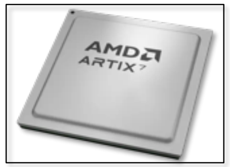
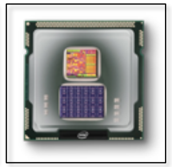
Christof Teuscher

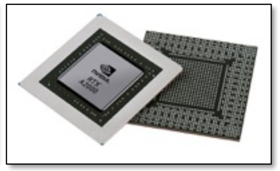
ECE 410/510: Hardware for AI and ML

Neuromorphic chips

Portland State University
Department of Electrical and Computer Engineering (ECE)
www.teuscher-lab.com
teuscher@pdx.edu





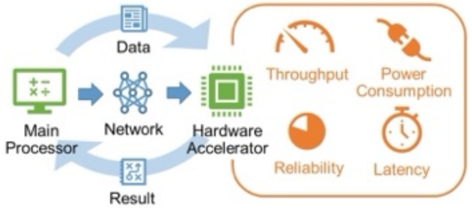


How can we accelerate an algorithm?

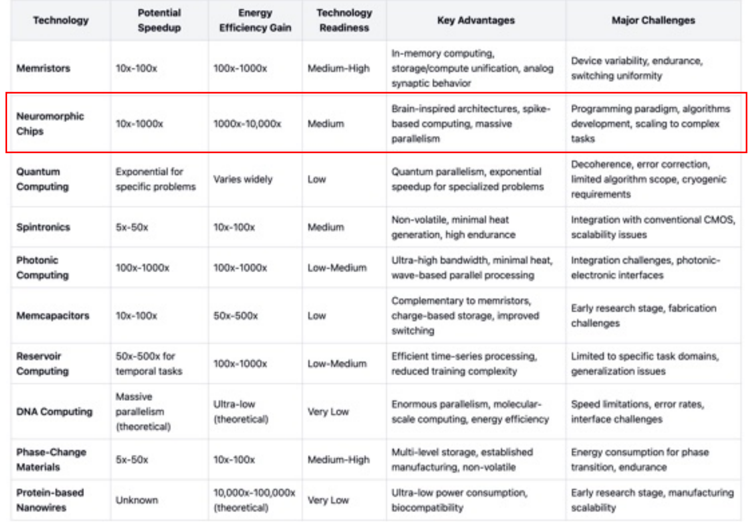
- Technology scaling (but Moore's law is flattening out)
- Cache optimizations
- Code optimization
- Exploiting parallelism
- GPUs
- TPUs
- Fancier CPUs? More pipelining? Faster storage? Better networking?
- HW/SW co-design
- Improve the algorithm (trade time for space, approximations, etc.)
- Emerging technology

How can we accelerate an algorithm?

Or in other words: solve a problem as fast as possible?



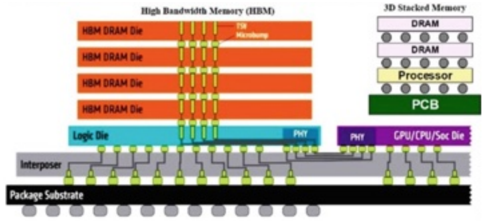
Emerging technology



Technology	Potential Speedup	Energy Efficiency Gain	Technology Readiness	Key Advantages	Major Challenges
Memristors	10x-100x	100x-1000x	Medium-High	In-memory computing, storage/compute unification, analog synaptic behavior	Device variability, endurance, switching uniformity
Neuromorphic Chips	10x-1000x	1000x-10,000x	Medium	Brain-inspired architectures, spike-based computing, massive parallelism	Programming paradigm, algorithms development, scaling to complex tasks
Quantum Computing	Exponential for specific problems	Varies widely	Low	Quantum parallelism, exponential speedup for specialized problems	Decoherence, error correction, limited algorithm scope, cryogenic requirements
Spintronics	5x-50x	10x-100x	Medium	Non-volatile, minimal heat generation, high endurance	Integration with conventional CMOS, scalability issues
Photonic Computing	100x-1000x	100x-1000x	Low-Medium	Ultra-high bandwidth, minimal heat, wave-based parallel processing	Integration challenges, photonic-electronic interfaces
Memcapacitors	10x-100x	50x-500x	Low	Complementary to memristors, charge-based storage, improved switching	Early research stage, fabrication challenges
Reservoir Computing	50x-500x for temporal tasks	100x-1000x	Low-Medium	Efficient time-series processing, reduced training complexity	Limited to specific task domains, generalization issues
DNA Computing	Massive parallelism (theoretical)	Ultra-low (theoretical)	Very Low	Enormous parallelism, molecular-scale computing, energy efficiency	Speed limitations, error rates, interface challenges
Phase-Change Materials	5x-50x	10x-100x	Medium-High	Multi-level storage, established manufacturing, non-volatile	Energy consumption for phase transition, endurance
Protein-based Nanowires	Unknown	10,000x-100,000x (theoretical)	Very Low	Ultra-low power consumption, biocompatibility	Early research stage, manufacturing scalability

Non-von Neuman architectures
In-memory computation (IMC or PiM)

Stacked 3D chips



Mishra et al.

