Advanced School for Computing and Imaging

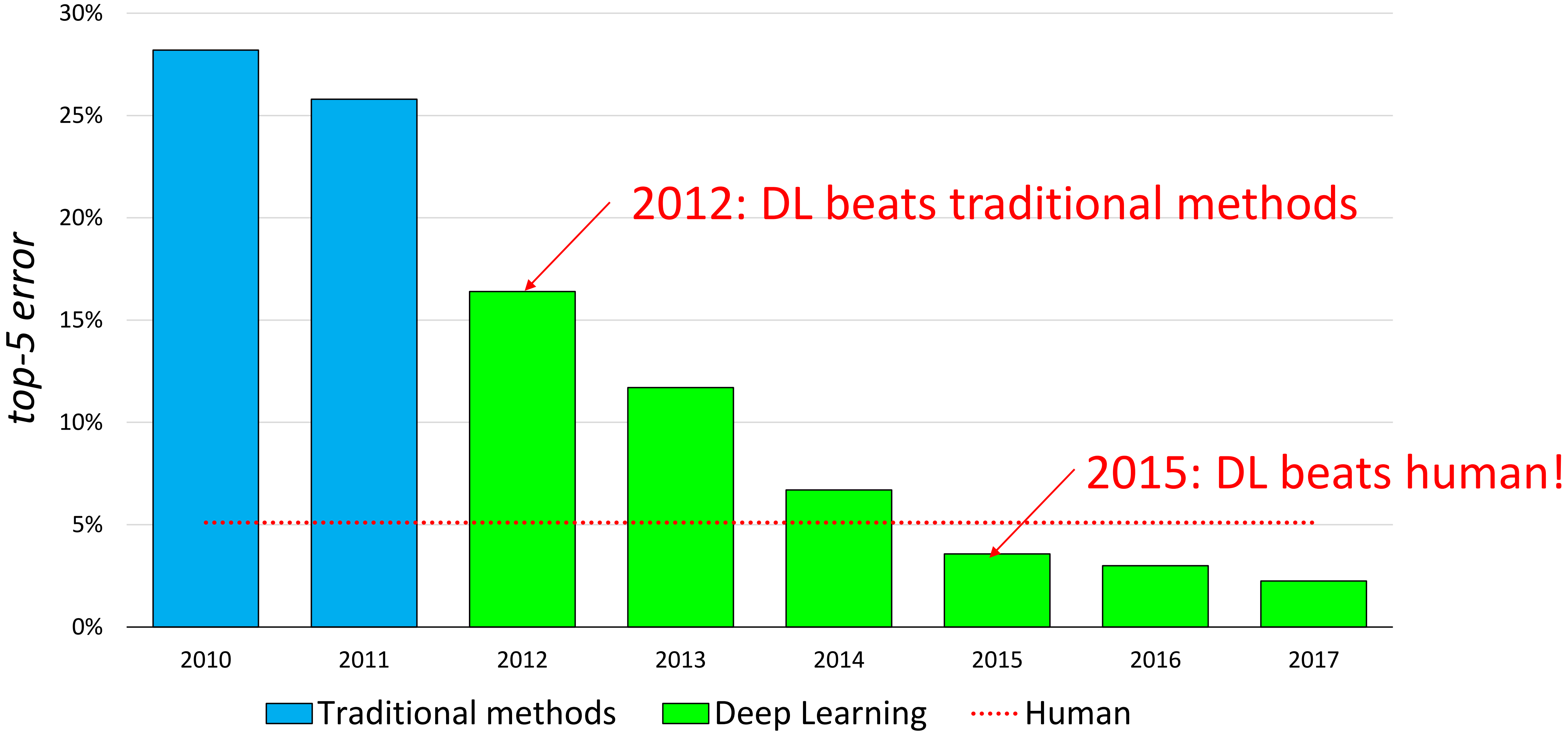
ASCI – Winterschool on Efficient Deep Learning Systems

Oud Poelgeest: Nov 28-Dec 1, 2023

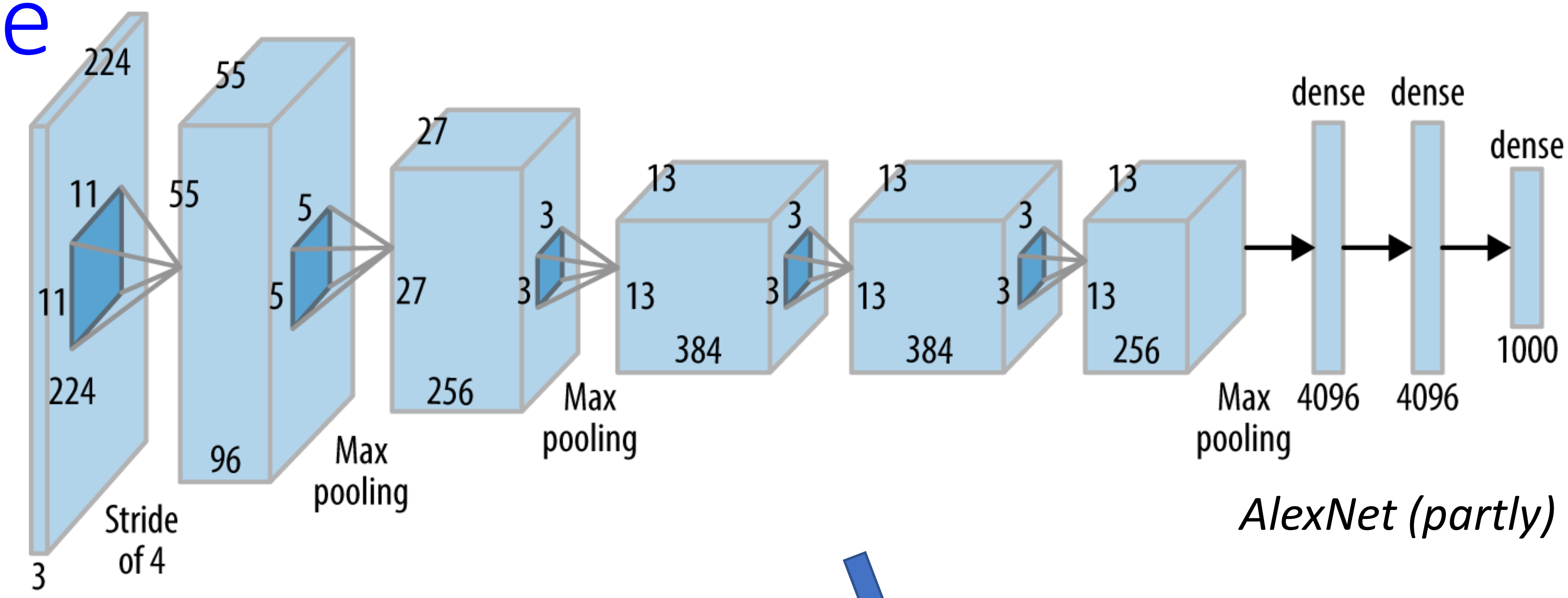
Once-Over-Lightly

Henk Corporaal
www.ics.ele.tue.nl/~heco
TUE

ImageNet Winners (top-5 classification error)



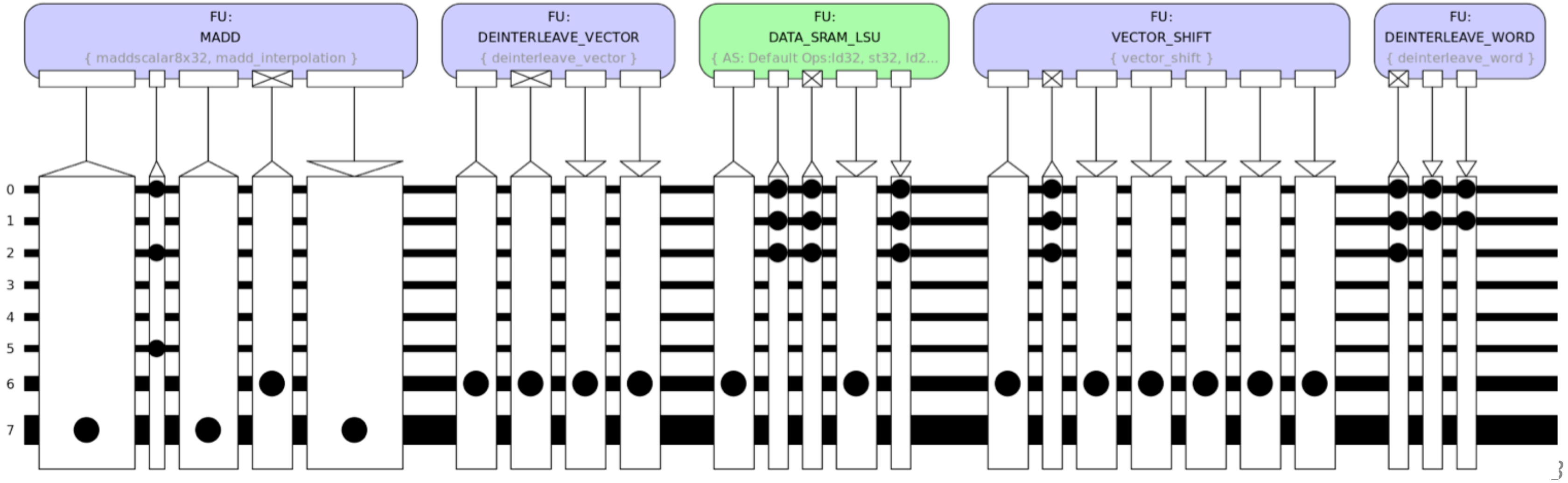
The BIG picture



AlexNet (partly)

NVIDIA Jetson Orin Nano embedded GPU

AivoTTA (partly)



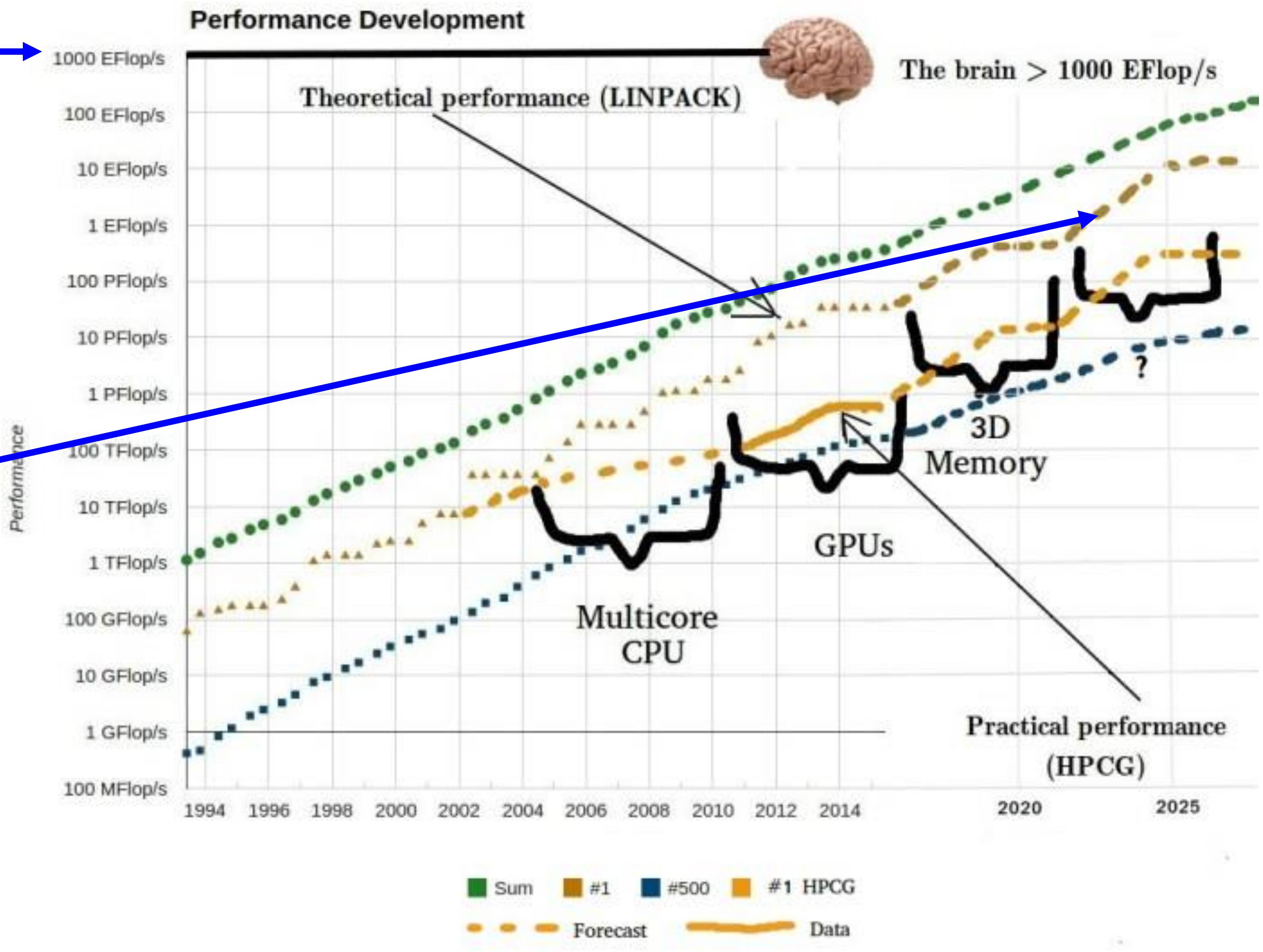
Frontier #1 top500.org

21 MWatt
606k CPU cores
8.3M GPU cores

Inspiration: Our Brain is extremely powerful & efficient

- Power = 20 Watt
- Speed = 1000 Exa Op/s*
- Energy/operation =
  Power / Speed =
  2×10^{-5} fJ/operation

- Compare to **Frontier**
  - Power = 21 MW
  - Peak = 1.194 ExaFlop/s
  - 680 m^2
  - Energy/op. = 19 pJ/op



*Tim Dettmers:
"making deep learning accessible"
2015

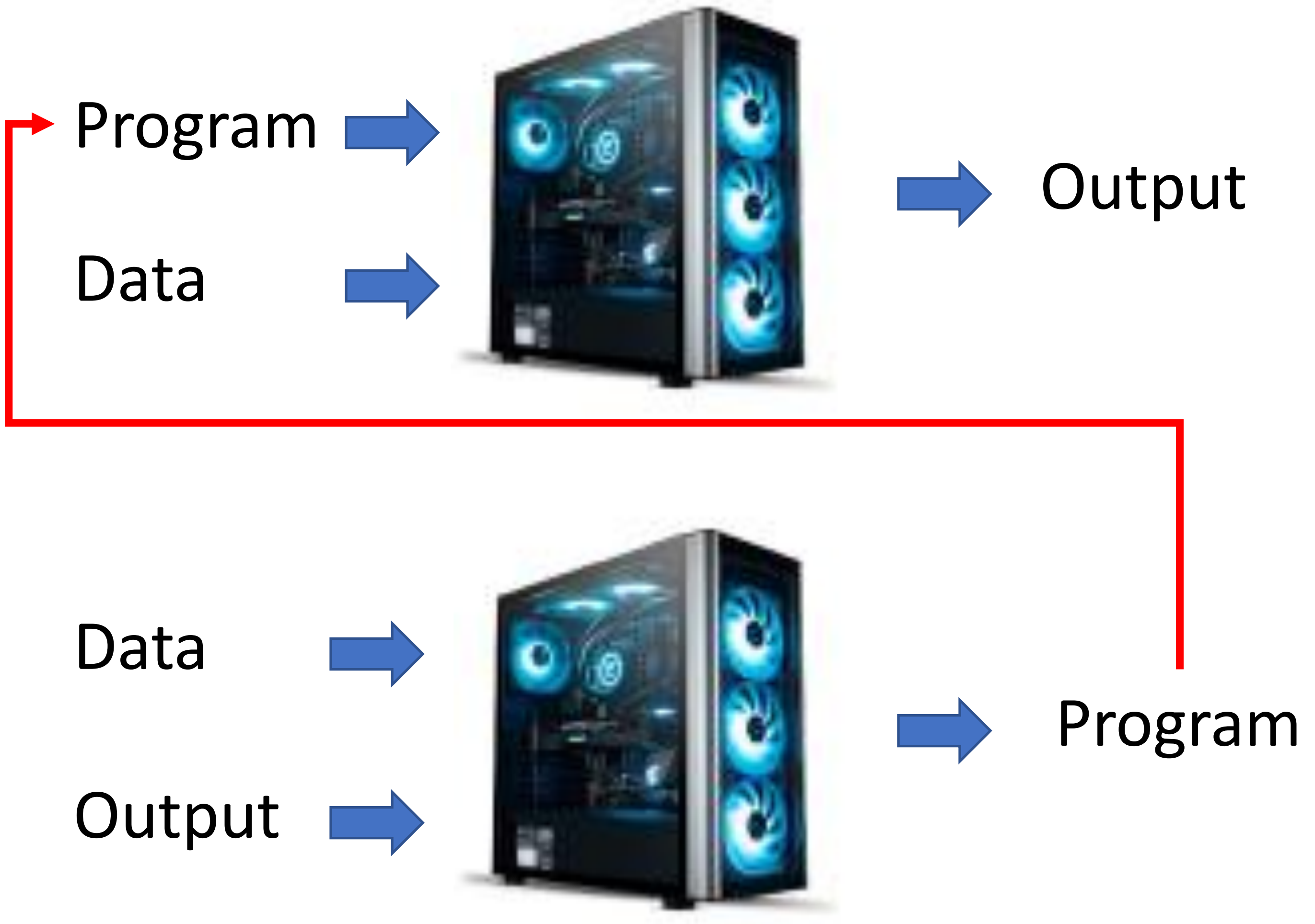
Once over lightly

- **What's (Deep) Learning?**
  - **self learning** algorithms
  - using **huge data sets** to learn
  - deep: **many** "learning **layers**"
  - **brain inspired**, based on neurons and synapses (connections)
  - high classification **accuracy**
- CNN: Convolutional Neural Network
  - Learning
- Other Network Models
- Optimizations
- Architectures

