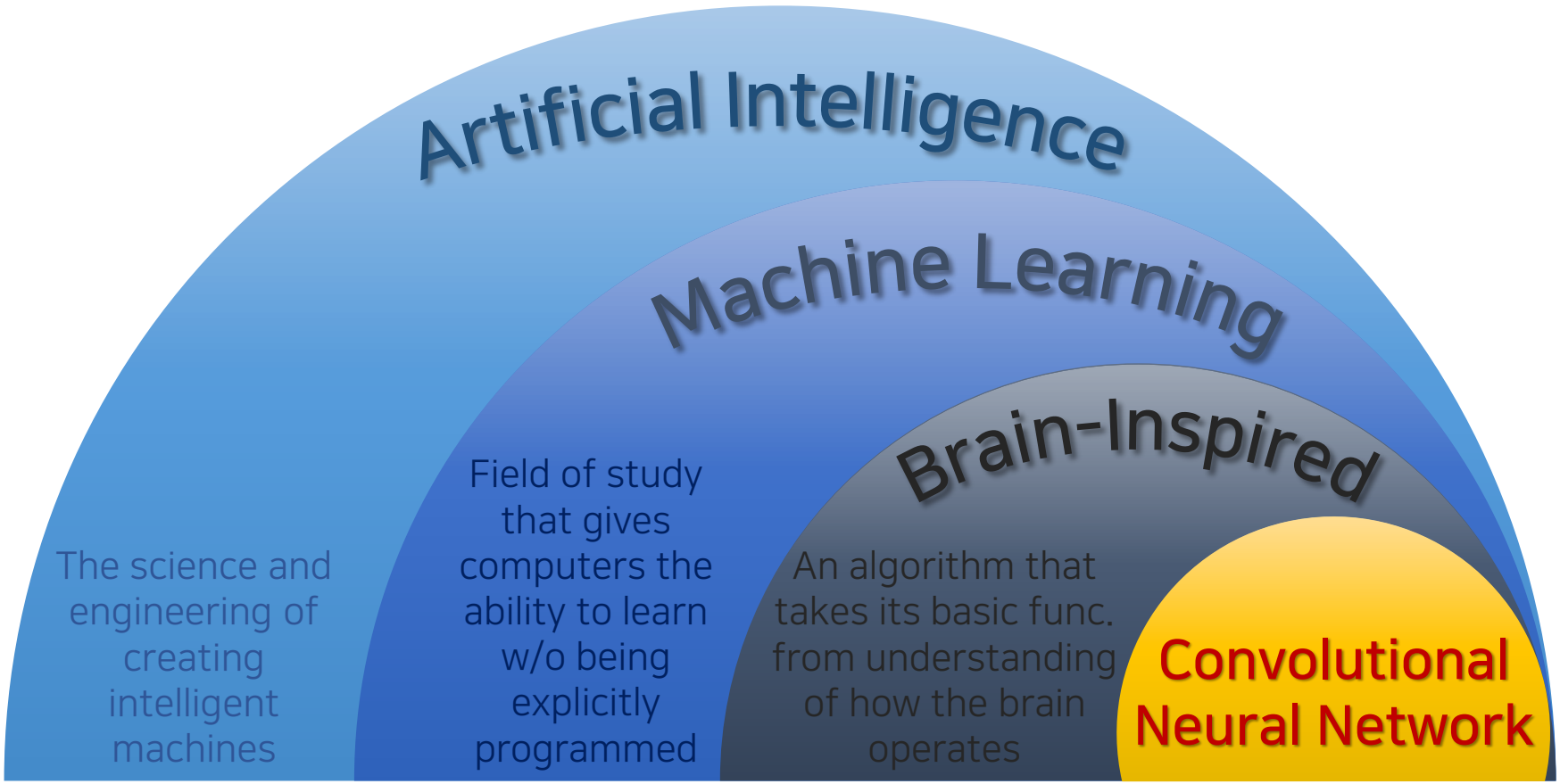
Outline

- 인공지능 관련 동향
- 딥러닝의 기초
  - 용어 및 분류(Classification)/회기(Regression) 모델
  - 퍼셉트론 (Perceptron) 및 MNIST Example
- 학습 및 추론 알고리즘의 이해
  - Backpropagation
  - Challenge of Learning
  - Optimizer of Learning
- **딥러닝 가속기**
  - CNN 및 인공 신경망 가속기 연구 동향

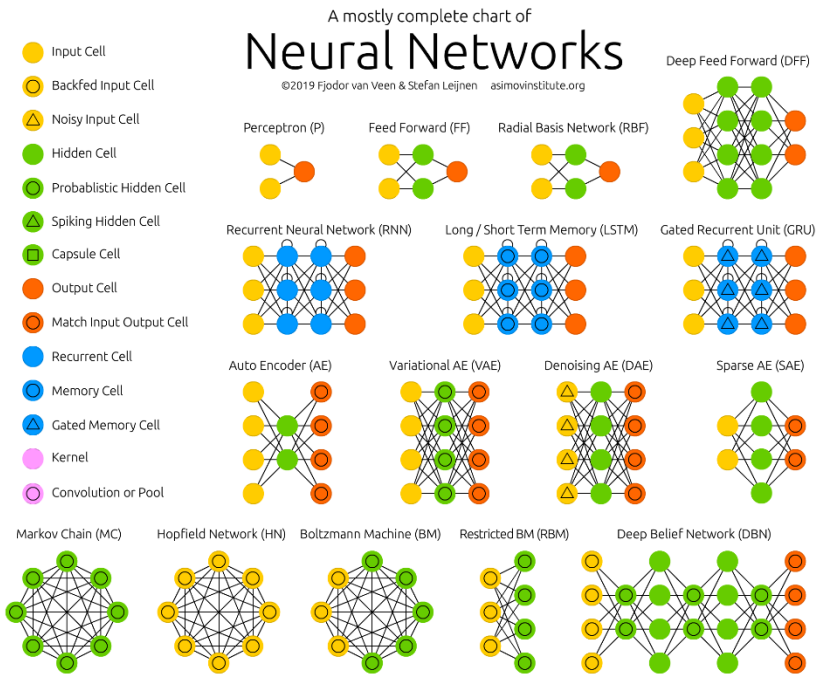
AI & Neural Network

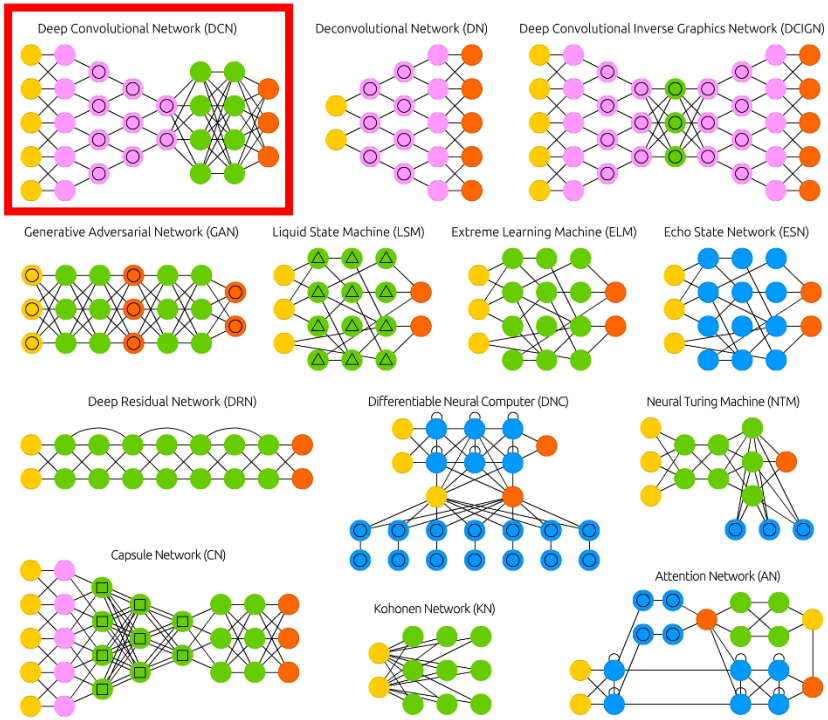


Various Neural Networks

❑ Convolutional Neural Network (CNN)

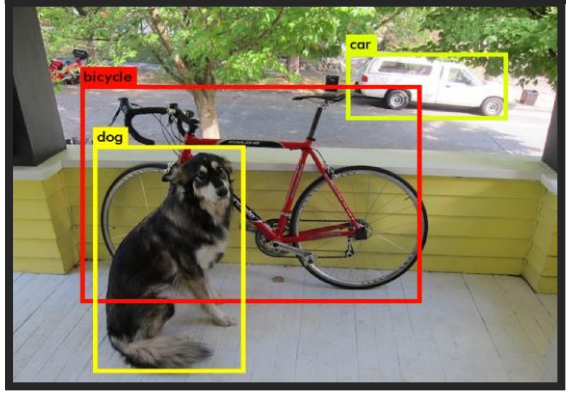
- One of the most popular neural network





CNN Applications

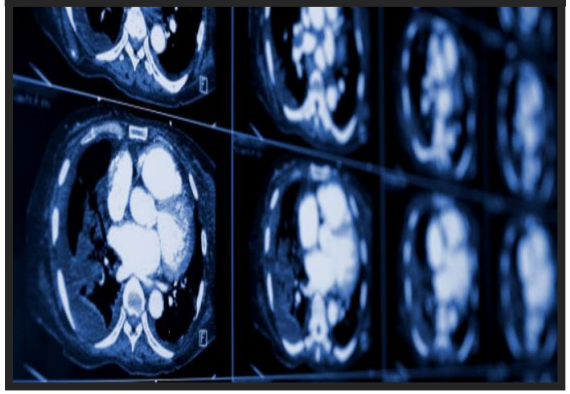
Image Process



Autonomous Machines

Security & Defense

Medical

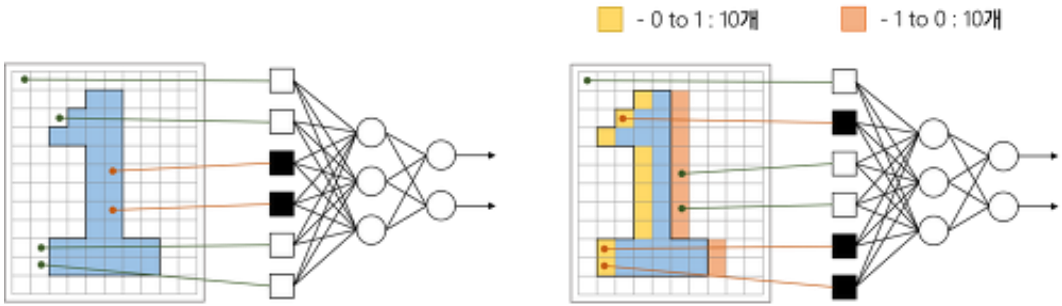


Game

Convolutional Neural Network

❑ MLP(Multi-Layer Perceptron)의 문제점

- Whole Feature에 대해서 인식



한 칸만 움직였는데,
변하는 값이 20개

- 부분적 Feature에 대해 인식하게 할 순 없을까?

