CS4223
Architectures for Deep Learning

Trevor E. Carlson

National University of Singapore

tcarlson@comp.nus.edu.sg

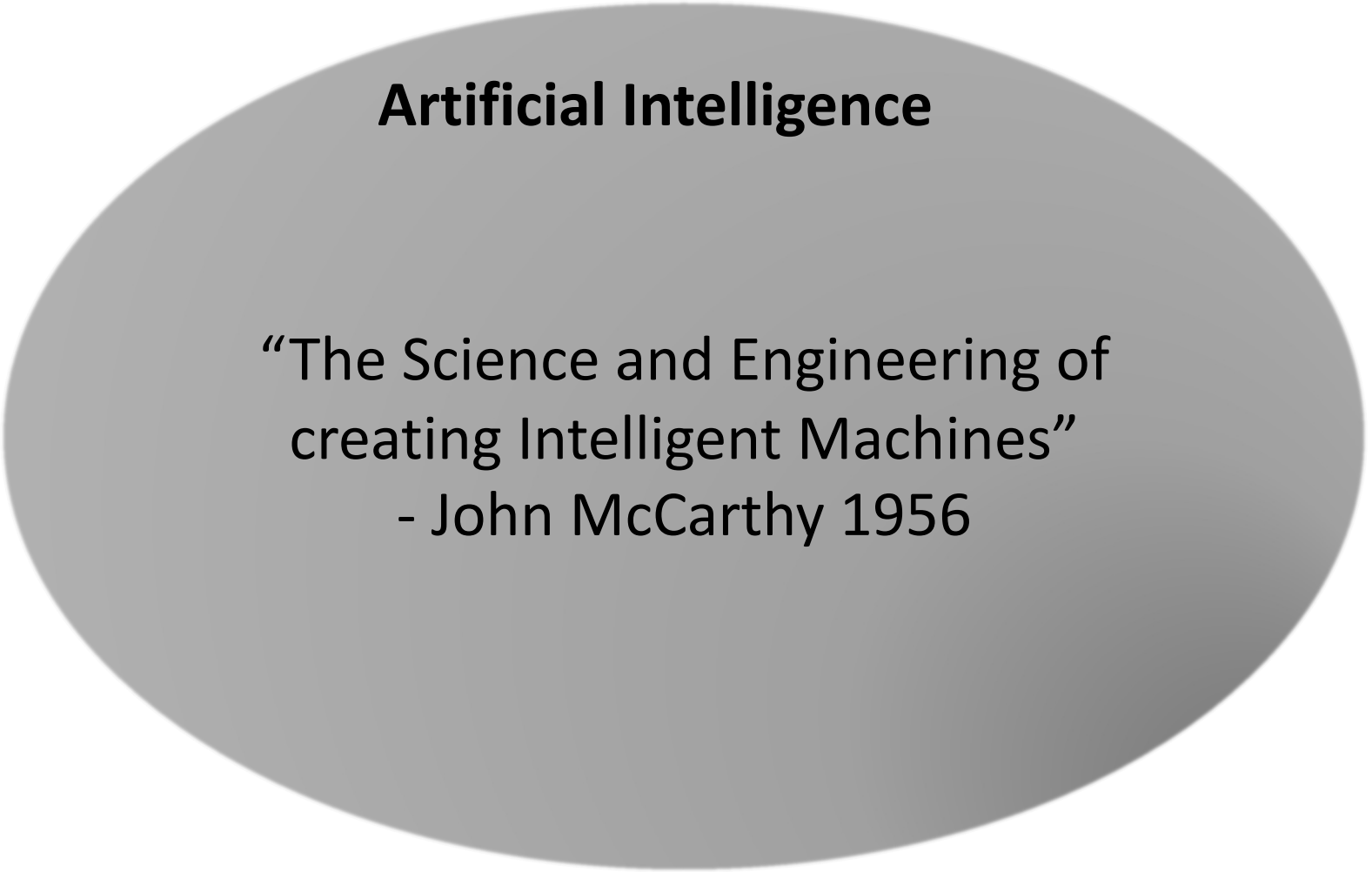
(Originally from Tulika Mitra)

Disclaimer

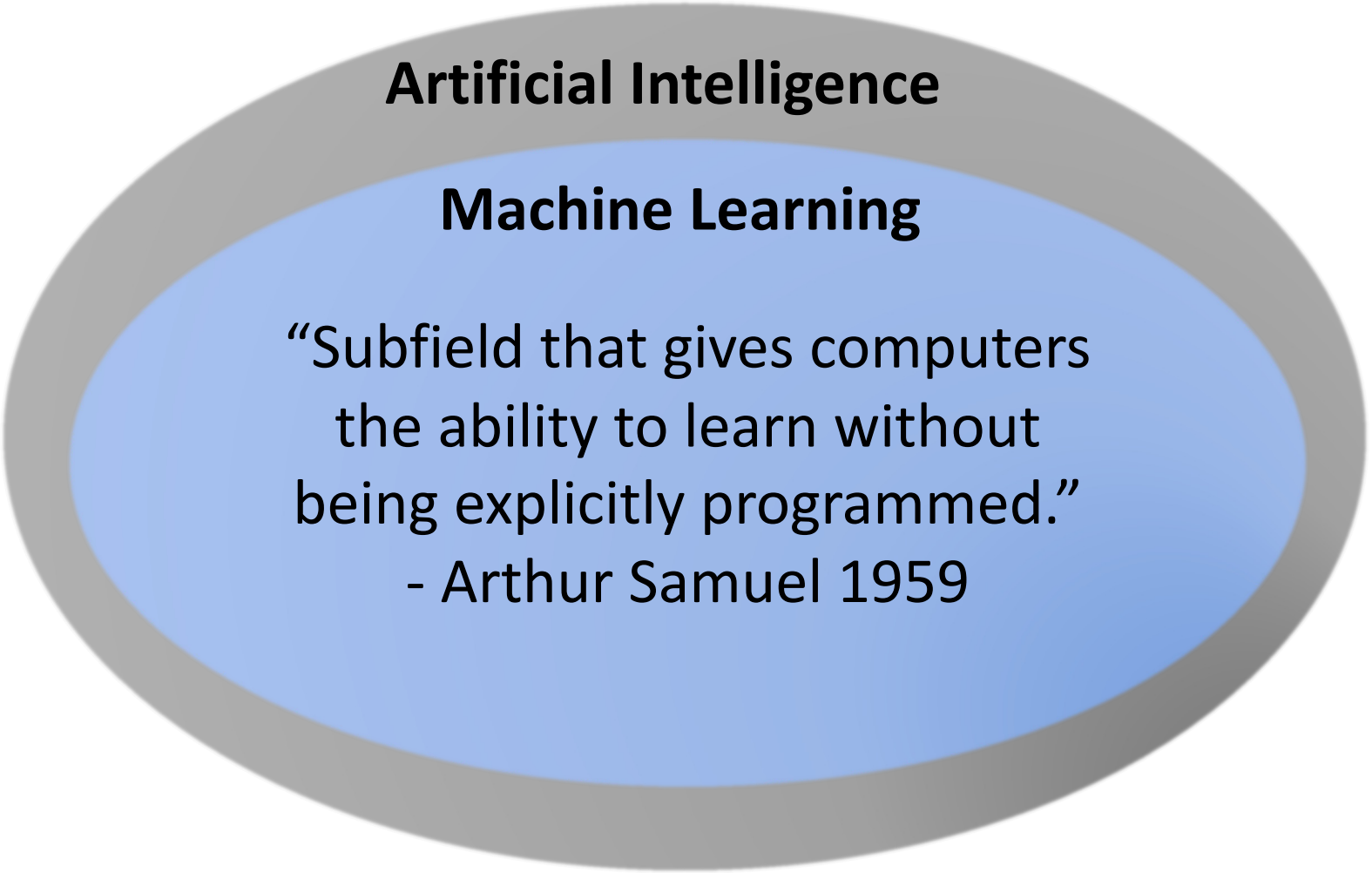
- Slides are either adapted or taken from the following:
  - David Patterson, "Evaluation of the Tensor Processing Unit: A Deep Neural Network Accelerator for the Datacenter“
  - Tutorial on Hardware Architectures for Deep Neural Networks, Joel Emer, Vivienne Sze, Yu-Hsin Chen

    http://eyeriss.mit.edu/tutorial.html
  - High-Performance Hardware for Machine Learning, William Dally, NIPS Tutorial
  - Deep Learning and GPUs: Intro and hands-on tutorial, Julie Bernauer

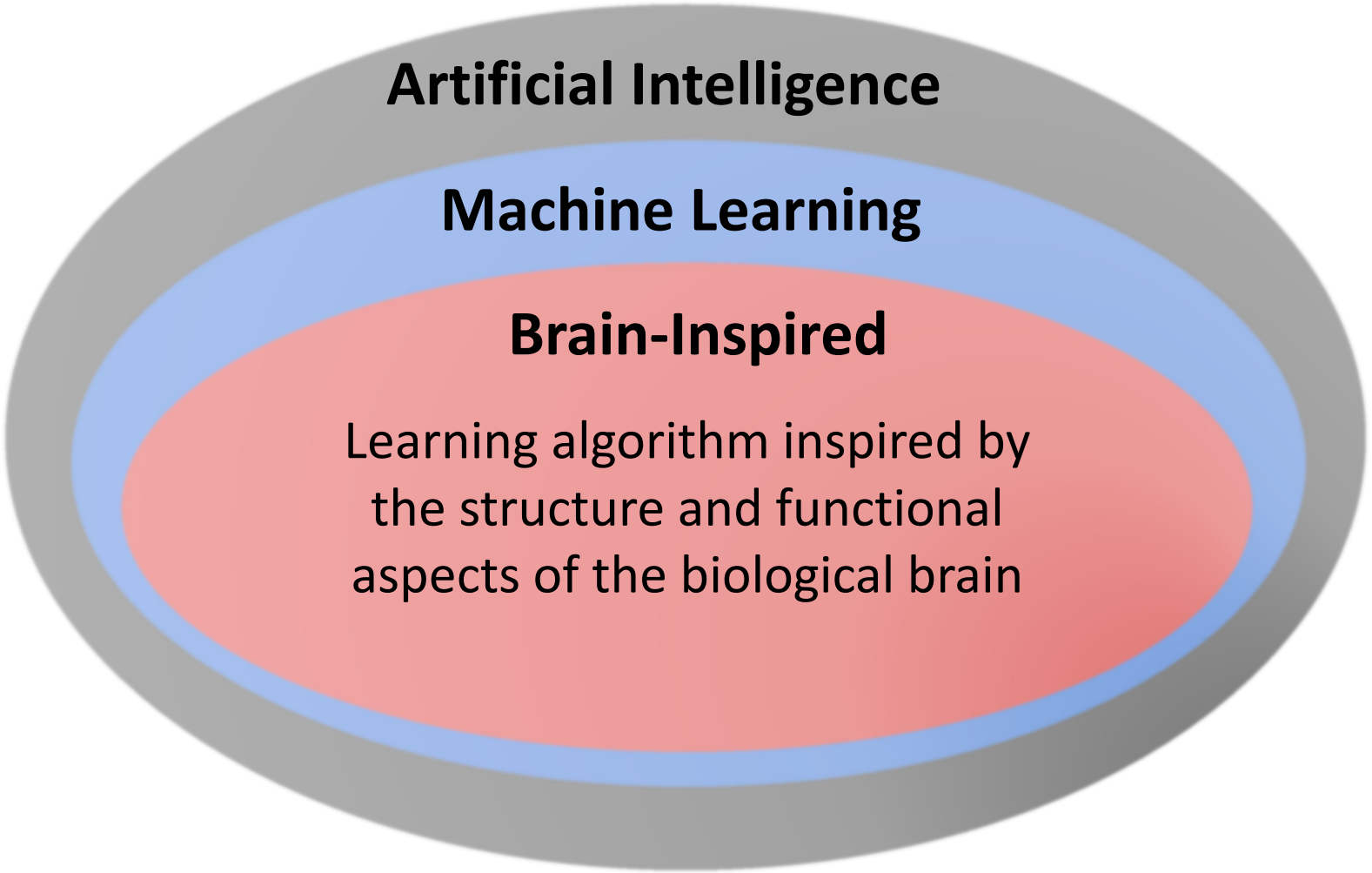
Artificial Intelligence (AI)



Machine Learning (ML)

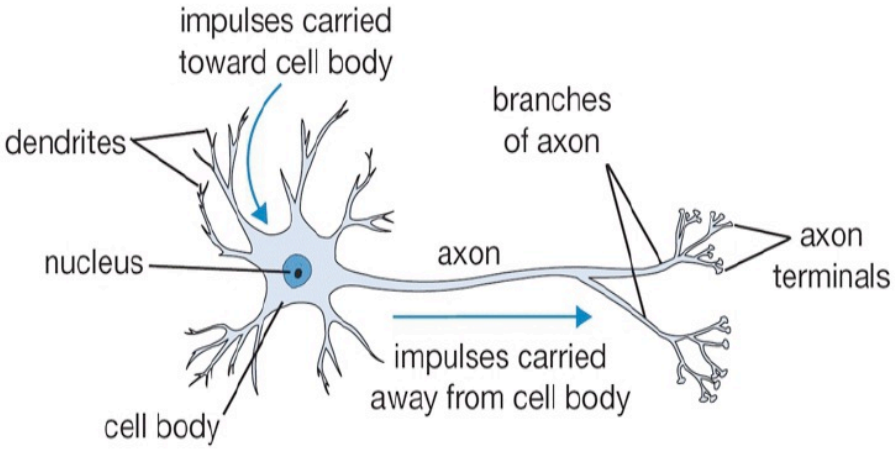


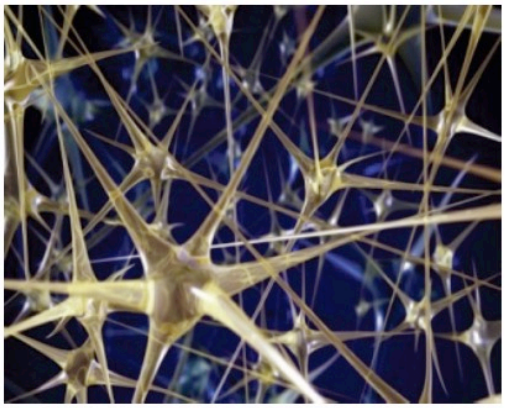
Brain-Inspired Machine Learning



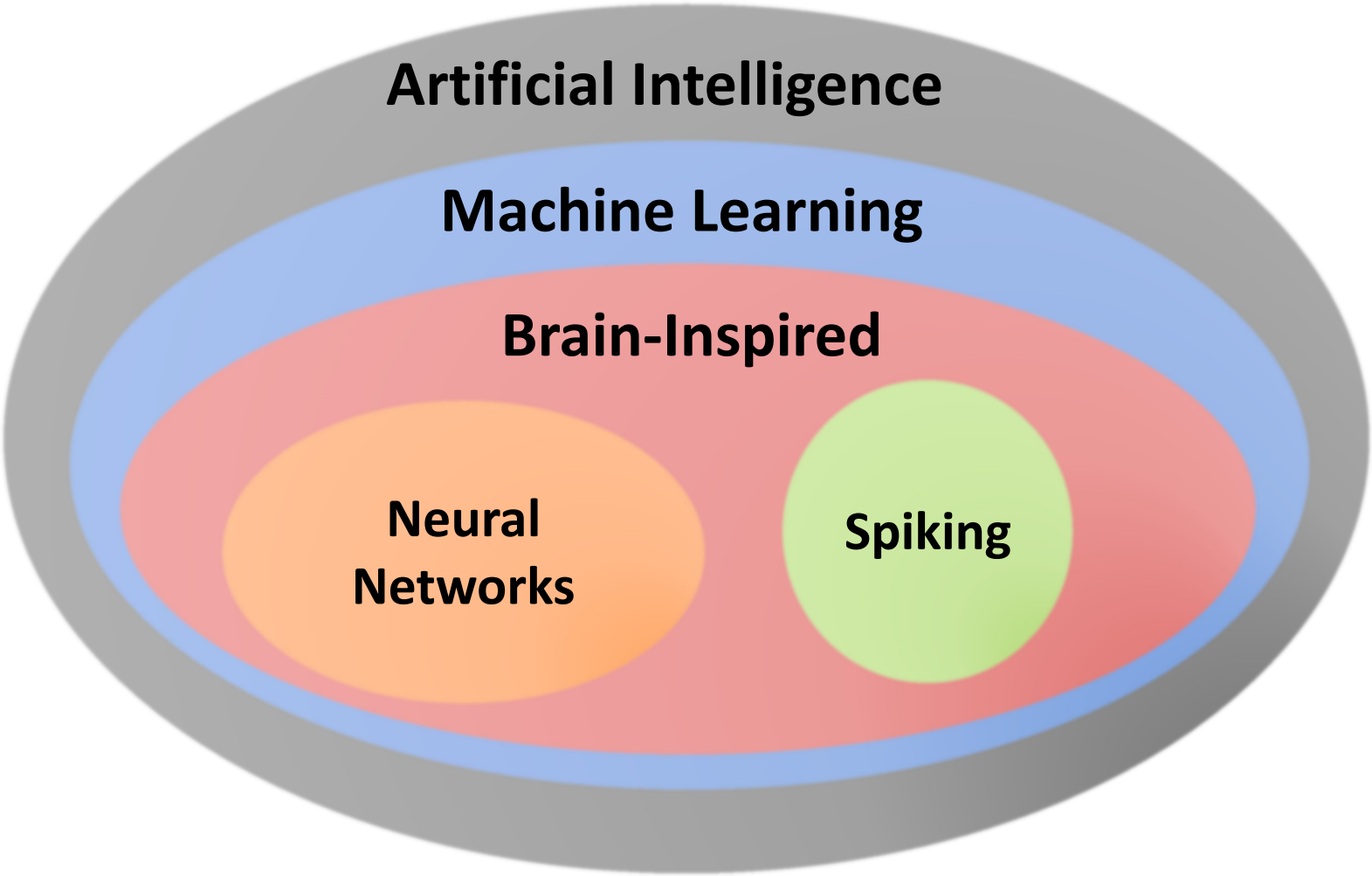
How does the brain work?

- Basic computational unit of brain is a neuron:
  - 86B neurons in the brain
- Neurons are connected with nearly $10^{14} – 10^{15}$ synapses
- Neurons receive input signal from dendrites and produces output signal along axon, which interact with the dendrites of other neurons via synaptic weights
- Synaptic weights are learnable and control influence strength

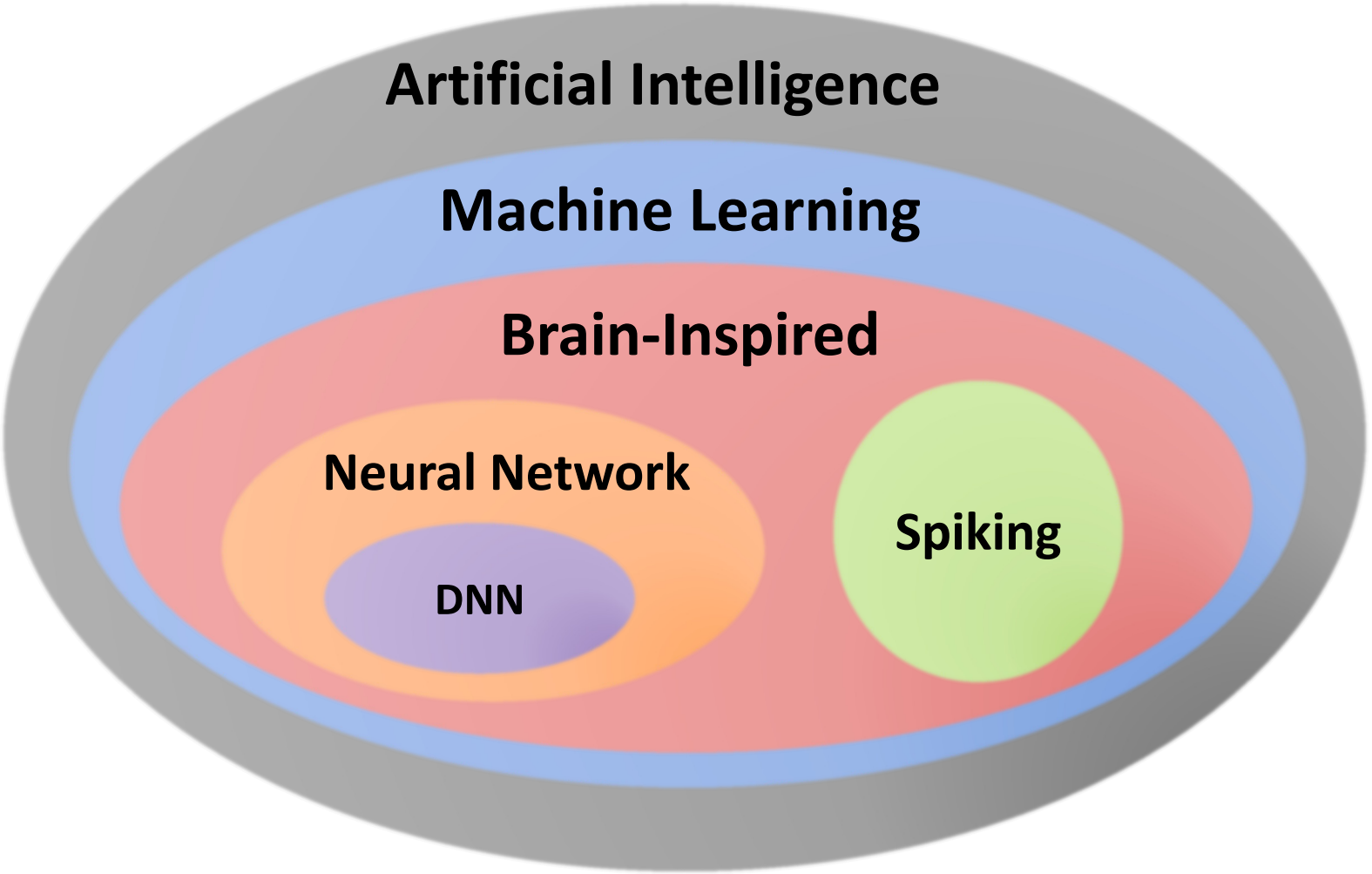




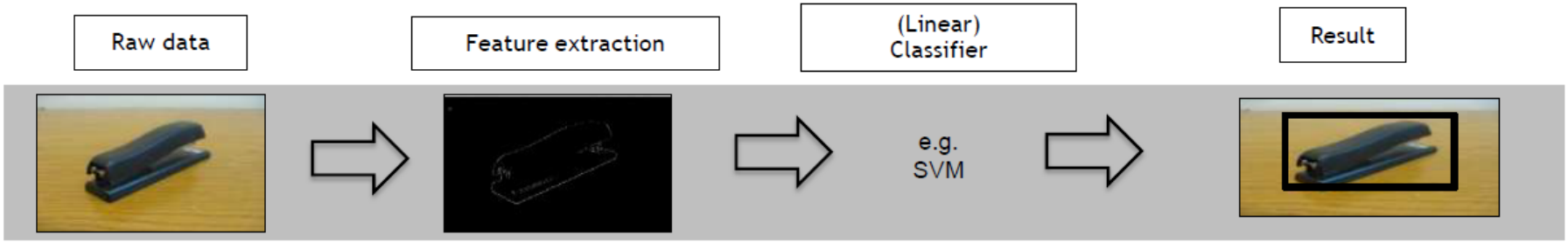
Neural Networks and Spiking



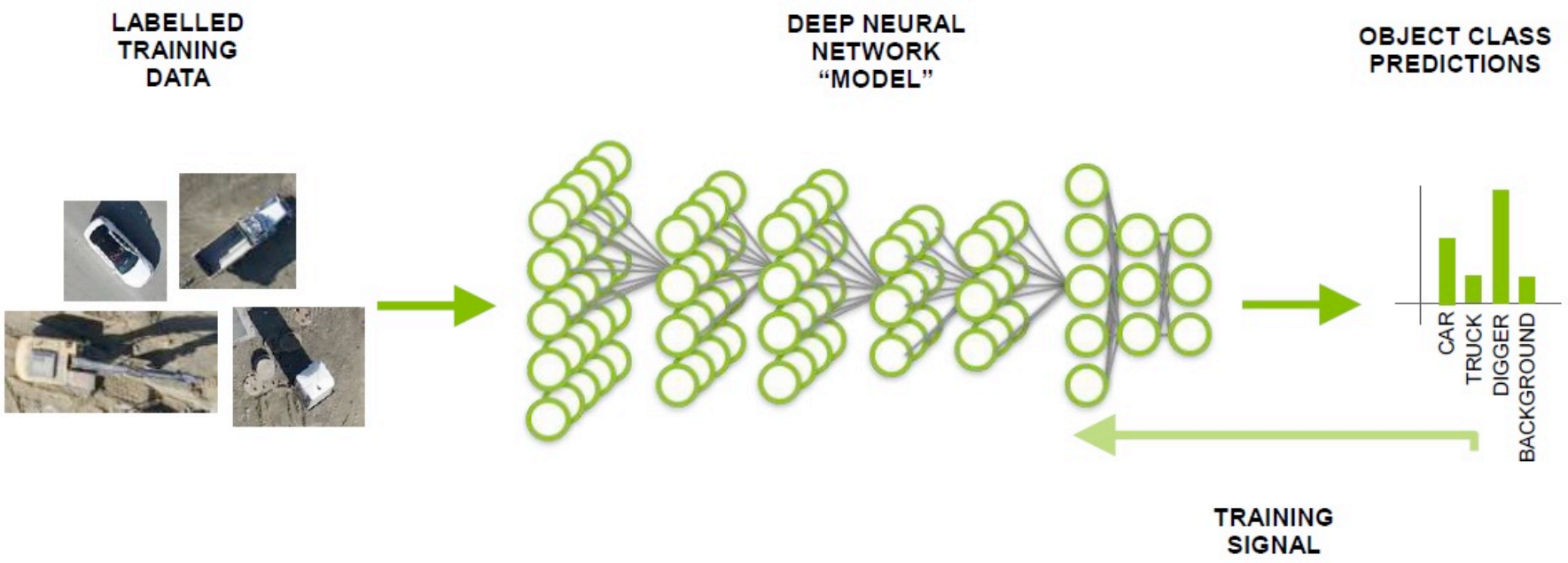
Deep Neural Network (DNN)