Learning

Traditional CS

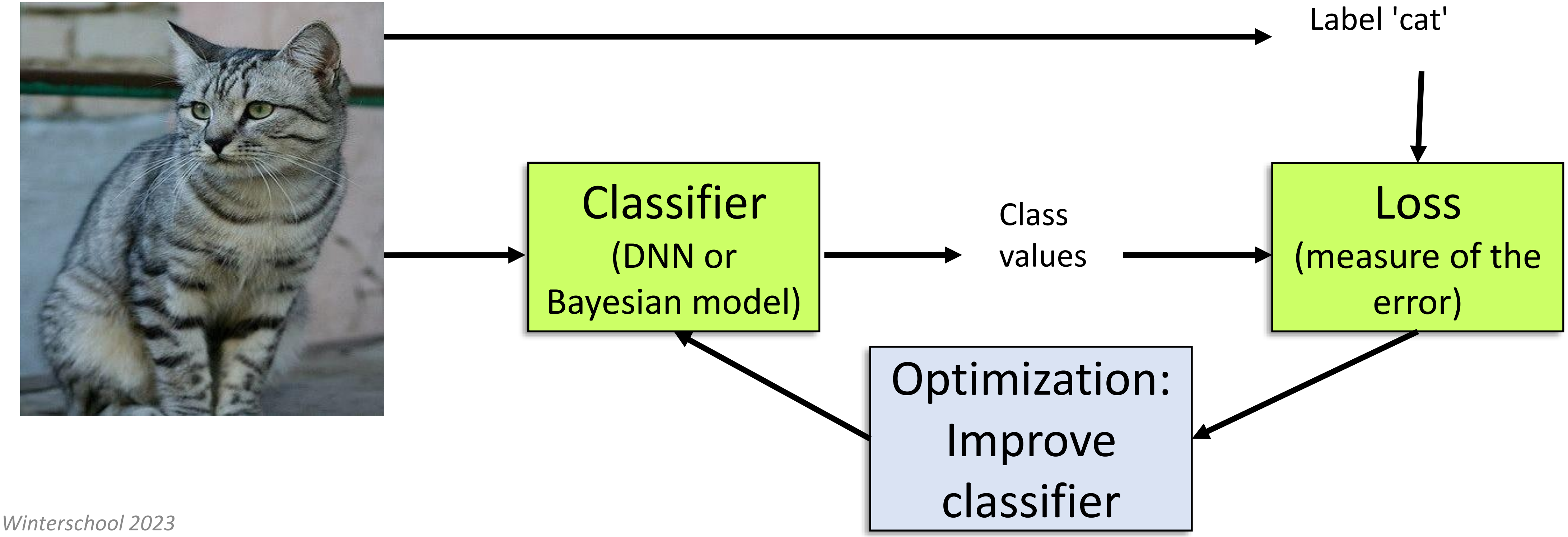
Machine Learning



Concl: We learn 'by example'

3 key components

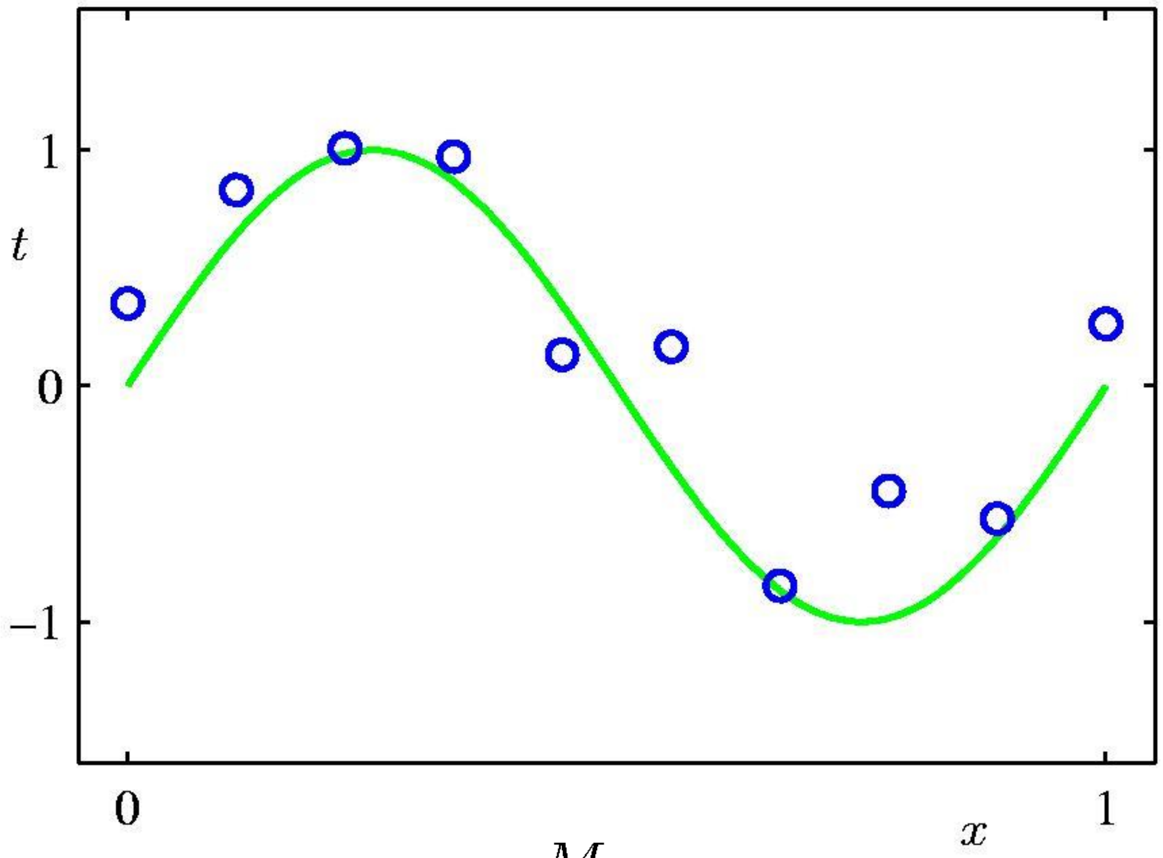
- Score function (**Classifier**) : Function to map input to output
- **Loss** Function : Evaluate quality of mapping
- **Optimization** Function : Update classifier



Example: Polynomial Curve Fitting

- ○ Measured values (x_i, t_i)
- **Generated** by *t=sin(x) + noise*

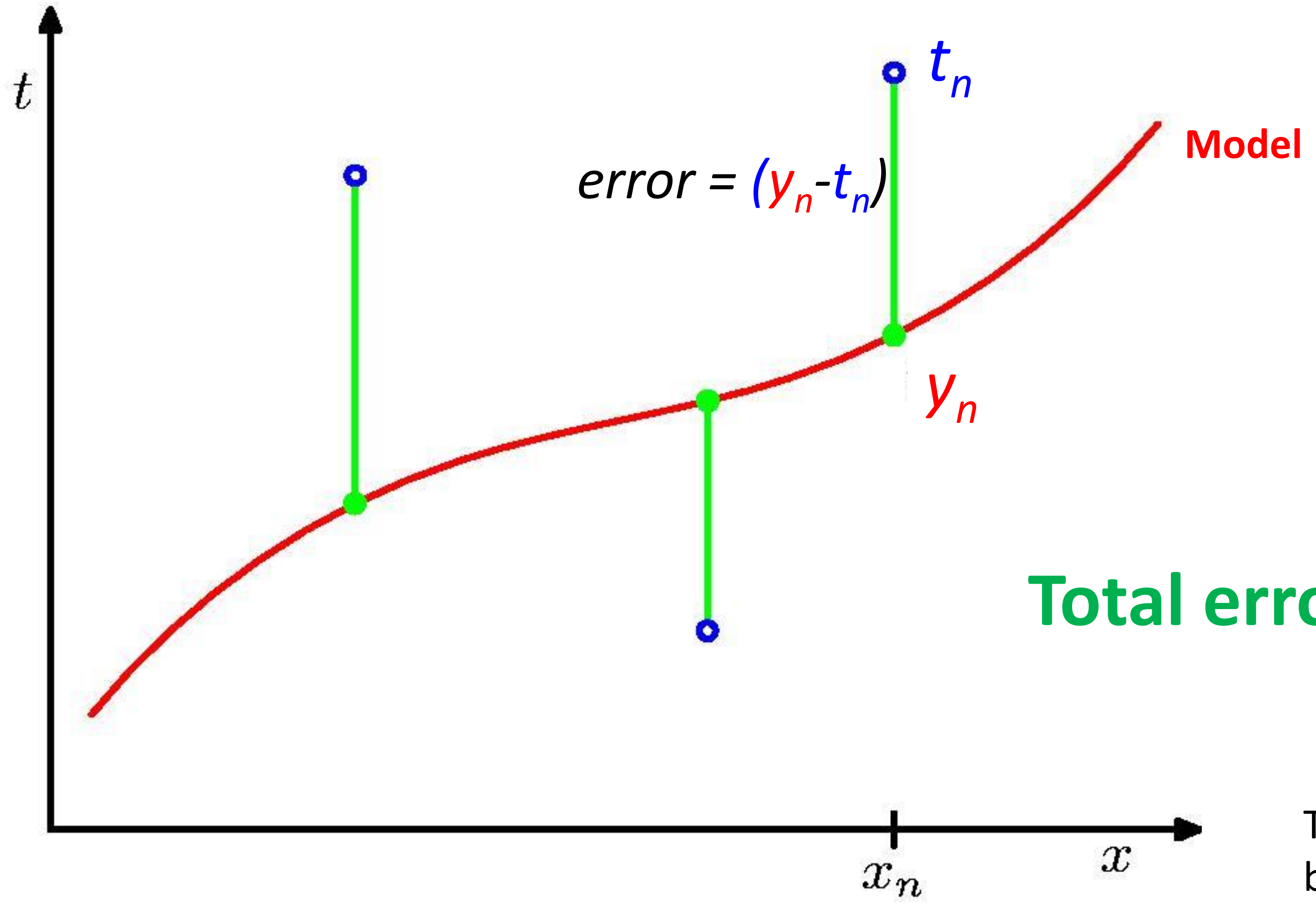
- Can we learn this curve from the measurements?



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Example taken from *C.M. Bishop: Pattern Recognition and Machine Learning*

Loss E(w): Sum-of-Squares Error Function



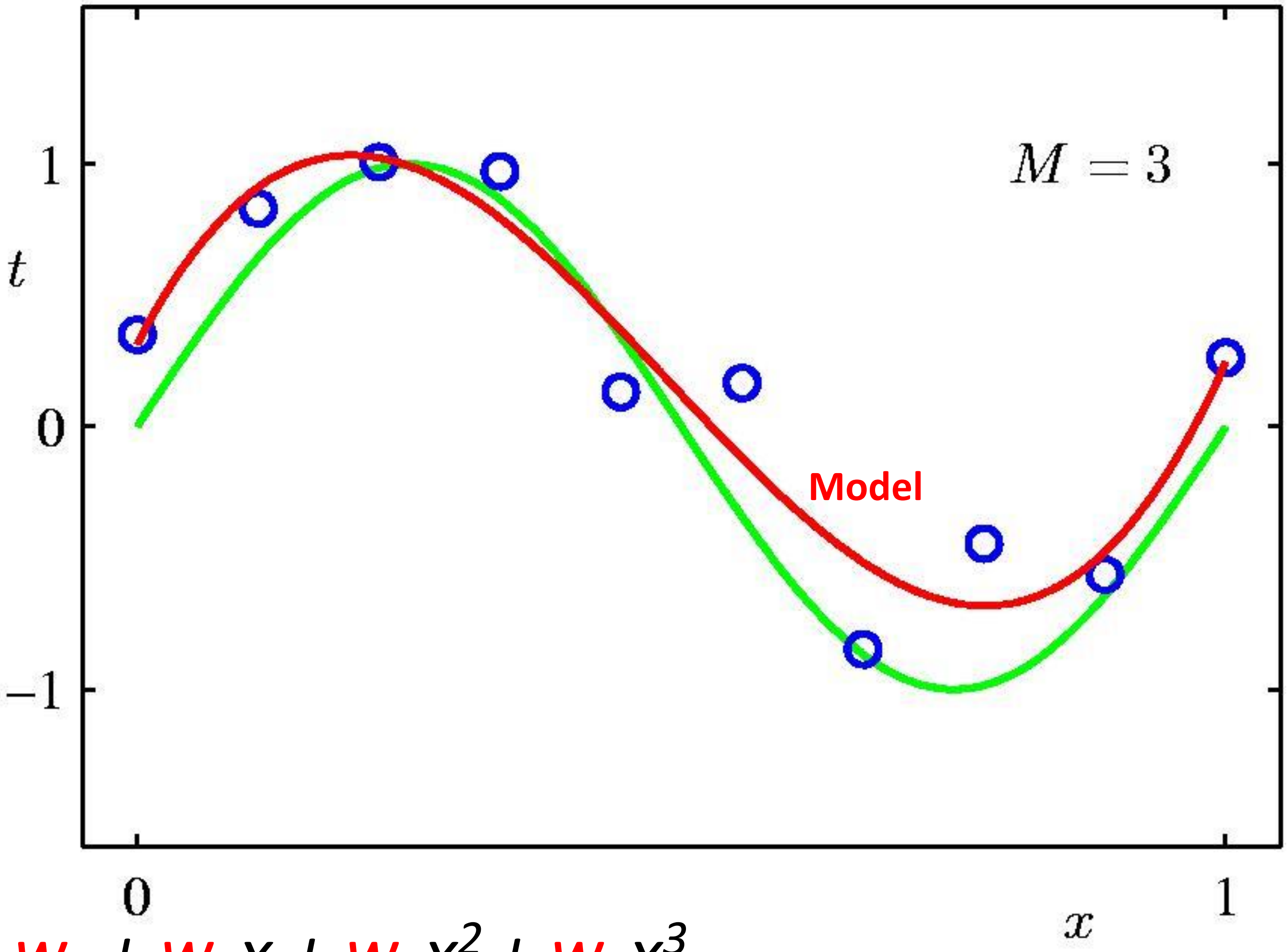
Total error: $E(\mathbf{w}) = \sum \{y_n - t_n\}^2$

