Fully Custom Deep Learning Hardware

Chixiao Chen

Announcement

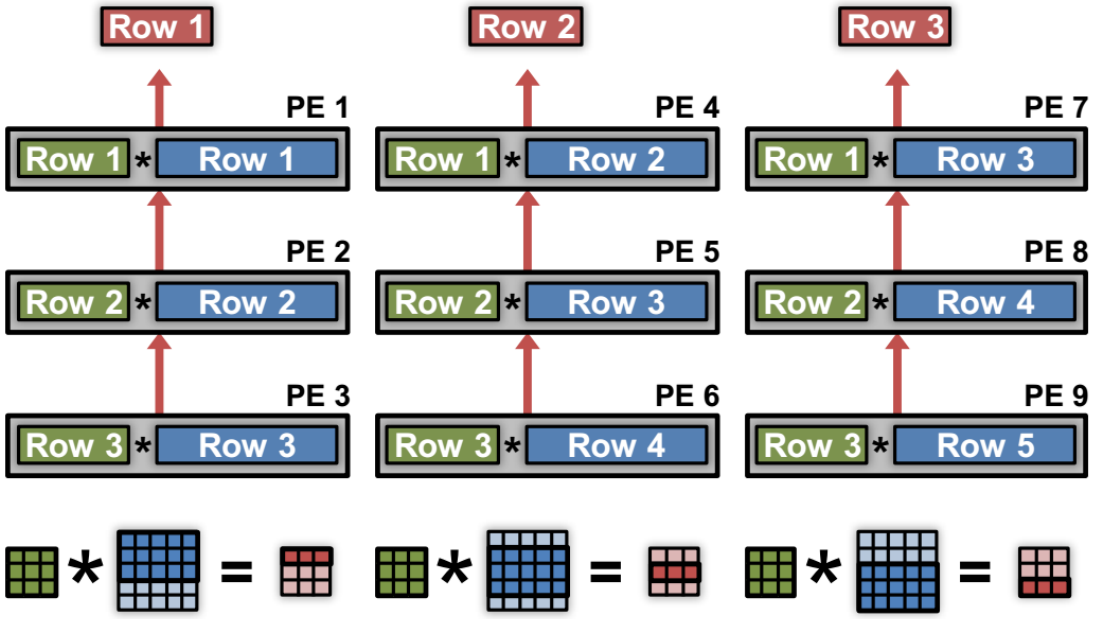
- This is the last class of our courses this semester.

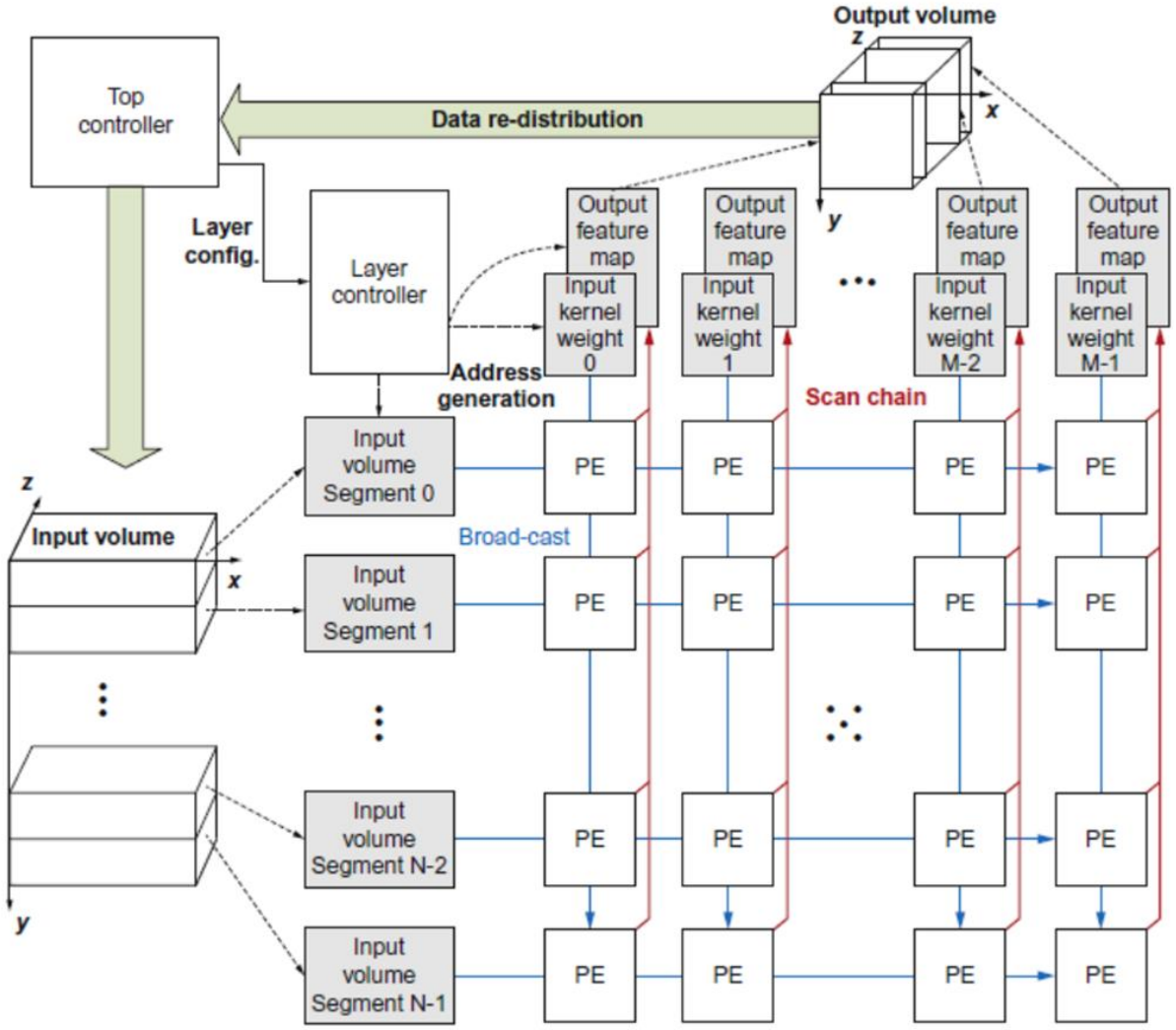
- Our final project will be presented on Jun 20, 9:30am.

  - Location: 新金博大楼 @ Handan Campus
  - 20min for each student, 2 hours for total
  - 15 min for presentation, 5 min for Q&A
  - **High recommend**: do not use more than 25 slides.

Lecture Last time

- PE (CGRA) Based Arch for Deep Learning
- Efficient Data flow to avoid repeated data bandwidth





Overview

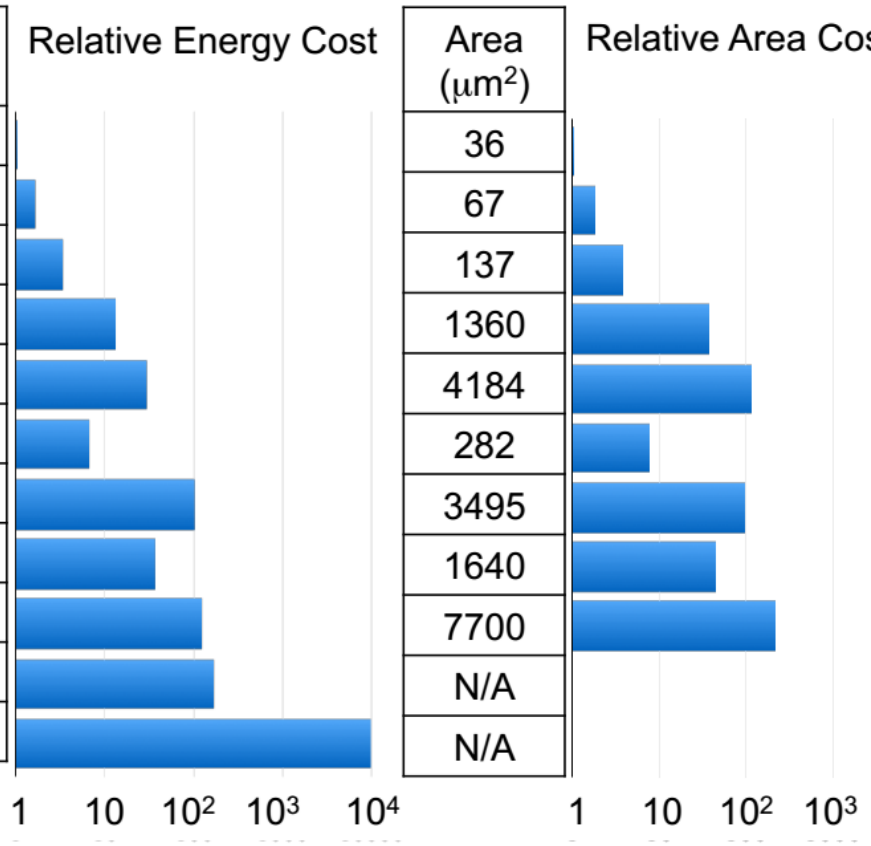
- Algorithm / Software / Hardware Codesign
  - Bit-width reduction
  - Sparsity
- Processing in Memory (PiM, CiM)

Cost of Operation

MAC is the key energy consumer in a deep learning processor, and it is highly related to the bit width

Multiplication follows a square rate law.

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]

$$A \times B = (A_{MSB} << 8 + A_{LSB}) \times (B_{MSB} << 8 + B_{LSB})$$

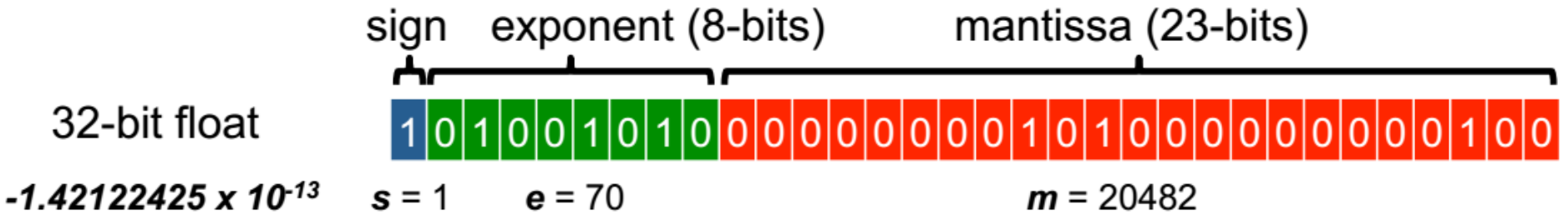
$$= A_{MSB}B_{MSB} << 16 + [A_{MSB}B_{LSB} + A_{LSB}B_{MSB}] << 8 + A_{LSB}B_{LSB},$$

Floating Point → Fixed Point

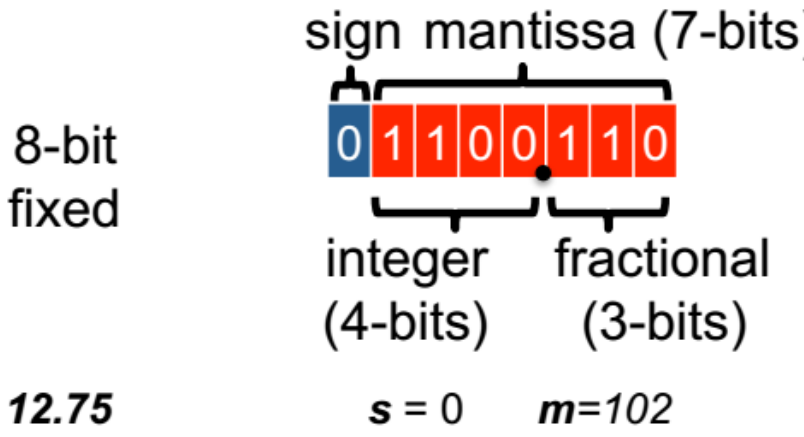
Question:

Human brains never have 32b floating range and accuracy, why DL need to use such presentation?

