DNN Model and Hardware Co-Design

ISCA Tutorial (2019)

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

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

Approaches

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

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

Taxonomy

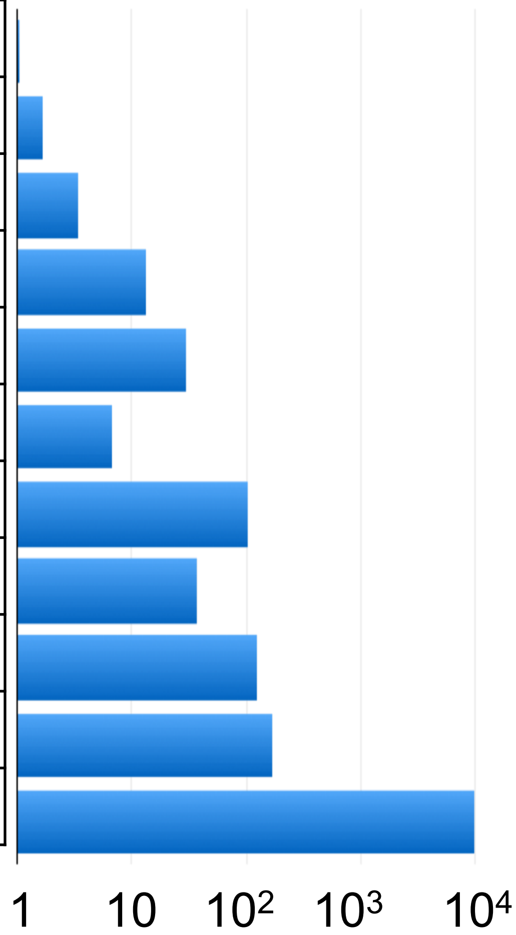
- **Precision** refers to the **number of levels**
  - Number of bits = $\log_2$ (number of levels)
- **Quantization:** mapping data to a smaller set of **levels**
  - Linear, e.g., fixed-point
  - Non-linear
    - Computed (e.g., floating point, log-domain)
    - Table lookup (e.g., learned)

Objective: Reduce size to improve speed and/or reduce energy while preserving accuracy

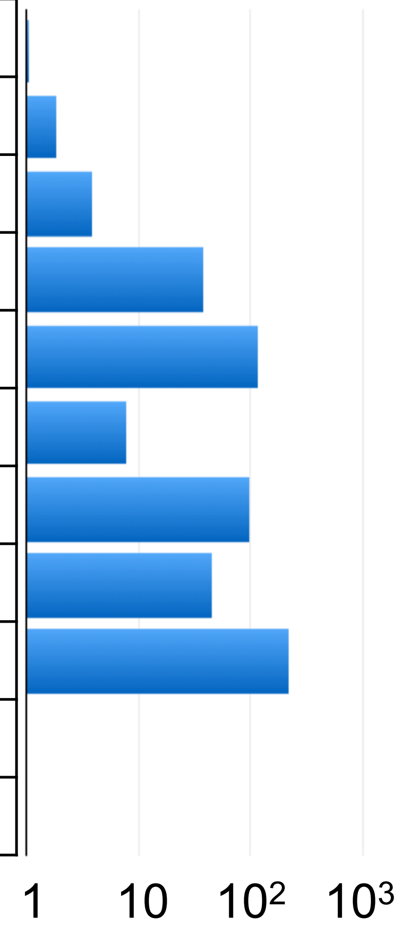
Cost of Operations

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

Relative Energy Cost

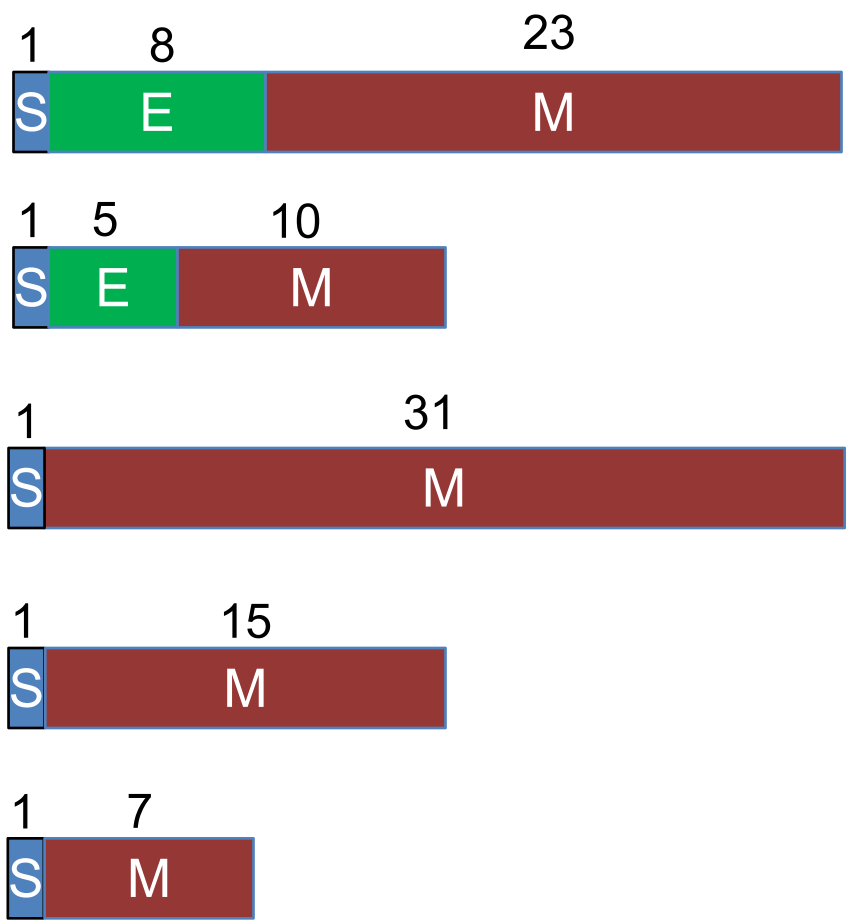


Relative Area Cost



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

Number Representation



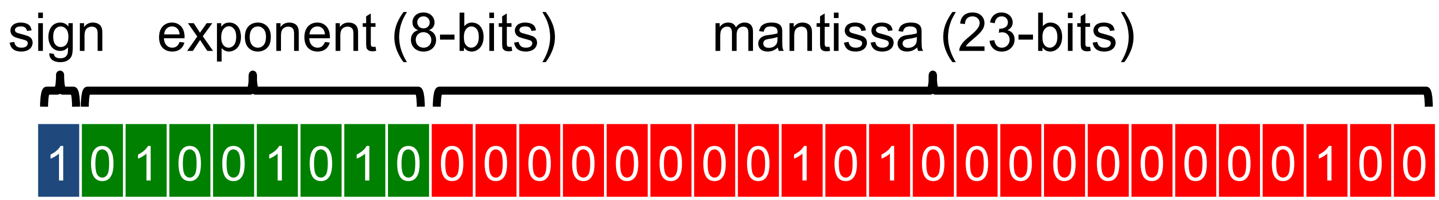
	Range	Accuracy
FP32	$10^{-38} – 10^{38}$	.000006%
FP16	$6x10^{-5}$ - $6x10^{4}$	.05%
Int32	$0 – 2x10^{9}$	½
Int16	$0 – 6x10^{4}$	½
Int8	0 – 127	½

