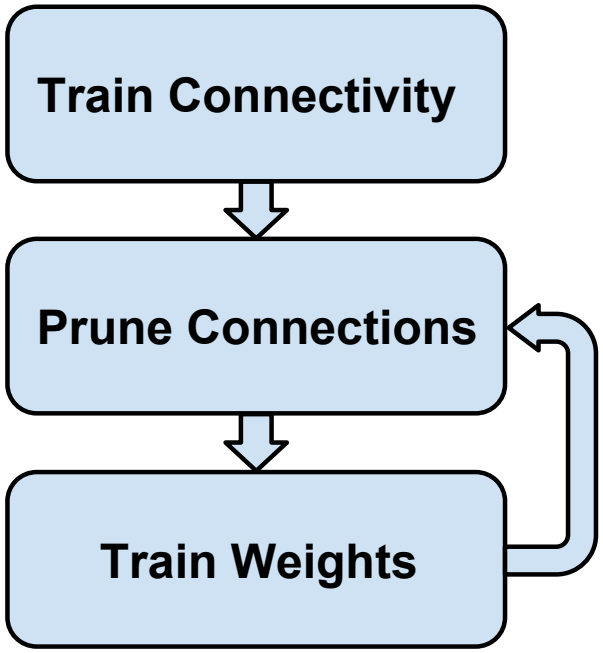
Projected Weights (α_d)

[Suleiman et al., VLSI 2016]

Pruning – Make Weights Sparse

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





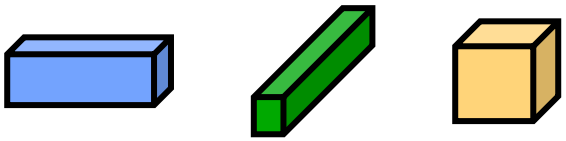
Example: *AlexNet*
Weight Reduction: *CONV layers 2.7x, FC layers 9.9x*
(Most reduction on fully connected layers)
Overall: *9x weight reduction, 3x MAC reduction*

[Han et al., NIPS 2015]

Key Metrics for Embedded DNN

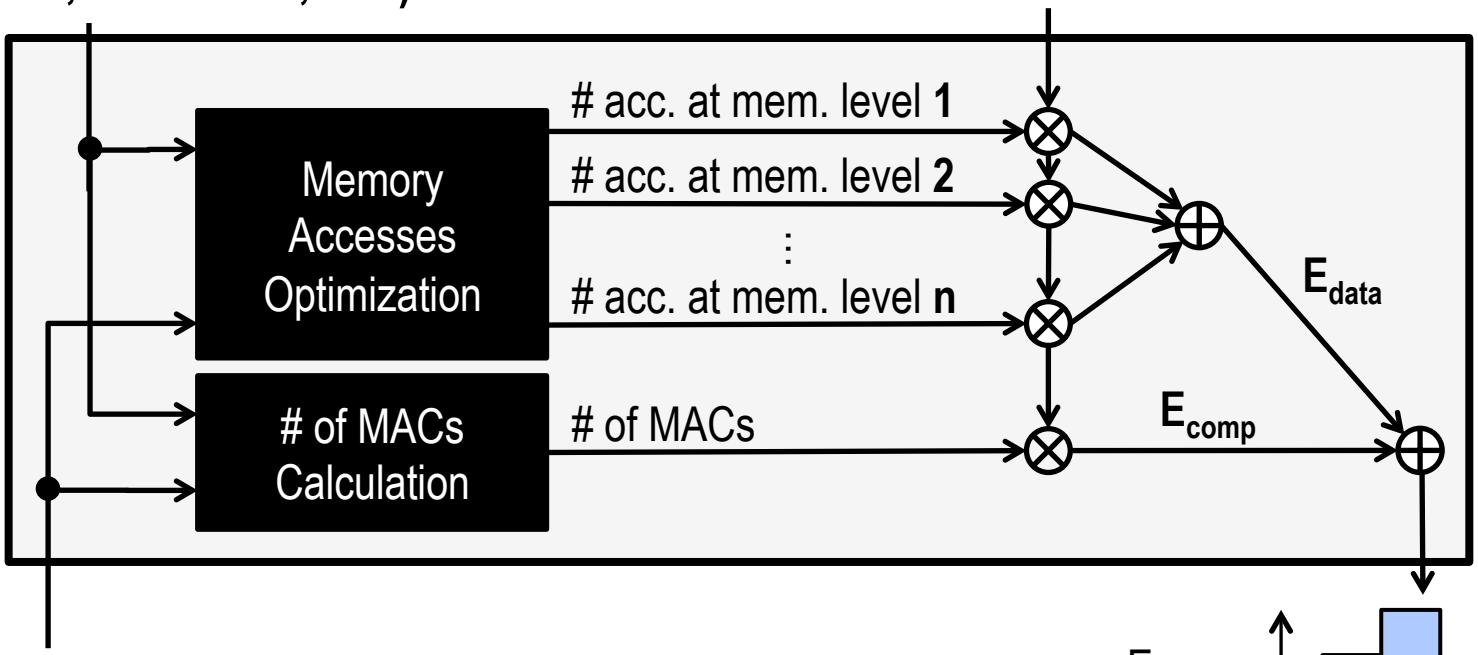
- Accuracy → Measured on Dataset
- Speed → Number of MACs
- Storage Footprint → Number of Weights
- Energy → ?

Energy-Evaluation Methodology



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

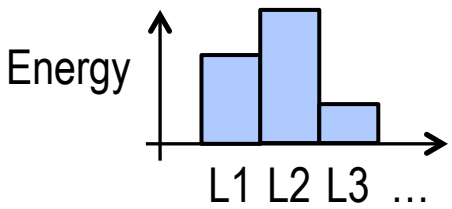
Hardware Energy Costs of each MAC and Memory Access



CNN Weights and Input Data

[0.3, 0, -0.4, 0.7, 0, 0, 0.1, …]

[Yang et al., CVPR 2017]