Floating Point

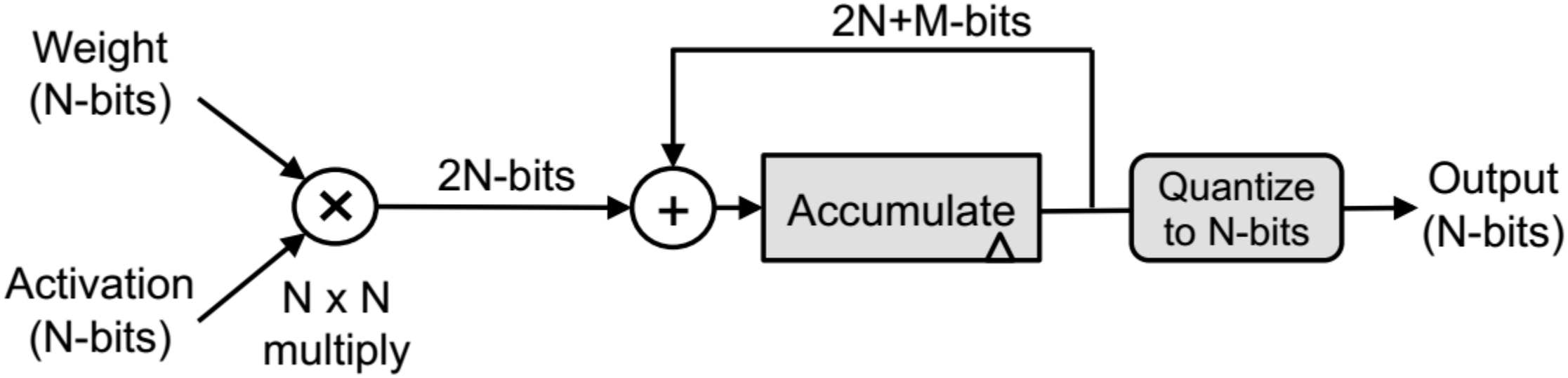


Fixed Point



Ideal Precision

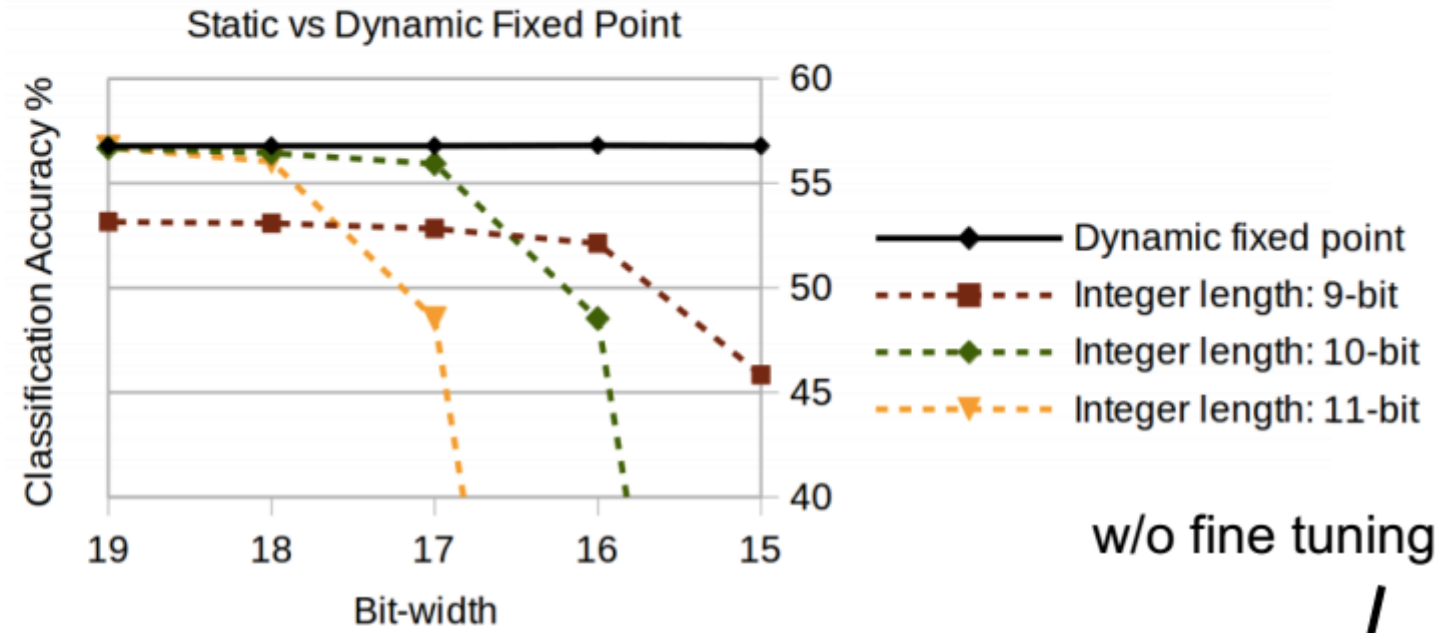
- For no loss in precision, **M** is determined based on largest filter size (in the range of 10 to 16 bits for popular DNNs)



- But we do not really need that, many sum are zeros (ReLU).

Impact on Accuracy

Top-1 accuracy on of CaffeNet on ImageNet



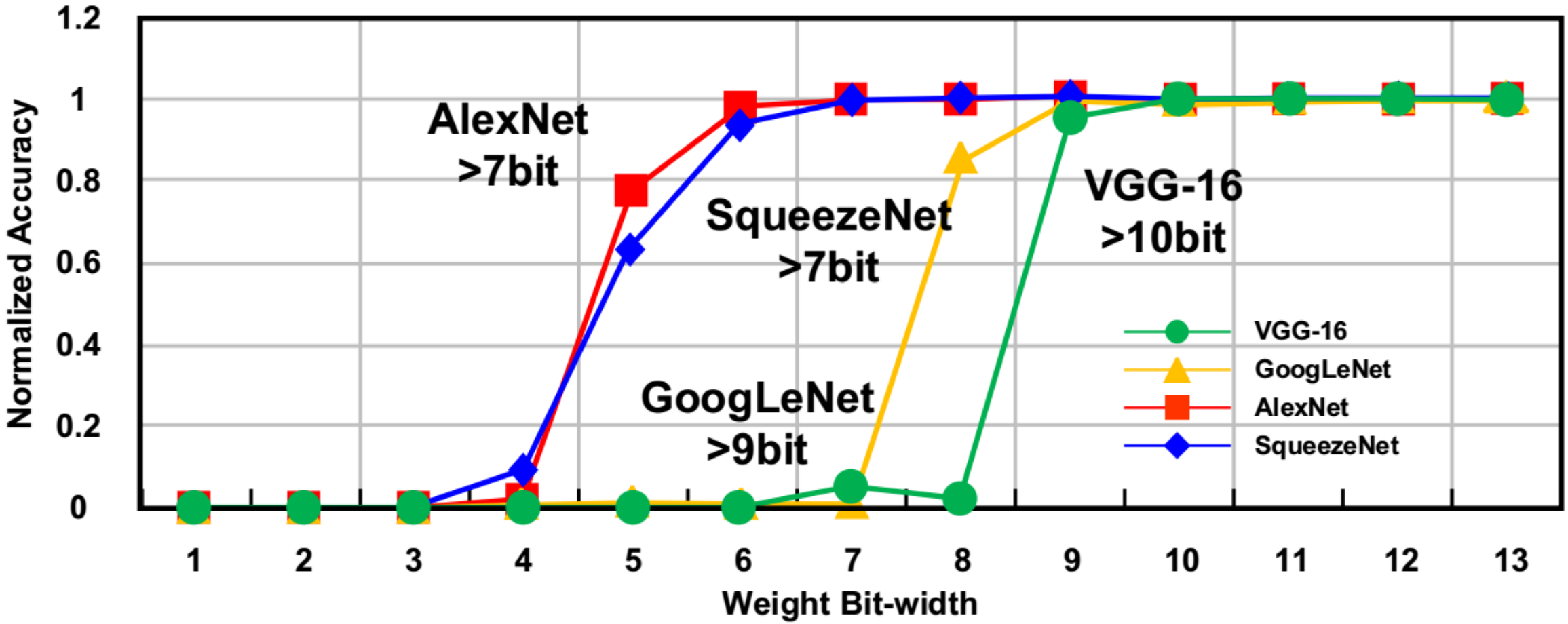
w/o fine tuning

	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]

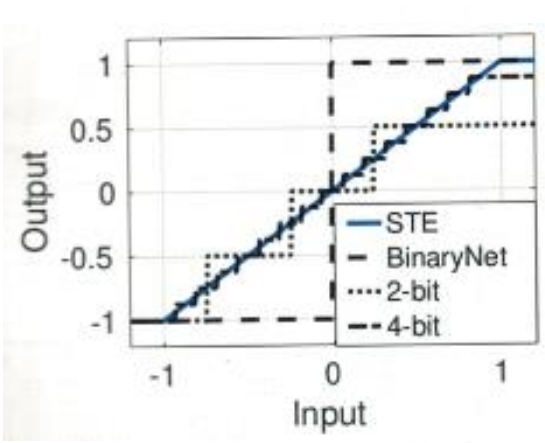
2. Bit Precision Reconfigurability

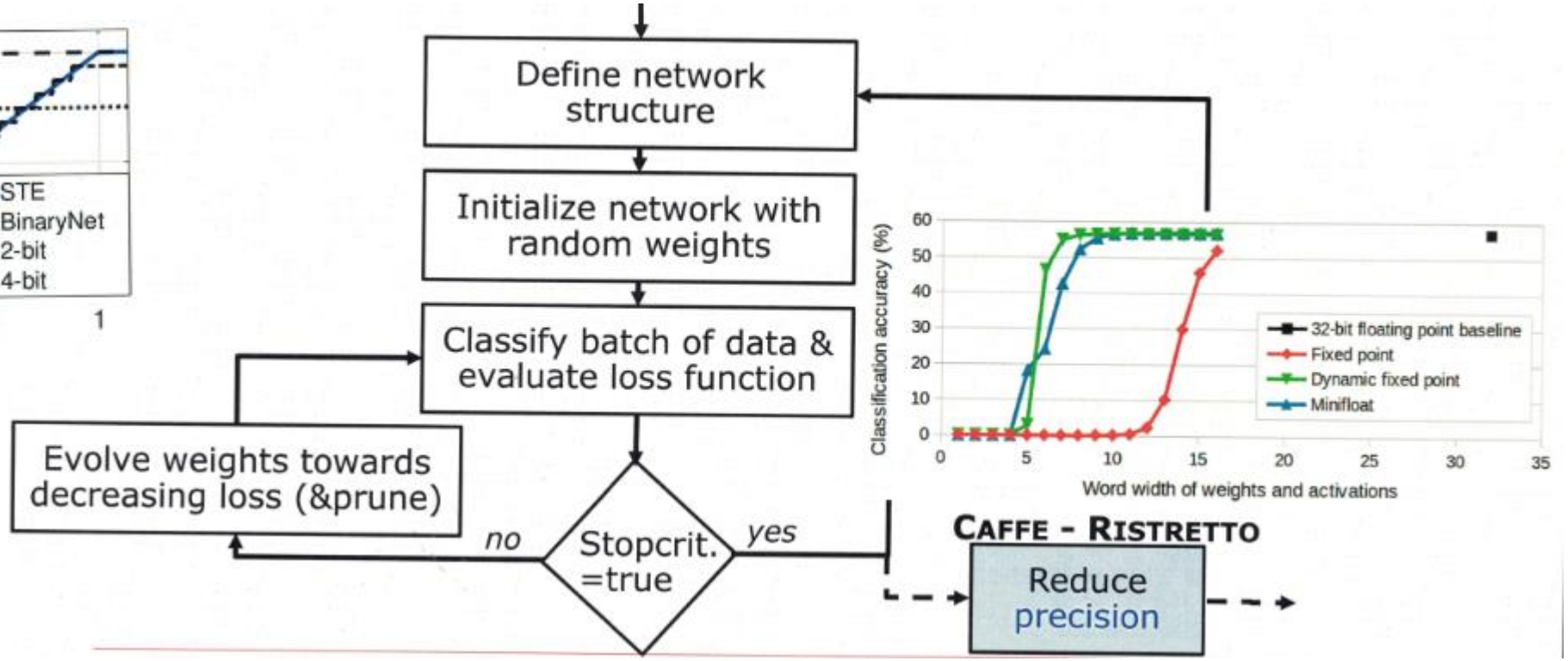
❑ Optimal Weight Precision Depends on Networks



Fine Tuning Methodology

- Training after quantization





Avoiding Dynamic Fixed Point

Batch normalization ‘centers’ dynamic range

AlexNet
(Layer 6)

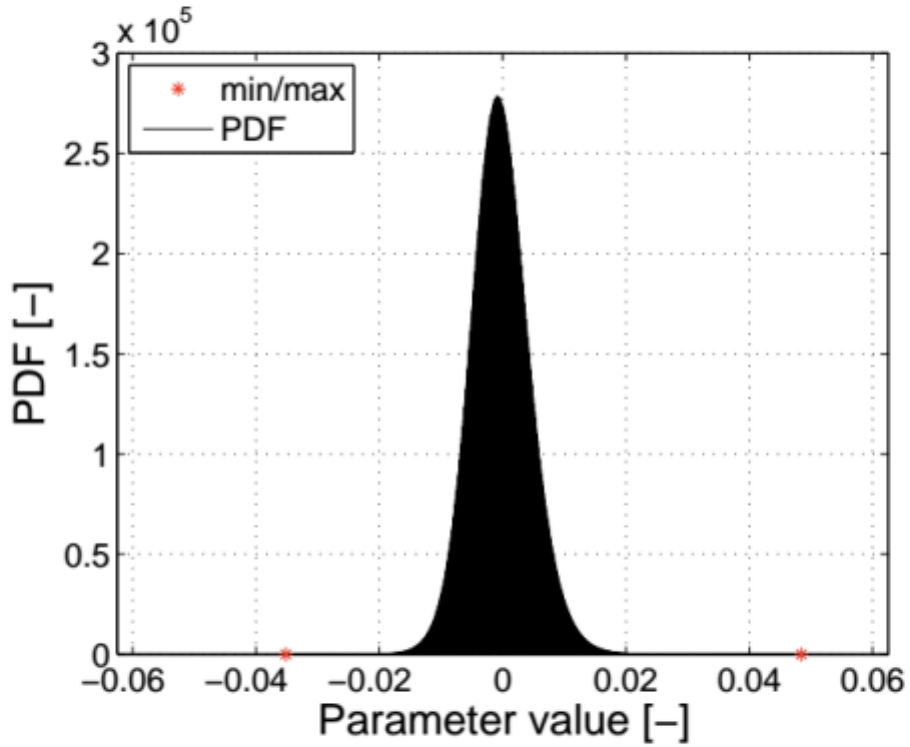
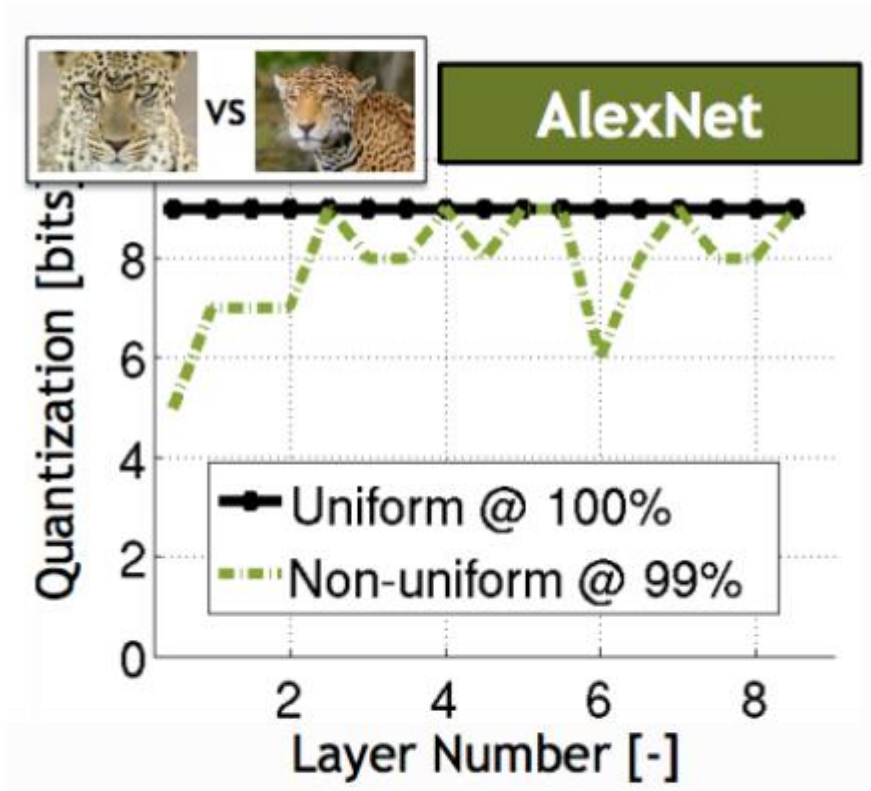


