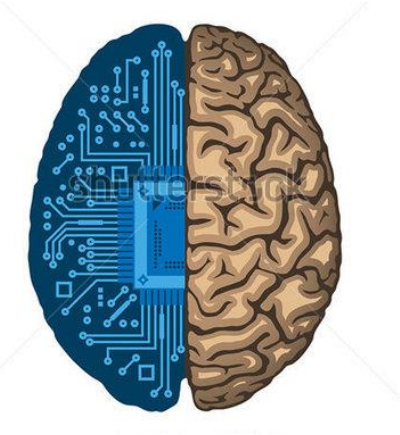


딥러닝 가속 하드웨어

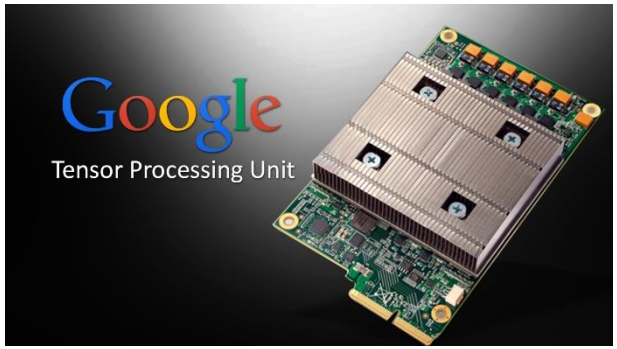
GPU or Accelerator

Won Woo Ro, Ph.D.

School of Electrical and Electronic Engineering

YONSEI UNIVERSITY

AlphaGo (2016. 3)



TPU: CPU와 GPU가아닌 전용 ASIC (2016. 5)

Mastering the Game of Go with Deep Neural Networks and Tree Search

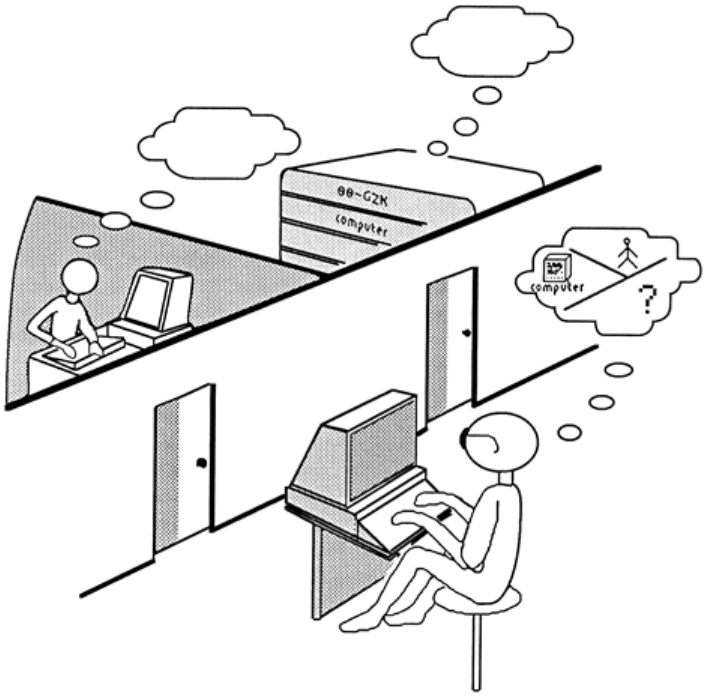
David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1], Demis Hassabis[1].

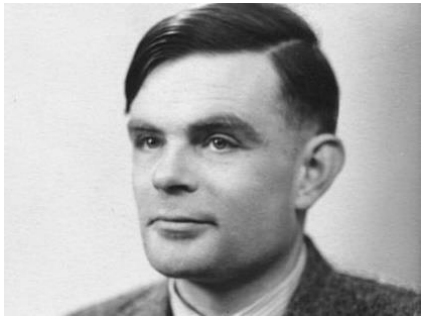
[1] Google DeepMind, 5 New Street Square, London EC4A 3TW.
[2] Google, 1600 Amphitheatre Parkway, Mountain View CA 94043.

Nature (2016.1)

computes policy and value networks in parallel on GPUs. The final version of *AlphaGo* used 40 search threads, 48 CPUs, and 8 GPUs. We also implemented a distributed version of *AlphaGo* that exploited multiple machines, 40 search threads, 1202 CPUs and 176 GPUs. The Methods section

A. I. & Turing Test





"인공지능의 발명이란 자동차에서 바퀴를 떼어낸 뒤 그 자리에 발을 달기 위해 고심하는 것이다."

– 앨런 튜링

1997년 세계 체스 챔피언 이겨 - IBM 딥블루

2011

왓슨: 켄 제닝스 74연승 저지

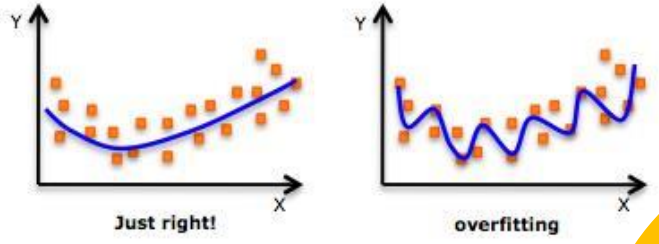
2014

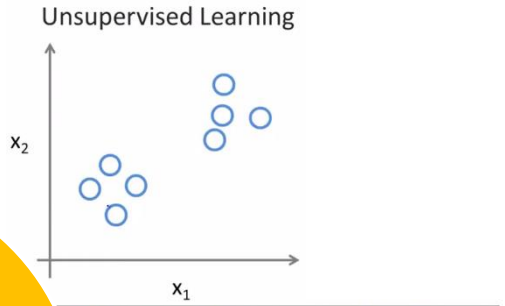
우크라이나에 사는 13살
유진 구스트만 Turing Test 통과

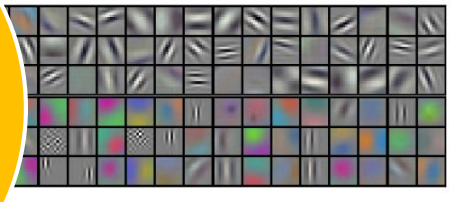
Agenda

- Deep Learning Born Again
- Machine Learning vs. Neuroscience
- Overview of Convolutional Neural Networks
- GPU Basic
- Accelerator for Neural Networks

Deep Learning Born Again

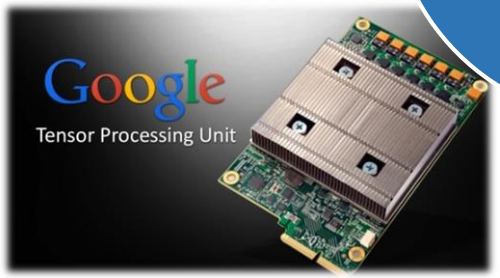






기존 신경망
모델
단점 극복

하드웨어
기술 발전

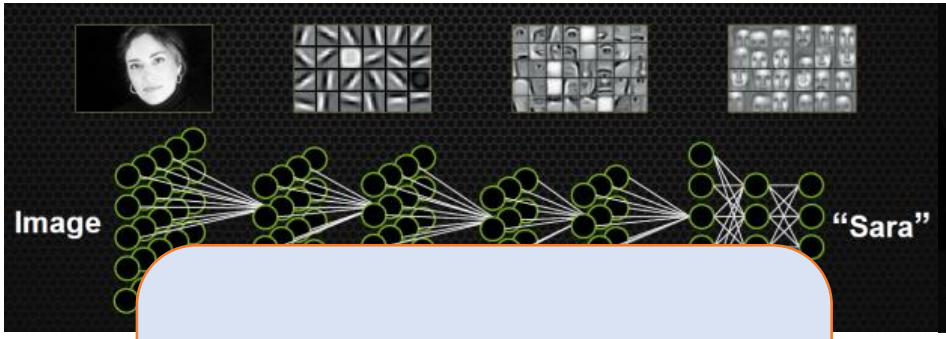


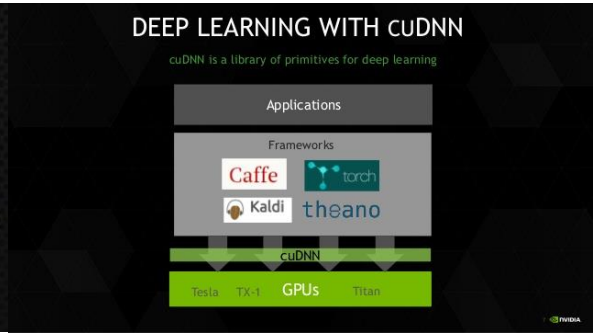
빅 데이터

People who are really serious about software should make their own hardware.

Alan Kay

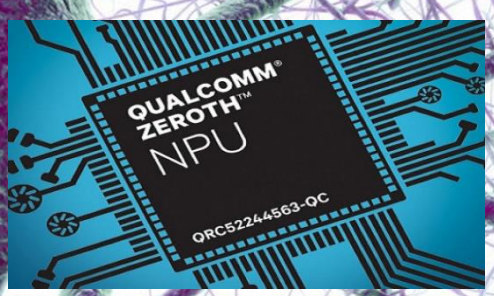
Point 1: Machine Learning vs. Neuroscience





Machine-Learning Approach

Neuroscience-Inspired Model



Point 1: Machine Learning vs. Neuroscience

Machine-Learning Approach

Multi Layer Perceptron + Back-Propagation)

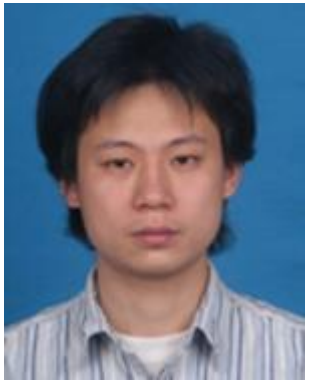
Neuroscience-Inspired Model

Spiking Neural Networks + Spike-Timing Dependent Plasticity

출처: **Neuromorphic Accelerators**: A Comparison Between Neuroscience and Machine-Learning Approaches (Micro 2015)

The DianNao 電腦 Project

The project was an academic collaboration between Prof. Yunji Chen (ICT/CSA) and Prof. Olivier Temam (Inria), within a joint ICT/Inria lab.



