Christof Teuscher

ECE 410/510: Hardware for AI and ML

Spiking transformer + In-memory computing

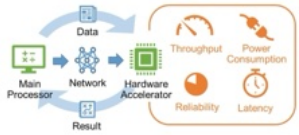
Portland State University
Department of Electrical and Computer Engineering (ECE)
www.teuscher-lab.com
teuscher@pdx.edu

teuscher Lab teuscher-lab.com

Portland State University

How can we accelerate an algorithm?

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



teuscher Lab teuscher-lab.com

Portland State University

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

teuscher Lab teuscher-lab.com

Portland State University

Traditional technology

Optimization Technique	Potential Speedup	Difficulty	Applicability	Notes
Algorithm Selection	10x-1000x+	Medium-High	High	Changing from $O(n^2)$ to $O(n \log n)$ yields massive gains at scale
Data Structure Optimization	2x-100x	Medium	High	E.g., Hash tables vs. arrays for lookups
GPU Acceleration	10x-1000x	High	Medium	For parallelizable computations only
Multithreading/Parallelization	2x-64x	High	Medium	Limited by number of cores and parallelizable portions
Compiler Optimizations	1.2x-4x	Low	High	Simply changing compiler flags
Memory Hierarchy Optimization	2x-10x	Medium	High	Cache-friendly data structures and access patterns
Memoization/Caching	2x-100x	Low-Medium	Medium	For algorithms with repeated calculations
Hardware Upgrade (CPU)	1.5x-4x	Low	High	Generational improvement in processors
SSD vs HDD Storage	2x-100x	Low	Medium	For I/O-bound algorithms
FPGA/ASIC Implementation	10x-1000x	Very High	Low	Custom hardware, high development cost
Code Micro-optimizations	1.1x-2x	Medium	High	Loop unrolling, reducing function calls, etc.
Language Change (Interpreted to Compiled)	3x-50x	High	Medium	E.g., Python to C++

teuscher Lab teuscher-lab.com

Portland State University

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

teuscher Lab teuscher-lab.com

Portland State University

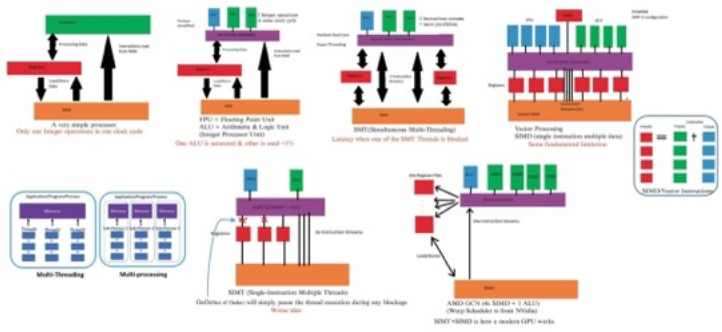


Fig. 14 A few examples of processor designs used for hardware acceleration

Mishra et al.

teuscher Lab teuscher-lab.com

Portland State University

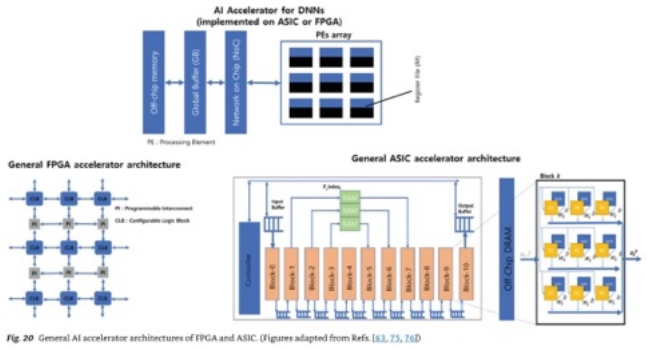
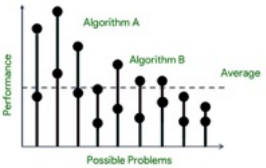


Fig. 20 General AI accelerator architectures of FPGA and ASIC. (Figures adapted from Refs. [63, 75, 76])

Mishra et al.