Image Source: Moons et al, WACV 2016

‘Centered’ dynamic ranges might reduce need for dynamic fixed point

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]

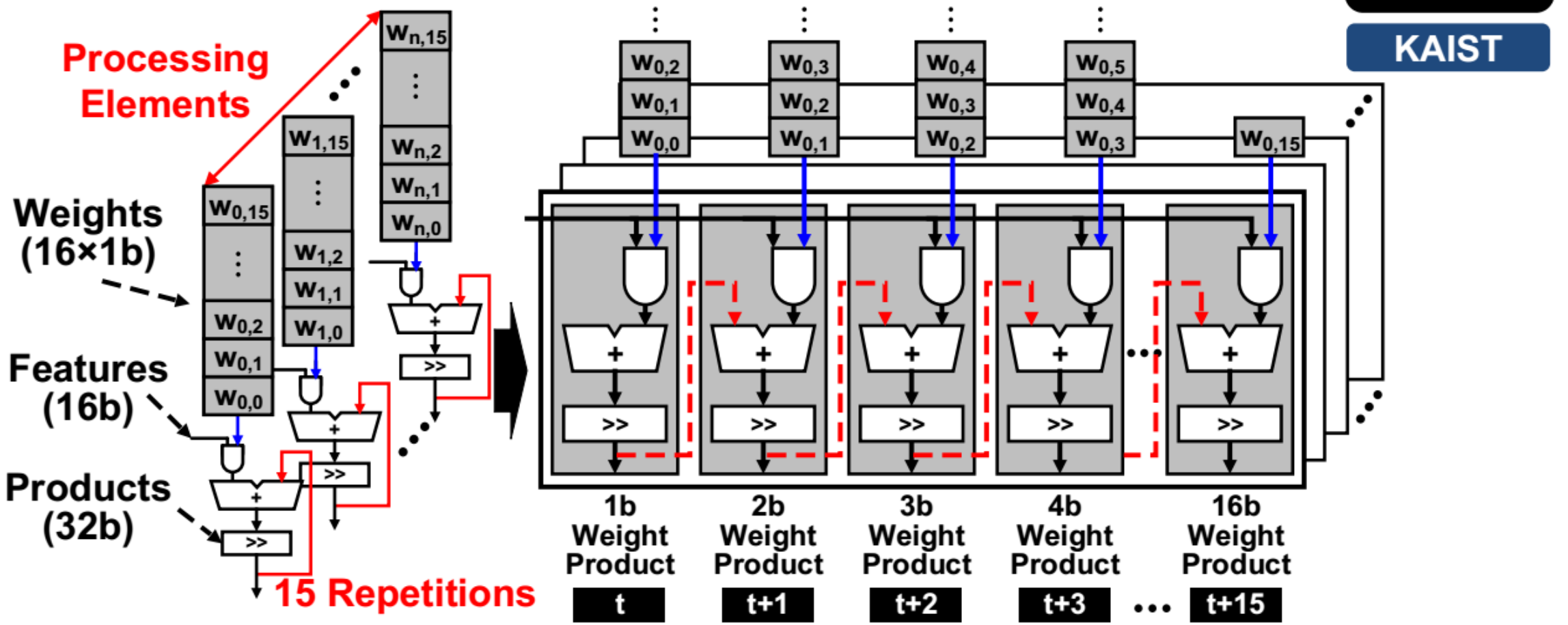
[Moons et al., WACV 2016]

Fully-Variable Weight Precision

❑"UNPU"-Bit-serial Processing – Variable from 1b to 16b

UNPU ISSCC2018

KAIST



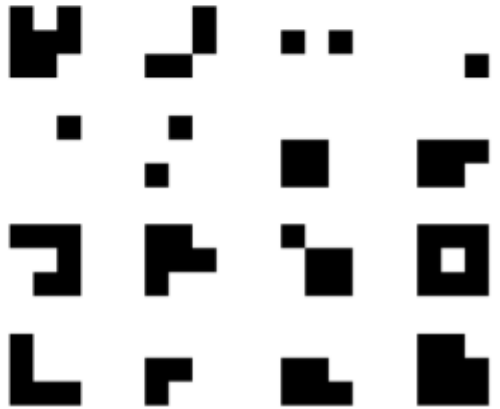
Lee, Jinmook, et al. "UNPU: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision." *ISSCC2018*

Binary Nets

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

    [Courbariaux, NIPS 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

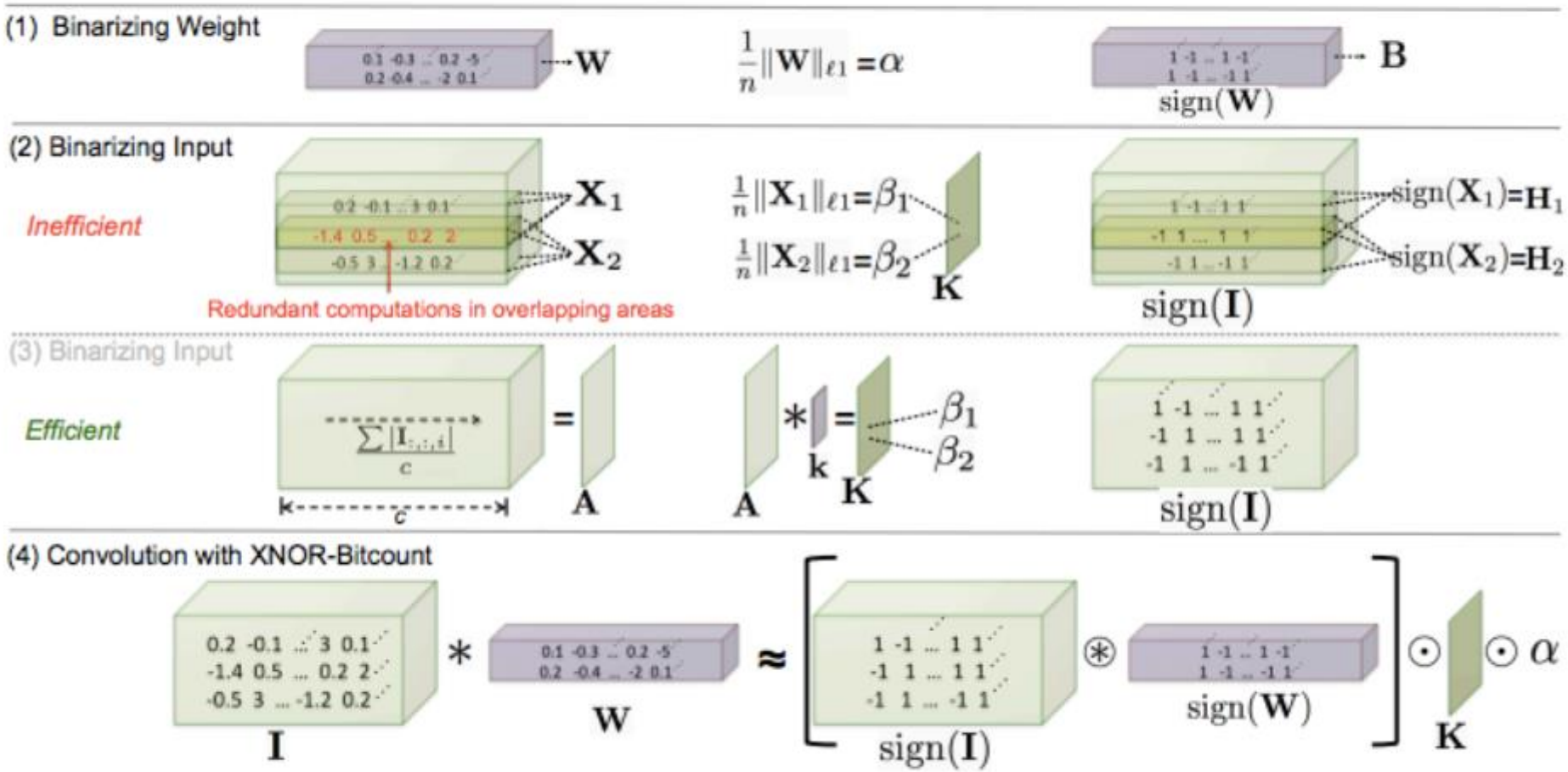
- **Binary Weight Nets (BWN)**
  - Weights {-α, α} → except first and last layers are 32-bit float
  - Activations: 32-bit float
  - α determined by the l_1-norm of all weights in a layer
  - Accuracy loss: **0.8%** on AlexNet
- **XNOR-Net**
  - Weights {-α, α}
  - Activations {-β_i, β_i} → except first 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 layer or position in filter

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

XNOR-Net



[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

Non-Linear Quantization

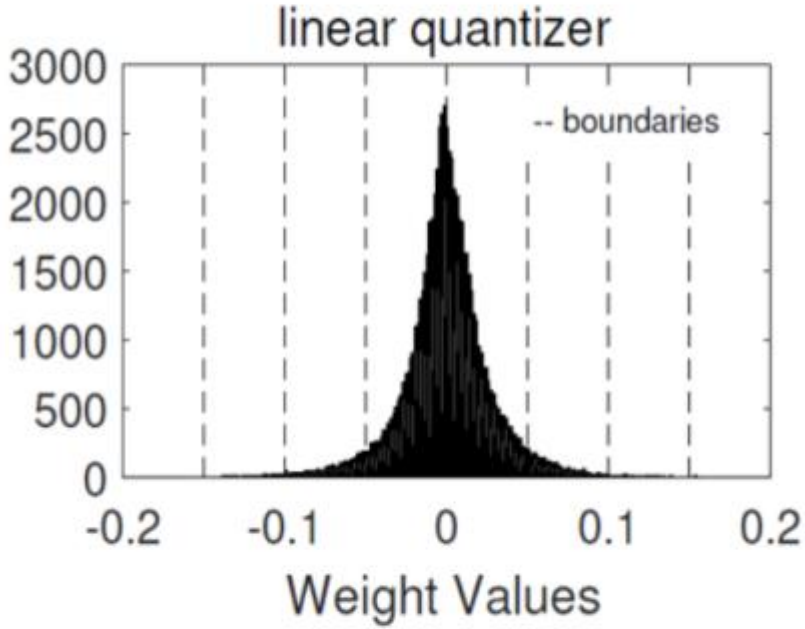
- **Precision** refers to the **number of levels**
  - Number of bits = $\log_2$ (number of levels)

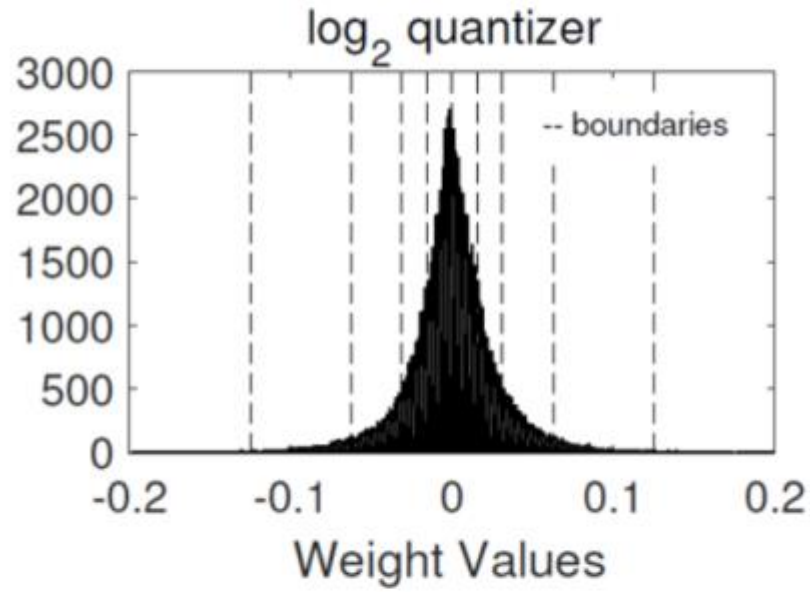
- **Quantization:** mapping data to a smaller set of **levels**
  - Linear, e.g., fixed-point
  - Non-linear
    - Computed
    - Table lookup