Currently, he leads the NOVEL group to build non-traditional computer, especially neural network computer. Before that, he participated in the **Godson/Loongson** project for more than ten years, and was a chief architect of Godson-3 microprocessor.

http://pages.saclay.inria.fr/olivier.temam/homepage/research.html

Cambricon: An Instruction Set Architecture for Neural Networks

ISCA, 2016 Summer

Neuromorphic Accelerators: A Comparison Between Neuroscience and Machine-Learning Approaches

Micro, 2015 Winter

ShiDianNao: Shifting Vision Processing Closer to the Sensor

ISCA, 2015 Summer

PuDianNao: A Polyvalent Machine Learning Accelerator

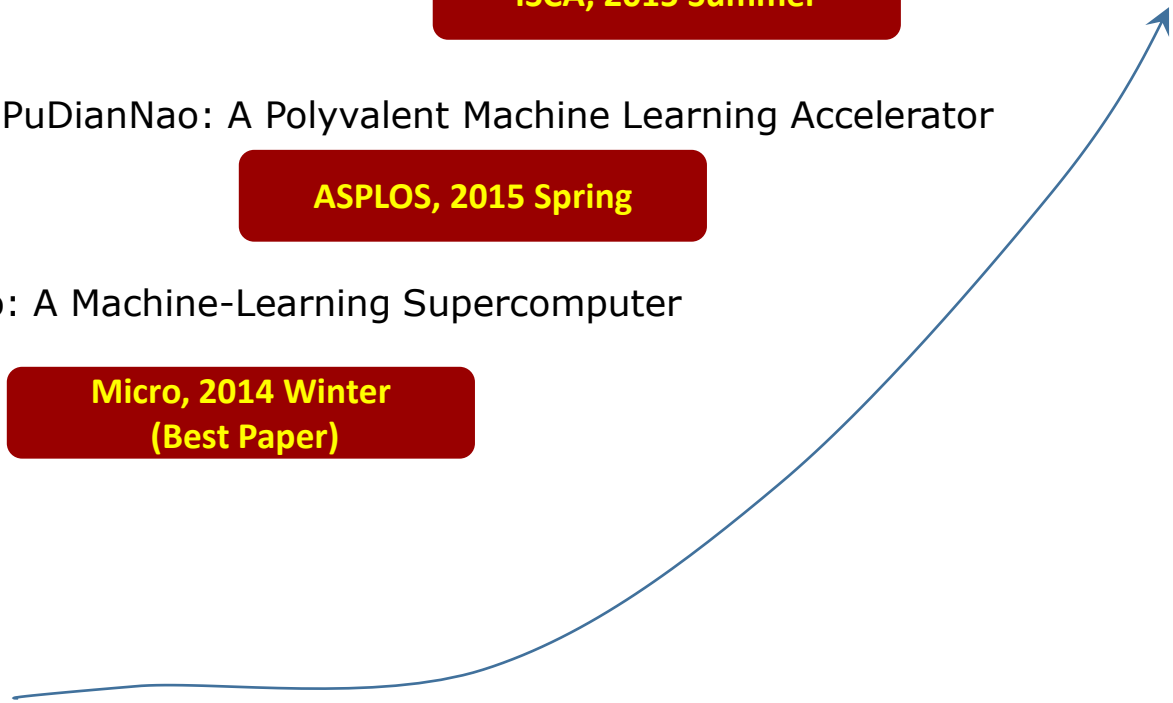
ASPLOS, 2015 Spring

DaDianNao: A Machine-Learning Supercomputer

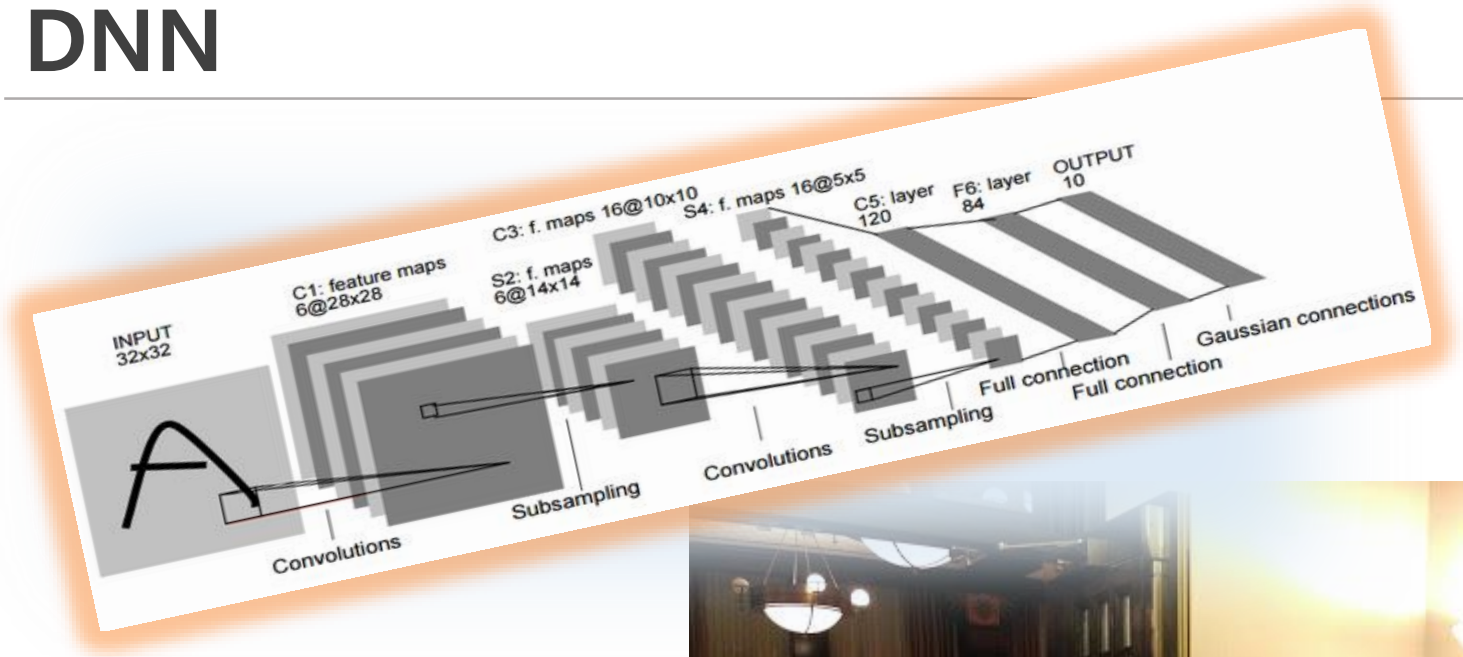
Micro, 2014 Winter (Best Paper)

DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning

ASPLOS, 2014 Spring (Best Paper)



