

REVIEW ARTICLE

A device–circuit–algorithm review of physics-driven platforms for next-generation artificial intelligence

Neha Singh^{1*}  and Sonal Shreya^{2*} ¹Department of Electronics and Communication Engineering, Indian Institute of Information Technology, Bhopal, Madhya Pradesh, India²Department of Electrical and Computer Engineering, Biomedical Engineering Section, Aarhus University, Aarhus, Central Denmark Region, Denmark

Abstract

Recent advancements in artificial intelligence (AI) have significantly enhanced the perception and cognition of language and the decision-making process. However, the increase in computational power and energy efficiency has also led to the emergence of computational and energy needs. Traditional Von Neumann architectures, with the separation of memory and processing, are costly due to numerous data transfers between hierarchical memory systems. Therefore, in large-scale AI workloads, energy consumption and data transfer latency are major challenges to scalability and efficiency. Using physical phenomena to compute and combining devices, circuits, and algorithms by co-design, these issues are resolved by physics-driven computing. In-memory operations, nonlinearity, and parallelism are also enabled by resistive switching, spin dynamics, phase transitions, and optical interference at the device level, thereby significantly reducing data transmission and energy consumption. Unless the entire hardware stack, including devices, circuits, architectures, and learning algorithms, is coordinated, one achieves limited system-level benefits. This review presents a unified physics-based AI hardware design by jointly optimizing device, circuit, and algorithm. It also discusses new device platforms, including memristive, spintronic, phase-change, photonic, and neuromorphic complementary metal-oxide-semiconductor (CMOS), and how the physics underlying these principles can be used to implement computational primitives. It is discussed further at the circuit level, e.g., in-memory computing arrays, event cameras, neuromorphic processors, and algorithms such as hardware-conscious training, spike-based neural computation, phase-based training, and stochastic inference are also discussed in these physical resources. Together, these innovations enable the development of scalable, adaptable, and energy-efficient intelligent hardware through integrated co-design of materials, electronics, and machine learning.

Keywords: Physics-driven computing; Neuromorphic circuits; Device–circuit–algorithm co-design; Memristors; Spintronics; Phase change devices; Photonic artificial intelligence

***Corresponding authors:**

Neha Singh
(neha.singh@iiitbhopal.ac.in)
Sonal Shreya
(sshreya@ece.au.dk)

Citation: Singh N, Shreya S. A device–circuit–algorithm review of physics-driven platforms for next-generation artificial intelligence. *Int J AI Mater Design*. 2026;3(2):026160009. doi: 10.36922/IJAMD026160009

Received: April 15, 2026

Revised: June 11, 2026

Accepted: June 17, 2026

Published online: June 30, 2026

Copyright: © 2026 Author(s). This is an Open-Access article distributed under the terms of the Creative Commons Attribution License, permitting distribution, and reproduction in any medium, provided the original work is properly cited.

Publisher's Note: AccScience Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

The advancement of deep neural networks and data-driven models with large amounts of information has fueled tremendous growth in the field of artificial intelligence (AI) over the past decade. Thanks to digital complementary metal-oxide-semiconductor (CMOS) technology and massively parallel technologies such as graphical processing units (GPUs) and tensor processing units (TPUs), they can perform at the human level in some tasks, including image recognition, natural language processing, healthcare diagnostics, robotics, and strategic decision-making.¹⁻⁴ There's been a massive uptick in computing power; however, modern neural networks require lots of computational resources for training and deployment, and are stretching current hardware infrastructure as the number of parameters increases to billions or even trillions.

The main problem behind this is that the Von Neumann architecture of computation consists of separate memory and computing units. This paradigm of building has been followed for decades in digital computers, and in this paradigm, data must be reloaded from memory and moved to the processing units during the computation. With the increasing workload of a neural network, data movement cost at the system level in terms of energy consumption and system latency becomes increasingly important.^{5,6} In current machine learning systems, transfer can often be more expensive (in terms of energy) than arithmetic and is extremely expensive for large-scale AI systems. As a result, the Von Neumann bottleneck has become one of the key limitations in terms of scaling and energy efficiency of today's AI hardware.

