Artificial Neural Networks and RL : Lecture 14

Wulfram Gerstner
EPFL, Lausanne, Switzerland

from brain-computing to neuromorphic computing

Objectives for today:

- local learning rules for hardware
- Spiking Neural Networks (SNN)
- neuromorphic chips
- reducing energy consumption

Recent Development at IBM and INTEL:

Chip companies invest in neuromorphic

Potential reduction of energy consumption with SNN and local learning rules (three-factor rules)

Background reading:

IBM research lab

Bert Offrein et al., 2020, Prospects for photonic implementations of neuromorphic devices and systems, *IEEE Xplore,* *https://ieeexplore.ieee.org/abstract/document/9371915*

LOIHI Chip (intel)
https://en.wikichip.org/wiki/intel/loihi
https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf

Artificial Neural Networks and RL : Lecture 14

Wulfram Gerstner
EPFL, Lausanne, Switzerland

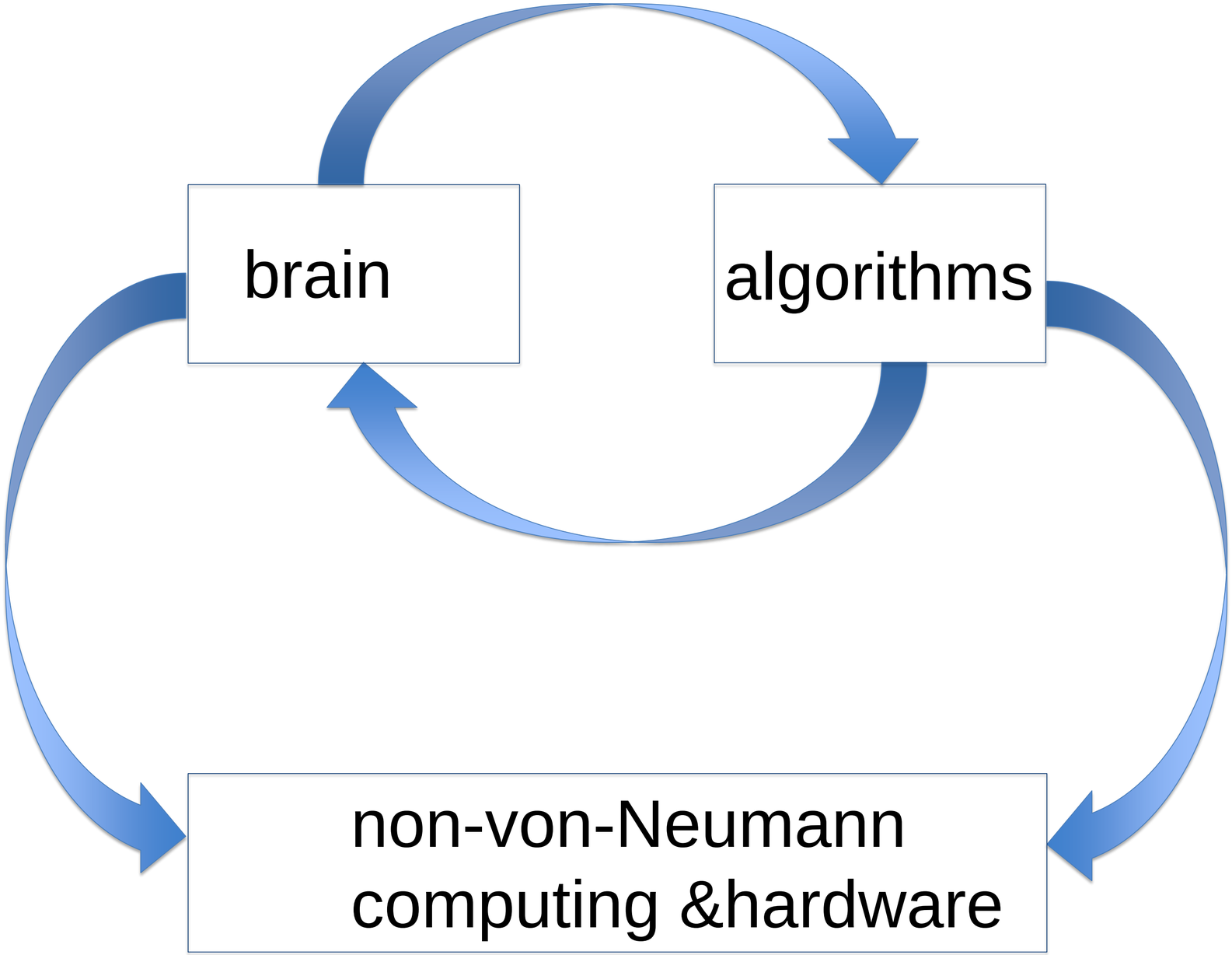
from brain-computing to neuromorphic computing

1. Review: Local Learning rules

Previous slide:

This final lecture is an outlook onto current developments for specialized, bio-inspired chips that will eventually use much less energy than conventional chips. The class of chips is often called neuromorphic chips since they take inspiration from biological principles in neuroscience.
In particular, they use communication with spiking neurons and local learning rules.

Learning Rules



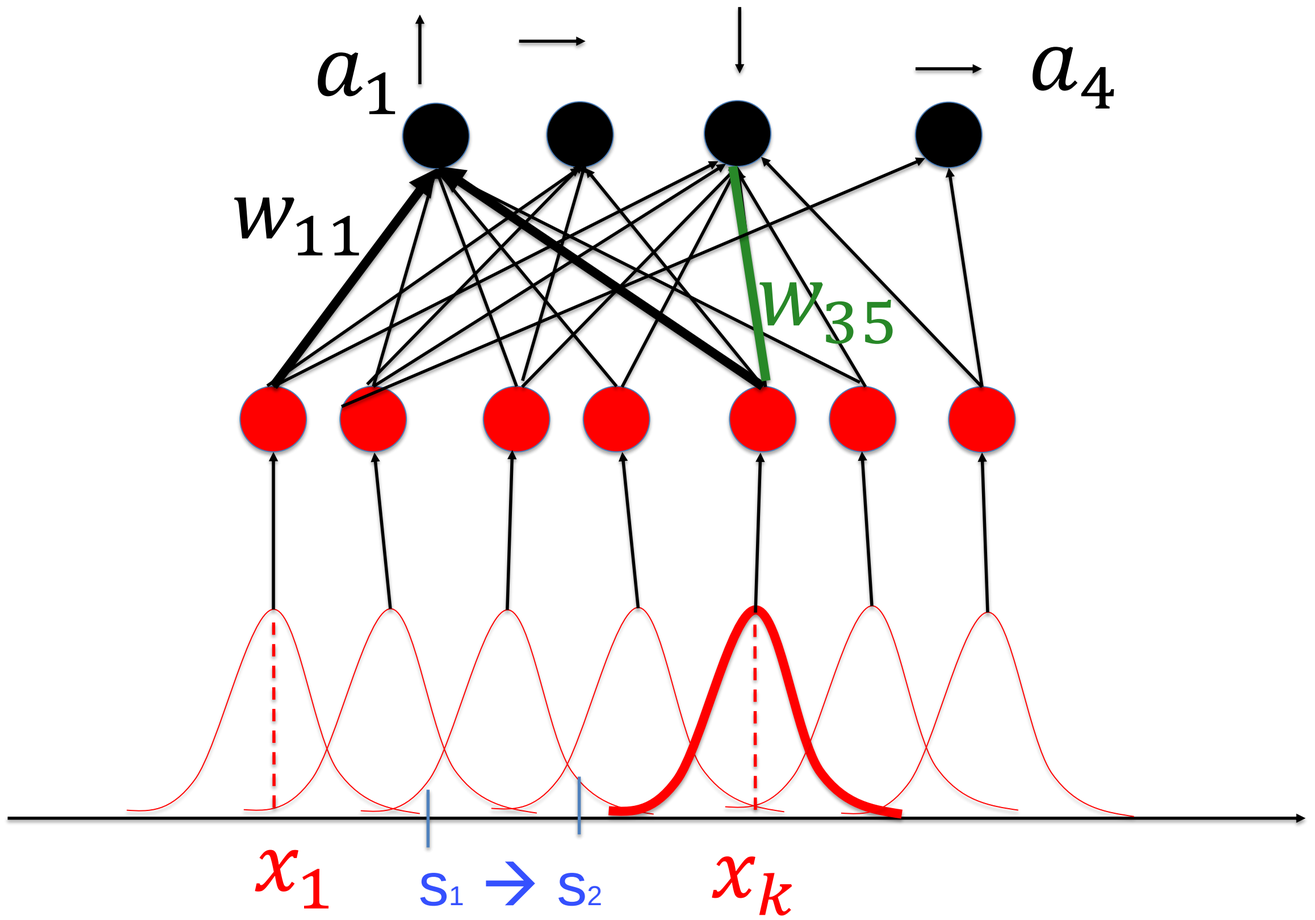
Previous slide:
The lecture last week covered the relation between learning rules used by the brain and those implemented in modern reinforcement learning algorithms.
This lecture will make the link to recent developments in modern neuromorphic computing architectures that are completely different than the class model of von-Neumann computing architectures.

One aspect is that these hardware approaches show potential advantage of Spiking Neural Networks.
Another aspect is that the rely on local learning rules, in particular three-factor rules.
A third aspect is that they could potentially reduce energy consumption

Review: Policy gradient rule with softmax and 1-hot coding

$North$: $\boldsymbol{a_1}$=**1**; a_2=a_3 =a_4 = 0

$East$: a_1 =a_2=a_3 =0; $\boldsymbol{a_4}$ = **1**



Discrete actions with
1 hot coding

If at time t, the action
$a_i^t = 1$ is chosen then
$a_j^t = 0$ for all other
output neurons $j \neq i$

Action choice:
Softmax policy

Previous slide:

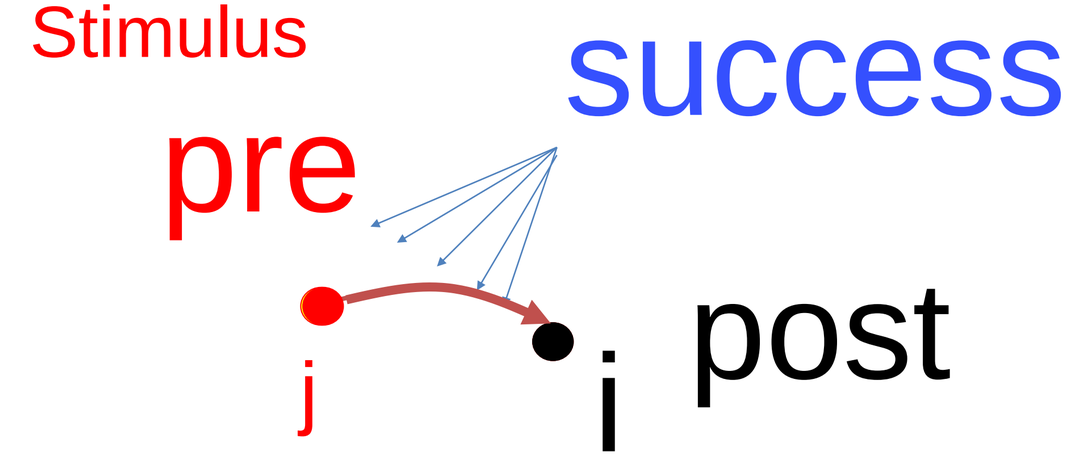
Last week we considered reinforcement learning in a simple network where the state is represented by some basis functions and reinforcement learning is restricted to the final layer of action choices.

The action choices are based one one-hot coding:
If the output symbol for action i is +1 ($\boldsymbol{a_1=1}$) then the symbol for all other actions j not equal i is zero ($\boldsymbol{a_j=0}$) .

In our example network, the policy of action choices is a stochastic softmax policy.

Review: Policy Gradient as three-factor rule

parameter = weight w_{ij}



Change depends on pre and post

Three factors: **success** **post** **pre**

$$\Delta w_{ij} = \eta \; S(a_i^t, \vec{x}) \overbrace{[\, a_i^t - \langle a_i(\vec{x}) \rangle]}^{\text{post}} x_j$$

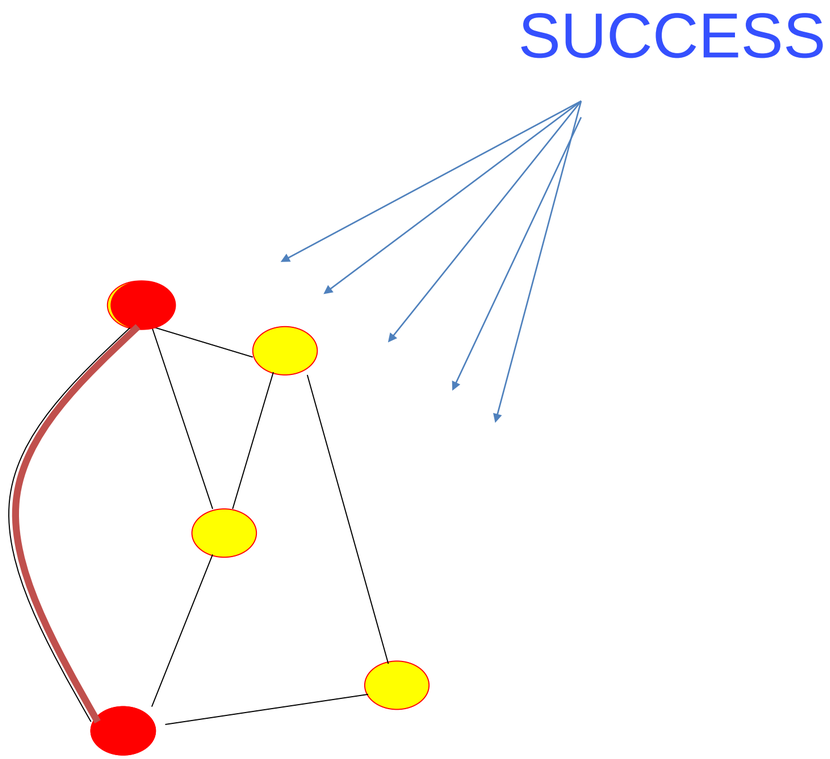
postsynaptic factor is

'activity – expected activity'

Previous slide:
For this example we have found that the update of weights in the output layer can be written as a three-factor learning rule:

- The presynaptic factor and the postsynaptic factor define the connection weight that is updated (the two local factors.
- The third factor is global (independent of neuronal indices) and signals success.
- Success can be reward or the TD error (for actor-critic).

Review: Local Rules for Reinforcement Learning



Reinforcement Learning = reward + Hebb

$$\Delta w_{ij} \propto F(pre, post, SUCCESS)$$

local global

broadly diffused signal:
neuromodulator
(e.g., dopamine)

Previous slide:
The traditional Hebb rule has the same two local factors as the three-factor rules.
Thus the three-factor rule can be seen as a Hebbian rule modulated by the global success factor.

These notions will be important for the discussion of modern neuromorphic hardware later on.

Review: Three-factor STDP for reward-based learning

$$\frac{dw_{ij}}{dt} = F(w_{ij}; \mathrm{PRE}_j\ , \mathrm{POST}_i\ , 3rd)$$

Eligibility trace:

$$\tau \frac{d}{dt} e_{ij} = \mathrm{HEBB}_{ij} - e_{ij}$$

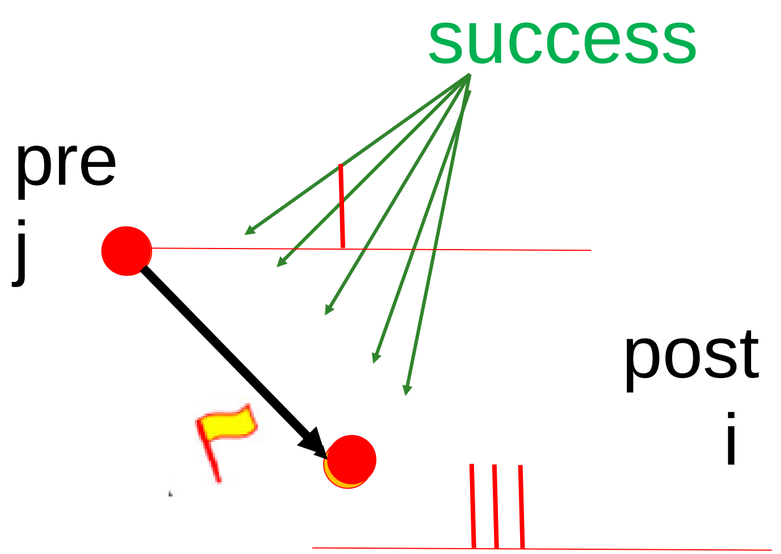
Weight

$$\frac{d}{dt} w_{ij} = e_{ij} \cdot S(t)$$

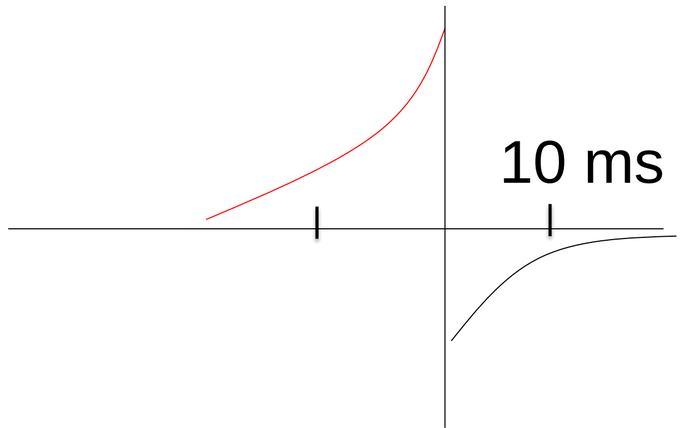
Success signal

Hebb rule/eligibility trace

Success signal:
TD error



STDP window



Xie and Seung 2003, Izhikevich, 2007; Florian, 2007; Legenstein et al., 2008, Fremaux et al. 2010, 2013

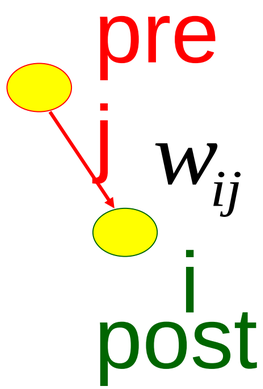
Previous slide:

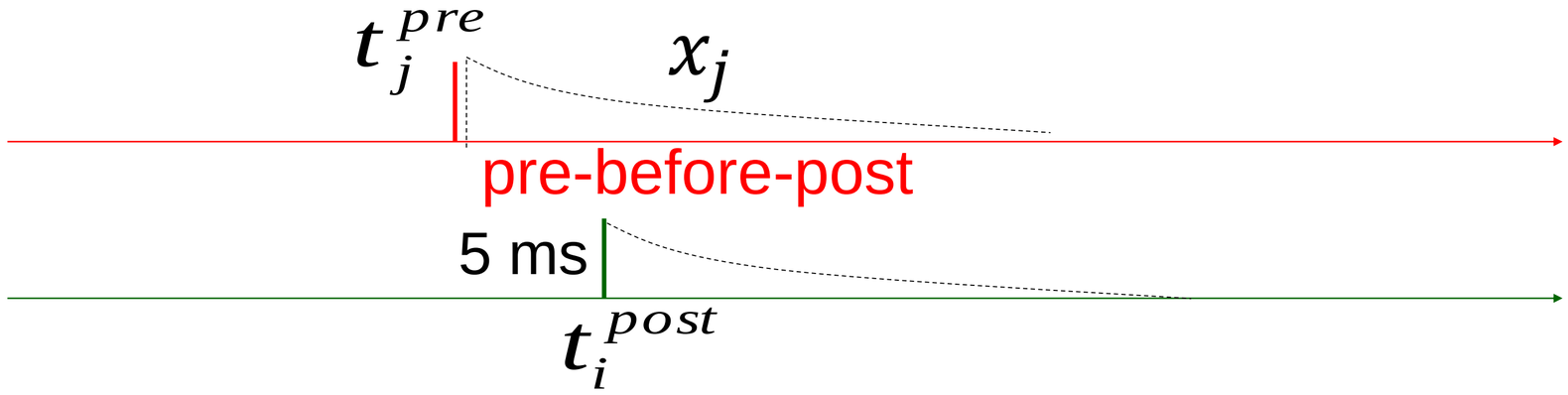
A specific biologically plausible three-factor rule with eligibility traces would be the following:

- Spike-Timing-Dependent Plasticity (STDP) picks up coincidences between pre and postsynaptic spikes on a time scale of 10 milliseconds. STDP is hence a spike-based version of Hebbian learning.

- If furthermore the success signal arrives within one second, then the weight is updated.

