



UNIVERSITY
OF TRENTO - Italy
Department of Physics

1st Summer School of

Interdisciplinary Research on
Brain Network Dynamics

June 24-28, 2019

Analog Crossbar Arrays – Future Neuromorphic Workhorses for Neural Networks

Tutorial

Roger Dangel, PhD

Neuromorphic Devices and Systems Group

IBM Zurich GmbH

June 24, 2019



IBM Zurich: Neuromorphic Devices & Systems Group



**Research Staff
Members (PhDs)**



**Group leader
Bert Offrein**

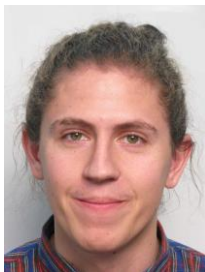
Post-Docs



Engineers



Pre-Docs



Outlook

- Introduction:
- Experiment: Human brain against computer
 - Conceptual comparison: Brain vs. computer
 - What makes the human brain so outstanding? – Can we mimic it ?

- Part1:
- Neuromorphic computing – what is it?
 - Neuromorphic tasks in AI
 - Anatomy of computational “heavy” workloads – the actual problem
 - Computational challenge: Matrix-vector multiplications
 - Current and future AI acceleration hardware

- Part 2:
- From brain-like to Deep Neural Networks (DNNs)
 - Training of DNN with backpropagation algorithm - Mathematical background
 - Status of today’s Deep Neural Network processing

- Part 3:
- Analog electrical crossbar array vs. DNN
 - Synaptic weight processing operations
 - Targeted device properties for analog electrical crossbar arrays

- Part 4:
- Memristive devices for synaptic weight implementation
 - Examples: Resistive Random Access Memory (ReRAM)
Phase Change Memory (PCM)
Ferroelectric Tunneling Junctions (FTJ)

Summary



Keywords of this Tutorial

Neuromorphic computing

Human brain

(Non) von-Neumann architecture

Deep Neural Networks

Analog vs. digital processing

Matrix-vector multiplication

Backpropagation algorithm

CPU / GPU / FPGA / ASIC

Accelerators

Training / Inference

Memristive devices

Non-volatile memory

Synaptic weights

Crossbar arrays

Multiply & accumulate

PCM / ReRAM / FTJ

Experiment: “Human Brain against Computer”

Task 1: Mathematics

Brain

$$\sqrt{2} = ?$$



Computer

$$\sqrt{2} = ?$$



Experiment: “Human Brain against Computer”

Task 1: Mathematics

Brain

Iterative approximation x_n of square root \sqrt{Z}

$$\sqrt{Z} \approx x_n \quad x_{n+1} = \frac{1}{2} \left(x_n + \frac{Z}{x_n} \right) \quad \text{HERON method}$$

Example: $Z=2$

$x_0 = 1$ (start value)

$$x_1 = \frac{1}{2} \left(x_0 + \frac{Z}{x_0} \right) = \frac{1}{2} \left(1 + 2 \right) = \frac{3}{2} = 1.5$$

$$x_2 = \frac{1}{2} \left(x_1 + \frac{Z}{x_1} \right) = \frac{1}{2} \left(\frac{3}{2} + \frac{2}{3/2} \right) = \frac{1}{2} \left(\frac{3}{2} + \frac{4}{3} \right) = \frac{1}{2} \left(\frac{9+8}{6} \right) = \frac{17}{12} = 1.41\bar{6}$$

$$x_3 = \frac{1}{2} \left(x_2 + \frac{Z}{x_2} \right) = \frac{1}{2} \left(\frac{17}{12} + \frac{2}{17/12} \right) = \frac{1}{2} \left(\frac{17}{12} + \frac{24}{17} \right) = \frac{1}{2} \left(\frac{289+288}{204} \right) = \frac{577}{408} = 1.4142$$

$$17 : 12 = 1.41\bar{6}$$

$$12$$

$$50$$

$$48$$

$$20$$

$$12$$

$$80$$

$$72$$

$$80$$

$$577 : 408 = 1.4142$$

$$408$$

$$1690$$

$$1632$$

$$550$$

$$408$$

$$1720$$

$$1632$$

$$880$$



Roger Dangel: Result obtained in ≈ 10 min:

$$\sqrt{2} \approx 1.4142$$

Computer



Computer: Result obtained in $\ll 1$ sec:

1.414213562373095048801688724209698078569
671875376948073176679737990732478462107038850387
5343276415727350138462309122970249248360558507372126441
2149709993583141322266592750559275579995050115278206057147010
9559971605970274534596862014728517418640889198609552329230484308714321450
839762603627995251407989687253396546331808829640620615258352395054745750287759961
729835575220337531857011354374603408498847160386899970699004815030544027790316454247823068
492936918621580578463111596668713013015618568987273235288509264861249497715421833420428568
60601468247207714358548741554595147226722926124850696173163809410821860045286102696547576...



Experiment: “Human Brain against Computer”

Task 2: Image recognition

Image to be recognized



Brain

What does the image show?



Computer

What does the image show?



Experiment: “Human Brain against Computer”

Task 1: Mathematics

Brain

Iterative approximations x_n of square root $\sqrt{2}$

$$\sqrt{2} \approx x_n \quad x_{n+1} = \frac{1}{2} \left(x_n + \frac{2}{x_n} \right) \quad \text{HERON method}$$

Example: $Z=2$

$x_0 = 1$ (start value)

$$x_1 = \frac{1}{2} \left(x_0 + \frac{2}{x_0} \right) = \frac{1}{2} \left(1 + 2 \right) = \frac{3}{2} = 1.5$$

$$x_2 = \frac{1}{2} \left(x_1 + \frac{2}{x_1} \right) = \frac{1}{2} \left(\frac{3}{2} + \frac{4}{3} \right) = \frac{1}{2} \left(\frac{9+8}{6} \right) = \frac{17}{12} \approx 1.41\bar{6}$$

$$x_3 = \frac{1}{2} \left(x_2 + \frac{2}{x_2} \right) = \frac{1}{2} \left(\frac{17}{12} + \frac{24}{17} \right) = \frac{1}{2} \left(\frac{289+288}{204} \right) = \frac{577}{408} \approx 1.4142$$

$$17 \cdot 12 = 204$$

$$17$$

$$12$$

$$204$$

$$17$$

$$12$$

$$204$$

$$17$$

$$12$$

$$204$$

$$577 \cdot 408 = 235416$$

$$577$$

$$408$$

$$235416$$

$$577$$

$$408$$

$$235416$$

$$577$$

$$408$$

$$235416$$



Roger Dangel: Result obtained in ≈ 10 min:

$$\sqrt{2} \approx 1.4142$$

Computer



Computer: Result obtained in $\ll 1$ sec:

1.414213562373095048801688724209698078569
67187537694807317667973799073247846210703880387
5343276415727350138462309122970249248360558507372126441
2149709993583141322266592750559275579995050115278206037147010
9559971605970274534596862014728517418640889198609552329230484308714321450
839762603627995251407989687253396546331808829640620615258352395054745750287759961
729835575220337531857011354374603408498847160386899970699004815030544027790316454247823068
492936918621580578463111596668713013015618568987237235288509264861249497715421833420428568
60601468247207714358548741554595147226722926124850696173163809410821860045286102696547576...



Task 2: Image recognition



Roger Dangel: Result obtained in < 1 sec:

Group of ring-tailed lemurs eating fruits



Computer: not able to solve this task (yet)



Experiment: “Human Brain against Computer”

Task 1: Mathematics

Brain

Iterative approximation x_n of squareroot \sqrt{Z}

$$\sqrt{Z} \approx x_n \quad x_{n+1} = \frac{1}{2} \left(x_n + \frac{Z}{x_n} \right) \quad \text{HERON method}$$

Example: $Z=2$

$x_0 := 1$ (start value)

$$x_1 = \frac{1}{2} \left(x_0 + \frac{Z}{x_0} \right) = \frac{1}{2} \left(1 + 2 \right) = \frac{3}{2} = 1.5$$

$$x_2 = \frac{1}{2} \left(x_1 + \frac{Z}{x_1} \right) = \frac{1}{2} \left(\frac{3}{2} + \frac{2}{3} \right) = \frac{1}{2} \left(\frac{9+8}{6} \right) = \frac{17}{12} = 1.41\bar{6}$$

$$x_3 = \frac{1}{2} \left(x_2 + \frac{Z}{x_2} \right) = \frac{1}{2} \left(\frac{17}{12} + \frac{2}{17/12} \right) = \frac{1}{2} \left(\frac{17}{12} + \frac{24}{17} \right) = \frac{1}{2} \left(\frac{289+288}{204} \right) = \frac{577}{408} = 1.4142$$

17.12 = 1.416
12
50
48
20
12
80
72
80

577 : 408 = 1.4142
408
1690
1632
580
408
1720
1632
880



Roger Dangel: Result obtained in ≈ 10 min:

$$\sqrt{2} \approx 1.4142$$

Task 2: Image recognition



Roger Dangel: Result obtained in < 1 sec:

Group of ring-tailed lemurs eating fruits

Computer



Computer: Result obtained in $<< 1$ sec:



1.414213562373095048801688724209698078569
671875376948073176679737990732478462107038860387
5343276415727350138462309122970249248360558507372136441
2149709993583141322266592750559275579995050115278206037147016
9559971605970274534596862014728517418640889198609552329230484308714321450
839762603627995251407989687253396546331808829640620615258352395054745750287759961
72983557520337531857011354374603408498847160386899970699004815030544027790316454247823068
492936918621580578463111596668713013015618568987237235288509264861249497715421833420428568
60601468247207714358548741554595147226722926124850696173163809410821860045286102696547576...