No Free Lunch theorem (NFL)

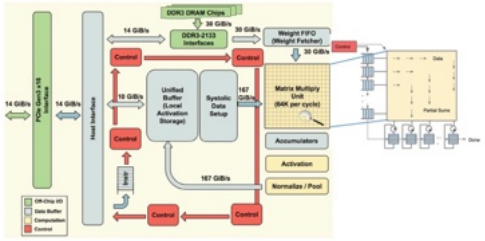
- The "No Free Lunch" (NFL) theorem in optimization and machine learning states that, when averaged across all possible problems, all optimization algorithms perform equally well. I.e., **no algorithm consistently outperforms random search or any other algorithm.**
- The NFL theorem was formalized by David Wolpert and William Macready in 1997.
- **This principle has profound implications for ML and HW optimization:**
  - It explains why specialized algorithms/architectures work well for specific domains but fail in others.
  - It highlights the importance of incorporating domain knowledge when selecting algorithms/architectures.
  - It reminds us that claims about algorithm/architecture superiority must specify the context/problem class.



https://www.geeksforgeeks.org/what-is-no-free-lunch-theorem

https://ieeexplore.ieee.org/document/585893

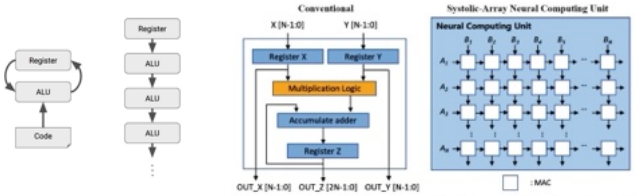
Architecture of a TPU accelerator



- The main component of TPU is the MXU that consists of 256 by 256 (i.e., 65,536) MACs to perform 8-bit mathematical operations.
- The MXU gets its input from the weighted FIFO and unified buffer (UB).

https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

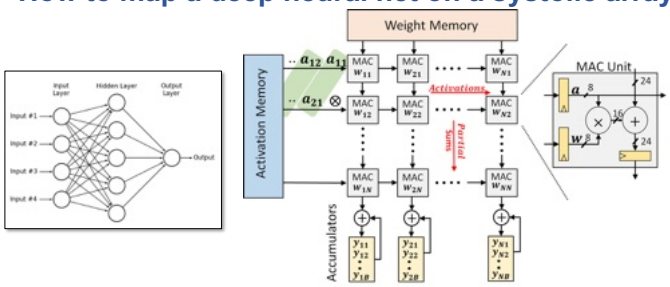
Systolic array



- The name "systolic" comes from an analogy to the human heart, where data pulses through the array of processors in a rhythmic, synchronized fashion.
- CPUs and GPUs often spend energy to access multiple registers per operation. A systolic array chains multiple ALUs together, reusing the result of reading a single register.

Mishra et al., 2023

How to map a deep neural net on a systolic array



https://doi.org/10.1145/3195970.3196129

A Neuromorphic Transformer Architecture Enabling Hardware-Friendly Edge Computing

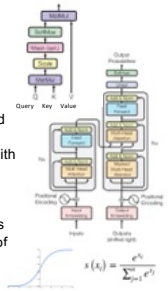
Publisher: IEEE

Abstract:
The transformer model has demonstrated significant capabilities in various intelligent tasks, attracting widespread attention in recent years. However, it involves numerous complex operations, including large-bit-width multiplication, division, matrix transposition, and exponentiation. These require substantial storage and computational resources, making it challenging to deploy on edge devices.

https://ieeexplore.ieee.org/abstract/document/10962199

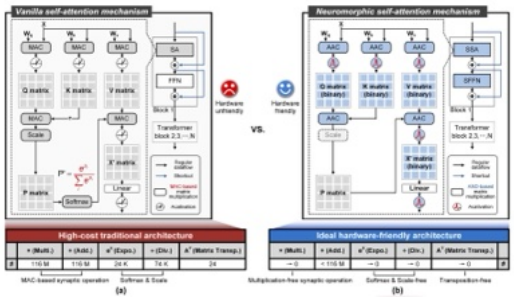
What transformer operations could we improve?

