High-Performance Hardware for Machine Learning

Cadence ENN Summit

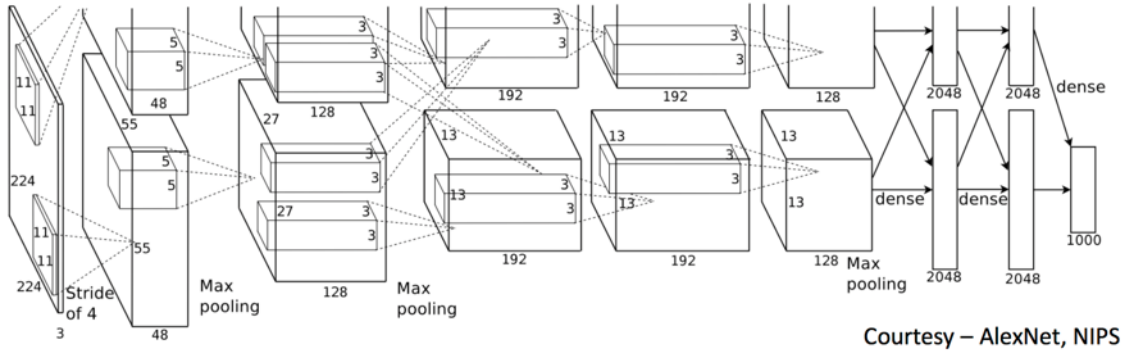
2/9/2016

Prof. William Dally

Stanford University

NVIDIA Corporation

Hardware and Data enable DNNs



Courtesy – AlexNet, NIPS 2012

The Need for Speed

Larger data sets and models lead to better accuracy but also increase computation time. Therefore progress in deep neural networks is limited by how fast the networks can be computed.

Likewise the application of convnets to low latency inference problems, such as pedestrian detection in self driving car video imagery, is limited by how fast a small set of images, possibly a single image, can be classified.

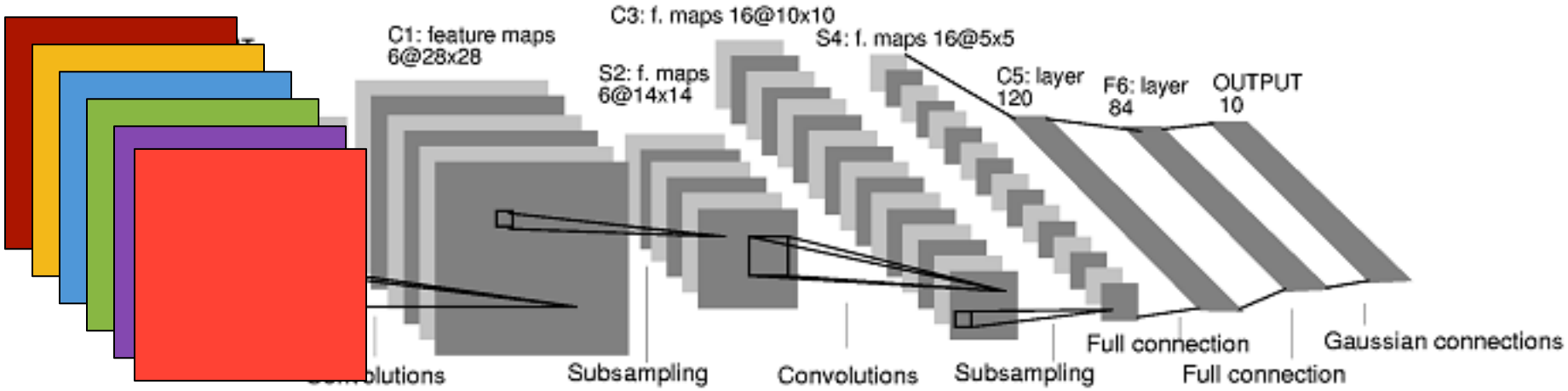
More data → Bigger Models →More Need for Compute

But Moore's law is no longer providing more compute…

Lavin & Gray, Fast Algorithms for Convolutional Neural Networks, 2015

To go faster, use more processors

Lots of parallelism in a DNN

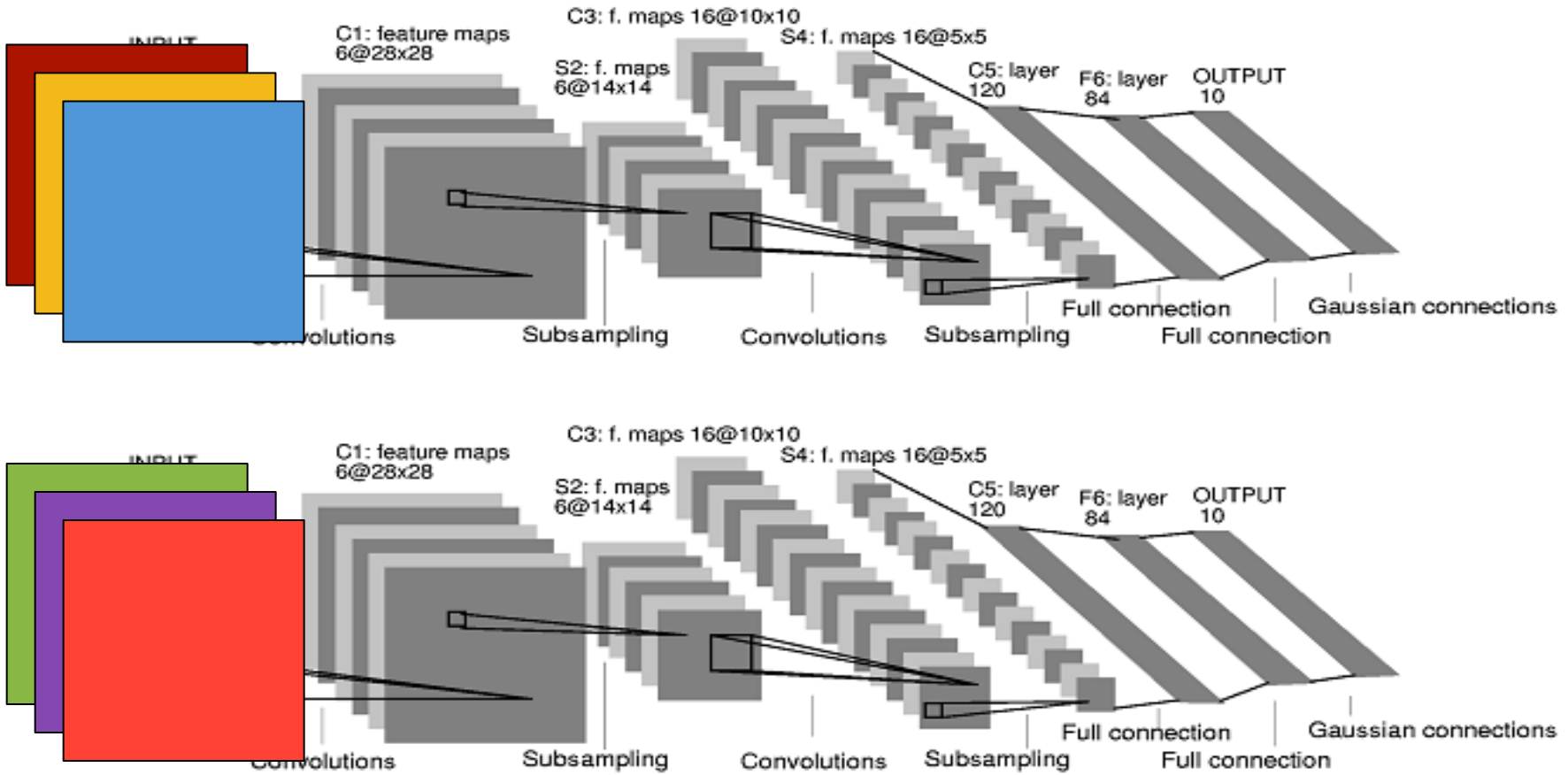


- Inputs
- Points of a feature map
- Filters
- Elements within a filter

- Multiplies within layer are independent
- Sums are reductions
- Only layers are dependent
- No data dependent operations

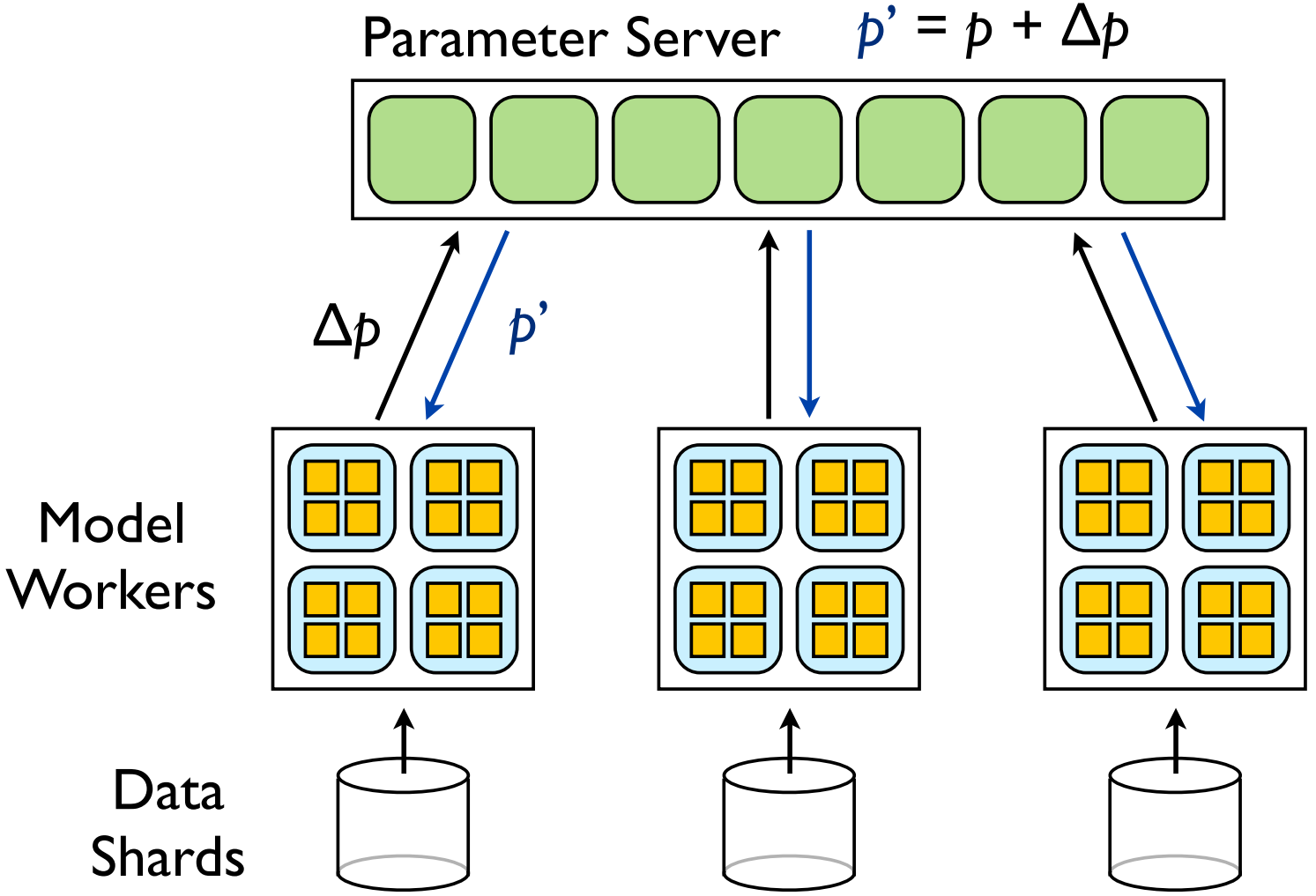
  => can be statically scheduled

Data Parallel – Run multiple inputs in parallel



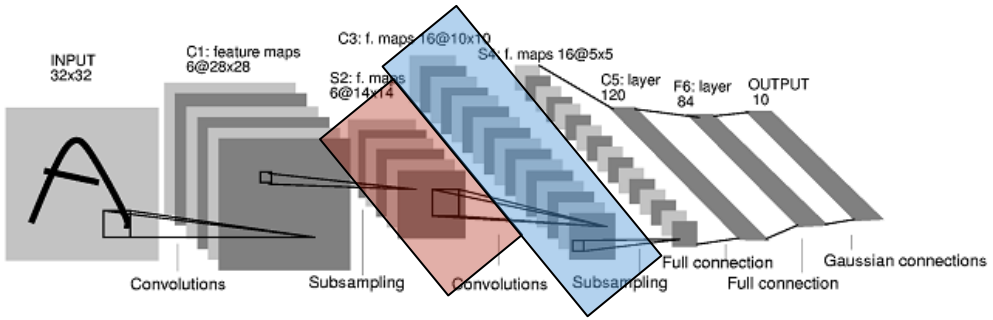
- Doesn’t affect latency for one input
- Requires P-fold larger batch size
- For training requires coordinated weight update

Parameter Update



Large Scale Distributed Deep Networks, Jeff Dean et al., 2013

Model-Parallel Convolution – by output region (x,y)