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

Computed Non-linear Quantization

Log Domain Quantization





Product = X * W

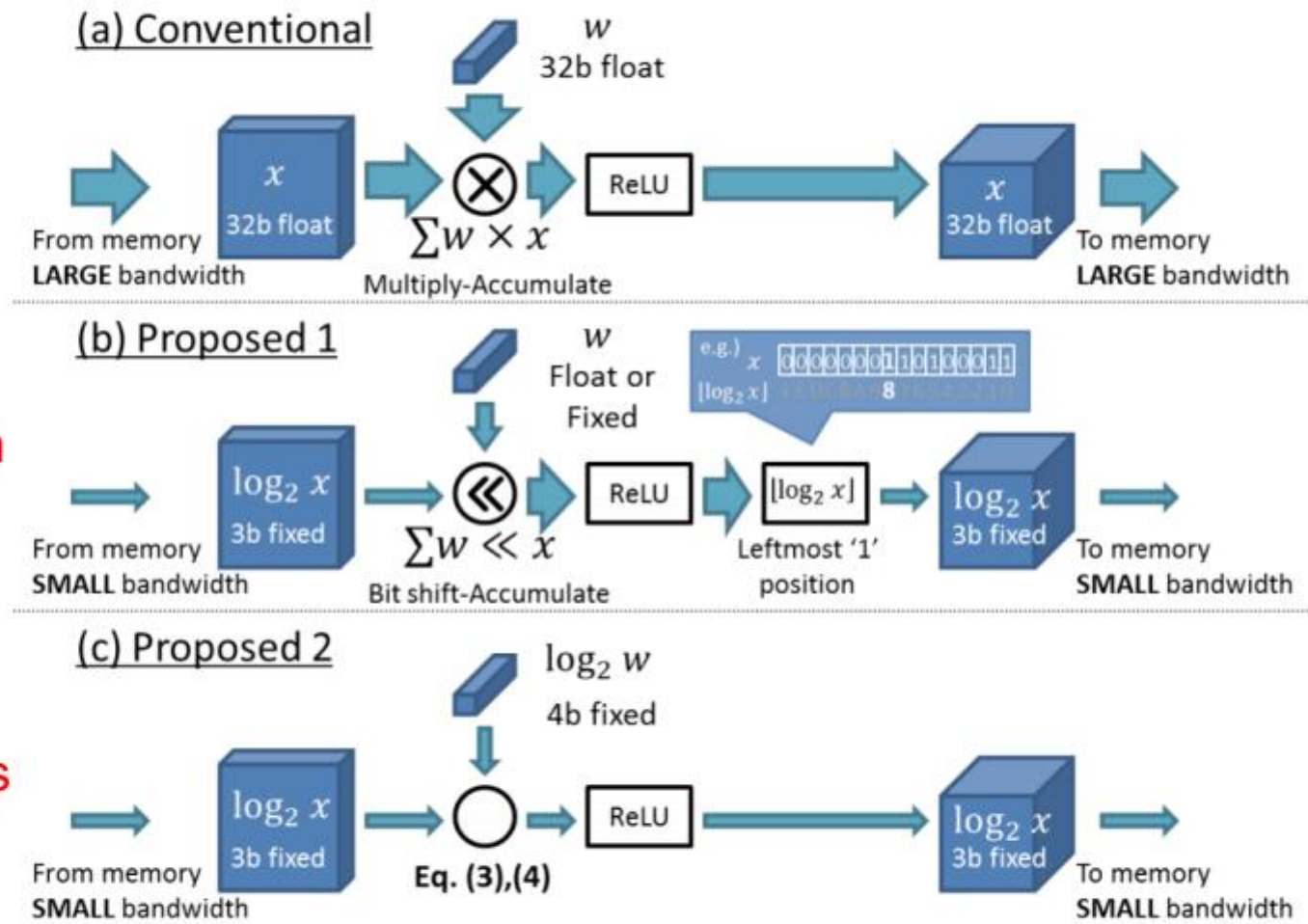
Product = X << W

[Lee et al., LogNet, ICASSP 2017]

Log Domain Computation

Only activation in log domain

Both weights and activations in log domain

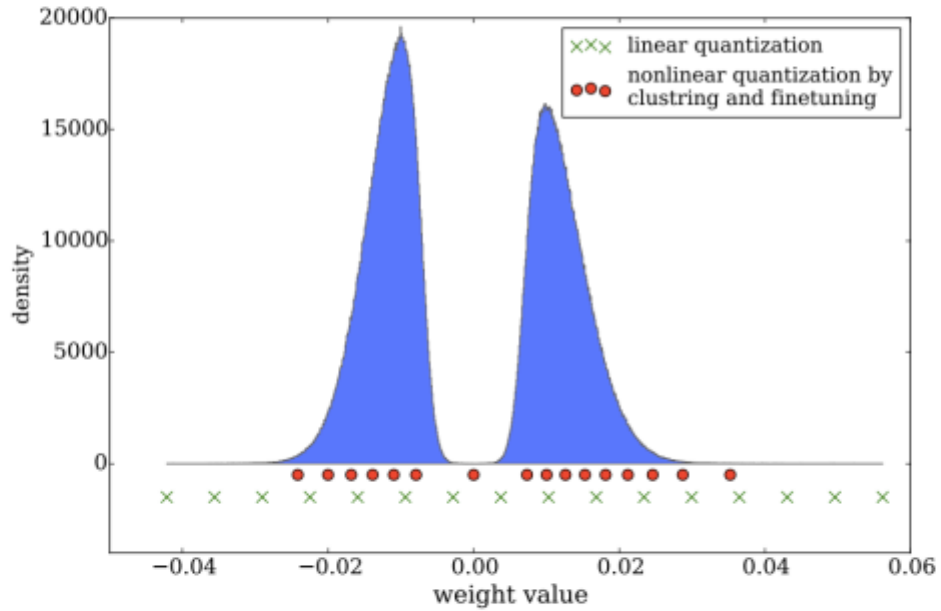


max, bitshifts, adds/subs

[Miyashita et al., arXiv 2016]

Reduce Precision Overview

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



Implement with look up table

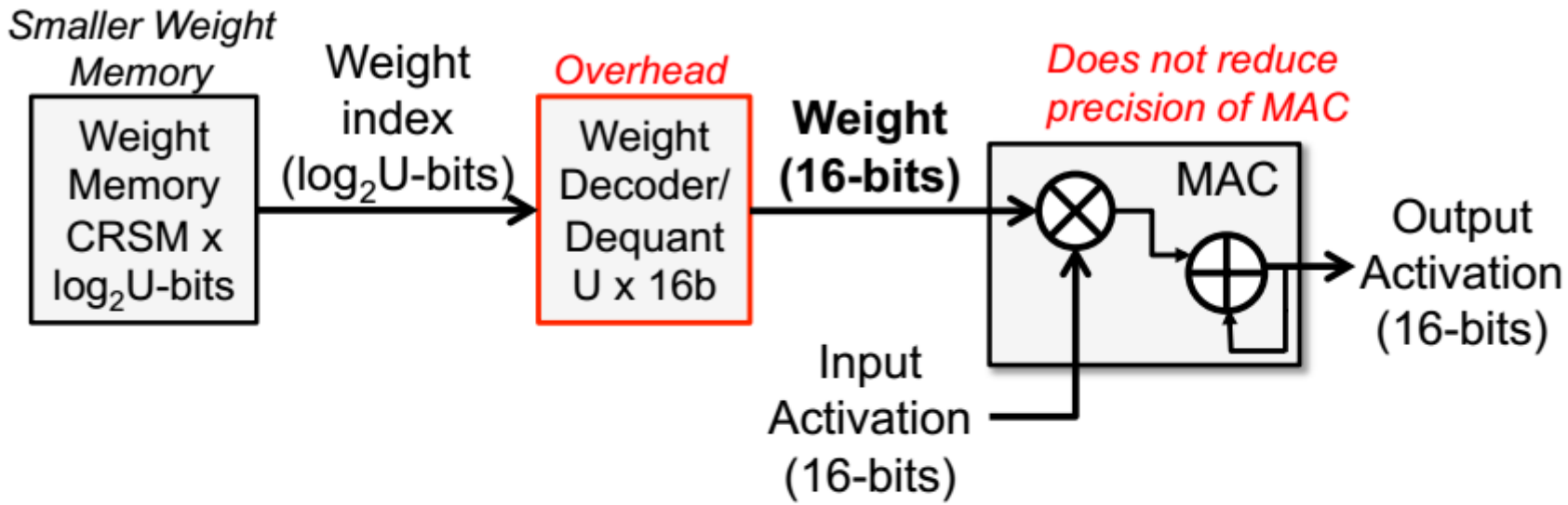
[Han et al., ICLR 2016]

- **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

Full list @ [Sze et al., arXiv, 2017]

Network Sparsity

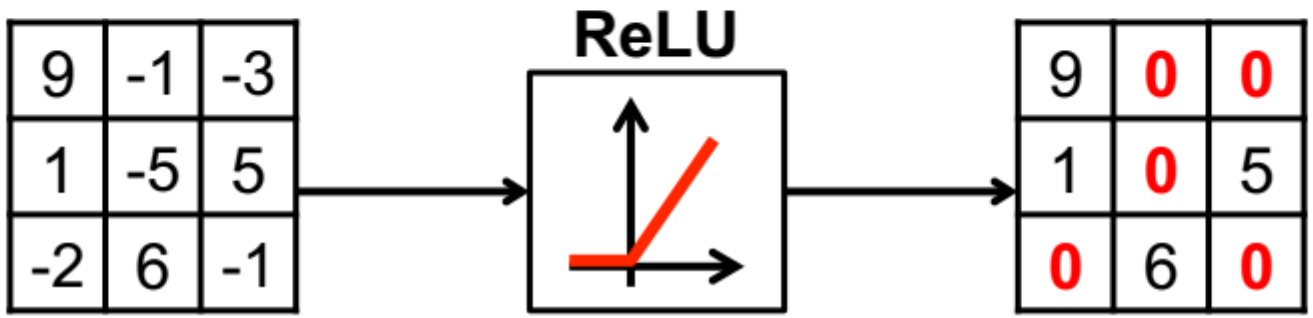
- Human's neurons are highly sparse
- So as the matrix multiplication in NNs

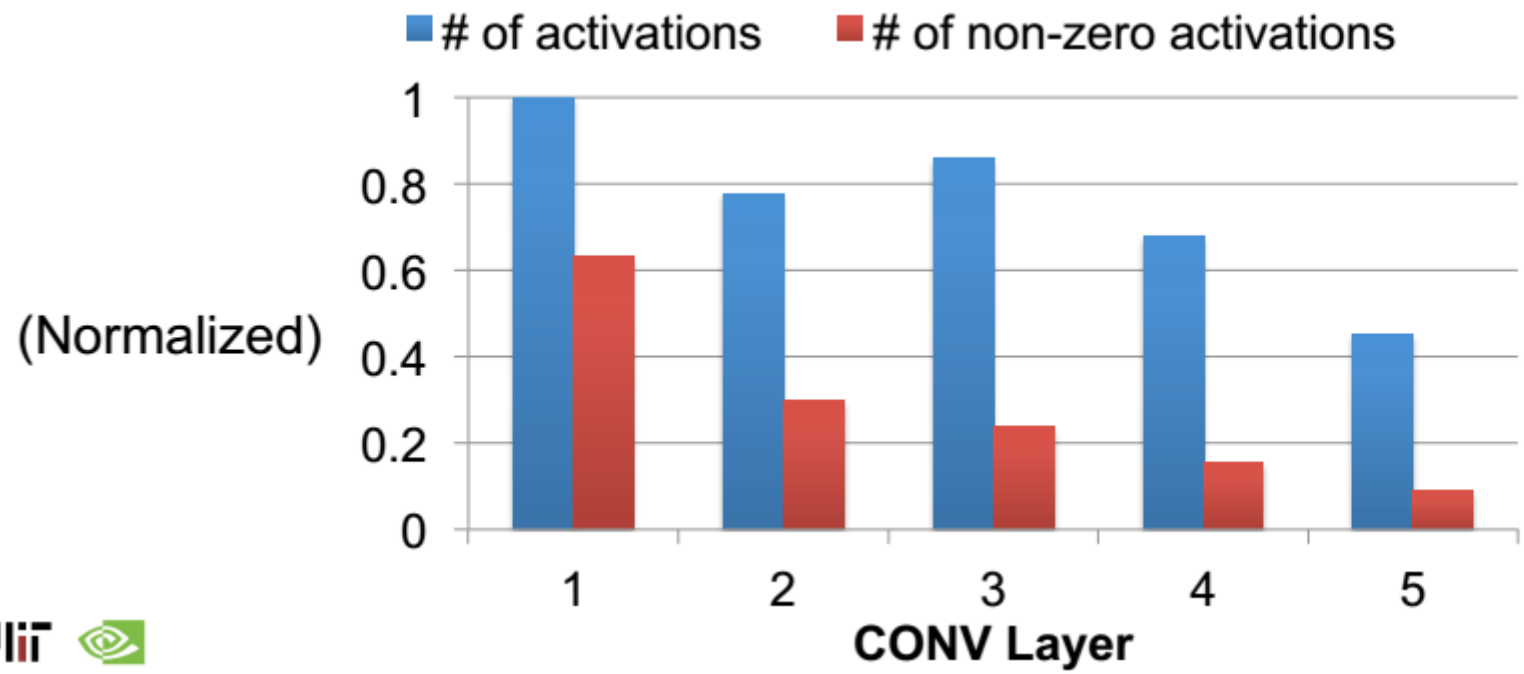
- Exploiting Sparsity in Deep learning hardware
- Reducing numbers of Ops and Weights
- Activation Statistics
- Network Prunning
- Knowledge Distillation