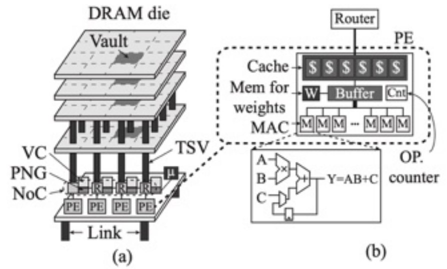
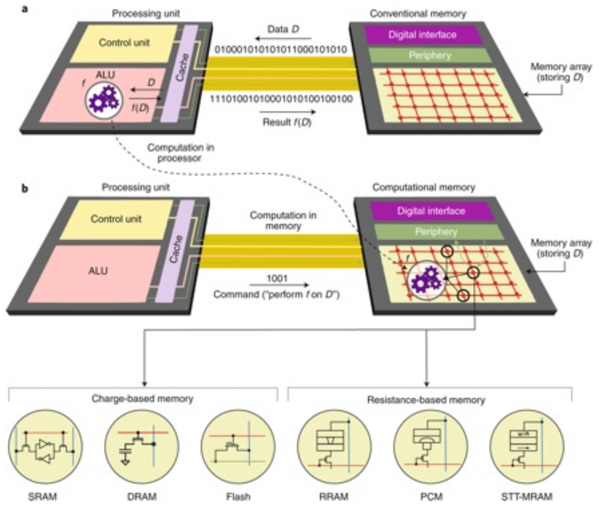


Figure 5. (a) Proposed Neurocube architecture and (b) Organization of the processing elements (PEs).

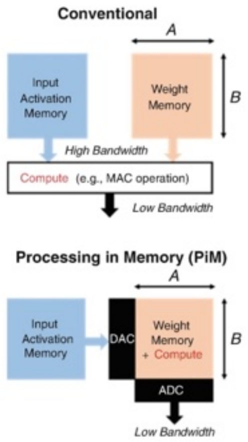
Kim et al., Neurocube

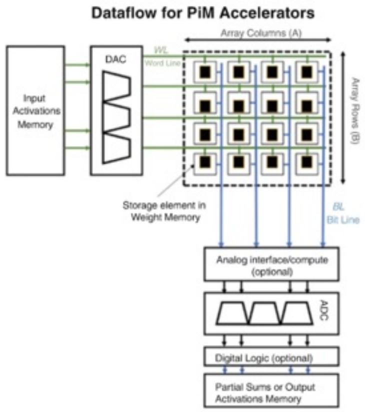
In-memory computing:
the basic idea is simple



Sebastian et al., Nature

Conventional vs Processing-in-Memory (PiM)




Dataflow for PiM Accelerators
Array Columns (A)
Input Activations Memory
DAC
WL
Word Line
Array Rows (B)
Storage element in Weight Memory
BL
Bit Line
Analog interface/compute (optional)
ADC
Digital Logic (optional)
Partial Sums or Output Activations Memory

Mishra et al.

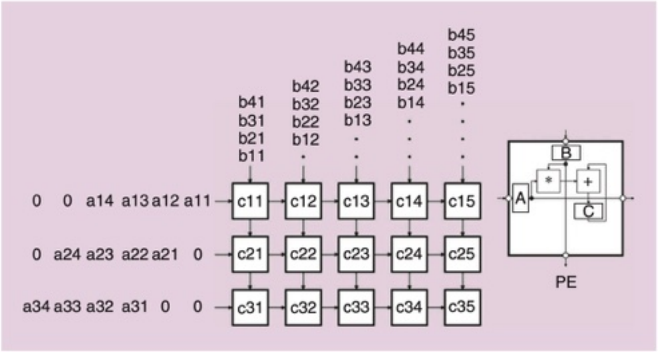


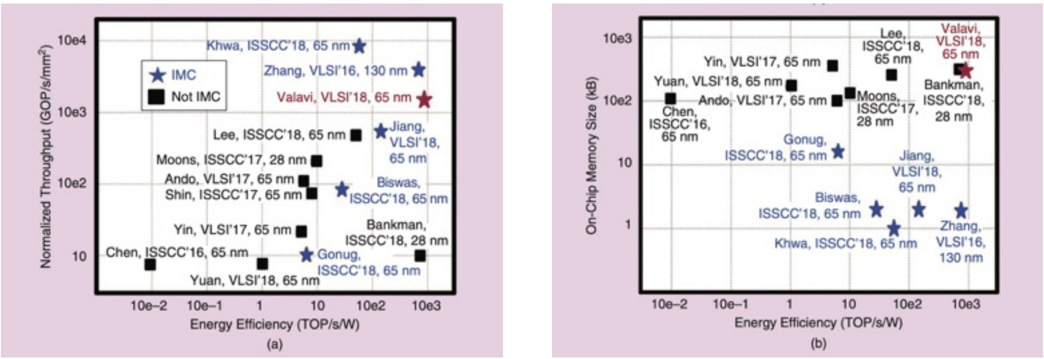
FIGURE 2: The spatial architecture for data-movement/memory-accessing amortization.

Matrix-vector multiplication (MCM)

MVM $\vec{c} = \mathbf{A} \times \vec{b}$

Verma et al., In-Memory Computing, 2019

IMC vs non-IMC comparison



IMC enables ~ 10x gains in each metric.

Verma et al., In-Memory Computing, 2019

MVM (Matrix-Vector Multiplication) can be executed in a crosspoint memory array by universal circuit laws, such as Kirchhoff's current law for summation and Ohm's law for multiplication.

