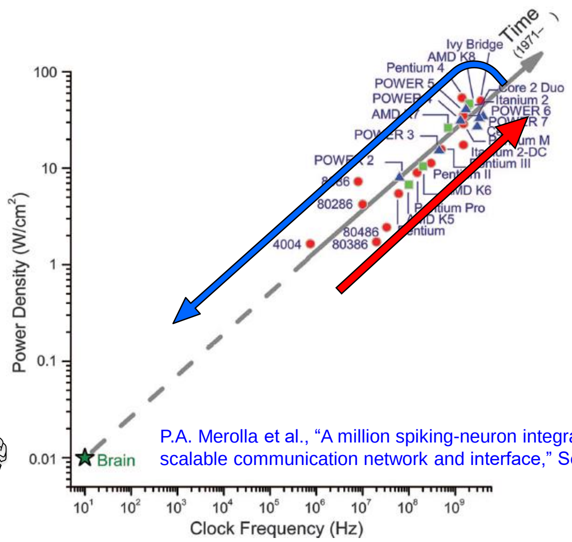
Algorithms and Architectures for Efficient Neural Processing

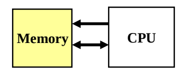
July 19, 2019
Kiyoung Choi
Dept. ECE, NPRC
Seoul National Univ

Neural Processing Research Center
뉴럴프로세싱 연구센터
서울대학교

Contents

Introduction

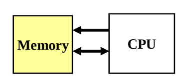




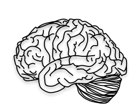
P.A. Merolla et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” Science, Aug. 2014.

Artificial Neural Network

◆ **Conventional computing**
- von Neumann architecture
- Accurate with full precision binary computing
- High cost in area and energy consumption
- Memory wall problem

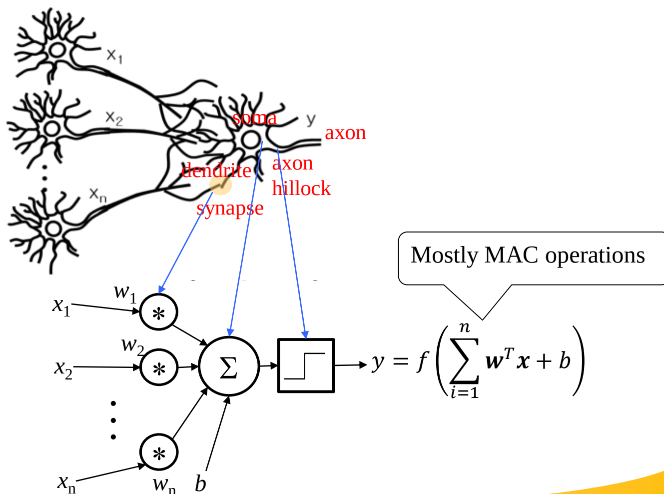


◆ **Human brain**
- Consumes ~20W power
- Does not perform precise computing
- Very well recognizes objects



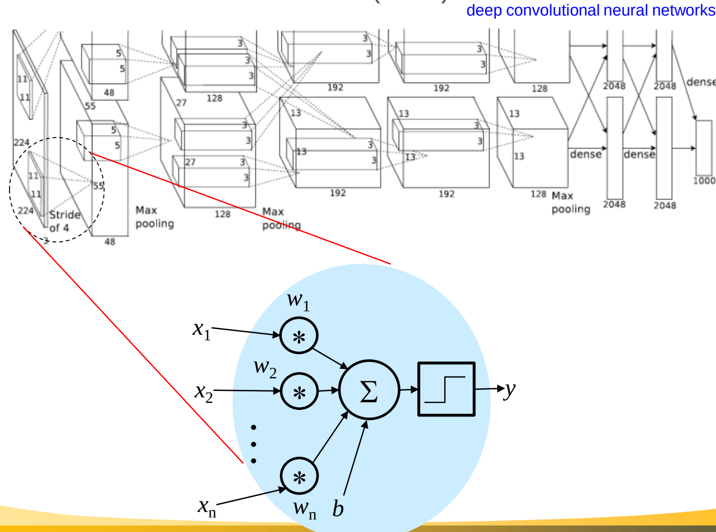
Artificial Neural Network

- **Neural network models & implementations**
  - Perceptron model



Artificial Neural Network

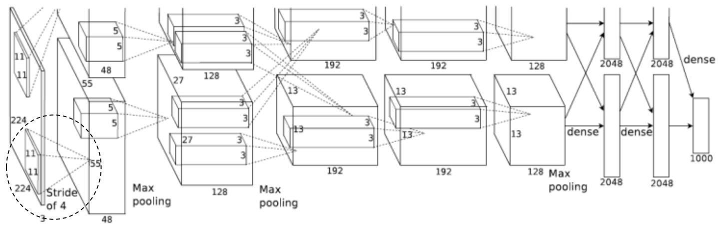
- **Neural network models & implementations**
  - Convolutional neural networks (CNNs) A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks, NIPS 2012.

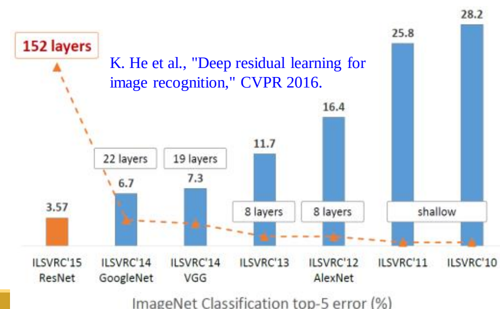


Artificial Neural Network

◆ Neural network models & implementations

- Convolutional neural networks (CNNs) A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks, NIPS 2012.