Image Source: B. Dally

Floating Point → Fixed Point

Floating Point

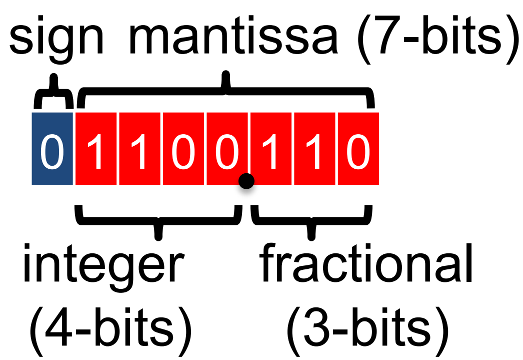
32-bit float



-1.112934 x 10^{-16} ***s*** = 1 ***e*** = 74 ***m*** = 20484

Fixed Point

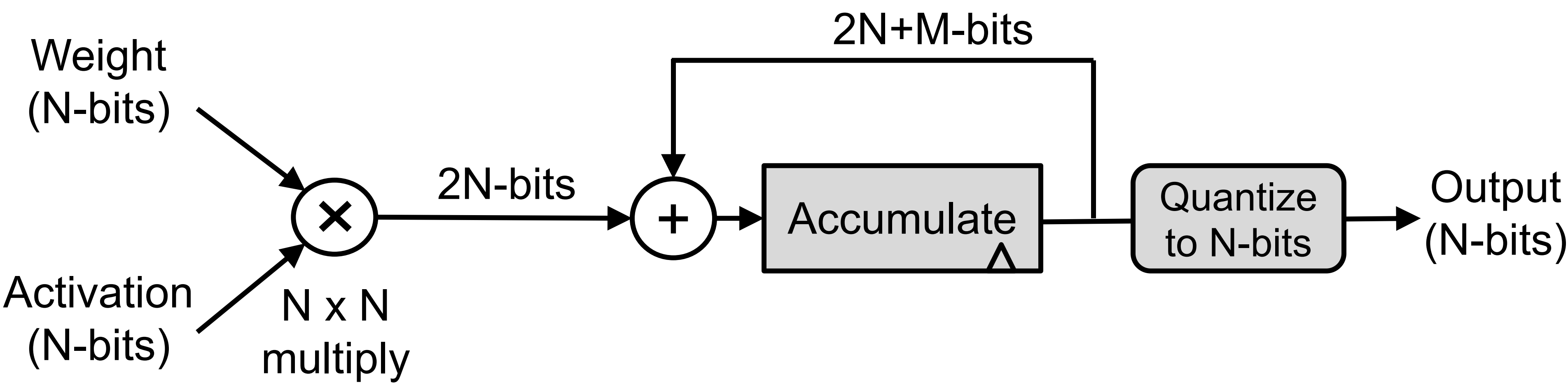
8-bit fixed



12.75 ***s*** = 0 ***m****=102*

N-bit Precision

For no loss in precision, **M** is determined based on largest filter size (in the range of 10 to 16 bits for popular DNNs)



Dynamic Fixed Point

Floating Point

sign | exponent (8-bits) | mantissa (23-bits)

32-bit float: 1 | 0 1 0 0 1 0 1 0 | 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0

-1.112934 x 10^{-16} — ***s*** = 1 ***e*** = 74 ***m*** = 20484

Fixed Point

8-bit dynamic fixed

sign | mantissa (7-bits)

0 | 1 1 0 0 . 1 1 0

integer ([7-***f***]-bits) | fractional (***f***-bits)

12.75 — ***s*** = 0 ***m***=*102* ***f*** = 3

8-bit dynamic fixed

sign | mantissa (7-bits)

0 . [ ] [ ] 1 1 0 0 1 1 0

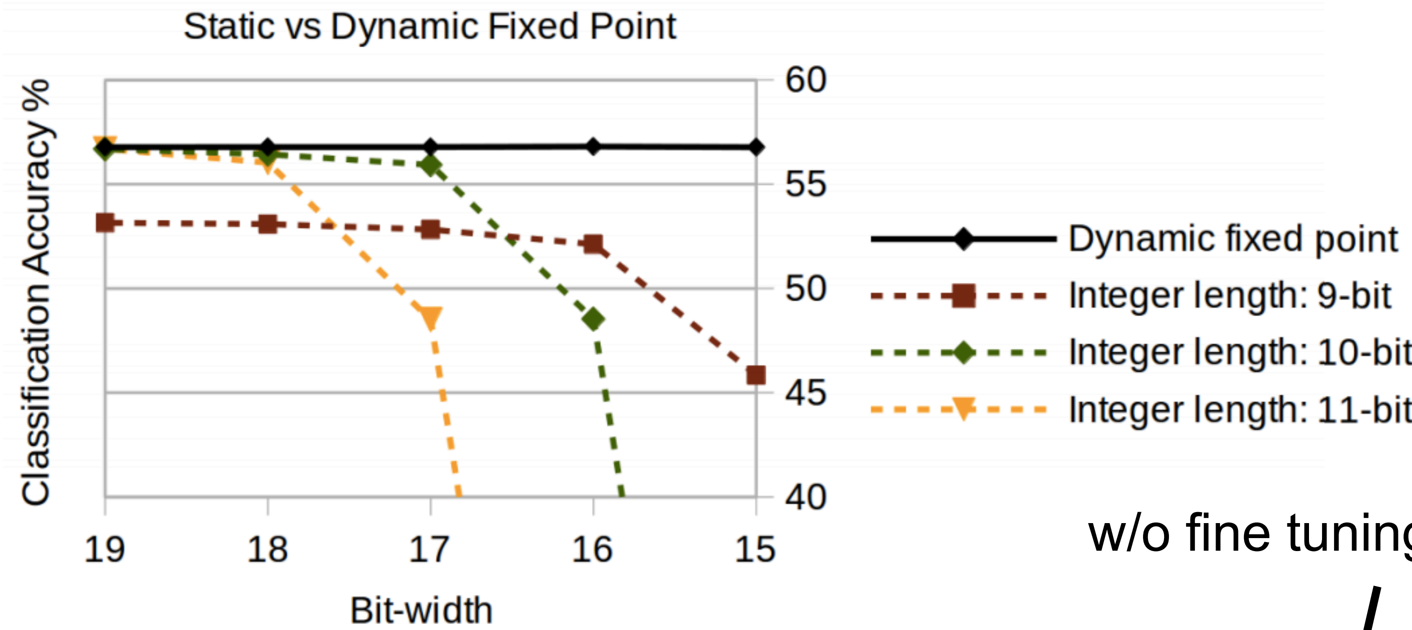
fractional (***f***-bits)

0.19921875 — ***s*** = 0 ***m***=*102* ***f*** = 9

Allow ***f*** to vary based on data type and layer

Impact on Accuracy

Top-1 accuracy on of CaffeNet on ImageNet



w/o fine tuning

	Layer outputs	CONV parameters	FC parameters	32-bit floating point baseline	Fixed point accuracy
LeNet (Exp 1)	4-bit	4-bit	4-bit	99.1%	99.0% (98.7%)
LeNet (Exp 2)	4-bit	2-bit	2-bit	99.1%	98.8% (98.0%)
Full CIFAR-10	8-bit	8-bit	8-bit	81.7%	81.4% (80.6%)
SqueezeNet top-1	8-bit	8-bit	8-bit	57.7%	57.1% (55.2%)
CaffeNet top-1	8-bit	8-bit	8-bit	56.9%	56.0% (55.8%)
GoogLeNet top-1	8-bit	8-bit	8-bit	68.9%	66.6% (66.1%)

