embedded VISION SUMMIT 2018

Approaches for Energy Efficient Implementation of Deep Neural Networks

Massachusetts Institute of Technology

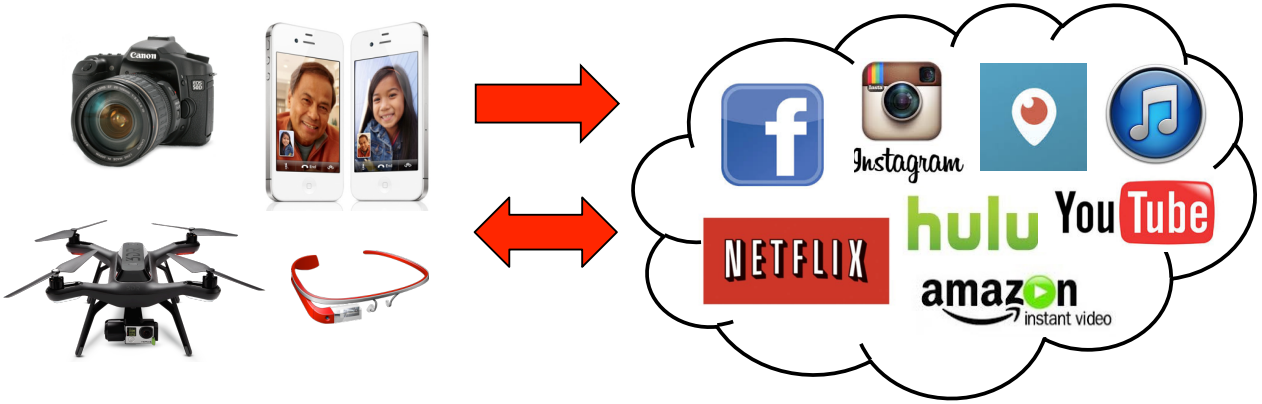
In collaboration with Yu-Hsin Chen, Joel Emer, Tien-Ju Yang

Vivienne Sze

May 23, 2018

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**

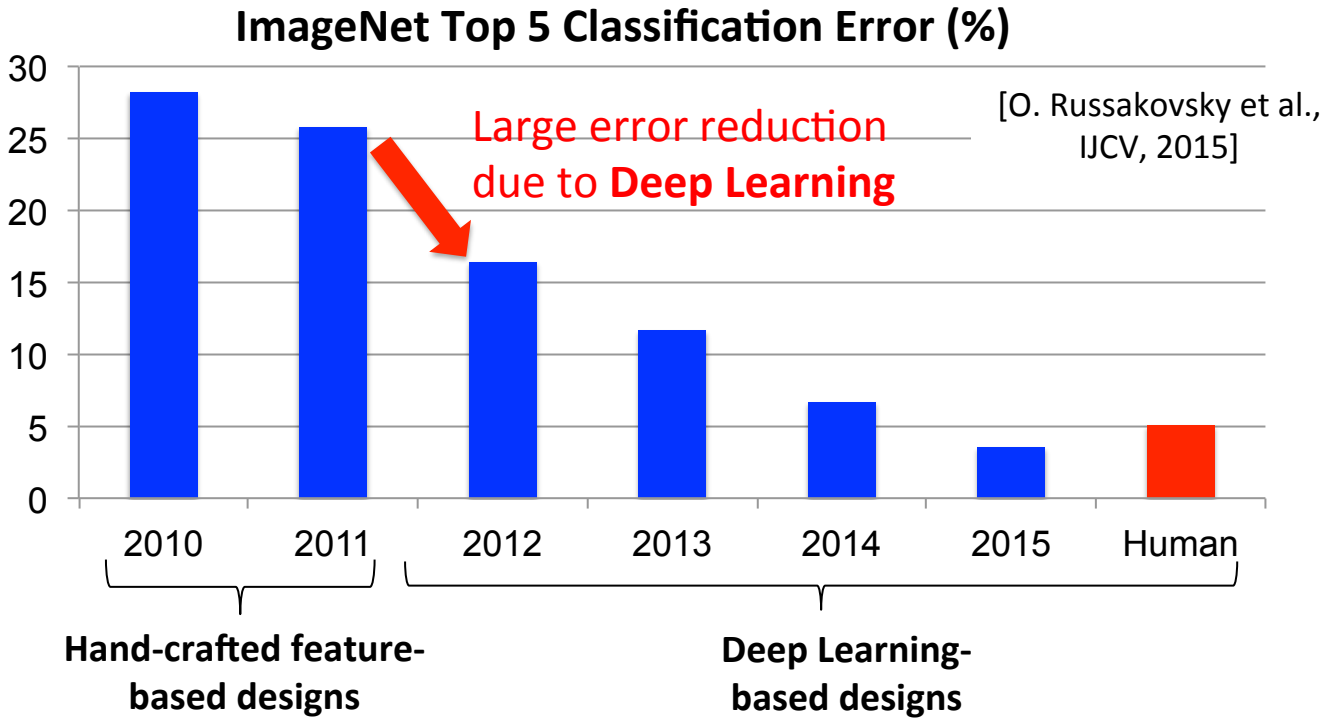


Energy limited due to battery capacity

Power limited due to heat dissipation

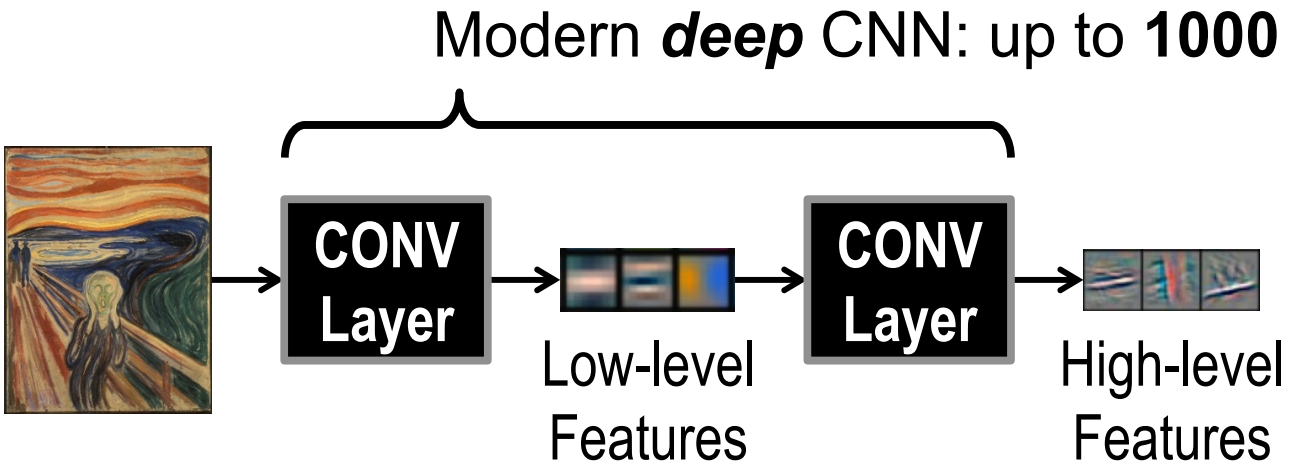
Need energy-efficient pixel processing!

Increased Accuracy with Deep Learning

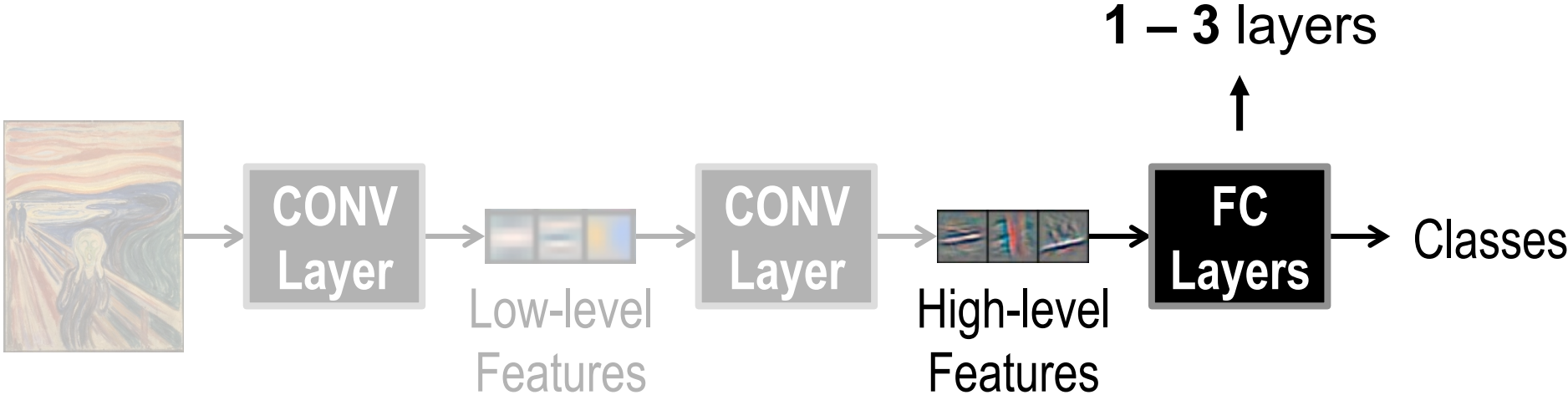


Deep Learning requires significantly more computation than previous approaches

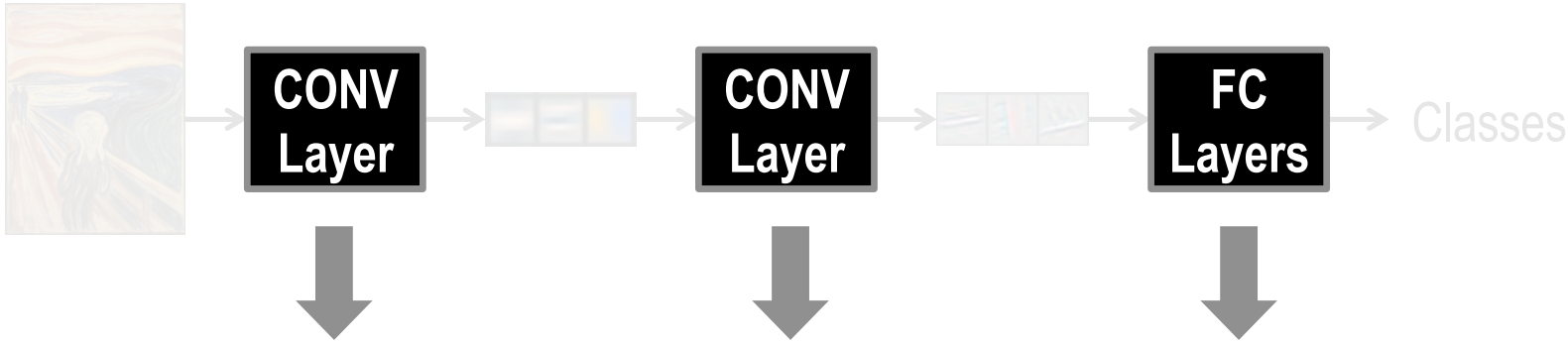
Deep Convolutional Neural Networks

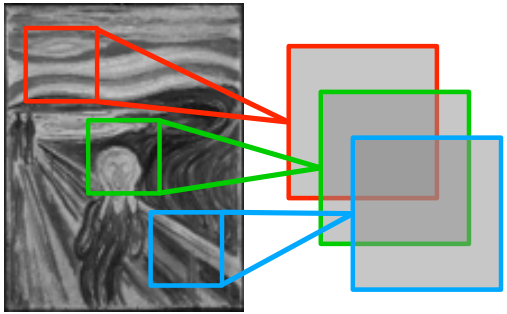


Deep Convolutional Neural Networks



Deep Convolutional Neural Networks



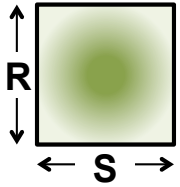


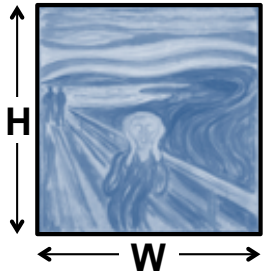
Convolutions account for more than 90% of overall computation, dominating **runtime** and **energy consumption**

High-Dimensional CNN Convolution

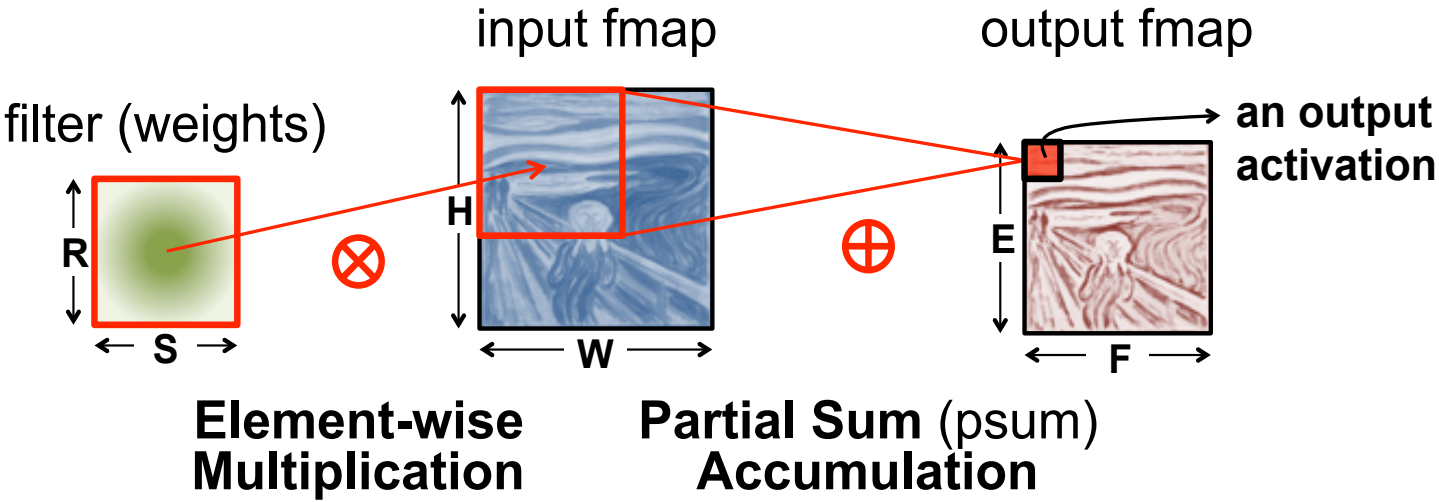
a plane of input activations
a.k.a. **input feature map (fmap)**

filter (weights)

