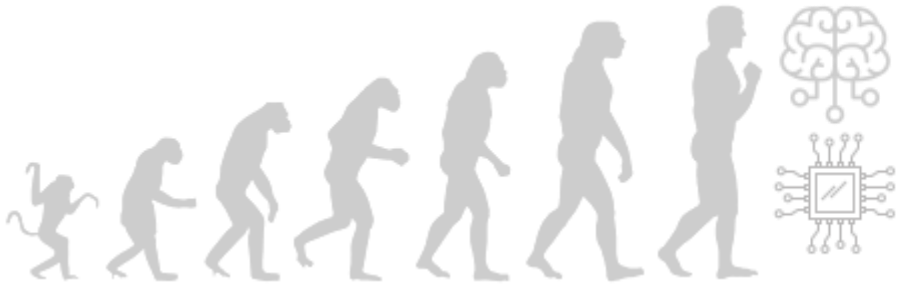
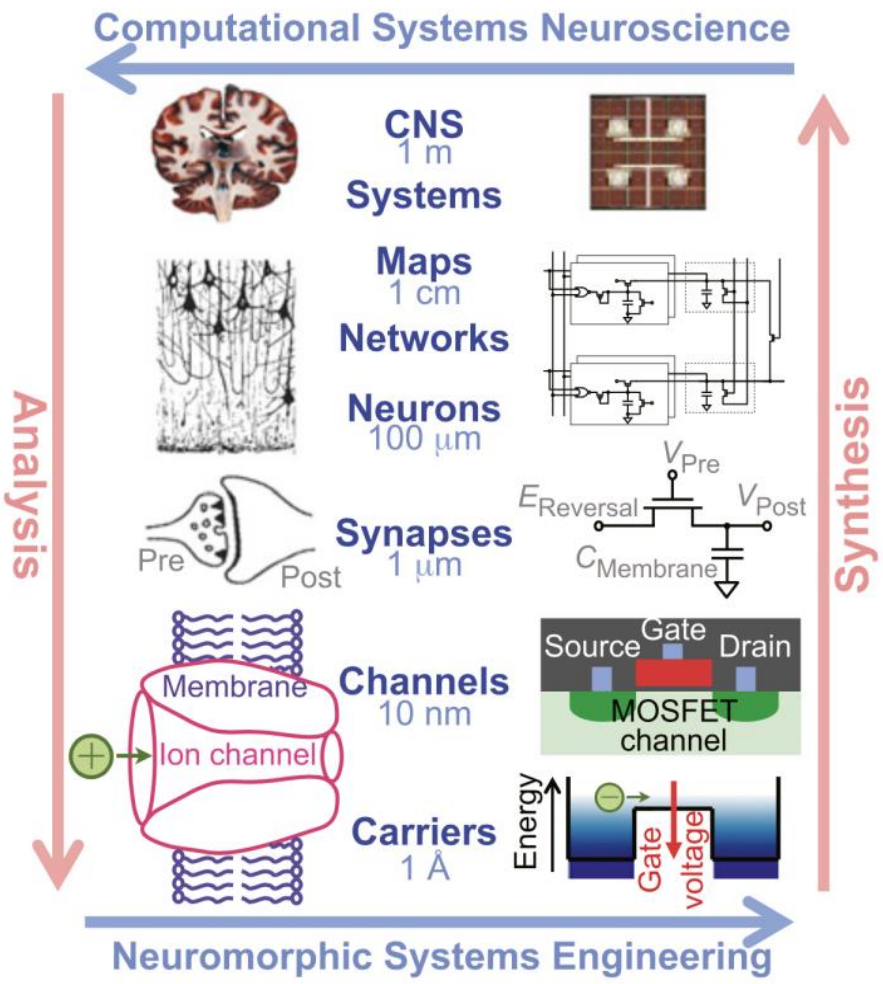
Evolvable Brain-inspired hardware Based on Memristor

Speaker: Dr. Xinming Shi
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast

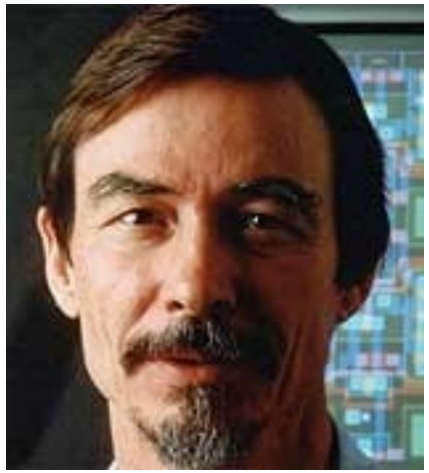
2025-8-28



What is Neuromorphic Computing Hardware?



Cauwenberghs, Gert. 2013

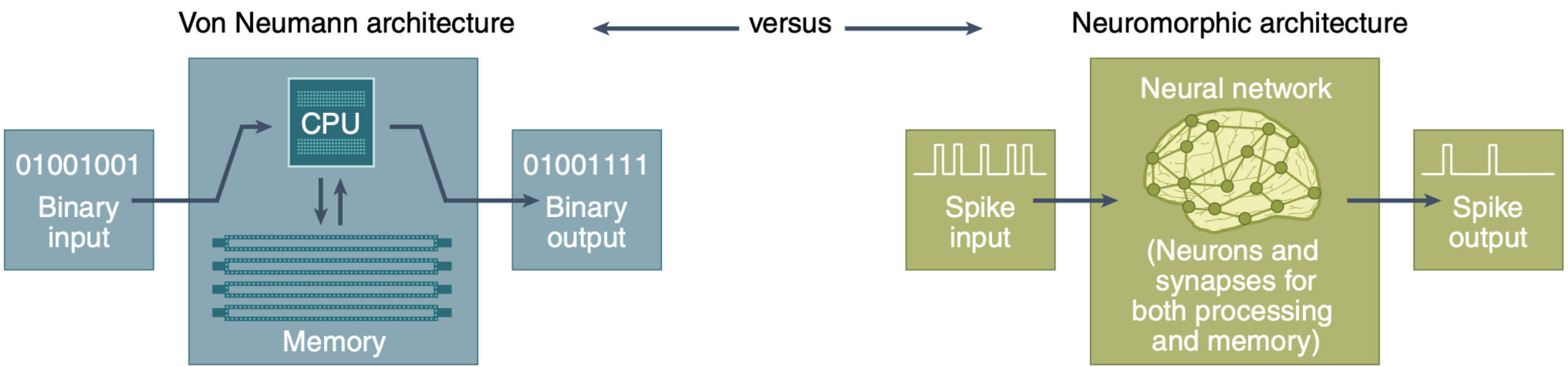


- Computation is a physical process
- "The Constrains on our analog silicon systems are similar to those on neural systems: **wire is limited, power is precious, robustness and reliability are essential**"
  – Carver Mead, 1989

- ✓ **Term proposed by Carver Mead in 1990, referring to analogue VLSI mimicking biological neural systems.**
- ✓ **The original idea of Brain-like computing is much older, discussed by Alan Touring and John von Neumann in the 1950's.**
- ✓ **Neuromorphic systems are typically based on some combination of:**

- ✓ **Efficiency**
- ✓ **Analog Computation**
- ✓ **Real-time Processing**
- ✓ **Low-Latency Decision Making**
- ✓ **Fault Tolerance and Robustness**
- ✓ **Adaptability and Learning in Dynamic Environments**

Neuromorphic vs von Neumann

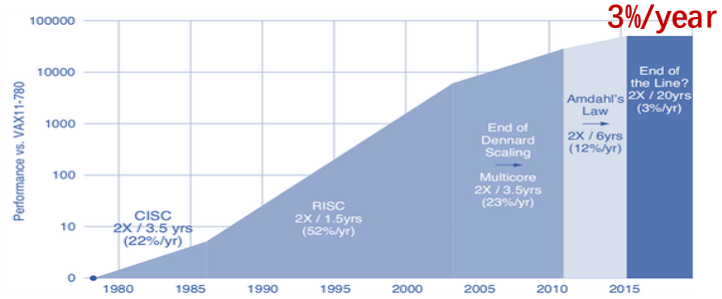


Sequential processing	Operation	Massively parallel processing
Separated computation and memory	Organization	Collocated processing and memory
Code as binary instructions	Programming	Spiking neural network
Binary data	Communication	Spikes
Synchronous (clock-driven)	Timing	Asynchronous (event-driven)

Image from Schuman, Catherine D., Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, Prasanna Date, and Bill Kay. 2022. "Opportunities for Neuromorphic Computing Algorithms and Applications." Nature Computational Science 2 (1): 10–19.

Why Neuromorphic Computing Hardware?

➢ Motivation: Contradiction between computation power of chips and high requirement for training



Growth in performance of chip

Source: Computer Architecture: A Quantitative Approach

Requirement of computation power for training (Petaflop/s-day)

Source: Intel, NIVIDIA, OpenAI, 2018

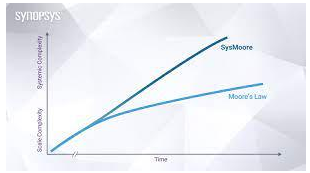
Contradiction between computation power of chips and high requirement for training

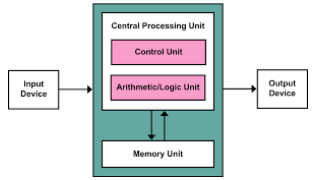
Due to

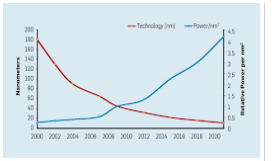
- **Device bottleneck → Moore Law**
  The size is close to the physical limit.

- **Architecture bottleneck → von Neumann**
  Architecture processing and storage units are separated, data exchange exists and storage wall exists.

- **Energy efficiency bottleneck → Dennard Scaling**
  The power density of single-core processors has reached a bottleneck. It is expected that by 2040, 10^40 big data operations will require 10^27 J of energy consumption.







Require

New devices

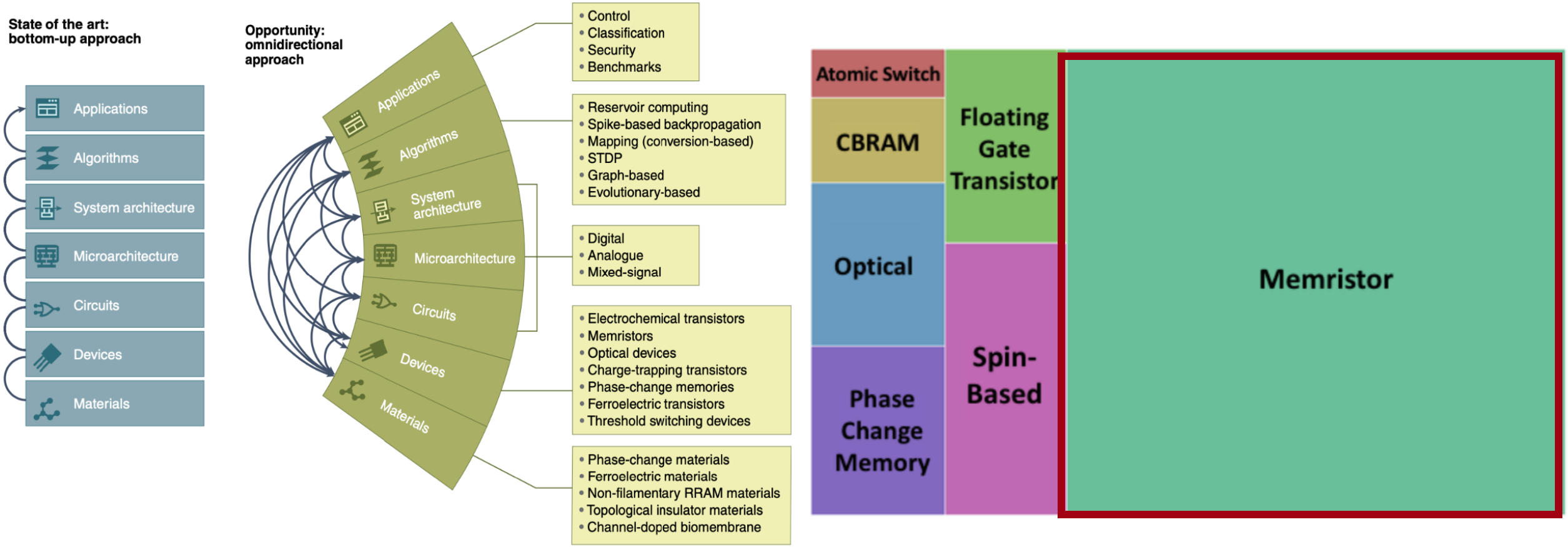
New architecture

New system

New devices, architecture and system are required to tackle these bottlenecks

Neuromorphic computing hardware is the one with new devices, new architecture and new system

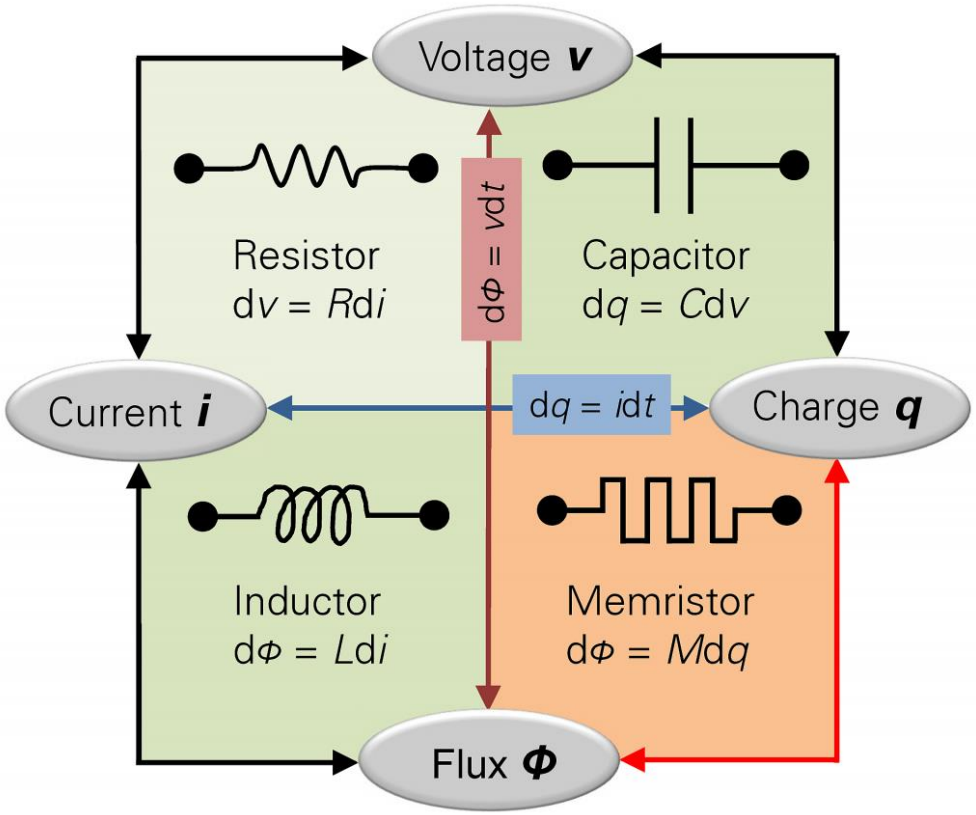
New Devices of Neuromorphic Computing



Mehonic, Adnan, and Anthony J. Kenyon. "Brain-inspired computing needs a master plan." *Nature* 604.7905 (2022): 255-260.
Schuman, Catherine D., et al. "Opportunities for neuromorphic computing algorithms and applications." *Nature Computational Science* 2.1 (2022): 10-19.

Memristor Devices: Introduction

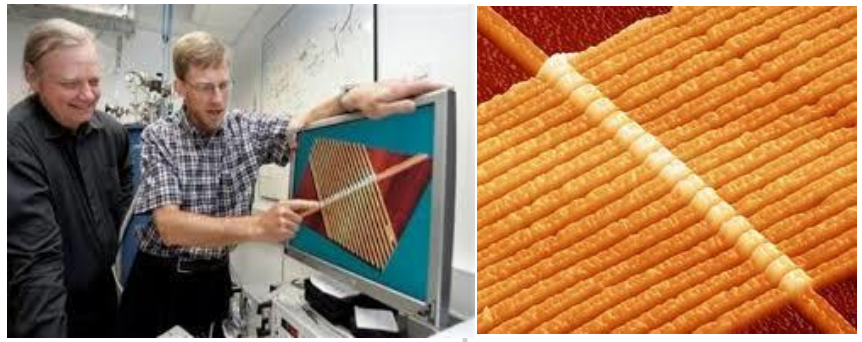
L. O. Chua, "memristor – the missing circuit element" IEEE Trans. Circuit Theory (1971)

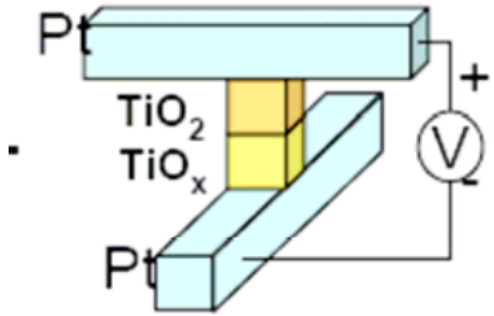


We know that every electronic circuit can be designed by **using several passive components namely resistors, capacitors, as well as inductors, but there will be an essential fourth component which is termed as a memristor**. These are semiconductors used for jointing passive components to form a fourth component, and the resistance is named as memristance. It is a resistance depends on charge in memristor circuits & the resistance unit is ohm.

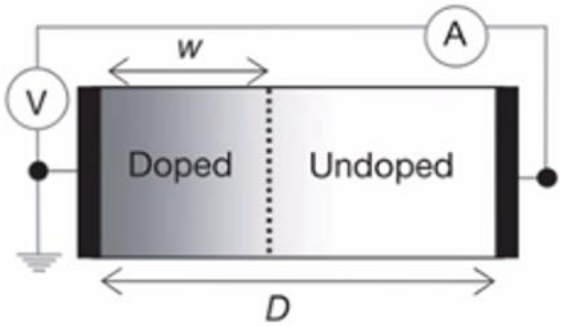
Memristor Devices: Introduction

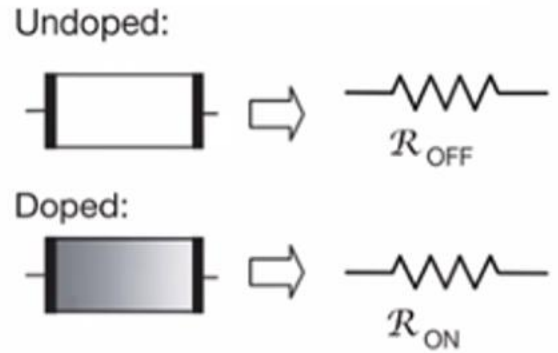
In 2008，HP company fabricated the first physical memristor





< 30x30 nm^2

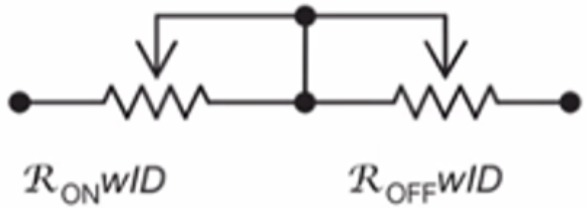




Nanoscale potentiometer

R_{on}: Low resistance

R_{off}: High resistance



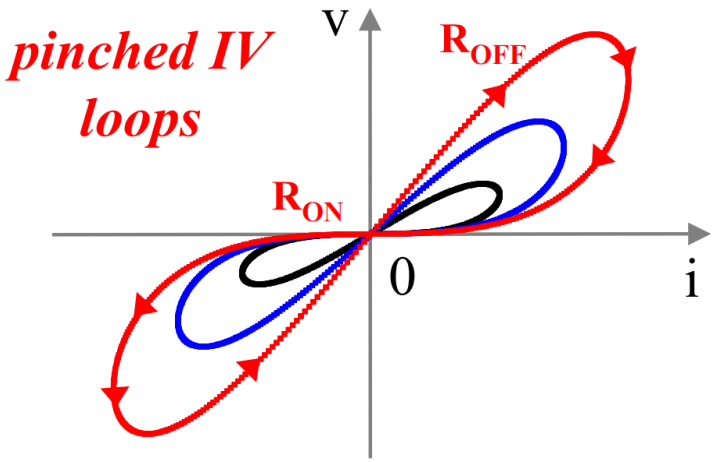
Strukov, Snider, Stewart & Williams, Nature 453 (2008)

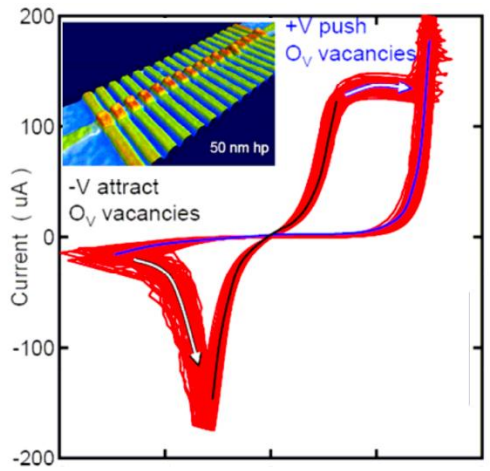
Memristor Devices: Introduction

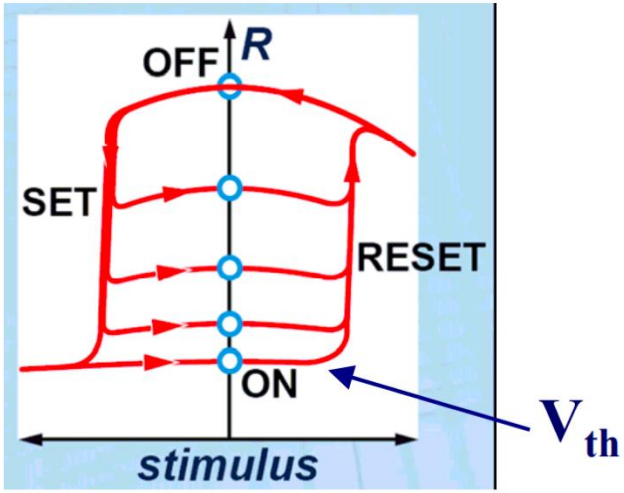
$v = M(q)\, i$

M is a resistance that "remembers" how much current was injected, and how long
continuously tunable between R_{ON} and R_{OFF}

- Nano resistance
- Tunable (multi-states of resistance available)
- Non volatile
- Non-linear : V < Vth read, V > Vth write



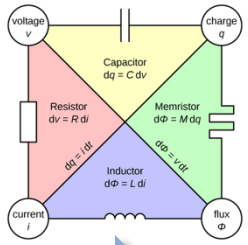




Yang et al., Nature Nano (2008)