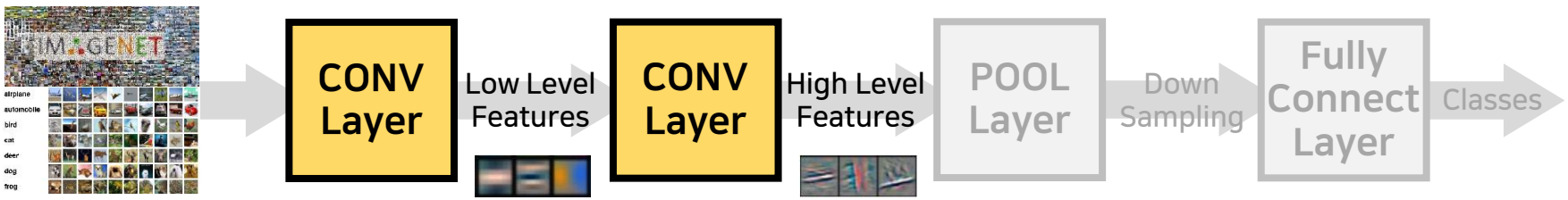
1단계 : 가로, 동그라미, 세모, 부드러움

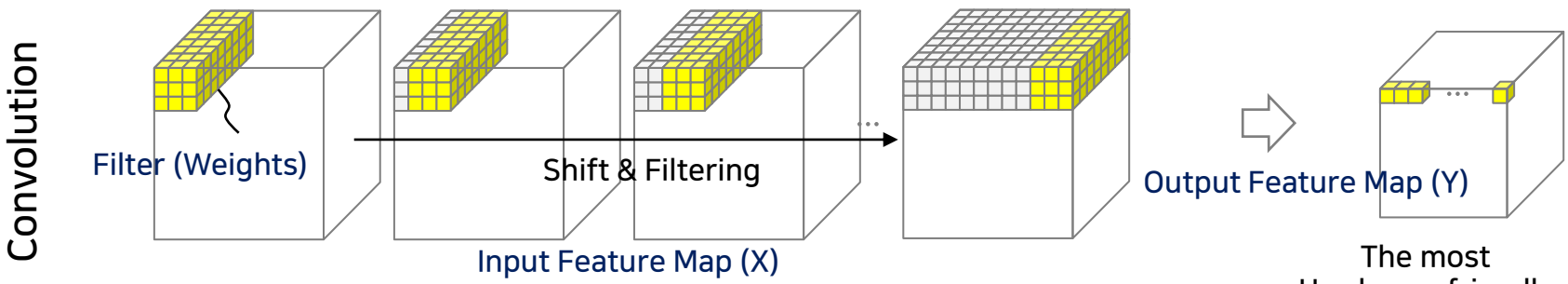
2단계 :눈, 코, 귀, 발

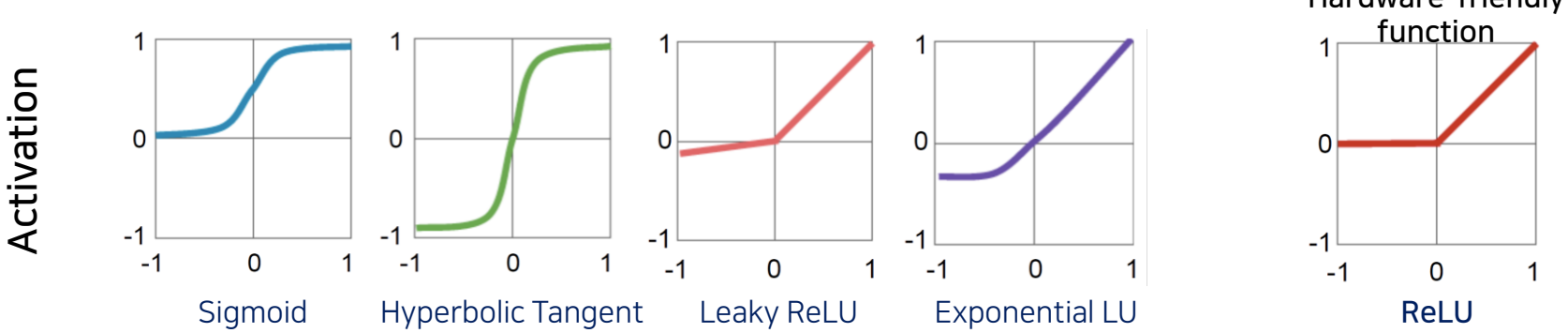
3단계 :고양이!

CNN!