$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$

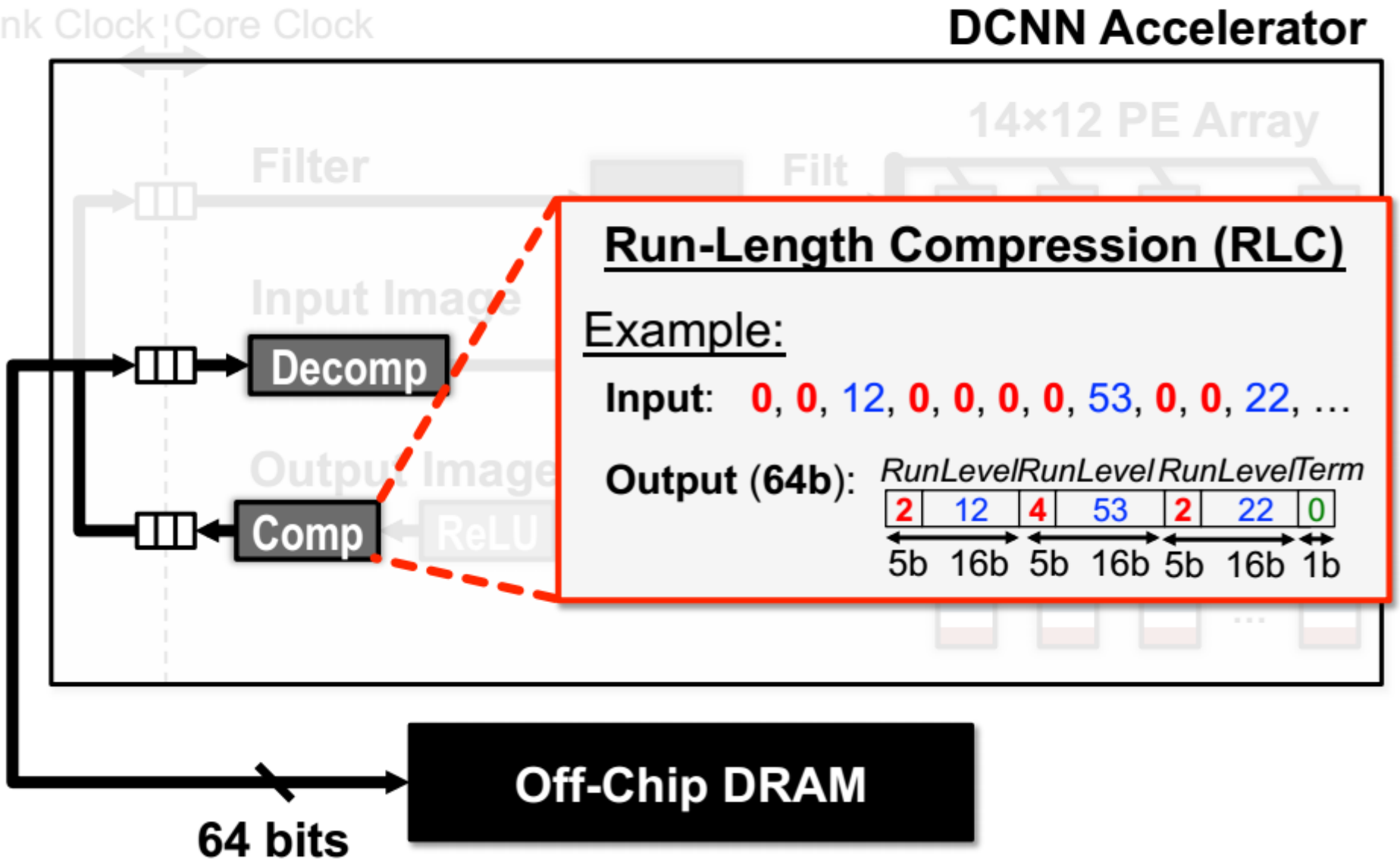
Sparsity in Fmaps

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



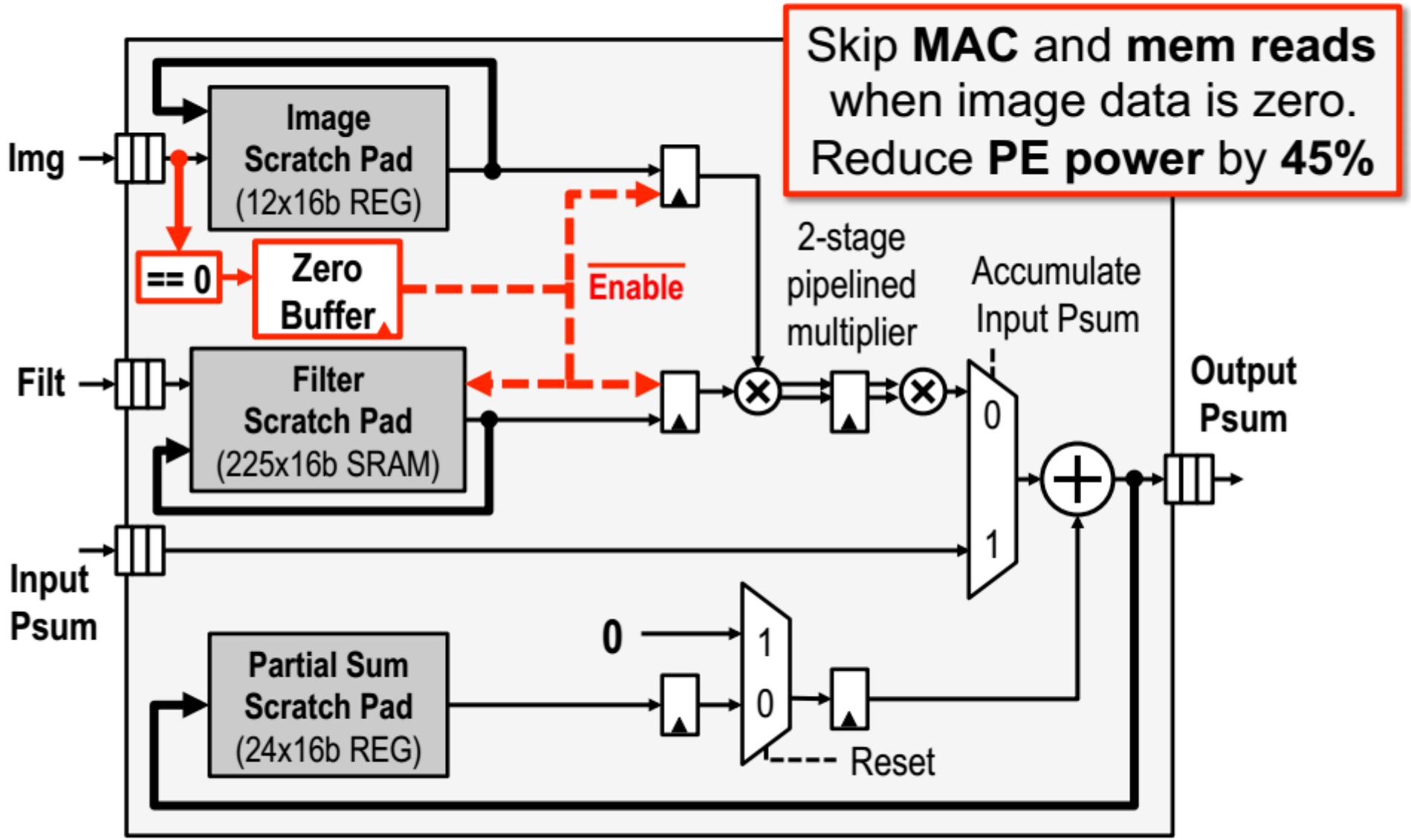


I/O Compression in Eyeriss



[Chen et al., ISSCC 2016]

Data Gating / Zero Skipping in Eyeriss

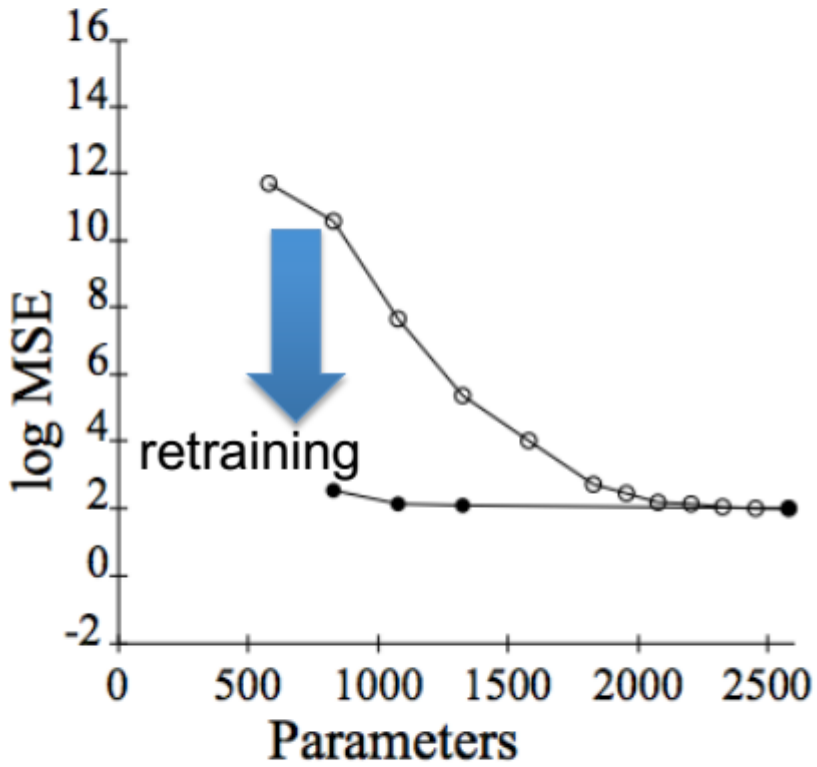


[Chen et al., ISSCC 2016]

Pruning – Make Weights Sparse

- **Optimal Brain Damage**

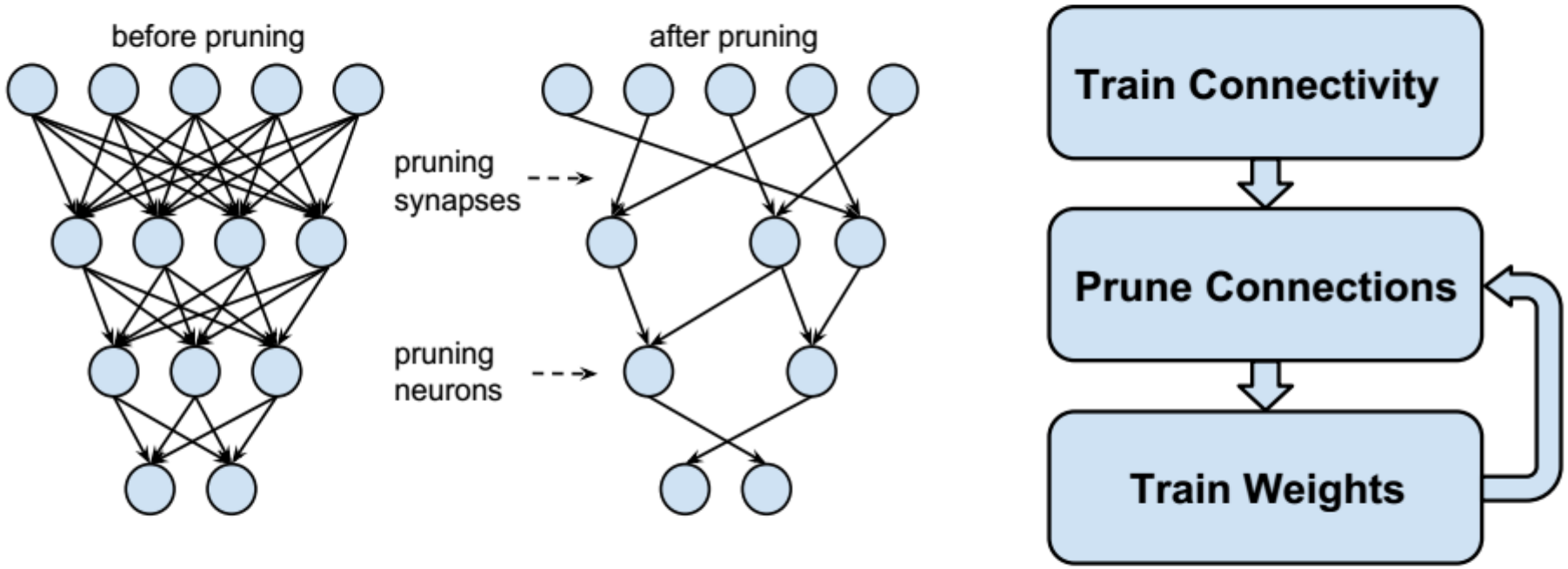
1. **Choose a reasonable network architecture**
2. **Train network until reasonable solution obtained**
3. **Compute the second derivative for each weight**
4. **Compute saliencies (i.e. impact on training error) for each weight**
5. **Sort weights by saliency and delete low-saliency weights**
6. **Iterate to step 2**