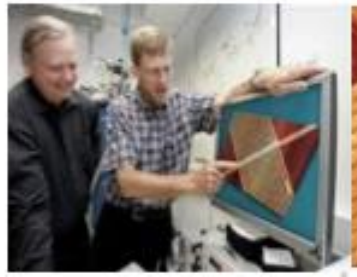
Memristor Devices: Brief History

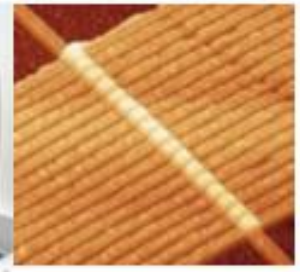
1970s, Leon O Chua predicted and generalized the fourth circuit element, memristor and memristive system



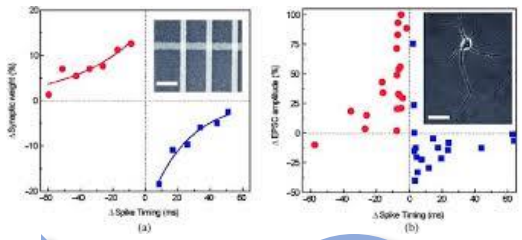
1970s

2008, HP company fabricated the first physical memristor



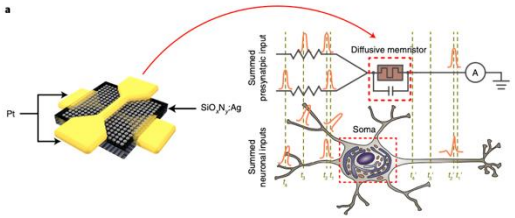


Wei Lu of UMich fabricated non-volatile memristor device for artificial synapse with learning mechanism



2010

Joshua Yang of USC fabricated volatile memristor device

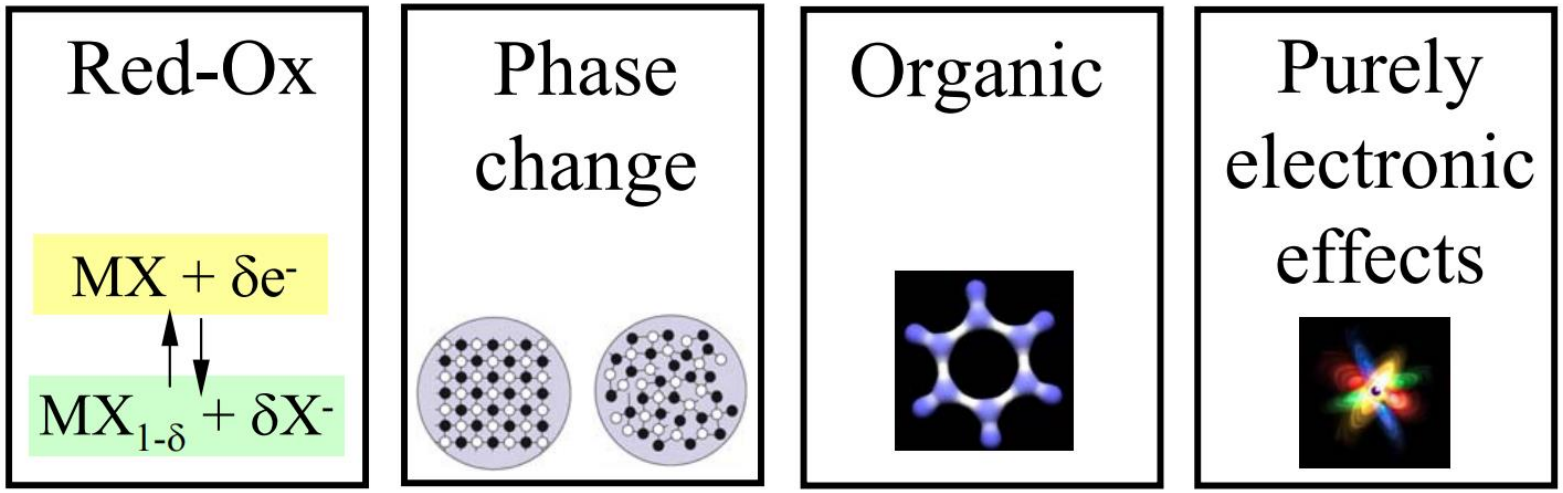


2017

Current

Foundries like **TSMC**, Global Foundry, Winbond
Starups like **Tetramem** (by Joshua Yang) are comercializing meristors, memristive crossbar(RRAM)

Memristor Devices: Classification



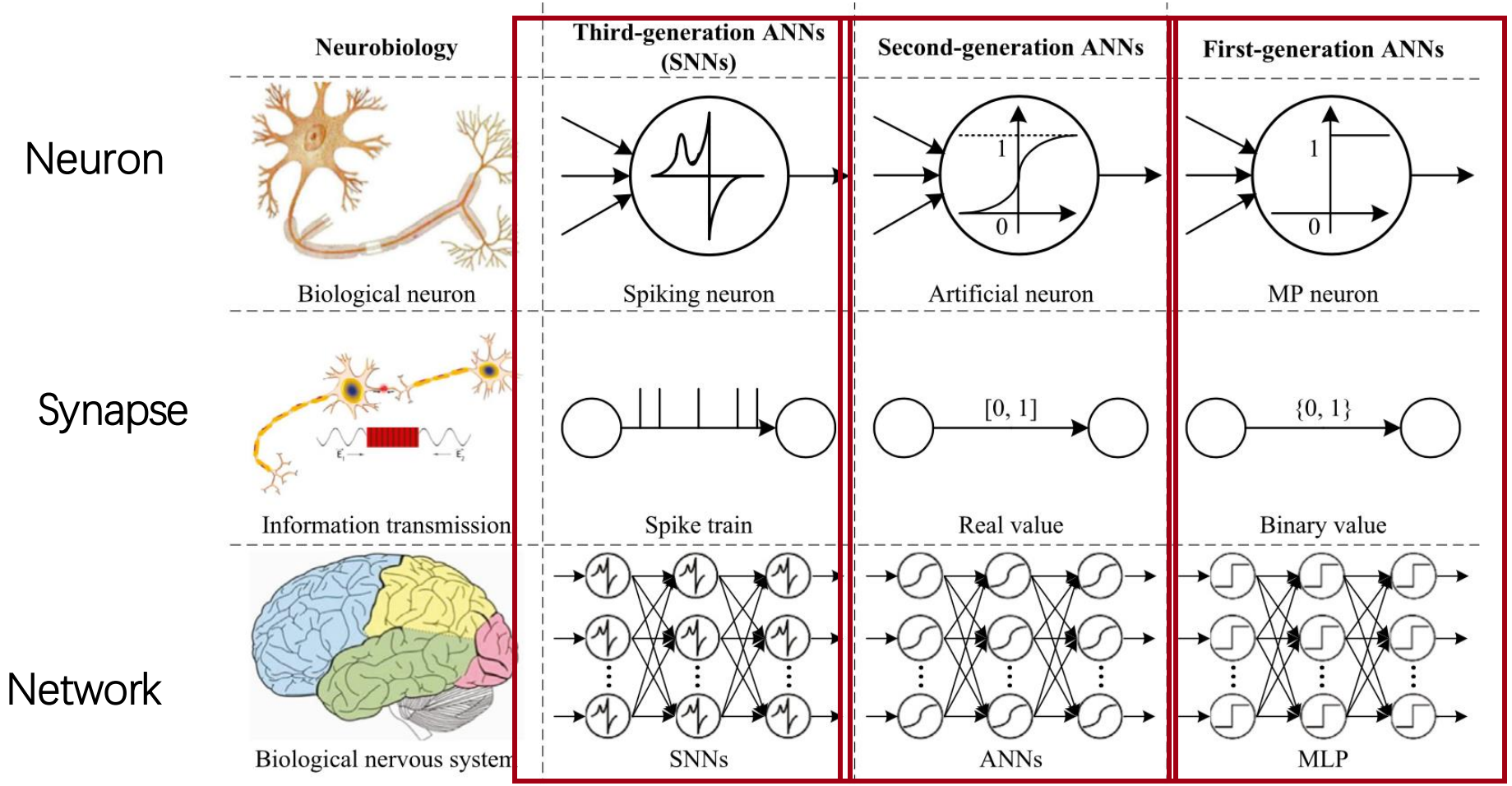
Memristor Devices: Classification

Type	Mechanism	Typical Materials	Advantages	Limitations	Applications
Red-Ox (Filamentary RRAM)	Ionic migration and redox reactions forming/rupturing conductive filaments	TiO_2, HfO_2, Ta_2O_5	Low power, non-volatile, widely studied	Variability, limited endurance	Memory, neuromorphic computing
Phase Change	Reversible transition between crystalline and amorphous states (thermal process)	$Ge_2Sb_2Te_5$ (GST), other chalcogenides	Fast switching, multi-level storage	High energy due to heating, material fatigue	Storage-class memory, synapses
Organic	Charge transfer and molecular conformational changes in organic molecules	Conductive polymers, small molecules	Flexible, low-cost, printable electronics	Poor stability, sensitive to environment	Flexible electronics, sensors
Purely Electronic Effects	Electron trapping/detrapping, tunneling, Coulomb blockade	2D materials, oxide heterostructures	Ultra-fast switching, potential scalability	Retention issues, complex fabrication	High-speed logic, in-memory computing

Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S. (2008). *The missing memristor found*. **Nature**, 453(7191), 80–83.
Ovshinsky, S. R. (1968). *Reversible electrical switching phenomena in disordered structures*. **Physical Review Letters**, 21(20), 1450–1453.
Erokhin, V., & Fontana, M. P. (2008). *Electrochemical memristive devices for neuromorphic networks*. **Nanotechnology**, 19(19), 195201.
Lee, M. J., et al. (2007). *A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures*. **Nature Materials**, 10(8), 625–630.

New Computing Paradigm of Neuromorphic Computing

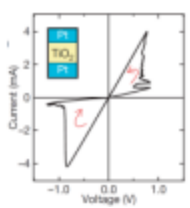
Spiking neural network, spike, sparse, dynamic,

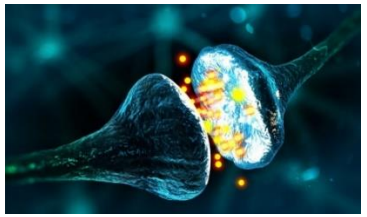


Wang, Xiangwen, et al. 2020

New Computing Paradigm of Neuromorphic Computing

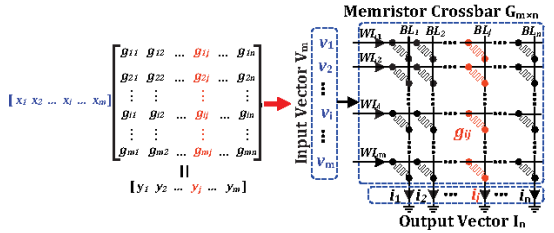
➢ Synapse and Neuron Basic model





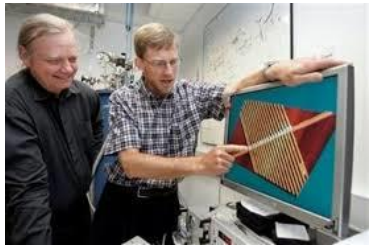
- Memory capacity
- Variable memristance

Single memristor is used to emulate the synapse



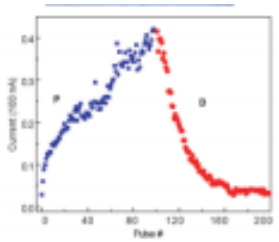
- Memory capacity
- Quick switching

Memristor crossbar is applied to map weight matrix



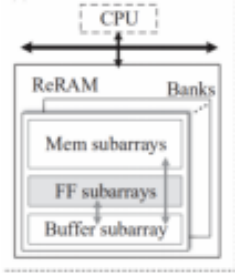
fabricated the physical memristor

(Nature 2008)



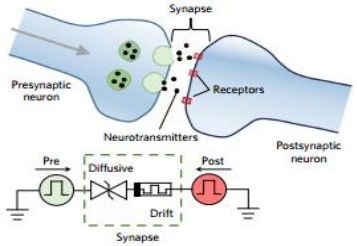
Wei proposed nonvolatile electronic synapse

(Nano Lett. 2010)



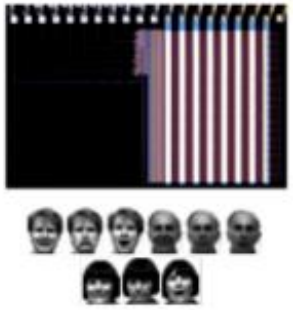
Yuan proposed PRIME architecture

(ISCA 2016)



Zhongrui proposed diffusive memristor to emulate synapse with forgetting function

(Nature 2017)



128×8 crossbar Face recognition

(Nature Comm. 2017)

2008
First physical memristor

2010
Continuous memristance

2016
First Processing-in-memory Architecture

2017
Emulate different dynamic behaviors of synapse

2017
Different Processing-in-memory Architecture

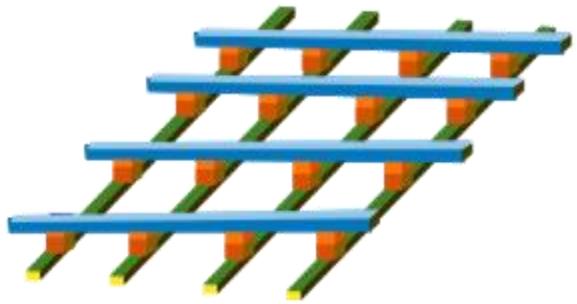
New Computing Paradigm of Neuromorphic Computing

➢ Vector-matrix multiplication (VMM)

Vector-matrix multiplication (VMM)

Ohm's law and Kirchhoff's law

$$I = G \cdot V \qquad I_{\text{out}} = \sum I_{\text{out}}$$

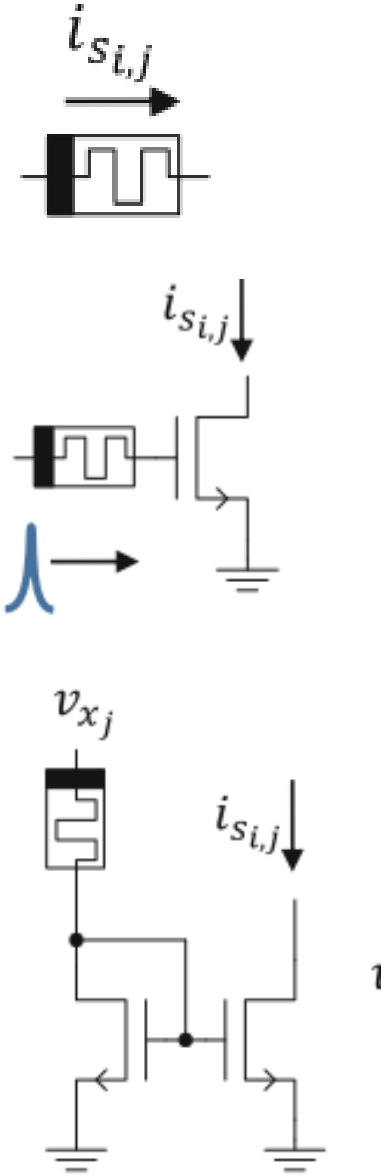


In memristor crossbar, this multiplication is achieved directly by physical laws.

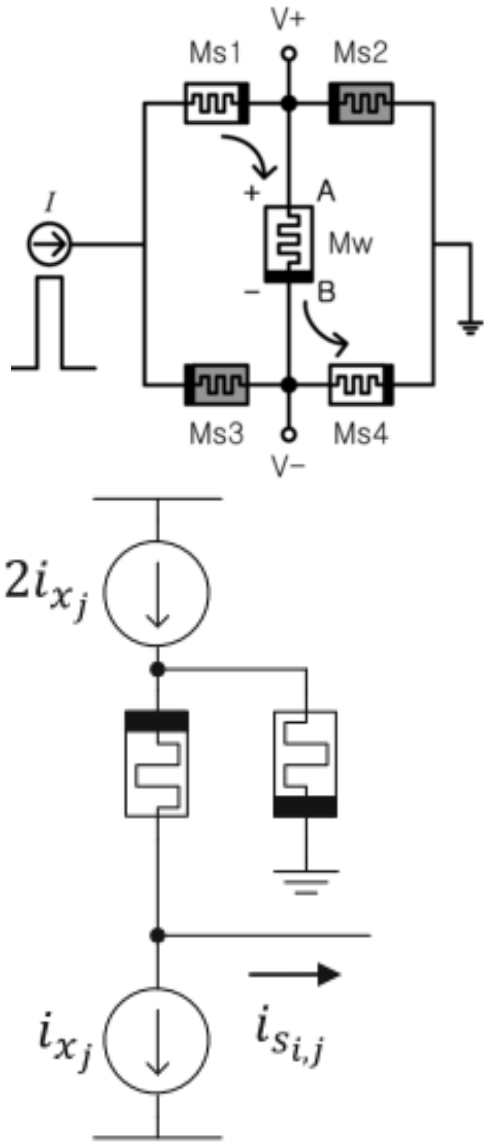
- In a crossbar array, we apply input voltages to the horizontal lines, often called word lines.
- Each cross-point is a memristor device whose conductance value directly represents a synaptic weight. A high conductance corresponds to a strong connection, and a low conductance corresponds to a weak one.
- According to Ohm's law, the current through each memristor is simply the product of the input voltage and its conductance: $I=G\times V$
- Then, by Kirchhoff's current law, all currents along a column automatically sum together. This gives us the weighted sum of inputs in one step, without explicitly performing digital multiplications and additions.

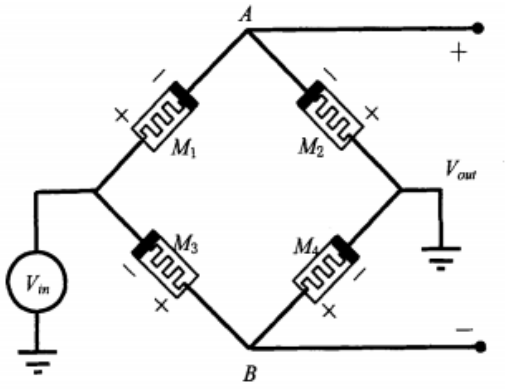
New Computing Paradigm of Neuromorphic Computing

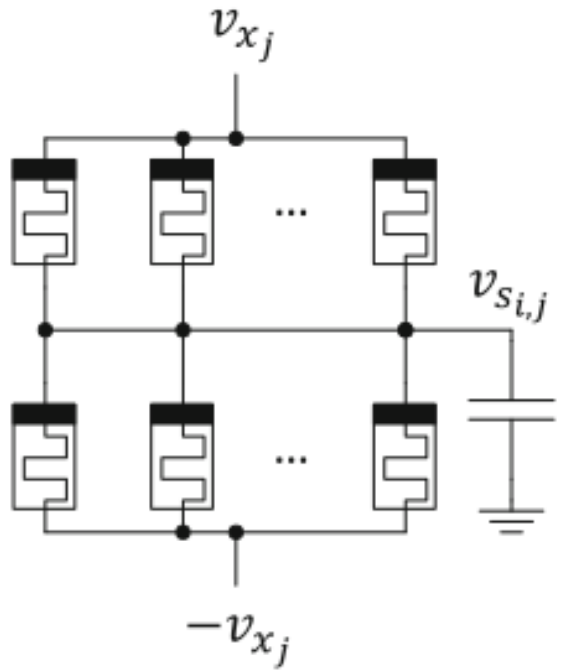
unipolar weight values



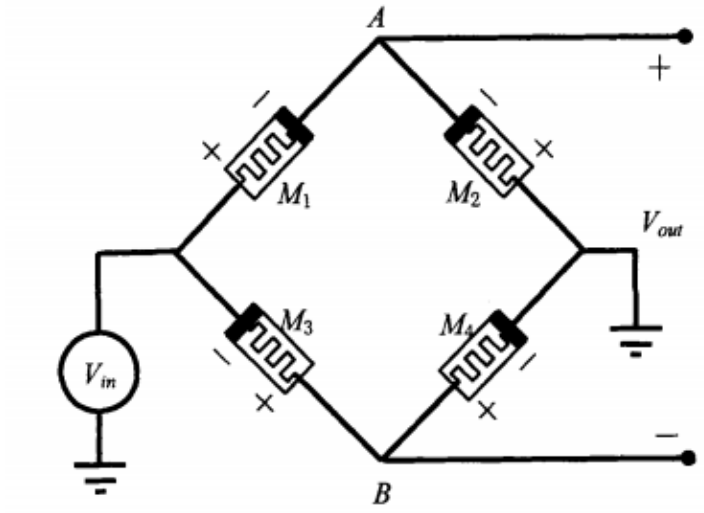
Bipolar weight values







New Computing Paradigm of Neuromorphic Computing



$$V_{M1} = \frac{M_1}{M_1 + M_2} V_{in}$$

$$V_{M2} = \frac{M_2}{M_1 + M_2} V_{in} = V_A$$

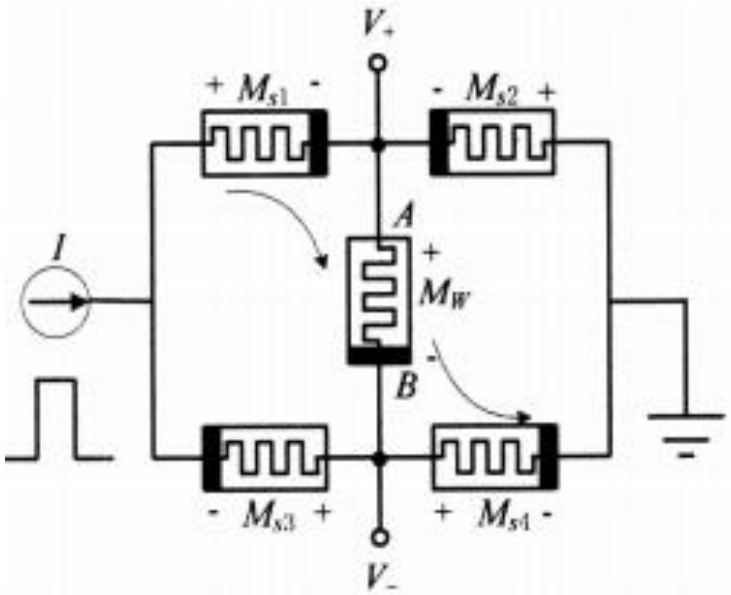
$$V_{M3} = \frac{M_3}{M_3 + M_4} V_{in}$$