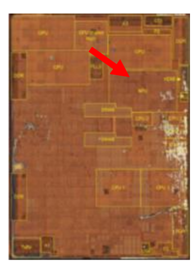


Artificial Neural Network

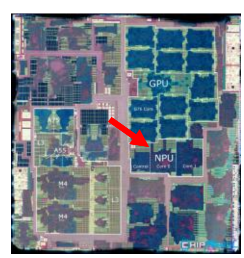
◆ Deployments

Qualcomm Snapdragon 845

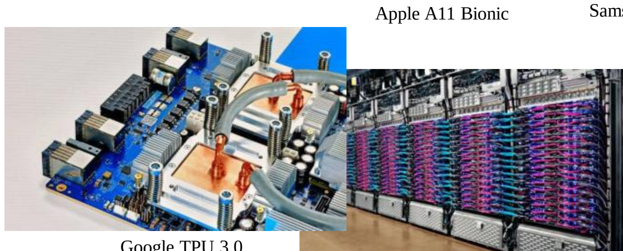
Huawei Kirin 970



Apple A11 Bionic



Samsung Exynos 9820

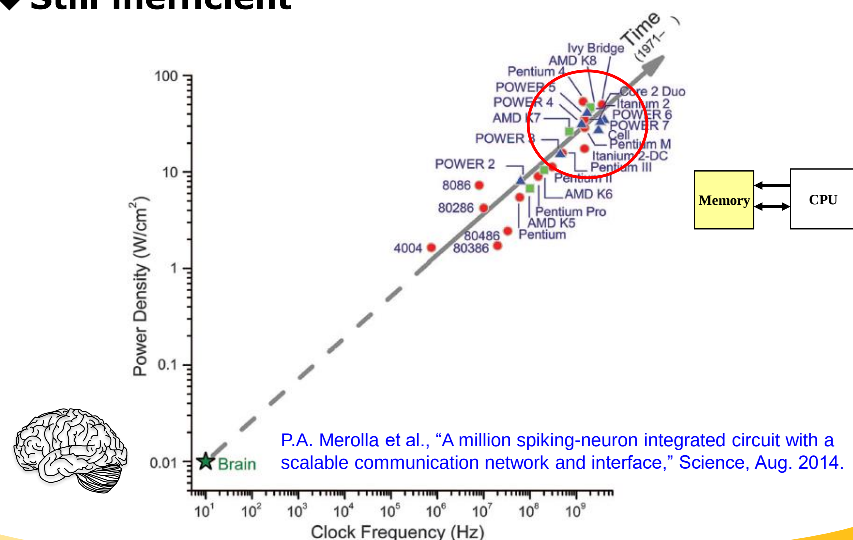


Google TPU 3.0

8

Artificial Neural Network

◆ Still inefficient

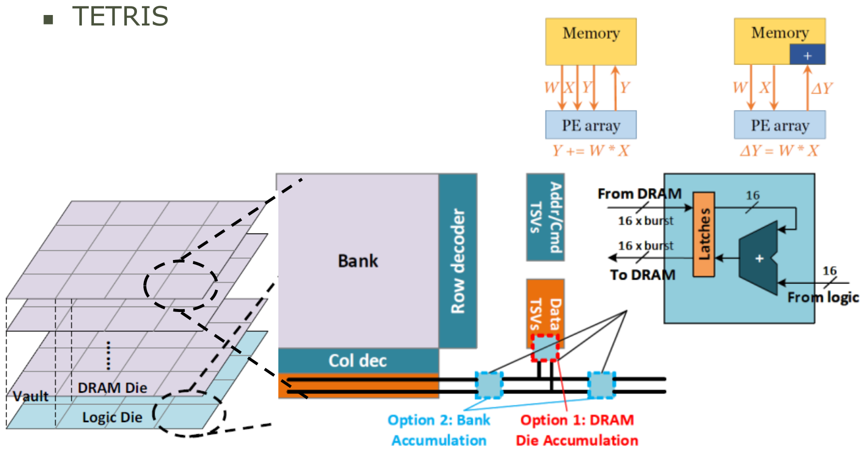


P.A. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," Science, Aug. 2014.

Artificial Neural Network

◆ Near-data processing

- TETRIS



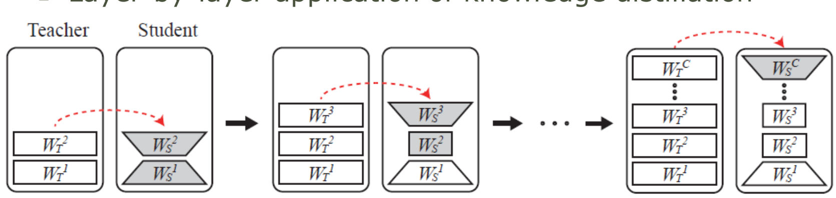
M. Gao et al., "TETRIS: scalable and efficient neural network acceleration with 3D memory," ASPLOS 2017.