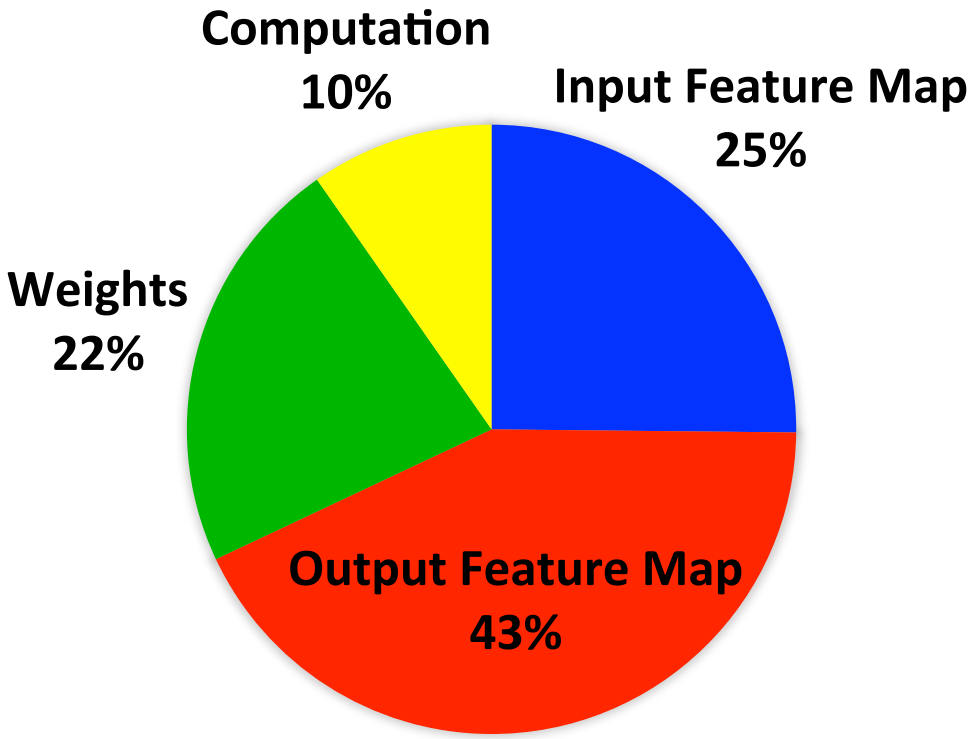


CNN Energy Consumption

Key Observations

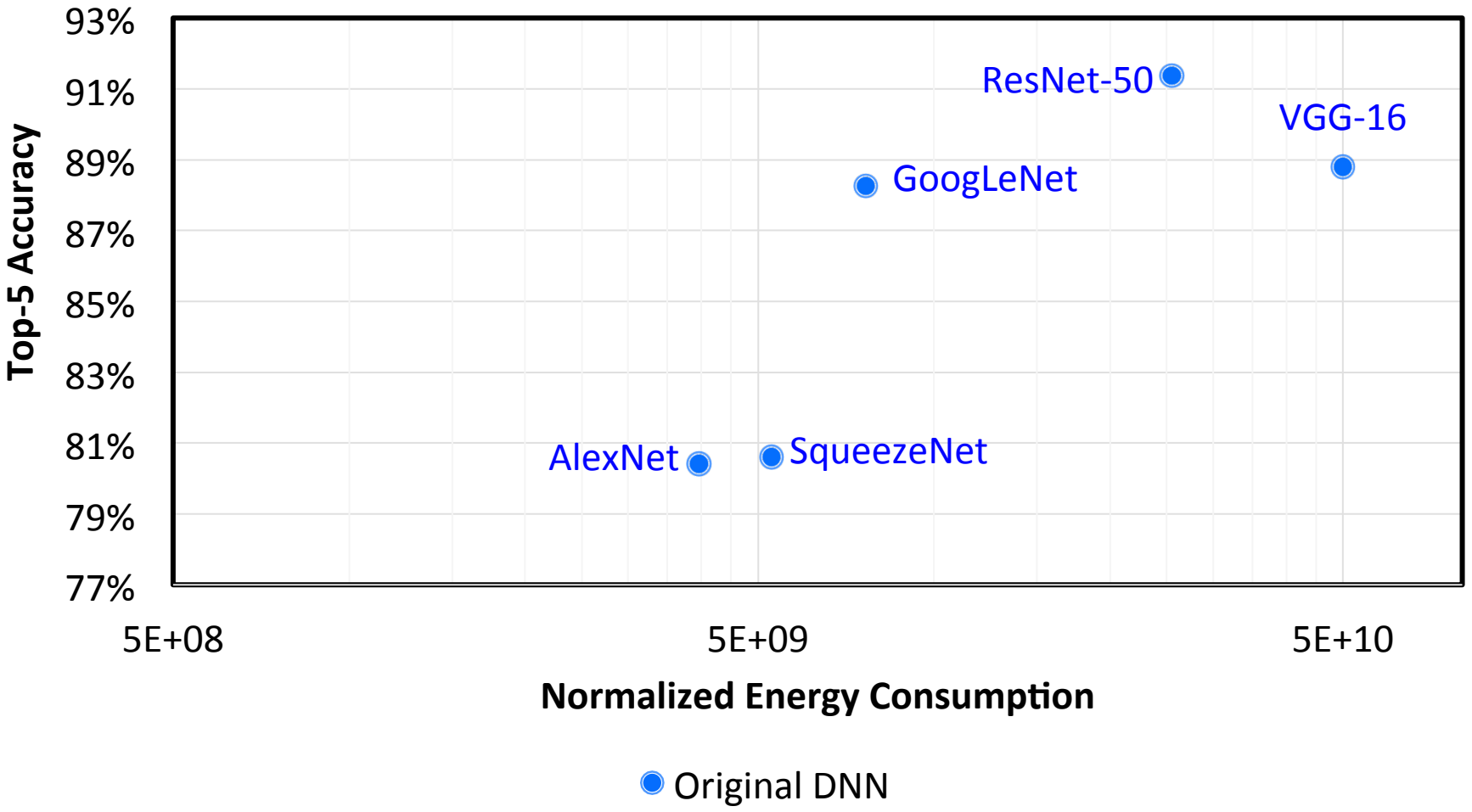
- Number of weights ***alone*** is not a good metric for energy
- **All data types** should be considered

Energy Consumption of GoogLeNet



[Yang et al., CVPR 2017]