[Gysel et al., Ristretto, ICLR 2016]

Avoiding Dynamic Fixed Point

Batch normalization 'centers' dynamic range

AlexNet
(Layer 6)

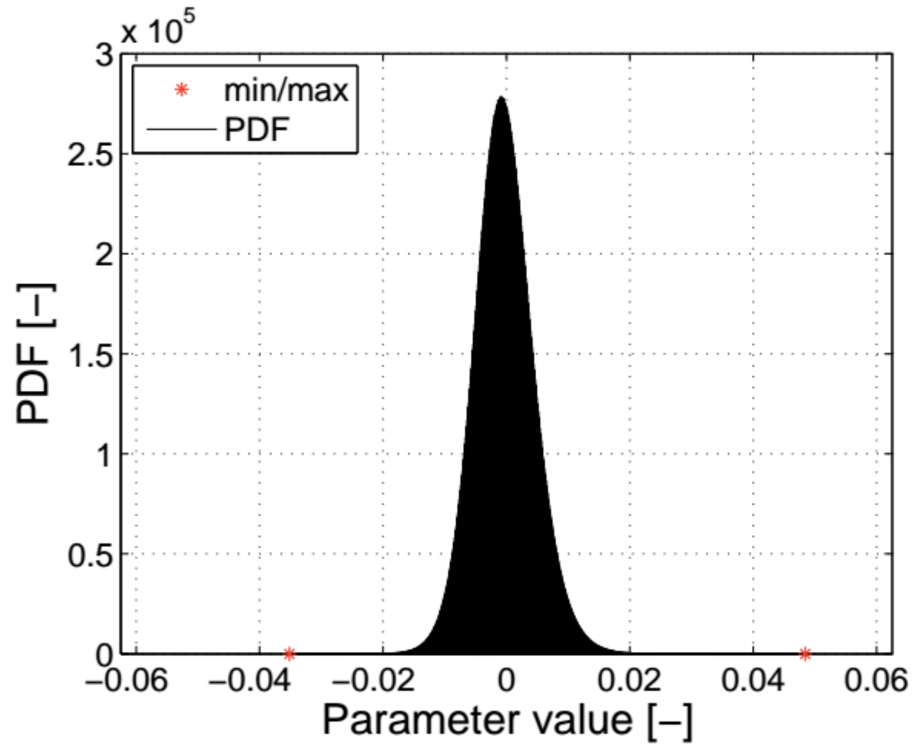


Image Source: Moons et al, WACV 2016

'Centered' dynamic ranges might reduce need for dynamic fixed point

Nvidia PASCAL

“New half-precision, **16-bit floating point instructions deliver over 21 TeraFLOPS** for unprecedented training performance. **With 47 TOPS (tera-operations per second) of performance, new 8-bit integer instructions** in Pascal allow AI algorithms to deliver real-time responsiveness for deep learning inference.”

– Nvidia.com (April 2016)

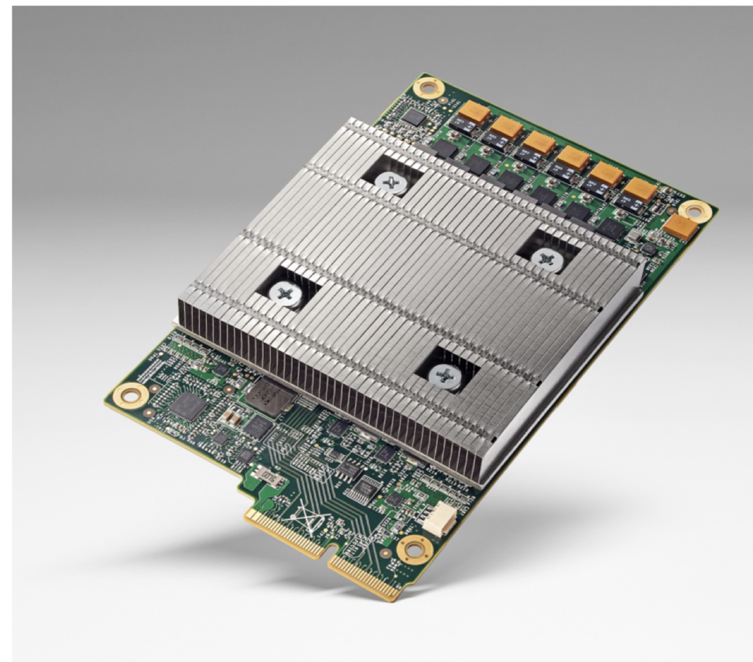
Google’s Tensor Processing Unit (TPU)

“ With its TPU Google has seemingly focused on delivering the data really quickly by **cutting down on precision**. Specifically, it doesn’t rely **on floating point precision like a GPU**

….

Instead the chip uses integer math…TPU used **8-bit integer**.”

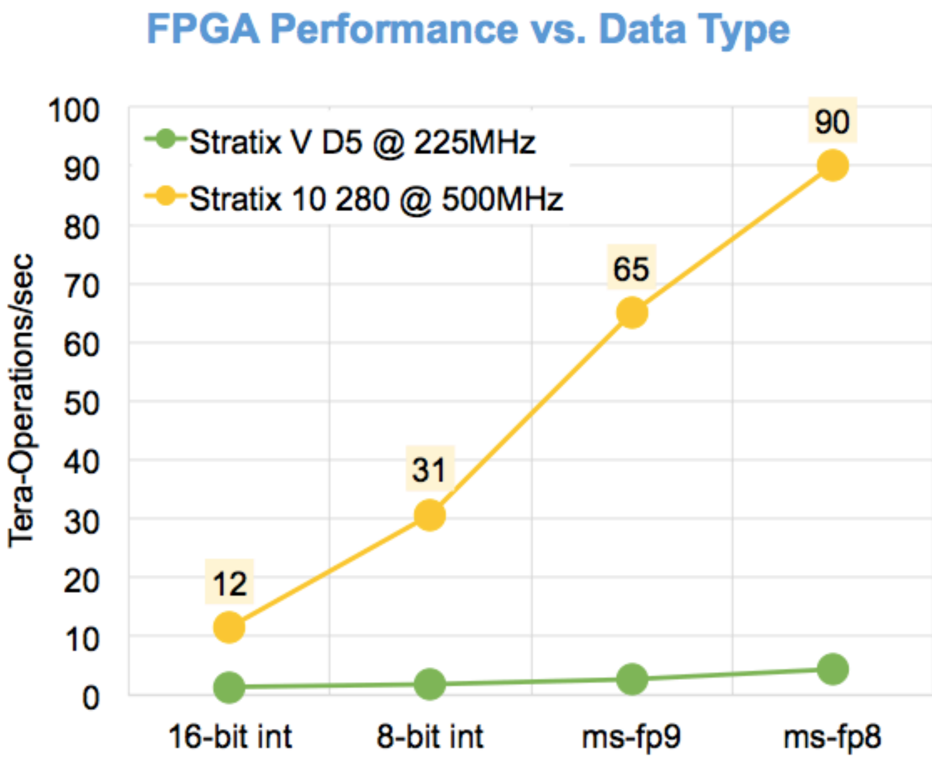
- Next Platform (May 19, 2016)