Network Reduction

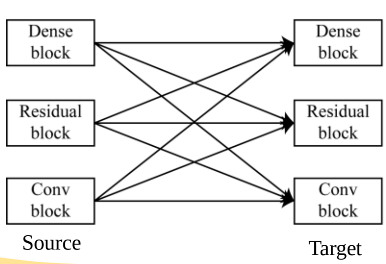
- **Weight pruning**
  - Y. Guo et al., "Dynamic network surgery for efficient dnns," NIPS 2016
  - S. Han et al., "Learning both weights and connections for efficient neural network," NIPS 2015
- **Filter pruning**
  - H. Hu et al., " Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," arXiv:1607.03250, 2016.
  - J.-H. Luo and J. Wu, "Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference," arXiv:1805.08941, 2018
- **Knowledge distillation**
  - G. Hinton et al.' "Distilling the knowledge in a neural net work," NIPS Workshop 2014
  - J. Yoo et al., "Network recasting: a universal method for network architecture transformation," AAAI, 2019.

Network Reduction

- **Network recasting**
  - Layer-by-layer application of knowledge distillation



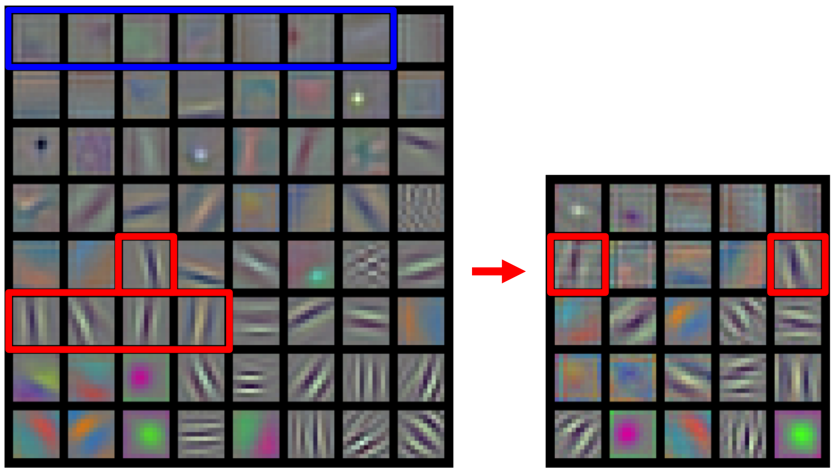
  - Recasting into an arbitrary target block



J. Yoo et al., "Network recasting: a universal method for network architecture transformation," AAAI, 2019.

Network Reduction

◆ Filter-pruning effect



Network Reduction

◆ Performance

- Much less memory access due to reduced activation

Method	Top1	Top5	Params	Mults	Actual speed-up
		ResNet-50			
Recasting(C+R_{bt})	**25.00**	**7.71**	21.72M	2.40B	**2.1×**
ThiNet-30 [1]	31.58	11.7	8.66M	1.10B	1.3×
AutoPruner ($r = 0.3$) [2]	27.47	8.89	-	1.32B	-
		VGG-16			
Recasting(C_A)	**30.05**	**10.38**	120.61M	3.12B	**3.2×**
ThiNet-Conv [1]	30.20	10.47	131.44M	4.79B	2.5×
RNP (3×) [3]	-	12.42	-	-	2.3×
Channel Pruning (3×) [4]	-	11.10	-	-	2.5×
AutoPruner ($r = 0.4$) [2]	31.57	11.57	-	4.09B	-

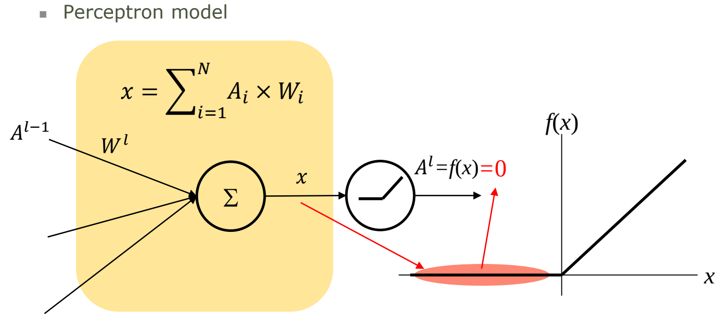
Comparison with previous works. (batch size is 64, NVIDIA Titan X (pascal))

Zero Skipping

- **Exploiting zeros in inputs**
  - J. Albericio et al., "Cnvlutin: ineffectual-neuron-free deep neural network computing," ISCA, 2016
  - P. Judd et al., "Stripes: Bit-serial Deep Neural Network Computing ," Computer Architecture Letters, 2016
  - D. Kim et al., "ZeNA: Zero-Aware Neural Network Accelerator," IEEE Design & Test, Feb. 2018
- **Exploiting zeros in outputs**
  - V. Akhlaghi et al., "SnaPEA: Predictive Early Activation for Reducing Computation in Deep Convolutional Neural Networks," ISCA 2018
  - D. Lee et al., "ComPEND: computation pruning through early negative detection," ICS, 2018
- **For training**
  - G. Lee et al., "Acceleration of DNN Backward Propagation by Selective Computation of Gradients," DAC 2019, to be presented.

Zero Skipping

- **ComPEND**
  - Computation Pruning through Early Negative Detection
  - Motivation
    - Perceptron model



  - Rectified linear unit (ReLU, [$f(x)$ = max(0,x)]) is widely used as an activation function for DNN
  - If we know *a priori* that $x \leq 0$, we can skip unnecessary computations

D. Lee et al., "ComPEND: computation pruning through early negative detection," ICS, June 2018.

Zero Skipping

◆ Early Negative Detection (END)

- Two's complement number representation (4 bits)

Negative

1111 = -8+7 = -1
1110 = -8+6 = -2
1101 = -8+5 = -3
1100 = -8+4 = -4
1011
1010 .
1001 .
1000 .