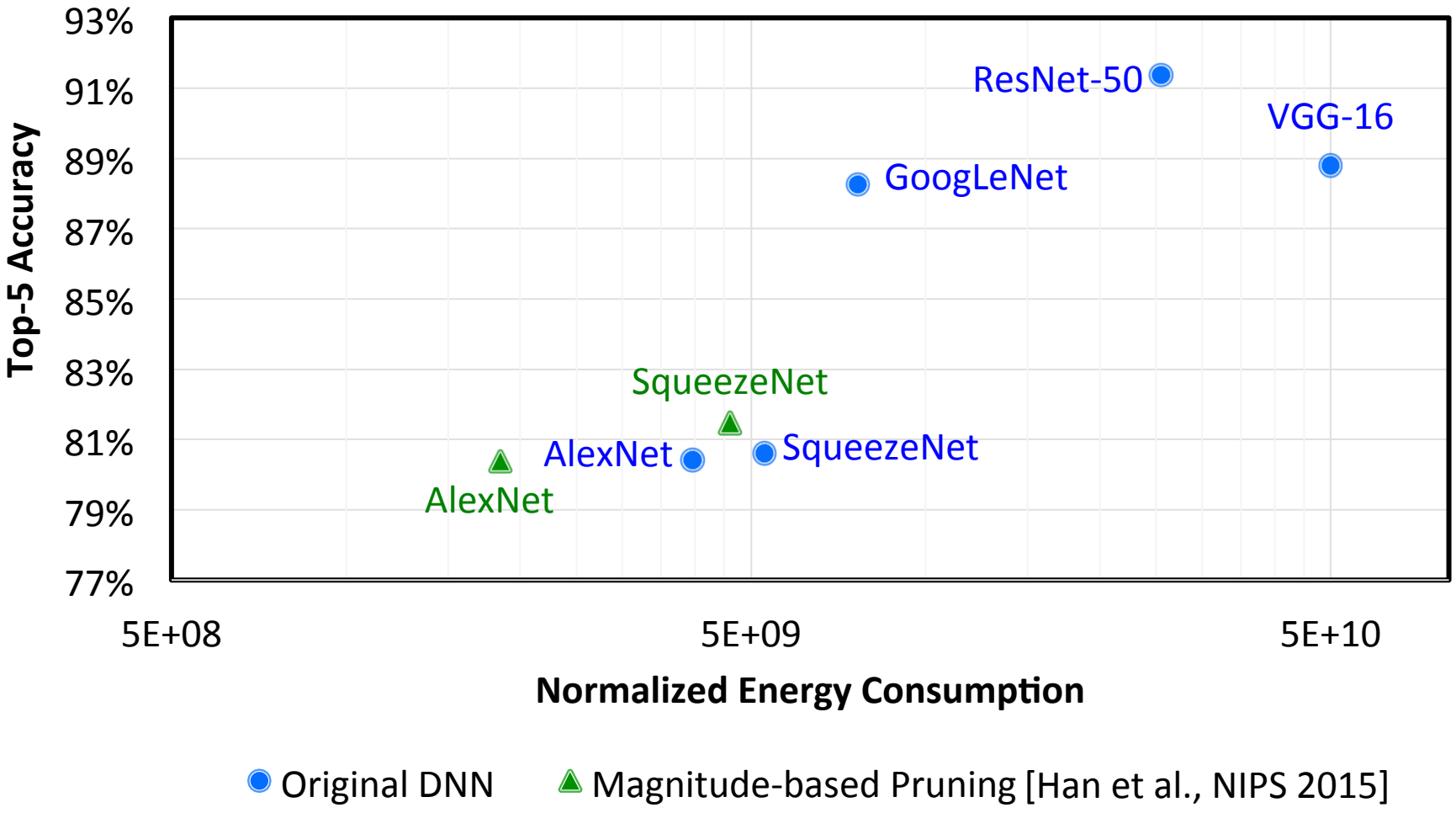
Energy Consumption of Existing DNNs



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

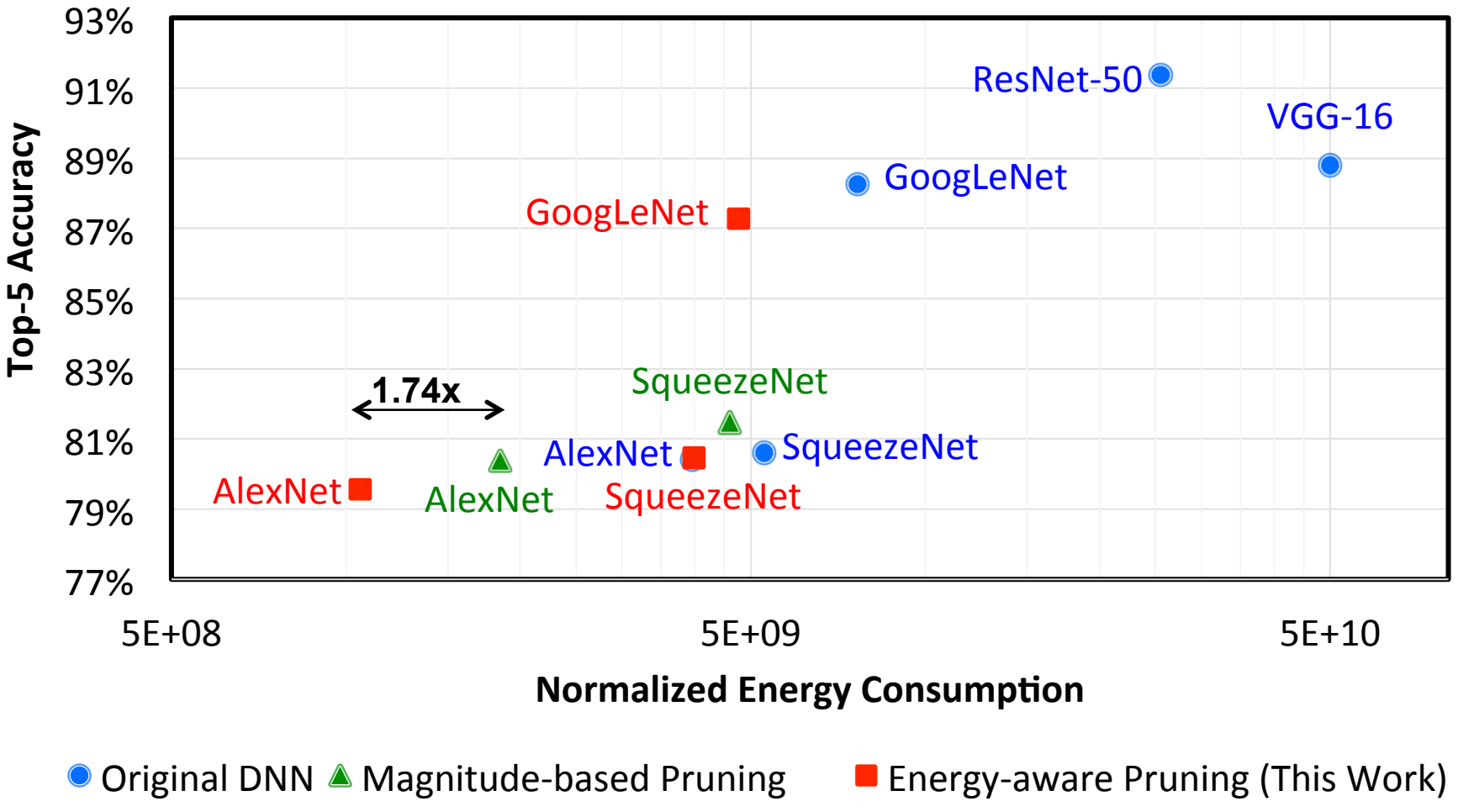
[Yang et al., CVPR 2017]

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]

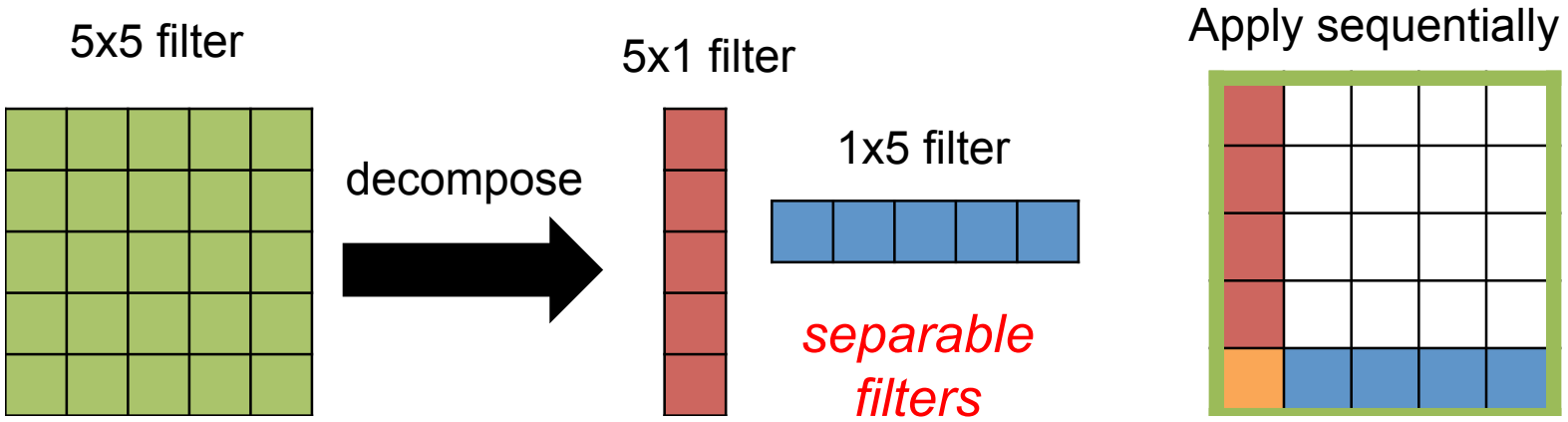
Compact Network Architectures

- **Break large convolutional layers into a series of smaller convolutional layers**
  - **Fewer weights, but same effective receptive field**
- **Before Training: Network Architecture Design**
- **After Training: Decompose Trained Filters**

