EINDHOVEN UNIVERSITY OF TECHNOLOGY

Efficient Processing of Deep Neural Networks: from Algorithms to Hardware Architectures

Dr. ir. Maurice Peemen

Electrical Engineering

My Background

- Masters Electrical Engineering at TU/e
- PhD work at TU/e
  - Thesis work with Prof. Dr. Henk Corporaal
  - Topic: Improving the Efficiency of Deep Convolutional Networks

- Staff Scientist at Thermo Fisher Scientific (formerly FEI Company)
  - Electron Microscopy Imaging Challenges
- Manager Research & Development
  - Leading a research team to progress microscopy with cutting edge algorithms

The World Leader in Serving Science

We enable our customers to make the world **healthier, cleaner** and **safer**

- Technology innovation leadership
- Unique customer value proposition
- Unparalleled global reach

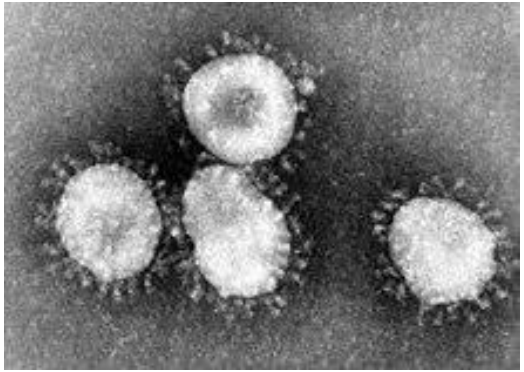
- $24 billion in revenues
- >$900 million spent on R&D
- >70,000 employees
- 5 premier brands

ThermoFisher SCIENTIFIC

thermo scientific | applied biosystems | invitrogen | fisher scientific | unity lab services

Recent News: Queen Maxima visiting the researchers working on vaccines for Corona

- Queen Maxima at the group of Prof. Eric Snijder at LUMC

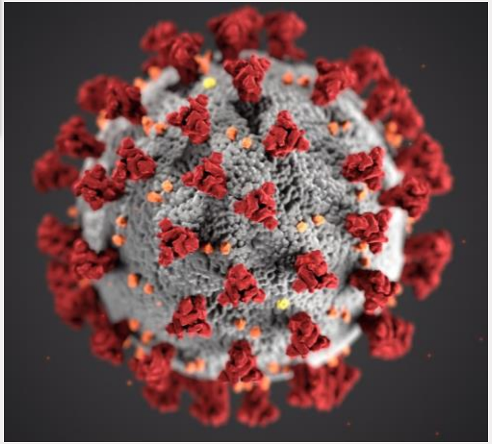


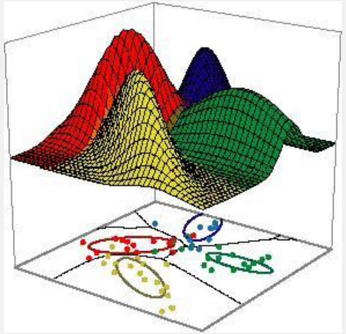
Older generation of microscopes
Not yet very data driven

and not suited for cryo-EM

High Performance Data Processing: Combining Domains

High Performance Computing and Algorithms





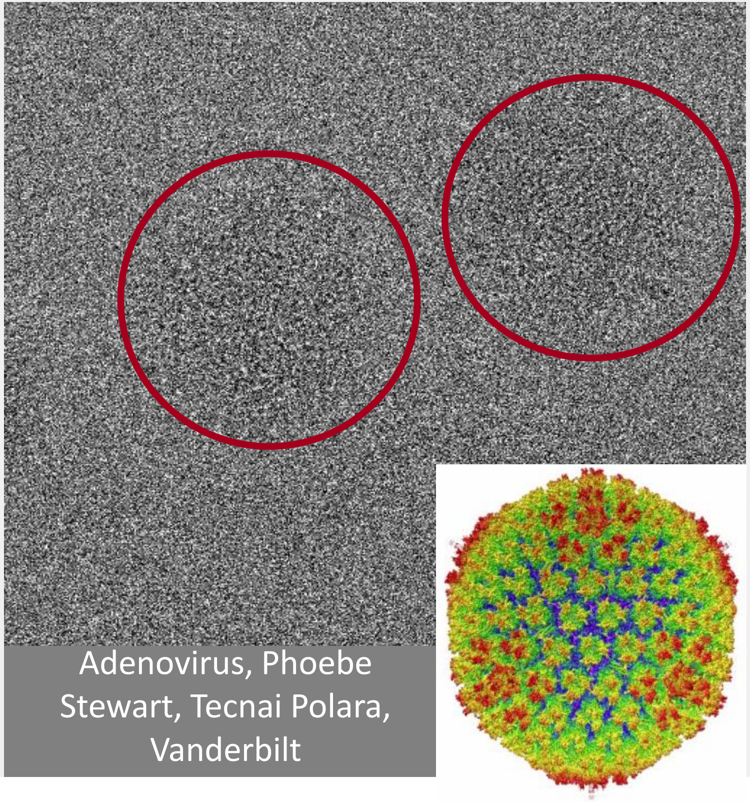
The key method to study the mechanics of the spikes of the new Corona virus

Imaging Cryo-Specimen: Need 4 Signal

Contrast problems

- Radiation damage
- Low electron dose

- Solutions to extract info
  - Cameras + Algorithms



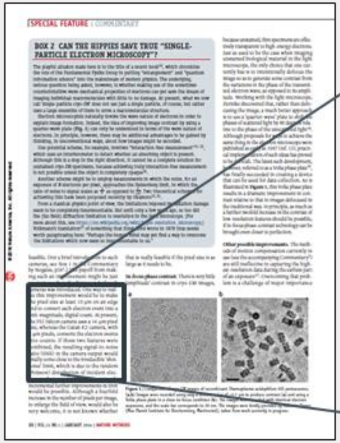
Adenovirus, Phoebe Stewart, Tecnai Polara, Vanderbilt

Single Particle Analysis (SPA) workflow

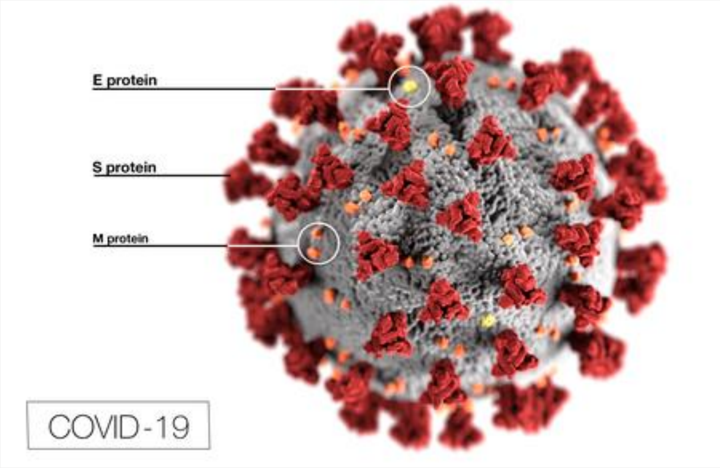
Nature Method of the year 2015

How good can cryo-EM become?

Robert M Glaeser

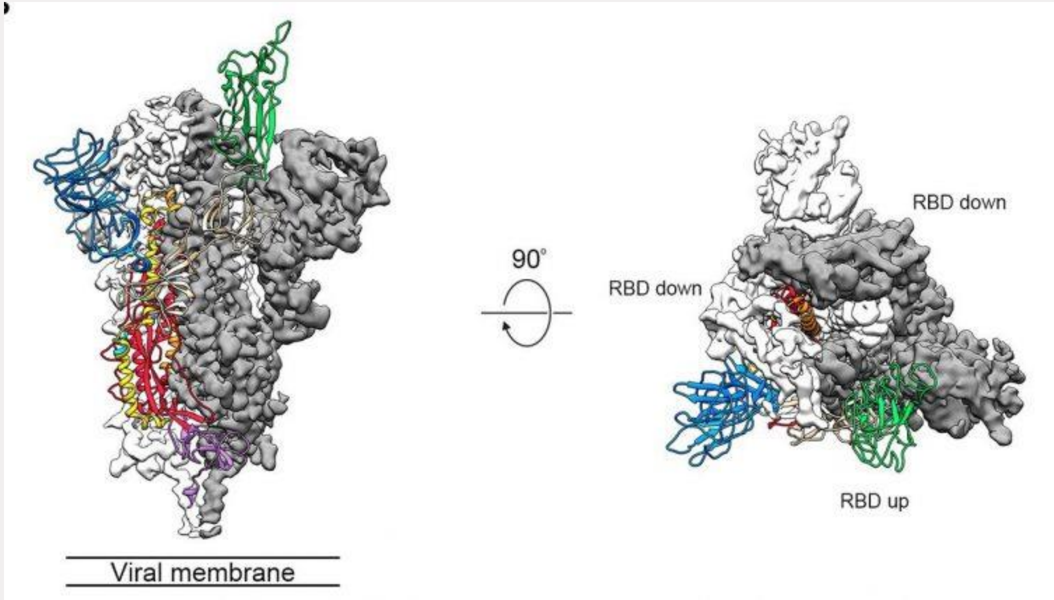


unit-magnitude, digital count. At present, the FEI Falcon camera uses a 14-μm pixel size, whereas the Gatan K2 camera, with 5-μm pixels, converts the electron events into counts. If these two features were combined, the resulting signal-to-noise ratio (SNR) in the camera output would finally come close to the irreducible 'shot-noise' limit, which is due to the random



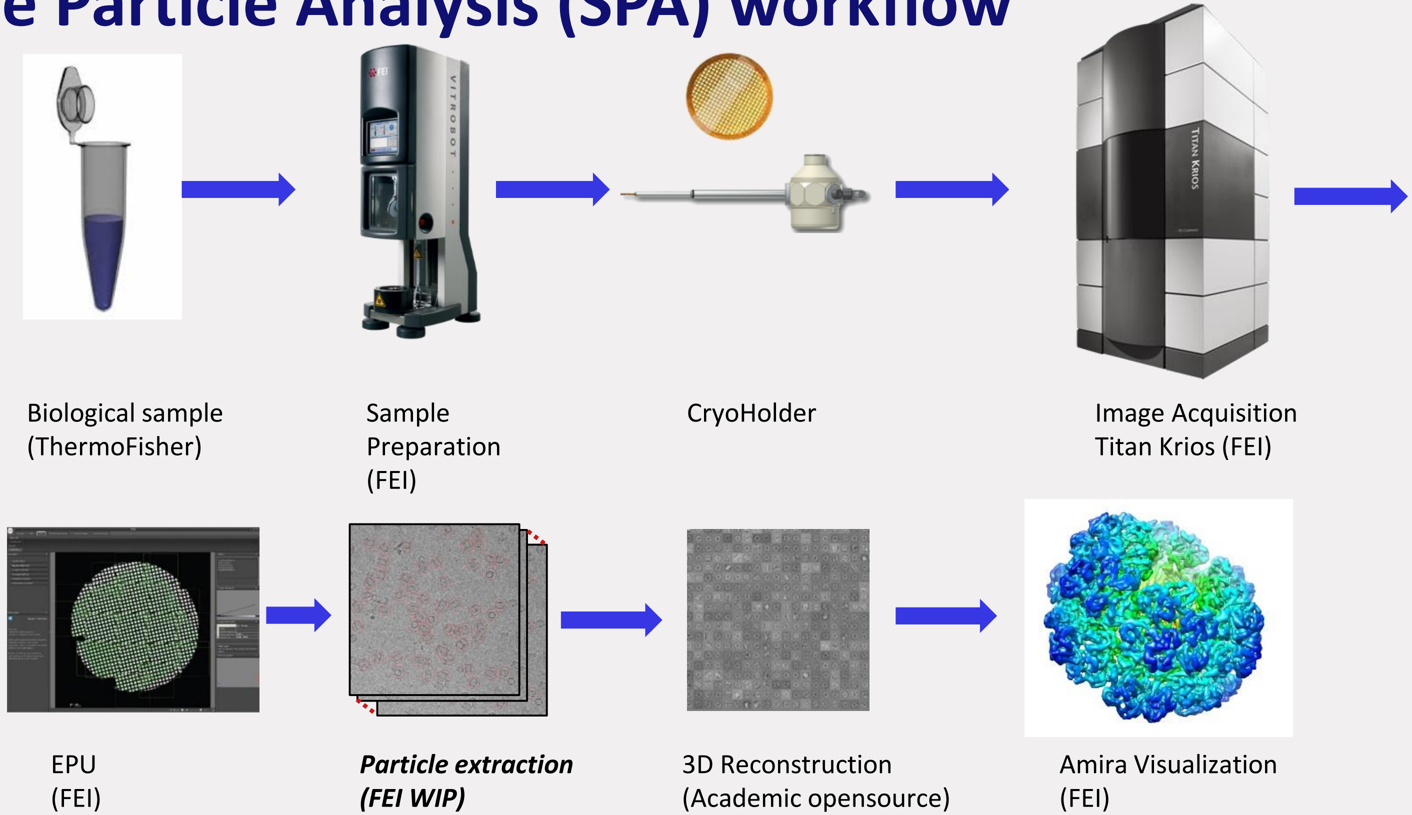
The key method to study the mechanics of the spikes of the new Corona virus

Detailed ACE2 receptor: How does it bind to the human cell?



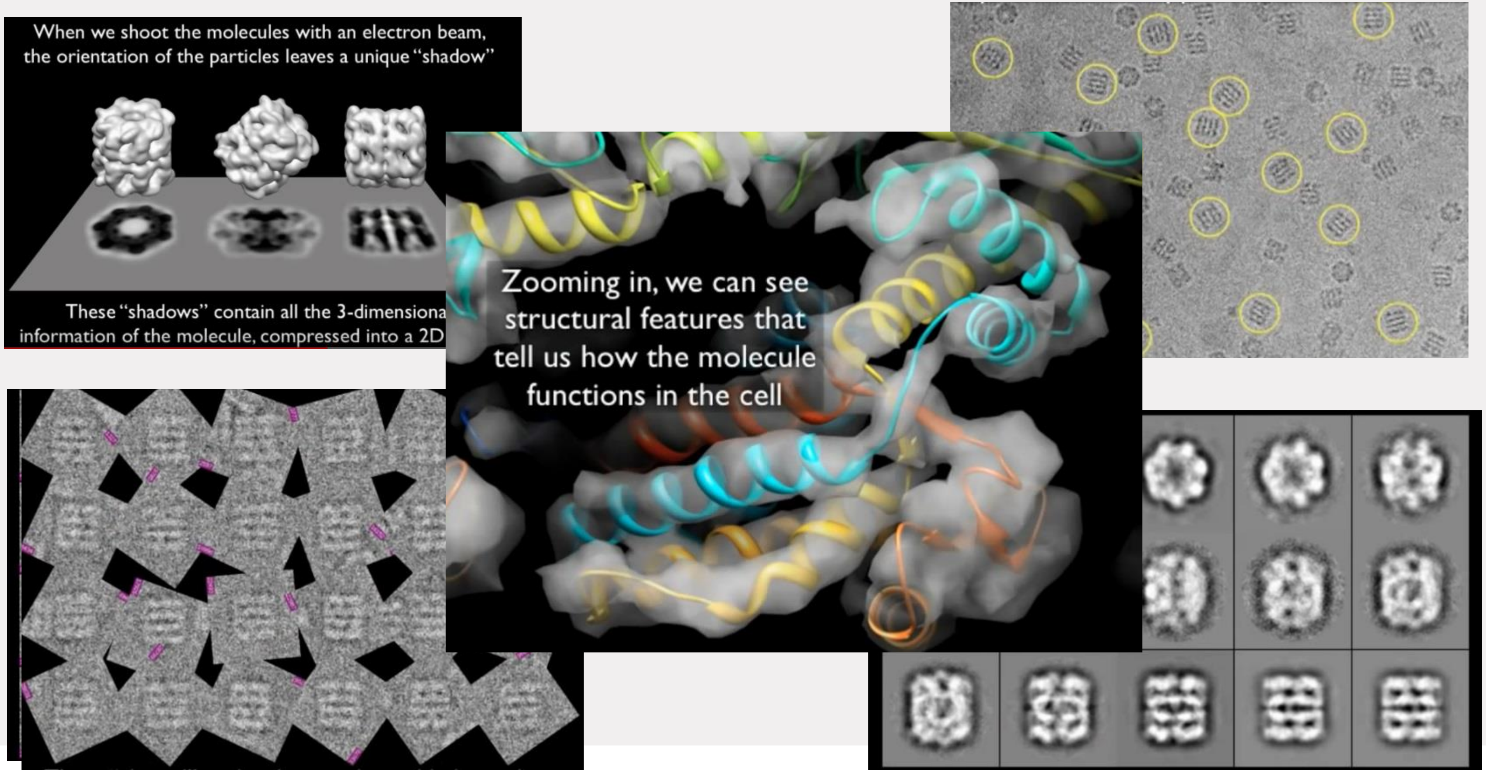
Side and top views of the pre-fusion structure of the COVID-19 S protein with a single RBD in the up conformation. The two RBD-down protomers are shown as cryo-EM density in either white or grey and the RBD-up protomer is shown in ribbons coloured green (credit: adapted from Wrapp, D, et al.).

Single Particle Analysis (SPA) workflow

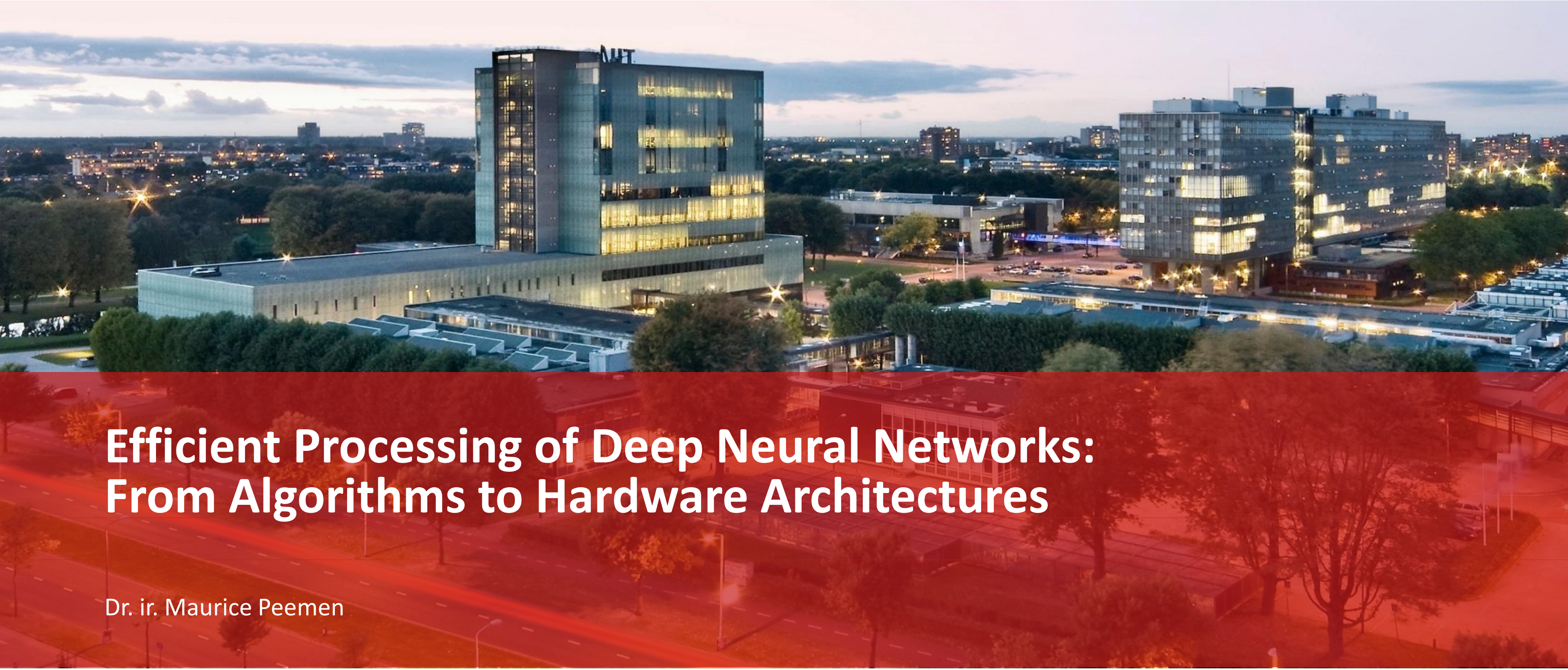


TU/e

Image processing challenges in SPA workflow



Large volume data acquisition – Mapping the brain



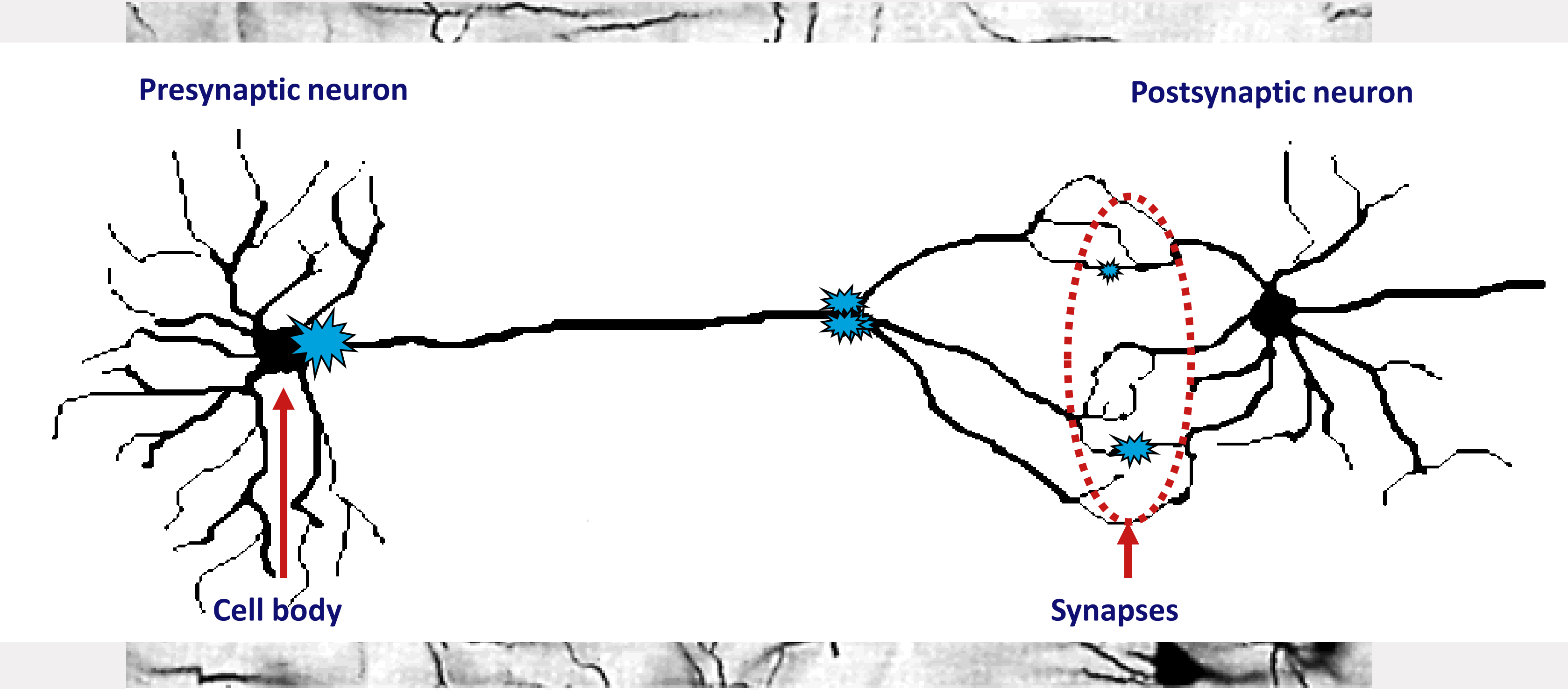
Efficient Processing of Deep Neural Networks: From Algorithms to Hardware Architectures

Dr. ir. Maurice Peemen

Electrical Engineering

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

What are Neural Networks?