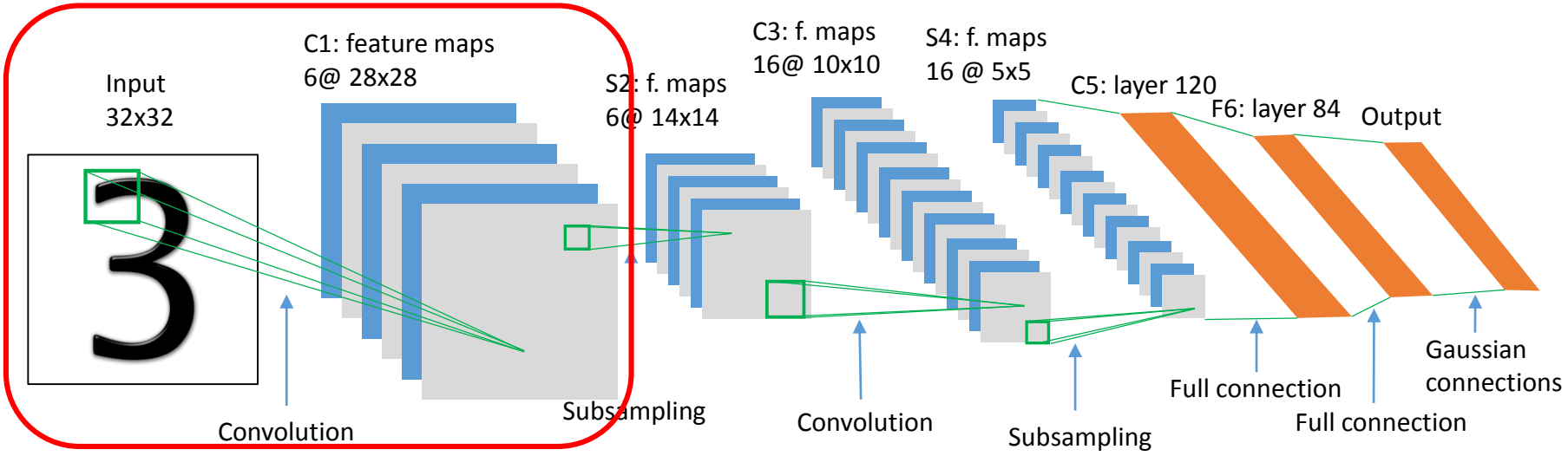
DNN

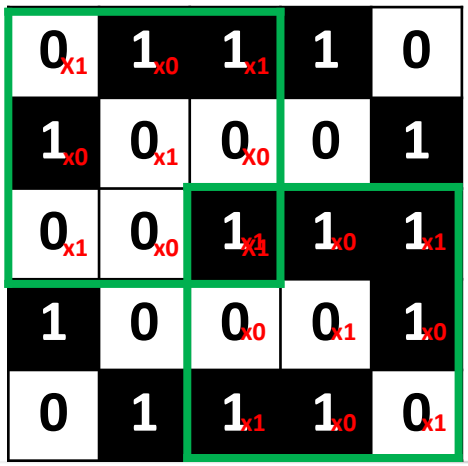

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


Yann LeCun
Geoffrey Hinton
Yoshua Bengio
Andrew Ng

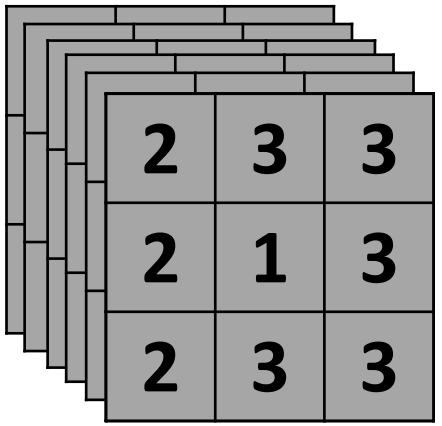
Convolutional Neural Networks

LeNet-5: Convolution

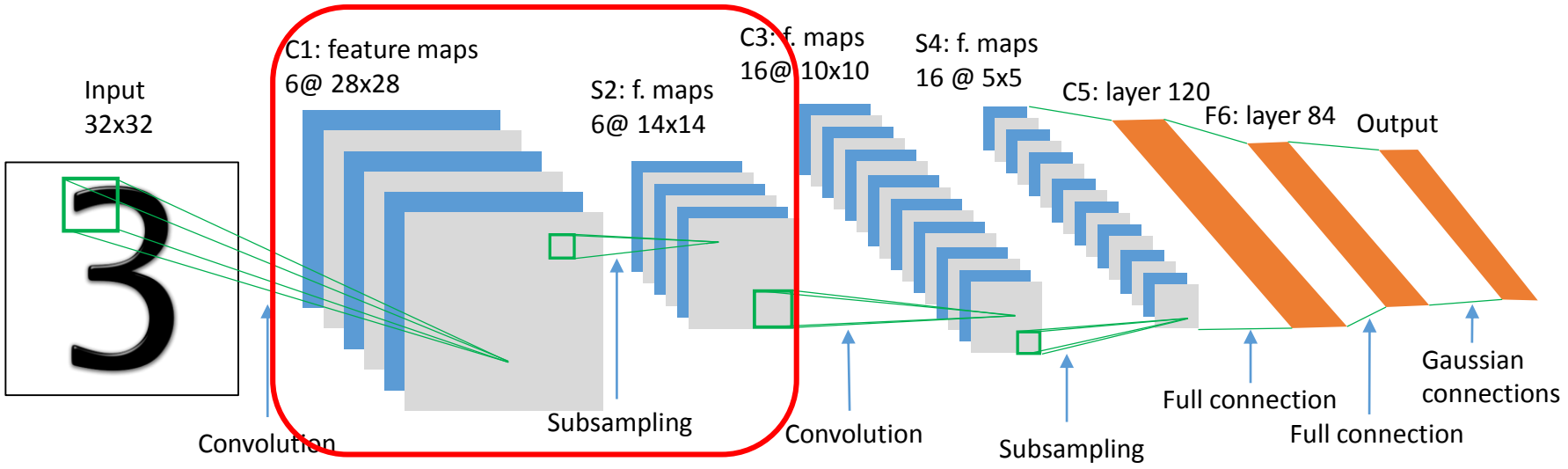




Convolution



LeNet-5: Subsampling

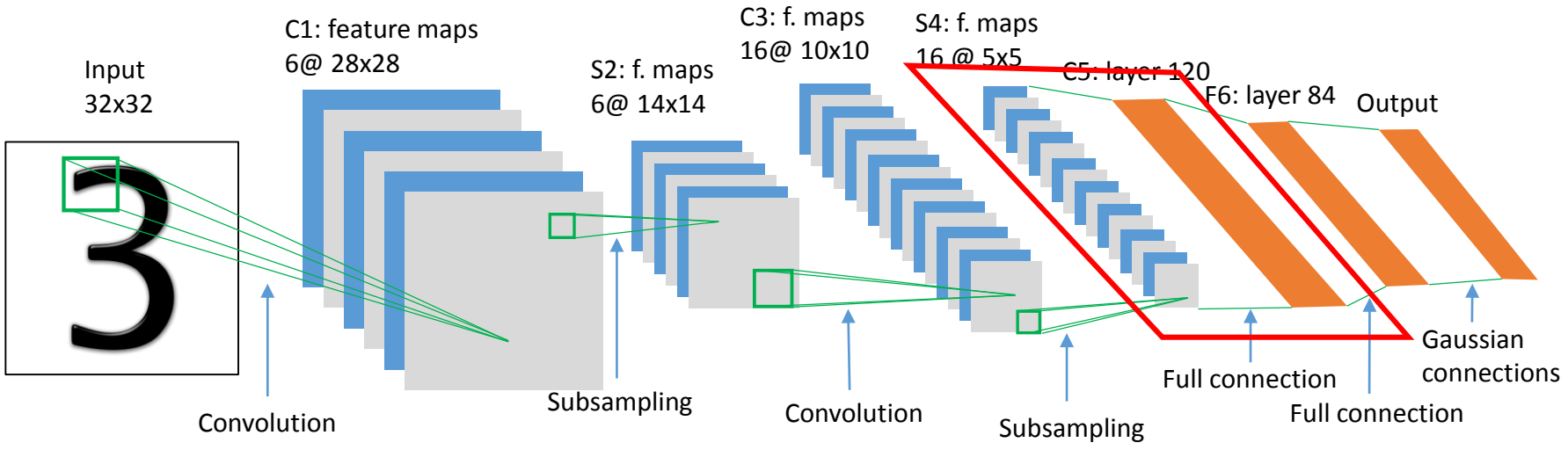


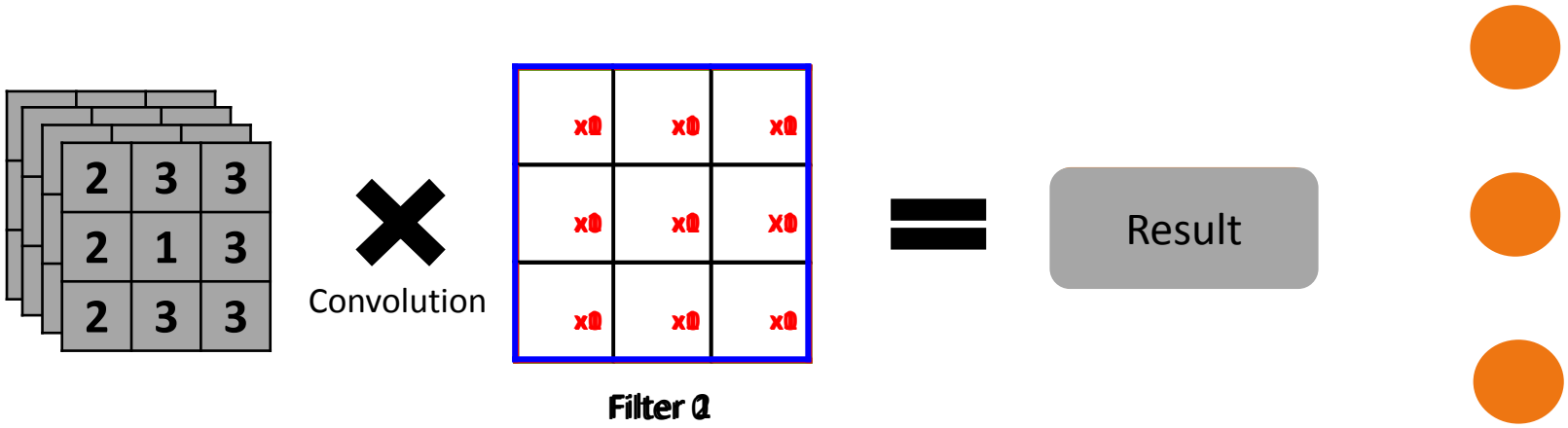
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Subsampling

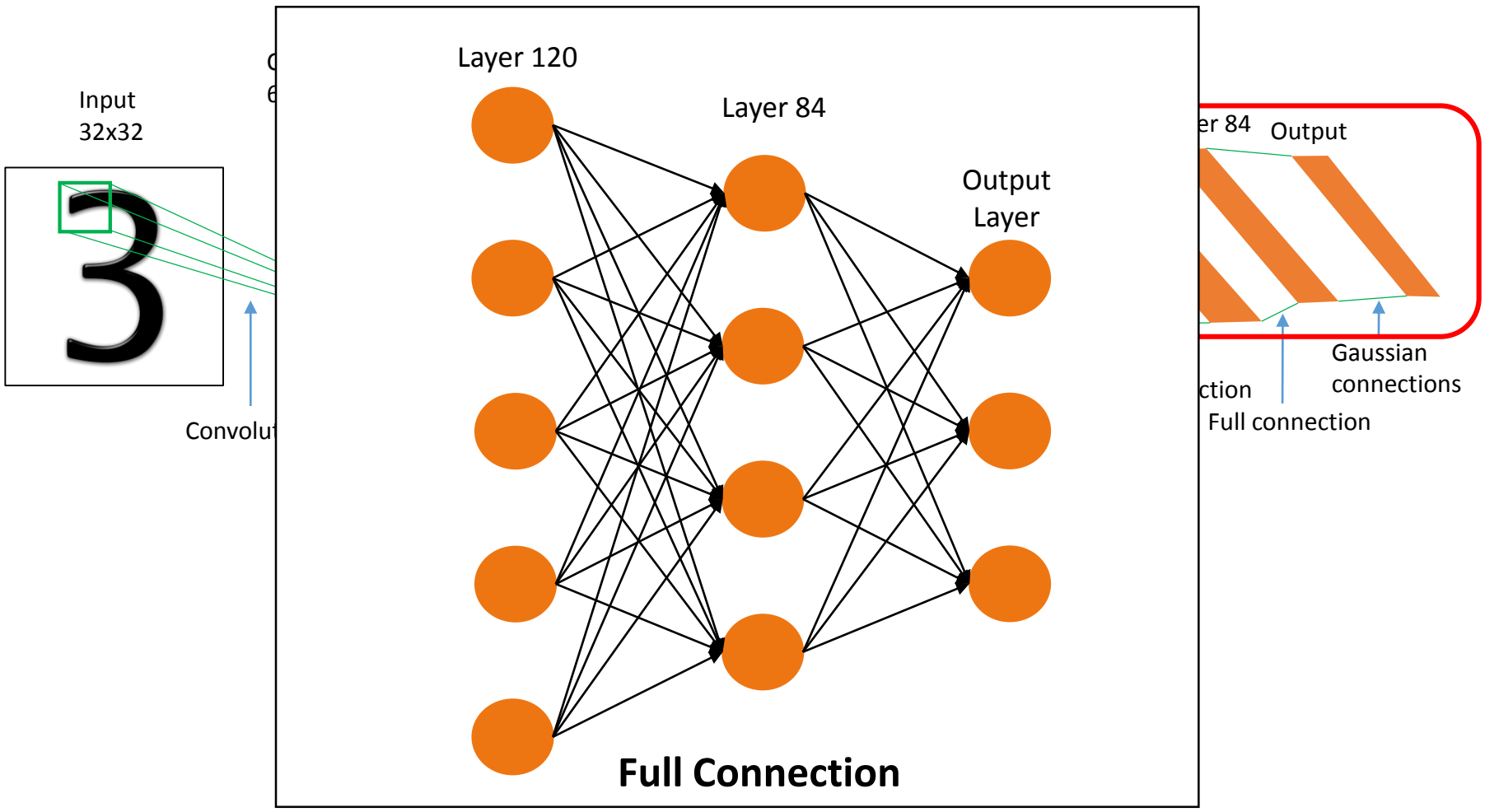
6	8
3	4

LeNet-5: Conv + Full connection





LeNet-5: Fully connected layer



“CONVOLUTION” account for over 90% of the CNN operations and dominates runtime

~90%

* Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks (ISCA 2016)

From AlexNet (2012)

	Layer type	Tunable parameters	Time (%)
1	Convolution	34,944	37.20
2	Non-linear	-	0.05
3	Normalization	-	0.12
4	Pooling	-	0.15
5	Convolution	307,456	2.05
6	Non-linear	-	0.05
7	Normalization	-	0.21
8	Pooling	-	1.11
9	Convolution	885,120	30.89
10	Non-linear	-	0.46
11	Convolution	663,936	13.56
12	Non-linear	-	0.08
13	Convolution	442,624	7.45
14	Non-linear	-	0.38
15	Pooling	-	0.74
16	Feed-forward	37,752,832	0.49
17	Non-linear	-	0.15
18	Dropout	-	0.06
19	Feed-forward	16,781,312	0.19
20	Non-linear	-	0.14
21	Dropout	-	0.07
22	Feed-forward	4,097,000	4.34
22	Softmax	-	0.06

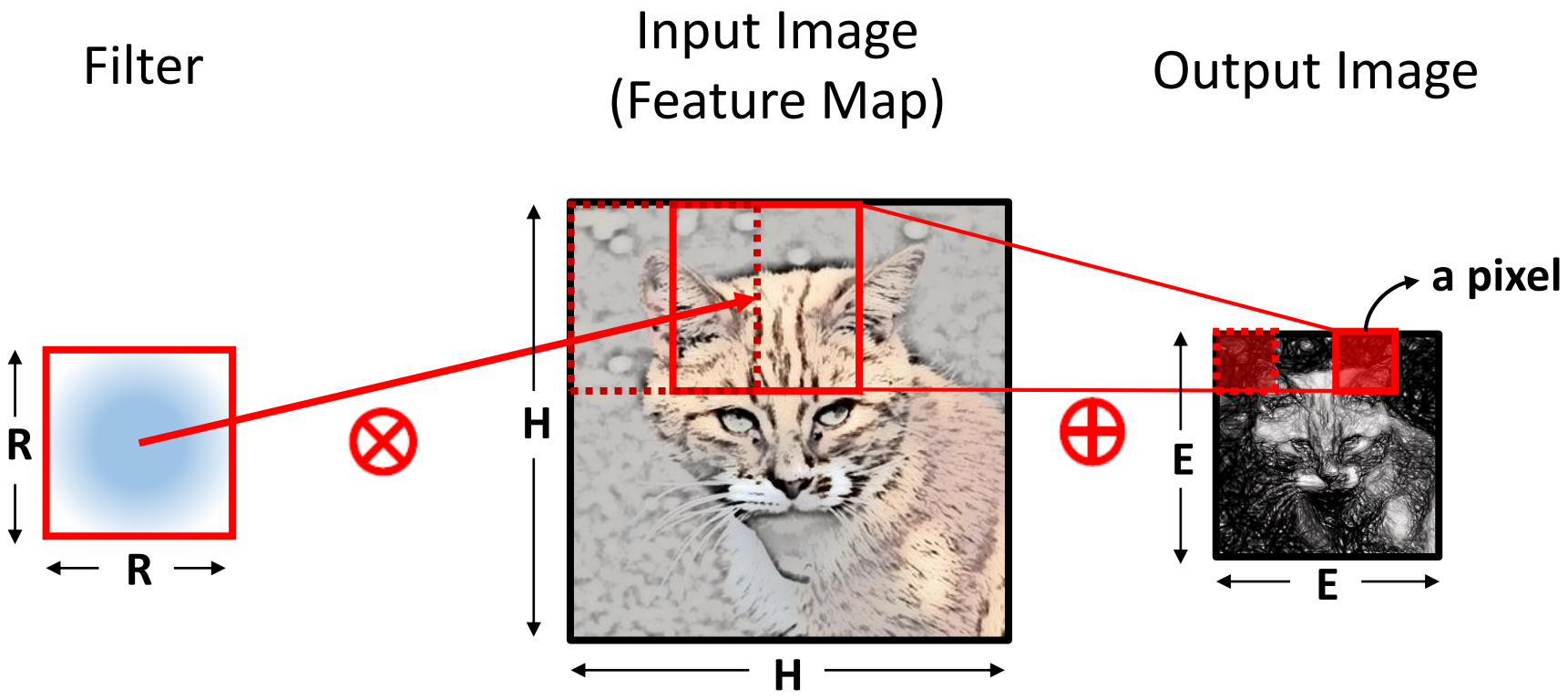
ILSVRC 2012 대회에서 우승

영상 인식 분야에서 CNN이 다시 주목받는 계기가 됨

5개의 컨볼루션 레이어와 그 사이에 끼워진 맥스풀링 레이어, 3개의 완전 연결 레이어, 그리고 1000 갈래로 출력하는 소프트맥스로 구성

An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices (IoT-App 2015)