6D Loop

Forall region XY

For each output map j

For each input map k

For each pixel x,y in XY

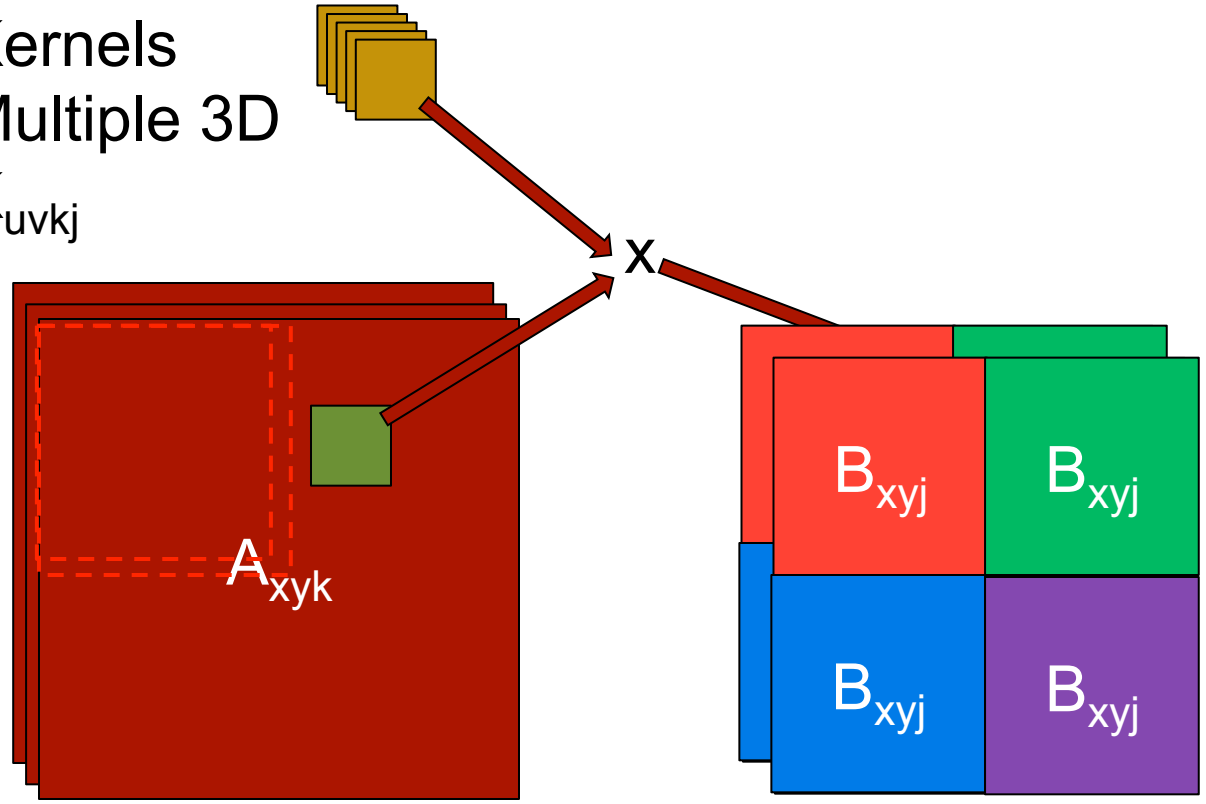
For each kernel element u,v

B_{xyj} += $A_{(x-u)(y-v)k}$ x K_{uvkj}

Kernels

Multiple 3D

K_{uvkj}



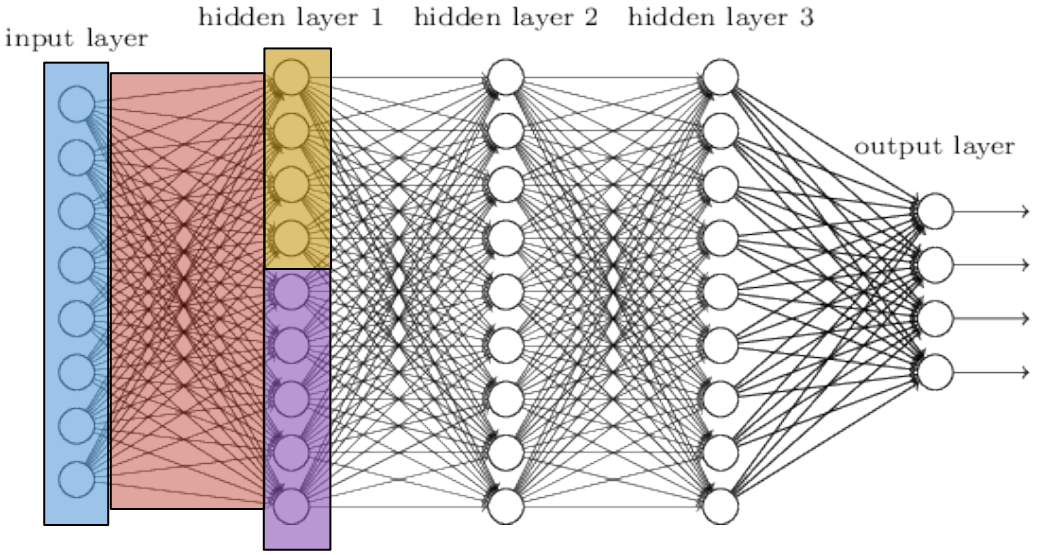
Input maps

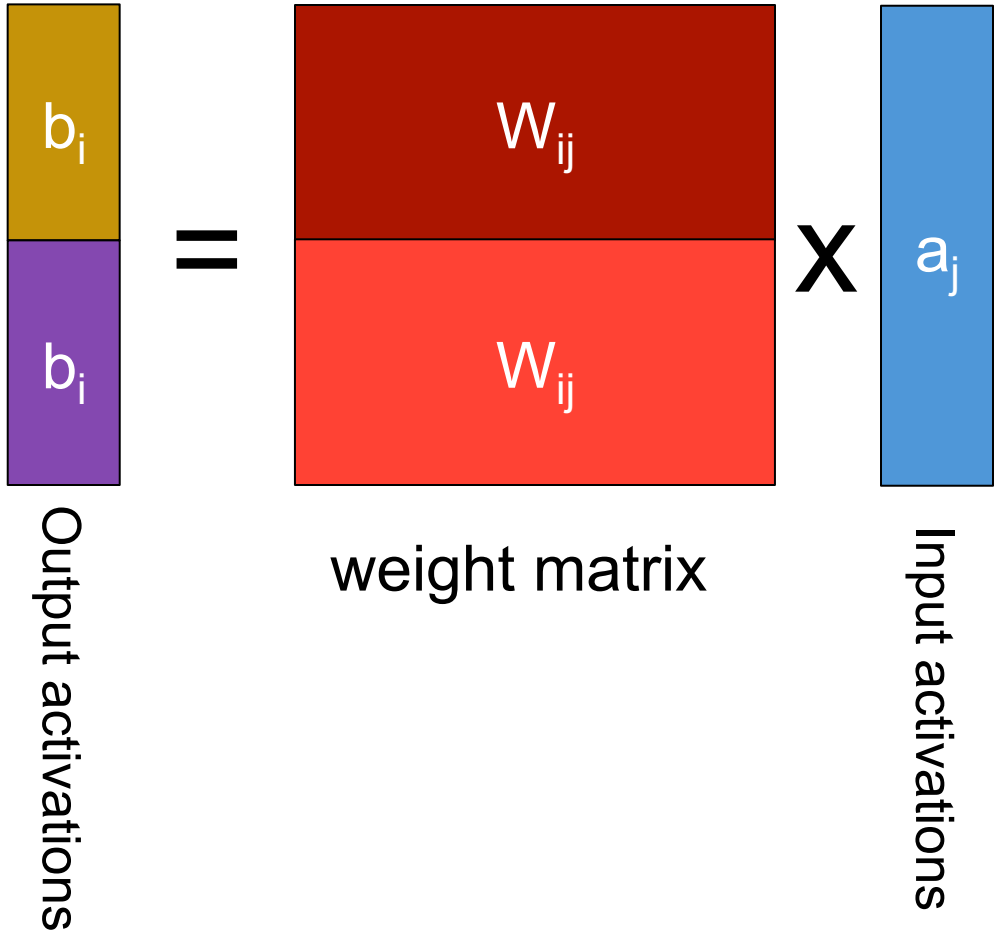
A_{xyk}

Output maps

B_{xyj}

Model Parallel Fully-Connected Layer (M x V)





To go fast, use multiple processors

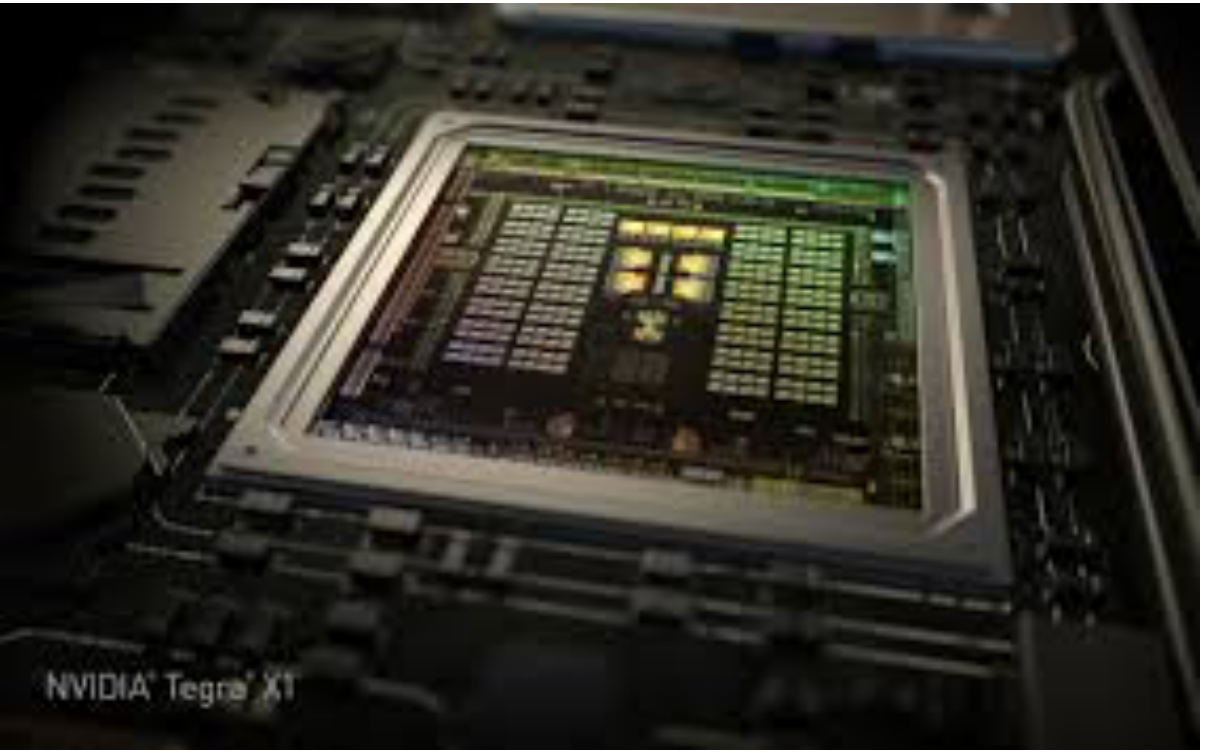
To be efficient and fast, use GPUs

To be efficient and go really fast, use multiple GPUs

Titan X

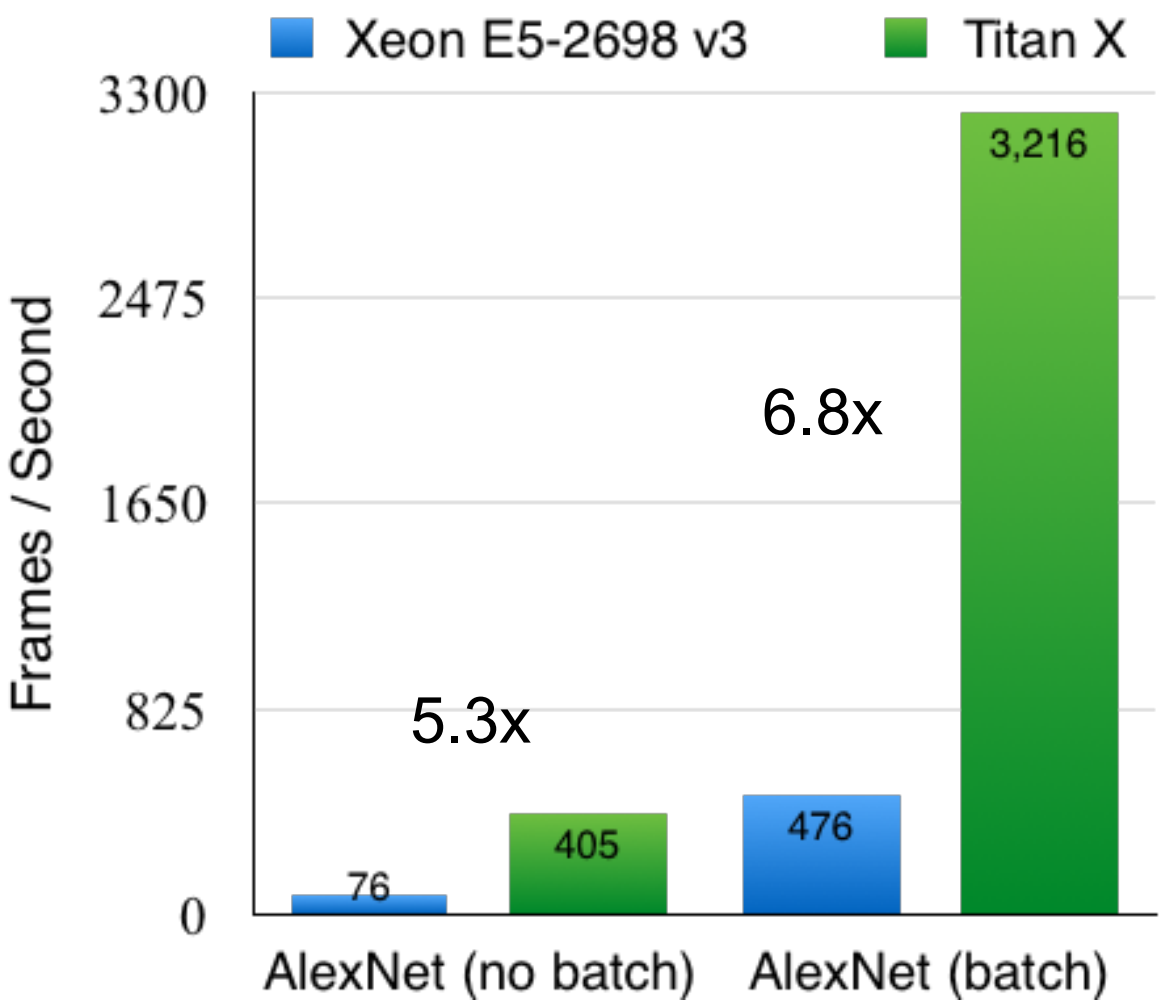
- 3072 CUDA cores @ 1 GHz
- 6 Teraflops FP32
- 12GB of GDDR5 @ 336 GB/sec
- 250W TDP
- 24GFLOPS/W
- 28nm process

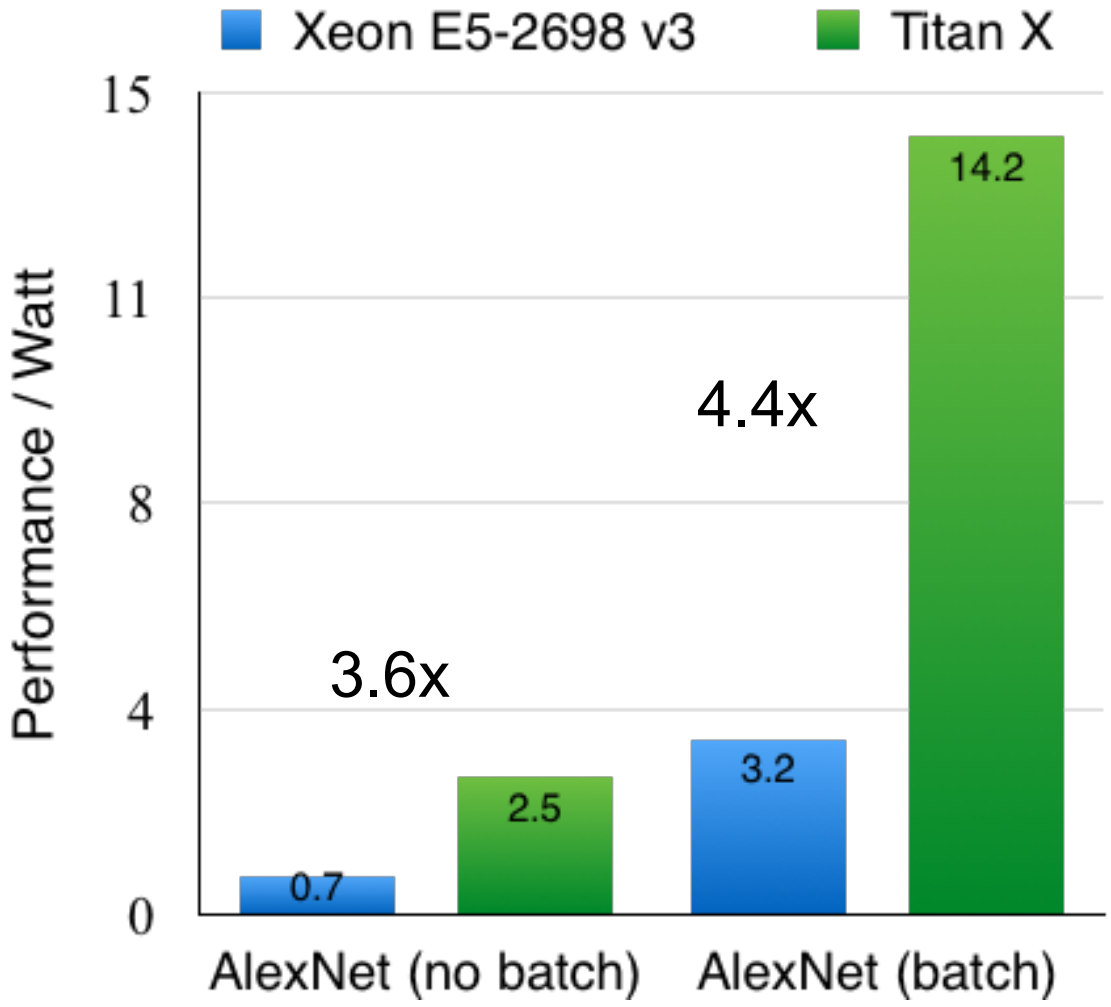
Tegra X1



- 256 CUDA cores @ ~1 GHz
- 1 Teraflop FP16
- 4GB of LPDDR4 @ 25.6 GB/s
- 15 W TDP (1W idle, <10W typical)
- 100GFLOPS/W (FP16)
- 20nm process