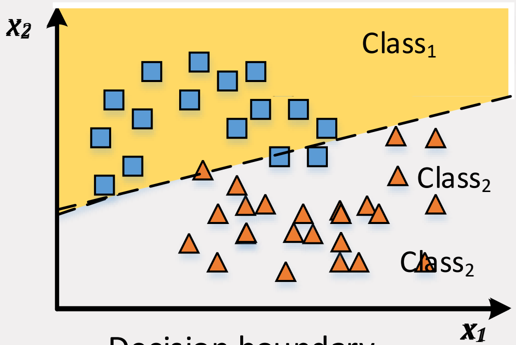
Perceptron Model (1957)

Feed forward processing

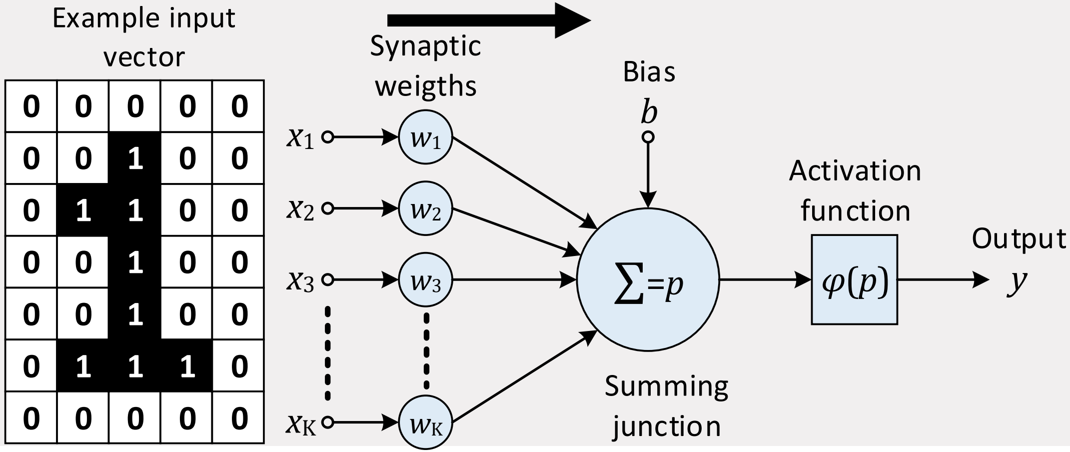
Tuning the weights by learning

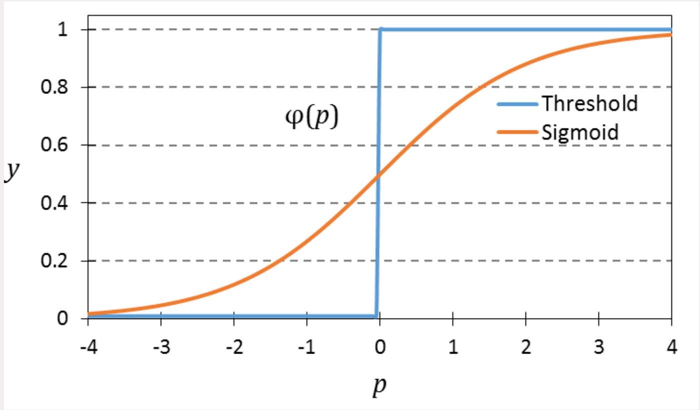
Non-linear separability (1969)

$$y = \varphi\left(b + \sum_i x_i \cdot w_i\right)$$



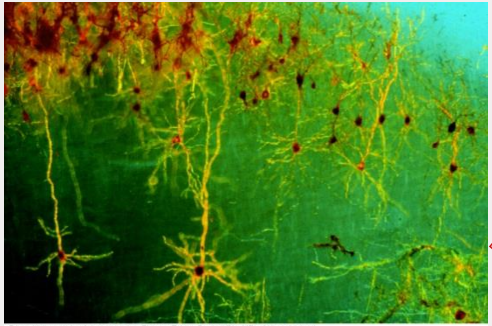
Decision boundary
$w_1x_1 + w_2x_2 + b = 0$





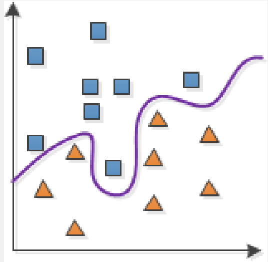
Convergence of different domains

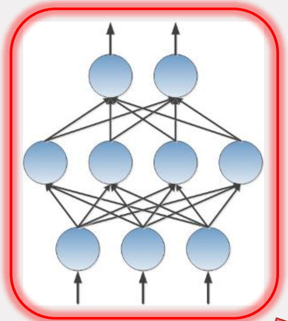
Neurobiology



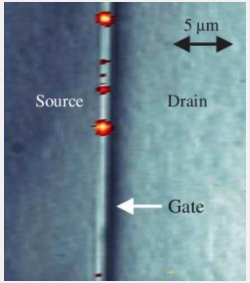
Pyramidal neurons of mouse cortex. Picture: Tobias Bonhoeffer, MPI of Neurobiology

Machine Learning



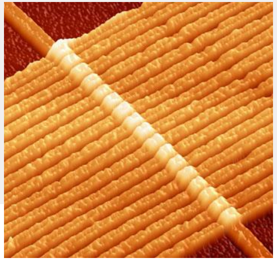


Applications

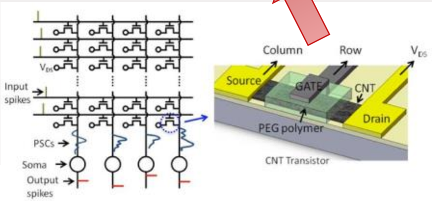


Constraints

Technology



Innovations

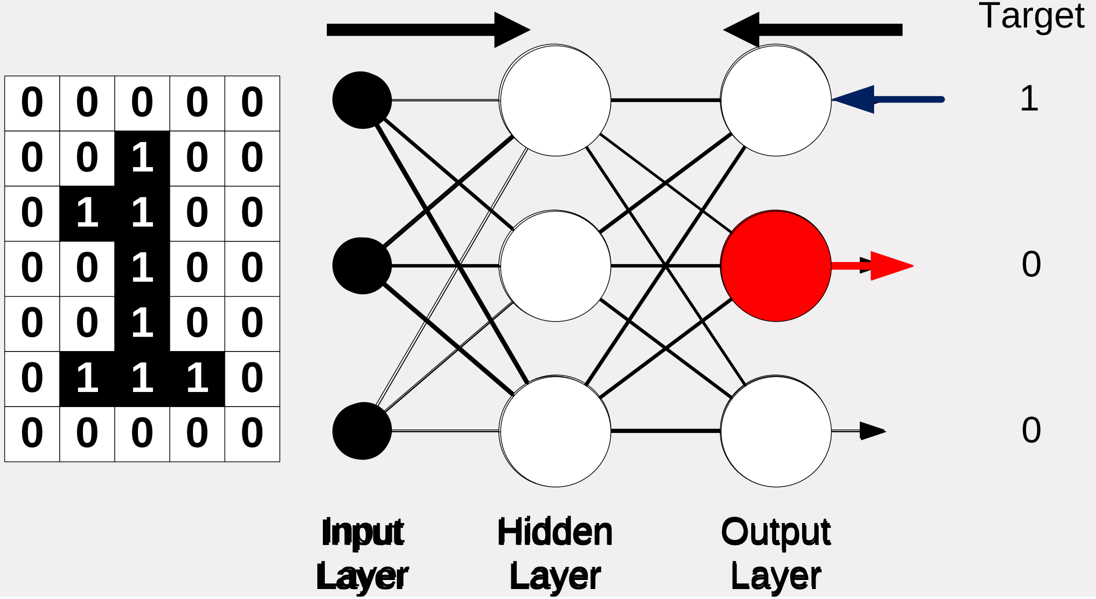


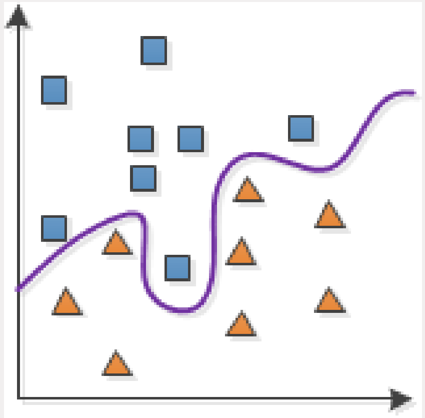
Neuromorphic

Multi Layer Perceptron (1979)

Training is done by error back-propagation

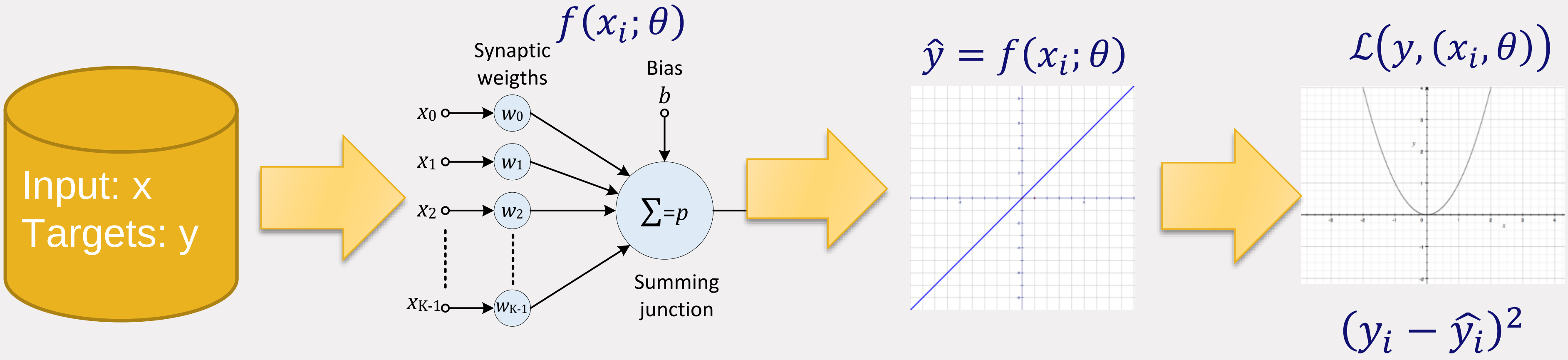
$$W_{new} = W_{old} - \eta \frac{\partial E_{CE}}{\partial W_{old}}$$





Training a network: Stochastic Gradient Descent (SGD)

- Collect annotated training data
- The neural network is a function of inputs x_i and weights θ: $f(x_i;\theta)$
- Start with feed forward run through the network: $x_i \rightarrow \widehat{y_i}$
- Evaluate predictions, i.e. results and labels into a loss function: $\mathcal{L}\big(y, f(x_i, \theta)\big)$



Training a network: Stochastic Gradient Descent (SGD) 2

- Compute the error gradients
- Update the coefficients to reduce error
- Repeat

$$\theta^{t+1} = \theta^{t} - \eta_t \nabla_\theta \mathcal{L}$$

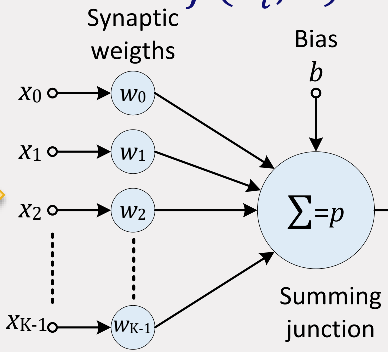
$$\frac{\partial f(x_i;\theta)}{\partial \theta}$$

$$\frac{\partial \hat{y}}{\partial f}$$

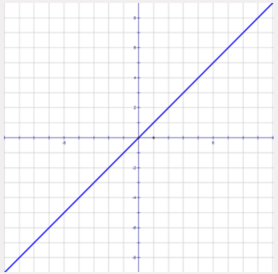
$$\frac{\partial \mathcal{L}(\theta;\hat{y},f)}{\partial \hat{y}}$$

Input: x
Targets: y

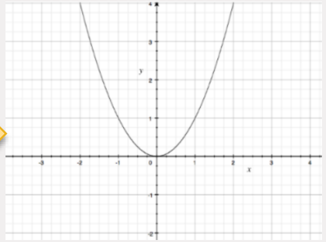
$$f(x_i;\theta)$$



$$\hat{y} = f(x_i;\theta)$$



$$\mathcal{L}(\theta;y,f)$$



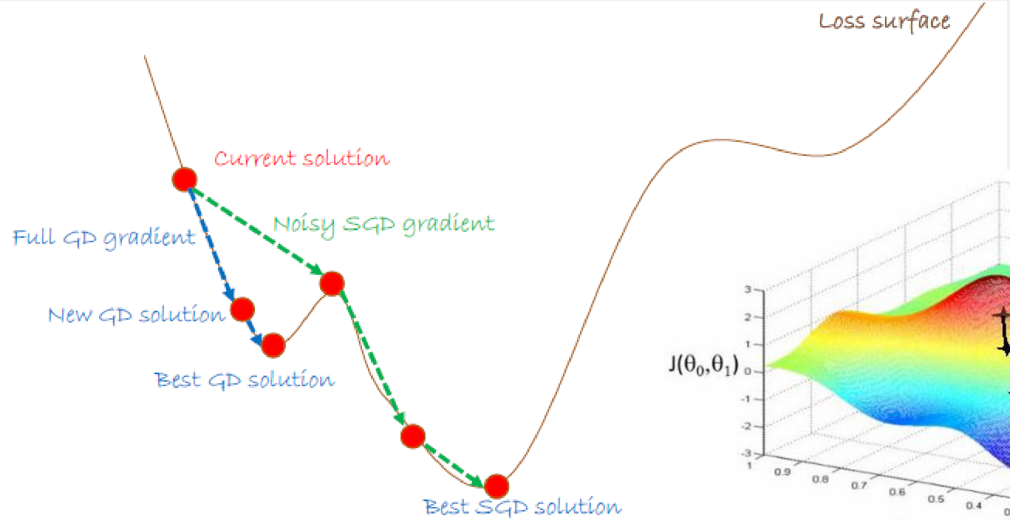
$$(y_i - \hat{y}_i)^2$$

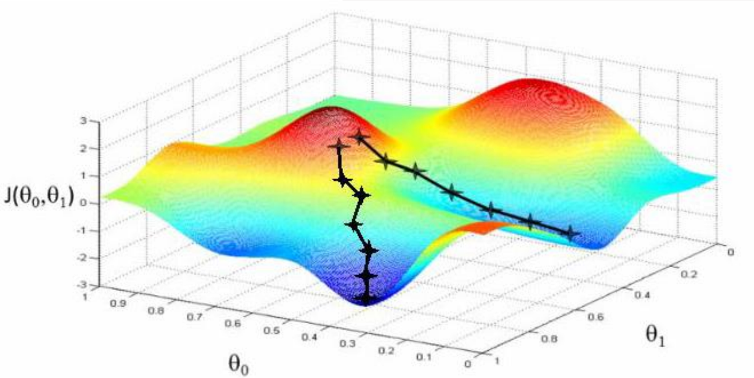
Optimization Through Gradient Descent (3)

Follow the path towards local minima in the parameter space

Stochastic Gradient Descent

Every random sample update the parameter space





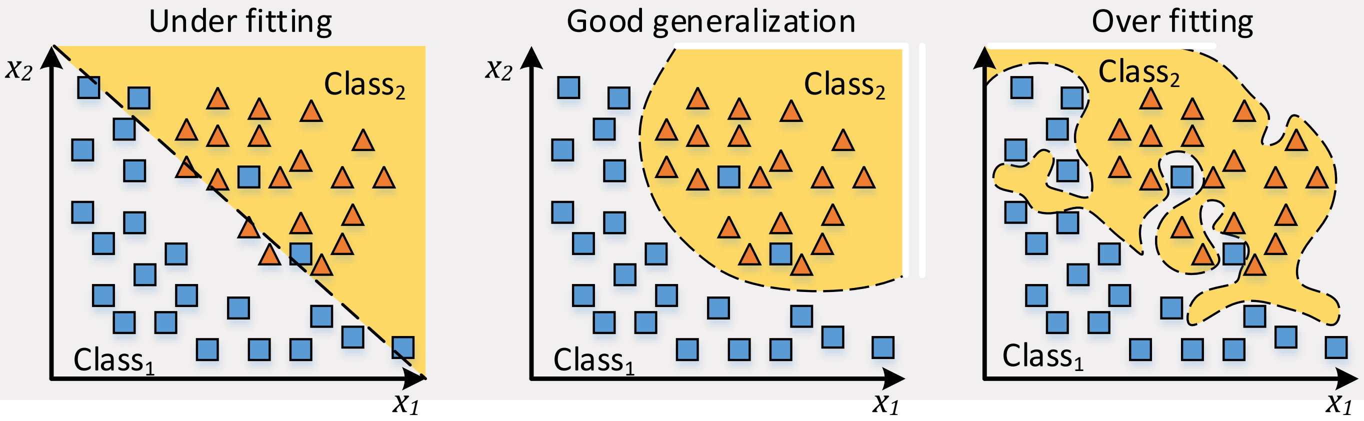
Generalization

Take an abstract representation

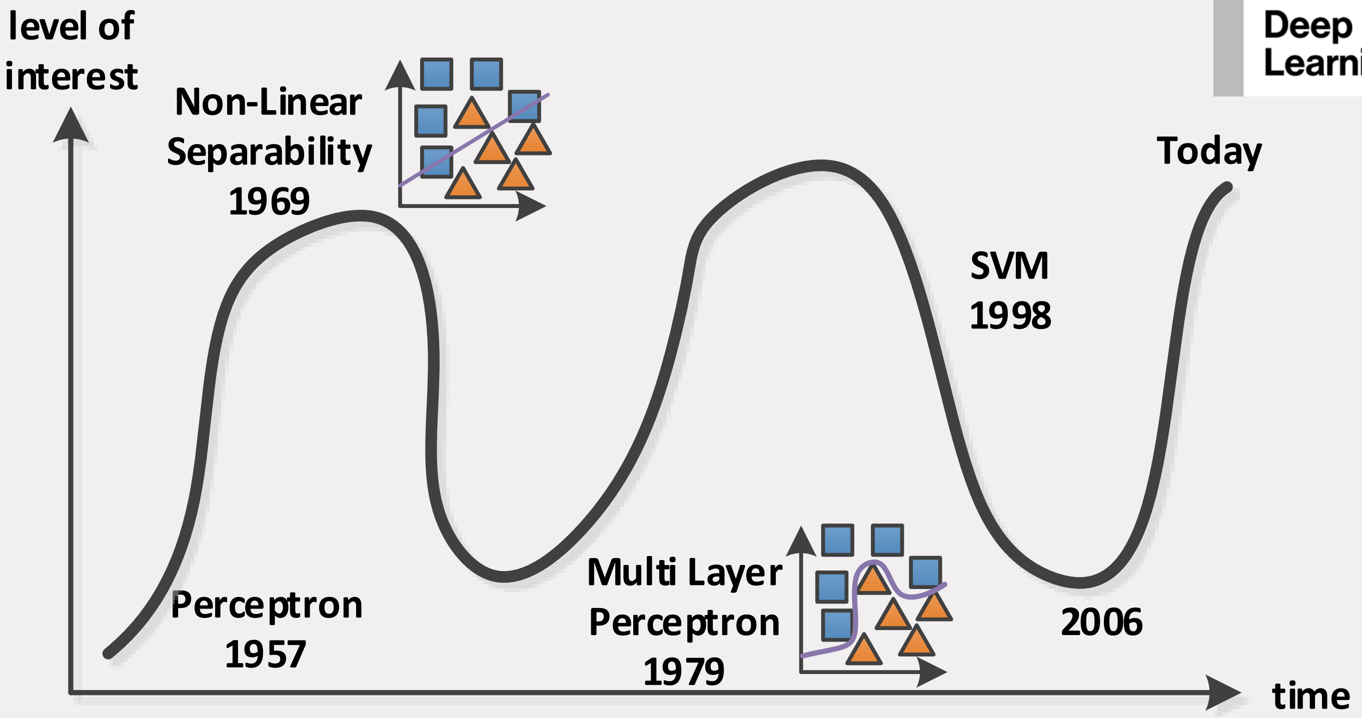
Add details

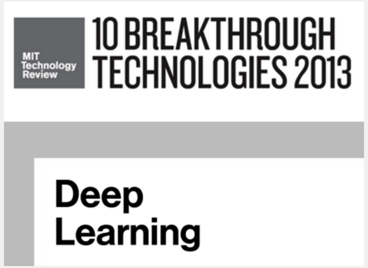
Add too many details

A common challenge in Machine Learning problems



The Hype Curve of Neural Networks





Deep Big Neural Networks

Deep Big Neural networks outperform SVM

Complex models with over a billion of parameters

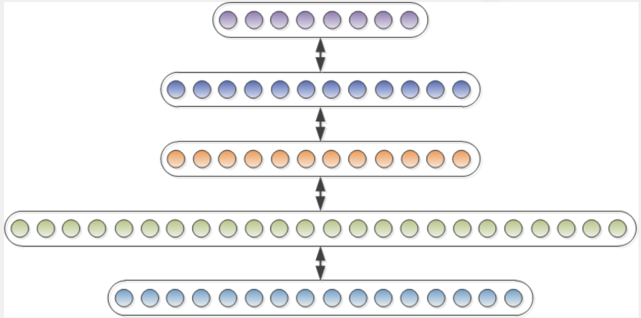
Unreasonably effective at classification tasks

Why would this be so effective?

≥ 5 layers

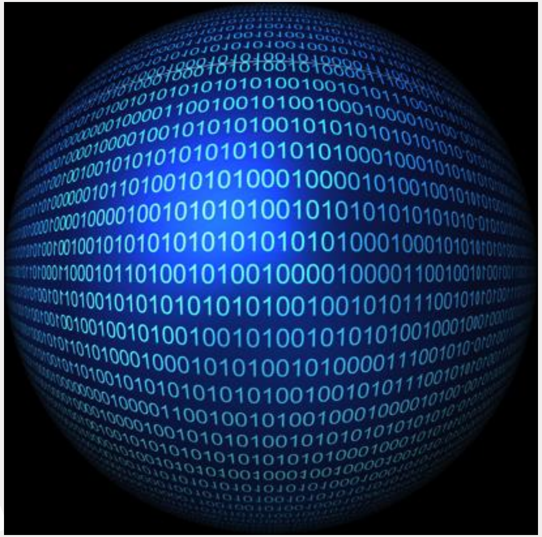
1000s of nodes

connection constraints

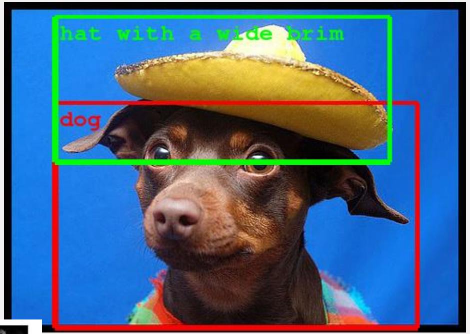


Big Deep Network

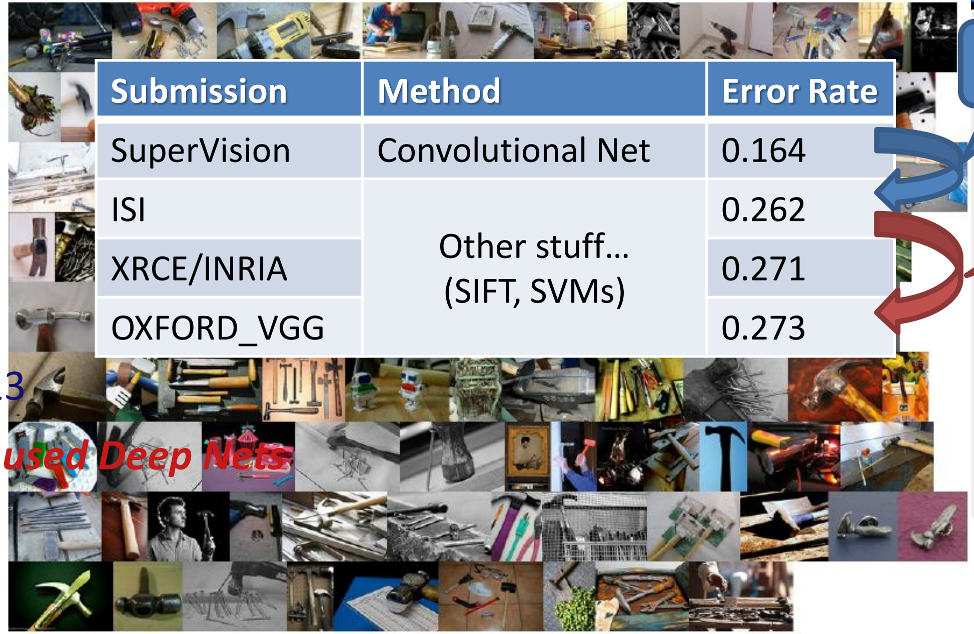
Data is Everywhere



Large Scale Visual Recognition Challenge



- IMAGENET 2012 Classification
- Many training samples: 1,281,167
- Large number of classes: 1000



Submission	Method	Error Rate
SuperVision	Convolutional Net	0.164
ISI	Other stuff… (SIFT, SVMs)	0.262
XRCE/INRIA		0.271
OXFORD_VGG		0.273

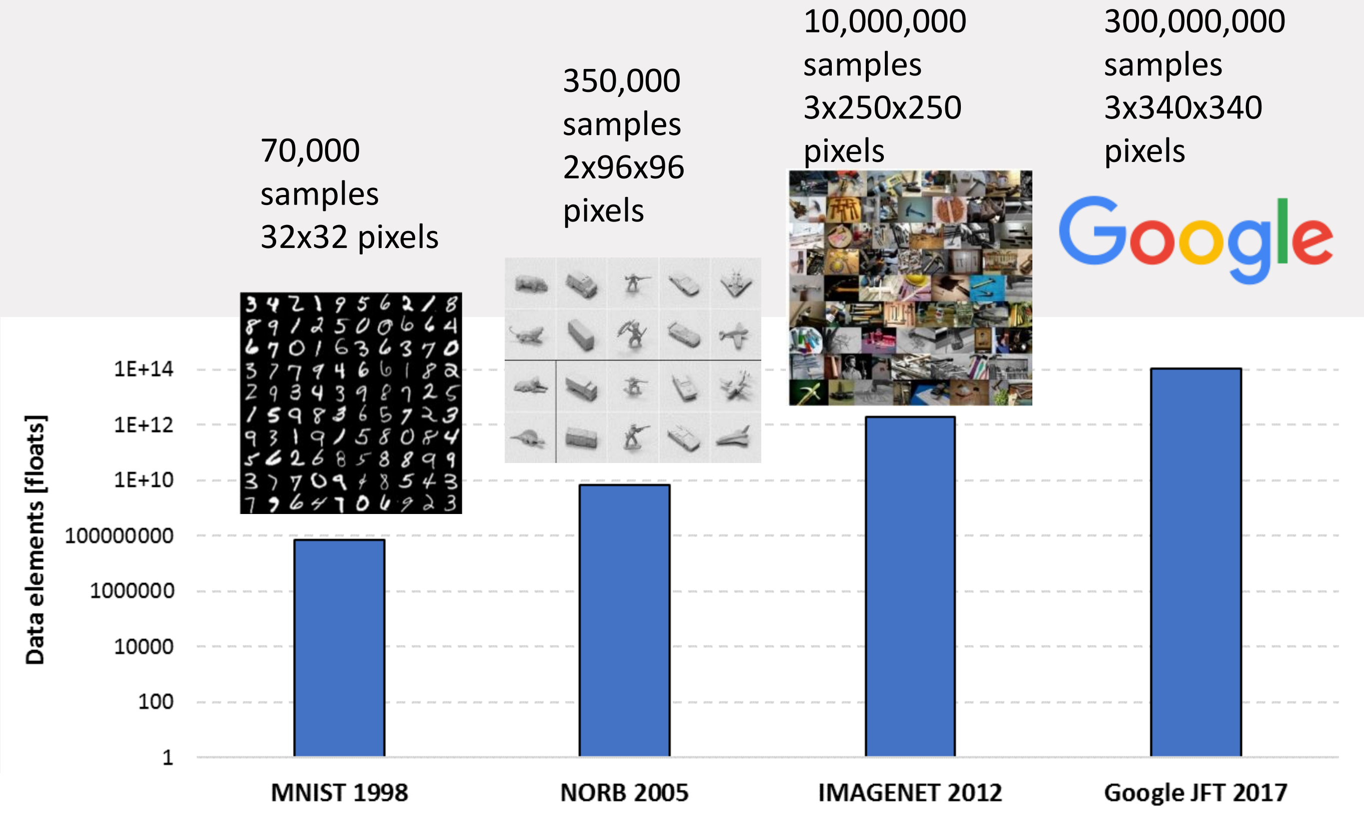
9.8%

1.2%

- ImageNet 2013

Complete top 10 used Deep Nets

Data set growth over time

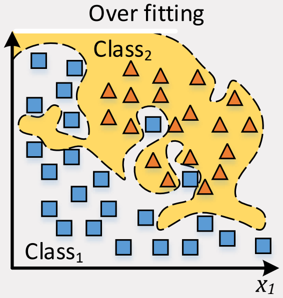


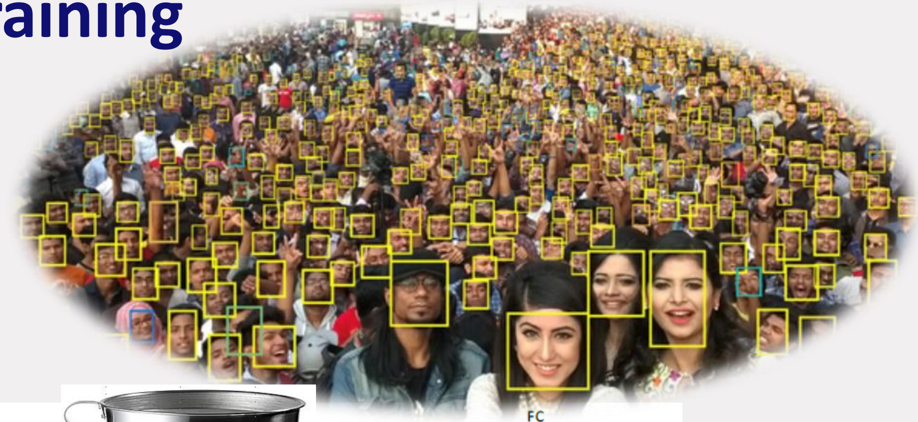
Huge amounts of data for training

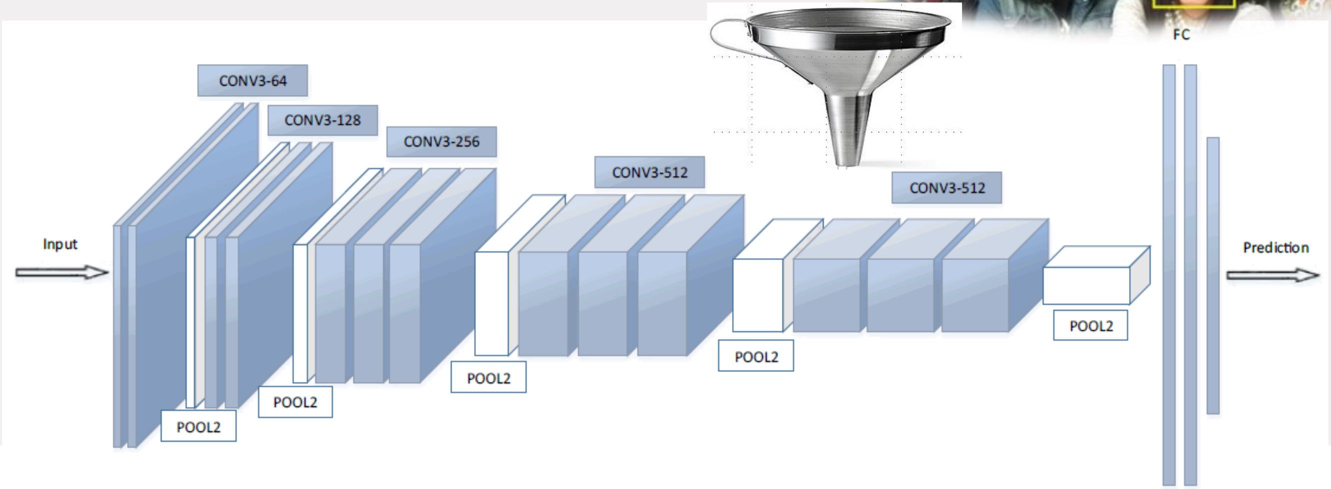
Large datasets help a lot:

- Less overfitting
- Accurate classification

Better Machine Learning

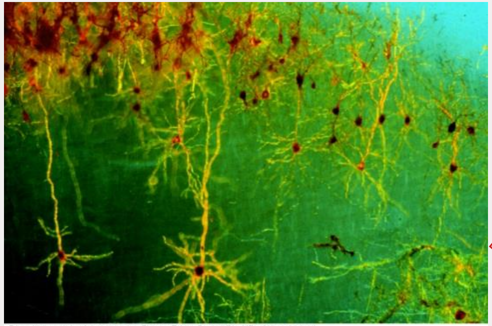






Deep Neural Networks

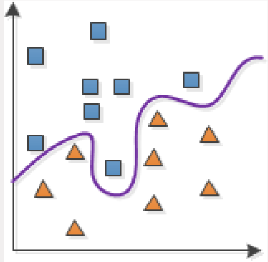
Neurobiology

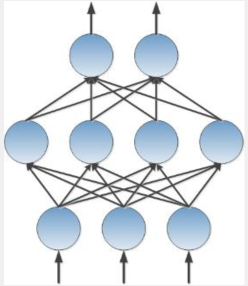


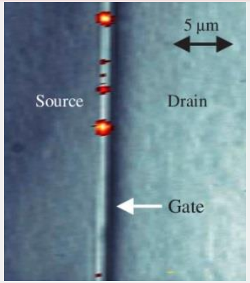
Pyramidal neurons of mouse cortex. Picture: Tobias Bonhoeffer, MPI of Neurobiology

Applications

Machine Learning

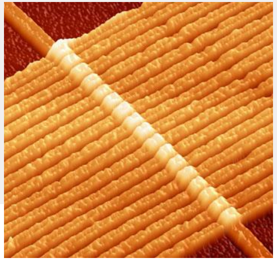




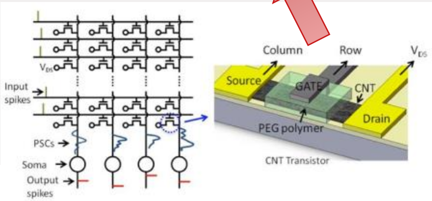


Constraints

Technology



Innovations



Neuromorphic

Deep learning applications are changing our lives

Self-driving cars

AlphaGo

Live machine translation

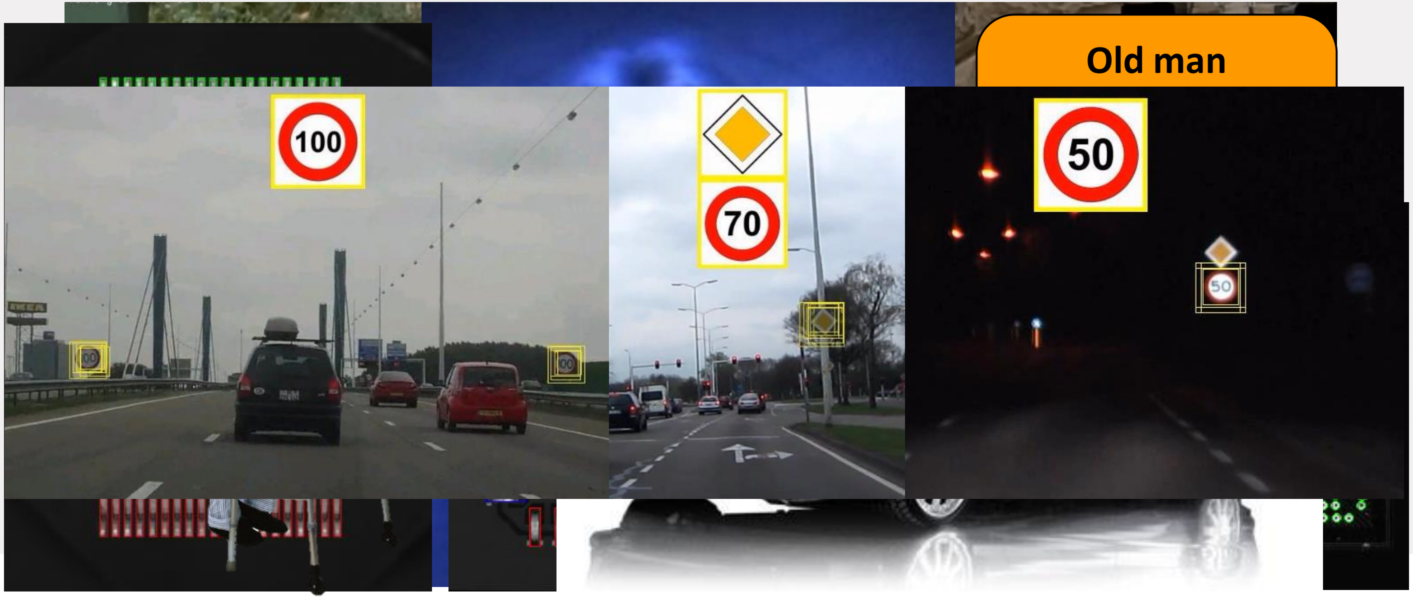
Smart robots

Face Detection

Intelligent vision applications are everywhere

Applications in many domains

Examples: Security, Industrial, Medical, Automotive



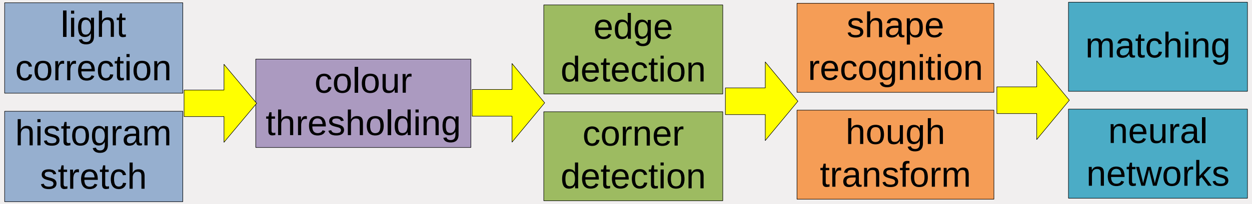
Classical recognition systems are developer limited

Design is based on knowledge of the task

Carefully tuned pipeline of algorithms

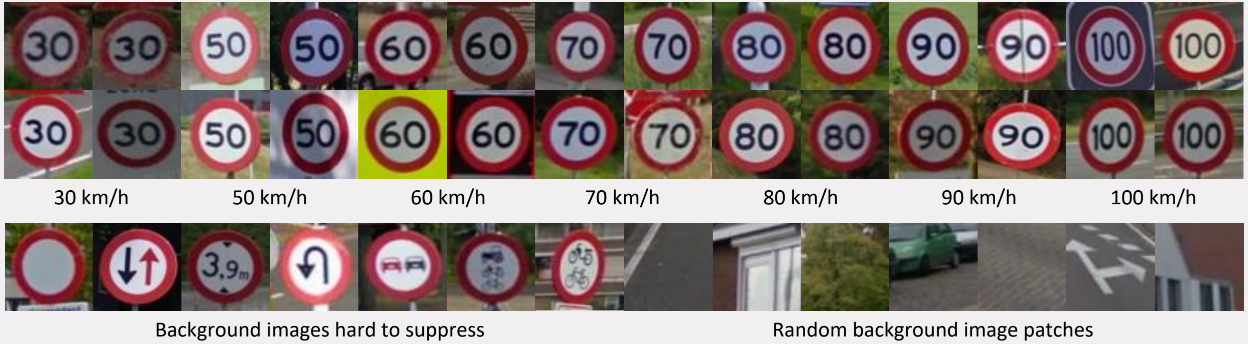
Really complex for real world problems

Design must be redone if the task changes

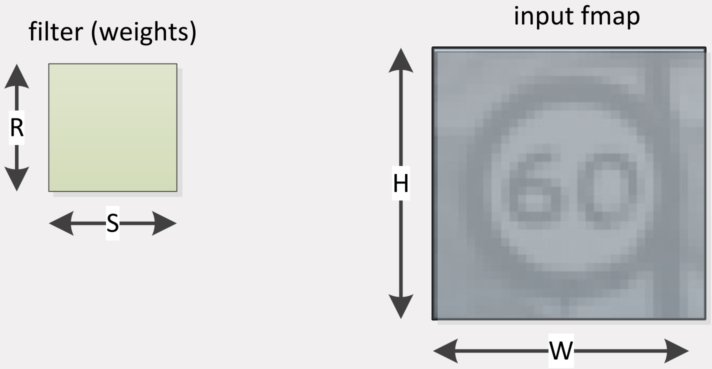


Train a Neural Network for the task

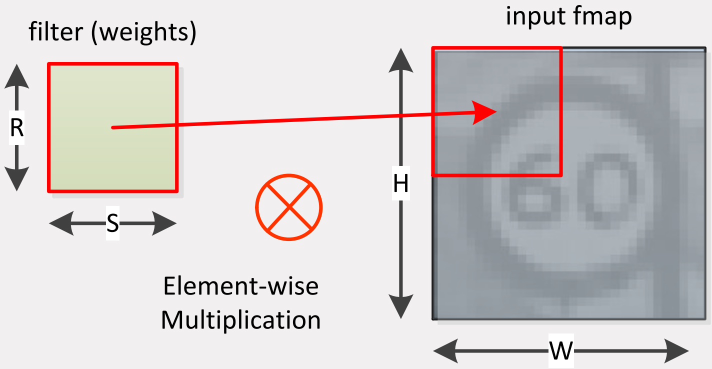
- Focus on data instead of algorithm complexity
- Pre-process data to generate more examples
- Use a test set to verify generalization



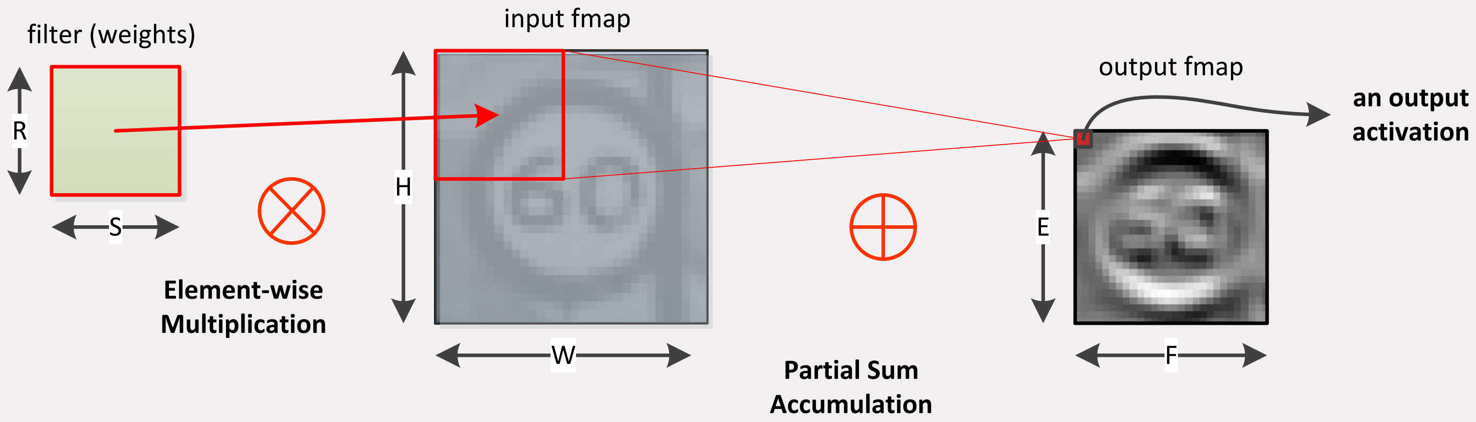
Building more restricted neurons



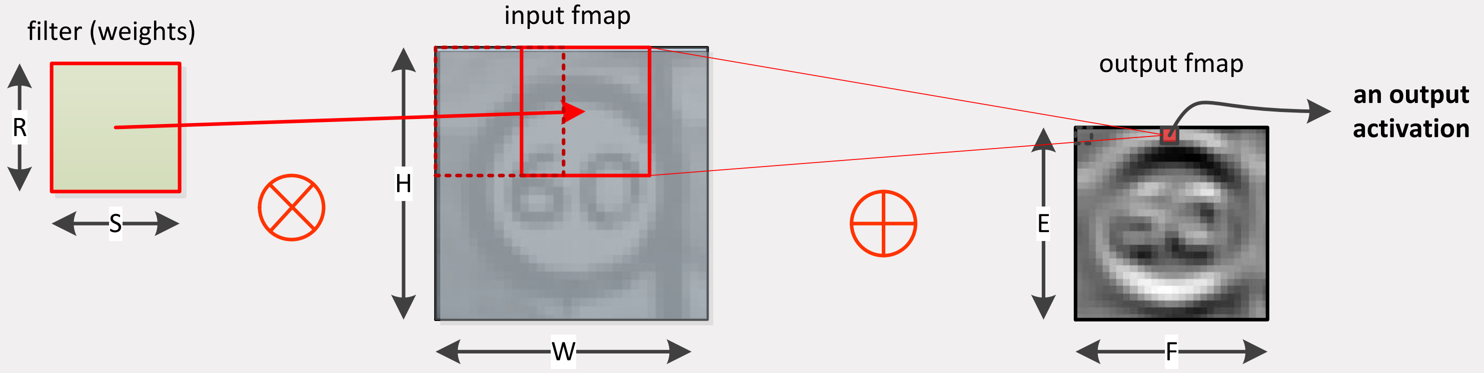
Building more restricted neurons



Building more restricted neurons



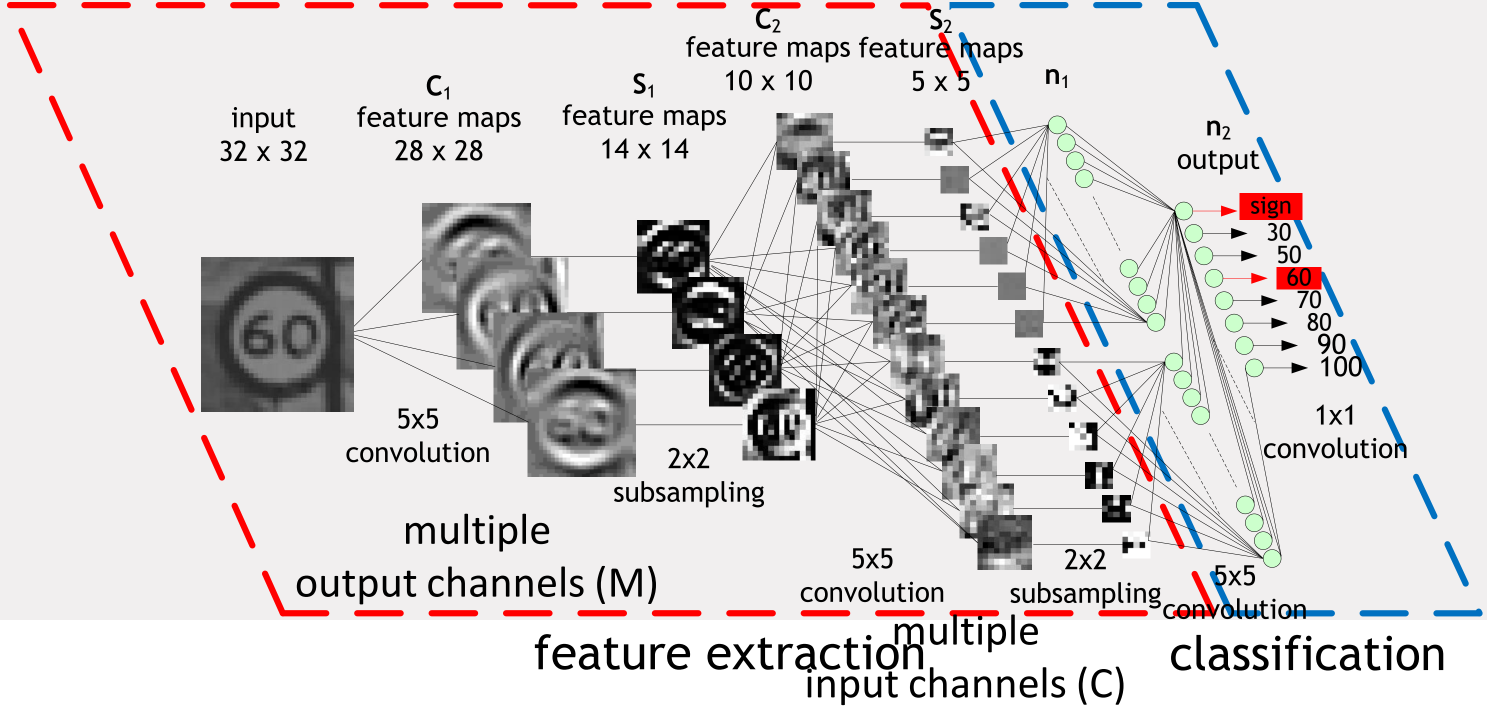
Building more restricted neurons



Sliding window processing

Biologically inspired object recognition

Convolutional Neural Network



Detection and Recognition Application

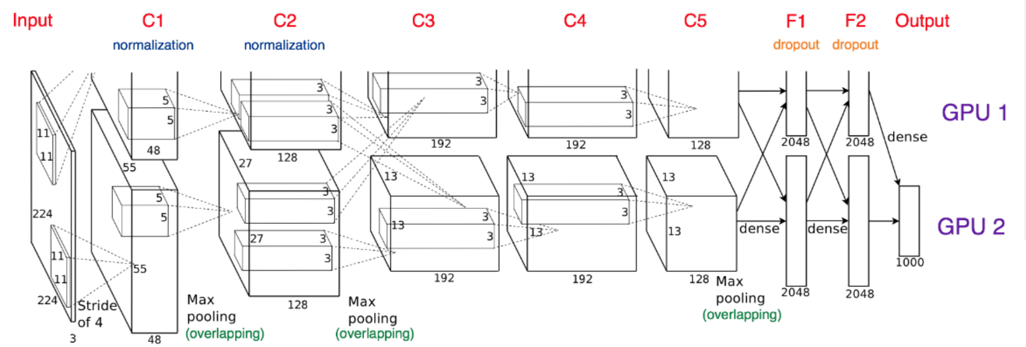
Define shape for each layer

Shape varies across layers

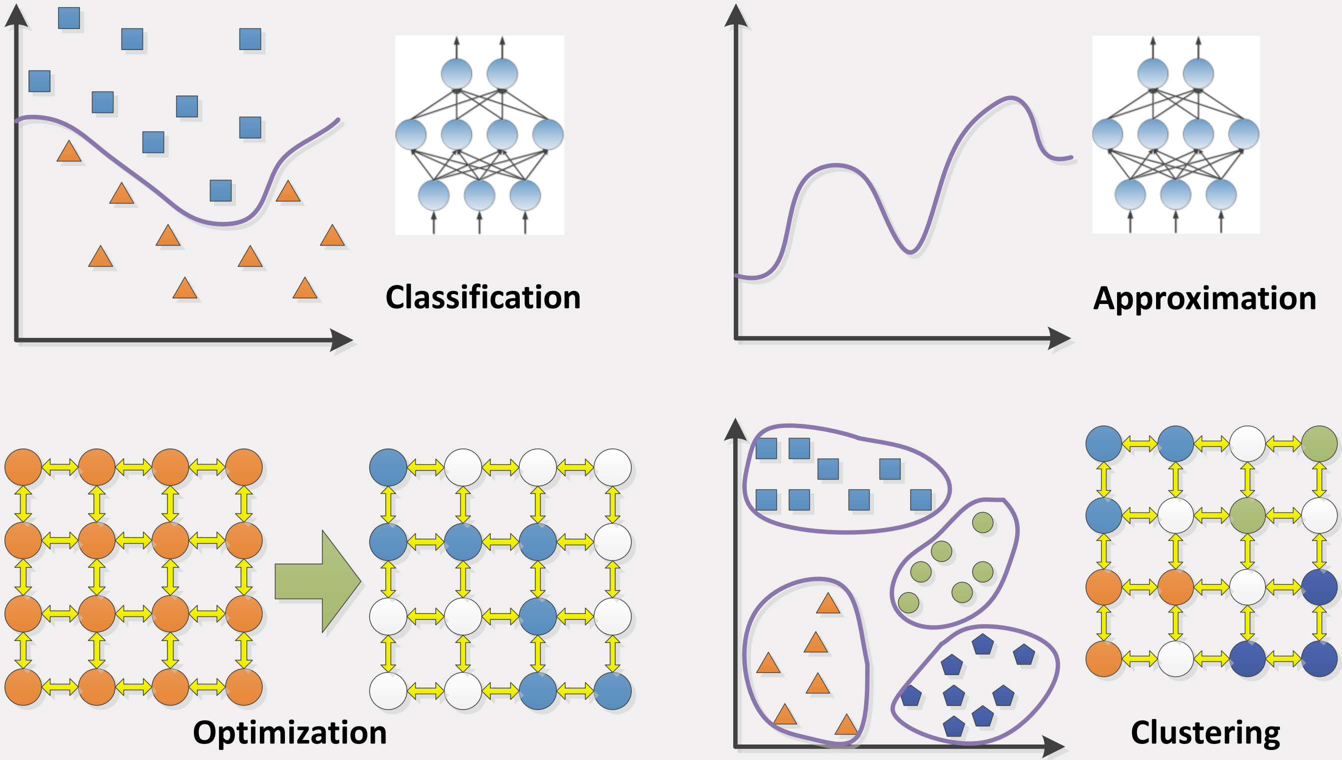
- H – Height of input fmap (activations)
- W – Width of input fmap (activations)
- C – Number of 2D input fmaps / filters (channels)
- R – Height of 2D filter (weights)
- S – Width of 2D filter (weights)
- M – Number of 2D output fmaps (channels)
- E – Height of output fmap (activations)
- F – Width of output fmap (activations)
- N – Number of input fmaps/output fmaps (batch size)

AlexNet (2012) ILSVRC winner:

- 8 layers, 62Mparameters
- 1.4 GFLOP inference



What Else Can Deep Neural Nets Do?