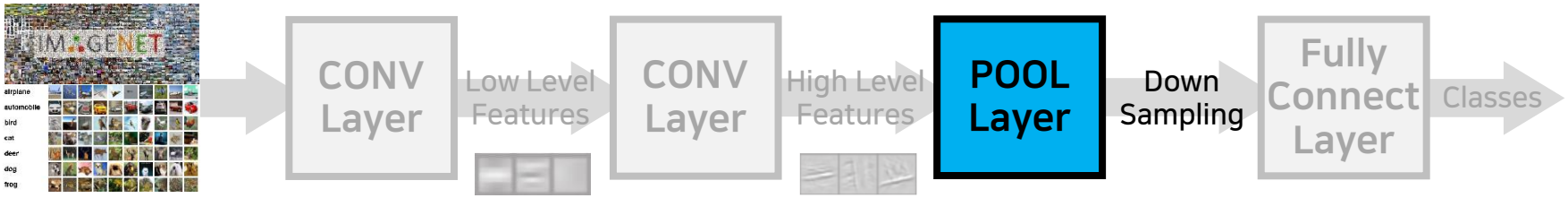
Convolutional Neural Network

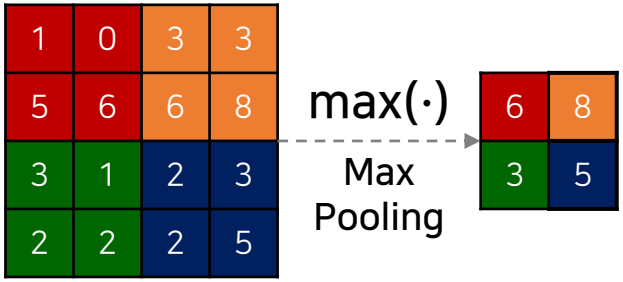


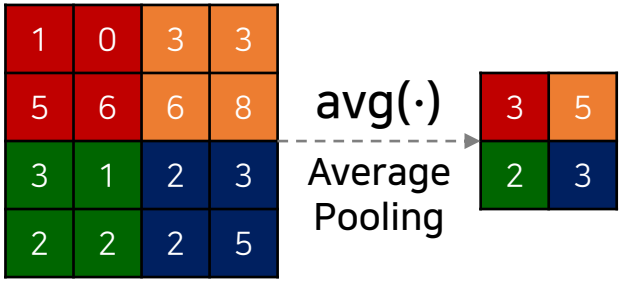




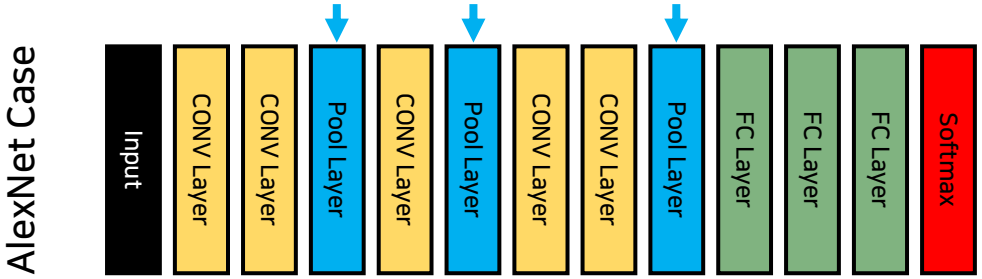
Convolutional Neural Network





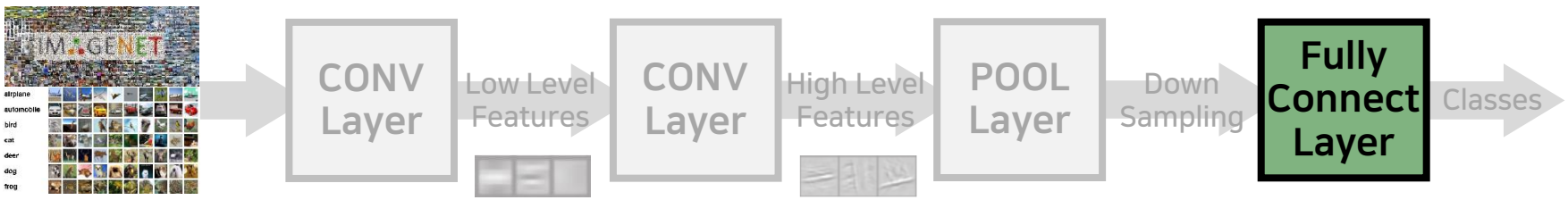


Location of Pooling Layers

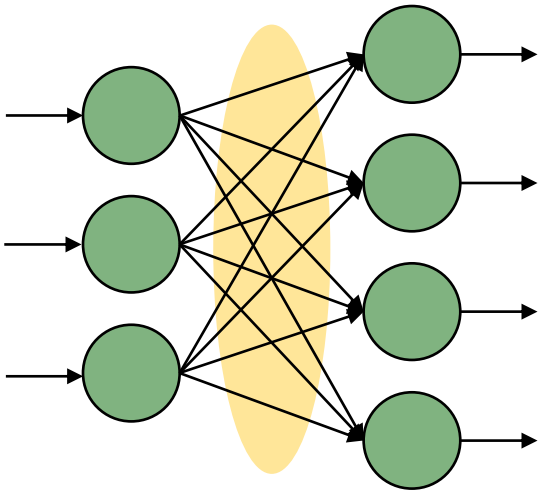


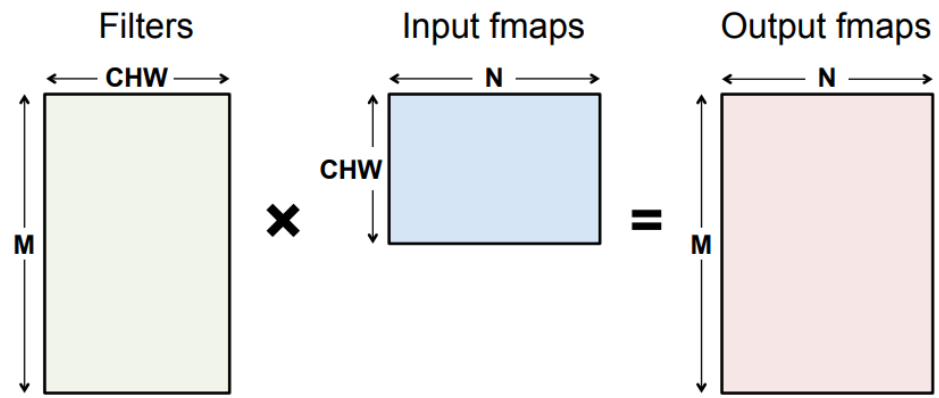
- Reduce resolution of each channel independently
- Increase translation-invariance and noise-resilience

Convolutional Neural Network



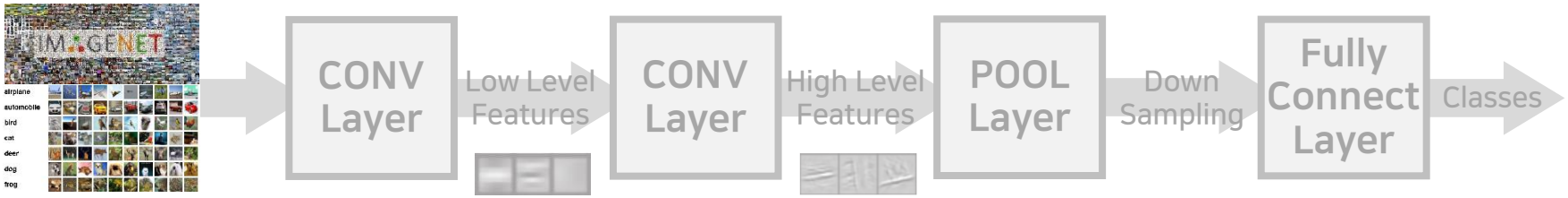
Fully-connected





- Fully-connected layer account for more than 90% of total number of parameters, dominating memory and energy
- Simple matrix multiplication

And Others …

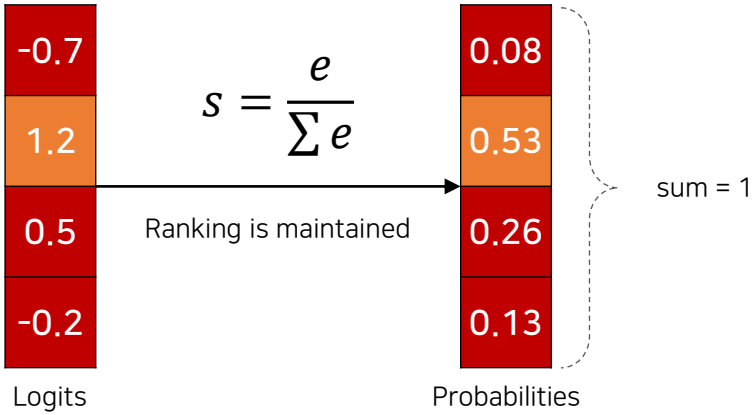


Normalization

[Sergey Ioffe et al., ICML 2015]

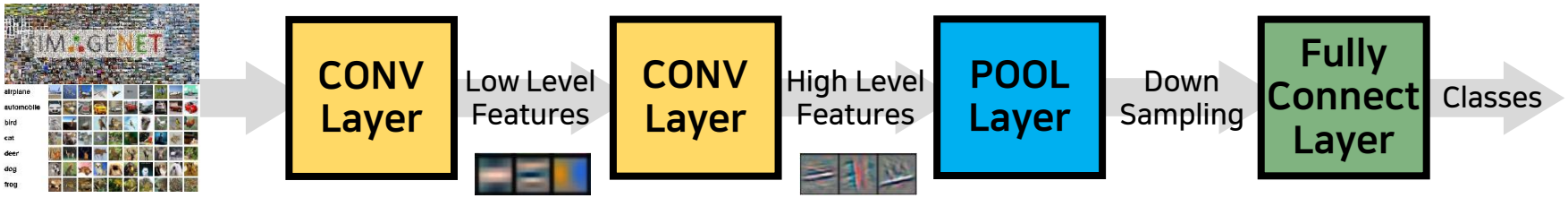
- Pre-processing to balance between the training and inference (accuracy highly relies on these procedure)

Softmax



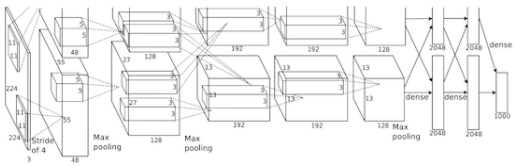
- Not essential when the difference between each class is not seriously importance

Advantage of CNN



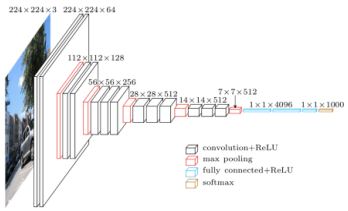
- ❑ **MLP 의 한계점 극복**
  - MLP는 1차원 입력 데이터만 받음
    - 이미지는 (RGB) 3 차원 데이터
- ❑ **파라미터 (Weight & Bias)를 줄이고 오버피팅 방지**
- ❑ **풀링 레이어 등으로 노이즈에 내성**
- ❑ **기존 Training 알고리즘 유지**

Various CNN Configurations



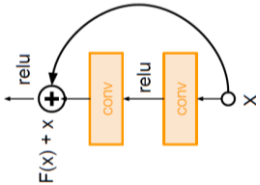
AlexNet [1]

Layer	Filter Size	# Filters	# Channels	Stride
1	11X11	96	3	4
2	5X5	256	96	1
3	3X3	384	256	1
4	3X3	384	384	1
5	3X3	256	384	1
6	Fully-Connected Layer			
7	Fully-Connected Layer			
8	Fully-Connected Layer			



VGG-16 [2]

Layer	Filter Size	# Filters	# Channels	Stride
1	3X3	64	3	1
2	3X3	64	64	1
3	3X3	128	64	1
4	3X3	128	128	1
5	3X3	256	128	1
6	3X3	256	256	1
7	3X3	256	256	1
8	3X3	512	256	1
9	3X3	512	512	1
10	3X3	512	512	1
11	3X3	512	512	1
12	3X3	512	512	1
13	3X3	512	512	1
14	Fully-Connected Layer			
15	Fully-Connected Layer			
16	Fully-Connected Layer			



ResNet-50 [3]

Layer	Filter Size	# Filters	# Channels	Stride
1	7X7	64	3	2
2	1X1	256	64	1
3	1X1	64	64	1
4	3X3	64	64	1
5	1X1	256	64	1
	Addition Layer			
6	1X1	64	64	1
7	3X3	64	64	1
8	1X1	256	64	1
	Addition Layer			
9	1X1	64	256	1
10	3X3	64	64	1
11	1X1	256	64	1
	Addition Layer			
12	1X1	512	64	2
13	1X1	128	64	2
14	3X3	128	128	1
15	1X1	512	128	1
	Addition Layer			
16	1X1	128	512	1
17	3X3	128	128	1
18	1X1	512	128	1
	Addition Layer			
19	1X1	128	512	1

⊕ ⊕ ⊕ ⊕

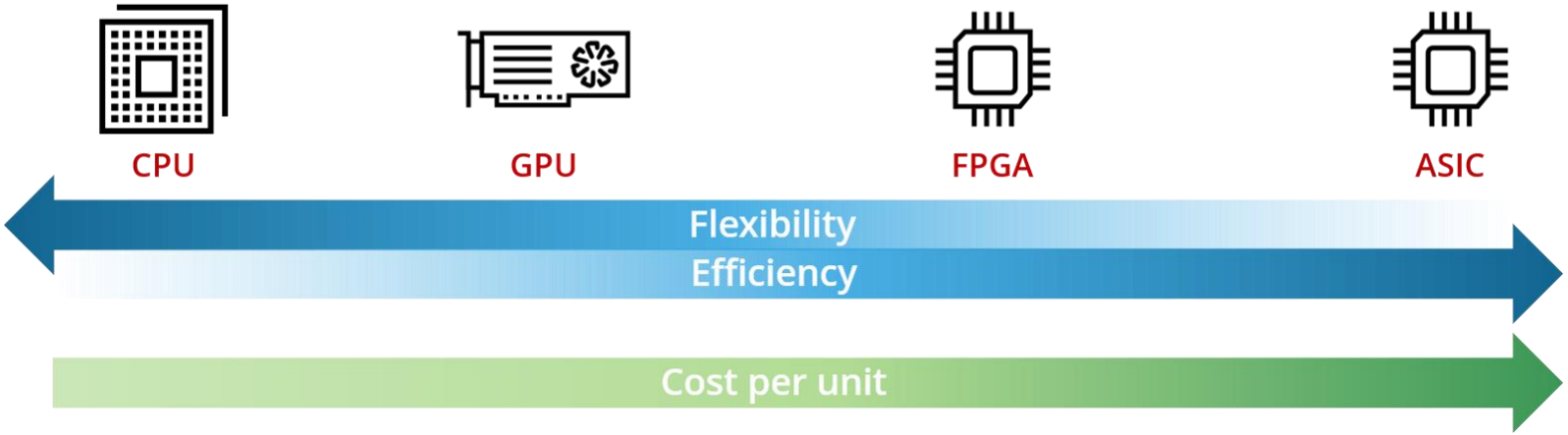
⋮

[1] [Krizhevsky et al. NIPS 2012]
[2] [Simonyan and Zisserman, ICLR 2015]
[3] [He et al., CVPR 2016]