This is schematically shown in Fig. 5(a), where the application of a voltage V_j at the jth column results in a current at the ith row, connected to ground, given by

$$I_i = \sum_j^N G_{i,j} \cdot V_j, \qquad (1)$$

where $G_{i,j}$ is the conductance of the memory element at position i,j and N is the number of rows and columns.[7,307] Equation (1) can be written in the compact matrix form $\mathbf{i} = \mathbf{G}\mathbf{v}$, thus evidencing the multiplication of the conductance matrix $\mathbf{G}$ with the voltage vector $\mathbf{v}$.

crosspoint array or **crossbar**

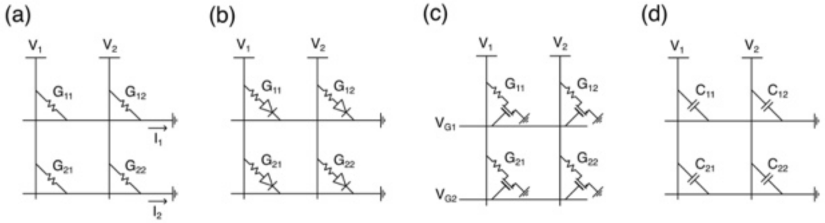
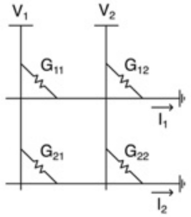


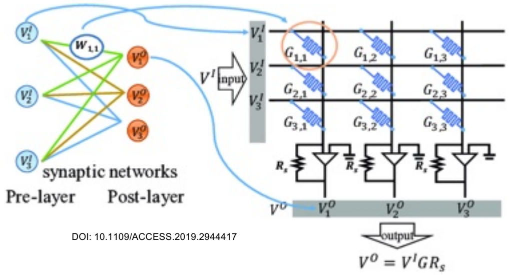
FIG. 5. Various cell structures for crosspoint array circuits. (a) One-resistor (1R) structure where the cell consists of a passive resistive device. (b) One-selector/one-resistor (1S1R) structure where the sneak path problem is circumvented by a non-linear selector device without affecting the integration density. (c) One-transistor/one-resistor (1T1R) structure allows for the selection of individual cells during programming and reading at the cost of a lower integration density. (d) One-capacitor (1C) structure, which prevents static leakage during MVM.

Mannocci et al.

How can we man a NN onto a crossbar?

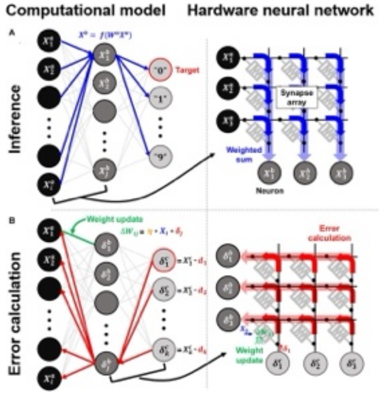


How can we map a NN onto a crossbar?



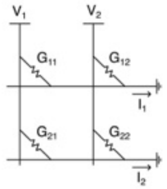
DOI: 10.1109/ACCESS.2019.2944417

Parallel multiplication: When voltage is applied to the rows, the current flowing through each memristive junction is proportional to the product of the input voltage and the junction's conductance. This effectively performs multiplication at each intersection point simultaneously.
Current summation: The currents from all intersections in a column are naturally summed according to Kirchhoff's current law, effectively adding up all the partial products.

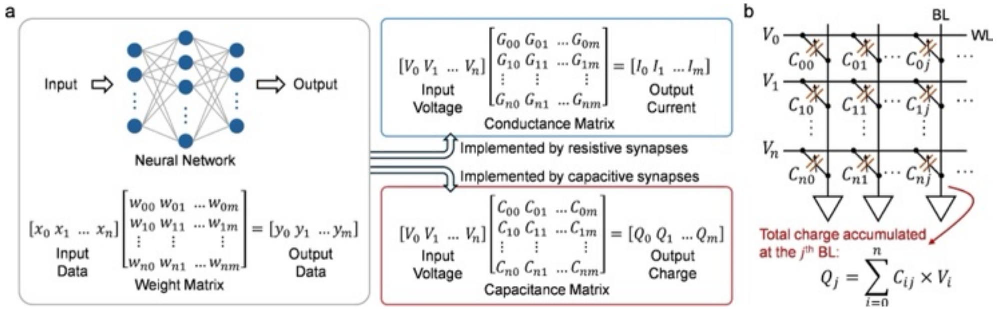


https://doi.org/10.3389/fnins.2021.690418

Sneak paths (1)

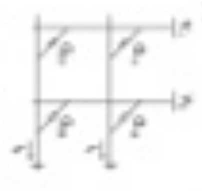


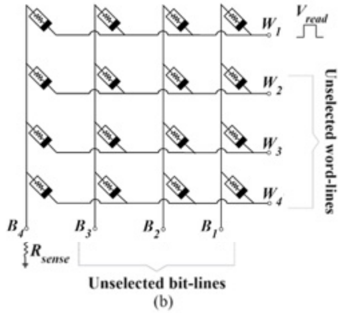
How can we map a NN onto a crossbar?



https://nanoconvergencejournal.springeropen.com/articles/10.1186/s40580-024-00463-0

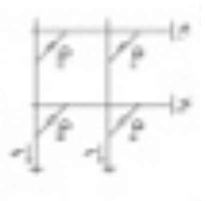
Sneak paths (2)