While cloud-based training infrastructure can be power-hungry, new AI applications need real-time inference and adaptive learning on edge devices. There are many applications out there, such as wearable healthcare devices, autonomous vehicles, drones, industrial sensors, and space systems, that demand high power, high thermal efficiency, low latency, and energy-efficient computation. Moreover, the explosive success of on-chip learning, where systems grow, change, and adapt to new data, further calls into question traditional architectures originally developed to enable deterministic digital logic but not data-rich, iterative learning.

Physics-driven computing, also known as physical computing, is an exciting alternative paradigm in this context. It is not based on ignoring nonidealities and analog behavior of devices, but on natural physical behavior, such as nonlinearity, memory, stochastic switching, phase synchronization, and collective dynamics, for native computing in hardware. It allows computation to occur physically in these processes, resulting in reduced

data transfers, enhanced energy efficiency, and novel computational constructs more suited to intensive AI applications. This is a topic that has been gaining more interest in the past few years, with research in computer architecture based on device physics and dynamical systems for processing information.⁷⁻¹⁰

This has stimulated the exploration of other ways of computing that have the potential to overcome the limitations of traditional digital ways of computing. One approach is to use neuromorphic and physics-based computing, where electronic devices have computing structures analogous to biological neural structures. Motivated by the distributed and power-efficient operation of the brain, the main idea of neuromorphic architectures is to integrate memory and computation on a single physical platform and enable the flow of information and the processing of events in a highly parallel fashion.^{8,9} Neuromorphic computing began as a concept to design electronic systems that mimic the architecture and function of biological neural circuits in the late 1980s. Over the past few years, several large-scale neuromorphic processors have been developed, such as the SpiNNaker system, the TrueNorth system developed by IBM, and Loihi developed by Intel, suggesting that it is now feasible to run large-scale spiking neural networks (SNNs) that are much more efficient in terms of energy usage than their current digital implementation.¹⁰⁻¹²

Research on physics-based AI hardware has been widely conducted for various emerging devices, such as memristors, spintronic devices, ferroelectric memories, phase change materials, and photonic computing platforms, as substrates for physics-based AI hardware. Memristive devices and resistive-switching devices (RSDs) can be utilized for in-memory vector-matrix multiplication (VMM), enabling neural computing with in-memory arrays.^{11,12} Indeed, the ability to exploit electron spin dynamics has been used to develop spintronic devices for the realization of non-volatile memory, stochastic neurons, and oscillatory networks, which are naturally coupled to phase and frequency.^{7,13} Taking advantage of materials that can change their phase allows for a multilevel analog state for synaptic weighting and in-memory learning. To boost linear algebraic operations, which constitute the majority of workloads in deep learning applications, high-bandwidth signal propagation and optical interference properties are exploited in the photonic circuits.^{7,14} All these platforms hint toward the embedding of the computation in the intrinsic material properties and dynamics of physical devices.

Despite rapid change in the field, it is relatively fragmented, which is helping drive the gap between advancements in

hardware and software. The demonstrations are mostly at the device level and are based on changes in physics or material properties, not on changes at the circuit or system level. Models of device behavior in circuits are often highly ideal; however, real-world devices are variable, unreliable, have low yield, limited precision, limited endurance, and stochastic behavior. At the algorithmic level, several machine learning models exist that are designed without accounting for device limitations, such as noise, analog computation errors, and nonlinearity.^{7,15,16} Efficient AI acceleration, however, is based on the close association between the physics of the device, circuit architecture, and the learning algorithms. Moreover, quantization-aware techniques for training memristive devices and for designing peripheral circuits to implement in-memory computing architectures are also important for the efficiency of memristive-based computing architectures. Likewise, a high level of co-optimization among the device coupling strength, learn-based synchronization circuits, and a phase-based learning algorithm must be achieved in a spintronic oscillatory network. The benefit in terms of power and processing, however, cannot be scaled up to the

system level without co-design across several levels of the computing stack.