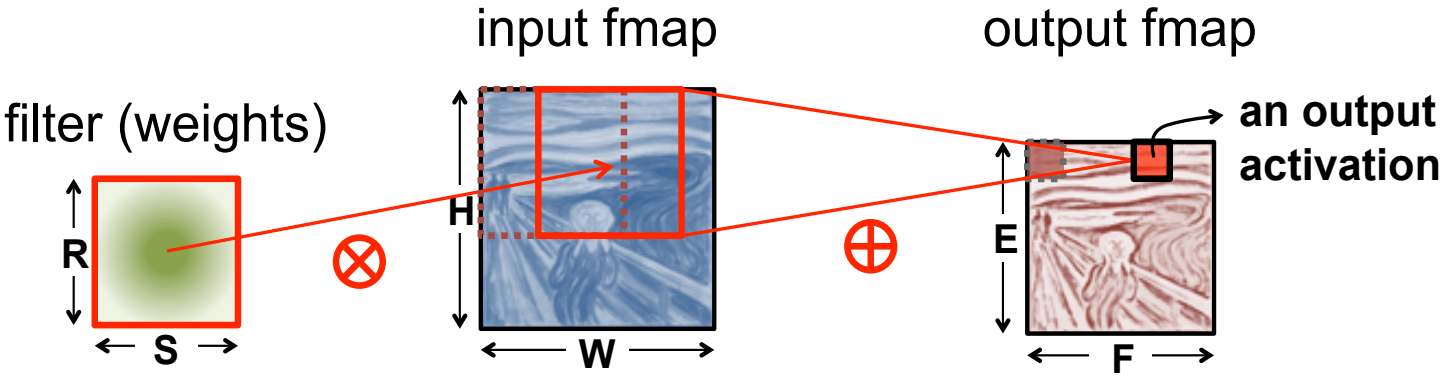




High-Dimensional CNN Convolution

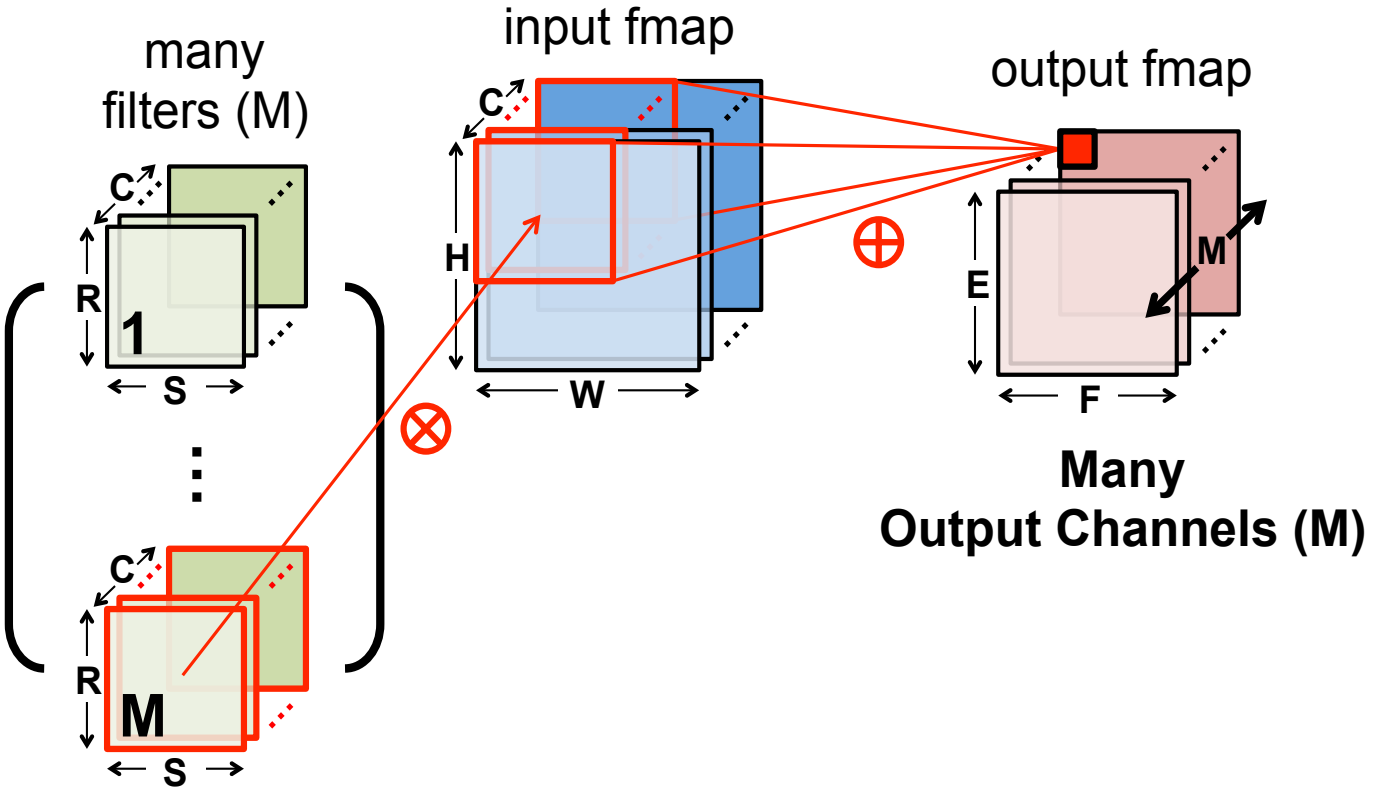


High-Dimensional CNN Convolution



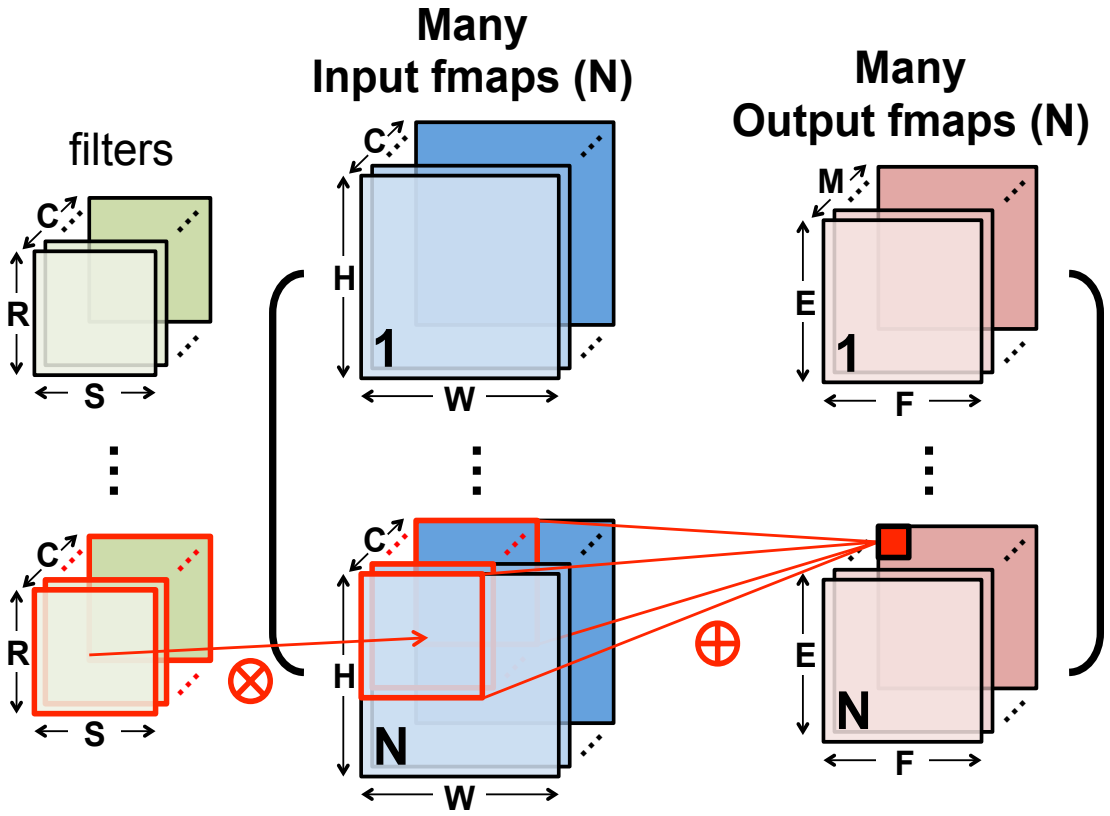
Sliding Window Processing

High-Dimensional CNN Convolution



High-Dimensional CNN Convolution

Image batch size: 1 – 256 (N)



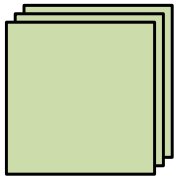
Large Size with Varying Shapes

AlexNet 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

[Krizhevsky, NIPS 2012]

Layer 1

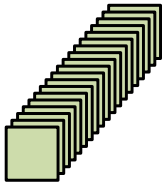


34k Params
105M MACs

Layer 2

307k Params
224M MACs

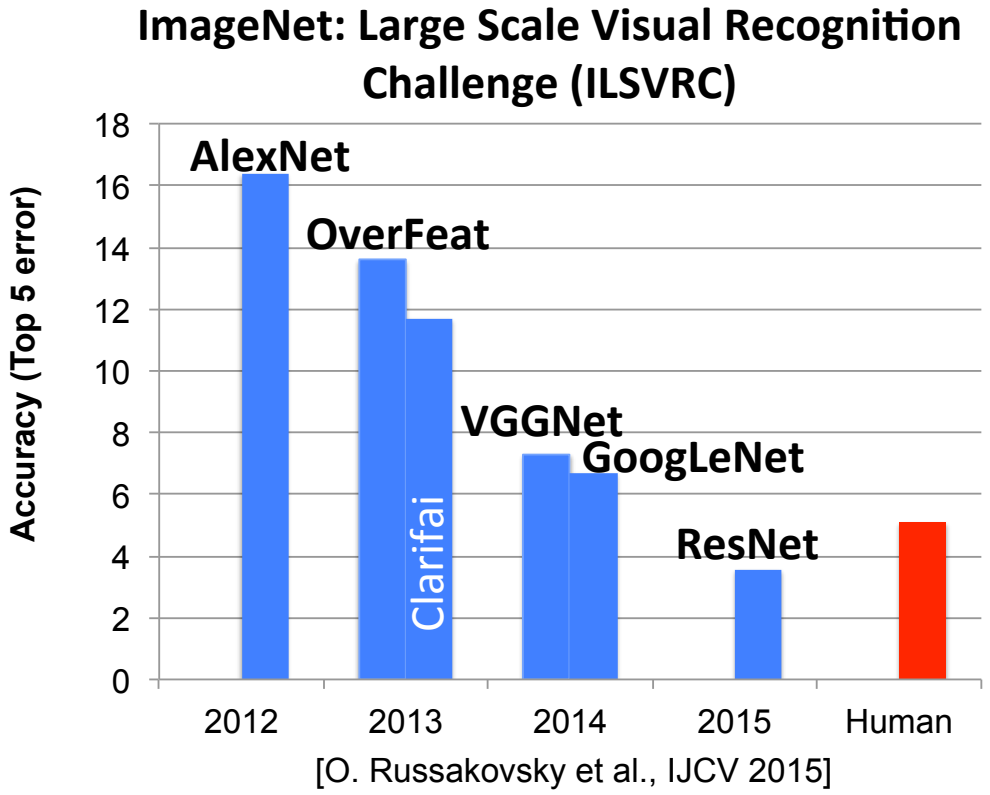
Layer 3



885k Params
150M MACs

Popular DNNs

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



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**
# of Weights	2.6k	2.3M	14.7M	6.0M	23.5M
# of MACs	283k	666M	15.3G	1.43G	3.86G
# of FC layers	**2**	**3**	**3**	**1**	**1**
# of Weights	58k	58.6M	124M	1M	2M
# of MACs	58k	58.6M	124M	1M	2M
Total Weights	**60k**	**61M**	**138M**	**7M**	**25.5M**
Total MACs	**341k**	**724M**	**15.5G**	**1.43G**	**3.9G**

CONV Layers increasingly important!

Training versus Inference

Training
(determine weights)

Weights

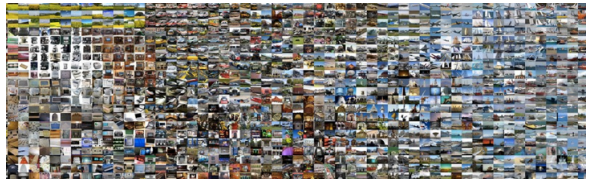
Inference
(use weights)

Key Metrics

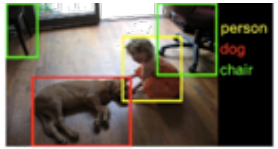
- Accuracy
  - Well defined dataset, DNN Model and task
- Programmability
  - Support various DNN Models with different filter weights
- Energy/Power:
  - Energy per operation and DRAM Bandwidth
- Throughput/Latency
  - GOPS, frame rate, delay, batch size
- Cost
  - Area (memory and logic size)

[Sze et al., CICC 2017]

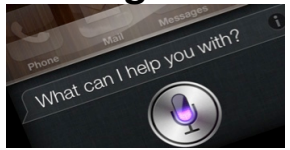
ImageNet



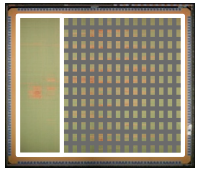
Computer Vision



Speech Recognition



Chip



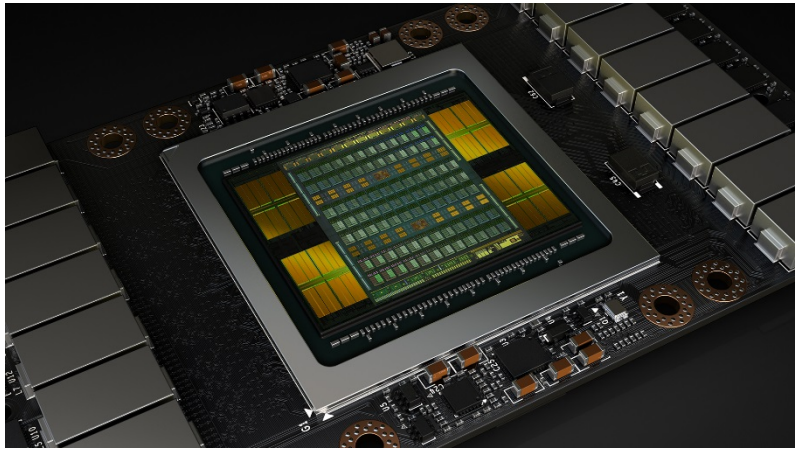
DRAM

GPUs and CPUs Targeting Deep Learning

Intel Knights Landing (2016)
Intel Knights Mills (2017)

Xeon Phi "optimized for deep learning"

Nvidia PASCAL GP100 (2016)
Nvidia VOLTA GV100 (2017)