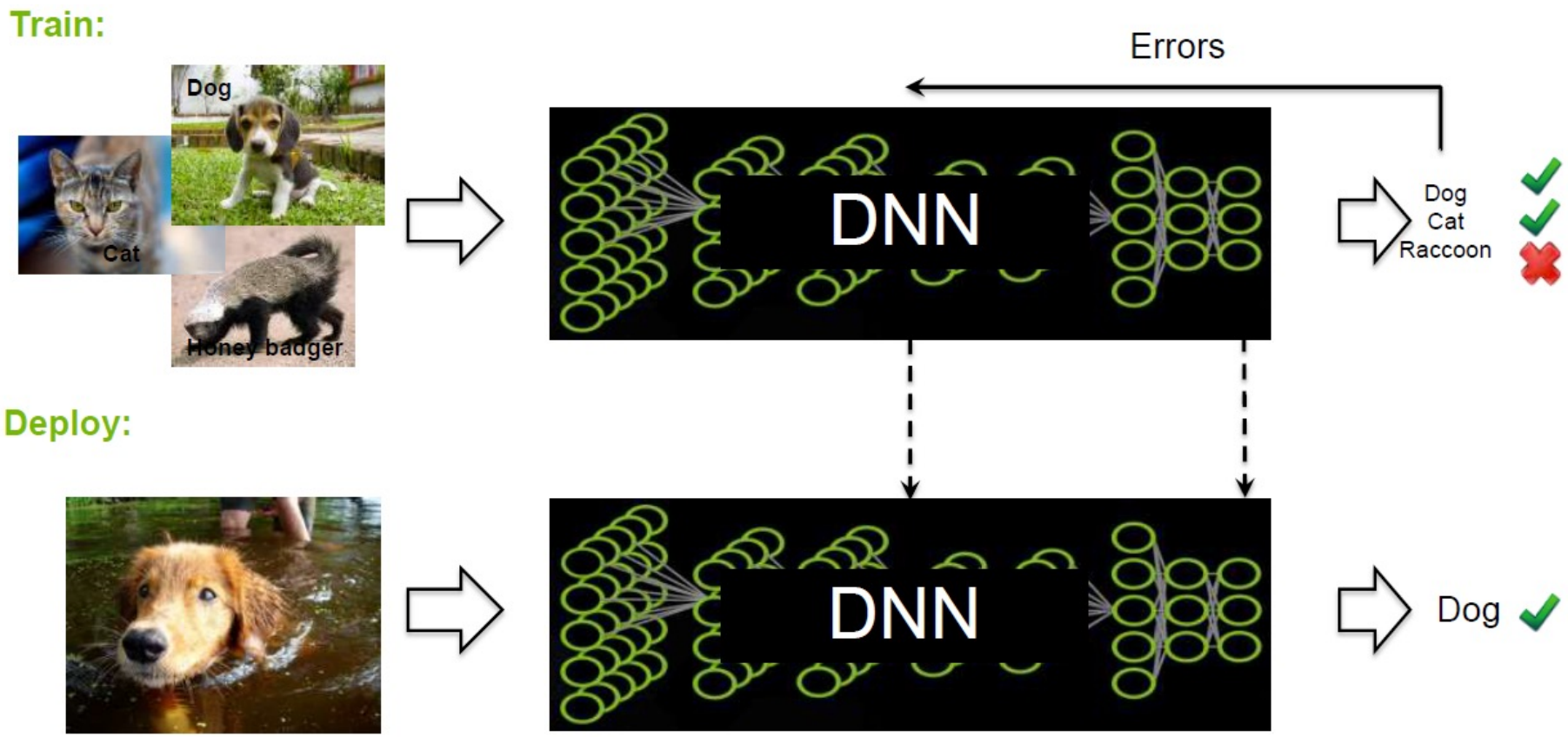
Traditional Machine Learning/ Computer Vision Approach



Deep Learning Approach



Deep Learning Approach

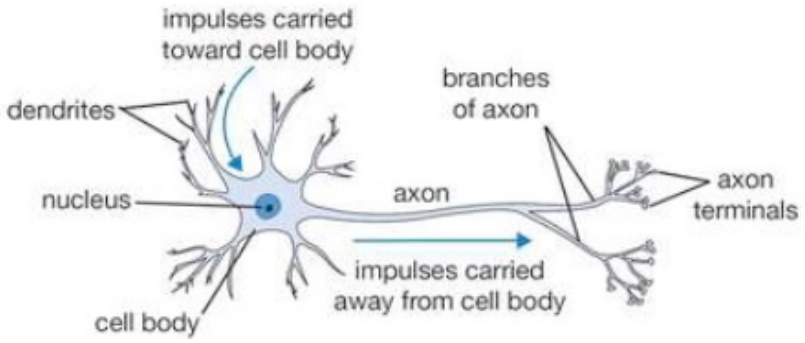


Artificial Neural Network (ANN)

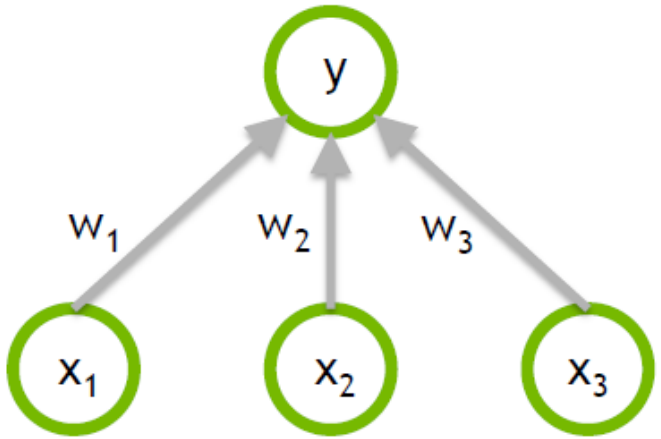
- A collection of simple, trainable mathematical units that collectively learn complex functions
- Given sufficient training data, an artificial neural network can approximate very complex functions mapping raw data to output decisions

Artificial Neuron

Biological neuron

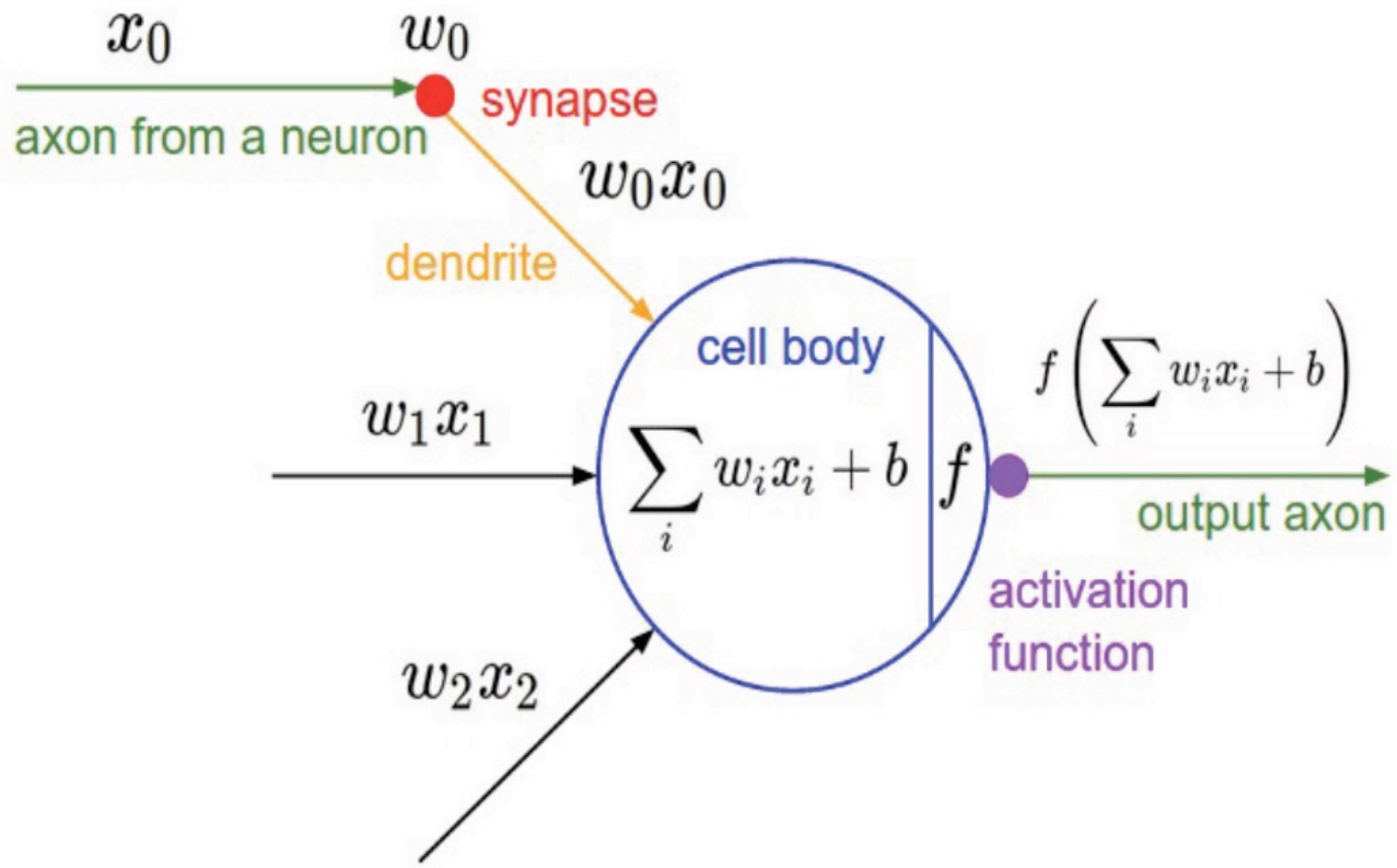


Artificial neuron



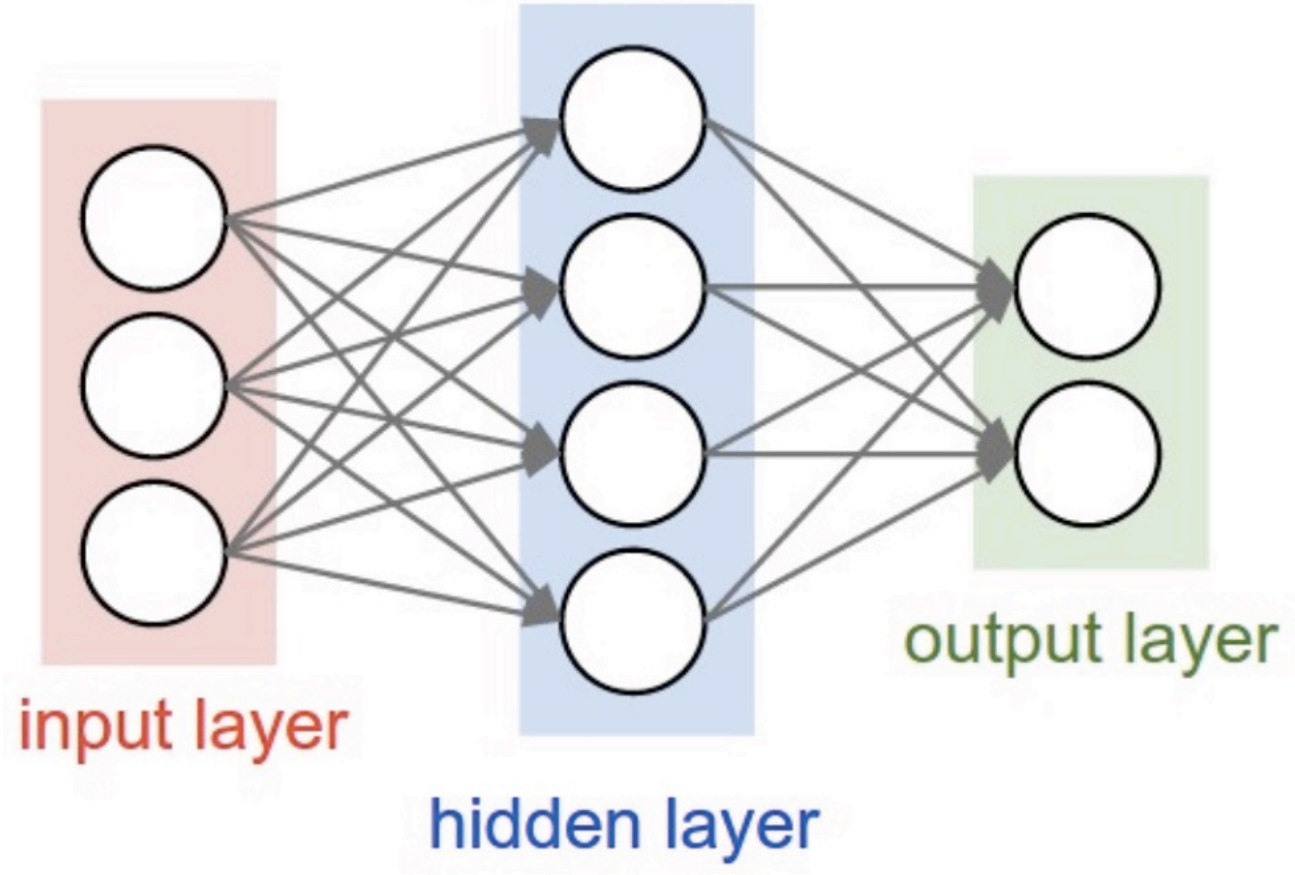
$$y=F(w_1x_1+w_2x_2+w_3x_3)$$

Artificial Neuron: Weighted Sum



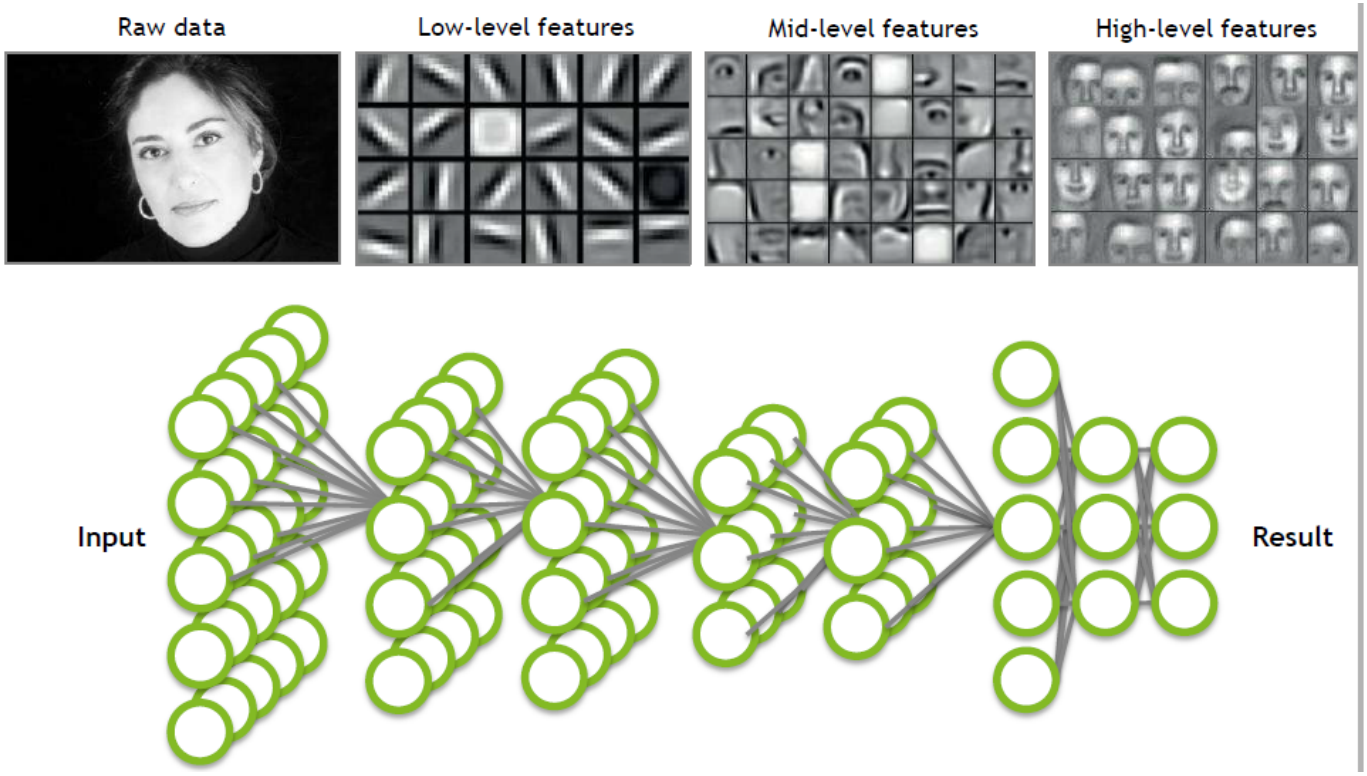
Neural Network

- Network of interconnected simple neurons



Deep Neural Network (DNN)

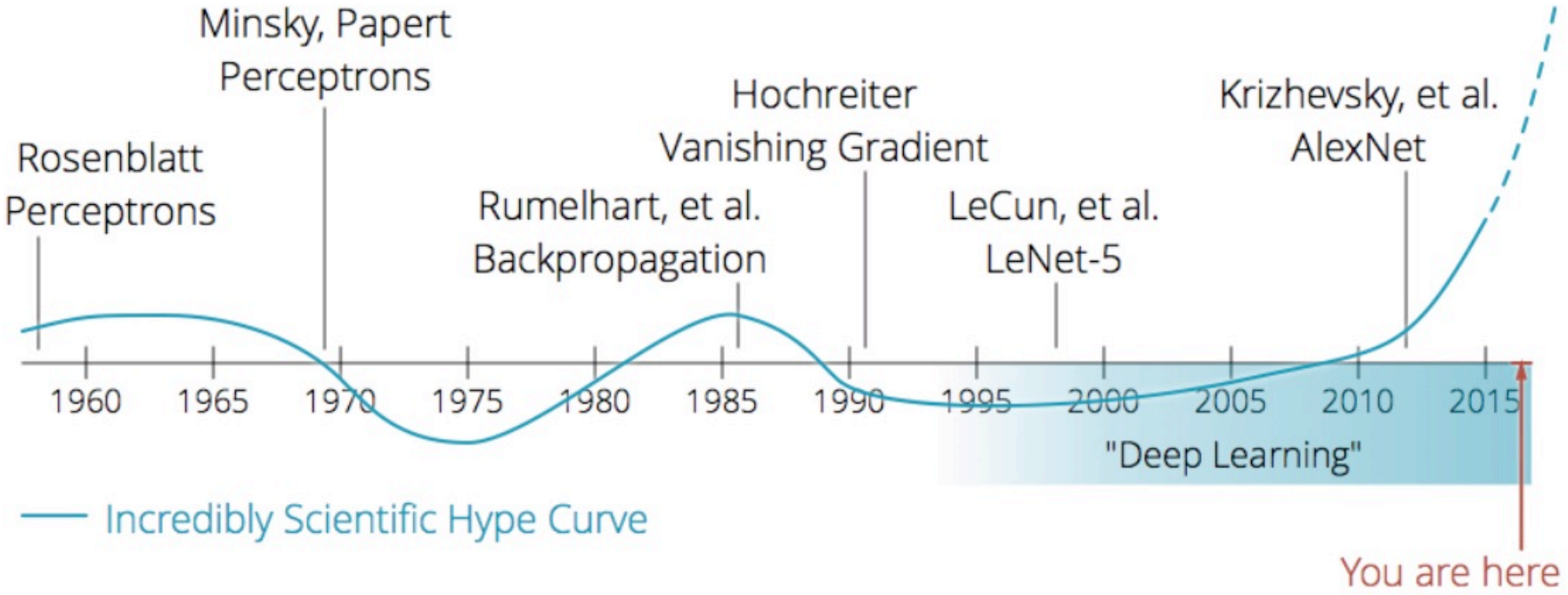
- Neural network consisting of multiple hidden layers



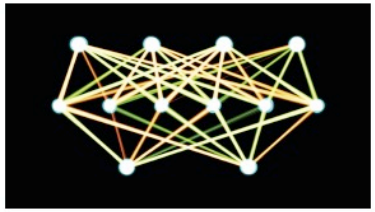
Application components:

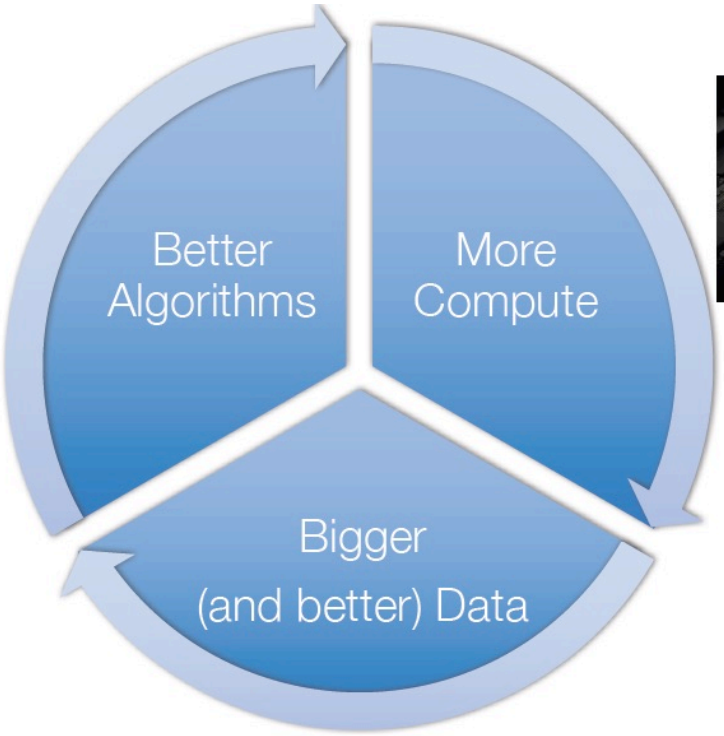
Task objective
e.g. Identify face
Training data
10-100M images
Network architecture
~10s-100s of layers
1B parameters
Learning algorithm
~30 Exaflops
1-30 GPU days