- **Matrix Multiplication** - Used throughout the architecture, particularly in the attention mechanism and feed-forward networks.
- **Attention Mechanism** - The core operation involving:
  - Query, Key, Value transformations (matrix multiplications)
  - Scaled dot-product attention: $\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
  - Multi-head attention which runs multiple attention operations in parallel
- **Layer Normalization** - Normalizes the outputs of sub-layers using mean and variance statistics
- **Residual Connections** (Skip Connections) - Addition operations that help with gradient flow
- **Position-wise Feed-Forward Networks** - Two linear transformations with a ReLU activation in between
- **Softmax** - Used in the attention mechanism to convert scores to probabilities
- **Positional Encoding** - Often implemented using sine and cosine functions of different frequencies
- **Embedding** - Converting tokens to continuous vector representations



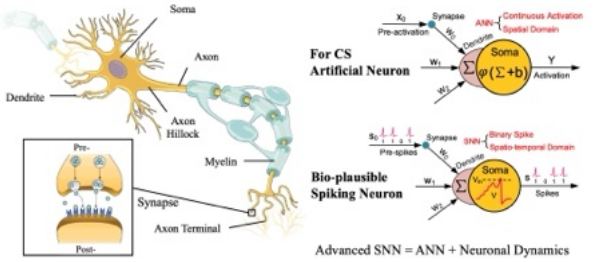
$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1} e^{x_j}}$$

Neuromorphic self-attention mechanism



https://ieeexplore.ieee.org/abstract/document/10962199

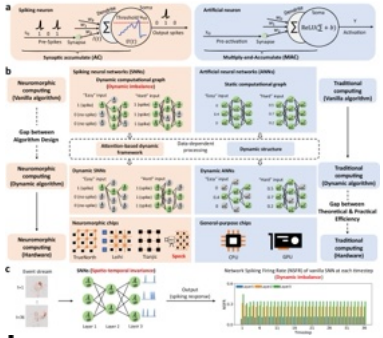
To spike or not to spike (1)



Advanced SNN = ANN + Neuronal Dynamics

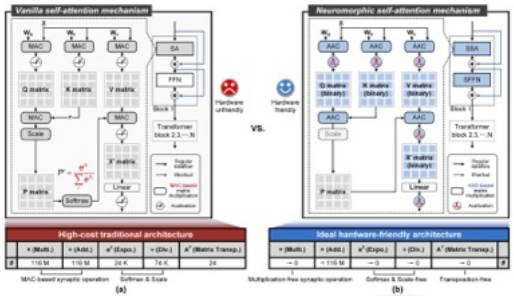
To spike or not to spike (2)

Characteristic	Spiking Neural Networks (SNNs)	Analog Neural Networks (ANNs)
Information Encoding	Discrete spikes/events (binary 0/1 signals)	Continuous values (floating-point)
Biological Plausibility	High - mimics actual neuron behavior	Low - simplified abstraction
Temporal Dynamics	Inherently time-based processing	No native temporal processing
Energy Efficiency	Potentially very high (event-driven)	Moderate to high (depends on implementation)
Training Methods	STDP, converted ANN weights, specialized algorithms	Backpropagation, gradient descent
Accuracy (Current)	Lower but improving	State-of-the-art for most tasks
Hardware Implementation	Specialized neuromorphic chips (Loihi, TrueNorth, SpiNNaker)	GPUs, TPUs, traditional computing hardware
Computational Model	Asynchronous, event-driven	Synchronous, clock-driven
Sparsity	Naturally sparse (neurons only fire when threshold reached)	Dense computations (all neurons compute at each step)
Neuron Models	Leaky integrate-and-fire, Hodgkin-Huxley, Izhikevich	Simple activation functions (ReLU, sigmoid, tanh)
Best Application Domains	Real-time processing, temporal data, power-constrained devices	General ML tasks, image classification, NLP
Maturity	Emerging technology, active research area	Mature technology, widely deployed



https://www.nature.com/articles/s41467-024-47811-6

Neuromorphic self-attention mechanism



AAC = AND accumulate

Why is that equivalent to MAC?