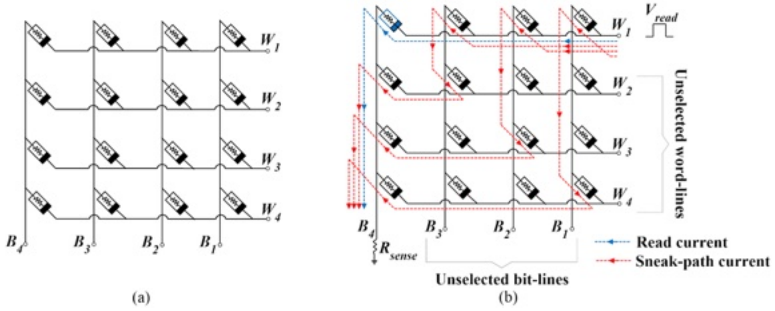




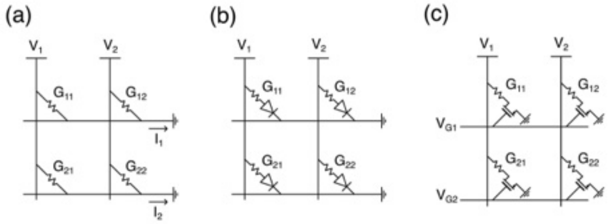
Make a drawing of where the current flows

Sneak paths (3)





Sneak paths (4)



Sneak-paths in crossbars are unintended current pathways
that occur in memory array architectures.

Solutions: **diodes** or **1T1R structures**

Questions

- What is the primary advantage of in-memory computing.
- In a typical crossbar implementation of a neural network, where are the synaptic weights physically represented?
- What are the benefits of spiking neural nets over analog neural nets?

Neuromorphic chips

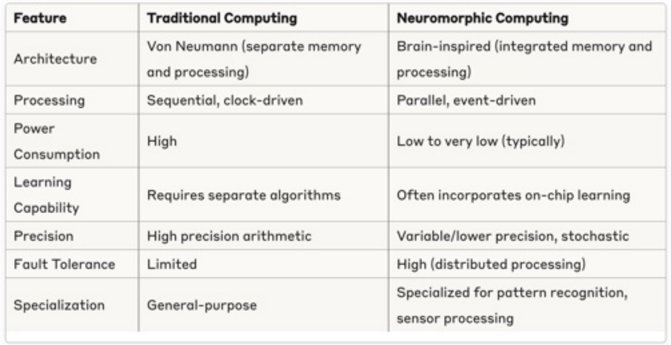
What are neuromorphic chips?

Neuromorphic chips are specialized hardware designed to mimic the structure and function of the human brain.

What neuromorphic HW would you build?
What are characteristics one would like to see in such HW?
Compute? Communication? Memory?
Do we need neuromorphic HW at all? If so, for what?
Why not do everything in software?

Key characteristics of all neuromorphic chips

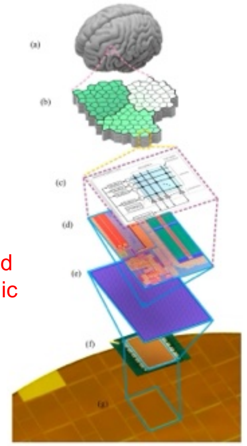
- **Parallel processing** - Unlike traditional Von Neumann computing where processing and memory are separate, neuromorphic chips integrate computation and memory in the same physical structures (similar to how brain cells work).
- **Event-driven computation** - Most neuromorphic chips operate based on events or "spikes" rather than continuous clock cycles, making them more energy-efficient by only consuming power when processing information.
- **Adaptability** - Many neuromorphic designs incorporate on-chip learning mechanisms that allow synaptic connections to strengthen or weaken based on activity patterns.
- **Distributed memory** - Information is stored across the network in the connection strengths between neurons, rather than in a centralized memory unit.



Feature	Traditional Computing	Neuromorphic Computing
Architecture	Von Neumann (separate memory and processing)	Brain-inspired (integrated memory and processing)
Processing	Sequential, clock-driven	Parallel, event-driven
Power Consumption	High	Low to very low (typically)
Learning Capability	Requires separate algorithms	Often incorporates on-chip learning
Precision	High precision arithmetic	Variable/lower precision, stochastic
Fault Tolerance	Limited	High (distributed processing)
Specialization	General-purpose	Specialized for pattern recognition, sensor processing

What are neuromorphic chips?

- Neuromorphic chips are specialized hardware designed to mimic the structure and function of the human brain. Unlike traditional computing architectures, these chips are built to process information in ways similar to biological neural networks.



- Cerebras WSE-3
- 300-millimeter wafer
- 4 trillion transistors
- TSMC 5nm
- 57x larger than an NVIDIA H100
- Accelerate AI workloads, particularly training and inference of large language models.

Not considered a neuromorphic chip

What are the different approaches?

- Implement neurons directly in silicon (Loihi, TrueNorth, etc.)
- Simulate neurons with specialized processors (SpiNNaker).
- Analog
- Digital
- Mixed-signal

- What is a key building block for directly implementing neurons on silicon in many modern neuromorphic chips?
- Can you draw a generic architecture diagram of a neuromorphic chip?

The usual trade-offs…!

Why not do everything in software?

SpiNNaker

Akida
Loihi
Loihi/2
BrainScaleS/2

IBM's TrueNorth
IBM's NorthPole

Flexibility

More digital circuits

Other trade-offs:
- Power
- Reliability
- Programmability
- On-chip learning
- Cost
- Scalability

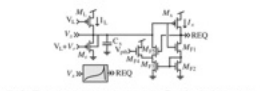


Fig. 2. The neuron circuit comprises a capacitor C_m, driven by three currents: an input (M_s), a leak (M_L), and positive feedback (M_{F1-4}). When the input and positive feedback currents drive V_m low enough, REQ transitions from low to high, signaling a spike.