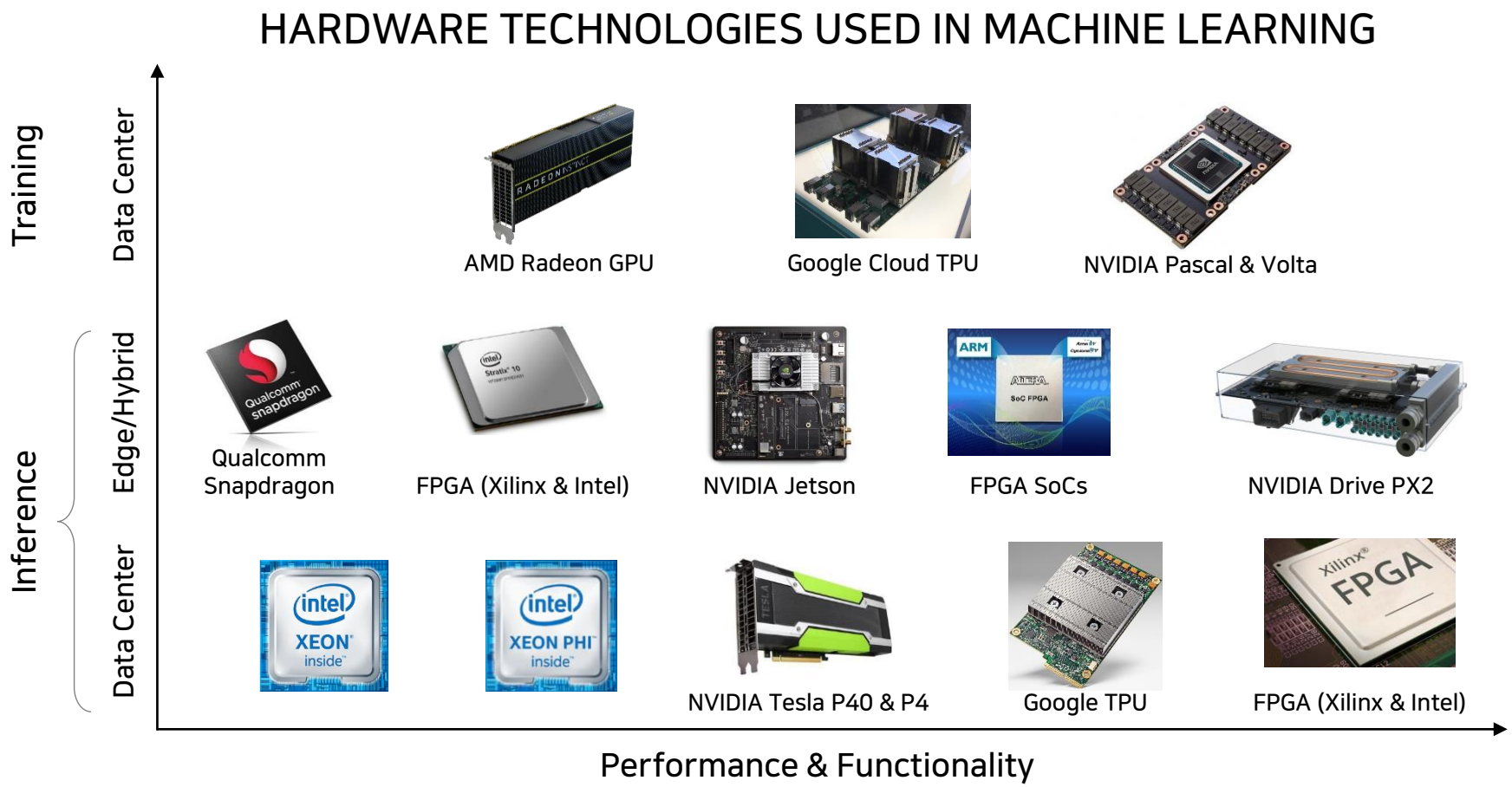
AI Accelerator

❑ Emergence of dedicated AI accelerator ASICs

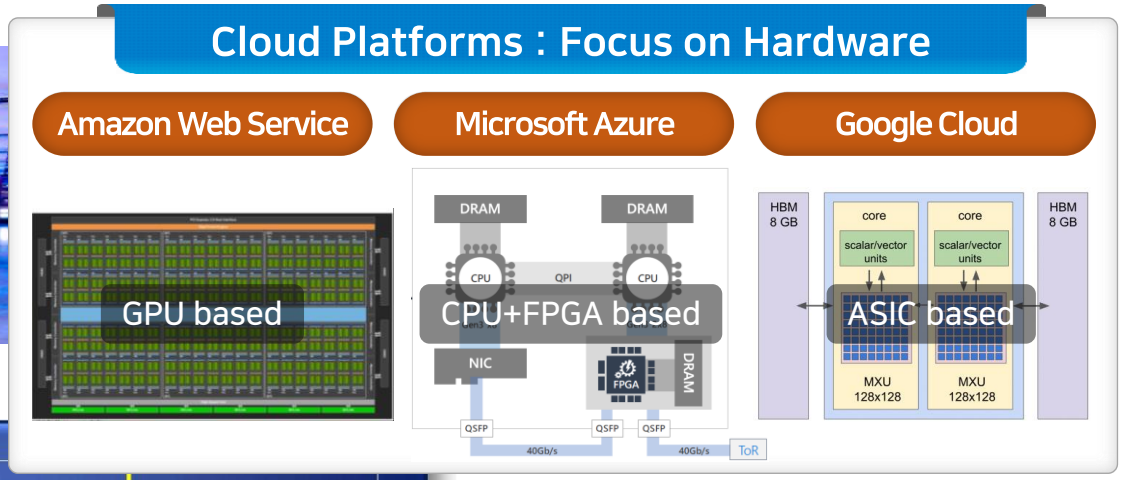
While GPUs and FPGAs perform far better than CPUs for AI related tasks, a factor of up to 10 in efficiency may be gained with a more specific design, via an application-specific integrated circuit (ASIC). These accelerators employ strategies such as optimized memory use and the use of lower precision arithmetic to accelerate calculation and increase throughput of computation. Some adopted low-precision floating-point formats used AI acceleration are half-precision and the bfloat16 floating-point format.

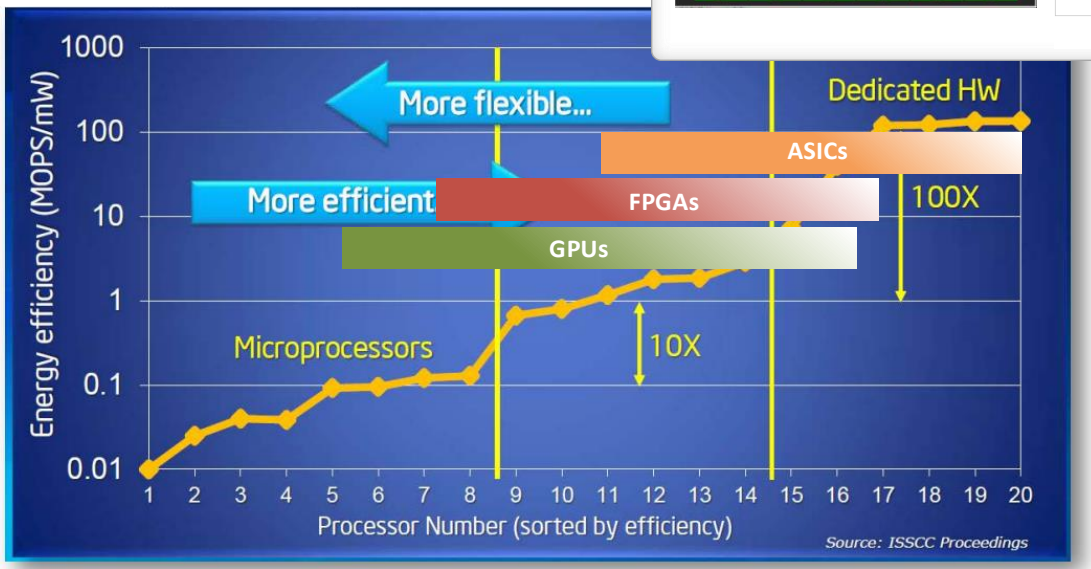


Hardware Technologies



Deep CNN on "Cloud Platforms"

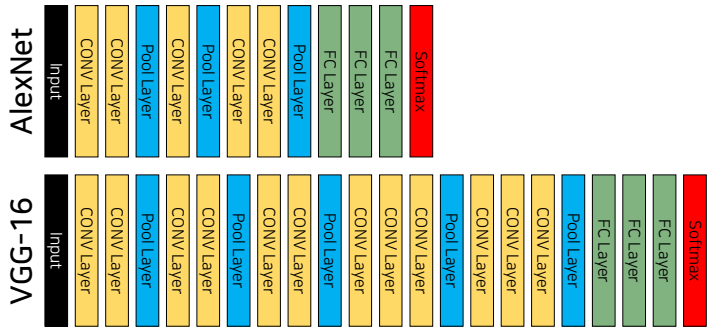




Source: Bob Broderson, Berkeley Wireless group

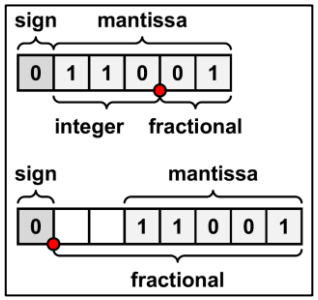
- Accelerator is more efficient in terms of power and energy consumption
- Trade off between Energy Efficiency and Flexibility