https://ieeexplore.ieee.org/abstract/document/10962199

AND accumulate

Traditional:
`output = (input1 × weight1) + (input2 × weight2) + ... + (inputN × weightN)`

AND accumulate:
`output = (input1 AND weight1) + (input2 AND weight2) + ... + (inputN AND weightN)`

This process essentially counts how many input-weight pairs are both "on" (1) at the same time, which serves as an approximation of multiplication in binary or highly quantized networks.

Binary multiplication = AND operation

Architecture

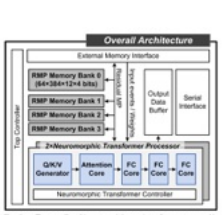


Fig. 6. The overall architecture mainly consists of two neuromorphic transformer processors and peripheral modules.

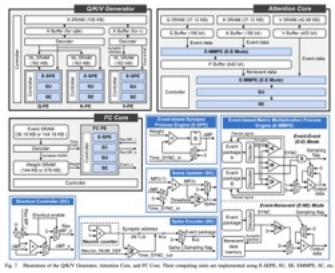
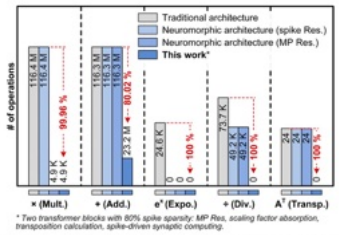


Fig. 7. Illustration of the Q/K/V Generator, Attention Core, and FC Core. Their computing units are implemented using 6 SEPE, 32, 32, 4 MMPE, 32, etc.

 https://ieeexplore.ieee.org/abstract/document/10962199

Resource overhead



* Two transformer blocks with 80% spike sparsity; MP Res, scaling factor absorption, transposition calculation, spike-driven synaptic computing.

Fig. 8. Computational resource overhead of different transformer architectures.

Res. = Residuals
MP = membrane potential

 https://ieeexplore.ieee.org/abstract/document/10962199

Comparison with state-of-the-art

	JSSC22 [18]	JSSC23 [19]	JSSC24 [48]	TBCAS19 [56]	JSSC24 [45]	NSR24 [57]	This work
Architecture	ANN (4×systolic-based attention core)	ANN (1×systolic-based general core)	ANN&SNN (64×FC core)	SNN (4×FC core)	SNN (1024×FC core)	SNN (575×FC core)	SNN (2×transformer block)
Technology (nm)	28	65	28	65	28	22	**28**
Die area (mm²)	6.82	6.40	1.63	3.5	537.98	358.53	**23.28 (all-mem on chip)***
Frequency (MHz)	50-510	600-1100	40-210	210	24-600	300-400	**400**
Power (mW)	12.06-272.7	400-1675	2.91-56.8	19.9	9970	1800	**798.59***
On-chip Memory	336 KB	64 KB	266.5 KB	288 KB	181.865 MB	N. A.	**5.13 MB**
Synaptic precision	INT12	INT8	INT8/10	INT1	INT1/2/4/8	INT1/2/4/8/16	**INT8**
Softmax	Required	Required	-	-	-	-	**Unrequired**
Scale	Required	Required	-	-	-	-	**Unrequired**
Matrix transposition	Required	Required	-	-	-	-	**Unrequired**
Synaptic MAC	Required	Required	Required	Unrequired	Unrequired	Unrequired	**Unrequired**
Accuracy (Cifar-10 dataset)	N. A.	91.5% (CNN)	N. A.	N. A.	93.91% (CNN)	90.17% (CNN)	**94.03 %***
Computing efficiency per core* (GSOP/s)	65.25 (MM) 65.25-157.5 (QK^T) 65.25-508.75 (PV) (@510 MHz)	512	N. A.	0.105 (@210 MHz)	20.25 (@600 MHz)	N. A.	**G^T: 1.49-2.93 A^T: 1.10-1.84 F^T: 1.00-1.98**
Energy efficiency* (pJ/SOP)	1.047 (MM) 0.77-0.32 (QK^T) 0.77-0.14 (PV) (@510 MHz)	3.33-2 (@1 GHz)	4.16 (@210 MHz)	65 (@210 MHz)	2.444 (@120 MHz)	5.4 (@300-400 MHz)	**G^T: 0.64-0.33 A^T: 4.34-2.79 F^T: 5.73-3.07**