LIF neuron

DOI: 10.1109/TCSI.2010.2089556

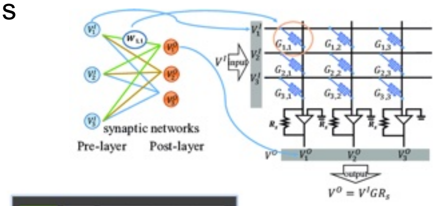
Performance

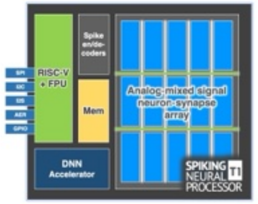
More analog circuits

But:
- Digital backend needed
- DAC/ADC

Key building block: the crossbar

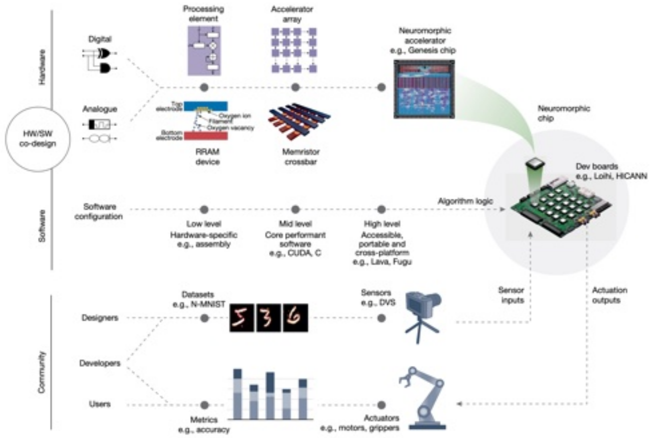
- Crossbar arrays provide the critical architecture for efficiently implementing synaptic connections between neurons.
- Why?
  - Efficient matrix-vector multiplication
  - In-memory computing
  - Implementation of synaptic weights
  - Parallel processing
  - Integration with emerging technologies





Generic diagram?

Neuromorphic computing ecosystem



https://doi.org/10.1038/s41586-024-08253-8

Program ming model

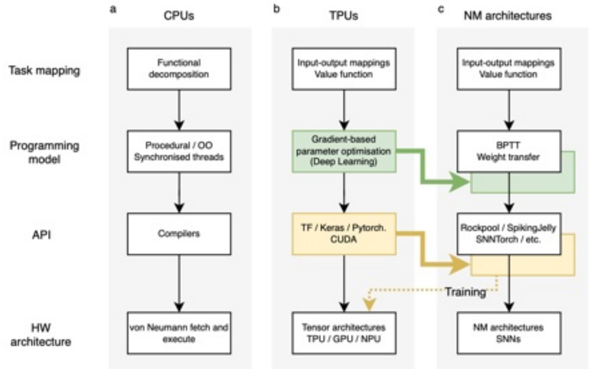
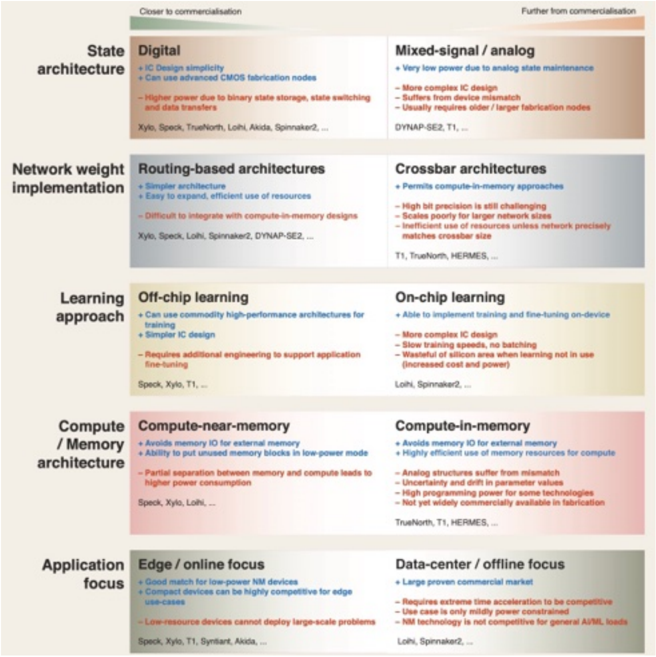

a
CPUs
b
TPUs
c
NM architectures
Task mapping
Programming model
API
HW architecture
Functional decomposition
Procedural / OO Synchronised threads
Compilers
von Neumann fetch and execute
Input-output mappings Value function
Gradient-based parameter optimisation (Deep Learning)
TF / Keras / Pytorch. CUDA
Tensor architectures TPU / GPU / NPU
Input-output mappings Value function
BPTT Weight transfer
Rockpool / SpikingJelly SNNTorch / etc.
Training
NM architectures SNNs

Fig. 1 | Programming models for von Neumann, tensor processor, and Neuromorphic hardware architectures. a von Neumann processors such as CPUs and MCUs use a number of programming models, all of which compile to a predominately serial, fetch-and-execute approach (small-scale multi-processing and optimised pre-fetch notwithstanding). Programming these systems requires decomposing a desired task into a set of explicit procedures for a CPU to execute. **b** Tensor processors such as GPUs, TPUs and NPUs have achieved great success for general-purpose applications, by adopting an example-driven, supervised machine-learning (ML) programming model based on gradient-based parameter optimisation (green box). This is supported by APIs (yellow box) which efficiently implement the ML programming model on tensor processors. **c** Neuromorphic architectures are increasingly adopting a similar programming model, based on methods developed for deep learning (green arrow). At the same time, new software APIs for Neuromorphic application development are making efficient use of the SW tools for deep learning (yellow arrow), which permit rapid parallelised training of NM architectures on commodity tensor processors (dashed arrow).