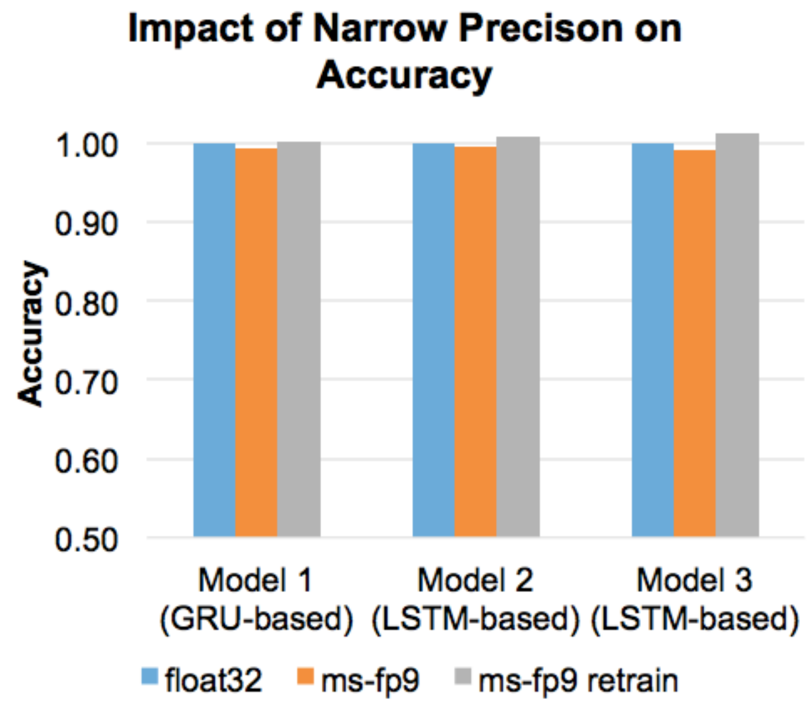


[Jouppi et al., ISCA 2017]

Microsoft BrainWave

Narrow Precision for Inference



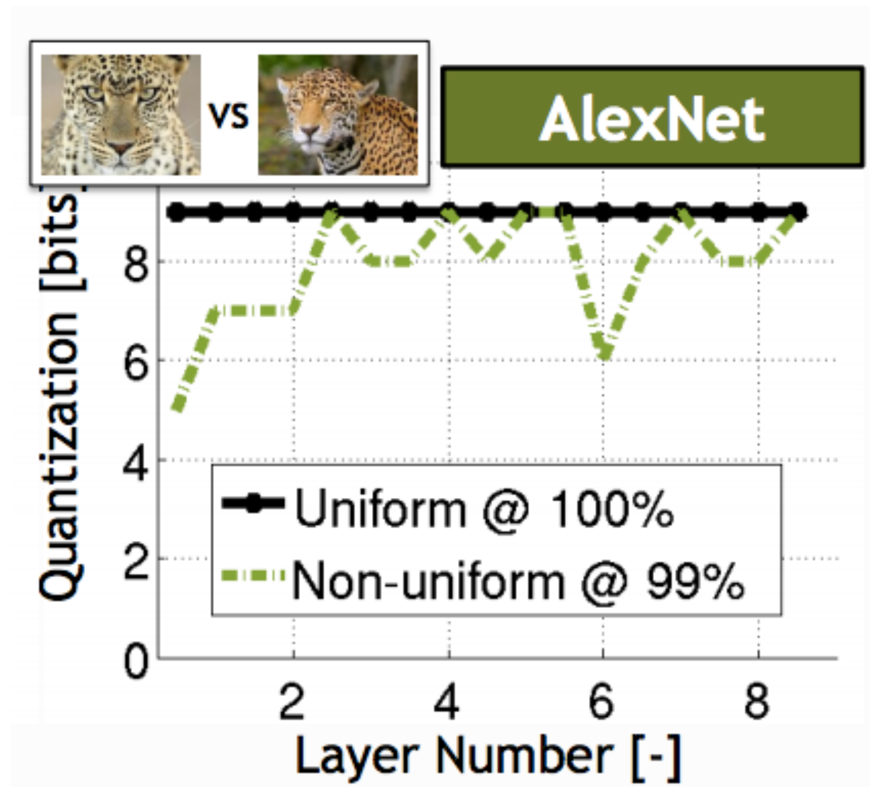


Custom 8-bit floating point format ("ms-fp8")

[Chung et al., Hot Chips 2017]

Precision Varies from Layer to Layer

Tolerance	Bits per layer (I+F)
AlexNet (F=0)	
1%	10-8-8-8-8-8-6-4
2%	10-8-8-8-8-8-5-4
5%	10-8-8-8-7-7-5-3
10%	9-8-8-8-7-7-5-3



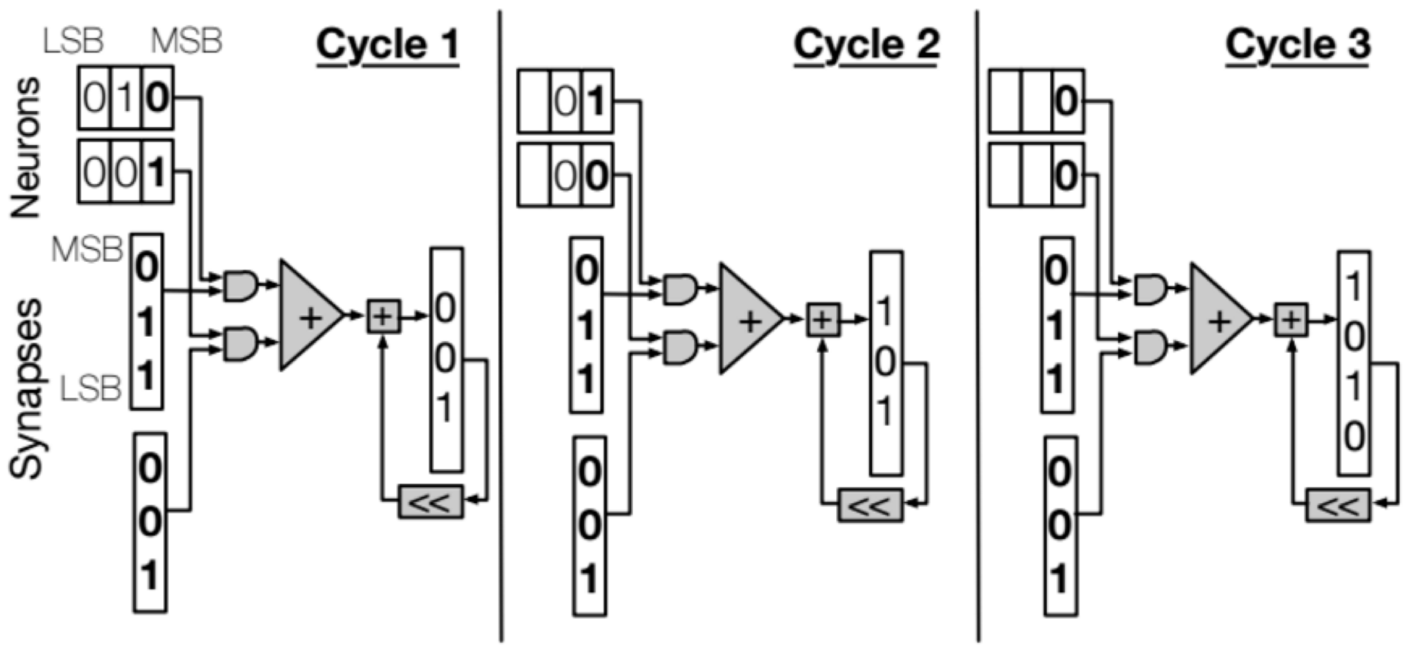
[Judd et al., ArXiv 2016]