SOP = Standard Operations Per Second

GSOP/s = Giga Standard Operations Per Second

Different cores of their design

 https://ieeexplore.ieee.org/abstract/document/10962199

What are the biggest bottlenecks in traditional computer architecture?

Impact of major bottlenecks

Bottleneck	Performance Impact	Trend	Mitigation Strategies	Relative Importance
Memory Wall (Von Neumann)	50–80% performance gap between CPU and DRAM speeds	Growing (CPU performance improves ~40% annually vs. memory at ~10%)	Cache hierarchies, HBM, on-chip memory	★★★★★
Power Consumption	~100x increase in power density since 1990s	Exponential growth halted clock speeds at ~4GHz	Dark silicon, specialized cores, power gating	★★★★★
Sequential Processing	Amdahl's Law: 5x parallel cores yields only ~3x speedup with 10% sequential code	Sequential portion becomes dominant constraint	Speculative execution, out-of-order processing	★★★★
I/O Bottlenecks	Storage access ~10^5-10^6 times slower than memory access	Improving but gap remains significant	NVMe, storage class memory, computational storage	★★★★
Instruction-Level Parallelism	Diminishing returns beyond 4–6 issue width	Limits reached in most architectures	Branch prediction, speculative execution	★★★
Cache Coherence	Up to 40% overhead in heavily shared workloads	Worsens with core count	Directory-based protocols, relaxed consistency	★★★

Innovations to address bottlenecks

Architecture Innovation	Memory Wall	Power	Sequential	I/O	ILP	Cache Coherence
Processing-in-Memory	✓✓✓	✓✓	✓	✓✓	-	✓
Domain-Specific Architectures	✓	✓✓✓	✓✓	-	✓✓	✓
Chiplet Designs	✓✓	✓✓	-	✓	-	-
Quantum Computing	✓	-	✓✓✓	-	✓✓✓	-
Neuromorphic Computing	✓✓	✓✓✓	✓✓	-	✓	✓✓

Checkmarks indicate level of impact in addressing the bottleneck (more checkmarks = greater impact)

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

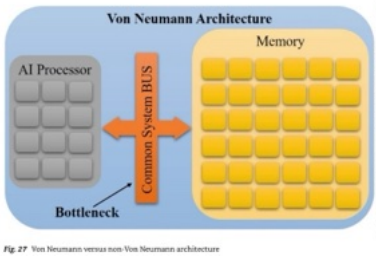


Fig. 27 Von Neumann versus non-Von Neumann architecture

?

Mishra et al.