Positive

0111
0110
0101
0100
0011
0010
0001 = -0+1 = +1
0000 = -0+0 = +0

For a B-bit number W : ($w_{B-1}\ w_{B-2}\ w_{B-3} \dots w_1\ w_0$)

$$W = w_{B-1} \times (-2^{B-1}) + \sum_{k=0}^{B-2} w_k \times (+2^k)$$

Zero Skipping

◆ Early Negative Detection (END)

- Inverted two's complement number representation (4 bits)

Positive

1111 = +8-7 = +1
1110 = +8-6 = +2
1101 = +8-5 = +3
1100 = +8-4 = +4
1011
1010 .
1001 .
1000 .

Negative

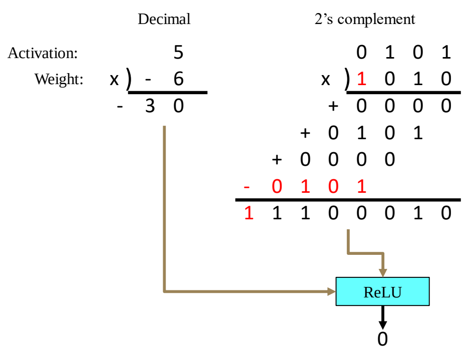
0111
0110
0101
0100
0011
0010
0001 = +0-1 = -1
0000 = +0-0 = -0

For a B-bit number W : ($w_{B-1}\ w_{B-2}\ w_{B-3} \dots w_1\ w_0$)

$$W = w_{B-1} \times (+2^{B-1}) + \sum_{k=0}^{B-2} w_k \times (-2^k)$$

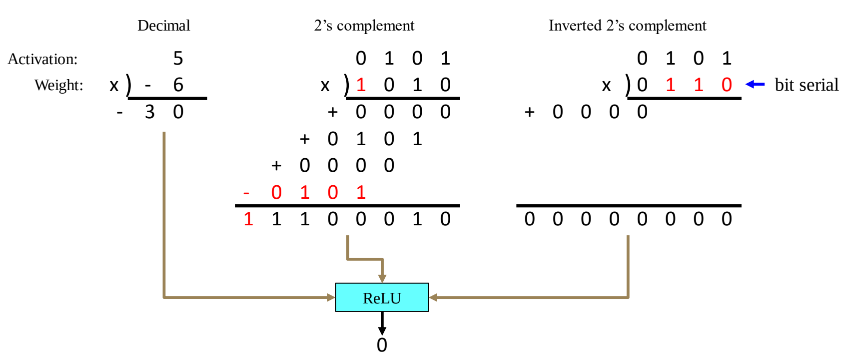
Zero Skipping

- **Early Negative Detection (END)**
  - Inverted two's complement for negative detection



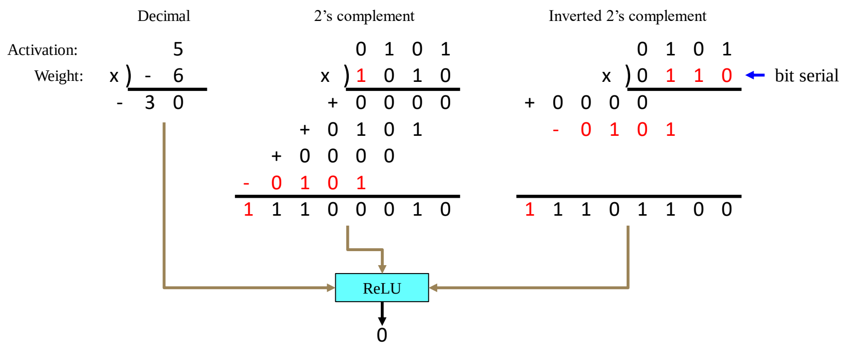
Zero Skipping

- **Early Negative Detection (END)**
  - Inverted two's complement for negative detection



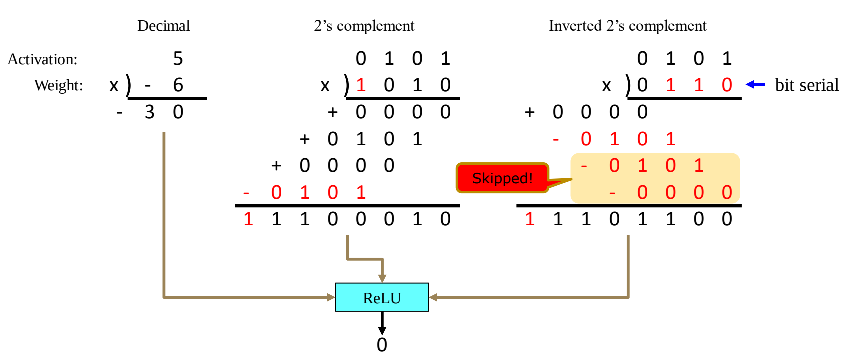
Zero Skipping

- **Early Negative Detection (END)**
  - Inverted two's complement for negative detection