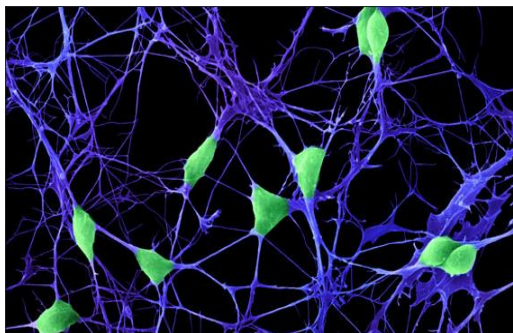
Computer: not able to solve this task (yet)



What makes the Human Brain so Outstanding ?

- **Power efficiency:** human brain $\leftrightarrow \approx 20$ Watts / supercomputers \leftrightarrow up to MWatts
- Brain **recognizes patterns and images** / **can deduce facts from raw (noisy) data**

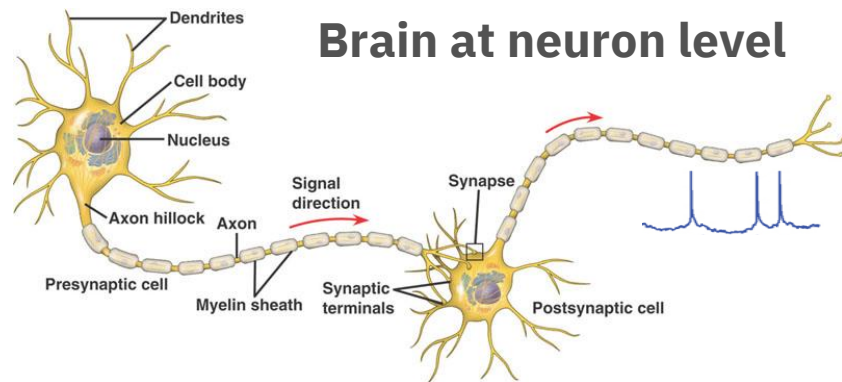
Brain at neural network level



 <http://www.sciencephoto.com/dennis-kunkel-microscopy-collection>

- Human brain: ≈ 100 billions nerve cells (= **neurons**)
- Each neuron receives signals from **1'000 – 10'000** other neurons via synapses \rightarrow **massive connectivity**
- Signals transmitted by synapses are adjustable:
 \rightarrow **“synaptic weight”**

Brain at neuron level



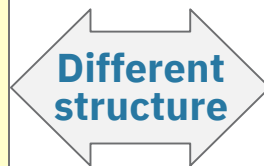
 <http://biomedicalengineering.yolasite.com/neurons.php>

- Signaling between neurons: Spikes, spike trains
- Neuron activation: “Integrate and Fire”
- Learning: Adjustment of the synaptic weights
Spike Timing Dependent Plasticity:
“Neurons that fire together wire together”

Conceptual Comparison: “Human Brain vs. Computer”



- Nerve cells (**neurons**) are processing units
- **Analog** operation
- **Distributed** processor and memory
- Massively, massively **parallel** processing
- **Slow** information processing
- **Redundancy** and **fault-tolerance** properties
-



- **Transistors** are processing units
- **Digital** operation
- **Centralized** processor and memory
- (Mostly) **serial** processing
- **Very fast** information processing
- **Reliable** and **precise**
-

Question: **Can we mimic the human brain to exploit its superiority in certain applications ?**



Outlook

- Part1:
- Neuromorphic computing – what is it?
 - Neuromorphic tasks in AI
 - Anatomy of computational “heavy” workloads – the actual problem
 - Computational challenge: Matrix-vector multiplications
 - Current and future AI acceleration hardware




Neuromorphic Computing – What is it ?

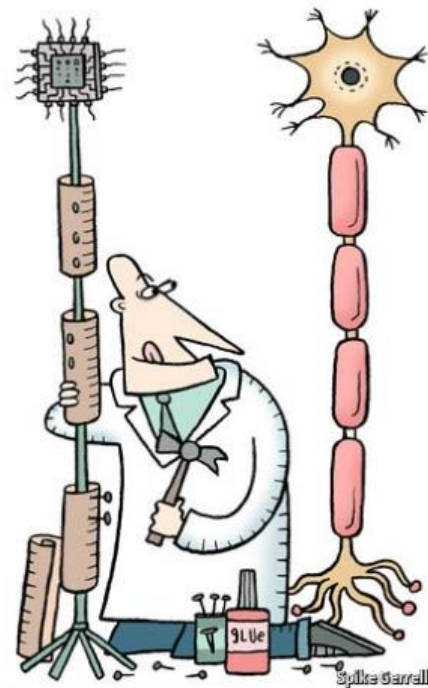
Ethymological: “**neuro**” \Leftrightarrow related to nerves or nervous system
 “**morphic**” \Leftrightarrow having form or structure of...

Definition: Neuromorphic computing is a **brain-inspired signal processing technology** that tries to **mimic** the **neuro-biological architecture** of the **brain and its functions**.

As interdisciplinary technology, it involves

- biological,
 - physical,
 - mathematical,
 - computer science,
 - and electronic engineering concepts
- to design and realize new artificial neural network systems.

 <http://www.web3.lu/category/science-philosophy/>



Neuromorphic Tasks in AI

Neuromorphic challenges in AI are tasks which normally require human “intellect”, e.g.:

- Memorizing complex information
- Deducing facts from raw (unstructured) Data
- Making recommendations and decisions

in the presence of
uncertainty and ambiguity

Elevator Control:

Deduce Facts from Data:

- floor-to/floor button

Make Decisions:

- stop at floor(s)

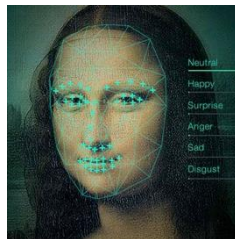
Memorizing Information:

- remember pending call



Detect or Extract

Emotions



Anomalies

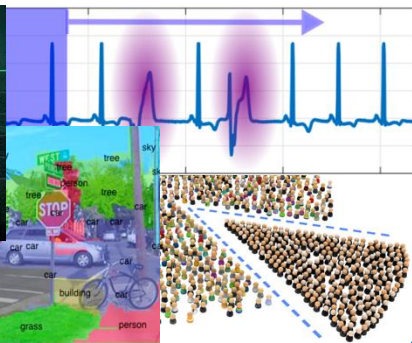
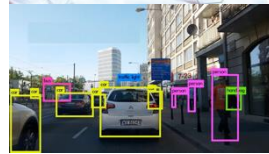


Image Classes

Segments

Act

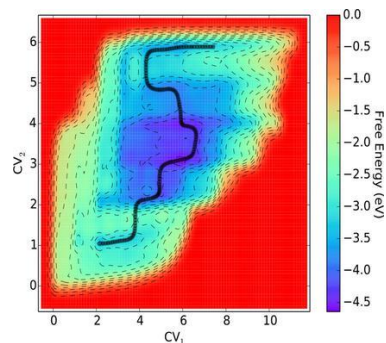
Recommendations



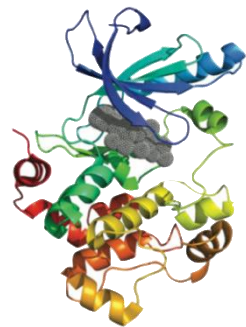
Autonomous Decisions

Anatomy of “Heavy” Computational Workloads – The Actual Problem

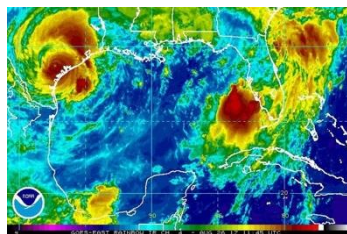
Scientific Workloads



(electro) Chemistry



Drug Discovery



Weather/Climate

PDEs

$$\frac{\partial^2 u_1}{\partial x_1^2} + \frac{\partial^2 u_2}{\partial x_2 \partial x_1} + \frac{\partial^2 u_1}{\partial x_1 \partial x_2} + \frac{\partial^2 u_2}{\partial x_2^2} + \frac{\partial^2 u_1}{\partial x_1 \partial x_3} + \frac{\partial^2 u_2}{\partial x_2 \partial x_3} + \frac{\partial^2 u_3}{\partial x_3^2} = 0$$