Backpropagation through time (BPTT)

https://doi.org/10.1038/s41467-025-57352-1



	Closer to commercialisation	Further from commercialisation
State architecture	**Digital** + IC Design simplicity + Can use advanced CMOS fabrication nodes – Higher power due to binary state storage, state switching and data transfers Xylo, Speck, TrueNorth, Loihi, Akida, Spinnaker2, ...	**Mixed-signal / analog** + Very low power due to analog state maintenance – More complex IC design – Suffers from device mismatch – Usually requires older / larger fabrication nodes DYNAP-SE2, T1, ...
Network weight implementation	**Routing-based architectures** + Simpler architecture + Easy to expand, efficient use of resources – Difficult to integrate with compute-in-memory designs Xylo, Speck, Loihi, Spinnaker2, DYNAP-SE2, ...	**Crossbar architectures** + Permits compute-in-memory approaches – High bit precision is still challenging – Scales poorly for larger network sizes – Inefficient use of resources unless network precisely matches crossbar size T1, TrueNorth, HERMES, ...
Learning approach	**Off-chip learning** + Can use commodity high-performance architectures for training + Simpler IC design – Requires additional engineering to support application fine-tuning Speck, Xylo, T1, ...	**On-chip learning** + Able to implement training and fine-tuning on-device – More complex IC design – Slow training speeds, no batching – Wasteful of silicon area when learning not in use (increased cost and power) Loihi, Spinnaker2, ...
Compute / Memory architecture	**Compute-near-memory** + Avoids memory IO for external memory + Ability to put unused memory blocks in low-power mode – Partial separation between memory and compute leads to higher power consumption Speck, Xylo, Loihi, ...	**Compute-in-memory** + Avoids memory IO for external memory + Highly efficient use of memory resources for compute – Analog structures suffer from mismatch – Uncertainty and drift in parameter values – High programming power for some technologies – Not yet widely commercially available in fabrication TrueNorth, T1, HERMES, ...
Application focus	**Edge / online focus** + Good match for low-power NM devices + Compact devices can be highly competitive for edge use-cases – Low-resource devices cannot deploy large-scale problems Speck, Xylo, T1, Syntiant, Akida, ...	**Data-center / offline focus** + Large proven commercial market – Requires extreme time acceleration to be competitive – Use case is only mildly power constrained – NM technology is not competitive for general AI/ML loads Loihi, Spinnaker2, ...

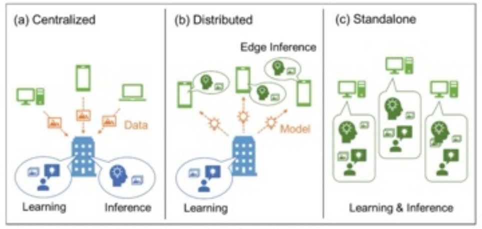
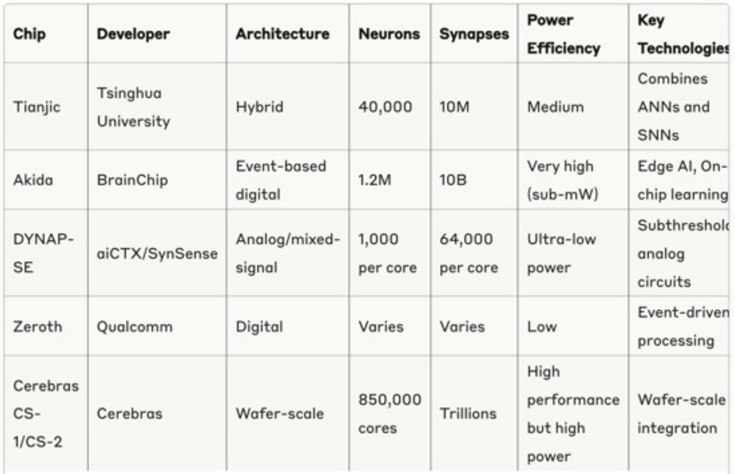


Fig. 1 Overview of learning and inference scheme in (a) centralized, (b) distributed, and (c) standalone computing

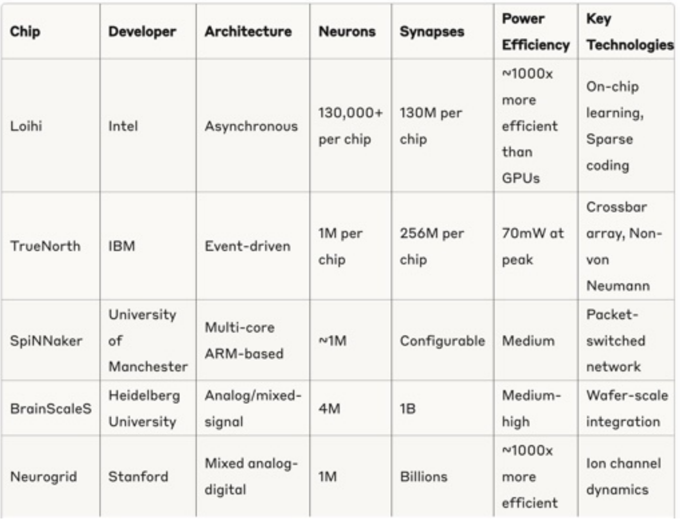
https://doi.org/10.1038/s41467-025-57352-1