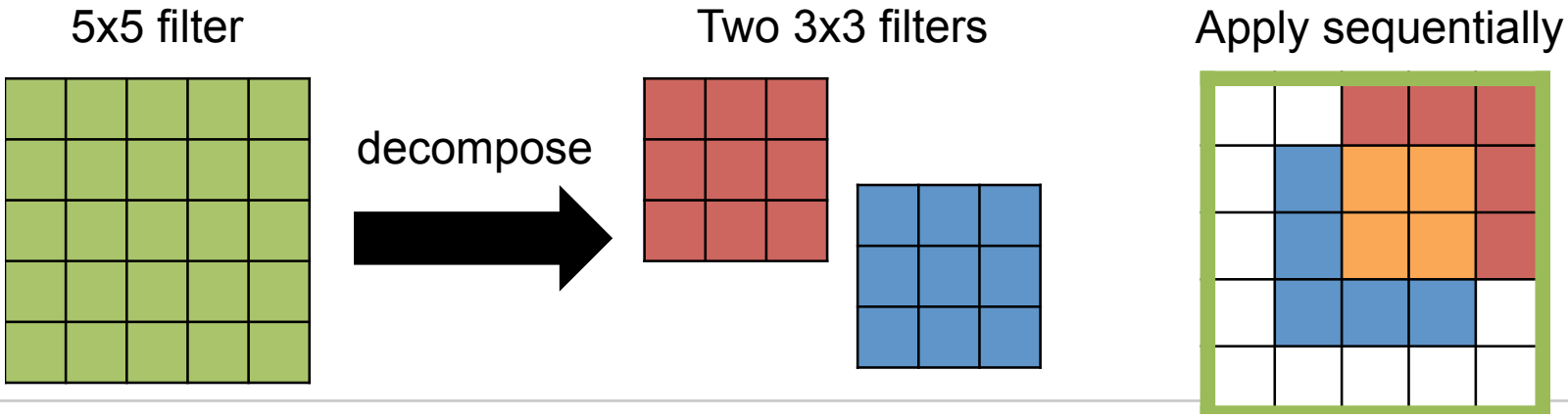
Network Architecture Design

Build Network with series of Small Filters

GoogleNet/Inception v3



VGG-16



Network Architecture Design

Reduce size and computation with 1x1 Filter (**bottleneck**)

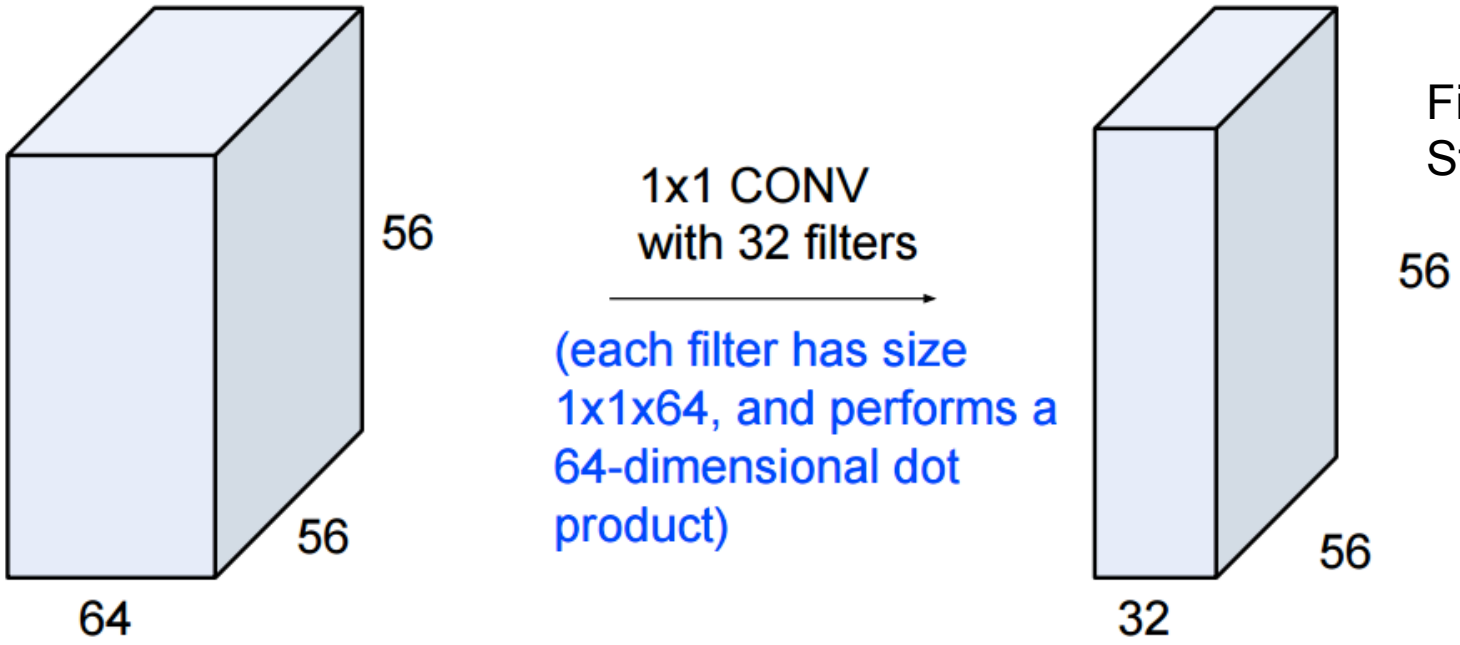


Figure Source:
Stanford cs231n

Used in Network In Network(NiN) and GoogLeNet

[Lin et al., ArXiV 2013 / ICLR 2014] [Szegedy et al., ArXiV 2014 / CVPR 2015]

Network Architecture Design

Reduce size and computation with 1x1 Filter (**bottleneck**)

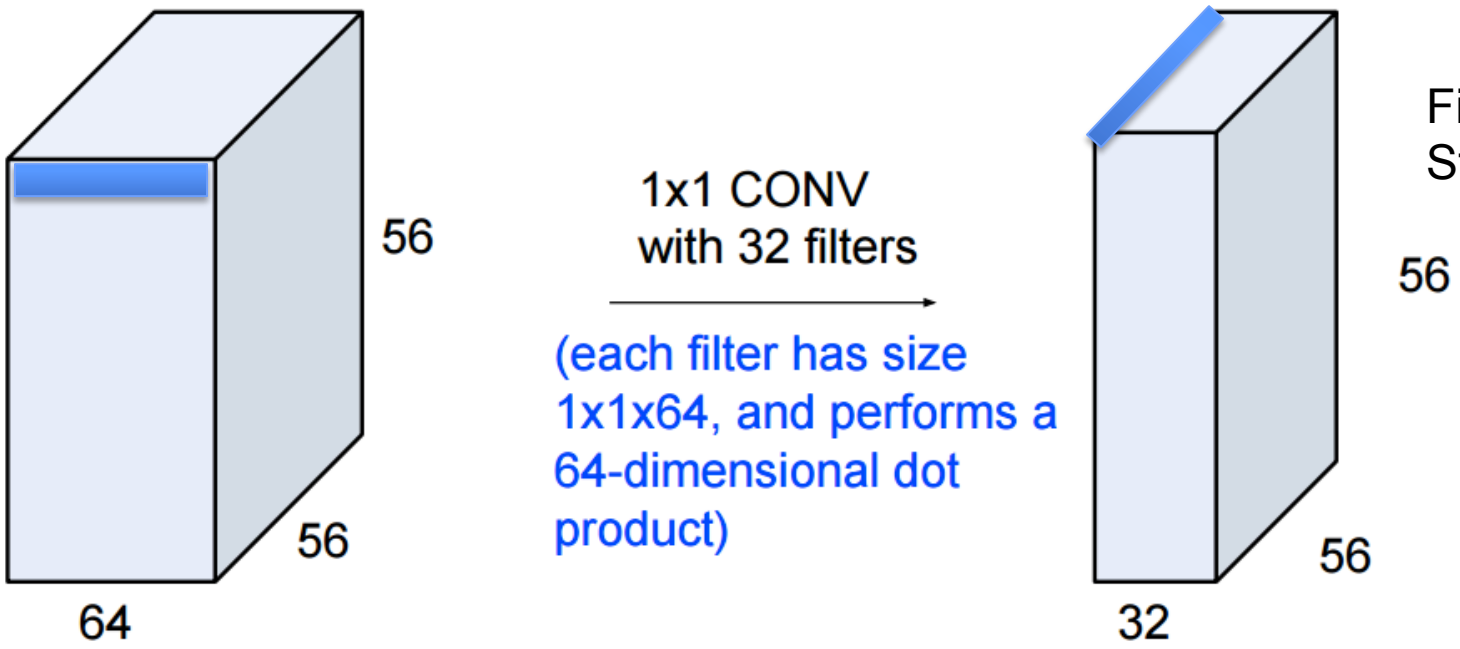


Figure Source:
Stanford cs231n

Used in Network In Network(NiN) and GoogLeNet

[Lin et al., ArXiV 2013 / ICLR 2014] [Szegedy et al., ArXiV 2014 / CVPR 2015]