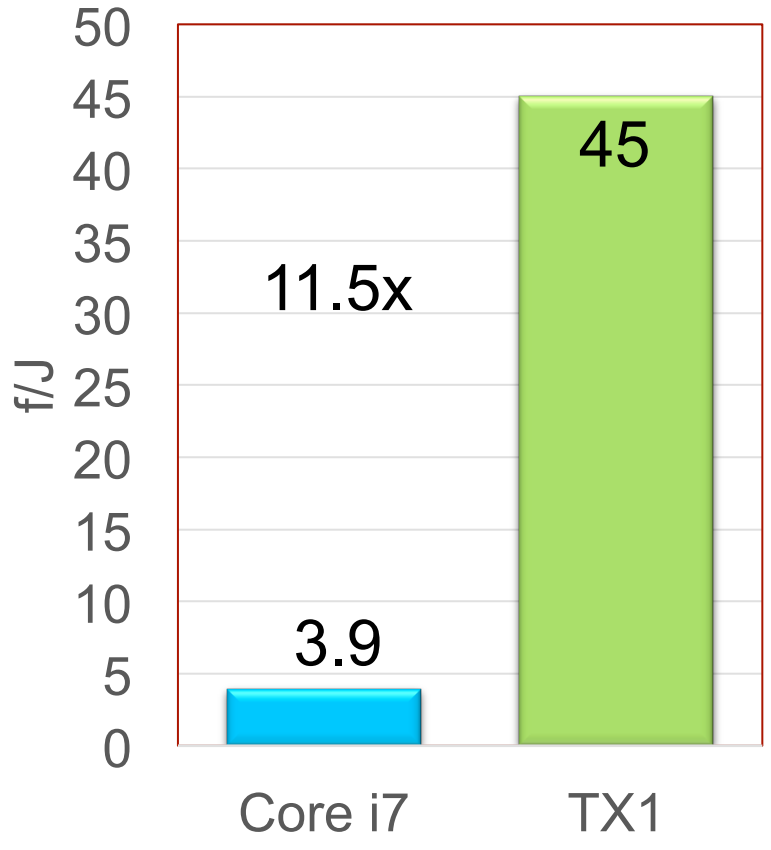
Xeon E5-2698 CPU v.s. TitanX GPU





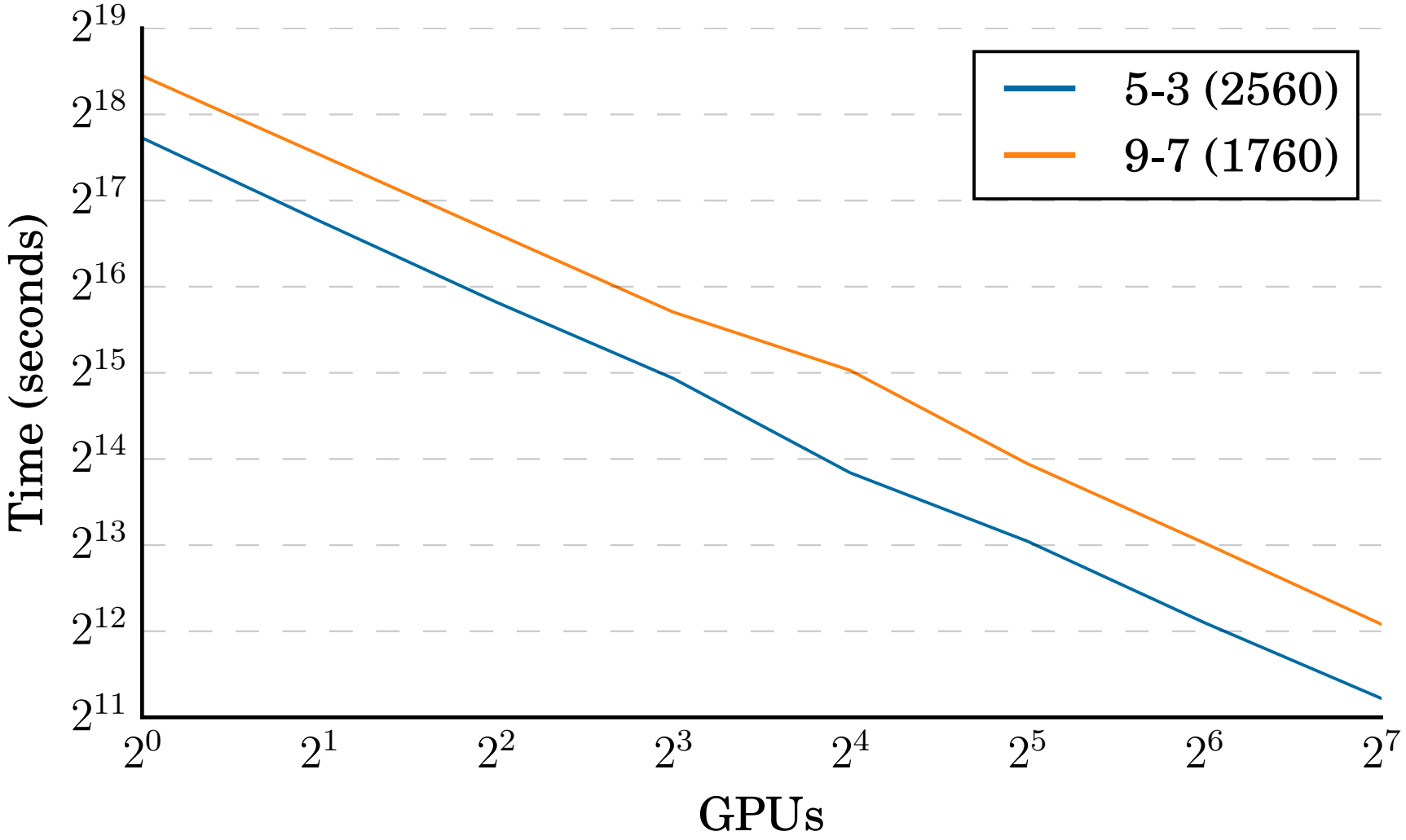
NVIDIA, “Whitepaper: GPU-based deep learning inference: A performance and power analysis.”

Tegra X1 vs Core i7



NVIDIA, “Whitepaper: GPU-based deep learning inference: A performance and power analysis.”

Parallel GPUs on Deep Speech 2



Baidu, Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, 2015

Summary of GPUs

- Titan X ~6x faster, 4x more efficient than Xeon E5
- TX1 11.5x more efficient than Core i7
- On inference
- Larger gains on training

- Data parallelism scales easily to 16GPUs
- With some effort, linear speedup to 128GPUs

Reducing precision

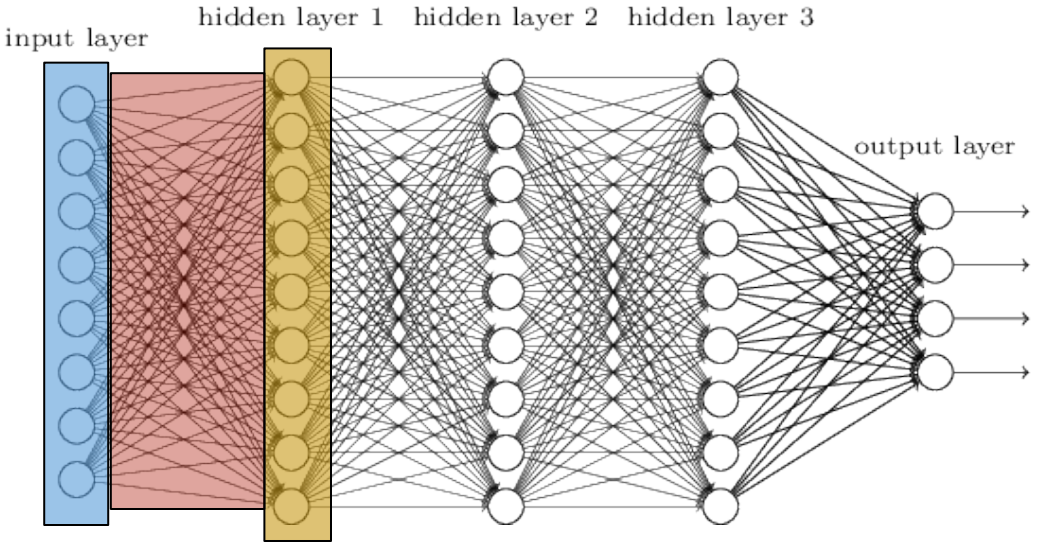
Reduces storage

Reduces energy

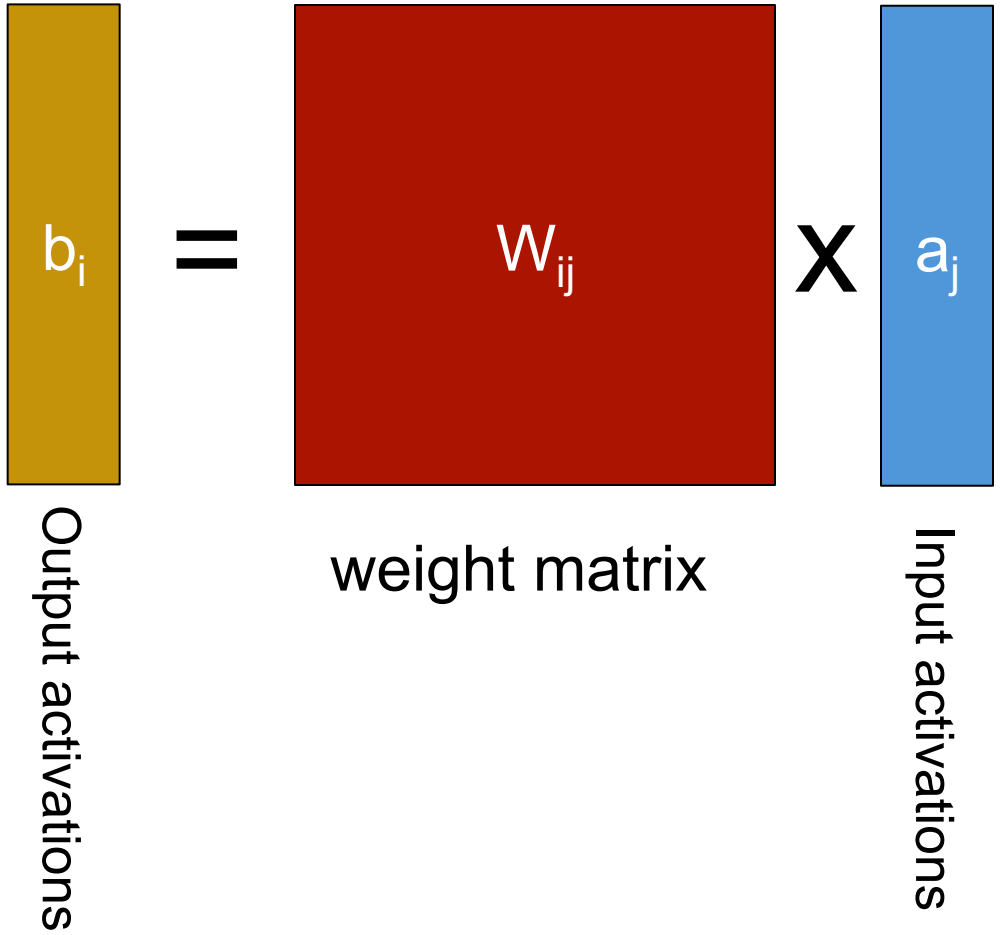
Improves performance

Has little effect on accuracy – to a point

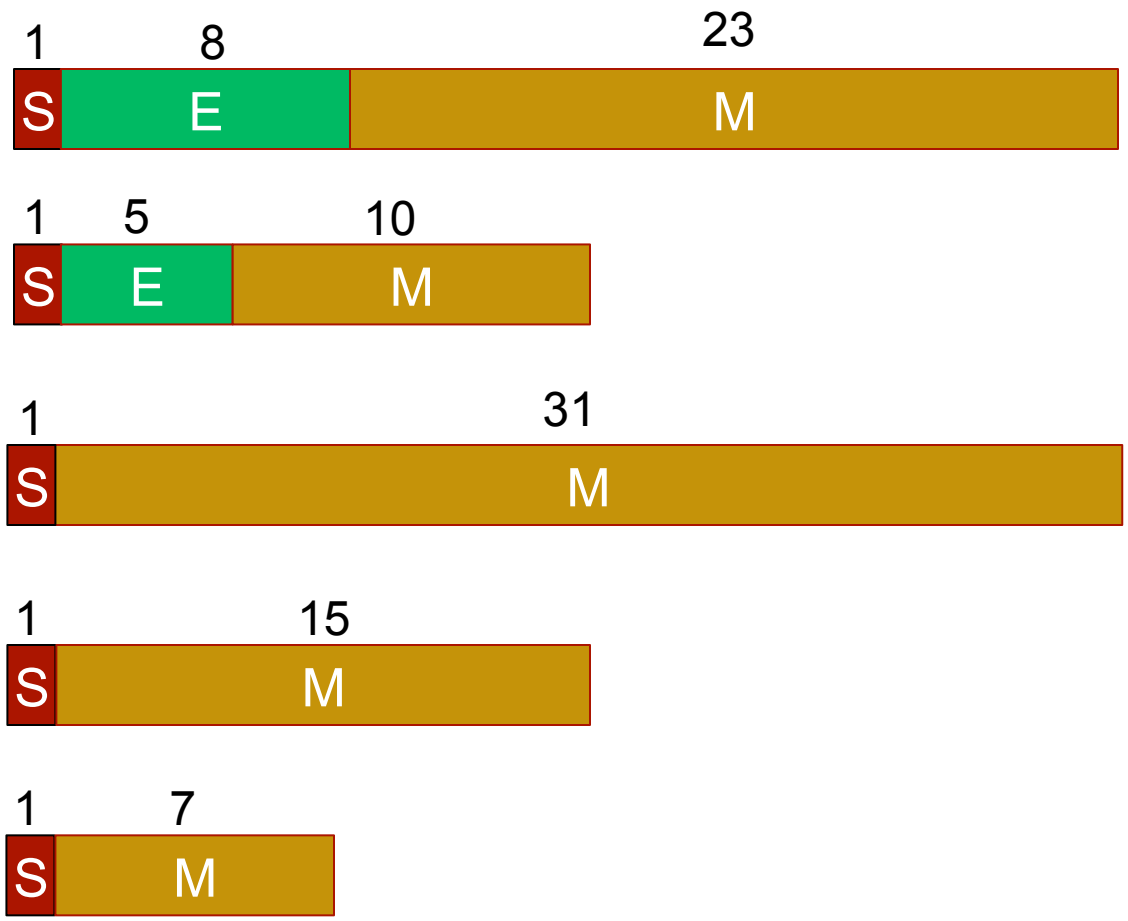
DNN, key operation is dense M x V



$$b_i = f\left(\sum_j w_{ij}\, a_i\right)$$



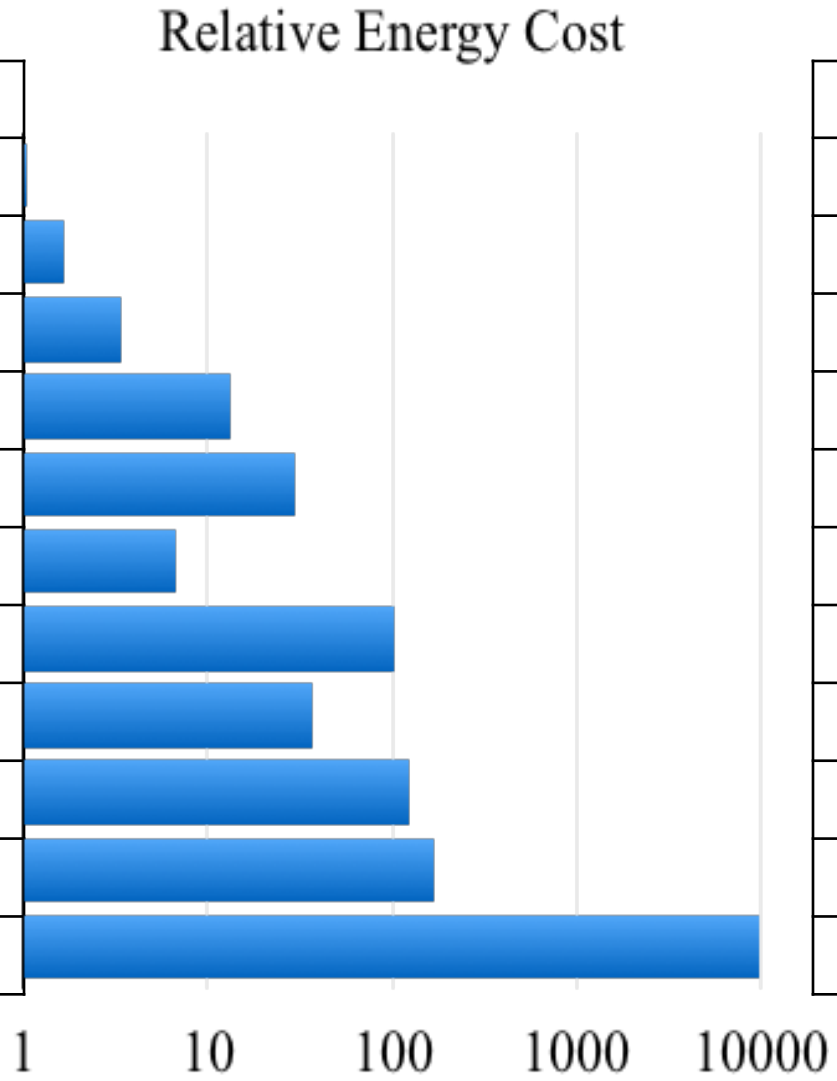
Number Representation

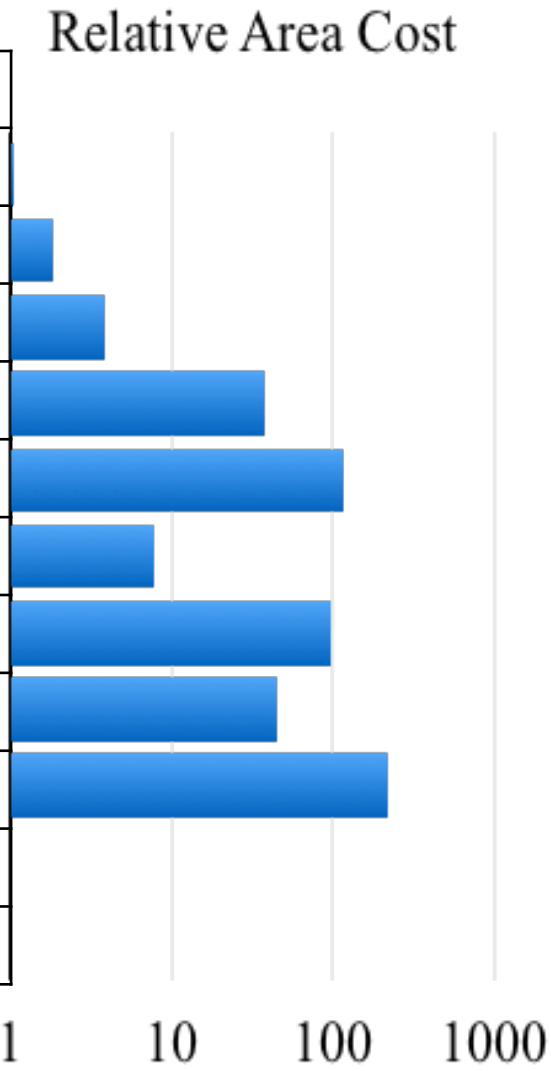


	Range	Accuracy
FP32	10^{-38} - 10^{38}	.000006%
FP16	$6x10^{-5}$ - $6x10^{4}$	.05%
Int32	0 – $2x10^{9}$	½
Int16	0 – $6x10^{4}$	½
Int8	0 – 127	½