Commercial and emerging neuromorphic chips



Chip	Developer	Architecture	Neurons	Synapses	Power Efficiency	Key Technologie
Tianjic	Tsinghua University	Hybrid	40,000	10M	Medium	Combines ANNs and SNNs
Akida	BrainChip	Event-based digital	1.2M	10B	Very high (sub-mW)	Edge AI, On-chip learning
DYNAP-SE	aiCTX/SynSense	Analog/mixed-signal	1,000 per core	64,000 per core	Ultra-low power	Subthreshol analog circuits
Zeroth	Qualcomm	Digital	Varies	Varies	Low	Event-driven processing
Cerebras CS-1/CS-2	Cerebras	Wafer-scale	850,000 cores	Trillions	High performance but high power	Wafer-scale integration

Academic and research chips



Chip	Developer	Architecture	Neurons	Synapses	Power Efficiency	Key Technologies
Loihi	Intel	Asynchronous	130,000+ per chip	130M per chip	~1000x more efficient than GPUs	On-chip learning, Sparse coding
TrueNorth	IBM	Event-driven	1M per chip	256M per chip	70mW at peak	Crossbar array, Non-von Neumann
SpiNNaker	University of Manchester	Multi-core ARM-based	~1M	Configurable	Medium	Packet-switched network
BrainScaleS	Heidelberg University	Analog/mixed-signal	4M	1B	Medium-high	Wafer-scale integration
Neurogrid	Stanford	Mixed analog-digital	1M	Billions	~1000x more efficient	Ion channel dynamics

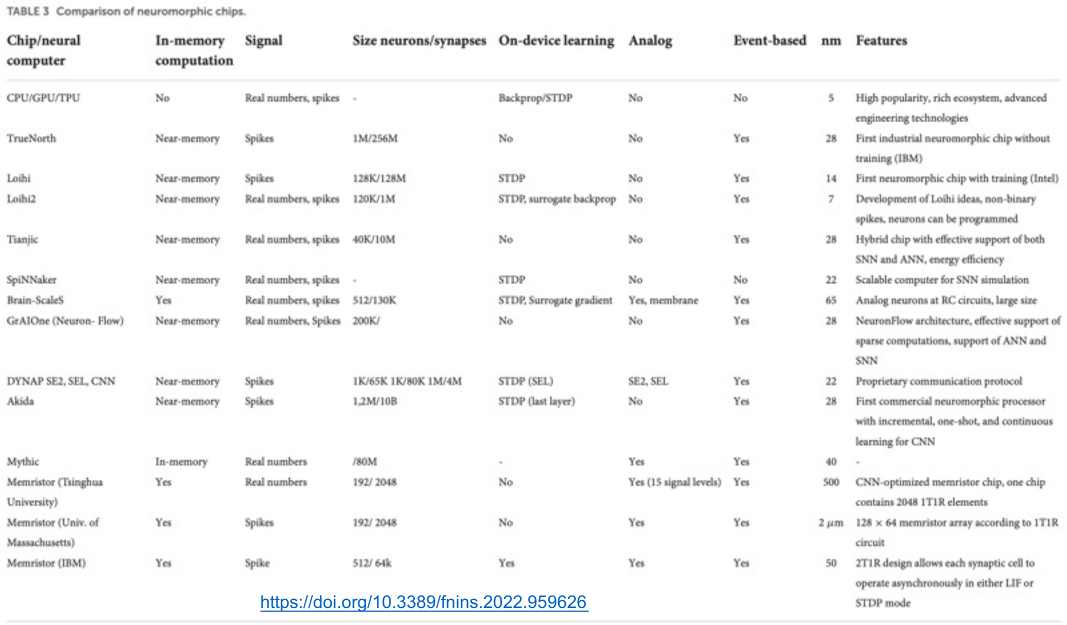


TABLE 3 Comparison of neuromorphic chips.