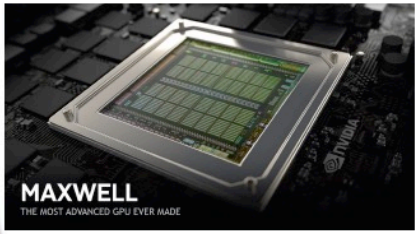
The Rise of Deep Learning



Today's Virtuous Cycle

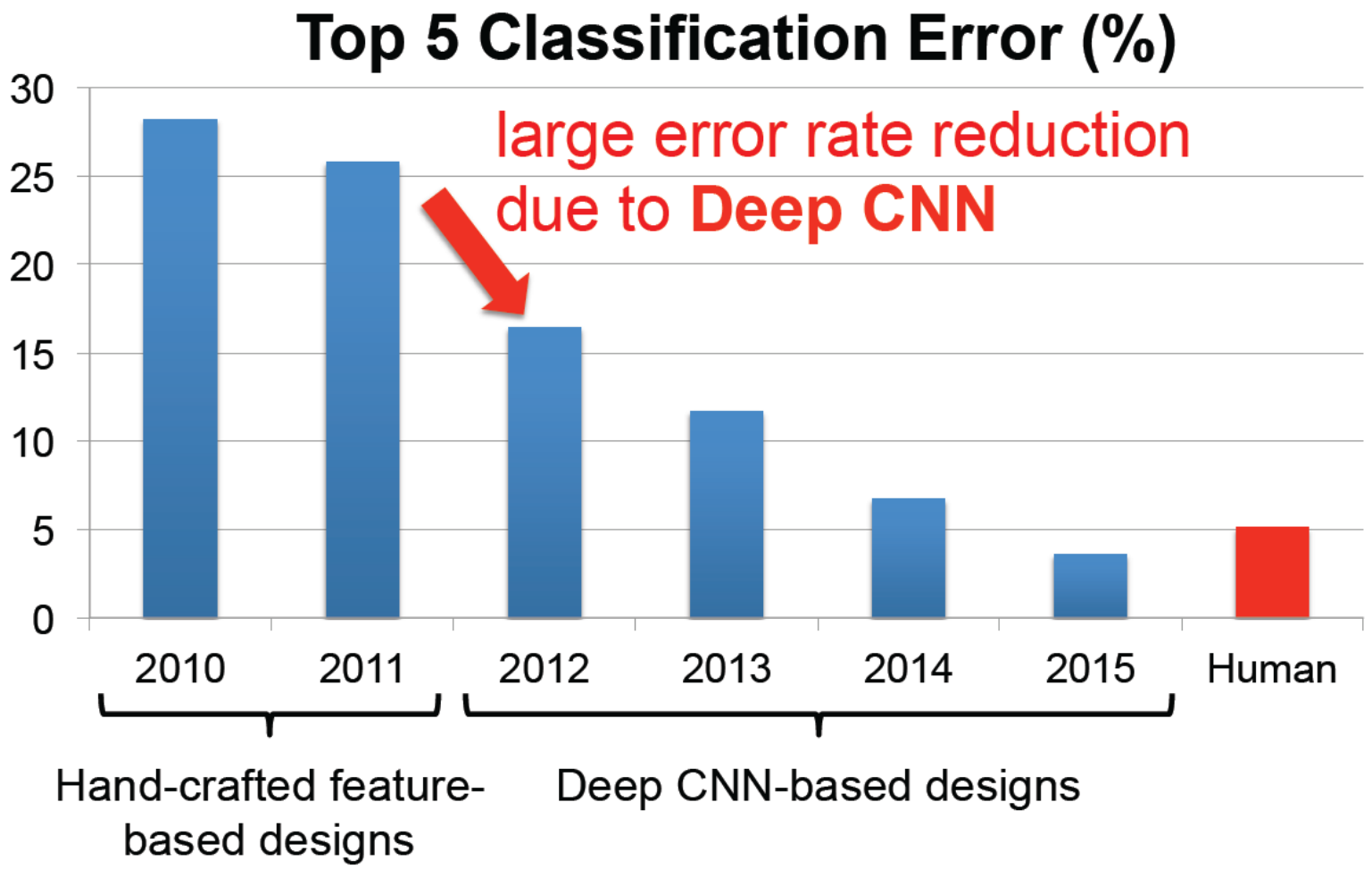






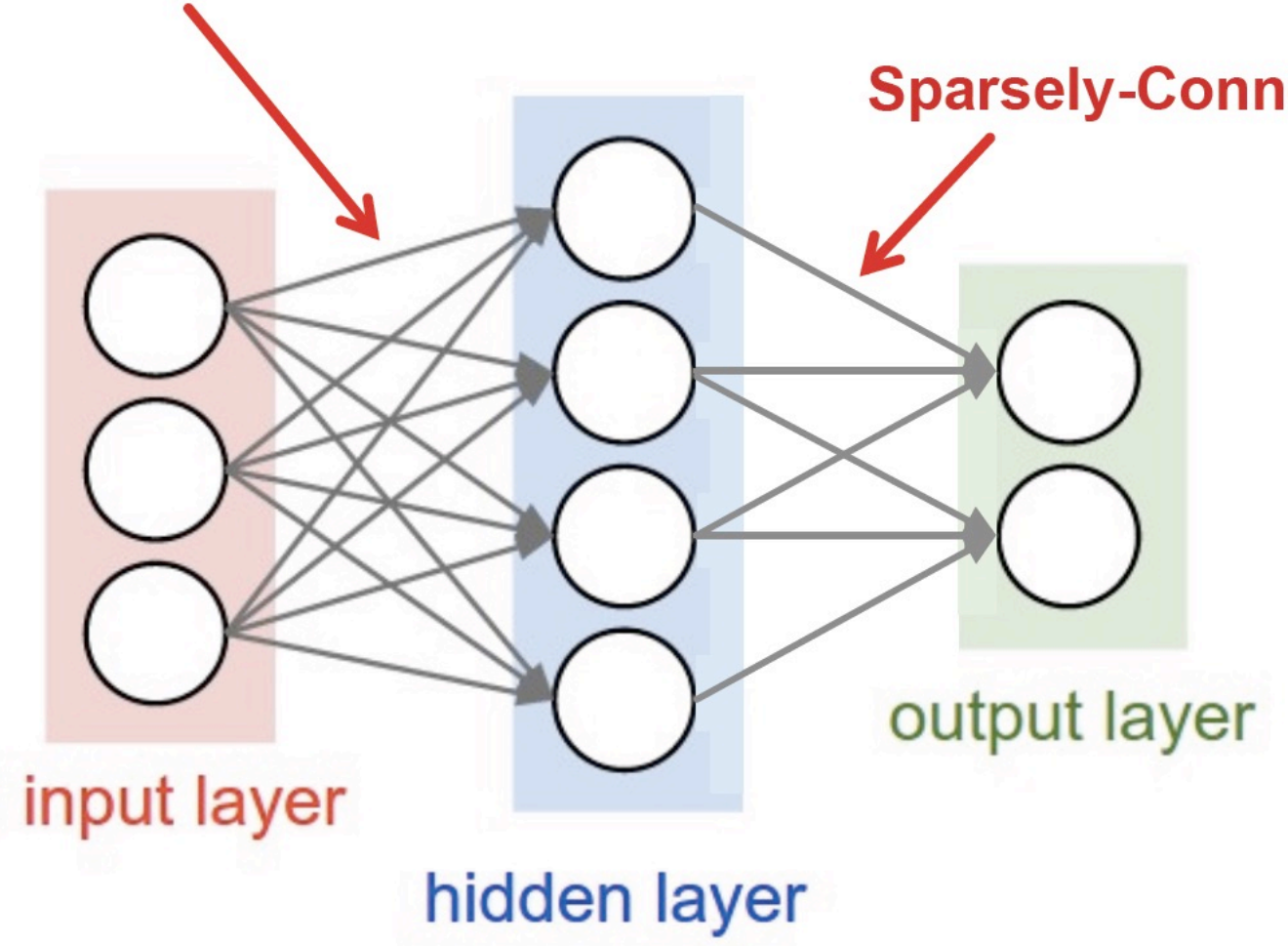
IMAGENET

ImageNet: Image Classification Task

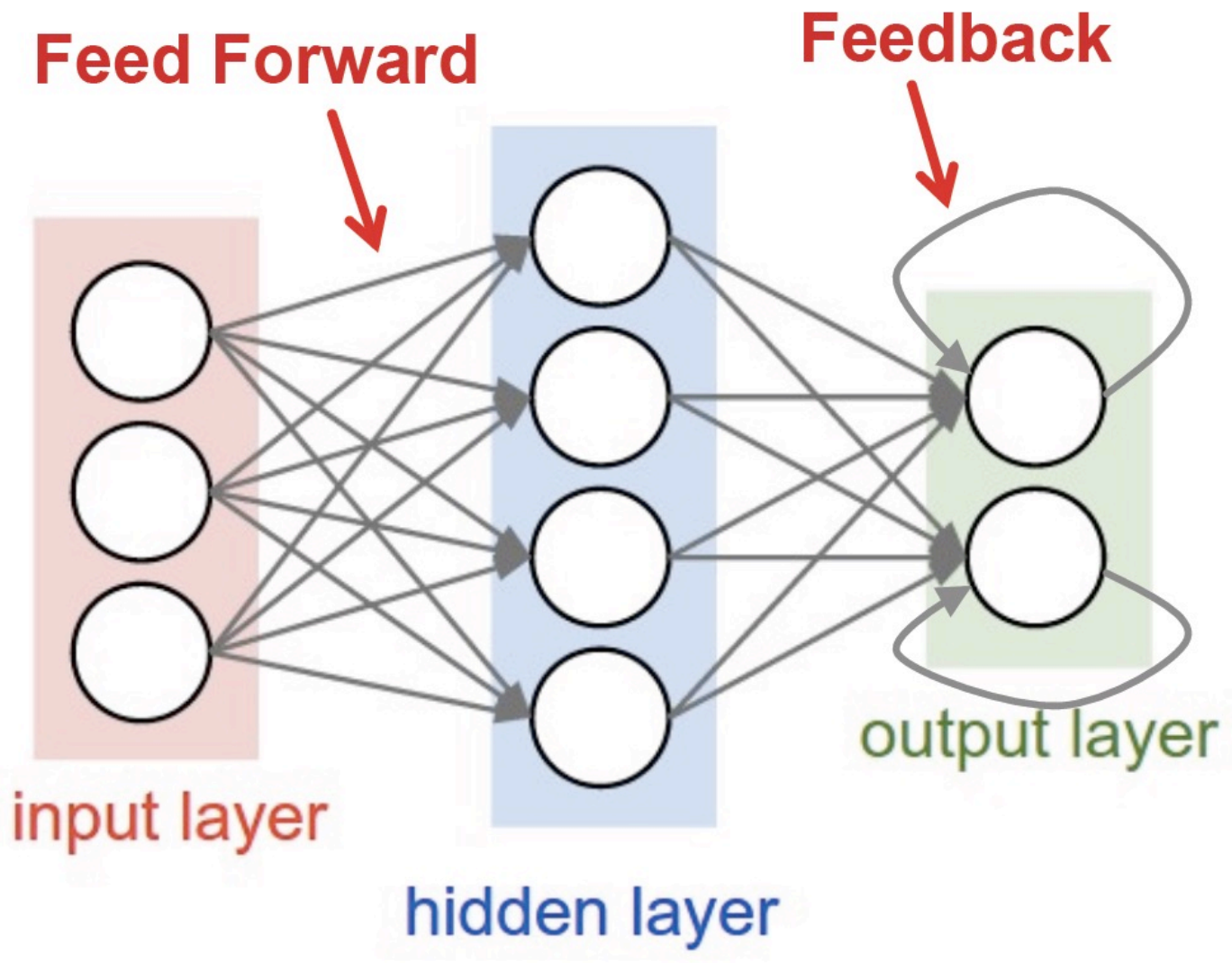


DNN Terminology 101

Fully-Connected: all i/p neurons connected to all o/p neurons



DNN Terminology 101



Popular Types of Neural Networks

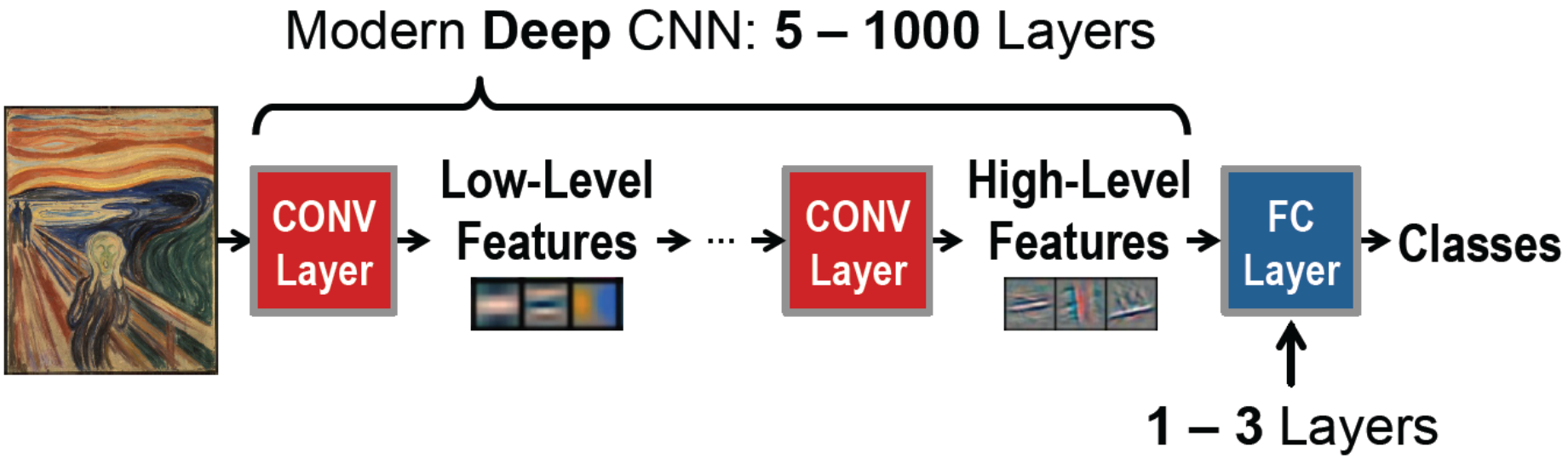
- **Fully connected NN**
  - Feed forward; also called multilayer perceptron (MLP)
- **Convolutional NN (DNN)**
  - Feed forward, sparsely connected with weight sharing
- **Recurrent NN (RNN)**
  - Feedback
- **Long Short-Term Memory (LSTM)**
  - Feedback + Storage
- **Residual Network**
  - Feed forward, skipping layers

Inference versus Training

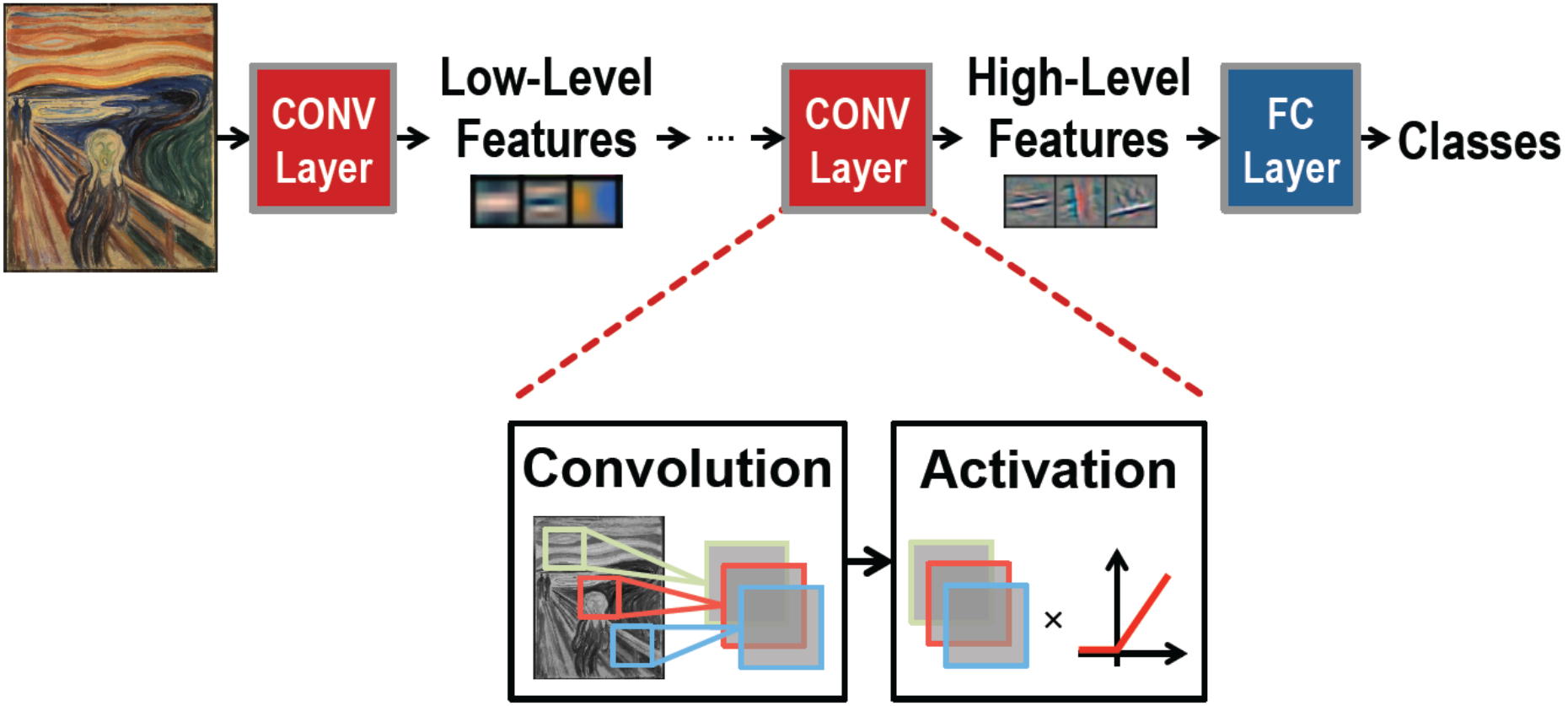
- **Training:** Determine weights
  - **Supervised:** Training set has inputs and outputs, labeled
  - **Unsupervised:** Training set in un-labled
  - **Semi-supervised:** Training set is partially labeled
  - **Reinforcement:** output assessed via rewards and punishments

- **Inference:** Apply weights to determine output

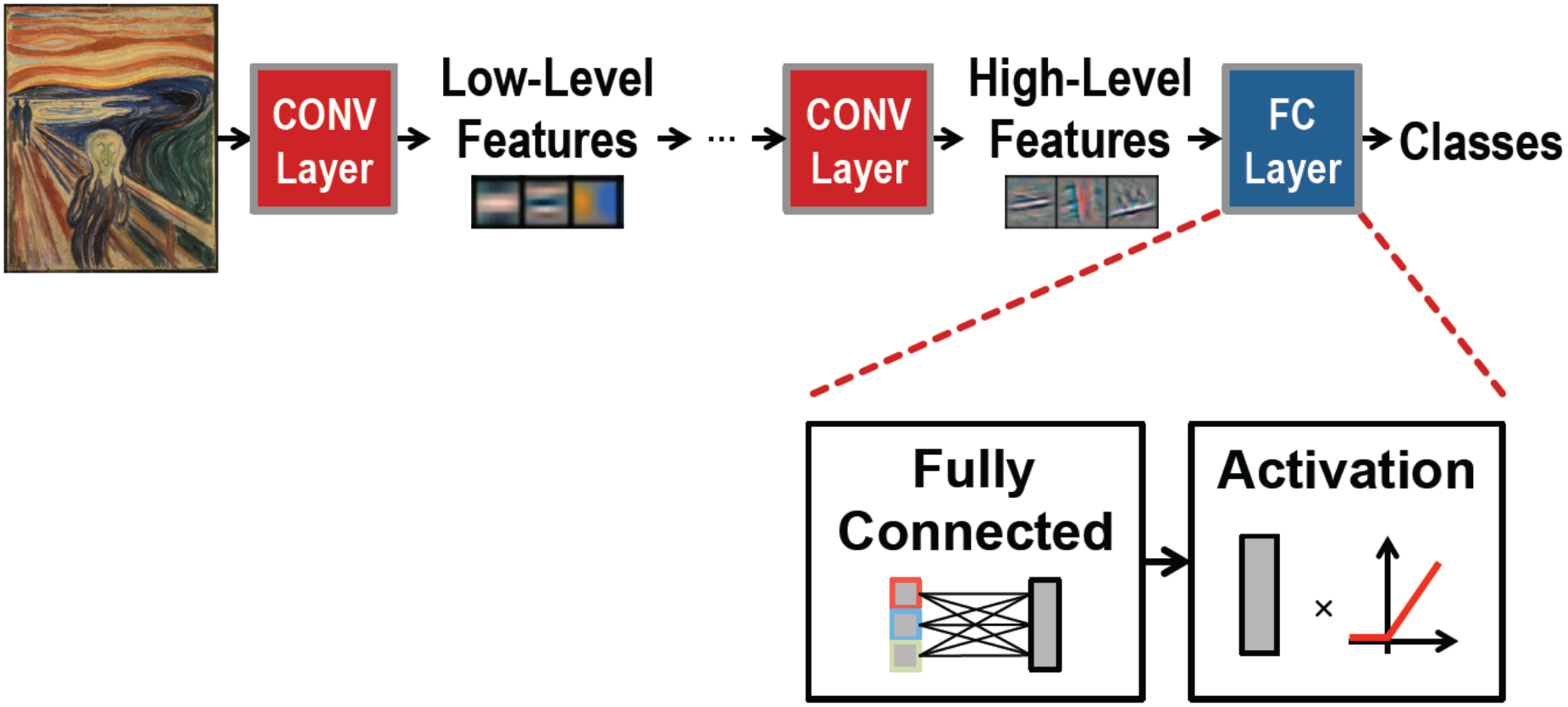
Convolutional Neural Networks (CNN)



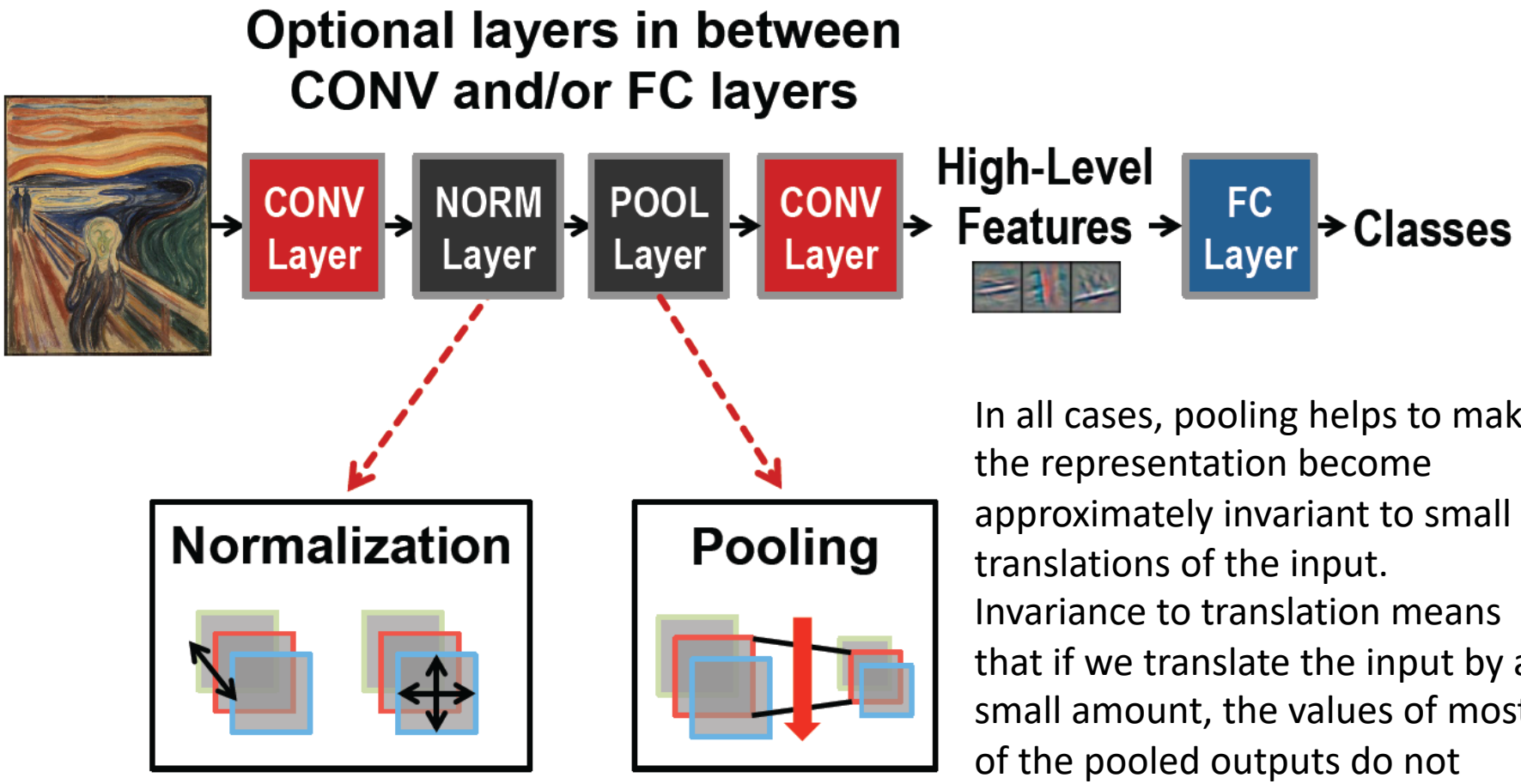
Convolutional Neural Networks (CNN)



Convolutional Neural Networks (CNN)



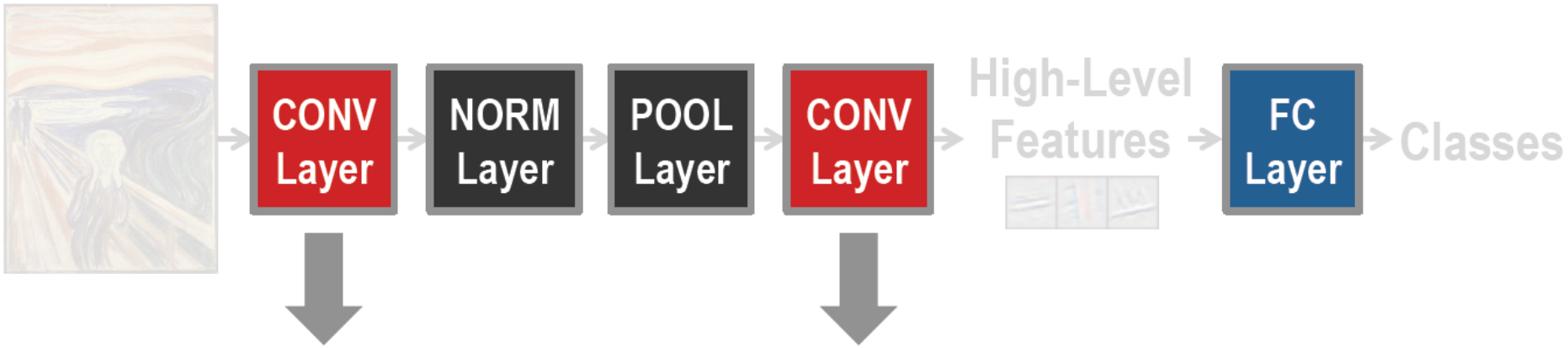
Convolutional Neural Networks (CNN)

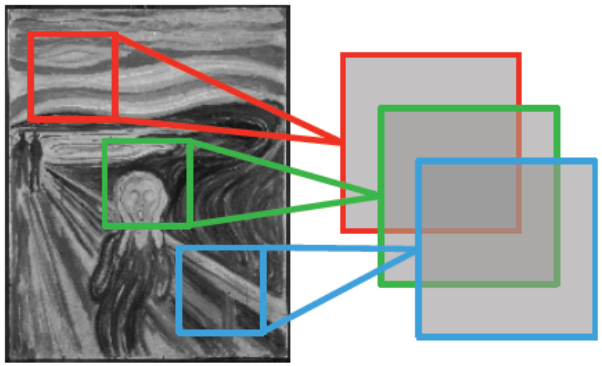


In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.

Page 342, Deep Learning, Goodfellow, Bengio, Courville 2016.