Cost of Operations

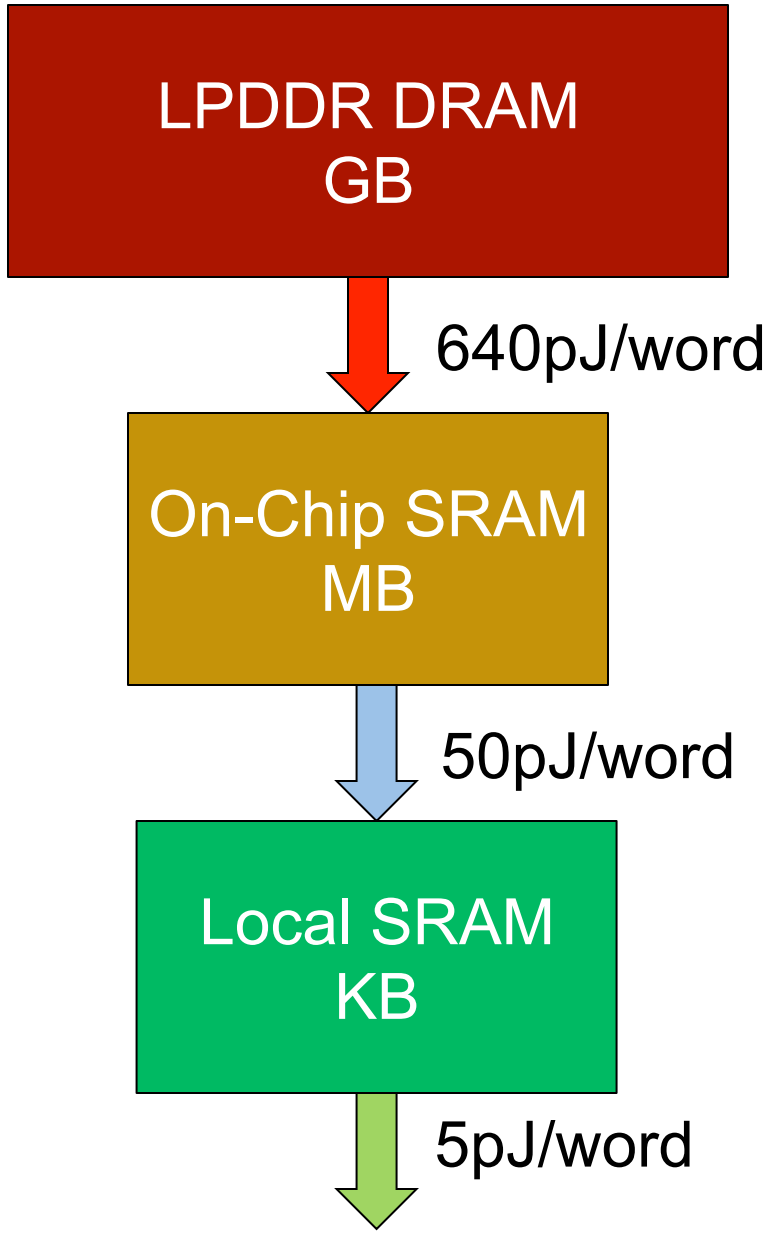
Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A





Energy numbers are from Mark Horowitz “Computing’s Energy Problem (and what we can do about it)”, ISSCC 2014
Area numbers are from synthesized result using Design Compiler under TSMC 45nm tech node. FP units used DesignWare Library.

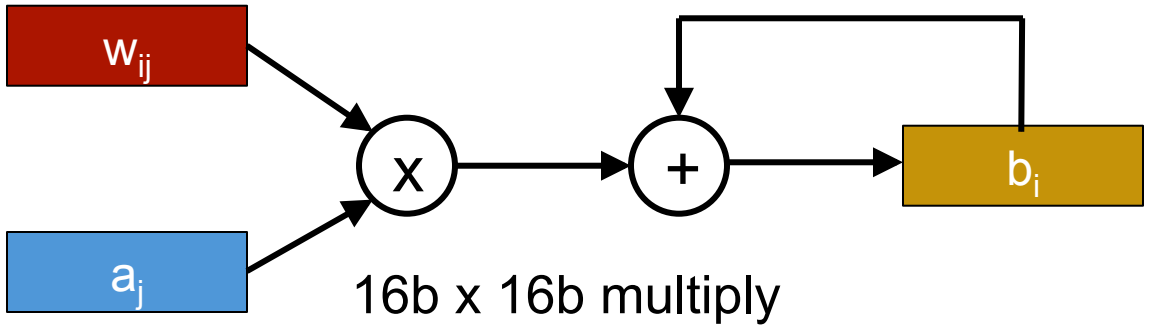
The Importance of Staying Local



Mixed Precision

Store weights as 4b using Trained quantization, decode to 16b

Store activations as 16b



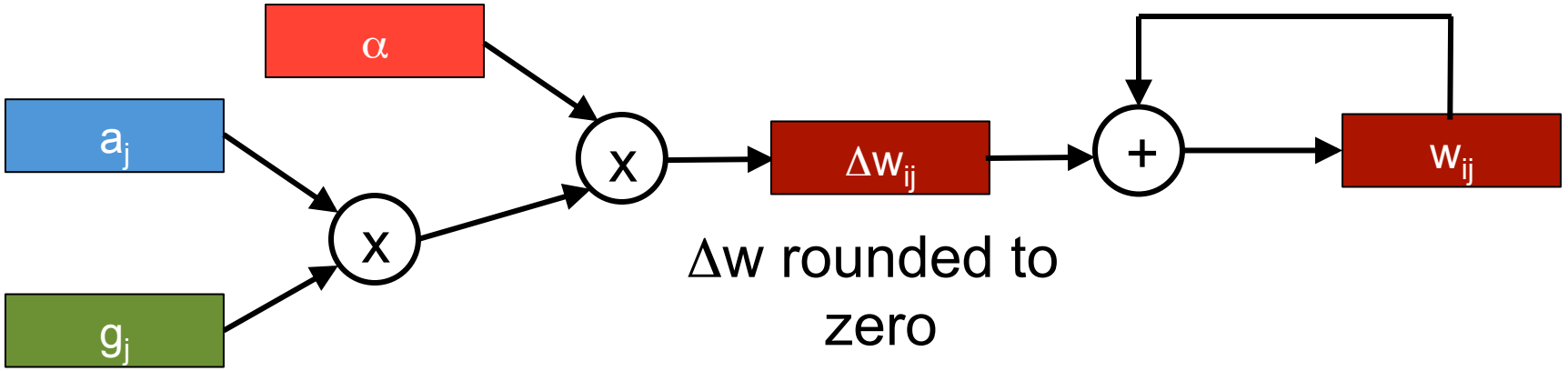
16b x 16b multiply
round result to 16b

accumulate 24b or 32b to avoid saturation

Batch normalization important to ‘center’ dynamic range

Weight Update

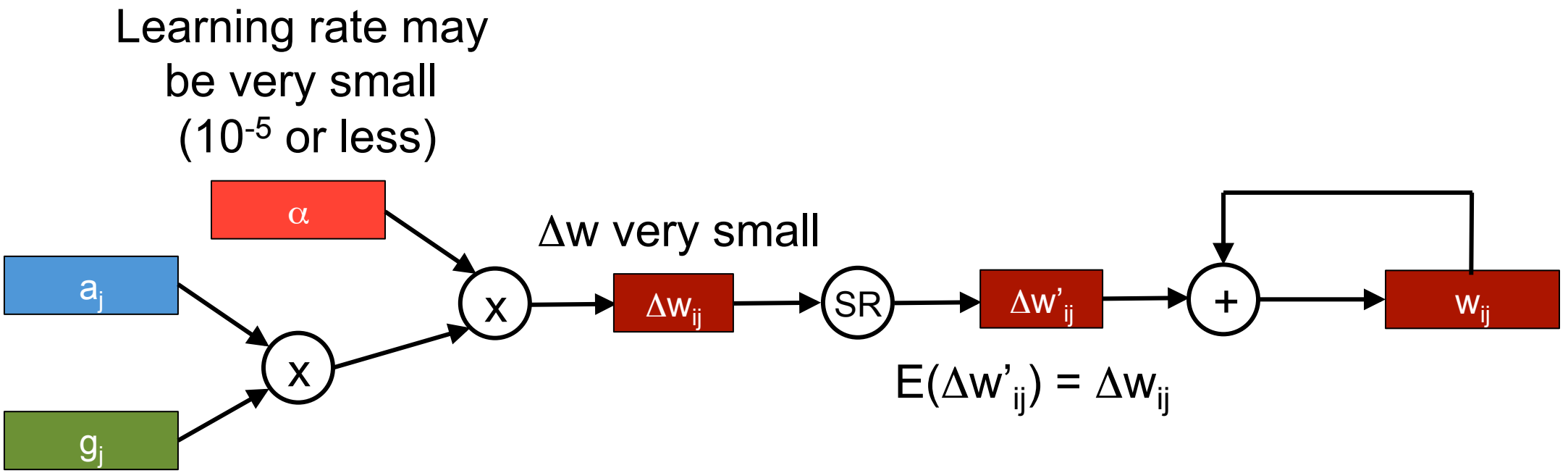
Learning rate may be very small (10^{-5} or less)



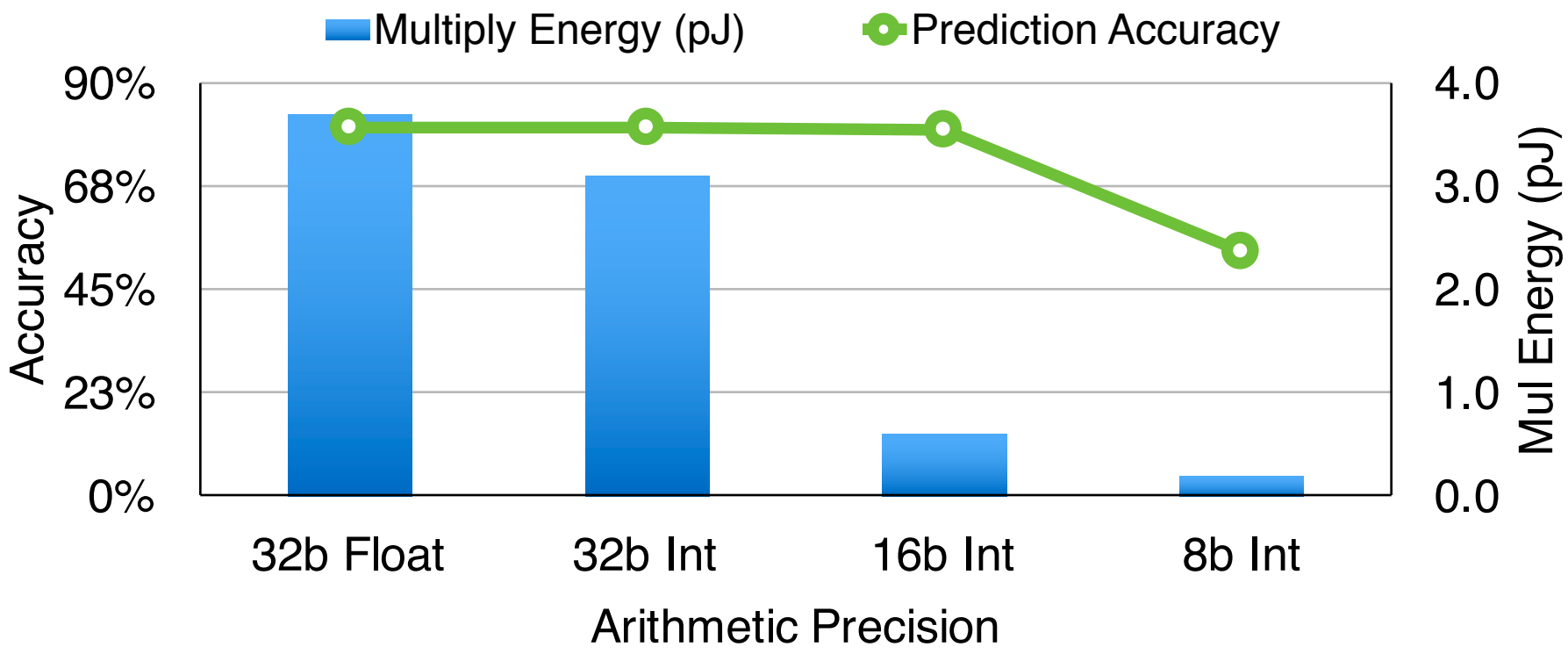
Δw rounded to zero

No learning!

Stochastic Rounding



Reduced Precision for Inference

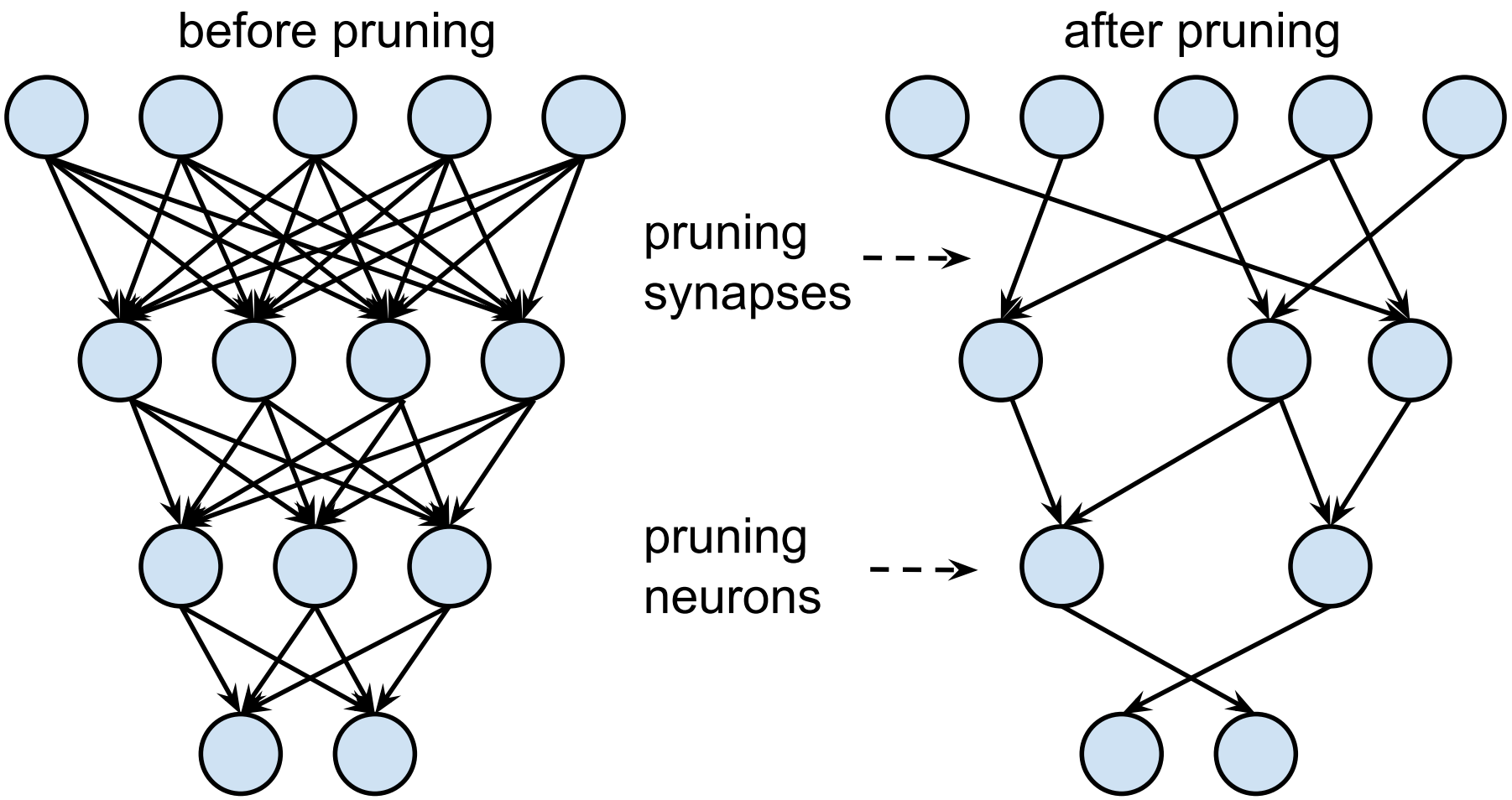


Summary of Reduced Precision

- Can save memory capacity, memory bandwidth, memory power, and arithmetic power by using smaller numbers
- FP16 works with little effort
  - 2x gain in memory, 4x in multiply power
- With care, one can use
  - 8b for convolutions
  - 4b for fully-connected layers
- Batch normalization – important to ‘center’ ranges
- Stochastic rounding – important to retain small increments