Application: Super Resolution from NNs

Low Resolution Image

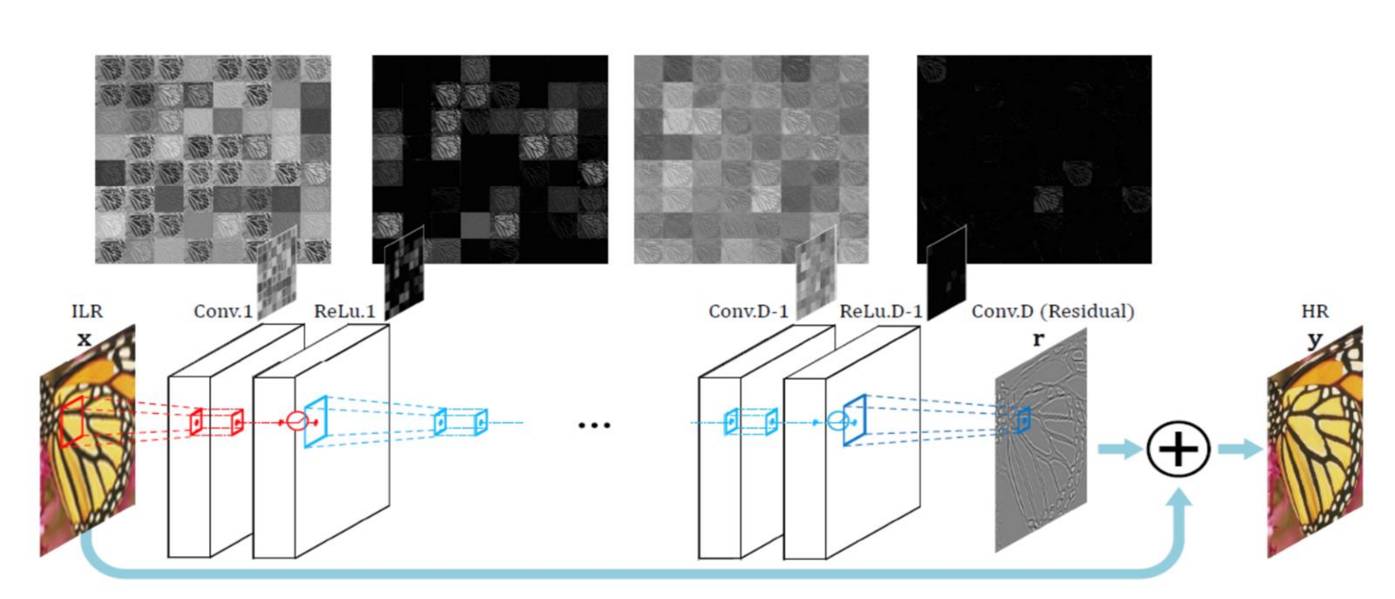
Input Image

Output Image

Kim et al. “Accurate Image Super-Resolution Using Very Deep Convolutional Networks” CVPR16

Very Deep Super Resolution: VDSR

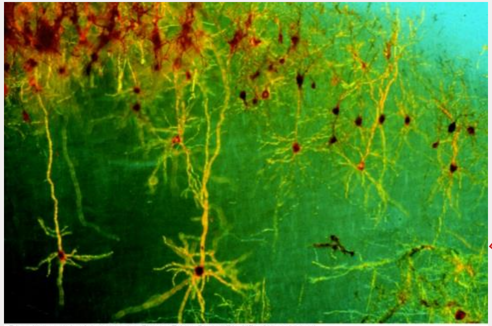
- Residual learning



Kim et al. “Accurate Image Super-Resolution Using Very Deep Convolutional Networks” CVPR16

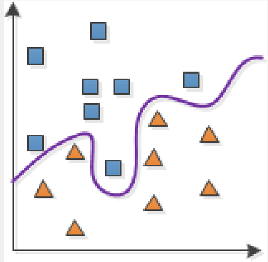
Deep Neural Networks

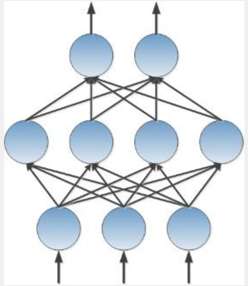
Neurobiology

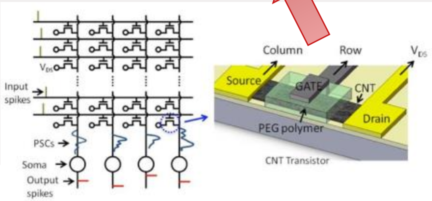


Pyramidal neurons of mouse cortex. Picture: Tobias Bonhoeffer, MPI of Neurobiology

Machine Learning

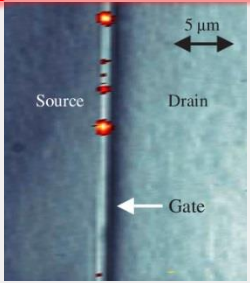






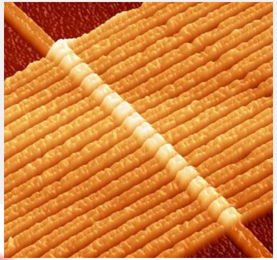
Neuromorphic

Applications



Constraints

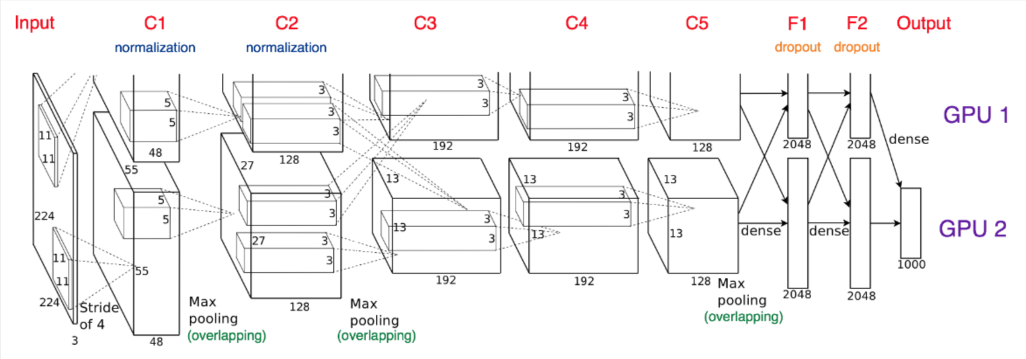
Technology



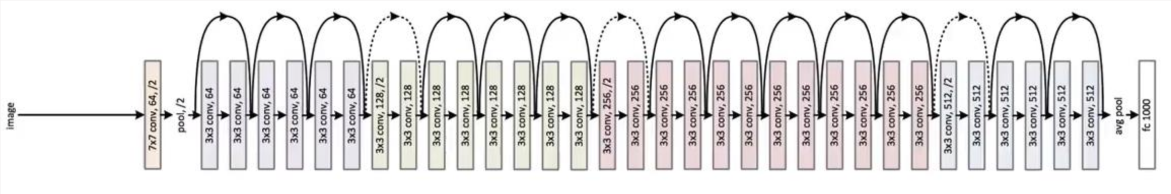
Innovations

Deep learning models have become huge

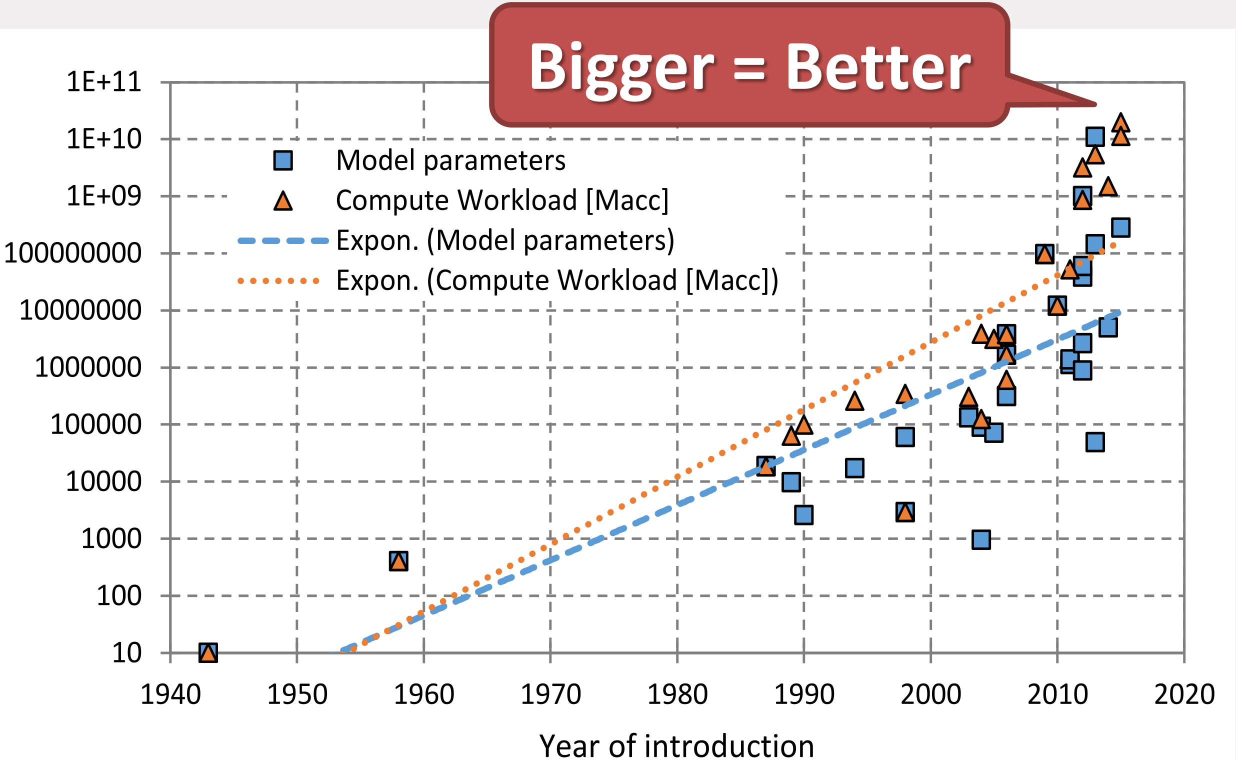
- Image Recognition
- AlexNet (2012) ILSVRC winner
  - 8 layers, 62Mparameters
  - 1.4 GFLOP inference
  - 16% error rate



- ResNet (2015) ILSVRC winner
  - 152 layers, 60Mparameters
  - 22.6 GFLOP inference
  - 6.16% error rate

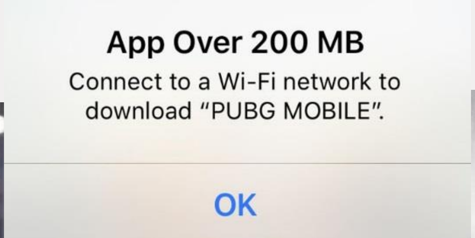


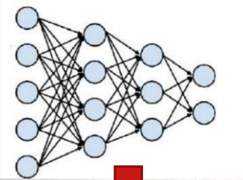
Trends in Deep Learning



The first challenge: Model Size

- Competition winning networks like AlexNet (2012) and ResNet152 (2015) are large
  - About 60M parameters resulting in a coefficient file of 240MB
- Hard to distribute large models through over-the-air updates





The second challenge: Speed

- Network design and training time have become a huge bottleneck

	Error rate	Training time
ResNet 18:	10.76%	2.5 days
ResNet 50:	7.02%	5 days
ResNet 101:	6.21%	1 week
ResNet 152:	6.16%	1.5 weeks

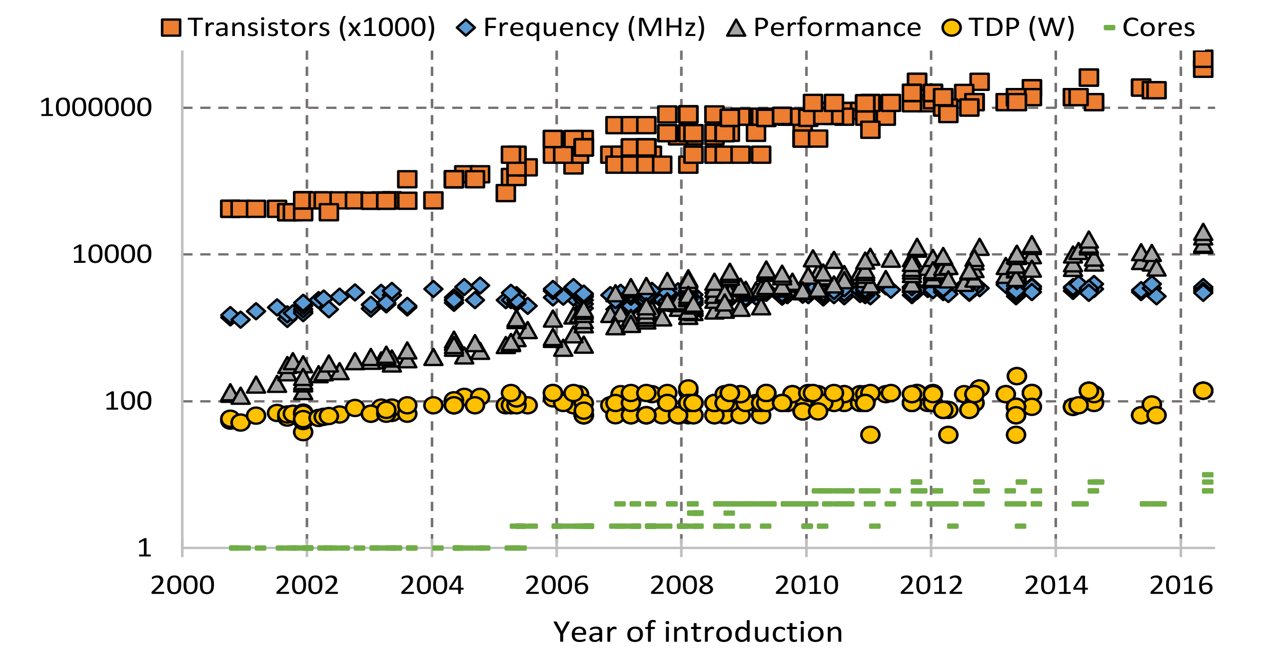
Training time benchmarked with fb.resnet.torch using four M40 GPUs

- Such a long training time limits the productivity of ML researchers

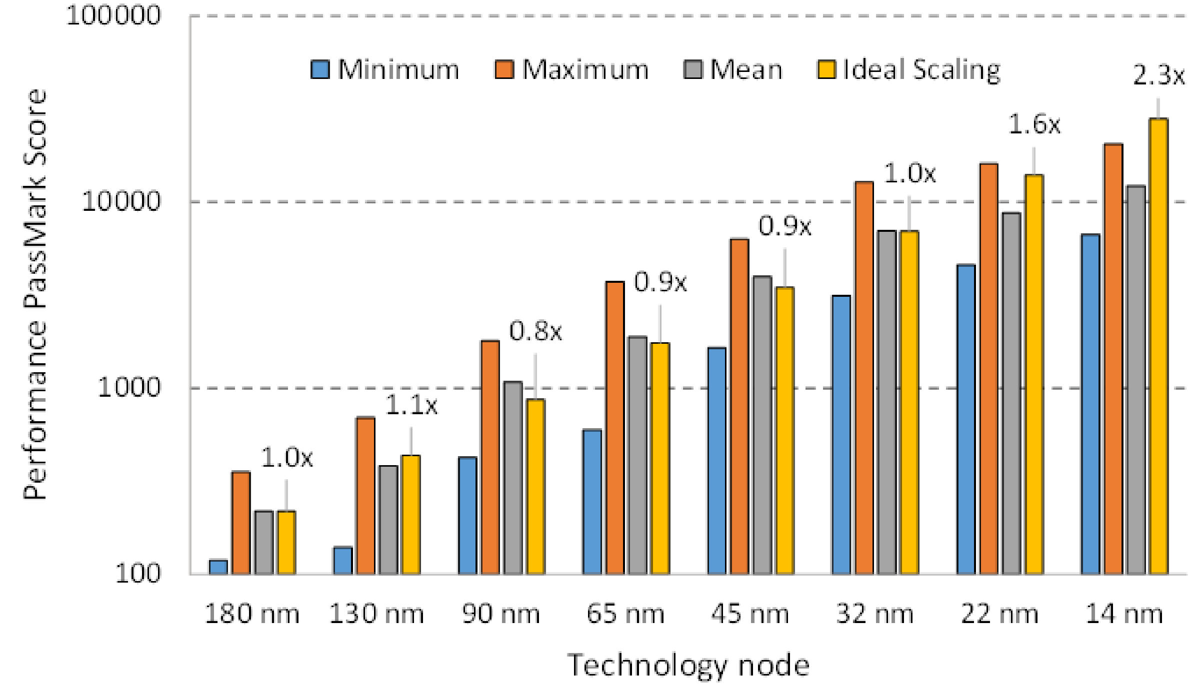
Transistors are not getting more efficient

Slowdown of Moore's law and Dennard Scaling

General purpose microprocessors not getting faster or more efficient



Performance scaling is slowing down



TU/e

The third challenge: Energy Efficiency

AlphaGo **1920 GPUs and 280 GPUs, $3000 electric bill per game**

On smartphone: battery drains quickly
On data-center: Increased TCO

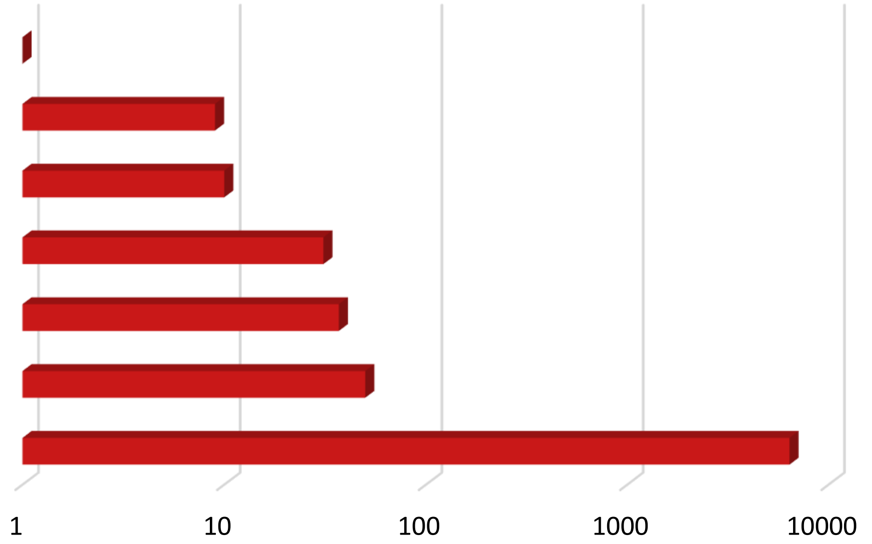
TU/e

Where is the Energy Consumed?

- Larger model → More memory references → More energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit fload ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

Relative Energy Cost



Where is the Energy Consumed?

- Deep learning models with many parameters are expensive!
- Reduce parameters to improve efficiency

1 External Memory Transfer

=

1000 Multiply or ADD operations

+ x + x + x + x + x + x + x + x + x + x +
x + x + x + x + x + x + x + x + x + x + x
+ x + x + x + x + x + x + x + x + x + x +
x + x + x + x + x + x + x + x + x + x + x
+ x + x + x + x + x + x + x + x + x + x +
x + x + x + x + x + x + x + x + x + …….

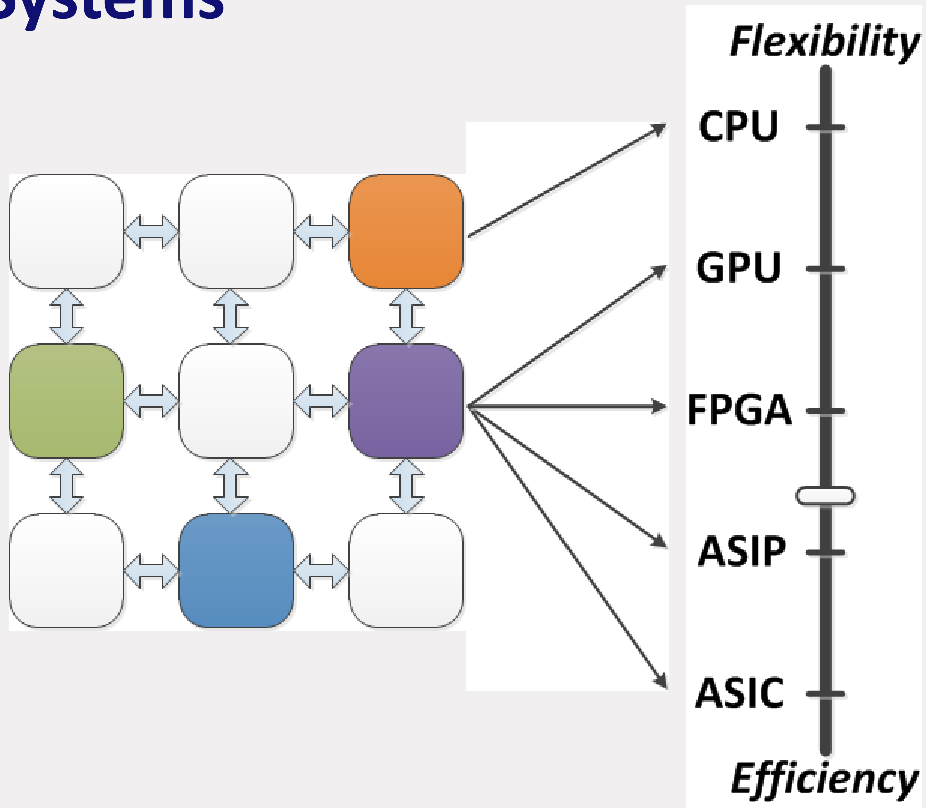
Toward Heterogeneous Systems

Efficient accelerators

Multi-purpose ASICs

Accelerators for Neural Networks

- Flexible functionality
- State-of-the-art results
- Parallelism



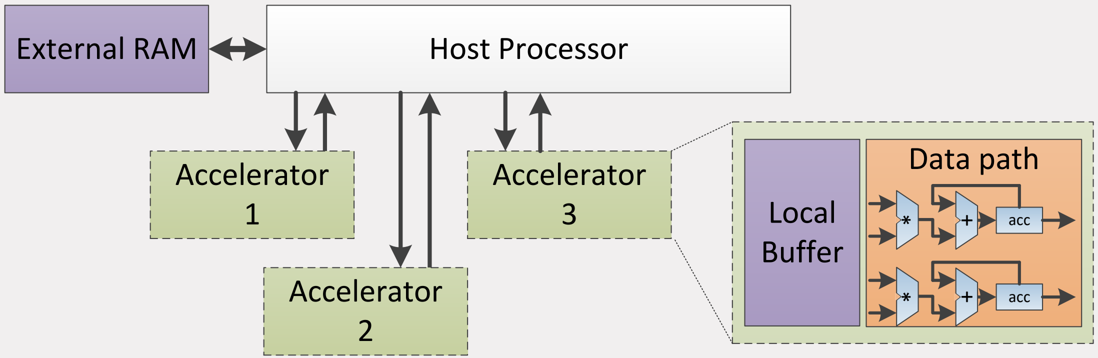
Accelerator Challenges

Data Movement: Get parameters + activations from RAM

Data movement is expensive

- Energy, latency, bandwidth

Focus on data locality



*Action	Energy	Relative
ALU op	1 pJ – 4 pJ	1x
SRAM Read	5 pJ – 20 pJ	5x
Move 10mm across chip	26 pJ – 44 pJ	25x
Send to DRAM	200 pJ – 800 pJ	200x
Read from image sensor	3.2 nJ – 4 nJ	4,000x
Send over LTE	50 uJ – 600 uJ	50,000,000 x

DianNao (2014)

Split Buffers for BW

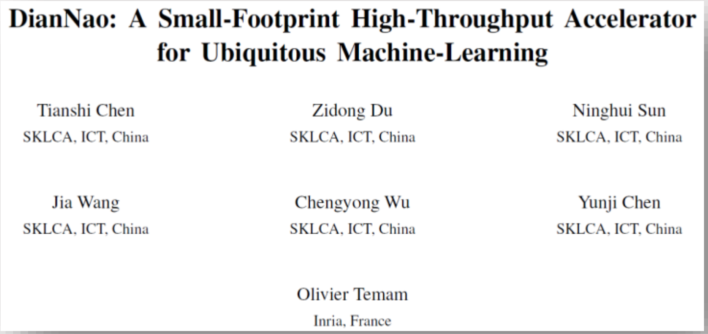
Vector vector multiply

Hardware adder tree

Partial layer computation

Fetch input neurons

Fetch synapses



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

Tianshi Chen
SKLCA, ICT, China

Zidong Du
SKLCA, ICT, China

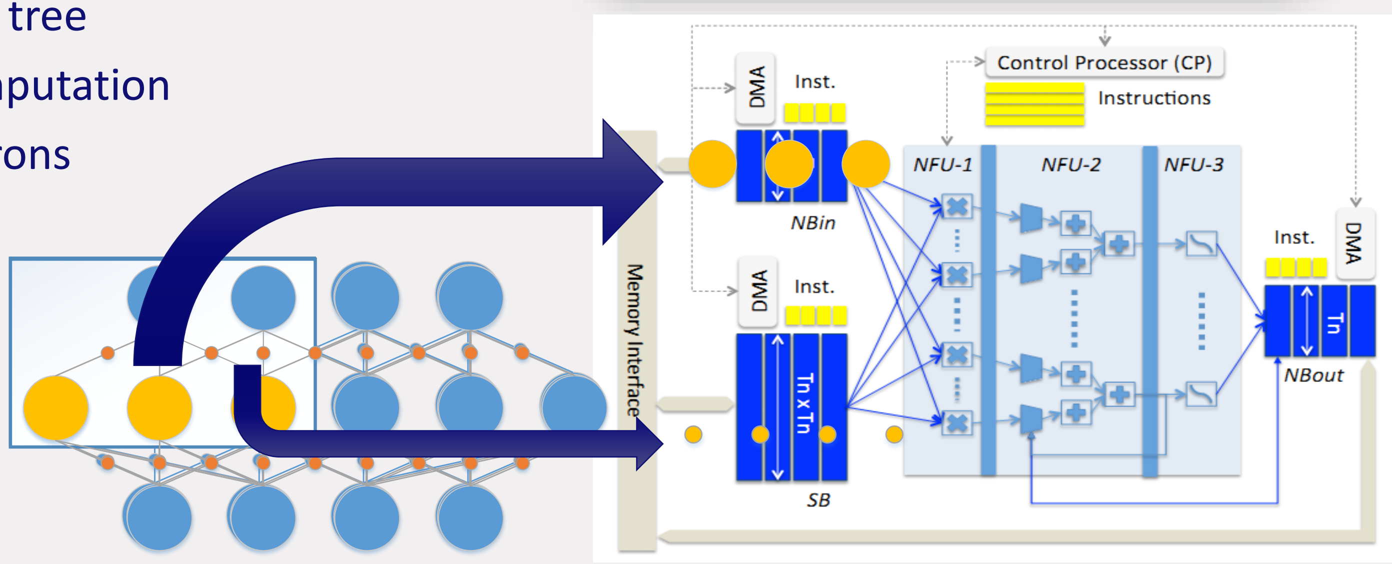
Ninghui Sun
SKLCA, ICT, China

Jia Wang
SKLCA, ICT, China

Chengyong Wu
SKLCA, ICT, China

Yunji Chen
SKLCA, ICT, China

Olivier Temam
Inria, France



DianNao (2014)

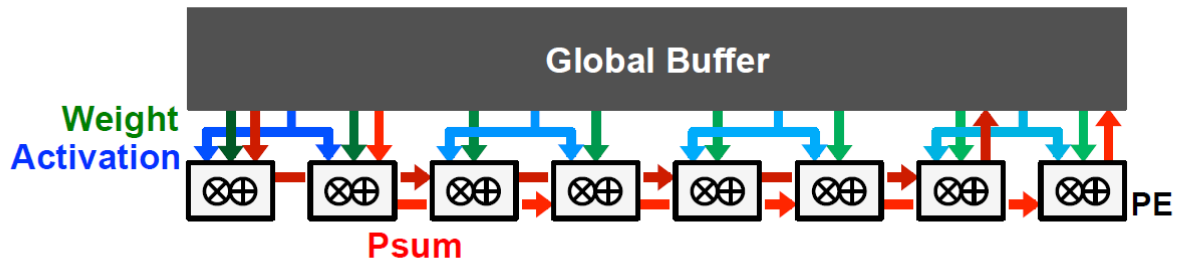
Multiply

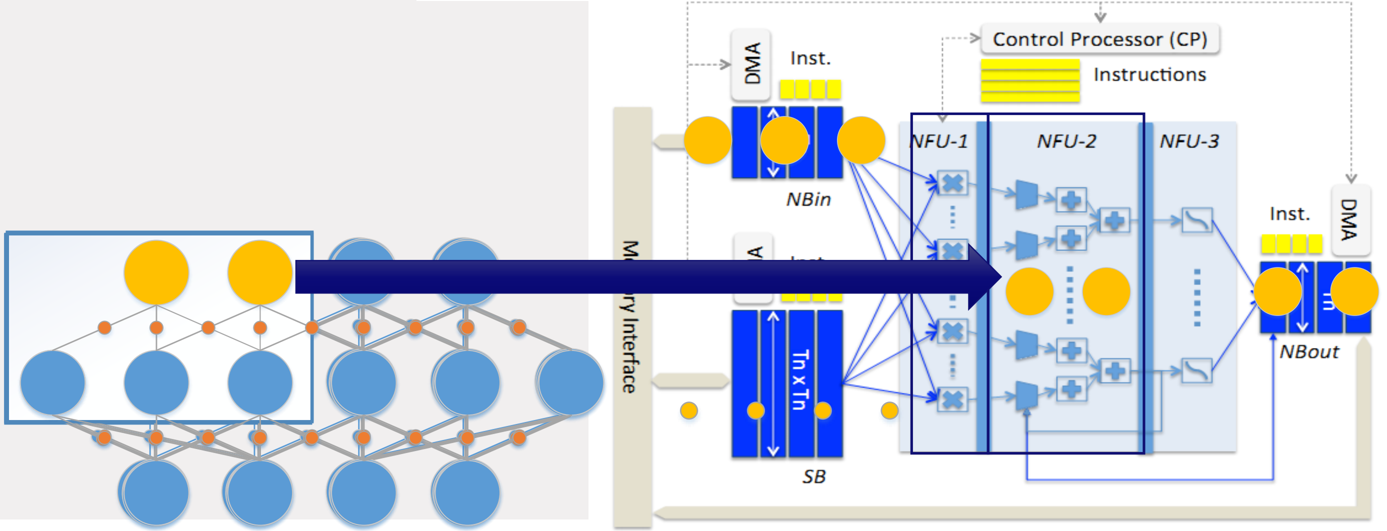
Sum neuron inputs

Buffer partial sums

According to V. Sze et al. "Efficient Processing of Deep Neural Networks" Proceedings of the IEEE 2017

Non Local Reuse Architecture





In-depth analysis of the Google Tensor Processing Unit

- 2013: Google prepares for the success-disaster of new DNN apps
  - Users speaking to phones 3 minutes per day:
    With only CPUS, need 2X-3X times whole datacenter fleet
  - DNNs applicable to a wide range of problems, so hardware accelerator can be reused for speech, vision, language translation, search ranking, etc.