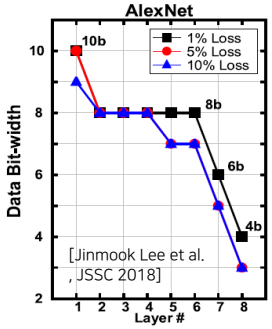
Need Reconf. Accelerator



In different Network

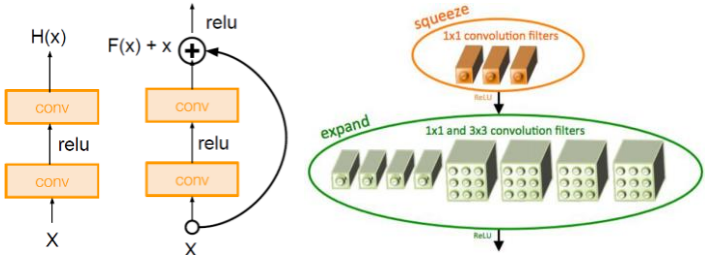
- Different number of layers
- Different number of filters / channels





In different Quantization

- Different bit-width of different layers

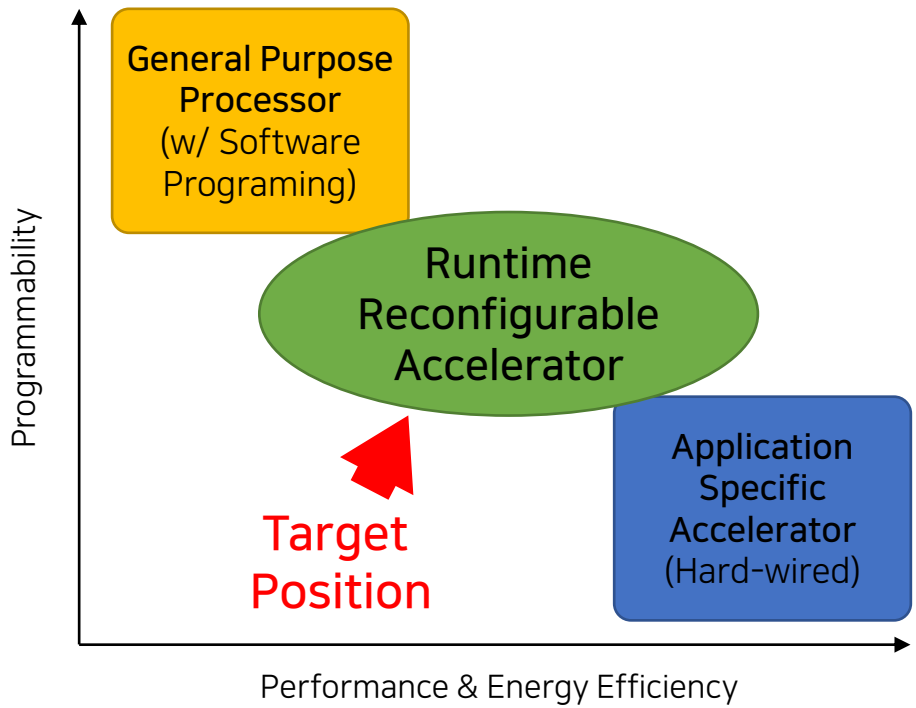


In different Architecture

- Different algorithmic structures

Reconfig. Vs Energy Efficiency

Dynamically Reconfigurable DNN Accelerator

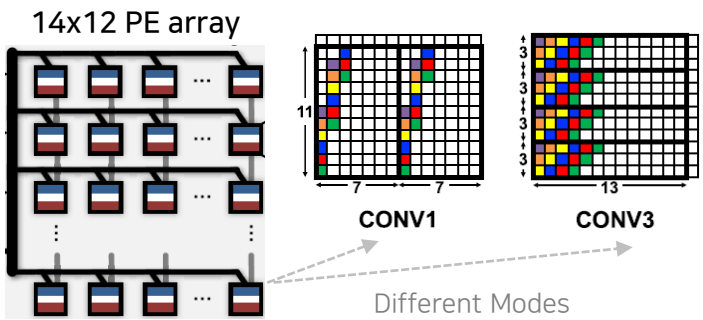


High-level Reconfiguration

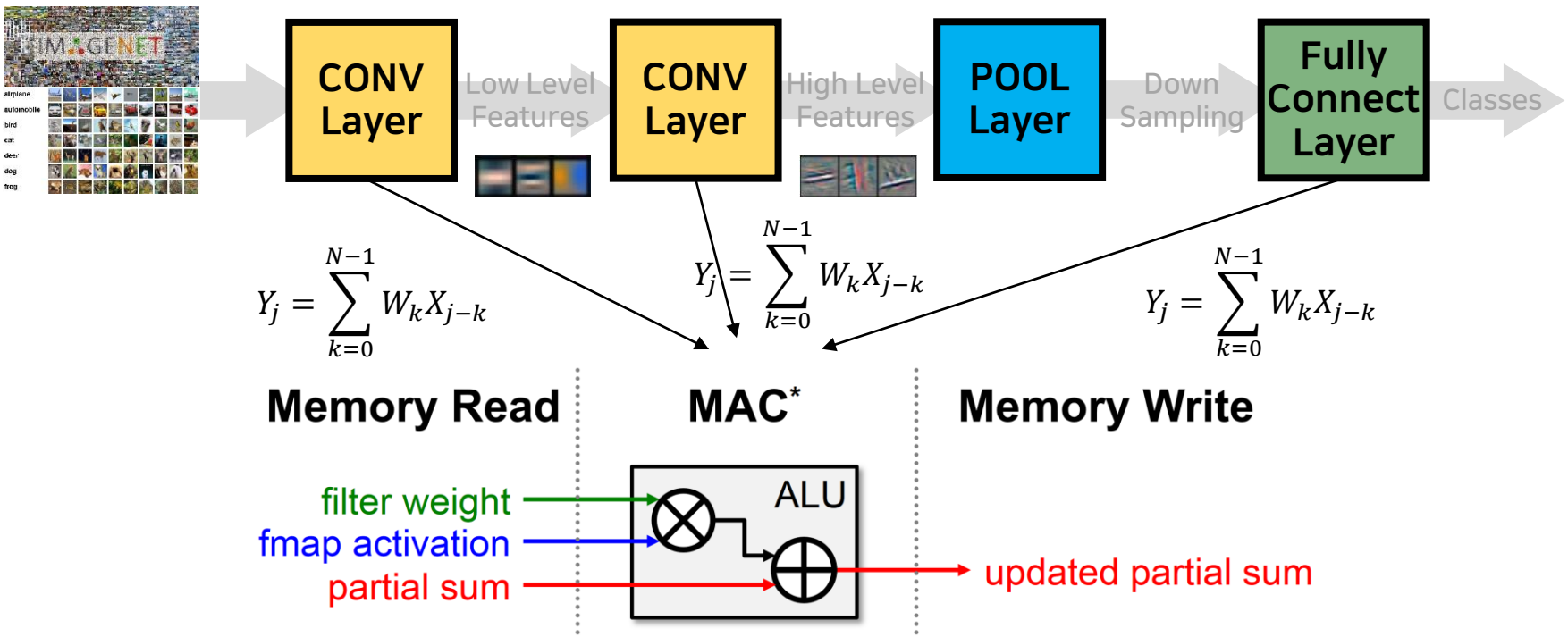
```
VLOAD   $3, $0, #100        // load input vector from address (100)
MLOAD   $4, $2, #300        // load weight matrix from address (300)
MMV     $7, $1, $4, $3, $0  // Wx
VAV     $8, $1, $7, $5      // tmp=Wx+b
VEXP    $9, $1, $8          // exp(tmp)
VAS     $10, $1, $9, #1     // 1+exp(tmp)
VDV     $6, $1, $9, $10     // y=exp(tmp)/(1+exp(tmp))
VSTORE  $6, $1, #200        // store output vector to address (200)
```

Use Instruction Set Architecture

Low-level Reconfiguration

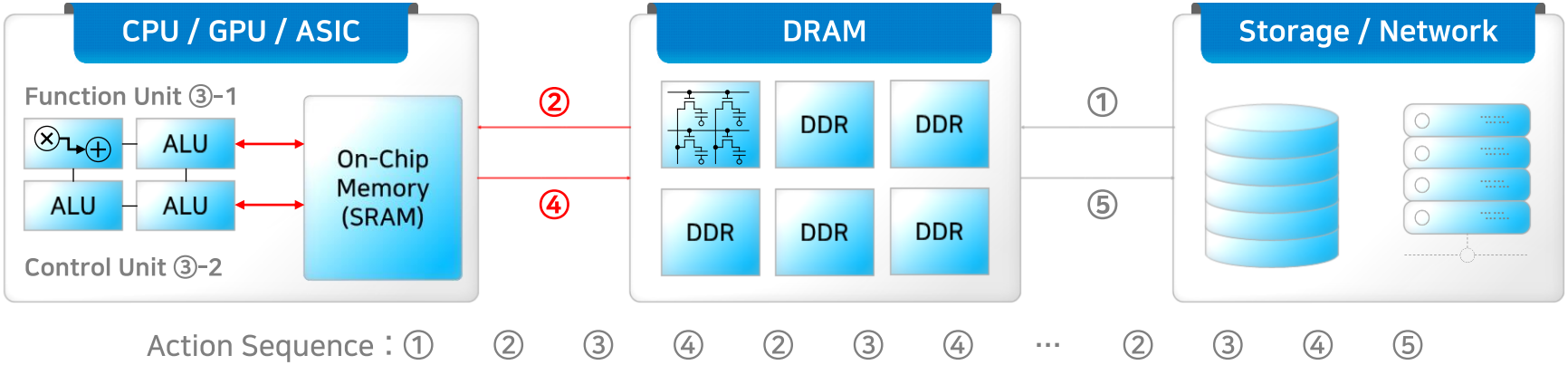


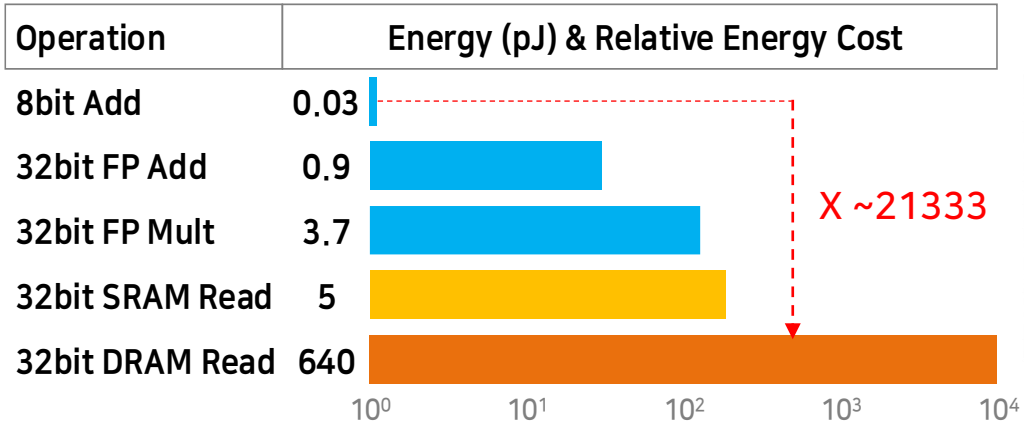
Main Operation in CNN



Architecture	Weight Size	Ifmap Size	# Multiply-Adds	Top-1 Accuracy
AlexNet	238 MB	1.6 MB	724 M	57.10 %
VGG-16	528 MB	34.8 MB	15.5 B	70.50 %
ResNet-50	99 MB	37.5 MB	3.9 MB	75.20 %