[Lecun et al., NIPS 1989]

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]

Compression of Weights & Activations

- **Compress weights and activations between DRAM and accelerator**
- **Variable Length / Huffman Coding**

Example:

Value: **16'b0** → Compressed Code: {**1'b0**}

Value: **16'bx** → Compressed Code: {**1'b1**, **16'bx**}

- Tested on AlexNet → **2× overall BW Reduction**

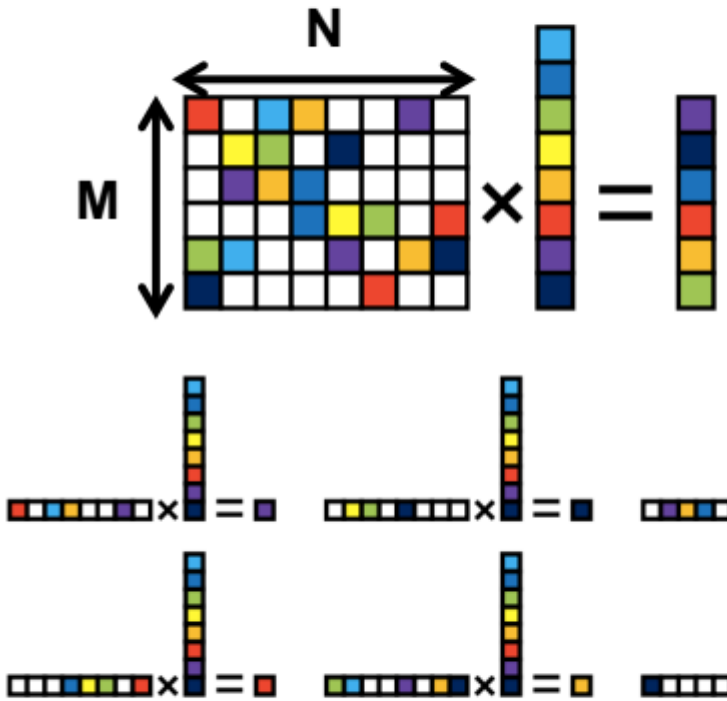
Layer	Filter / Image bits (0%)	Filter / Image BW Reduc.	IO / HuffIO (MB/frame)	Voltage (V)	MMACs/ Frame	Power (mW)	Real (TOPS/W)
General CNN	16 (0%) / 16 (0%)	1.0x		1.1	–	**288**	**0.3**
AlexNet l1	7 (21%) / 4 (29%)	1.17x / 1.3x	1 / 0.77	0.85	105	85	0.96
AlexNet l2	7 (19%) / 7 (89%)	1.15x / **5.8x**	3.2 / 1.1	0.9	224	55	1.4
AlexNet l3	8 (11%) / 9 (82%)	1.05x / 4.1x	6.5 / 2.8	0.92	150	77	0.7
AlexNet l4	9 (04%) / 8 (72%)	1.00x / 2.9x	5.4 / 3.2	0.92	112	95	0.56
AlexNet l5	9 (04%) / 8 (72%)	1.00x / 2.9x	3.7 / 2.1	0.92	75	95	0.56
Total / avg.	–	–	19.8 / **10**	–	–	**76**	**0.94**
LeNet-5 l1	3 (35%) / 1 (87%)	1.40x / 5.2x	[illegible]	0.7	0.3	25	1.07
LeNet-5 l2	4 (26%) / 6 (55%)	1.25x / 1.9x	0.050 / 0.042	0.8	1.6	35	1.75
Total / avg.	–	–	0.053 / **0.043**	–	–	**33**	**1.6**

[Moons et al., VLSI 2016; Han et al., ICLR 2016]

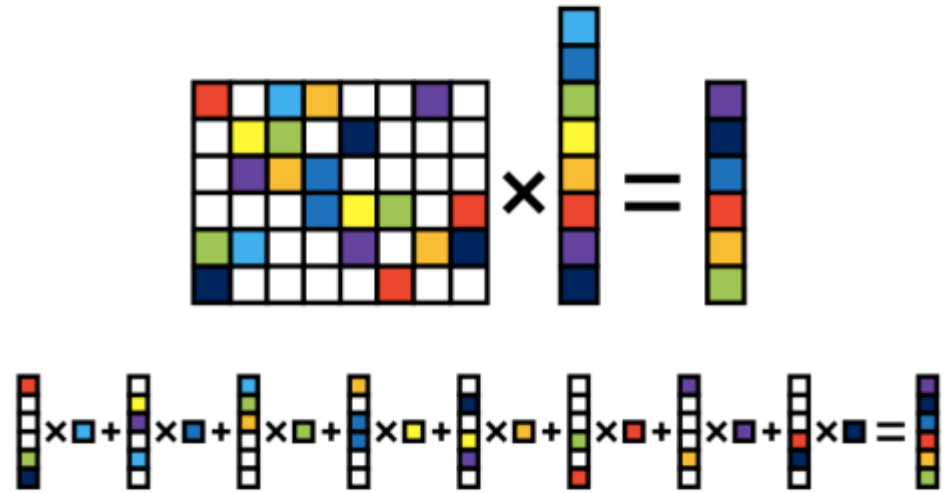
Sparse Matrix-Vector DSP

- **Use CSC rather than CSR for SpMxV**

Compressed Sparse Row (CSR)



Compressed Sparse Column (CSC)



Reduce memory bandwidth (when not **M** >> **N**)

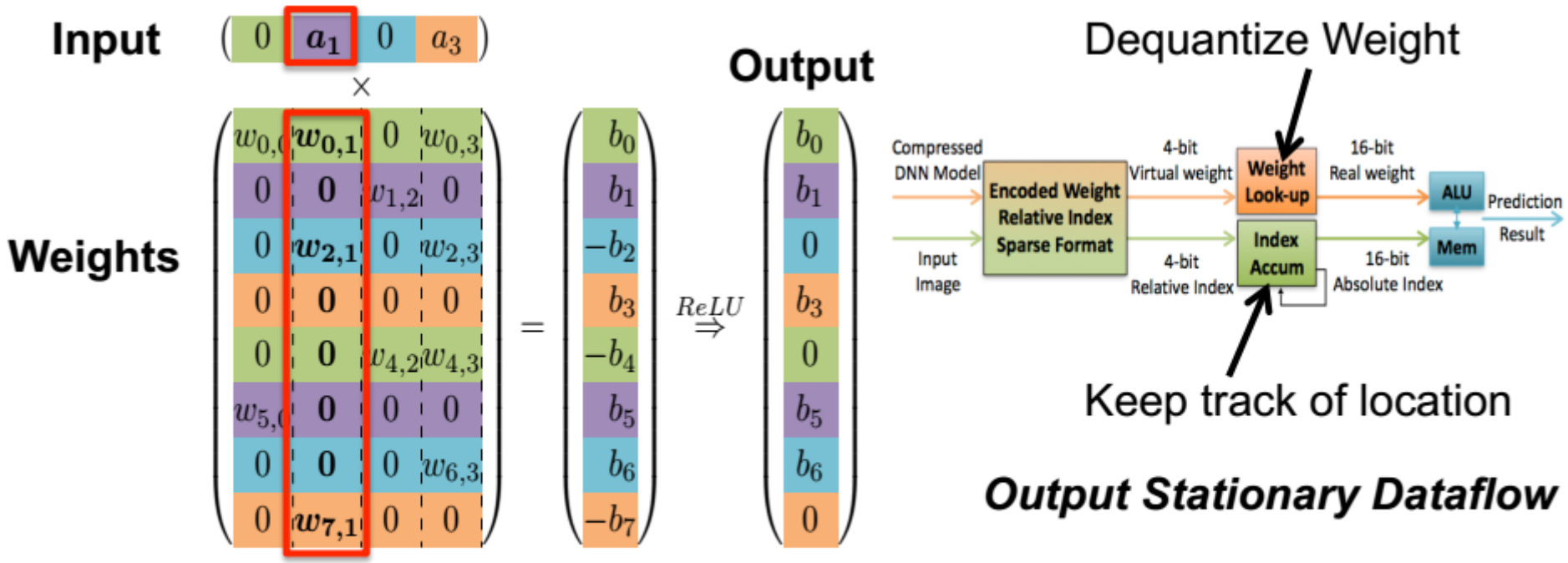
*For DNN, **M** = # of filters, **N** = # of weights per filter*

[Dorrance et al., FPGA 2014]

EIE: A Sparse Linear Algebra Engine

- **Process Fully Connected Layers (after Deep Compression)**
- **Store weights column-wise in Run Length format**
- **Read relative column when input is non-zero**

Supports Fully Connected Layers Only



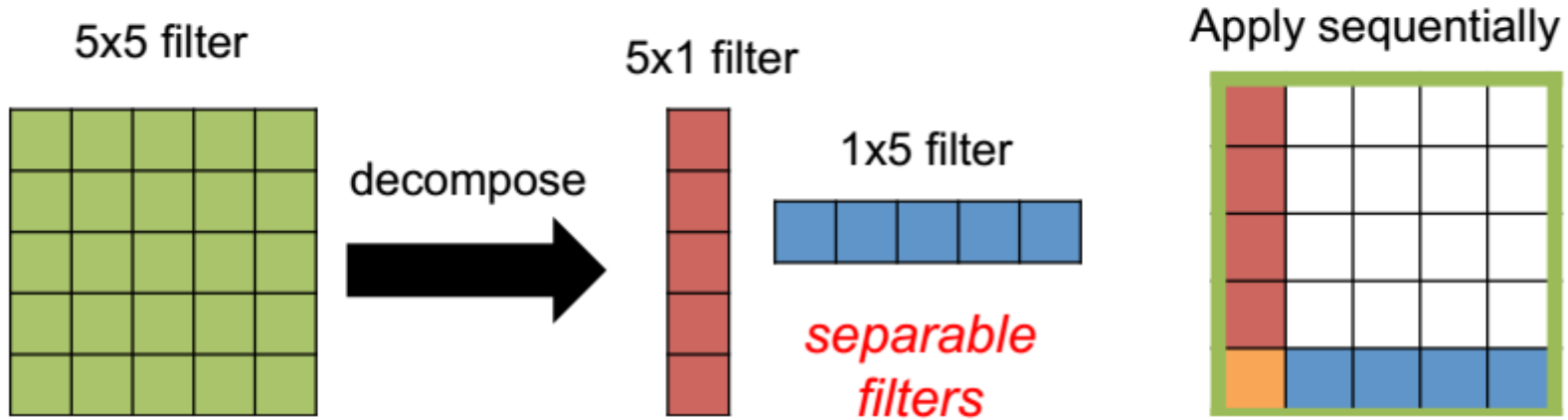
Output Stationary Dataflow

[Han et al., ISCA 2016]