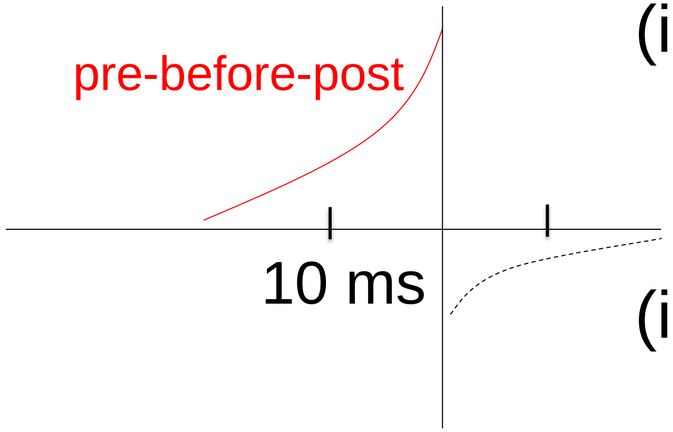
‘traces’ for STDP: how to implement Hebb with spikes





Simple STDP model
(Gerstner et al. 1996, Song-Miller-Abbott 2000, etc)

STDP window



(i) Trace left by presynaptic spike (discrete time steps of 1ms):

$$\Delta x_j = 1 \quad if \quad t = t_j^{pre}$$

$$x_j \leftarrow \lambda_+ \, x_j$$

(ii) Update of eligibility trace at moment of postsynaptic spike

$$\Delta e_{ij} = x_j \;\; if \quad t = t_i^{post}$$

$$e_{ij} \leftarrow \lambda \quad e_{ij}$$

(iii) Update of weights prop to eligibility trace and Success S

$$\Delta w_{ij} = e_{ij} \, S$$

Previous slide:

$$\tau_+ = 1/\lambda_+$$

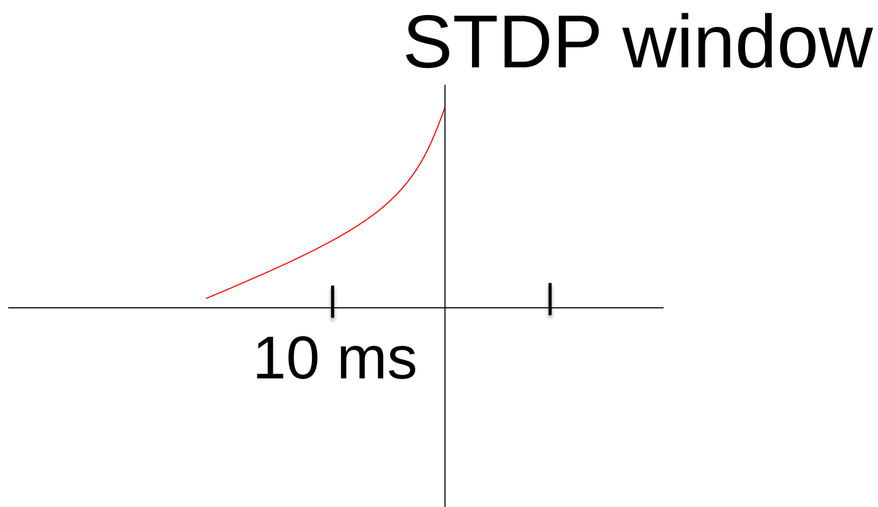
- For example, the **pre-before-post 'HEBB' condition** can be implemented by saying that the presynaptic spike leaves an **exponential trace** (decaying with a time constant $\tau_+ = 1/\lambda_+$ of 10 millisecond); if the postsynaptic spike arrives a few milliseconds afterwards, it sets an eligibility trace that is proportional to the value of the **presynaptic trace.**

- The eligibility trace decays on slower time scale (time scale $\tau=1/\lambda$ = 1 second).

- If the success signal arrives within one second, then the weight is updated.

- We can consider a special case (all time scales are the same discrete time step):
  If (i) the eligibility trace has a time constant of $\tau_+ = 1/\lambda_+$ 10ms
  (ii) the Hebbian STDP window is one-sided with a time scale of 10ms
  (iii) the discrete time step is 10ms,
  then the three-factor STDP is very similar to the three-factor policy gradient rule

- A two-sided STDP window can be implemented by stating that the postsynaptic spike leaves another trace (postsynaptic trace) which leads to a negative updated of the eligibility trace at the moment of the next presynaptic spike arrival.

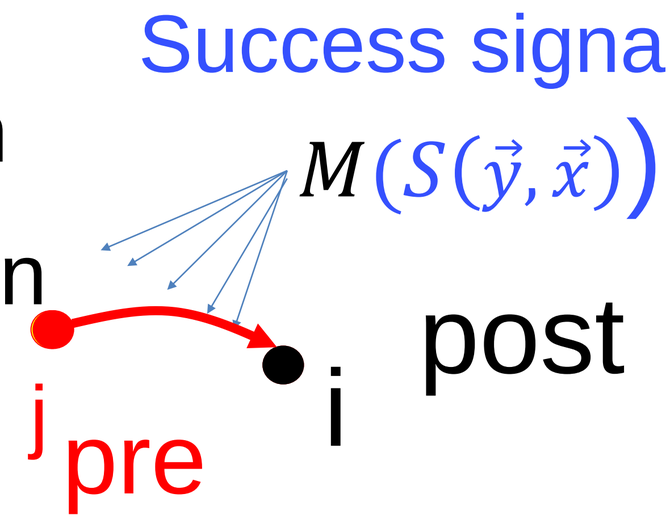


Formalism of Three-factor rules with eligibility trace

Three-factor rule defines a framework

x_j = activity-trace left by of presynaptic neuron

y_i = activity-trace left by of postsynaptic neuron



Step 1: co-activation sets eligibility trace

$$\Delta e_{ij} = \eta \; f(y_i)\, g(x_j)$$

Step 2: eligibility trace decays over time

$$e_{ij} \leftarrow \lambda\, e_{ij}$$

Step 3: eligibility trace translated into weight change

$$\Delta w_{ij} = \eta\, M\left(S(\vec{y},\vec{x})\right) e_{ij}$$

Previous slide:

There are many different Hebbian rules or STDP rules. Similarly, there is not a single three-factor rule. Rather three-factor rules are a framework formulated as follows:

- The trace left by presynaptic activity contributes some nonlinear factor $g(x_j)$

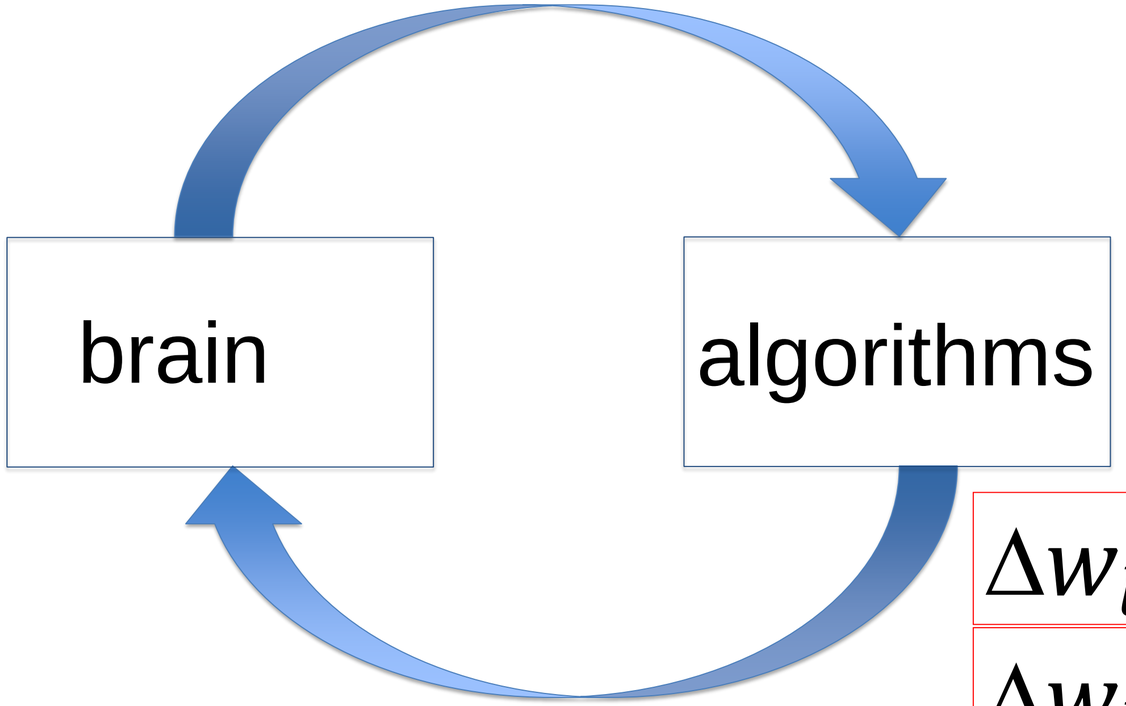
- The trace left by postsynaptic activity contributes some nonlinear factor $f(y_i)$

- The eligibility trace e_{ij} is changed proportional to the two factors f times g

- The eligibility trace decays by a factor λ

- Weights are updates proportional to eligibility trace e_{ij} times M with a modulator M that is a nonlinear function of the success S.

Learning Rules



$$\Delta w_{lk} = \eta \; r_t \; e_{lk}$$

$$\Delta w_{lk} = \eta \; \delta_t \; e_{lk}$$

The learning rule of the (advantage) actor-critic or REINFORCE with eligibility traces are both compatible with three-factor rules

Updates proportional to the reward r or TD error δ_t

Previous slide:

The main difference between REINFORCE with eligibility trace and Advantage-actor-critic is just the way the success signal is defined.

Quiz: Relation of Advantage-Actor-Critic to other Policy Gradient Algos

Assume the transition to state x^{t+1} with a reward of r^{t+1} after taking action a^t at state x^t. The learning rule for the Advantage Actor-Critic with Eligibility traces is

'learning rule' of Advantage Actor-Critic with eligibility trace

$$\delta \leftarrow r^{t+1} + \gamma \hat{v}_w(x^{t+1}) - \hat{v}_w(x^t)$$
$$z^w \leftarrow \lambda^w z^w + \nabla_w \hat{v}_w(x^t)$$
$$z^\theta \leftarrow \lambda^\theta z^\theta + \nabla_\theta \pi_\theta(a^t|x^t)$$
$$w \leftarrow w + \alpha^w z^w \delta$$
$$\theta \leftarrow \theta + \alpha^\theta z^\theta \delta$$

[ ] We get the Advantage Actor-Critic **without eligibility trace** if we set $\lambda^w = \lambda^\theta = 0$.

[ ] We get REINFORCE **without baseline** (with eligibility trace) if set $\delta \leftarrow r^{t+1}$

[ ] We get REINFORCE **without baseline** and **without eligibility trace**

if set $\delta \leftarrow R = r^{t+1} + \gamma r^{t+2} +$

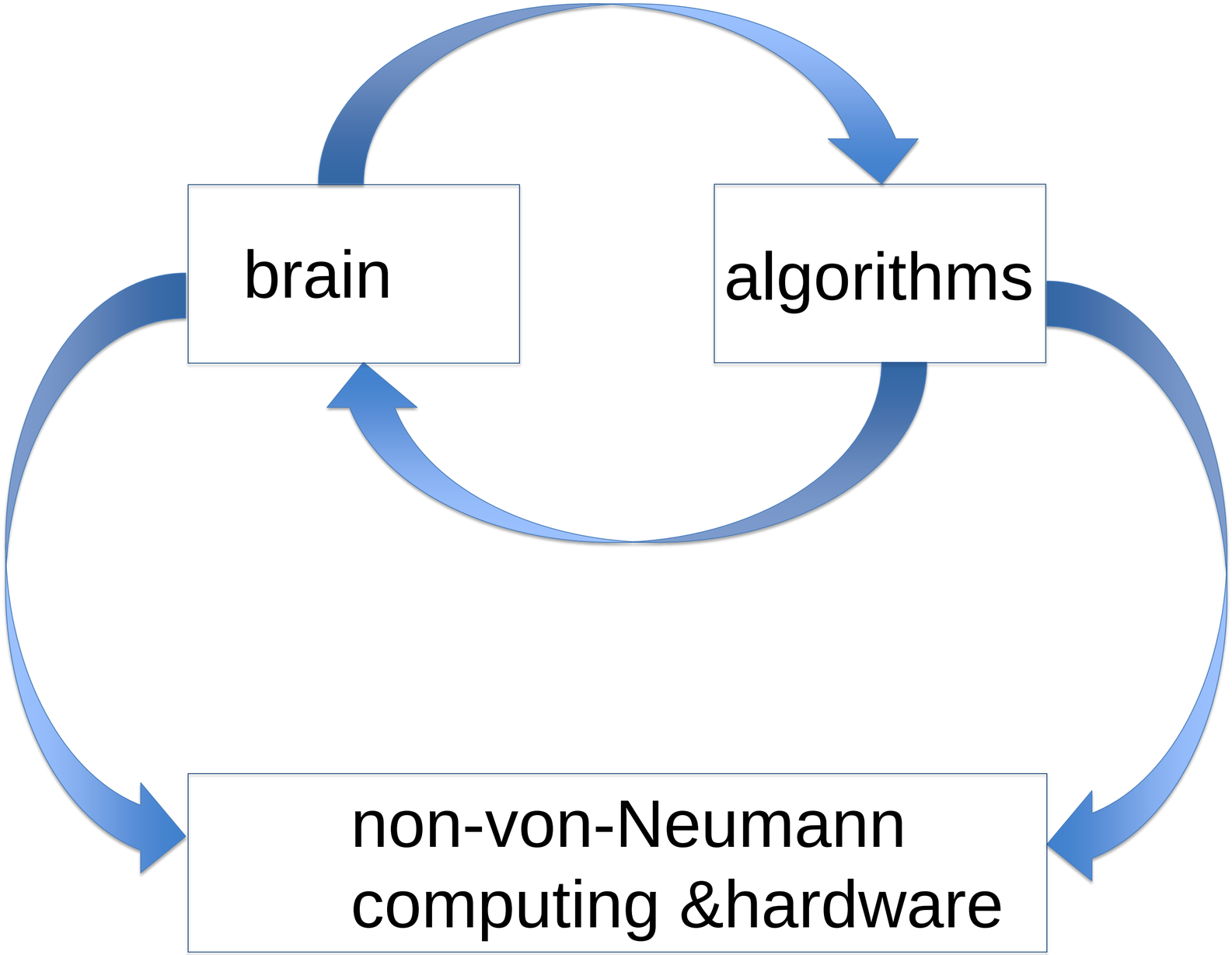
[ ] REINFORCE **without baseline and without eligibility trace** has many terms

propto $\nabla_\theta \pi_\theta(a^t, x^t)$, $\nabla_\theta \pi_\theta(a^{t+1}, x^{t+1})$, … and is therefore not an online algorithm

Previous slide. Your notes. All these algorithms have been covered in the lecture on policy gradient methods.

R denotes the return (sum of discounted rewards, starting from state t)

Learning Rules



Previous slide:

After this review of three-factor rules, we make a small detour before we prepared to look at the first hardware implementation.

Artificial Neural Networks and RL : Lecture 14

Wulfram Gerstner
EPFL, Lausanne, Switzerland

from brain-computing to neuromorphic computing

1. Review: Local Learning rules
2. **Detour: Spiking Neural Networks (SNN)**

Previous slide:

The standard model of Spiking Neural Networks (SNN) is the leaky integrate-and-fire model.

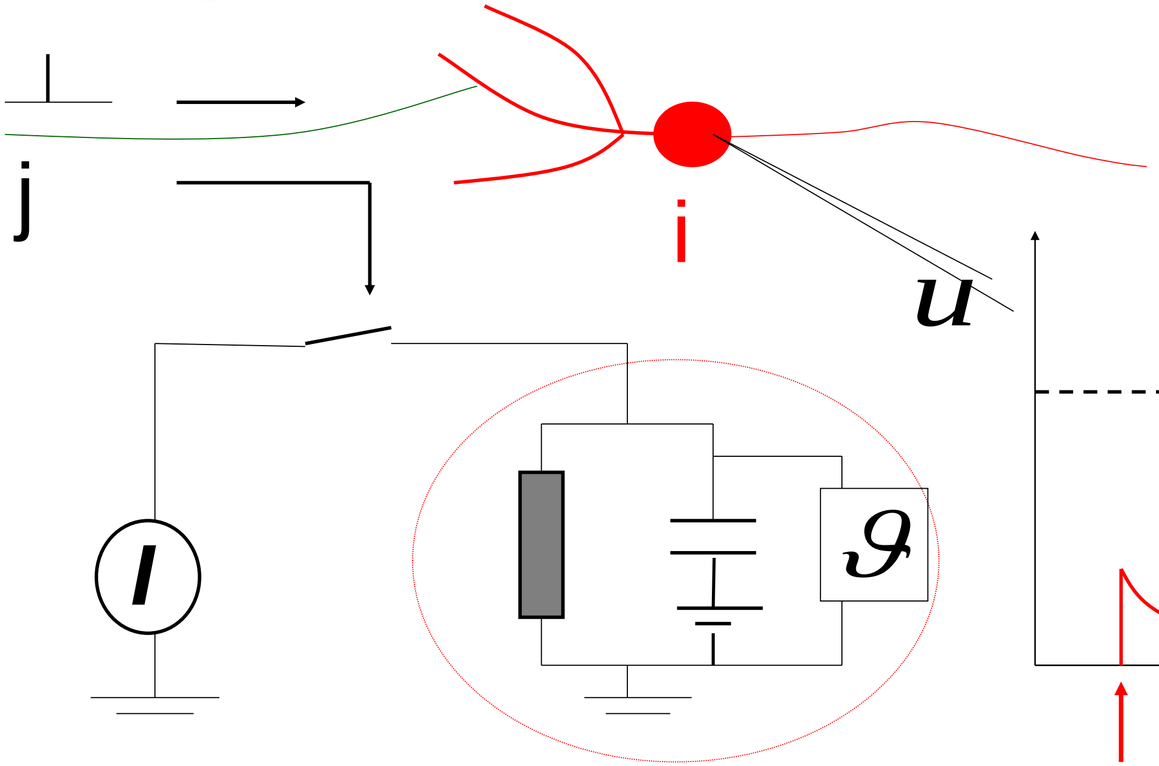
In the community of computational neuroscience it is usually written in continuous time.

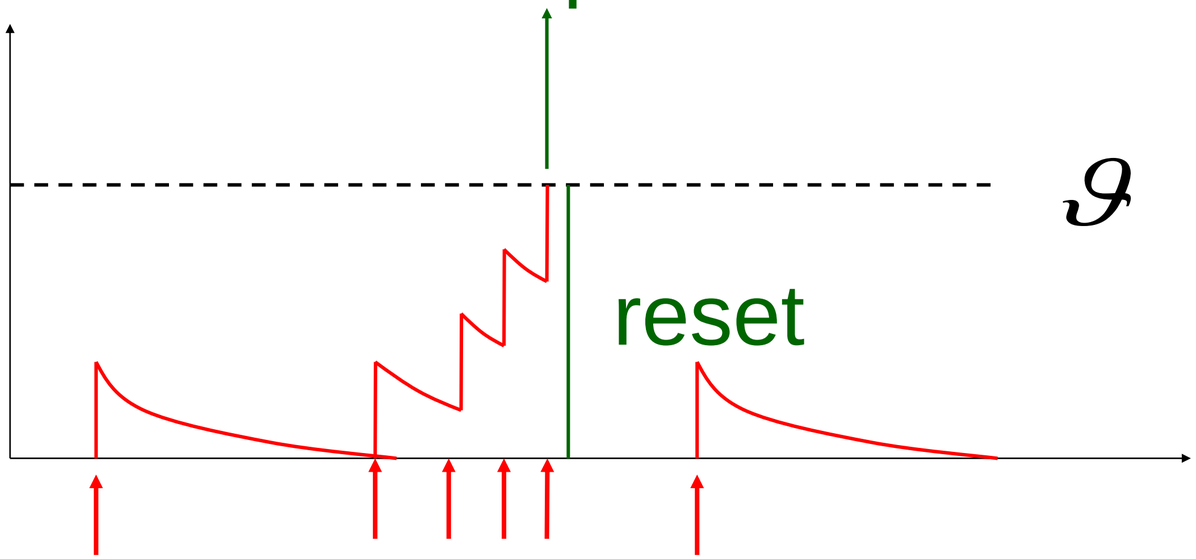
For computer science applications, it is usually written in discrete time.

We show both versions.

Spiking Neural Network – Leaky Integrate-and-Fire Model

(continuous time formulation)





$$\mathrm{I_i(t)} = \sum_j w_{ij}\ \delta(t - t_j^{pre})$$

$$\tau \cdot \frac{d}{dt} u = -(u - u_{rest}) + R\ \mathrm{I_i(t)}$$

linear

$$u(t) = \vartheta \Rightarrow \text{Fire+reset}\ \ u \longrightarrow 0.$$

threshold

Previous slide:

The Leaky integrate-and-fire model written in **continuous time** involves a LINEAR differential equation that can be interpreted as an electrical RC circuit charged by a current I(t). This current I(t) consists of short electrical pulses that present spike arrivals. The $\delta(t - t_j^{pre})$ denotes the Dirac delta function for each presynaptic spike arrival at times t_j^{pre} and w_{ij} are the weights.