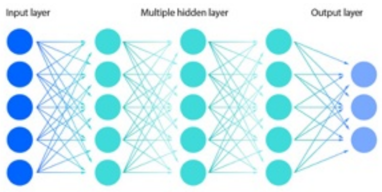
Chip/neural computer	In-memory computation	Signal	Size neurons/synapses	On-device learning	Analog	Event-based	nm	Features
CPU/GPU/TPU	No	Real numbers, spikes	-	Backprop/STDP	No	No	5	High popularity, rich ecosystem, advanced engineering technologies
TrueNorth	Near-memory	Spikes	1M/256M	No	No	Yes	28	First industrial neuromorphic chip without training (IBM)
Loihi	Near-memory	Spikes	128K/128M	STDP	No	Yes	14	First neuromorphic chip with training (Intel)
Loihi2	Near-memory	Real numbers, spikes	120K/1M	STDP, surrogate backprop	No	Yes	7	Development of Loihi ideas, non-binary spikes, neurons can be programmed
Tianjic	Near-memory	Real numbers, spikes	40K/10M	No	No	Yes	28	Hybrid chip with effective support of both SNN and ANN, energy efficiency
SpiNNaker	Near-memory	Real numbers, spikes	-	STDP	No	No	22	Scalable computer for SNN simulation
Brain-ScaleS	Yes	Real numbers, spikes	512/130K	STDP, Surrogate gradient	Yes, membrane	Yes	65	Analog neurons at RC circuits, large size
GrAIOne (Neuron- Flow)	Near-memory	Real numbers, Spikes	200K/	No	No	Yes	28	NeuronFlow architecture, effective support of sparse computations, support of ANN and SNN
DYNAP SE2, SEL, CNN	Near-memory	Spikes	1K/65K 1K/80K 1M/4M	STDP (SEL)	SE2, SEL	Yes	22	Proprietary communication protocol
Akida	Near-memory	Spikes	1,2M/10B	STDP (last layer)	No	Yes	28	First commercial neuromorphic processor with incremental, one-shot, and continuous learning for CNN
Mythic	In-memory	Real numbers	/80M	-	Yes	Yes	40	-
Memristor (Tsinghua University)	Yes	Real numbers	192/ 2048	No	Yes (15 signal levels)	Yes	500	CNN-optimized memristor chip, one chip contains 2048 1T1R elements
Memristor (Univ. of Massachusetts)	Yes	Spikes	192/ 2048	No	Yes	Yes	2 μm	128 × 64 memristor array according to 1T1R circuit
Memristor (IBM)	Yes	Spike	512/ 64k	Yes	Yes	Yes	50	2T1R design allows each synaptic cell to operate asynchronously in either LIF or STDP mode

https://doi.org/10.3389/fnins.2022.959626

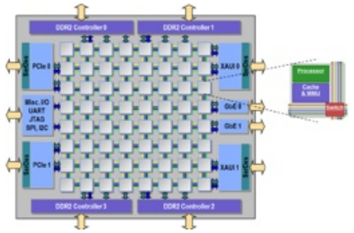
Current limitations and challenges

- **Programming Models**: Lack of standardized programming frameworks for neuromorphic hardware
- **Application Development**: Limited software ecosystem compared to traditional computing
- **Scaling**: Challenges in scaling to human-brain levels while maintaining efficiency
- **Algorithm Mapping**: Difficulty in mapping existing AI algorithms to neuromorphic architectures
- **Adoption**: Industry adoption still in early stages outside of research applications.
- **Cost**: Silicon cost of large-scale applications is prohibitive. Also: most neuromorphic chips rely on SRAM, which is ~2 orders of magnitude more expensive than DRAM.

How to communicate spikes?



Map on grid of processors



Applications

Real-Time Sensing and Processing
- Autonomous Vehicles
- Edge Computing and IoT
- Advanced Robotics

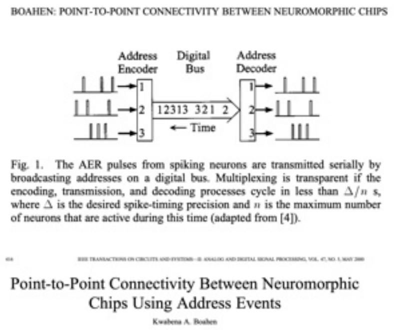
Intelligent Analysis and Security
- Cybersecurity
- Healthcare Monitoring
- Voice and Speech Recognition
- Advanced AI Applications

Advanced AI Applications
- Efficient Machine Learning
- Brain Research
- Aerospace and Defense

Autonomous Vehicles
- **NVIDIA** DRIVE AGX provides "a scalable and energy-efficient AI computing platform designed to process the complex workloads required for autonomous driving."
- Full Self-Driving (FSD) Computer: **Tesla** initially used NVIDIA hardware but has since developed its own custom SoC. According to Elon Musk, designing their own chip allows Tesla "to be able to run the neural network at a fundamental, bare metal level" and optimize specifically for their needs. Tesla's FSD computer delivers 144 TOPS while consuming 72 watts of power.
- **Intel Mobileye EyeQ**: BMW, Volkswagen, and Nissan use Mobileye technology.
- **Qualcomm's Snapdragon Ride**: "designed to power all levels of automated driving" with a scalable architecture that can handle everything from basic driver assistance to fully autonomous driving. Volkswagen, BMW, General Motors, and Toyota have partnered with Qualcomm for their autonomous technology

It's all about AER (Address Event Representation)

- Address Event Representation (AER) is a spike event message passing protocol for Network-on-Chip (NoC).
- Almost all hardware and software simulation environments use a variant of AER
- The simplest and most efficient implementation of AER sends a firing neuron's unique identification number to all of the nodes containing any of that neuron's targets.
- Each spike is typically encoded as a packet containing source/destination information
- Only active neurons that produce spikes generate messages.
- The timing information is implicit in when the message is sent.
- Efficient, event-driven communication system.
- AER allows for asynchronous, sparse communication between neuromorphic cores
- A NoC routes these spike event messages between different neural processing elements.


BOAHEN: POINT-TO-POINT CONNECTIVITY BETWEEN NEUROMORPHIC CHIPS

Address Encoder
Digital Bus
Address Decoder
1
2
3
12313 321 2
← Time
1
2
3

Fig. 1. The AER pulses from spiking neurons are transmitted serially by broadcasting addresses on a digital bus. Multiplexing is transparent if the encoding, transmission, and decoding processes cycle in less than Δ/n s, where Δ is the desired spike-timing precision and n is the maximum number of neurons that are active during this time (adapted from [4]).

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING, VOL. 47, NO. 5, MAY 2000

Point-to-Point Connectivity Between Neuromorphic Chips Using Address Events

Kwabena A. Boahen