- Goal: Custom hardware to reduce TCO of DNN <u>inference</u> by <u>10X</u> vs. CPUs
  - Must run existing apps developed for CPUs and GPUs
  - Very short development cycle: Project start 2014, running in datacenter 15 months later
  - Architecture invention, Compiler invention, Hardware, Design, Build, Test, Deploy

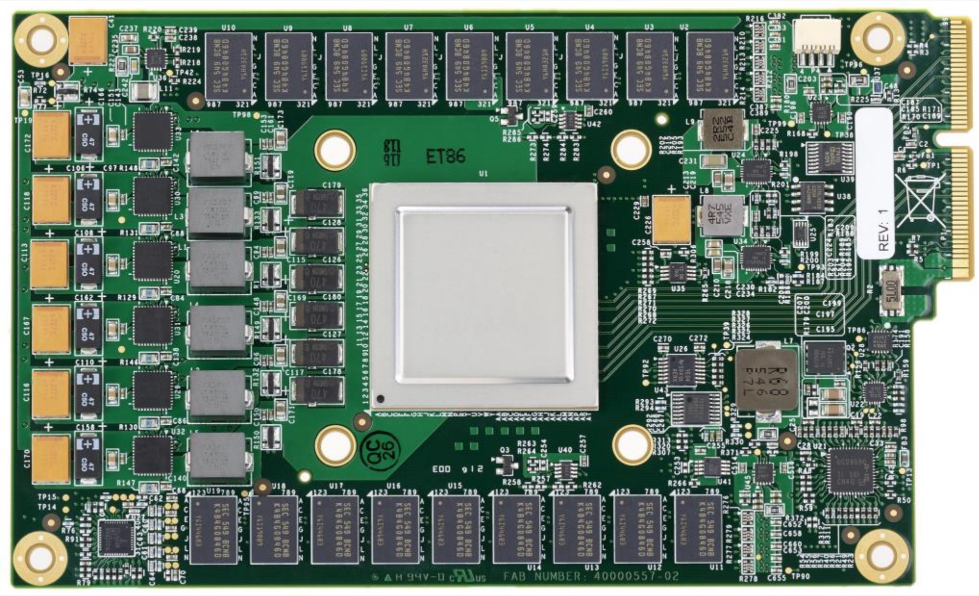
Based upon: In-Data Center Performance Analysis of a tensor Processing Unit
ISCA 2017, Jouppi, Young, Patil, Patterson, et al.

What came out: TPUv1 card & package

- Accelerator card for servers
  - Up to 4 cards / server

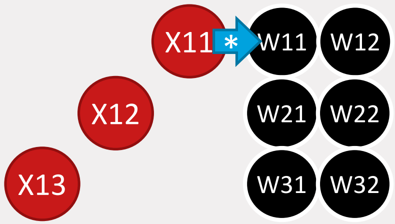
- Coprocessor on the PCIe I/O bus
  - Host CPU sends TPU instructions to execute (to simplify HW design)
  - Unlike GPU that fetches and executes own instructions

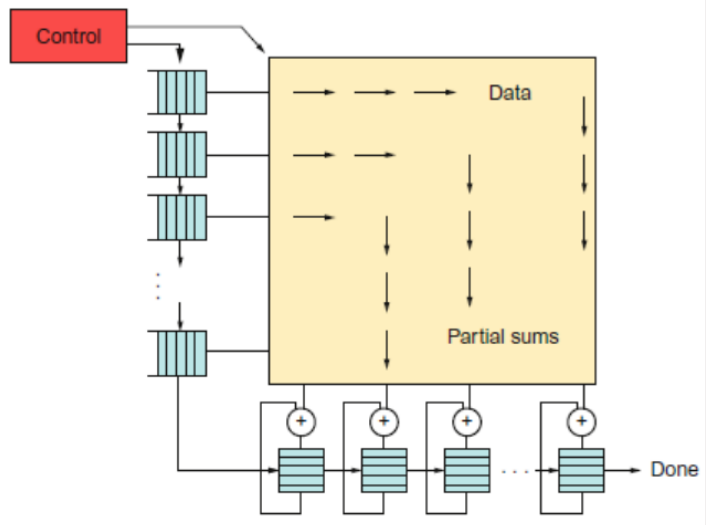
- Large parallel chunks of work done in custom hardware



Implement large 2D systolic matrix multiplication unit

- Systolic Execution to compute data on the fly in buffers
- Pipeline control and data

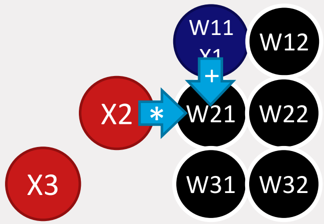




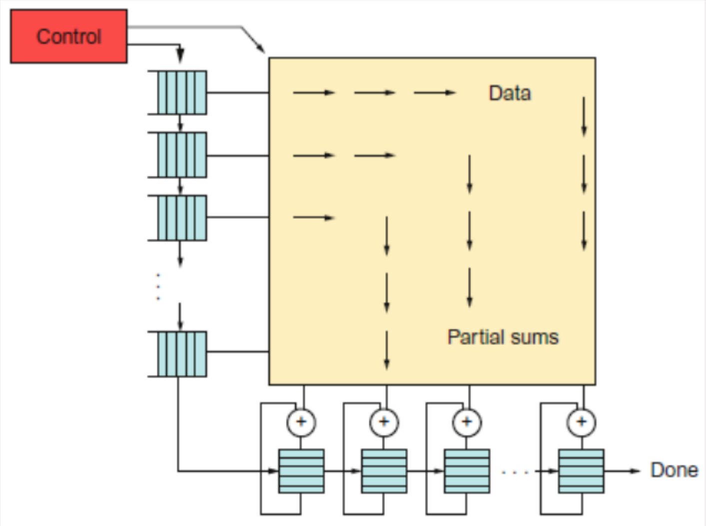
- Relies on data from different directions arriving at regular intervals and being combined

Implement large 2D systolic matrix multiplication unit

- Systolic Execution to compute data on the fly in buffers
- Pipeline control and data

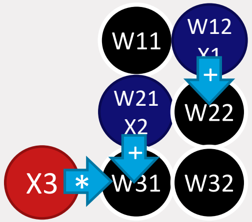


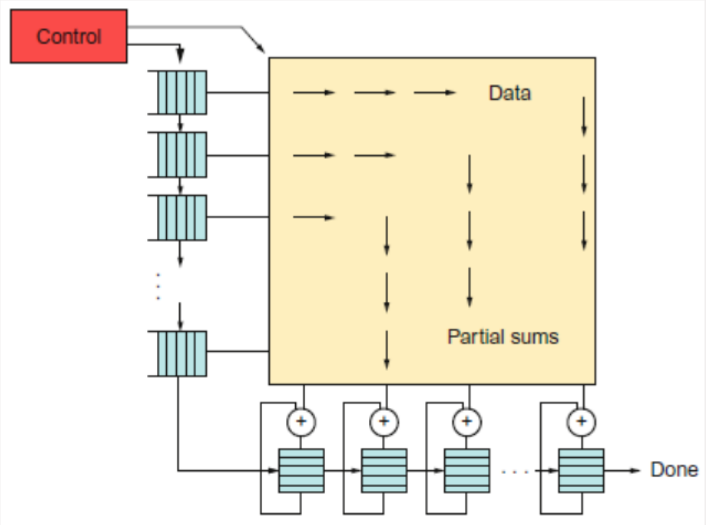
- Relies on data from different directions arriving at regular intervals and being combined



Implement large 2D systolic matrix multiplication unit

- Systolic Execution to compute data on the fly in buffers
- Pipeline control and data

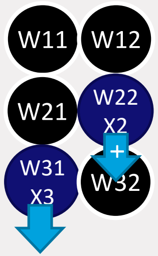




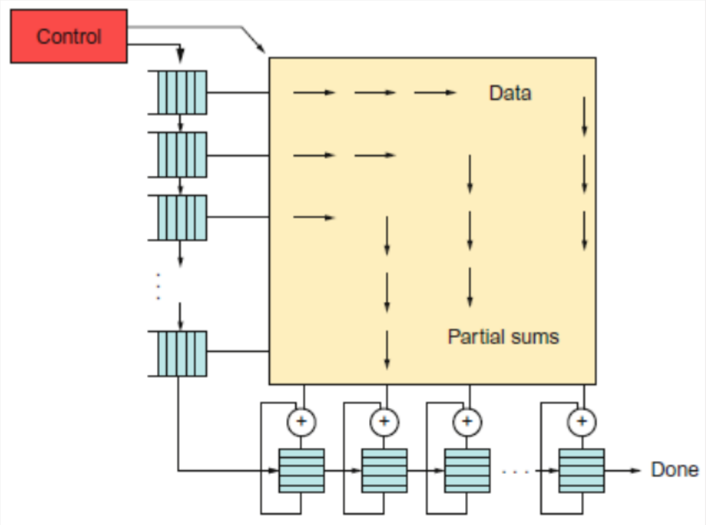
- Relies on data from different directions arriving at regular intervals and being combined

Implement large 2D systolic matrix multiplication unit

- Systolic Execution to compute data on the fly in buffers
- Pipeline control and data



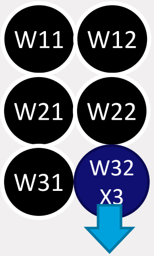
$$y_1=w_{11}x_1+w_{21}x_2+w_{31}x_3$$



- Relies on data from different directions arriving at regular intervals and being combined

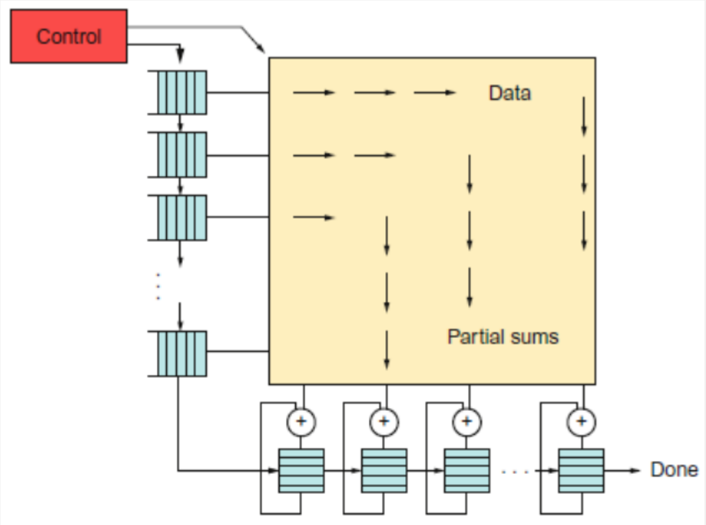
Implement large 2D systolic matrix multiplication unit

- Systolic Execution to compute data on the fly in buffers
- Pipeline control and data



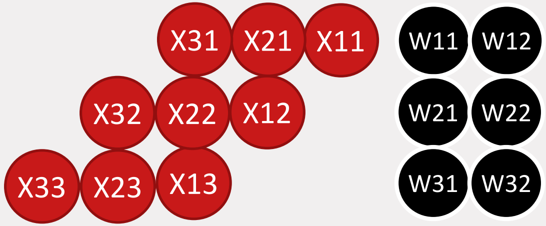
$$y_2 = w_{12}x_1 + w_{22}x_2 + w_{32}x_3$$
$$y_1 = w_{11}x_1 + w_{21}x_2 + w_{31}x_3$$

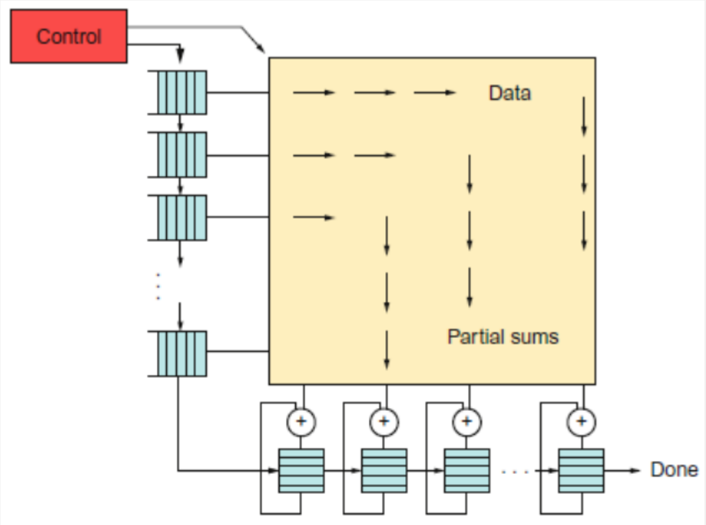
- Relies on data from different directions arriving at regular intervals and being combined



Implement large 2D systolic matrix multiplication unit

- Systolic Execution to compute data on the fly in buffers
- Pipeline control and data



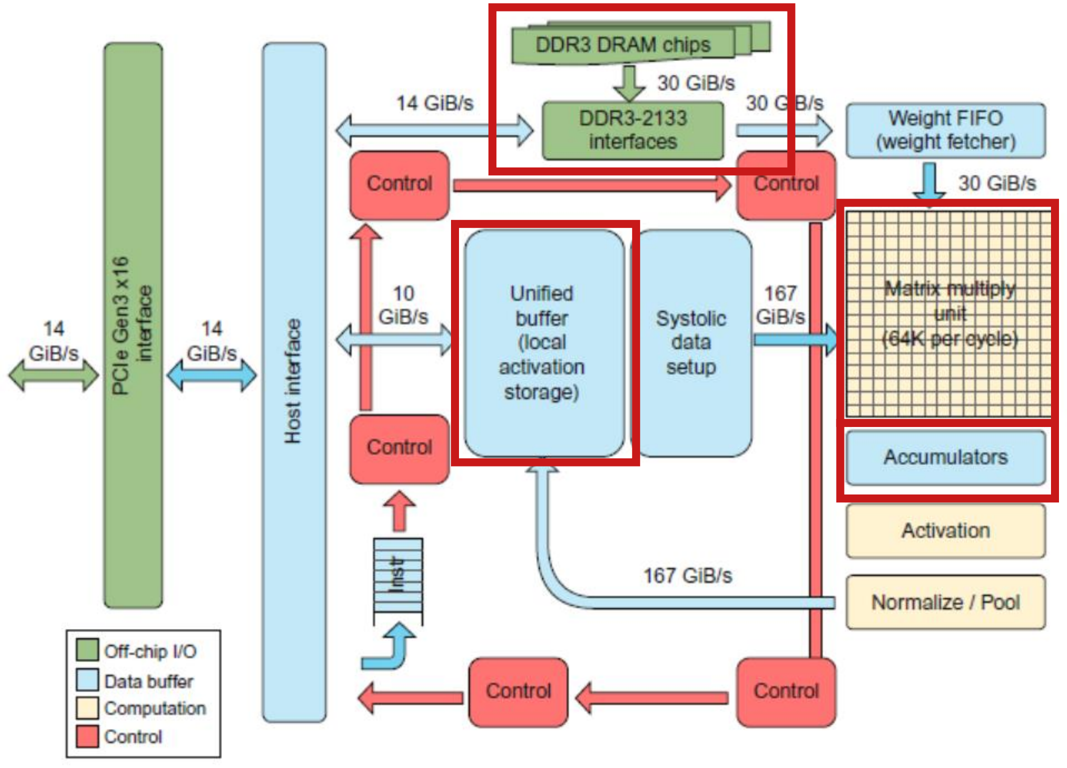


- Relies on data from different directions arriving at regular intervals and being combined
- Matrix Matrix Multiply takes more cycles and has better utilization

High-level Chip Architecture

- Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92 Tera operations/second
  - 65,536 * 2 * 700M

- 4 MiB of on-chip Accumulator mem.
- 24 MiB of on-chip Unified Buffer (activation memory)
- 8 GiB of off-chip weight DRAM mem.
- Two 2133MHz DDR3 DRAM channels

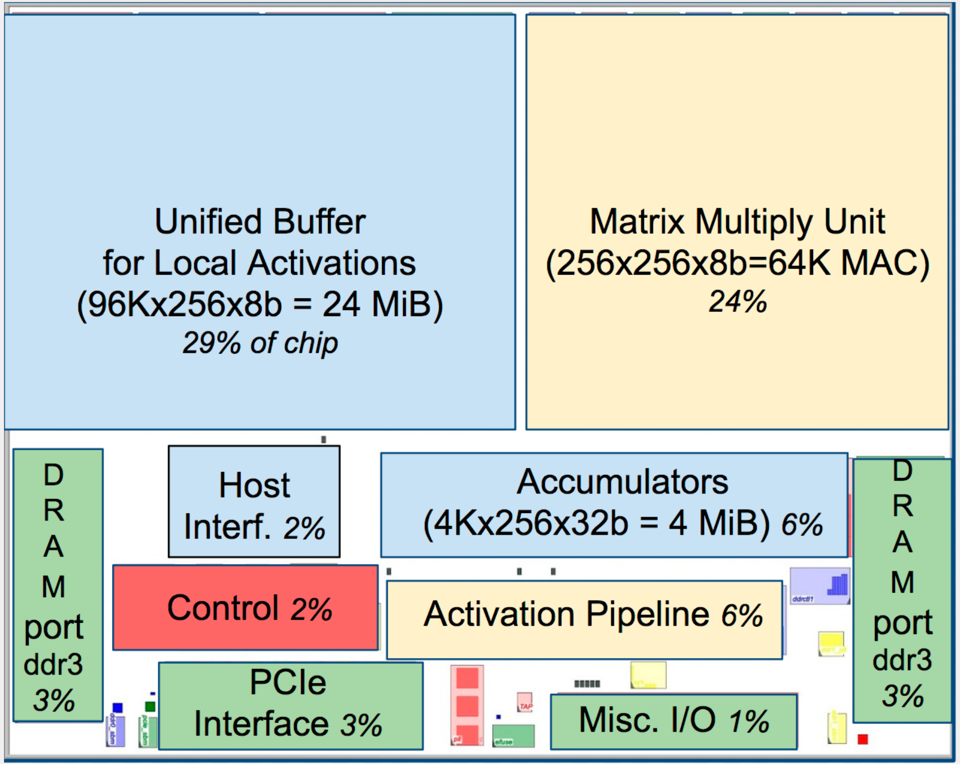


TU/e

TPUv1 Chip area breakdown

- Main focus:
  - On-chip memory 35%
  - Matrix Multiply Unit 24%

- Control is just 2%



TPUv1 from a programmer's view

- 5 main (CISC) instructions
  - Read_Host_Memory
    - Reads memory from the CPU memory into the unified buffer
  - Read_Weights
    - Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit
  - MatrixMatrixMultiply/Convolve
    - Perform a matrix-matrix multiply, a vector-matrix multiply, an element-wise matrix multiply, an element-wise vector multiply, or a convolution from Unified Buffer into the accumulators
    - Takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produce a B*256 output, taking B pipelined cycles to complete
  - Activate (ReLU, Sigmoid, Maxpool, LRN, …)
    - Computes activation function
  - Write_Host_Memory
    - Writes data from unified buffer into host memory

TPUv1 from a programmer’s view

- 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

Relative TPU Performance: 3 Contemporary Chips 2015

Processor	*mm^2*	*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
TPUv1	<331	700	75	28	34	91.8	--

TPUv1 is less than half die size of the Intel Haswell processor

TU/e

Relative Performance of TPUv1 server

Processor	*Chips / Server*	*DRAM*	*TDP Watts*	*Idle Watts*	*Observed Busy Watts in Datacenter*
CPU: Haswell (18 cores)	2	256GB	504	159	455
NVIDIA K80 (2 die per card; 4 cards per server)	8	256GB (host) + 12GB x 8	1838	357	991
TPUv1 (1Core)	4	256GB (host) + 8GB x 4	861	290	384

Performance: Inference Datacenter Workload (95%)

Name	LOC	Layers					Nonlinear function	Weights	TPUv1 Ops / Weight Byte	TPUv1 Batch Size	% Deployed
		FC	Conv	Vector	Pool	Total					
MLP0	0.1k	5				5	ReLU	20M	200	200	61%
MLP1	1k	4				4	ReLU	5M	168	168	
LSTM0	1k	24		34		58	sigmoid, tanh	52M	64	64	29%
LSTM1	1.5k	37		19		56	sigmoid, tanh	34M	96	96	
CNN0	1k		16			16	ReLU	8M	2888	8	5%
CNN1	1k	4	72		13	89	ReLU	100M	1750	32	

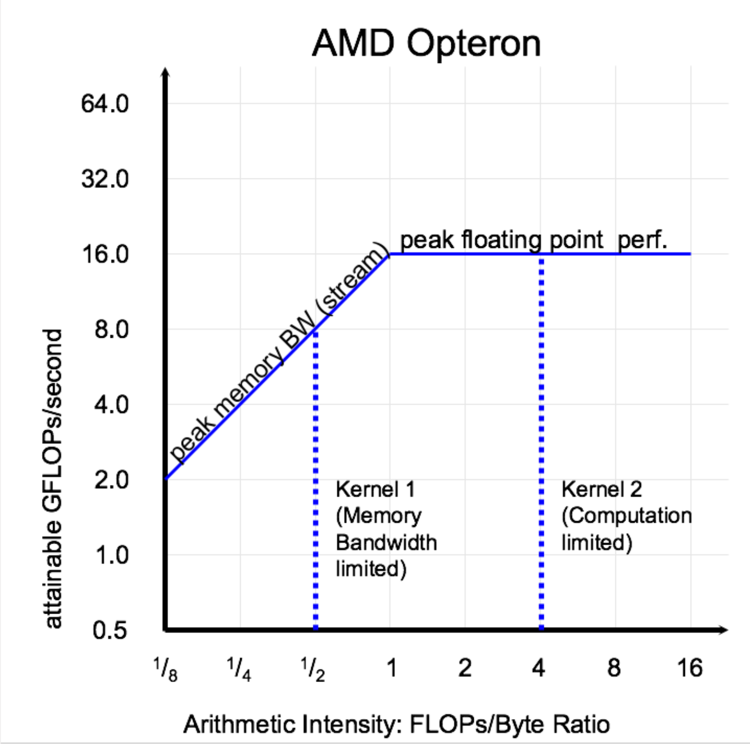
Roofline Visual Performance Model

2 Limits to performance

1. Peak Computation
2. Peak Memory Bandwidth
   (for apps with large data that don't fit in chache)

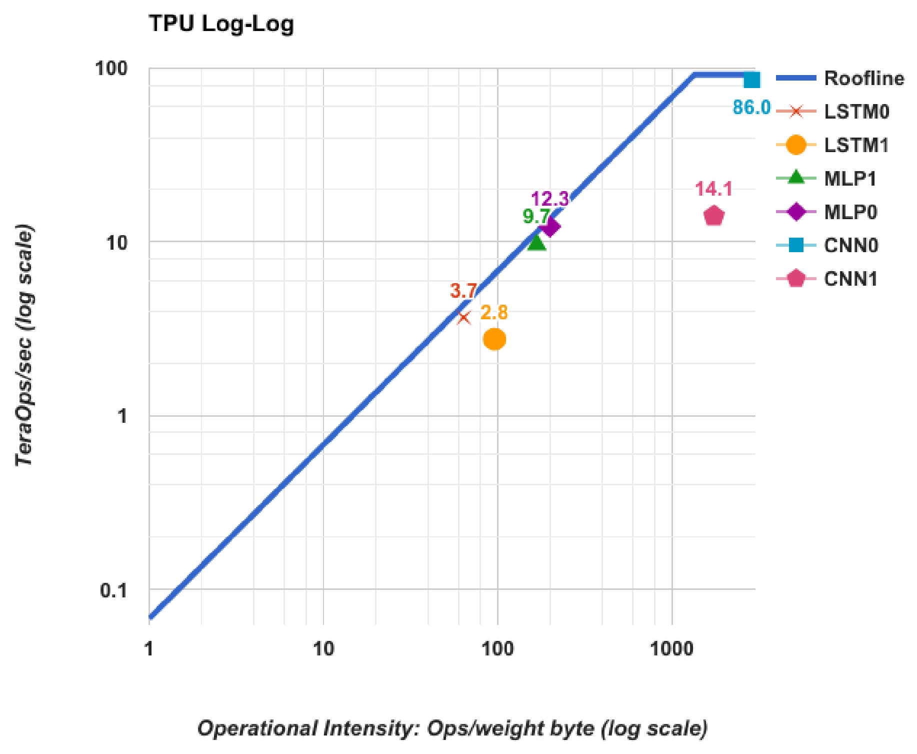
Arithetic Intensity (FLOP/Byte or reuse) determines which limit

Weight-reuse = Arithmetic Intensity for DNN roofline

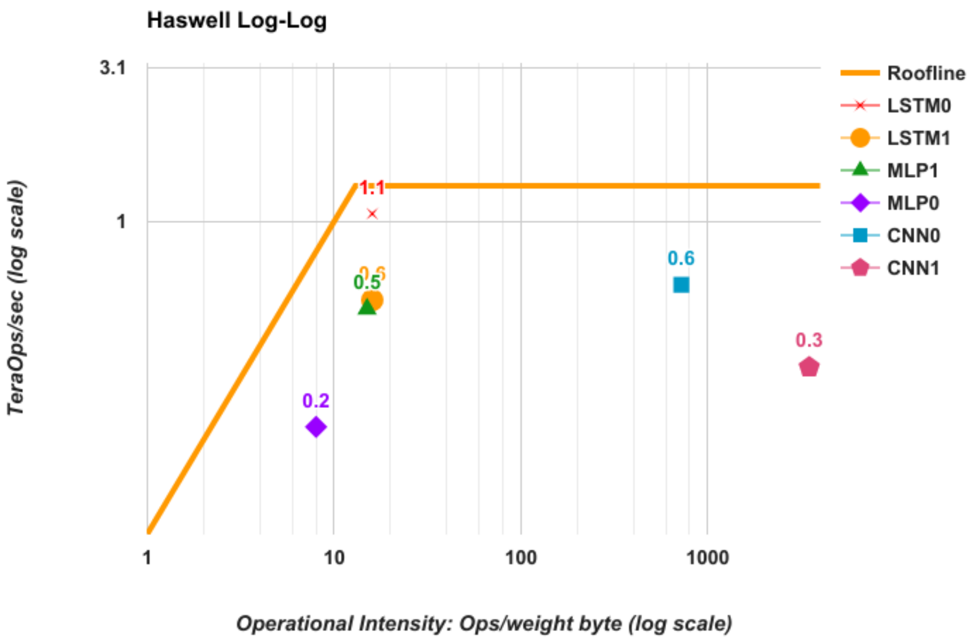


Samuel Williams, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures."*Communications of the ACM* 52.4 (2009): 65-76.