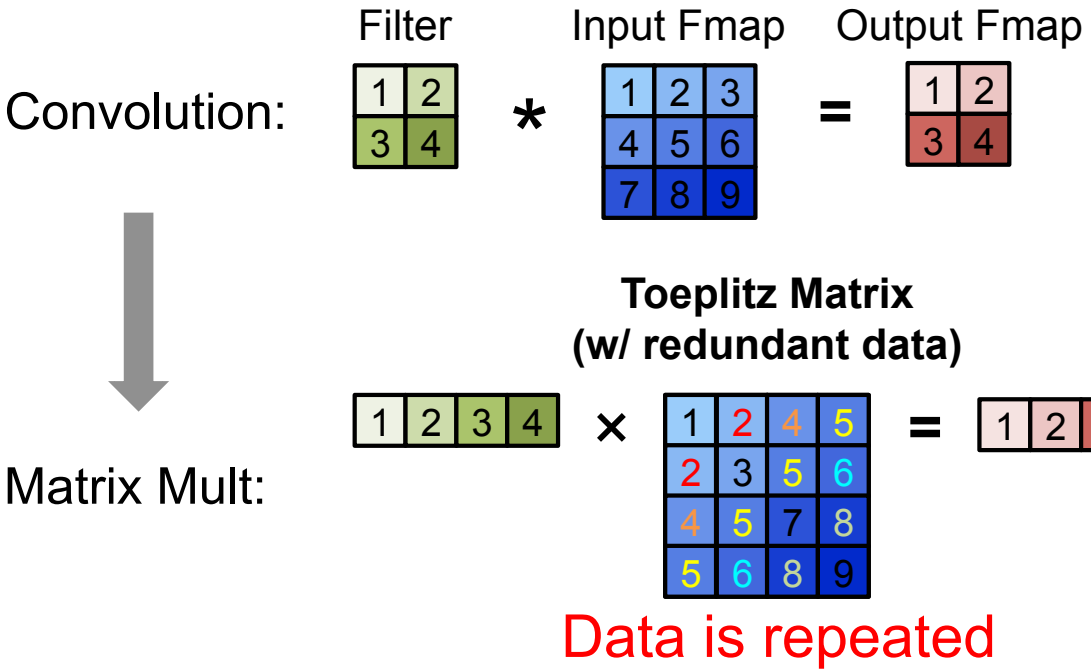
Use **matrix multiplication libraries** on CPUs and GPUs

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**



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

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

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

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

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

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

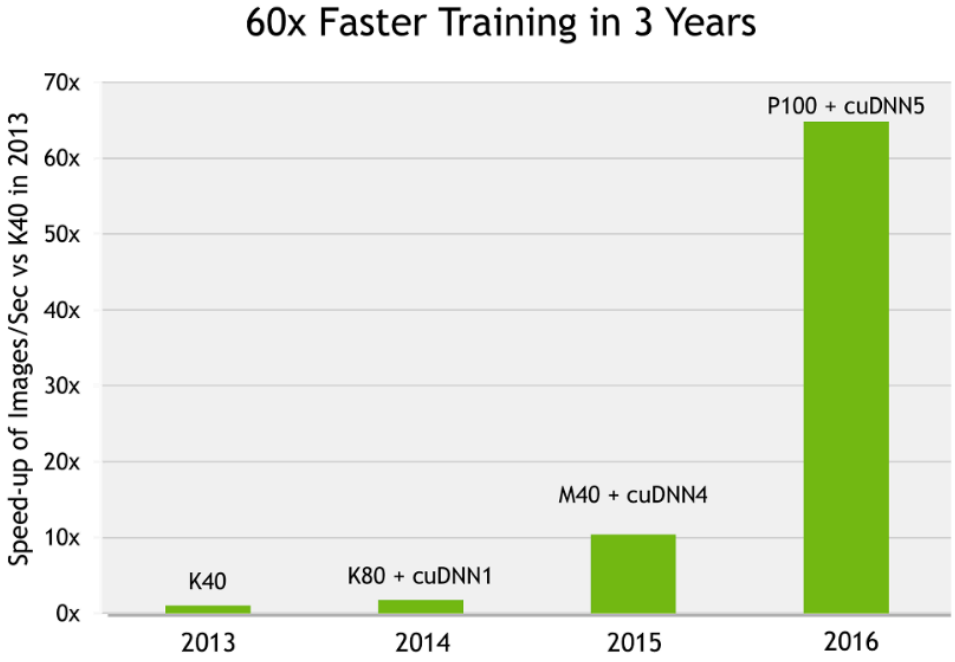
3 multiplications + 5 additions

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

cuDNN: Speed up with Transformations

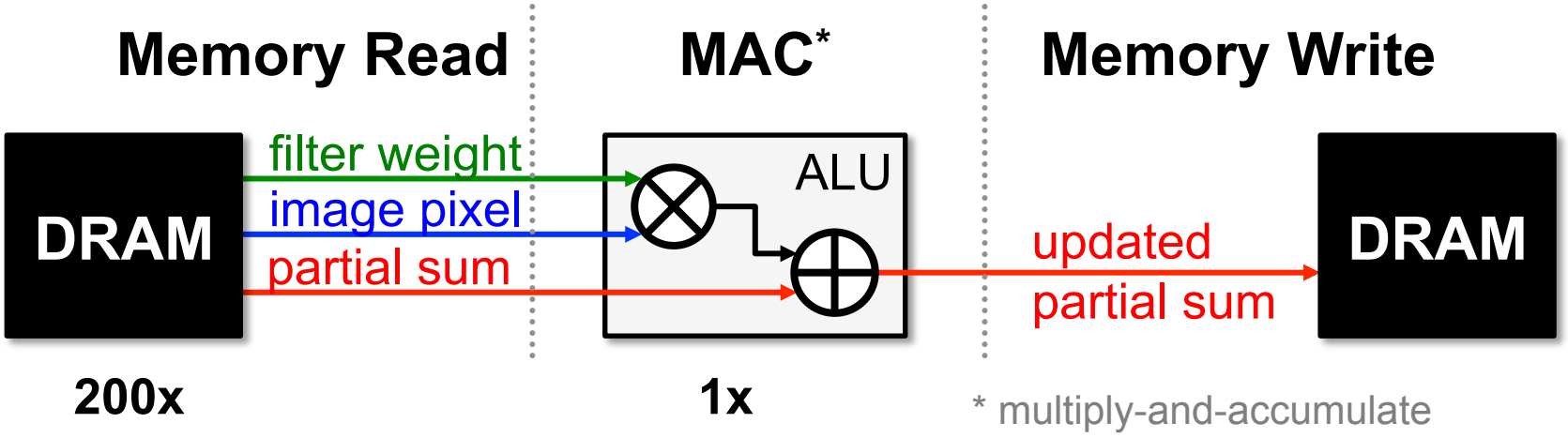


Source: Nvidia

Designing Specialized Hardware (Accelerators) for DNNs

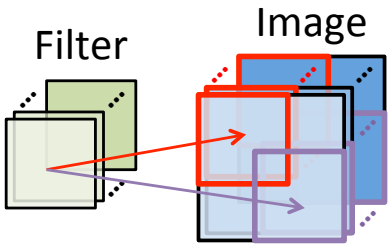
Properties We Can Leverage

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



Properties We Can Leverage

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



Convolutional Reuse
(pixels, weights)

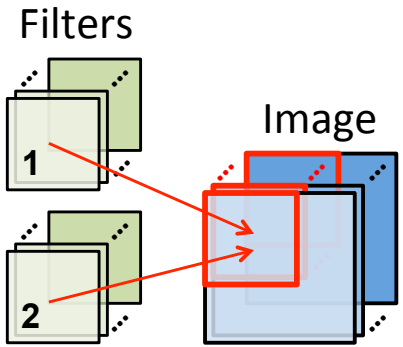
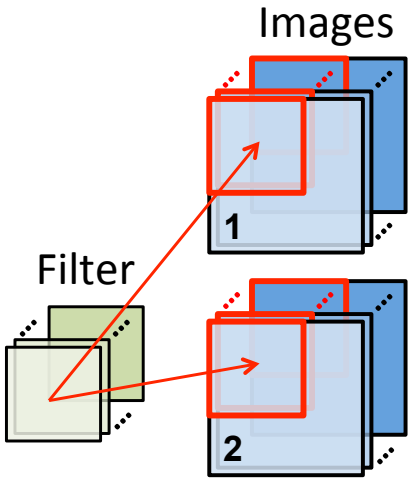


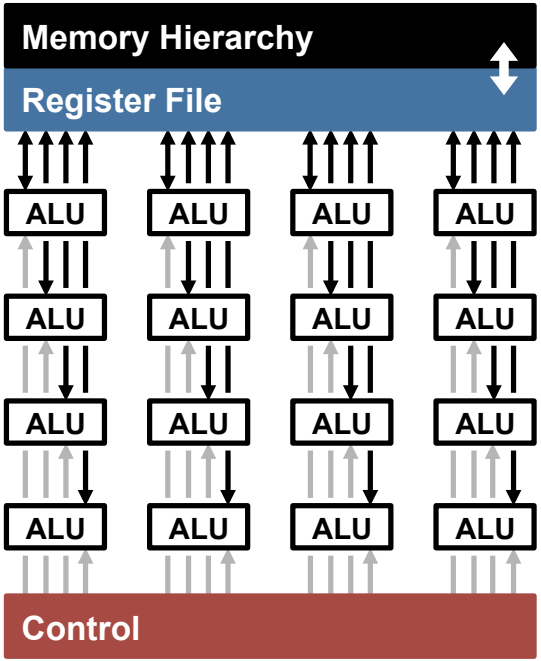
Image Reuse
(pixels)



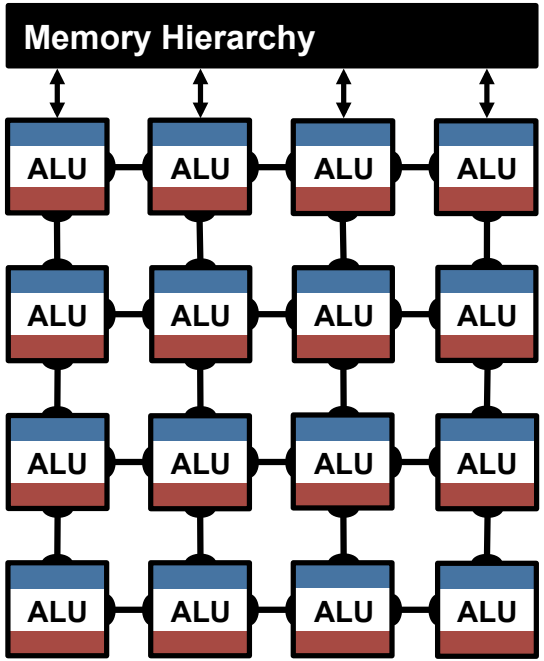
Filter Reuse
(weights)

Highly-Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)



Spatial Architecture (Dataflow Processing)

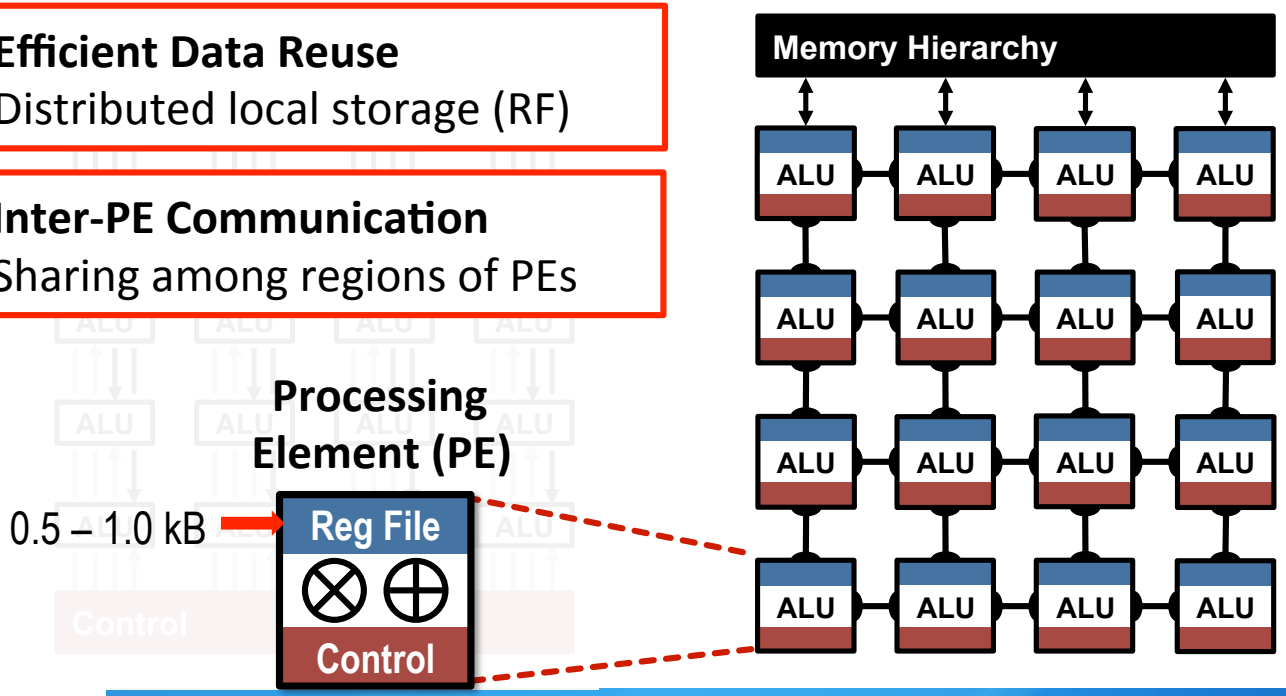


Advantages of Spatial Architecture

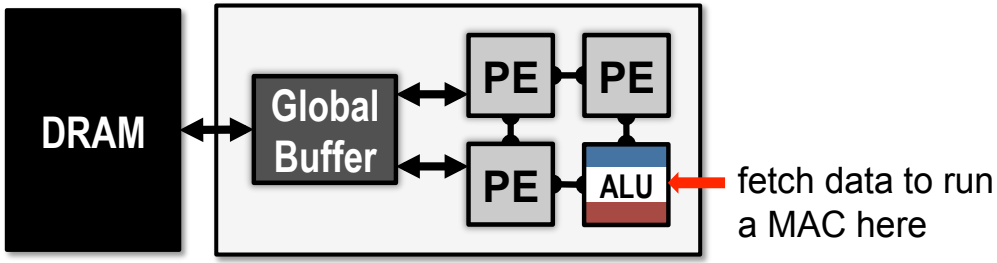
Spatial Architecture (Dataflow Processing)

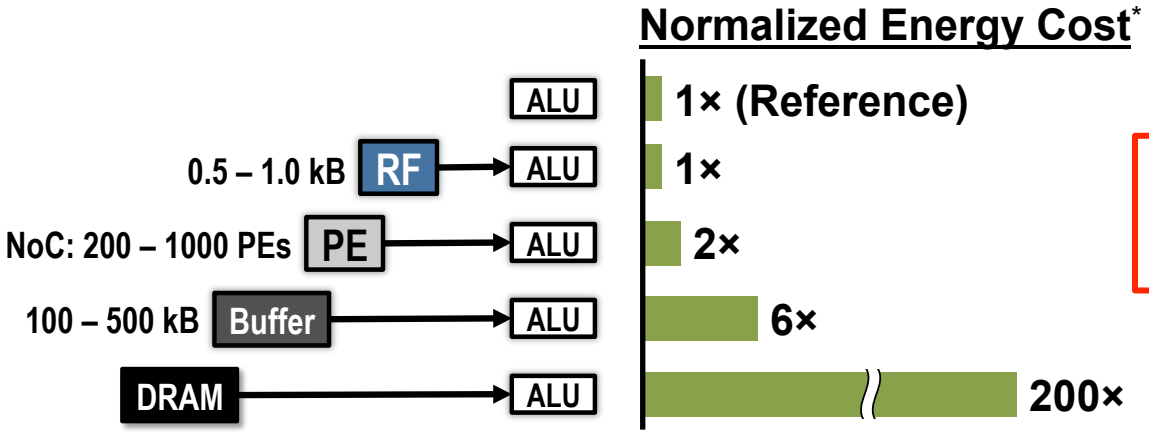
Efficient Data Reuse
Distributed local storage (RF)