Network Architecture Design

Reduce size and computation with 1x1 Filter (**bottleneck**)

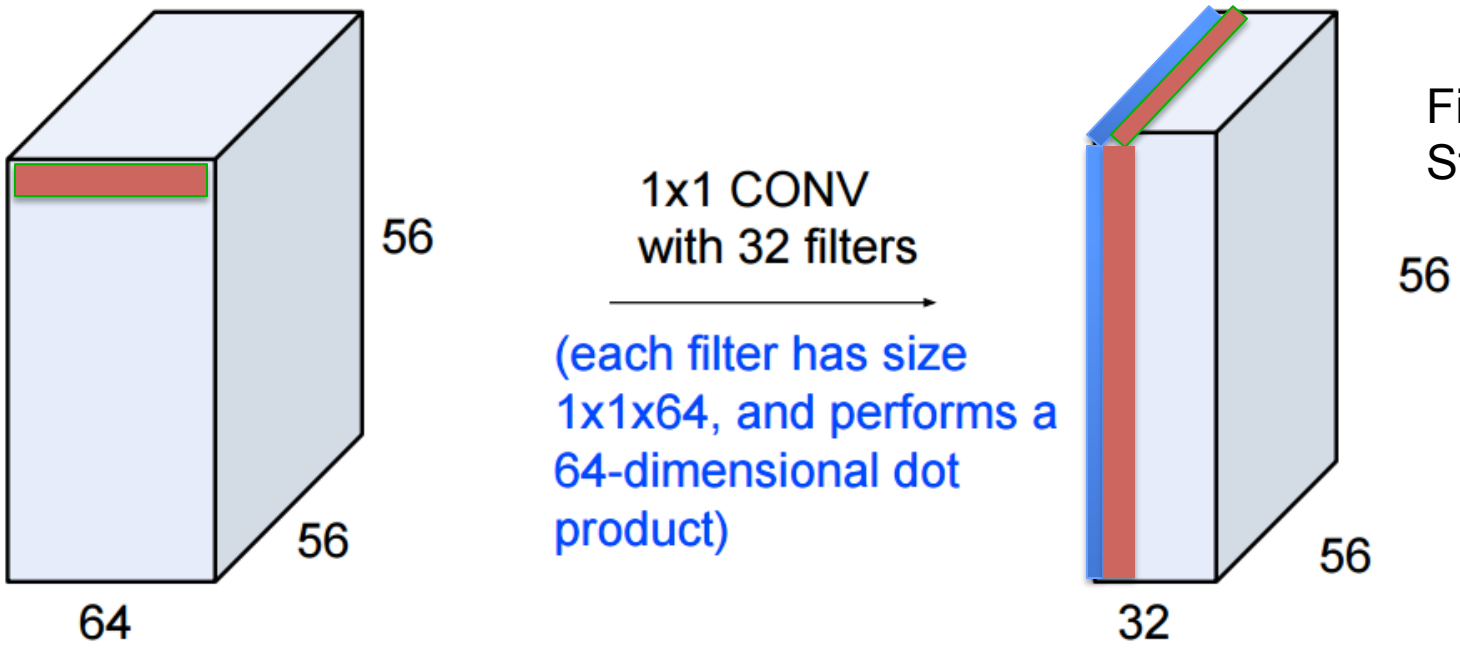


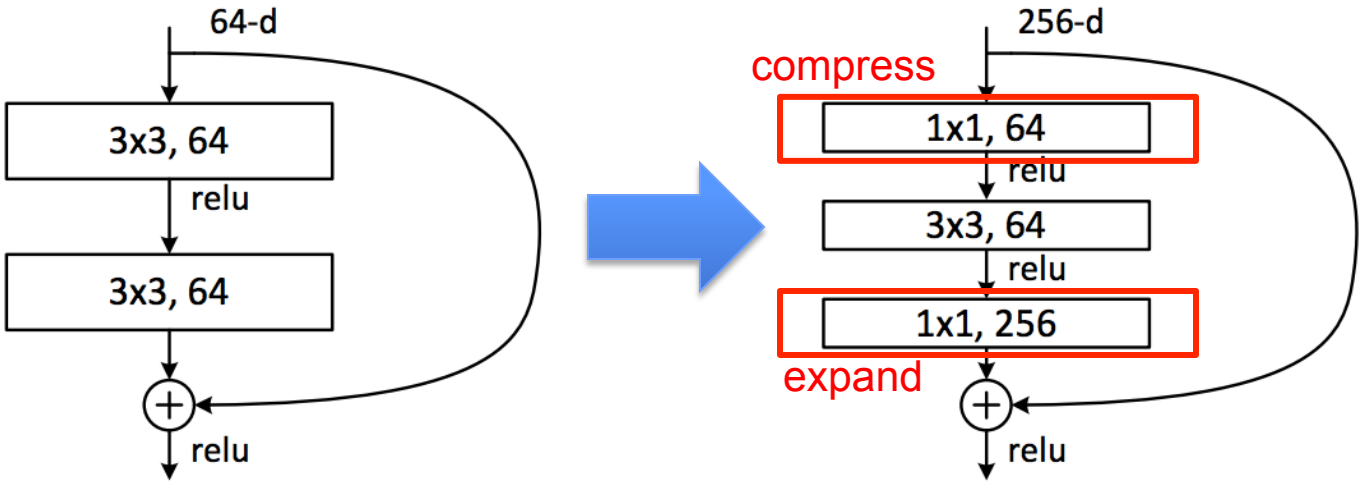
Figure Source:
Stanford cs231n

Used in Network In Network(NiN) and GoogLeNet

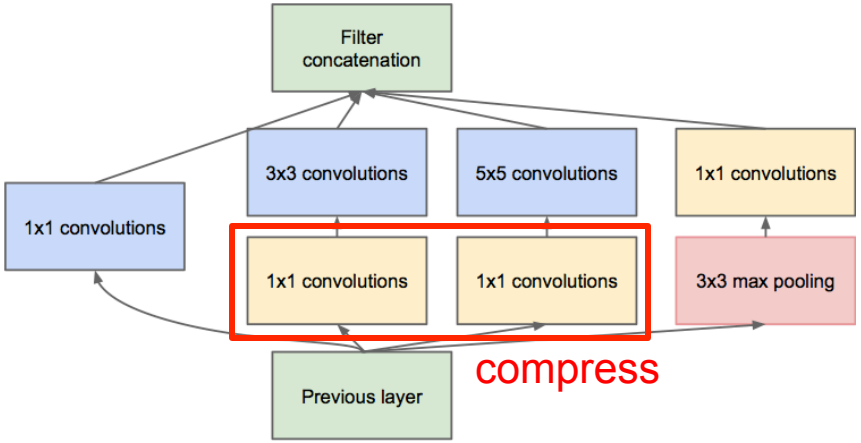
[Lin et al., ArXiV 2013 / ICLR 2014] [Szegedy et al., ArXiV 2014 / CVPR 2015]