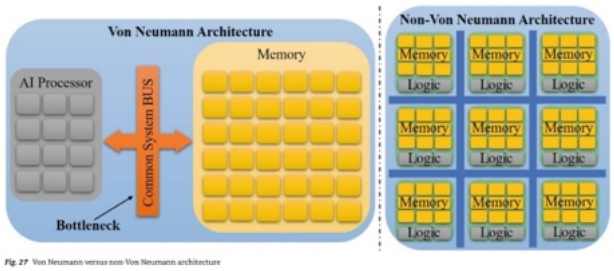
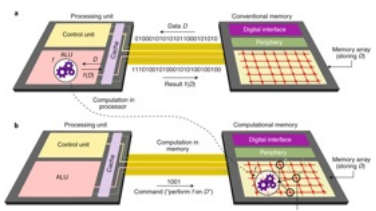


Fig. 27 Von Neumann versus non-Von Neumann architecture

Fundamental problem: separation of memory and computation. Communication cost is fundamental, computer architectures cannot avoid it, but they can attempt to amortize it.

Mishra et al.

IMC: the basic idea is simple



Sebastian et al., Nature

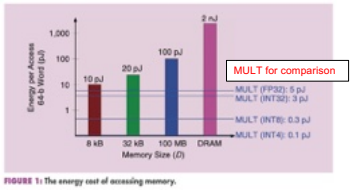


FIGURE 1: The energy cost of accessing memory.

A comparison of the energy required to access one word of data (64 b) from different-sized memories in a 45-nm technology to the energy of multiplication operations (considering the lowered precision levels that are increasingly relevant for deep-learning inference computations.

Verma et al., In-Memory Computing, 2019

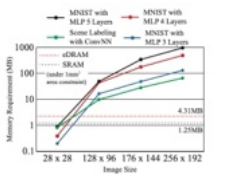
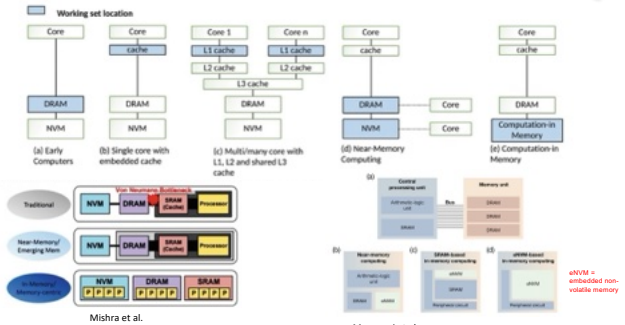


Figure 1. Required memory for scene labeling [9] with different image size using convolutional neural network and for MNIST [10]. Memory capacity of SRAM and eDRAM is normalized by 1mm^2 area constraint [11], [12].

Kim et al., Neurocube



Mishra et al.

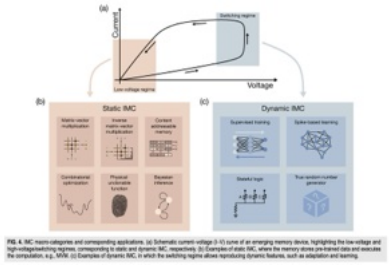
Mannocci et al.

It's all about trade-offs

Mishra et al.

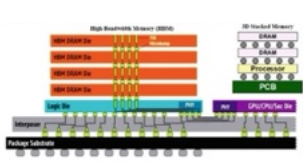
Mannocci et al.

IMC macro-categories



Mannocci et al.

Stacked 3D chips



Mishra et al.

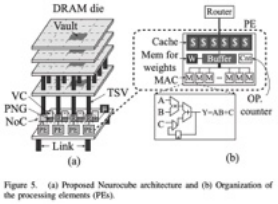
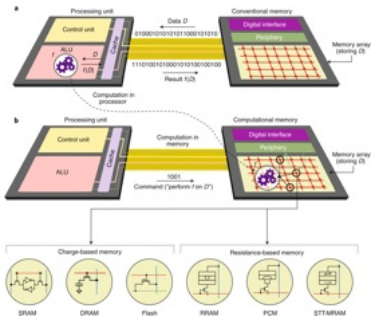


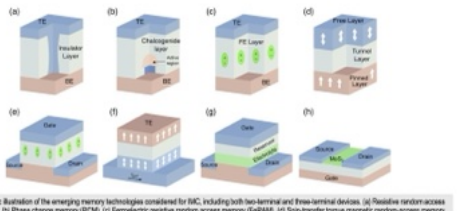
Figure 5. (a) Proposed Neurocube architecture and (b) Organization of the processing elements (PEs).