Inter-PE Communication
Sharing among regions of PEs



Data Movement is Expensive

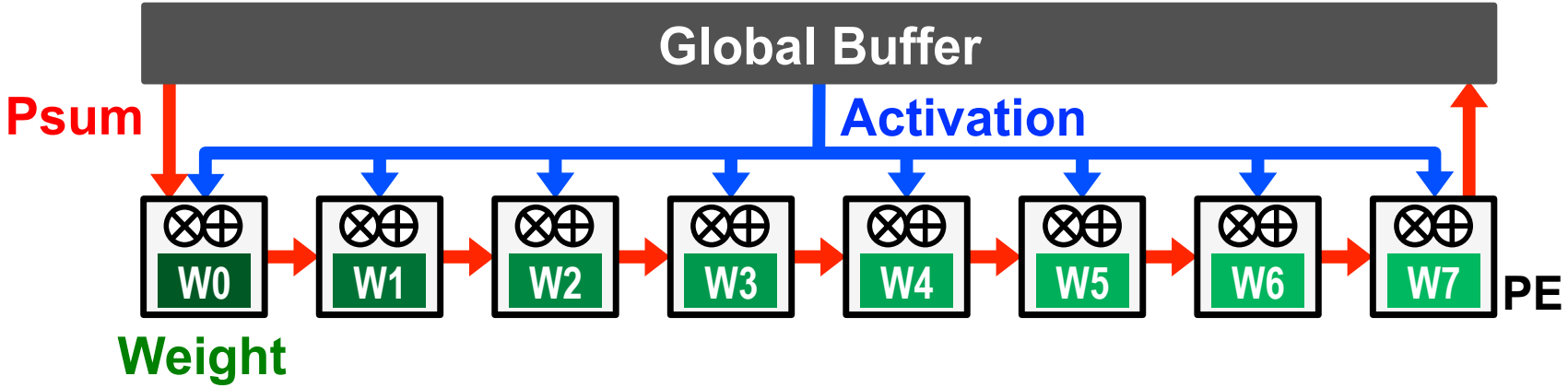




Maximize data reuse at low cost levels of hierarchy

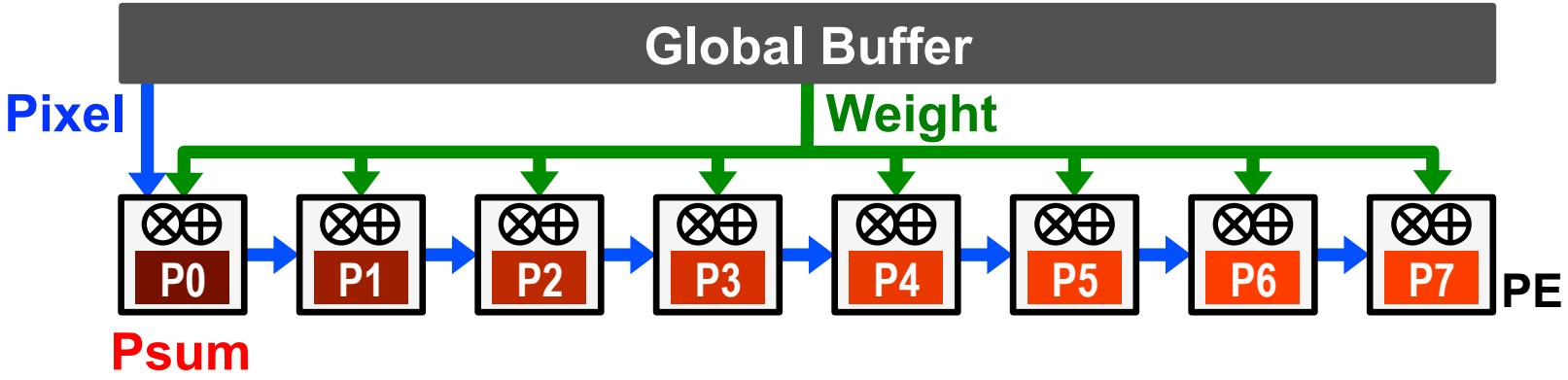
* measured from a commercial 65nm process

Weight Stationary (WS)



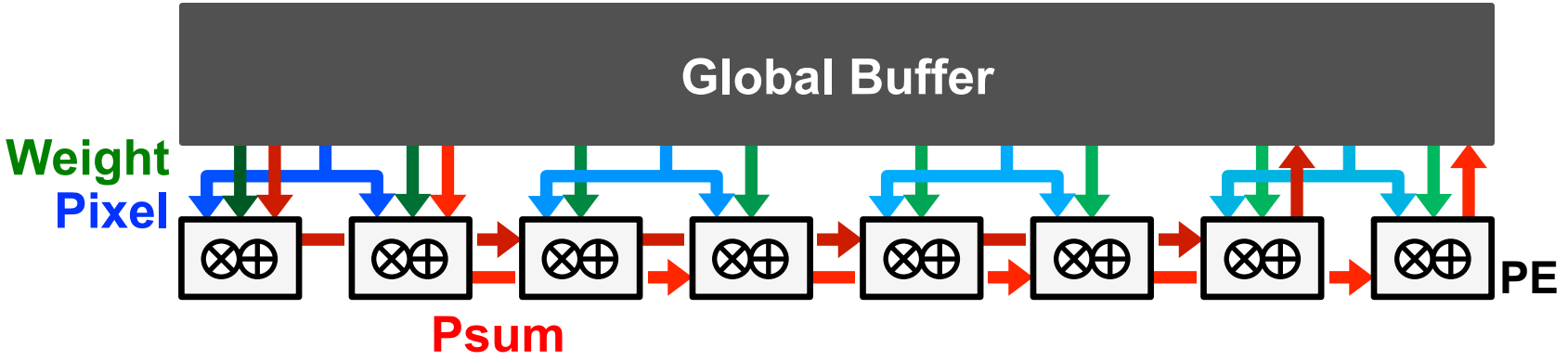
- **Minimize weight** read energy consumption
  - maximize convolutional and filter reuse of weights
- **Examples:** [**nn-X (NeuFlow)**, *CVPRW* 2014] [**Park**, *ISSCC* 2015] [**Origami**, *GLSVLSI* 2015] [**Google TPU**, *ISCA* 2017]

Output Stationary (OS)



- **Minimize partial sum** R/W energy consumption
  - maximize local accumulation
- **Examples:** [**Gupta**, *ICML* 2015] [**ShiDianNao**, *ISCA* 2015] [**ENVISION**, *ISSCC* 2017] [**Thinker**, *JSSC* 2017]

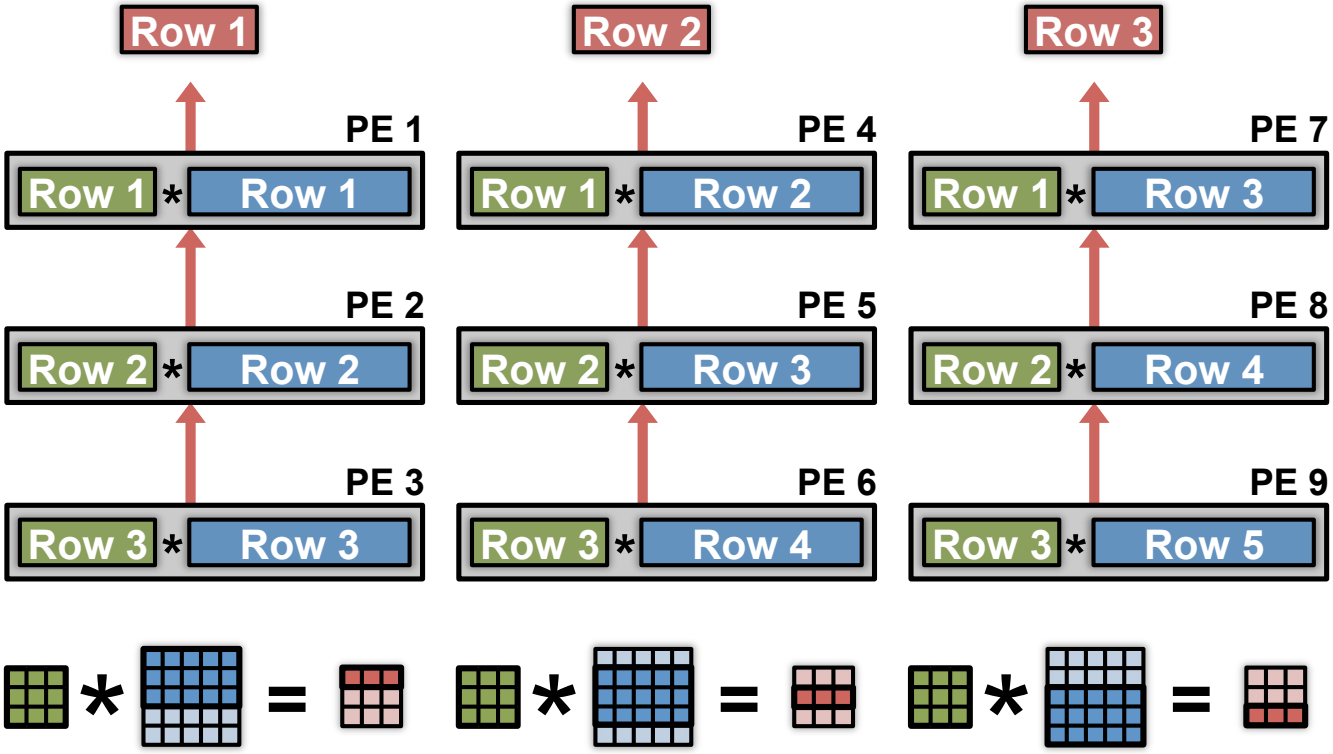
No Local Reuse (NLR)



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

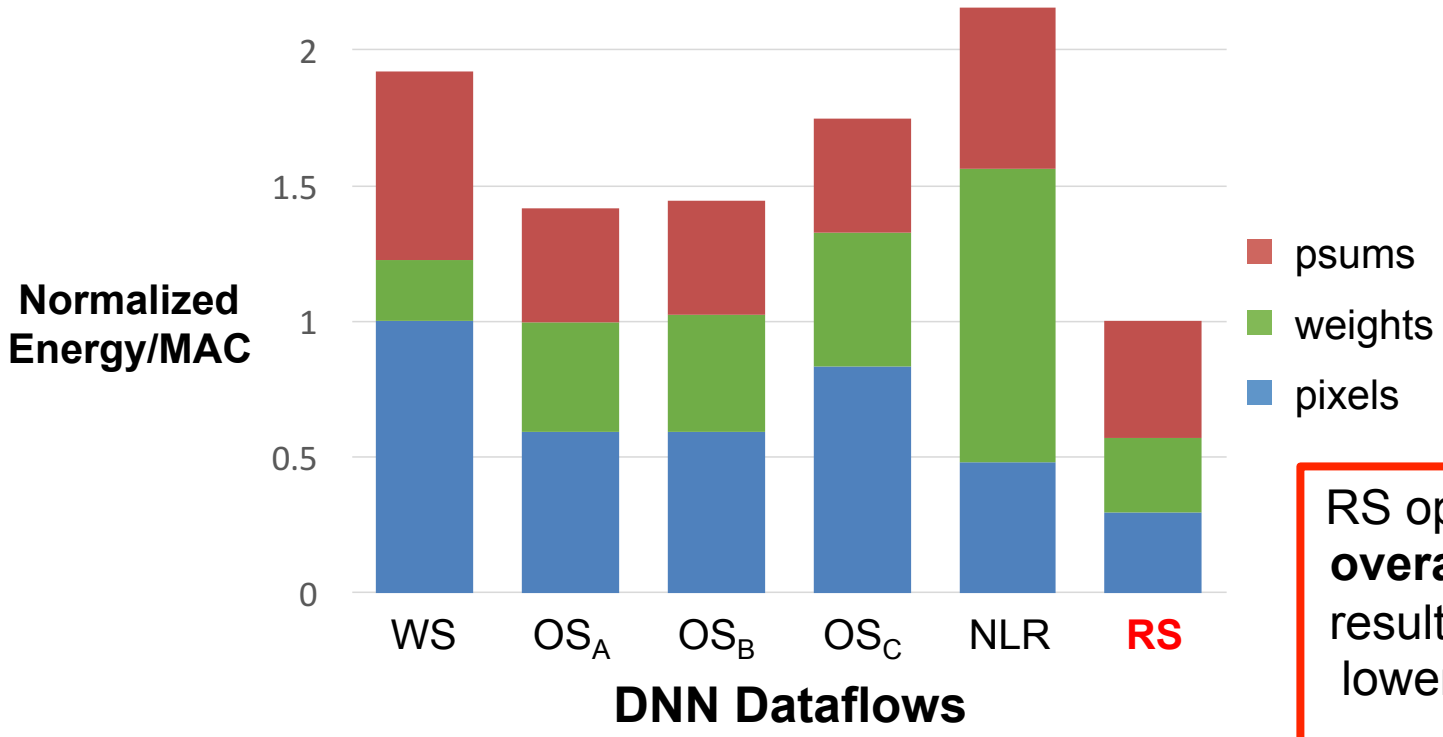
- **Examples:** [**DianNao**, *ASPLOS* 2014] [**DaDianNao**, *MICRO* 2014]
  [**Zhang**, *FPGA* 2015]

Row Stationary Dataflow



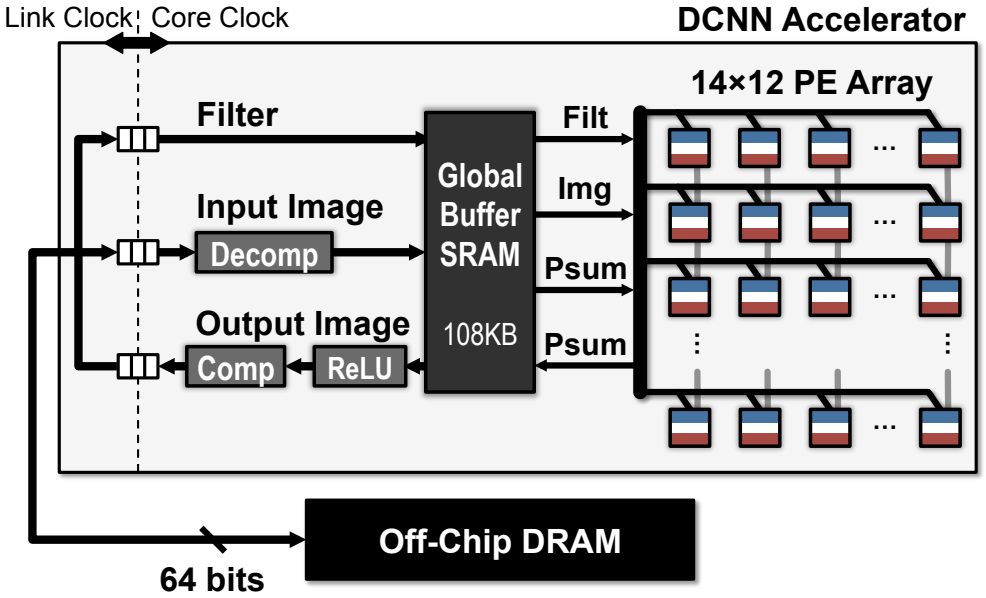
Optimize for **overall energy efficiency** instead for only a certain data type