Data-Centric CNN



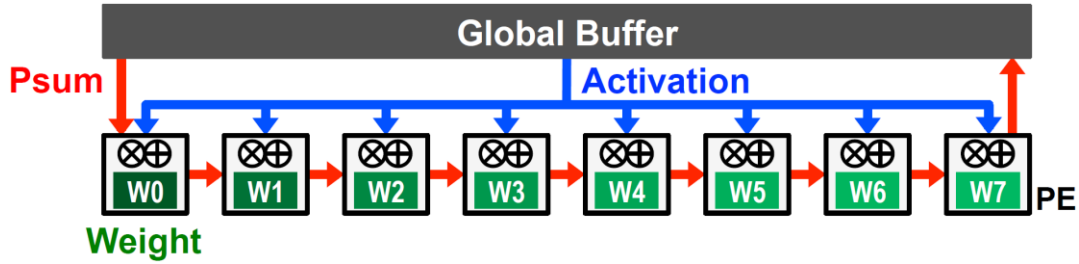


[M Horowitz et al., ISSCC 2014]

Accelerator Design

- Maximize Data Reuse
- Reduction: Computation Size
- Reduction: Computation Number
- Processing-in-memory

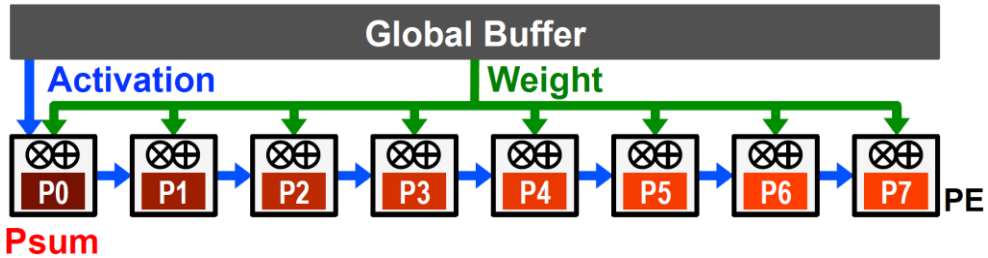
Maximize Data Reuse : Data Flow



[nn-X (NeuFlow), *CVPRW* 2014] [Park, *ISSCC* 2015] [ISAAC, *ISCA* 2016] [PRIME, *ISCA* 2016]

Weight Stationary

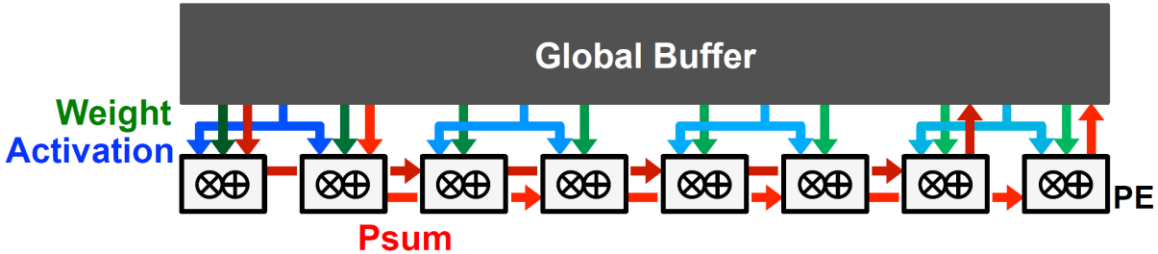
- Maximize weight reuse
- Broadcast activation
- Accumulate pSUMs spatially



[Peemen, *ICCD* 2013] [ShiDianNao, *ISCA* 2015] [Gupta, *ICML* 2015] [Moons, *VLSI* 2016]

Output Stationary

- Maximize pSUM reuse
- Broadcast weight
- Reuse activation spatially



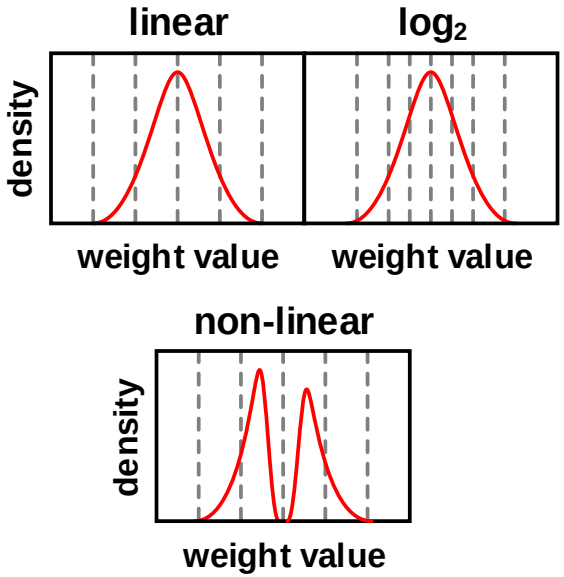
[DianNao, *ASPLOS* 2014] [DaDianNao, *MICRO* 2014] [Zhang, *FPGA* 2015] [TPU, *ISCA* 2017]

No Local Reuse

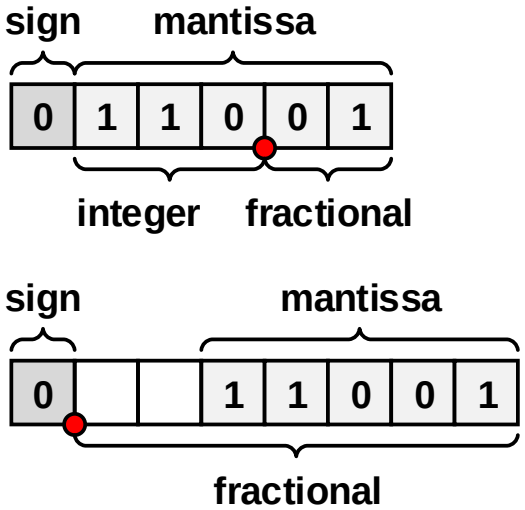
- Use a large global buffer
- Reduce DRAM access
- Multicast activation & weight
- Accumulate pSUMs spatially

Reduction: Computation Size

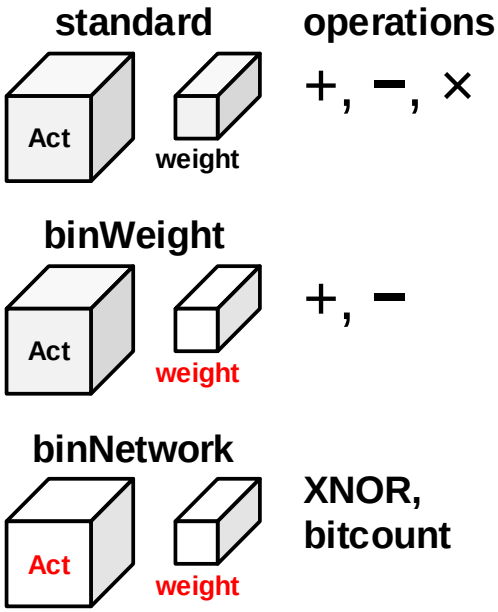
Non-linear Quantization



Dynamic Fixed Point



Binary Neural Network

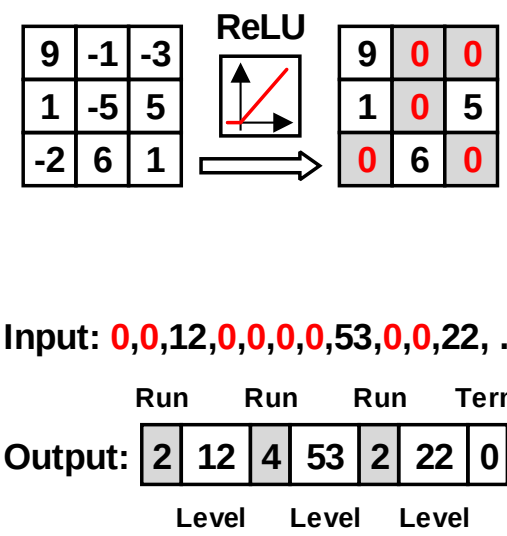


Processing-in-Memory

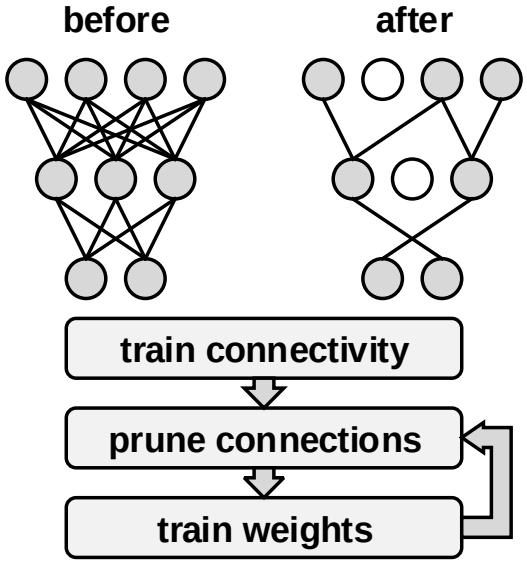
- Directly reduced the memory & PEs
- Trade-off : Bit-width ↔ Accuracy