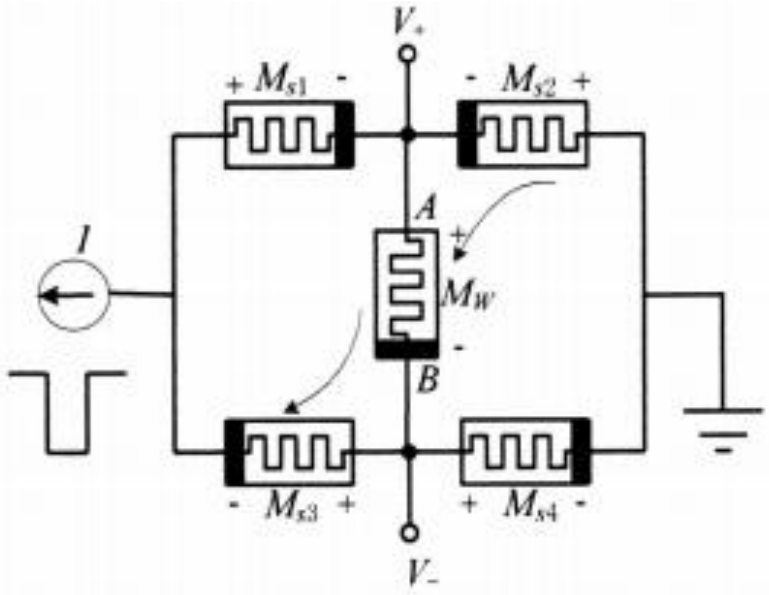
$$V_{M4} = \frac{M_4}{M_3 + M_4} V_{in} = V_B$$

$$V_{out} = V_A - V_B = \left(\frac{M_2}{M_1 + M_2} - \frac{M_4}{M_3 + M_4}\right) V_{in}$$

$$V_{out} = W \times V_{in}$$

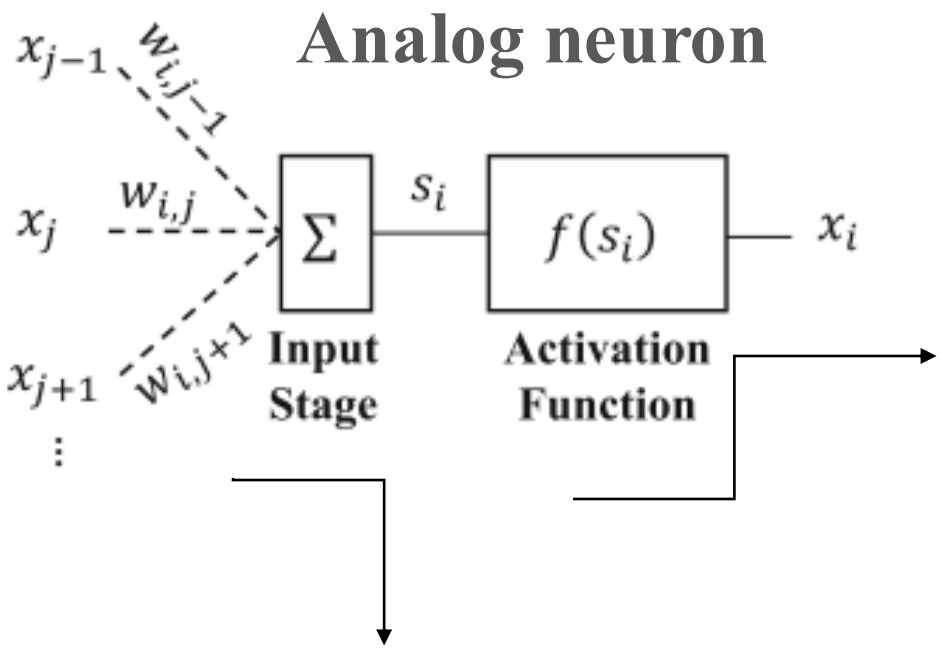
$$W = \frac{M_2}{M_1 + M_2} - \frac{M_4}{M_3 + M_4}$$

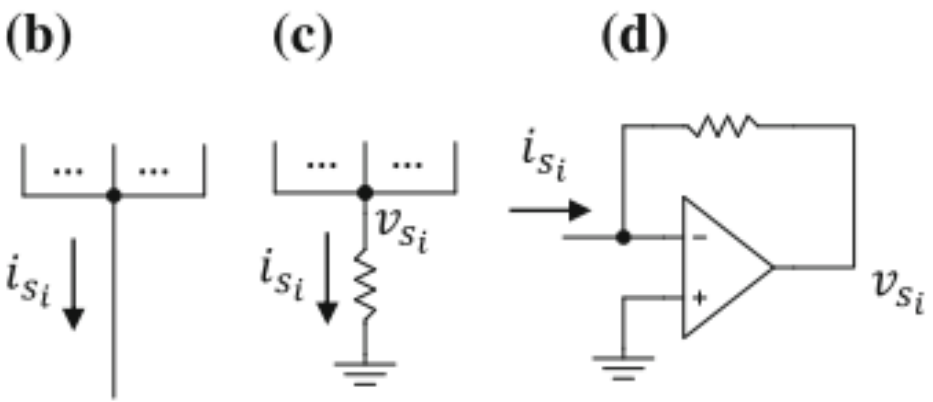


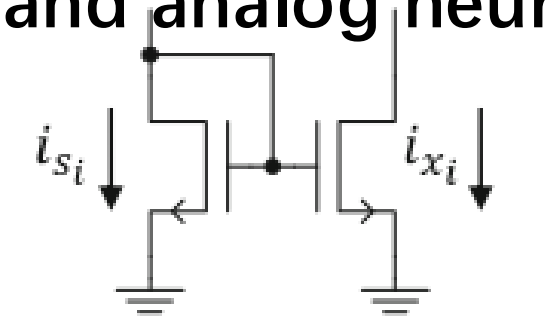


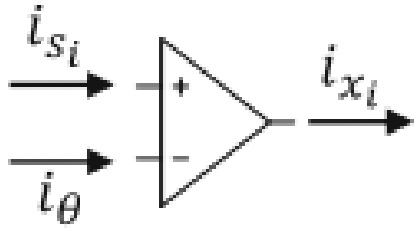
Valsa, J., Biolek, D., Biolek, Z., & Kolka, Z."Analysis of a memristor bridge circuit." *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 24, no. 4, pp. 400–408, 2011.

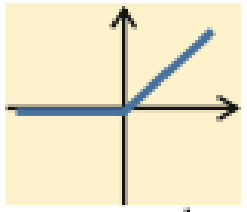
Differences between spike neuron and analog neuron

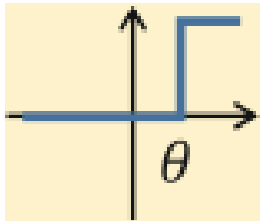


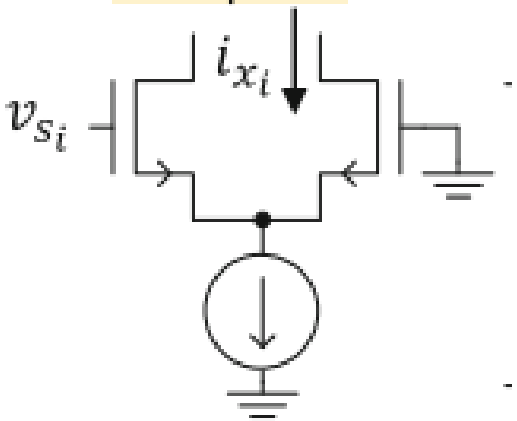


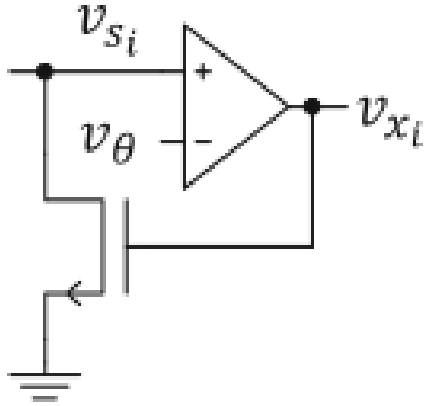


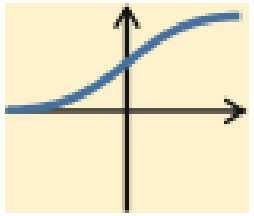


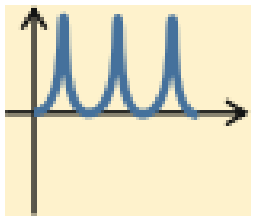




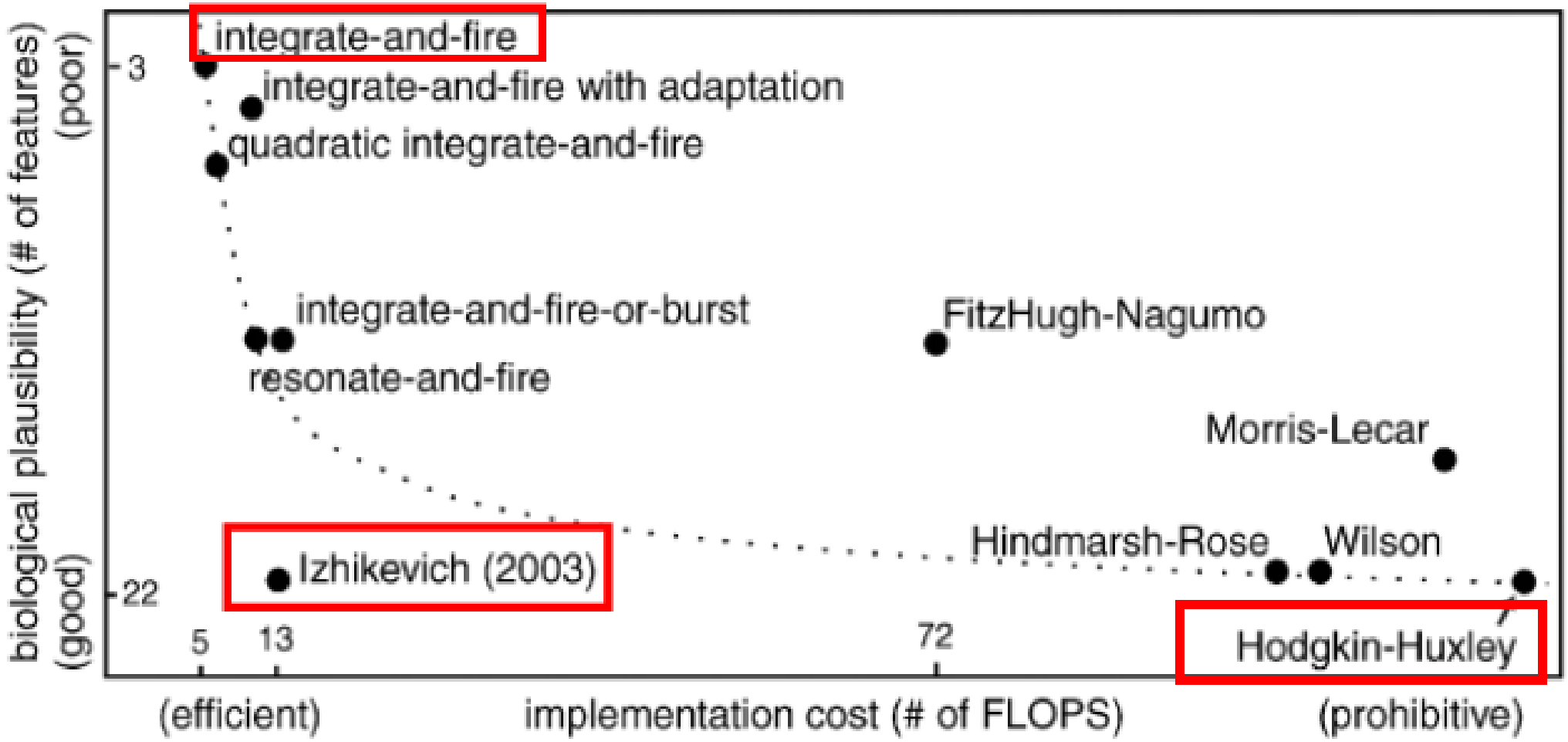




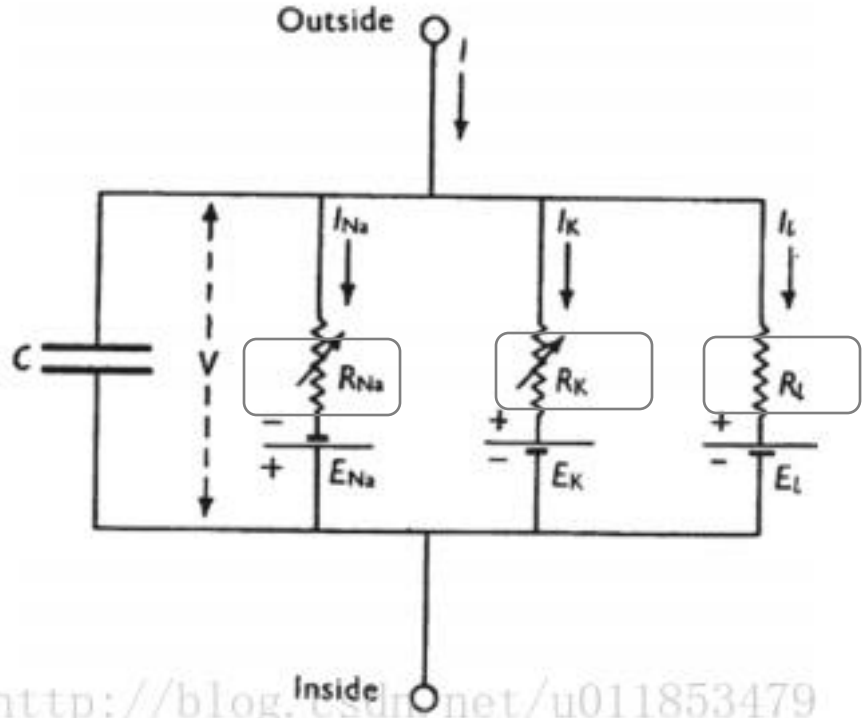




New Computing Paradigm of Neuromorphic Computing



Spike neuron： Hodgkin-Huxley neuron model



$$C\dot{V} = I - \bar{g}_K n^4 \left(V - E_K\right) - \bar{g}_{Na} m^3 h \left(V - E_{Na}\right) - \bar{g}_{Cl} n^4 h \left(V - E_{Cl}\right)$$

$$\dot{n} = \alpha_n \left(V\right)(1-n) - \beta_n \left(V\right) n$$

$$\dot{m} = \alpha_m \left(V\right)(1-m) - \beta_m \left(V\right) m$$

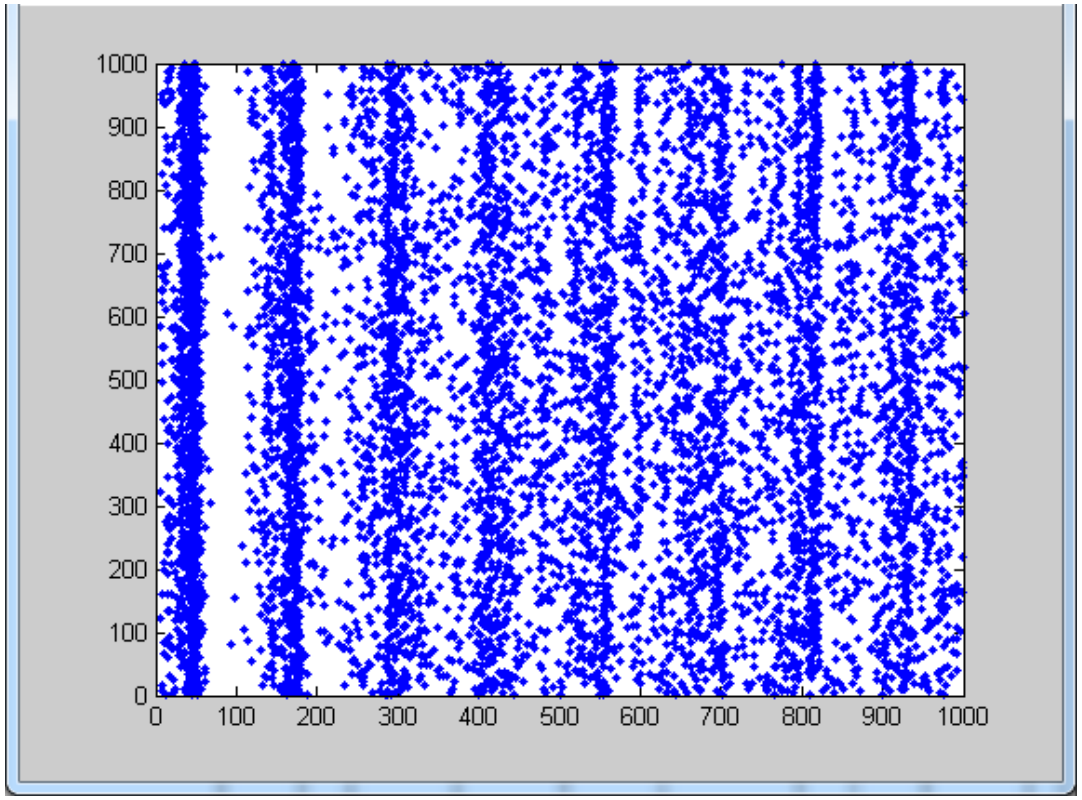
$$\dot{h} = \alpha_h \left(V\right)(1-h) - \beta_h \left(V\right) h$$

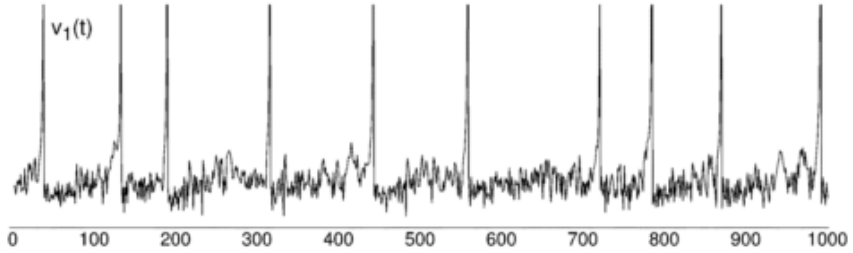
Izhikevich model

$$V_i^{'} = 0.04V_i^2 + 5V_i + 140 - U_i + I + I_i^{syn}$$

$$U_i^{'} = a(bV_i - U_i) + D\xi_i$$

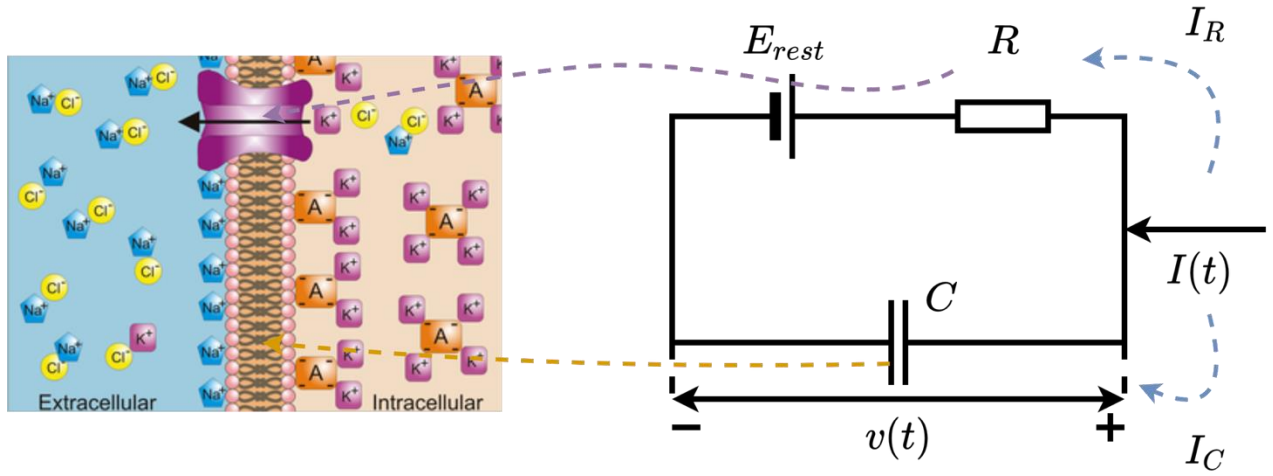
$$if \quad V_i > 30mV, then \begin{cases} V_i \leftarrow c \\ U_i \leftarrow U_i + d \end{cases}$$