Dataflow Comparison: CONV Layers

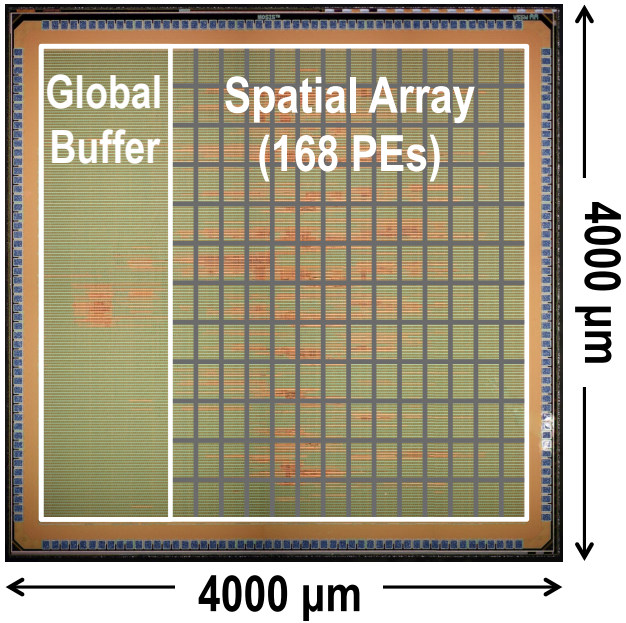


RS optimizes for the best **overall** energy efficiency resulting in a 1.4× – 2.5× lower energy than other dataflows

Eyeriss Deep CNN Accelerator



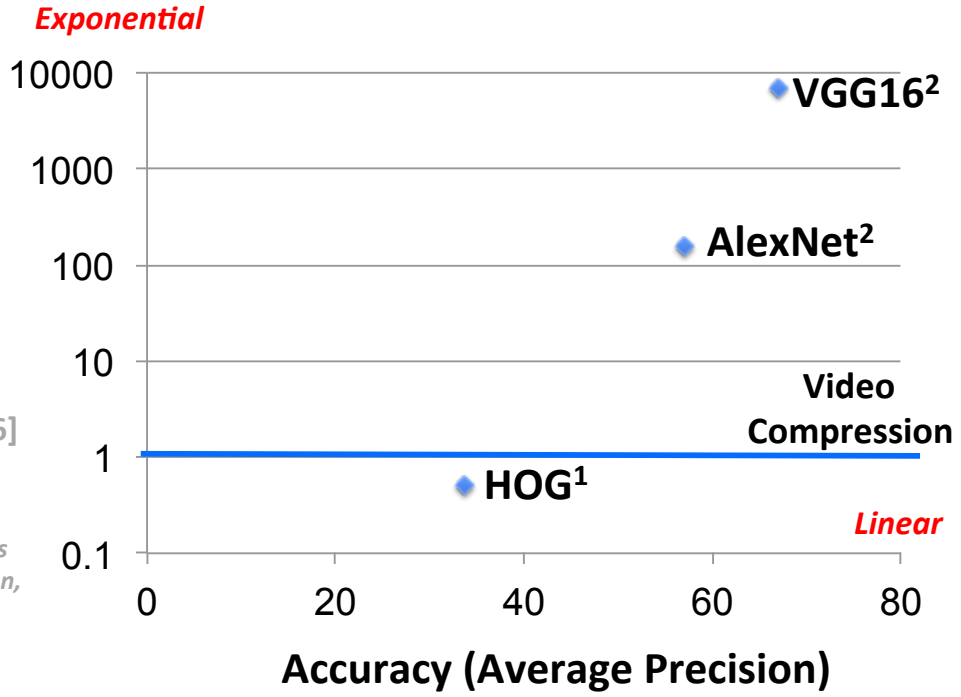
Fabricated in a 65nm process



AlexNet @ 35 fps while consuming 278mW
>10x more energy efficient than a mobile GPU

[Chen et al., ISSCC 2016]

Features: Energy versus 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]

Designing Efficient DNN Models

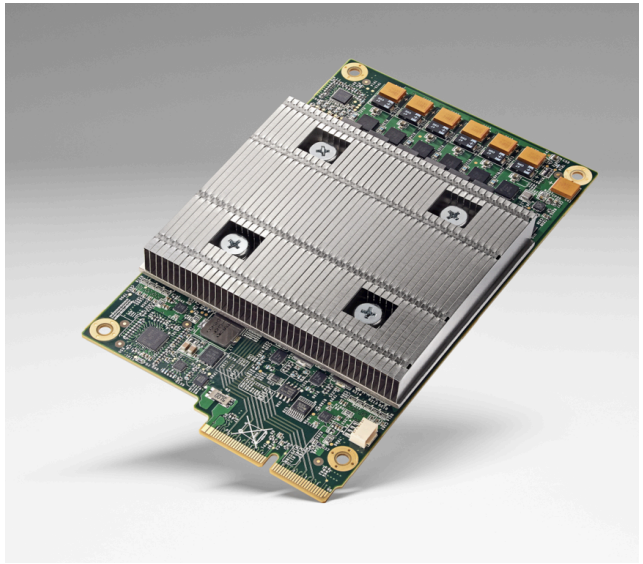
Approaches

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

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

Commercial Products using 8-bit Integer

Nvidia's Pascal (2016)

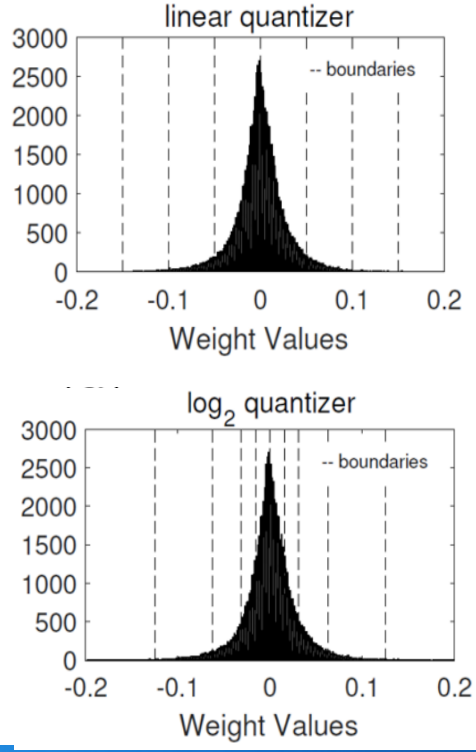


Google's TPU (2016)

Reduced Precision in Research

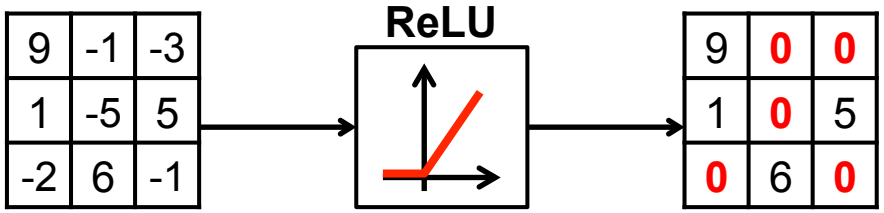
- **Reduce number of bits**
  - Binary Nets [Courbariaux, NIPS 2015]
- **Reduce number of unique weights**
  - Ternary Weight Nets [Li, arXiv 2016]
  - XNOR-Net [Rategari, ECCV 2016]
- **Non-Linear Quantization**
  - LogNet [Lee, ICASSP 2017]

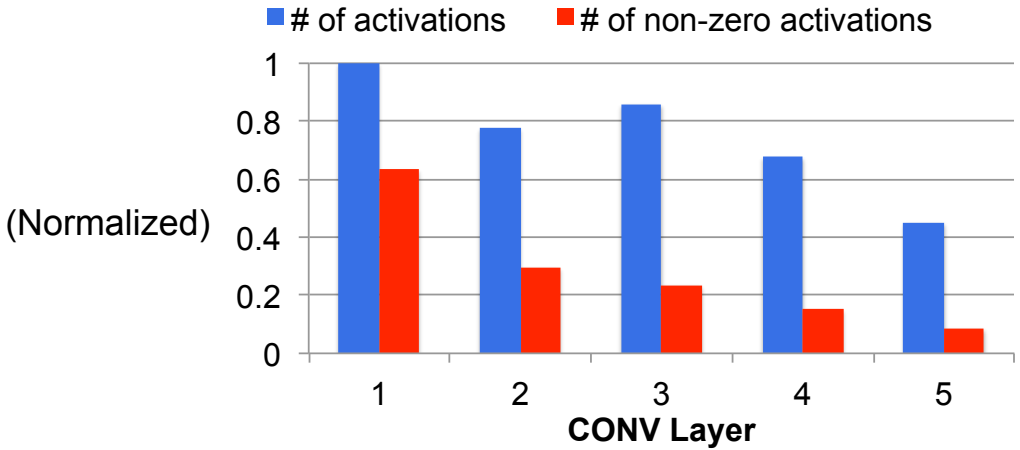
Log Domain Quantization



Sparsity in Feature Map

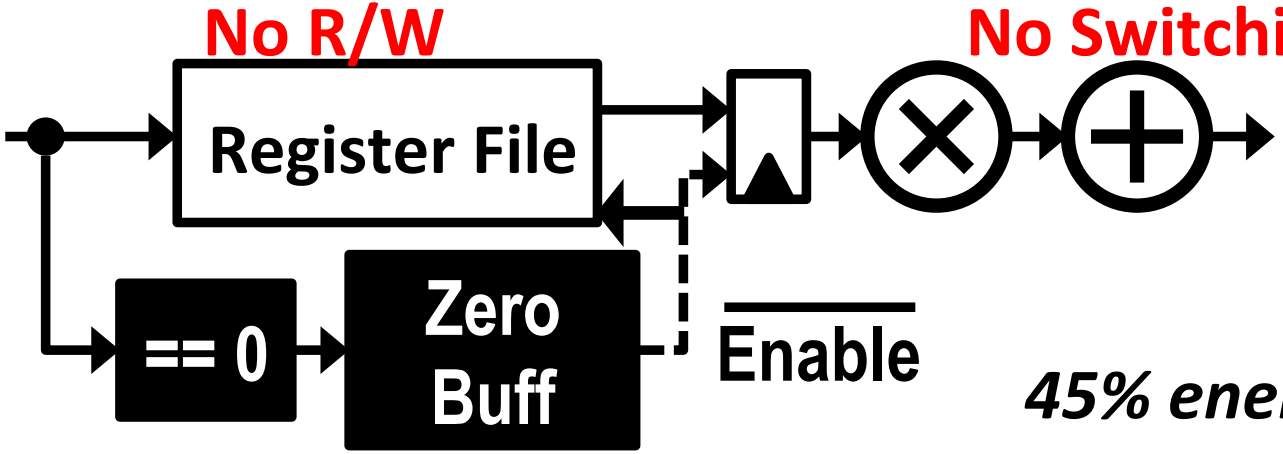
Many **zeros** in **output fmaps** after **ReLU**





Exploit Sparsity

Method 1: Skip memory access and computation

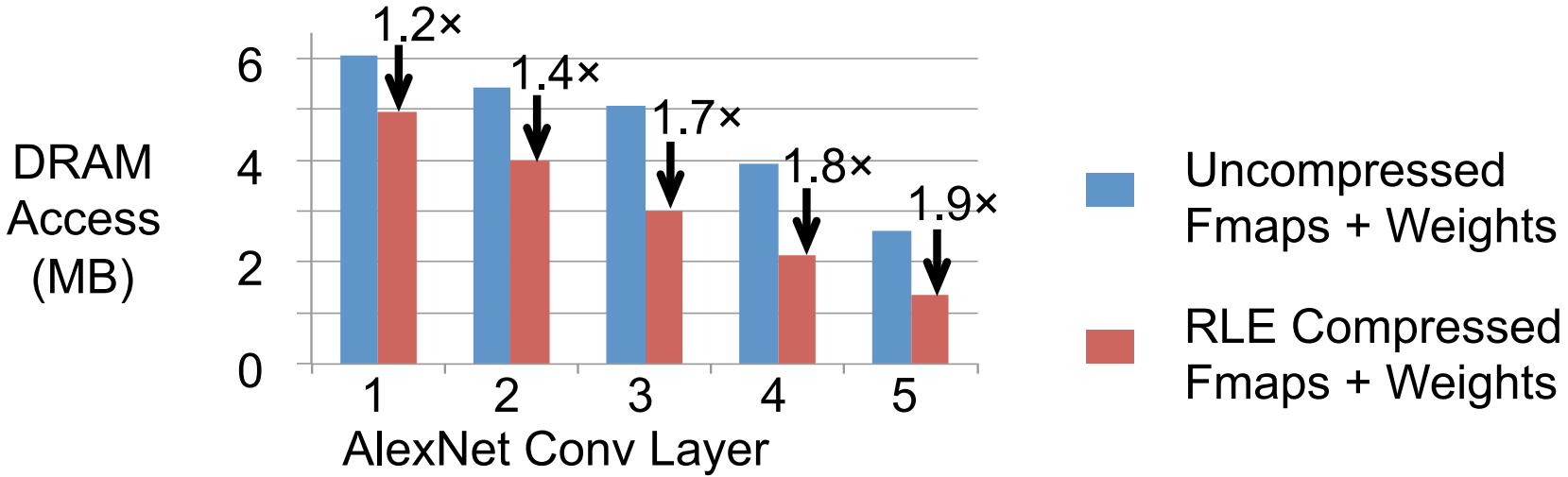


45% energy savings

[Chen et al., ISSCC 2016]

Exploit Sparsity

Method 2: Compress data to reduce storage and data movement

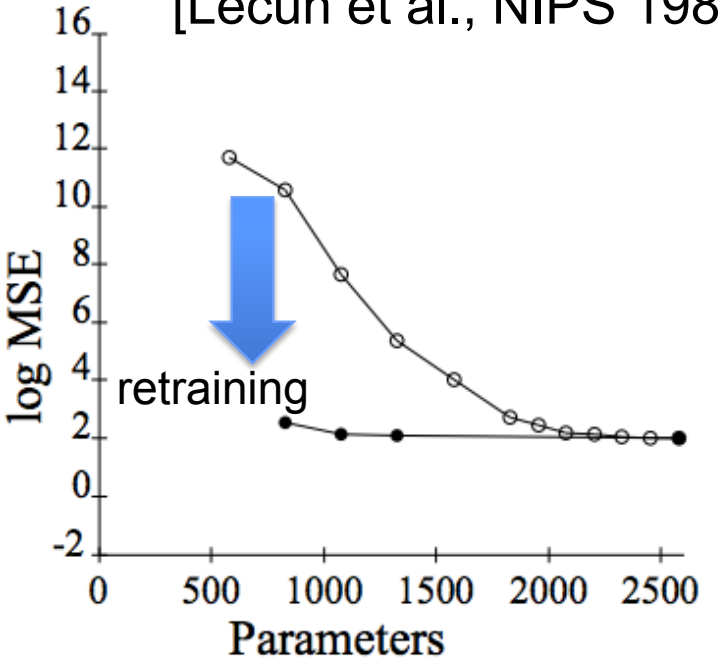


[Chen et al., ISSCC 2016]