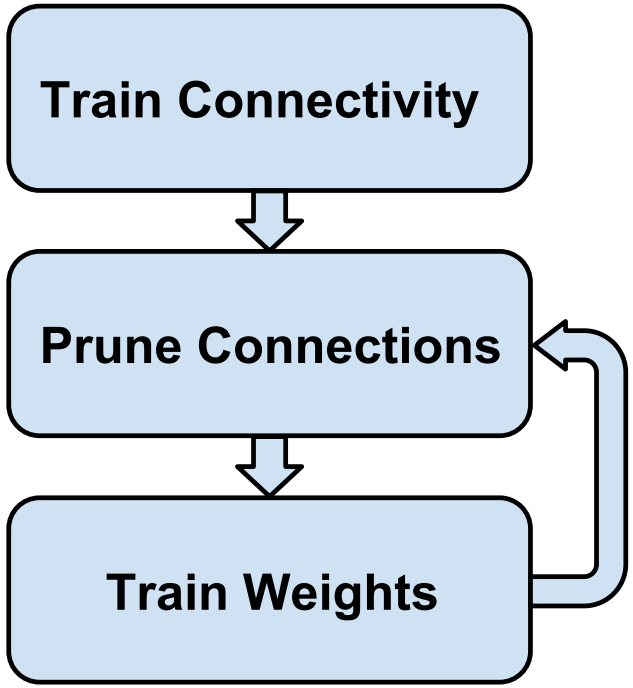
Reducing Size of Network Reduces Work and Storage

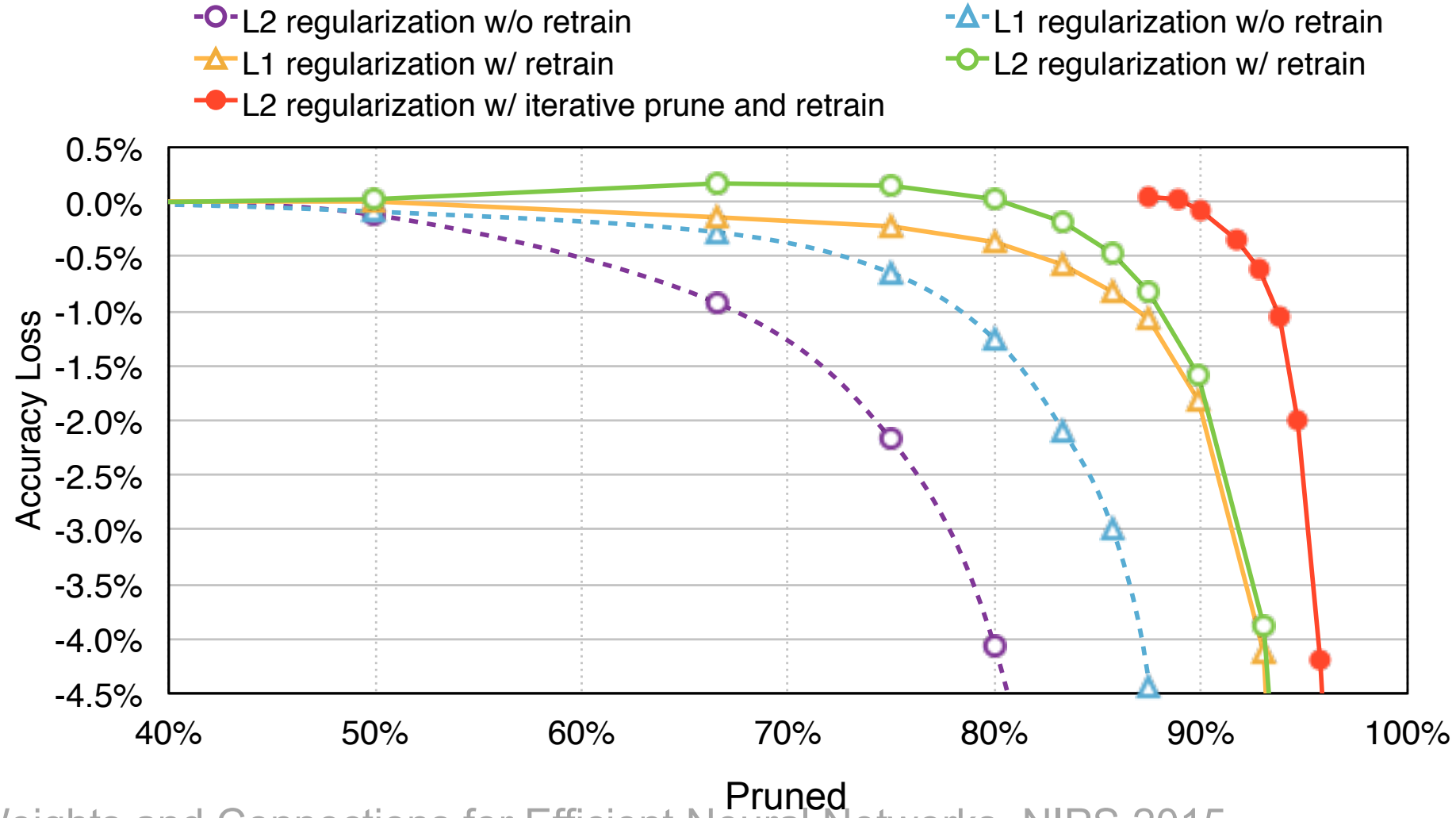
Pruning



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

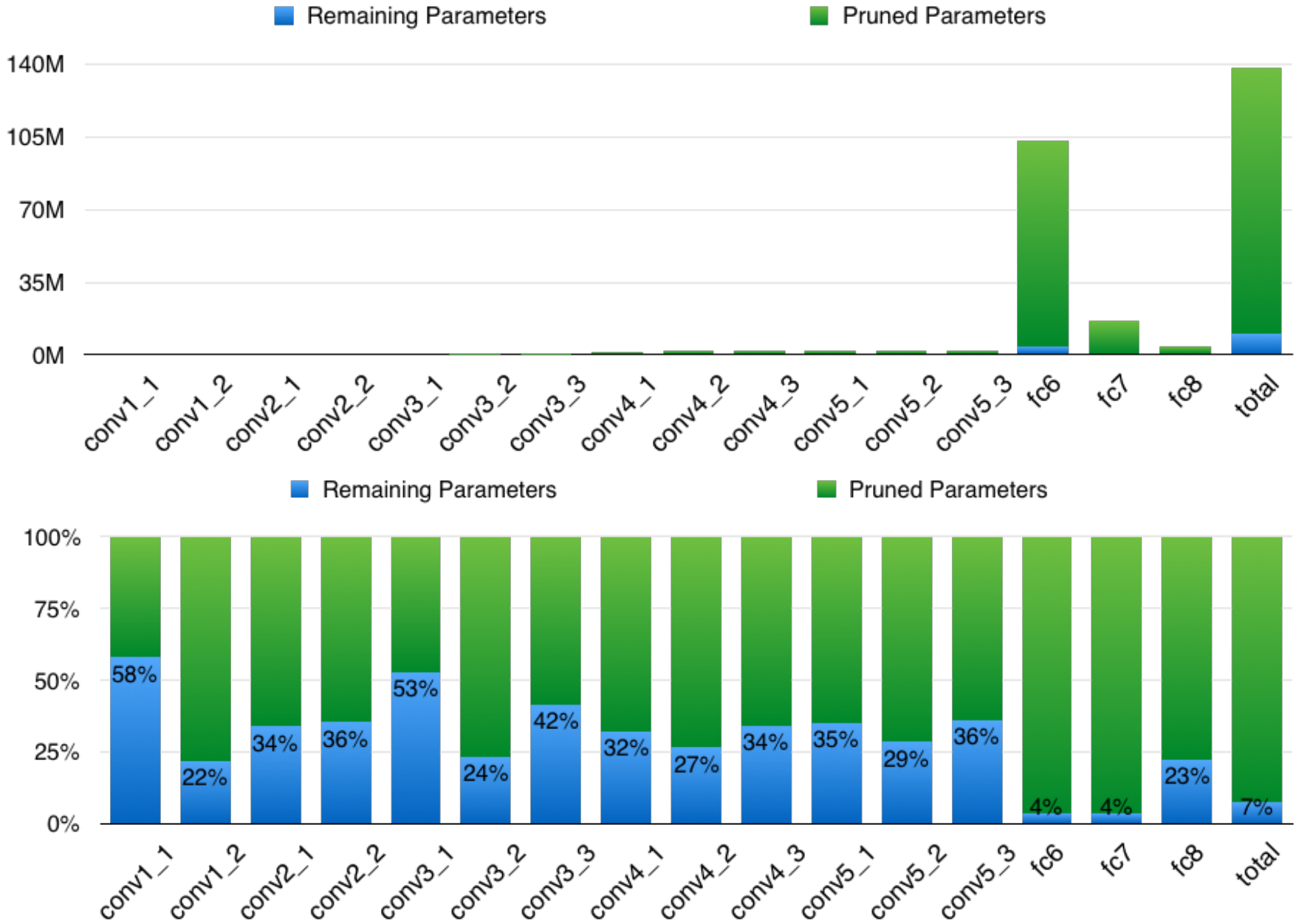
Retrain to Recover Accuracy





Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

Pruning of VGG-16



Pruning Neural Talk and LSTM

- **Original**: a basketball player in a white uniform is playing with a ball
- **Pruned 90%**: a basketball player in a white uniform is playing with a basketball

- **Original** : a brown dog is running through a grassy field
- **Pruned 90%**: a brown dog is running through a grassy area

- **Original** : a man is riding a surfboard on a wave
- **Pruned 90%**: a man in a wetsuit is riding a wave on a beach

- **Original** : a soccer player in red is running in the field
- **Pruned 95%**: a man in a red shirt and black and white black shirt is running through a field

Speedup of Pruning on CPU/GPU

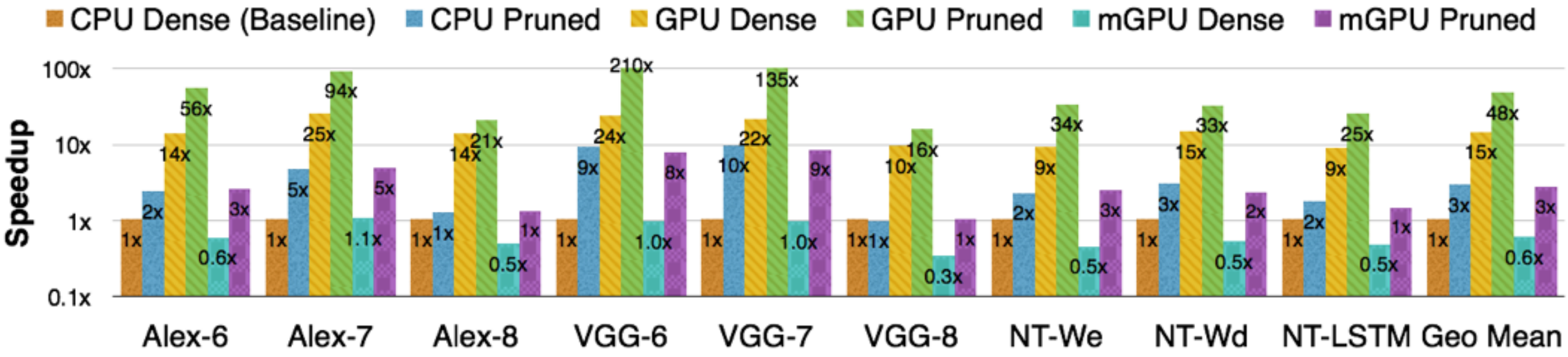
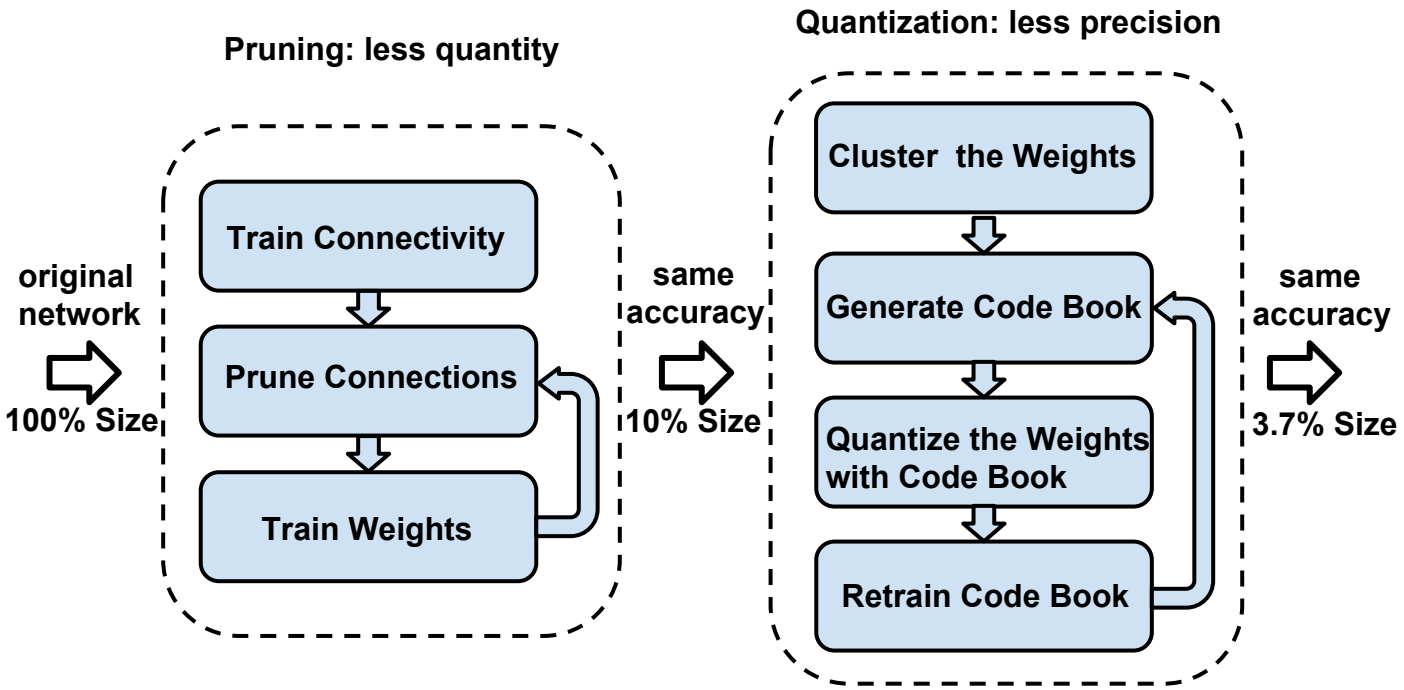


Figure 9: Compared with the original network, pruned network layer achieved 3× speedup on CPU, 3.5× on GPU and 4.2× on mobile GPU on average. Batch size = 1 targeting real time processing. Performance number normalized to CPU.

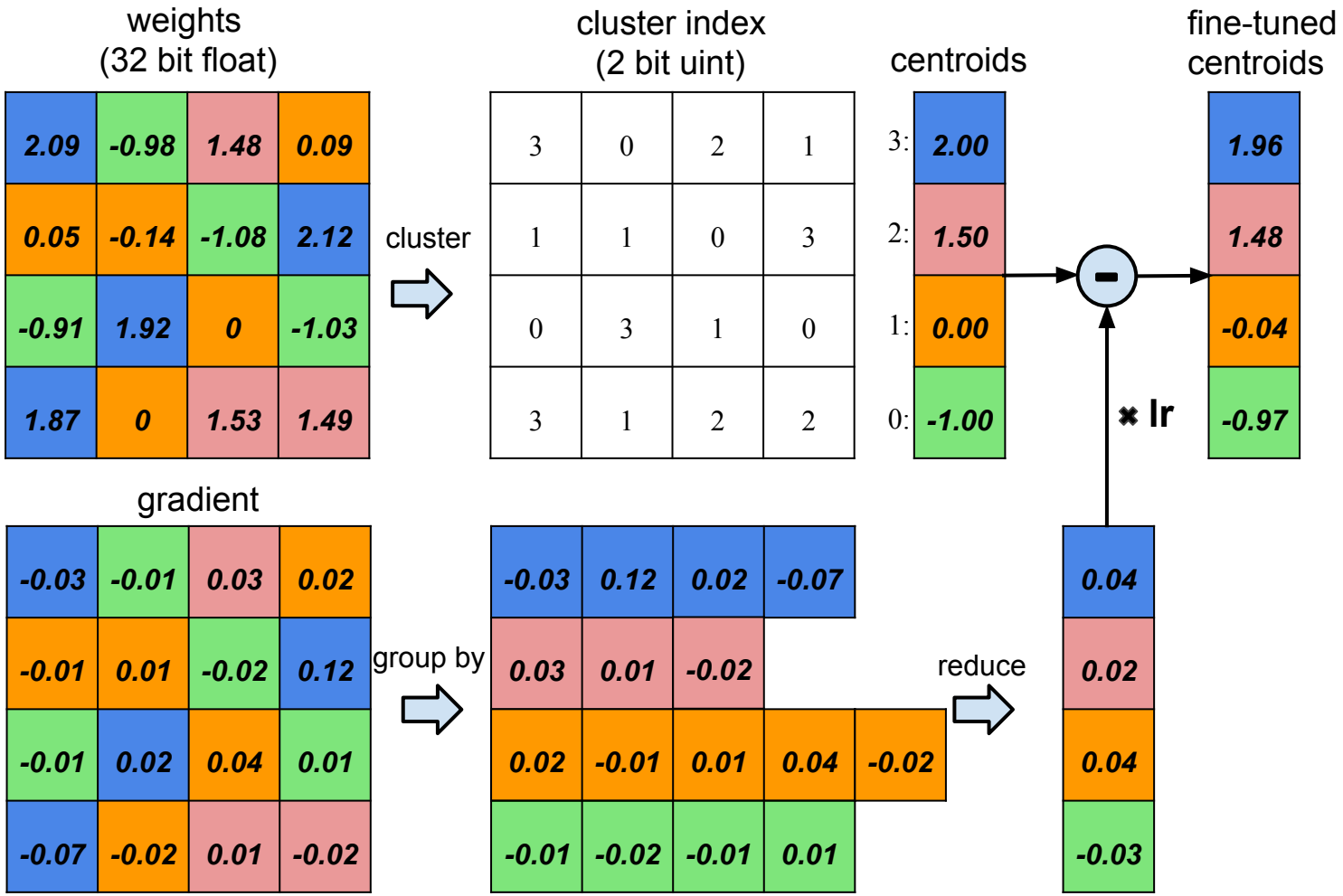
Intel Core i7 5930K: MKL CBLAS GEMV, MKL SPBLAS CSRMV
NVIDIA GeForce GTX Titan X: cuBLAS GEMV, cuSPARSE CSRMV
NVIDIA Tegra K1: cuBLAS GEMV, cuSPARSE CSRMV