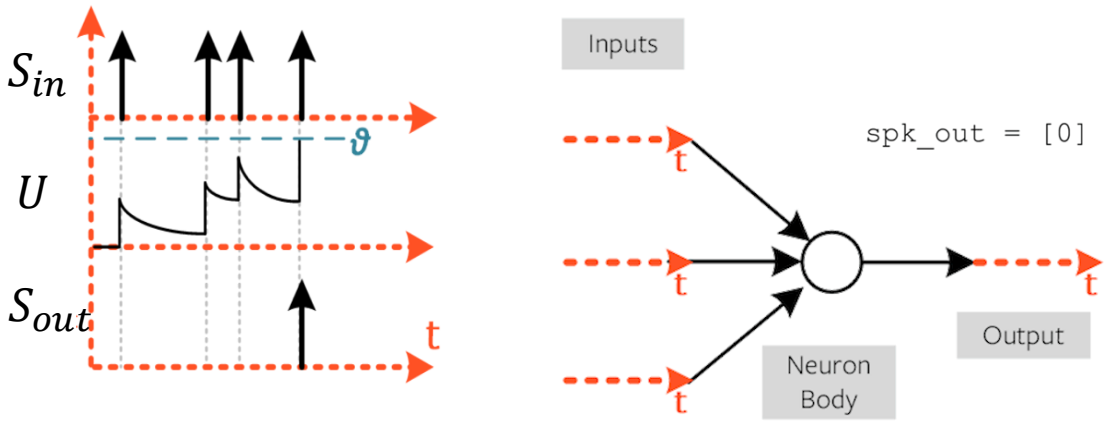




New Computing Paradigm of Neuromorphic Computing

➢ LIF neuron model



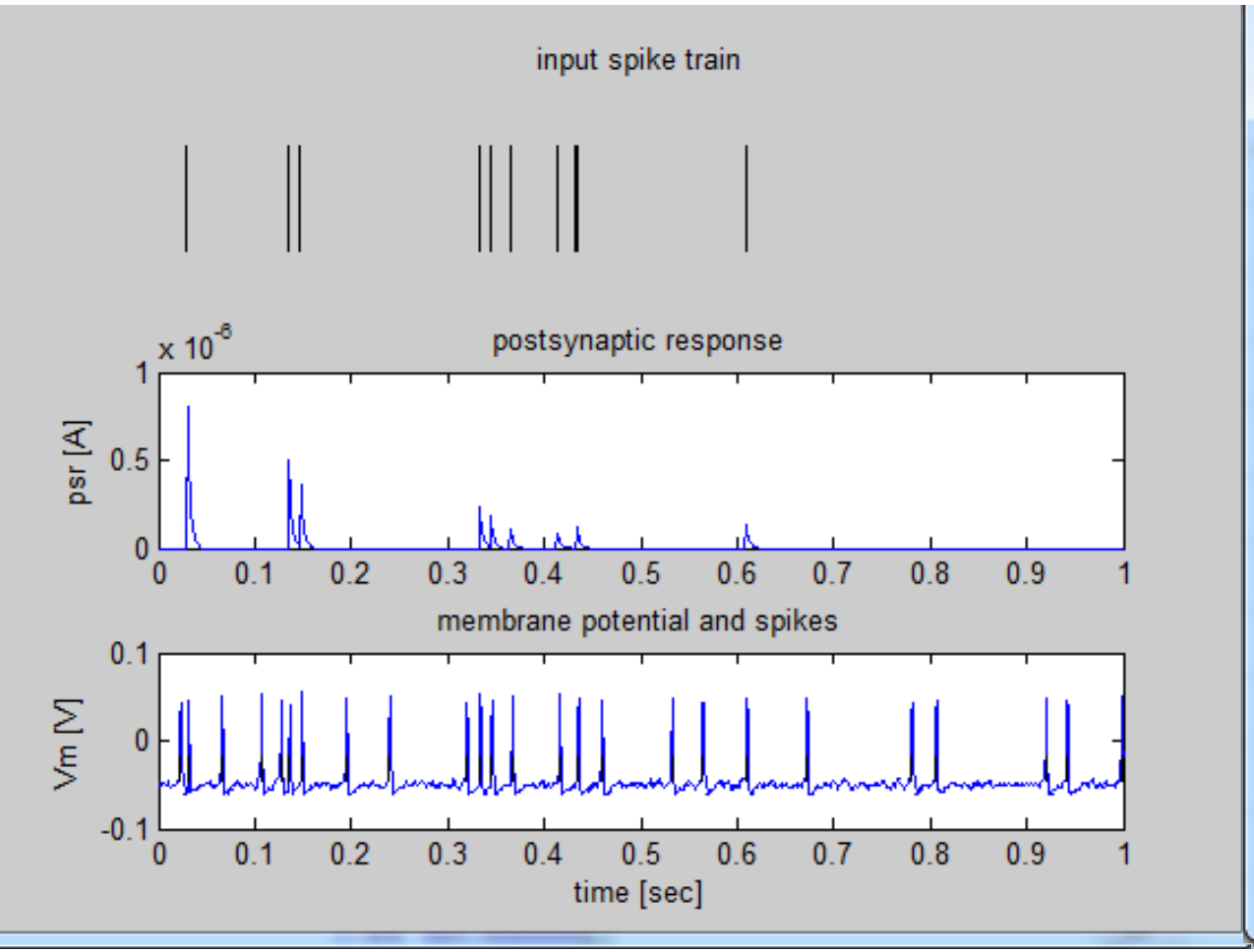


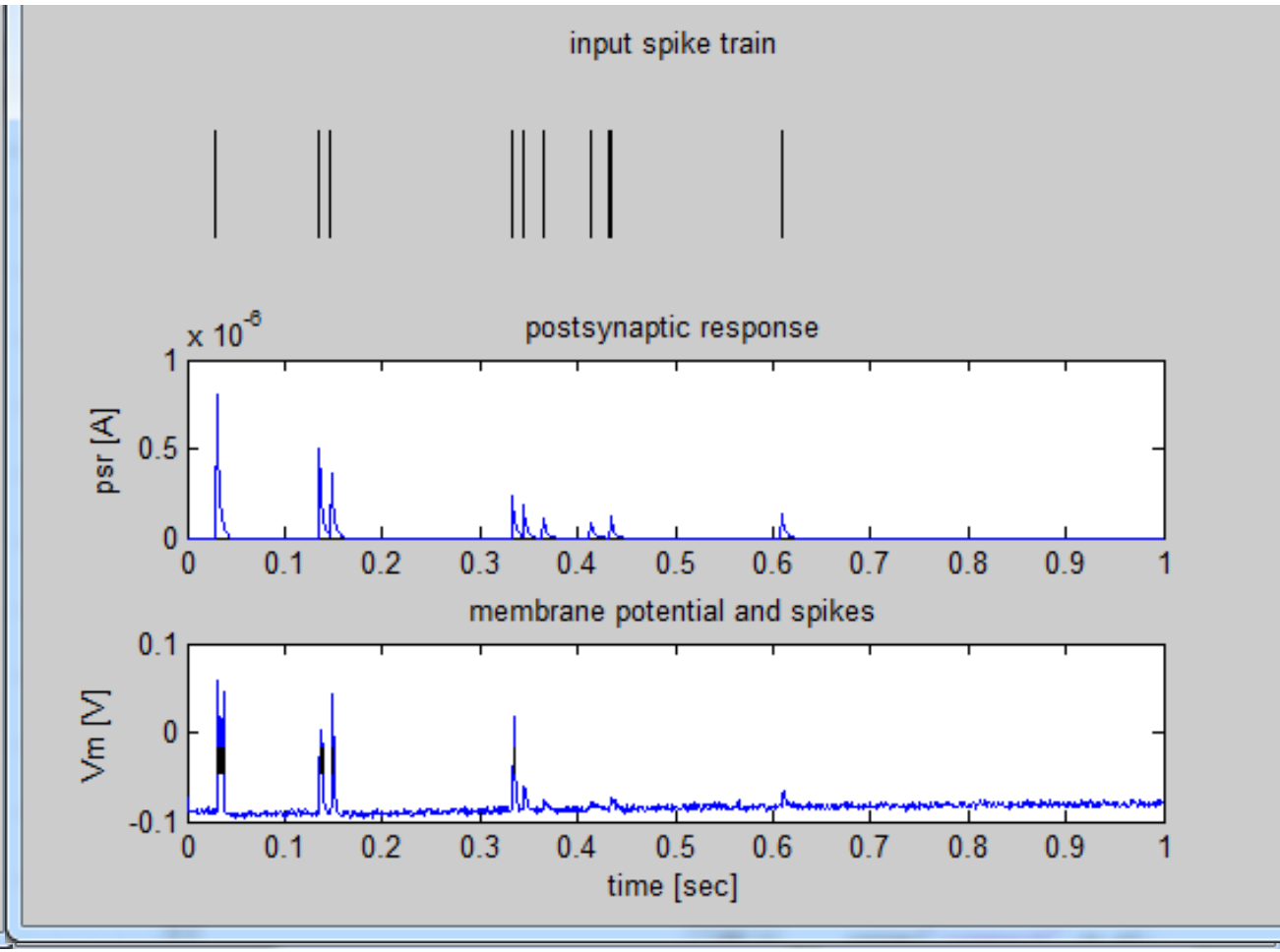
Leaky Integrate-and-Fire (LIF) Neuron

- $I(t) = I_R + I_C$
- $I(t) = \frac{v(t) - E_{rest}}{R} + C\frac{dv(t)}{dt}$
- $RC\frac{dv(t)}{dt} = -(v(t) - E_{rest}) + RI(t)$

Continous dynamic -> Discretized status

- synaptic current: (decay, input)
$$I[t+1] = \alpha I[t] + WS_{in}[t]$$
- membrane potential: (decay, input, reset)
$$U[t+1] = \beta U[t] + I[t] - S[t]$$
- spike:
$$S[t] = \Theta(U[t] - U_{th})$$
- Decay factor:
$$\alpha = e^{-1/\tau_{syn}} \quad \beta = e^{-1/\tau_{mem}}$$

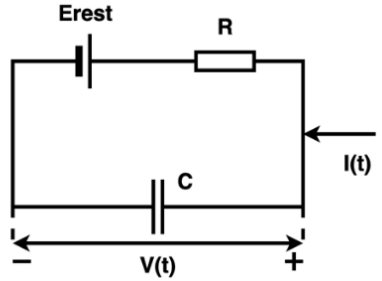

input spike train
x 10^-6
postsynaptic response
psr [A]
1
0.5
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
membrane potential and spikes
Vm [V]
0.1
0
-0.1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
time [sec]

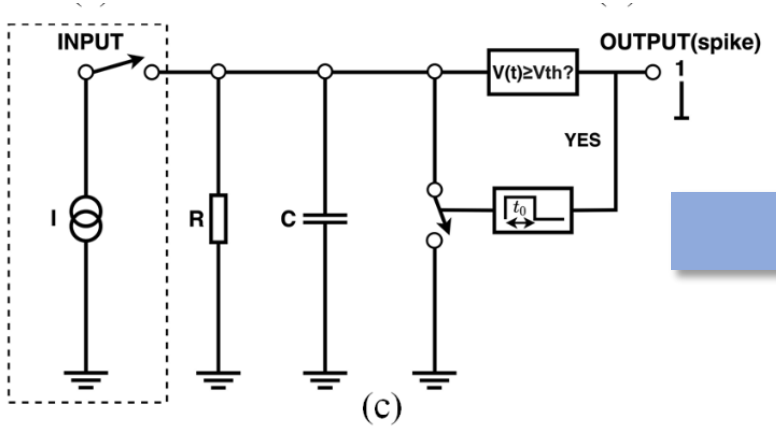

input spike train
x 10^-6
postsynaptic response
psr [A]
1
0.5
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
membrane potential and spikes
Vm [V]
0.1
0
-0.1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
time [sec]

New Computing Paradigm of Neuromorphic Computing

➢ Memristive Neuron

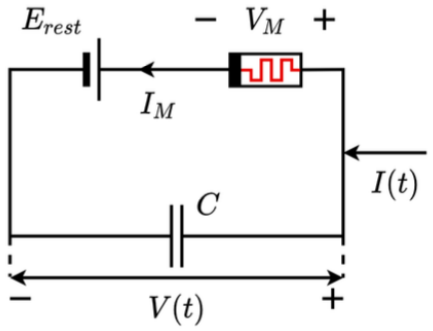
LIF neuron required hard reset





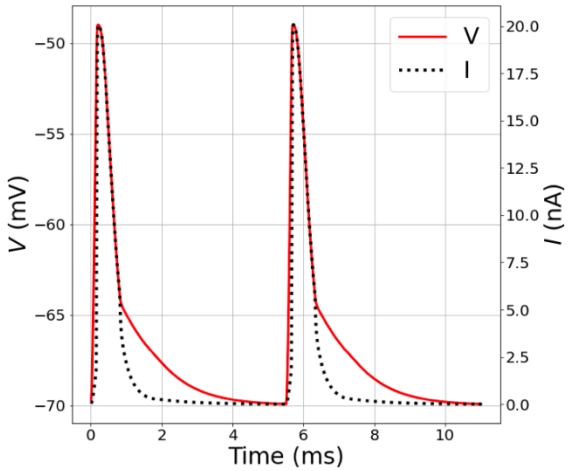
Autonomous Switching?

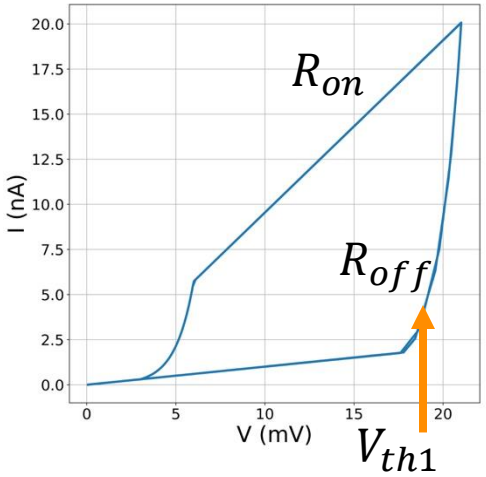
Memristive Integrate-and-Fire Neuron



MIF1

A volatile memristor





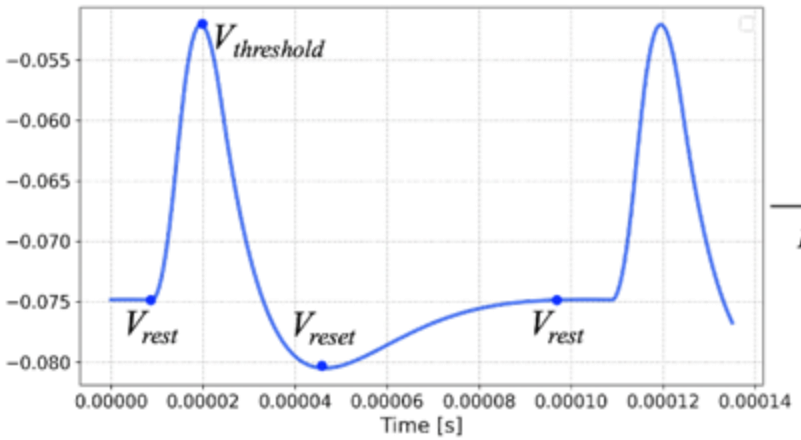
$V(t)$ increases →
V_M reaches memristor threshold value V_{th1} →
R_M switches from R_{off} to R_{on} →
V_M falls from $R_{off}C$ to $R_{on}C$ towards V_{rest} →
Switch back to R_{off} before next input

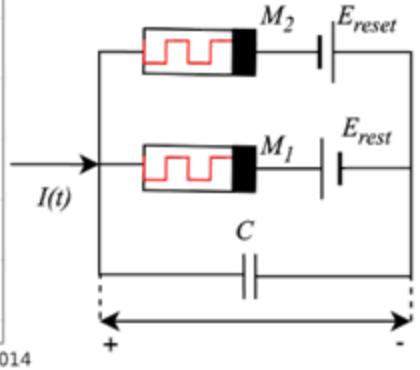
1. **Memristive Integrate-and-Fire (MIF)** are more efficient, compact than **CMOS-based** or hybrid approach
2. **Compatible** with memristor crossbar/RRAM
3. Use **commercial available** memristor to conduct experiments

***Xinming Shi**, et al, MIND, 2025*

New Computing Paradigm of Neuromorphic Computing

➢ Memristive Neuron





MIF Neuron Model

$$\mathbf{I} = \mathbf{G} \times \mathbf{V},$$

$$I_n = \sum_{i=1}^{m} v_i \cdot w_{ni}.$$

$$v^{t+1} = \frac{I^t - G_1^t(v^t - E_{\text{rest}}) - G_2^t(v^t - E_{\text{reset}})}{C} + v^t,$$

Memristor model

$$\begin{cases} i = Gv_M = G\dfrac{1}{1+GR}v = G_1 v, \\ \dot{x}_{\text{MR}} = \dfrac{1}{\tau}\left(1 + e^{-\beta\left(\frac{1}{1+RG}v - V_{\text{ON}}\right)}\right)(1 - x_{\text{MR}}) \\ \qquad\quad - \dfrac{1}{\tau}\left(1 - \left(1 + e^{-\beta\left(\frac{1}{1+RG}v + V_{\text{OFF}}\right)}\right)^{-1}\right) x_{\text{MR}} \end{cases}$$

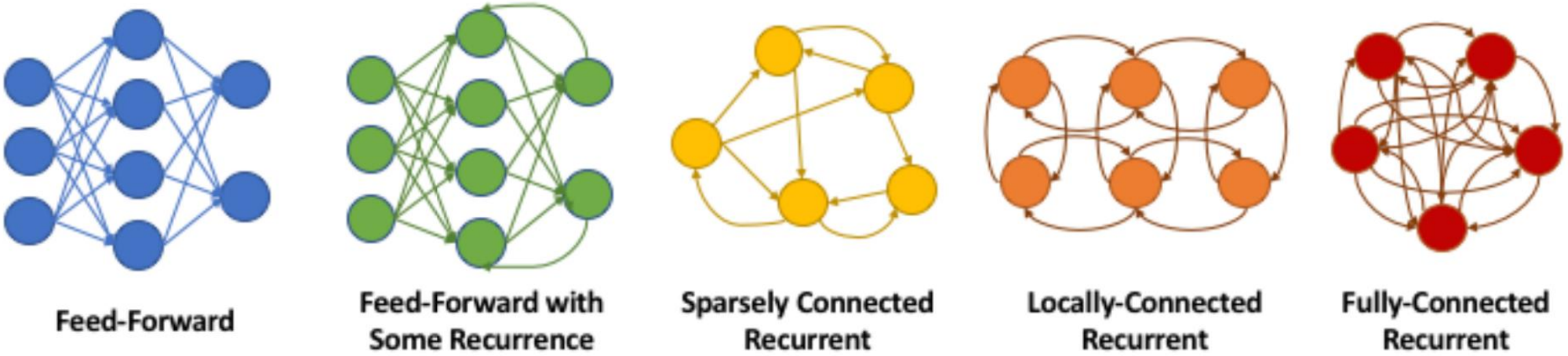
$$x_1^{t+1} = \frac{1}{\tau_1}\left(\frac{1 - x_1^t}{1 + \exp\left(\frac{v_{\text{on1}} - (v^t - E_{\text{rest}})}{V_{\text{th}} \cdot k_{\text{v}}}\right)} - \frac{x_1^t}{1 + \exp\left(\frac{(v^t - E_{\text{rest}}) - v_{\text{off1}}}{V_{\text{th}} \cdot k_{\text{v}}}\right)}\right) + x_1^t$$

- ✓ Use **commercial available** memristor to conduct experiments
- ✓ **Compatible** with memristor crossbar/RRAM
- ✓ NO extra ADC/DAC required

Xinming Shi, et al, MIND, 2025

New Computing Paradigm of Neuromorphic Computing

➢ Network

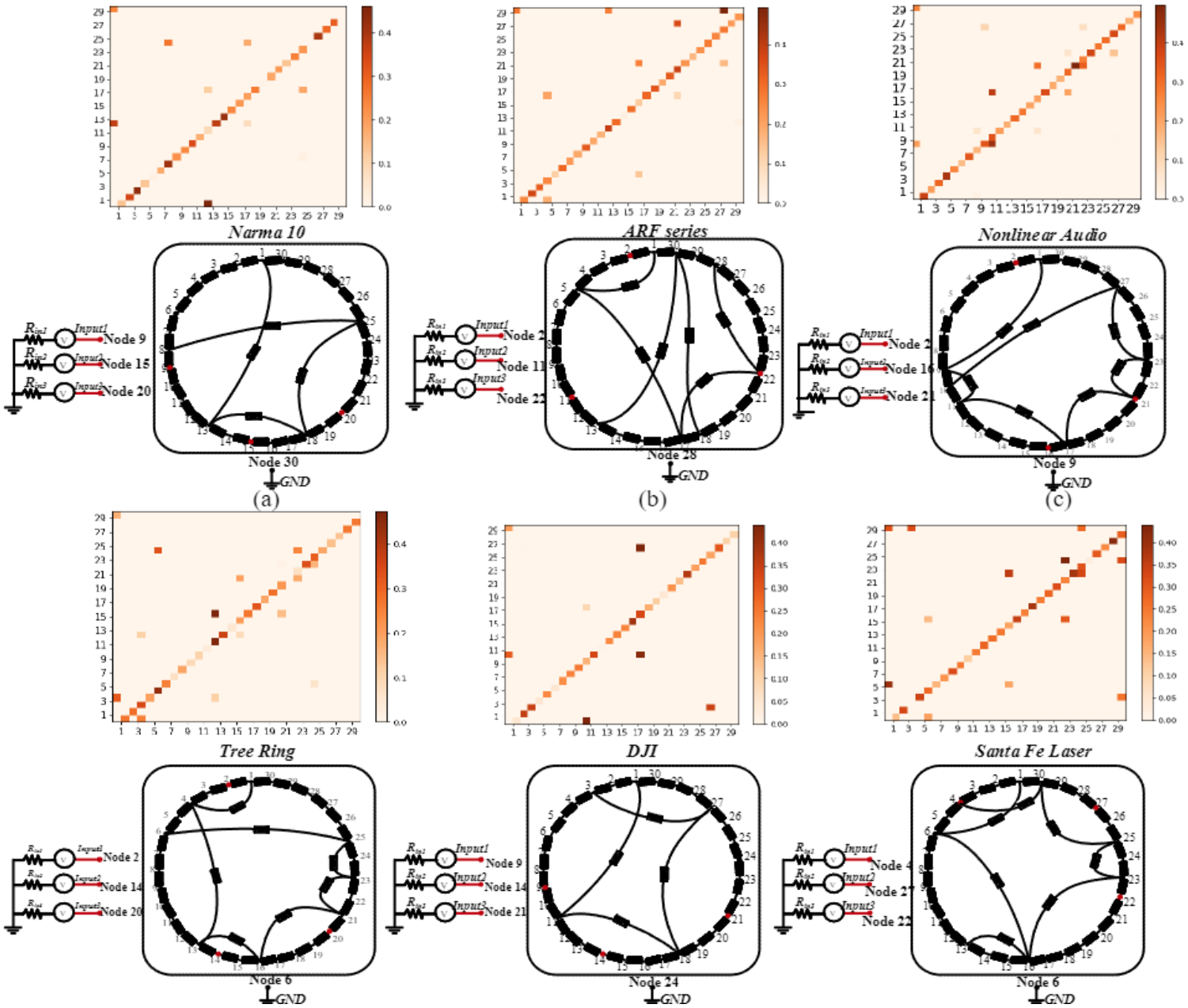


Different network topologies that might be desired for neuromorphic systems. Determining the level of connectivity that is required for a neuromorphic implementation and then finding the appropriate hardware that can accommodate that level of connectivity is often a non-trivial exercise.

Mehonic, Adnan, and Anthony J. Kenyon. "Brain-inspired computing needs a master plan." *Nature* 604.7905 (2022): 255-260.
Schuman, Catherine D., et al. "Opportunities for neuromorphic computing algorithms and applications." *Nature Computational Science* 2.1 (2022): 10-19.