Pruning – Make Weights Sparse

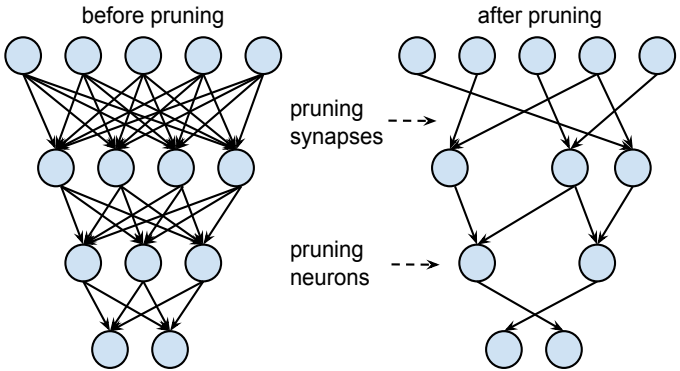
Optimal Brain Damage

[Lecun et al., NIPS 1989]



Prune DNN based on
magnitude of weights

[Han et al., NIPS 2015]

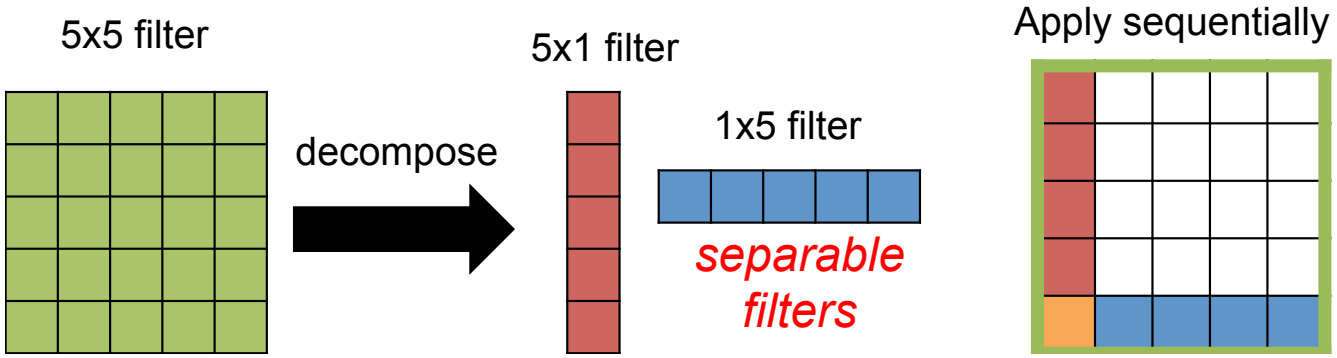


Example: *AlexNet*
Weight Reduction:
CONV layers 2.7x, ***FC layers 9.9x***
Overall Reduction:
Weights 9x, MACs 3x

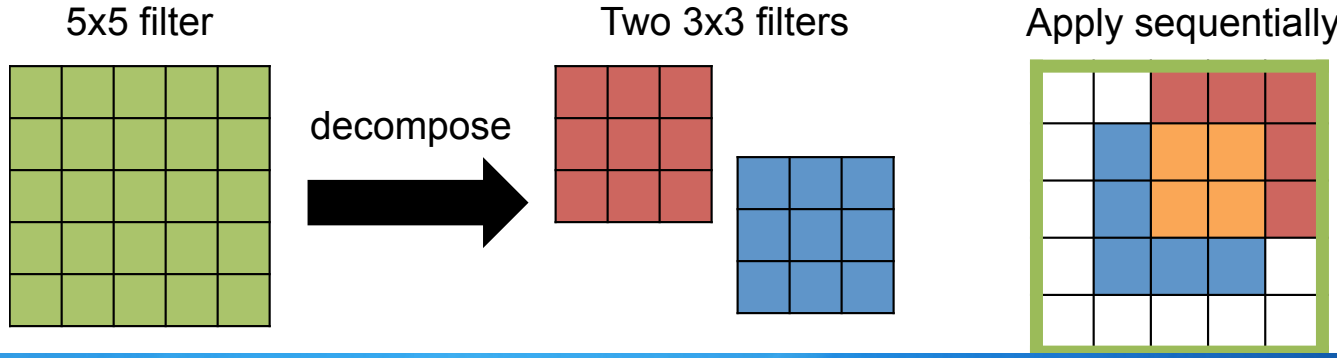
Network Architecture Design

Build Network with series of Small Filters

GoogleNet/ Inception v3

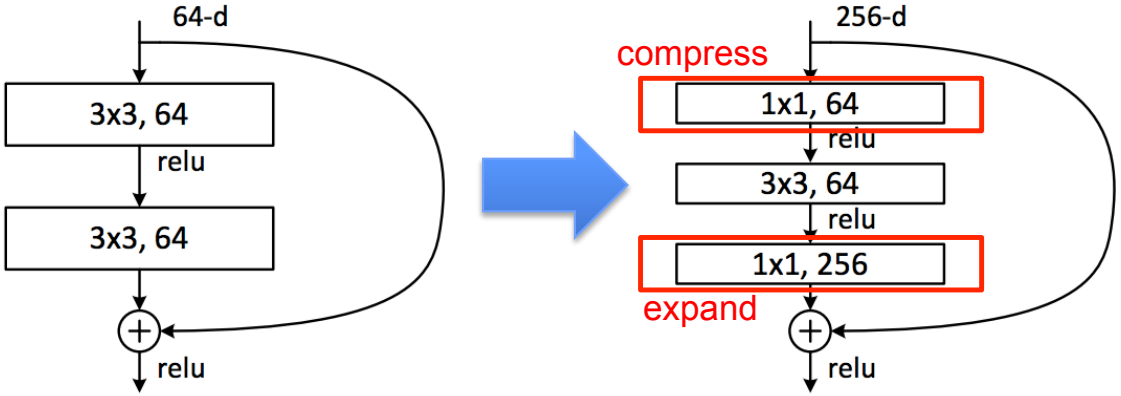


VGG-16

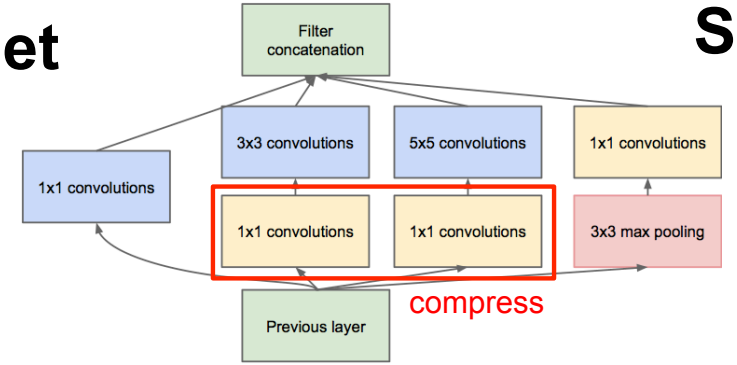


1x1 Bottleneck in Popular DNN models

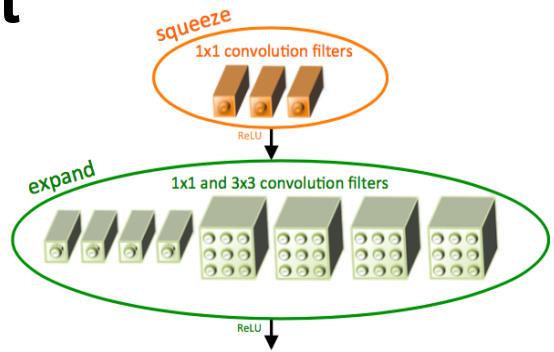
ResNet



GoogleNet



SqueezeNet

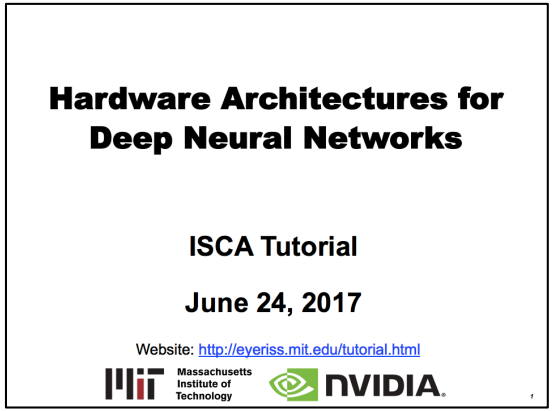


Understanding the Limitations of Existing Energy-Efficient Design Approaches for Deep Neural Networks

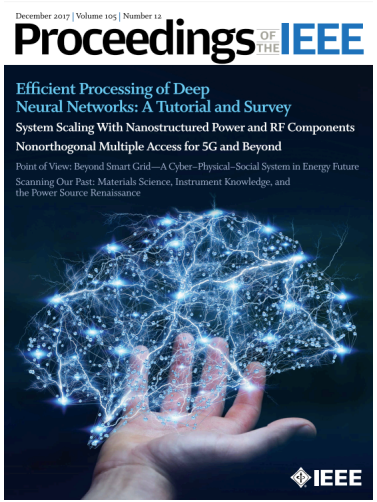
[Y.-H. Chen et al., SysML Conference, February 2018]

Energy-Efficient Processing of DNNs

A significant amount of algorithm and hardware research on energy-efficient processing of DNNs



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



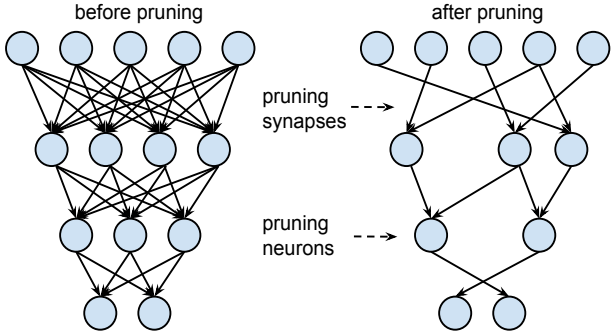
V. Sze, Y.-H. Chen, T-J. Yang, J. Emer, "***Efficient Processing of Deep Neural Networks: A Tutorial and Survey***," Proceedings of the IEEE, Dec. 2017

We identified various limitations to existing approaches

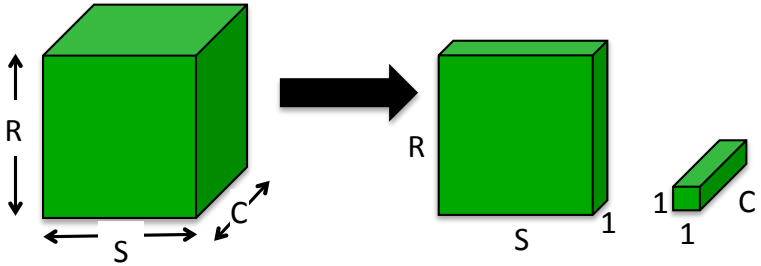
Design of Efficient DNN Algorithms

- Popular efficient DNN algorithm approaches

Network Pruning



Compact Network Architectures

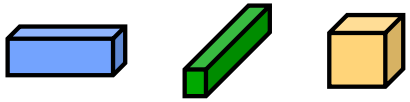


Examples: SqueezeNet, MobileNet

... also reduced precision

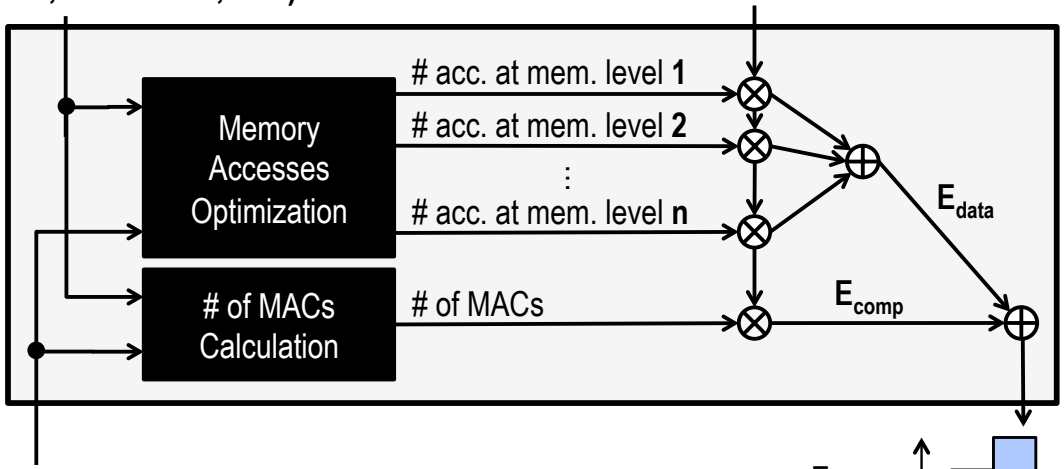
- Focus on reducing **number of MACs and weights**
- **Does it translate to energy savings?**

Energy-Evaluation Methodology

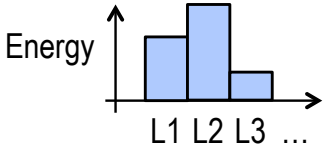


CNN Shape Configuration
(# of channels, # of filters, etc.)

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, …]



CNN Energy Consumption

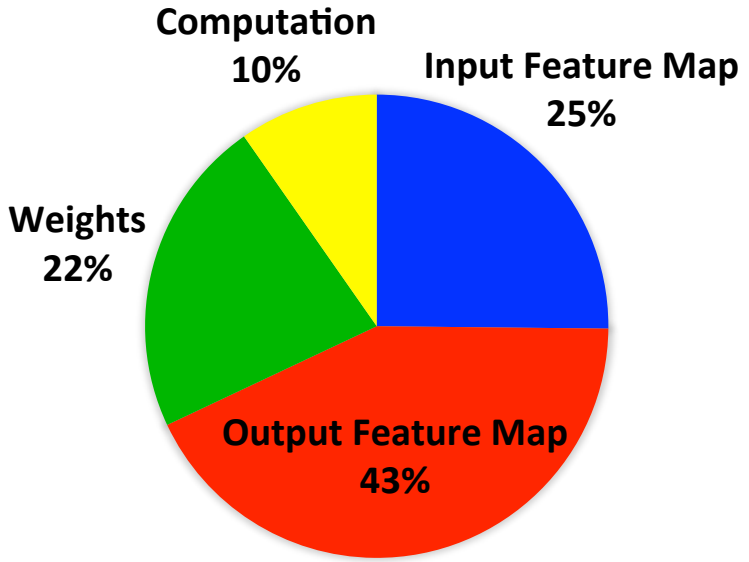
Energy estimation tool available at http://eyeriss.mit.edu

[Yang et al., CVPR 2017]