Bottleneck in Popular DNN models

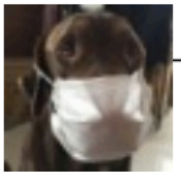
ResNet

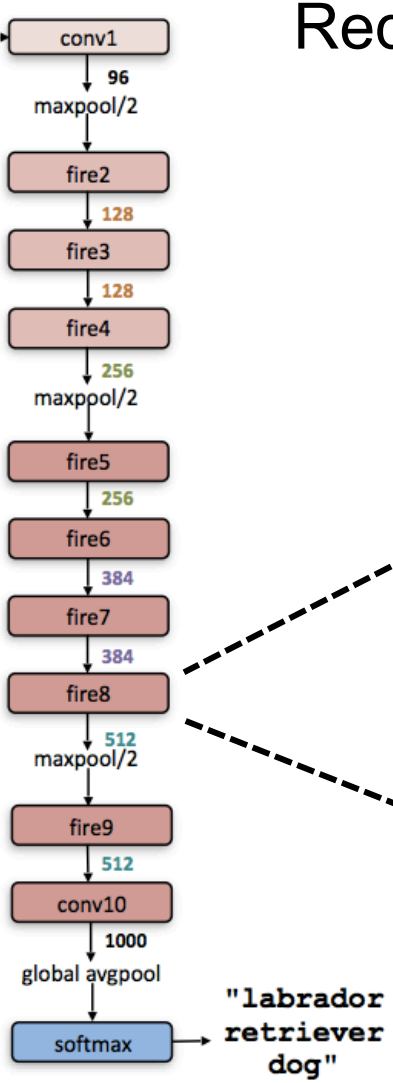


GoogleNet



SqueezeNet

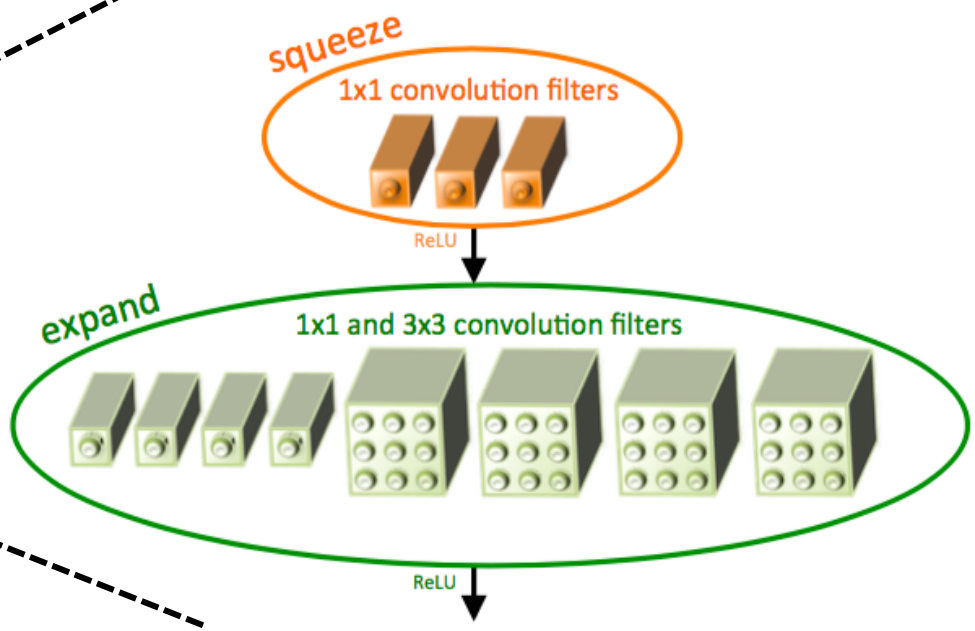




Reduce weights by reducing number of input channels by "squeezing" with 1x1

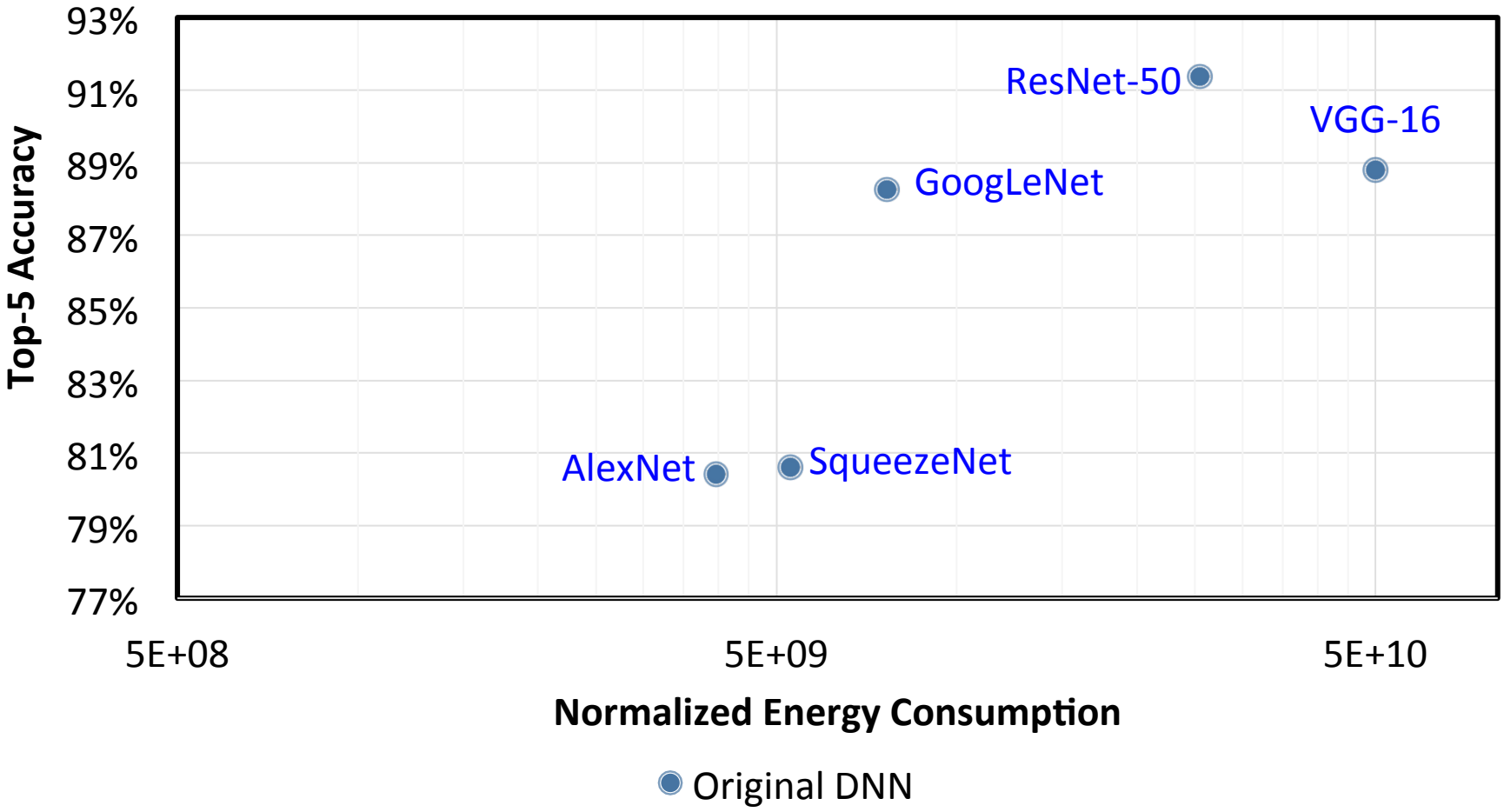
50x fewer weights than AlexNet (no accuracy loss)

Fire Module



[F.N. Iandola et al., ArXiv, 2016]