Tune model such that difference between y_n and t_n gets smaller

E.g. assume model = 3rd Order Polynomial

- 4 parameters w_i

Parameter	Value
w_0	0.31
w_1	7.99
w_2	-25.43
w_3	17.37

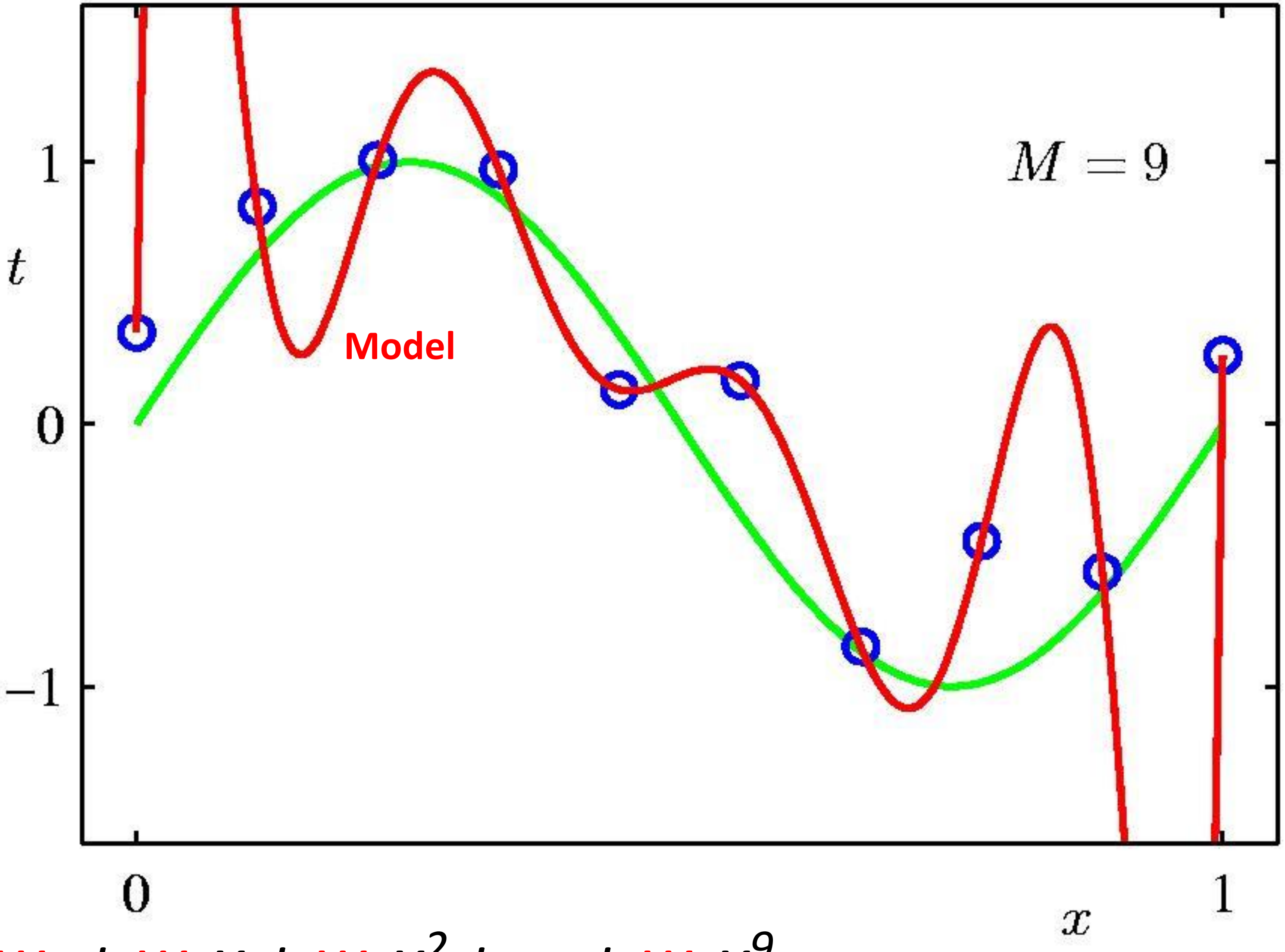


Model: $y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

9^{th} Order Polynomial => Overfitting

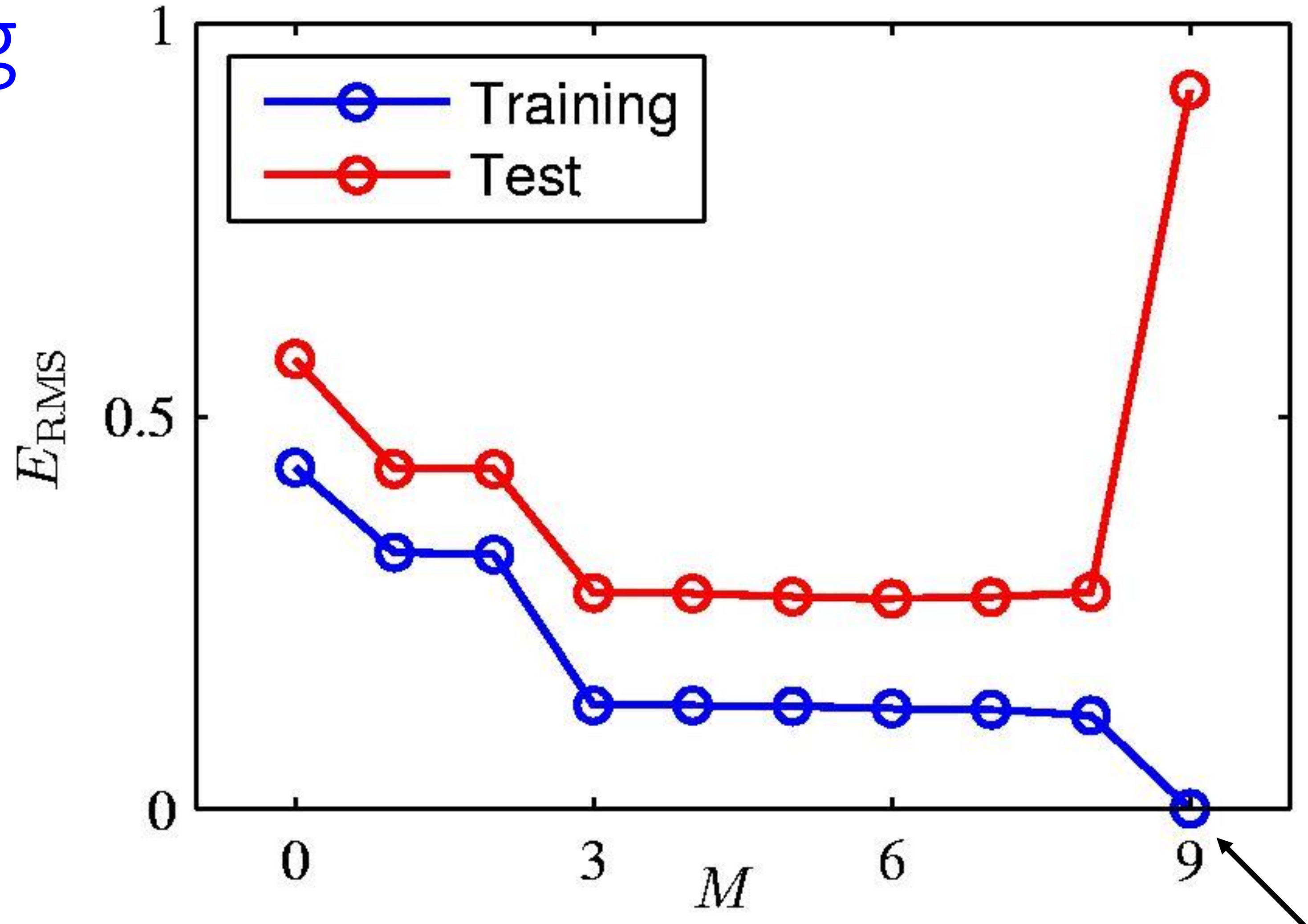
- 10 parameters w_i

Parameter	Value
w_0	0.35
w_1	232
w_2	-5321
w_3	45688
w_4	-231630
w_5	640042
w_6	-1061800
w_7	1042400
w_8	-557682
w_9	125201



Model: $y(x,\mathbf{w}) = w_0 + w_1x + w_2x^2 + \ldots + w_9x^9$

Over-fitting



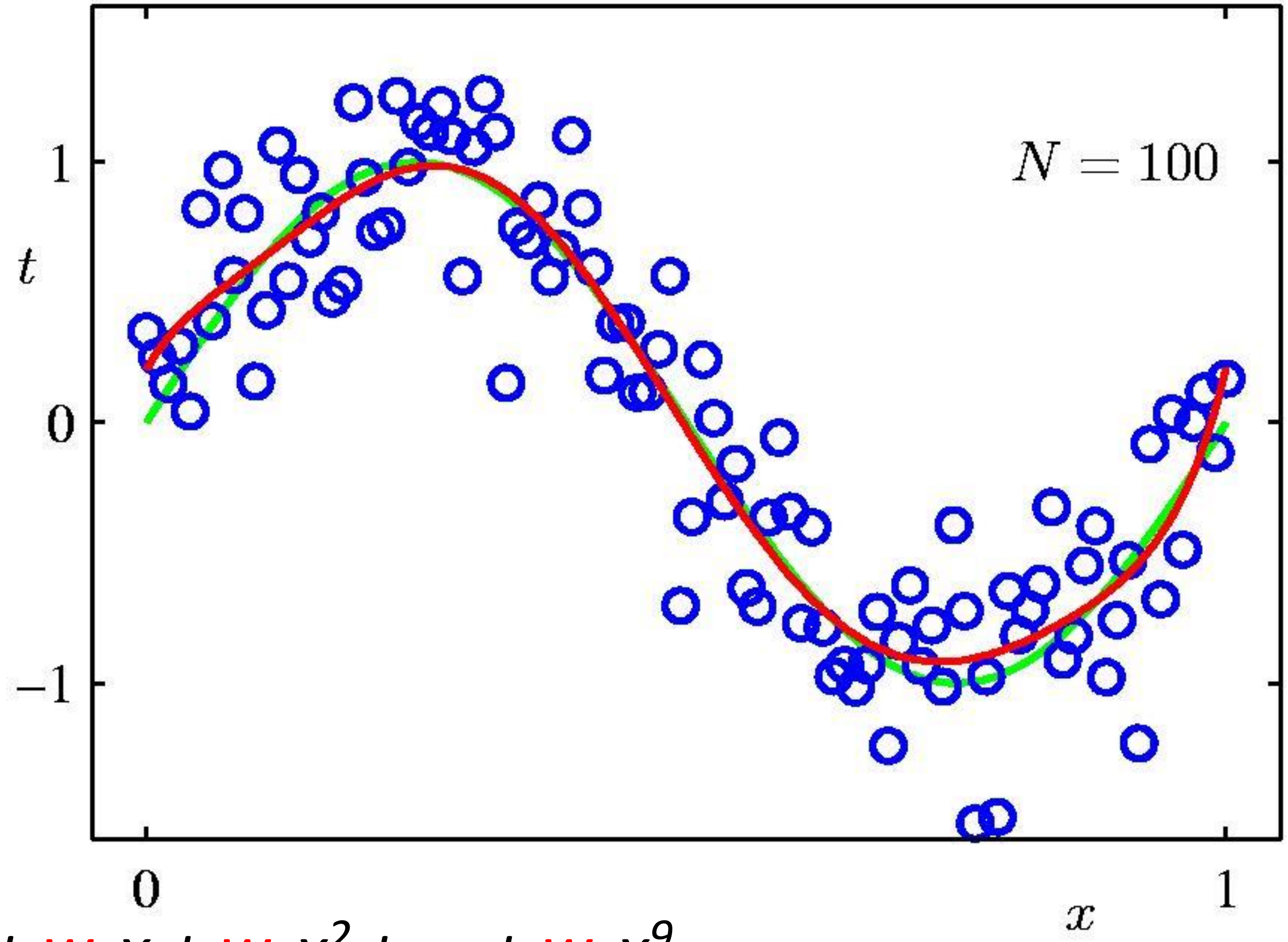
Overfitting

Root-Mean-Square (RMS) Error: $$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$$

Increasing Data Set Size to 100 points

9^{th} Order Polynomial

N = 100



Model: $y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_9 x^9$

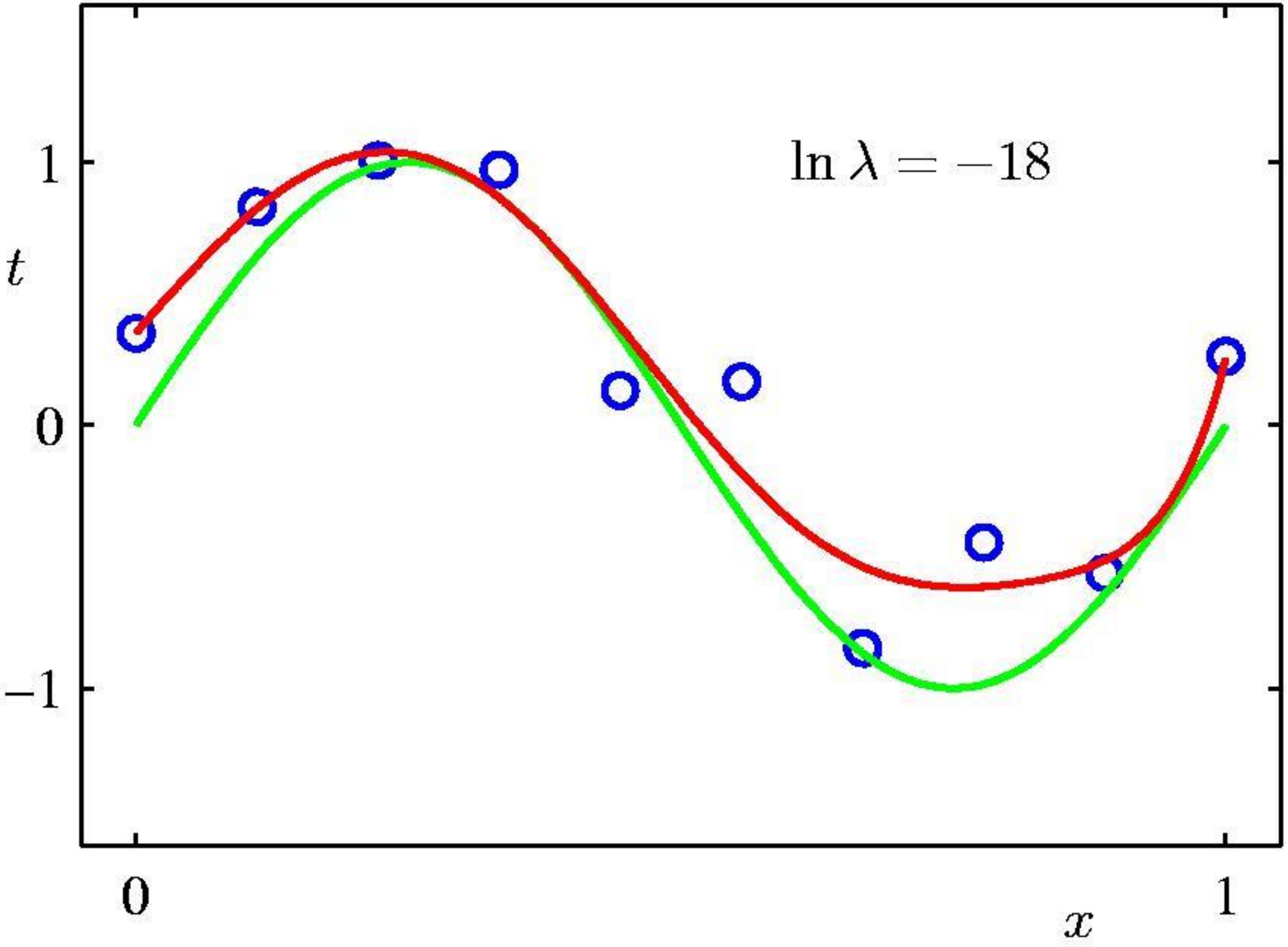
Regularization

- Penalize large coefficient values => add regularization term:

$$E(\mathbf{w}) = \sum \{y_n - t_n\}^2 + \boldsymbol{\lambda |w|^2}$$

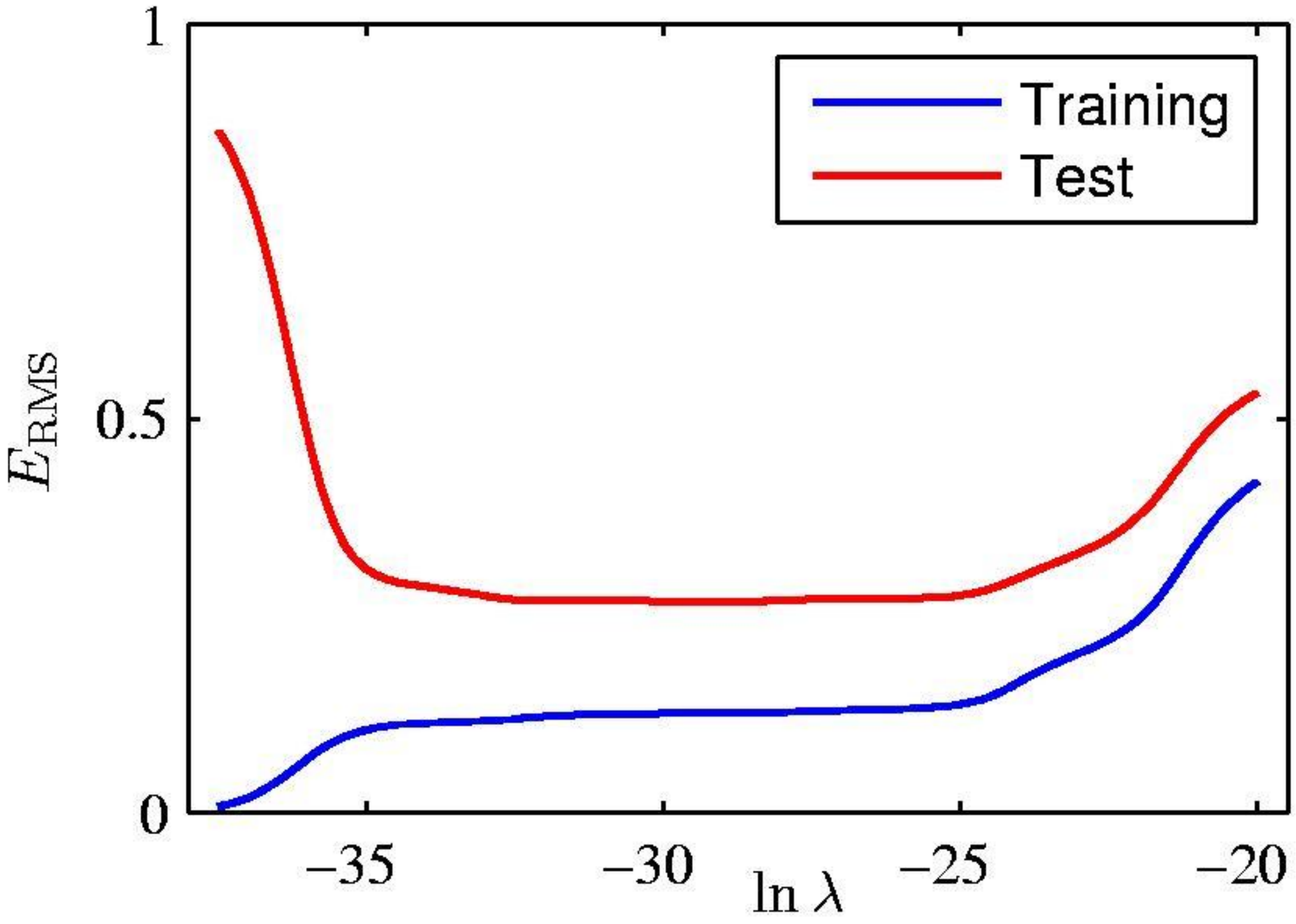
- λ is one of the many hyper parameters for learning

Regularization: ln λ = -18, M=9 (10 coeff)



Regularization: E_{RMS} vs. ln $\boldsymbol{\lambda}$

- **Hyperparameter tuning**: λ is critical !