Reduction: Computation Number

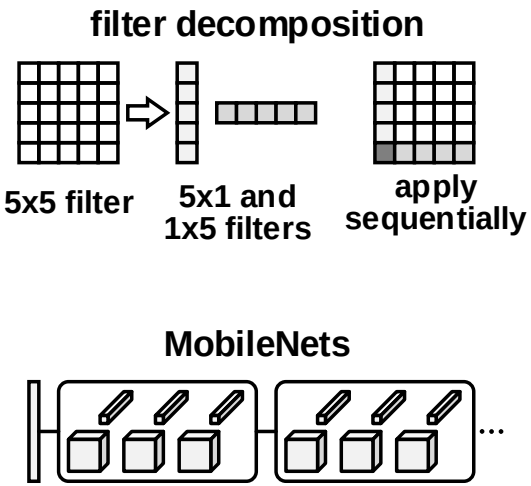
Activation Sparsity



Network Pruning

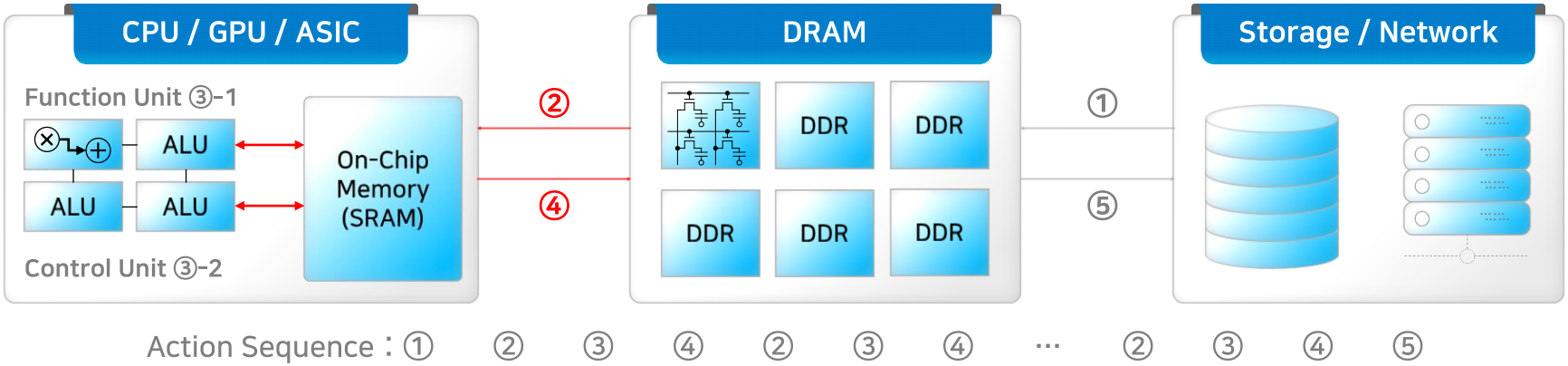


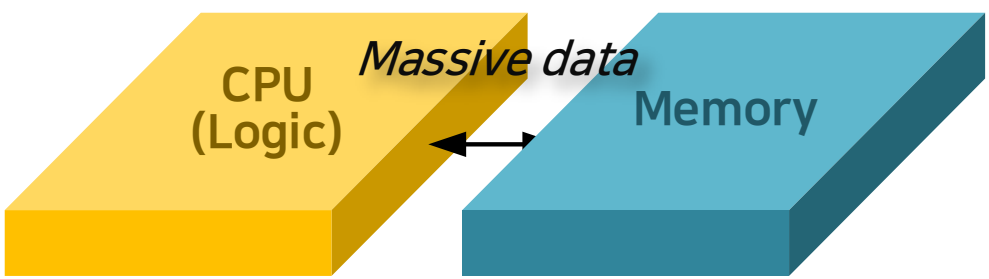
Compact Architecture



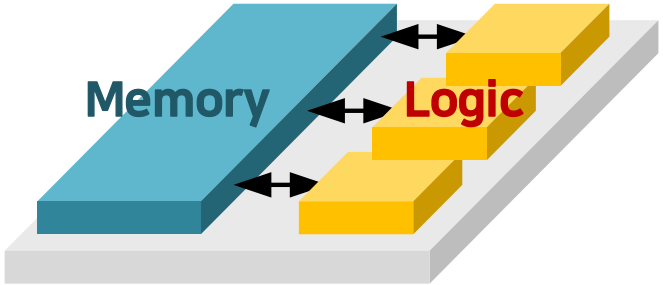
- Reduced Activation → Gating or skipping cycle & memory access
- Accuracy loss depending on the techniques

Processing in Memory





von Neumann architecture

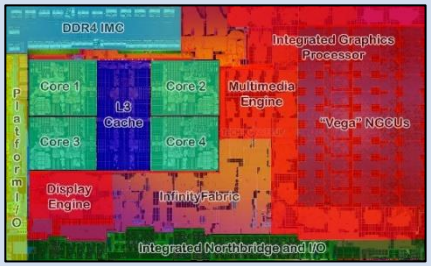
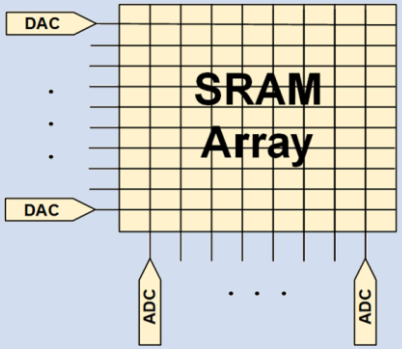
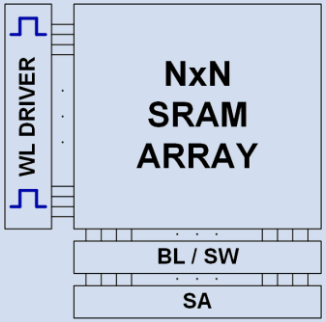


Processing-in-memory

- PIM: A technique that performs simple logic within a memory device to reduce the amount of data being passed to the processor.

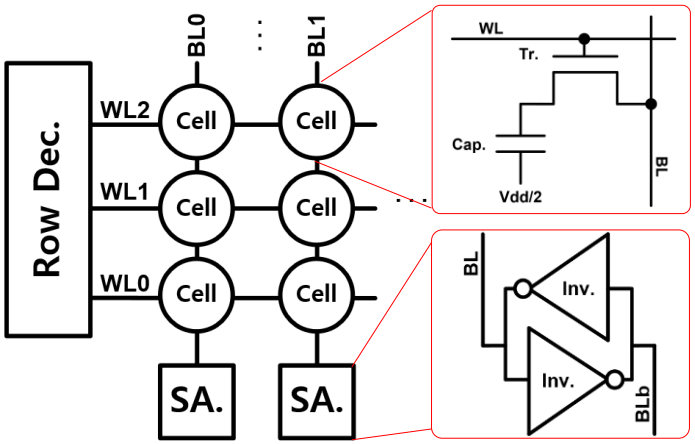
Processing in Memory

❑ Comparison of PIM

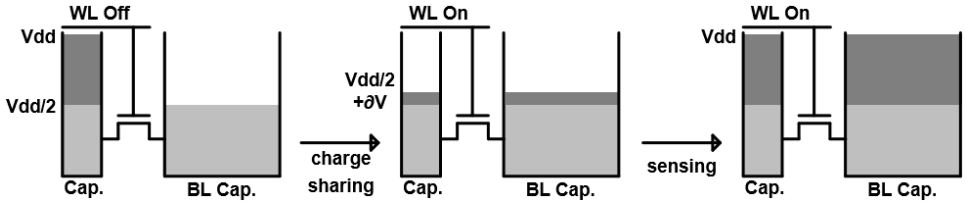
	Traditional Computing	Analog In-Memory Computing	Digital In-Memory Computing
Pro	• General • Accurate • Robust	• High Energy Efficiency • High Performance	• Moderate Energy Efficiency • High Performance • Robust
Con	• Poor Energy Efficiency • Bandwidth Problem	• Poor Precision • Weak to Noise/PVT Variation	• Limited Functionality • Poor Precision
Example	• CPU • GPU 		

Processing in Memory

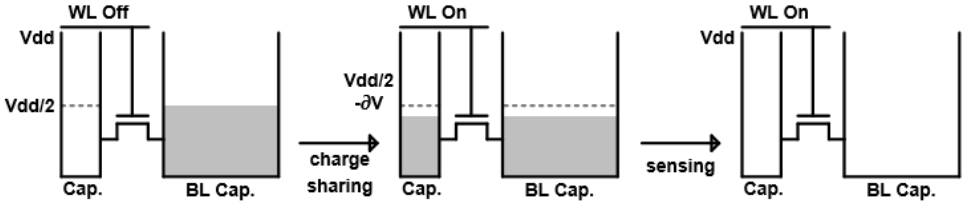
❑ Basic DRAM Operation : Read → Write-Back



DRAM Array Architecture



DRAM read operation: sensing "1"



DRAM read operation: sensing "0"

- DRAM consists of Cell(1T1C) array
- WL on → charge sharing btw. cell cap. and Bit-line(BL) cap. → sensing