[Moons et al., WACV 2016]

Bitwidth Scaling (Speed)

Bit-Serial Processing: Reduce Bit-width → Skip Cycles
Speed up of 2.24x vs. 16-bit fixed

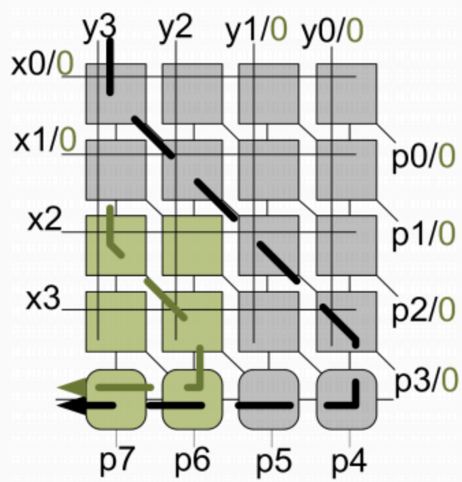
$$\sum_{i=0}^{N_i-1} s_i \times n_i = \sum_{i=0}^{N_i-1} s_i \times \sum_{b=0}^{P-1} n_i^b \times 2^b = \sum_{b=0}^{P-1} 2^b \times \sum_{i=0}^{N_i-1} n_i^b \times S_i$$

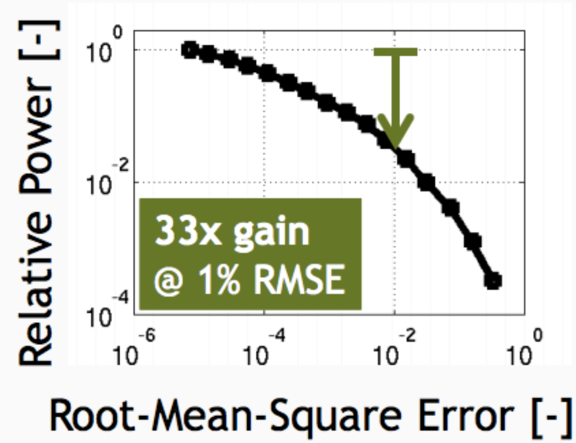


[Judd et al., Stripes, CAL 2016]

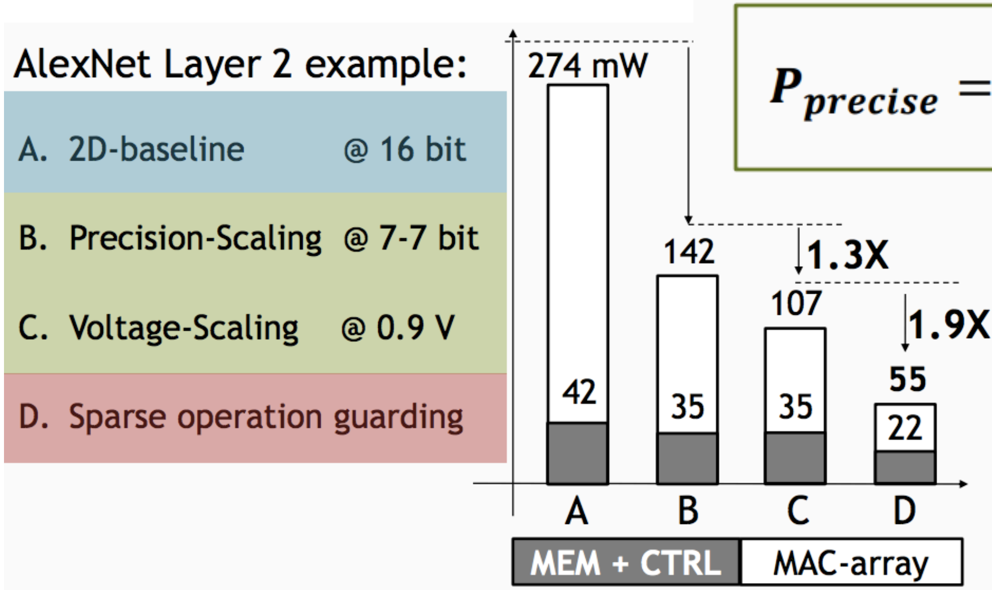
Bitwidth Scaling (Power)

Reduce Bit-width → Shorter Critical Path → Reduce Voltage





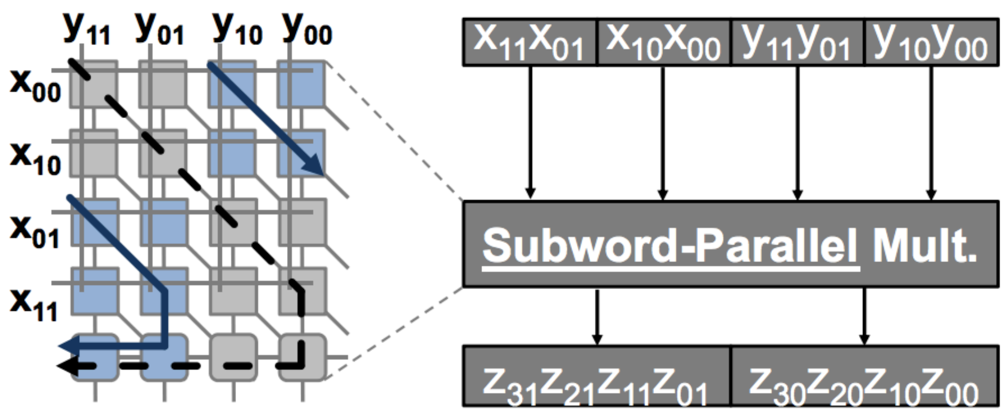
$$P_{precise} = \alpha C f V^2 \Rightarrow P_{imprecise} = \frac{\alpha}{k_1} C f \left(\frac{V}{k_2}\right)^2$$

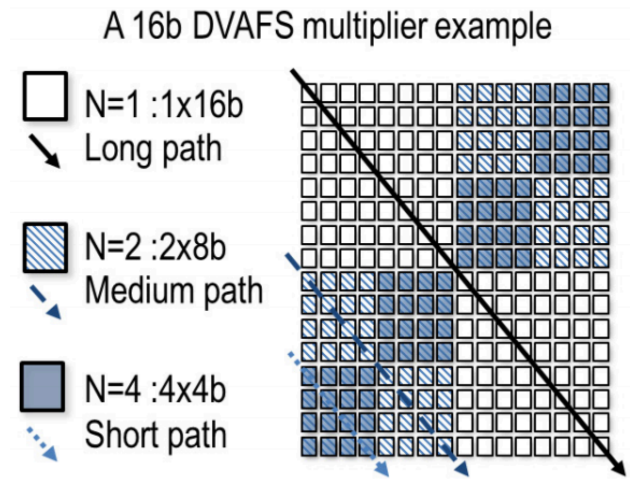


Power reduction of 2.56x vs. 16-bit fixed On AlexNet Layer 2

[Moons et al., VLSI 2016]