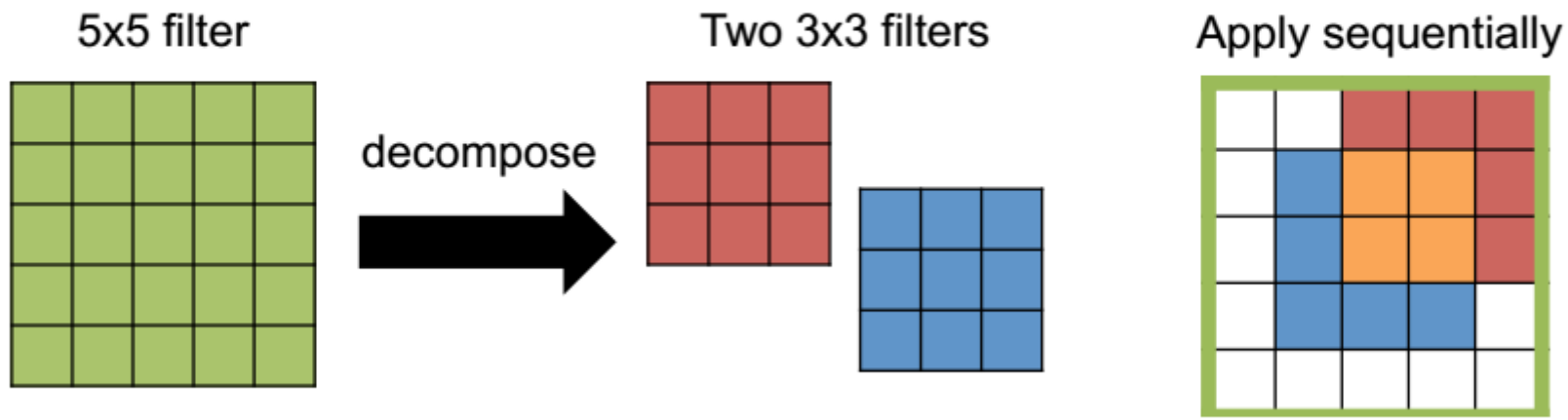
Network Architecture Design

Build Network with series of Small Filters

GoogleNet/Inception v3



VGG-16



Network Architecture Design

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

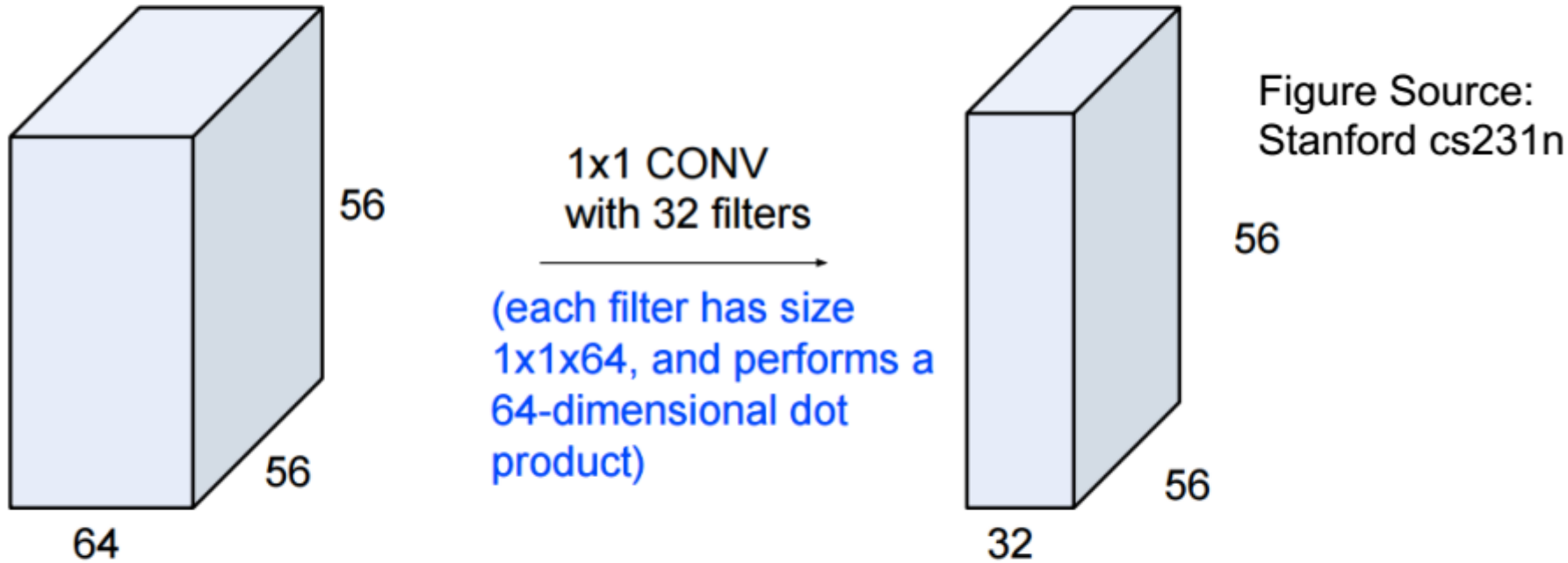


Figure Source:
Stanford cs231n

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**)

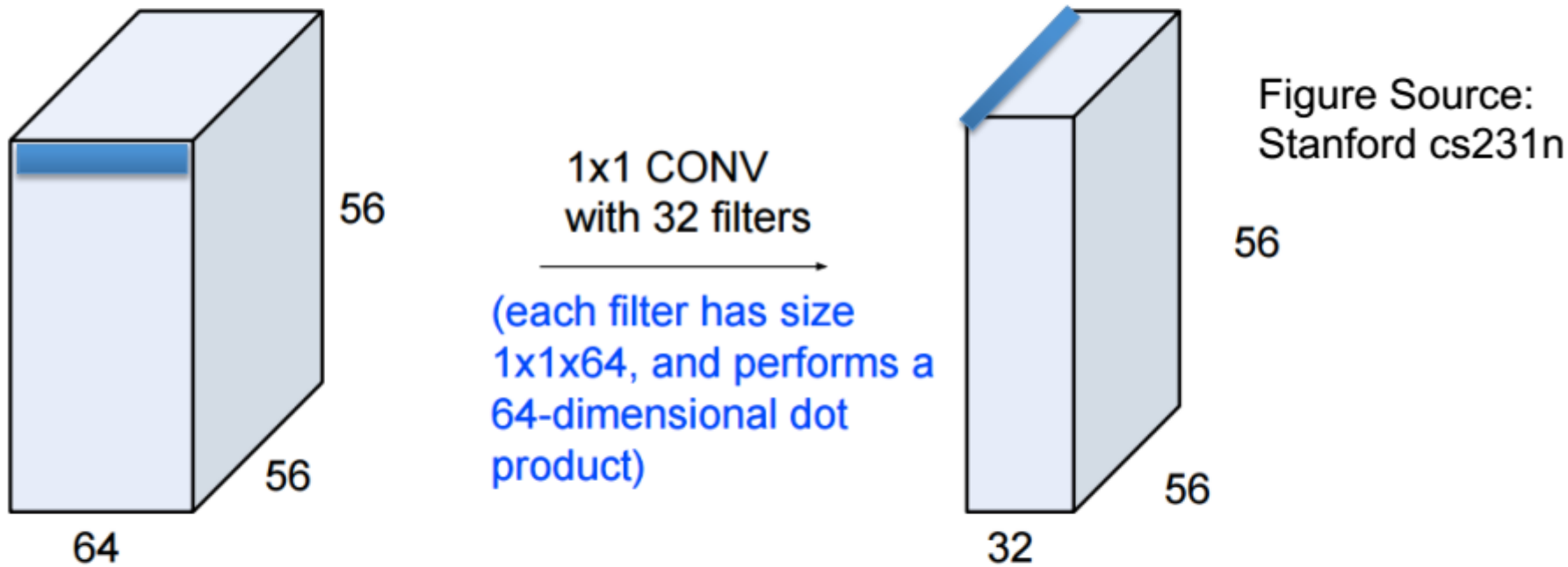


Figure Source:
Stanford cs231n

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**)

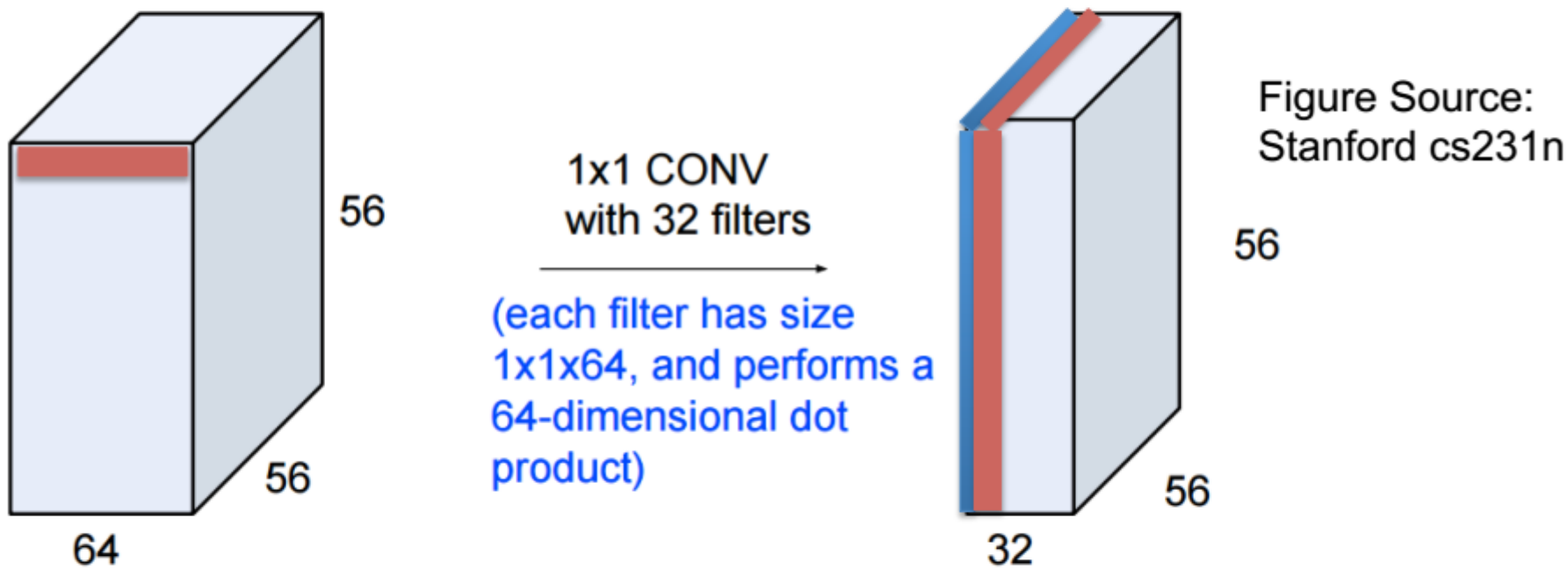
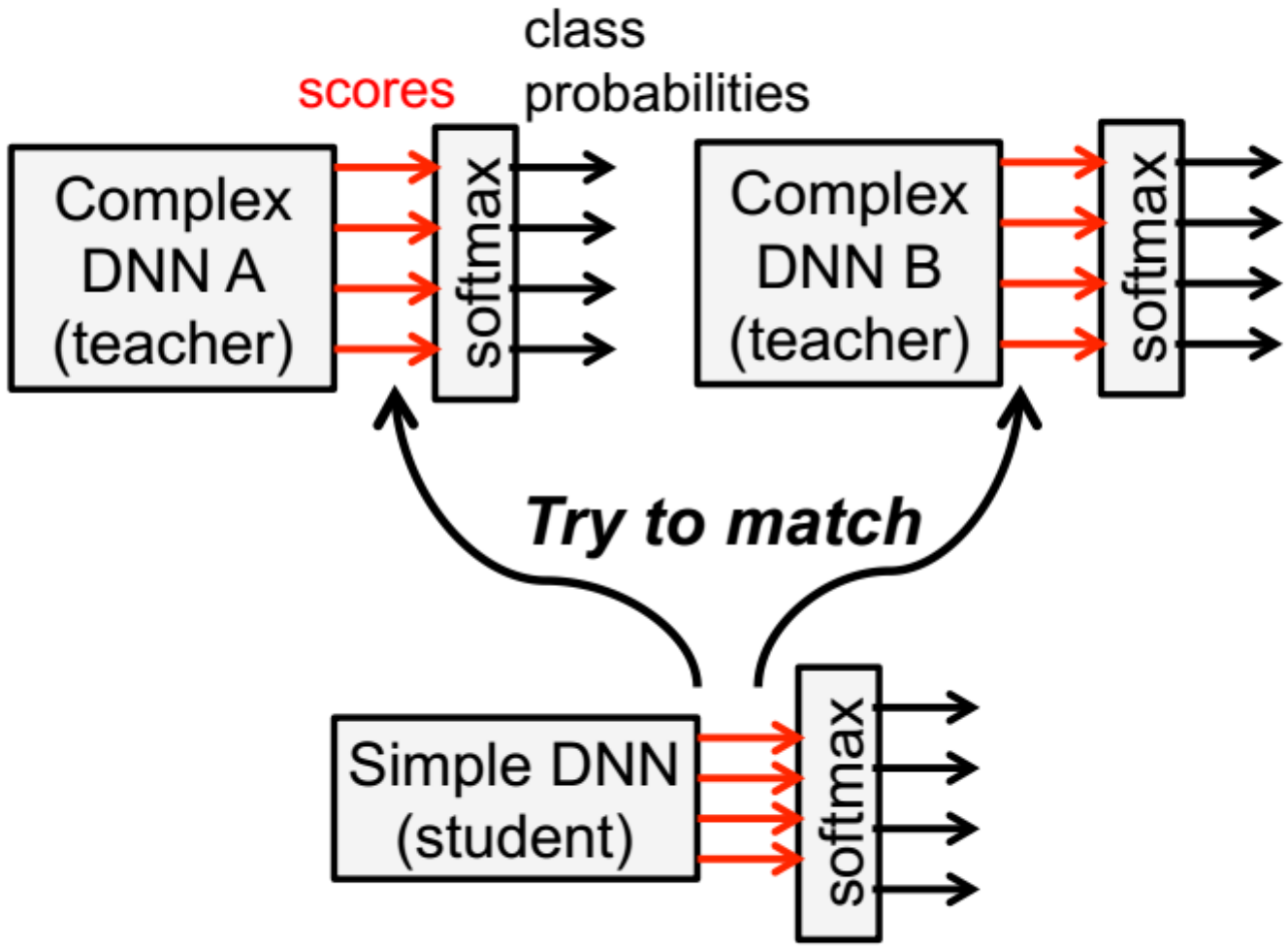