Backpropagation

$$x_{i,j}^l = \sum_m \sum_n w_{m,n}^l o_{i+m,j+n}^{l-1} + b_{i,j}^l$$

$$o_{i,j}^l = f(x_{i,j}^l)$$

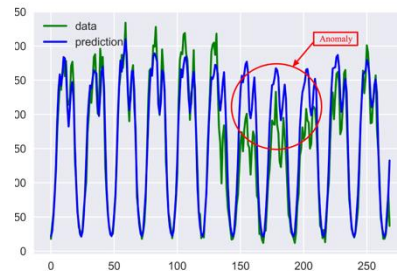
$$\delta_{i,j}^l = \frac{\partial E}{\partial x_{i,j}^l}$$

$$\frac{\partial E}{\partial x_{i,j}^l} = \sum_{m=0}^{k_l-1} \sum_{n=0}^{k_l-1} \delta_{i-m,j-n}^{l+1} w_{m,n}^{l+1} f'(x_{i,j}^l)$$

$$\frac{\partial E}{\partial w_{m',n'}^l} = \sum_{i=0}^{B-k_l} \sum_{j=0}^{W-k_l} \delta_{i,j}^l o_{i+m',j+n'}^{l-1}$$

$f_1 = 0$
 $f_2 = 0$
 $f_3 = 0$

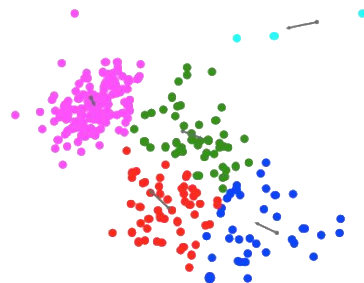
AI / Machine Learning



Time-Series Predictions



Graph Analytics



Clustering Algorithms

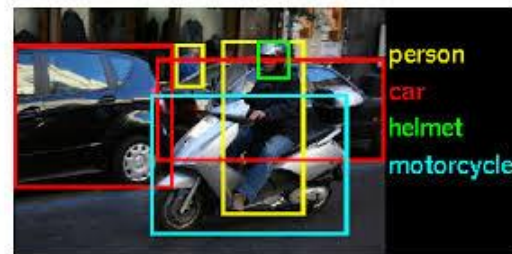
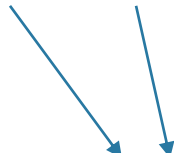


Image Classification

Computational Challenge: Matrix-Vector Multiplications

Matrix-vector multiplications of the form


$$\mathbf{W}\mathbf{x} = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,2} & \dots & w_{0,N} \\ w_{1,0} & w_{1,1} & w_{1,2} & \dots & w_{1,N} \\ w_{2,0} & w_{2,1} & w_{2,2} & \dots & w_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{M,0} & w_{M,1} & w_{M,2} & \dots & w_{M,N} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^N w_{0,i} x_i \\ \sum_{i=0}^N w_{1,i} x_i \\ \sum_{i=0}^N w_{2,i} x_i \\ \vdots \\ \sum_{i=0}^N w_{M,i} x_i \end{bmatrix}$$

are common to the mentioned workloads and **dominate** the computation time and energy consumption.

Matrix-vector multiplications are “computationally expensive” !

Our mission

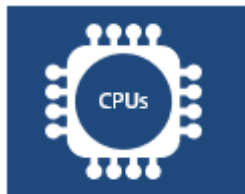
Develop dedicated hardware (→ Analog Crossbar Arrays) which enables **efficient analog implementation of matrix-vector multiplications** and therefore acceleration of Deep Neural Network Learning

Current and Future AI Acceleration Hardware



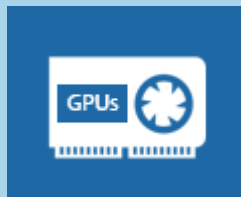
C o - P r o c e s s o r s

Central Processing Unit (CPU)



- CPUs were originally designed for general computing workloads

Graphics Processing Unit (GPU)



Current Workhorse

- GPUs were originally developed for manipulation of images which relies on similar mathematical basis than neural networks
- GPUs operate on vectors of data in parallel
- GPUs are effective at processing same set of operations in parallel (single instruction, multiple data (SIMD))
- GPUs have well-defined instruction-set and fixed data width

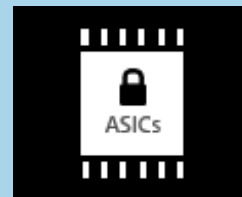
Field-Programmable Gate-Array (FPGA)



Limited spread

- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects
- FPGAs are reconfigurable, what makes evolution of hardware, framework and software easier
- FPGAs are effective at processing same or different set of operations in parallel (multiple instructions, multiple data (MIMD))
- FPGAs do not have predefined instruction-set or fixed data width.

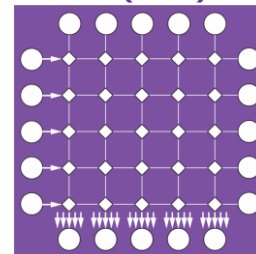
Application-Specific Integrated Circuit (ASIC)



Under investigation

- ASICs are application-specifically designed hardware
- ASICs employ special strategies, e.g. optimized memory use or use of lower precision arithmetics

Resistive Processing Unit (RPU)



based on
Analog Crossbar Arrays

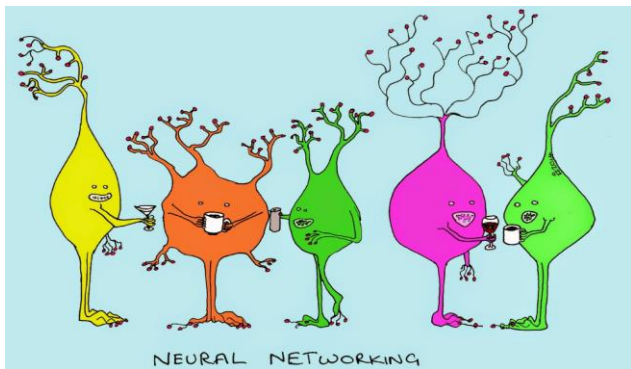
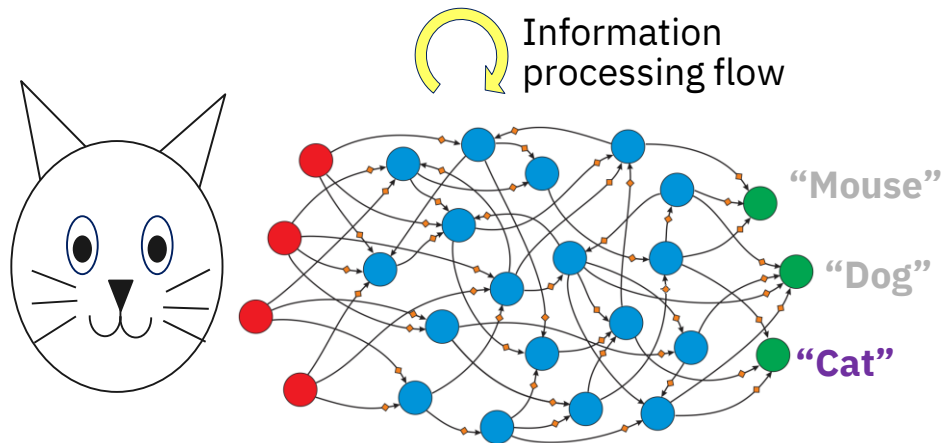
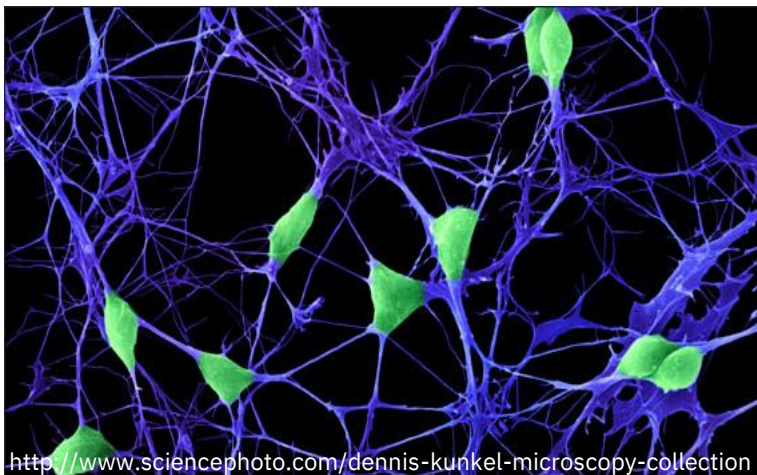