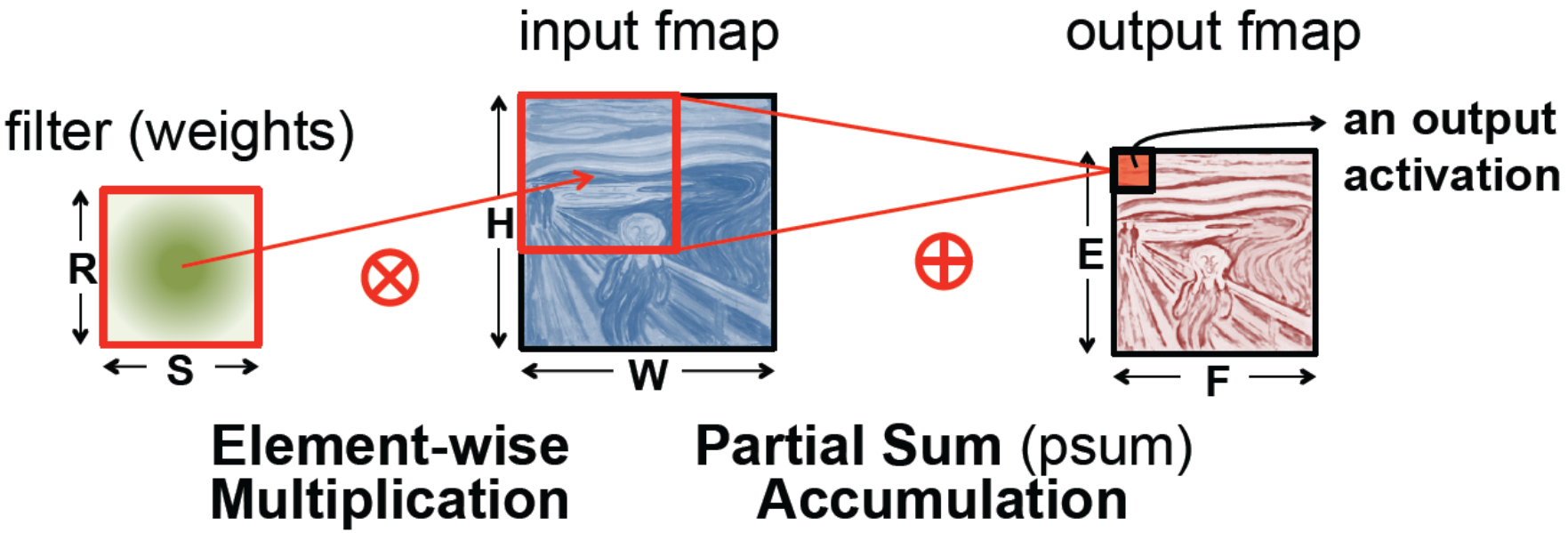
Convolutional Neural Networks (CNN)



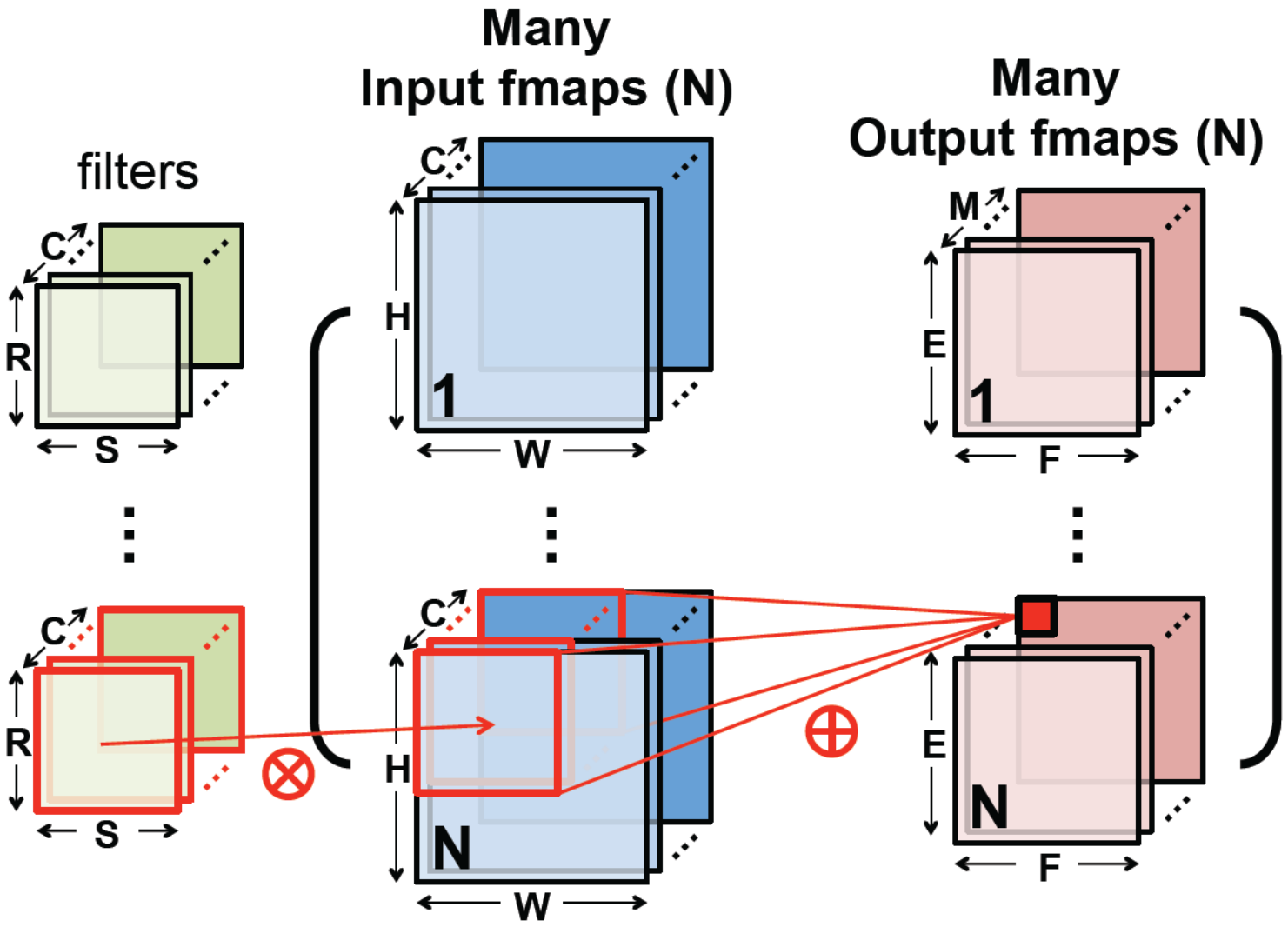


Convolutions account for more than 90% of overall computation, dominating **runtime** and **energy consumption**

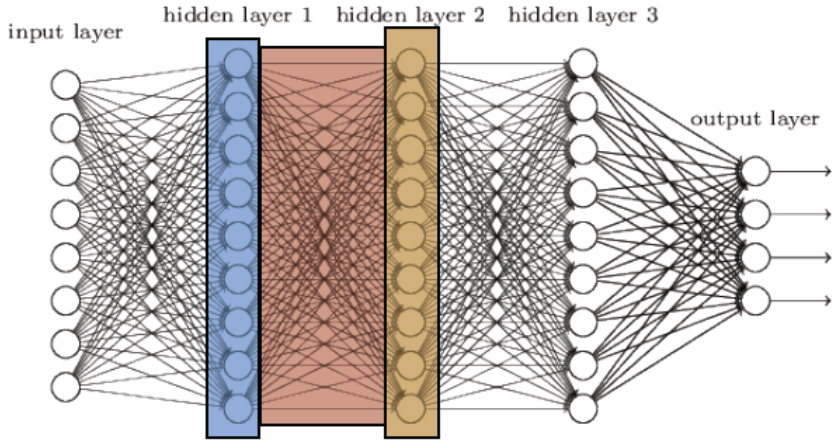
Convolution (CONV) Layer



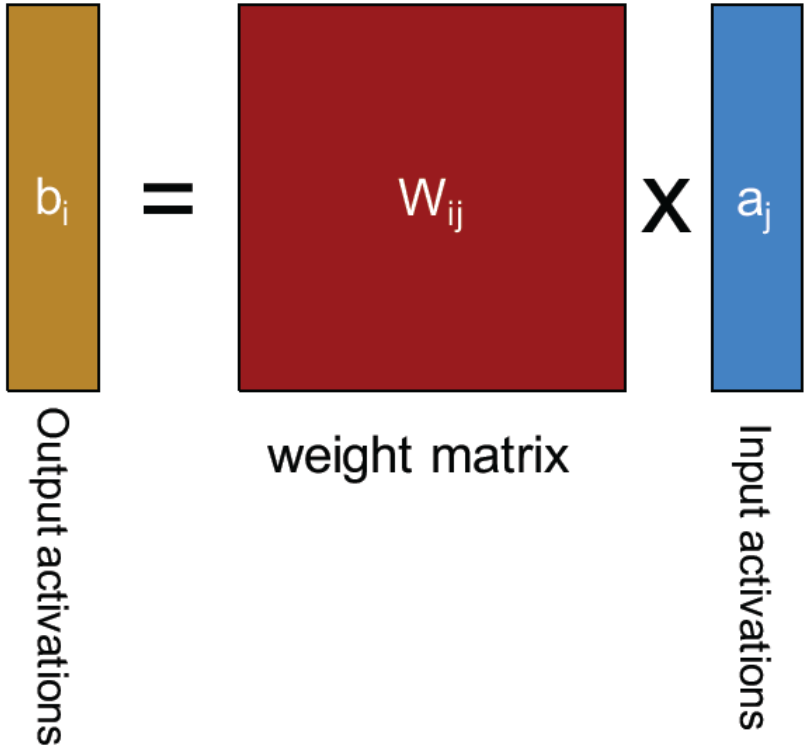
Convolution (CONV) Layer



Key Operation: Matrix-Vector Multiply

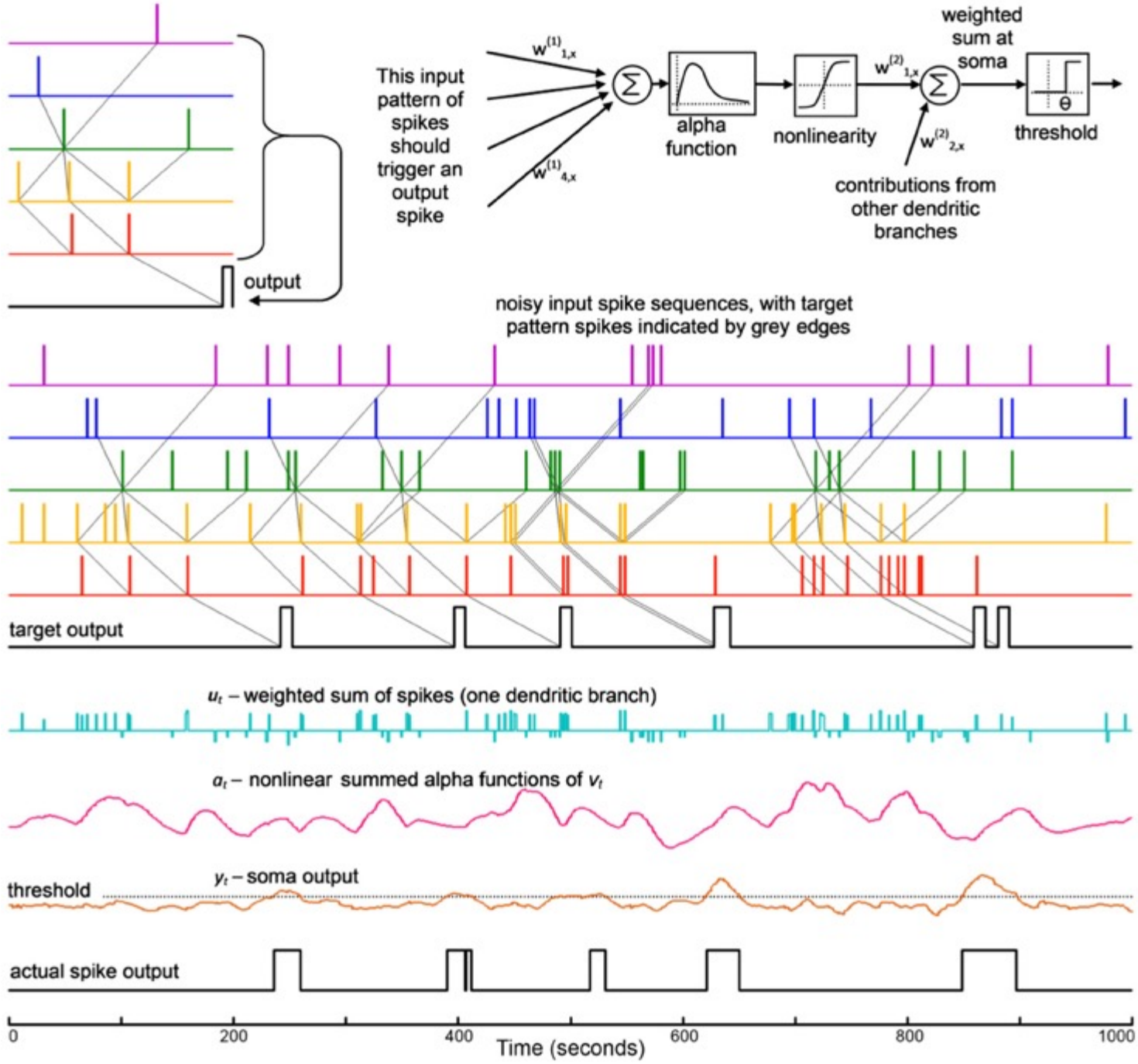


Repeat for each layer



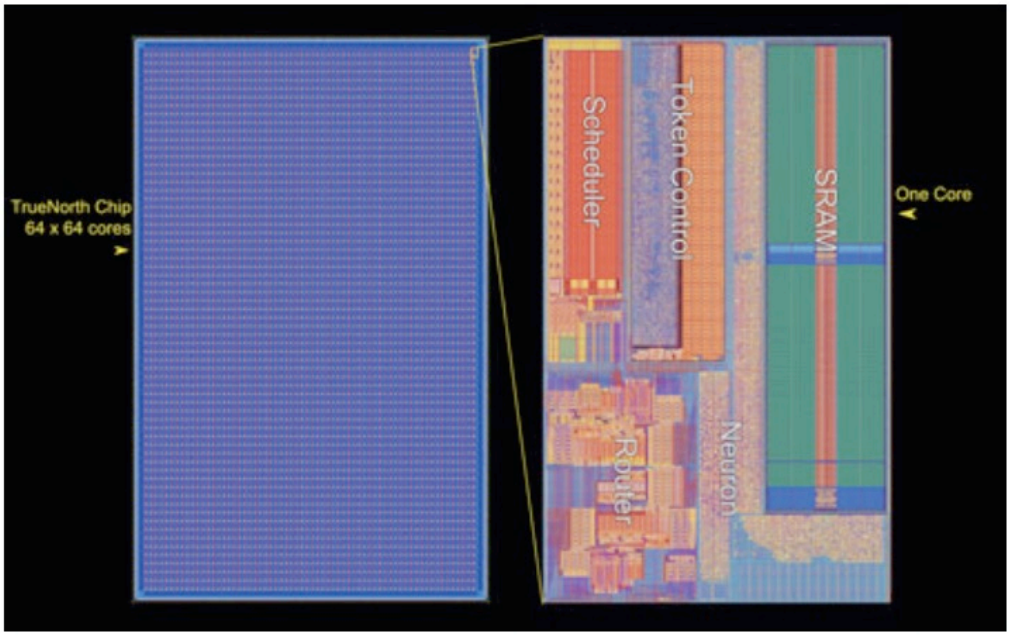
Spiking Neural Network (SNN)

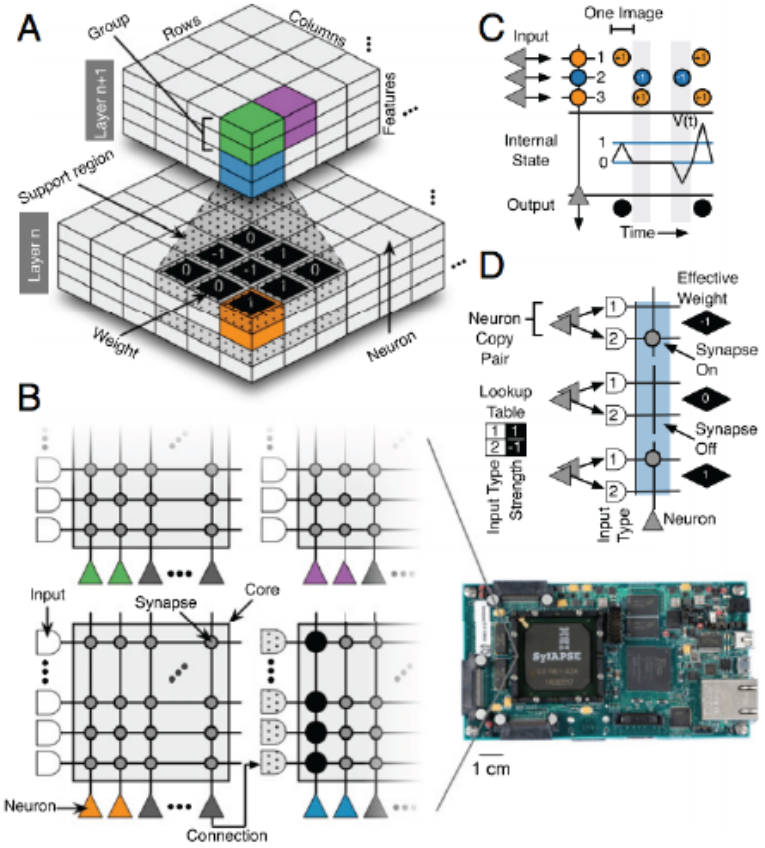
- In addition to neuronal and synaptic states, SNN incorporates the concept of **time**
- Neurons in SNN do not fire at each propagation cycle, but rather fire only when a membrane potential reaches the threshold
- When a neuron fires, the generated signal travel to other neurons, which in turn changes their potentials


This input pattern of spikes should trigger an output spike
$w^{(1)}_{1,x}$
$w^{(1)}_{4,x}$
alpha function
nonlinearity
$w^{(2)}_{1,x}$
weighted sum at soma
$w^{(2)}_{2,x}$
contributions from other dendritic branches
θ
threshold
output
noisy input spike sequences, with target pattern spikes indicated by grey edges
target output
u_t – weighted sum of spikes (one dendritic branch)
a_t – nonlinear summed alpha functions of v_t
threshold
y_t – soma output
actual spike output
0
200
400
600
800
1000
Time (seconds)

Spiking Neural Network (SNN)

- Brain-inspired
- Integrate and Fire
- Example: IBM TrueNorth





Spiking Neural Network (SNN)

- New neuromorphic techniques have shown significant progress recently
  - ANN <-> SNN equivalence
  - Time-to-First-Spike Implementations
  - Potential for low-power designs, co-designing the hardware, the network and the application

Why computer architecture for deep learning?

- Larger data sets and models lead to better accuracy but also increases computation time
- Progress in deep learning is limited by how fast the networks can be computed
- Application of deep learning to low latency inference problems (e.g., pedestrian detection in self-driving car) is limited by how fast an image can be classified

Why computer architecture for deep learning?

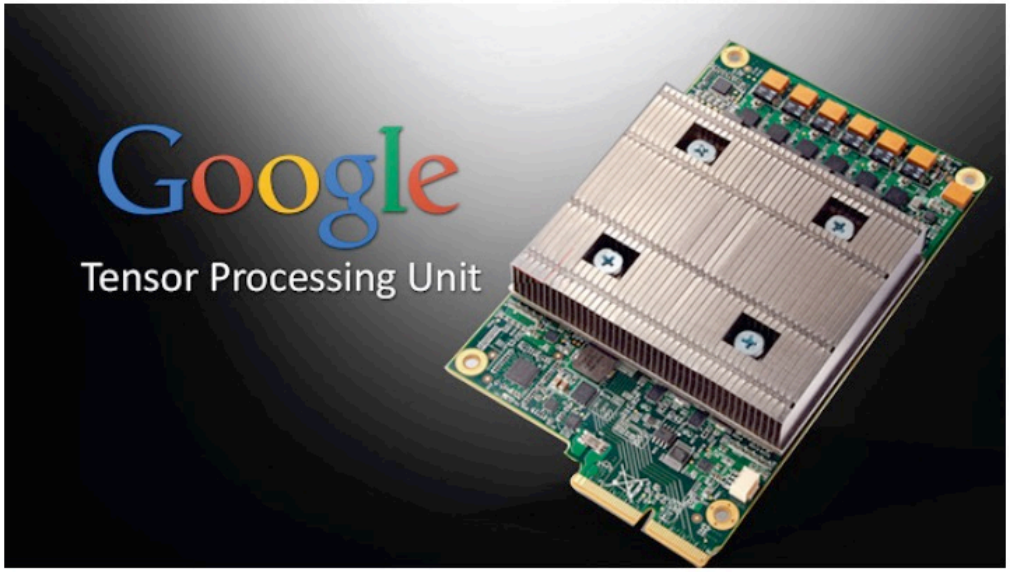
"Today the job of training machine learning models is limited by compute, if we had faster processors we'd run bigger models…in practice we train on a reasonable subset of data that can finish in a matter of months. We could use improvements of several orders of magnitude – 100x or greater."

- Greg Diamos, Senior Researcher, SVAIL, Baidu
(EE Times, Sep 2016)

Why computer architecture for deep learning?

"By May, the (Google) Brain team understood that the only way they were ever going to make the system fast enough to implement as a product was if they could run it on T.P.U.s, the special-purpose chips that (Jeff) Dean had called for. As (Zhifeng) Chen put it: "We did not even know if the code would work. But we did know that without T.P.U.s, it definitely wasn't going to work."

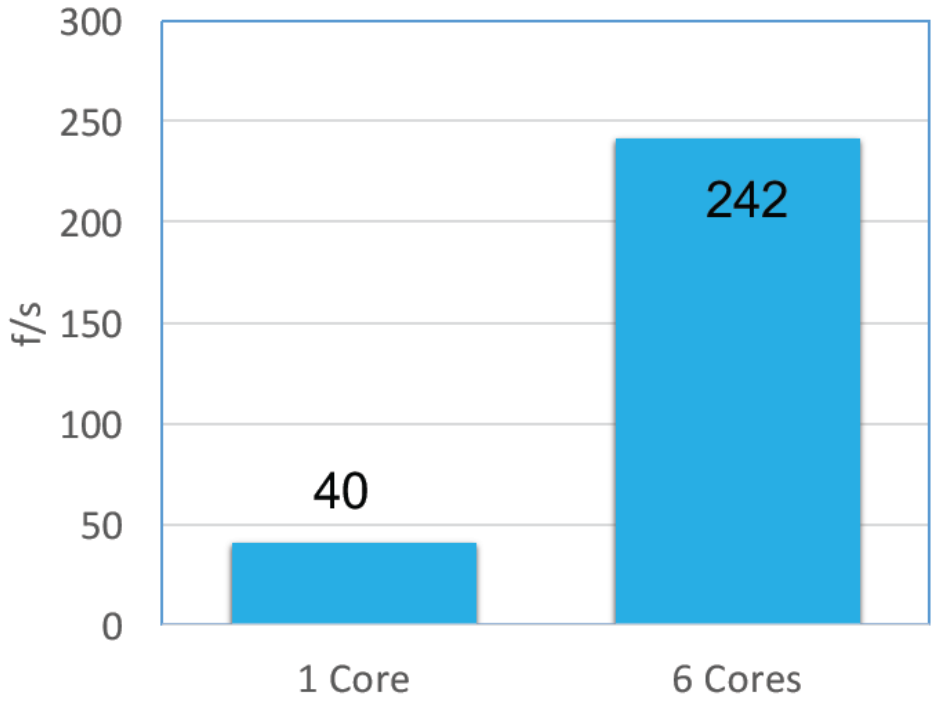
- The Great A.I. Awakening, New York Times, Dec 2016



Baseline Performance on Single Core: Xeon E5-2698

- AlexNet --- inference
- 30 frames per second
- 3.2 frames / Joule
- Most operations on AVX (SIMD) unit

CPU Parallelism
Core i7: 1 core versus 6 cores



Intel Knights Landing

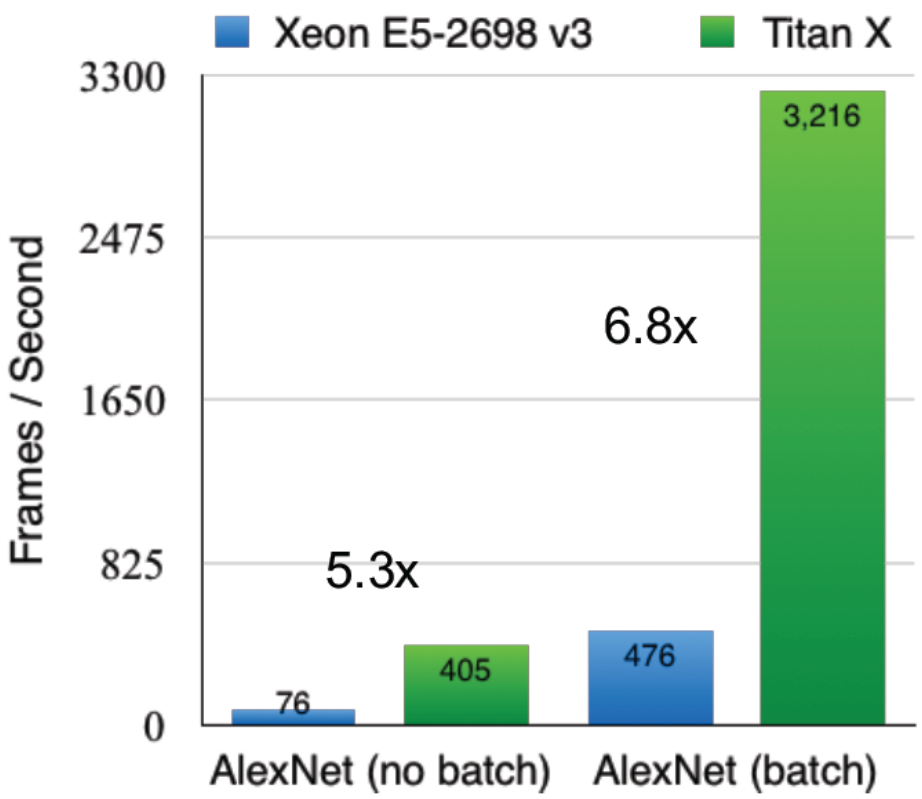
- 7 TFLOPS FP32
- 16GB DRAM: 400 GB/sec
- 245W TDP
- 29 GFLOPS/W FP32
- 14nm process