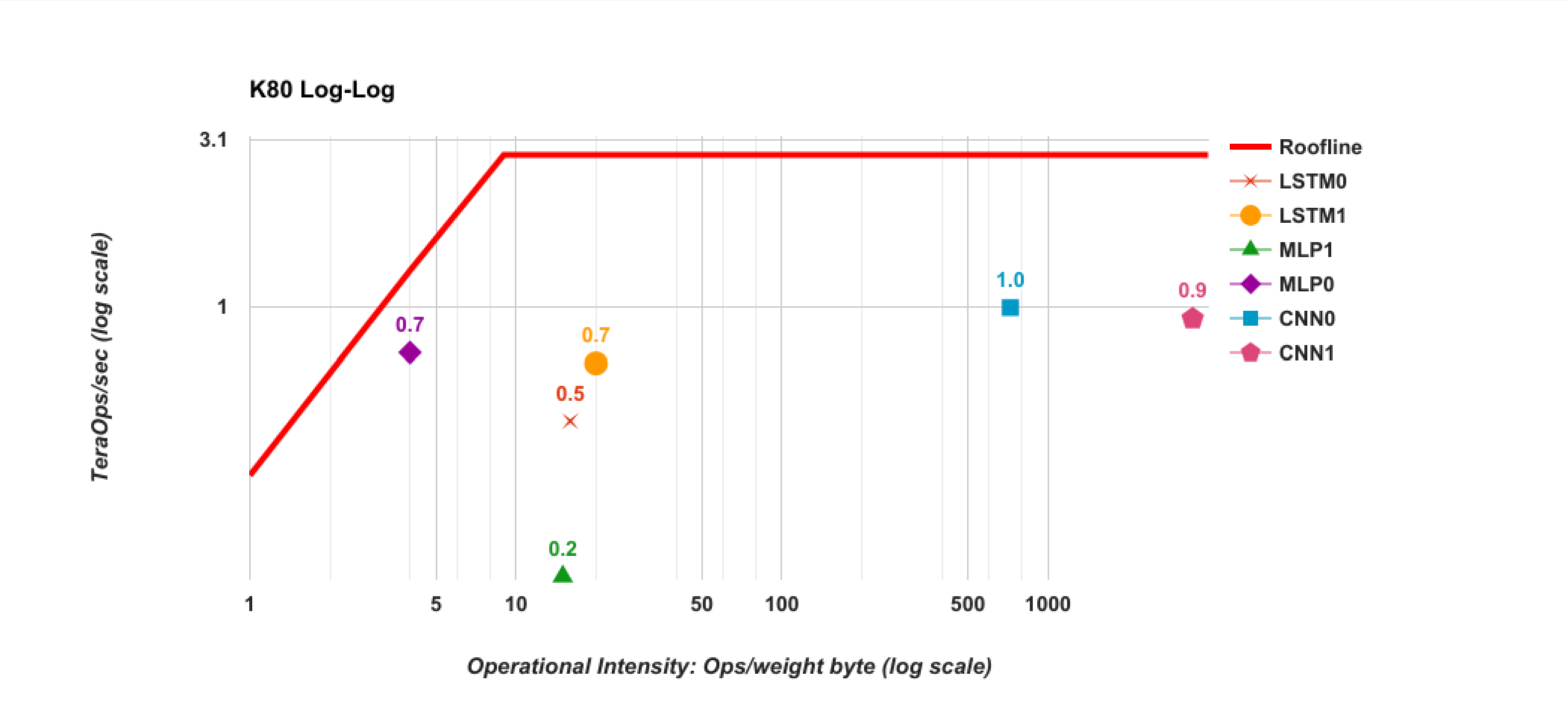
TPUv1 Die Roofline



Haswell (CPU) Die Roofline



K80 (GPU) Die Roofline

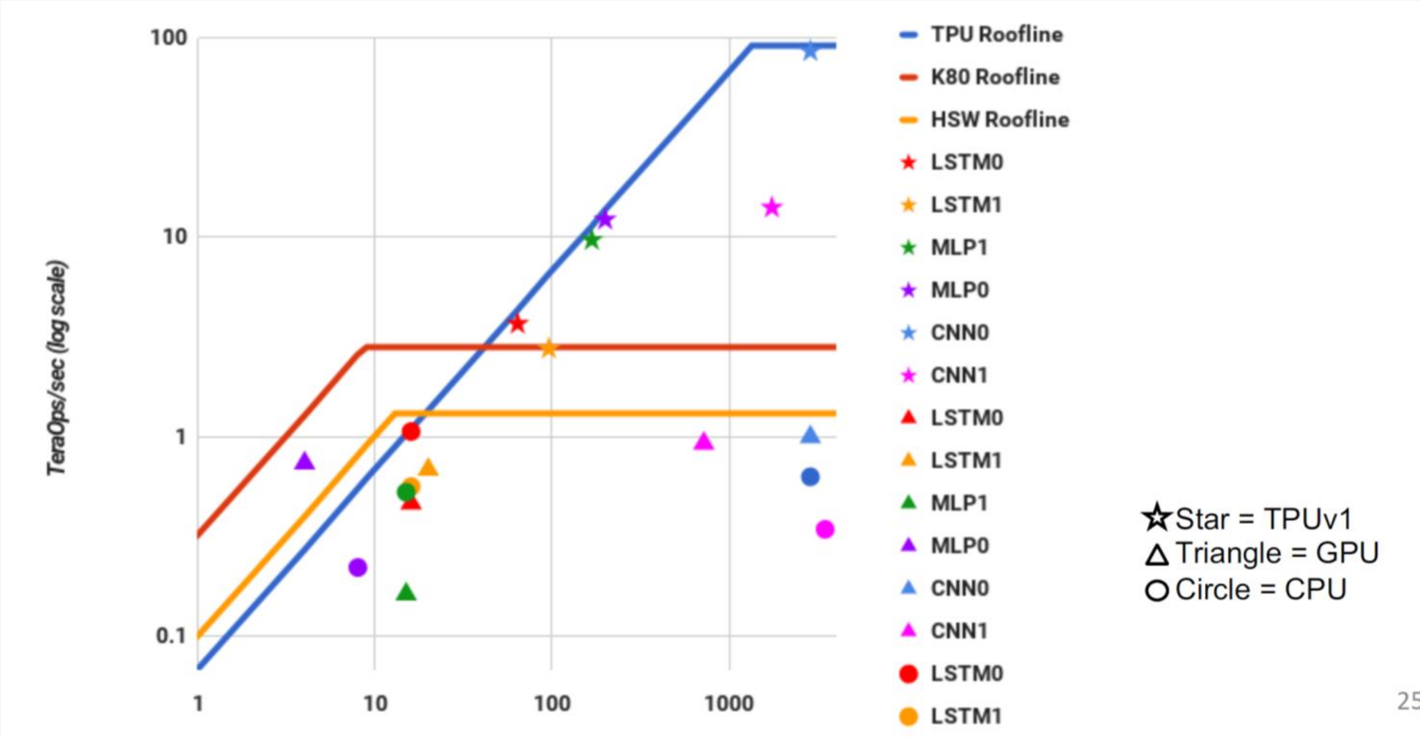


Many benchmark far below roofline, e.g. MLP0

Increase Batch Size = More Weight Reuse

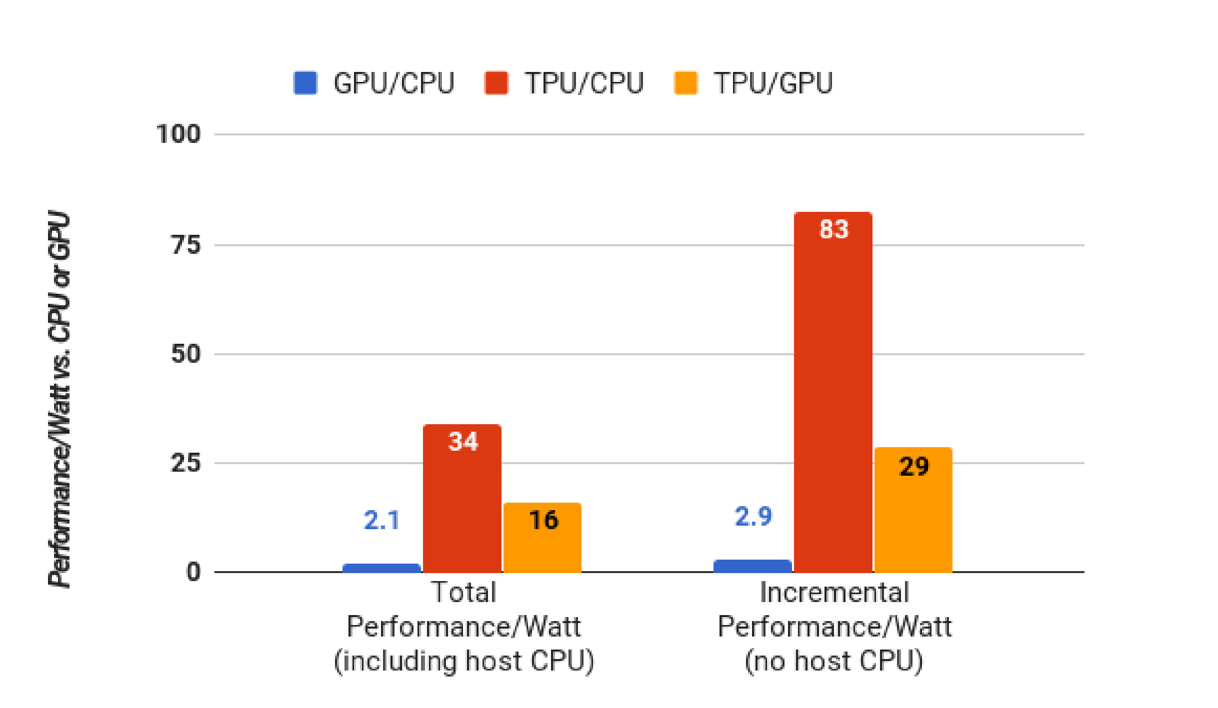
Type	*Batch*	*99th% Response*	*Inf/s (IPS)*	*% Max IPS*
CPU	16	7.2 ms	5,482	42%
CPU	64	21.3 ms	13,194	100%
GPU	16	6.7 ms	13,461	37%
GPU	64	8.3 ms	36,465	100%
TPUv1	200	7.0 ms	225,000	80%
TPUv1	250	10.0 ms	280,000	100%

Combined log rooflines CPU, GPU, TPUv1



TU/e

Perf/Watt TPUv1 vs CPU & GPU



TPUv1 success due to…

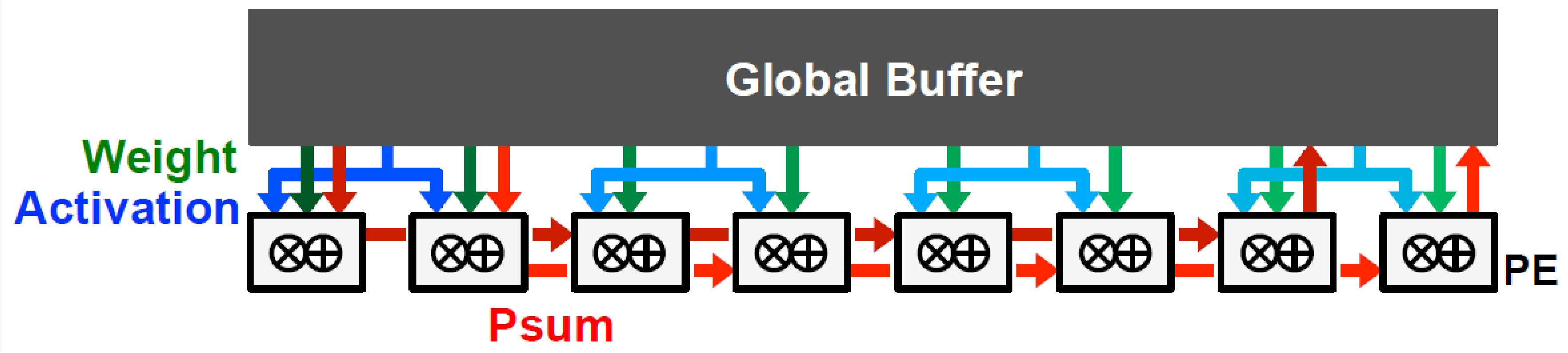
- Large Matrix Multiply Unit
- Software controlled on-chip memory
- Omission of GPU features small, low power die
- Use of 8-bit integers in quantized apps
- Apps in Tensorflow are easy to port at speed

- 15-month design & live introduction

- 30x faster than Haswell CPU, K80 GPU (inference)
- <0.5 die size, 0.5 Watts

Previous architectures are "Non Local Reuse Architectures"

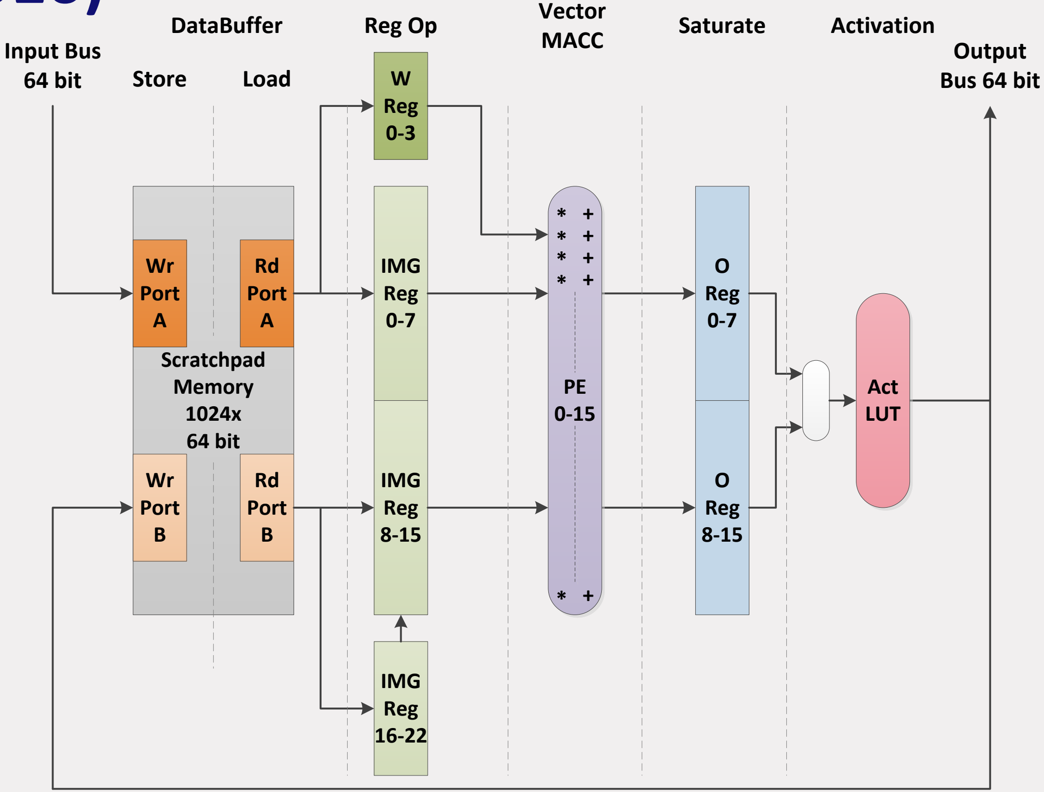
- Use of large global buffers as shared storage
  - Reduce DRAM accesses
- Multicast activations, single cast weights, accumulate partial sums spatially
- Often matrix multiplication based



Neuro Vector Engine (2016)

Operate on vectors

- Dual port vector memory
- Vector shuffle registers
- Vector MACC array
- Vector Activation



M. Peemen et al. "The neuro vector engine: Flexibility to improve convolutional net efficiency" DATE2016

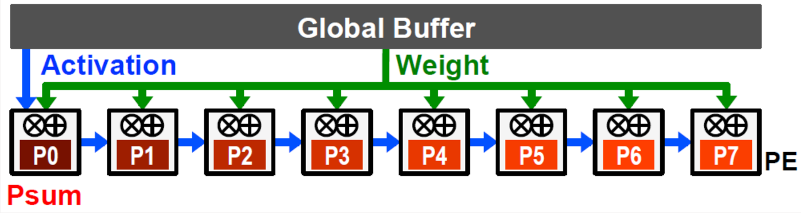
Output Stationary

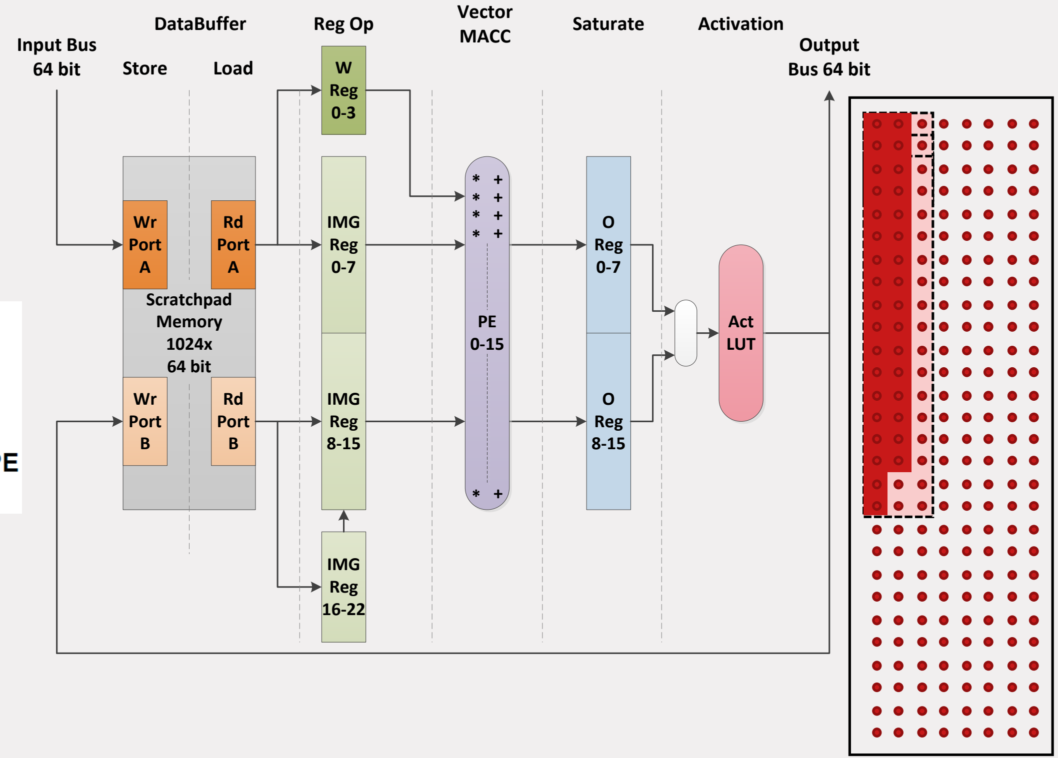
According to V. Sze et al. "Efficient Processing of Deep Neural Networks" Proceedings of the IEEE 2017

Minimize partial sum R/W energy consumption

Maximize local accumulation

Broadcast filter weights and



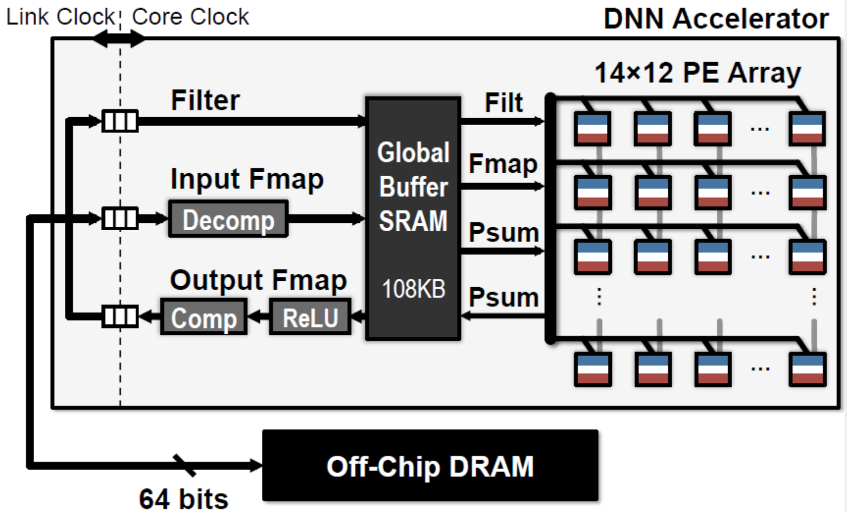


Energy-efficient dataflow Eyeriss

Row Stationary

- Maximize reuse and accumulation at register file
- Optimize for overall energy efficiency instead on one data type
- Filter row in RF
- Activation sliding window in RF
- Partial sum accumulation in RF

Accumulator can move in out memory and between PE



Y-H. Chen et al. “Eyeriss: an Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks”, ISSCC 2016

GPU: NVIDIA Volta V100 (GTC May 2017)

- up to 80 cores, 5120 PEs (FP32)
- 20 MB register space
- 815 mm^2
- 21.1 Btransistors
- 12 nm
- 300 W
- peak: 120 TFlops/s
- (FP16) => 2.5 pJ/op

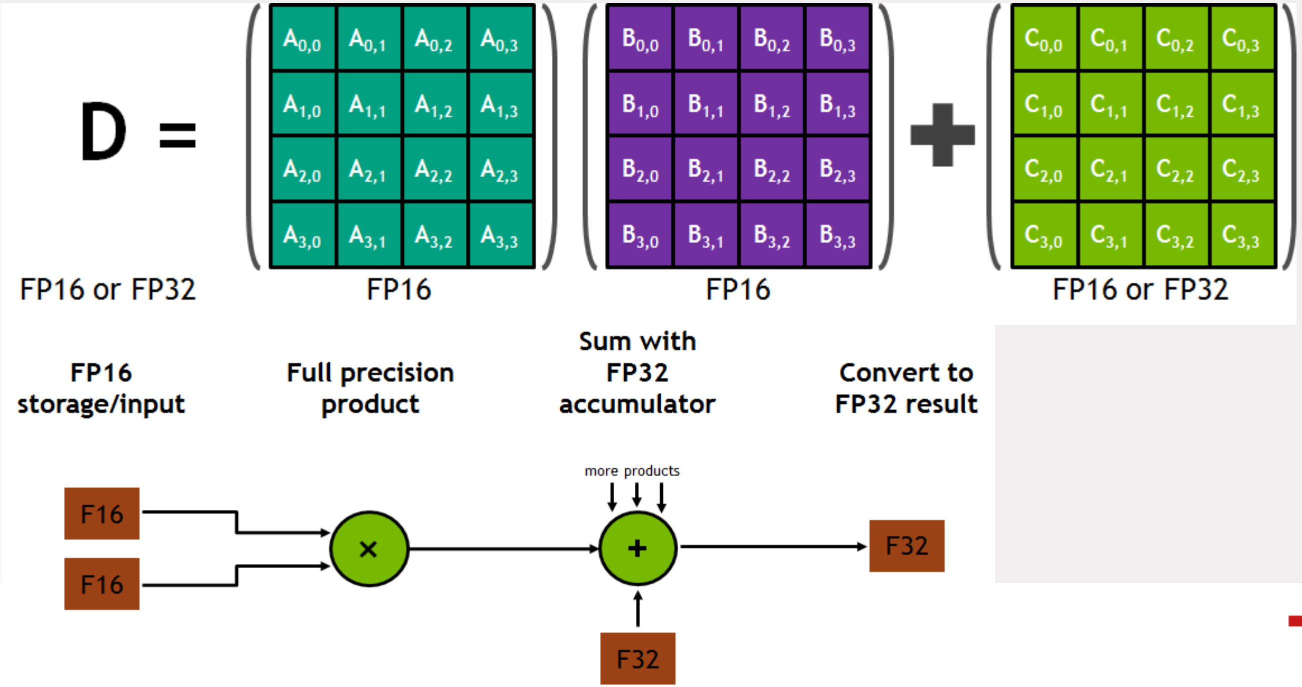
1 SM core

Units:

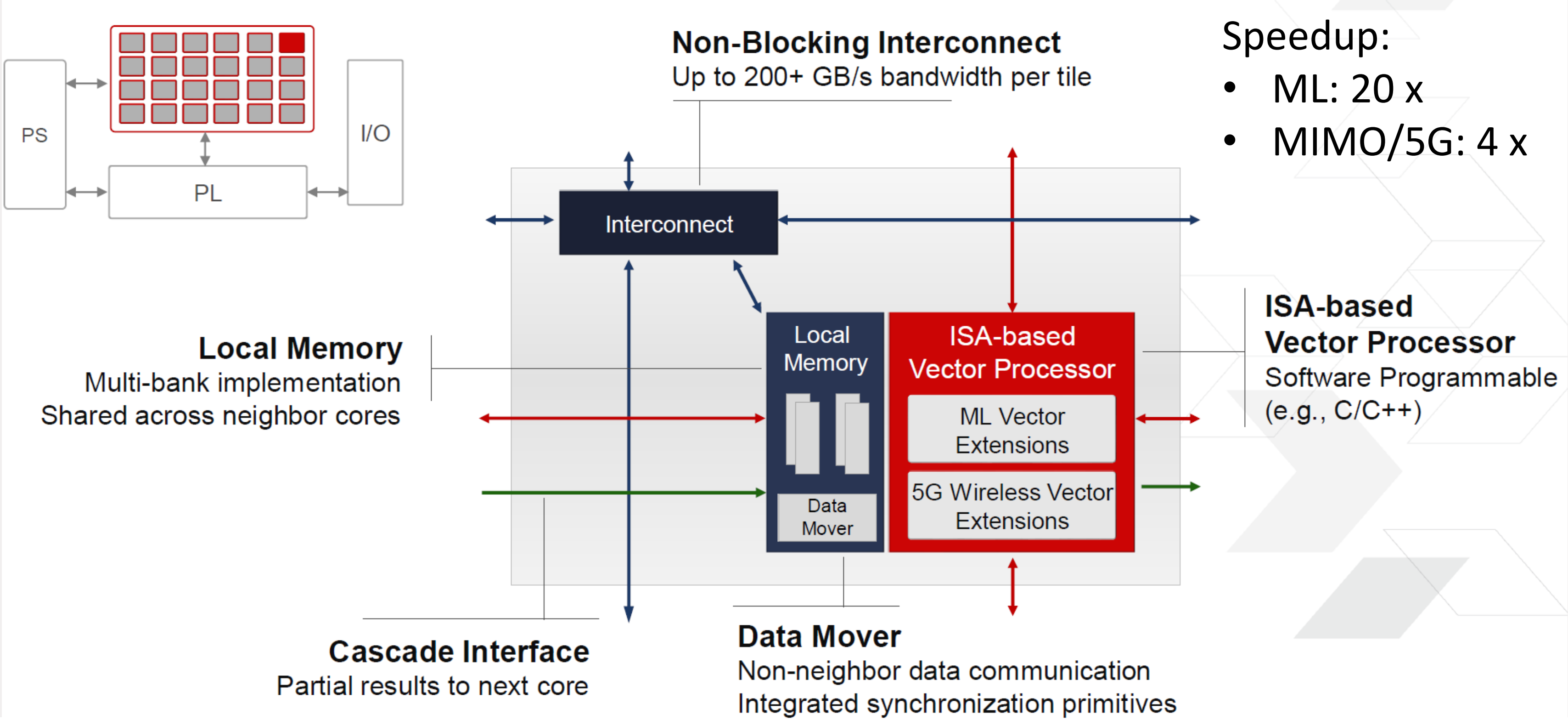
- 8 tensor cores/SM
- 64 Int units
- 64 FP32
- 32 FP64
- 32 Ld/St
- 4 SFUs
- 128 LB L1 Data $
- 4 warp schedulers

Tensor core operation

- D = AxB + C, all 4x4 matrices
- 64 floating point MAC operations per clock



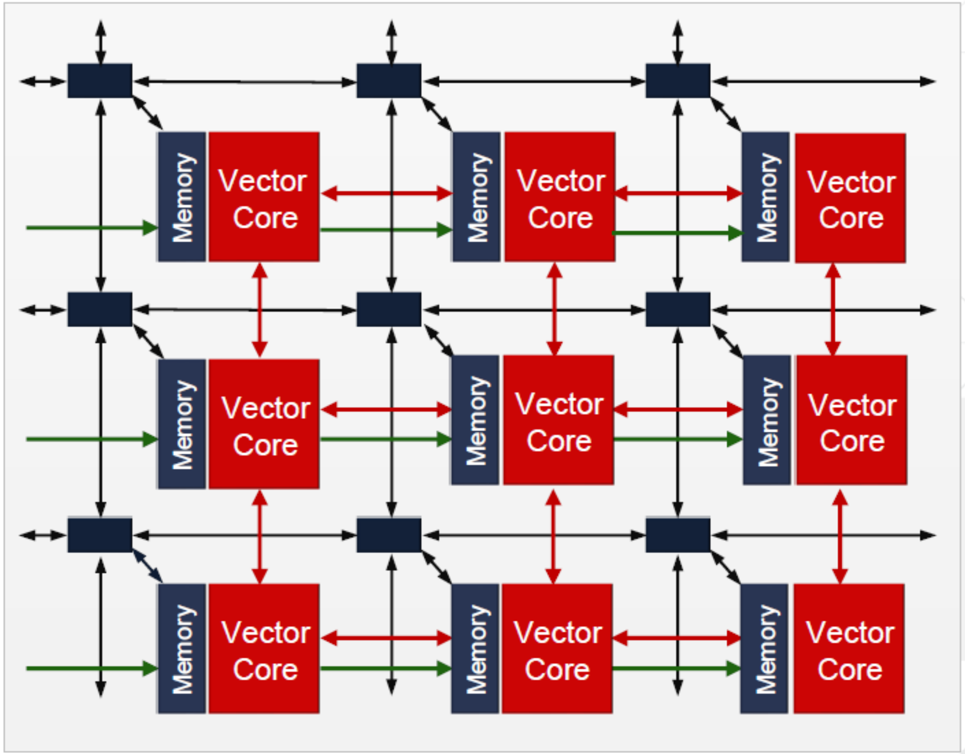
HotChips 2018: Xilinx adds Vector blocks



Speedup:

- ML: 20 x
- MIMO/5G: 4 x

Xilinx Versal AI cores: scalable VLIW vector units



Compute efficiency for neural network acceleration

Well known paradigm that can be made quantitative

- Energy Efficiency: Gops/Watt
- Sometimes expressed as: pJ/op
- Area Efficiency: Gops/mm2

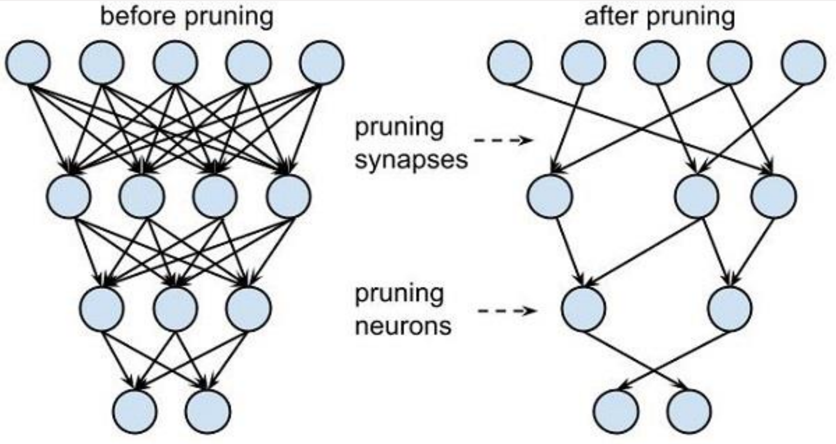
Accelerator	mW	V	Tech.	MHz	Gops	Gops/W	On-Chip Mem.	Control	mm^2	$Gops/mm^2$
NeuFlow [110]	600	1.0	45 SOI	400	294	490	75 kB	Dataflow	12.5	23.5
Origami [15]	93	0.8	umc 65	189	55	803	43 kB	Config	1.31	42
NVE [109]	54	-	TSMC 40	1000	30	559	20 kB	VLIW	0.26	115
DianNao [20]	485	-	TSMC 65	980	452	931	44 kB	FSM	3.02	149.7
ShiDianNao [46]	320	-	TSMC 65	1000	128*	400	288 kB	FSM	4.86	26.3
Desoli et al. [43]	51	0.58	28 FD-SOI	200	78	1277	6 MB	Config	34	2.29
Shin et al. [132]	35	0.77	65 1P8M	50	73	2100**	290 kB	FSM	16	4.5
TPU [77]	40,000	-	28 nm	700	46,000	1150	28 MB	CISC	<331	>139

* Only gives theoretical peak performance, as opposed to actual measurements on a real network.
** Q-table lookup of precomputed multiplication results

Network Efficiency is Important

Less compute and storage without loss of accuracy

- This sounds too good to be true?



Network	Original Size	Compressed Size	Compression Ratio	Original Accuracy	Compressed Accuracy
Lenet-5	1720KB	44KB	39x	99.20%	99.26%
AlexNet	240MB	6.9MB	35x	80.27%	80.30%
VGGNet	550MB	11.3MB	49x	88.68%	89.09%
GoogleNet	28MB	2.8MB	10x	88.90%	88.92%

S.Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”

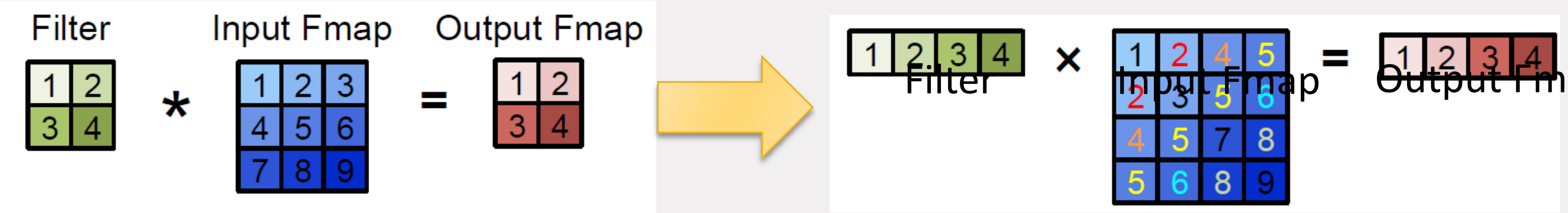
Examples of inefficiency

TPU (256x256 systolic array)

Effective utilization for small (efficient) network layers?

- E.g. 16 kernels of 3x3
- Where to obtain the parallelism?

Matrix conversion overhead?



N.P. Jouppi et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit" ISCA 2017
V. Sze et al. "Efficient Processing of Deep Neural Networks", Proceedings of the IEEE 2017

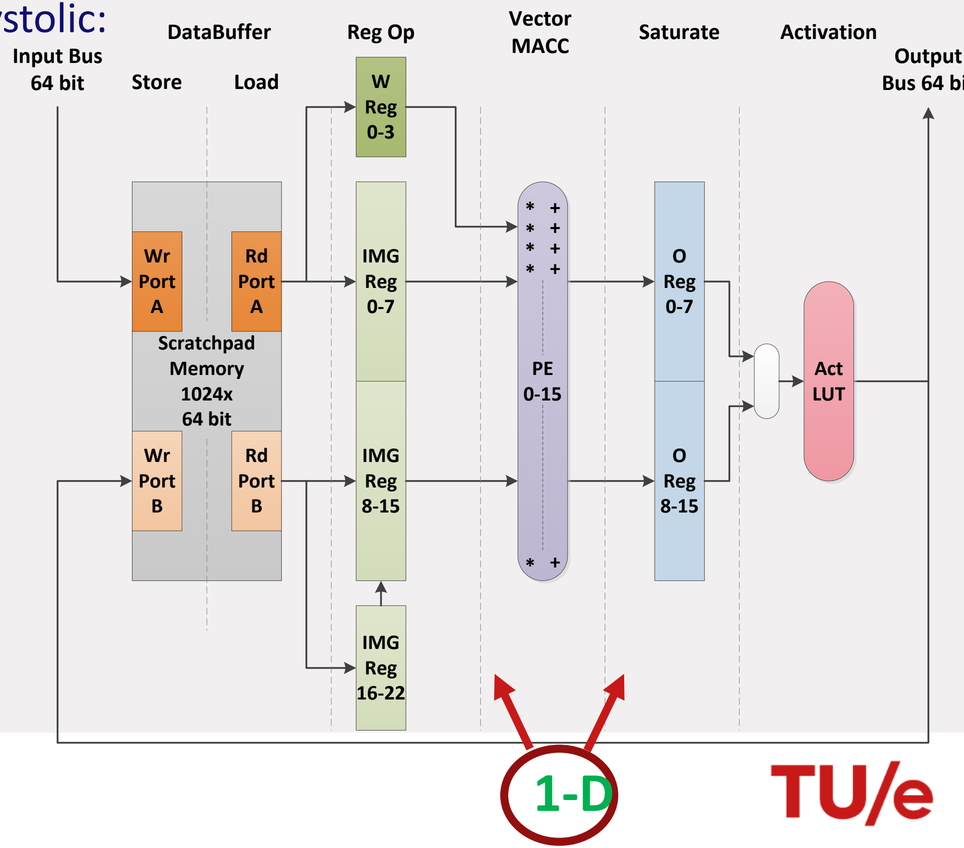
Improving flexibility and efficiency

Processing 1D vectors more flexible than 2D systolic:

More programming flexibility

Operations that perform vector permute and merge

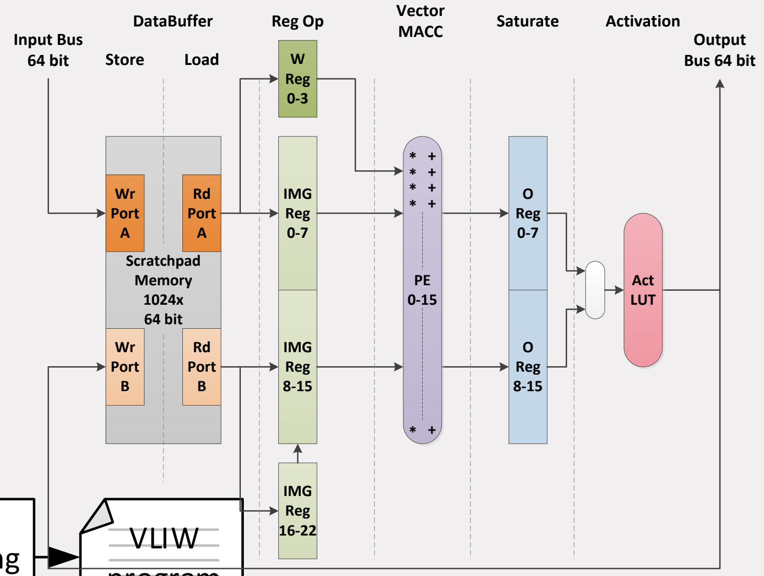
Register files like Intel AVX



TU/e

NVE Compiler needed

Abstract from the hardware



.xml ConvNet → Task Graph Generation → Block Scheduling → Schedule Bundling → Assembling → VLIW program

M.Peemen e.a., VLIW Code Generation for Convolutional Network Accelerator SCOPS 2015

10 Cycles

Write Local	Read Local	Reg Operation	EX VMAC	WB Sgm
	Ld a 0, [b0 w0-1]	Set i3, Shift i0 i1, Shift W	MAC,w6, c2 i0	
	Ld ab 4 5, [c0,i0-15]	Shift i0 i1, Shift W	MAC,w7, c2 i1	
Wr[c3,i0-i7], 4	Ld b 6, [c0,i16-17]	Set W0	MAC,w8, c2 i2	
	Ld a 1, [w2-4]	Set i0 i1, Shift W	Set, b0	
	Ld ab 7 8, [c1,i0-15]	Set i3, Shift i0 i1, Shift W	MAC,w0, c0 i0	
Wr[c3,i8-i15], 5	Ld b 9, [c1,i16-17]	Shift i0 i1, Set W1	MAC,w1, c0 i1	
	Ld a 2, [w5-8]	Set i0 i1, Shift W	MAC,w2, c0 i2	Sigm0, Wrback
	Ld ab 10 11, [c2,i0-15]	Set i3, Shift i0 i1, Shift W	MAC,w3, c1 i0	Sigm1, Wrback
Wr[c3,i16-i19 - - - -], 6	Ld b 12, [c2,i16-17]	Shift i0 i1, Set W2	MAC,w4, c1 i1	
	NOP	Set i0 i1, Shift W	MAC,w5, c1 i2	

TU/e

AivoTTA architecture

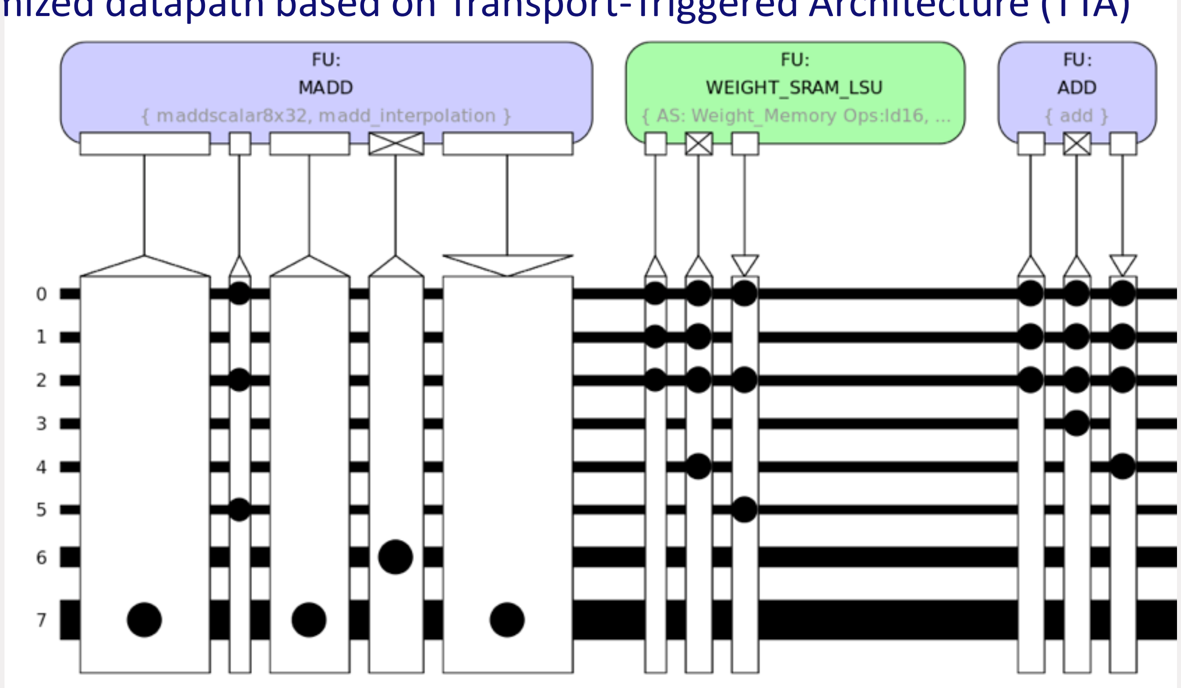
Convolutional Network optimized datapath based on Transport-Triggered Architecture (TTA) Template

Weight datapath

bus 0-2: general-purpose scalar 32-bit

bus 3-5: scalar weights, 32-bit

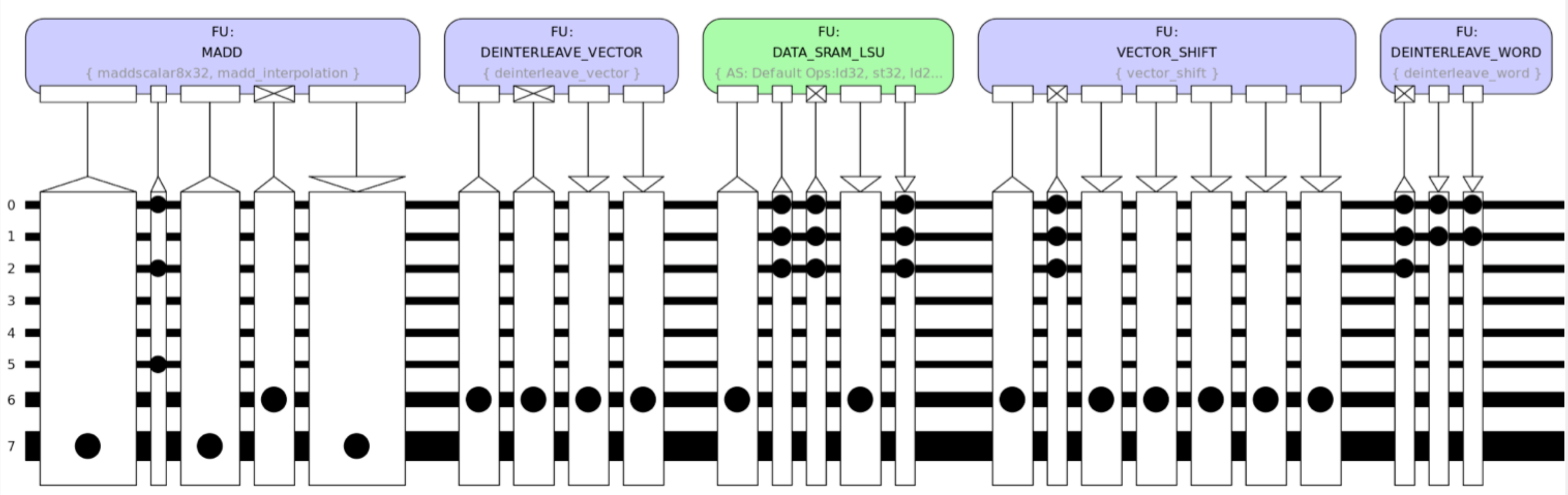
bus 6: input vectors, 256-bit
bus 7: accumulator vectors, 1024-bit



J.IJzerman et al., "AivoTTA: An Energy Efficient Programmable Accelerator for CNN-Based Object Recognition" SAMOS 2018 (research from Tampere Univ of Technology and TU/e)

AivoTTA (several vector units)

C and OpenCL programmable



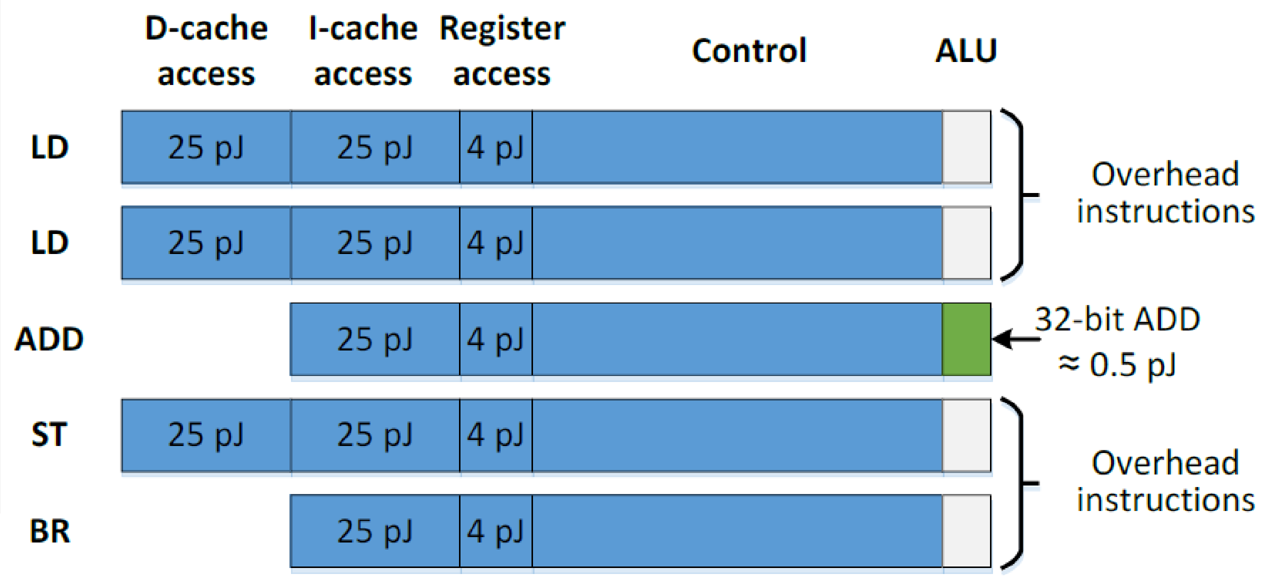
Flexibility has its price

Where does the energy go?

RISC @ 45 nm, 0.9 V

ADD op. 0.5 pJ out of 70 pJ for the ADD instruction

Overall efficiency:
1 / 850 = 0.12 %

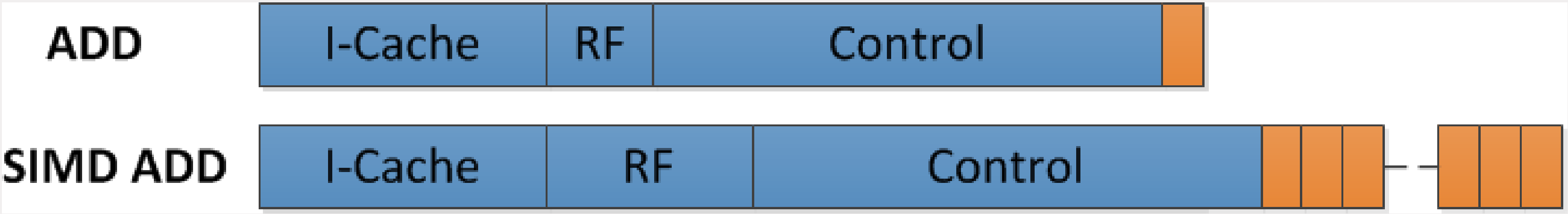


Reduce the price of flexibility: SIMD

Perform more effective work per instruction

Very programmable

1/140 vs 16/180 → 10 times efficiency improvement

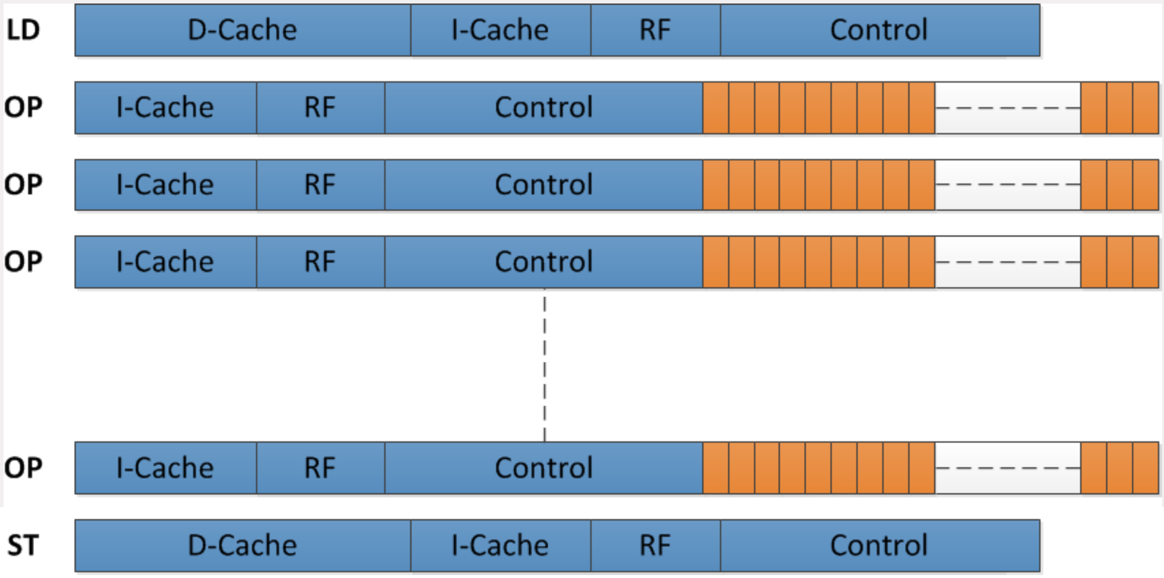


Further improve efficiency

Maximize effective operations

Minimize load stores

E.g. 1/5 effective vs 9/11 → additional 4 times efficiency increase



~40 times efficiency improvement

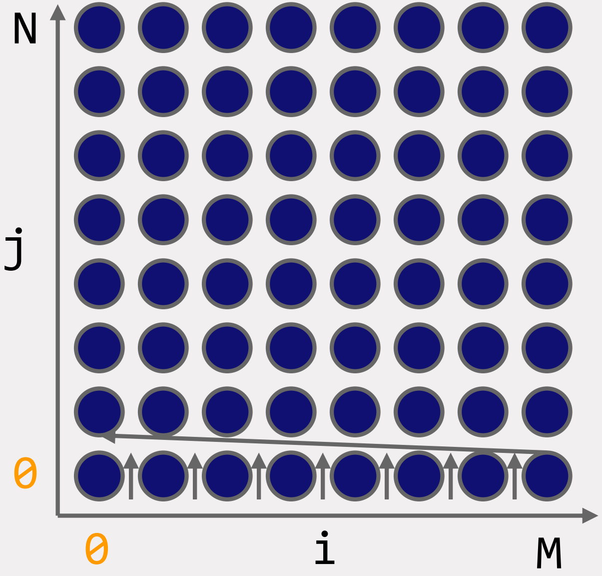
The importance of Loop Transformations

Reduce external memory accesses

- Tiling
- Loop interchange

Loop Tiling

Original loop

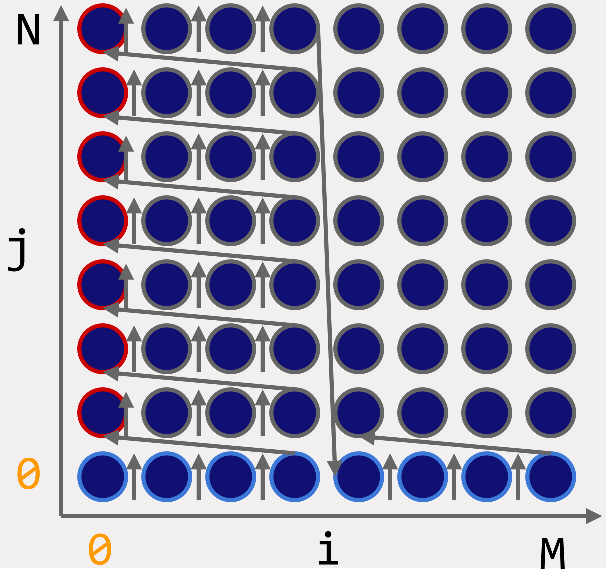



```
for(j=0; j<N; j++)
  for(i=0; i<M; i++)
    B[j][i]+=A[i]+C[j];
```