New Computing Paradigm of Neuromorphic Computing

➢ Network

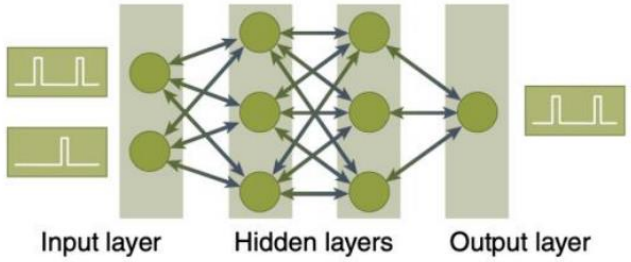


Xinming Shi, et al, TNNLS, 2025

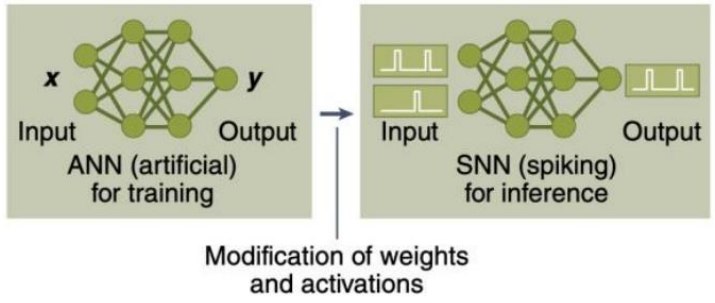
New Computing Paradigm of Neuromorphic Computing

➢ Learning Rules and Training Methods

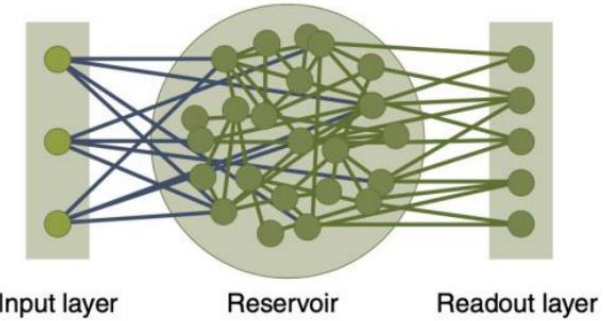
a (Quasi) Backpropagation



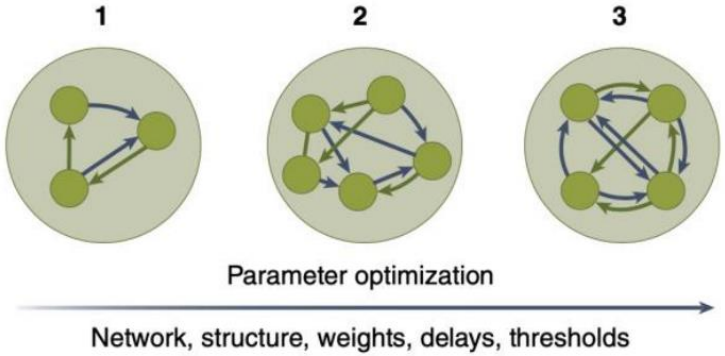
b Mapping Post-training



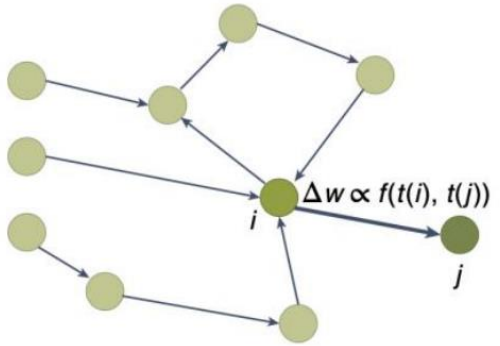
c Reservoir Computing



d Evolutionary Optimisation



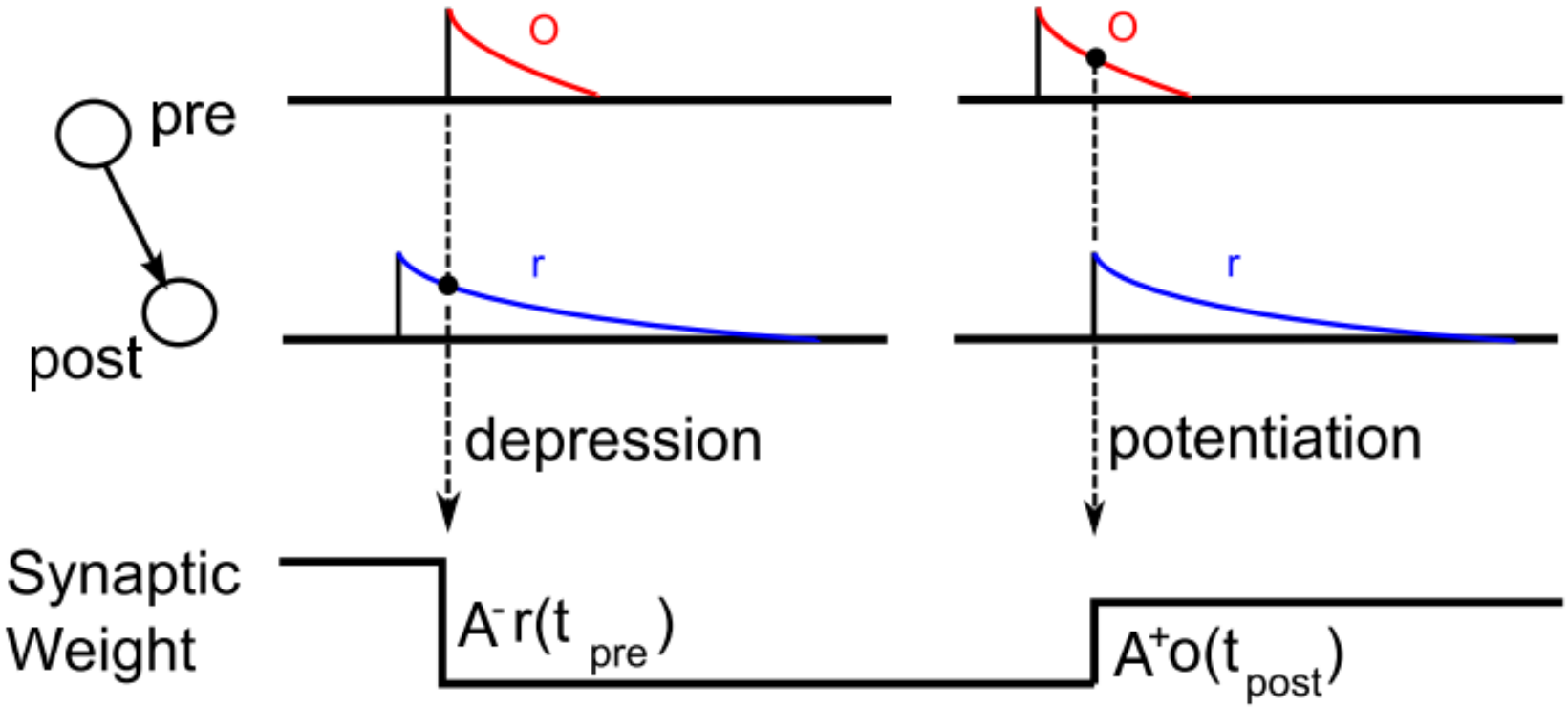
e Hebbian Rules, STDP



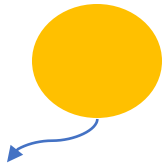
ALGORITHMS PROS AND CONS

Algorithm Class	Any Model	Device Quirks	Complex to Implement	On-Line	Fast Time to Solution	Demonstrated Broad Applicability	Biologically-Inspired or Plausible
Back-Propagation	No	No	Yes	No	Yes	Yes	No
Evolutionary	Yes	Yes	No	No	No	Yes	Maybe
Hebbian	No	Yes	No	Yes	Maybe	No	Yes
STDP	No	Yes	Maybe	Yes	Maybe	No	Yes

➢ Learning Rules and Training Methods--STDP



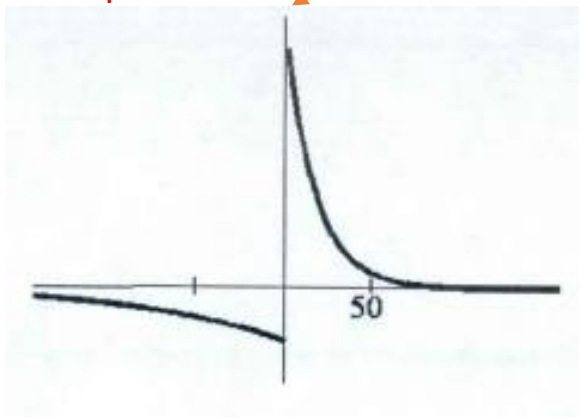
➢ Learning Rules and Training Methods--STDP

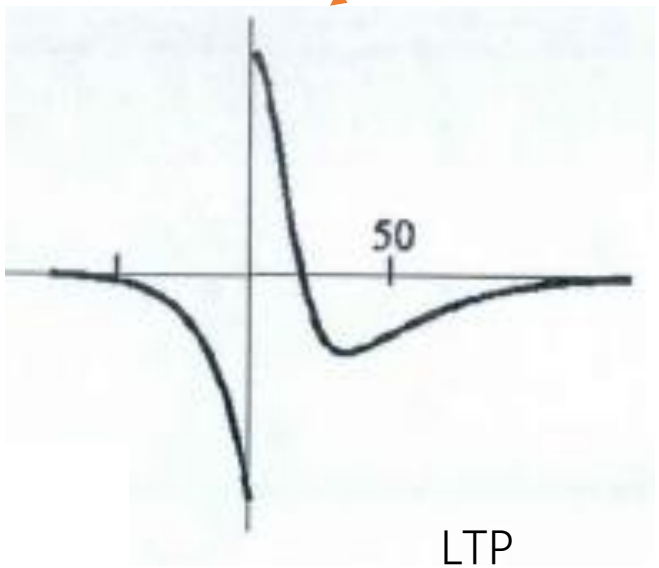


transmitter

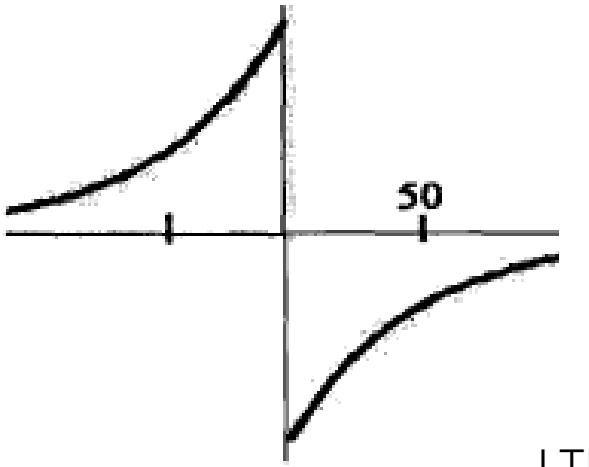
excitatory synapses:
trigger the next neuron

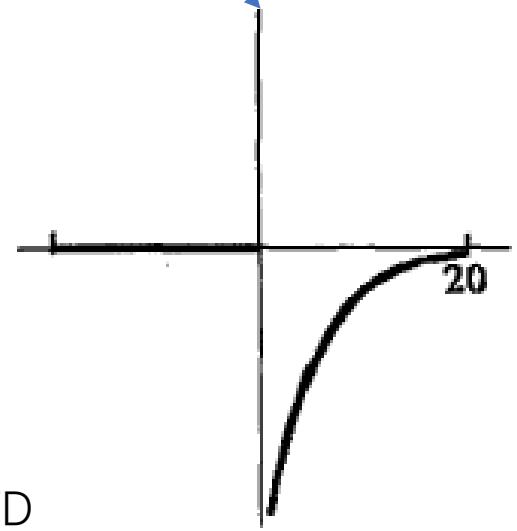
E-stdp





LTP

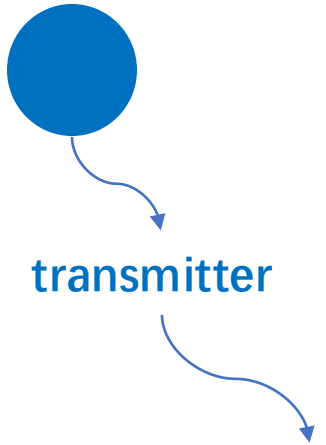




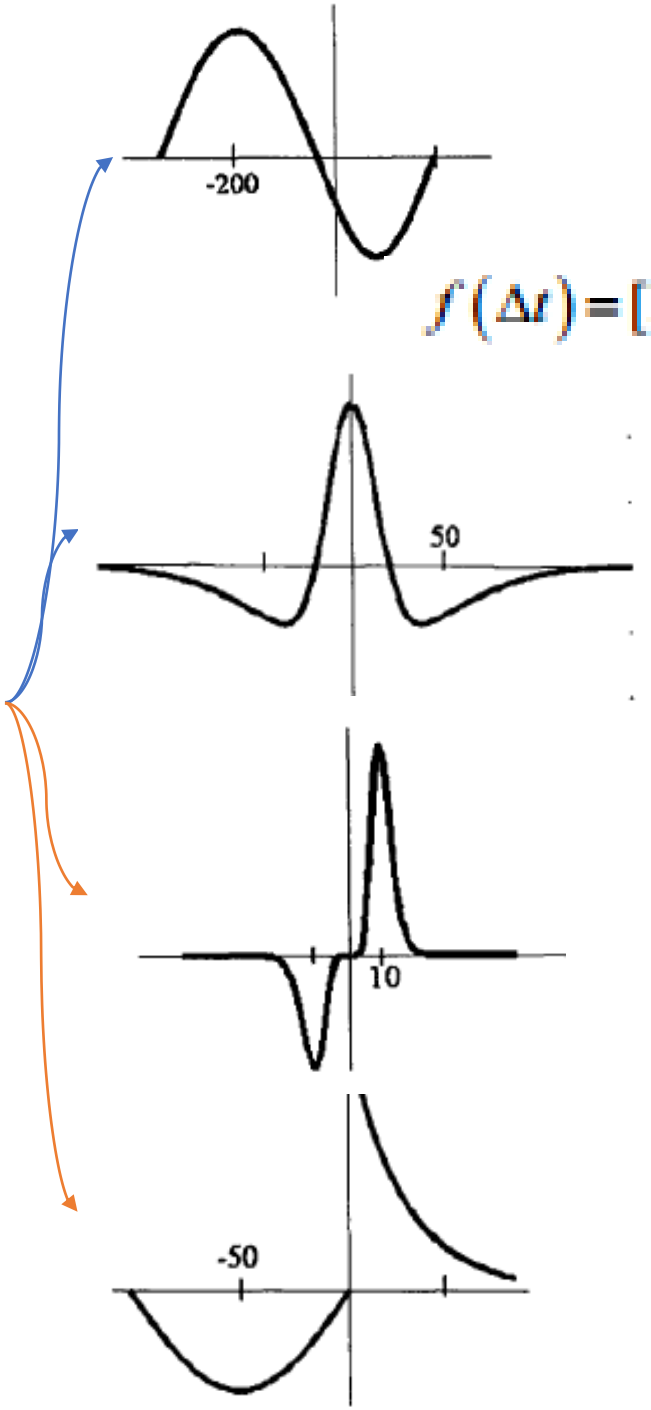
LTD

$$\Delta g_{ij} = \overline{g} \cdot f(\Delta t)$$

$$f(\Delta t) = \begin{cases} -A^{-} \cdot \exp\left(\dfrac{\Delta t}{\tau^{-}}\right), & if \ \Delta t \le 0 \\ A^{+} \cdot \exp\left(-\dfrac{\Delta t}{\tau^{+}}\right), & if \ \Delta t > 0 \end{cases}$$



inhibitory synapses:
inhibit the next neuron



$$f(\Delta t) = [1.5 \times \exp(-0.004\Delta t^2) - 0.5 \times \exp(-0.0003\Delta t^2)] / \eta$$

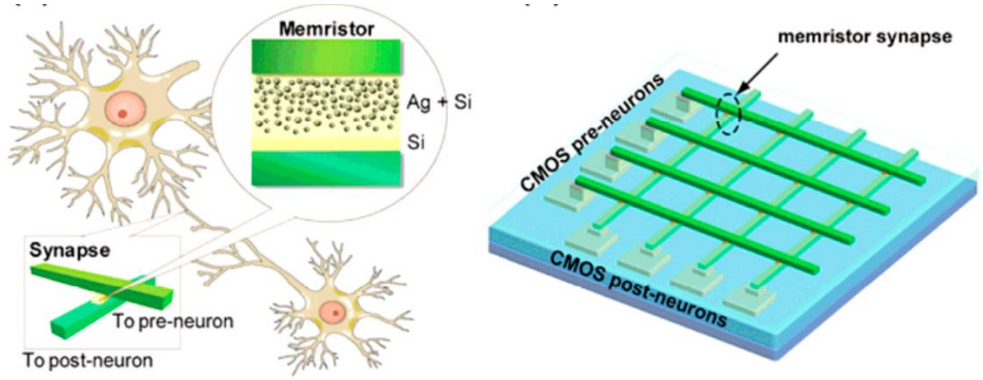
$$w_{ji\_new} = w_{ji\_old} + \eta \bullet w_{ji\_old} F(\Delta t)$$

$$f(\Delta t) = \begin{cases} [a_1 \cdot (\Delta t)^{10} e^{a_2 \Delta t}] / \eta, & if \ \Delta t < 0 \\ [a_3 \cdot (\Delta t)^{10} e^{a_4 \Delta t}] / \eta, & if \ \Delta t > 0 \end{cases}$$

New Computing Paradigm of Neuromorphic Computing

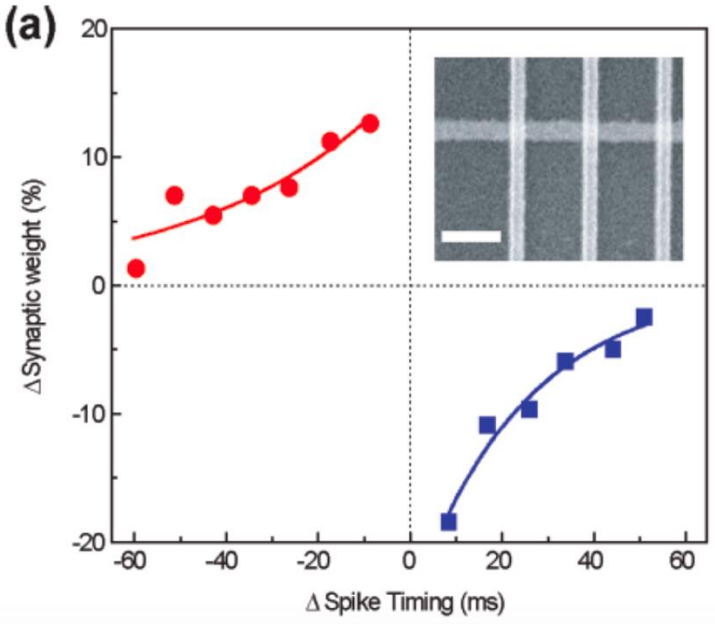
➢ Learning Rules and Training Methods--STDP

Jo *et al.*, Nanoletters 2010

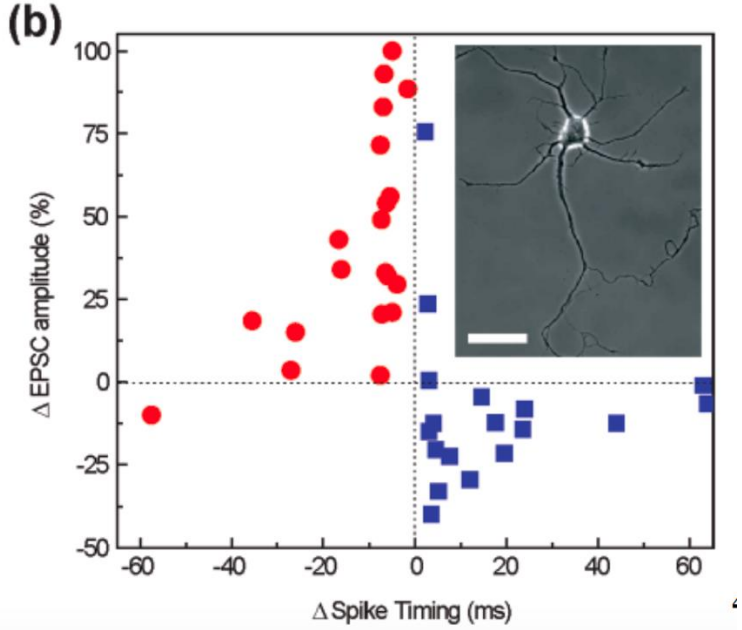


STDP (Spike-Timing-Dependent Plasticity) is a biological learning rule where the strength of a synapse is adjusted based on the precise timing between pre- and post-synaptic spikes.

Memristor STDP curve



Bi & Poo 1998

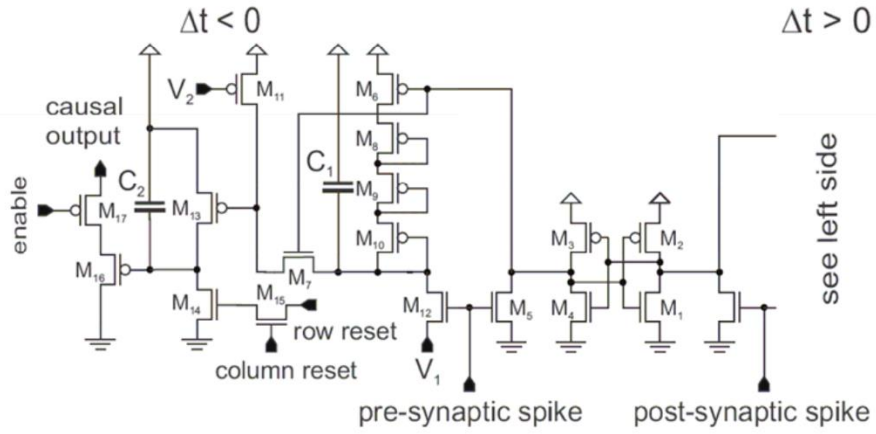


STDP (Spike-Timing-Dependent Plasticity) enables unsupervised learning because synaptic strengths self-adjust based on the timing correlation of pre- and post-synaptic spikes, allowing the network to autonomously discover patterns in input data without external labels.