CONVOLUTION



Sliding Window Processing

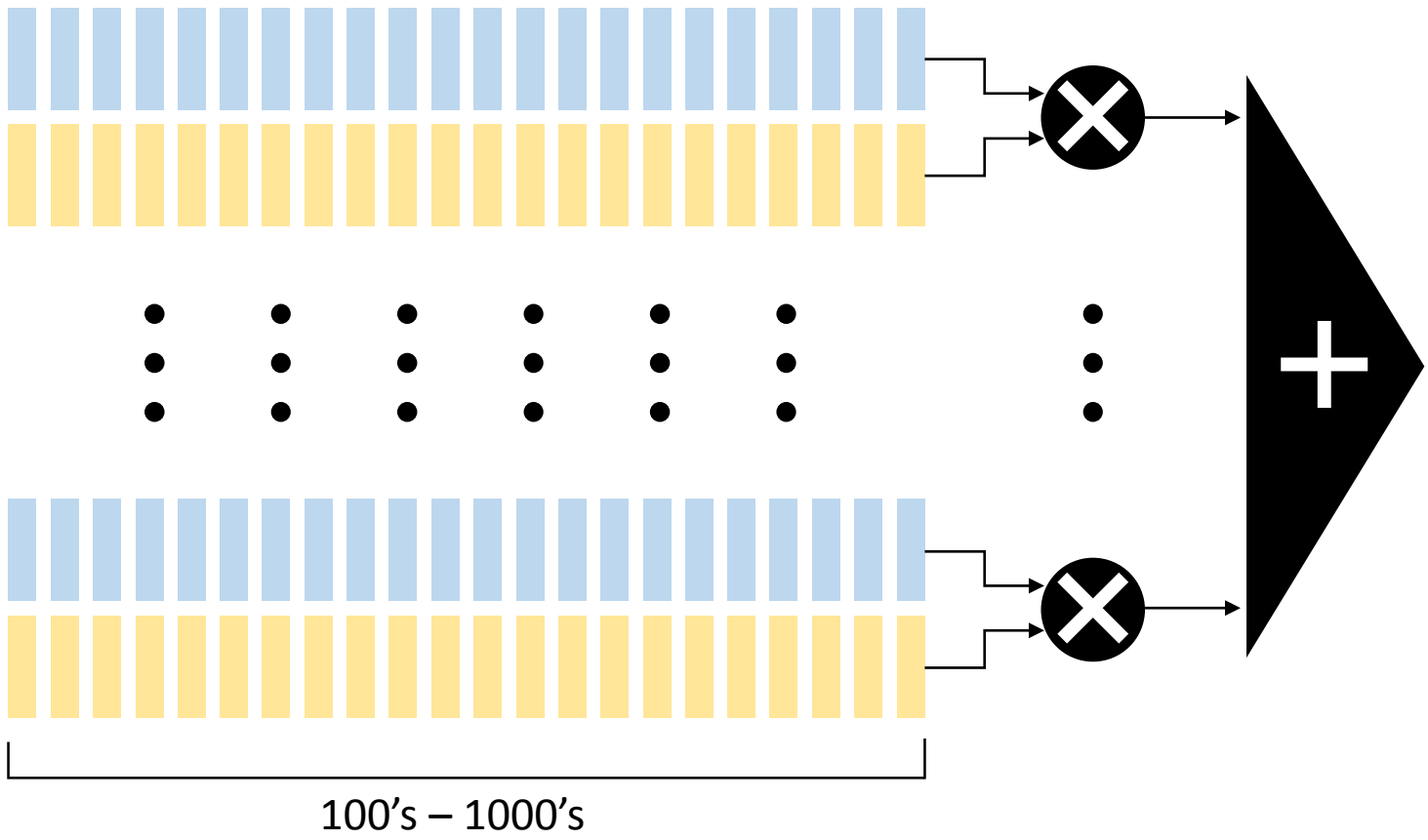
Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks (ISCA 2016)

Convolution Computation

```
for(r=0; r<R; r++)        //output feature map
  for(q=0; q<Q; q++)      //input feature map
    for(m=0; m<M; m++)    //row in feature map
      for(n=0; n<N; n++)  //column in feature map
        for(k=0; k<K; k++)    //row in convolution kernel
          for(l=0; l<L; l++)  //column in convolution kernel
            Y[r][m][n]+=W[r][q][k][l]*X[q][m+k][n+l];
```

Minimizing Computation in Convolutional Neural Networks (ICANN 2014)

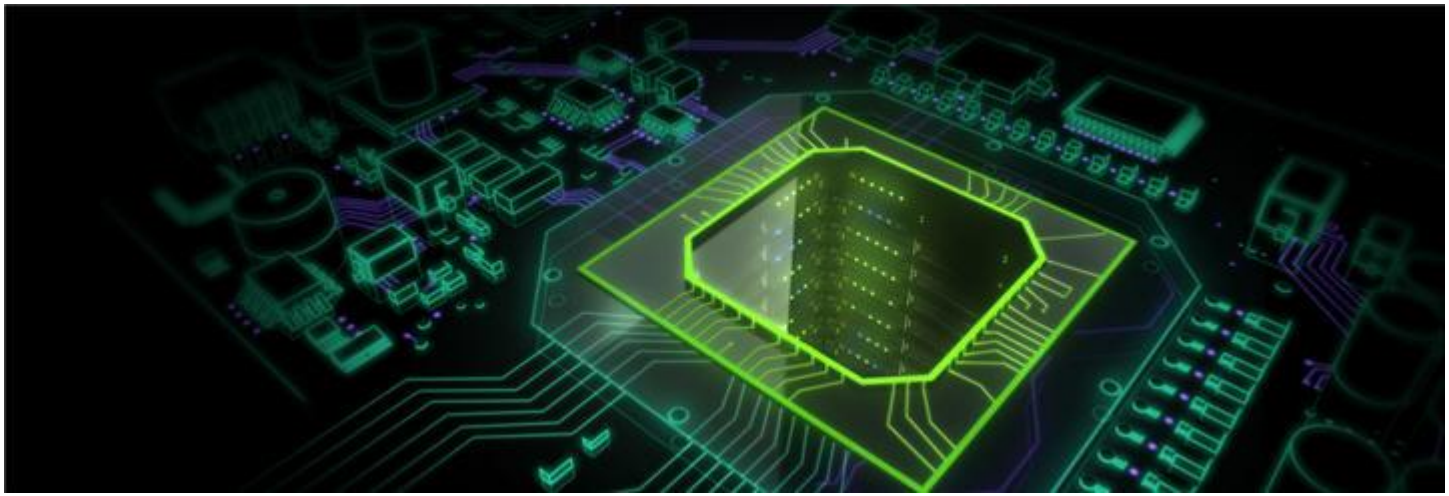
CONVOLUTION = SIMD Paradise!



CNVLUTIN: Ineffectual-neuron-free DNN computing (ISCA 2016)

Why GPUs?

- **Many core structure**
  - More than thousands of cores to compute floating point operations
  - Exploiting high Thread Level Parallelism (TLP)
- **Power efficiency**
  - Low control overheads due to SIMT architecture
- **Scalability**
  - Good scalable performance across multiple GPUs
  - 4 GPUs improve 1.9x better performance than 2 GPUs



History of GPU

- **Graphics Accelerator (~1999)**

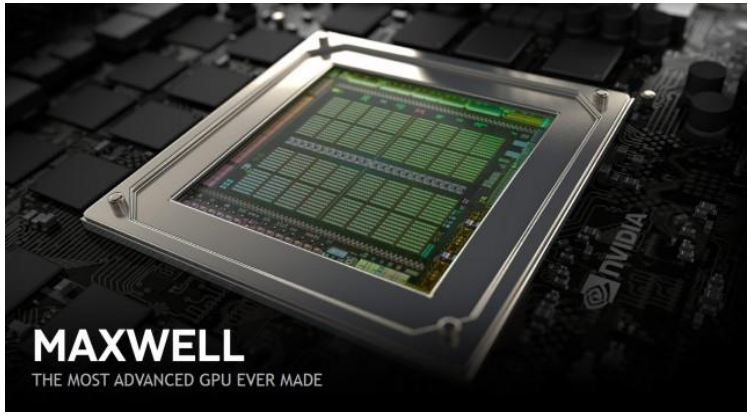
  단순 화면 표시 장치, 제한적인 3D graphics 가속 기능

- **Graphics Processing Unit (1999~2006)**

  복잡한 3D graphics 처리를 위한 하드웨어적 발전

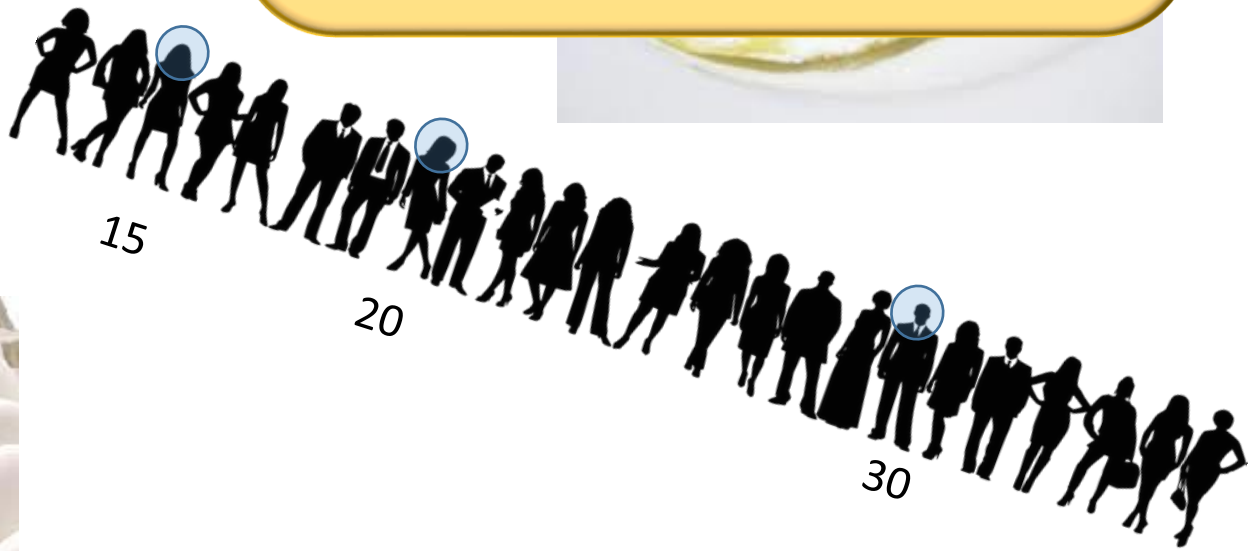
- **General-Purpose Computation on GPU (~현재)**