Outlook

- Part 2:
- From brain-like to Deep Neural Networks (DNNs)
 - Training of DNN with backpropagation algorithm – Mathematical background
 - Status of today's Deep Neural Network processing



From Brain to Brain-like Neural Network



- Omni-directional signal flow
- A-synchronous pulse signals
- Information encoded in signal timing

➡ **Difficult to implement efficiently on standard computer hardware**

Artificial Neural Network

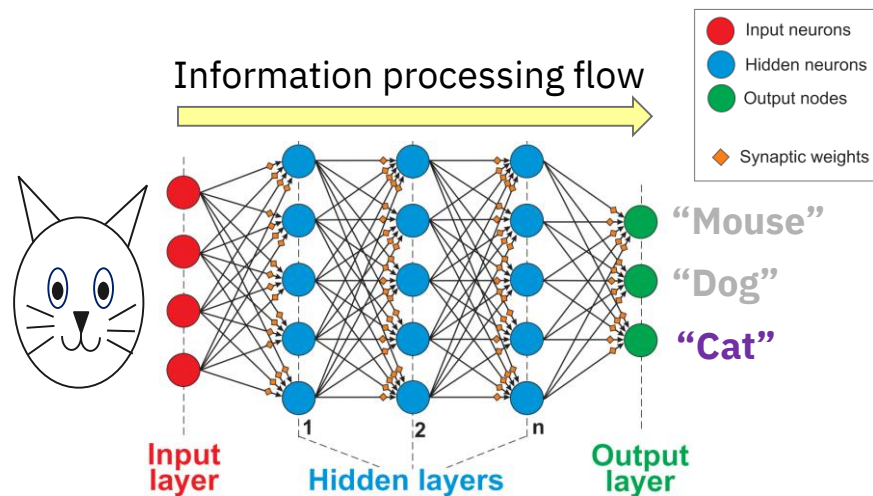
Brain-like neural network

Simplified model

Artificial Neural Network (ANN)

- ANNs are neuromorphic computing models, which mimic the brain **in a simplified way**.
- ANNs are composed of **multiple nodes (= artificial neurons)** which can be arranged in special configurations
- The first developed, easiest and most common ANN is the:

Feed-forward Deep Neural Network (DNN)



DNN better fit to standard hardware

- Feed-forward sequential processing
- Information encoded in signal amplitude
- Neuron activation: Weighted sum + Threshold
- Training with “Backpropagation algorithm”



Operation Phases of Deep Neural Network (DNN)

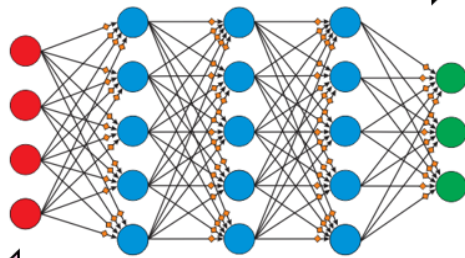
Phase 1

very large number of **known** samples

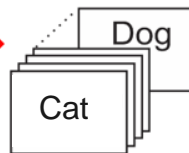


Training/
Learning

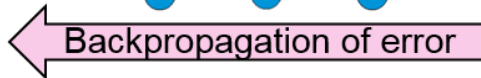
Forward propagation



answer with error



Backpropagation of error



Computational speed and efficiency are extremely important because **training** of Deep Neural Networks **can range from days to weeks** (even with high-performance computers) !

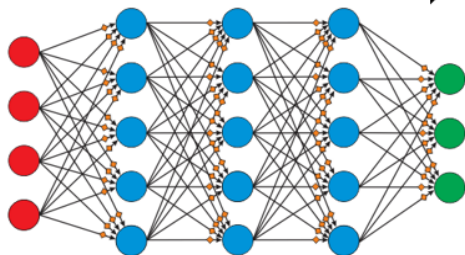
Phase 2

smaller number of **unknown** samples

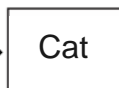


Inference/Use

Only forward propagation



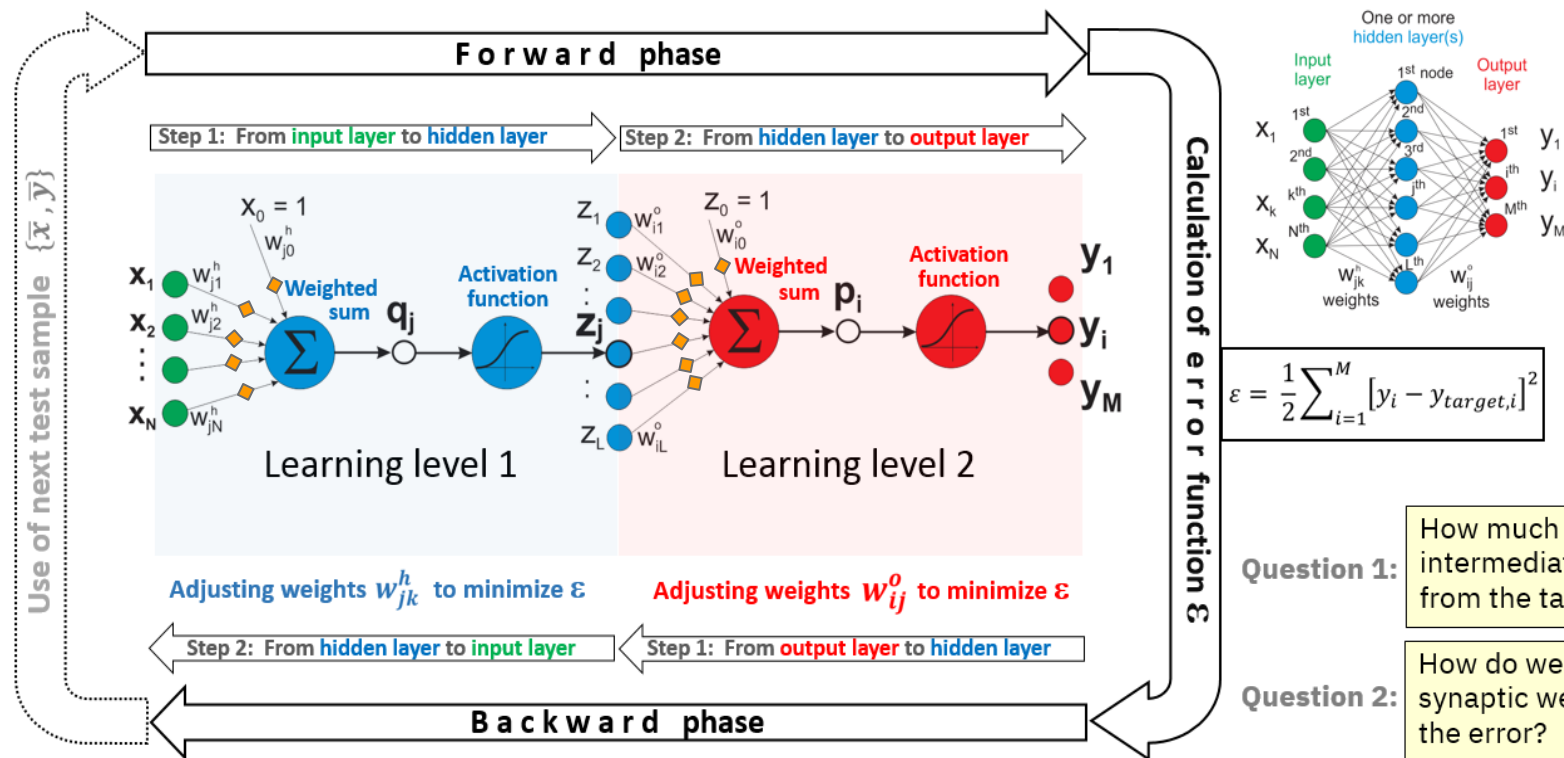
correct answer



DNN Training by Backpropagation Algorithm

(1 of 3)