Loop, after Tiling

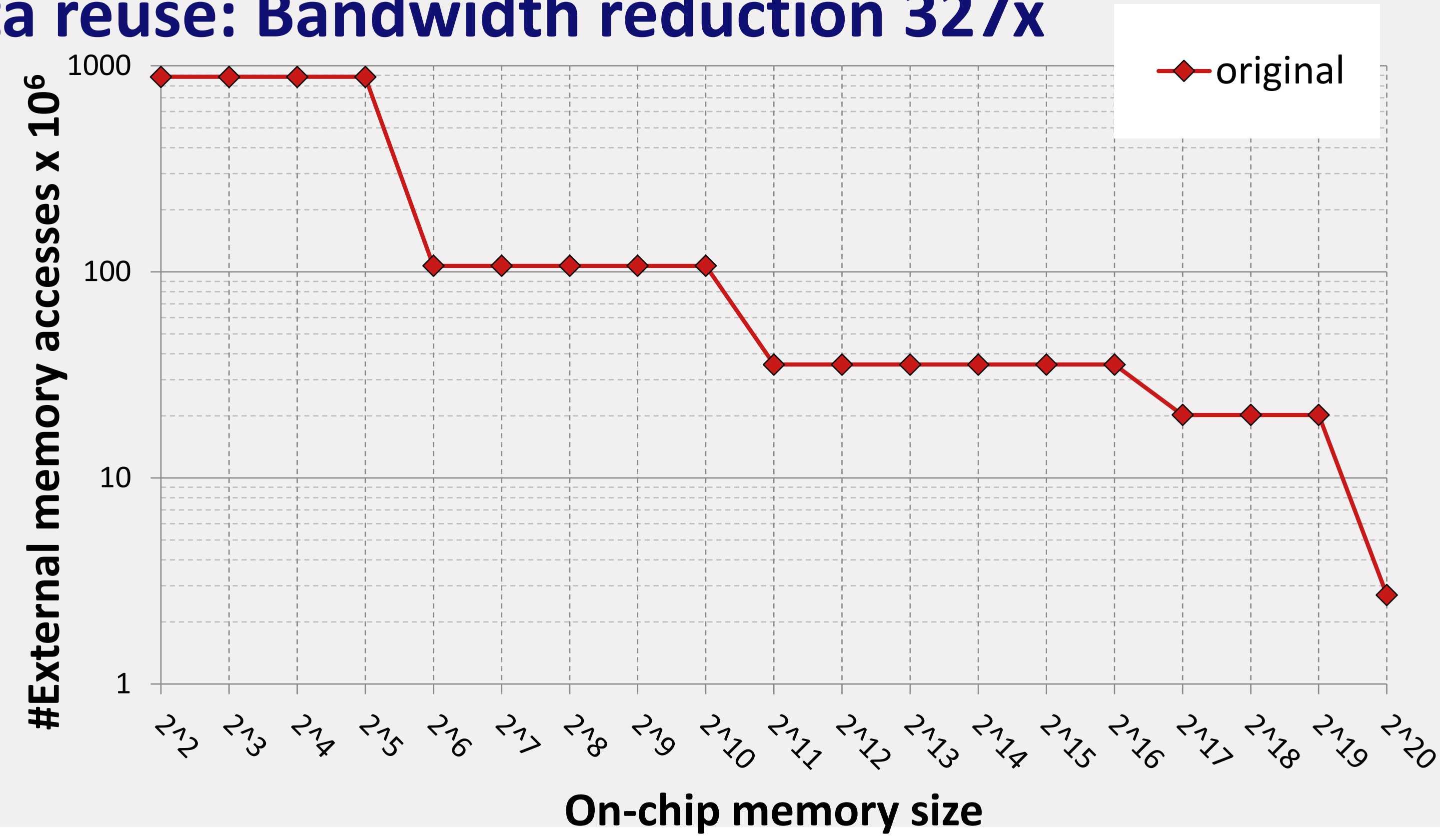
Tile i loop into (i, ii)

Interchange j and ii - loops



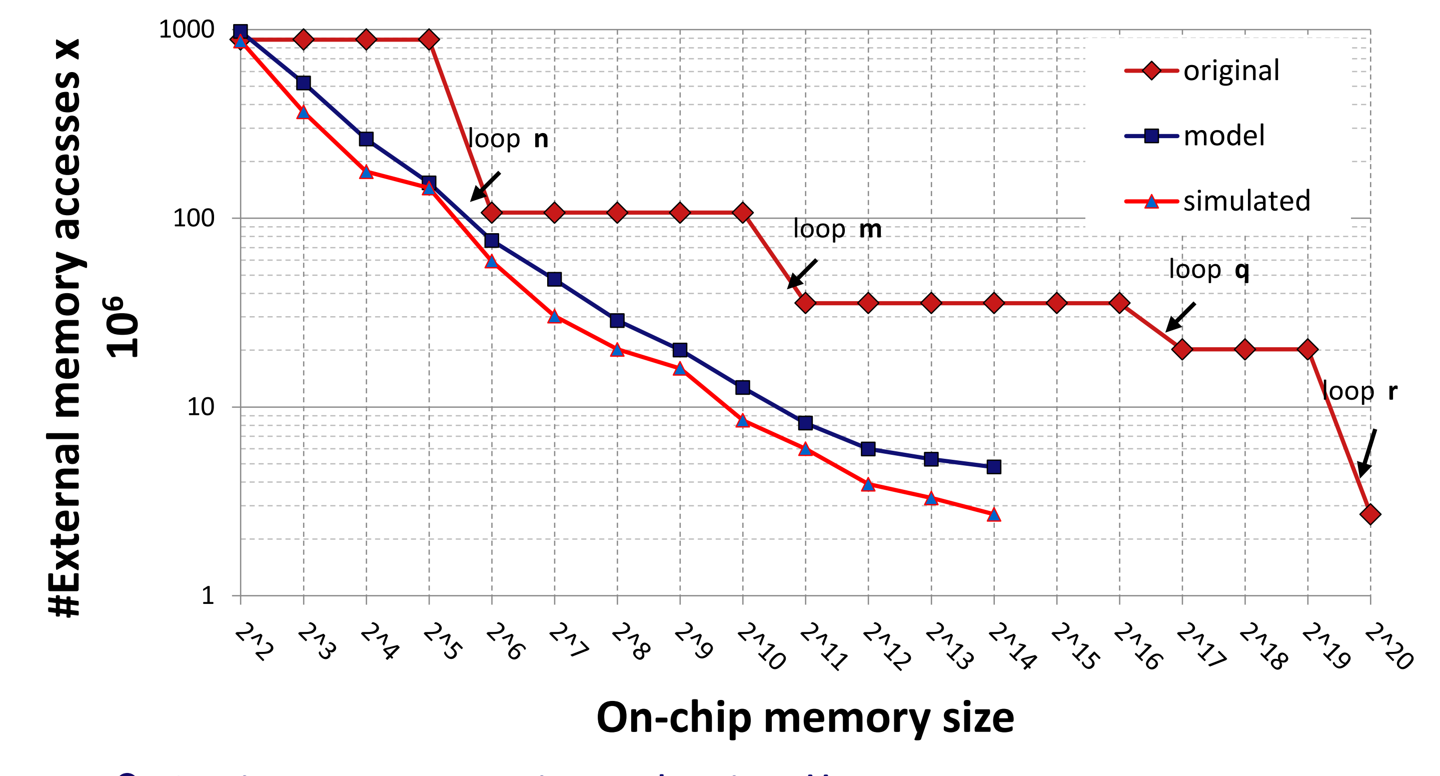

```
for(i=0; i<M; i+=Ti)
  for(j=0; j<N; j++)
    for(ii=i; ii<i+Ti; ii++)
      B[j][i]+=A[i]+C[j];
```

Data reuse: Bandwidth reduction 327x



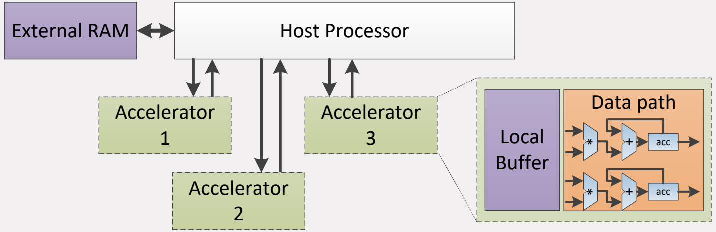
However: Huge on-chip memory required

After using our code transformer:



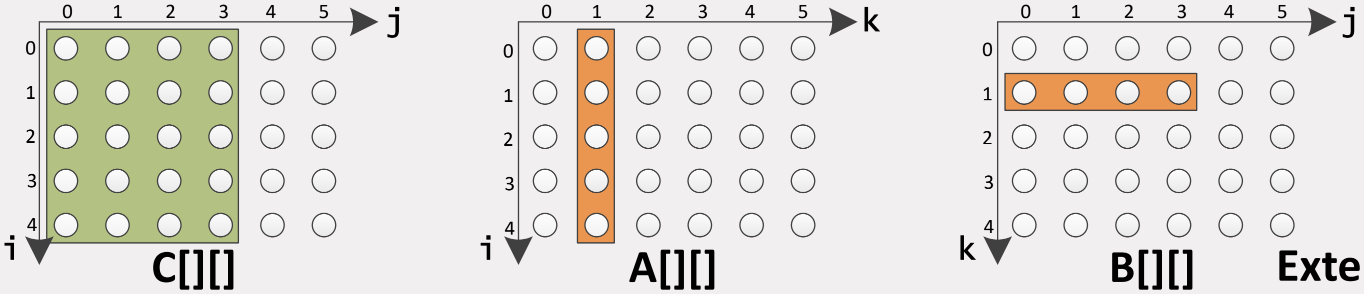
- 64 times memory size reduction !!

Inter-tile data reuse optimization



Operation List

```
for(i=0; i<Bi; i++){
  for(j=0; j<Bj; j++){
    C[i][j]=0;
    for(k=0; k<Bk; k++){
      C[i][j]+=A[i][k]*B[k][i];
}}}
```



External Access Reduction
Up to 2.4 times

M. Peemen et al., "Inter-Tile Reuse Optimization." (DATE 2015)

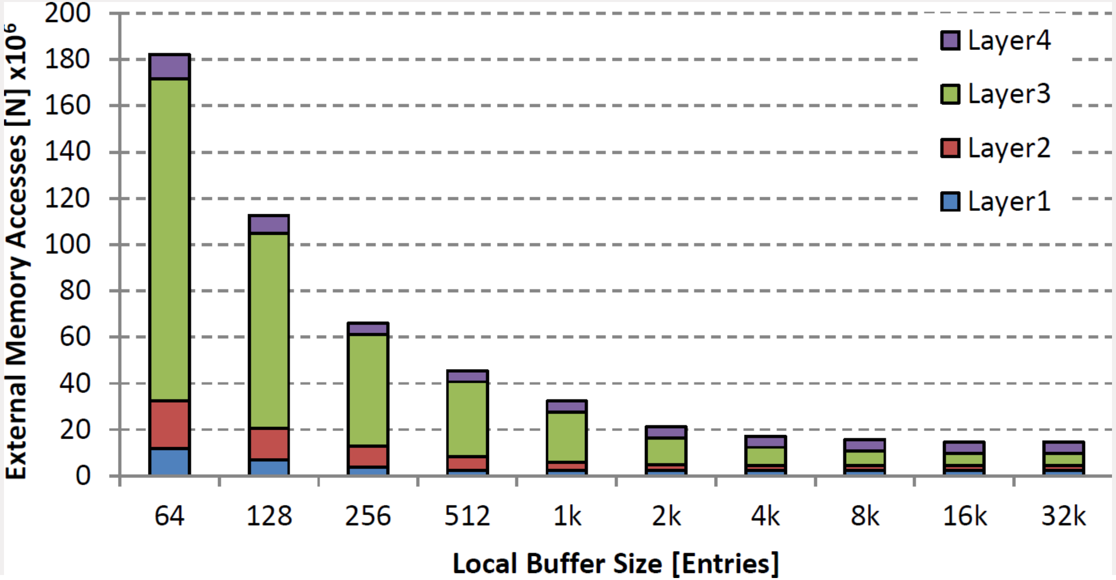
TU/e

The benefits of inter-tile reuse

Reuse data in a small local scratchpad

Reduce external communication by ~100x

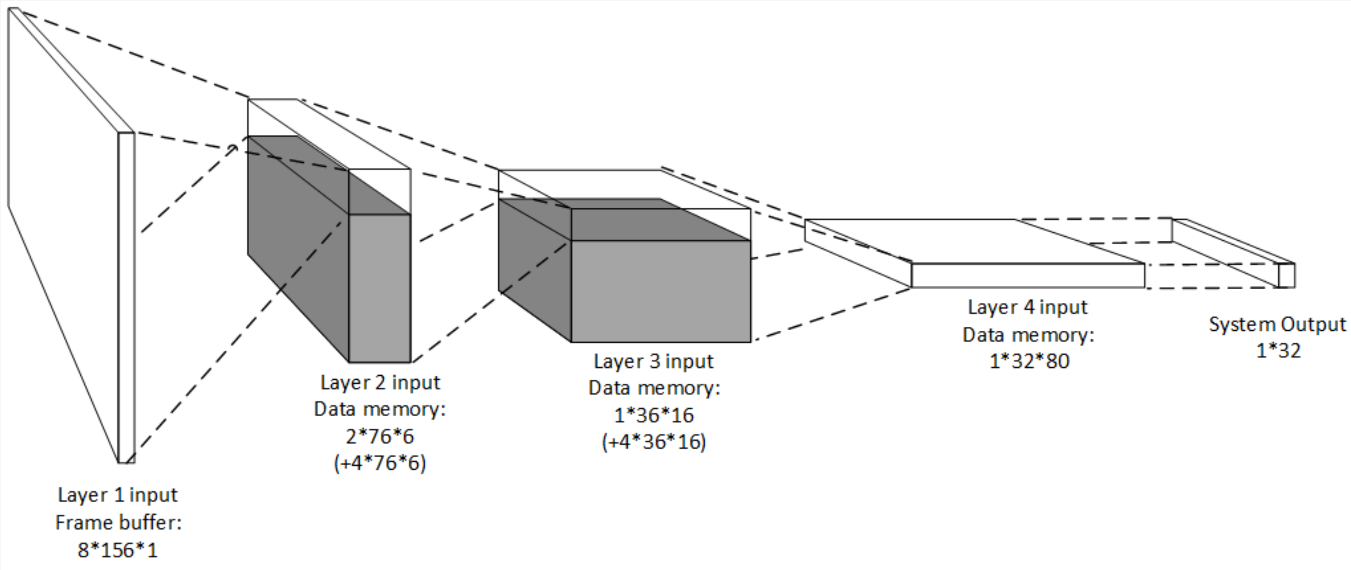
Against state-of-the-art tiling 2.1x reduction



M. Peemen et al., "Inter-Tile Reuse Optimization." (DATE 2015)

Next step in loop transformations: FUSION

- *Alwani e.a. Micro 2016*
- *AivoTTA (SAMOS 2018)*



Summary

Huge processing and storage demands

A lot of research on
efficiency improvements

DL networks get far more irregular

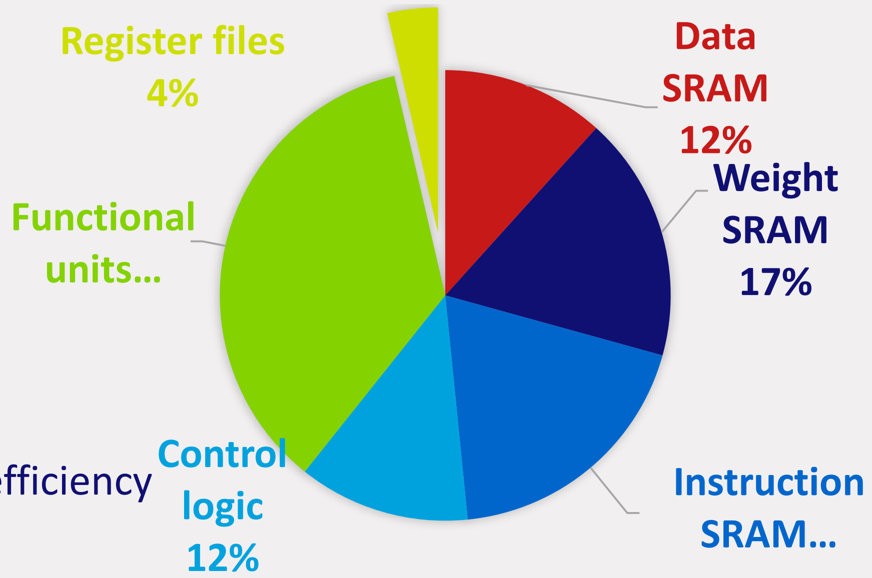
Flexible, but efficient, architectures needed

Select the sweet spot between flexibility and efficiency

Complex code:

Advanced code transformations and automated code generation needed

Energy breakdown AivoTTA, *SAMOS 2018*
400 MHz, total 11.3 mW



Being flexible does not have to be inefficient

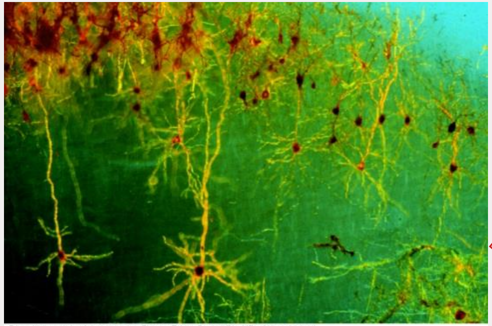
Example AivoTTA

- Very efficient Gops/W
- Memory efficient Gops/GB

	Tech. (nm)	GOPS /w	GOPS /GB	Control
Neuflow	45	490	50	Dataflow
Origami	65	803	521	Config.
NVE	40	559	125	VLIW
DianNao	65	931	-	FSM
ShiDianNao	65	400	-	FSM
SoC	28	1542	-	Config
DNPU	65	**2100**	-	FSM
TPU	28	1150	**1352**	CISC
AivoTTA 0.6V	**28**	**1434**	**1081**	**TTA**

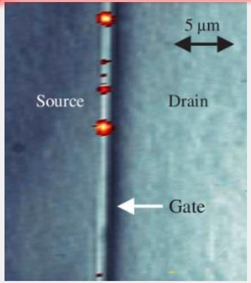
Deep Neural Networks

Neurobiology



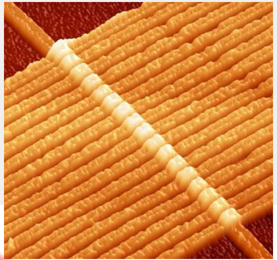
Pyramidal neurons of mouse cortex. Picture: Tobias Bonhoeffer, MPI of Neurobiology

Applications

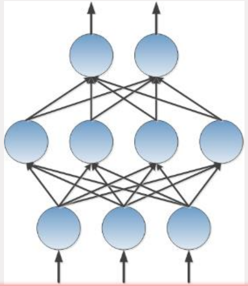


Constraints

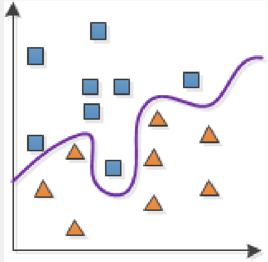
Technology

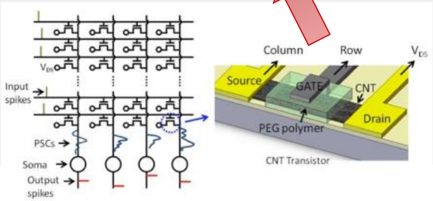


Innovations



Machine Learning





Neuromorphic

Synapses as Memristors from Intel

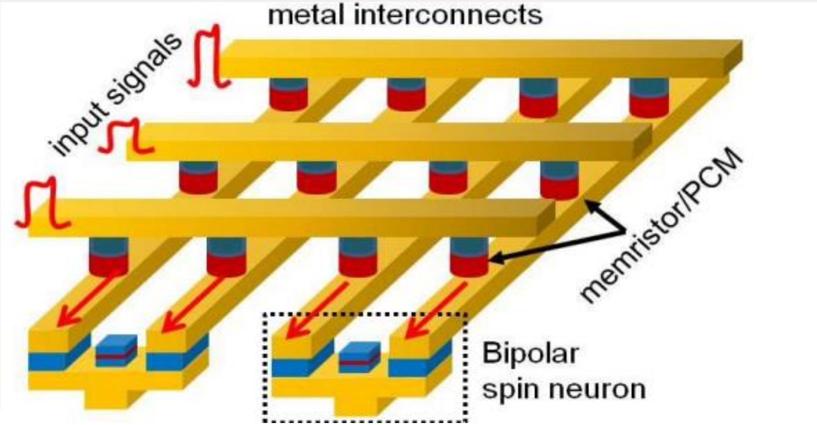
Proposal For Neuromorphic Hardware Using Spin Devices

[1]Mrigank Sharad, [2]Charles Augustine, [1]Georgios Panagopoulos, [1]Kaushik Roy
[1]Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
[2]Circuit Research Lab, Intel labs, Intel Corporation, Hillsboro, OR, US
msharad@.purdue.edu

Abstract: We present a design-scheme for ultra-low power Rest of the paper is organized as follows. Section 2 introduces

Memristor can be used as switch

Also analog storage of memristance

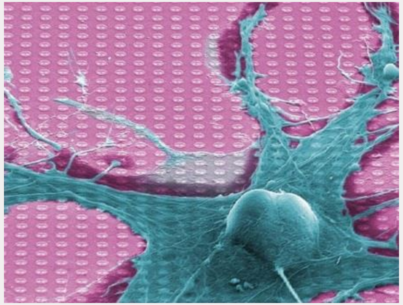


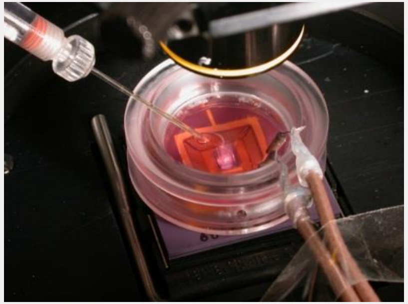
Beyond Silicon

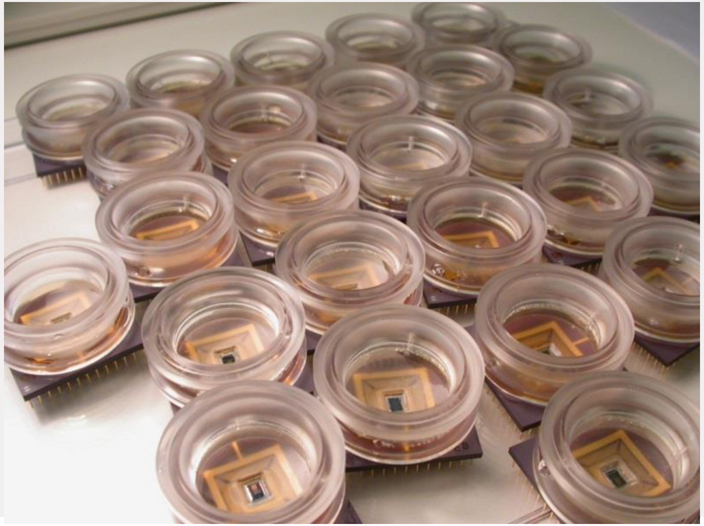
Infineon NeuroChip

Directly uses biological networks

Difficult to connect to other devices

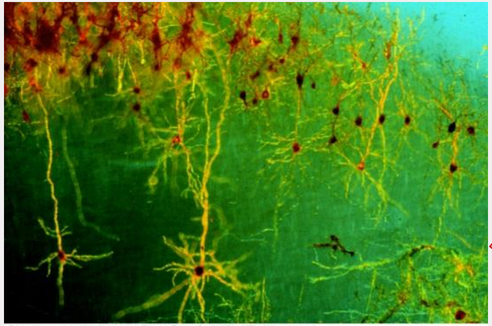






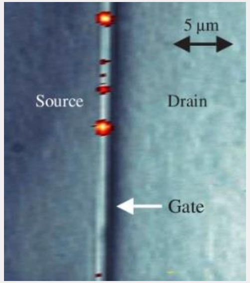
Deep Neural Networks

Neurobiology



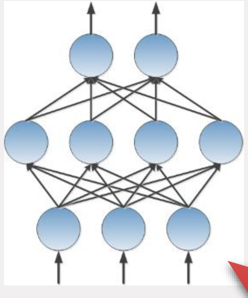
Pyramidal neurons of mouse cortex. Picture: Tobias Bonhoeffer, MPI of Neurobiology

Applications

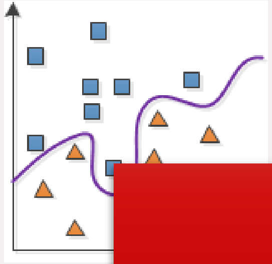


Constraints

Technology



Machine Learning



Innovations

Thank you for your attention

Neuromorphic

TU/e