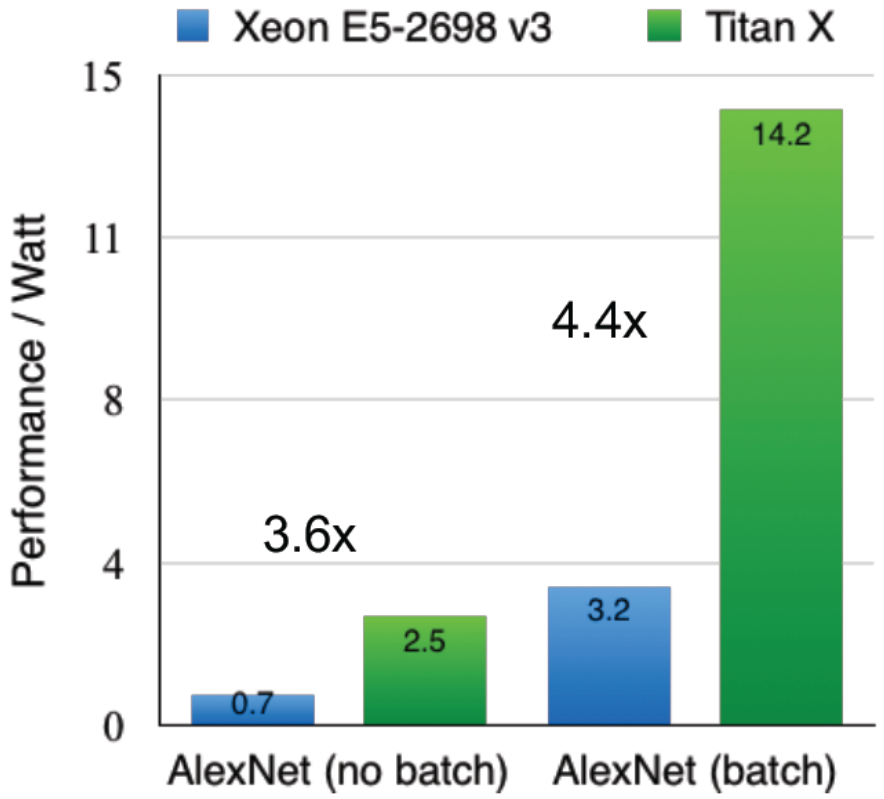
- Knights Mill: next gen Xeon Phi optimized for deep learning
- Intel announced addition of new vector instructions for deep learning, October 2016

Titan X GPU

- Pascal architecture: 3027 CUDA cores @ 1 GHz
- 6 TFLOPS FP32
- 12GB of GDDR5 @ 336 GB/sec
- 250 Thermal Design Power (TDP)
- 24 GFLOPS/W
- 28nm process

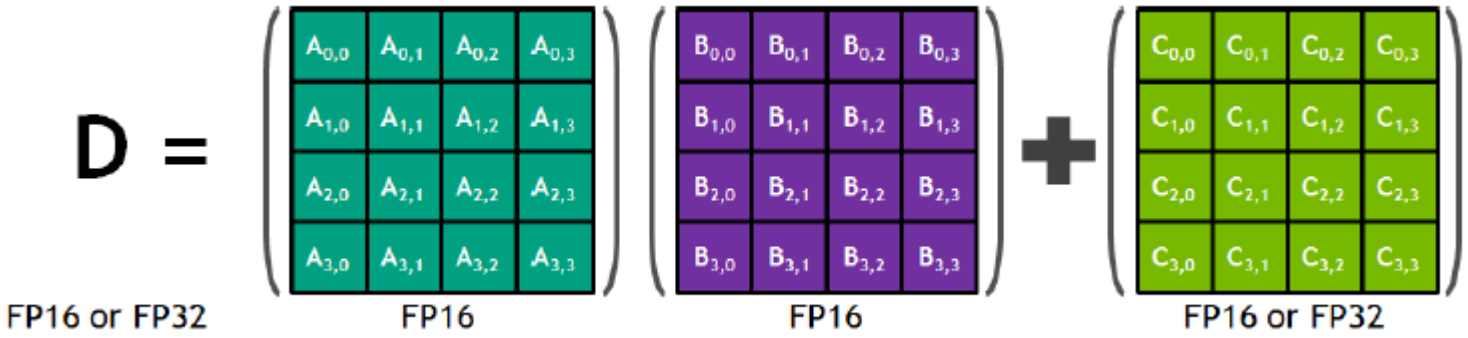
CPU versus GPU



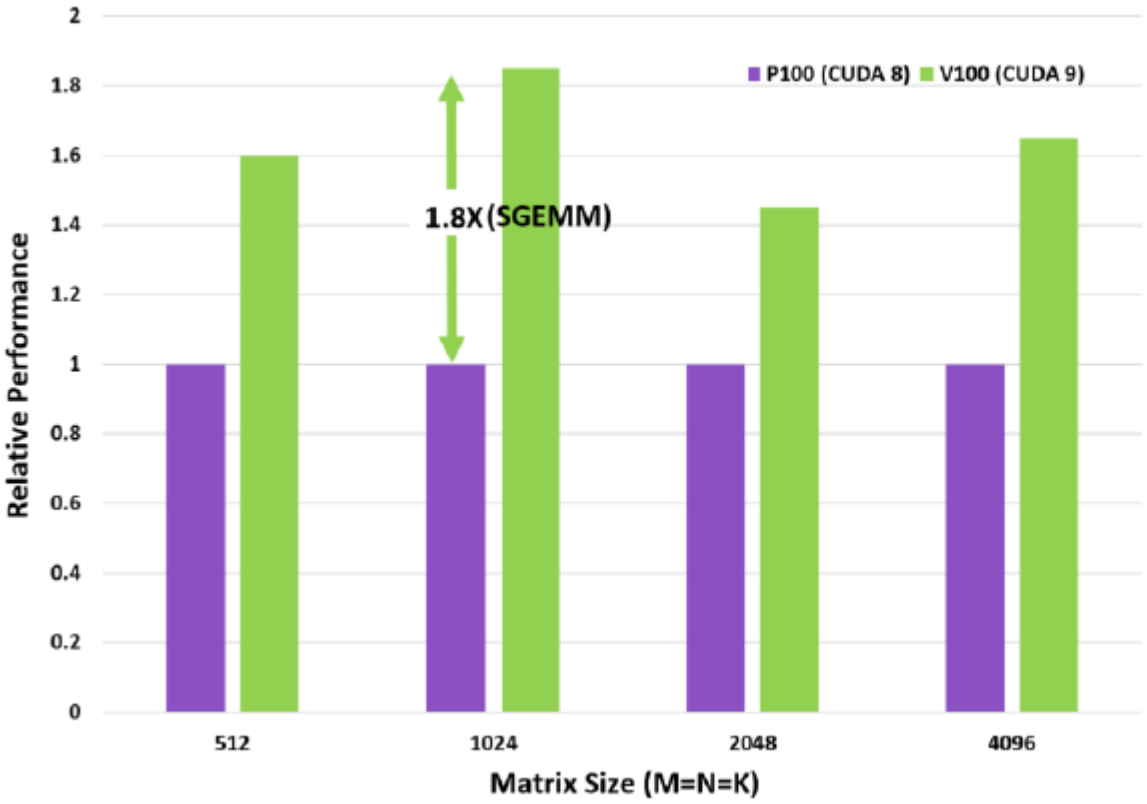


Volta Tensor Cores

- Volta Gv1000 contains 640 Tensor Cores
- Each Tensor Core operates on 4x4 matrix D = A x B + C
- Exposed as specialized warp-level matrix load, multiply and accumulate and store operations in CUDA
- 12x faster than Pascal architecture



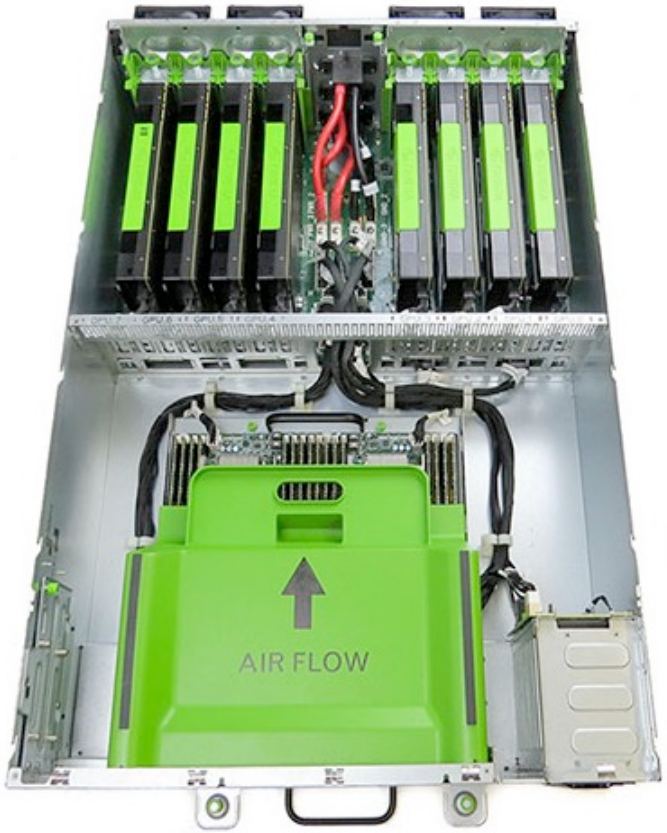
Volta Tensor Core: Matrix-Matrix Multiply



Single-precision (FP32) Matrix-Matrix Multiplies are up to 1.8x faster on Tesla V100 with CUDA 9 than on Tesla P100 with CUDA 8

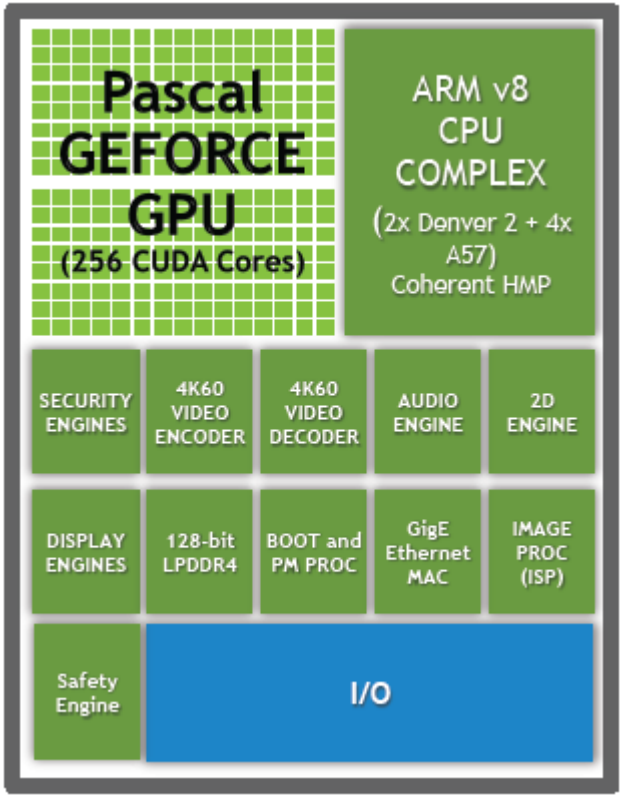
Cloud Systems for Deep Learning

- Facebook Deep Learning Machine
  - 8 Tesla M40 GPUs

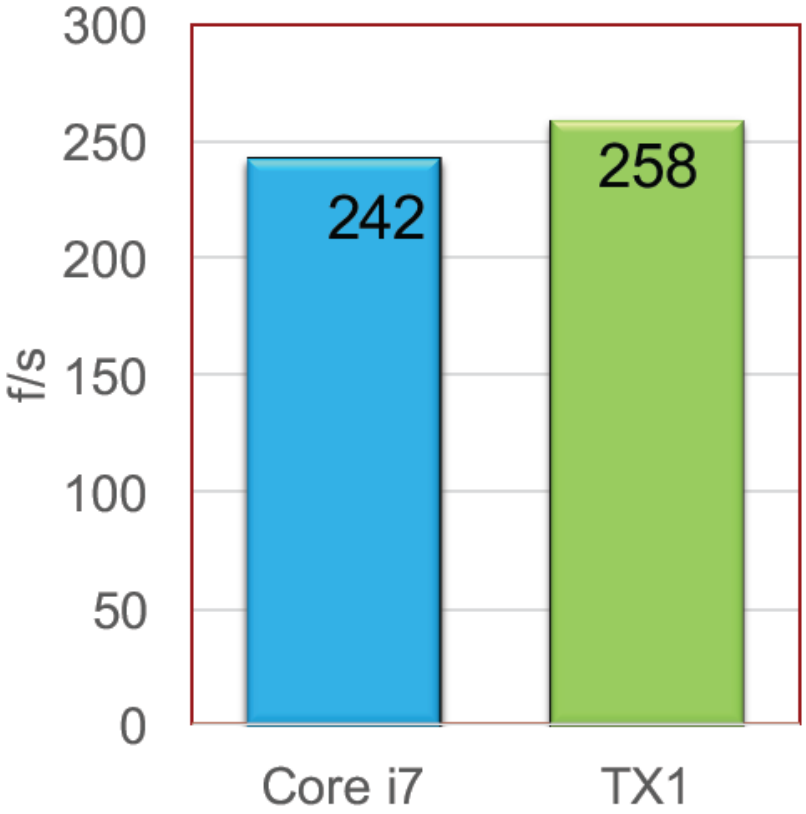


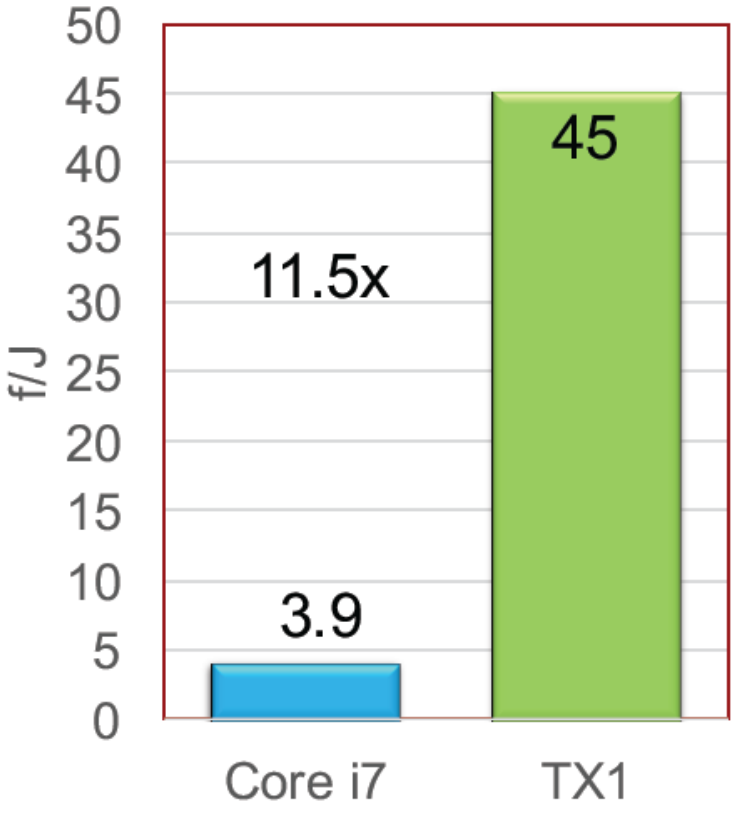
SoC for Deep Learning Inference

- NVIDIA Tegra Parker
- GPU: 1.5 TFLOPS FP16
- 4GB DRAM, 25.6 GB/sec
- 15W TDP
- 100 GFLOPS/W FP16
- 16nm process
- Xavier: next generation Tegra "AI Supercomputer"



Tegra X1 versus Core i7





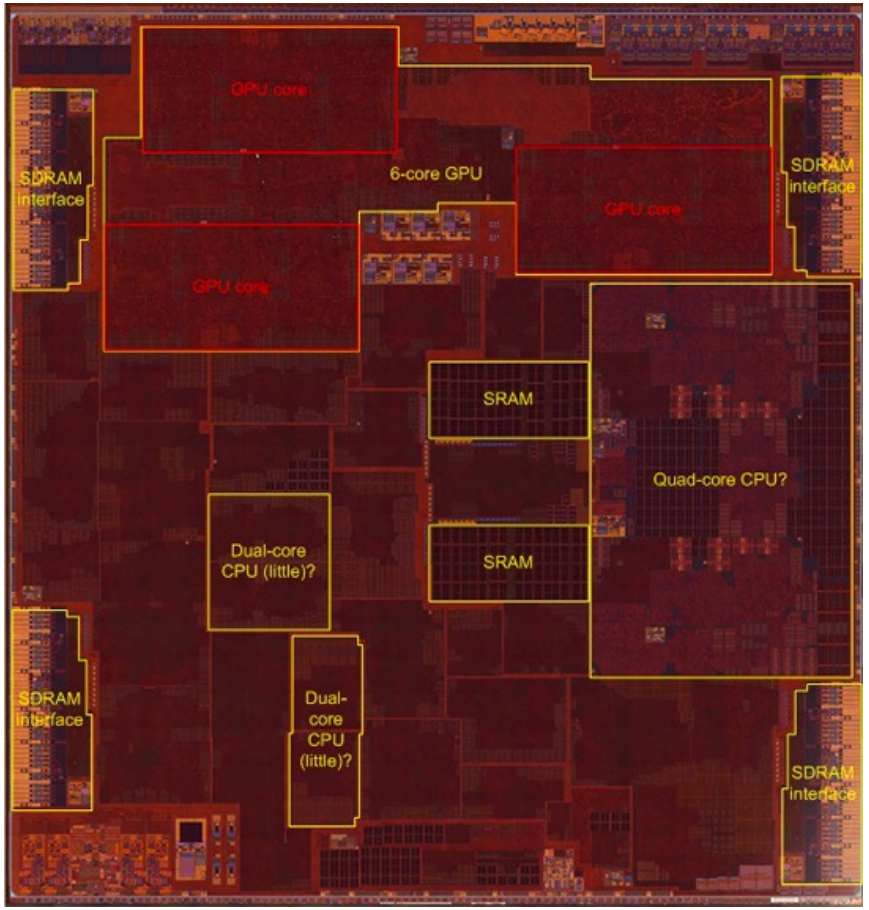
Huawei Kirin 970

- Dedicated Neural-network Processing Unit (NPU)
- 4x ARM Cortex A73 high-performance cores
- 4x ARM Cortex A53 efficiency cores

Apple Mobile Processors

- Apple A11 Bionic contains a neural engine

- A11 – 0.6 T/OPS[1]
- A12 – 5.0 T/OPS[1]
- A13 – 6.0 T/OPS[1]
- A14 – 8.0 T/OPS[1] (2nd-gen)



Apple A10

[1]https://www.macworld.com/article/3575331/a14-bionic-faq-performance-features-cpu-gpu-neural-engine.html

FPGAs for Deep Learning

- Intel/Altera Stratix 10
  - 10 TFLOPS FP32
  - 80 GFLOPS/W
- Xilinx Virtex UltraScale+
  - DSP: upto 21.2 TMACS

CPU, FPGA, GPU

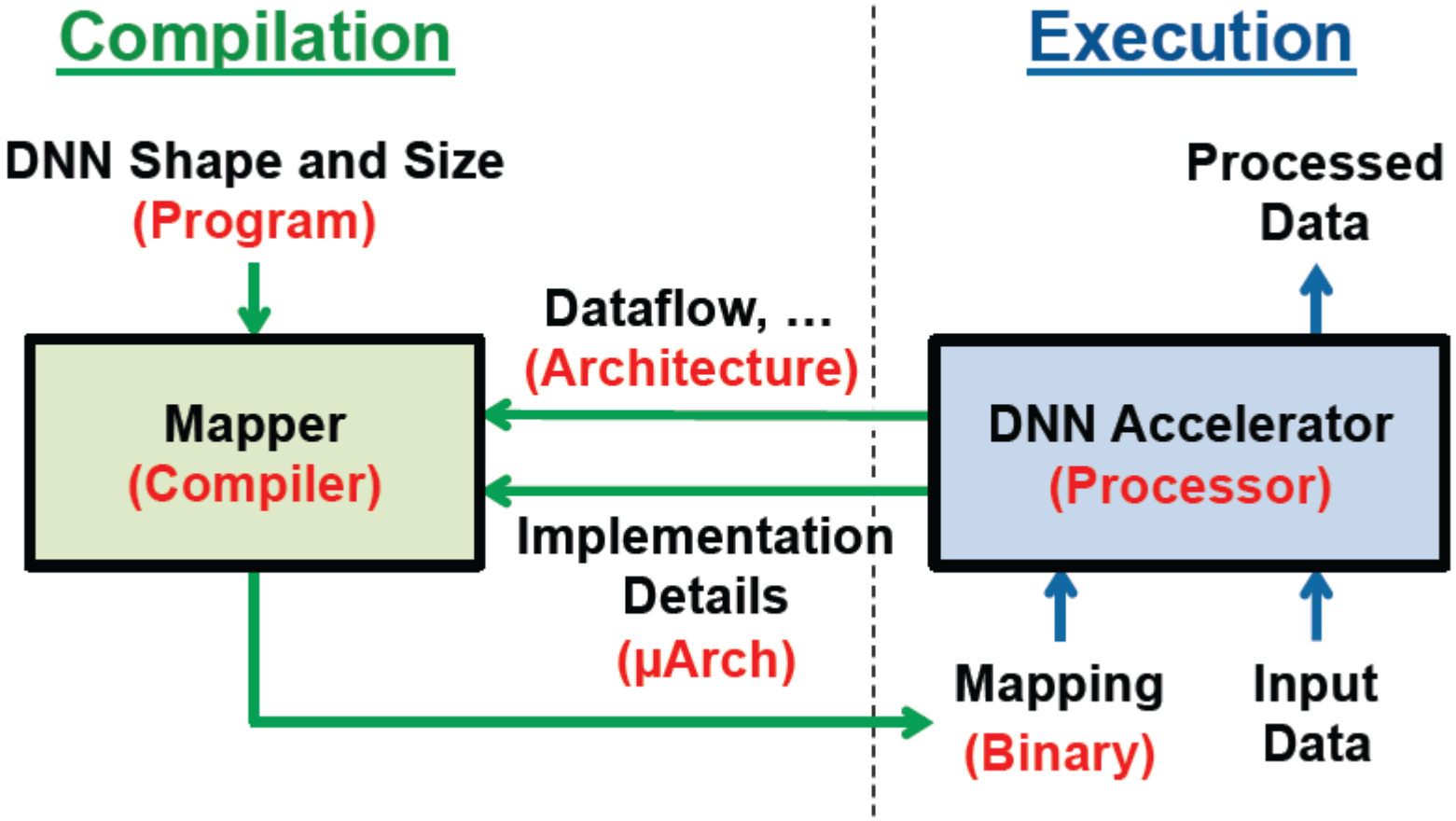
ImageNet-1K Classification Performance

Platform	Library/OS	ImageNet 1K Inference Throughput	Peak TFLOPs	Effective TFLOPs	Estimated Peak Power with Server	Estimated GOPs/J (assuming peak power)
16-core, 2-socket Xeon E5-2450, 2.1GHz	Caffe + Intel MKL Ubuntu 14.04.1*	53 images/s	0.27T	0.074T (27%)	~225W	~0.3
Arria 10 GX1150	Windows Server 2012	369 images/s[1]	1.366T	0.51T (38%)	~265W	~1.9
NervanaSys-32 on NVIDIA Titan X	NervanaSys-32 on Ubuntu 14.0.4	4129 images/s[2]	6.1T	5.75T (94%)	~475W	~12.1

Includes server power; however, CPUs available to other jobs in the datacenter

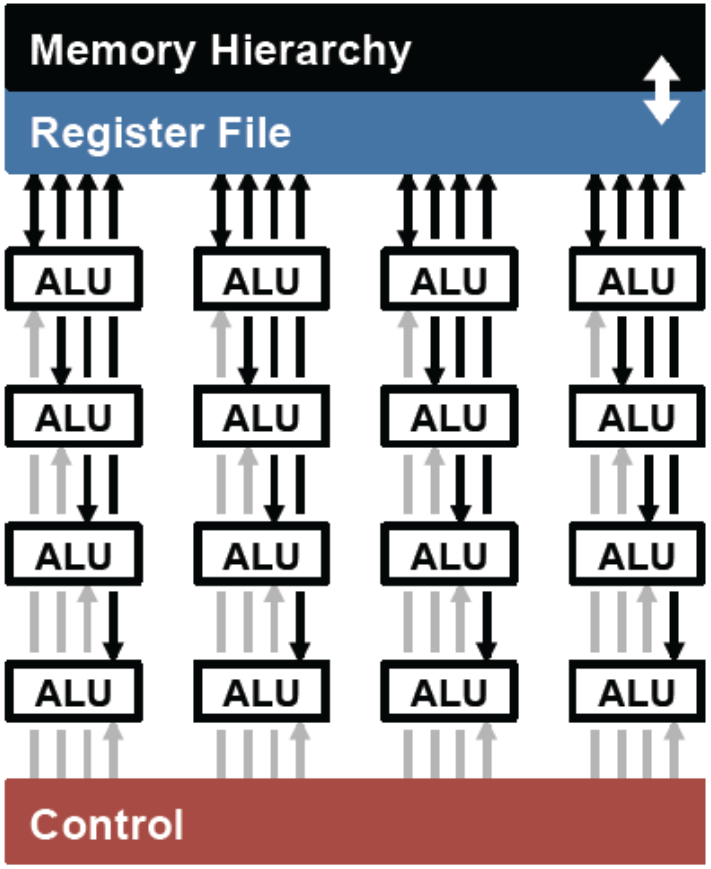
[1]Dense layer time estimated
[2]https://github.com/soumith/convnet-benchmarks