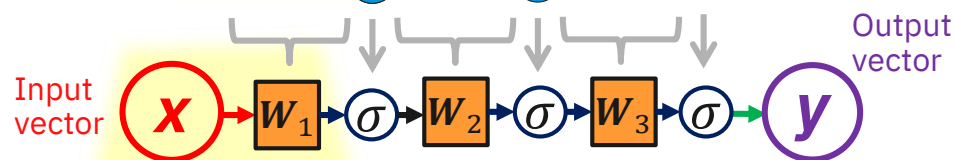
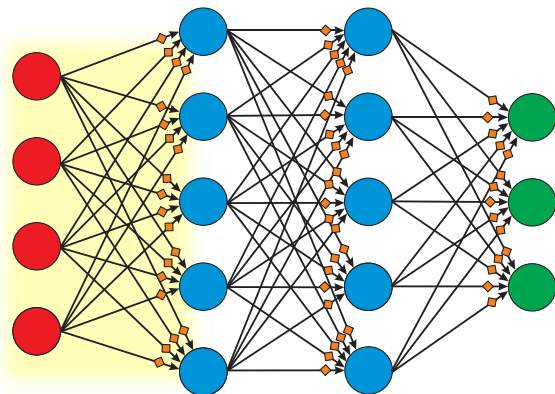
Generic scheme for iterative error minimization by adjusting the synaptic weights



DNN Training by Backpropagation Algorithm

(2 of 3)

Neural net as chain of vector operations:



$$W_1 x = \begin{bmatrix} w_{1,0,0} & w_{1,0,1} & \dots & w_{1,0,N} \\ w_{1,1,0} & w_{1,1,1} & \dots & w_{1,1,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,M,0} & w_{1,M,1} & \dots & w_{1,M,N} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^N w_{1,0,i} x_i \\ \sum_{i=0}^N w_{1,1,i} x_i \\ \vdots \\ \sum_{i=0}^N w_{1,M,i} x_i \end{bmatrix}$$

Accumulate
Multiply

Components:

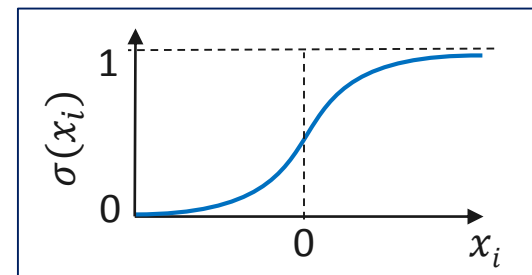
- Layers of neurons ● ● ● ●
- Synaptic interconnections → ◆ →

Mathematical operations:

→ : Signal vector

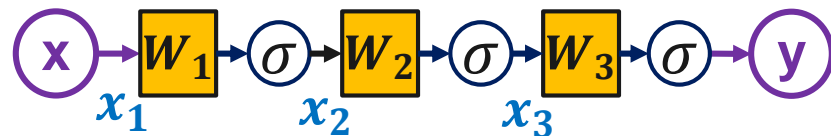
W_n : Synaptic weight matrix $[W_n]$

σ : Per-element neural (non-linear) activation function (sigmoid):



For many training case inputs \mathbf{X} with target response $\mathbf{y}_{\text{target}}$:

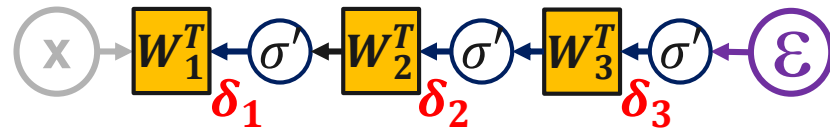
- ① **Forward Propagate:** Input $\mathbf{X} \rightarrow$ Response \mathbf{y}
Store neuron activation patterns \mathbf{x}_i for later use



- ② **Determine output error \mathcal{E} :**

$$\mathcal{E} = \frac{1}{2} \sum_i [y_i - y_{\text{target}_i}]^2$$

- ③ **Backward Propagate:** Which neuron inputs have strongest influence on \mathcal{E} ?
 \rightarrow Error gradient vectors δ_j



- ④ **Adjust weights** that were active ($\propto \mathbf{x}$), proportionally to their influence on error \mathcal{E} ($\propto \delta$):

$$\Delta W = -\eta \mathbf{x} \otimes \delta$$

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



Status of Today's Deep Neural Network Processing

Current
situation

- Processing dominated by large matrix operations

Forward propagation:

Backward propagation:

Weight update:



Scale $\propto N^2$

Neurons/layer

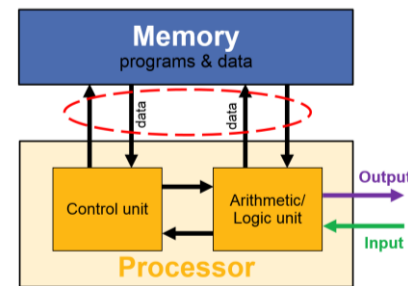
- Large training datasets: Thousands of training cases

- Inefficient on standard Von-Neumann architecture systems:

- (Mostly) serial processing
- Low computation to IO ratio
- → Memory bottleneck



High performance computer



Today's standard computer architecture
(→ proposal by John Von-Neumann in 1945)

Need for faster
and more efficient
DNN processing

Borrow some concepts from the brain:

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

Analog Crossbar Arrays

Outlook

- Part 3:
- Analog electrical crossbar array vs. DNN
 - Synaptic weight processing operations

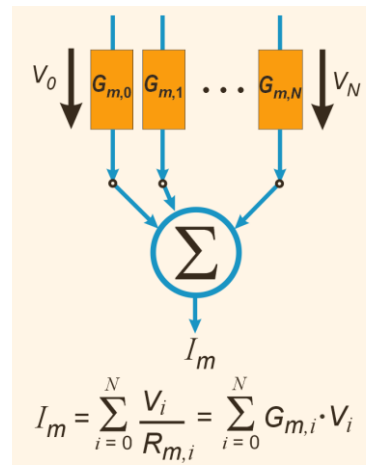


Analog Electrical Crossbar Array

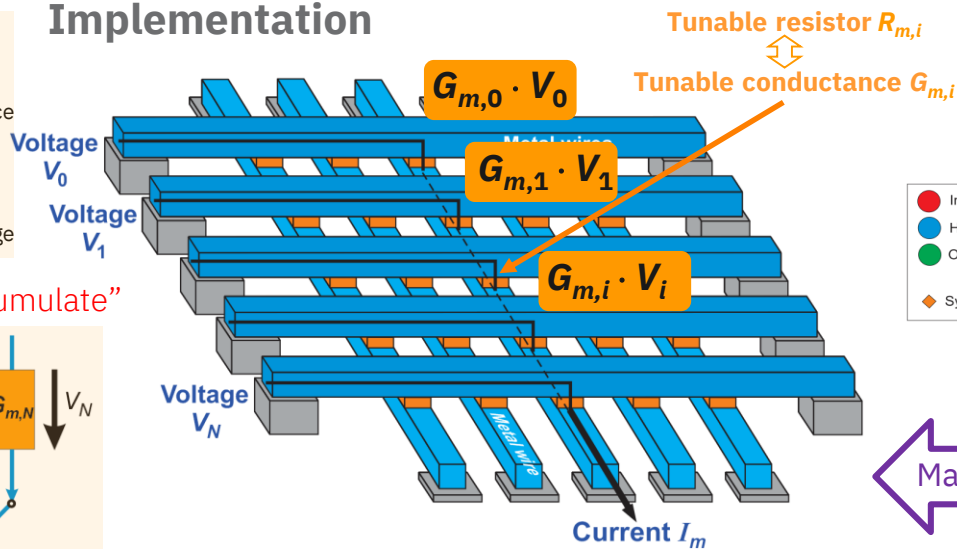
Conductance $G = \frac{1}{R}$
Resistance

Current $I = \frac{V}{R} = G \cdot V$
Voltage

“Multiply and Accumulate”