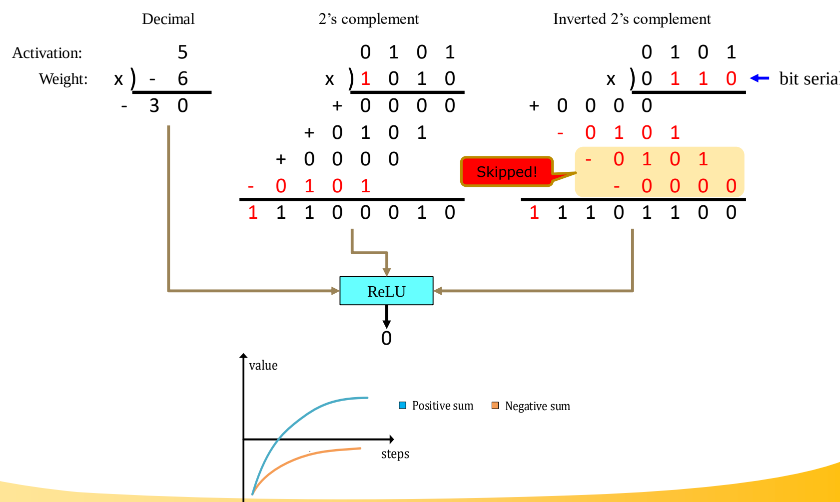
Zero Skipping

- **Early Negative Detection (END)**
  - Inverted two's complement for negative detection



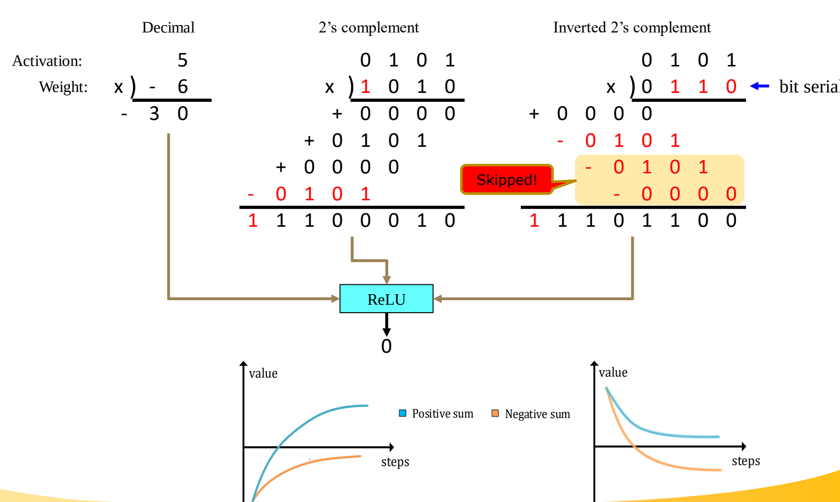
Zero Skipping

- **Early Negative Detection (END)**
  - Inverted two's complement for negative detection



Zero Skipping

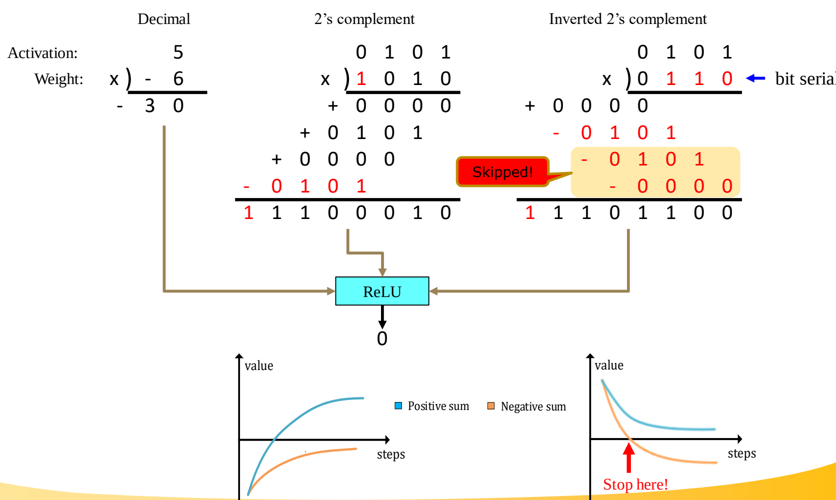
- **Early Negative Detection (END)**
  - Inverted two's complement for negative detection



Zero Skipping

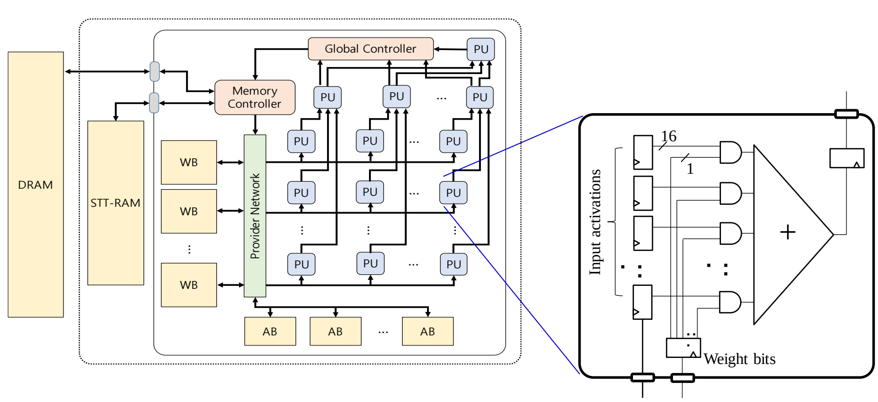
◆ Early Negative Detection (END)