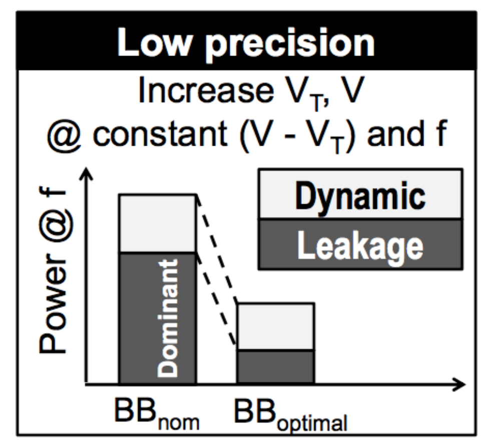
Reconfigure Spatial Multiply





$$\underset{\text{constant throughput}}{E_{precise} \sim \alpha C f V^2} \Rightarrow E_{imprecise} \sim \frac{\alpha}{\boldsymbol{k_3}} C \frac{f}{\boldsymbol{N}} \left(\frac{V}{\boldsymbol{k_4}}\right)^2$$

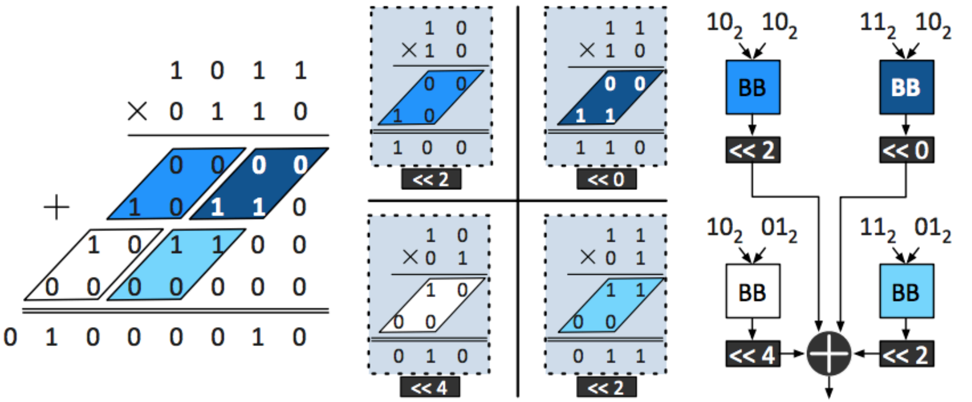
Configure 16bx16b multiplication into two 8x8b or four 4x4b (up to 256-64=192 adders are idle).
Body bias to reduce leakage at low precision since more adders are idle (1.2x reduction)



[Envision, ISSCC 2017]

Reconfigure Spatial Multiply

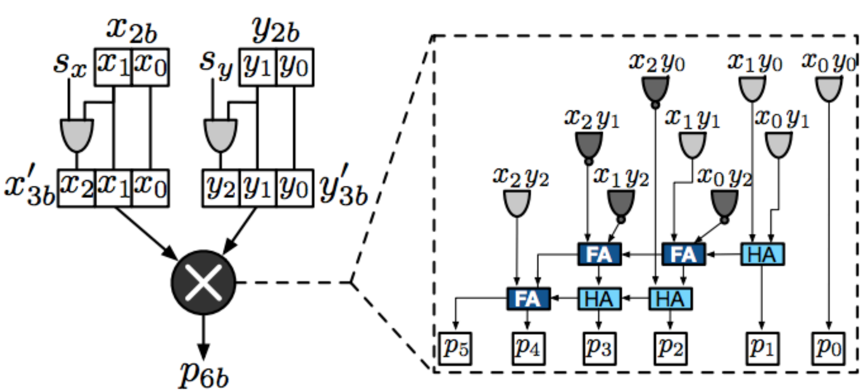
Build larger multipliers (Fused Unit) from small 2x2 multipliers with programmable shifters (BitBrick)

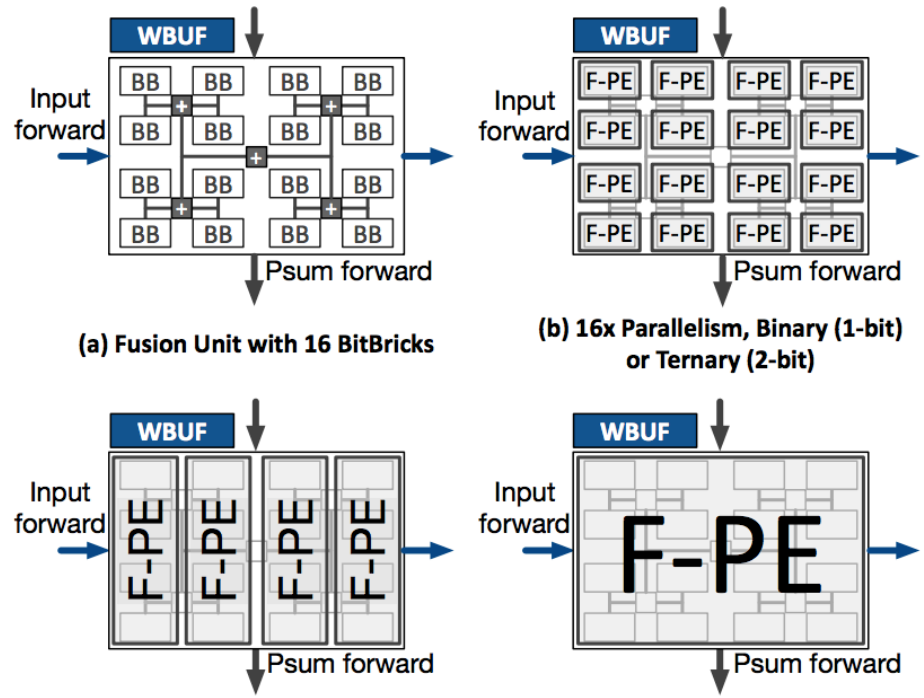


(a) A 4-bit multiplication (6_{10} x 11_{10} = 66_{10})

(b) Decomposing the 4-bit multiplication to four 2-bit multiplications.

(c) Mapping decomposed multiplications to BitBricks (BBs).