Polynomial Coefficients

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

No regularization

Too much regularization

Once over lightly

- What's Deep Learning?
- **CNN: Convolutional Neural Network**
  - **Learning**
- Other Network Models
- Optimizations
- Architectures

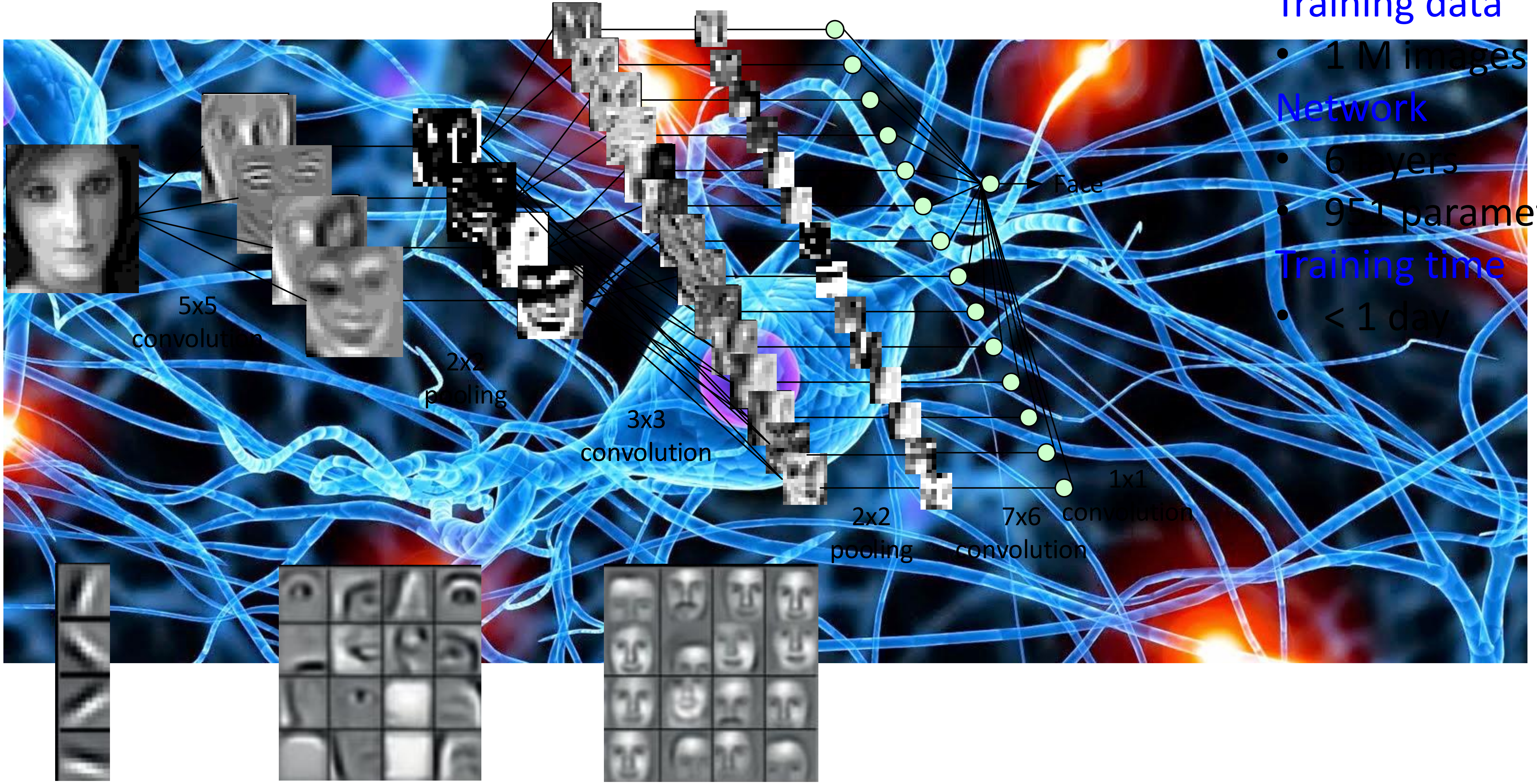
Deep Learning, a quick tour

A Simple Task
- Detect face

Training data
- 1 M images

Network
- 6 layers
- 951 parameters

Training time
- < 1 day

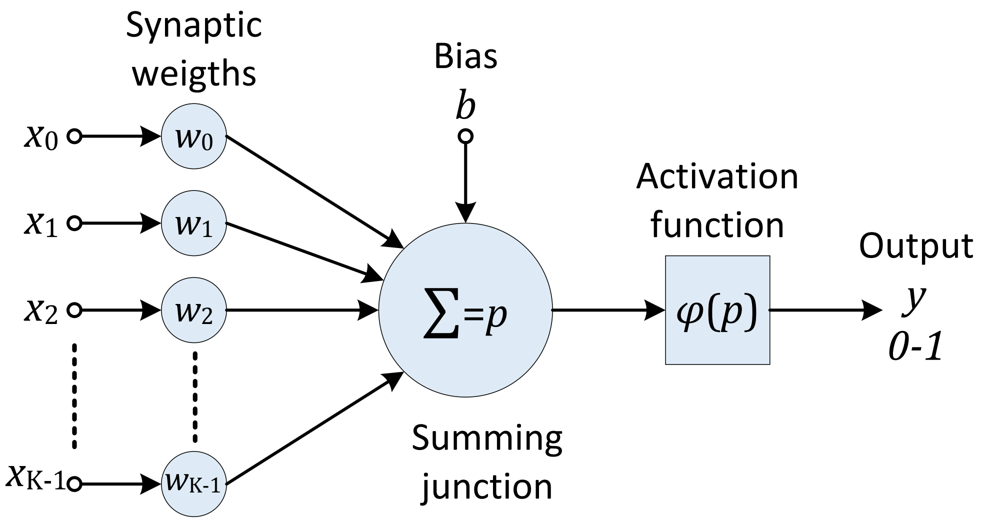


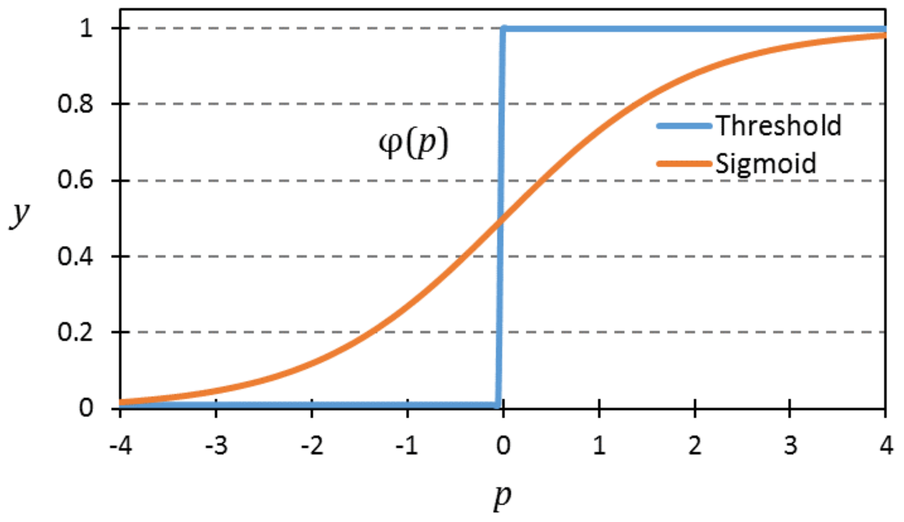
Low-level features
Mid-level features
High-level features

Convolutional network as a deep loop-nest

Example input vector

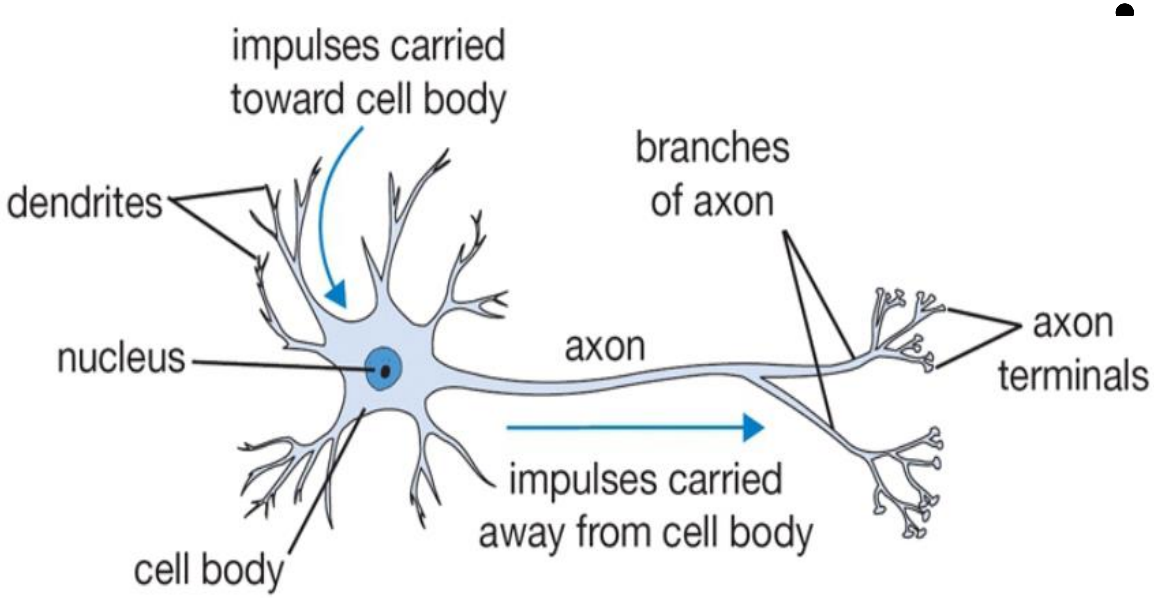
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
0	0	1	0	0
0	0	1	0	0
0	1	1	1	0
0	0	0	0	0





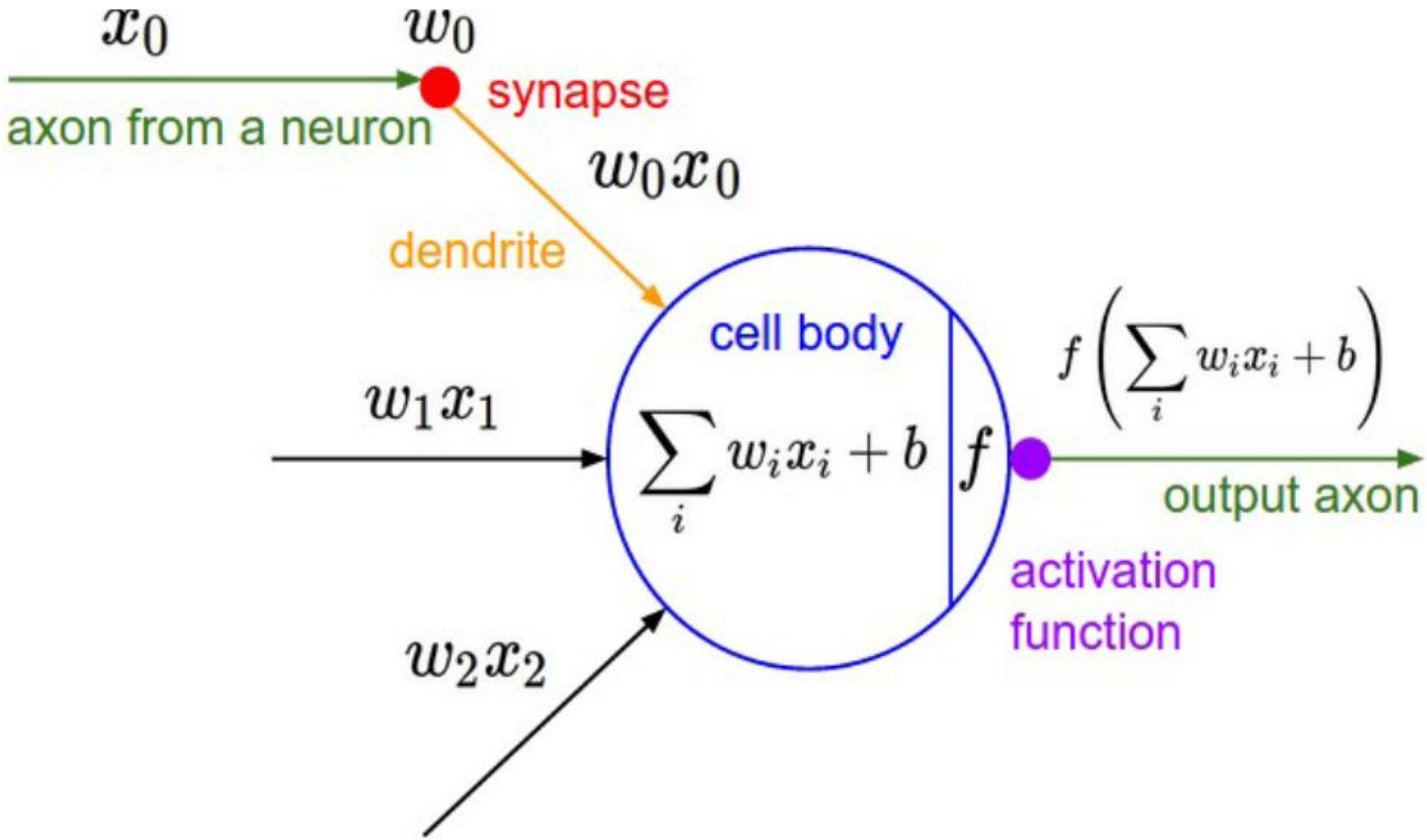

```
for l in layers:
  for o in output_maps[l]:
    for i in input_maps[l]:
      for x in columns[l]:
        for y in rows[l]:
          for kx in kernel_widht[l]:
            for ky in kernel_height[l]:
              out[l][o][x][y] += in[l][i][x+kx][y+ky] * w[l][i][o][kx][ky]
          fout[l][o][x][y]= f_act(out[l][o][x][y])
```

Our Brain

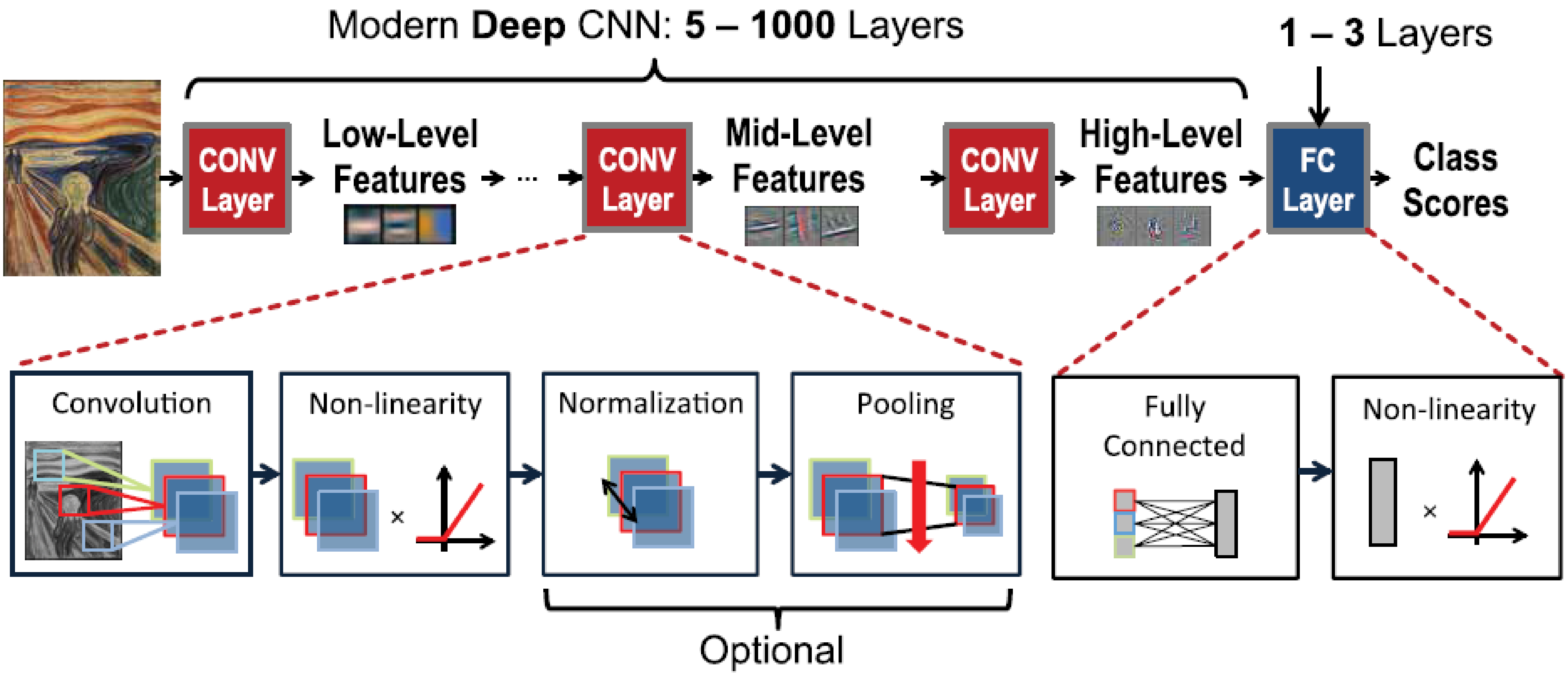


- The basic computational unit of the brain is a **neuron**
  - about 100 Billion neurons in our brain
  - Neurons are connected with nearly **10^{14} – 10^{15} synapses**
  - Neurons receive input signals from **dendrites** and produce output signal along **axon**, which interact with the dendrites of other neurons via **synaptic weights**
- Synaptic **weights** – learnable & control influence strength

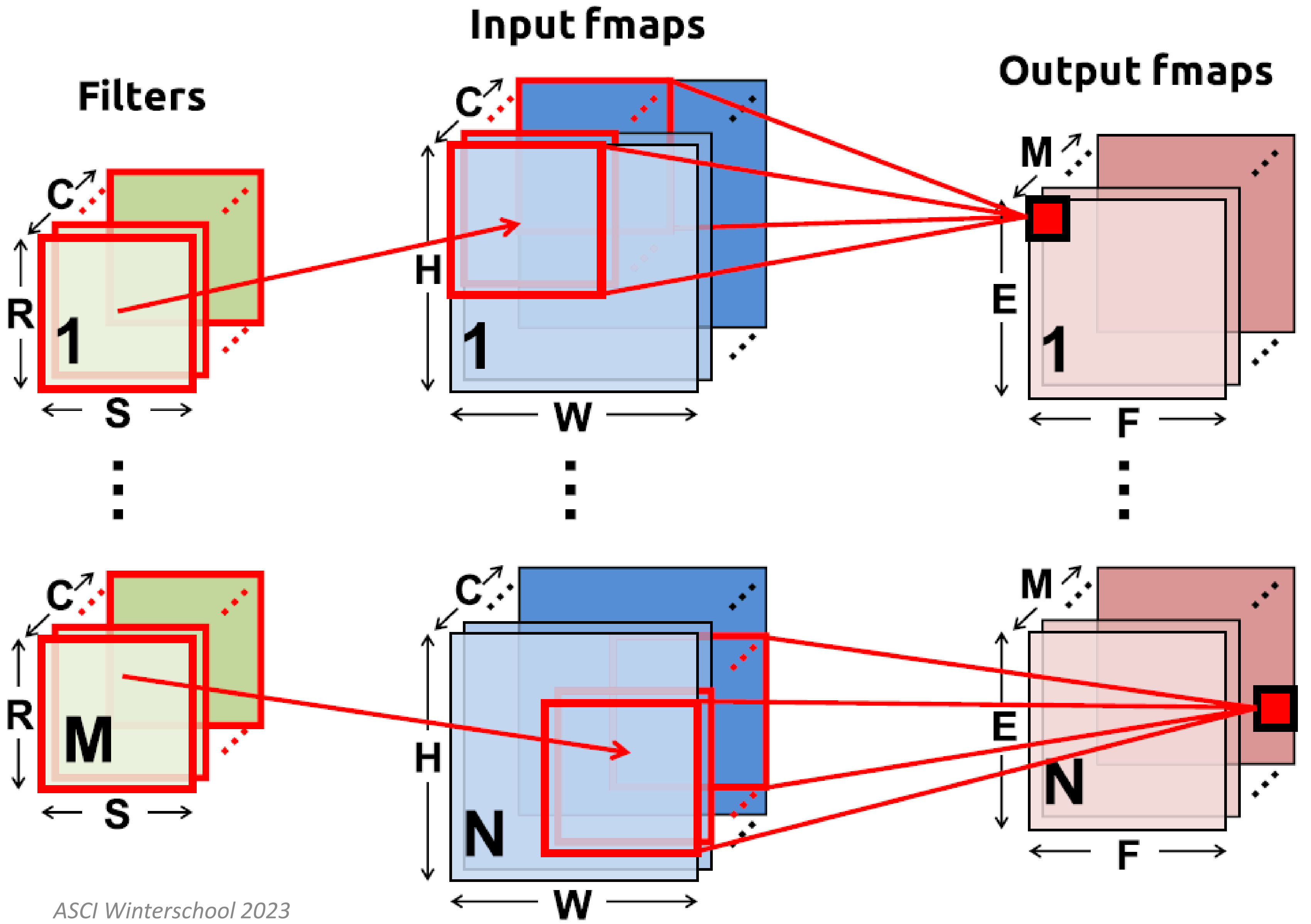
Artificial Neuron



DNN structure



Convolution in CNNs: **1 layer**



- N = batch size
- C input feature maps of size HxW
- M output feature maps of size ExF
- M filters of size RxS