- Inverted two's complement for negative detection



Zero Skipping

◆ ComPEND architecture



Zero Skipping

◆ Comparison with other architectures

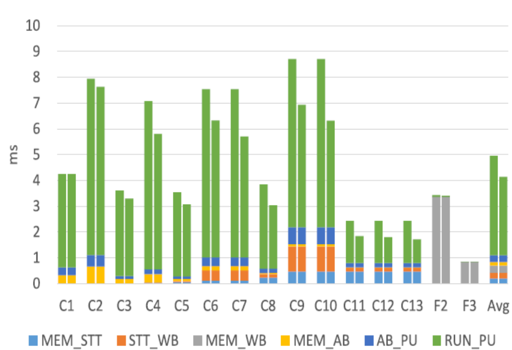
	Eyeriss	DaDianNao	ComPEND (w/o zero-skipping)
Precision	16	16	16
Technology	65 nm	28 nm	45 nm
Clock frequency	250 MHz	606 MHz	1000 MHz
Throughput	42 GMACS	2,790 GMACS	288 GMACS
Core Area	12.25 mm²	67.73 mm²	5.62 mm²
Area efficiency	3.43 GMACS/mm²	41.19 GMACS/mm²	51.25 GMACS/mm²
Power	450 mW	15,970 mW	1,180 mW
Power efficiency	93.3 GMACS/W	174.7 GMACS/W	244.1 GMACS/W

27

Zero Skipping

◆ Runtime

- ComPEND reduces runtime by 16.62% on average



- MEM_STT: reads/writes between off-chip memory and STT-RAM
- STT_WB: runtime of reads/writes between STT-RAM and WB
- MEM_WB: reads/writes between off-chip memory and WB
- MEM_AB: reads/writes between off-chip memory and AB
- AB_PU: reads/writes between AB and registers in PUs
- RUN_PU: computation in PUs

Left bars: without ComPEND
Right bars: with ComPEND

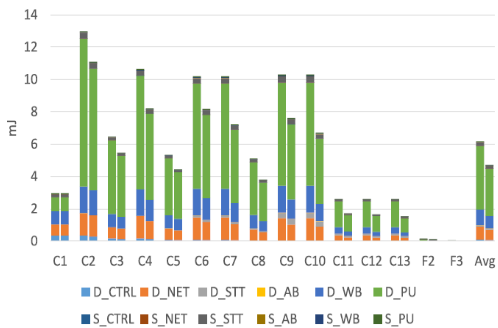
< for VGG-16 layers >

28

Zero Skipping

◆ Energy (dynamic & static) consumption

- ComPEND reduces energy by 23.50% on average



- D/S_CTRL: global controller
- D/S_NET: provider network
- D/S_STT: STT-RAM.
- D/S_AB: activation buffers
- D/S_WB: weight buffer
- D/S_PU: processing units

Left bars: without ComPEND
Right bars: with ComPEND

< for VGG-16 layers >

Zero Skipping

◆ Zero skipping for training

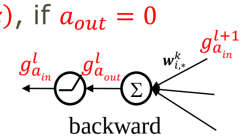
- Skipping gradient computation on zero activation
- $a_{out} = f_{\mathrm{ReLU}}(a_{in}) = \begin{cases} a_{in}, & a_{in} > 0 \\ 0, & a_{in} \le 0 \end{cases}$

  $\rightarrow a_{out} = 0 \Rightarrow \frac{\partial a_{out}}{\partial a_{in}} = 0$

- In backward propagation $\frac{\partial E}{\partial a_{in}} = \frac{\partial E}{\partial a_{out}} \cdot \cancelto{0}{\frac{\partial a_{out}}{\partial a_{in}}}$

  →No need to compute gradient $g^l_{a_{out}}(x,y,z)$, if $a_{out} = 0$

  $$g^l_{a_{out}}(x,y,z) = \sum_{i=0}^{F_x-1}\sum_{j=0}^{F_y-1}\sum_{k=0}^{F_n-1} g_a^{l+1}(x+i, y+j, k) \times w^l(i,j,z,k)$$



backward propagation

  →saves $F_x \times F_y \times F_n$ MAC operations

Zero Skipping

◆ **Zero skipping for training**

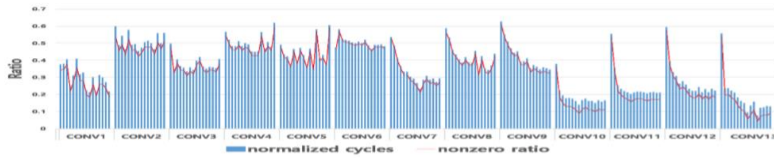


Figure 8: Performance improvement on VGG-16.

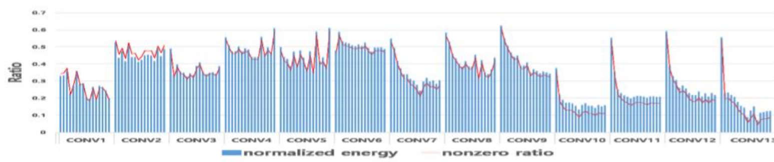


Figure 9: DRAM energy reduction on VGG-16.