From this recognition, the device–circuit–algorithm (D–C–A) co-design paradigm has emerged, in which devices, circuits, architectures, and learning algorithms are co-designed in an optimized fashion.^{17,18} This is not done by considering each layer separately, but by using the laws of physics of the device to determine what kind of computational primitive can be implemented at the circuit level and then the circuit-level architecture to inform the algorithm-level adaptations that take into account aspects of the device like stochasticity, limited precision of numbers, or analog computation.

Figure 1 illustrates the aim and contribution of the D–C–A co-design approach described in this paper, and how it relates to the Von Neumann bottleneck through hardware co-design solutions inspired by physics.

There have been several reviews on each technology (including memristive in-memory computing⁷, neuromorphic spintronics¹⁹, phase-change accelerators¹⁴, and photonic neural networks), but a synthesis of various

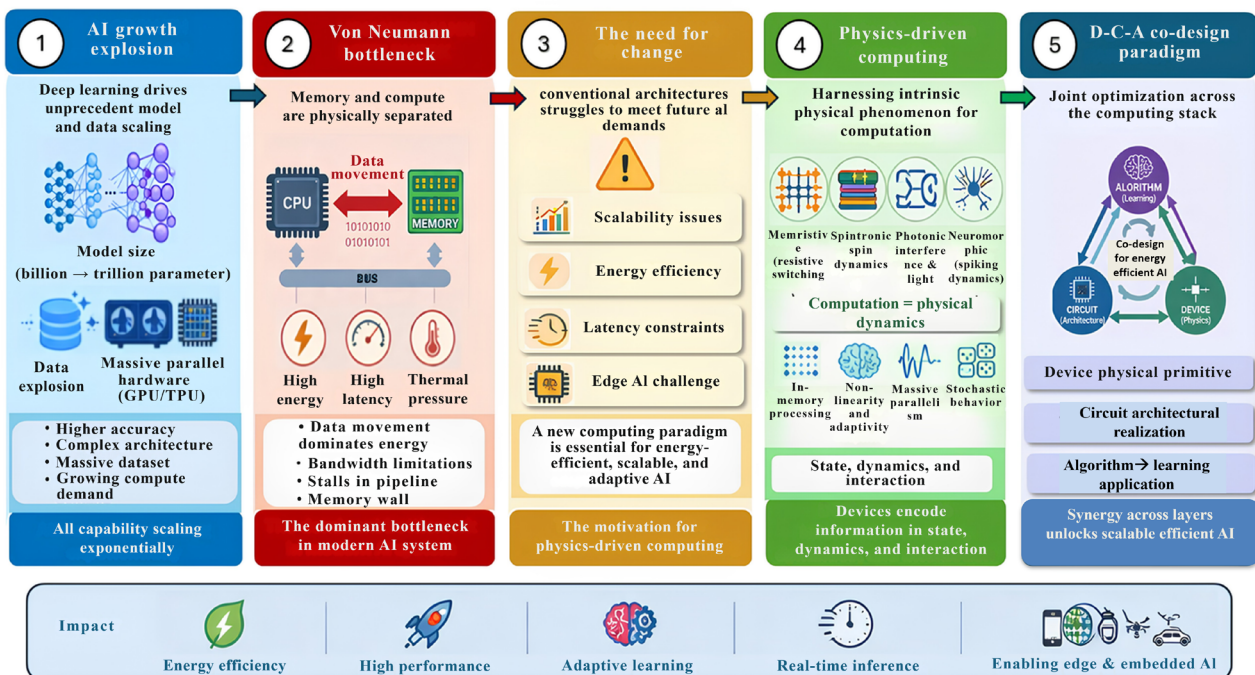


Figure 1. Conceptual overview and motivation of a physics-based device–circuit–algorithm (D–C–A) co-design paradigm of energy-efficient, scalable AI hardware

Abbreviations: AI: Artificial intelligence; GPU: Graphics processing unit; TPU: Tensor processing unit.

device types in a D-C-A framework has not been provided to date. In addition, the new developments in hardware-aware training, in-memory computing, stochastic computing, SNNs, oscillatory neural networks, and hybrid mixed-signal architectures necessitate a new perspective that encompasses them. The new hardware platforms, the booming requirement for intelligence at the system level, and the rapid evolution of AI algorithms make it imperative to review the material physics to system-level intelligence.