Convolution code

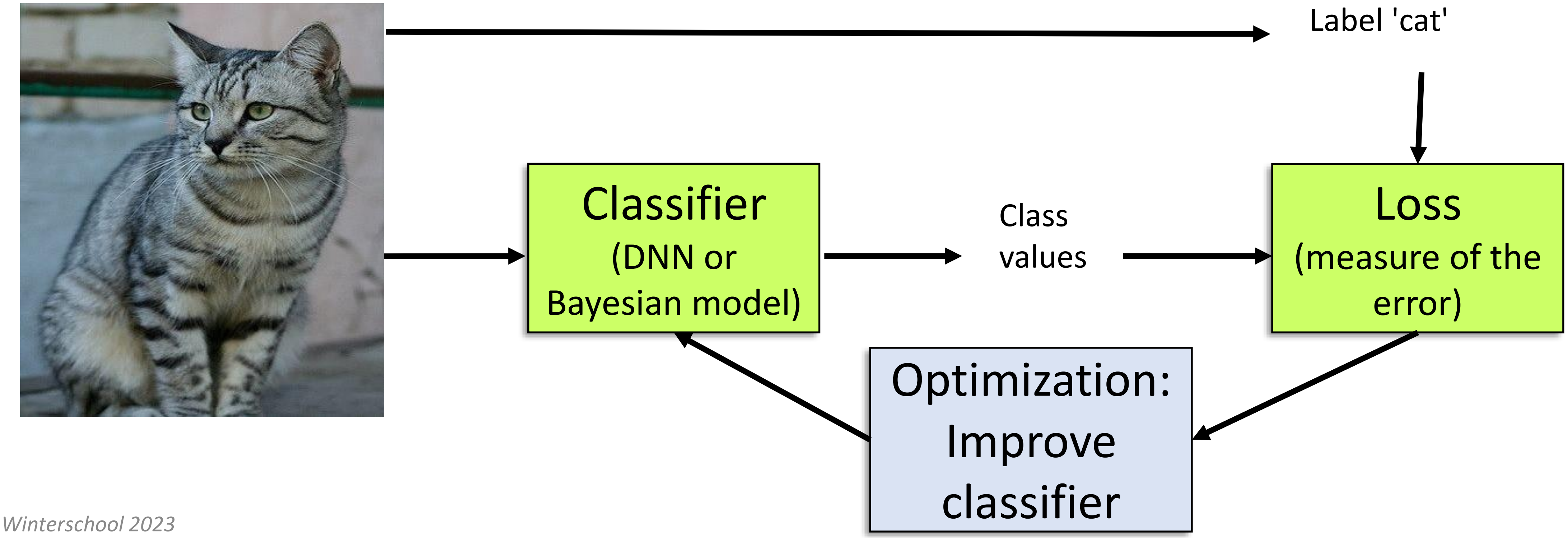
for each output fmap value

convolve
a window
and apply
activation

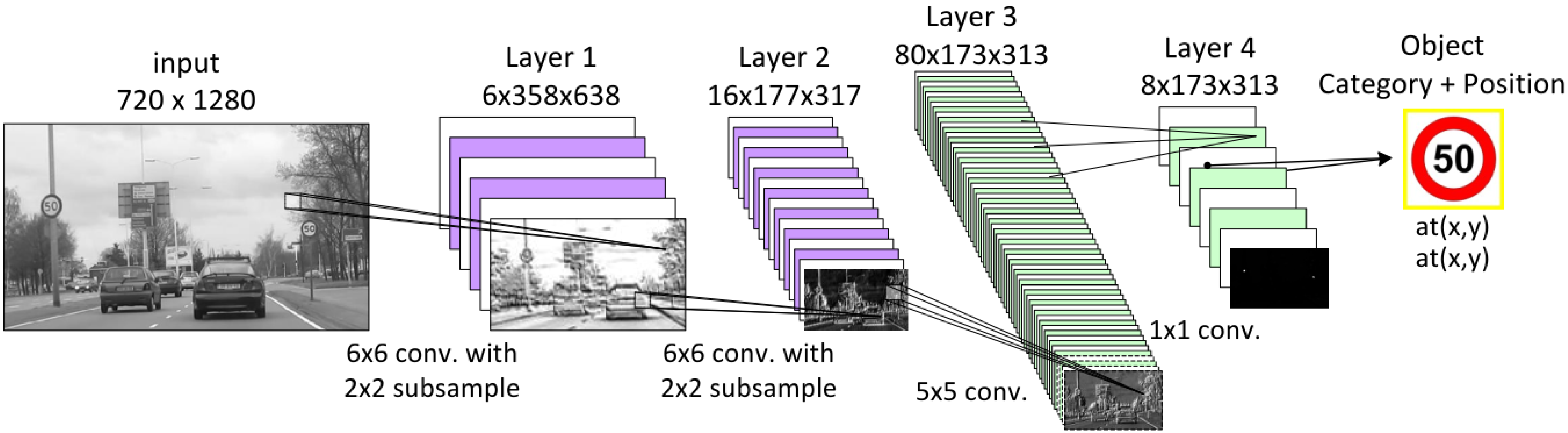
```
for (n=0; n<N; n++) {
    for (m=0; m<M; m++) {
        for (x=0; x<F; x++) {
            for (y=0; y<E; y++) {

                O[n][m][x][y] = B[m];
                for (i=0; i<R; i++) {
                    for (j=0; j<S; j++) {
                        for (k=0; k<C; k++) {
                            O[n][m][x][y] += I[n][k][Ux+i][Uy+j] × W[m][k][i][j];
                        }
                    }
                }

                O[n][m][x][y] = Activation(O[n][m][x][y]);
            }
        }
    }
}
```

Learning

- Score function (**Classifier**) : Function to map input to output
- **Loss** Function : Evaluate quality of mapping
- **Optimization** Function : Update classifier



How do we learn all these coefficients

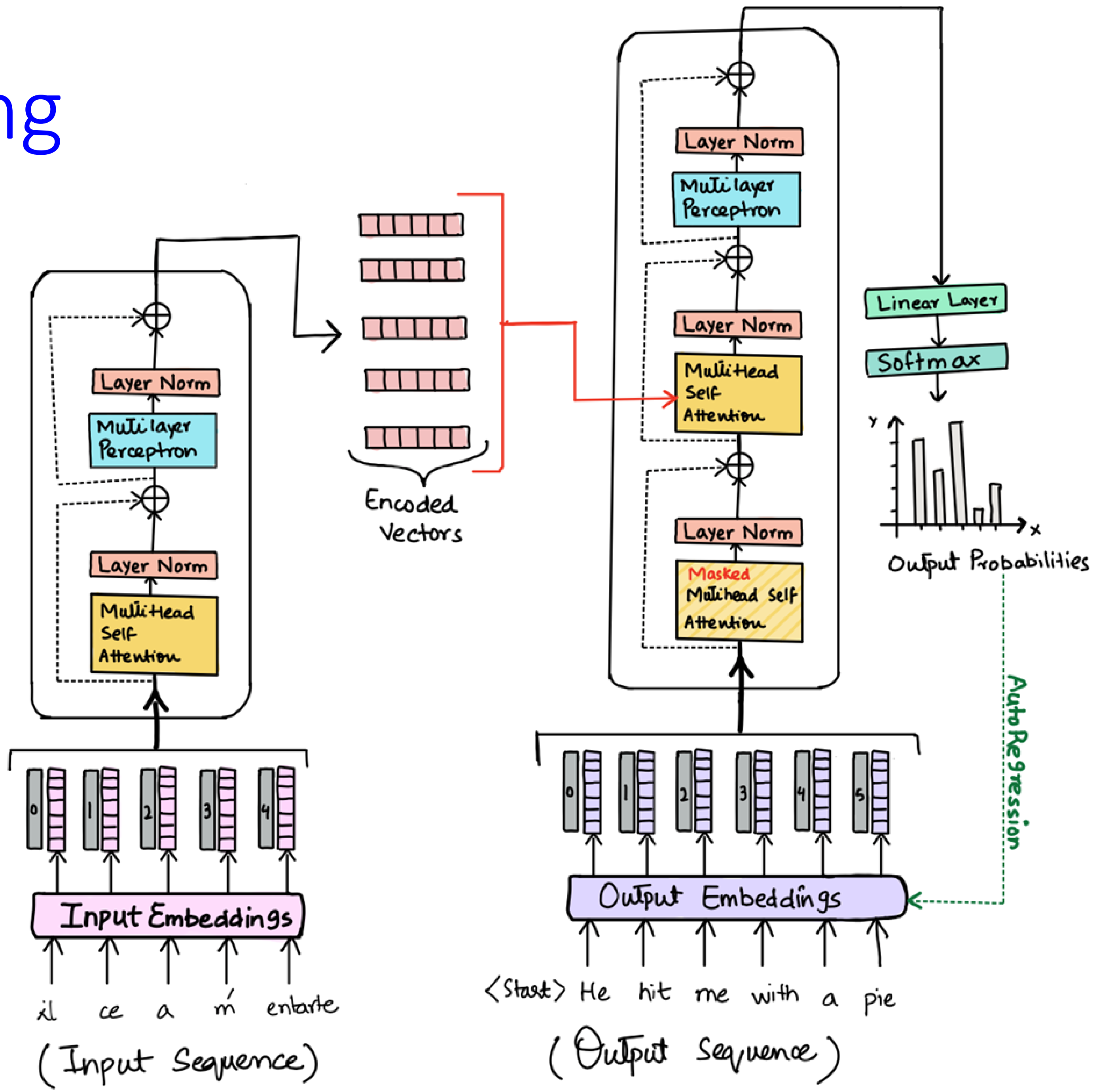


- **Back Propagation !!**
  - calculate partial derivatives: **δ**Loss / **δ**w , for all w
  - update w
  - repeat many times, with many labeled inputs

Once over lightly

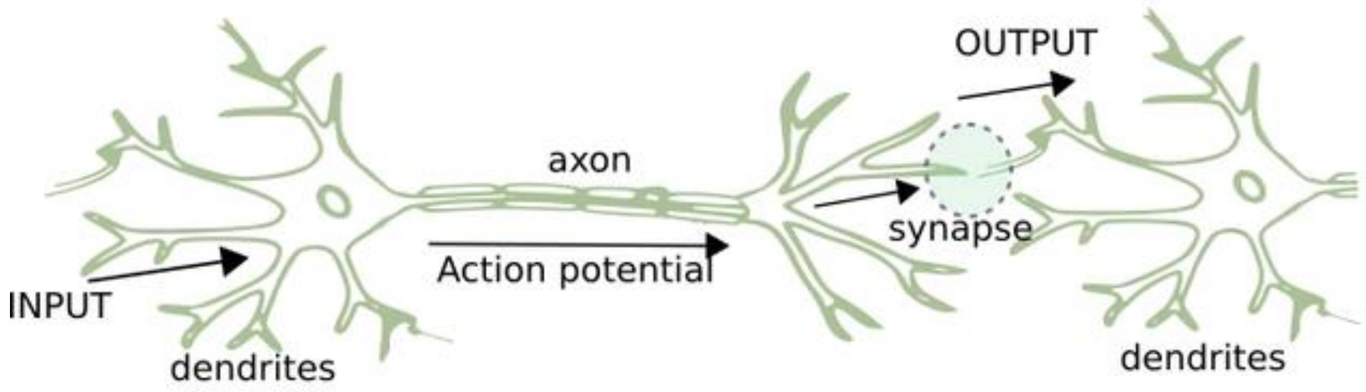
- What's Deep Learning?
- CNN: Convolutional Neural Network
  - Learning
- **Other Network Models**
  - **Transformer**
  - **SNN: Spiking Neural Network**
- Optimizations
- Architectures

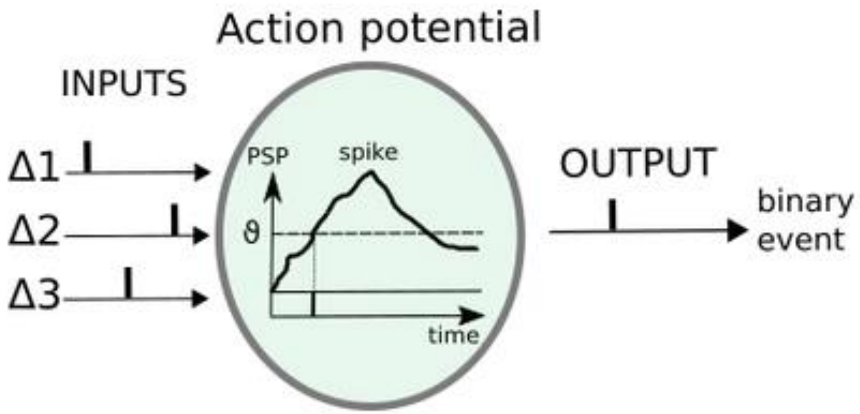
Transformer using Attention NW



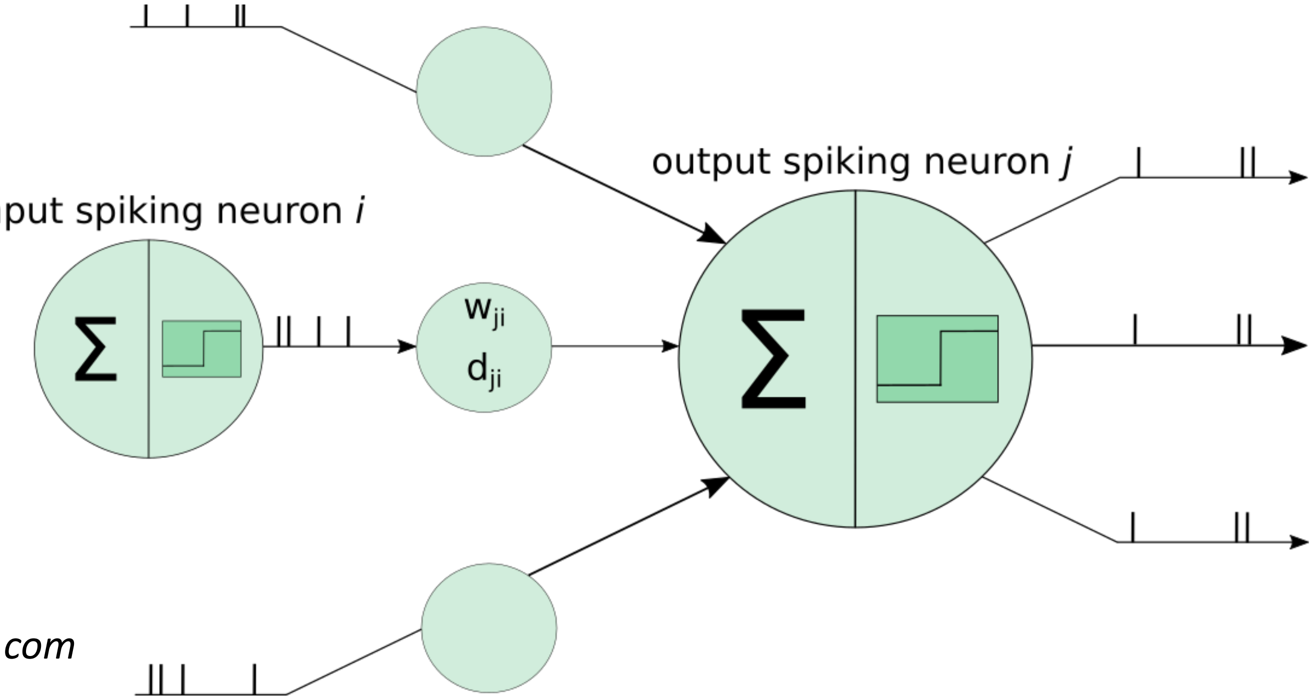
Sudipto Baul @ medium.com

SNN: Spiking Neural Network (more brain inspired)