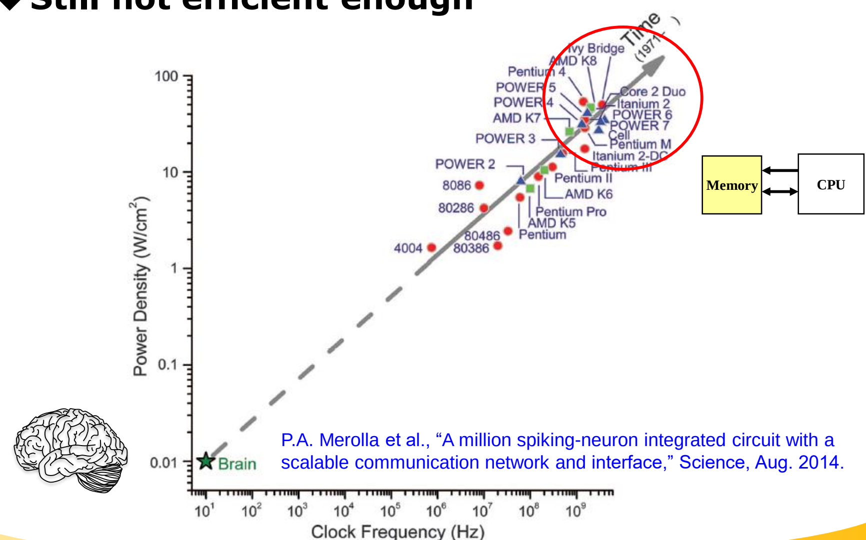
Low-Precision Computing

◆ **Inference**
- 8-bit
  - Google TPU 1
- Binary
  - Trade-off between precision and accuracy

◆ **Training**
- Dynamic fixed-point
  - Bengio
  - DAL
- 16-bit FP
  - NVDIA: half-precision FP, 1-5-10, scaling
  - Google: bfloat, 1-8-7
- 8-bit
  - IBM: stochastic rounding
  - Intel: range batch normalization + bifurcation

Computing in Analog

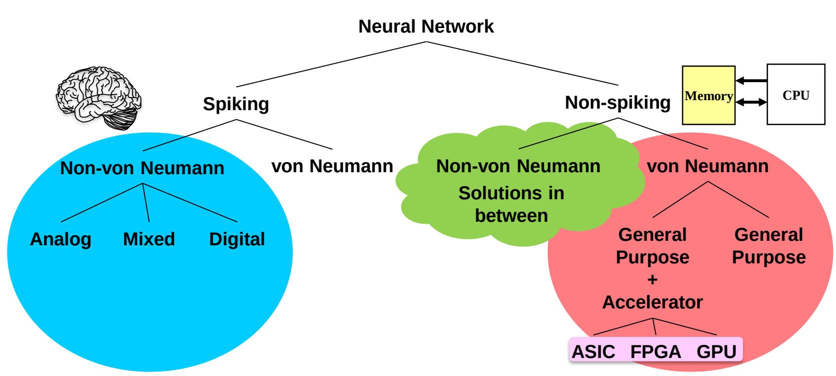
◆ **Still not efficient enough**



P.A. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," Science, Aug. 2014.

Computing in Analog

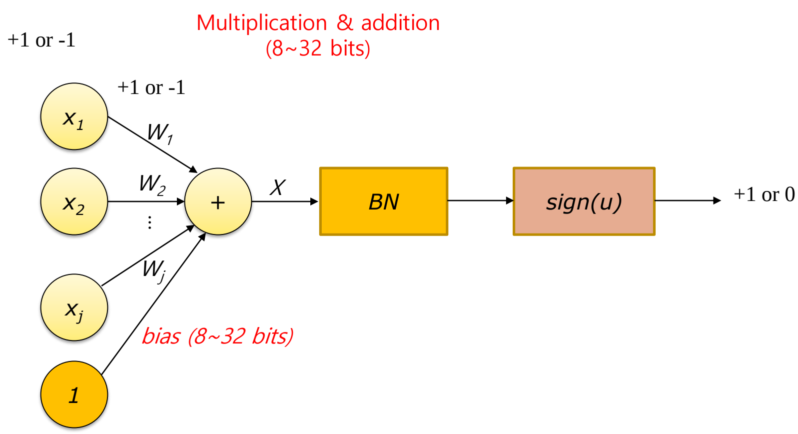
◆ **Various ways of implementing neural networks**



Computing in Analog

◆ **BNN (Binarized Neural Network)**

- MAC operations in analog



Computing in Analog

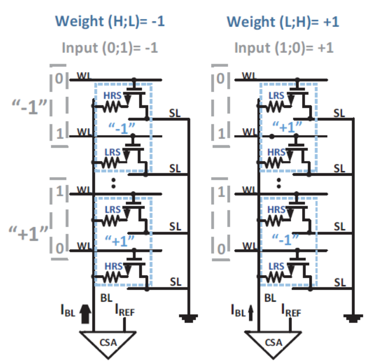
◆ **Accuracy**

Network	W	Act	CIFAR-10 ACC (4 conv, 2 fc)	CIFAR-10 ACC (6 conv, 3 fc)
DNN (baseline)	Float32	Float32	85.60%	91.11%
BWN	1-bit (1, -1)	Float32	84.21% (-1.39%p)	90.64% (-0.47%p)
BNN	1-bit (1, -1)	1-bit (1, -1)	77.13% (-8.47%p)	88.89% (-2.22%p)