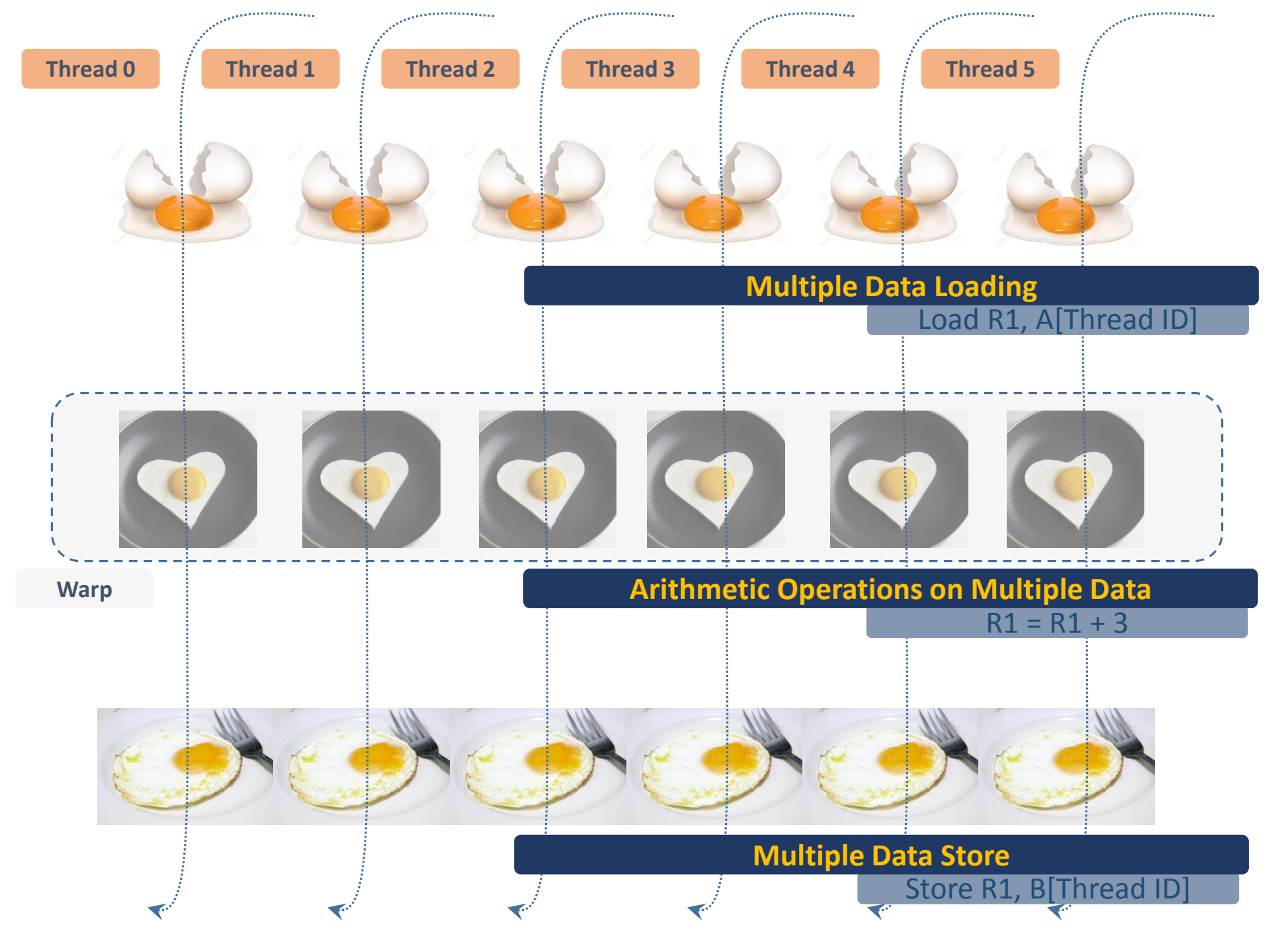
  3D graphics 만이 아닌, 일반 범용 연산 분야에서의 사용 시작



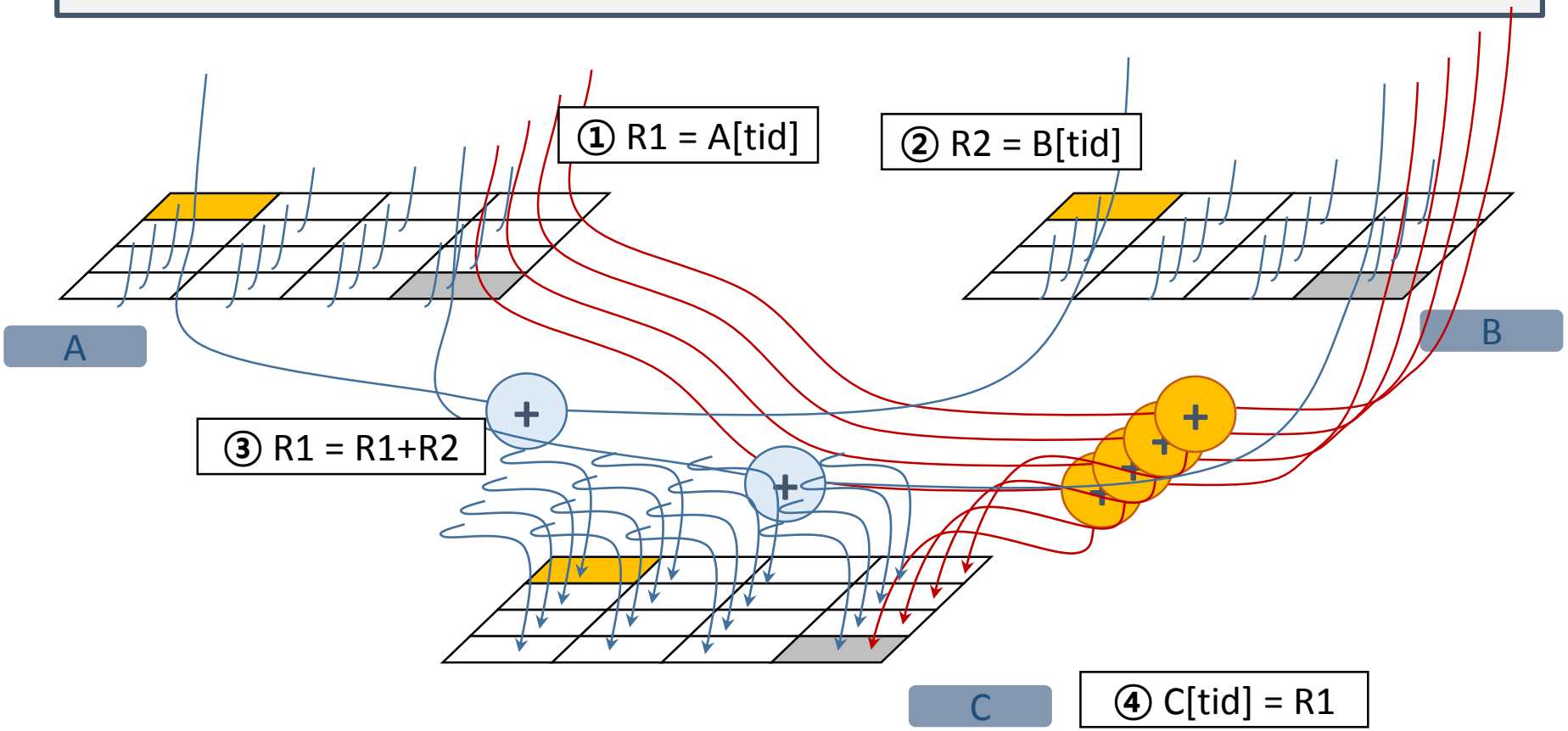

```
for (남은사람 > 0)
{
        계란 깨기;
        계란 굽기;
        덜어 주기;
}
```

1 2 3 4 5 6 7 8 9 10 11 12



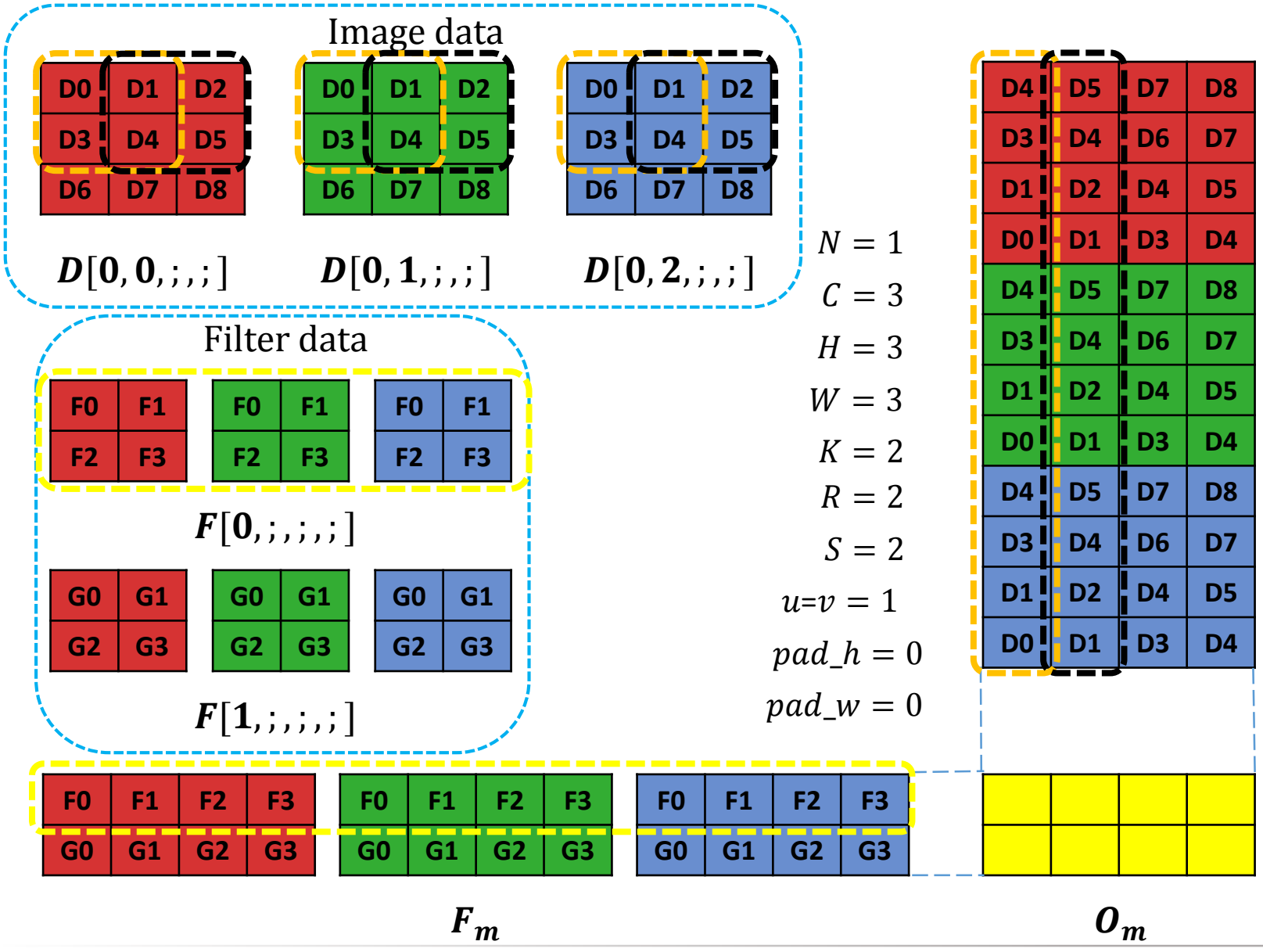

Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Multiple Data Loading
Load R1, A[Thread ID]
Warp
Arithmetic Operations on Multiple Data
R1 = R1 + 3
Multiple Data Store
Store R1, B[Thread ID]

```
__global__
void GPU_kernel_sum(float* A, float* B, float* C)
{
  // calculate the index value inside of a thread
  int tid = 쓰래드 아이디!
  C[tid] = A[tid] + B[tid];
}
```