Systems View of Accelerators

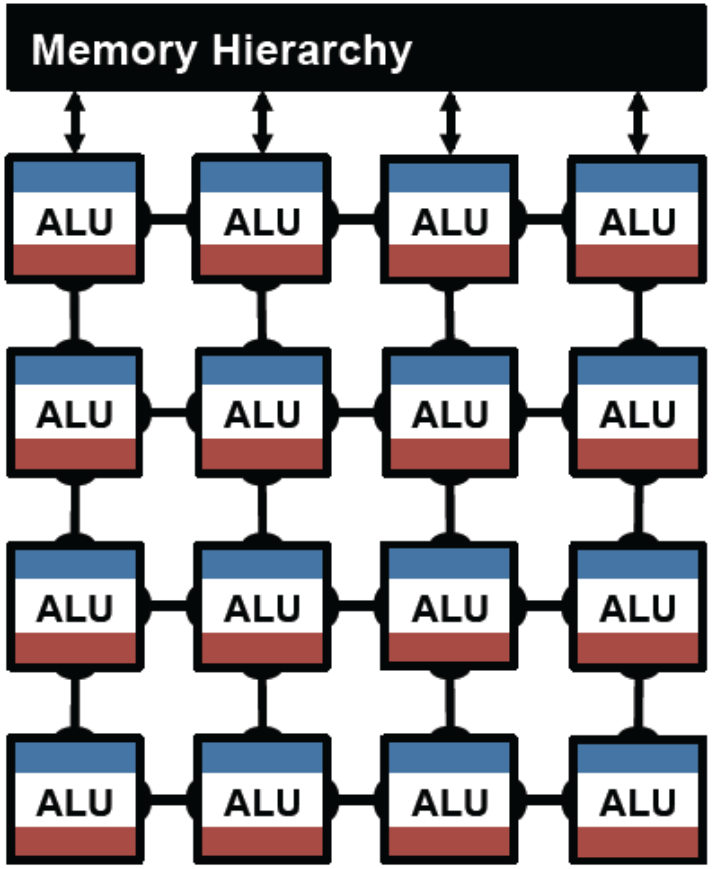


Compute Paradigm Choices

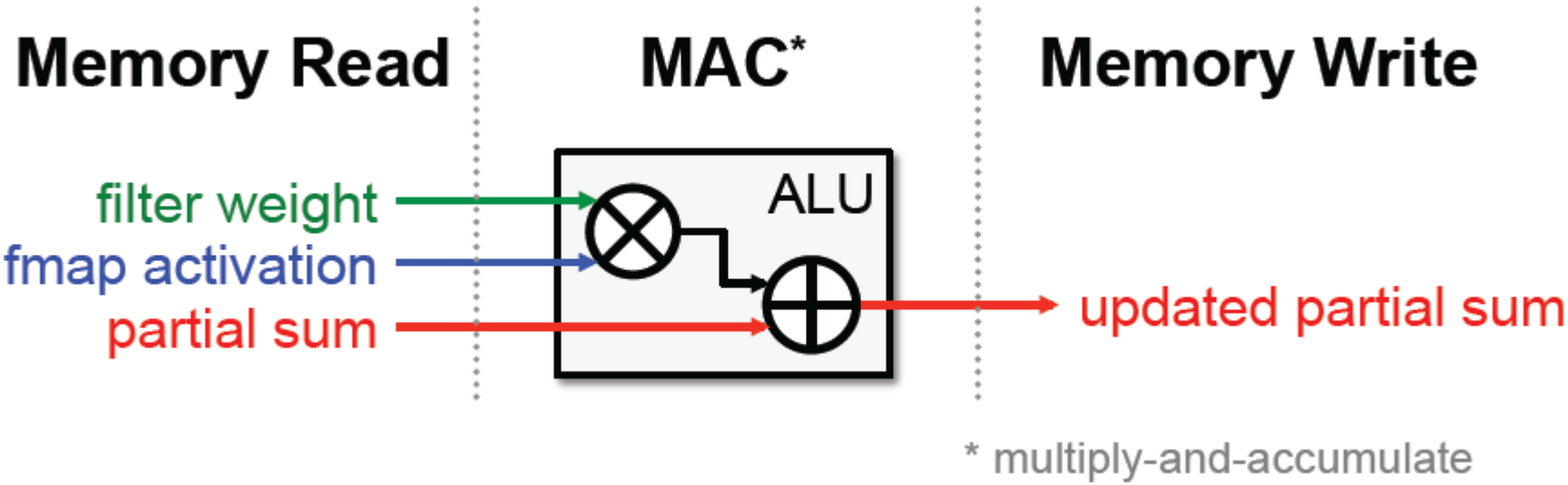
Temporal Architecture (SIMD/SIMT)



Spatial Architecture (Dataflow Processing)



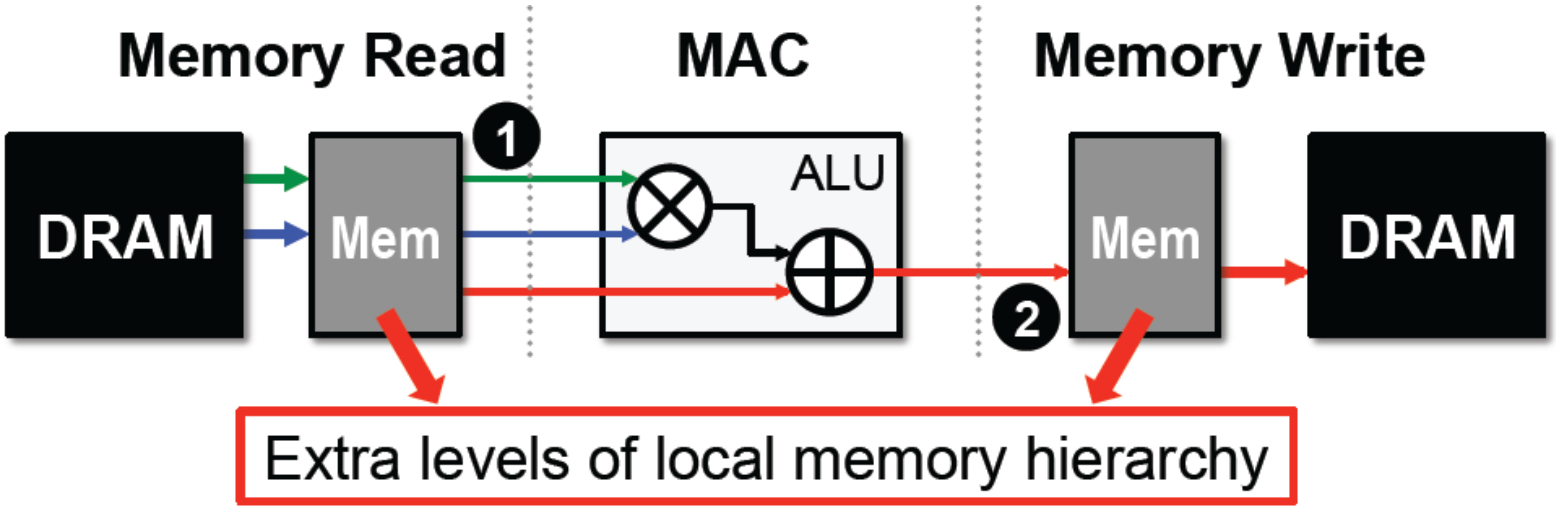
Memory Access is the Bottleneck



Worst Case: all memory R/W are **DRAM** accesses

- Example: AlexNet [NIPS 2012] has **724M** MACs
  → **2896M** DRAM accesses required

Solving Memory Bottleneck



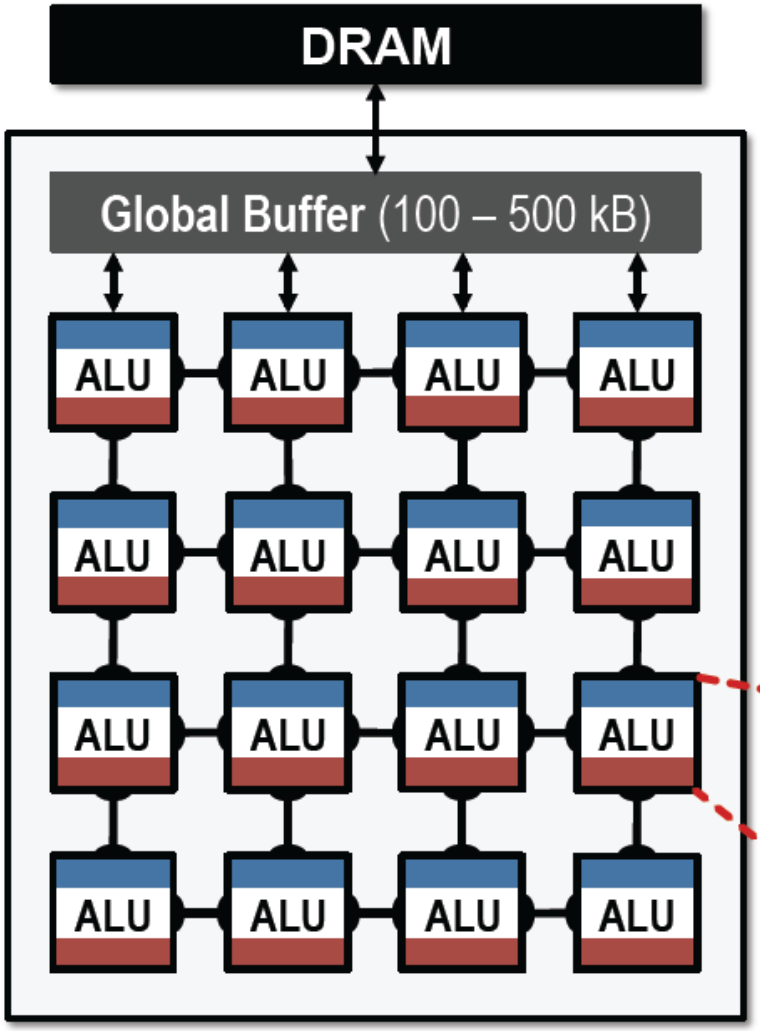
Opportunities: ❶ **data reuse** ❷ **local accumulation**

❶ Can reduce DRAM reads of filter/fmap by up to **500×**

❷ Partial sum accumulation does **NOT** have to access DRAM

- Example: DRAM access in AlexNet can be reduced from **2896M** to **61M** (best case)

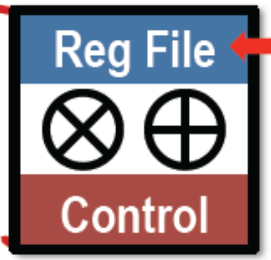
Spatial Architecture for DNN



Local Memory Hierarchy

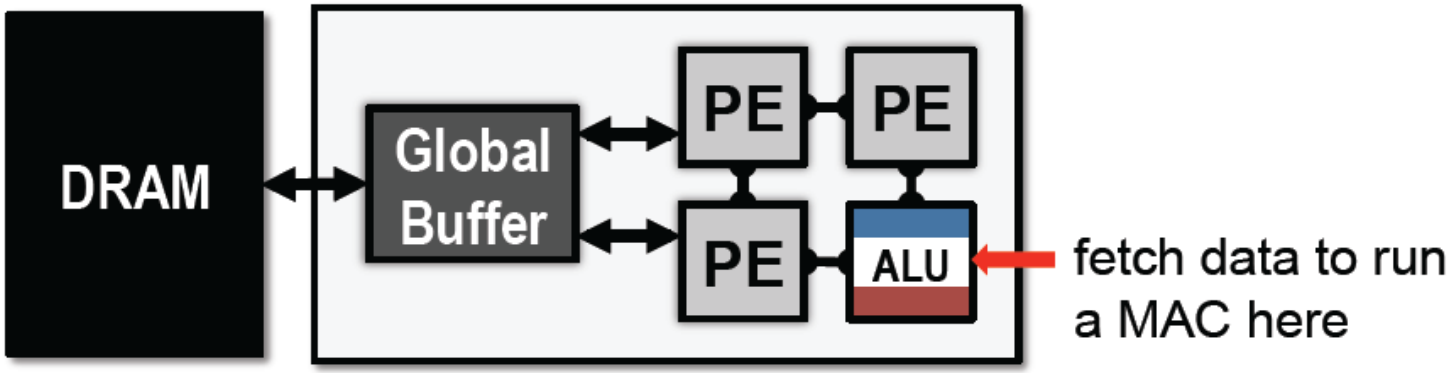
- Global Buffer
- Direct inter-PE network
- PE-local memory (RF)

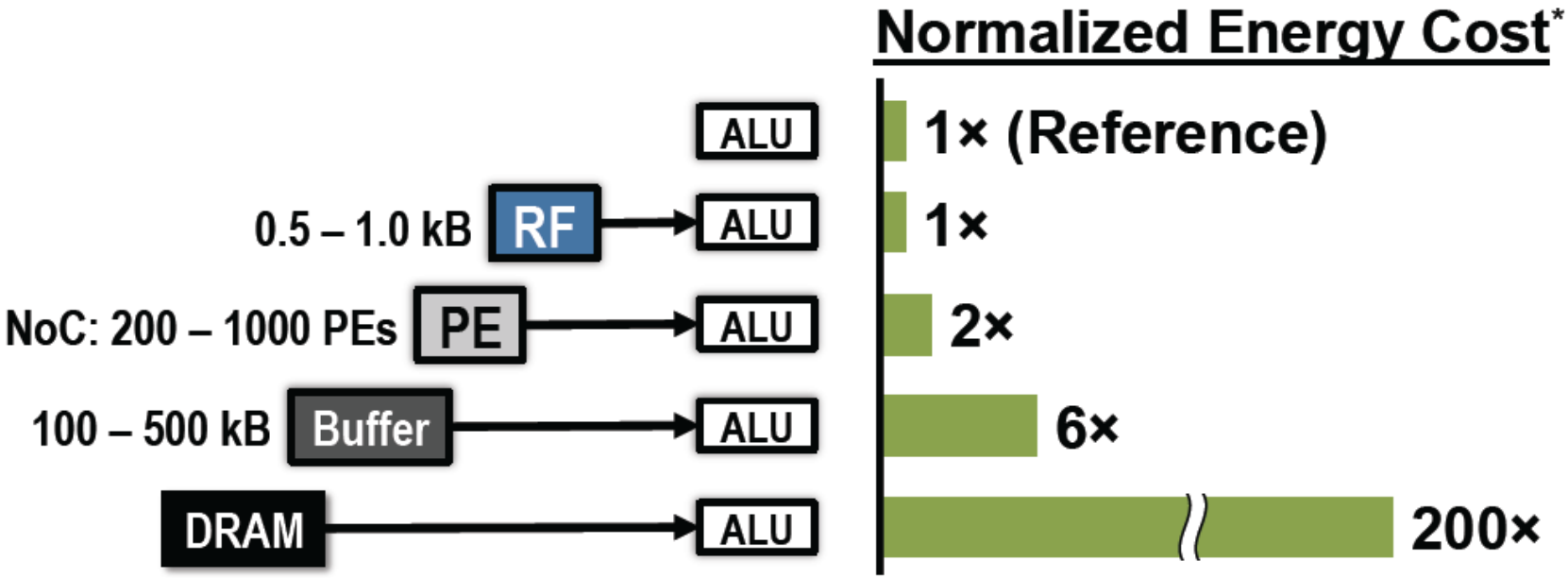
Processing Element (PE)



0.5 – 1.0 kB

Low-Cost Local Data Access





Accelerator: Diannao (Electric Brain)

- Dedicated functional units and memory buffers optimized for CNN workload
- 452 GOP/sec, 3.02 mm^2, 485 mW

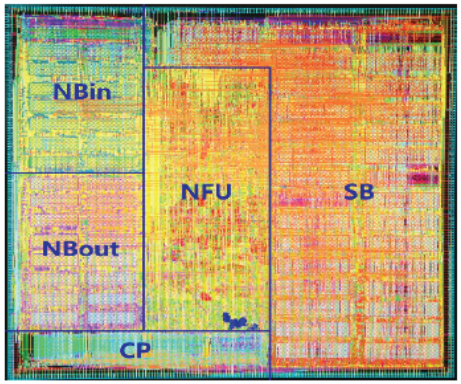


Figure 15. *Layout (65nm).*

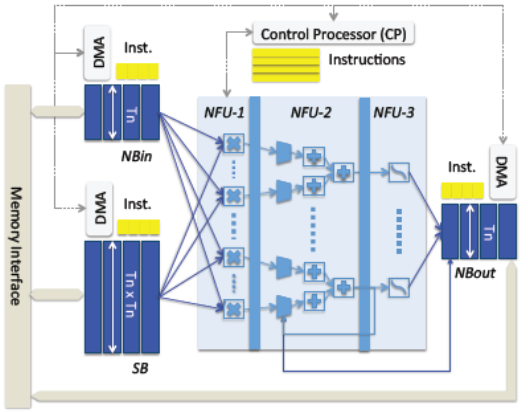
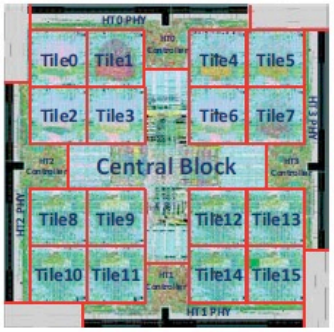


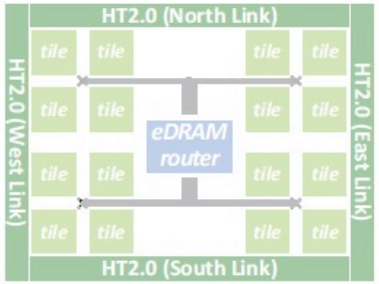
Figure 11. *Accelerator.*

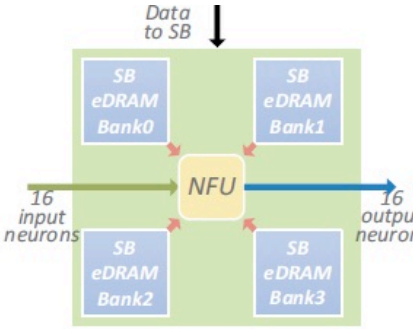
Component or Block	Area in μm^2	(%)	Power in mW	(%)	Critical path in ns
ACCELERATOR	3,023,077		485		1.02
Combinational	608,842	(20.14%)	89	(18.41%)	
Memory	1,158,000	(38.31%)	177	(36.59%)	
Registers	375,882	(12.43%)	86	(17.84%)	
Clock network	68,721	(2.27%)	132	(27.16%)	
Filler cell	811,632	(26.85%)			
SB	1,153,814	(38.17%)	105	(22.65%)	
NBin	427,992	(14.16%)	91	(19.76%)	
NBout	433,906	(14.35%)	92	(19.97%)	
NFU	846,563	(28.00%)	132	(27.22%)	
CP	141,809	(5.69%)	31	(6.39%)	
AXIMUX	9,767	(0.32%)	8	(2.65%)	
Other	9,226	(0.31%)	26	(5.36%)	

Table 6. *Characteristics of accelerator and breakdown by component type (first 5 lines), and functional block (last 7 lines).*

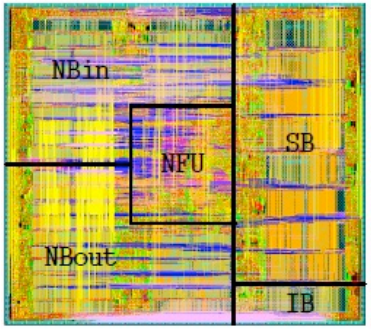
Diannao and Friends

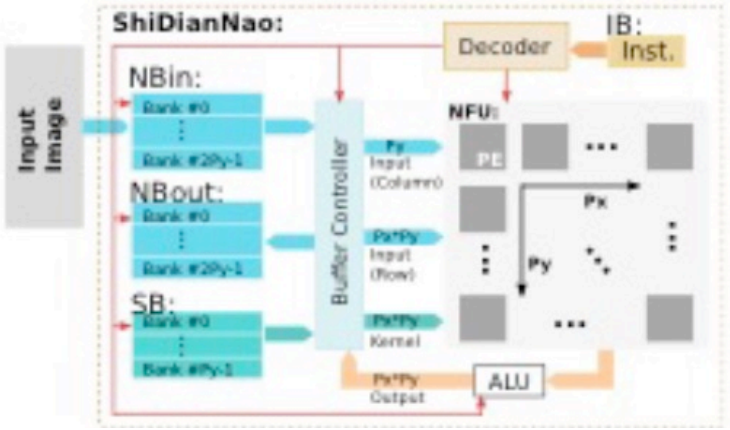






DaDiannao uses multi-chip and EDRAM to fit larger models. Each chip is 68 mm^2 fitting 12 M parameters, consumes 16W

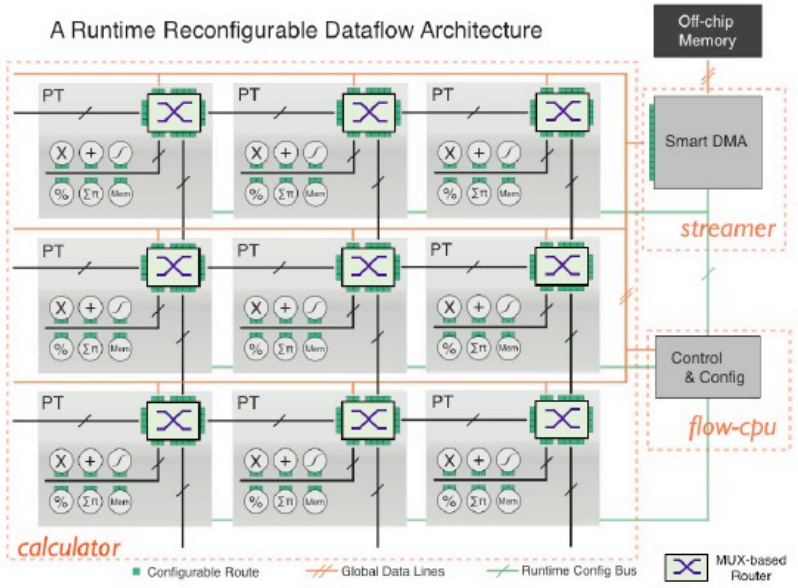


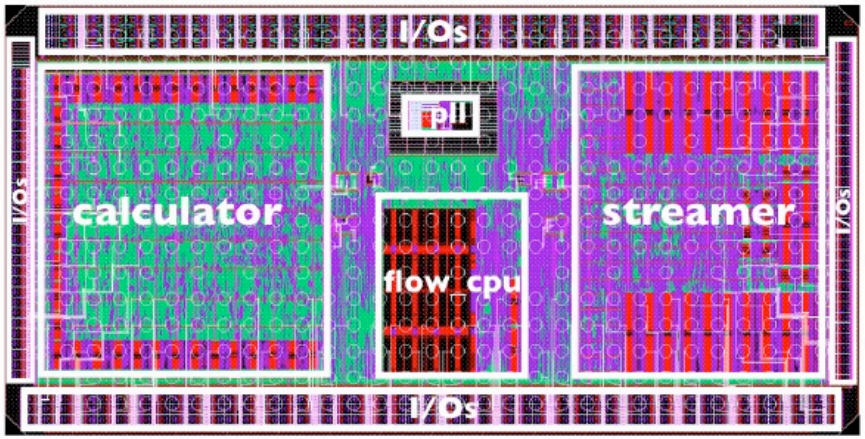


ShiDiannao fits small models (64K parameters). Maps computation on 2D PE array. The chip is 4.86 mm^2 and consumes 320mW

NeuFlow

- SoC designed to accelerate neural networks and vision algorithms based on convolutions/matrix multiply
- 160 GOPS, 570 mW, 12.5 mm^2