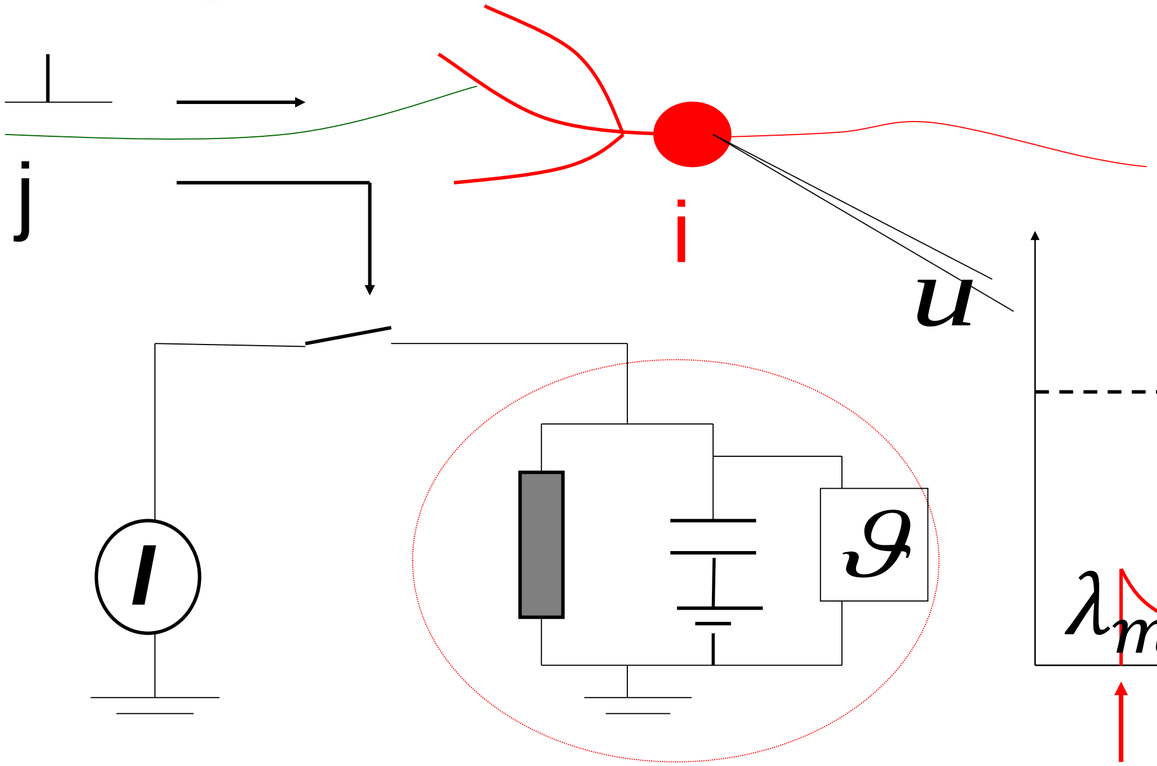
The linear equation is combined with a NONLINEAR FIRE-and-RESET process. If the variable u ('membrane potential of the neuron') reaches the threshold theta, then u is reset to zero.

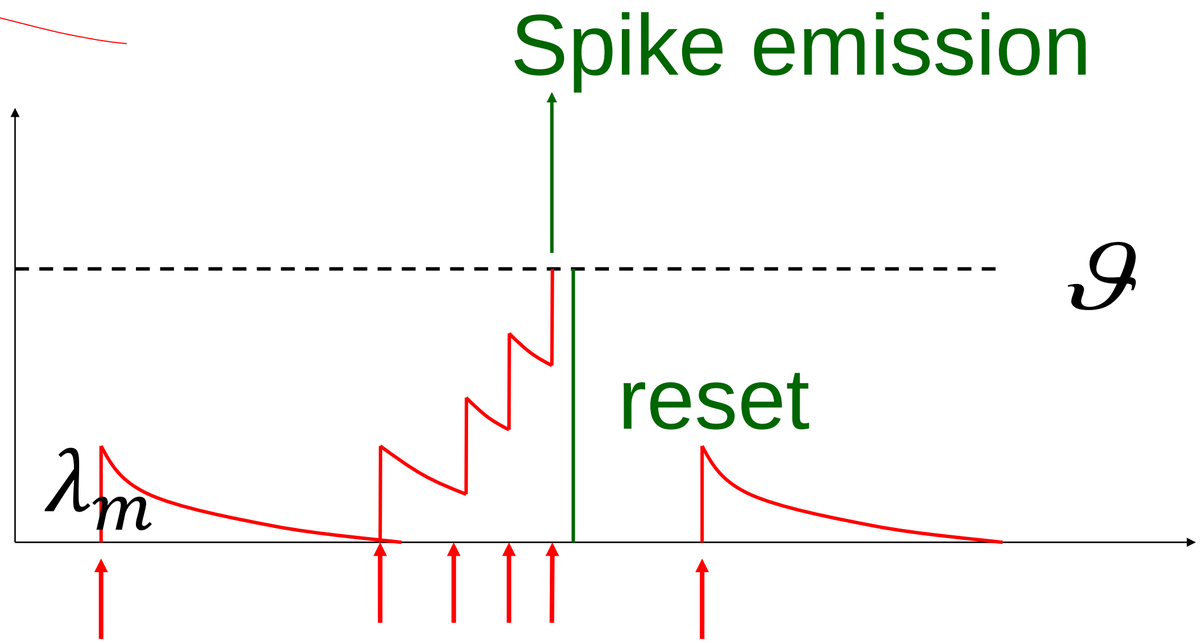
Side Note: An electrical RC circuit consists of a capacitance C and a resistor R and has a time constant

$$\tau = RC$$

Spiking Neural Network – Leaky Integrate-and-Fire Model

(continuous time formulation)





$$\Delta u_i = w_{ij} \quad if \quad t = t_j^{pre}$$

$$u_i \leftarrow \lambda_m u_i$$

$$if \quad u_i = \vartheta \qquad u_i \leftarrow 0$$

Linear, discrete time steps

Threshold → fire+reset

Previous slide:

The Leaky integrate-and-fire model written in discrete time (say time step = 1ms)
has two linear update steps:

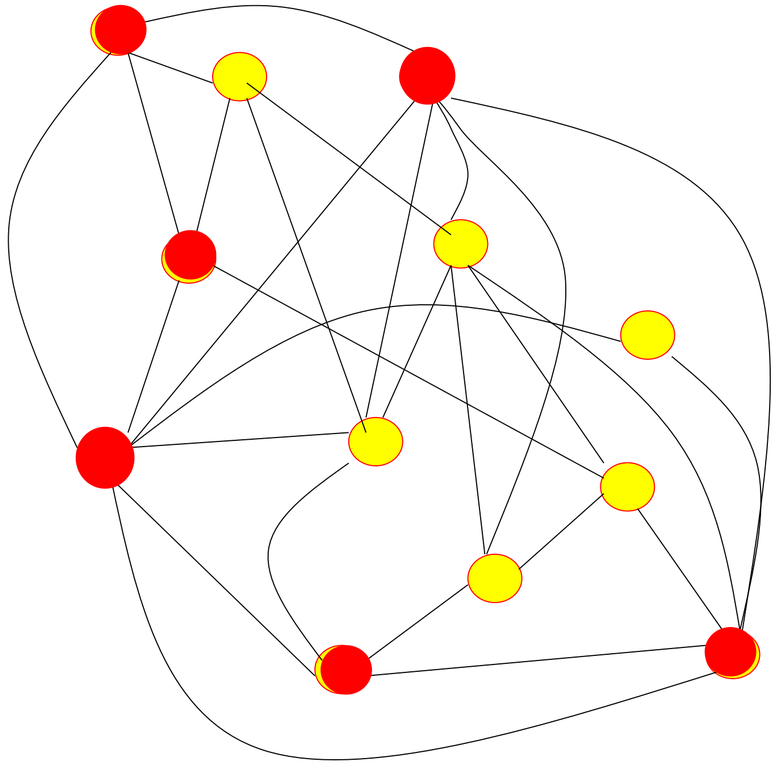
- each presynaptic spike causes a jump by the synaptic weight w_{ij}.
- In each time step the membrane potential decays with a factor λ_m<1.

continuous time involves a LINEAR differential equation that can be interpreted as an electrical RC circuit charged by a current I(t). This current I(t) consists of short electrical pulses that present spike arrivals.

The linear equation is combined with a NONLINEAR FIRE-and-RESET process. If the variable u ('membrane potential of the neuron') reaches the threshold theta, then u is reset to zero.

Summary: the brain is a large recurrent network of neurons

- Active neuron = spike emission



- spikes are rare events
- only events are transmitted → low bandwith

Previous slide:

This slide has already been shown in the very first week. In a spiking neural network, most neurons are most of the time silent. Spikes are rare events.

This is exploited in spiking hardware.

Artificial Neural Networks and RL : Lecture 14

Wulfram Gerstner
EPFL, Lausanne, Switzerland

from brain-computing to neuromorphic computing

1. Review: Local Learning rules
2. Detour: Spiking Neural Networks
3. Loihi chip (INTEL)

Previous slide:

The Loihi chip of Intel that appeared as a research support chip in 2017/2018 is interesting because it gives a direct implementation of the above three-factor rule.

INTEL:
Loihi (announced 2017, appeared 2018)

Loihi2 (announced 2021, appears 2022)

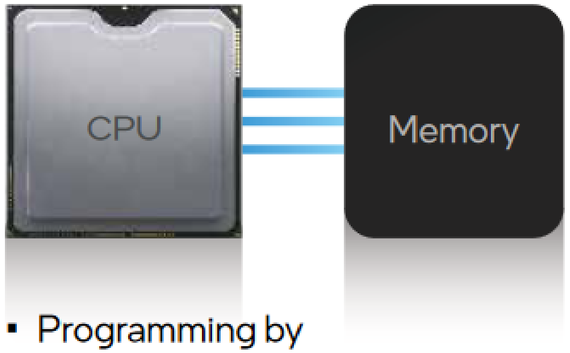
Previous slide:

More recently the first generation of Loihi has been replaced by Loihi2 with more general functionalities.

INTEL, Loihi research chip

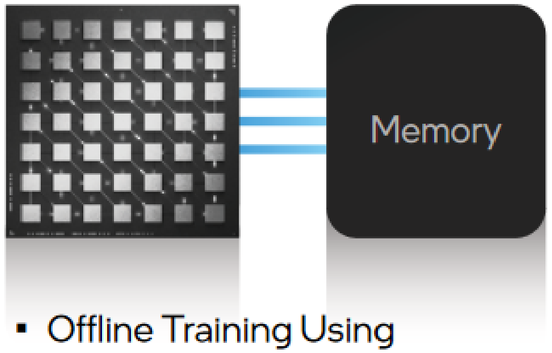
Computing Architectures

Conventional Computing

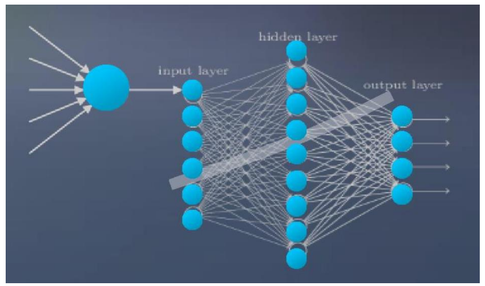


- Programming by Encoding Algorithms
- Synchronous Clocking
- Sequential Threads of Control

Parallel Computing

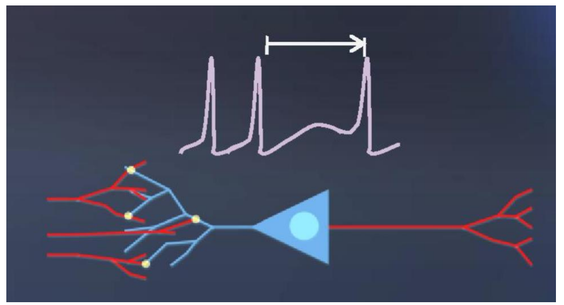


- Offline Training Using Labeled Datasets
- Synchronous Clocking
- Parallel Dense Compute



Neuromorphic Computing

- Learn On-the-Fly Through Neuron Firing Rules
- Asynchronous Event-Based Spikes
- Parallel Sparse Compute



https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf

Previous slide:

This slide from INTEL emphasize the differences in the computing architecture.

LEFT: classical von Neumann computing with separation of CPU and memory. Compute operations are mapped to logical operations performed in discrete time.

MIDDLE: Parallel computing and GPU architectures. The separation of computing and memory remains, and operations are still performed in discrete time. The only difference is that certain operations (such as convolutions) or updates of layer-wise dynamics in ANNs can be performed in parallel.

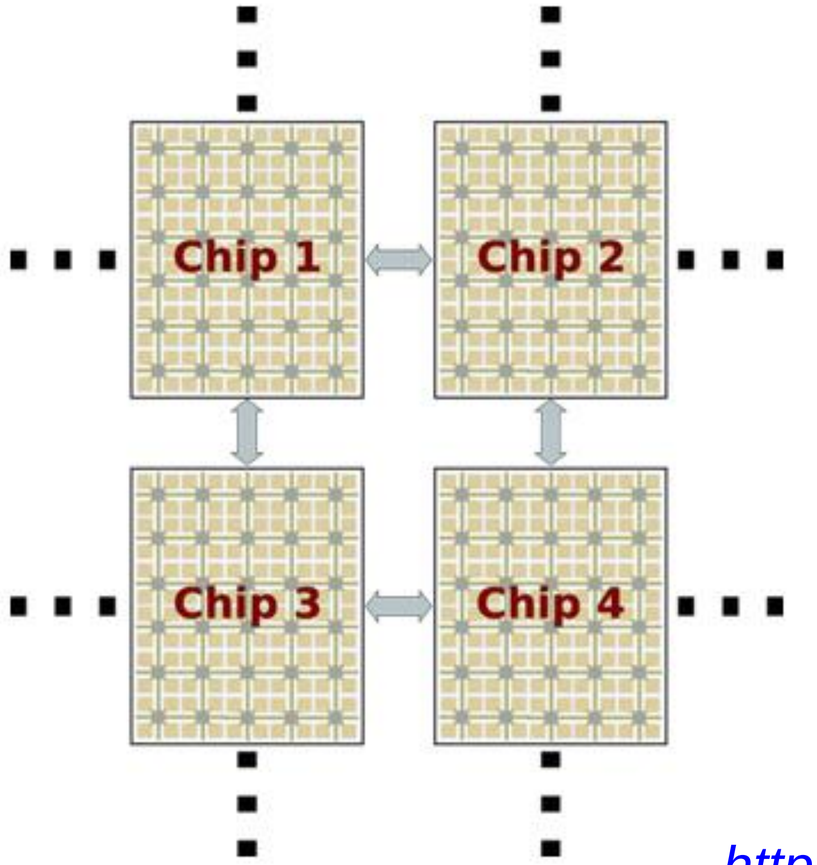
RIGHT: Neuromorphic computing architectures. Neurons compute with spikes which leads to nonlinear compute operations and signal transmission at rare moments in time defined by the moments of threshold-crossing. In between neurons are updated in ‘subthreshold’ mode with simple linear operations (leaky integration). Ideally, computing is asynchronous and in continuous time (even though this specific INTEL hardware implementation is still ‘digital’).

Two related arguments:

- energy consumption:

  Loihi < 1 W (GPU > 300W)

- asynchronous computing/event-based messaging



1 chip = mesh of 128 neuromorphic cores

Spiking neural network (SNN)

1 core = 1024 simple spiking neurons

On-chip integrated learning rule

https://en.wikichip.org/wiki/intel/loihi

Previous slide:

Why would one want to change the computing architecture?

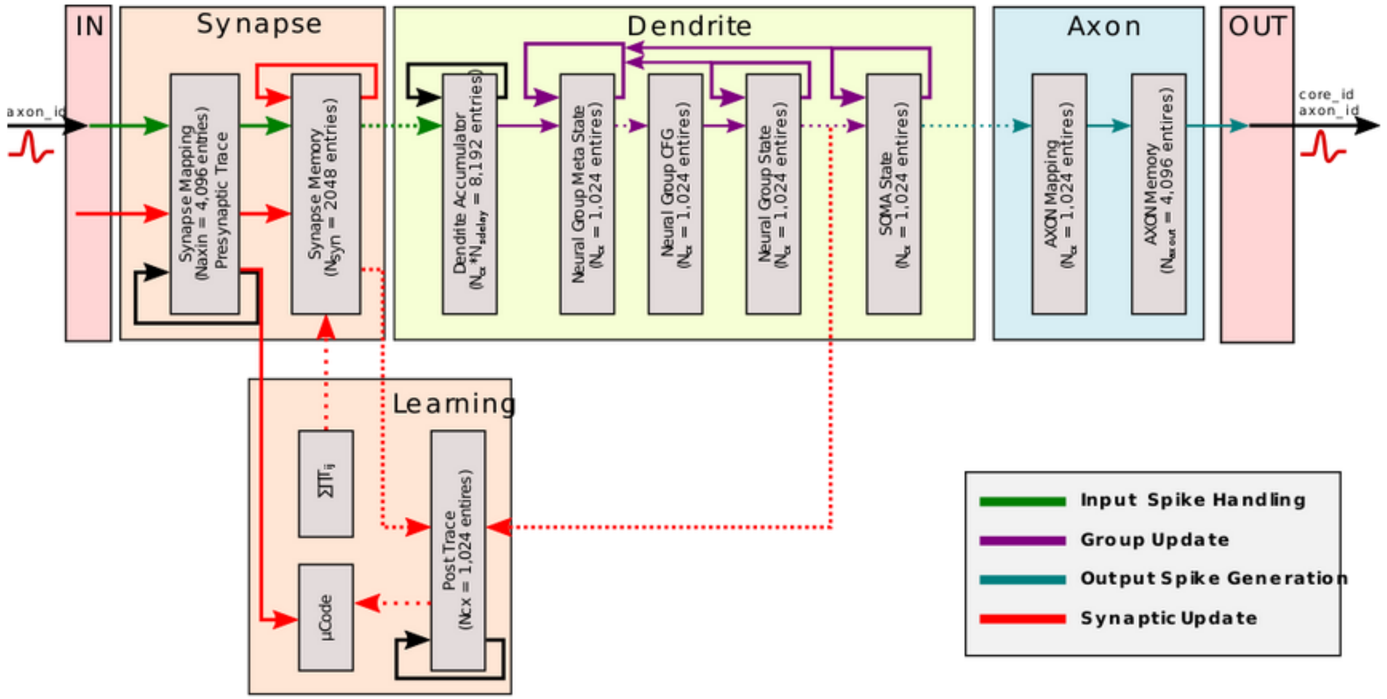
Essentially because asynchronous, event-based computing could lead to enormous reductions in energy consumptions, because expensive nonlinear processing steps and transmission steps are sparse in time: they are rare compared to the elementary time step in a discrete-time implementation.

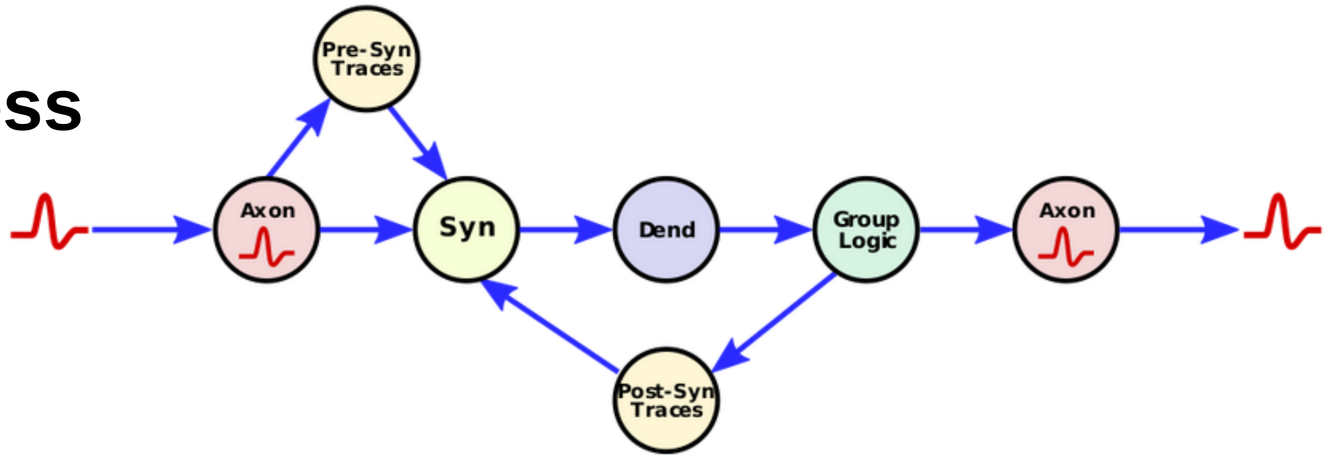
1 chip contains 128 cores, each one able to simulate about 1000 simple leaky integrate-and-fire neurons.

Loihi: (first chip, 2018)

- 128 neuron cores per chip
- Upto 128’000 neurons per chip
- 2 billion transistors

- Standard integrate-fire neuron model
- **Three-factor learning rule**
  trace(pre)trace(post)success





Previous slide:

Importantly, the framework of the learning rule that is possible on the Loihi chip is exactly that of three-factor rules explained above.