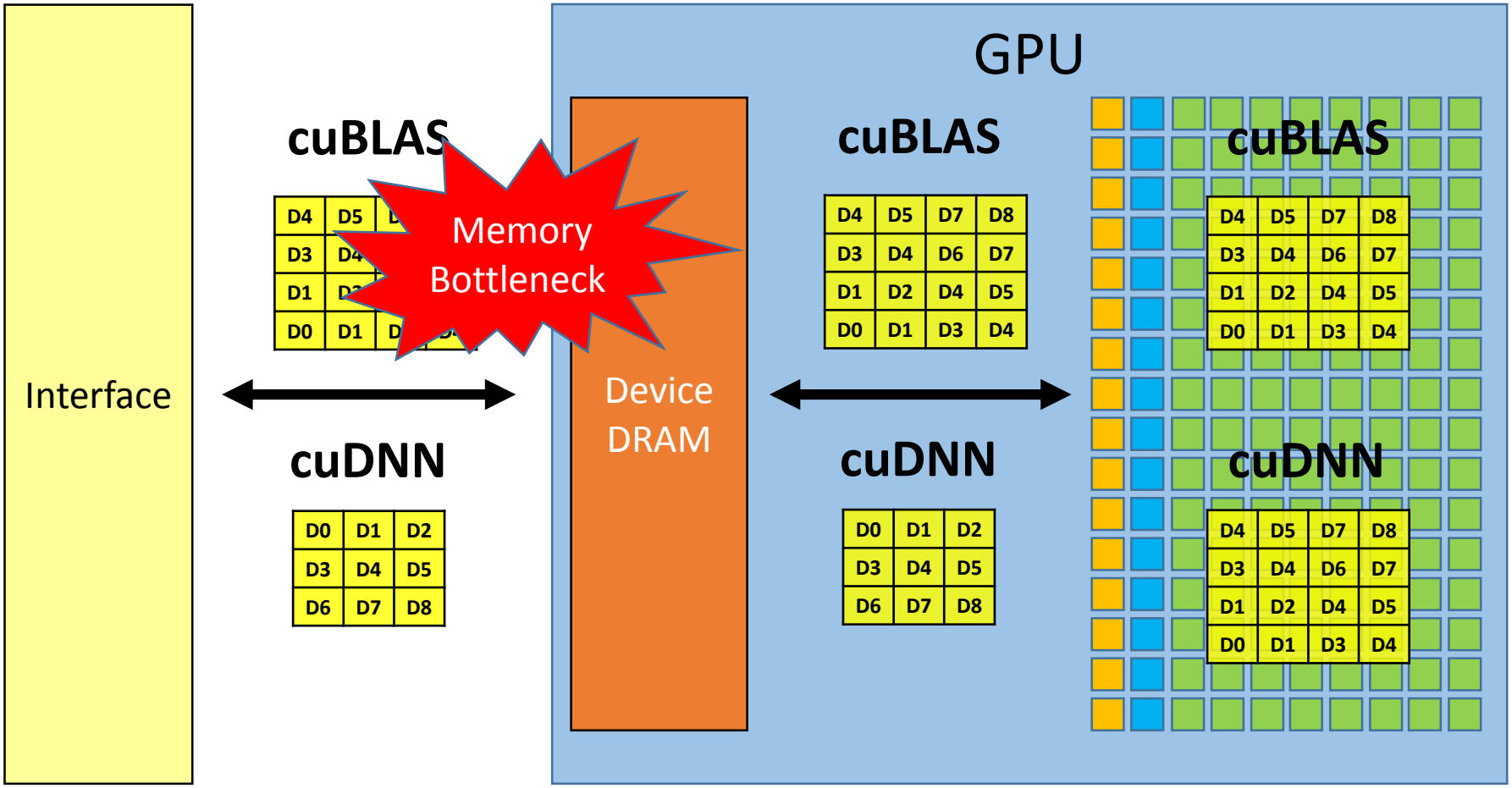


GPU Architecture: NVIDIA Maxwell (2014)

Convolution Lowering



Memory Usage in cuDNN



Weight Sharing

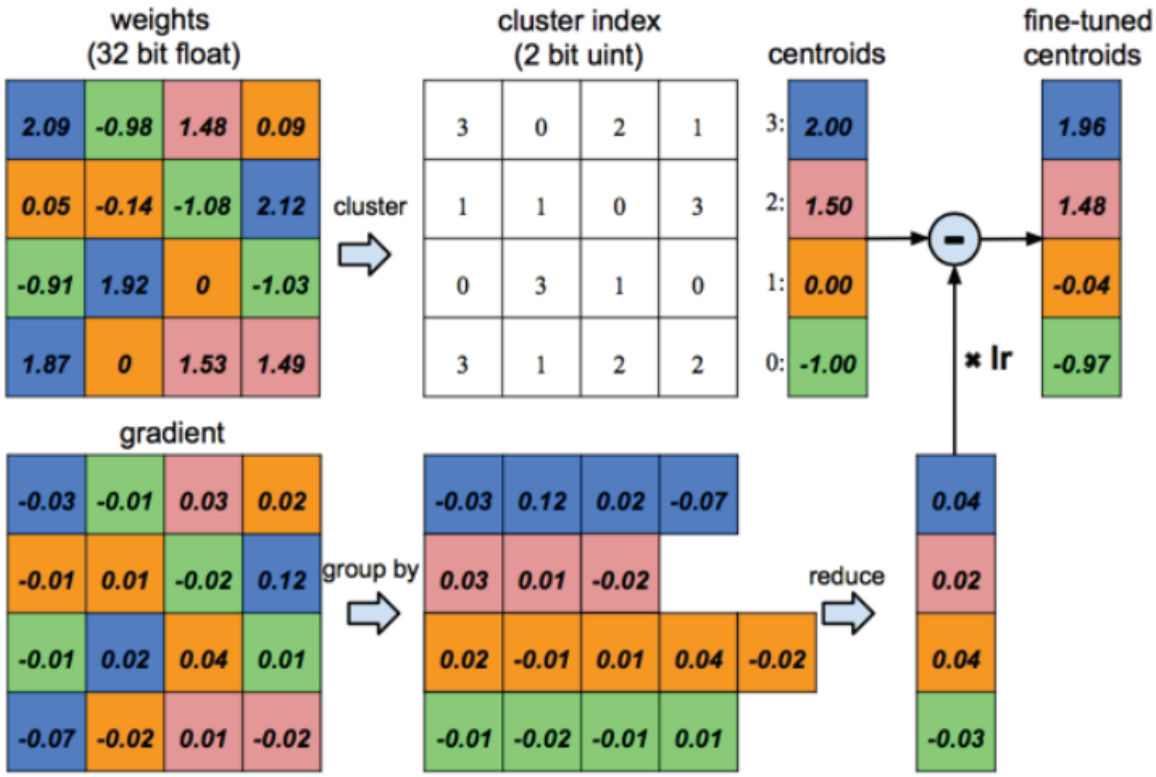


Figure 3: Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom)

Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016

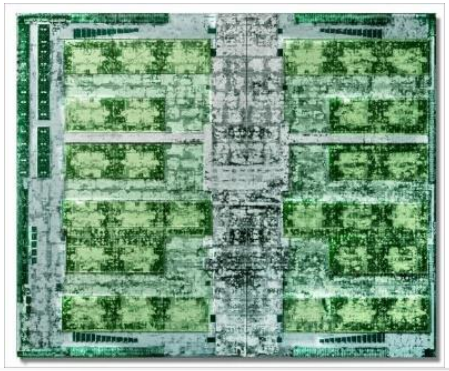
For pruned AlexNet, we are able to quantize to 8-bits (256 shared weights) for each CONV layers, and 5-bits (32 shared weights) for each FC layer without any loss of accuracy

NVIDIA DGX-1

- 170 TFLOPS
- 8x Tesla P100 16GB
- NVLink Hybrid Cube Mesh
- 3RU
- 3200W

Tesla P100
for NVLink-enabled Servers

5.3 TF DP · 10.6 TF SP · 21 TF HP
720 GB/sec Memory Bandwidth, 16 GB

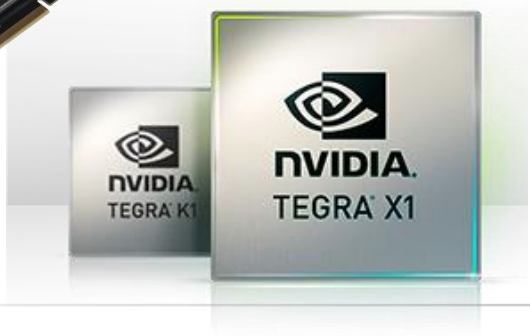


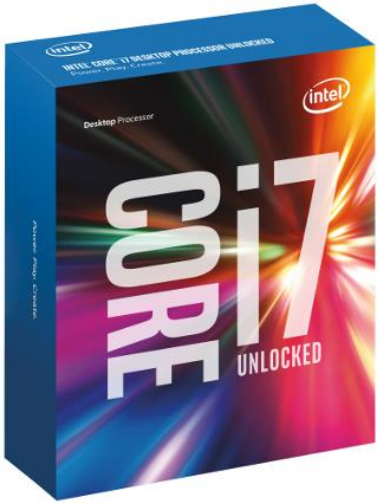
- GP100 GPU: 610 mm^2 in die size on TSMC's 16nm FinFET (15B Transisotrs)
- 16GB of HBM2 RAM

Execution Time & Power (AlexNet)

Network: AlexNet	Batch Size	Titan X (FP32) 15.03	Tegra X1 (FP32) 15.01	Tegra X1 (FP16) 15.01	i7 6700K (FP32) 15.08	Xeon E5-2698 v3 (FP32) 14.09
Inference Performance(img/sec)	1	**405**	47	67	62	76
Power(W)		164.0	5.5	5.1	49.7	111.7
Performance/Watt(img/sec/W)		**2.5**	8.6	13.1	1.3	0.7
Inference Performance(img/sec)	128(Titan X) 128(Tegra X1) 48(Core i7) 48(Xeon E5)	**3216**	155	258	242	476
Power(W)		227.0	6.0	5.7	62.5	149.0
Performance/Watt(img/sec/W)		**14.2**	25.8	45.0	3.9	3.2

AlexNet Execution Time & Power





Execution Time & Power at GoogleNet

Network: GoogLeNet	Batch Size	Titan X (FP32)	Tegra X1 (FP32)	Tegra X1 (FP16)
Inference Performance(img/sec)	1	138	33	33
Power(W)		119.0	5.0	4.0
Performance/Watt(img/sec/W)		1.2	6.5	8.3
Inference Performance(img/sec)	128(Titan X) 128(Tegra X1) 48(Core i7) 48(Xeon E5)	863	52	75
Power(W)		225.0	5.9	5.8
Performance/Watt(img/sec/W)		3.8	8.8	12.8