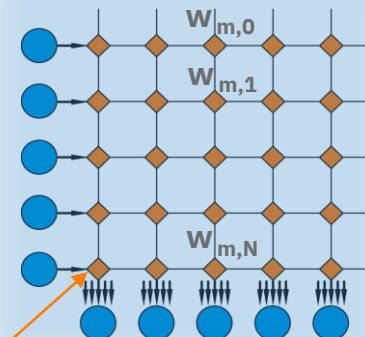
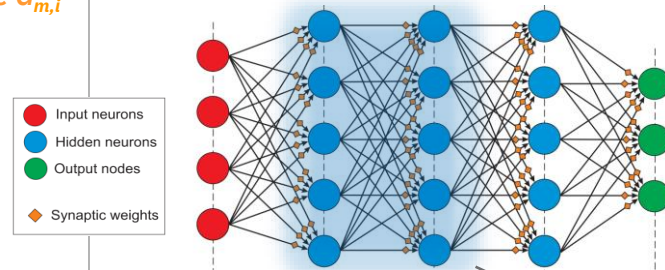


Implementation



$$\begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \vdots \\ I_M \end{bmatrix} = \begin{bmatrix} G_{0,0} & G_{0,1} & G_{0,2} & \dots & G_{0,N} \\ G_{1,0} & G_{1,1} & G_{1,2} & \dots & G_{1,N} \\ G_{2,0} & G_{2,1} & G_{2,2} & \dots & G_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ G_{M,0} & G_{M,1} & G_{M,2} & \dots & G_{M,N} \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ \vdots \\ V_N \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^N G_{0,i} V_i \\ \sum_{i=0}^N G_{1,i} V_i \\ \sum_{i=0}^N G_{2,i} V_i \\ \vdots \\ \sum_{i=0}^N G_{M,i} V_i \end{bmatrix}$$

Feedforward DNN w/ fully connected neural layers



Memristor

Synaptic weight

Cross-point

Synaptic weight

Synaptic Weight Processing Operations

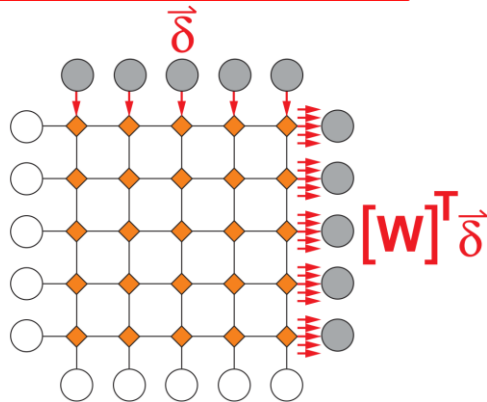
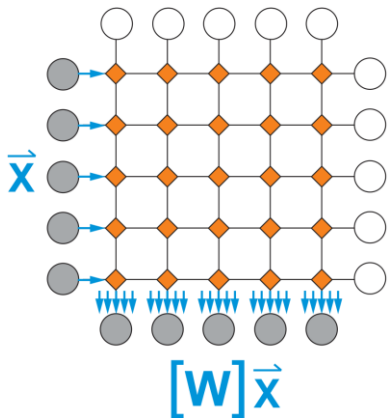
Forward propagation



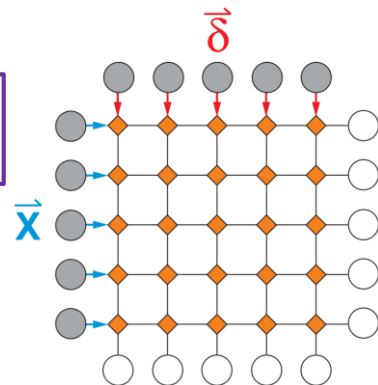
Backward propagation



Synaptic weight update



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



\vec{x} Input vector
 $[W]$ Weight matrix
 $[W]^T$ Transposed weight matrix

Challenge

Update must be proportional to signals on rows ($\propto x_i$) and on columns ($\propto \delta_j$)

- **Symmetric** increase and decrease of weight
- **Analog behavior:** > 100 levels preferred (ca. 8 bit)

Outlook

- Part 4:
- Targeted device properties for analog electrical crossbar arrays
 - Memristive devices for synaptic weight implementation
 - Examples: Resistive Random Access Memory (ReRAM)
Phase Change Memory (PCM)
Ferroelectric Tunneling Junctions (FTJ)



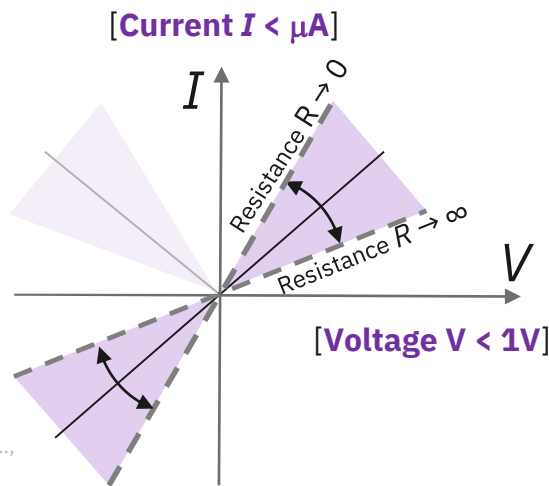
Targeted Device Properties for Analog Electrical Crossbar Arrays

Our Dream-Device:

- CMOS compatibility
- Low voltage operation
- Small device footprint
- Very short (re-)set time
- Long retention time (\leftrightarrow NVM)
- Low drift
- High dynamic range
- Large resistance range (high-resistance \rightarrow low power)
- Reproducibility, low variability
- (Some) linearity & symmetry

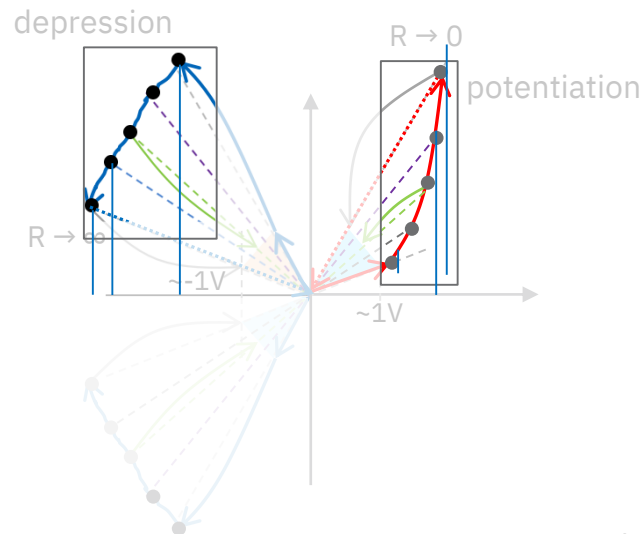
Gokmen & Vlasov, Acceleration of Deep NN Training..., *Frontiers in Neuroscience*, 2016

Operation:



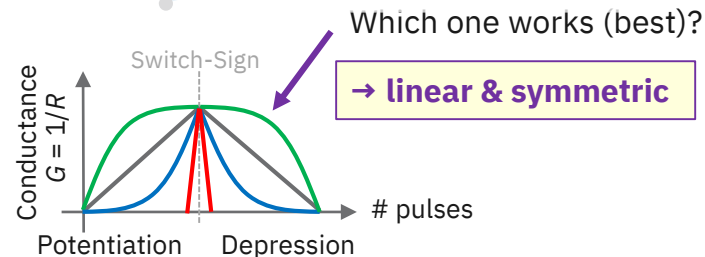
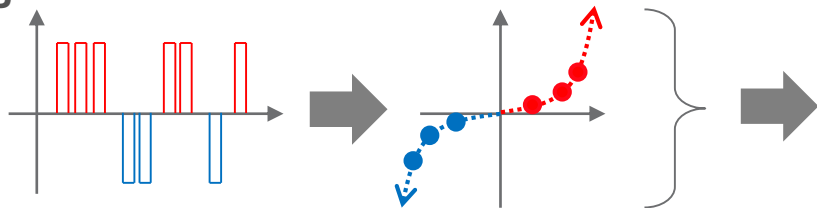
“Programming” Resistance:

(representative, generic characteristic)



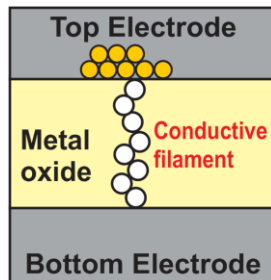
Programming Scheme:

Pulse Encoding
(Incremental)