(a) Fusion Unit with 16 BitBricks

(b) 16x Parallelism, Binary (1-bit) or Ternary (2-bit)

(c) 4x Parallelism, Mixed-Bitwidth (2-bit weights, 8-bit inputs)

(d) No Parallelism, 8-bits

One 8bx8b, four 2bx8b, sixteen 2bx2b

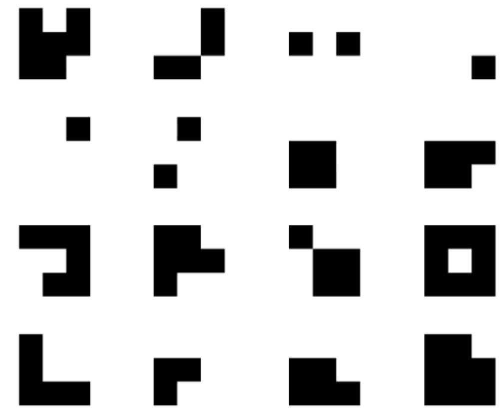
[Bit Fusion, ISCA 2018]

Binary Nets

- **Binary Connect (BC)**
  - Weights {-1,1}, Activations 32-bit float
  - MAC → addition/subtraction
  - Accuracy loss: **19%** on AlexNet

    [Courbariaux, NeurIPS 2015]

Binary Filters



- **Binarized Neural Networks (BNN)**
  - Weights {-1,1}, Activations {-1,1}
  - MAC → XNOR
  - Accuracy loss: **29.8%** on AlexNet

    [Courbariaux, arXiv 2016]

Scale the Weights and Activations

- **Binary Weight Nets (BWN)**
  - Weights {-α, α} → except first and last layers are 32-bit float
  - Activations: 32-bit float
  - α determined by the l_1-norm of all weights in a filter
  - Accuracy loss: **0.8%** on AlexNet
- **XNOR-Net**
  - Weights {-α, α}
  - Activations {-β_i, β_i} → except first and last layers are 32-bit float
  - β_i determined by the l_1-norm of all activations across channels ***for given position i*** of the input feature map
  - Accuracy loss: **11%** on AlexNet

Hardware needs to support both activation precisions

Scale factors (α, β_i) can change per filter or position in filter

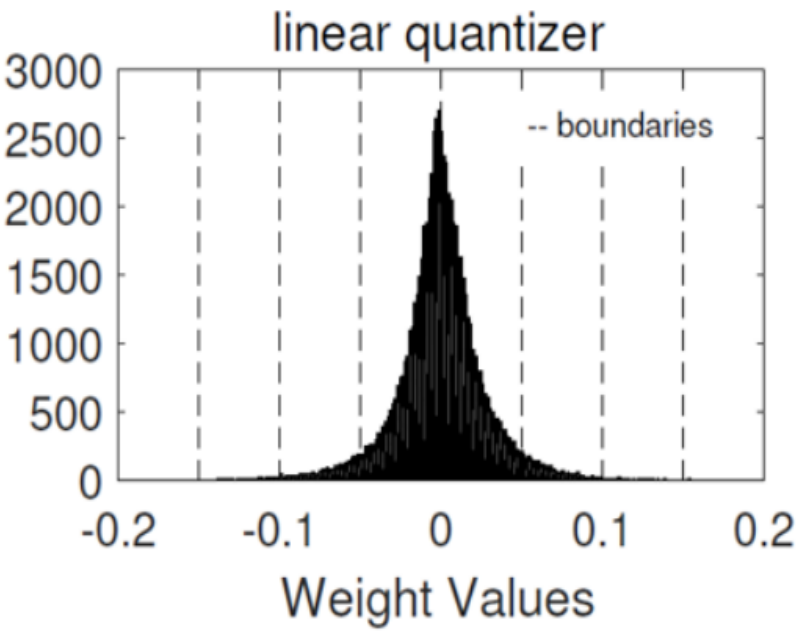
[Rastegari et al., BWN & XNOR-Net, ECCV 2016]

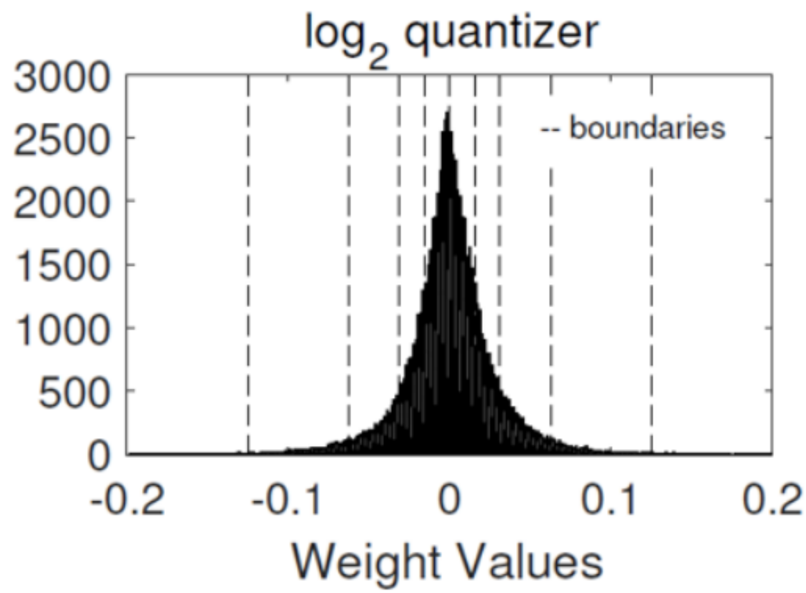
Ternary Nets

- **Allow for weights to be zero**
  - Increase sparsity, but also increase number of bits (2-bits)
- **Ternary Weight Nets (TWN)** [Li et al., arXiv 2016]
  - Weights {-w, 0, w} → except first and last layers are 32-bit float
  - Activations: 32-bit float
  - Accuracy loss: 3.7% on AlexNet
- **Trained Ternary Quantization (TTQ)** [Zhu et al., ICLR 2017]
  - Weights $\{-w_1, 0, w_2\}$ → except first and last layers are 32-bit float
  - Activations: 32-bit float
  - Accuracy loss: 0.6% on AlexNet