Trained Quantization
(Weight Sharing)



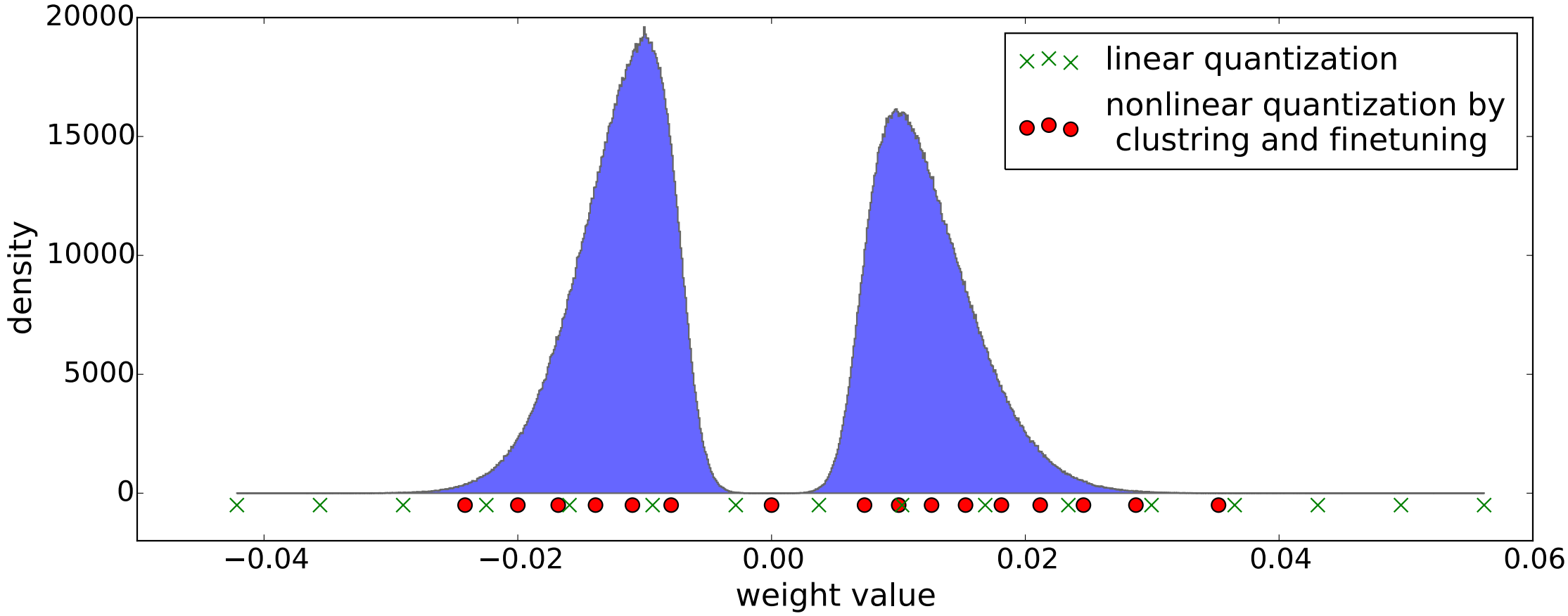
Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, arXiv 2015

Weight Sharing via K-Means



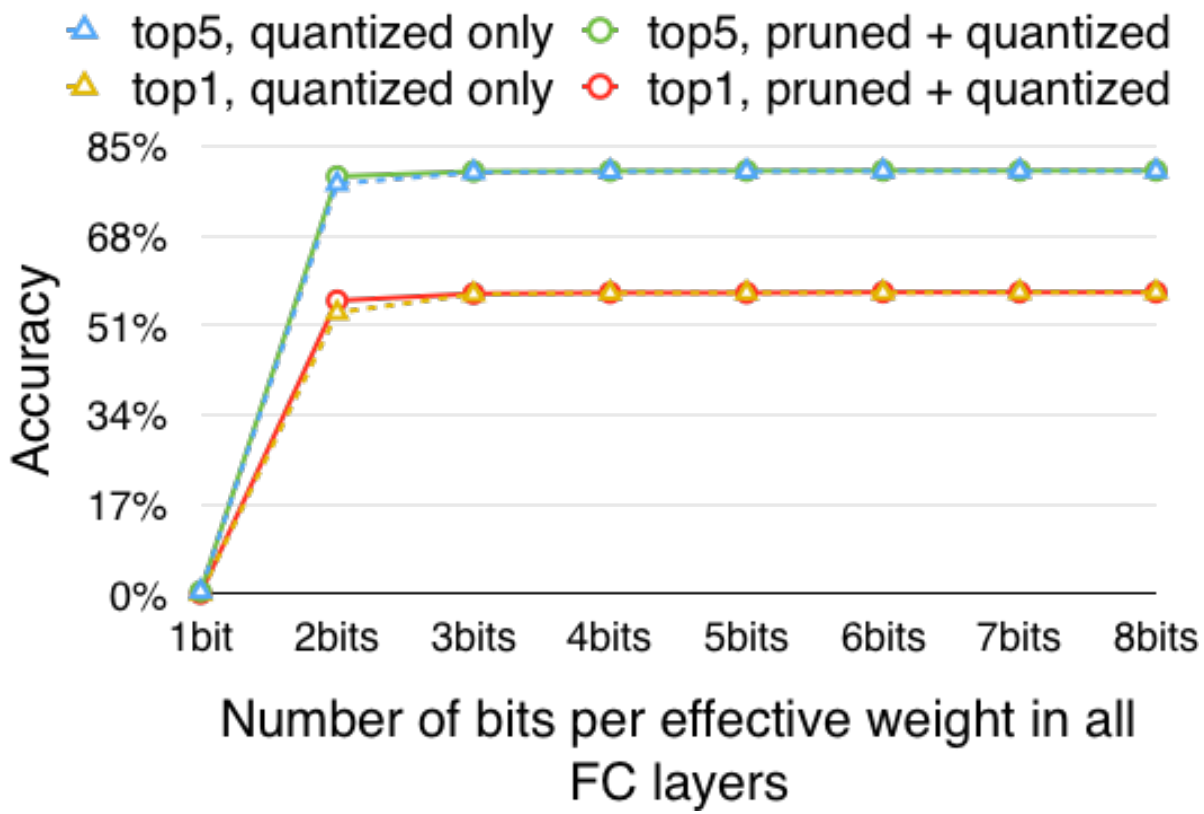
Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, arXiv 2015

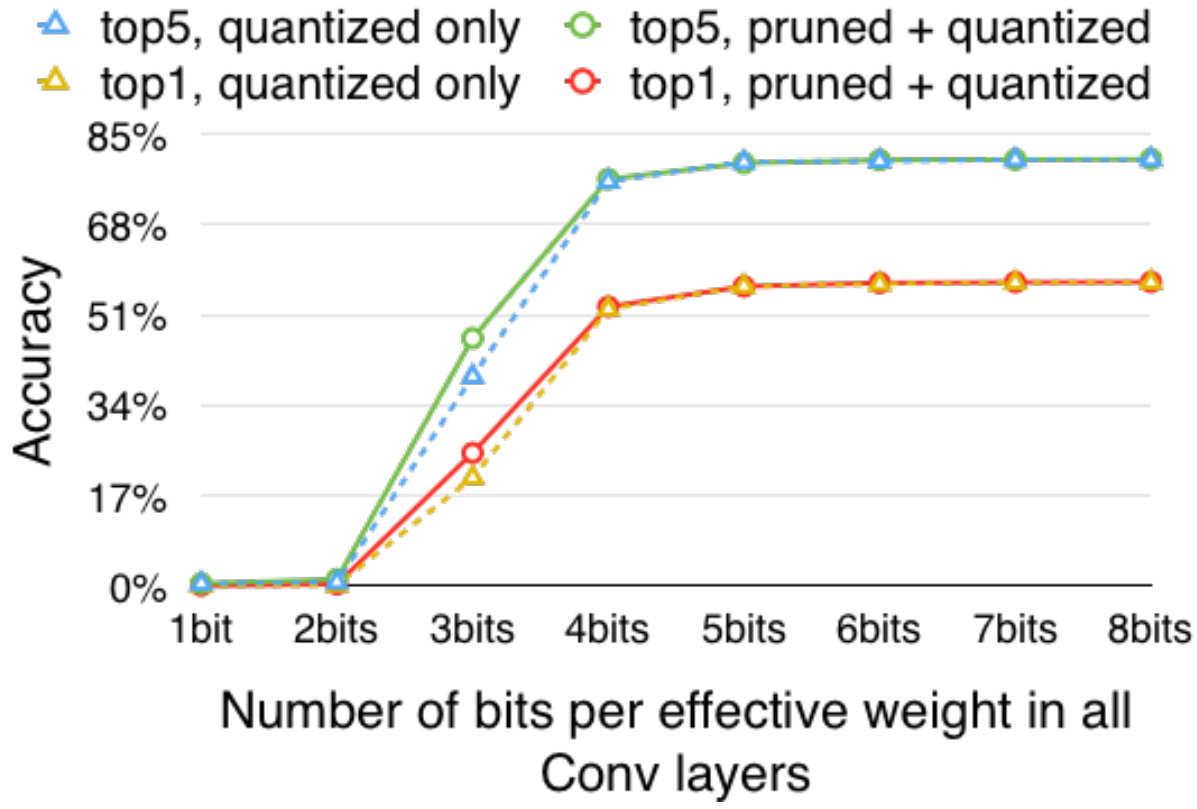
Trained Quantization



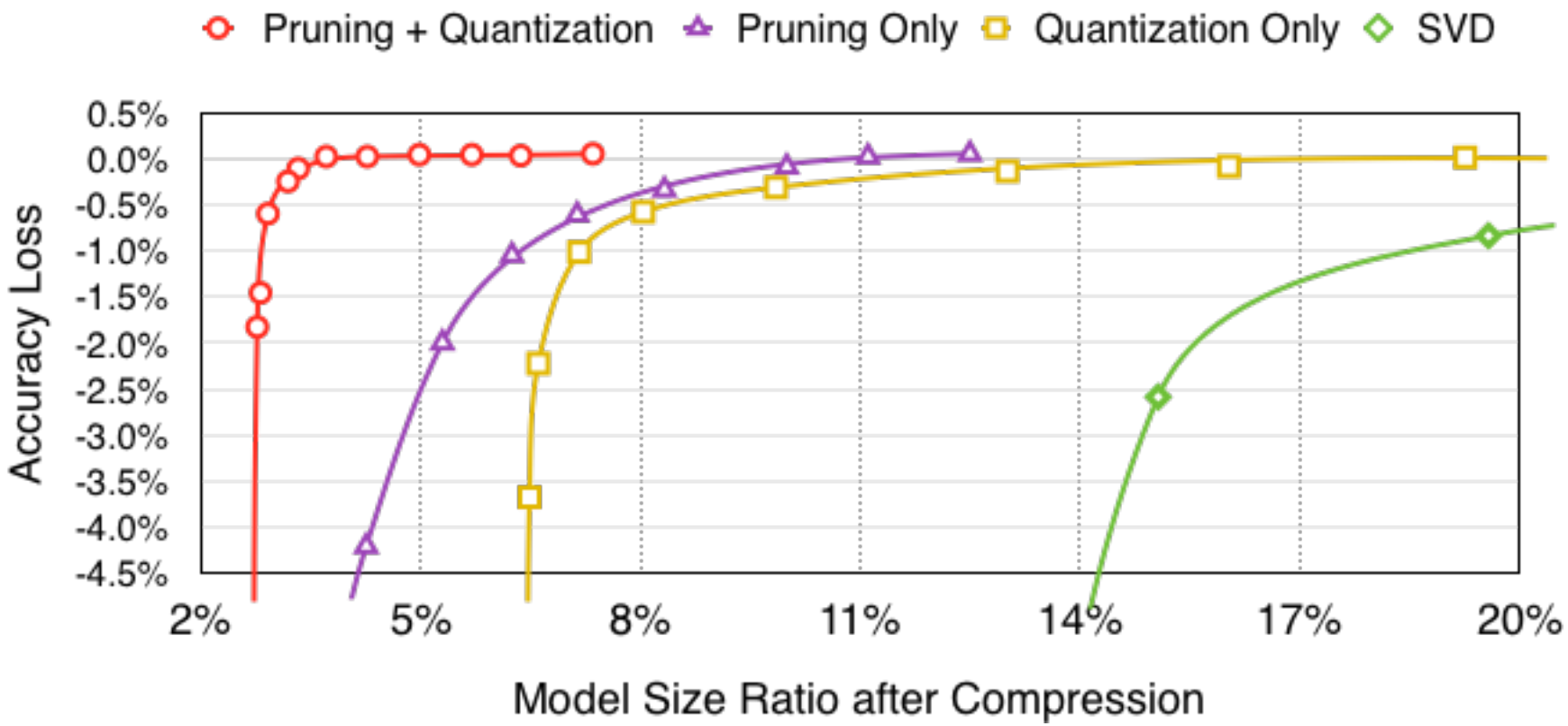
Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, arXiv 2015

Bits per Weight





Pruning + Trained Quantization



30x – 50x Compression Means

- Complex DNNs can be put in mobile applications (<100MB total)
  - 1GB network (250M weights) becomes 20-30MB
- Memory bandwidth reduced by 30-50x
  - Particuarly for FC layers in real-time applications with no reuse
- Memory working set fits in on-chip SRAM
  - 5pJ/word access vs 640pJ/word

To be maximally efficient use special-purpose hardware

Unless you are memory limited

Diannao (Electric Brain)