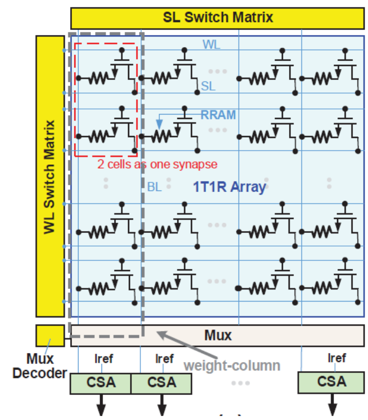
Computing in Analog

◆ Mixed-signal implementation of BNN

XNOR: Neuron (-1, +1); Weight (-1, +1)



Parallel XNOR-RRAM

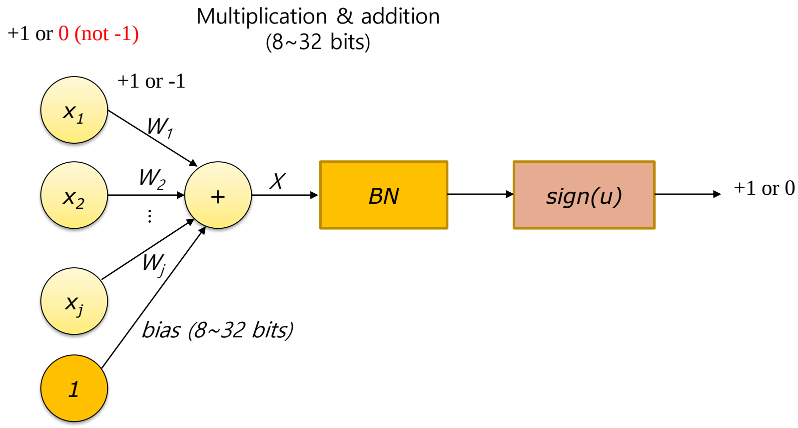


X. Sun et al., "XNOR-RRAM: A Scalable and Parallel Resistive Synaptic Architecture for Binary Neural Networks," DATE 2018.

Computing in Analog

◆ BNN (Binarized Neural Network)

- MAC operations in analog



Computing in Analog

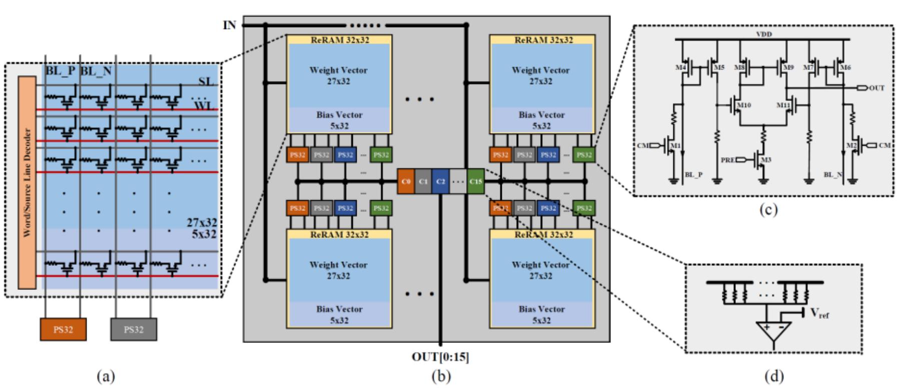
◆ Binarized Spiking Neural Network (BSNN)

Network	W	Act	CIFAR-10 ACC (4 conv, 2 fc)	CIFAR-10 ACC (6 conv, 3 fc)
DNN (baseline)	Float32	Float32	85.60%	91.11%
BWN	1-bit (1, -1)	Float32	84.21% (-1.39%p)	90.64% (-0.47%p)
BNN	1-bit (1, -1)	1-bit (1, -1)	77.13% (-8.47%p)	88.89% (-2.22%p)
SNN	Float32	1-bit (1, 0)	78.73% (-6.87%p)	88.01% (-3.1%p)
BSNN	1-bit (1, -1)	1-bit (1, 0)	77.25% (-8.35%p)	87.85% (-3.26%p)

Computing in Analog

◆ ReRAM-based implementation of BNN

- 27x32 binary weights + 5x32 biases in a tile
- Array of tiles



Computing in Analog

◆ Comparison

Implementation	JSSC '17	ASP-DAC '17	ISLPED '17	DATE '18	ISSCC '18	Ours
Network	CNN	CNN	CNN	MVM	CNN	MLP
# Parameters	0.26M	1.26M	14.03M	0.07M	1.88M	0.53M
Technology	65nm	45nm	40nm	65nm	28nm	32nm
						2.1ns
Ar						0.15
Po						519.6
Er						1.09
Energy efficiency (TOPS/W)	**0.048**	**0.962***	**126**	**141**	**532**	**970**

- Human brain
  - Power consumption: ~10W
  - Number of synapses: ~10^{15}
  - Firing rate of one synapse: ~10 spikes/sec
  - Max. power efficiency: $10^{15} \times 10 / 10$ = 1 POPS/W

* Data calculated based on the numbers in the paper

How good is ~1 POPS/W?

Conclusion

◆ For an efficient neural processing

- Network reduction
- Zero skipping
- Low-precision computing
- Computing in analog
- …

◆ Many new areas to be explored

- Exploiting NVMs and in-memory-computing
- Exploiting information in timing
- Spiking neural network
- New training algorithm for efficiency
- …