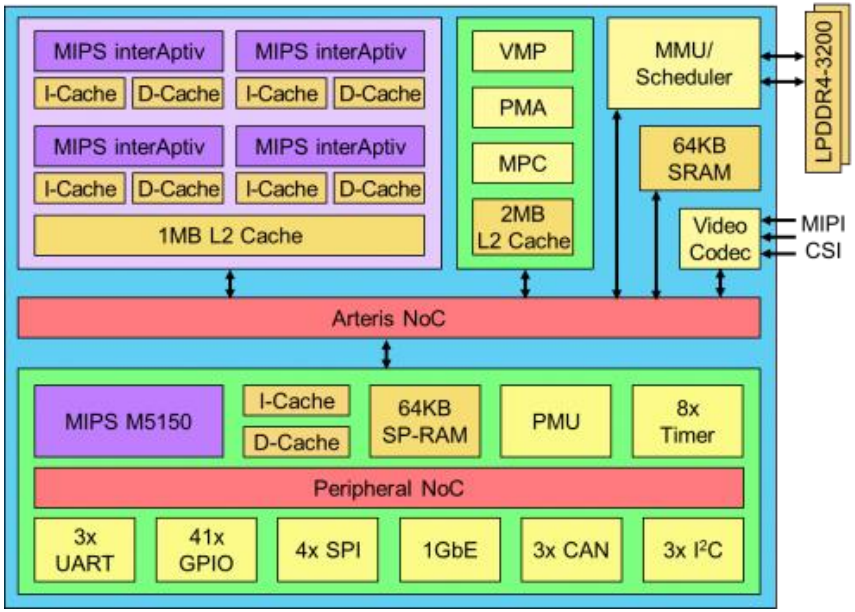
GoogLeNet Execution Time & Power

Mobileye Eye Q4 vs. EyeQ3

Mobileye EyeQ4 Specifications

Parameter	VMP	PMA	MPC
Clock Speed	1Ghz	0.75GHz	1
Cores	6	2	2
MAC/Core	76	372	32
TFLOPS	912000	1116000	128000

*Vector Microcode Processors (VMP)
*Programmable Macro Array (PMA)
*Multithreaded Processing Cluster (MPC)



Mobileye EyeQ3 vs. Tegra X1 Specifications Comparison

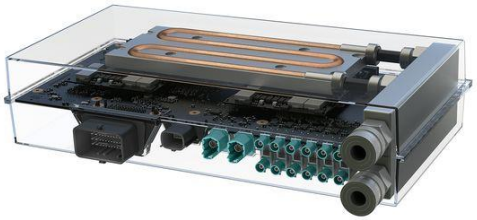
Parameter	EyeQ3	Tegra X1
Clock Speed	0.5Ghz	1Ghz
Cores	4	256
MAC/Core	64	2
MFLOPS	256000	1024000
Utilization	0.8	0.28
Effective MAC/s	102M	143M
Power	2.5W	10W
Silicon Area	$42mm^2$	$126mm^2$
Fabrication Process	40nm	20nm

http://eyesofthings.eu/?p=762
http://www.linleygroup.com/mpr/article.php?id=11437

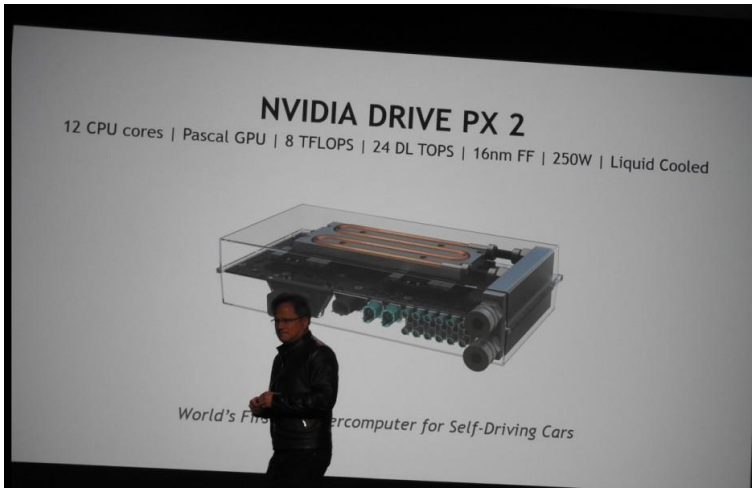
ADAS

Audi zFAS (2015)

NVIDIA Drive PX 1 (2015)



NVIDIA Drive PX 2 (2016)



	DRIVE PX 1 (2015)	DRIVE PX 2 (2016)
SoCs	2x Tegra X1	2x Tegra "Parker"
Discrete GPUs	None	2x Pascal
CPU Cores	8x ARM Cortex-A57 + 8x ARM Cortex-53	4x NVIDIA Denver + 8x ARM Cortex-A57
GPU Cores	2x Tegra X1 (Maxwell)	2x Tegra "Parker" (Pascal) + 2x Pascal MXM GPUs
FP32 TFLOPS	> 1 TFLOPS	8 TFLOPS
FP16 TFLOPS	> 2 TFLOPS	16 TFLOPS (24-8)

SW vs. HW

가속기

FPGAs

Device	Performance (Gflops)	Power (W)	Energy Efficiency (Gflops/W)
Intel Xeon X5675	22	95	0.23
Nvidia Tesla K20	486	235	2.07
FPGA2015*	61.62	18.61	3.31
MS CNN**	182	25	7.28

- **Performance**
  - x2.6~ better performance on GPU
  - ~x8.3 better performance than CPU
- **Energy efficiency**
  - Low power requirement than other implementations
  - Higher energy efficiency than CPU and GPU

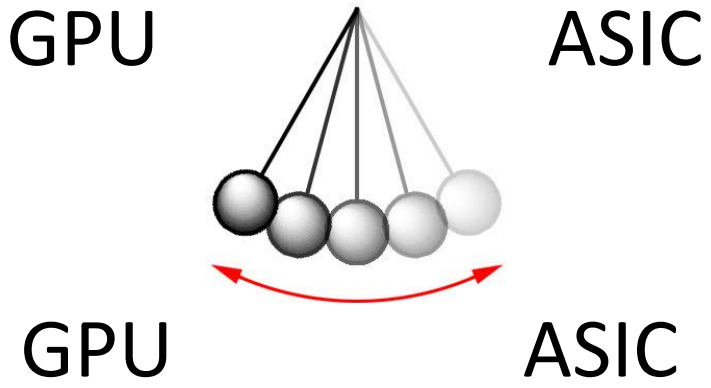
* Zhang, Chen, et al., "Optimizing fpga-based accelerator design for deep convolutional neural networks."

* * Ovtcharov, Kalin, et al., "Accelerating deep convolutional neural networks using specialized hardware."

Historical Debates

Performance
&
Energy Efficiency

Programmability
&
Flexibility



We may prefer to use **FPGAs** or **ASIC** when targeting a **single** network for best efficiency.

We may prefer to use **GPUs** when targeting **various** networks (different input sizes, filters, or layers).

The 1st Machine Learning Accelerator (ISCA 2012)

O. Temam. A Defect-Tolerant Accelerator for Emerging High-Performance Applications

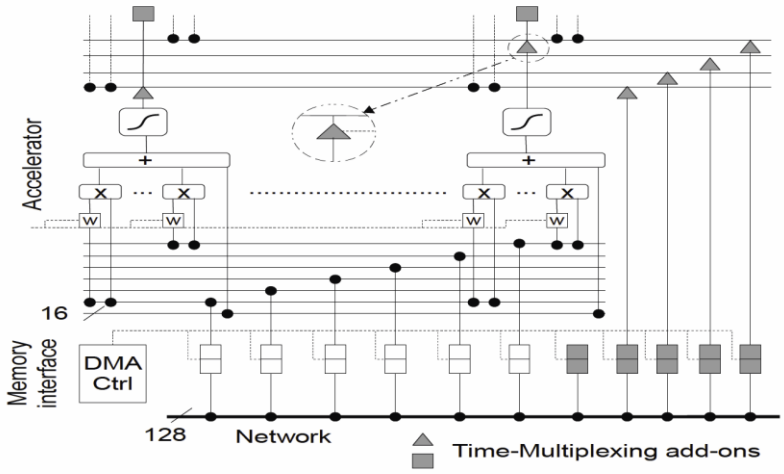


Fig. 3: *Accelerator implementation (memory interface & hidden layer).*

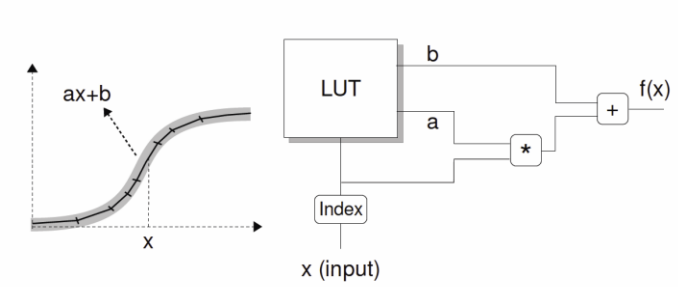


Fig. 4: *Sigmoid: piecewise approximation and hardware implementation.*

DianNao (ASPLOS 2014)

- **ISCA 2012 accelerator with local memories in order to capture the locality properties of deep neural networks and overcome memory bandwidth limitations.**

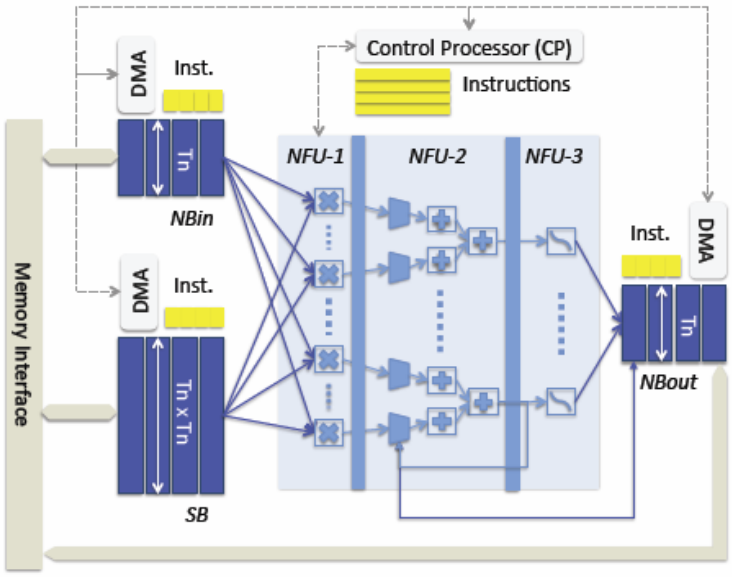


Figure 11. *Accelerator.*

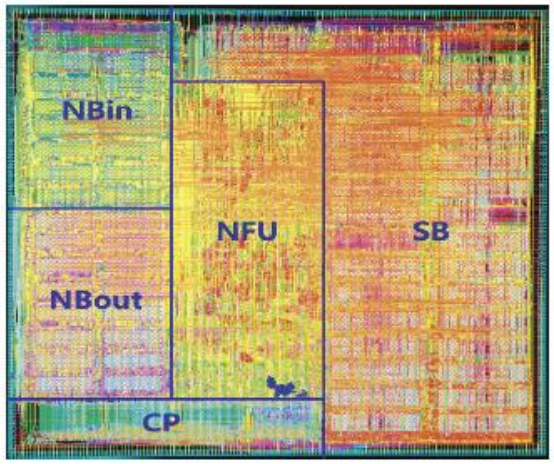


Figure 15. *Layout (65nm).*

DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning (ASPLOS 2014)

DaDianNao (MICRO 2014)

- **A multi-chip version of DianNao and had two main goals**
  - Excellent scalability properties thanks to the partitioning properties of neural network layers
  - Enough memory capacity to store the whole machine-learning model on-chip

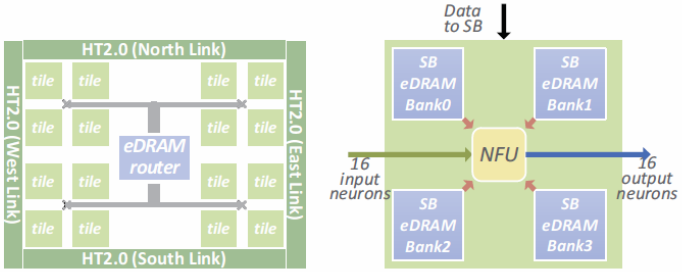


Figure 5: *Tile-based organization of a node (left) and tile architecture (right). A node contains 16 tiles, two central eDRAM banks and fat tree interconnect; a tile has an NFU, four eDRAM banks and input/output interfaces to/from the central eDRAM banks.*