Each presynaptic spike leaves a trace. The combination with the trace left by a postsynaptic spike and a success signal defines the weight update.

Learning rules

→Loihi (2017): Three-factor learning rules

presynaptic factor, postsynaptic factor, **global success**

→ **single-layer RL algorithms**

→ Loihi2 (2022): Detailed three-factor learning rules

presynaptic factor, postsynaptic factor, **neuron-specific feedback**

→ **approximate BackProp in Multi-Layer RL**

Previous slide:

In the new version, they generalized the learning rule so that it can now also implement an approximate version of BackProp.

Introducing Loihi 2

Programmable Neurons
Neuron models described by microcode instructions

Generalized Spikes
Spikes carry integer magnitudes for greater workload precision

Enhanced Learning
Support for powerful new "three factor" learning rules from neuroscience

10x Faster
2-10x faster circuits[2] and design optimizations speed up workloads by up to 10x[3]

8x More Neurons
Up to 1 million neurons per chip with up to 80x better synaptic utilization, in 1.9x smaller die

Better Scaling and Integration
3D scaling with 4x more bandwidth per link[4], >10x compression[5] with standard interfaces

Fabricated with Intel 4 process
(pre-production)

[2] *Based on silicon characterization of Loihi 1 and a combination of silicon and pre-silicon simulation estimates for Loihi 2.*

[3] *Based on simulation modeling of a 9-layer Sigma-Delta Neural Network implementation of the PilotNet DNN inference workload compared to a rate-coded SNN implementation on Loihi 1.*

[4] *Based on pre-silicon circuit simulations.*

[5] *Based on a 7-chip Locally Competitive Algorithm workload analysis.*

See backup for analysis details. Results may vary.

Previous slide:

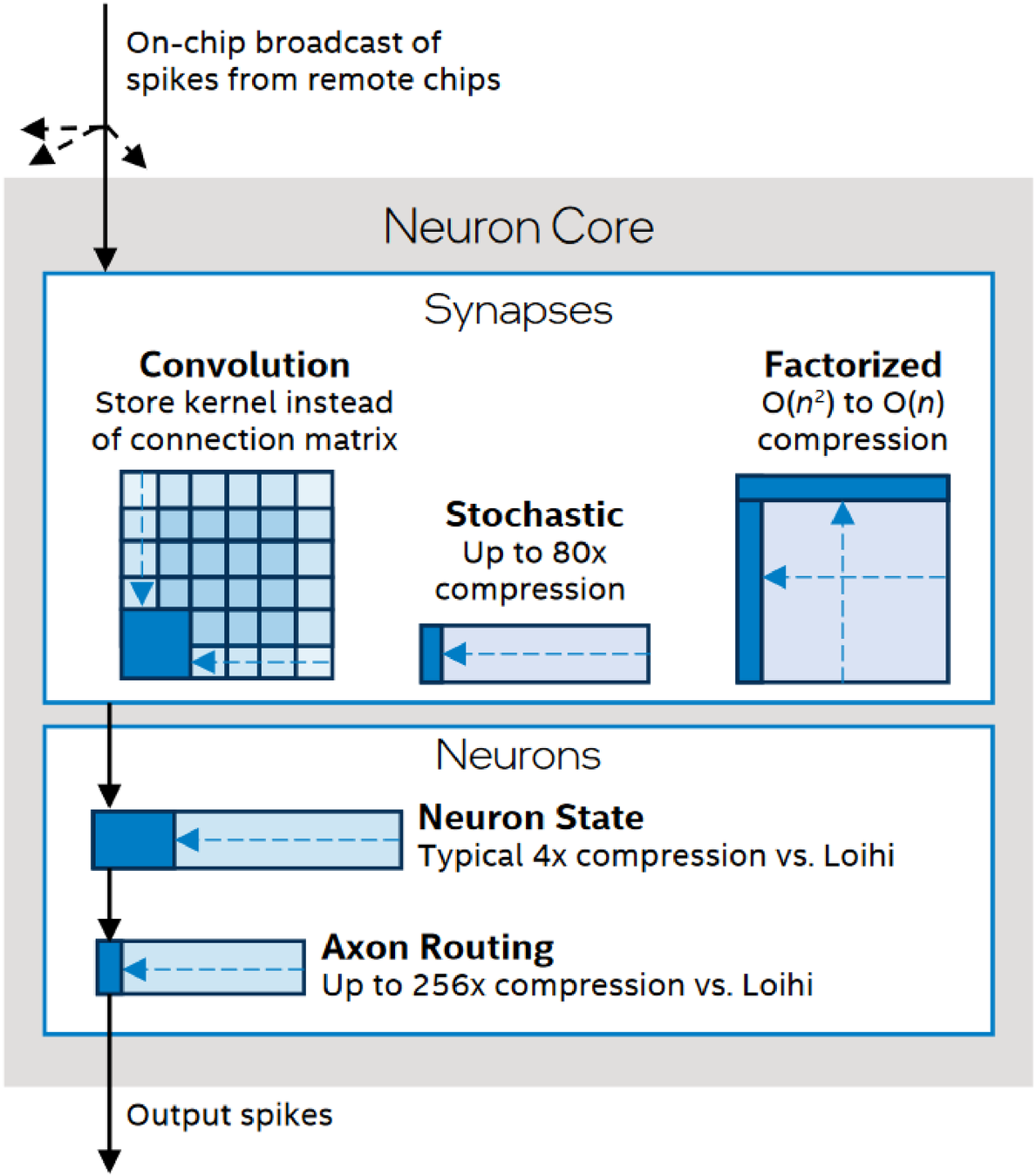
Official INTEL slide.

Loihi2 (2022):

- 128 neuron cores per chip
- Upto 1 Mio neurons per chip
- 2 billion transistors

- programmable neuron model
- programmable learning rule f(pre),g(post),3rd(neuron_i)

- spike broadcast at destination chip
- Convolutional networks
- outer-product weight matrix
- Linked to C/Phython programming interface



Previous slide:

Apart from spike broadcast (as opposed to targeted delivery lines), the chip also implements features such as weight matrices compatible with convolutional neural networks and outer-product weight matrices (factorial, see conv-net lecture).

Importantly, the learning rule framework now enables the user to switch from a GLOBAL third factor to a use-defined programmable NEURON-specific third factor.

Artificial Neural Networks and RL : Lecture 14

Wulfram Gerstner
EPFL, Lausanne, Switzerland

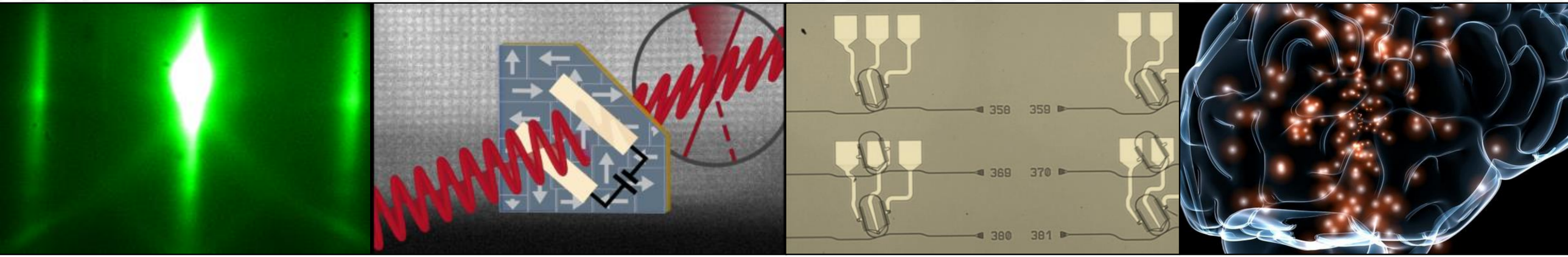
from brain-computing to neuromorphic computing

1. Review: Local Learning rules
2. Detour: Spiking Neural Networks
3. Loihi chip (INTEL)
4. Memristor chips (IBM research)

Previous slide:

Analog synaptic signal processing for neural network inference and training

Bert Jan Offrein



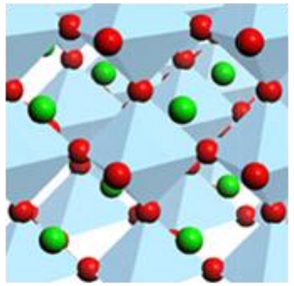
Neuromorphic Devices and Systems Group, IBM Research Europe - Zurich

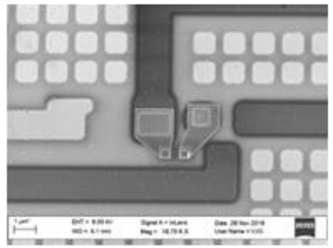
Reading

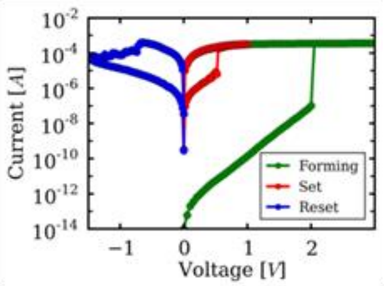
Bert Offrein et al., 2020, Prospects for photonic implementations of neuromorphic devices and systems, *IEEE Xplore, https://ieeexplore.ieee.org/abstract/document/9371915*

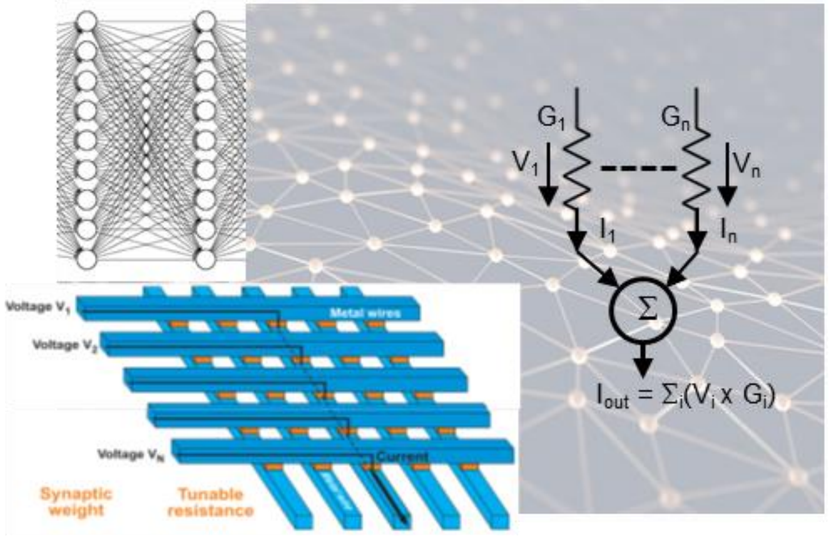
The slides are adapted from a presentation of Bert Offrein who leads a group of neuromorphic computing at IBM research in Zurich-Ruschlikon.

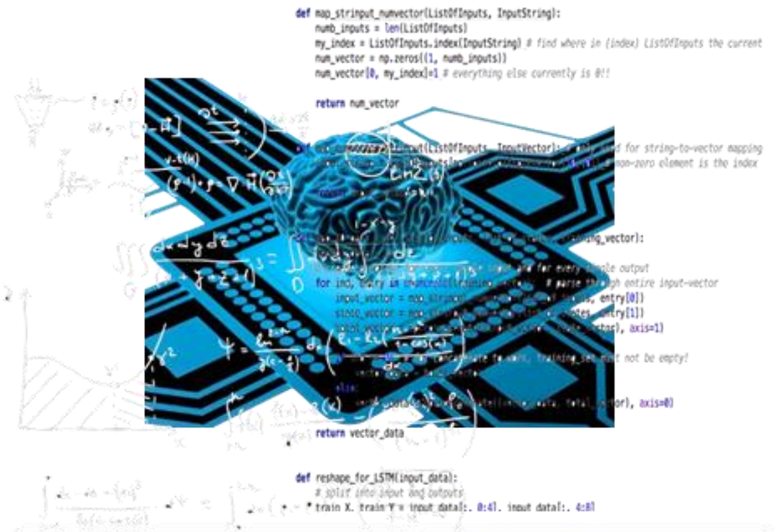
Accelerating Neuromorphic Workloads – Innovation required at all levels











New Materials and Devices

Non *von Neumann* Architecture

Hardware – Algorithm Interplay

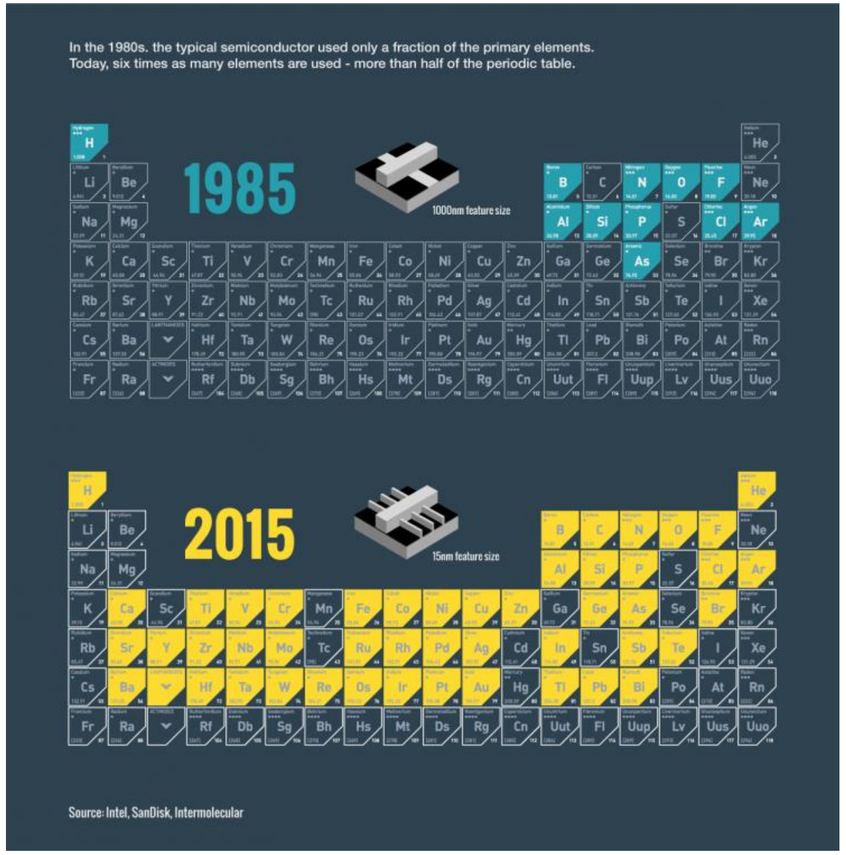
IBM slide

Previous slide:

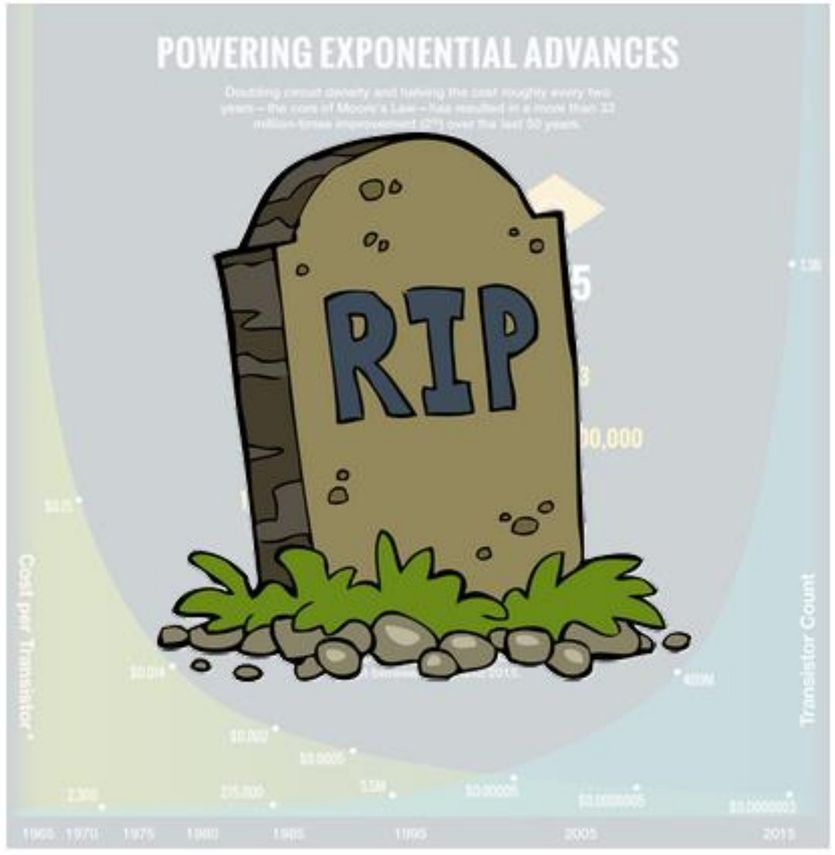
The project of IBM research focuses mostly on Matrix multiplication (middle) and update of the matrix elements as a result of a learning rule (‘algorithm’, right).

Three pillars for Si technology

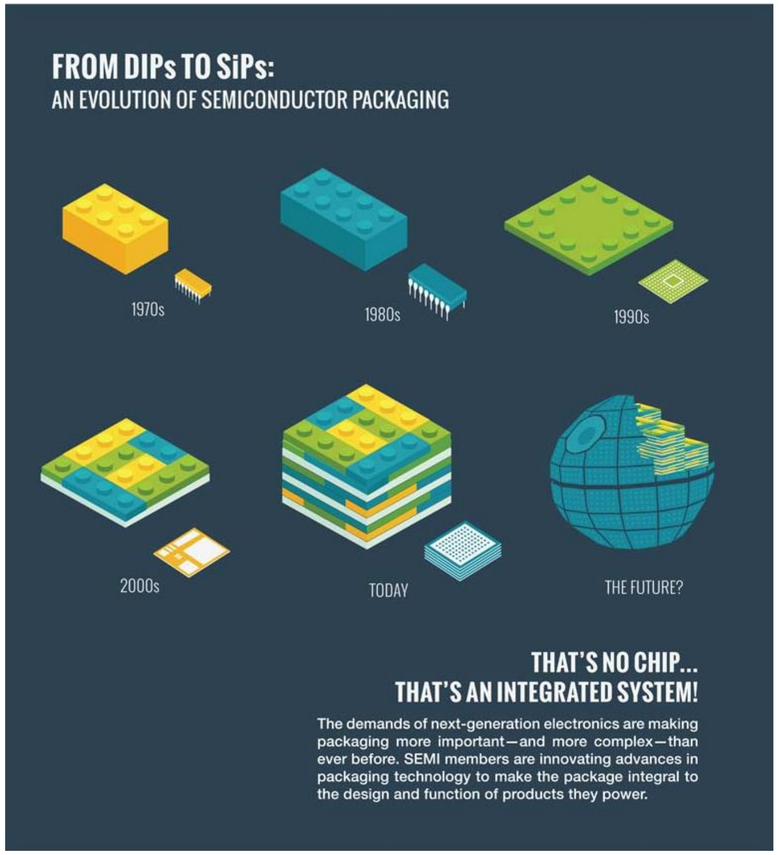
Materials



Scaling



Packaging



www.semi.org

The traditional scaling law is dead! IBM slide

Previous slide:

In the IBM research the focus is more on new materials that enable faster and energy-efficient matrix multiplication as well as weight-update rules.

The three drivers of the changes are:

Left: New materials combine many more elements than older ones.

Middle: Moore’s law, the traditional scaling law of hardware performance increase, has come to its end.

Right: Packaging has to go from 2-dim to 3-dim arrangements.

Experiment: “Human Brain vs. Computer”

Task 1: Mathematics

$$\sqrt{2} = ?$$

Task 2: Image recognition

Traditional silicon scaling ended
New types of problems gain interest

Explore new functionalities, More than Moore
Explore new computing paradigms

- approximate computing
- large parallel data streams

Previous slide:

This shows a simple theoretical experiment, where we want to compare the performance of the human brain with a computer based on two different tasks.

In task 1, both candidates have to calculate the square root of 2 as fast as possible.

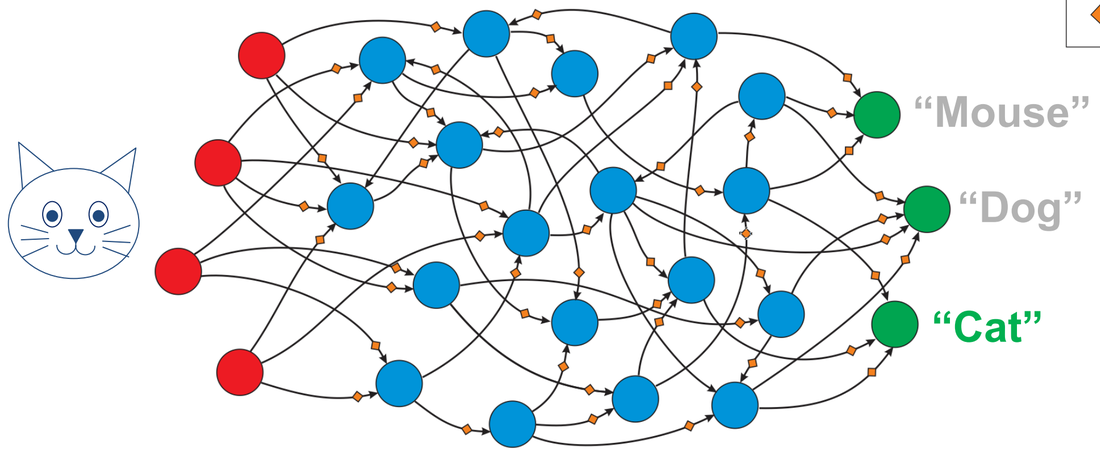
In task 2 both candidates have to interpret a scene.