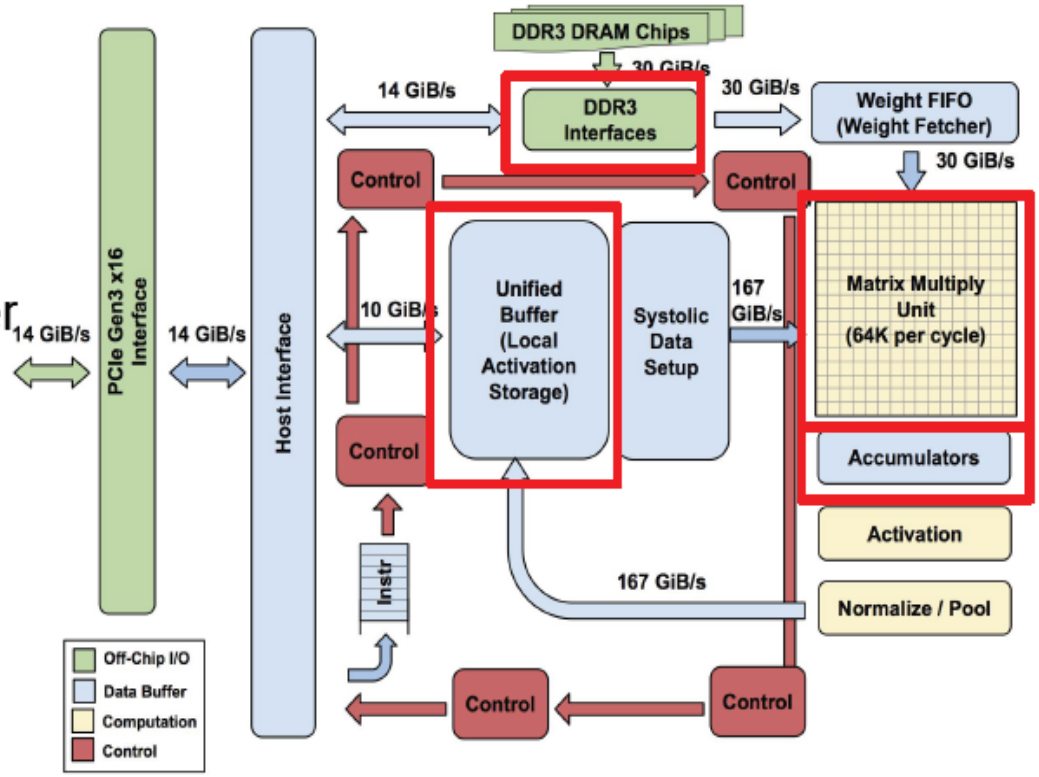


Google Tensor Processing Unit (TPU)

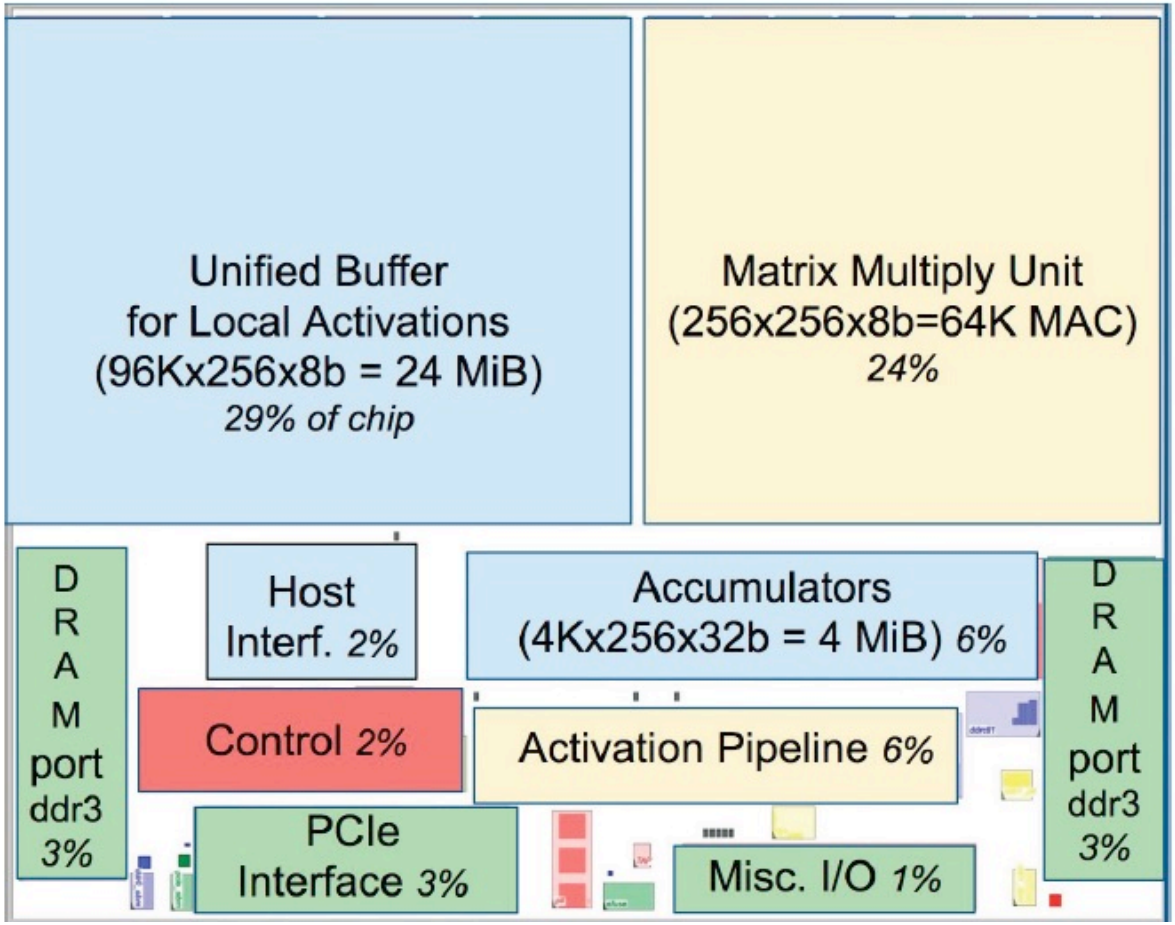
- Inference engine
- Add as accelerators to existing servers
  - Connect over I/O (PCI) bus
  - Matrix accelerator on I/O bus
- Host server sends it instructions like a floating-point unit unlike GPU that fetches and executes own instructions

TPU: High-level Chip Architecture

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
  - 65,536 * 2 * 700M
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory



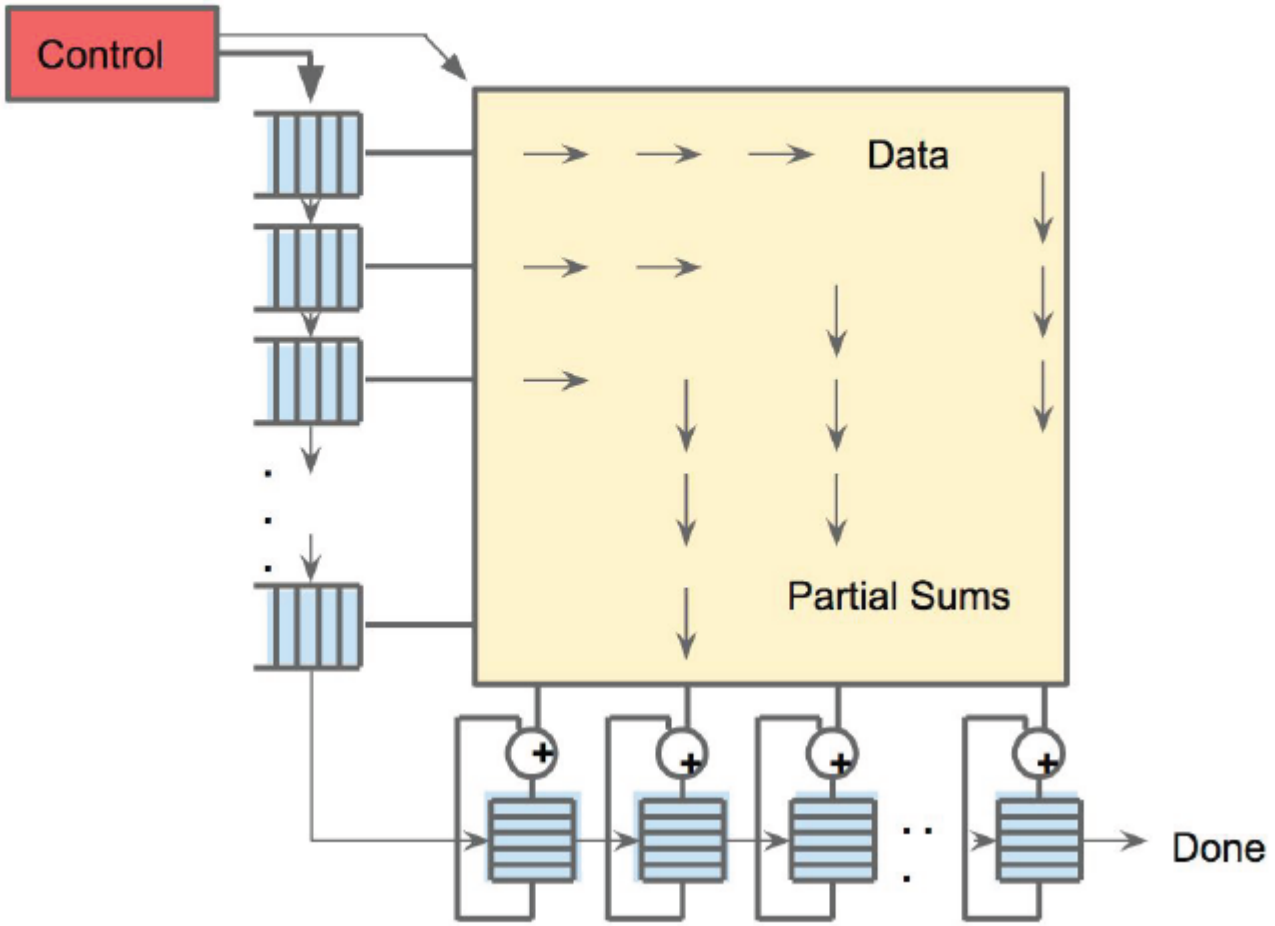
TPU: a Neural Network Accelerator Chip



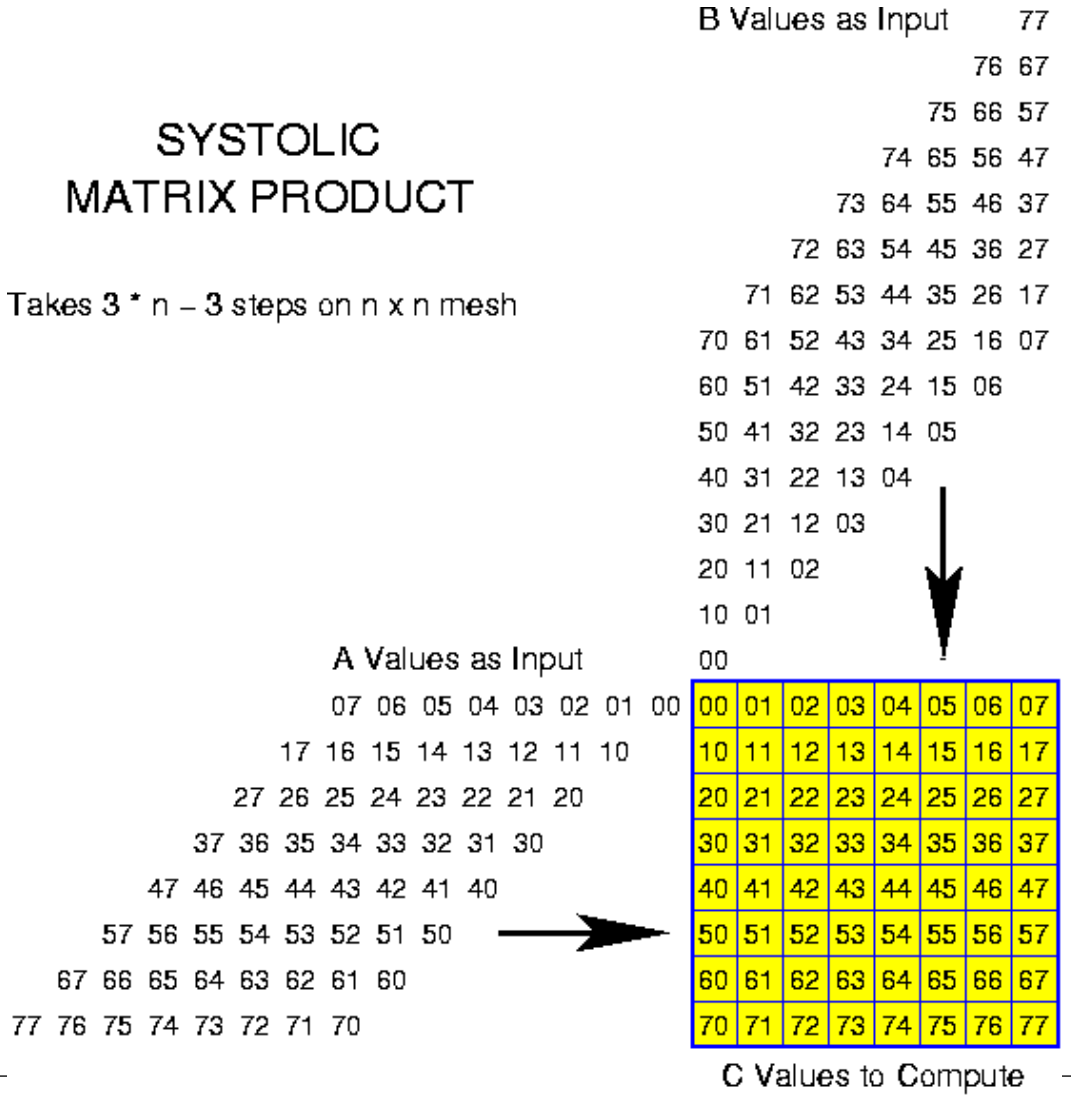
TPU Architecture, programmer's view

- 5 main (CISC) instructions
  `Read_Host_Memory`
  `Write_Host_Memory`
  `Read_Weights`
  `MatrixMultiply/Convolve`
  `Activate(ReLU,Sigmoid,Maxpool,LRN,…)`
- Average Clock cycles per instruction: >10
- 4-stage overlapped execution, 1 instruction type / stage
  - Execute other instructions while matrix multiplier busy
- Complexity in SW: No branches, in-order issue,
  SW controlled buffers, SW controlled pipeline synchronization

Systolic Execution: Control and Data are pipelined



Matrix Multiplication: Systolic Array



Relative Performance: 3 Contemporary Chips

Processor	*mm²*	*Clock MHz*	*TDP Watts*	*Idle Watts*	*Memory GB/sec*	*Peak TOPS/chip*	
						8b int.	*32b FP*
CPU: Haswell (18 core)	662	2300	145	41	51	2.6	1.3
GPU: Nvidia K80 (2 / card)	561	560	150	25	160	--	2.8
TPU	<331*	700	75	28	34	91.8	--

*TPU is less than half die size of the Intel Haswell processor

K80 and TPU in 28 nm process; Haswell fabbed in Intel 22 nm process

These chips and platforms chosen for comparison because widely deployed in Google data centers

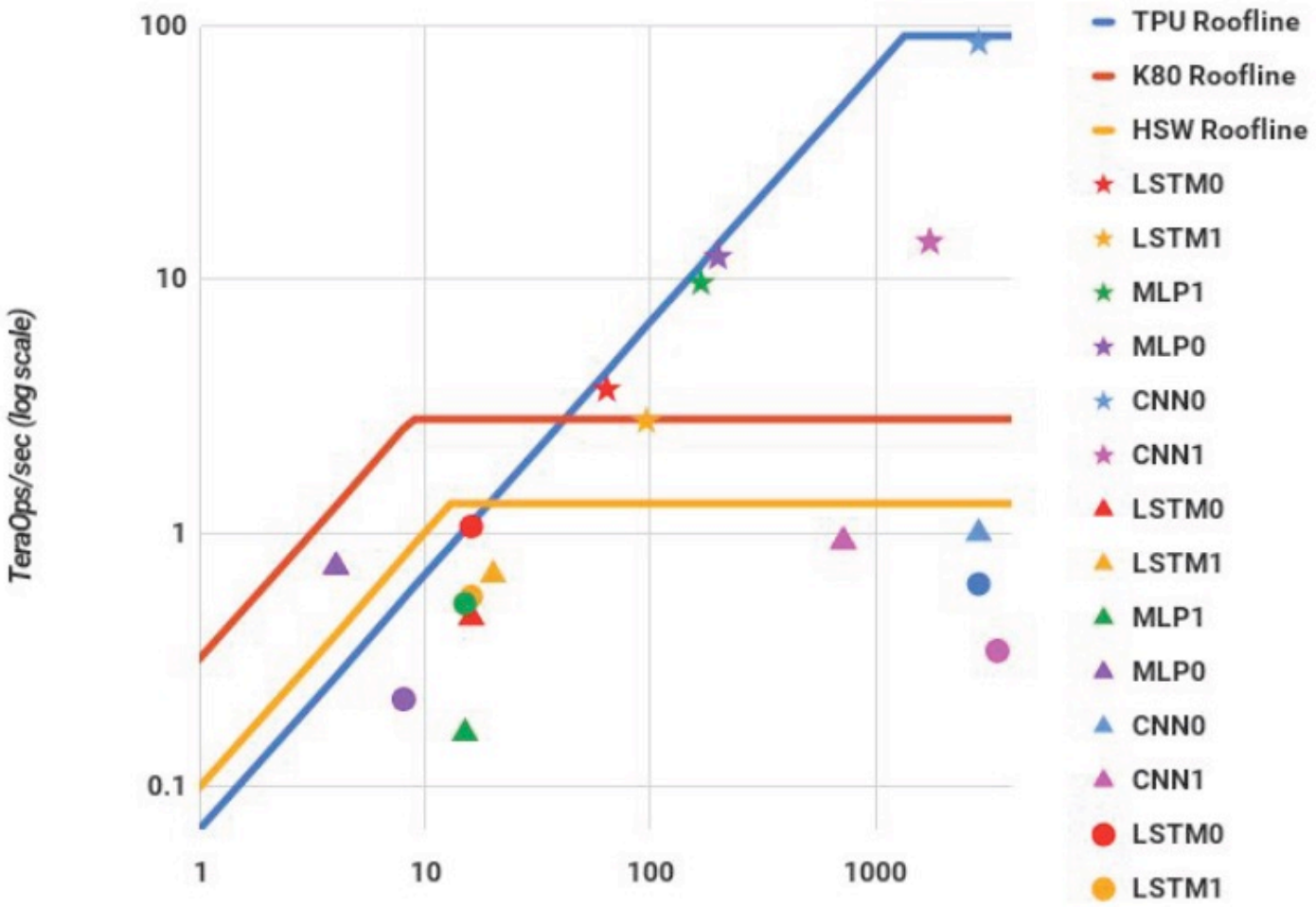
GPUs and TPUs added to CPU server

Relative Performance: 3 Platforms

Processor	*Chips/ Server*	*DRAM*	*TDP Watts*	Idle Watts	Observed Busy Watts in datacenter
CPU: Haswell (18 cores)	2	256 GB	504	159	455
NVIDIA K80 (13 cores) (2 die per card; 4 cards per server)	8	256 GB (host) + 12GB x 8	1838	357	991
TPU (1 core) (1 die per card; 4 cards per server)	4	256GB (host) + 8GB x 4	861	290	384

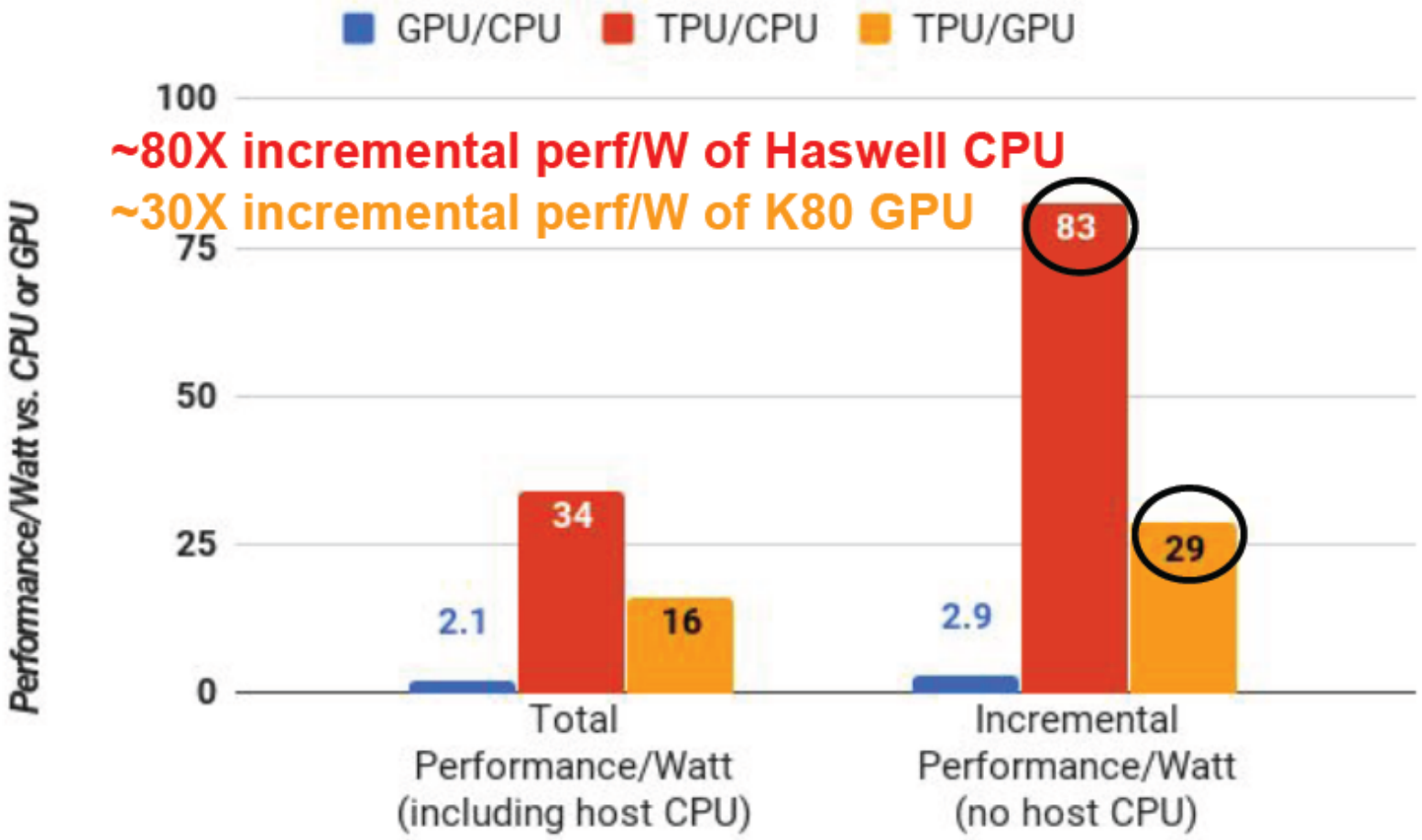
These chips and platforms chosen for comparison because widely deployed in Google datacenters

Log Rooflines for CPU, GPU, TPU



☆ Star = TPU
△ Triangle = GPU
○ Circle = CPU

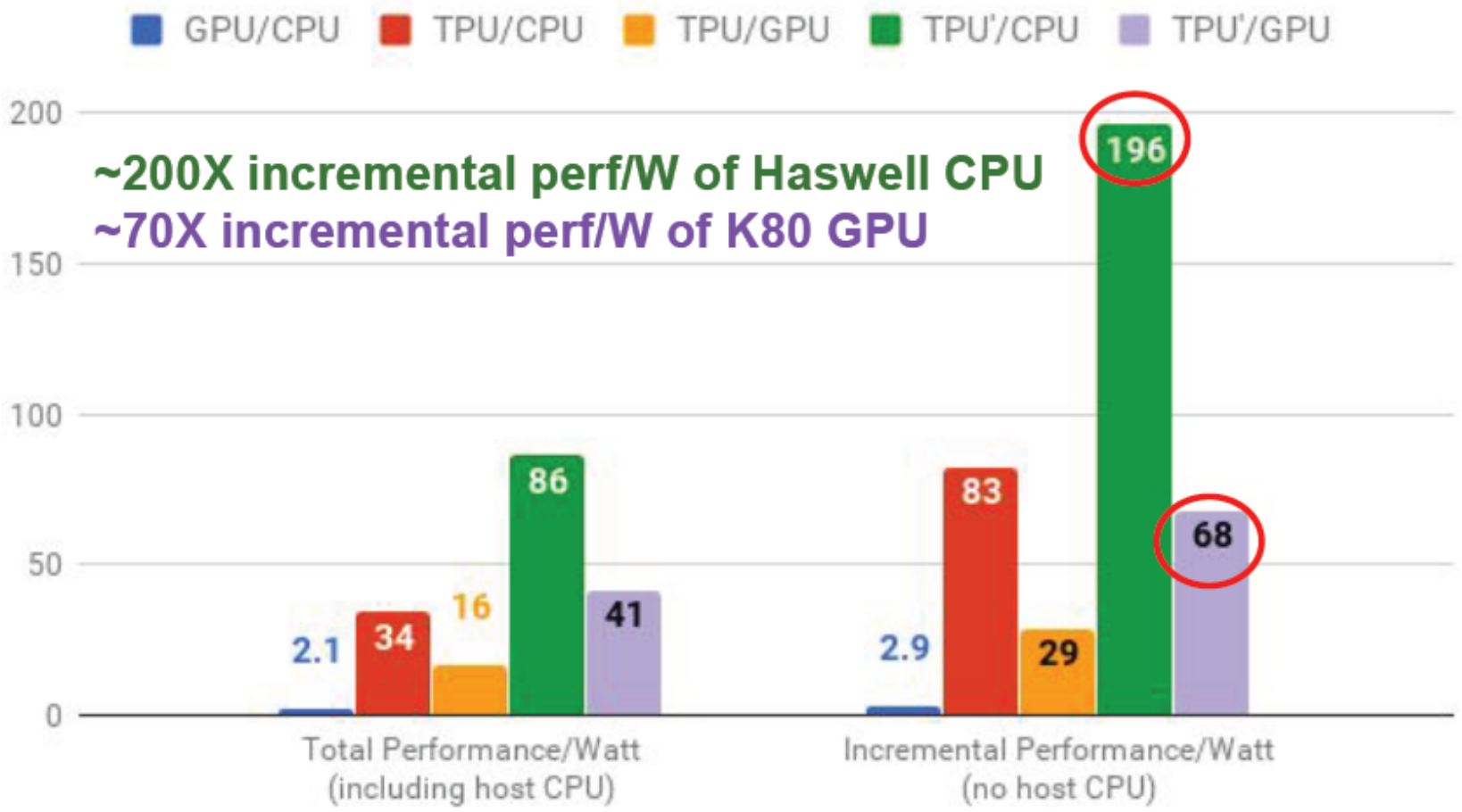
Perf/Watt TPU vs CPU & GPU



Improving TPU: Move “Ridge Point”