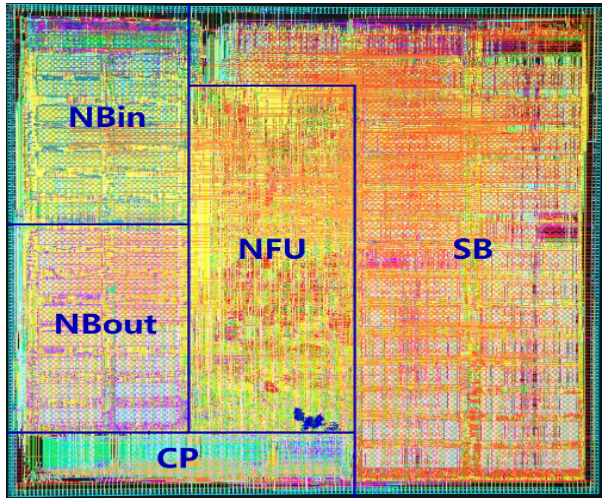


Figure 15. *Layout (65nm).*

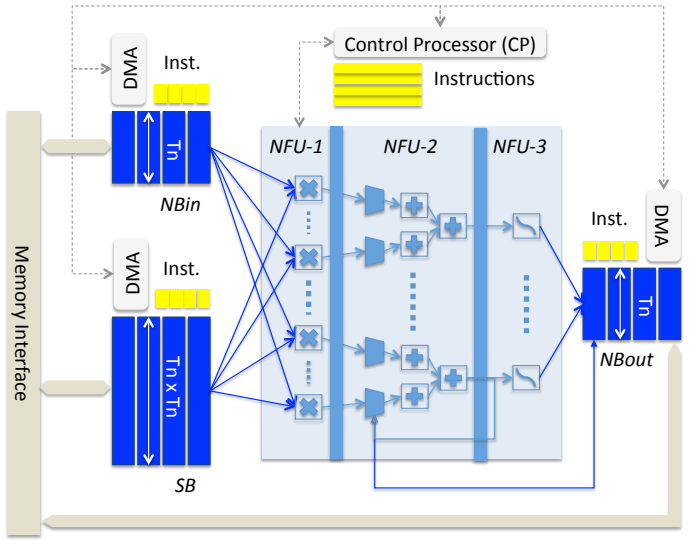


Figure 11. *Accelerator.*

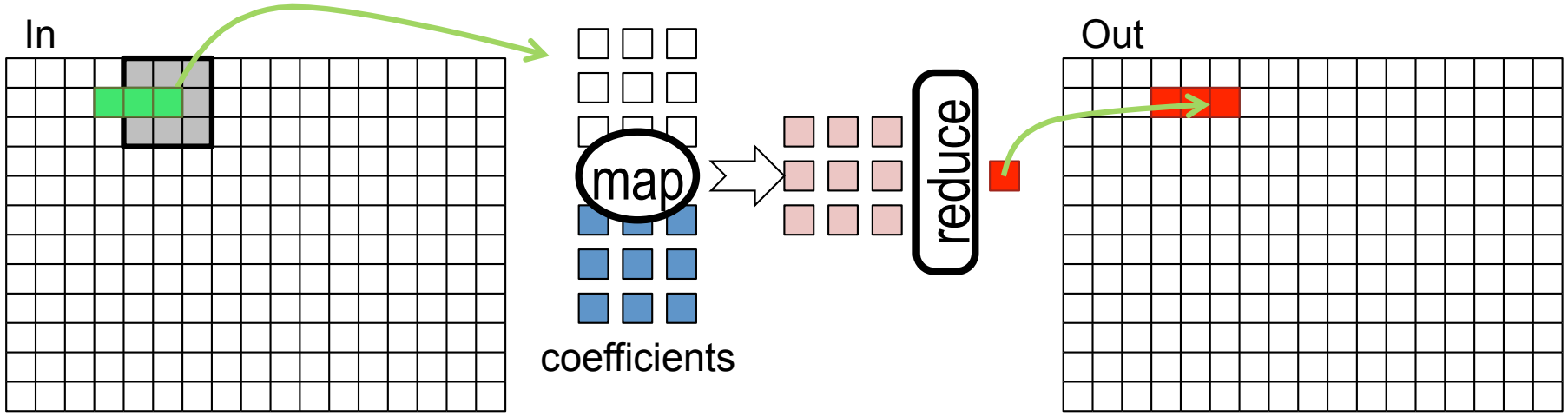
Component or Block	Area in μm^2	(%)	Power in mW	(%)	Critical path in ns
ACCELERATOR	3,023,077		485		1.02
Combinational	608,842	(20.14%)	89	(18.41%)	
Memory	1,158,000	(38.31%)	177	(36.59%)	
Registers	375,882	(12.43%)	86	(17.84%)	
Clock network	68,721	(2.27%)	132	(27.16%)	
Filler cell	811,632	(26.85%)			
SB	1,153,814	(38.17%)	105	(22.65%)	
NBin	427,992	(14.16%)	91	(19.76%)	
NBout	433,906	(14.35%)	92	(19.97%)	
NFU	846,563	(28.00%)	132	(27.22%)	
CP	141,809	(5.69%)	31	(6.39%)	
AXIMUX	9,767	(0.32%)	8	(2.65%)	
Other	9,226	(0.31%)	26	(5.36%)	

Table 6. *Characteristics of accelerator and breakdown by component type (first 5 lines), and functional block (last 7 lines).*

- Diannao improved CNN computation efficiency by using dedicated functional units and memory buffers optimized for the CNN workload.
- Multiplier + adder tree + shifter + non-linear lookup orchestrated by instructions
- Weights in off-chip DRAM
- 452 GOP/s, 3.02 mm^2 and 485 mW

Chen et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning, ASPLOS 2014

Convolution Engine

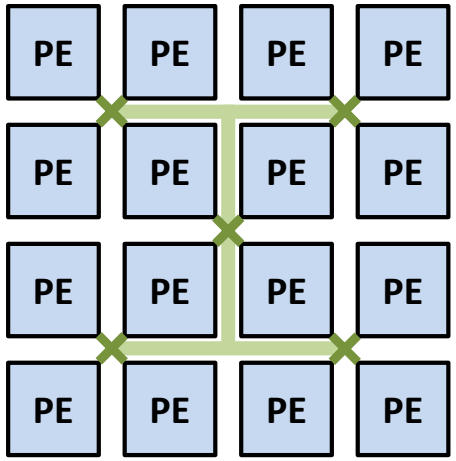


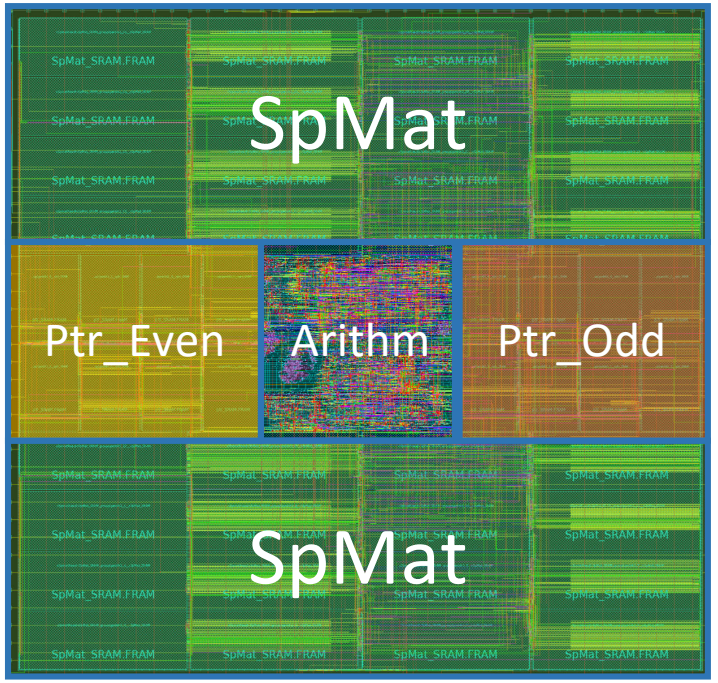
- Convolution Engine (CE), is specialized for the convolution-like data-flow that is common in image processing.
- CE achieves energy efficiency by capturing data reuse patterns, eliminating data transfer overheads, and enabling a large number of operations per memory access.
- With restricted the domain in image and video processing, flexible convolution engine improves improves energy and area efficiency by 8-15x over a SIMD engine.

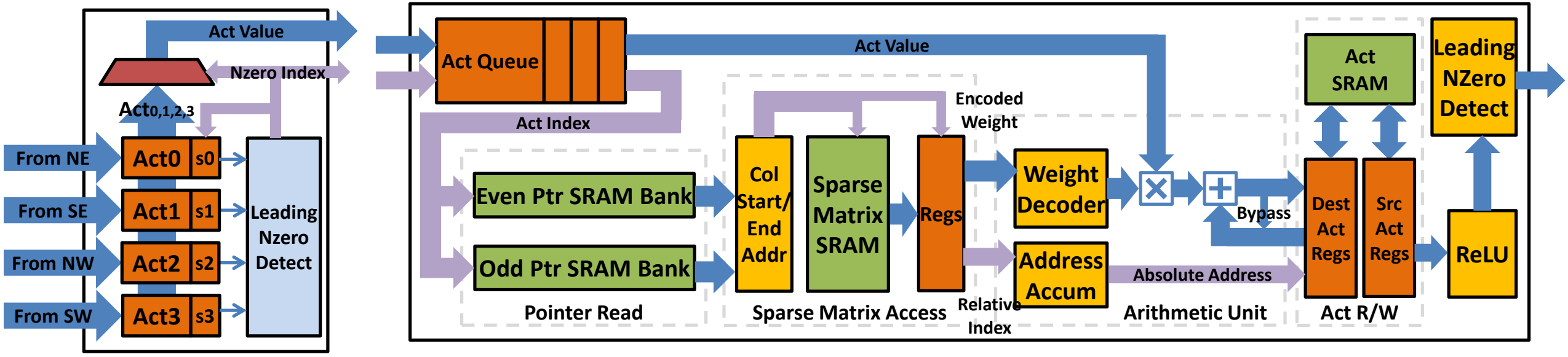
Wajahat Qadeer et.al, Convolution Engine: Balancing Efficiency & Flexibility in Specialized Computing

Efficient Inference Engine

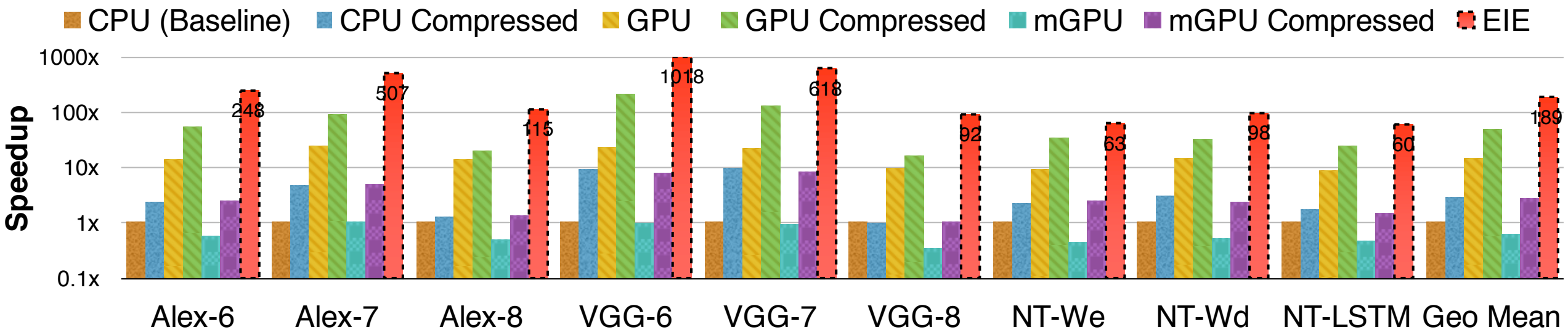
	Power (mW)	(%)	Area (μm^2)	(%)
Total	9.157		638,024	
memory	5.416	(59.15%)	594,786	(93.22%)
clock network	1.874	(20.46%)	866	(0.14%)
register	1.026	(11.20%)	9,465	(1.48%)
combinational	0.841	(9.18%)	8,946	(1.40%)
filler cell			23,961	(3.76%)
Act_queue	0.112	(1.23%)	758	(0.12%)
PtrRead	1.807	(19.73%)	121,849	(19.10%)
SpmatRead	4.955	(54.11%)	469,412	(73.57%)
ArithmUnit	1.162	(12.68%)	3,110	(0.49%)
ActRW	1.122	(12.25%)	18,934	(2.97%)
filler cell			23,961	(3.76%)



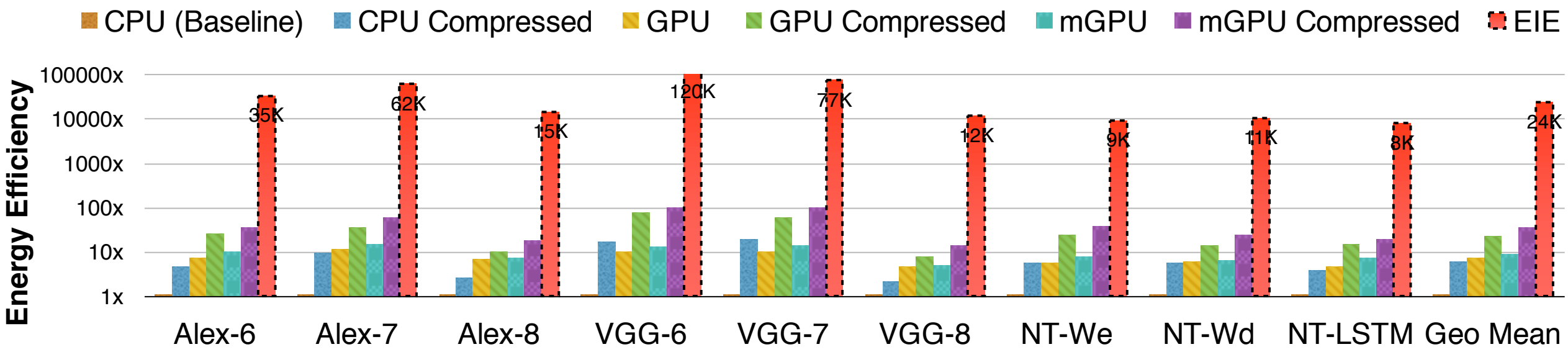




Speedup


CPU (Baseline)
CPU Compressed
GPU
GPU Compressed
mGPU
mGPU Compressed
EIE
Speedup
1000x
100x
10x
1x
0.1x
248
507
115
1018
618
92
63
98
60
189
Alex-6
Alex-7
Alex-8
VGG-6
VGG-7
VGG-8
NT-We
NT-Wd
NT-LSTM
Geo Mean

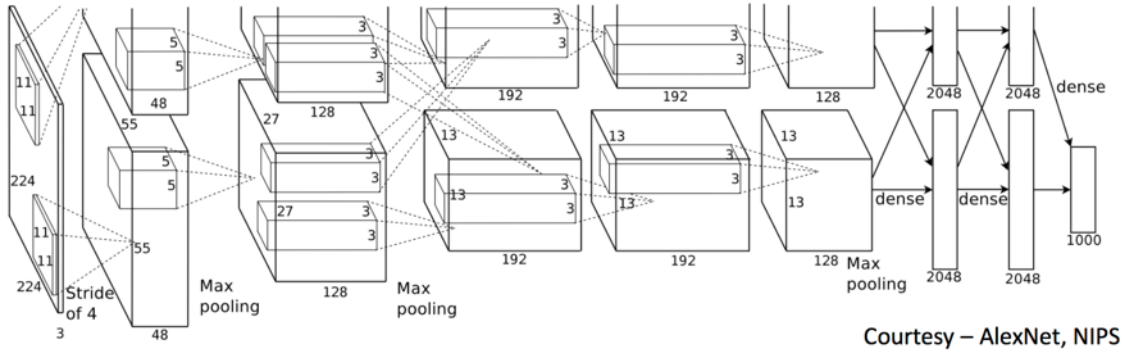
Energy Efficiency



Bottom Line

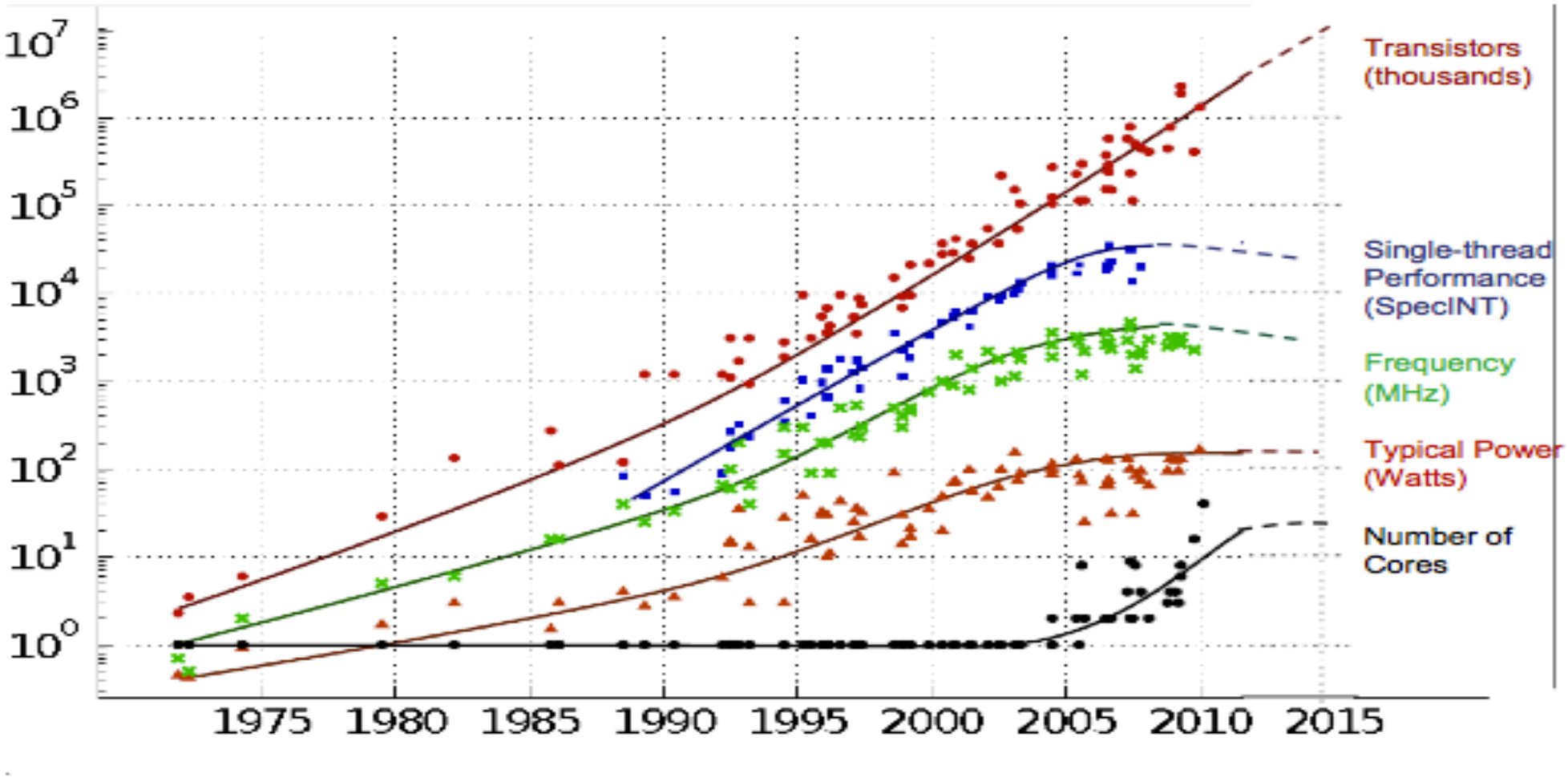
- Arithmetic perf/W of special purpose hardware is ~2x a GPU (FP16)
- Perf/W on memory limited layers (FC, not batch) is no better than GPU
- Big win from special-purpose hardware is
  - When entire network fits on chip
  - Decompressing highly-compressed networks
- Can do this with a GPU and HW compression