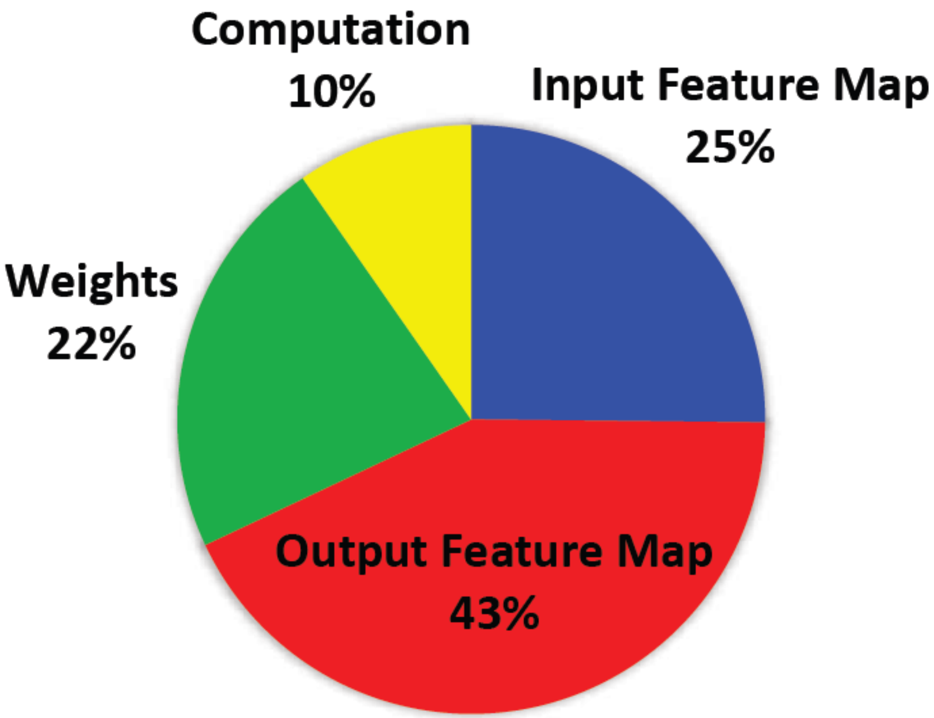
Model:



Jesus L.Lobo @ towardsdatascience.com

Once over lightly

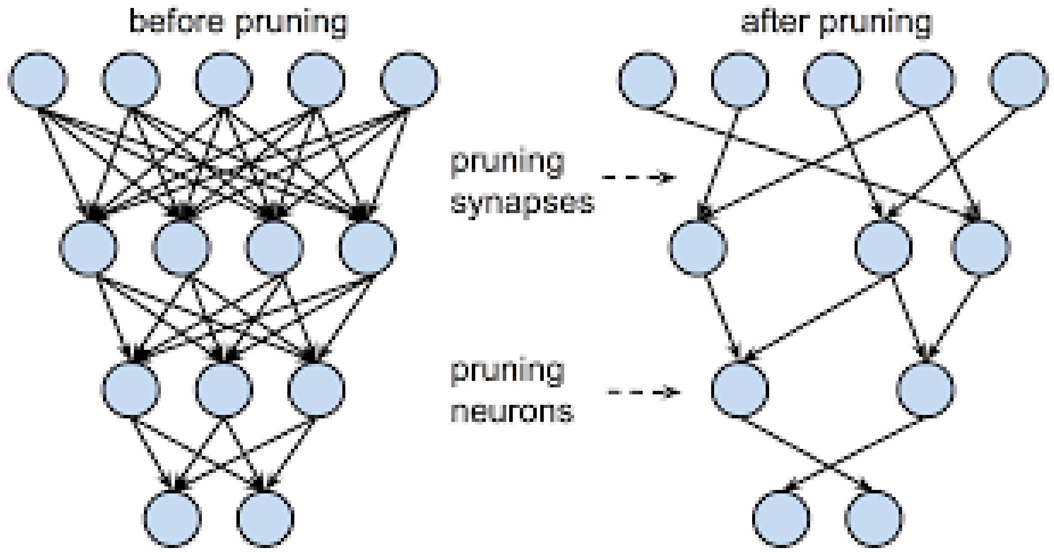
- What's Deep Learning?
- CNN: Convolutional Neural Network
  - Learning
- Other Network M
- **Optimizations**
  - **Pruning**
  - **Quantization**
  - **Data reuse**
- Architectures



Typical energy breakdown of a CNN

Yang.e.a CVPR'17

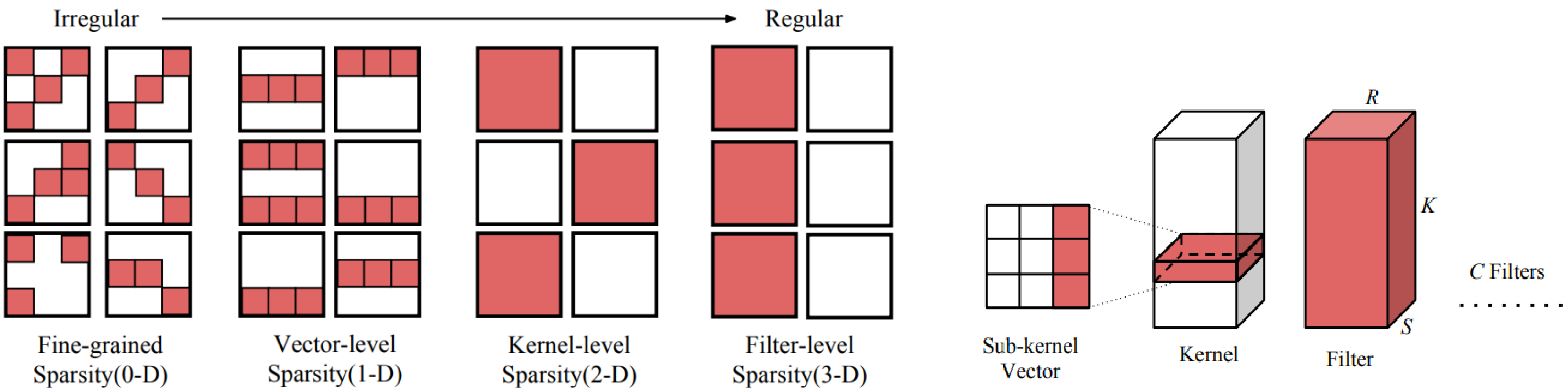
Pruning: Reduce Network



- Fine grain (irregular) pruning
  - removing connections with small weigths
  - needs special HW for efficiency
- Coarse grain (regular) pruning
  - remove kernel (2D): i.e., skip an input feature map for a certain filter
  - remove complete filter (3D), i.e. reduce nr. of output feature maps
- Structured pruning:
  - try to keep e.g. SIMD regularity (vector computing)
  - decompose filters
  - depth wise convolution
  - N:M type of pruning

From Fine to Coarse-Grained Pruning

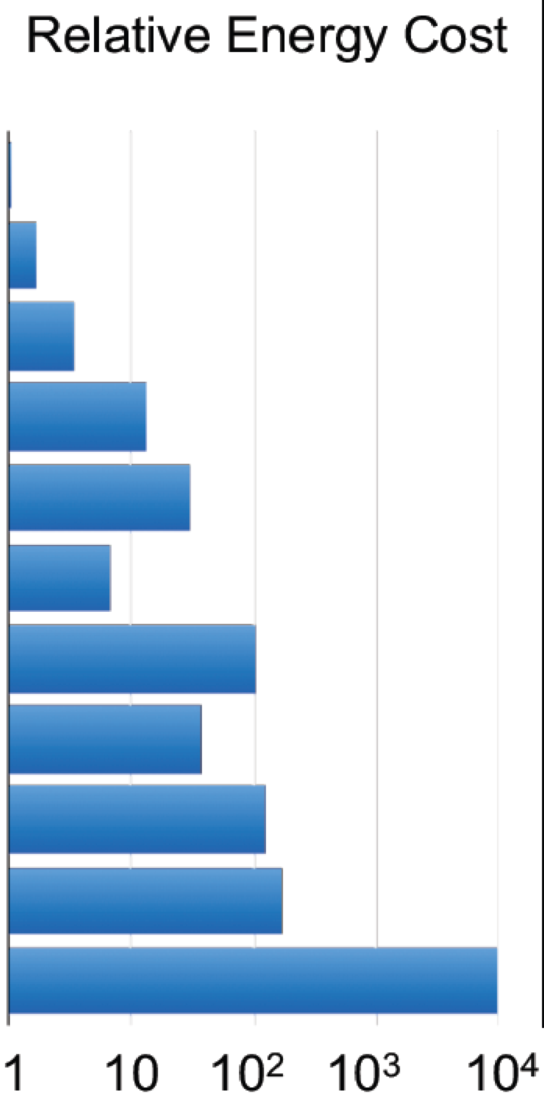
- Prune to match the underlying data-parallel hardware
  - E.g. prune by eliminating entire filter planes

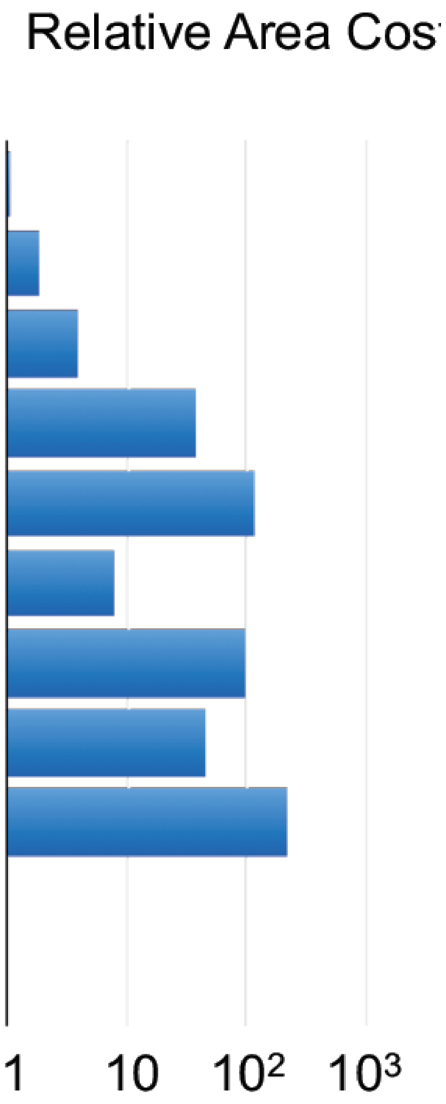


H. Mao et al. "Exploring the regularity of sparse structure in ConvNets" (CVPR 2017)

Quantization: why?

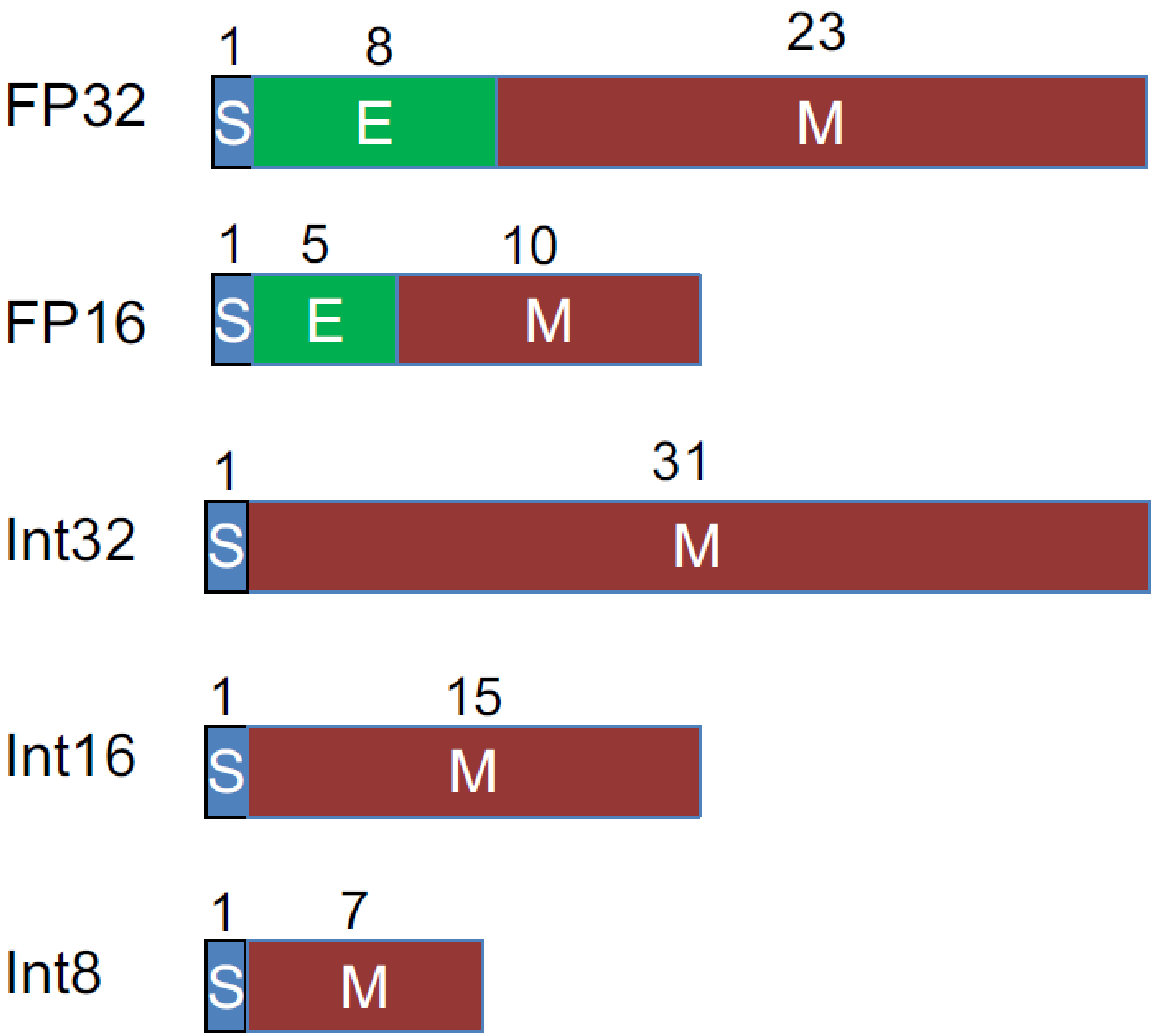
Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A





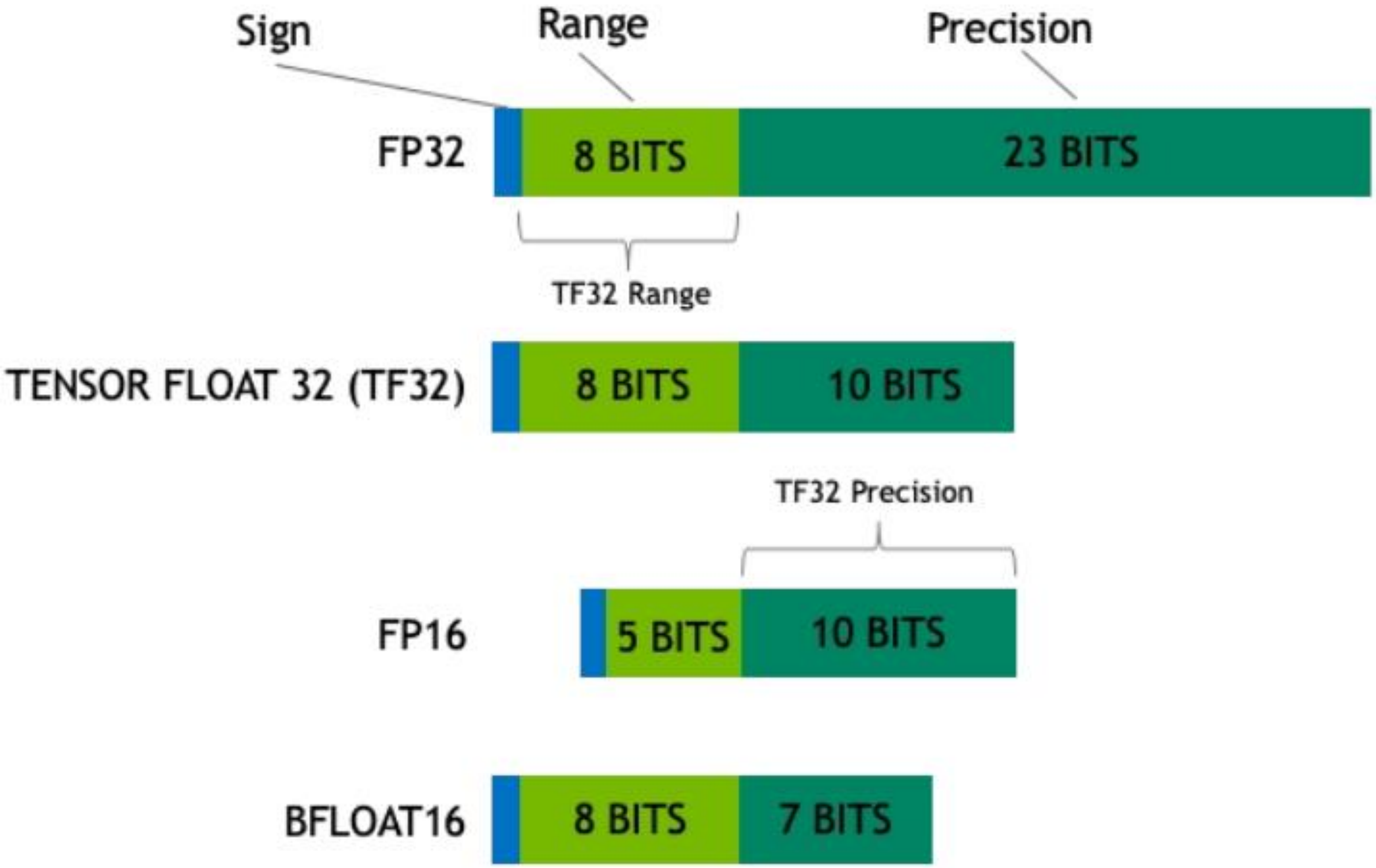
[Horowitz, "Computing's Energy Problem (and what we can do about it)", ISSCC 2014]

Number representations



	Range	Accuracy
FP32	$10^{-38} – 10^{38}$	.000006%
FP16	$6x10^{-5}$ - $6x10^{4}$	.05%
Int32	$0 – 2x10^{9}$	½
Int16	$0 – 6x10^{4}$	½
Int8	0 – 127	½