Energy Consumption of Existing DNNs

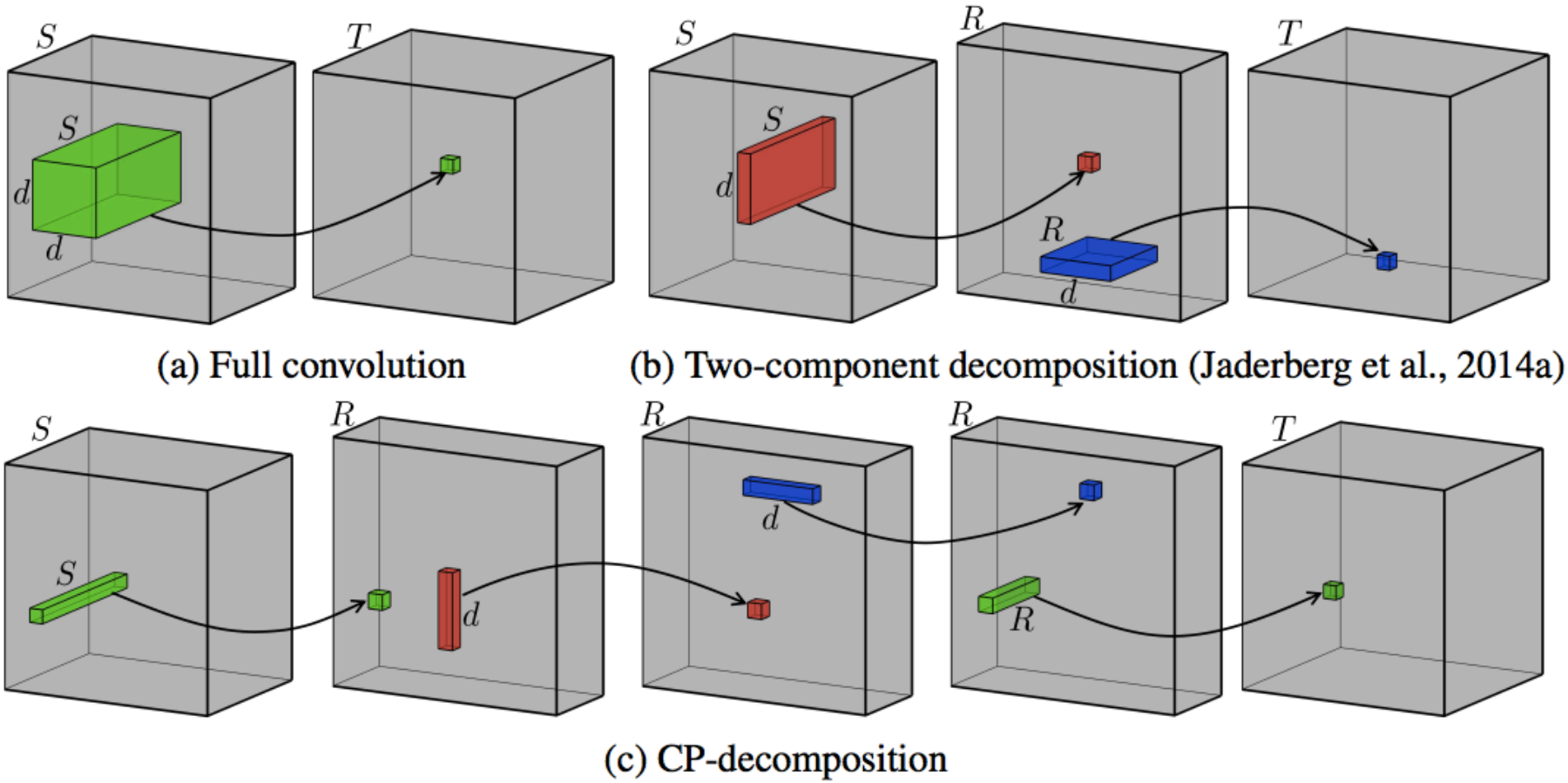


Deeper CNNs with fewer weights do not necessarily consume less energy than shallower CNNs with more weights

[Yang et al., CVPR 2017]

Decompose Trained Filters

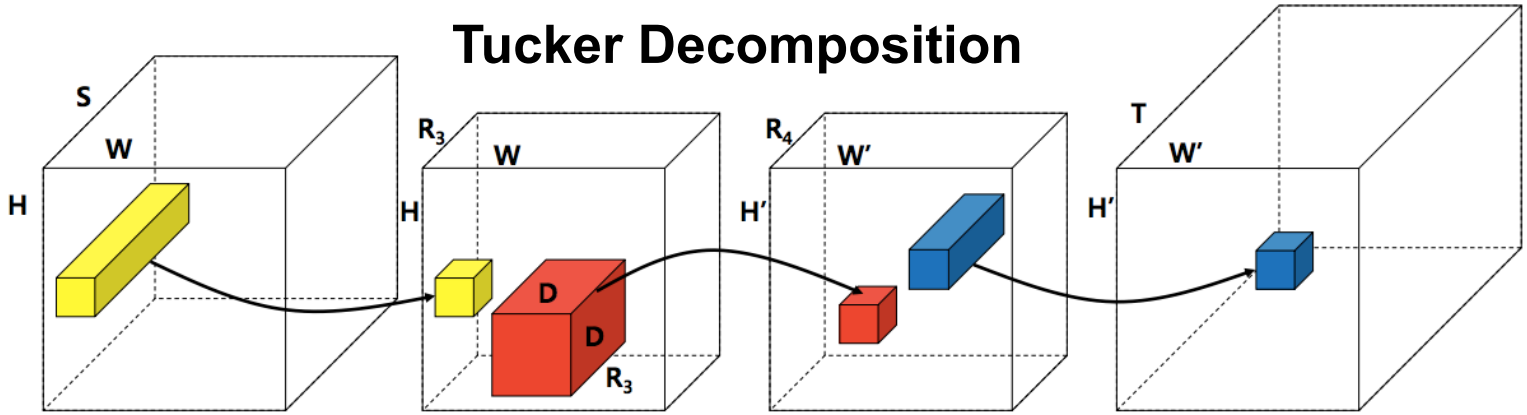
After training, perform **low-rank approximation by applying tensor decomposition** to weight kernel; then **fine-tune** weights for accuracy



R = canonical rank

[Lebedev et al., ICLR 2015]