Hardware and Data enable DNNs



Courtesy – AlexNet, NIPS 2012

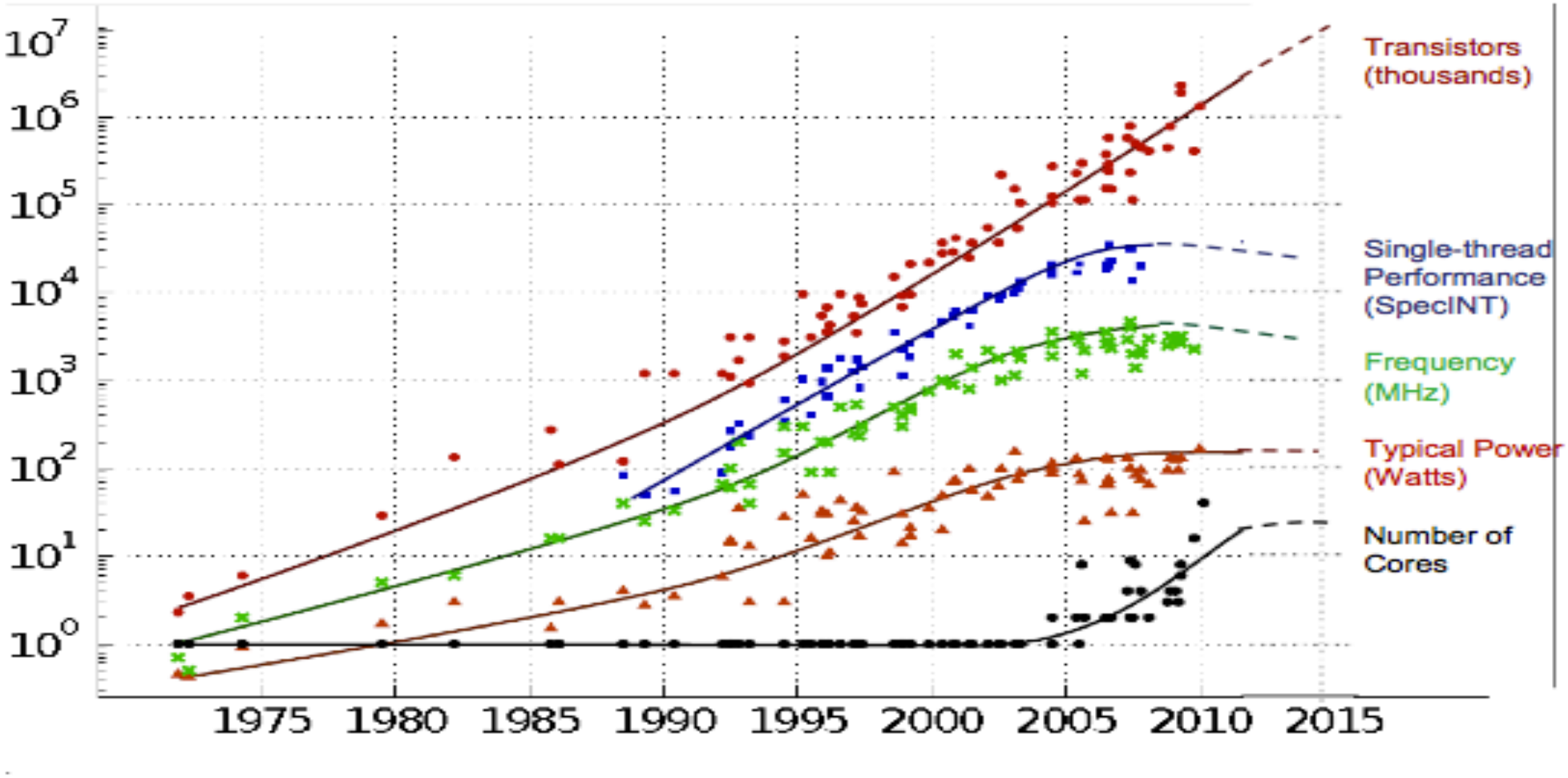
In 1990, CPUs had one 100 SpecINT Core



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

C Moore, *Data Processing in ExaScale-ClassComputer Systems,* Salishan, April 2011

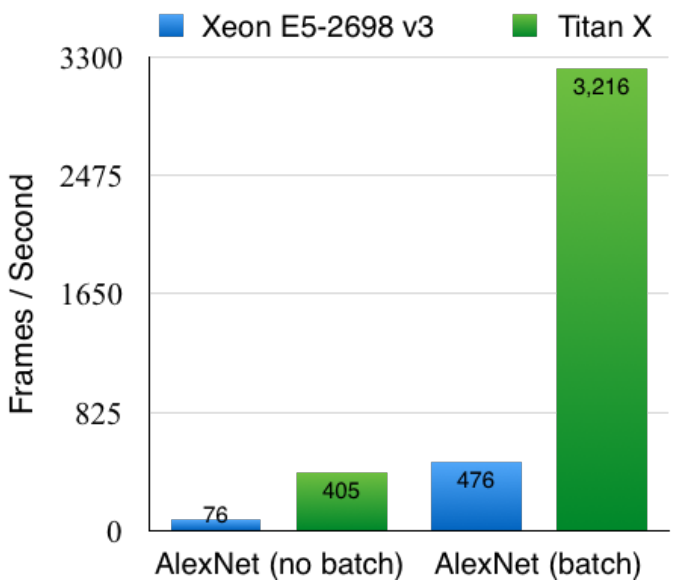
Today they have 6-8 30,000SpecINT cores (~200,000x) But Moore's Law is over…

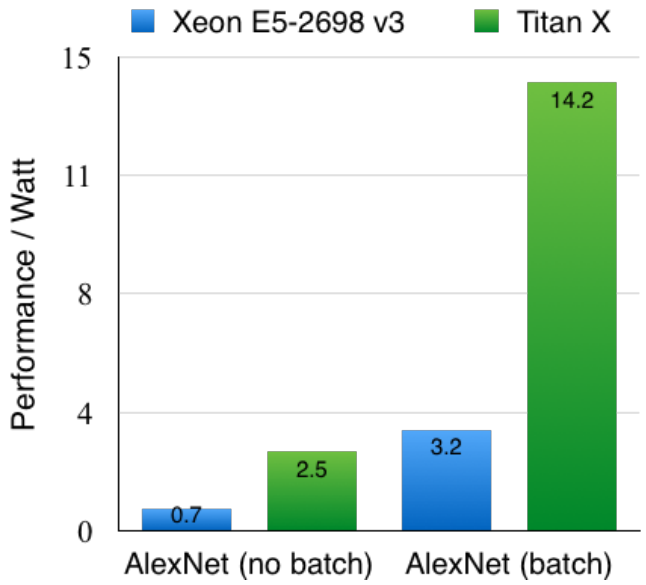


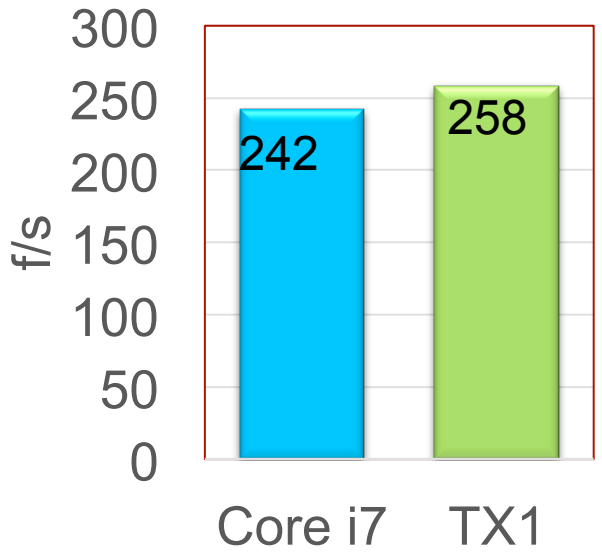
Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

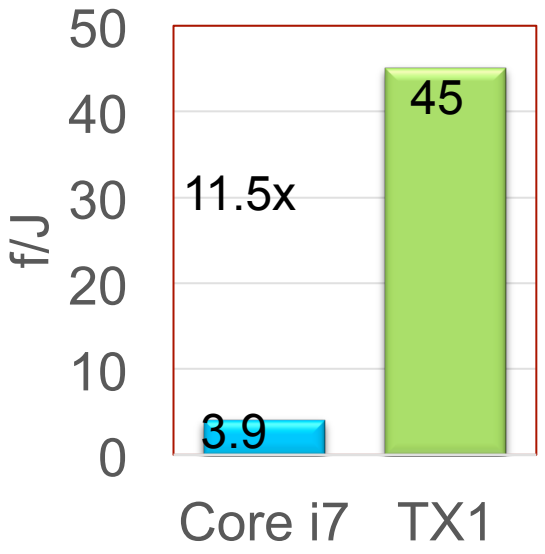
C Moore, *Data Processing in ExaScale-ClassComputer Systems,* Salishan, April 2011

GPUs give an additional 5-10x (2,000,000x)

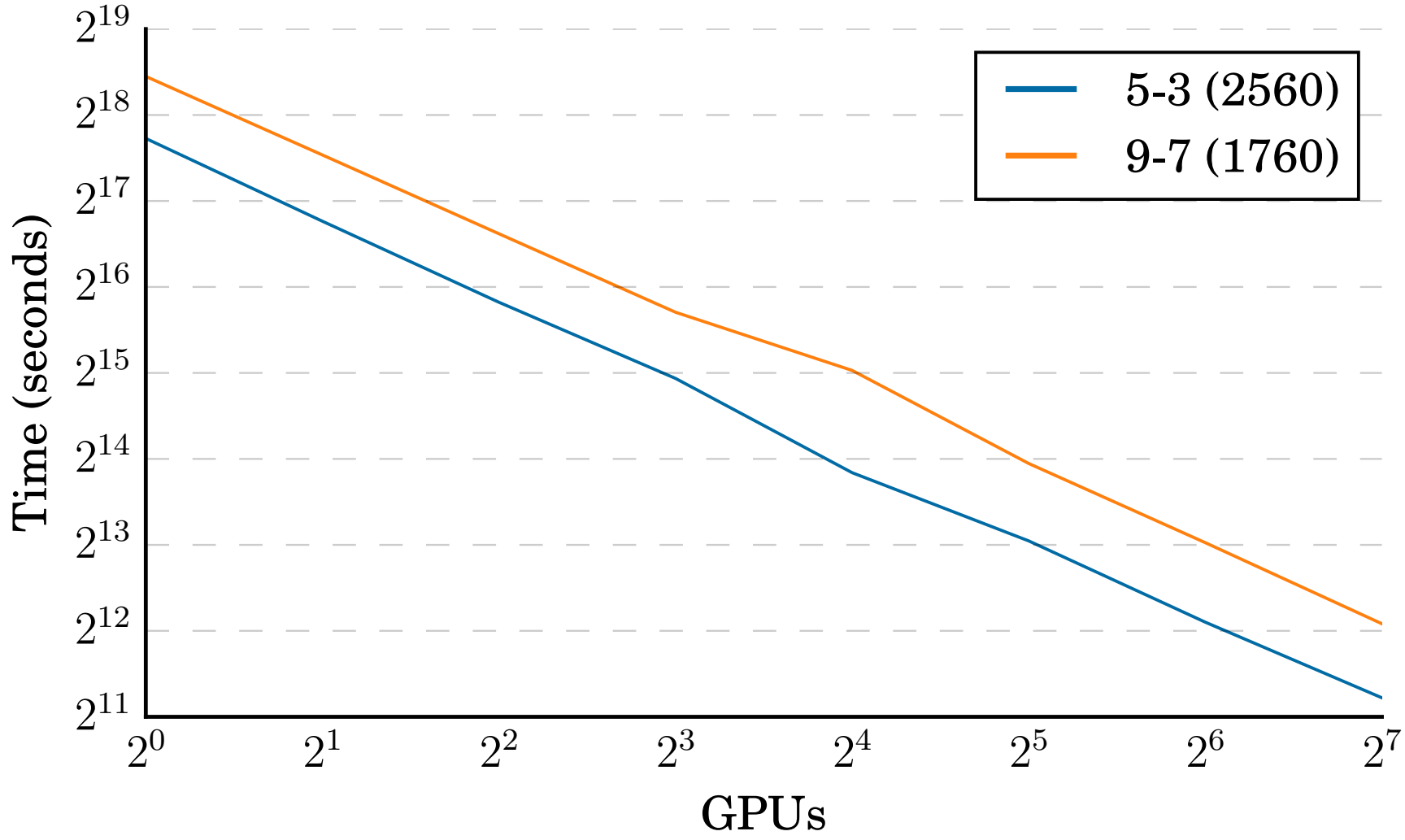






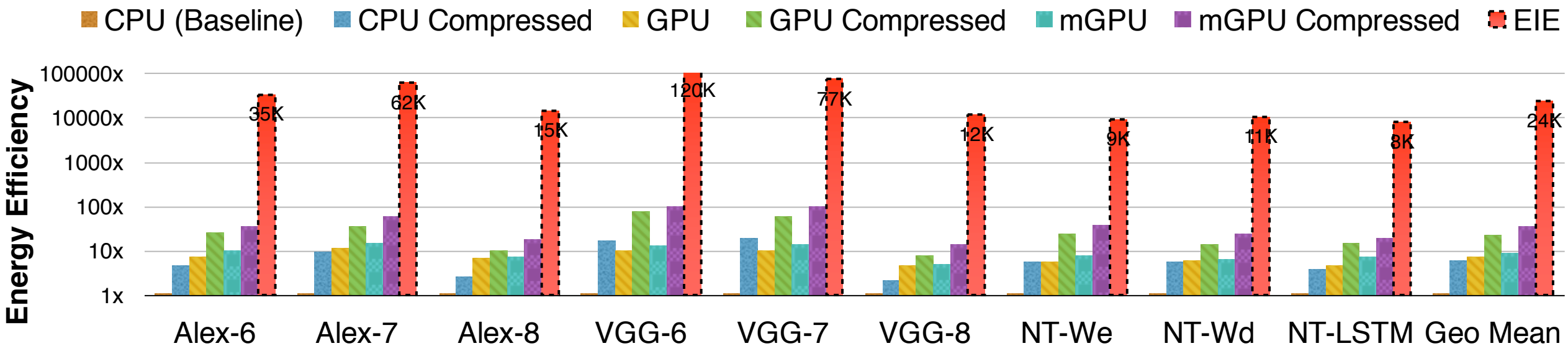


Data Parallelism Can get another 128x (256,000,000x) More with Model and Hyper-Parameter Parallelism



Baidu, Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, 2015

Special-Purpose Hardware Can Give another 100x (25,000,000,000x) Mostly from localizing memory



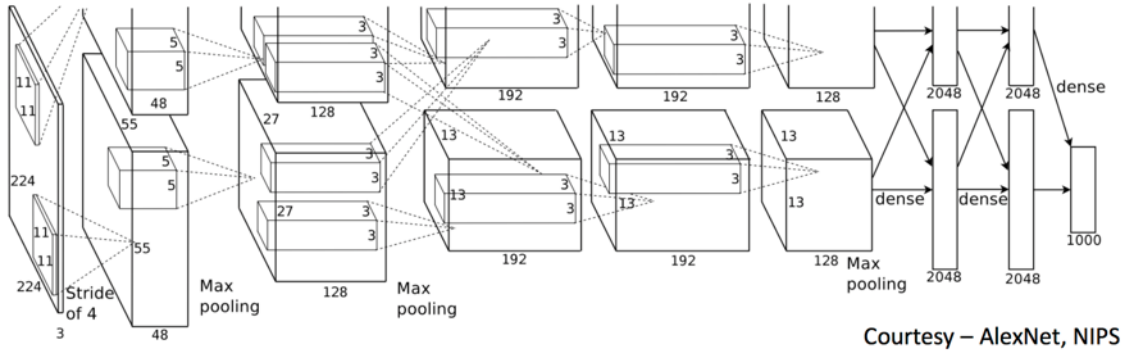
Accelerate the best algorithms
Prune the network
Compress the network
FFT convolutions

So what should you do?

- For training use clusters of 8-16GPUs
  - Best perf, perf/W, perf/$, and memory bandwidth
  - Easy parallelism
- For inference in the data center use single GPUs
  - Tesla M4 and M40
- For inference in mobile devices (Automotive, IoT)
  - Use a TX1 (11.5x perf/W of CPU)

- For the absolute best performance and efficiency use an ASIC
  - But make sure the model fits (memory limited ASICs no better than GPU)
  - And that your algorithm isn’t going to change

Thank You



Courtesy – AlexNet, NIPS 2012