New Computing Paradigm of Neuromorphic Computing

➢ Learning Rules and Training Methods--STDP

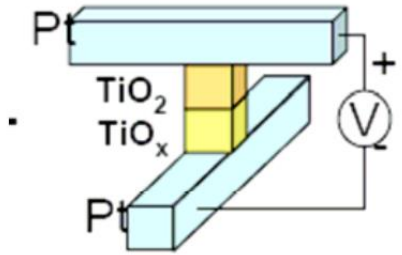
CMOS Implementation



Schemmel *et al.*, IJCNN 2006

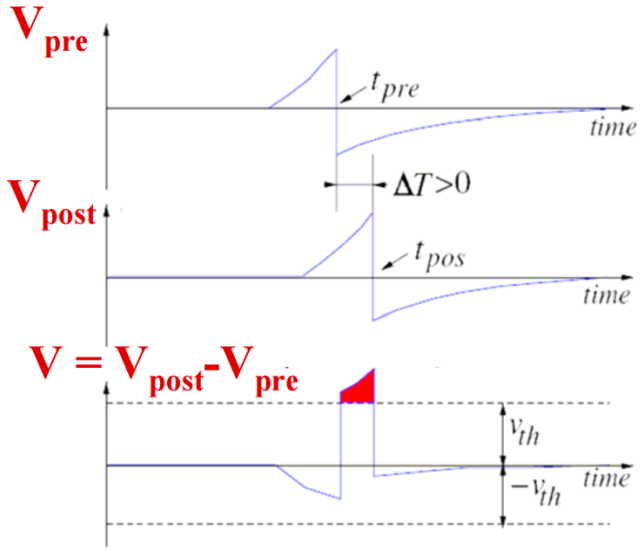
1 memristor directly implements STDP

< 30x30 nm²



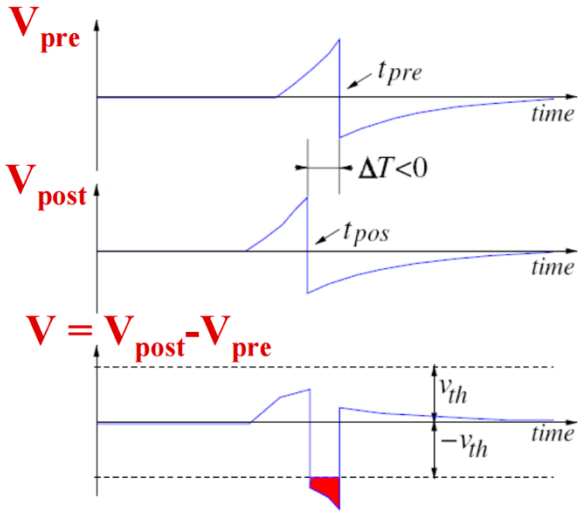
• Jo *et al.*, Nanoletters 2010

Conductance increase



potentiation

Conductance decrease

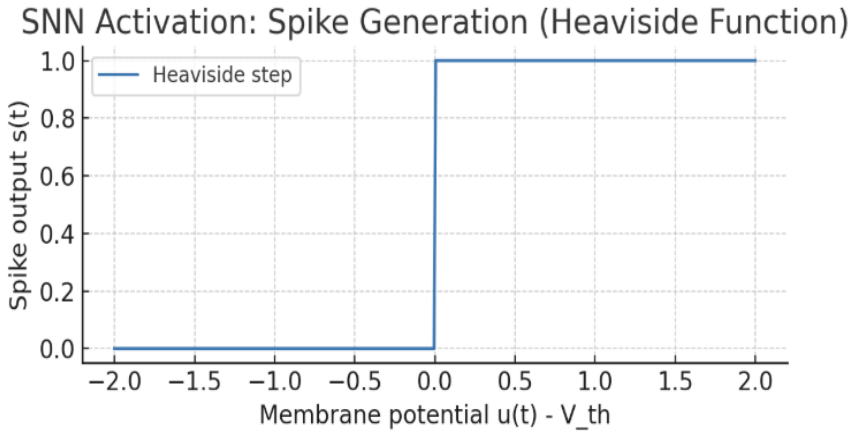


depression

Linarres-Barranco *et al.*, frontiers in Neuroscience, 2011

➢ Learning Rules and Training Methods-- Backpropagation

Derivative Problem In SNN



In Spiking Neural Networks (SNNs), spike firing is controlled by the **Heaviside step function**:

$$s(t) = H(u(t) - V_{\text{th}})$$

The derivative of the step function is **zero almost everywhere**, and **undefined at the threshold point**. During backpropagation:

$$\frac{\partial s}{\partial u} = 0$$

→ Gradients vanish completely, so the network cannot learn.

➢ Learning Rules and Training Methods-- Backpropagation

Mathematically, during backpropagation the original derivative $H'(x)$ is replaced by a smooth surrogate function $\sigma'(x)$:

$$\frac{\partial L}{\partial u} \approx \frac{\partial L}{\partial s} \cdot \sigma'(u - V_{\text{th}})$$

- Here, $\sigma'(x)$ is a chosen smooth function, for example:
  - **Sigmoid derivative**:

$$\sigma'(x) = \frac{\alpha e^{-\alpha x}}{(1 + e^{-\alpha x})^2}$$

  - **Piecewise linear (triangle/hat) function**:

$$\sigma'(x) = \max(0, 1 - |x|)$$

  - **Cosh-based surrogate (used in SLAYER)**:

$$\sigma'(x) = \frac{\alpha}{2\cosh^2(\alpha x/2)}$$

- Biological mismatch
- Approximation error
- Computational overhead
- Hardware–software gap

Surrogate gradients solve the gradient-vanishing problem in SNNs, making supervised training feasible while still keeping the discrete spike dynamics in the forward path.

New Computing Paradigm of Neuromorphic Computing

➢ Learning Rules and Training Methods-- Backpropagation

Gradients naturally emerge from the physical ODEs of memristive integrate-and-fire (MIF) circuits

$$\begin{cases} i = Gv_M = G\dfrac{1}{1+GR}v = G_1 v, \\ \dot{x}_{\mathrm{MR}} = \dfrac{1}{\tau}\left(1+e^{-\beta\left(\frac{1}{1+RG}v - V_{\mathrm{ON}}\right)}\right)(1-x_{\mathrm{MR}}) \\ \qquad\quad -\dfrac{1}{\tau}\left(1-\left(1+e^{-\beta\left(\frac{1}{1+RG}v + V_{\mathrm{OFF}}\right)}\right)^{-1}\right)x_{\mathrm{MR}} \end{cases}$$

Commercial memristor model

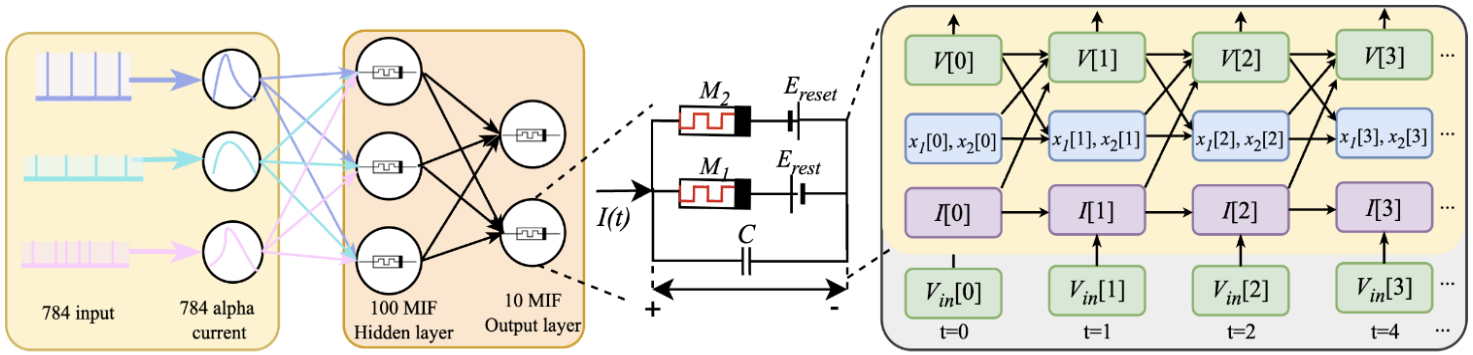


Fig. 1. A fully memristive spiking neural network (MSNN) architecture and its computational graph based on memristive dynamics.

All variables—including v, x1, x2, and τ —are embedded in a differentiable computational graph. The gradients naturally emerge from the physical ODEs of memristive integrate-and-fire (MIF) circuits. The result is a forward-mode simulation framework **that preserves the physical behavior of the MIF neuron while enabling efficient, gradient-based training in neuromorphic learning systems.**

$$v^{t+1} = \frac{I^t - G_1^t(v^t - E_{\mathrm{rest}}) - G_2^t(v^t - E_{\mathrm{reset}})}{C} + v^t,$$

$$x_1^{t+1} = \frac{1}{\tau_1}\left(\frac{1-x_1^t}{1+\exp\left(\frac{v_{\mathrm{on1}}-(v^t-E_{\mathrm{rest}})}{V_{\mathrm{th}}\cdot k_{\mathrm{v}}}\right)} - \frac{x_1^t}{1+\exp\left(\frac{(v^t-E_{\mathrm{rest}})-v_{\mathrm{off1}}}{V_{\mathrm{th}}\cdot k_{\mathrm{v}}}\right)}\right) + x_1^t$$

***Xinming Shi**, et al, MIND, 2025*

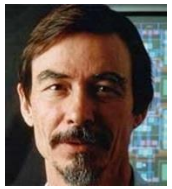
$$\tau_{\mathrm{syn}}\frac{\mathrm{d}I}{\mathrm{d}t} = a - I, \quad \tau_{\mathrm{syn}}\frac{\mathrm{d}a}{\mathrm{d}t} = -a + W_i\sum_n \delta(t - t_i^n)$$

Quick recap

What is Neuromorphic Computing Hardware?

Why Neuromorphic Computing Hardware?

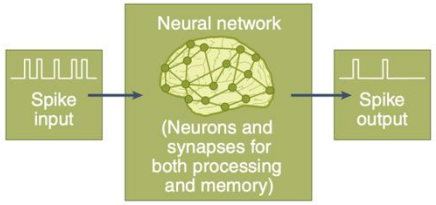
Referring to analogue VLSI mimicking biological neural systems.



New devices of NC: memristor

- **Nano resistance**
- **Tunable (multi-states of resistance available)**
- **Non volatile**
- **Non-linear : V < V_{th} read, V > V_{th} write**

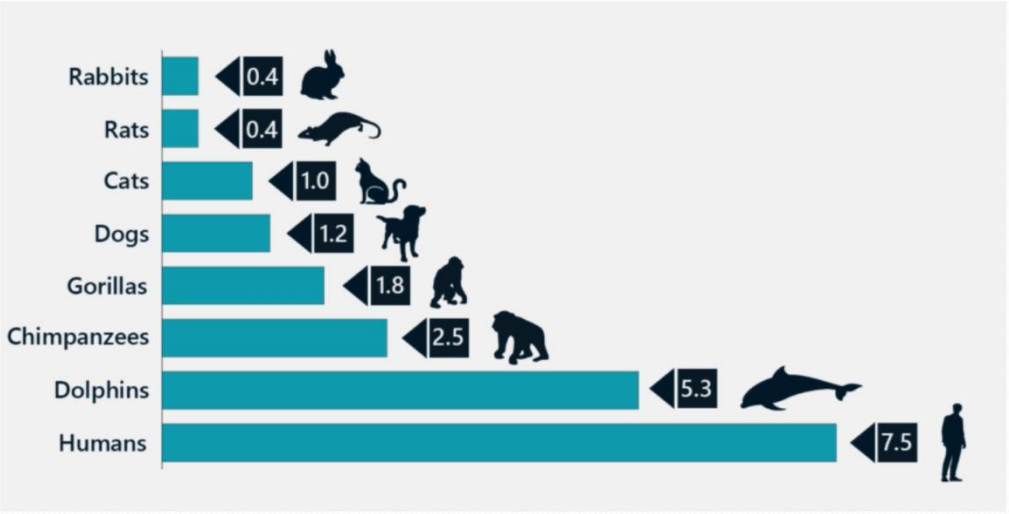
New computing paradigm of NC

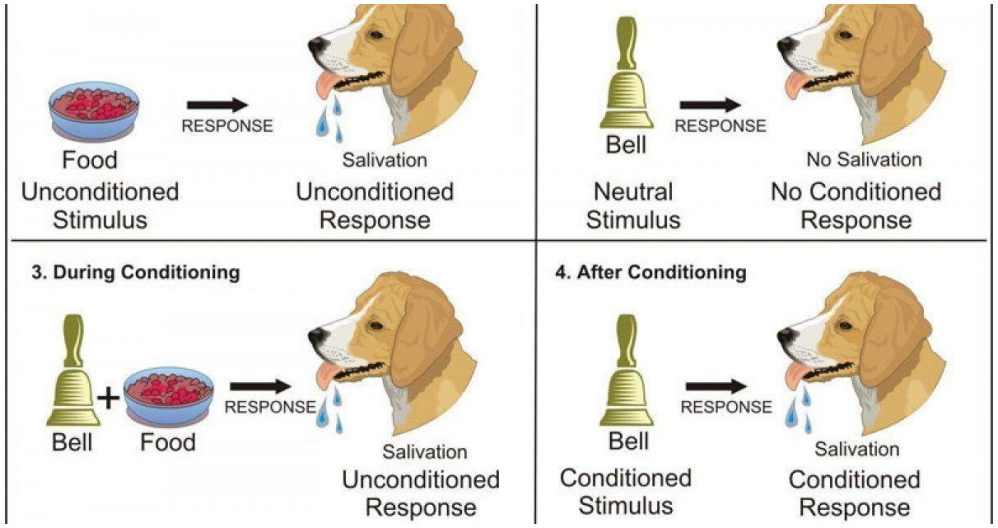


- **synapse**
- **neuron**
- **network**
- **learning rules and training methods**

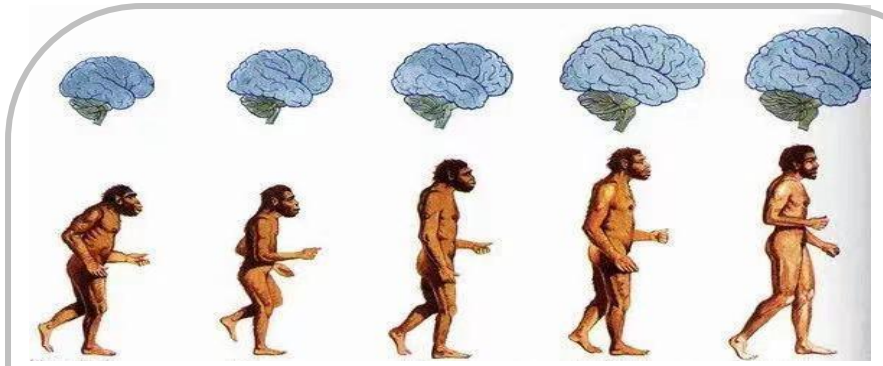
Why evolving the brain-inspired computing system?

Brains comparisons

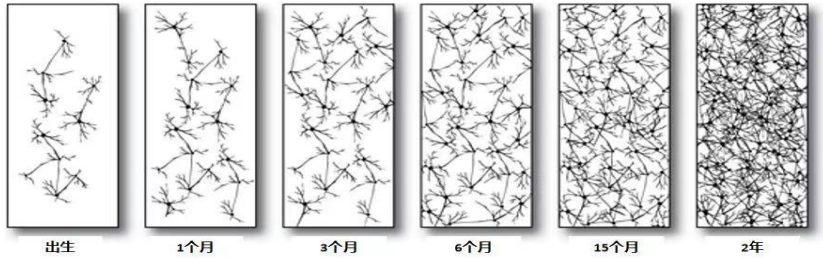




Evolution of the human brain



The brain has been through a long evolutionary process



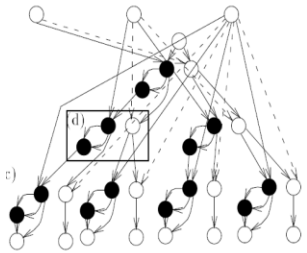
The process of neural circuit growth and development

Dogs can learn certain tasks, but because evolution has not enabled them with higher-level cognitive structures, they will never develop language or abstract reasoning. This highlights that **learning alone is insufficient—true breakthroughs in intelligence require evolutionary progress**.

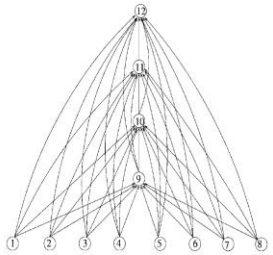
Falk, Dean. "Evolution of the primate brain." *Handbook of palaeoanthropology* 2 (2007): 1133-62.

Research Background

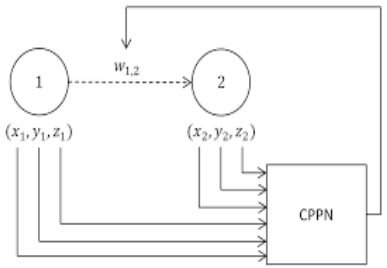
➢ Drawing on the inspiration of the nature: evolutionary computing and neural networks



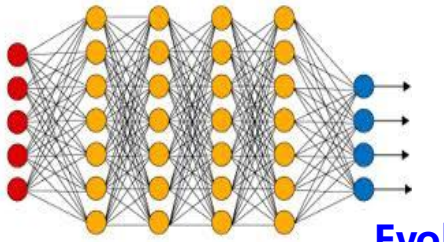
1994
F.Gruau
artificial developmental neural network



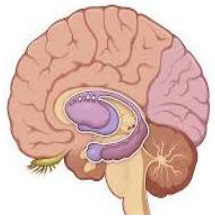
1997
Xin Yao
Evolutionary NN



2008
Stanley
HyperNEAT

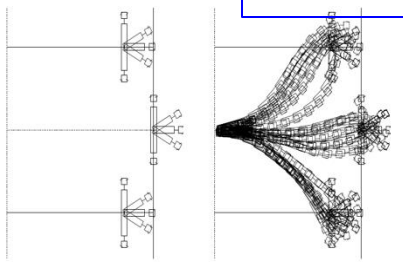


Evolution of DL



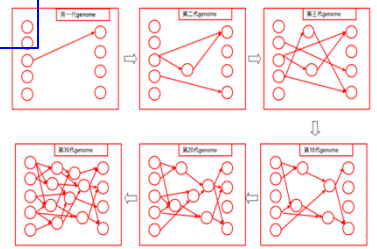
Evolution of brain-inspired computing

1994
Ronald
GA

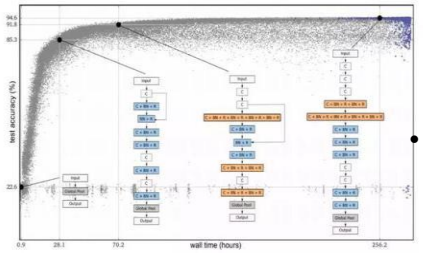


1990s
First research boom!

2002
Stanley
NEAT



2017
Real
Large-scale evolutionary image classification



- Neuroevolution
- Hyperparametric Optimisation
- Architecture Search
- Rule-based Reasoning...

...

2018-Present
Second Research Boom!

Evolving Artificial Neural Networks

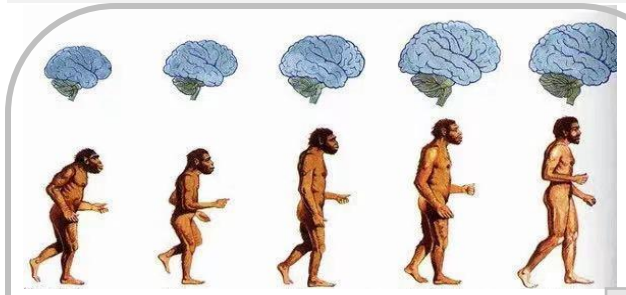
XIN YAO, SENIOR MEMBER, IEEE

Invited Paper

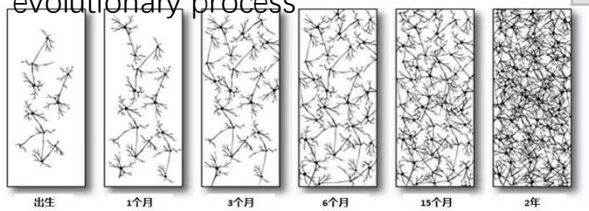
1999 Xin Yao

How to evolve the brain-inspired computing system?

Evolution of the human brain



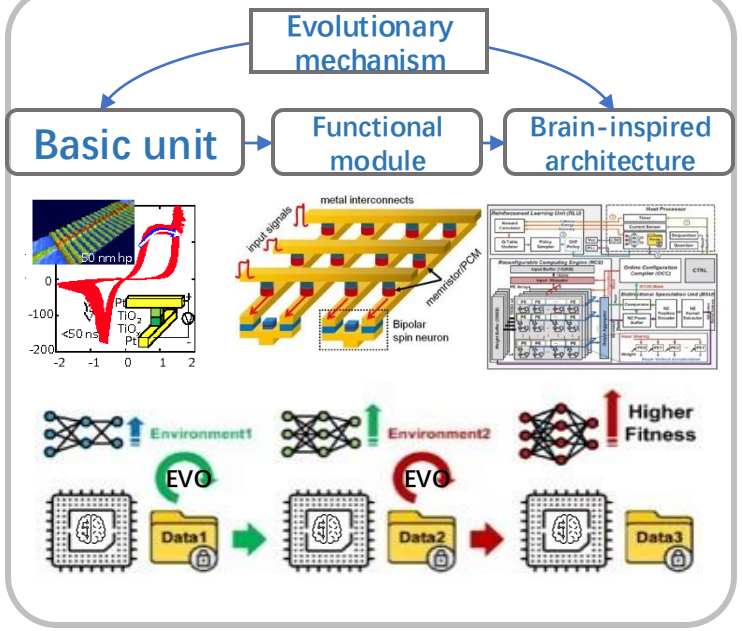
The brain has been through a long evolutionary process



The process of neural circuit growth and development

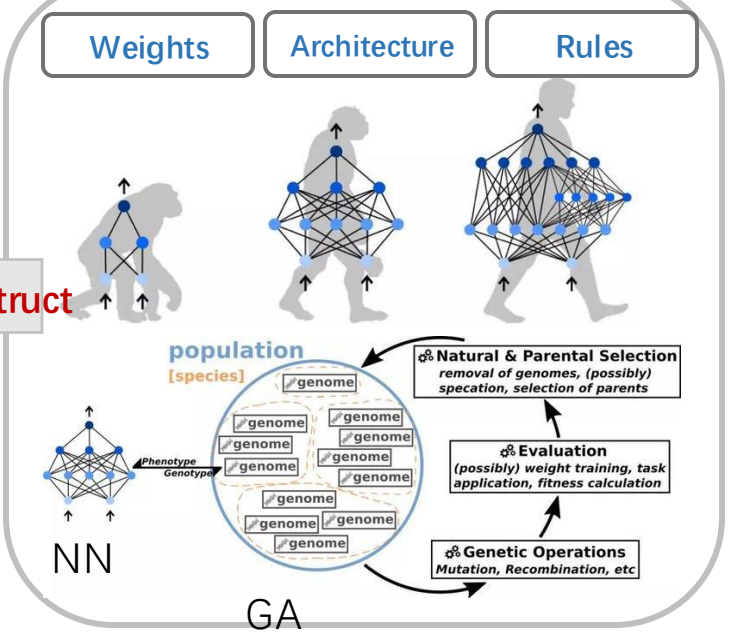
inspire

Evolvable brain-inspired computing systems



construct

Evolution of neural networks



■ Evolve the topology of NN

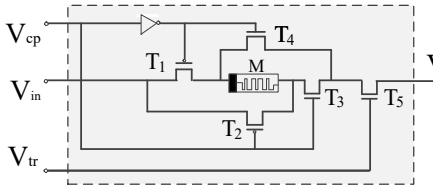
➢ 4T1R Architecture Evolution

A reconfigurable memristive unit built with 4 transistors and 1 memristor, where different signal states lead to different circuit configurations.