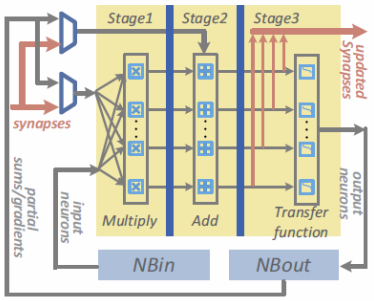


Figure 6: *The different (parallel) operators of an NFU: multipliers, adders, max, transfer function.*

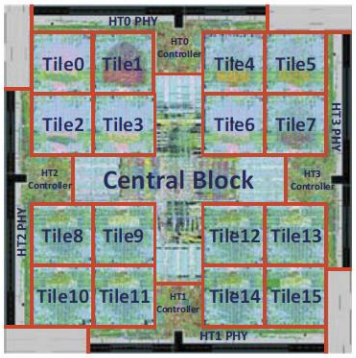


Figure 9: *Snapshot of the node layout.*

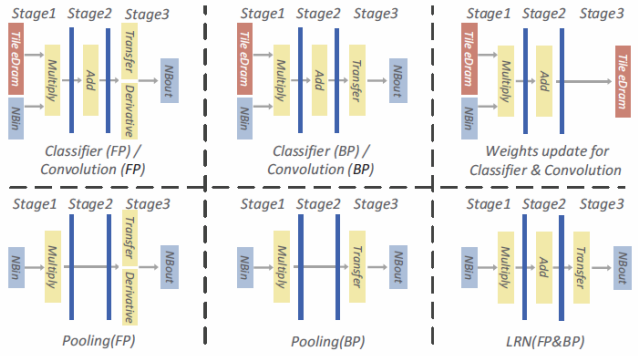
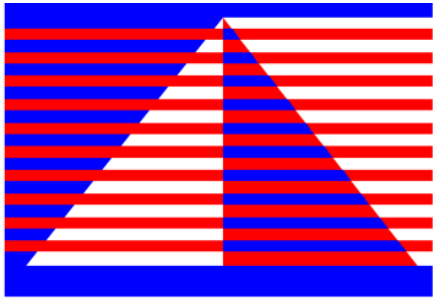


Figure 7: *Different pipeline configurations for CONV, LRN, POOL and CLASS layers.*

DaDianNao: A Machine-Learning Supercomputer (Micro 2014)

ISCA 2016 Neural Networks



The 43rd
ACM/IEEE
International
Symposium
on Computer
Architecture
Seoul, Korea

ISCA 2016

9 Papers in 3 Sessions (9/56 = 16%)

- **Instruction Set Architecture (ISA)**
  - Cambricon: An Instruction Set Architecture for Neural Networks
- **Processing In Memory (PIM)**
  - Neurocube: A programmable Digital Neuromorphic Architecture with High-Density 3D memory
  - PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory
- **Accelerators**
  - Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators
  - Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing
  - EIE: Efficient Inference Engine on Compressed Deep Neural Network
  - Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks
- **Analog**
  - ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars
  - RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision

New ISA

Cambricon: An Instruction Set Architecture for Neural Networks

Shaoli Liu, Zidong Du, Jinhua Tao, Dong Han, Tao Luo,
Yuan Xie, Yunji Chen, and Tianshi Chen

ICT, UCSB, Cambricon Ltd.

Cambricon: Contribution

- **Decomposing complex and informative instructions describing high-level functional blocks of NN into shorter instructions corresponding to low-level computational operations**
- **Simple and short instructions reduce design/verification complexity and power/area of the instruction decoder**

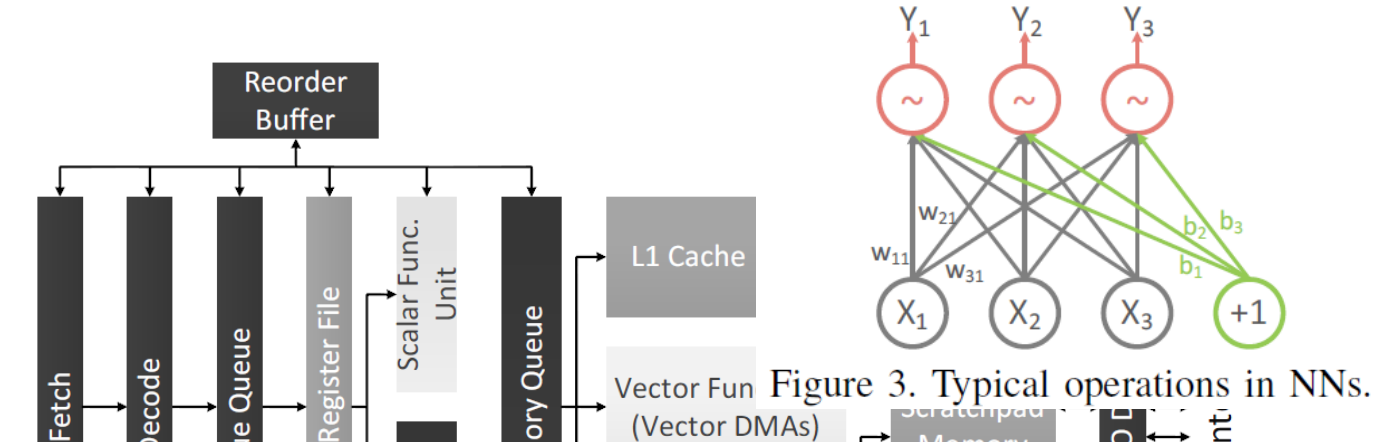


Figure 3. Typical operations in NNs.

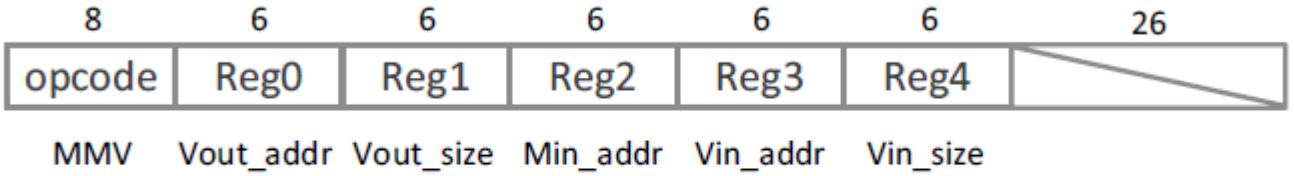


Figure 4. Matrix Mult Vector (MMV) instruction.

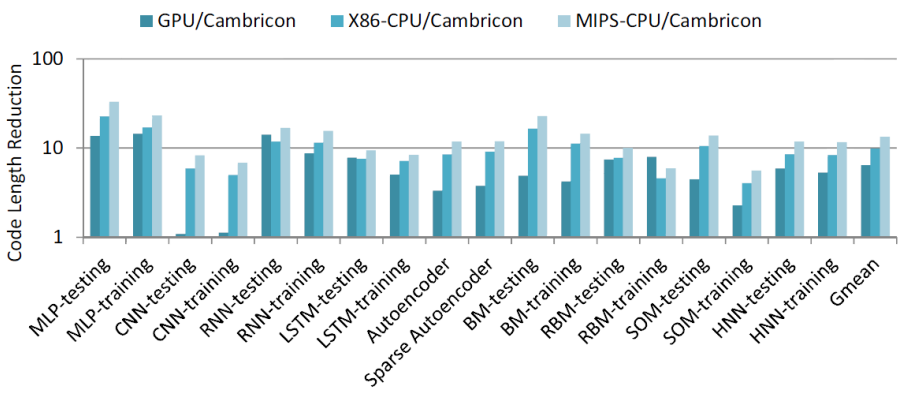
Cambricon: Design Guidelines

- **Data-level parallelism: data-level parallelism enabled by vector/matrix is more efficient than instruction level parallelism**
- **Customized vector/matrix instructions: many common operations of NN techniques are not covered by traditional linear algebra libraries**
- **Using on-chip scratchpad memory: using fixed-width power-hungry vector register files is no longer the most cost-effective choice**

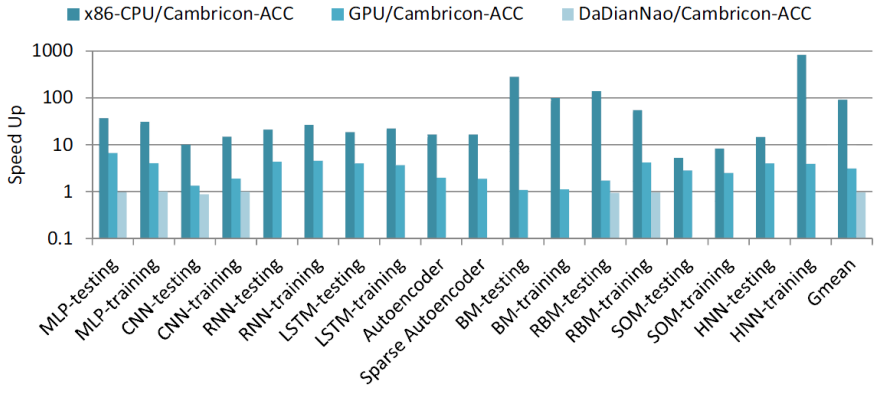
Instruction Type		Examples	Operands
Control		jump, conditional branch	register (scalar value), immediate
Data Transfer	Matrix	matrix load/store/move	register (matrix address/size, scalar value), immediate
	Vector	vector load/store/move	register (vector address/size, scalar value), immediate
	Scalar	scalar load/store/move	register (scalar value), immediate
Computational	Matrix	matrix multiply vector, vector multiply matrix, matrix multiply scalar, outer product, matrix add matrix, matrix subtract matrix	register (matrix/vector address/size, scalar value)
	Vector	vector elementary arithmetics (add, subtract, multiply, divide), vector transcendental functions (exponential, logarithmic), dot product, random vector generator, maximum/minimum of a vector	register (vector address/size, scalar value)
	Scalar	scalar elementary arithmetics, scalar transcendental functions	register (scalar value), immediate
Logical	Vector	vector compare (greater than, equal), vector logical operations (and, or, inverter), vector greater than merge	register (vector address/size, scalar)
	Scalar	scalar compare, scalar logical operations	register (scalar), immediate

Cambricon: Evaluation

- **Code density: meaningful ISA metric only when the ISA is flexible enough to cover a broad range of applications**
  - **13.62x**, 22.62x, and 32.92x compared **to GPU**, X86, and MIPS, respectively
- **Performance**
  - 91.72x compared to x86-CPU
  - **3.09x compared to GPUs**
  - -4.5% compared to DaDianNao: low-level computational instructions which may bring in additional pipeline bubbles between instructions



The Reduction of Code Length



The Speedup of Cambricon-ACC

* Due to the incomplete and restricted ISA, DaDianNao can only accommodate 3 out of 10 benchmarks

References

- cuDNN: Efficient Primitives for Deep Learning," CoRR, vol. abs/1410.0759, 2014
- Mastering the game of Go with deep neural networks and tree search Nature, vol. 529 (2016), pp. 484-503
- DaDianNao: A Machine-Learning Supercomputer (Micro 2014)
- Neuromorphic Accelerators: A Comparison Between Neuroscience and Machine-Learning Approaches (Micro 2015)
- A Defect-Tolerant Accelerator for Emerging High-Performance Applications (ISCA 2012)
- Optimizing FPGA-based accelerator design for deep convolutional neural networks (FPGA 2015)
- Accelerating deep convolutional neural networks using specialized hardware (Microsoft 2015)
- DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning (ASPLOS 2014)
- Cambricon: An Instruction Set Architecture for Neural Networks (ISCA 2016)
- Neurocube: A programmable Digital Neuromorphic Architecture with High-Density 3D memory (ISCA 2016)
- PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory Accelerators (ISCA 2016)
- Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators (ISCA 2016)
- Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing (ISCA 2016)
- EIE: Efficient Inference Engine on Compressed Deep Neural Network (ISCA 2016)
- Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks (ISCA 2016)