Kim et al., Neurocube



Sebastian et al., Nature

Emerging memory technologies



Mannocci et al.

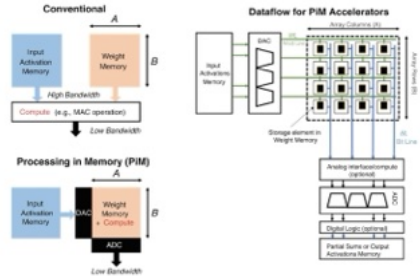
Emerging memory technologies

TABLE I. Comparison of different memory technologies suited for in-memory computing. Reproduced with permission from D. Ielmini and S. Ambrogio, Nanotechnology 31(9), 092001 (2020). Copyright 2020 Author(s), licensed under a Creative Commons Attribution 4.0 License.

Technology	NOR flash	NAND flash	RRAM	PCM	STT-MRAM	FeRAM	FeFET	SOT-MRAM	Li-ion
On/off ratio	10^4	10^4	10–10^2	10^2–10^4	1.5–2	10^2–10^3	5–50	1.5–2	40–10^3
Multilevel operation	2 bit	4 bit	2 bit	2 bit	1 bit	1 bit	5 bit	1 bit	10 bit
Write voltage (V)	<10	10	<3	<3	<1.5	<3	<5	<1.5	<1
Write time	1–10 μs	0.1–1 ms	<10 ns	~50 ns	<10 ns	~30 ns	~10 ns	<10 ns	<10 ns
Read time	~50 ns	~10 μs	<10 ns	<10 ns	<10 ns	<10 ns	~10 ns	<10 ns	<10 ns
Stand-by power	Low	Low	Low	Low	Low	Low	Low	Low	Low
Write energy [J/bit]	~100 pJ	~10 fJ	0.1–1 pJ	10 pJ	~100 fJ	~100 fJ	<1 fJ	<100 fJ	~100 fJ
Linearity	Low	Low	Low	Low	None	None	Low	None	High
Drift	No	No	Weak	Yes	No	No	No	No	No
Integration density	High	Very high	High	High	High	Low	High	High	Low
Retention	Long	Long	Medium	Long	Medium	Long	Long	Medium	···
Endurance	10^5	10^4	10^5–10^8	10^8–10^9	10^{15}	10^{12}	>10^5	>10^{15}	>10^5
Suitability for DNN training	No	No	No	No	No	No	Moderate	No	Yes
Suitability for DNN inference	Yes	Yes	Moderate	Yes	No	No	Yes	No	Yes
Suitability for SNN applications	Yes	No	Yes	Yes	Moderate	Yes	Yes	Moderate	Moderate

Mannocci et al.

Conventional vs Processing-in-Memory (PiM)



Mishra et al.

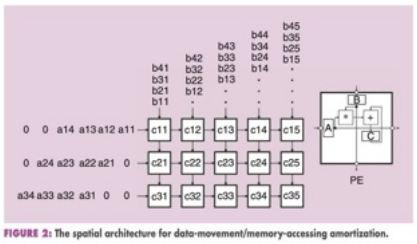


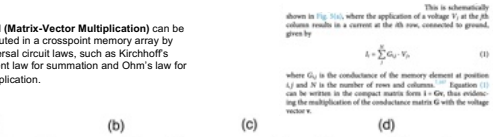
FIGURE 2: The spatial architecture for data-movement/memory-accessing amortization.