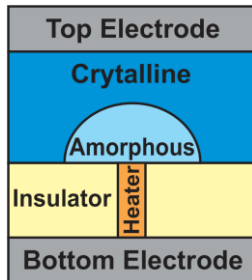


Memristive Devices for Synaptic Weight Implementation

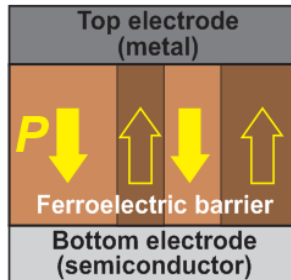
ReRAM



PCM

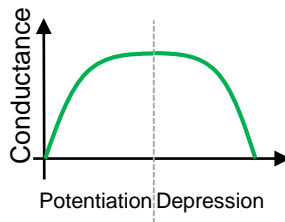
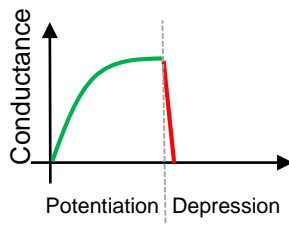
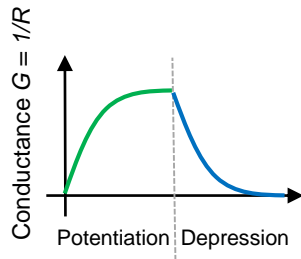


FTJ



Others:

- **FeRAM** (Ferro-Electric RAM)
- **MRAM** (Magnetic RAM)
- **ECRAM** (Electro-Chemical RAM)
- ⋮

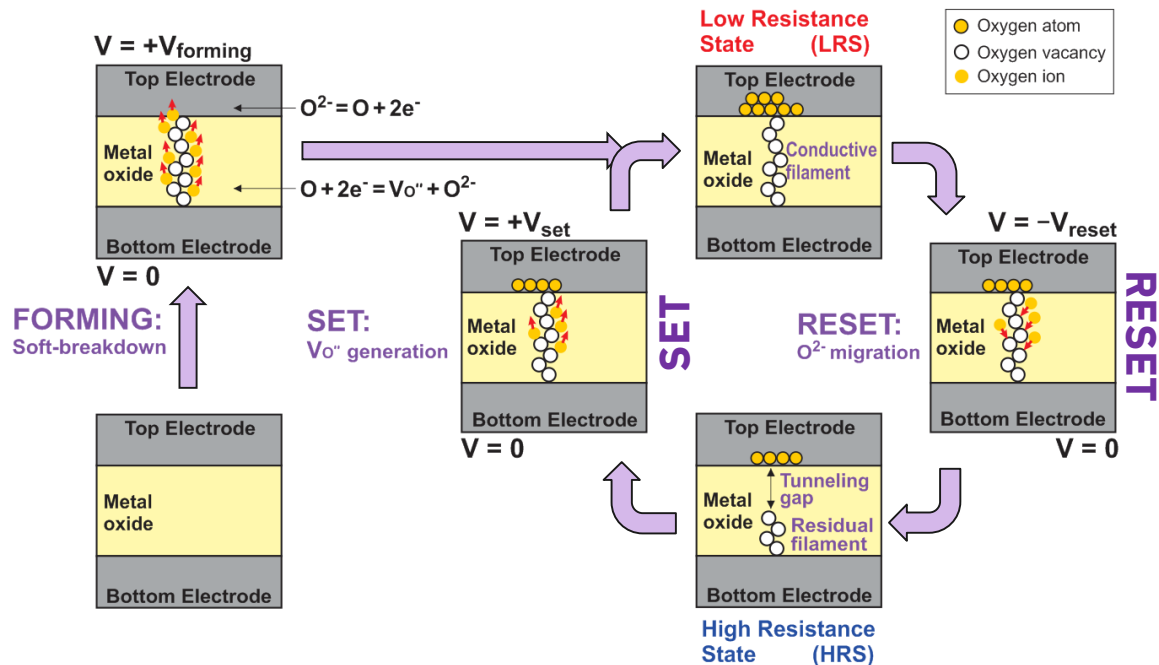


Resistive Random Access Memory (ReRAM)

- **ReRAM** (also called RRAM) is one type of memristive non-volatile memory that works by **changing the resistance across a dielectric solid-state material**

Sufficiently high voltage V_{forming} makes insulating dielectric material conductive

- **Filament-like or homogeneous current conduction path(s)** induced by defects (oxygen-vacancies)
- Switching between **Low Resistance State (LRS)** and **High Resistance State (HRS)** by applying suitable voltages $-V_{\text{reset}}$ and $+V_{\text{set}}$
- The oxygen vacancies act as charge carriers, meaning that the depleted area has a much lower resistance



ReRAM phases:

- **FORMING:** creation of conducting filament in dielectric material between electrodes
- **RESET** (LRS \rightarrow HRS): partial dissolution of filament
- **SET** (HRS \rightarrow LRS): recreation of filament
- **STORAGE:** retain last resistance

Resistive Random Access Memory (ReRAM)

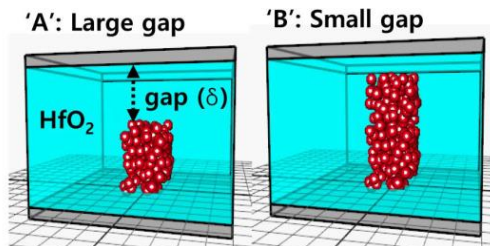
■ Challenge:

With only one (or a few) localized conductive filaments, **switching would be quite abrupt** (between 2 resistance states: **LRS** and **HRS**)

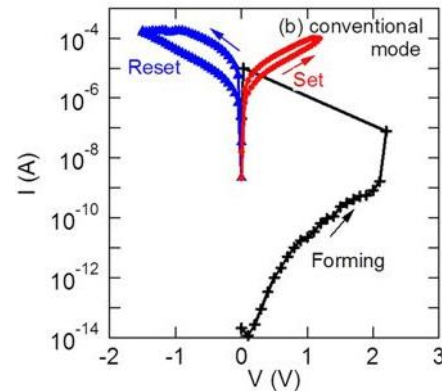
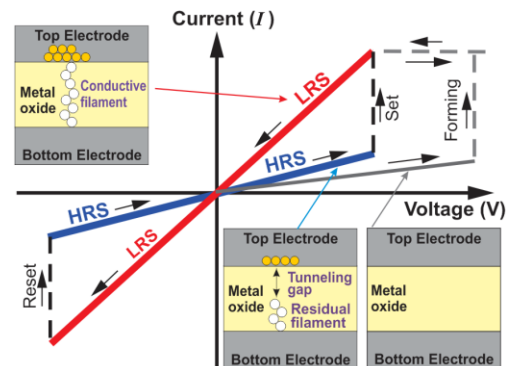
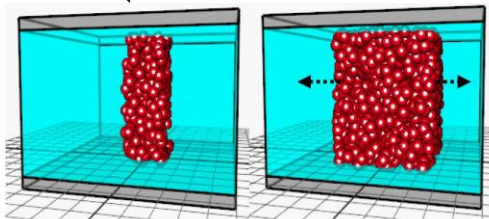
However, for use of ReRAM in analog crossbar arrays, **gradual tuning of resistance with many intermediate states is required**

Use of specifically **engineered oxides with suitable oxygen intercalation^(*)** properties as electrodes

Volumetric changes of conductive filament(s) (i.e., in lateral dimension)



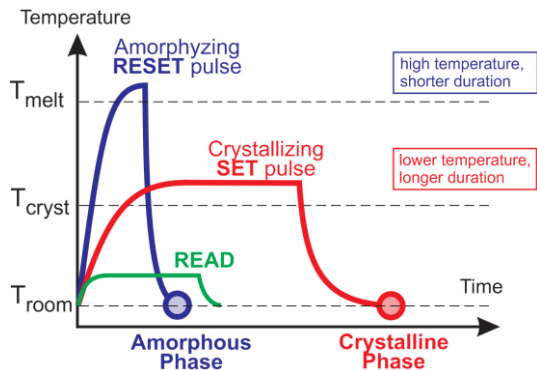
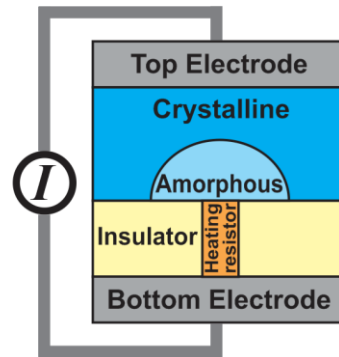
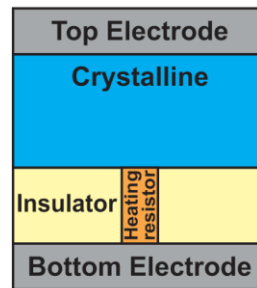
Woo et al. IEEE Electr. Dev. Lett. 38, 9 (2017)