Computed Non-linear Quantization

Log Domain Quantization





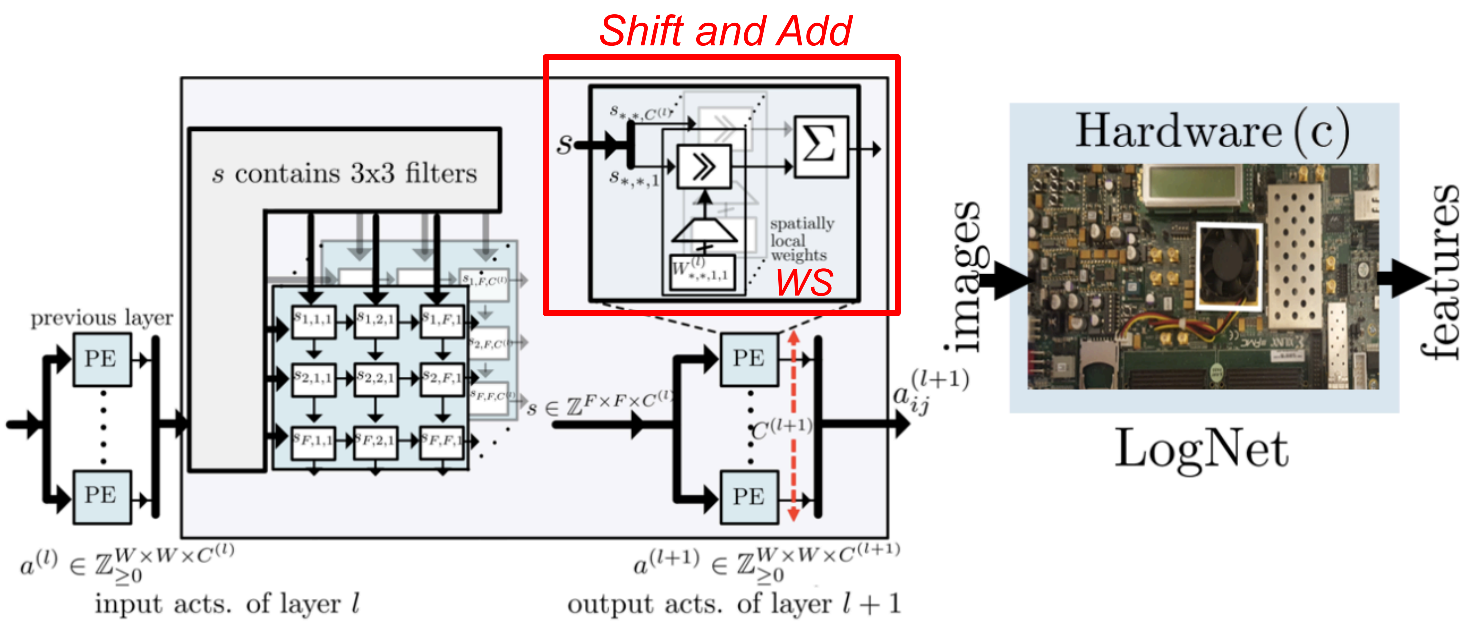
Product = X * W

Product = X << W

[Lee et al., LogNet, ICASSP 2017]

Log Domain Quantization

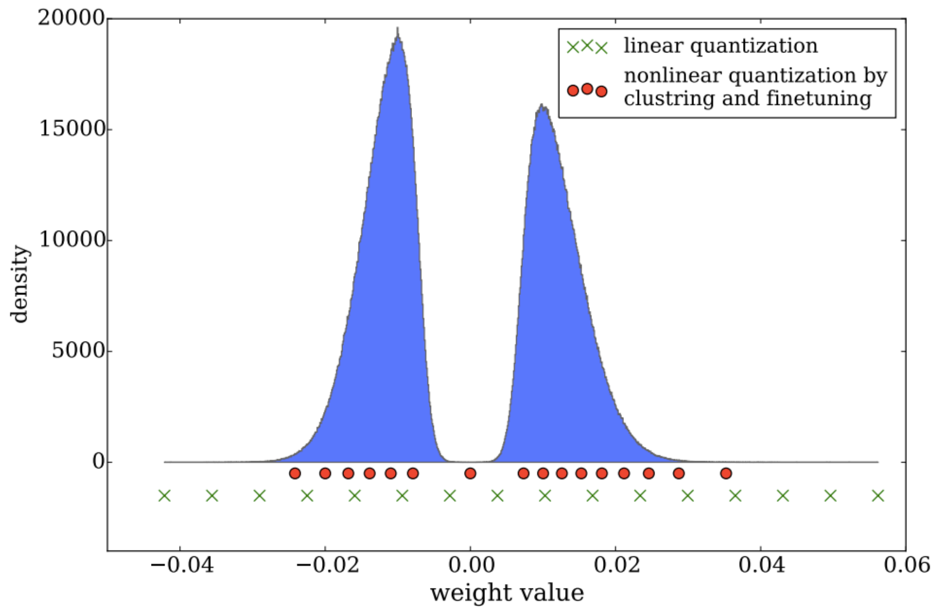
- **Weights: 5-bits for CONV, 4-bit for FC; Activations: 4-bits**
- Accuracy loss: 3.2% on AlexNet



[Miyashita et al., arXiv 2016],
[Lee et al., LogNet, ICASSP 2017]

Reduce Precision Overview

- **Learned mapping of data to quantization levels (e.g., k-means)**



Implement with look up table

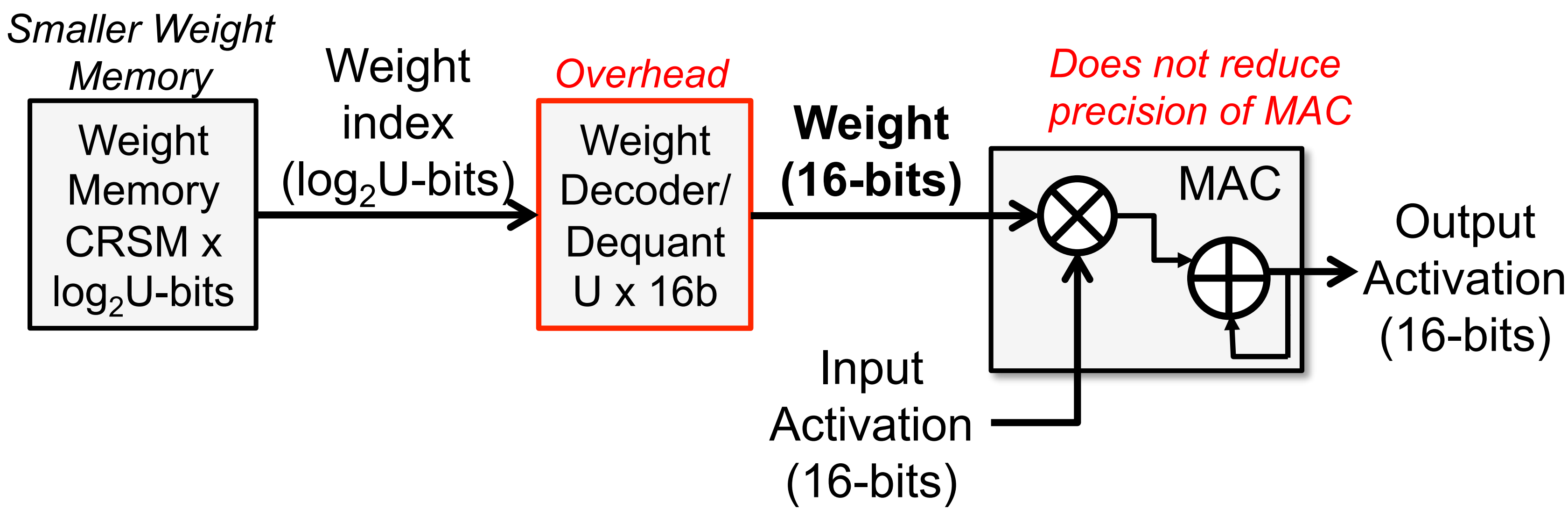
[Han et al., ICLR 2016]

- **Additional Properties**
  - **Fixed or Variable (across data types, layers, channels, etc.)**

Non-Linear Quantization Table Lookup

Trained Quantization: Find K weights via K-means clustering to reduce number of unique weights ***per layer*** (weight sharing)

Example: AlexNet (no accuracy loss)
256 unique weights for CONV layer
16 unique weights for FC layer



Consequences: Narrow weight memory and second access from (small) table

[Han et al., Deep Compression, ICLR 2016]

Summary of Reduce Precision

Category	Method	Weights (# of bits)	Activations (# of bits)	Accuracy Loss vs. 32-bit float (%)
Dynamic Fixed Point	w/o fine-tuning	8	10	0.4
	w/ fine-tuning	8	8	0.6
Reduce weight	Ternary weights Networks (TWN)	2*	32	3.7
	Trained Ternary Quantization (TTQ)	2*	32	0.6
	Binary Connect (BC)	1	32	19.2
	Binary Weight Net (BWN)	1*	32	0.8
Reduce weight and activation	Binarized Neural Net (BNN)	1	1	29.8
	XNOR-Net	1*	1	11
Non-Linear	LogNet	5(conv), 4(fc)	4	3.2
	Weight Sharing	8(conv), 4(fc)	16	0

* first and last layers are 32-bit float