Key Observations

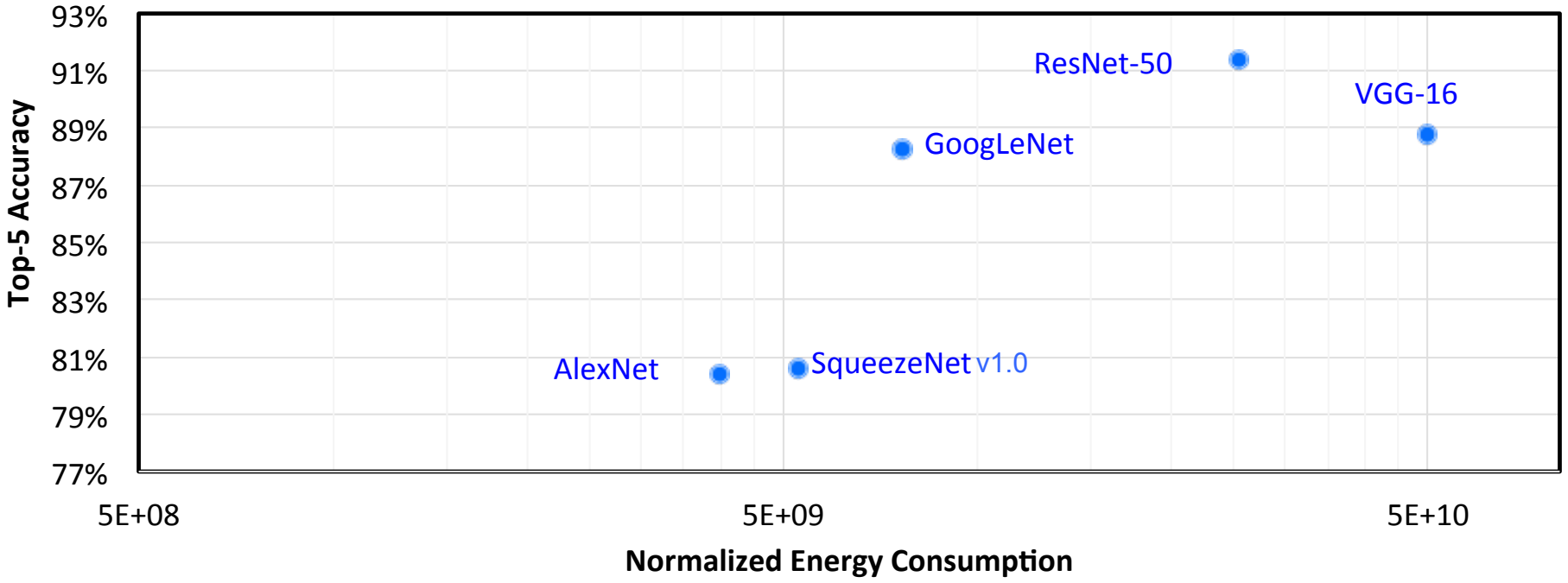
- Number of weights *alone* is not a good metric for energy
- All data types should be considered

Energy Consumption of GoogLeNet



[Yang et al., CVPR 2017]

Energy Consumption of Existing DNNs

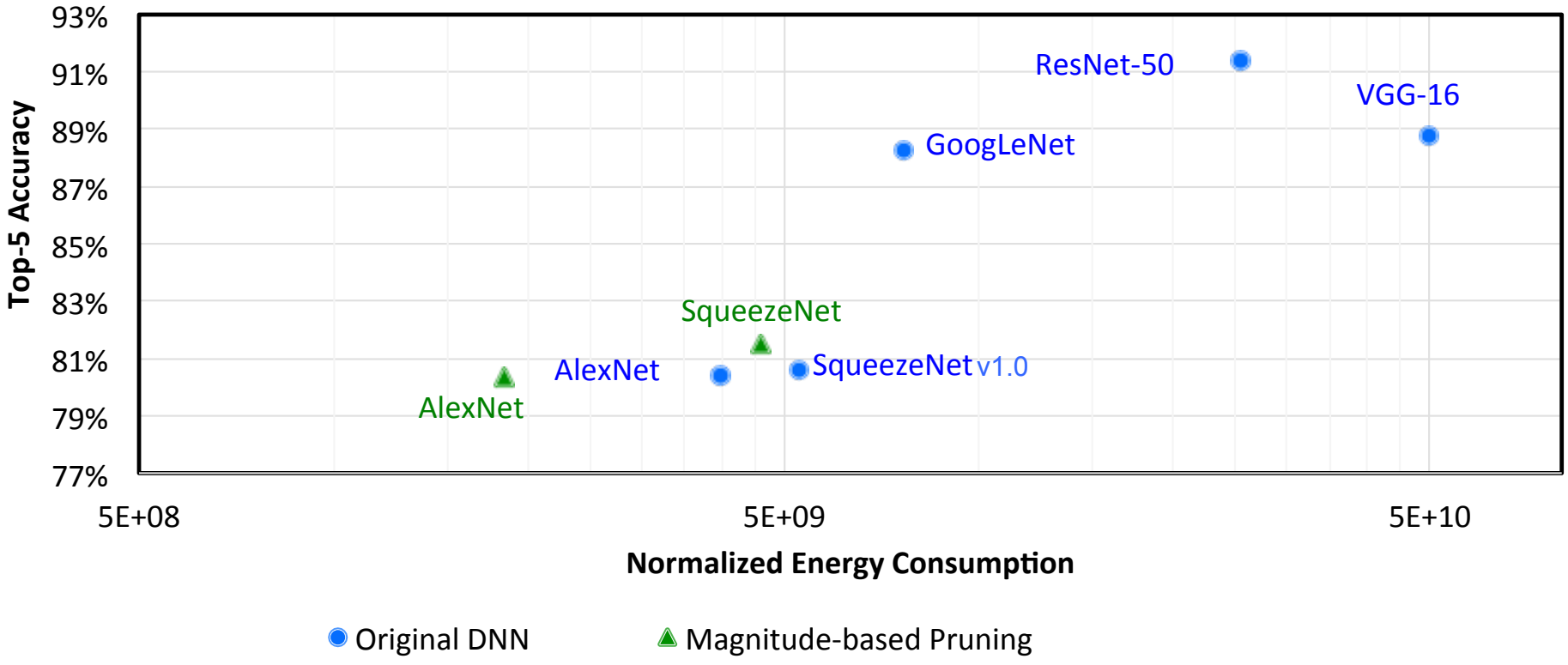


Batch sizes between 44 to 48

[Yang et al., CVPR 2017]

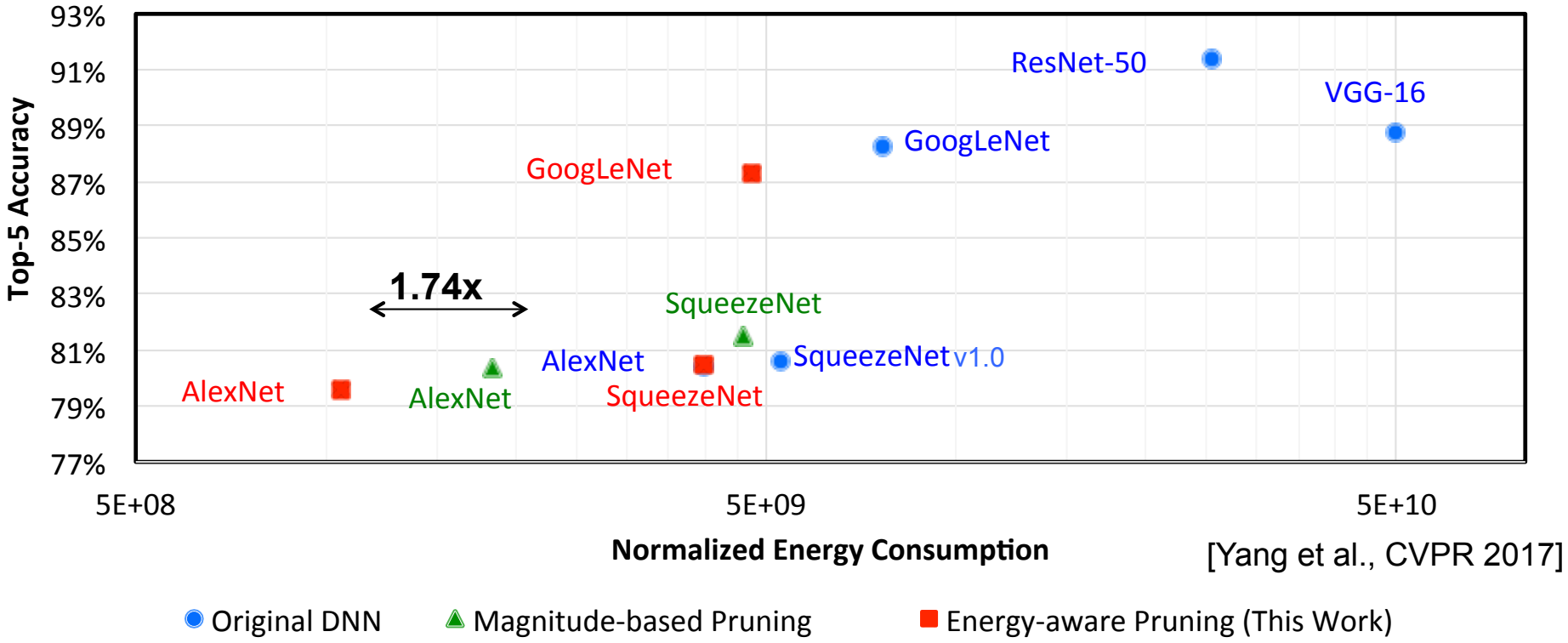
Deeper CNNs with fewer weights do not necessarily consume less energy than shallower CNNs with more weights

Magnitude-based Weight Pruning



Reduce number of weights by removing small magnitude weights

Energy-Aware Pruning

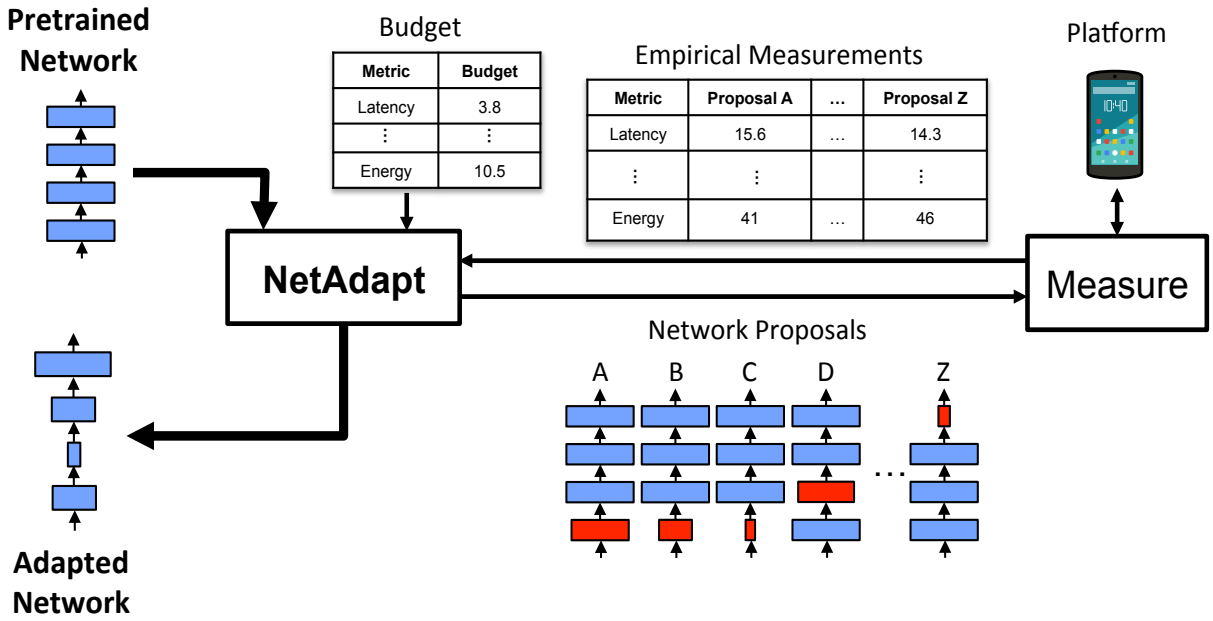


[Yang et al., CVPR 2017]

Directly target energy and incorporate it into the optimization of DNNs to provide greater energy savings

NetAdapt: Platform-Aware DNN Adaptation

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

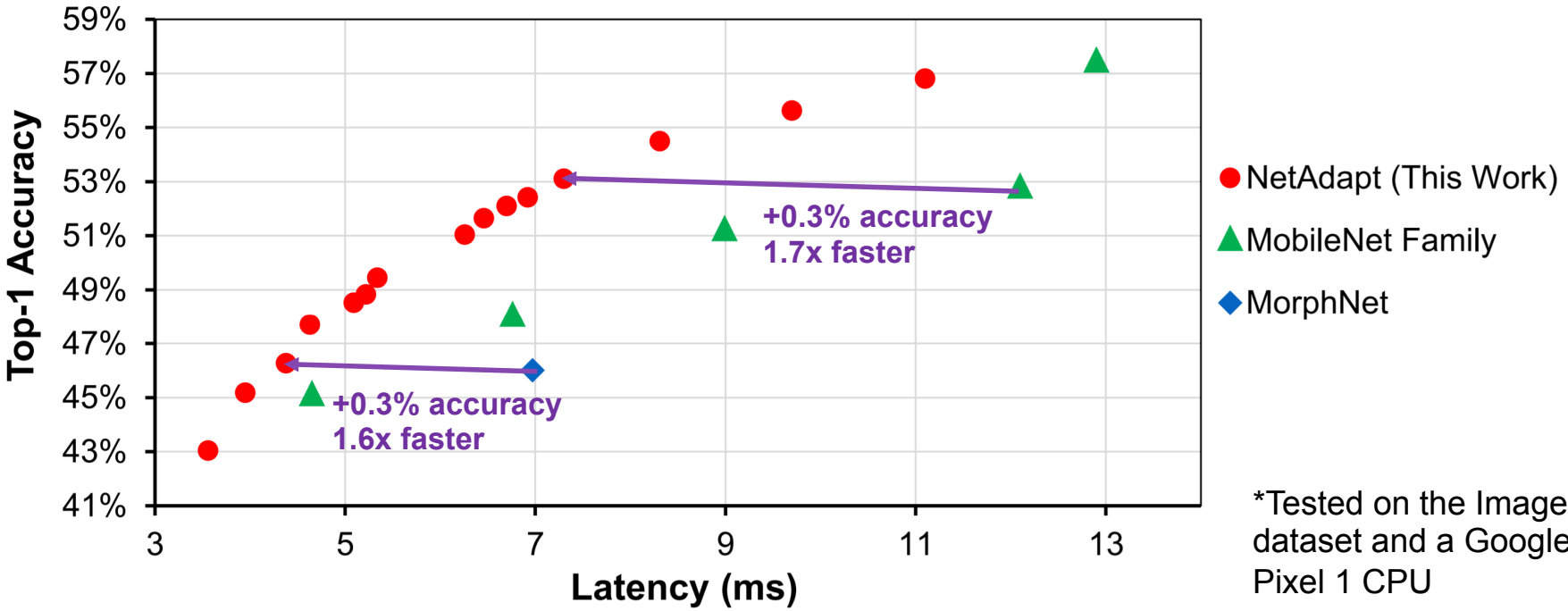


In collaboration with Google's Mobile Vision Team

[Yang et al., arXiv 2018]

Latency vs. Accuracy Tradeoff with NetAdapt

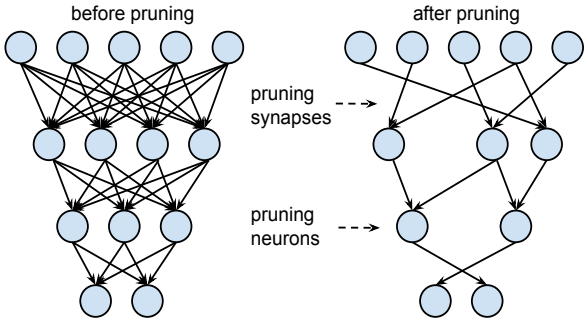
- NetAdapt boosts the real inference speed of MobileNet by 1.7x with higher accuracy