Matrix-vector multiplication (MCM)

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

Verma et al., In-Memory Computing, 2019

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

$$I_i = \sum_j G_{ij} \cdot V_j \quad (1)$$

where G_{ij} is the conductance of the memory element at position i,j and N is the number of rows and columns. [37] 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**

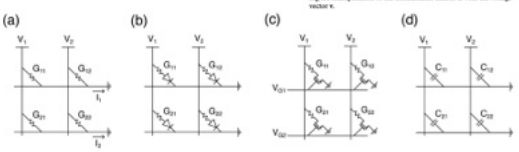


FIG. 5. Various cell structures for crosspoint array circuits. (a) One-resistor (1R) structure where the cell consists of a passive resistive device. (b) One-selector/one-resistor (1S1R) structure where the sneak path problem is circumvented by a non-linear selector device without affecting the integration 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.

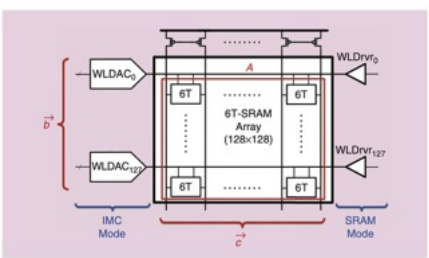
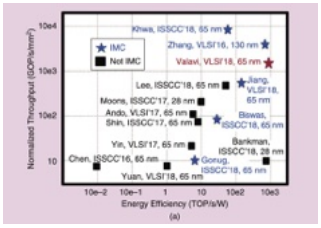


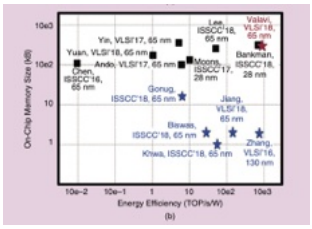
FIGURE 3: The basic approach of IMC illustrated in [7].

Verma et al., In-Memory Computing, 2019

- 6-T SRAM
- 2 modes
- **SRAM mode**:
  - One row accessed at a time via WL
- **IMC mode**:
  - Multiple or all rows accessed
  - Each WL digital-to-analog converter (WLDAC) applies an analog voltage corresponding to an input-vector element.
  - Taking the bitlines (BL/BLb) as a differential signal, the stored data then have the effect of multiplying the input-vector element by +1/–1, and the currents from all bit cells in a column accumulate, generating an analog discharge of BL/BLb. This yields a multiply–accumulate (MAC) operation as required for MVM.

IMC vs non-IMC comparison

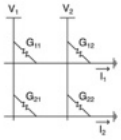




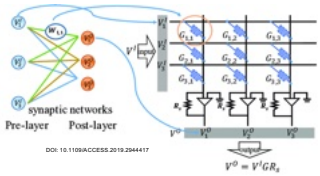
IMC enables ~ 10x gains in each metric.

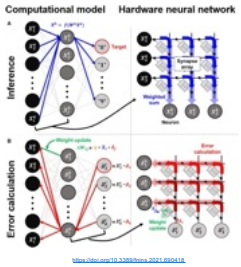
Verma et al., In-Memory Computing, 2019

How can we man a NN onto a crossbar?



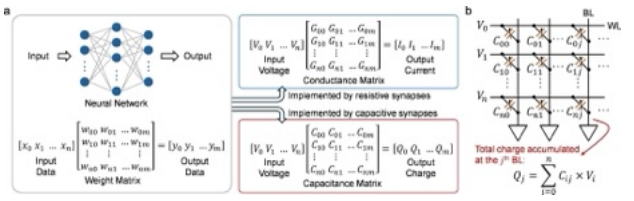
How can we man a NN onto a crossbar?





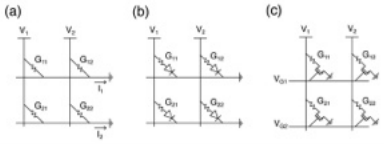
Parallel multiplication: When voltage is applied to the rows, the current flowing through each memristive junction is proportional to the product of the input voltage and the junction's conductance. This effectively performs multiplication at each intersection point simultaneously.
Current summation: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

How can we man a NN onto a crossbar?



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

Sneak paths



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?