Providing scalable platform for the circuit evolution

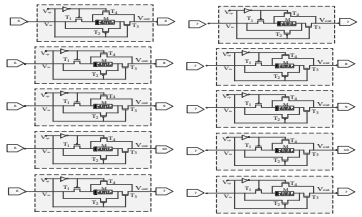
Evolving the duration of different signals

Applying evolutionary algorithms to self-adapt and evolve memristive array computing systems.



Reconfigurable unit

platform



Reconfigurable memristive reservoir computing system

NxN bool matrix → reservoir topology

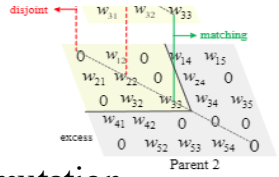
NxN ta matrix → the duration of positive voltage applied
NxN tb matrix → the duration of negative voltage applied

Reservoir weights

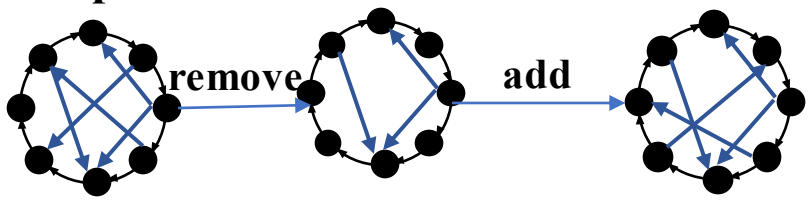
Sparse initialization + Circuit evolution

Six types of mutation and crossover

- Weight mutation
- Add node
- Delete node
- Jump mutation
- Sparsity mutation
- Input/GND node mutation



Adaptive connection based on evolution



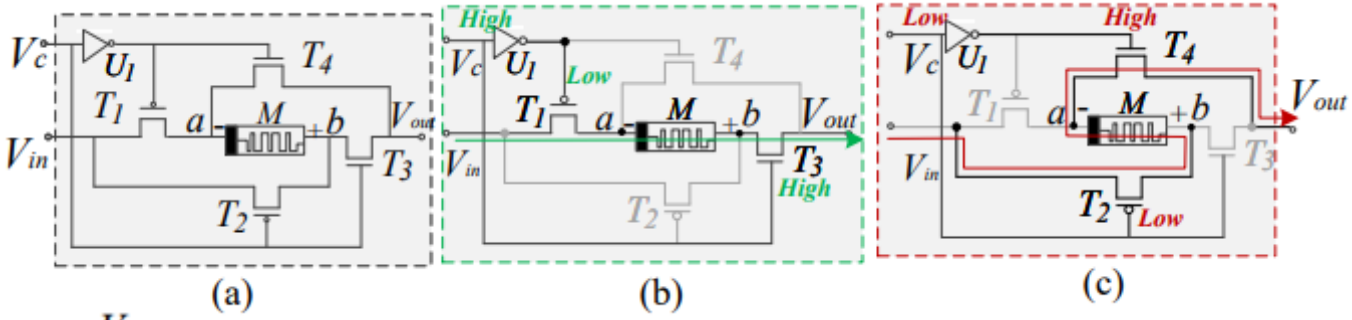
- ✓ **On-chip Evolvable Memristive Reservoir Circuit: proposed circuit utilizes memristor currents to enhance the adaptability and performance of reservoir computing systems.**
- ✓ **Scalable Evolutionary Algorithm: directly evolve the configuration signals of memristors rather than the memristive states themselves, addressing issues related to device variance which can affect circuit performance.**
- ✓ **The evolutionary algorithm incorporates mechanisms to maintain sparse topology and ensure legal configurations in accordance with circuit laws.**

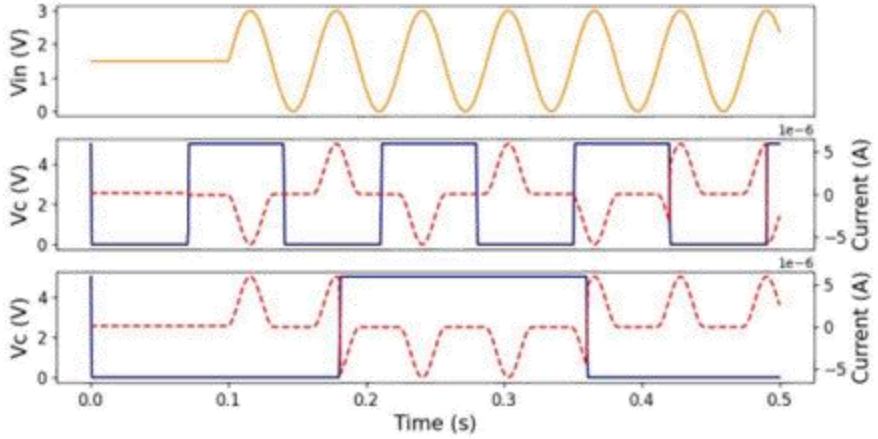
Xinming Shi, Leandro L. Minku and Xin Yao, "Evolving Memristive Reservoir," *IEEE Transactions on Neural Networks and Learning Systems,* Early Access, pp. 1-15, May 2023, doi: 10.1109/TNNLS.2023.3270224.

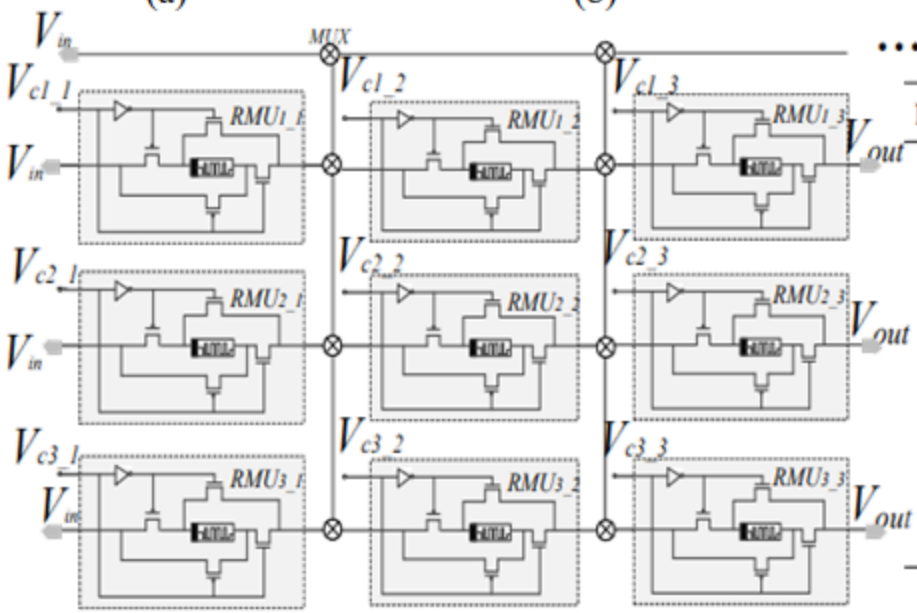
■ Evolve the Topology of NN

➢ 4T1R Architecture Evolution

Using the width of the control signal to represent the connection states instead of the circuit wires.





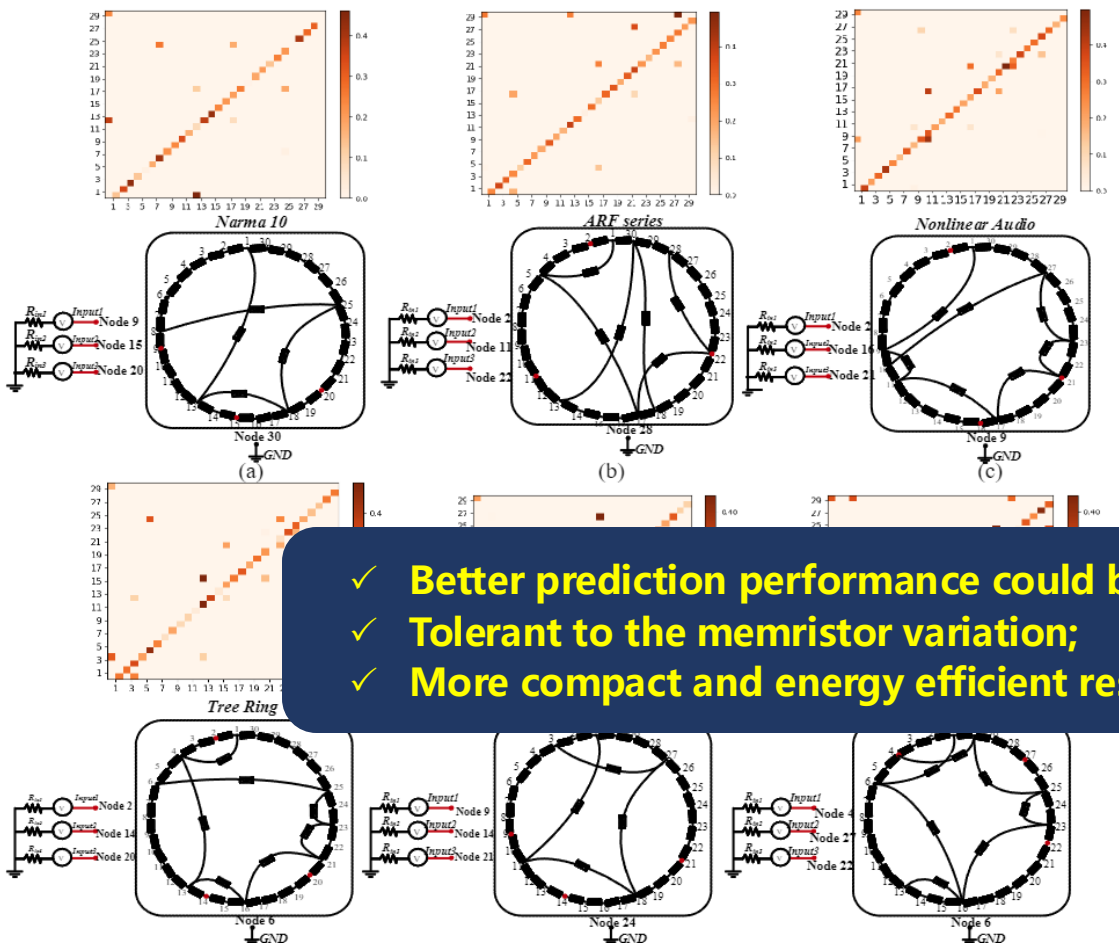


NO.	$V_{c1\text{-}1}$	$V_{c1\text{-}2}$	$V_{c1\text{-}3}$	$V_{c2\text{-}1}$	$V_{c2\text{-}2}$	$V_{c2\text{-}3}$	$V_{c3\text{-}1}$	$V_{c3\text{-}2}$	$V_{c3\text{-}3}$	Equivalent Circuits
1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	
2	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	
3	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	
4	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	

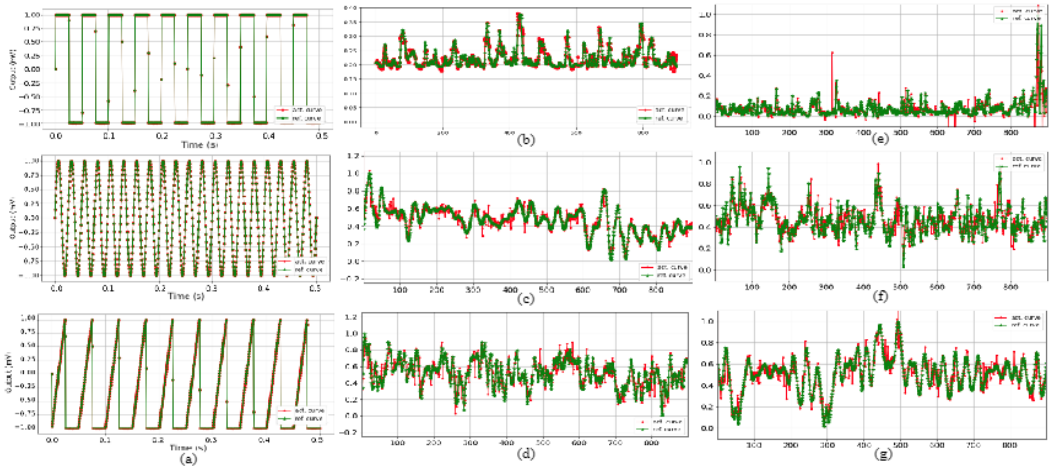
The key idea is that the MUX unit in conjunction with the memristive crossbar array allows for dynamic reconfiguration of logic functions.

■ Evolutionarily Reconfigurable Hardware

➢ 4T1R Architecture Evolution

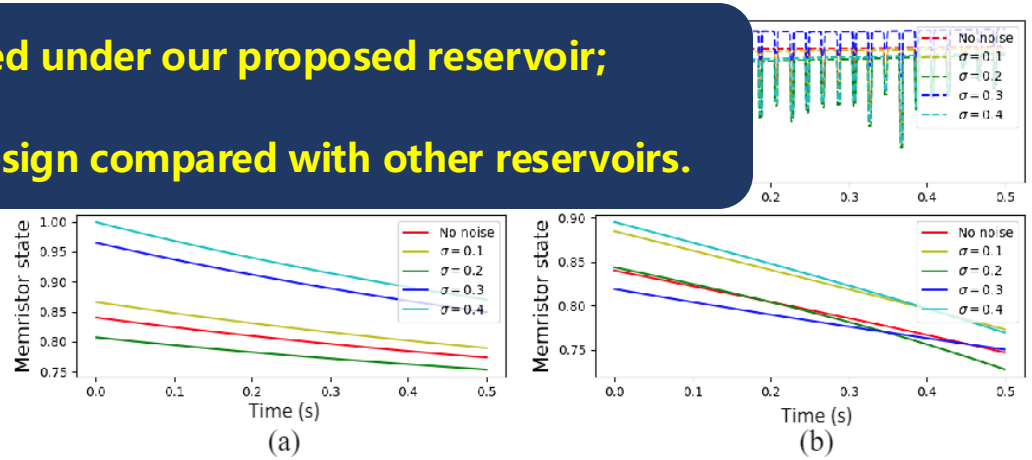


Evolved reservoir topology and configuration states



Evolved reservoir topology and configuration states

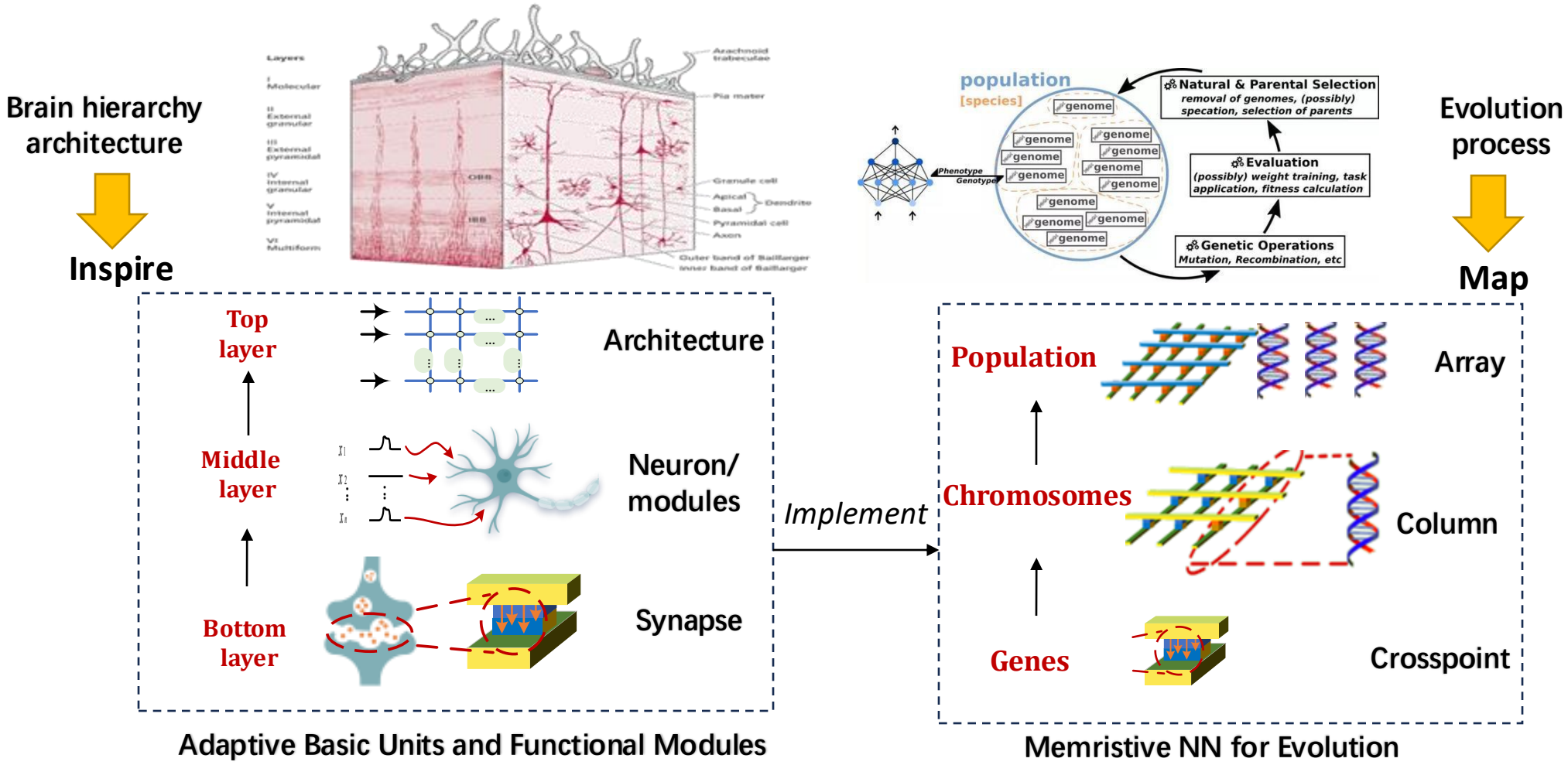
- ✓ Better prediction performance could be obtained under our proposed reservoir;
- ✓ Tolerant to the memristor variation;
- ✓ More compact and energy efficient reservoir design compared with other reservoirs.



Noise injection experiments: memristor current is more tolerant to noise compared with memrsitance

Xinming Shi, Leandro L. Minku and Xin Yao, "Evolving Memristive Reservoir," *IEEE Transactions on Neural Networks and Learning Systems,* Early Access, pp. 1-15, May 2023, doi: 10.1109/TNNLS.2023.3270224.

Memristive Implementation of Evolution Algorithm

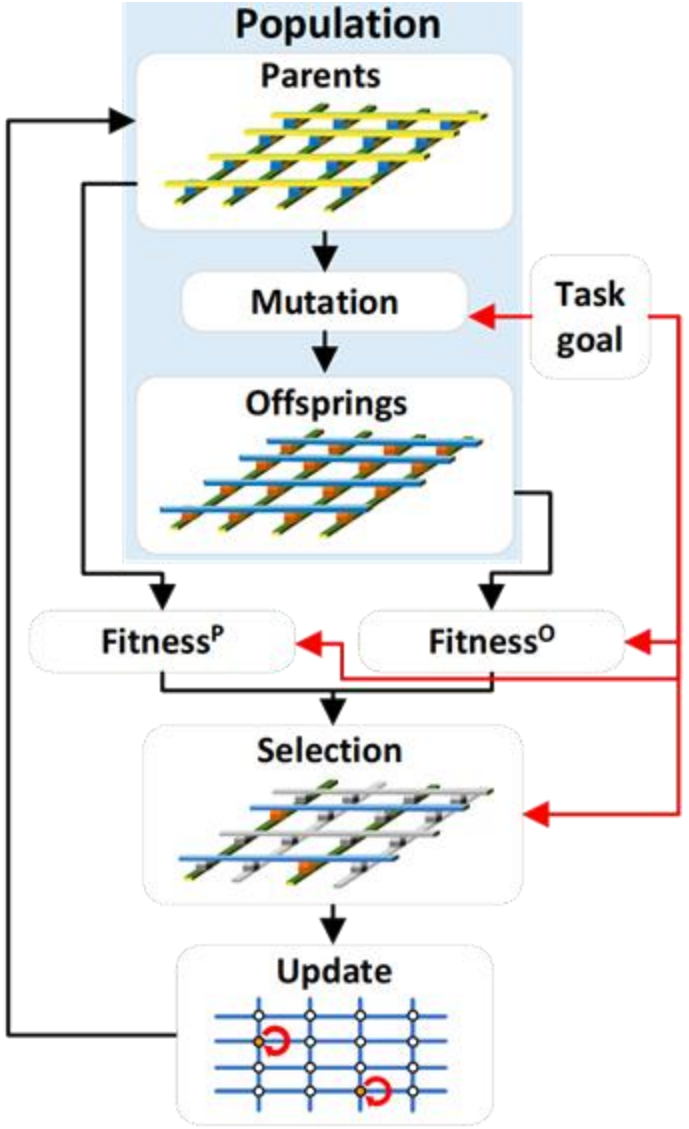


Zilu Wang, **Xingming Shi**, and Xin Yao, "A Brain-Inspired Hardware Architecture for Evolutionary Algorithms based on Memristive Arrays," in *ACM Transactions on Design Automation of Electronic Systems*, doi: 10.1145/3598421.

■ Memristive Implementation of Evolution Algorithm

➢ Memristive architecture implementation of evolutionary algorithm [3]

Framework of the architecture implementation



- **Parents stored in memristor array** – initial population of solutions.
- **Mutation module** – generates offspring by perturbing parent weights in parallel.
- **Fitness evaluation** – both parents and offspring are compared against task goals.
- **Selection module** – only fitter individuals are kept, enabling sparse updates.
- **Update module** – writes selected genes back into the memristor array.
- **Iterative loop** – repeats the cycle to realize hardware-accelerated evolutionary learning.