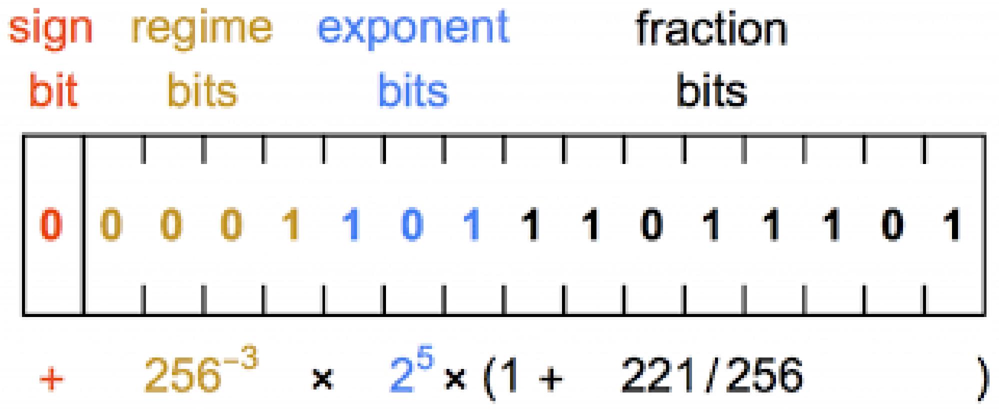
Newer floating point formats



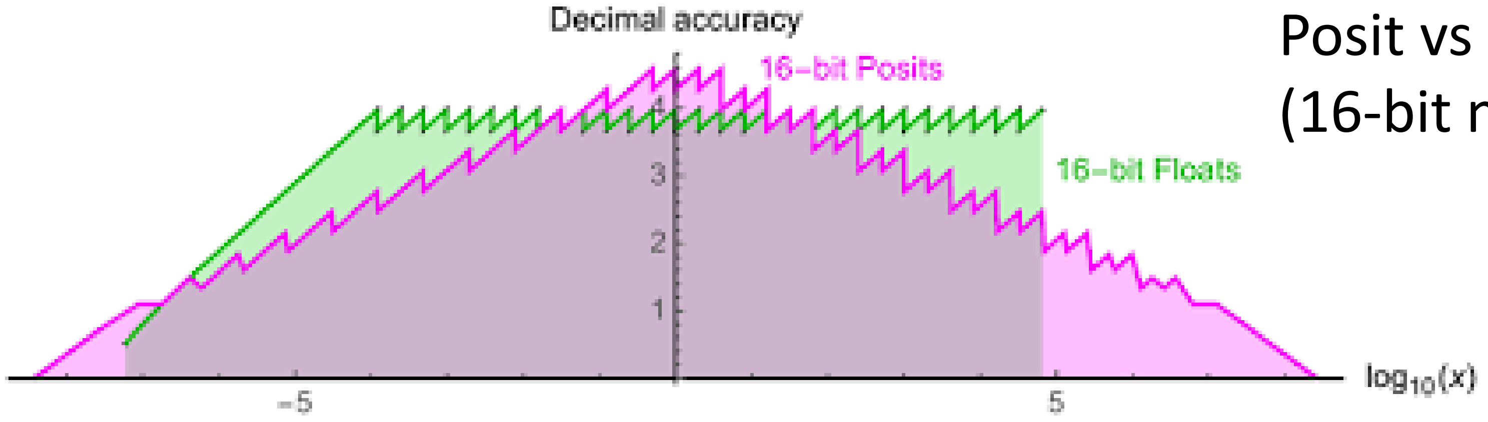
sign	exponent				significand		
0	0	0	0	0	0	0	0

8 bit, mini float

Going beyond traditional floating point: Posit



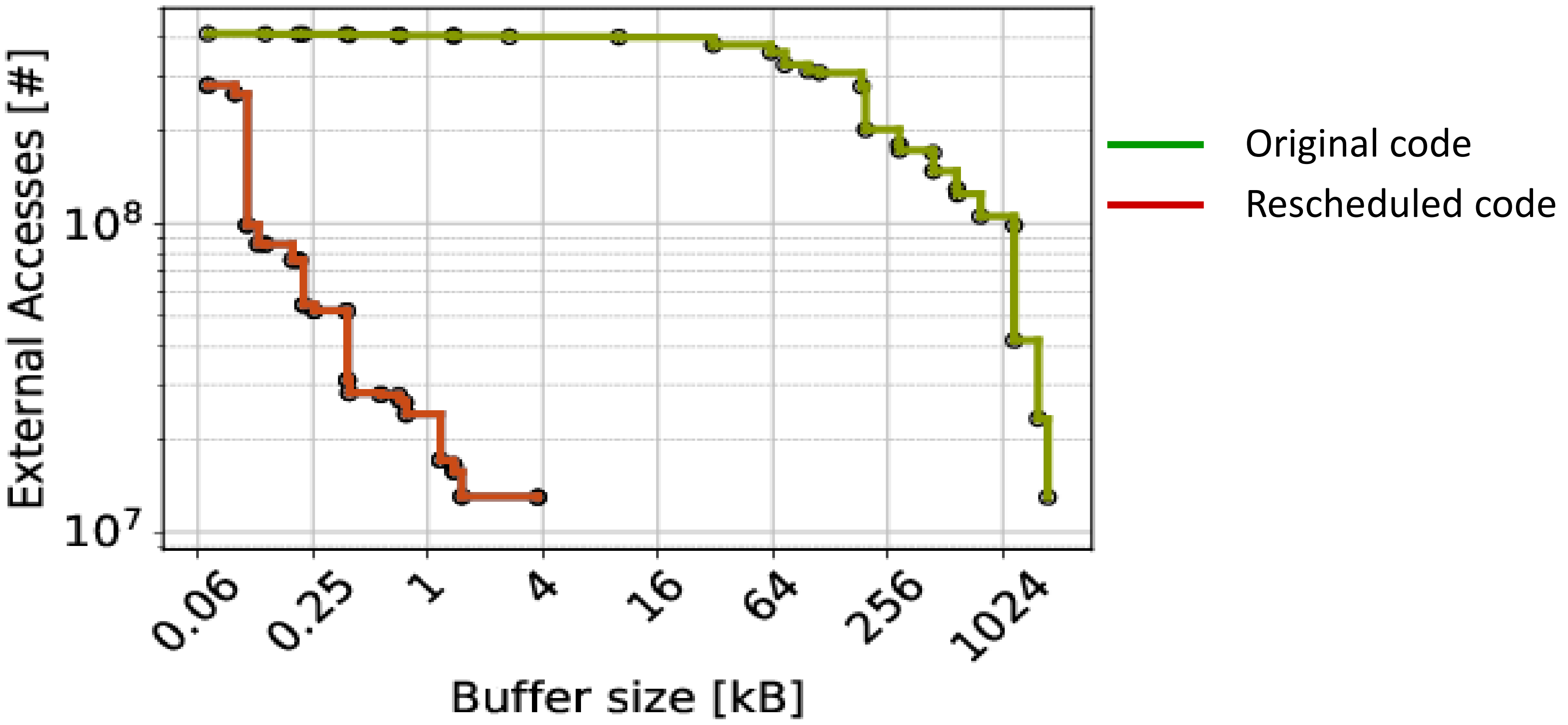
Posit (16 bits in this example)



Posit vs Floating point precision (16-bit numbers)

Reducing external memory accesses

VGG16 example:

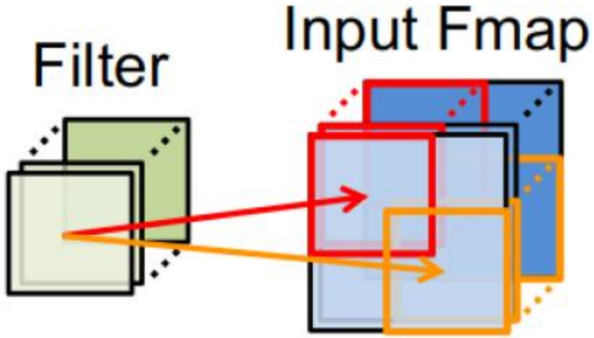


Conclusion: we need advanced loop transformation to exploit data locality (in local buffers), reducing external accesses

Types of reuse

Convolutional Reuse

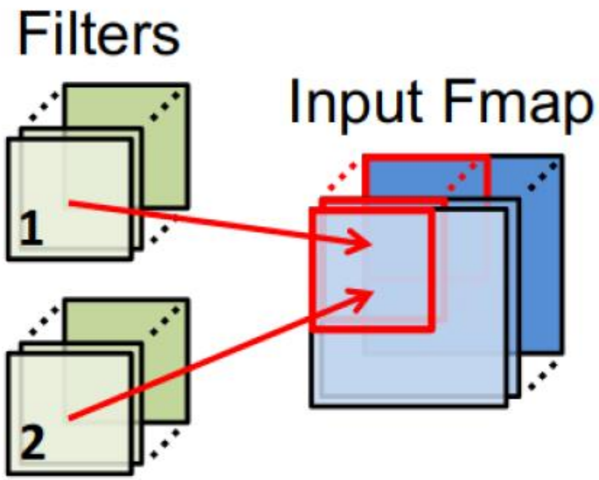
CONV layers only
(sliding window)



Reuse: Activations
Filter weights

Fmap Reuse

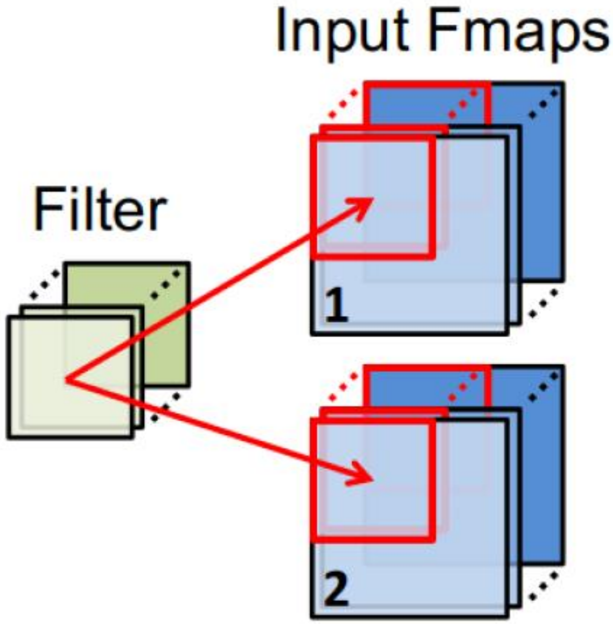
CONV and FC layers



Reuse: Activations

Filter Reuse

CONV and FC layers
(batch size > 1)



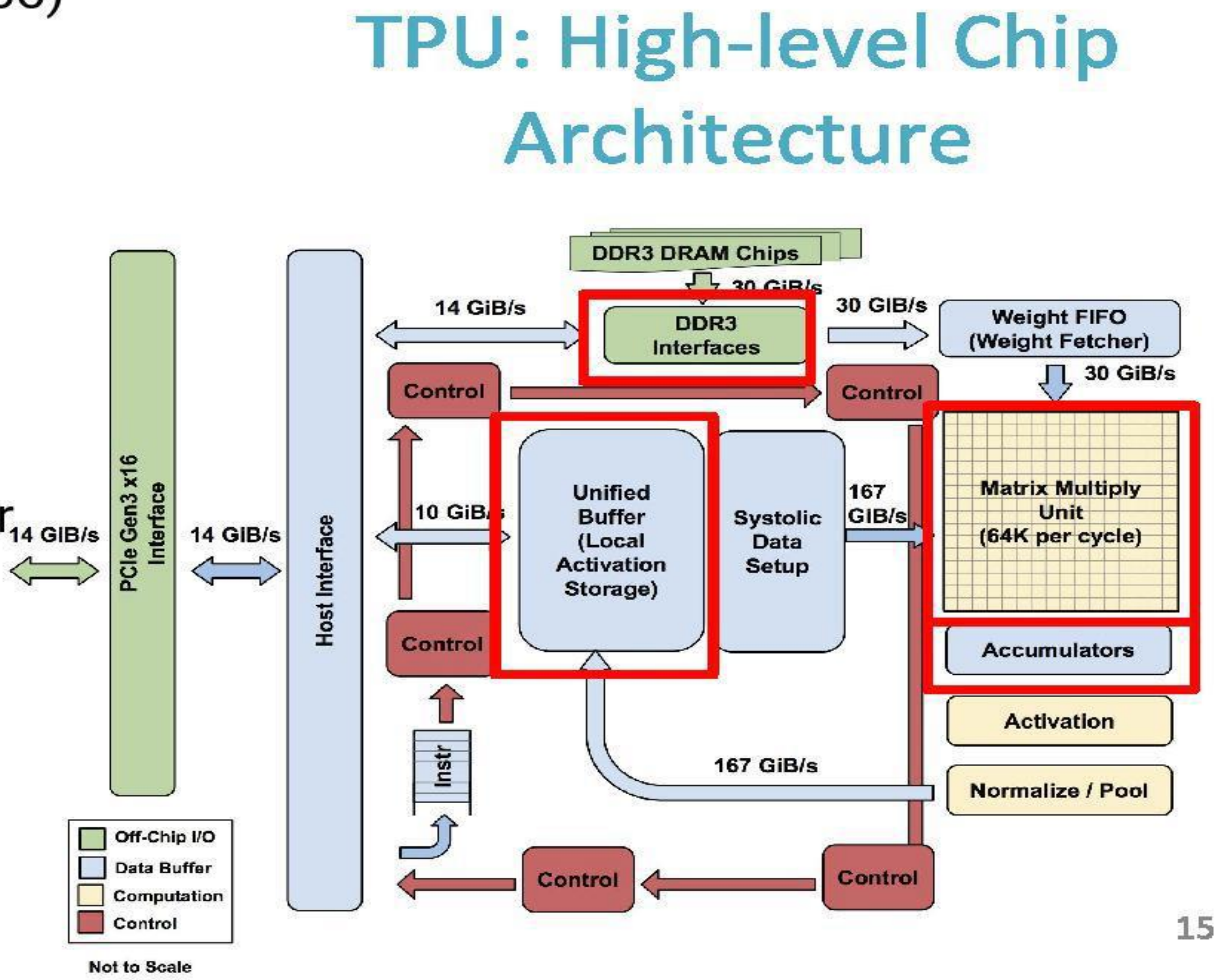
Reuse: Filter weights

Once over lightly

- What's Deep Learning?
- CNN: Convolutional Neural Network
  - Learning
- Other Network Models
- Optimizations
  - Pruning
  - Quantization
  - Data reuse
- **Architecture examples**

Accelerators: Google's TPU v1 (2016)

- 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

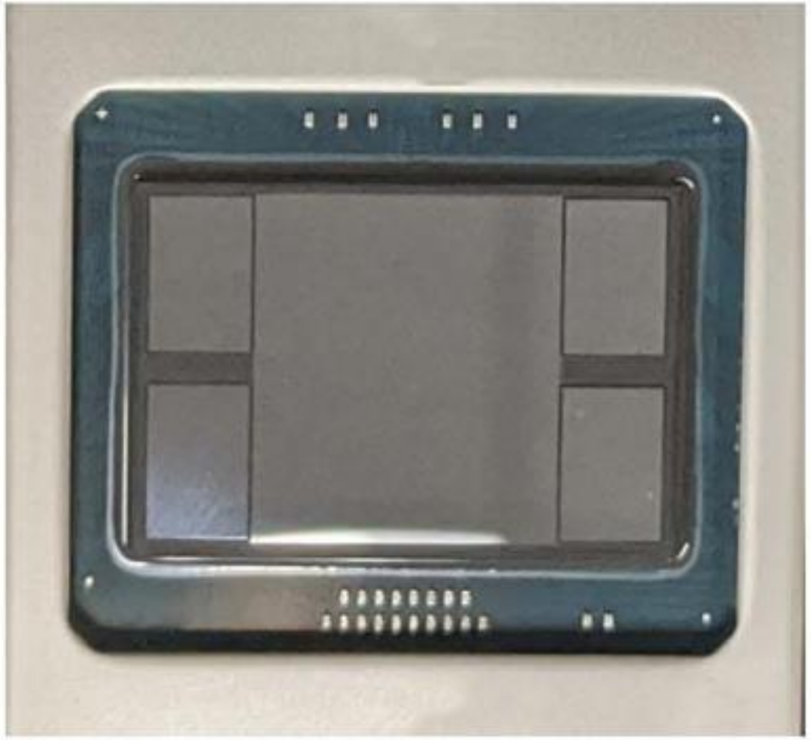


TPU v4 (2021)