Figure Source:
Stanford cs231n

Used in Network In Network(NiN) and GoogLeNet

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

Knowledge Distillation



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

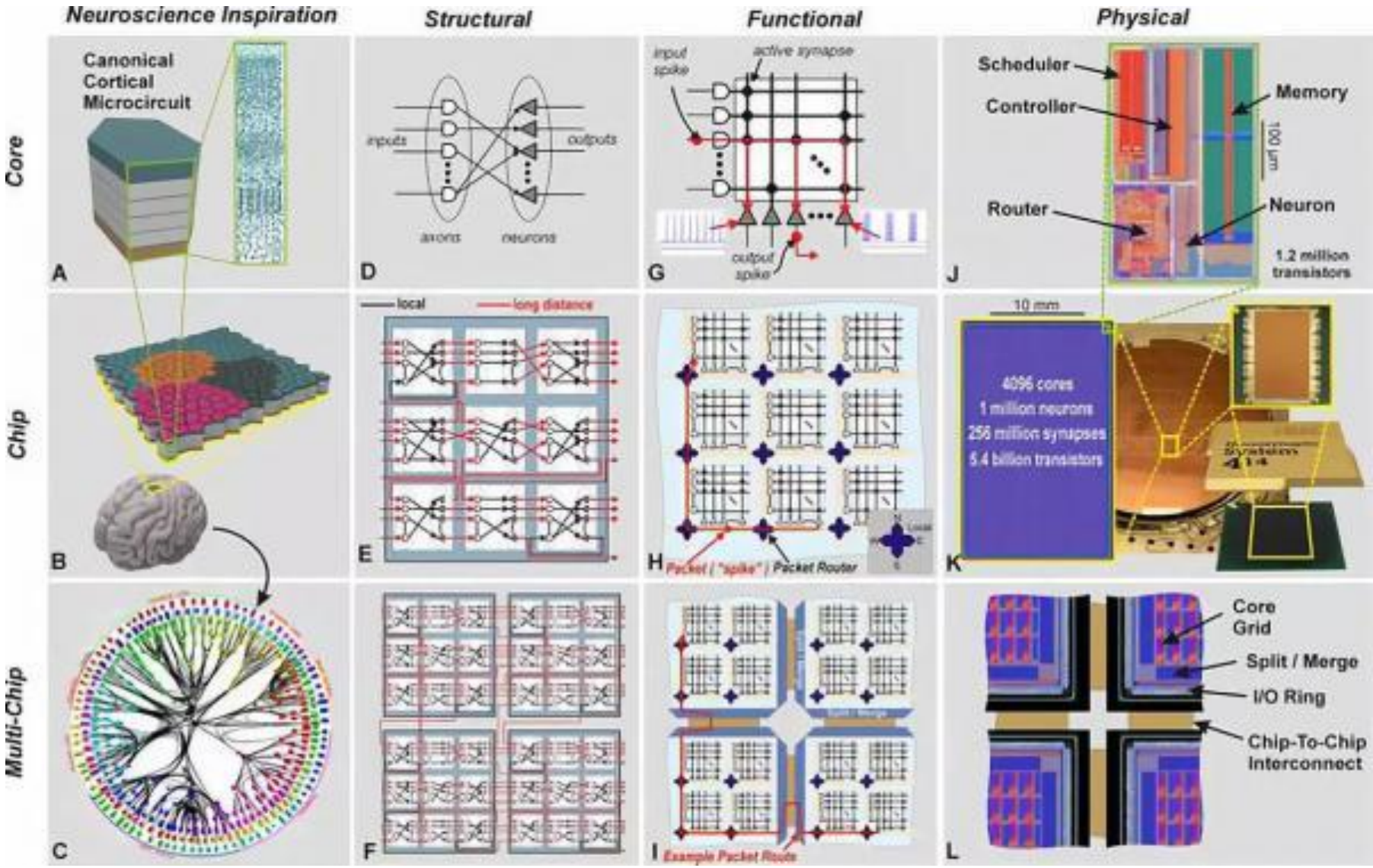
Neuromophic Implementation

Analog Computing etc.

True north

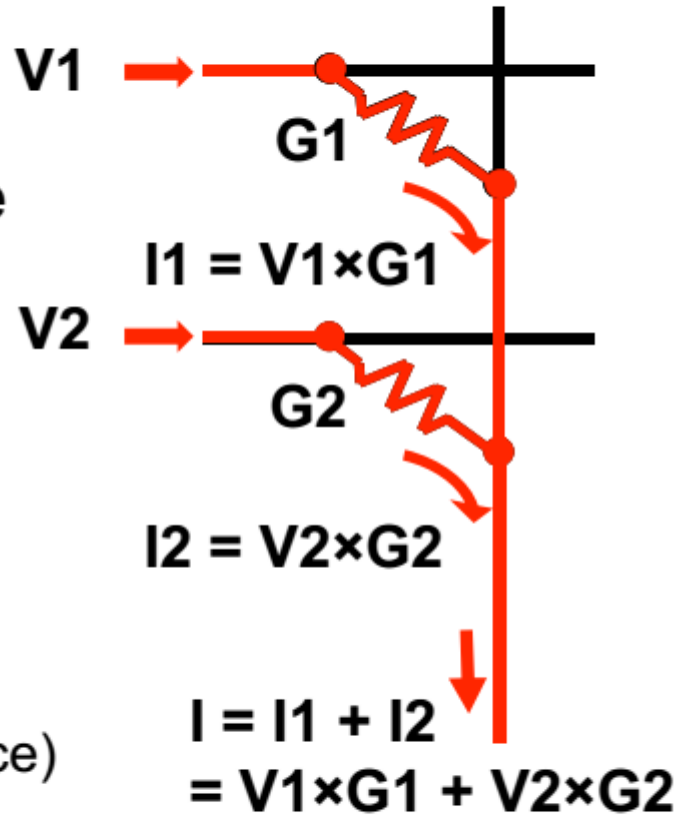
DARPA funded the first brain-inspired processor

1 Million Neurons
70 mW



Analog Computation

- **Conductance = Weight**
- **Voltage = Input**
- **Current = Voltage × Conductance**
- **Sum currents for addition**



$$Output = \sum Weight \times Input$$

Input = V1, V2, ...

Filter Weights = G1, G2, ... (conductance)

Weight Stationary Dataflow

Figure Source: ISAAC, ISCA 2016

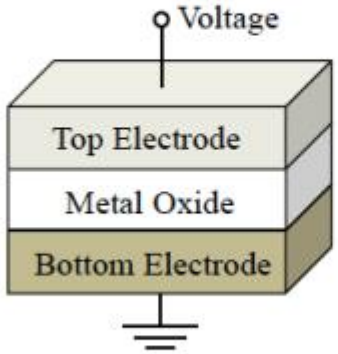
Memristor Computation

Use memristors as programmable weights (resistance)

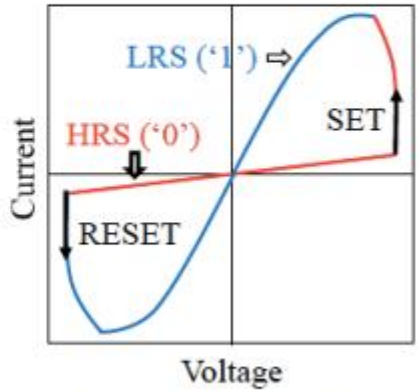
- **Advantages**
  - **High Density (< 10nm x 10nm size*)**
    - **~30x smaller than SRAM****
    - **1.5x smaller than DRAM****
  - **Non-Volatile**
  - **Operates at low voltage**
  - **Computation within memory (in situ)**
    - **Reduce data movement**

*[Govoreanu et al., IEDM 2011], **ITRS 2013

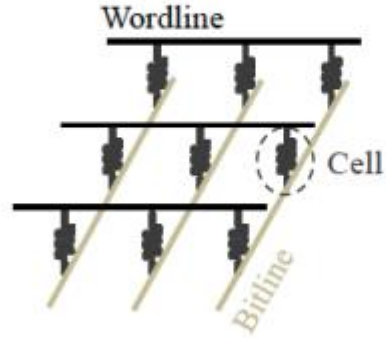
Memristor



(a) Conceptual view of a ReRAM cell

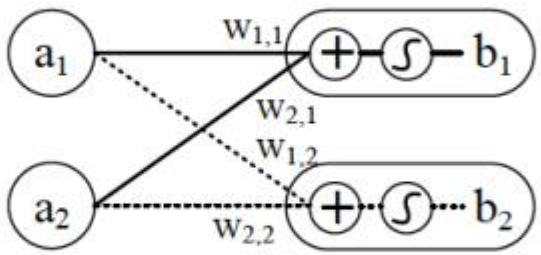


(b) I-V curve of bipolar switching

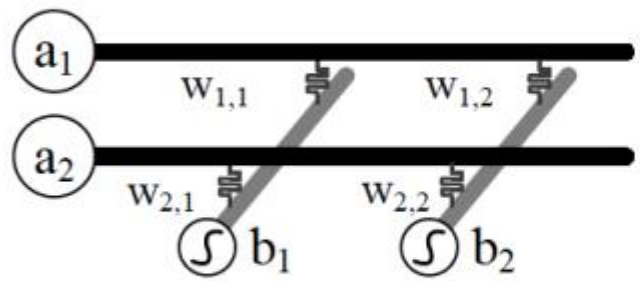


(c) schematic view of a crossbar architecture

$$b_j = \sigma\left(\sum_{\forall i} a_i \cdot w_{i,j}\right)$$



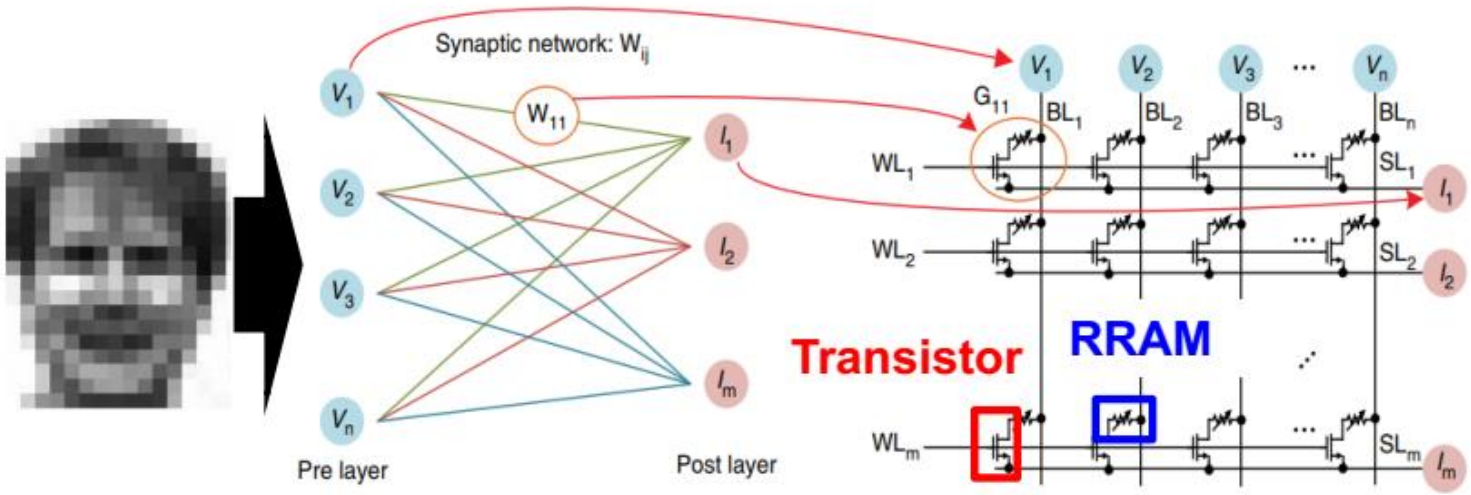
(a) An ANN with one input and one output layer

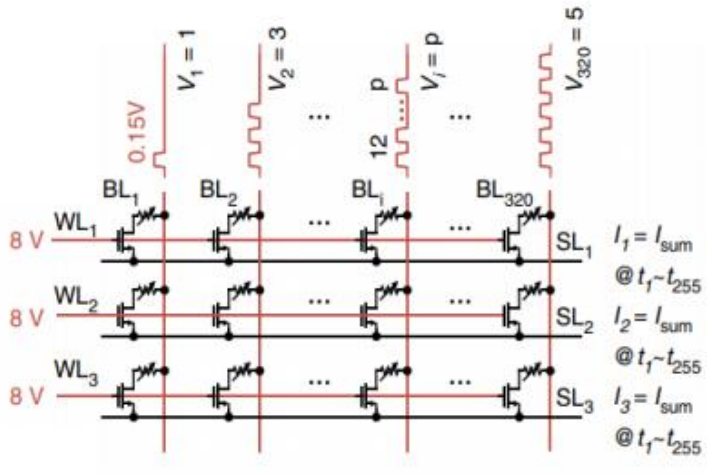


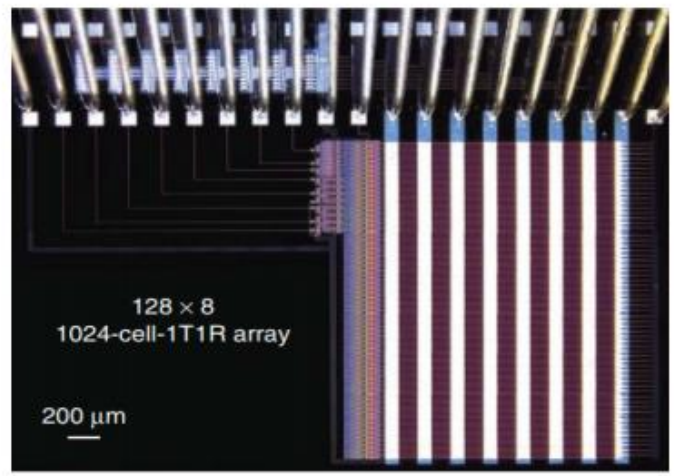
(b) using a ReRAM crossbar array for neural computation

[Chi et al., ISCA 2016]

RRAM Array for Analog Computation







P. Yao, Nature Comm. 2017
W. Wu, VLSI 2018

Tsinghua Univ.

❑ Applications

- Face recognition
- Tested on Yale face dataset

❑ Analog RRAM array

- 1K 1T1R cells
- Layer's input ➔ BL pulse input
- Synaptic weight ➔ cell conduc.
- Layer's output ➔ SL current

❑ MNIST with 8 2K-arrays:

- accuracy > 96%

Yao, Peng, et al. "Face classification using electronic synapses." *Nature communications* 8 (2017): 15199.

Challenges with Memristors

- **Limited Precision**
- **A/D and D/A Conversion**
- **Array Size and Routing**
  - **Wire dominates energy for array size of 1k × 1k**
  - **IR drop along wire can degrade read accuracy**
- **Write/programming energy**
  - **Multiple pulses can be costly**
- **Variations & Yield**
  - **Device-to-device, cycle-to-cycle**
  - **Non-linear conductance across range**

[Eryilmaz et al., ISQED 2016]

Conclusion