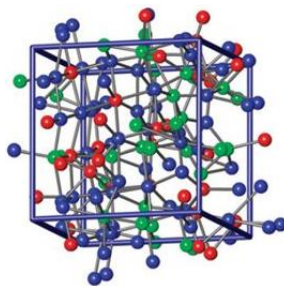
Intercalation: In chemistry, **intercalation** is the reversible inclusion or insertion of molecules (or ions) into materials... (Wikipedia)

Phase Change Memory (PCM)

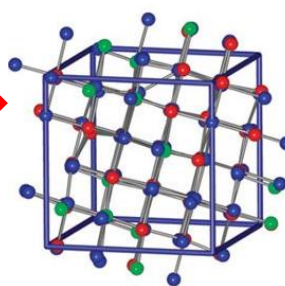
- PCM (also called PCRAM) is another memristive non-volatile memory
- PCM shows **amorphous** and **crystalline** phase
- Rapid and repeated switching between two phases possible
- Switching typically induced by optical or electrical heating
- Physical properties vary significantly between phases
 - crystalline phase → **Low Resistance State (LRS)**
 - amorphous phase → **High Resistance State (HRS)**
- Ratio of electrical resistances $R_{LRS} : R_{HRS} = 1 : 100$ to $1 : 1000$
- Many phase change materials are **chalcogenides**, most studied and utilized: **$\text{Ge}_2\text{Sb}_2\text{Te}_5$ (GST)**



High Resistance State (HRS)



Low Resistance State (LRS)



SET
RESET

Hegedüs, J. & Elliott, S. R., *Nature Mater.* **7**, 399–405 (2008).

Two-level-cell PCM

- only two states
- commercially available as Storage Class Memory (SCM)

Multi-level-cell PCM

- many intermediate states
- under development for emerging analog crossbar arrays



Ferroelectric Tunneling Junction (FTJ)

- **Ferroelectric materials** are dielectrics that exhibit a macroscopic electrical polarization P , even in absence of an external electric field ($E = 0 \rightarrow P = \pm P_{\text{remanent}}$).

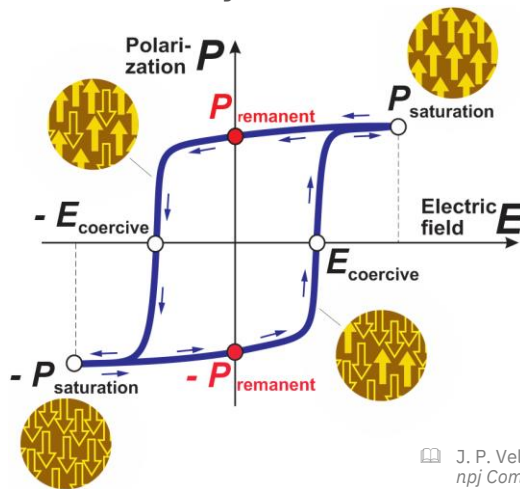
By applying an **electric field**, the macroscopic **polarization state** of ferroelectric material can be **gradually tuned** (\rightarrow ferroel. hysteresis curve) because of polarization switching of individual domains in the material from \uparrow to \downarrow or vice versa.

Ferroelectric material with boundary conditions:

- **Ferroelectric Tunneling Junction** is based on a few nm thick ferroelectric barrier layer sandwiched between two different electrodes (typically metal / semiconductor).

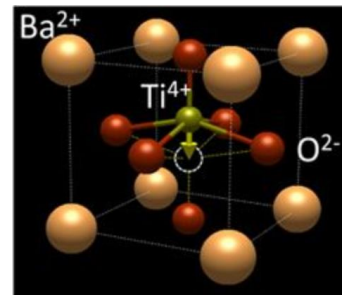
Gradual polarization state tuning possible by applying suitable positive or negative voltage pulses across FTJ.

Ferroelectric hysteresis curve

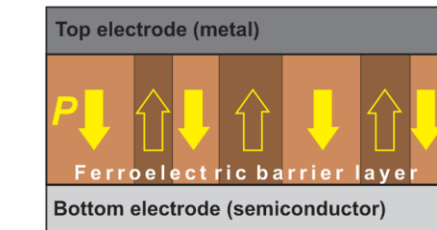


Material example: BaTiO₃

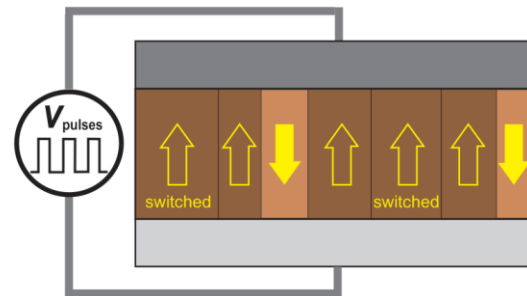
- Cubic phase of BaTiO₃
- Perovskite crystal
- **off-center-position of Ti⁴⁺**
- \rightarrow Ferroelectric behavior



J. P. Velev et al., "Predictive modelling of ferroelectric tunnel junctions", *npj Computational Materials*, vol. 2, Article no: 16009 (2016)



Ferroelectric tunneling junction



Ferroelectric Tunneling Junction (FTJ)

Gradual polarization state tuning can be achieved by applying suitable positive or negative voltage pulses across FTJ.

Many intermediate polarization states can be induced by **nucleation and growth of domains with opposite polarization**.



Electrical current through FTJ varies with macroscopic polarization state because of the **different tunnel widths for the two opposite polarizations** states in individual domains.



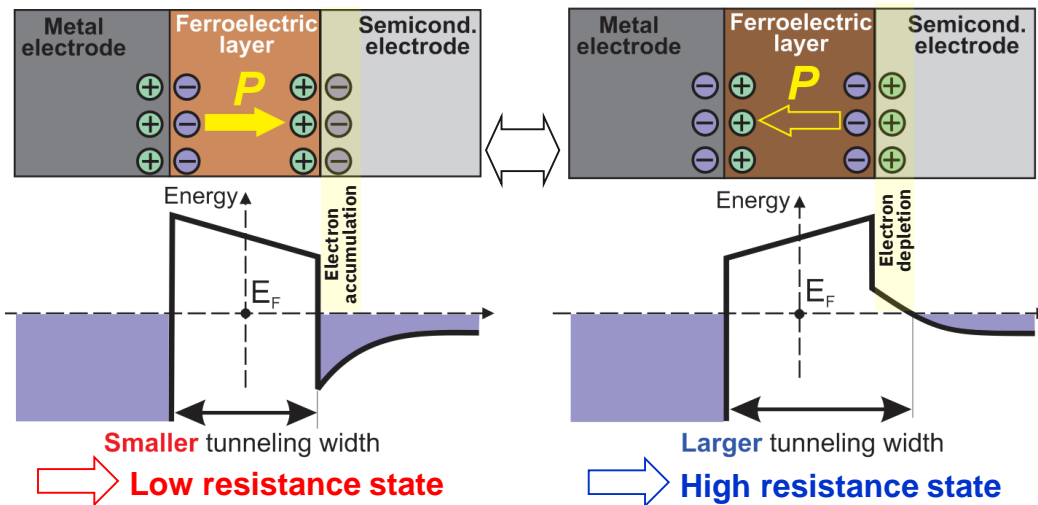
Electrical resistance of FTJ can be tuned by polarization state. → “Tunneling Electro-Resistance” (TER) with up to **10^4 x variation**.
FTJ retains last resistance value when power is turned off.

Dependence of tunneling current and resistance from polarization

(A)

Both situations A & B are for one domain only

(B)



Summary

For the learning (“**training**”) and use (“**inference**”) of **Artificial Neural Networks**, digital (co-)processors (**CPUs, GPUs, FPGAs and ASICs**) in computer systems based on **Von-Neumann architecture** are used almost exclusively today. One promising alternative to these **energy-hungry digital logic based computer systems** is **Analog Neuromorphic Computing**, where computationally time-consuming and therefore expensive operations are performed by specialized **accelerators** comprising **analog** elements with the promise to improve the performance and power efficiency by factors of **1000 to 10,000**.

In general, suitable compute elements are programmable **analog** devices with **non-volatile memory** capabilities that can be arranged in **crossbar arrays** to perform various mathematical operations. The main requirements for such emerging “**non Von-Neumann**” architectures are **vector-matrix multiplications** and the ability to provide the **transposed matrix** for learning as well as means to store **analog synaptic weights**. This mitigates the huge communication overhead for the operands in traditional systems, i.e. avoids the time and energy consuming massive data shuffling between processor and memory.