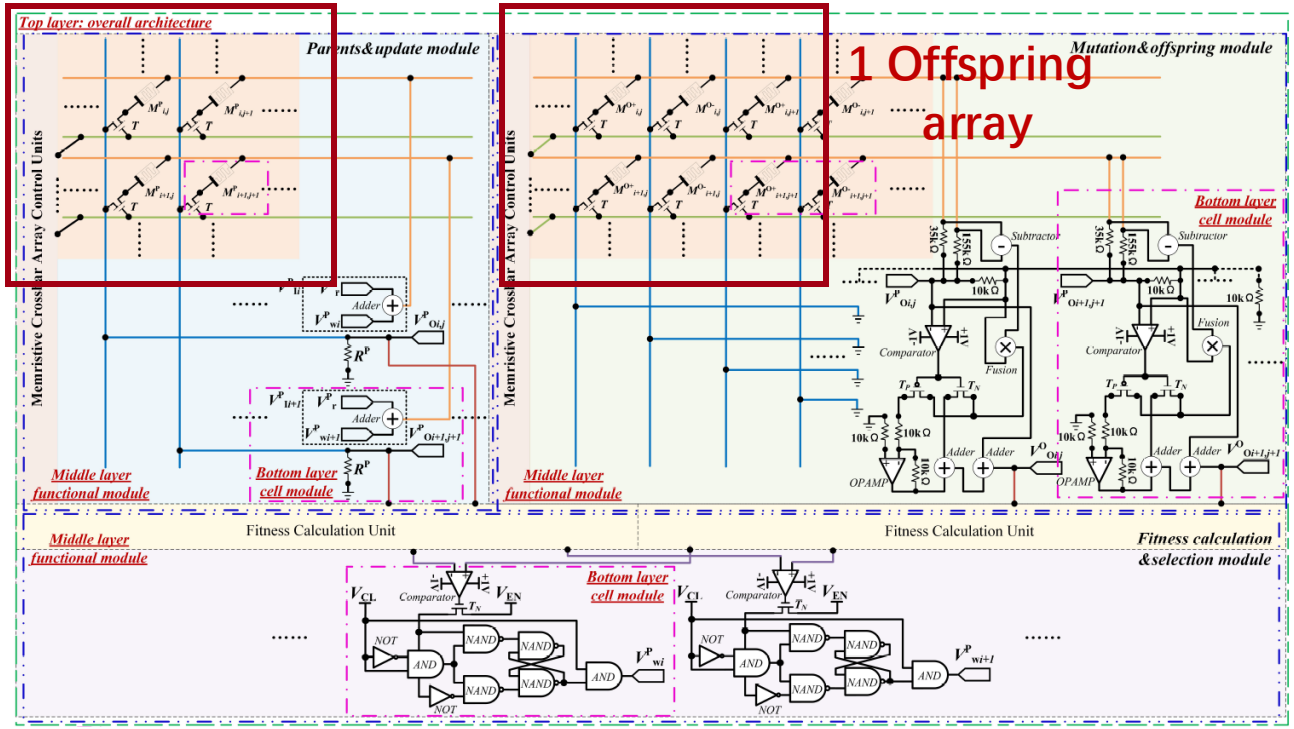
[3] Zilu Wang, **Xingming Shi**, and Xin Yao, "A Brain-Inspired Hardware Architecture for Evolutionary Algorithms based on Memristive Arrays," in *ACM Transactions on Design Automation of Electronic Systems*, doi: 10.1145/3598421.

Memristive Implementation of Evolution Algorithm

Memristive architecture implementation of evolutionary algorithm [3]

1 Parent array

1 Offspring array



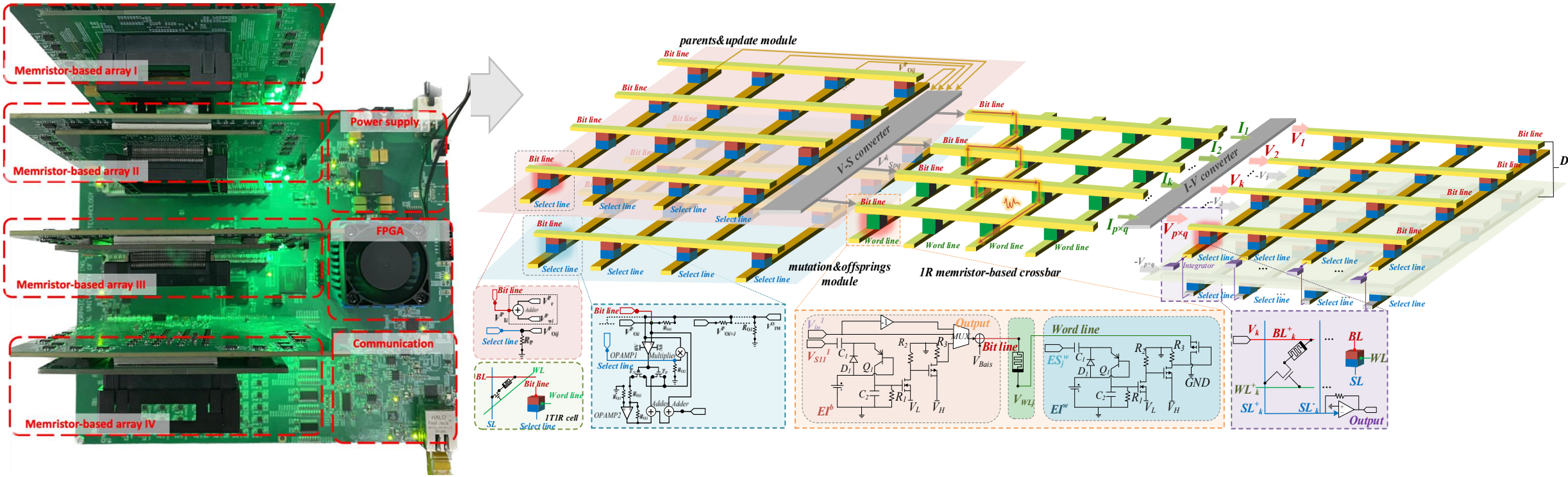
Circuit scheme of the memristive architecture

- This architecture consists of two arrays, one represents the parent, and another one represents the offspring.And each memristor represents one gene.
- Only the **one with worse fitness in paraent will be updated**, incurring the sparsity. Moreover, there is no need to change to the accurate value, we only ensure the changing direction is correct, incurring the approximation calculation.

[3] Zilu Wang, **Xingming Shi**, and Xin Yao, "A Brain-Inspired Hardware Architecture for Evolutionary Algorithms based on Memristive Arrays," in *ACM Transactions on Design Automation of Electronic Systems*, doi: 10.1145/3598421.

■ Memristive Implementation of Evolution Algorithm

➢ Memristive architecture implementation of evolutionary algorithm



✓ **Integrating the advantages of memristor arrays, such as in-memory computing, with sparse and approximate computing.**

Zilu Wang, **Xinming Shi**, and Xin Yao, "A Brain-Inspired Hardware Architecture for Evolutionary Algorithms based on Memristive Arrays," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 5, pp. 32, Sep. 2023, doi: 10.1145/3598421.

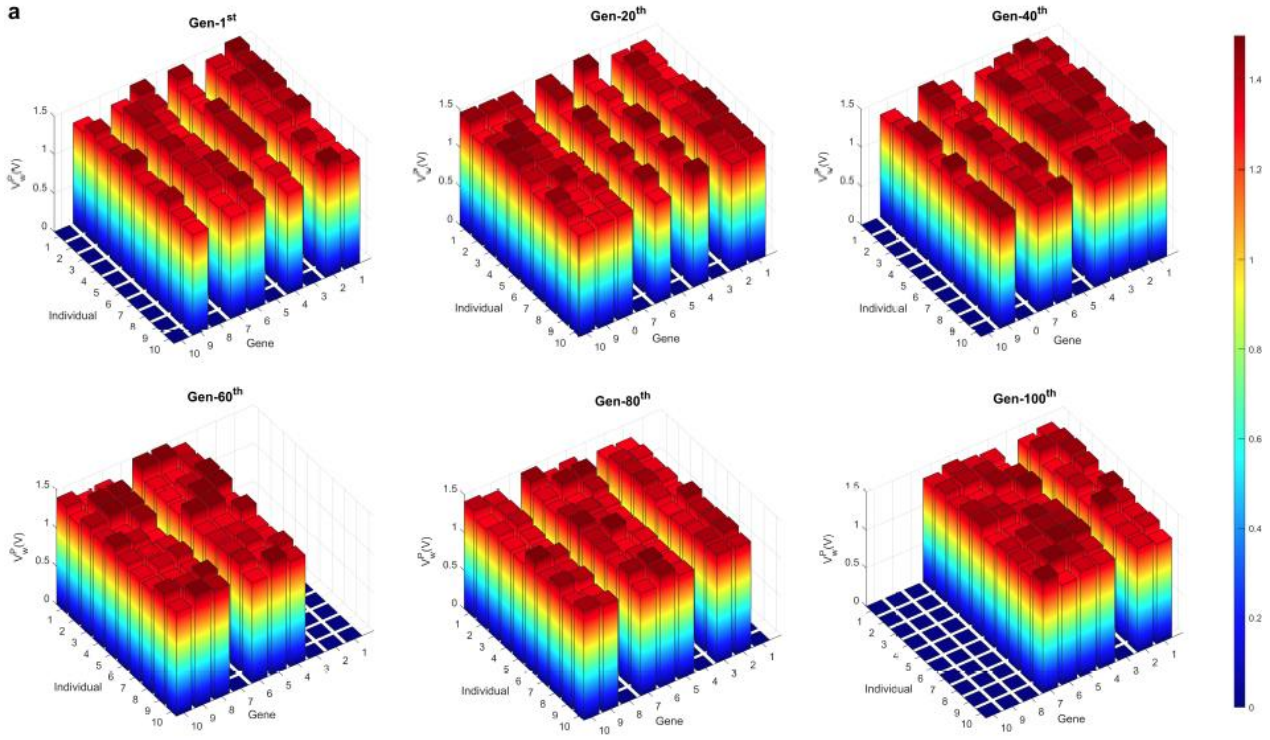
Xinming Shi et. al"Memristor-based Evolvable Brain-inspired Computing System", Under review, *Nature Machine Intelligence*

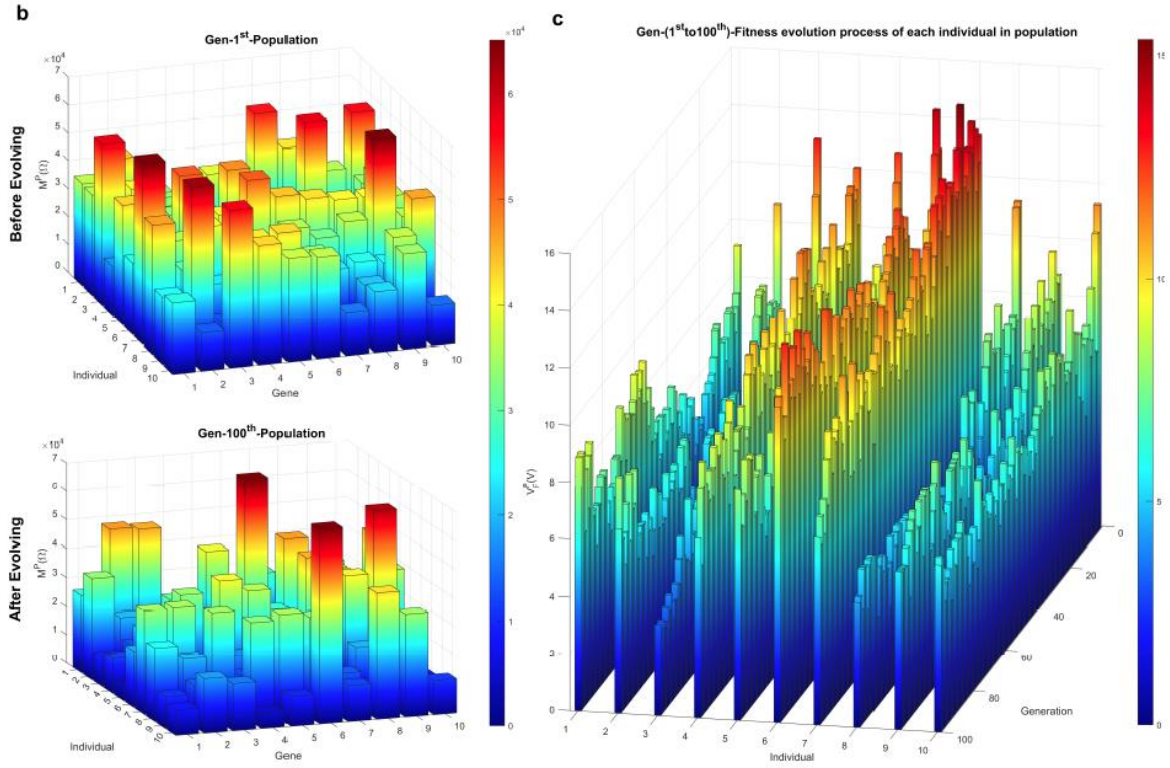
■ Memristive Implementation of Evolution Algorithm

➢ Memristive architecture implementation of evolutionary algorithm

Sparse update: not all genes will be updated

approximate updating of memristive array





■ Memristive Implementation of Evolution Algorithm

➢ Memristive architecture implementation of evolutionary algorithm [3]

Table 6. Execution Time with Different Population Sizes and Different Execution Ways for the Same Benchmark Problem

Population Sizes	*Traditional Software*★	*Memristive Hardware**
5 ×5	0.167s	2e-5s
10 ×10	0.398s	2e-5s
100 ×100	28.599s	2e-5s

★ The working frequency of software-based implementation is 2.5 GHz, and parallel computing is adopted during program execution. * The working frequency of hardware-based implementation is 5 MHz.

Table 7. Hardware Overhead of the Evolvable Brain-Inspired Hardware System Implemented Based on the Proposed Brain-Inspired Hardware Architecture for Evolutionary Algorithms

	Parents&update Module	*Mutation&offspring Module*	*Selection Module*
Memristor	$m \cdot n$	$3 \cdot m \cdot n$	0
Transistor	$8 \cdot m \cdot n$	$69 \cdot m \cdot n$	$33 \cdot m$
Resistor	$8 \cdot m \cdot n$	$54 \cdot m \cdot n$	$8 \cdot m$
Order of Magnitude for Hardware Overhead		$(m \cdot n)\ \mu m^2$	

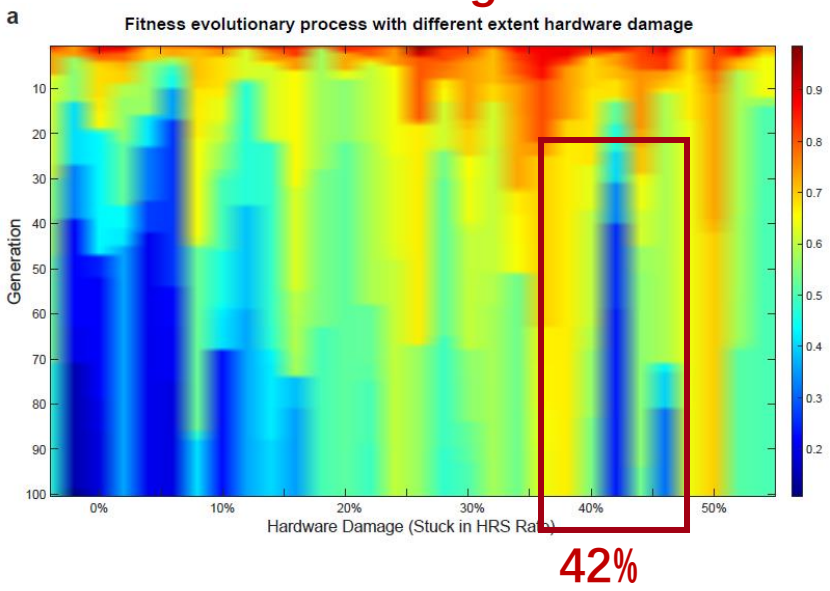
- **Speed**: Hardware-based implementation achieves million-fold acceleration compared to software, with execution time nearly constant regardless of population size.
- **Scalability**: The architecture naturally supports large-scale populations without increasing runtime.
- **Efficiency**: Hardware overhead grows linearly with problem size and remains within a manageable order of magnitude.
- **Resource utilization**: Modular design minimizes redundant hardware usage, especially in the Selection module.
- **Energy and parallelism**: Exploiting memristor's analog computing and parallel update capabilities results in low-power, high-throughput evolution.

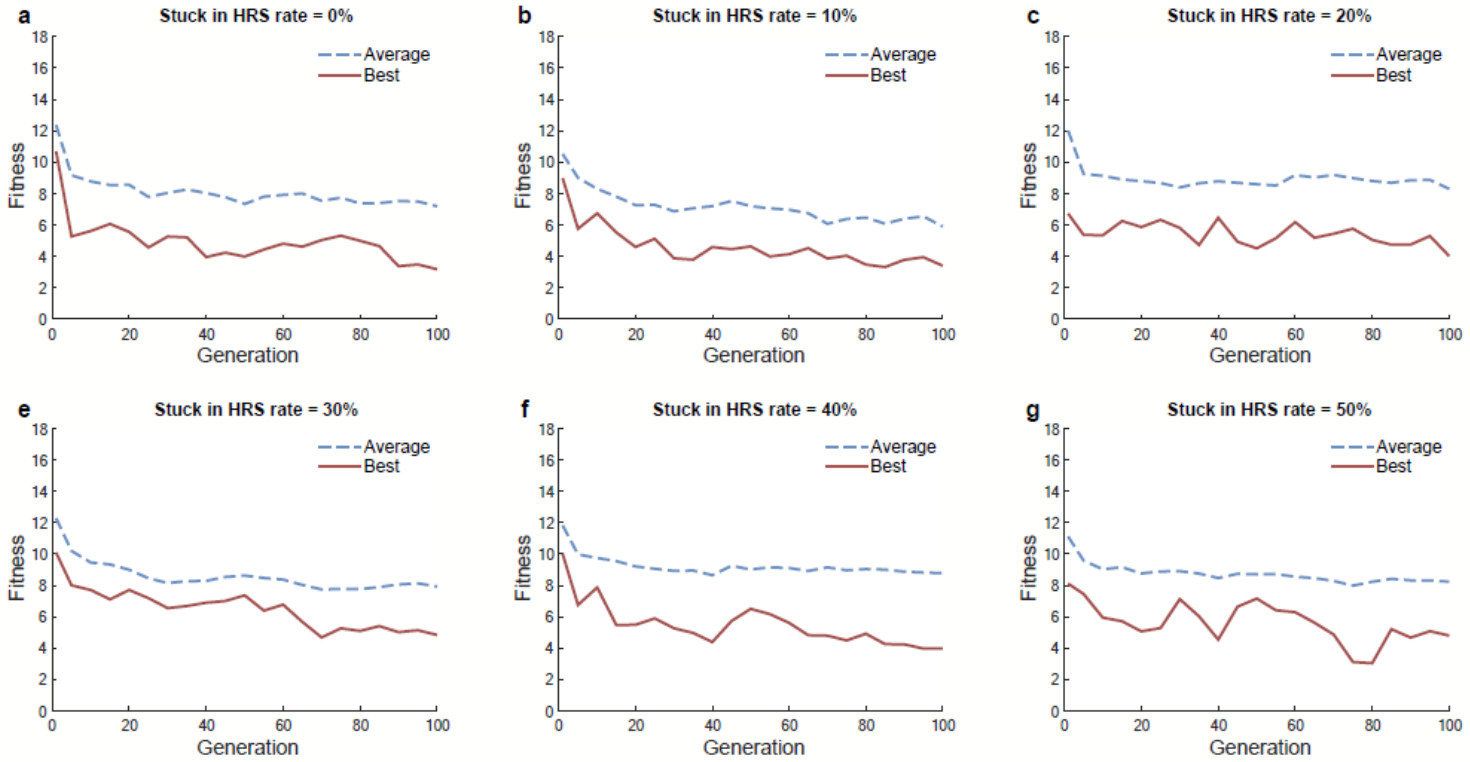
[3] Zilu Wang, **Xingming Shi**, and Xin Yao, "A Brain-Inspired Hardware Architecture for Evolutionary Algorithms based on Memristive Arrays," in *ACM Transactions on Design Automation of Electronic Systems*, doi: 10.1145/3598421.

■ Memristive Implementation of Evolution Algorithm

➢ Memristive architecture implementation of evolutionary algorithm [3]

Hardware damage and fitness





The average and best evolved results when the system is at 0%, 10% to 50% hardware damage, respectively.

- ✓ **We applied sparse, approximate adjustments to memristive arrays, which enables better power consumption and computing speed.**
- ✓ **The proposed brain-inspired computing architecture is fault tolerant.**

[3] Zilu Wang, **Xingming Shi**, and Xin Yao, "A Brain-Inspired Hardware Architecture for Evolutionary Algorithms based on Memristive Arrays," in *ACM Transactions on Design Automation of Electronic Systems*, doi: 10.1145/3598421.

Summary: Evolving Brain-Inspired Memristive Computing Systems

Background

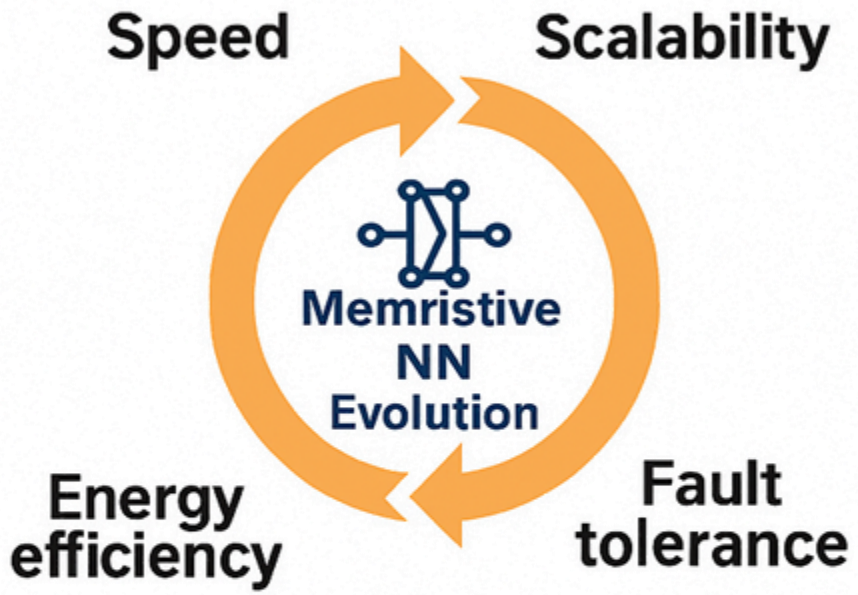
- Neuromorphic computing aims to mimic biological neural systems.
- Memristors provide nanoscale, tunable, non-volatile, and energy-efficient building blocks.

Approach

- Design spiking neuron molels (LIF, Hodgkin-Huxley, Izhikevich, MIF).
- Implement memristive neurons and networks (synapse, neuron, topology).
- Apply evolutionary algorithms to dapt circuit topology & weights.

Advantages

- **Speed:** Million-fold acceleration vs software.
- **Scalability:** Supports large-scale populations.
- **Efficiency:** Compact design with linear hardware overhead.



Results

- Demonstrated memputing paradigm combining brain-inspired models with evolutionary adaptation.
- Paves the way tor practical, adative, and resilient neuromorphic hardware systems.

QUEEN'S UNIVERSITY BELFAST

Thank you!

x.shi@qub.ac.uk