The point is that that task requirements in task 2 are very different!

For example, a single noisy pixel (or noisy compute process) is less relevant. Handling of large data streams is more important.

Review Brain inspired computing:

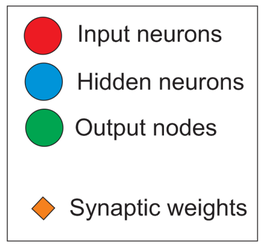
Brain-like Neural network:



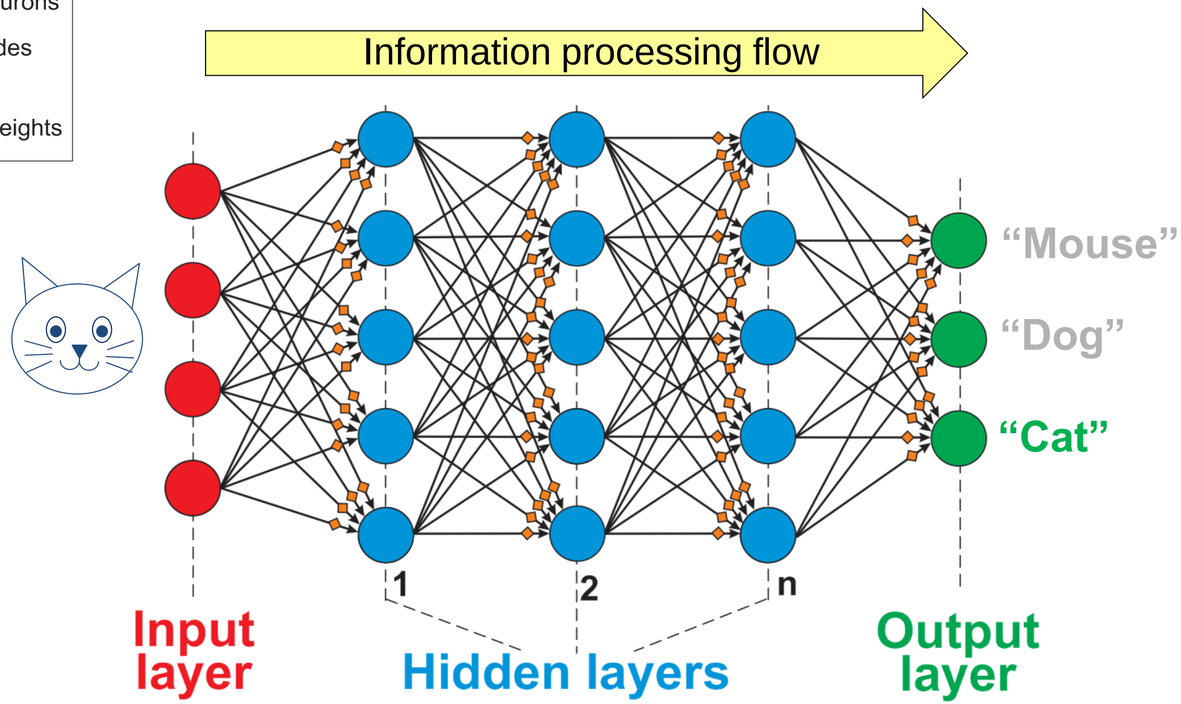
- Omni-directional signal flow
- Asynchronous pulse signals
- Information encoded in signal timing/Spiking Neural Networks

➔ Difficult to implement efficiently on standard computer hardware

Simplify



Deep Artificial Neural Network:



- Feed-forward sequential processing
- Information encoded in signal amplitude
- Neuron activation: Accumulate + Threshold
- **Training: Backpropagation Algorithm**

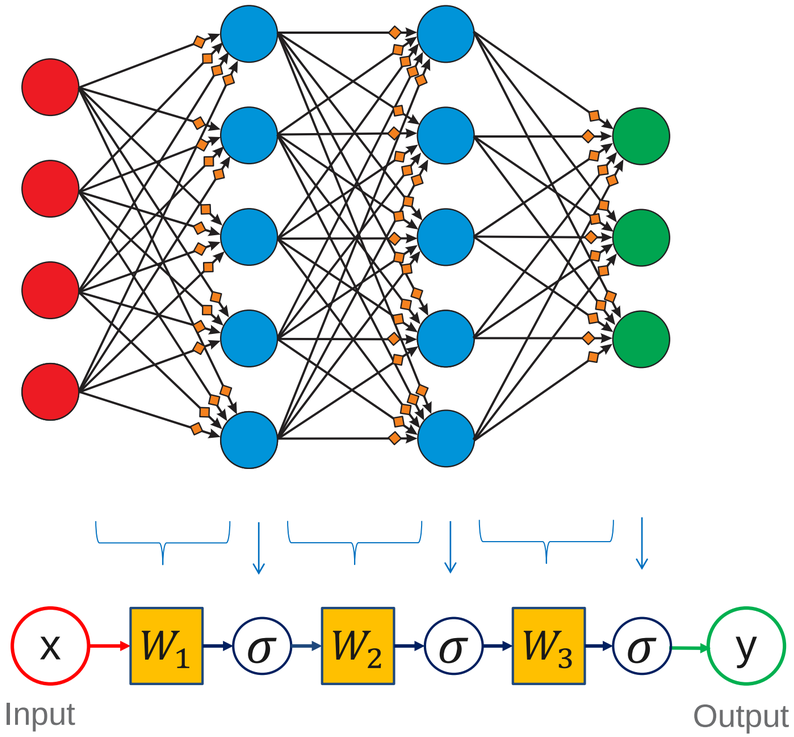
IBM slide

Previous slide:

Standard comparison of a few differences Brain vs ANN

Review: Training with Backpropagation algorithm

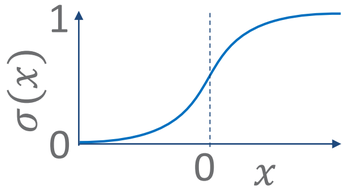
Neural net as chain of vector operations:



⟶ : Signal vector

W_n : Synaptic weight matrix

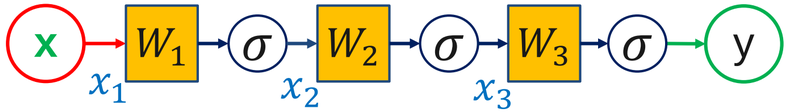
σ : Per-element neural activation function (sigmoid)



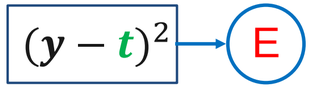
Backpropagation algorithm:

For many training cases **x** with target response **t**:

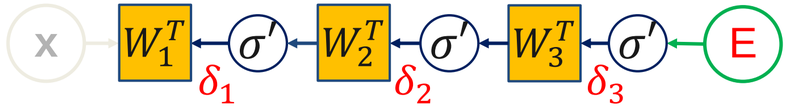
1. Forward Propagate:



2. Determine output error:



3. Backward Propagate: Determine neuron input influence δ on error **E**:



4. Adjust the active weights, proportional to their influence on the error: $\Delta W = -\eta\, \boldsymbol{x} \otimes \boldsymbol{\delta}$

IBM slide

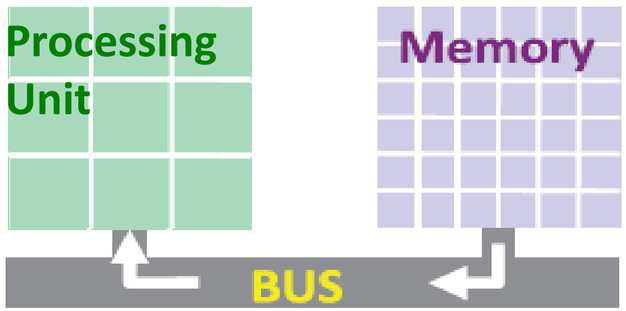
Previous slide:

Backpropagation involves

- multiple Matrix multiplications (weight matrix per layer)
- Update of the matrix elements (learning rule)

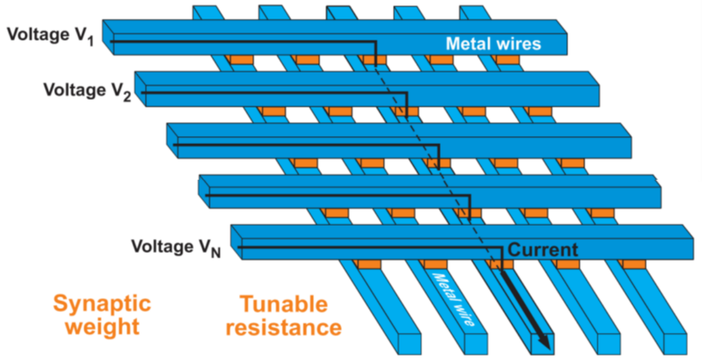
Analog signal processing for scalability

- **Limiting factors of von Neumann architecture**
  - Memory access
  - Sequential operations
  - Digital signal processing



Compute effort ~O(#Neurons^2)

- **Overcome by**
  - In-memory computing
  - Parallel operations
  - Analog signal processing



Compute effort ~O(N)

Electrical and optical solutions are viable candidates

Previous slide:

For these kind of matrix operations we should exploit new computing concepts.
The traditional von-Neumann paradigm is limited by signal flow and bad scaling as a the number of neurons per layer increases.

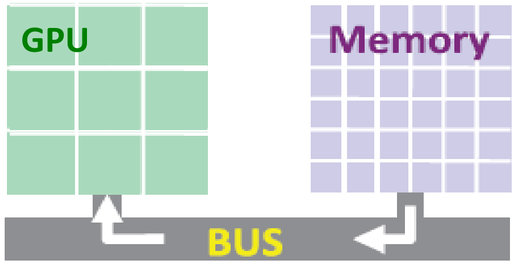
Efficient training of Deep Artificial Neural Networks:

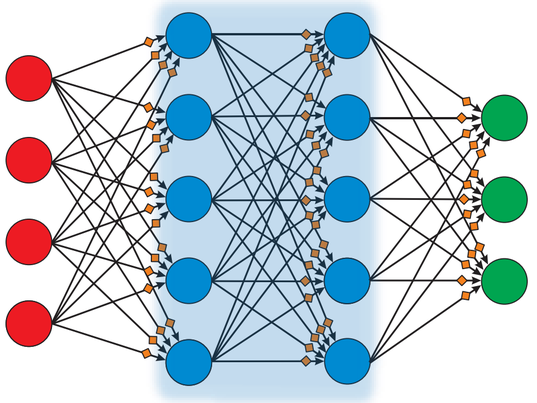
Training by Backpropagation Method:

- **Processing dominated by many large matrix operations**
  - Forward Propagation: $W_{1,2..}$
  - Backward Propagation: $W_{1,2..}^{T}$
  - Weight Update: $\Delta W_{1,2..}$

  Scale $\propto N^2$ → Neurons/layer

- **Inefficient on standard Von Neumann systems**:
  - (Mostly) Serial processing
  - Low computation to IO ratio → Memory bottleneck





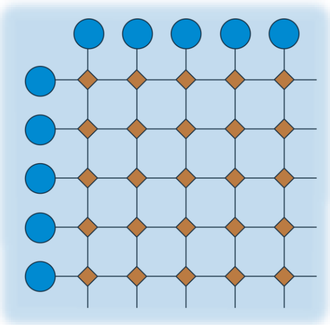
Wx

For fast and efficient neural network data processing:

- Fully parallel processing
- Tight integration of processing and memory
- Analog signal processing

Crossbar arrays
- Electrical
- Optical

x



G. W. Burr et al., “Tech. Dig. - Int. Electron Devices Meet. IEDM, vol. 2016–Febru, no. 408, p. 4.4.1-4.4.4, 2016.
T. Gokmen and Y. Vlasov, Front. Neurosci., vol. 10, no. JUL, pp. 1–13, 2016.

Previous slide:
Top:
In the week on BackPropagation we already discussed the scaling:

The algorithm scale proportional to the number of weights.
Assume that we have many layers and N neurons per layer. Then the scaling is O(N^2).

This is true for each of the three steps: forward pass, backward pass, weight update.

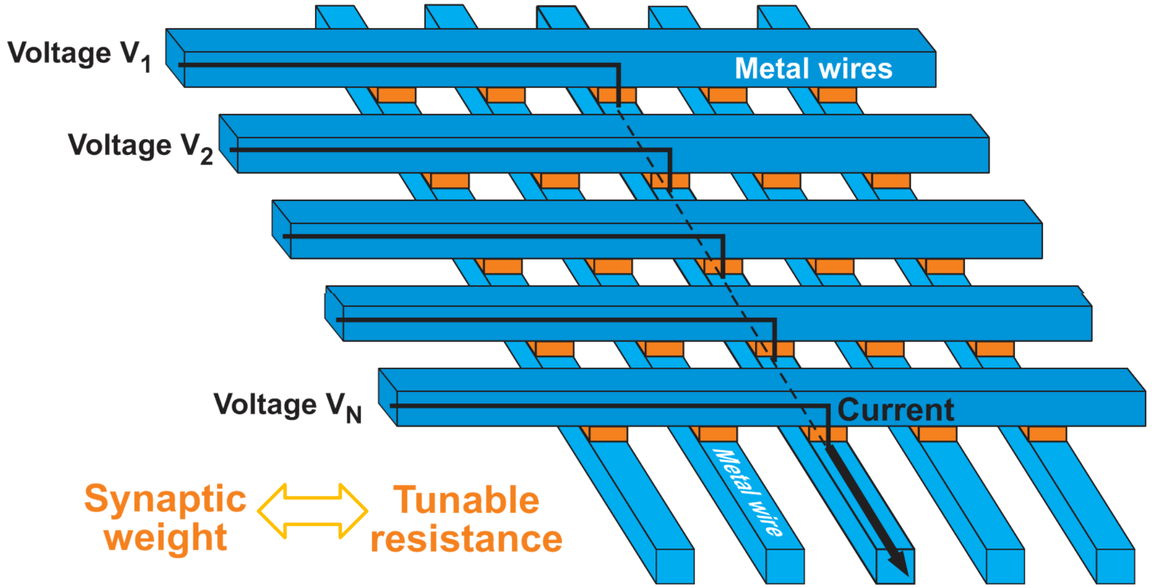
Bottom:
With analog implementation of the matrix multiplication we should be able to achieve a better scaling:

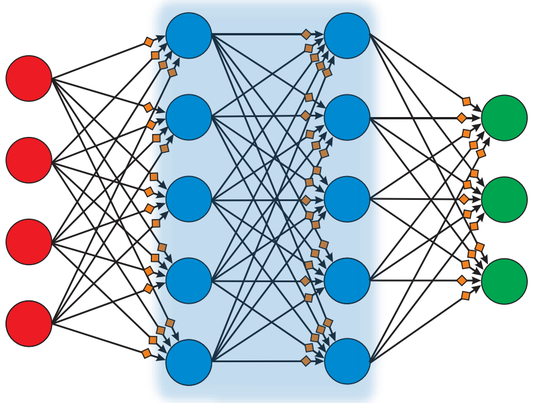
Forward pass: O(1)
Backward pass: O(1)
Weight update: O(N^2) ????

Efficient training of Deep Artificial Neural Networks:

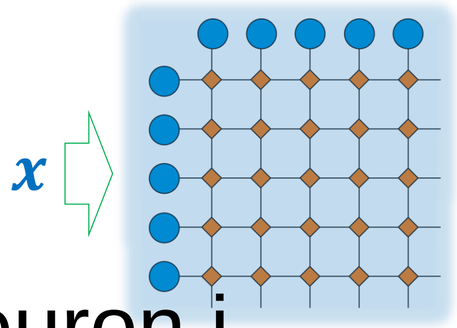
Matrix multiplication = Ohms law: $V=R\,I$

Electrical crossbar array:





Wx



Input signal $x_j = V_j$ voltage of neuron j

Weight $w_{ij} = 1/R_{ij}$ resistor at crossing

Output $I_i = \sum_j \frac{V_j}{R_{ij}} = \sum_j w_{ij}\, x_j$ current into neuron i

Previous slide:
Each blue bar is a perfect conductor. The red crossing points are tunable resistors that play the role of synaptic weights.

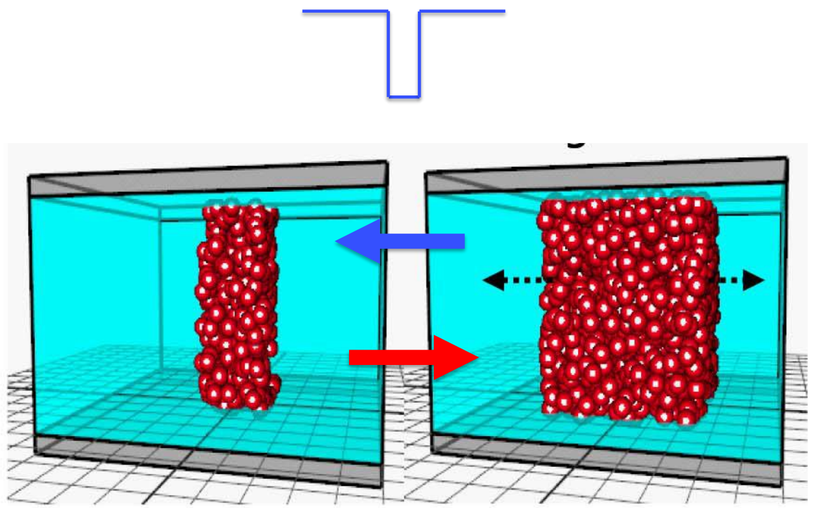
From Ohm's law follows that the current from neuron j to neuron i is $I_{ij} = V_j / R_{ij}$.

Kirchhoff's law (conservation of current) gives the final summation equation.

Tunable weights via Memristive Devices

‘memory of resistance’ = ‘memristor’

Understanding the mechanism



IBM MO_3+HfO_2

Continuous & symmetric change of R

Woo et al. IEEE Electr. Dev. Lett. 38, 9 (2017)

- Resistance depends on molecular configuration
- Resistance increase or decreases with voltage pulses above threshold value
- Resistance keeps memory

Previous slide:

Memristive material studied by IBM.

The basic function arises from the following principle.

The material in light blue is an electrical insulator (dielectric material). However, with a first strong voltage pulse one can create an initial breakdown in the material. This leads to a short-cut illustrated by a thin red column of molecules in a conducting state (lower left). Now the material is now longer insulating, but has a finite resistance.

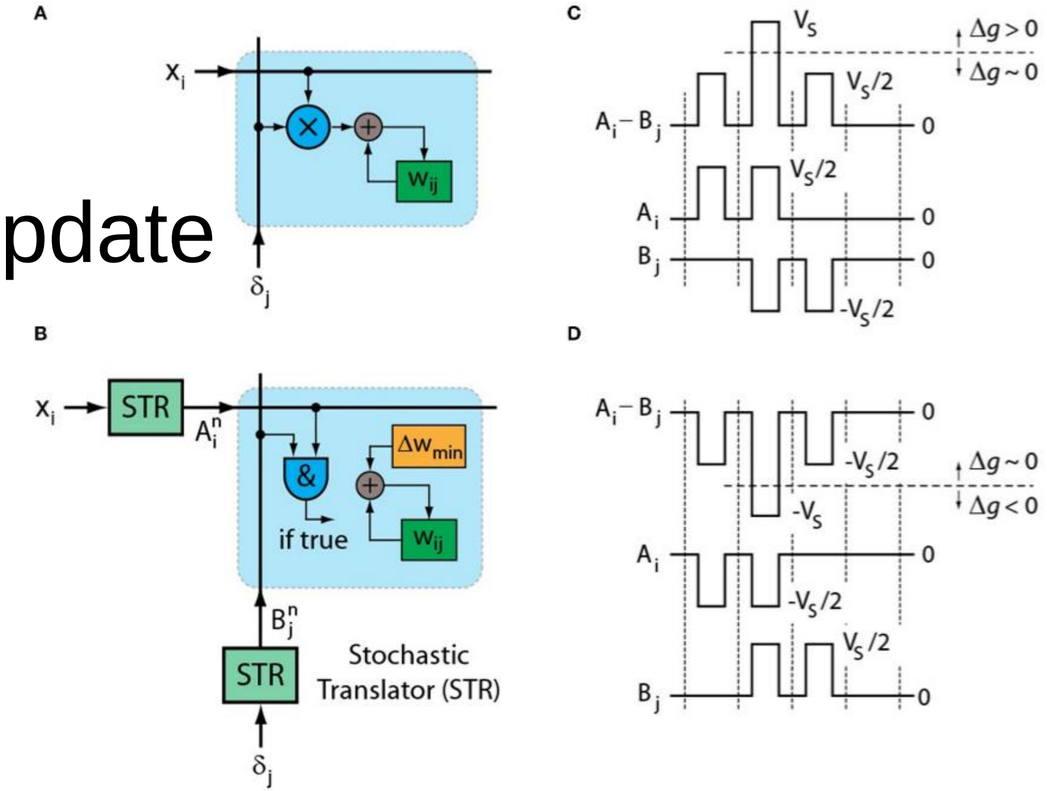
With an additional medium-sized **positive voltage pulse** (red), the column of conducting molecules can be made thicker so that the resistance decreases (lower right).