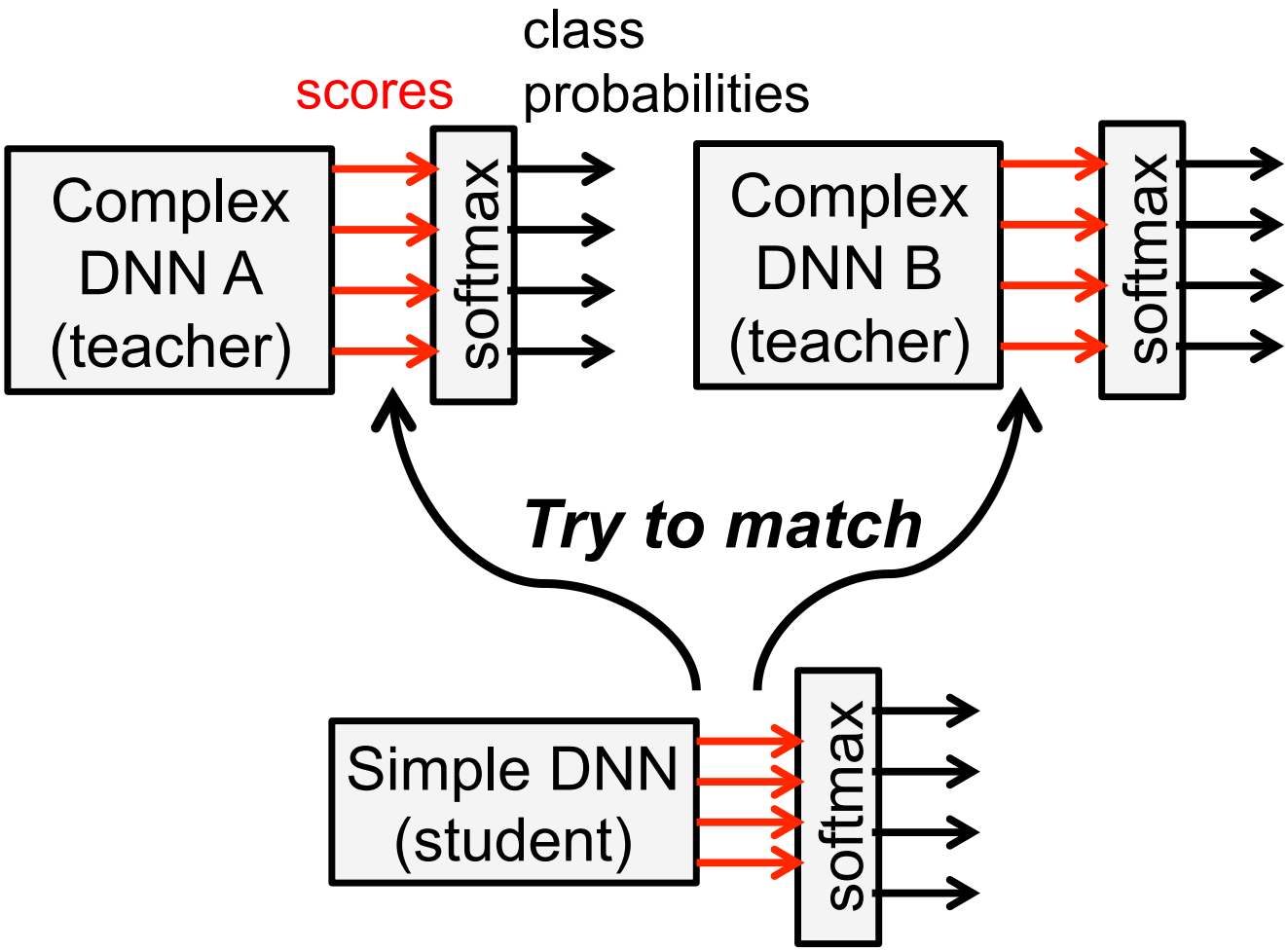
Decompose Trained Filters on Phone



Model	Top-5	Weights	FLOPs	S6		Titan X
AlexNet	80.03	61M	725M	117ms	245mJ	0.54ms
*AlexNet**	78.33	11M	272M	43ms	72mJ	0.30ms
(imp.)	(-1.70)	(×5.46)	(×2.67)	(×2.72)	(×3.41)	(×1.81)
VGG-S	84.60	103M	2640M	357ms	825mJ	1.86ms
*VGG-S**	84.05	14M	549M	97ms	193mJ	0.92ms
(imp.)	(-0.55)	(×7.40)	(×4.80)	(×3.68)	(×4.26)	(×2.01)
GoogLeNet	88.90	6.9M	1566M	273ms	473mJ	1.83ms
*GoogLeNet**	88.66	4.7M	760M	192ms	296mJ	1.48ms
(imp.)	(-0.24)	(×1.28)	(×2.06)	(×1.42)	(×1.60)	(×1.23)
VGG-16	89.90	138M	15484M	1926ms	4757mJ	10.67ms
*VGG-16**	89.40	127M	3139M	576ms	1346mJ	4.58ms
(imp.)	(-0.50)	(×1.09)	(×4.93)	(×3.34)	(×3.53)	(×2.33)

[Kim et al., ICLR 2016]

Knowledge Distillation



[Bucilu et al., KDD 2006],[Hinton et al., arXiv 2015]

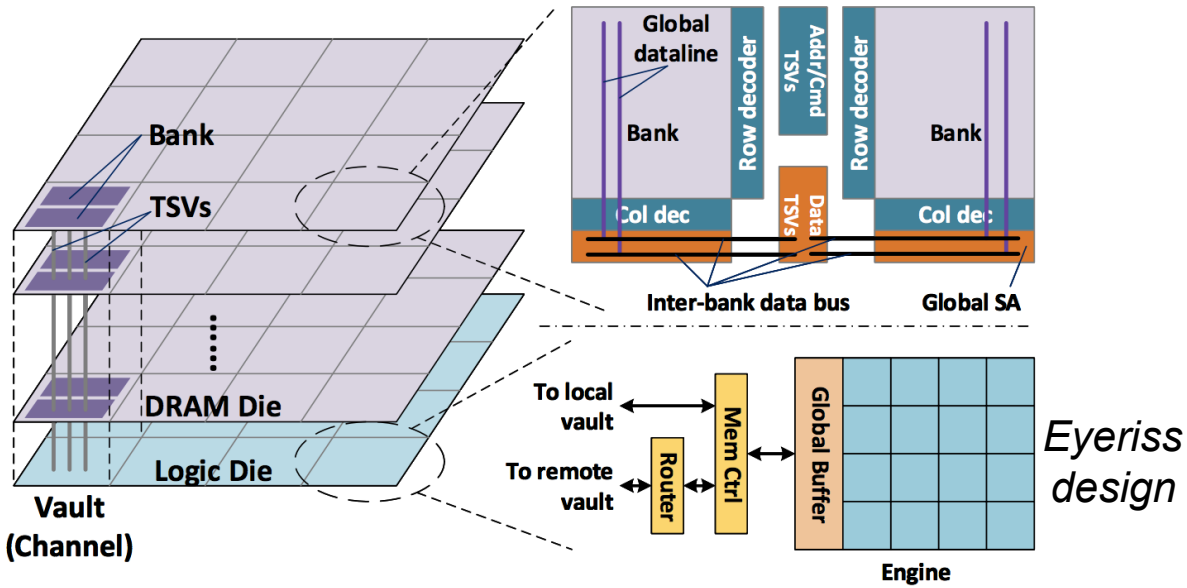
Opportunities in Advanced Technologies

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



Eyeriss design

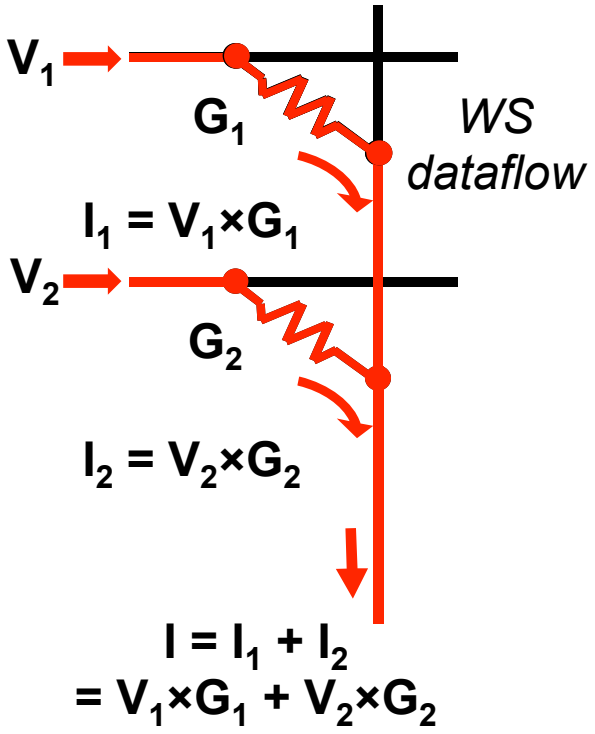
[Gao et al., Tetris, ASPLOS 2017]

[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]

Summary

- **Deep 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

Overview Paper

V. Sze, Y.-H. Chen, T-J. Yang, J. Emer, "*Efficient Processing of Deep Neural Networks: A Tutorial and Survey*", arXiv, 2017 https://arxiv.org/pdf/1703.09039.pdf

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

New Class Fall 2017

"6.S082/6.888 Hardware Architecture for Deep Learning"

For updates Follow @eems_mit

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