- Replace 2DDR3 (34 GB/sec) with GDDR5 in K80 (180 GB/sec)
- Move Ridge Point to the left

Perf/Watt Original & Revised TPU



You Only Spike Once (YOSO)

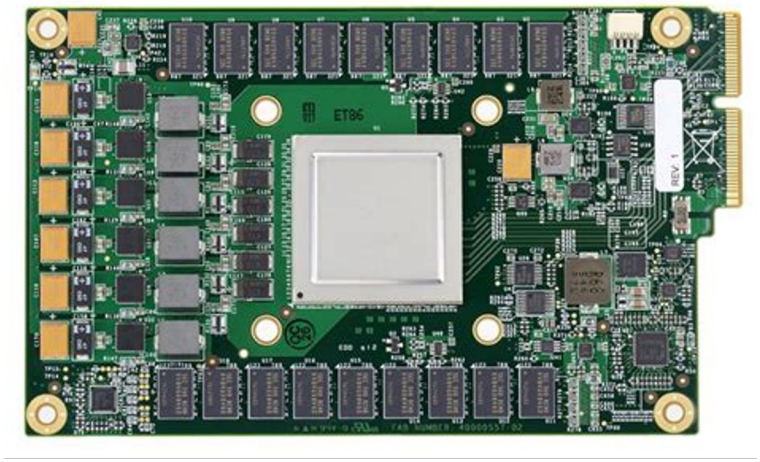
Improving Energy-Efficient Neuromorphic Inference to ANN-Level Accuracy

Srivatsa P, Kyle Timothy Ng Chu, Yaswanth Tavva, Jibin Wu,
Malu Zhang, Haizhou Li, Trevor E. Carlson

NUS
National University
of Singapore

Machine Learning Accelerators

TPU

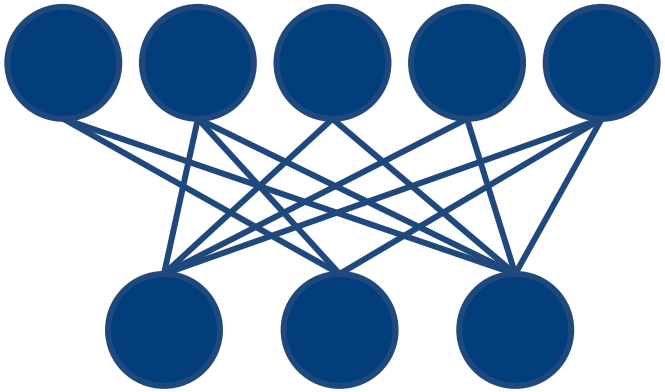


75W

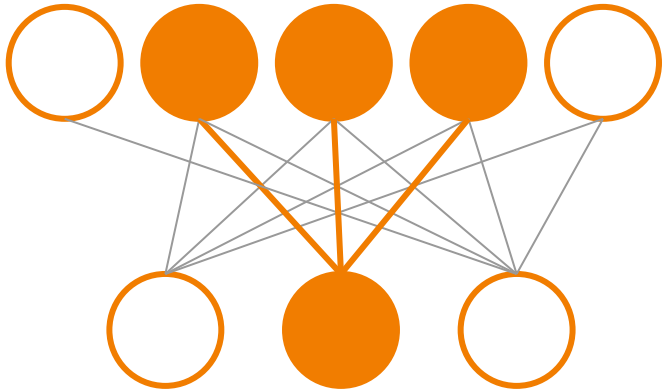
Human Brain

20W

SNNs - Why are they more power efficient?



Artificial Neural Network
(ANN)

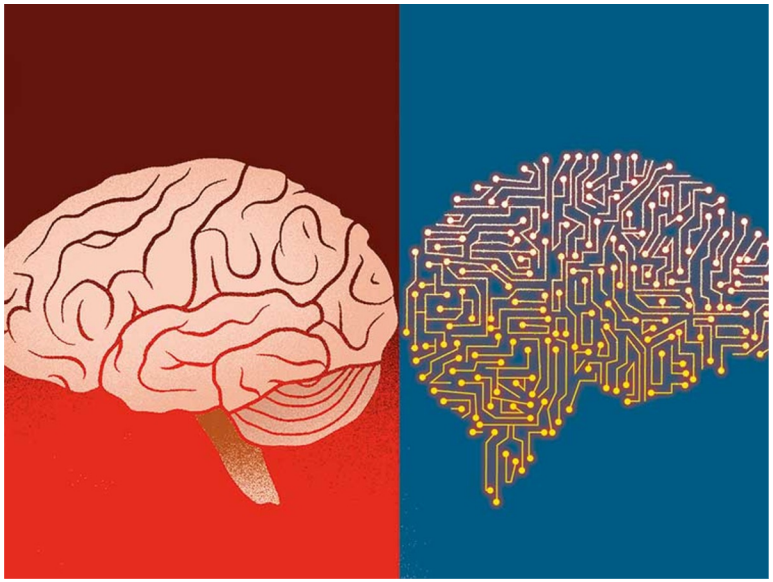


Spiking Neural Networks
(SNN)

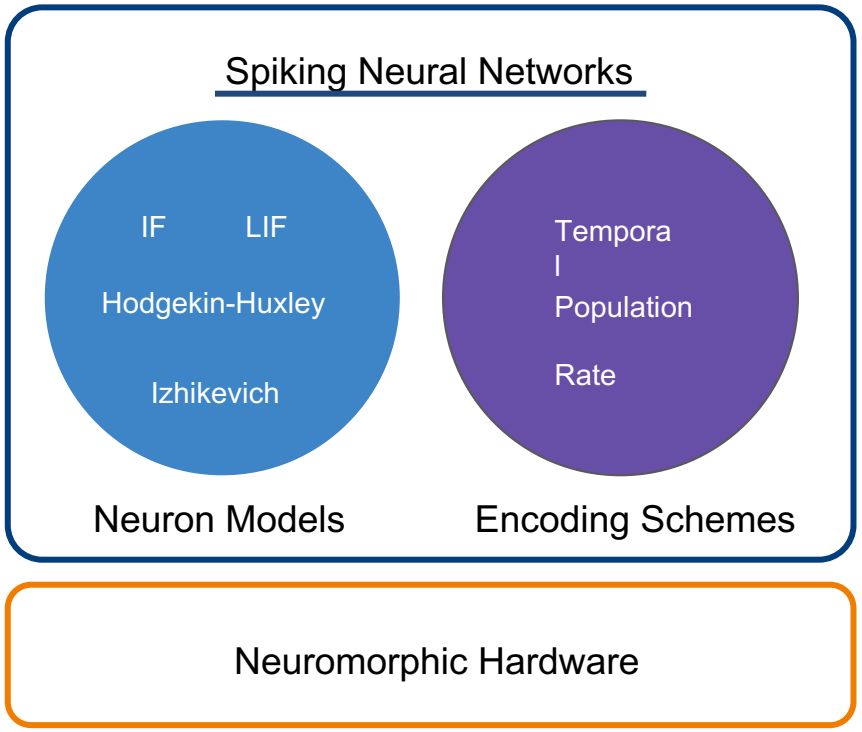
1. Cheaper compute operations

2. Not all neurons are used in computation

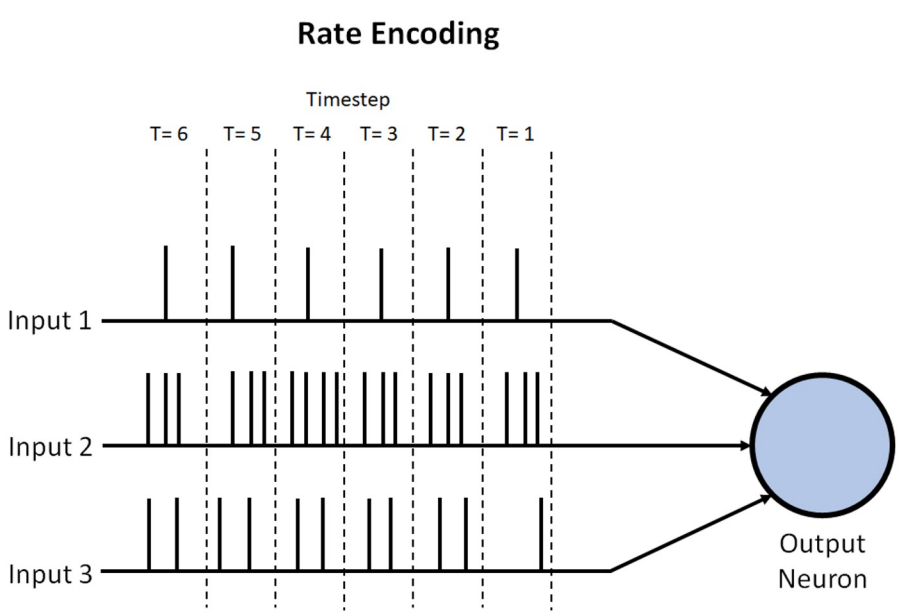
Spiking Neural Networks

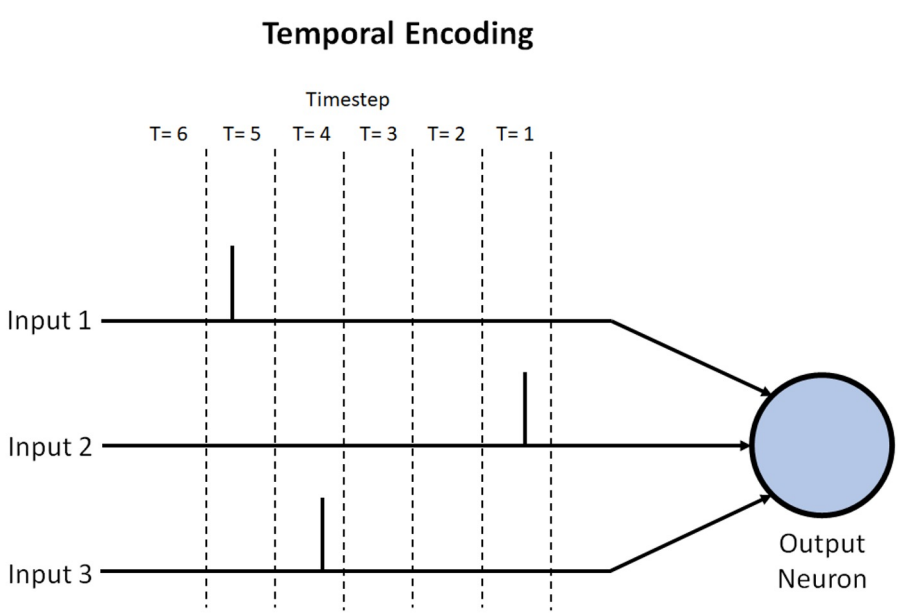


Source: https://gulfnews.com/opinion/op-eds/ai-cannot-replace-the-wonders-of-human-brain-1.2194745



Neural Encoding Techniques





Time-To-First-Spike (TTFS)

Prior work has demonstrated that:

1. TTFS networks generate significantly lower number of operations
2. Traditional ANNs can be directly converted into TTFS networks with low accuracy loss

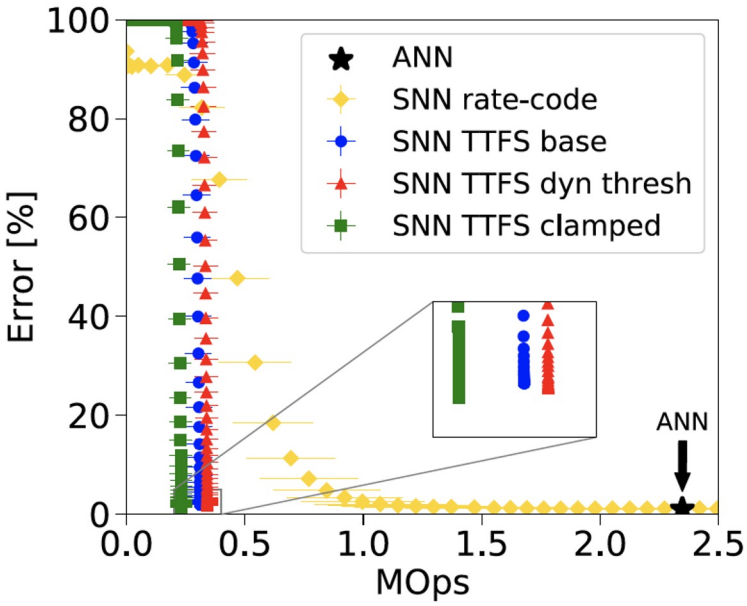
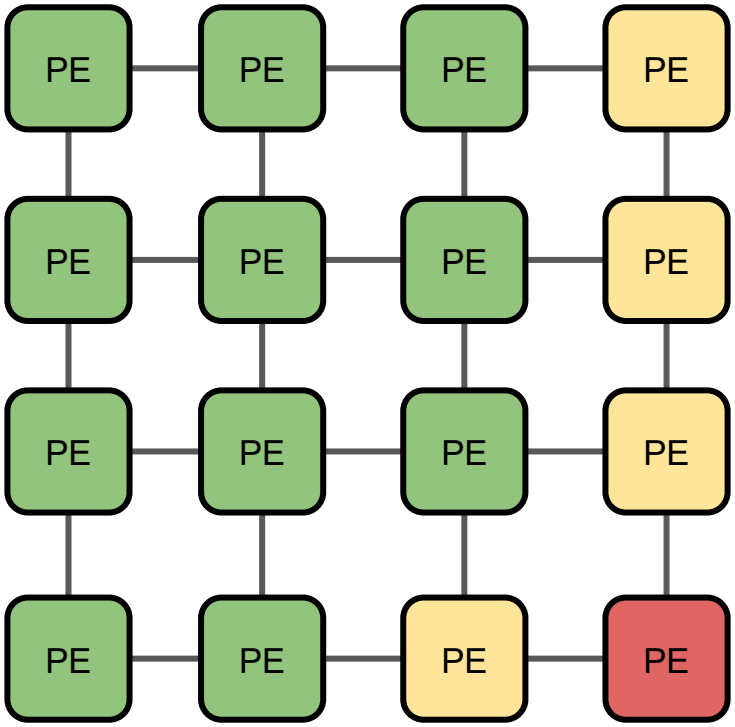


Image taken from :B. Rueckauer and S. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, Italy, 2018, pp. 1-5. doi: 10.1109/ISCAS.2018.8351295

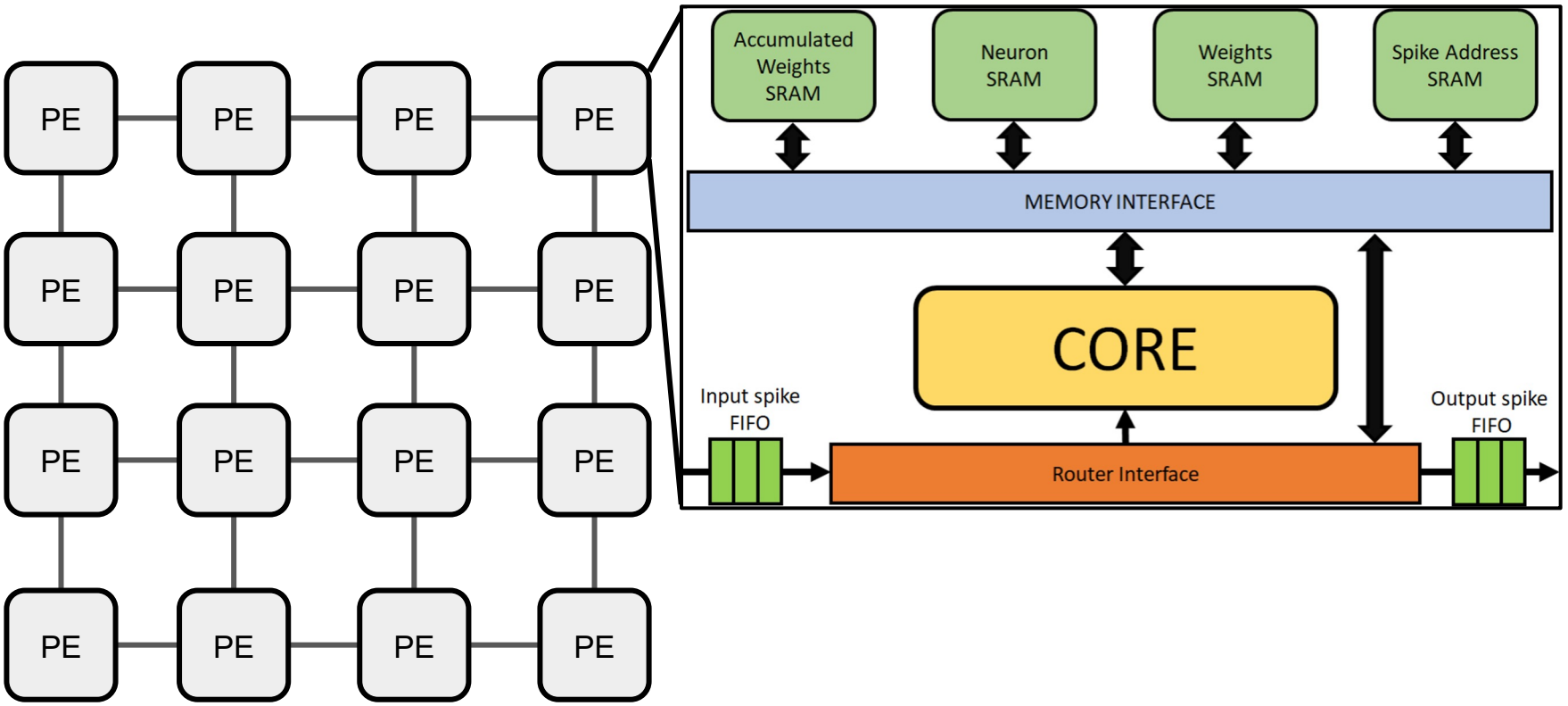
YOSO Accelerator Architecture



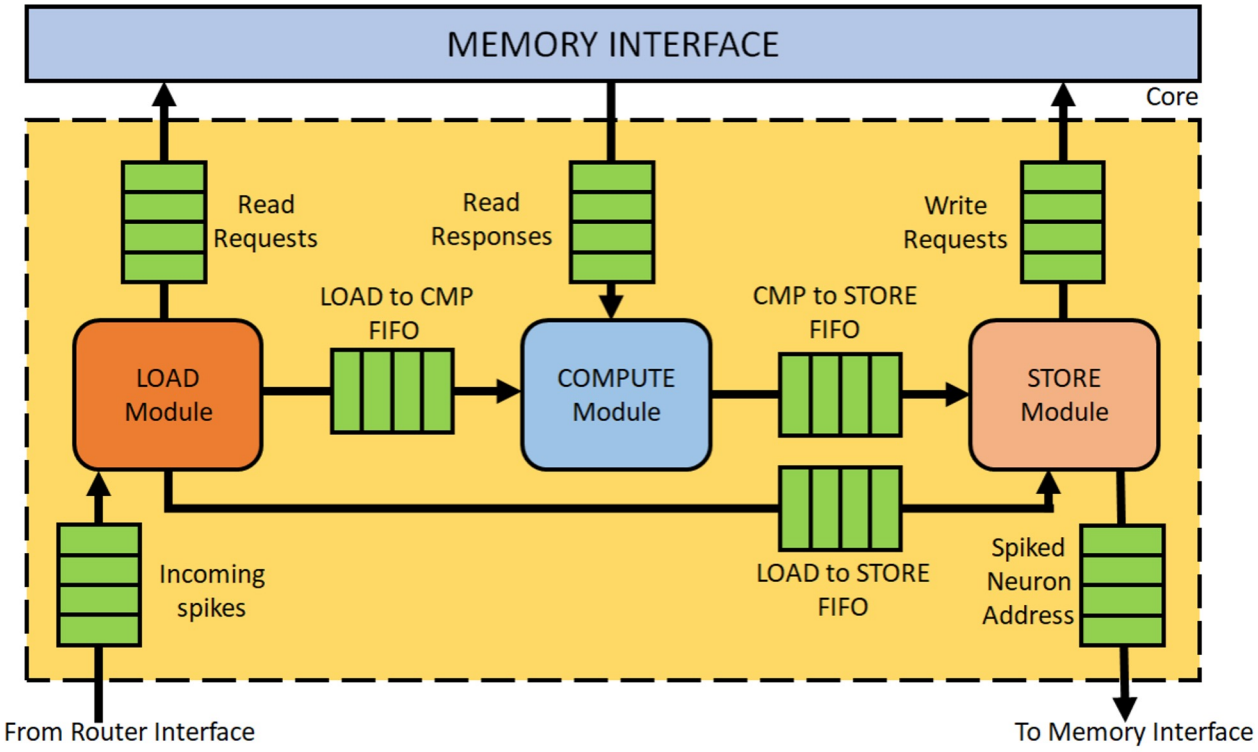
Layers are allocated cores depending on computational load

	Num. Neurons	Spikes Processed
Layer 1	300	116
Layer 2	300	30
Layer 3	10	55

YOSO Accelerator Architecture



YOSO Accelerator Architecture

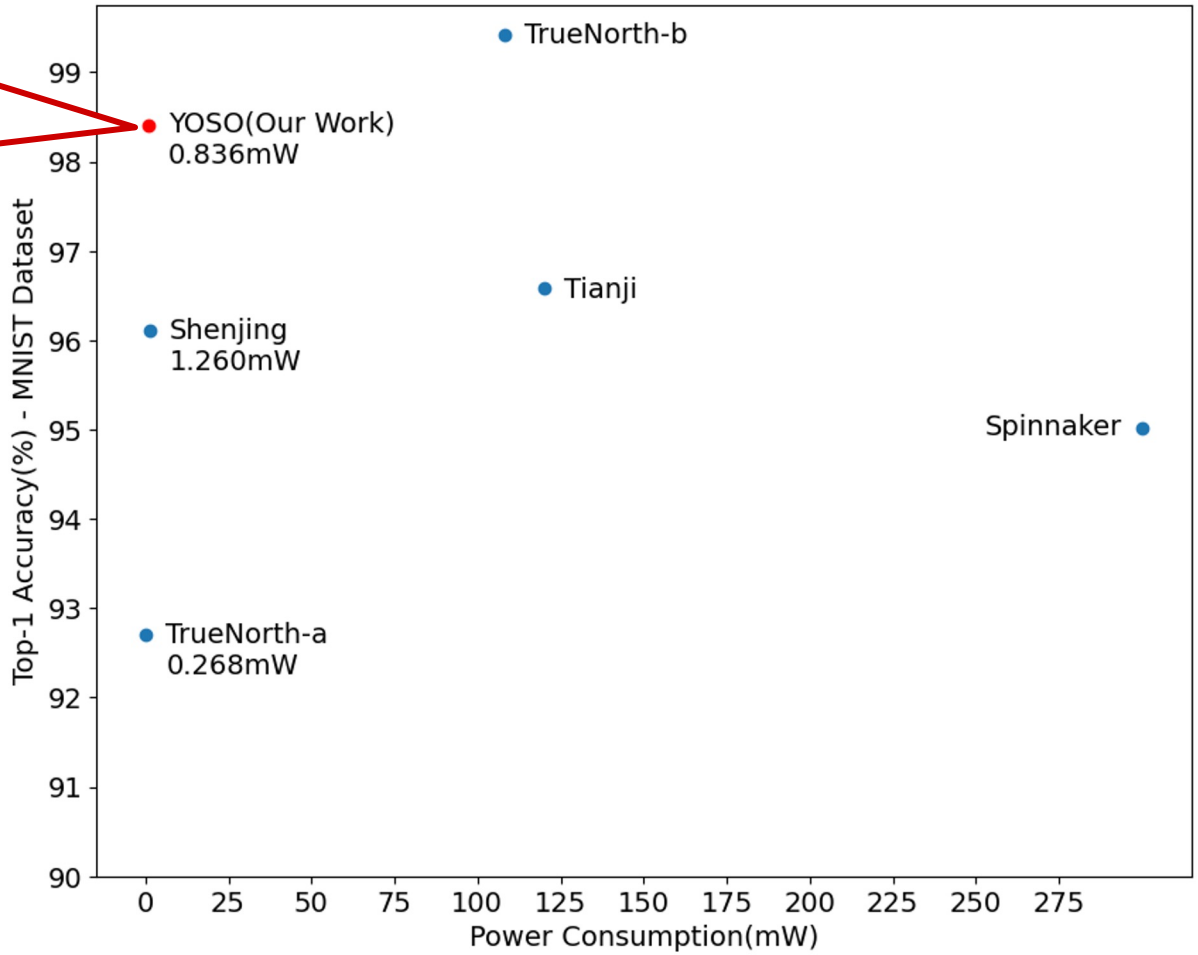


Evaluation

Our Work

High Accuracy

Low Power



Summary

- Training: Use clusters of 8-16 GPUs
  - Best perf, perf/W, perf/$, memory bandwidth
  - Easy parallelism
- Inference in data center
  - Single GPU (no real-time constraints)
  - Accelerator like Google TPU (real-time constraints)
- Inference in mobile devices (Automotive, IoT)
  - SoC with accelerators or GPU