With a later medium-sized **negative voltage pulse** (blue), one can return to the initial configuration (lower left).

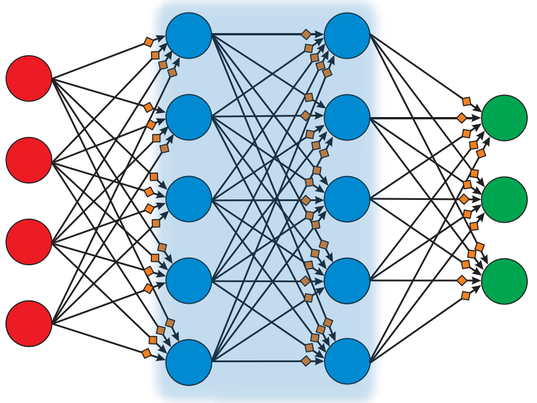
Weak currents and weak voltage pulses have no effect. Hence the material keeps its configuration and resistance for a long time. It has a **‘Memory of Resistance’ → Memristor.**

Efficient training of Deep Artificial Neural Networks: spiking network

Weight update order 1 !



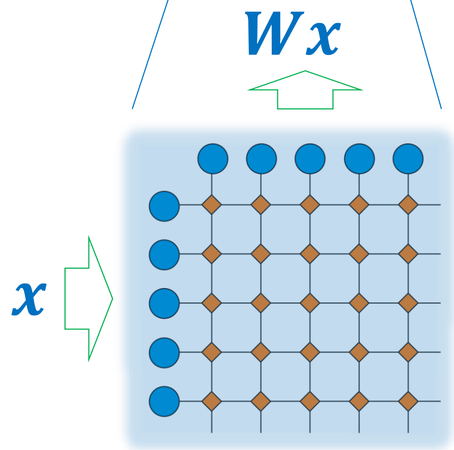
- Local Hebbian Learning rule
- Spike coding (SNN)



For fast and efficient neural network data processing:

- Fully parallel processing
- Pulse coding
- Stochastic Poisson Process

Crossbar arrays
- Electrical



Gokman and Vlasov, Acceleration of Deep Neural Networks with Resistive Cross-Point Devices Frontiers, 2016

Previous slide:

We now image the following coding principle (not yet implemented in hardware, but proposed some years ago).

Each presynaptic neurons sends voltage pulses (‘spikes’ of finite width) at random moments in time (Poisson process).

Each postsynaptic neuron sends voltage pulses (‘spikes’ of finite width) at random moments in time (an independent Poisson process).

The amplitude of the single pulse is such that it does not reach the switching amplitude of the memristive material. But if two pulses coincide, then it reaches the threshold and increases the weight (decreases the resistance).

Thus we have a proposition to implement a local (two-factor) Hebbian learning rule in hardware. And, unexpectedly, we need spike coding for this implementation scheme!

FIGURE 1 | (A) Schematics of original weight update rule of Equation (1) performed at each cross-point. **(B)** Schematics of stochastic update rule of Equation (2) that uses simple AND operation at each cross-point. Pulsing scheme that enables the implementation of stochastic updates rule by RPU devices for **(C)** up and **(D)** down conductance changes.

$$w_{ij} \leftarrow w_{ij} + \eta x_i \delta_j \tag{1}$$

where w_{ij} represents the weight value for the i^{th} row and the j^{th} column (for simplicity layer index is omitted) and x_i is the activity at the input neuron, δ_j is the error computed by the output neuron and η is the global learning rate.

In order to implement a **local and parallel** update on an array of two-terminal devices that can perform both weight storage and processing (RPU) we first propose to significantly simplify the multiplication operation itself by using stochastic computing techniques (Gaines, 1967; Poppelbaum et al., 1967; Alaghi and Hayes, 2013; Merkel and Kudithipudi, 2014). It has been shown that by using two stochastic streams the multiplication operation can be reduced to a simple AND operation (Gaines, 1967; Poppelbaum et al., 1967; Alaghi and Hayes, 2013). **Figure 1B** illustrates the stochastic update rule where numbers that are encoded from neurons (x_i and δ_j) are translated to stochastic bit streams using stochastic translators (STR). Then they are sent to the crossbar array where each RPU device changes its conductance (g_{ij}) slightly when bits from x_i and δ_j coincide. In this scheme we can write the update rule as follows.

$$w_{ij} \leftarrow w_{ij} \pm \Delta w_{min} \sum_{n=1}^{BL} A_i^n \wedge B_j^n \tag{2}$$

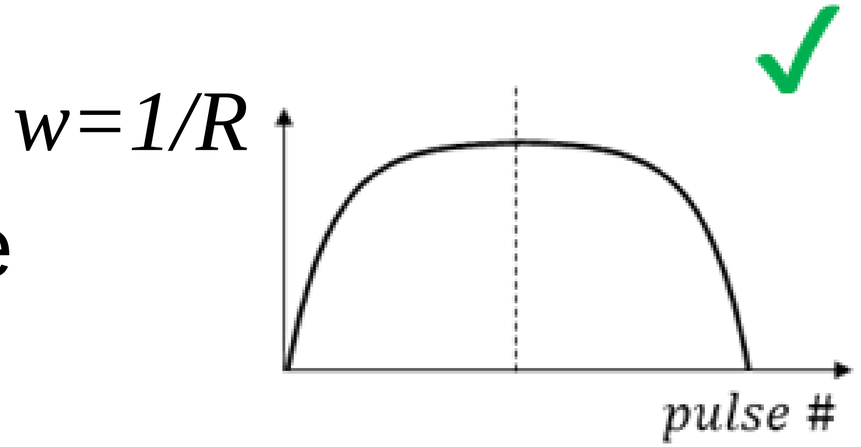
Gokman and Vlasov, Acceleration of Deep Neural Networks with Resistive Cross-Point Devices Frontiers in Neuroscience, 2016

Previous slide:

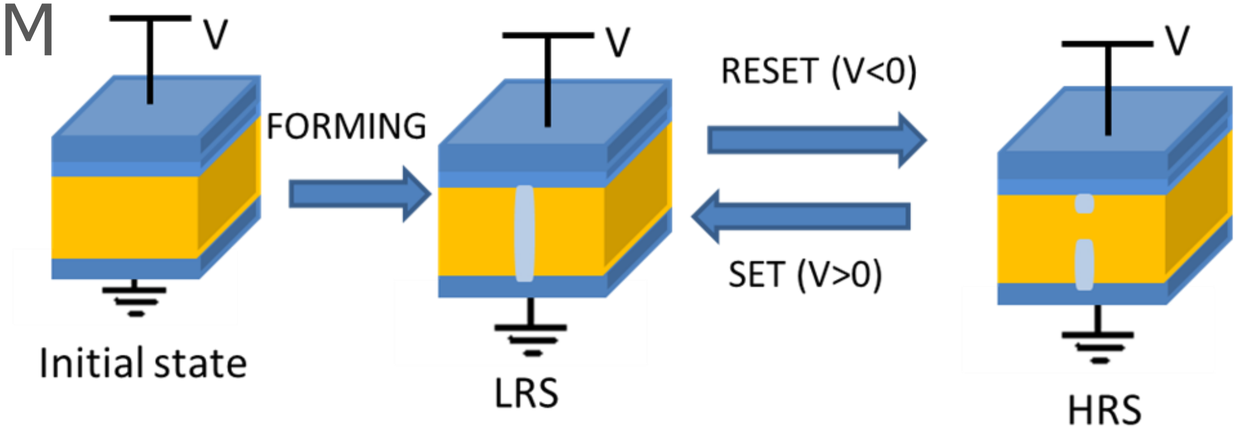
This is a copy of the relevant section of the original publication

The device challenge

- Create breakdown
  (‘mild shortcut)
- Make size of breakdown tunable



- RR
  AM



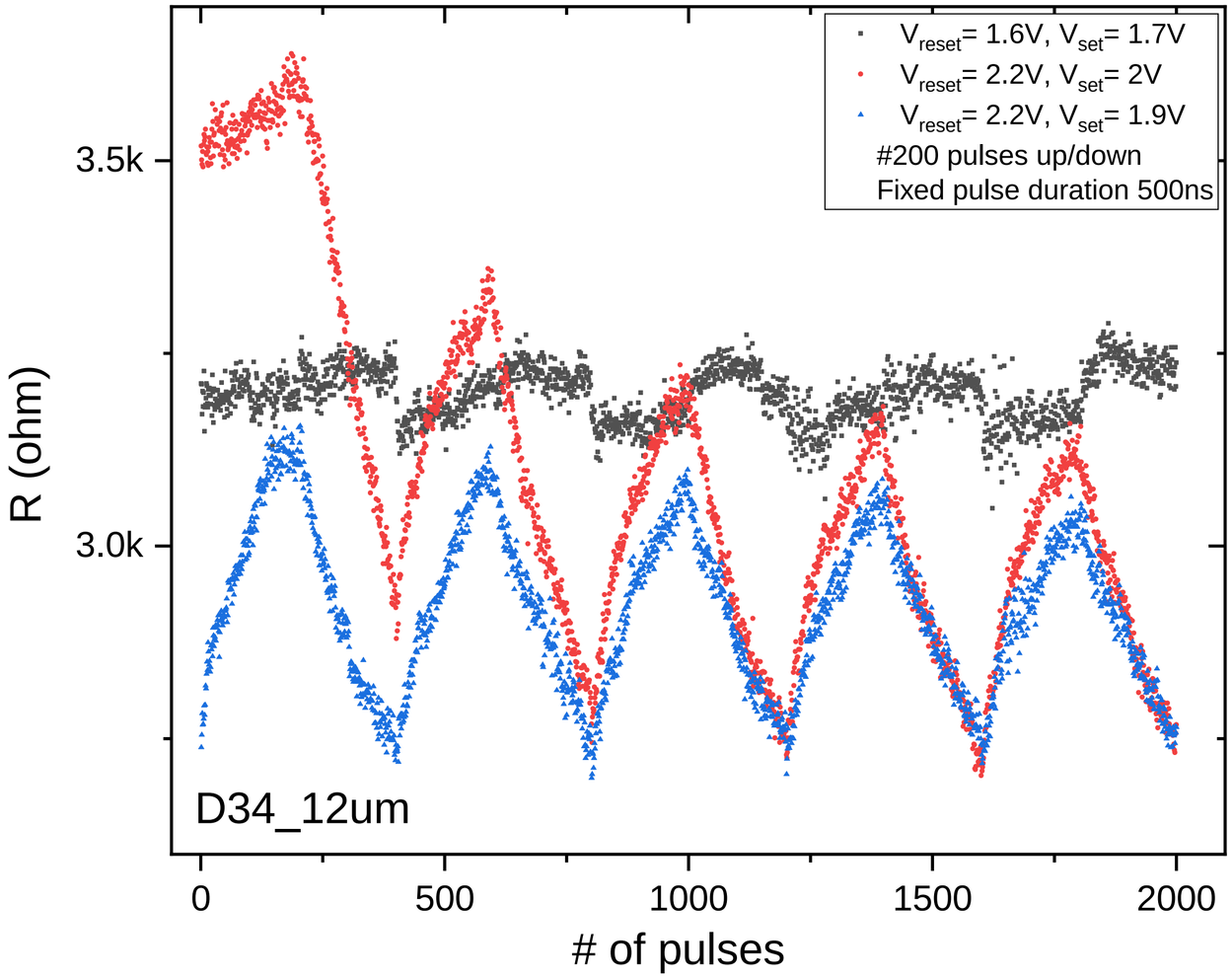
Courtesy E. Vianello

Previous slide:

Here The material in yellow is an electrical insulator (dielectric material). However, with a first strong voltage pulse one can create an initial breakdown (blue channel) in the material.

The question now is the following: Can we SMOOTHLY TUNE
sith several additional medium-sized **positive voltage pulse** (red), or **negative voltage pulse** (blue), one can return to the initial configuration (lower left).

Changes induced by 200 pulses up (and down)
→ change the weights of ANN by appropriate pulses



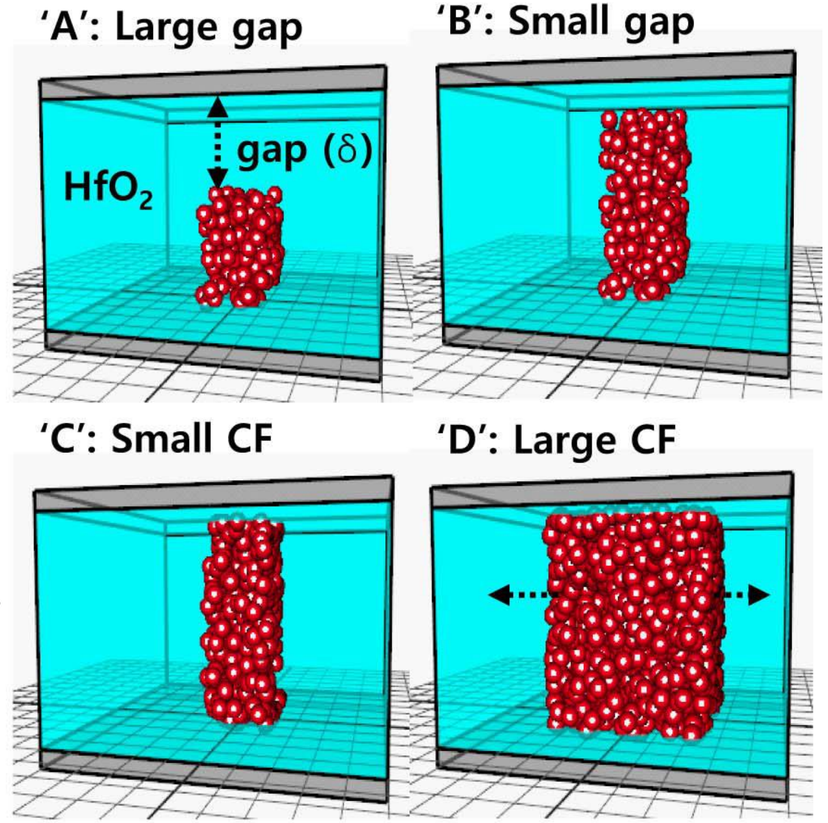
Understanding the mechanism

HfO_2 baseline

Abrupt switching

IBM MO_3+HfO_2

Continuous & symmetric change of G



Woo et al. IEEE Electr. Dev. Lett. 38, 9 (2017)

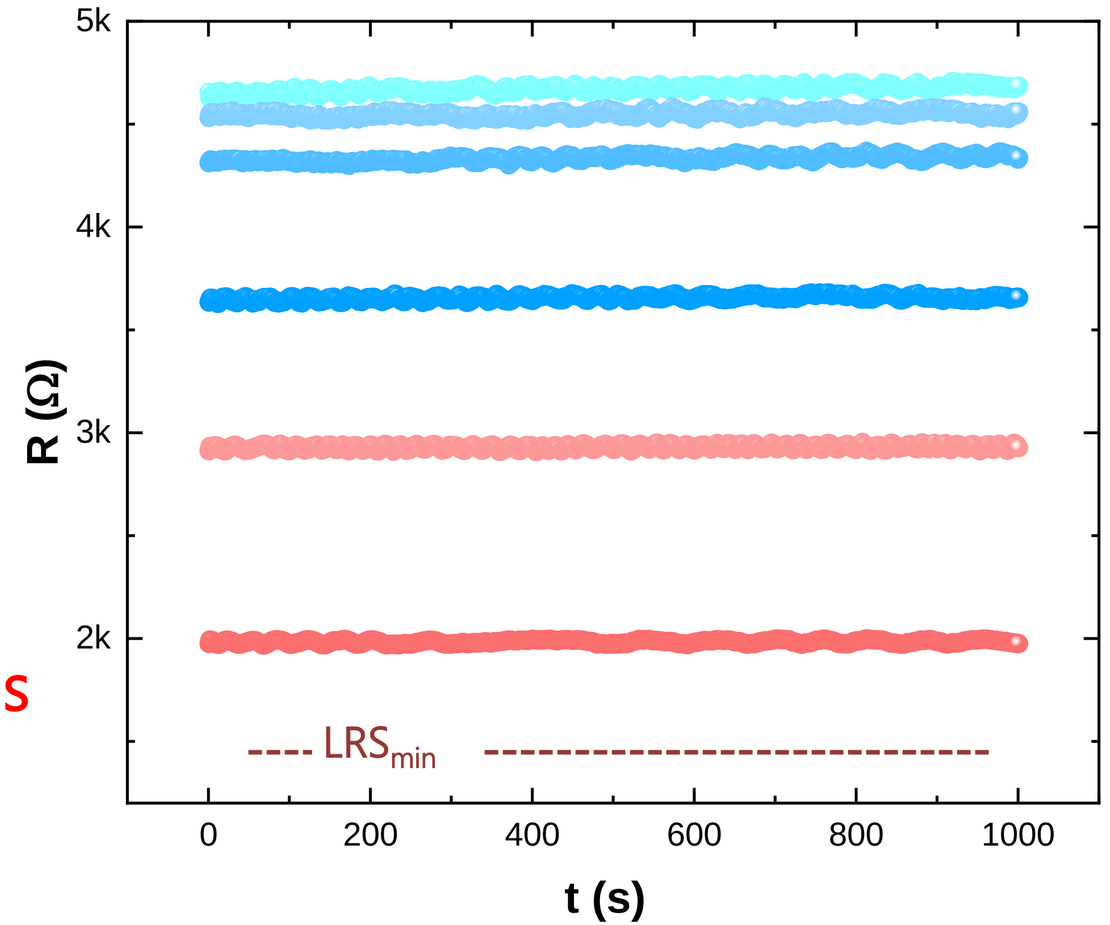
Experimental demonstration of symmetric and continuous change of G

Previous slide:

Experimental test with the material at the bottom shows that smooth tuning is possible (blue dots)

Retention of the Resistance Values over time (intermediate values)

1. Negative sweep to put it in Low-Resistance State
2. Read at 0.2V constantly for 1000s
3. Negative + positive cycle to put it in increasing High-Resistance State (HRS)
4. Read at 0.2V constantly for 1000s for each HRS



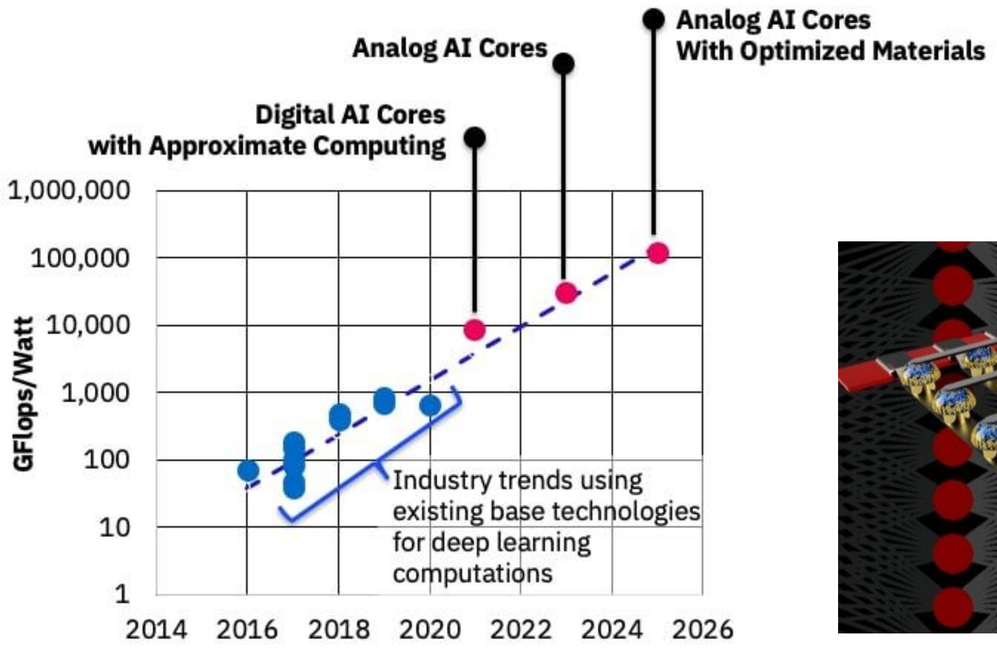
Intermediate states show no drift up to 1000 s
→We can change values of resistance
→New resistance is reliable over time
→We can change again
→'Online Learning'

Valeria Bragaglia Email: vbr@zurich.ibm.com

Previous slide:

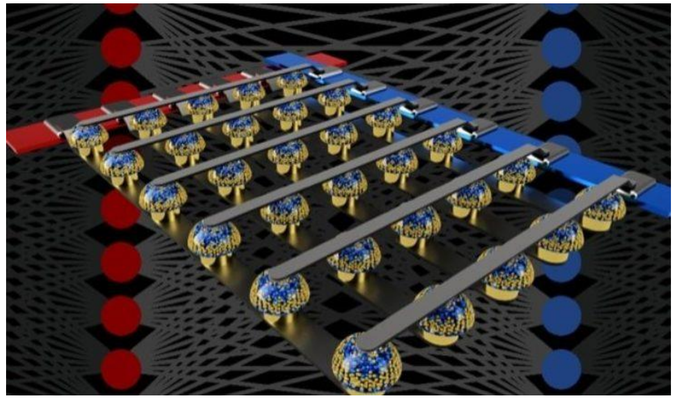
Moreover, after tuning the resistance remains constant

AI Technology roadmap



compute performance efficiency

New memristive devices required



Bloomberg the Company & Its Products | Bloomberg Anywhere Remote Login | Bloomberg Terminal Demo Request

Bloomberg

Technology

IBM Invests $2 Billion in New York Research Hub for AI

- Analog AI Cores
  - For the synaptic processing function
  - Apply memristive devices: Ohms law & Kirchhoff's law
  - Parallel forward inference & backward and weight update

Analog synaptic processing

	Inference	Training
Resistance	1-100 MΩ	1-100 MΩ
# Levels	100	1000
Weight set / update	To desired level	Symmetric

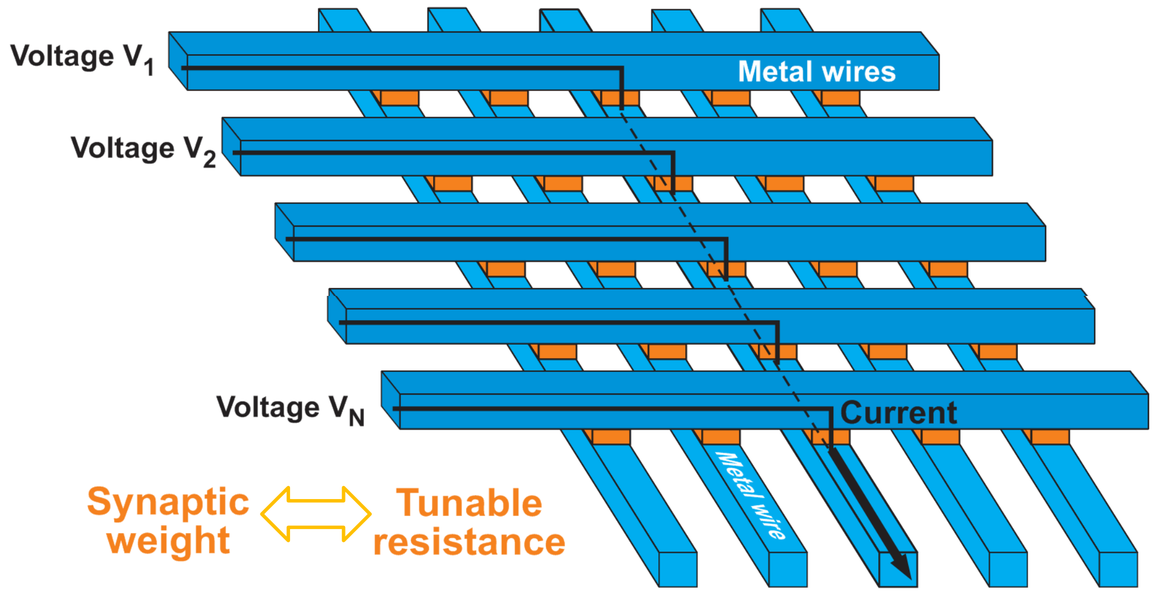
Previous slide:

The road map shows several aspects:

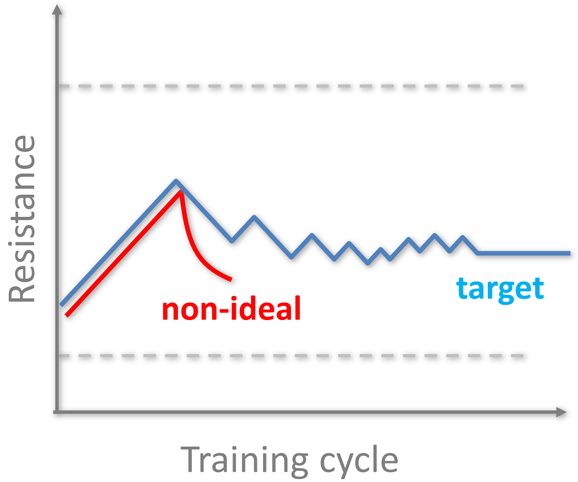
- Traditional scaling has increased not only the compute power, but also reduced the Watt per Flop.
- Traditional scaling expected to come to an end (or may continue, red dots)
- Staying digital, but allowing for approximate computing might give an extra jump in performance
- Going analog would yield a further jump.

Analog crossbar arrays: Update for BackProp

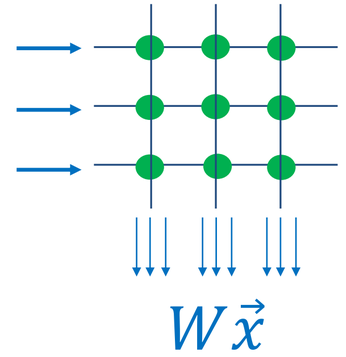
Electrical crossbar array:



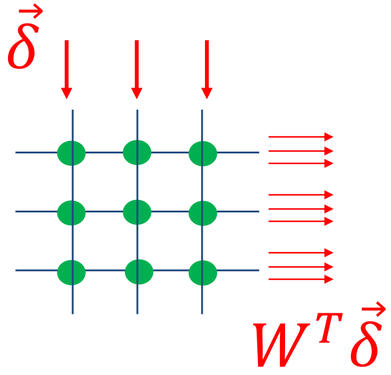
- **Weight update:** proportional to signals on row and column
  - –Symmetric increase and decrease of weight
  - –~1000 analog levels required
- **Physical challenge**: Identify material systems that meet these requirements



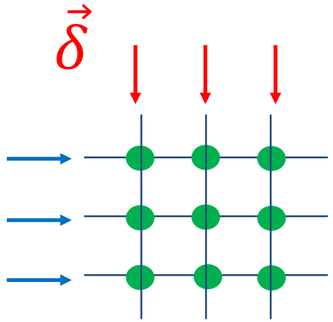
Forward propagation:



Backward propagation:



Weight update:



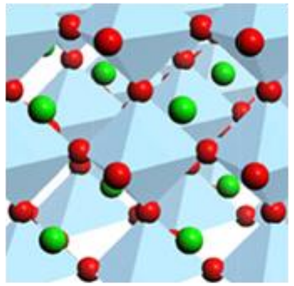
$$\Delta w_{ij} = -\eta\, x_i\, \delta_j$$

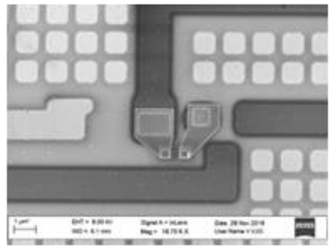
Previous slide:

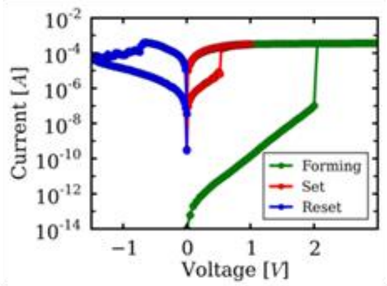
To have an impact, local Hebbian rules are not enough.

But one can also extend these ideas to local implementations of BackPROP.

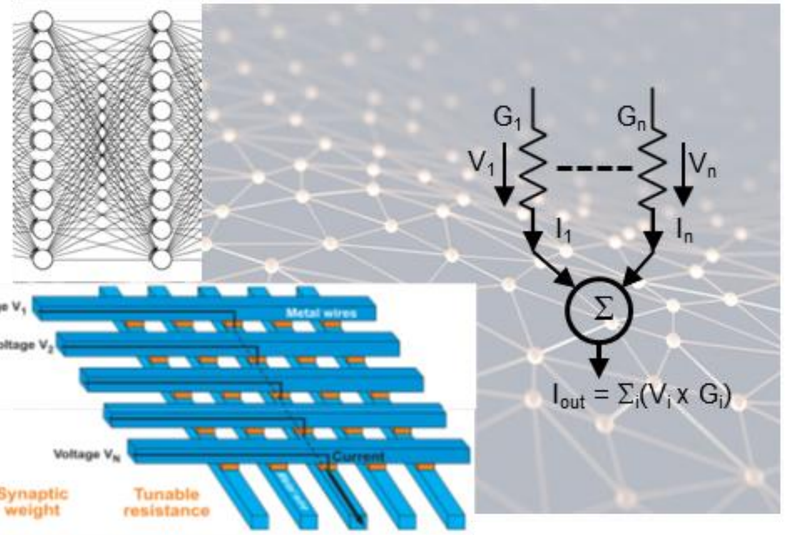
Neuromorphic – local learning rules in hardware



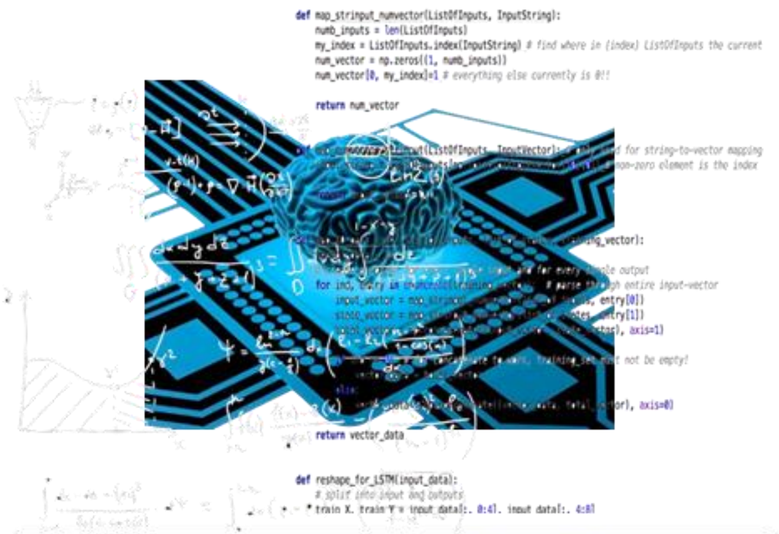




New Materials and Devices



Non *von Neumann* Architecture



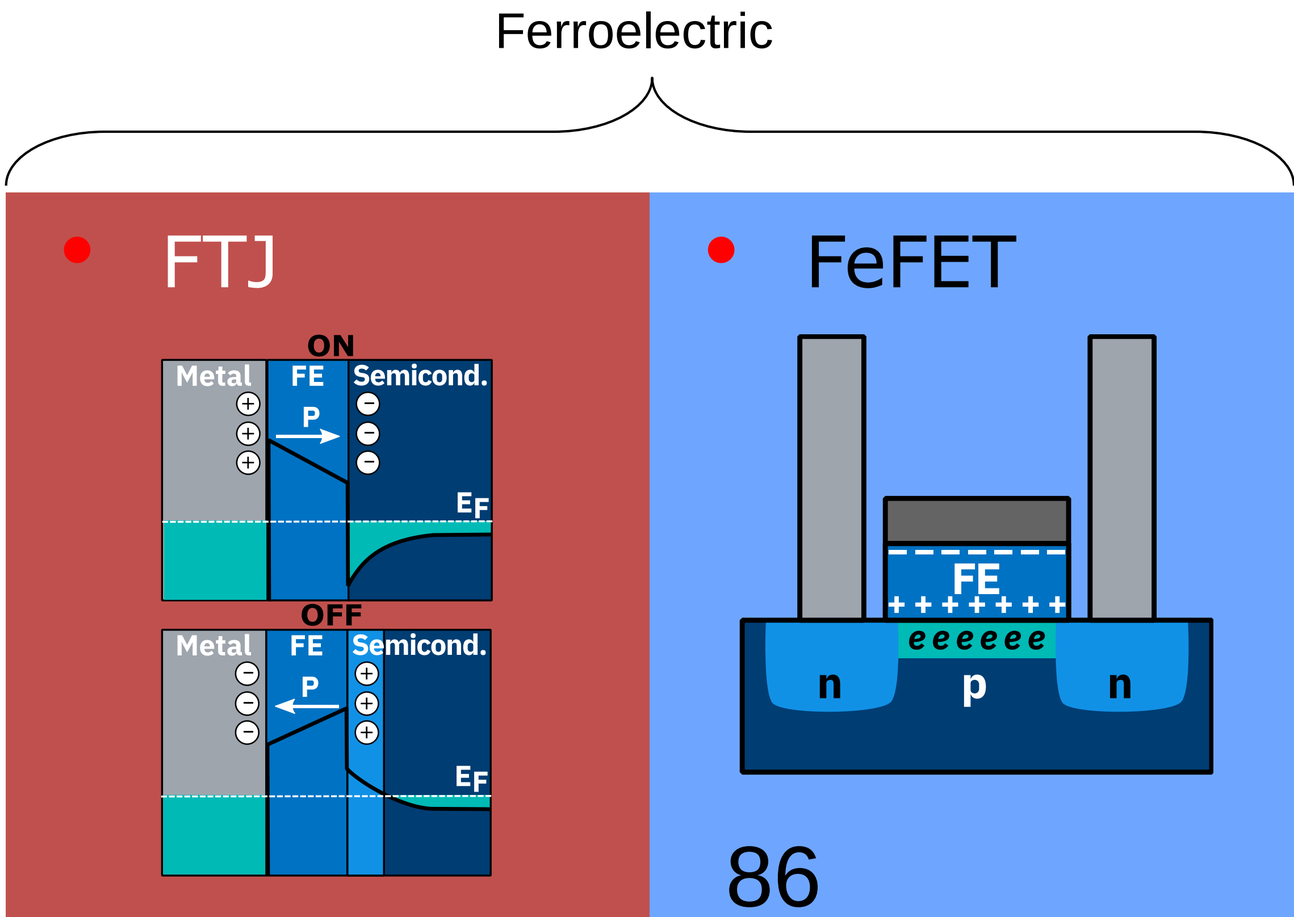
Hardware – Algorithm Interplay

→ Extension from two-factor to three-factor rules possible O(1)!

→ Extension to (approximative) Backprop possible O(N)!

Literature of ferroelectrics in AI hardware

- 2011: discovery of a ferroelectric phase in HfO_2.
- 2017: FeFET integrated in a 28nm HKMG technology (Mulaosmanovic *et al.*, VLSI 2017)
- 2018: IBM: crystallization of $HfZrO_4$ in the FE phase **below 400°C** (O'Connor *et a*l., APL Mater. 6, 121103 (2018))
- 2020: this work: first demonstration of a BEOL, CMOS FeFET



Summary

- Silicon technology remains the basis for computing devices
  - Leverage existing processes, infrastructure and know-how
  - Continuous extending of materials and function

- New computing paradigms – Neuromorphic computing - provides a path to handle unstructured data
  - Analog signal processing in crossbar arrays
  - Parallel processing of key algorithms in neural networks
  - Electrical and optical implementations

→ Extension to three-factor rule possible!

Previous slide:

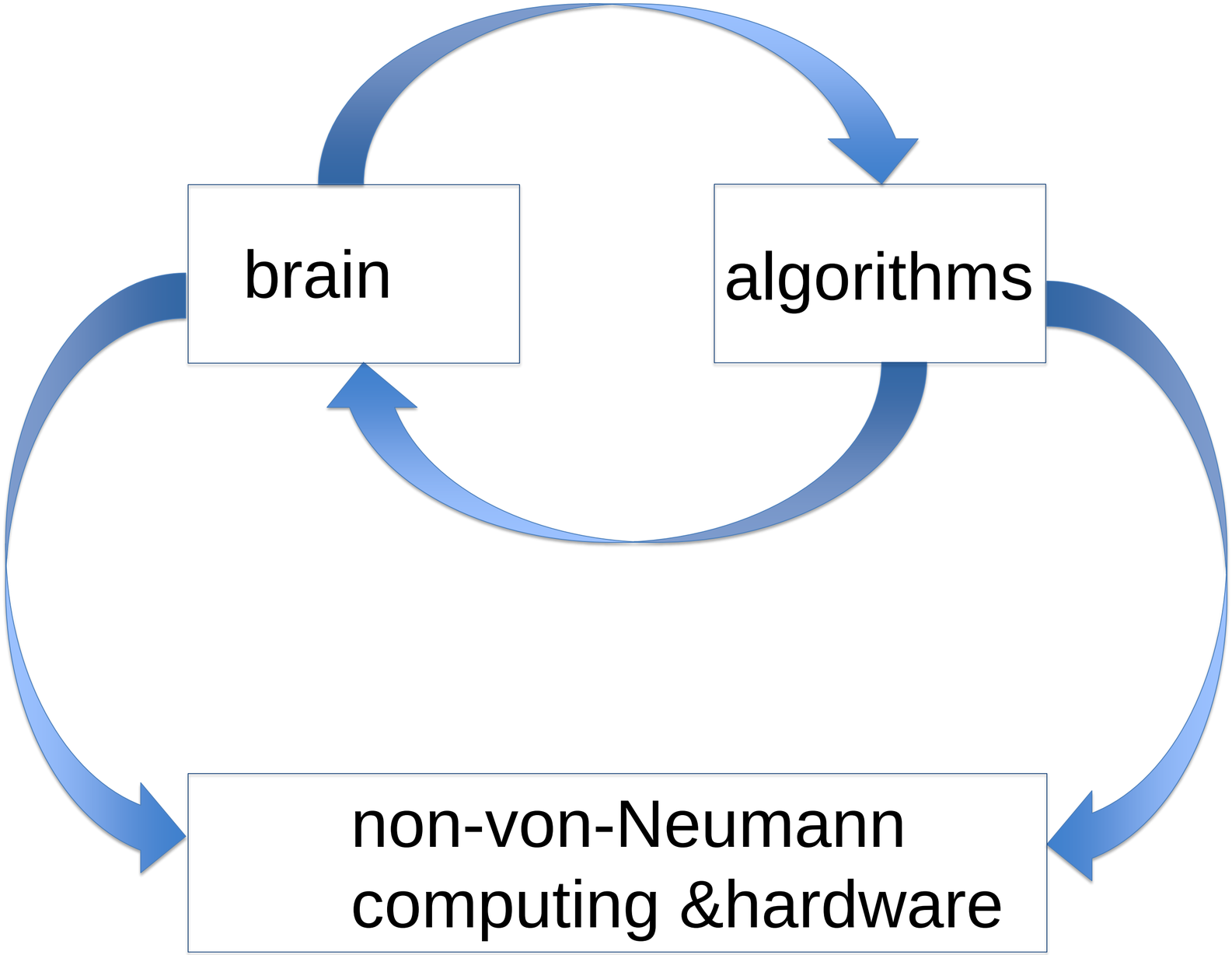
Artificial Neural Networks and RL : Lecture 14

Wulfram Gerstner
EPFL, Lausanne, Switzerland

from brain-computing to neuromorphic computing

1. Review: Local Learning rules
2. Detour: Spiking Neural Networks
3. Loihi chip (INTEL)
4. Memristor chips (IBM research)
5. The problem of energy consumption

Learning Rules



Energy consumption of the brain

- Sedentary humans eat and use 2500 kCal per day
- Translate to Joule → 10 000 kJ
- Brain facts: 20 percent of energy consumption of human at rest goes into the brain
- Most of it goes into synaptic signaling
- Brain uses 24 – 30 Watt (5 modern light bulbs)

The power consumption of the brain is relatively low!
→ 10h of hard thinking = 0.3kWh

https://www.brainfacts.org/Brain-Anatomy-and-Function/Anatomy/2019/How-Much-Energy-Does-the-Brain-Use-020119
https://biology.stackexchange.com/questions/16316/what-is-the-energy-consumption-of-the-brain

Previous slide.
Claim the power consumption of the brain (30W) is relatively low.

Low compared to what?
- Compare with GPU
= Compare with household power consumption.

Compare: Energy consumption of one GPU

- 300 W (RX 6800/6900 XT)
- 350 W (RTX 3080/3090)

https://www.tomshardware.com/features/graphics-card-power-consumption-tested

Previous slide:

Energy consumption of one GPU

- 300 W (RX 6800/6900 XT)
- 350 W (RTX 3080/3090)

→ 10h of training an ANN on 1 GPU = 3.5 kWh

1 day of training an ANN on 1 GPU = 8000Wh = 8 kWh

4 months GPU usage → 1000 kWh

12 months GPU usage → 3000 kWh

https://www.tomshardware.com/features/graphics-card-power-consumption-tested

Previous slide:

Electrical household energy consumption

Typical Swiss uses in household (fridge, TV, light)
→ about 1000 kWh per year and person.