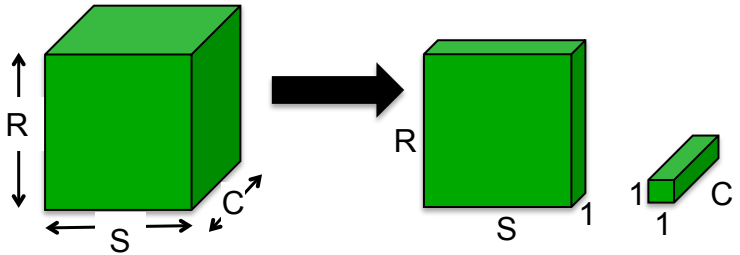
*Tested on the ImageNet dataset and a Google Pixel 1 CPU

Many Efficient DNN Design Approaches

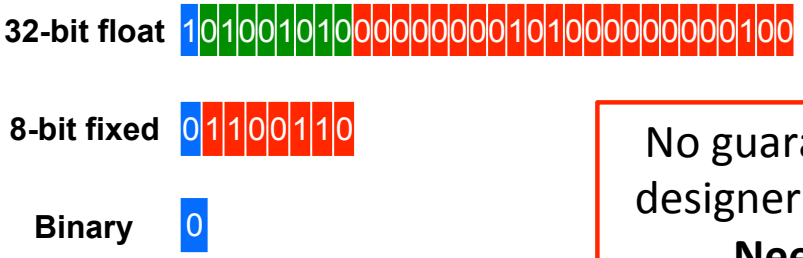
Network Pruning



Compact Network Architectures



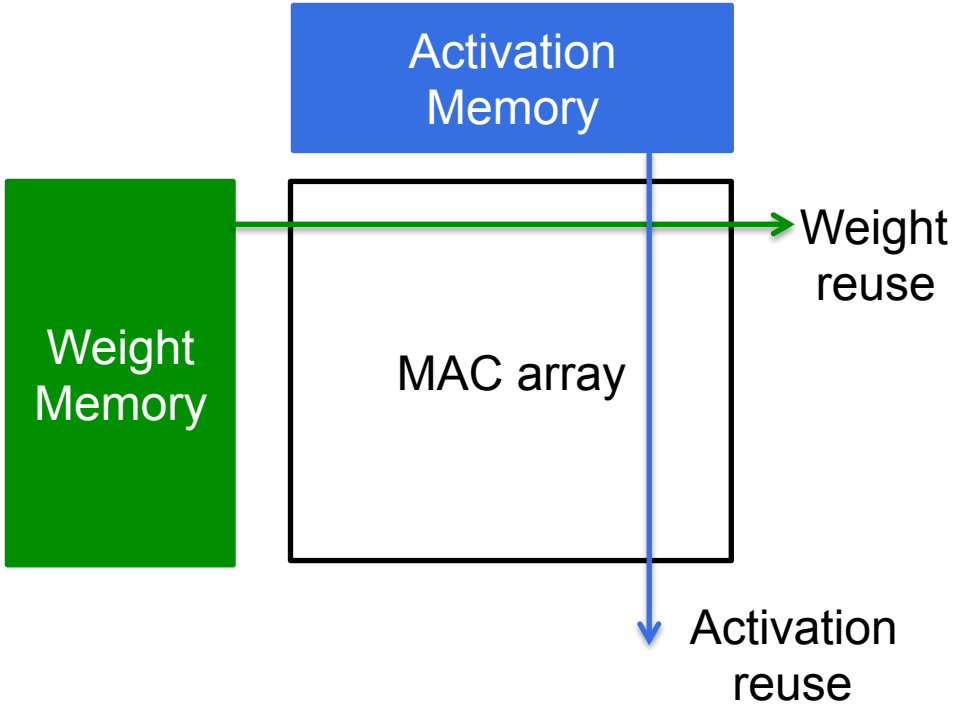
Reduce Precision



No guarantee that DNN algorithm designer will use a given approach.
Need flexible hardware!

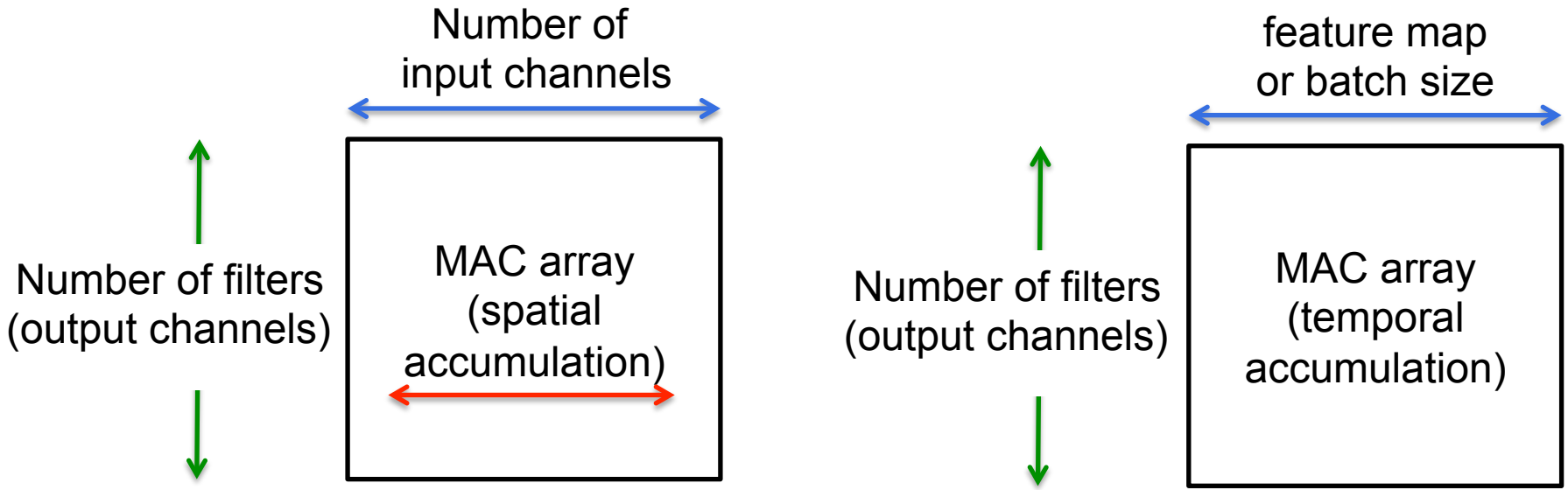
Existing DNN Architectures

- Specialized DNN hardware often rely on certain properties of DNN in order to achieve high energy-efficiency
- Example: Reduce memory access by amortizing across MAC array



Limitation of Existing DNN Architectures

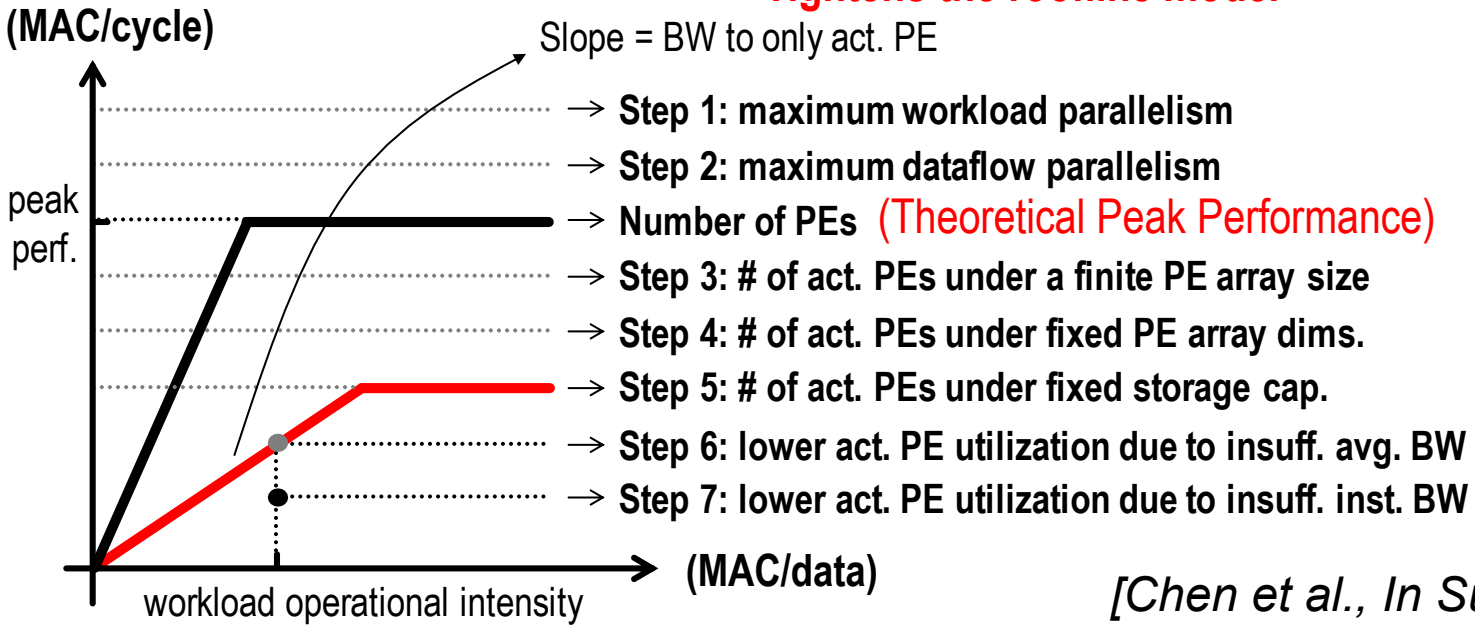
- Example: reuse depends on # of channels, feature map/batch size
  - Not efficient across all network architectures (e.g., compact DNNs)



Eyexam: Understanding Sources of Inefficiencies in DNN Accelerators

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

Tightens the roofline model

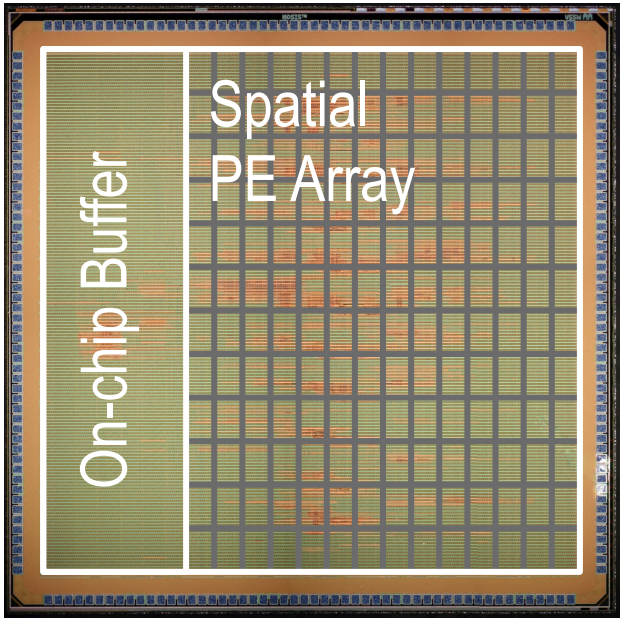


[Chen et al., In Submission]

Eyeriss v2

To efficiently support:

- Wide range of filter shapes
  - Large and Compact
- Different Layers
  - e.g., CONV and FC
- Wide range of sparsity
  - Dense and Sparse



Eyeriss (v1)
[Chen et al. ISSCC 2016, ISCA 2016]

Benchmarking Metrics for DNN Hardware

How can we compare designs?

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

Metrics for DNN Hardware

- **Accuracy**
  - Quality of result for a given task
- **Throughput**
  - Analytics on high volume data
  - Real-time performance (e.g., video at 30 fps)
- **Latency**
  - For interactive applications (e.g., autonomous navigation)
- **Energy and Power**
  - Edge and embedded devices have limited battery capacity
  - Data centers have stringent power ceilings due to cooling costs
- **Hardware Cost**
  - $$$

Specifications to Evaluate Metrics

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

Example: Metrics of Eyeriss Chip

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

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

Comprehensive Coverage

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

Evaluation Process

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

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

Summary

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

Massachusetts Institute of Technology

Acknowledgements: This work is funded by the DARPA YFA grant, MIT Center for Integrated Circuits & Systems, and gifts from Intel, Nvidia and Google.

References

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

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

- MIT Professional Education Course on **"Designing Efficient Deep Learning Systems"** http://professional-education.mit.edu/deeplearning

For updates on Eyerissv2, Eyexam, NetAdapt, etc.

Follow @eems_mit

or join EEMS news mailing list

References

- A. Suleiman*, Y.-H. Chen*, J. Emer, V. Sze, "Towards Closing the Energy Gap Between HOG and CNN Features for Embedded Vision," IEEE International Symposium of Circuits and Systems (ISCAS), May 2017.
- V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, Z. Zhang, "Hardware for Machine Learning: Challenges and Opportunities," IEEE Custom Integrated Circuits Conference (CICC), Invited Paper, May 2017.
- Y.-H. Chen, T. Krishna, J. Emer, V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," IEEE Journal of Solid State Circuits (JSSC), ISSCC Special Issue, Vol. 52, No. 1, pp. 127-138, January 2017.
- Y.-H. Chen, J. Emer, V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," International Symposium on Computer Architecture (ISCA), pp. 367-379, June 2016.
- Y.-H. Chen, T. Krishna, J. Emer, V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," IEEE International Conference on Solid-State Circuits (ISSCC), pp. 262-264, February 2016.

References

- T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, V. Sze, H. Adam, "NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications," arXiv, April 2018.
- Y.-H. Chen*, T.-J. Yang*, J. Emer, V. Sze, "Understanding the Limitations of Existing Energy-Efficient Design Approaches for Deep Neural Networks," SysML Conference, February 2018.
- V. Sze, T.-J. Yang, Y.-H. Chen, J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, December 2017.
- T.-J. Yang, Y.-H. Chen, J. Emer, V. Sze, "A Method to Estimate the Energy Consumption of Deep Neural Networks," Asilomar Conference on Signals, Systems and Computers, Invited Paper, October 2017.
- T.-J. Yang, Y.-H. Chen, V. Sze, "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.
- Y.-H. Chen, J. Emer, V. Sze, "Using Dataflow to Optimize Energy Efficiency of Deep Neural Network Accelerators," IEEE Micro's Top Picks from the Computer Architecture Conferences, May/June 2017.

References

- M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” in NIPS, 2015.
- F. Li and B. Liu, “Ternary weight networks,” in NIPS Workshop on Efficient Methods for Deep Neural Networks, 2016.
- M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNORNet: ImageNet Classification Using Binary Convolutional Neural Networks,” in ECCV, 2016
- E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, “Lognet: Energy-Efficient Neural Networks Using Logrithmic Computations,” in ICASSP, 2017.
- F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size,” ICLR , 2017
- A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv: 1704.04861, 2017.

References

- A. Lavin and S. Gray, “Fast algorithms for convolutional neural networks,” in CVPR, 2016.
- Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal Brain Damage,” in NIPS, 1990.
- S. Han, J. Pool, J. Tran, and W. J. Dally, “Learning both weights and connections for efficient neural networks,” in NIPS, 2015.