Processing in Memory

❑ DRAM-based PIM : AND, OR Operation

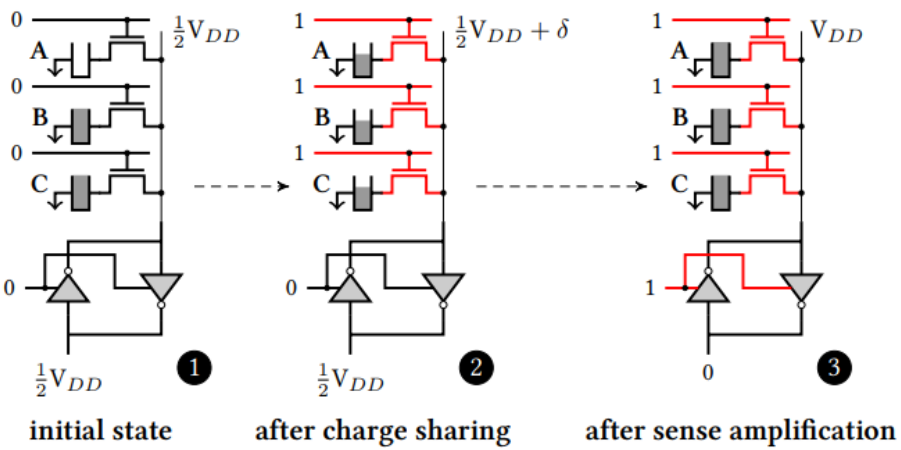


Figure 4: Triple-row activation

R = A *OR* B

A	B	C
1	1	1
1	0	1
0	1	1
0	0	1

Charge Sharing →

R
1
1
1
0

R = A *AND* B

A	B	C
1	1	0
1	0	0
0	1	0
0	0	0

Charge Sharing →

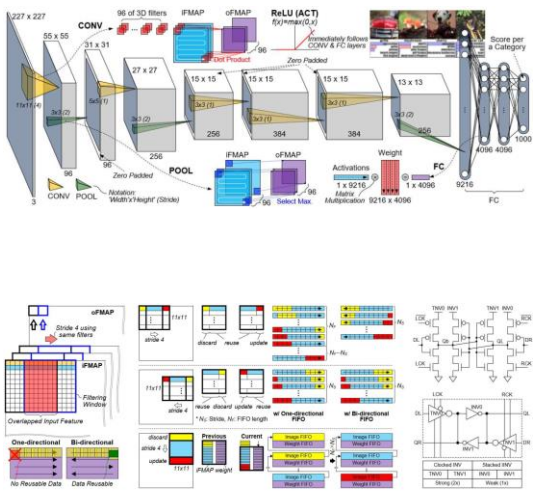
R
1
0
0
0

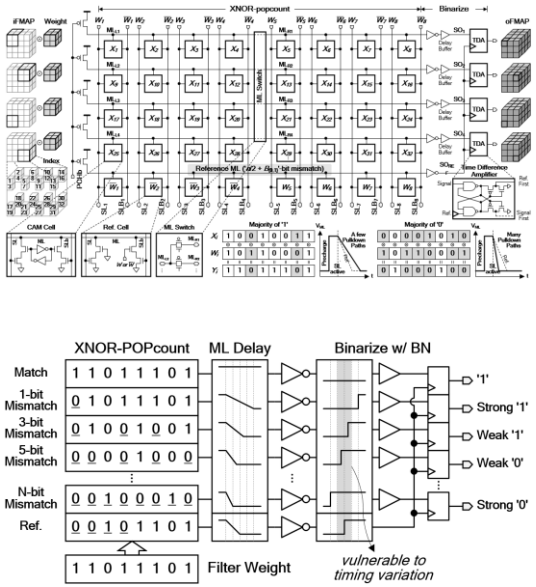
DRAM PIM: OR/AND Operation Table

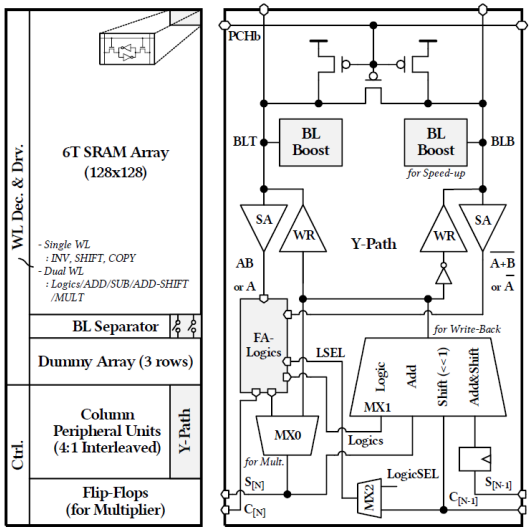
- AND/OR operation : 3 WLs on → charge sharing → sensing
- Majority function for A,B, and C input
- A & B when C = 1, A || B when C = 0

Processing in Memory

❑ Our Works





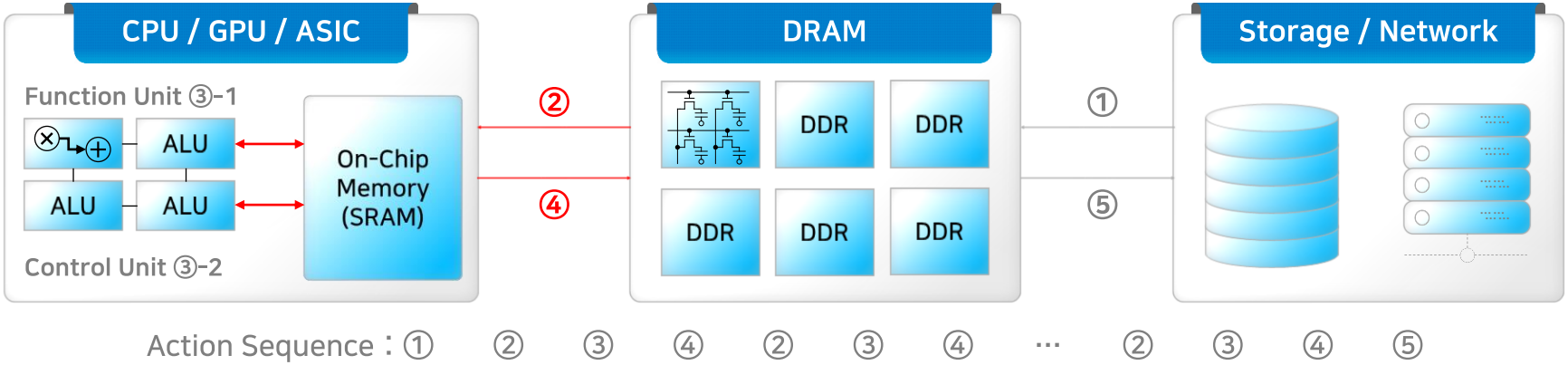


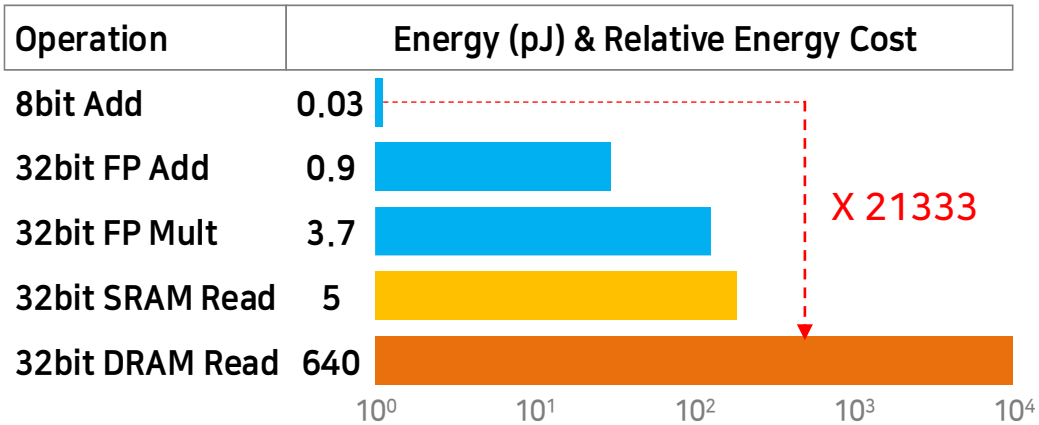
컨볼루션 인공신경망을 위한
양방향 선입선출 메모리 개발:
IEEE Access (2018)

딥러닝 추론연산에 적합한
CAM-based PIM 개발:
IEEE/ACM Design Automation Conference (2018)

복잡한 연산 지원 가능한
저지연 SRAM-based PIM 개발:
IEEE/ACM Design Automation Conference (2020)

Data-Centric DNN





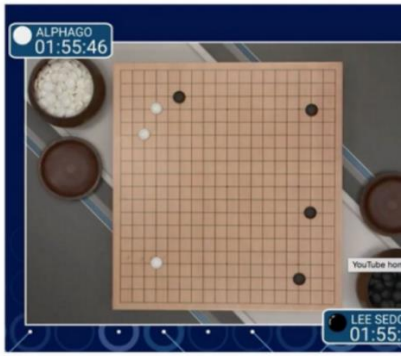
[M Horowitz et al., ISSCC 2014]

Accelerator Design

- Maximize Data Reuse
- Reduction: Computation Size
- Reduction: Computation Number
- Processing-in-memory

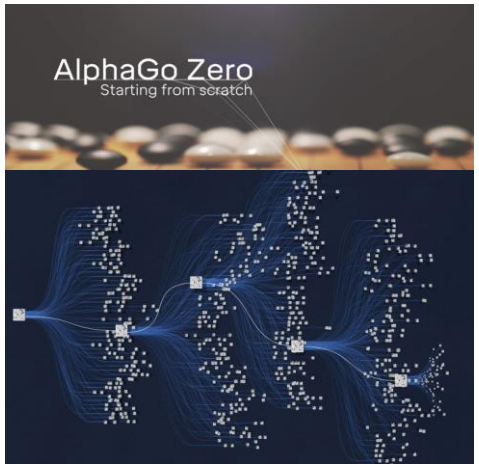
AI Hardware vs Human

❑ Energy Discrepancy



AlphaGO
1202 CPUs, 176 GPUs,
100+ Scientists.

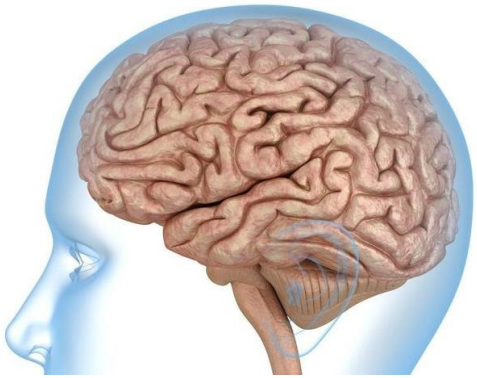
$5*10^4$ W



Alphago Zero

$1\sim2*10^3$ W

vs.



20 W

- Where does this inefficiency come from? Algorithm, Architecture, Circuits, Device, and Materials