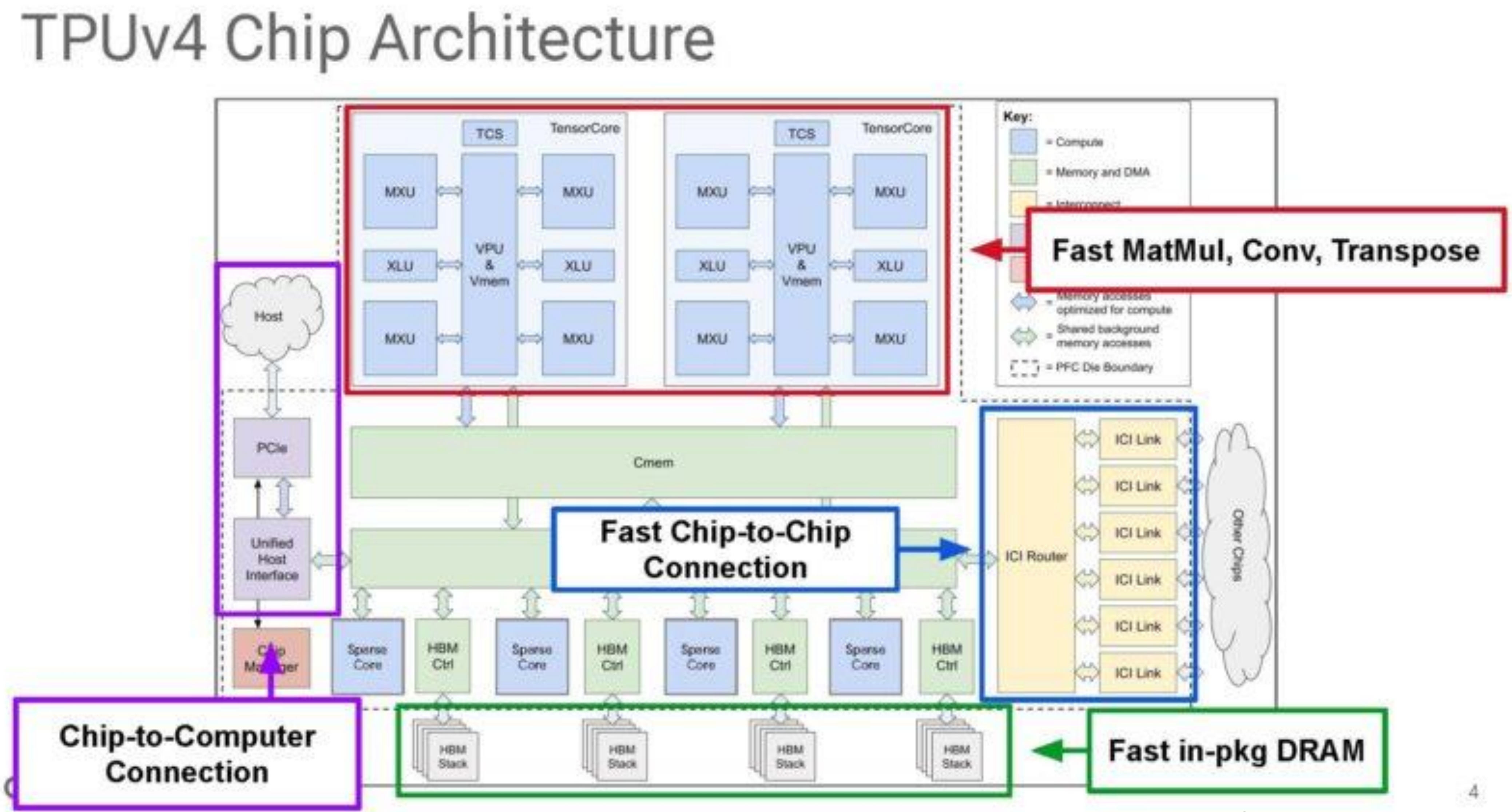
- 7nm process
- Optimized for training, superset of TPU v4i:
  - Two TPUv4i Tensorcores
  - 2X HBM of TPUv4i
  - 3D vs. 2D torus
- 275 peak TFLOPS
  - BF16 with FP32 accumulation
  - Also supports int8 like TPUv4i
- Typical power ~200W
  - TDP is higher to guarantee SLOs and prevent throttling
  - Peak power and water cooling are cheaper than SLO violations

200W/275 TFlops

= 0.73 pJ/flop

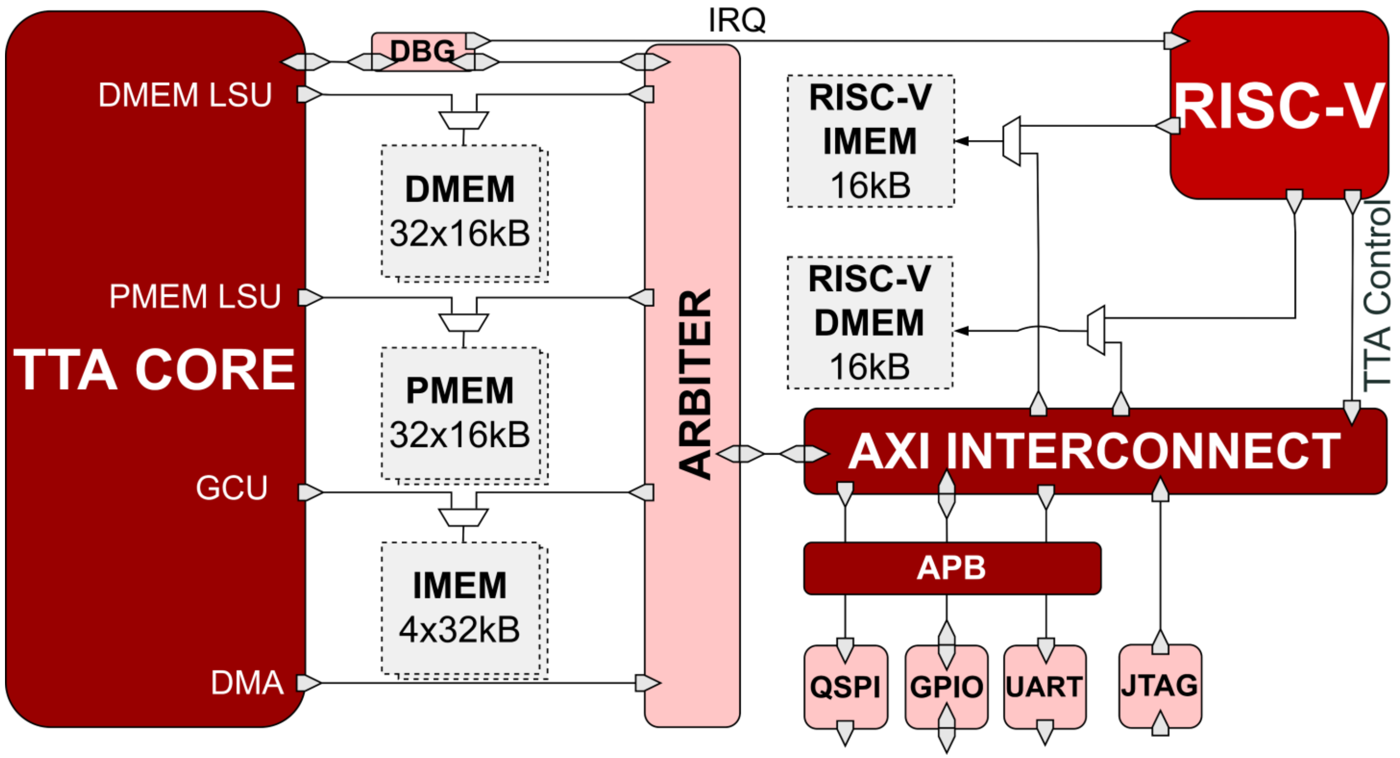


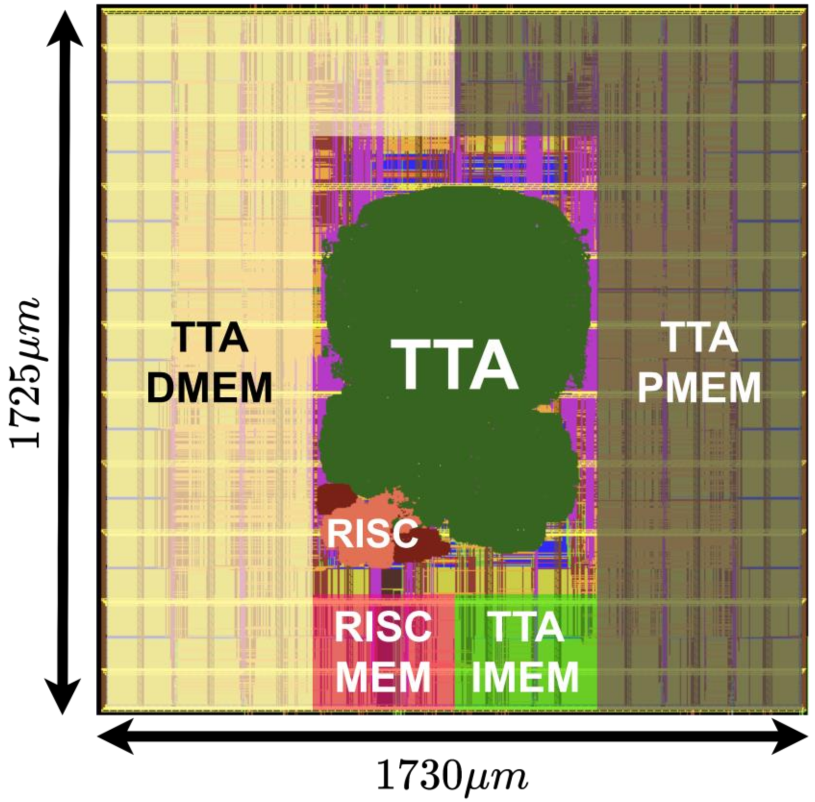
TPU v4 architecture



(see servethehome.com)

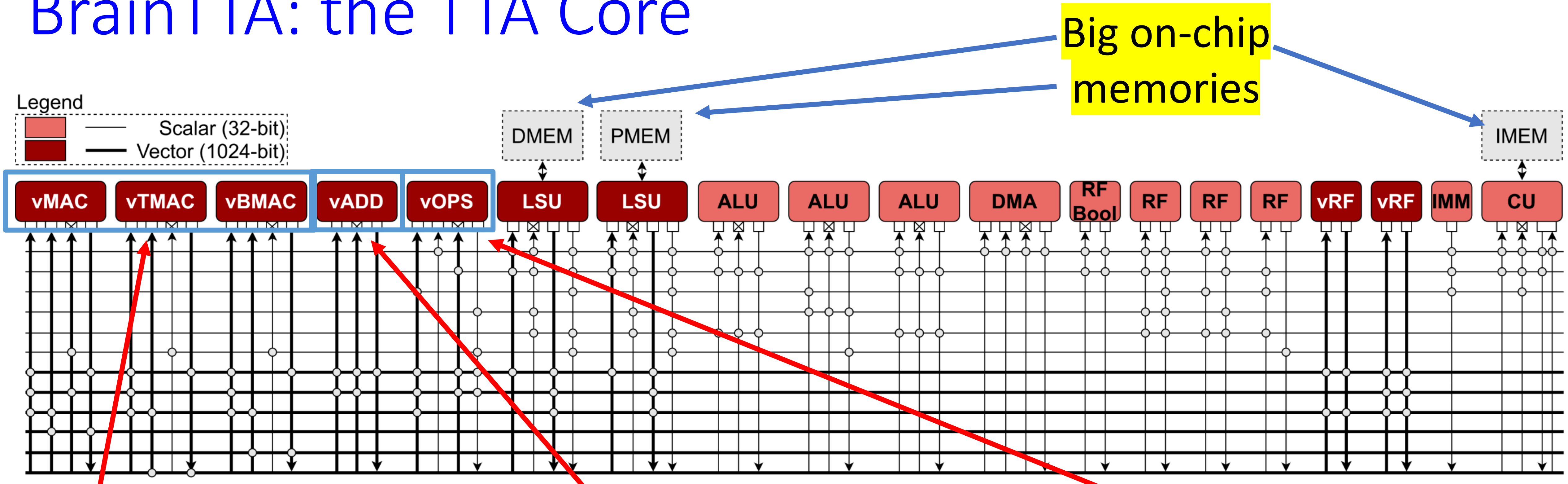
BrainTTA (TUE): System-on-Chip for Deep Learning





- Technology: 22 nm
- RISC-V + peripherals
- Split Data and Parameter memories (DMEM/PMEM) with banked access
- IMEM: Instruction memory

BrainTTA: the TTA Core



3 vMACs:

1. 8-bit MAC
   - Scalar-Vector MAC
   - Vector-Vector MAC
2. Binary MAC
3. Ternary MAC

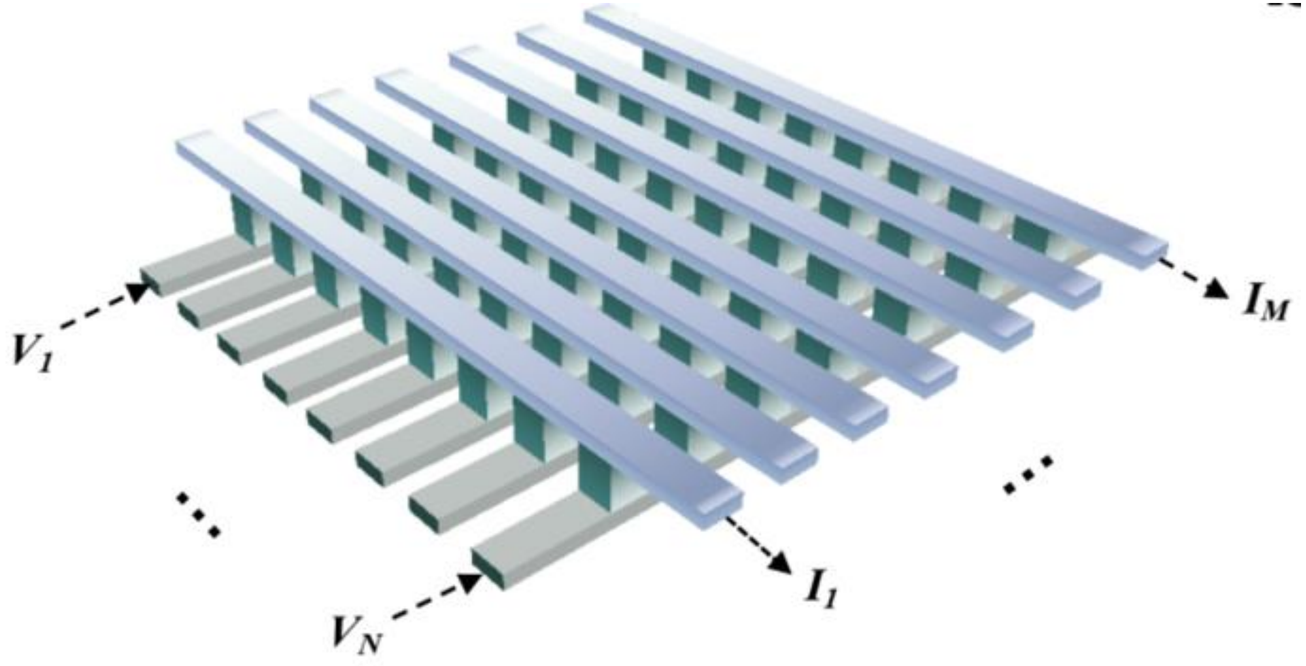
vADD:

- Vector-Vector addition
- Residual support

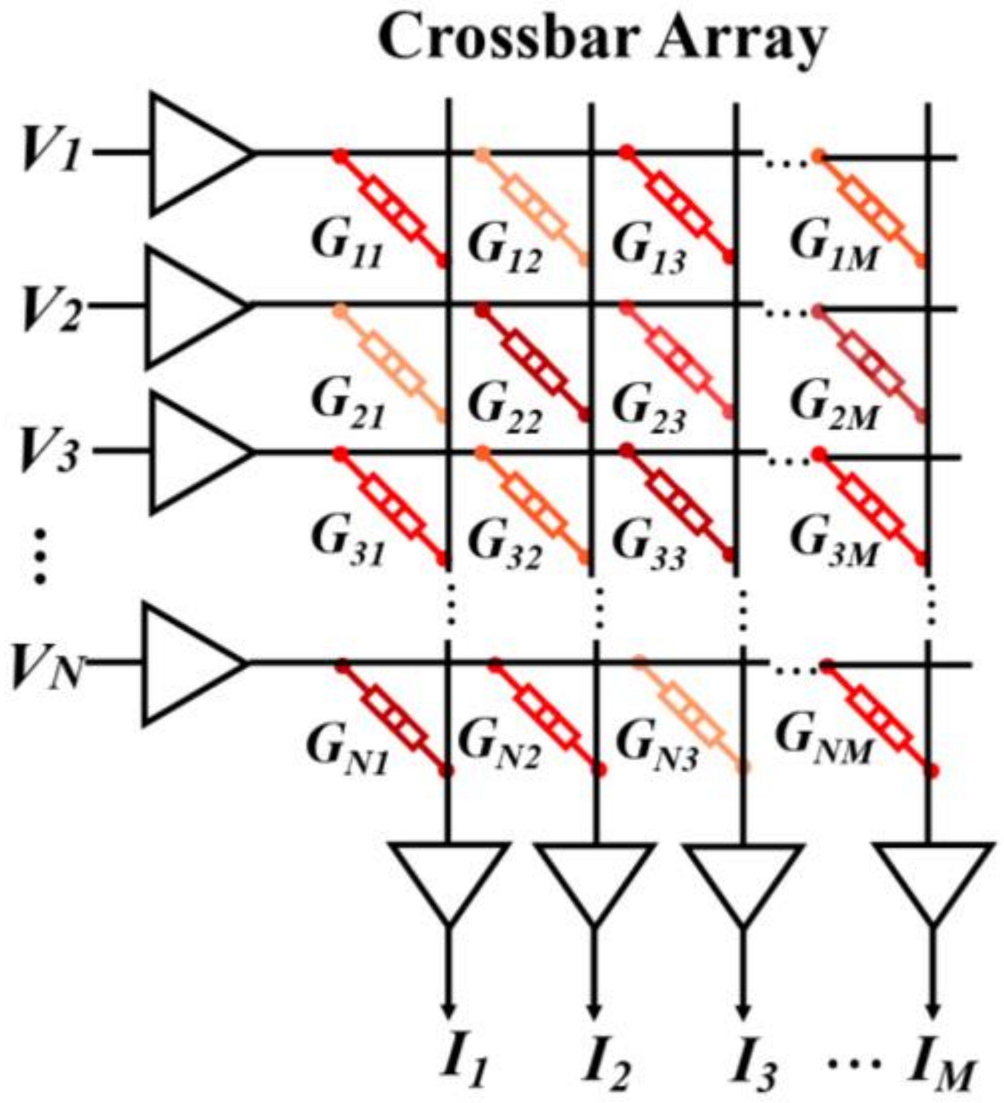
vOPS:

- Requantization
- Binarization
- Ternarization
- MaxPool
- Auxiliary ops

Really solving the memory bottleneck: **Computing in Memory (CIM)**



$$I_M = \sum_N G_{MN} V_N$$



Conclusion

- Deep learning is an extremely fast moving field
- All big players (Big five) invest Billions of Euros
- Efficiency is a big problem, especially when moving AI to the Edge
- Plenty of research opportunities

Enjoy this week