- 2 persons sharing apartment = 2200kWh per year
- 4 persons sharing house = 4000kWh per year

Heating/warm water with heat pump
4 persons sharing house 6000kWh per year
→ 1500 kWh per year and person

https://pubdb.bfe.admin.ch/de/publication/download/10559

Previous slide:

Comparison

- Brain 30W → 260 kWh per year and person
- Living in Switzerland → 2500 kWh per year and person
- GPU → 3000 kWh per year and GPU

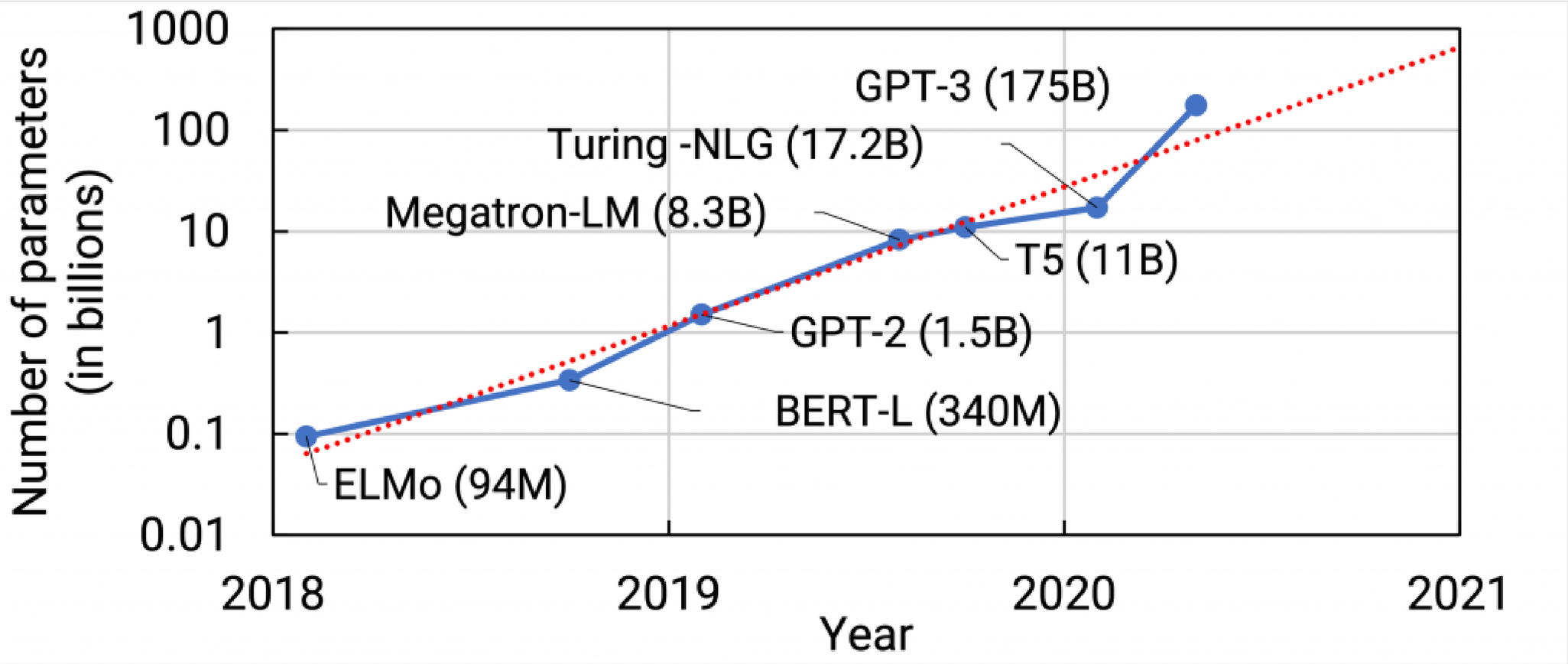
Problem!!!!

Solution? – use your machine carefully!

- think about better computer architecture!

Previous slide:

The neural network *size* explosion

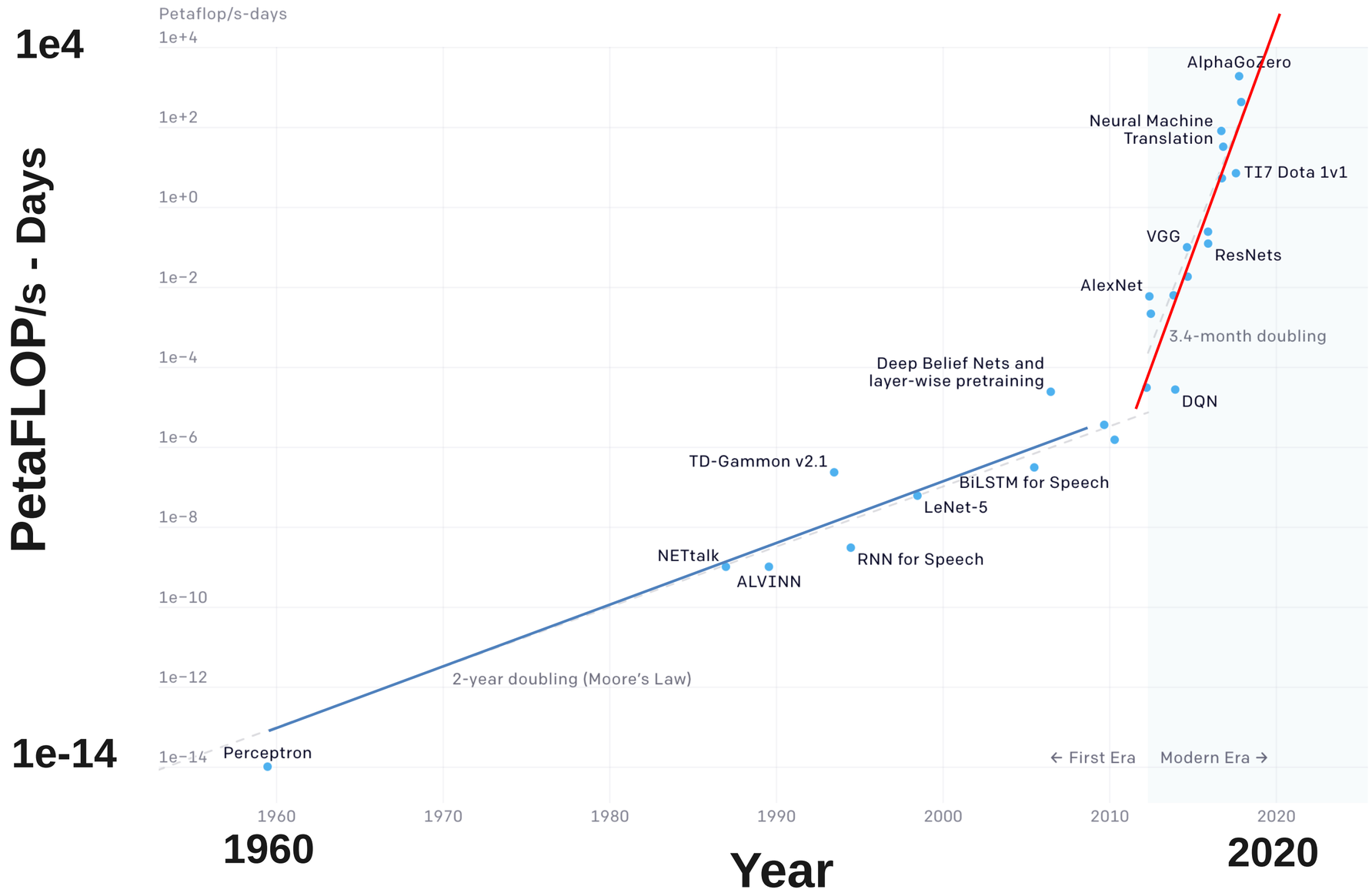


- Source: NVIDIA

Previous slide:

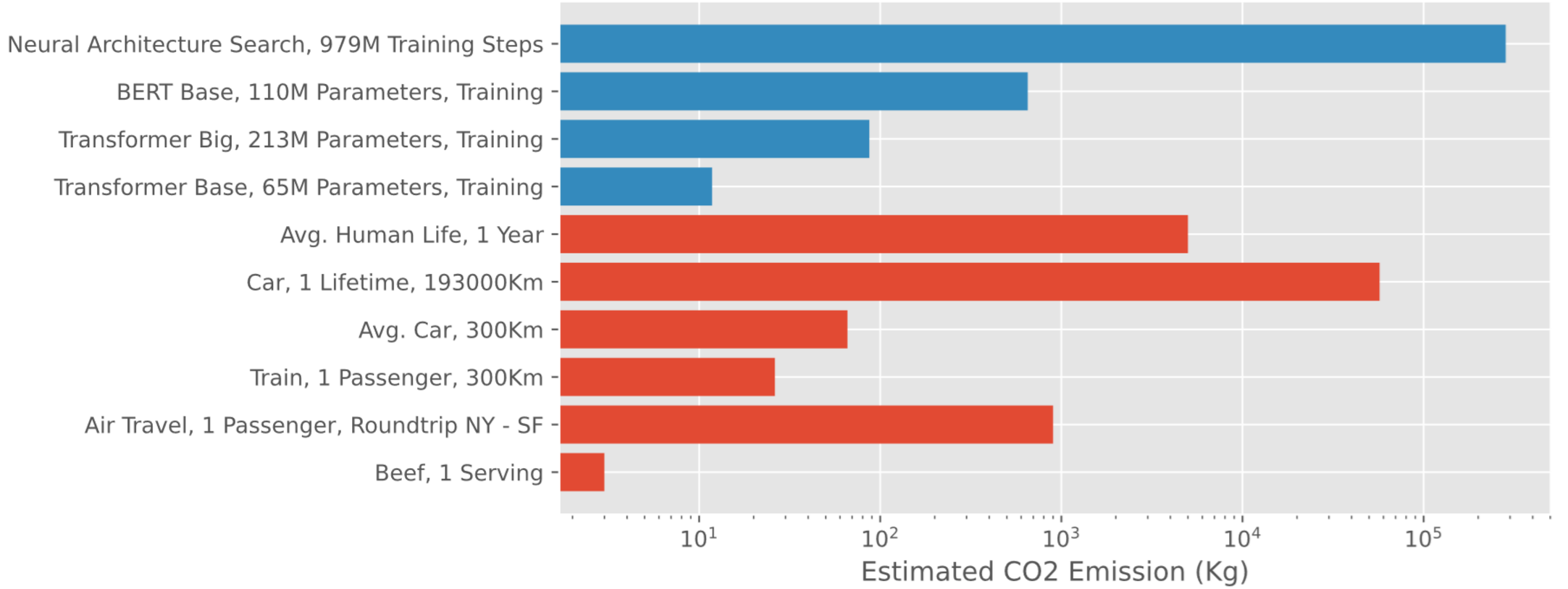
The *compute* explosion

Source:
Openai.com



Previous slide:

The *power and carbon emission* explosion



oil/gasoline: 1liter = 10kWh = 2.5 kg CO_2

- Source: E. Strubell et al., arXiv:1906.02243

Previous slide:

Energy consumption problem will further increase over time!

Solution? – use your machine carefully!

- think more, simulate less!
- invent better computer architecture!

Global warming is a reality!

→ Big migration waves to be expected

Solution? – tax on CO2

- reliable and predictable increase from 10cent to 10 dollars over 25 years.
- few countries start, others will follow

Previous slide:

Thanks!

The END

Exam:

- written exam: date fixed by SAC EPFL
- sample exams of previous years online
- miniproject counts 30 percent towards final grade

For written exam:

-bring 1 sheet A5 of own notes/summary
-HANDWRITTEN!
-no calculator, no textbook

1. Intro and Perceptrons

2. RL1: Reinforcement Learning and SARSA
3. RL2: TD learning, continuous space, eligibility traces
4. RL3: Policy gradient algorithms

5. DL1: BackProp and Regularization
6. DL2: Tricks of the Trade
7. DL3: Loss Landscape and optimization methods
8. DL4: Statistical Classification by Neural Networks
9. DL5: Convolutional Networks and NoFreeLunch

10. Deep RL1: DeepQ, Actor-Critic,
    Eligibility traces from Policy gradient, Model-based RL
11. Deep RL2: Discrete Games, Replay Buffer, and Continuous control
12. Deep RL3: Model-based Deep RL
13. Deep RL4: Biology and RL, three-factor rules
14. Deep RL5: Hardware

- Exam based on exercises and Quizzes.
- Clear questions/answers
- BackProp + RL

Feedback

Evaluation

Improvements

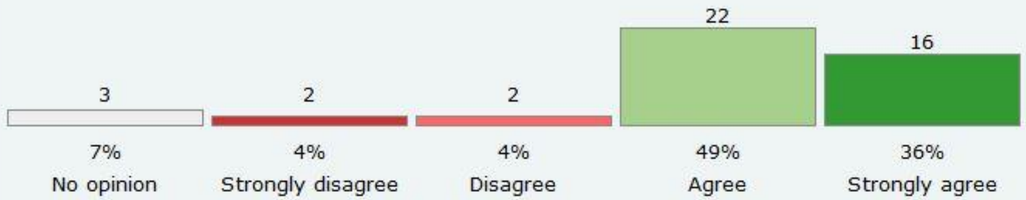
Old

2018

EPFL feedback by students: Overall I think this course is good

Year	2017-2018
Course	Artificial neural networks
Questionnaire	Evaluation indicative des enseignements (dès 2015-2016)
Nb Registered	144
Nb Answered	45

Overall, I think this course is good.



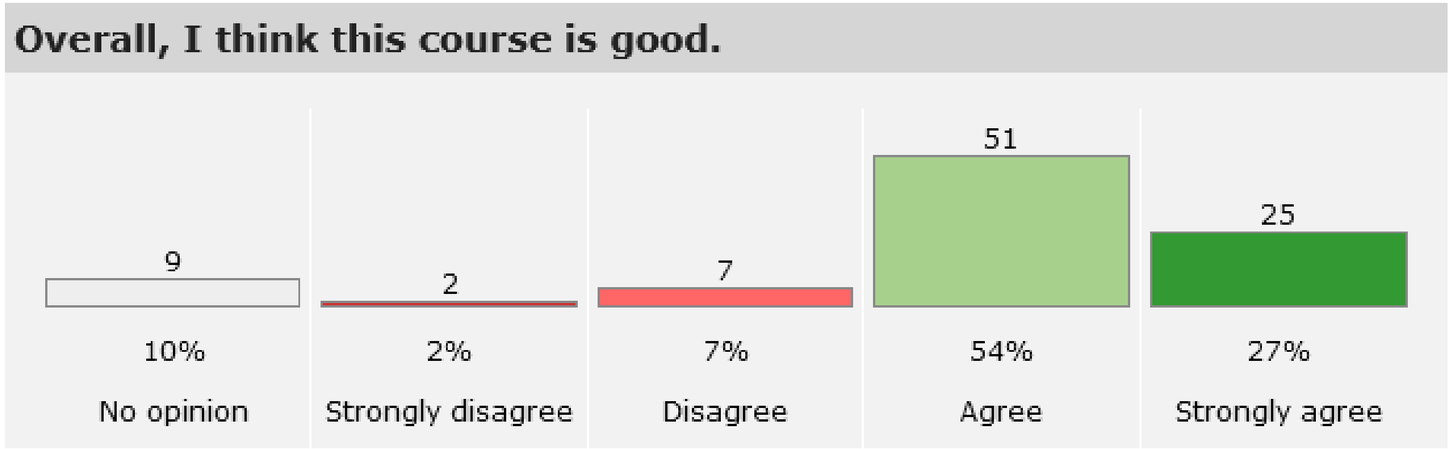
Response rate:
32 Percent
(45/144)

Thanks!

This year

2022

EPFL feedback by students: Overall I think this course is good



Response rate:
50 Percent
(94/197)

Thanks!

What can be improved next year ?

Main points:

- Overlap with other courses?
- Center even more on RL?
- Workload?
- Applied vs Formal Theory?
- More practical exercises?
- Support: Textbook, Slides, Video

What can be improved next year ?

Workload:

Lectures with integrated exercises take 120 min per week (=2h).

If I include everything (Lectures, exercises, miniproject, private reading), I have worked
PER WEEK (including the weeks of easter, ascension)

[ ] 5hours or less (< 3 hours in addition to lectures)
[ ] 5.5 - 8 hours (< 6 hours in addition to lectures)
[ ] > 8 hours (> 6 hours in addition to lectures)

What can be improved next year ?

- workload for a 5 credit course: 7.5 h p. week, for 18 weeks
  (includes 4 weeks of exam preparation time)

[1 credit = 27 hours work total = 1.5 h p. week, for 18 weeks]

Student in 2018

‘If I evaluate my worktime for this course I would say
1h minproject + 2h exercice + 2h lecture per week
which normally correspond to 5 ECTS credits and not 4.’

Support of Class: Textbook

How much time have you spent **during the term** with the textbook REINFORCEMENT LEARNING

[ ] 2 hours or less
[ ] 2-6 hours
[ ] more than 6 hours

IC evaluation 2018:
Question 4 – supporting material

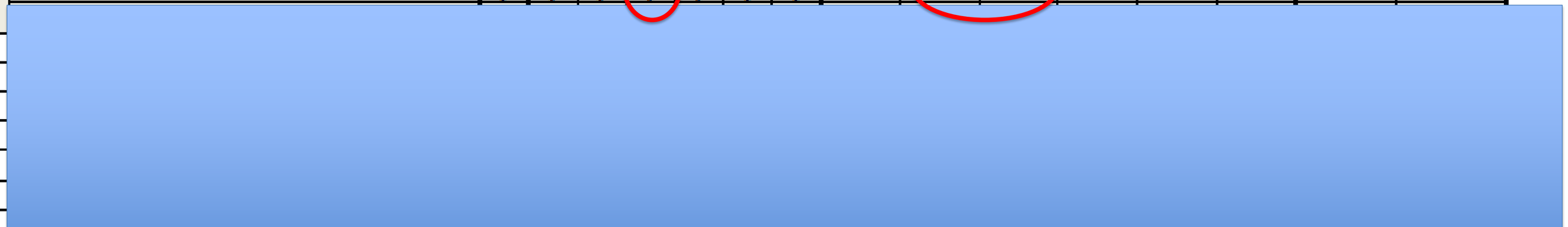
nication Sciences

rams

Master courses IC evaluation - 2nd and 4th semester

Academic year 2017-2018

		Questions														
		1	2	3	4	5	6	7	8						answers	registred
									less than 2 h	2h-4h	4h-6h	6h-8h	8h-10	over 10h		
		5,6	5,3	5,1	5,3	5,2	5,1	4,0	2	6	8	9	2	4	33	79
		5,6	5,6	5,1	5,4	5,6	5,7	5,4	0	3	7	6	4	1	23	33
		5,6	5,2	5,3	4,7	4,9	5,1	4,8	0	6	3	0	0	0	11	26
		5,8	5,5	5,0	5,2	5,5	5,0	4,1	0	0	0	2	5	6	14	39
		5,5	5,0	5,3	5,3	5,5	5,0	4,3	0	0	2	1	1	0	4	9
		5,8	5,6	4,8	5,2	6,0	5,2	5,6	0	2	2	0	1	0	5	14
	Artificial neural networks	5,2	5,5	5,2	4,5	5,3	5,1	5,2	2	9	19	3	3	0	38	76



What can be improved next year ?

‘The slides are not as good as in other courses, they should be improved.’

Agree – not sure - disagree

‘The slides do not contain enough text, just sketches’

A) When I look at the slides 4 weeks after the lecture, the slides are useful

B) When I sit in the course on Friday, the slides are useful

What can be improved next year ?

‘I also took the Deep Learning course
and the two are very complementary
despite the overlaps’

Overlap with ‘Deep Learning’.
Should we recommend that students take both classes:

[ ] Yes, very complementary
[ ] not sure
[ ] too much overlap, advice students against taking both

What can be improved next year ?

‘Overlap with Unsupervised and Reinforcement Learning’

Overlap with ‘Unsupervised and Reinforcement Learning’. Should we recommend that students take both classes:

[ ] Yes, very complementary (not more than 3 sessions overlap)
[ ] not sure
[ ] too much overlap, advice students against taking both

What can be improved next year ?

‘The lecture hall is bad for reading the blackboard’

‘The blackboard parts break the rhythm of the lecture.’

‘The blackboard parts should be handed out’

Blackboard part:
Like it – not sure – hate it

Blackboard part, but done on tablet:
Like it – not sure – hate it

What can be improved next year ?

‘The in-class exercises are a waste of time”

In-class exercises:
Like it – not sure – hate it

1. Intro and Perceptrons

2. RL1: Reinforcement Learning and SARSA
3. RL2: TD learning, continuous space, eligibility traces
4. RL3: Policy gradient algorithms

5. DL1: BackProp and Regularization
6. DL2: Tricks of the Trade
7. DL3: Loss Landscape and optimization methods
8. DL4: Statistical Classification by Neural Networks
9. DL5: Convolutional Networks and NoFreeLunch

10. Deep RL1: DeepQ, Actor-Critic,
    Eligibility traces from Policy gradient, Model-based RL
11. Deep RL2: Discrete Games, Replay Buffer, and Continuous control
12. Deep RL3: Model-based Deep RL
13. Deep RL4: Biology and RL, three-factor rules
14. Deep RL5: Hardware

15. Deep RL6: Exploration by Novelty/Surprise/InformationGain