This review aims to provide researchers with a coherent path toward the next generation of physical-computation-based AI concepts and approaches by combining insights from devices, circuits, and algorithms. In this way, AI learning becomes a fundamental opportunity to shift from hardware merely implementing an AI algorithm to hardware that is intrinsically an AI learning platform. The rest of this paper is organized as follows: Section 2 outlines the evolution of computing paradigms from Von Neumann architectures toward physics-driven platforms. In Section 3, the device-level basis for physics-driven AI accelerators is discussed, ranging from memristive, phase-change, spintronic, photonic, and neuromorphic CMOS devices. In Section 4, circuit-level implementations, such as in-memory computing arrays, neuromorphic processors, oscillatory circuits, and photonic systems, are discussed. In the algorithms and hardware co-design section (Section 5), we explore specific algorithm-hardware co-design approaches, such as hardware-aware training, SNNs, oscillatory algorithms, and probabilistic inference. Section 6 describes illustrative use cases for system-level applications, including edge AI, autonomous systems, healthcare monitoring, robotics, and space electronics. A brief overview of the key challenges, open problems, and directions for future research is given in Section 7. The conclusions are given in Section 8.

2. Evolution of computing paradigms

For decades, modern computing systems have been based on the Von Neumann architecture, a design paradigm that physically separates the processing and memory units, with a shared data bus for communication. In this architecture, instructions and data are stored in memory and moved to and from the processor for execution. This model worked very well for general-purpose computing, but is not suitable for contemporary AI applications, which require very high data throughput and massive parallelism.

The memory wall is one of the most well-known problems in the von Neumann architecture, characterized by an increasing mismatch between processor performance and memory bandwidth. With the advent of ever-faster processors, memory systems were unable to keep pace,

and today, latency and bandwidth in accessing memory are key factors in system performance. This was the first formal statement of the problem by Wulf and McKee²⁰, who noted that as processor speeds increased, memory access limitations would become a greater constraint in computing systems. This is particularly an issue for modern AI workloads, which involve iterated operations on vast numbers of model parameters. In deep or transformer networks, billions of parameters are repeatedly accessed during processing, which requires high performance and high energy costs to access on-chip caches, high-bandwidth memory, and off-chip dynamic random-access memory (RAM). However, as pointed out by Sze *et al.*²¹, Moore's law and Dennard scaling are slowing, and so is the ability to make processors more energy-efficient, as the need for increased processor performance continues to rise.

Energy consumed per data transfer is one of the important aspects of the memory wall. Memory accesses are known to often consume much more energy than arithmetic operations, such as a few pJ for a floating-point multiplication, compared to retrieving operands from off-chip memory, which can require tens to hundreds of pJ depending on the level in the memory hierarchy.^{21,22} In fact, industry studies have shown that memory access is the largest contributor of energy cost in today's AI accelerators²³ and that data movement is not necessarily the biggest contributor to system efficiency. Sequential instruction execution is supported in the traditional architecture. Despite the fact that modern processors are based on multi-core parallelism, out-of-order processing, and vector processing extensions, they are mainly used to mask memory latency rather than to remove the data transfer bottleneck. With the increasing complexity of AI workloads, these incremental improvements aren't enough to obtain the necessary throughput and efficiency.

Such constraints have led to a series of architectural innovations that represent successive evolutionary steps in AI computing. This evolution from Von Neumann processors to physics-driven platforms with increasing computation localization within or near the physical substrate to reduce data movement costs is shown in [Figure 2](#). To motivate the paradigm change outlined in the subsections below, spanning parallel accelerators to in-memory, neuromorphic, photonic, and other emerging physics approaches, it is important to understand these fundamental limitations.

2.1. Parallel and domain-specific AI accelerators (GPU/TPU era)

Massively parallel computing was the first major architectural solution to the Von Neumann bottleneck.

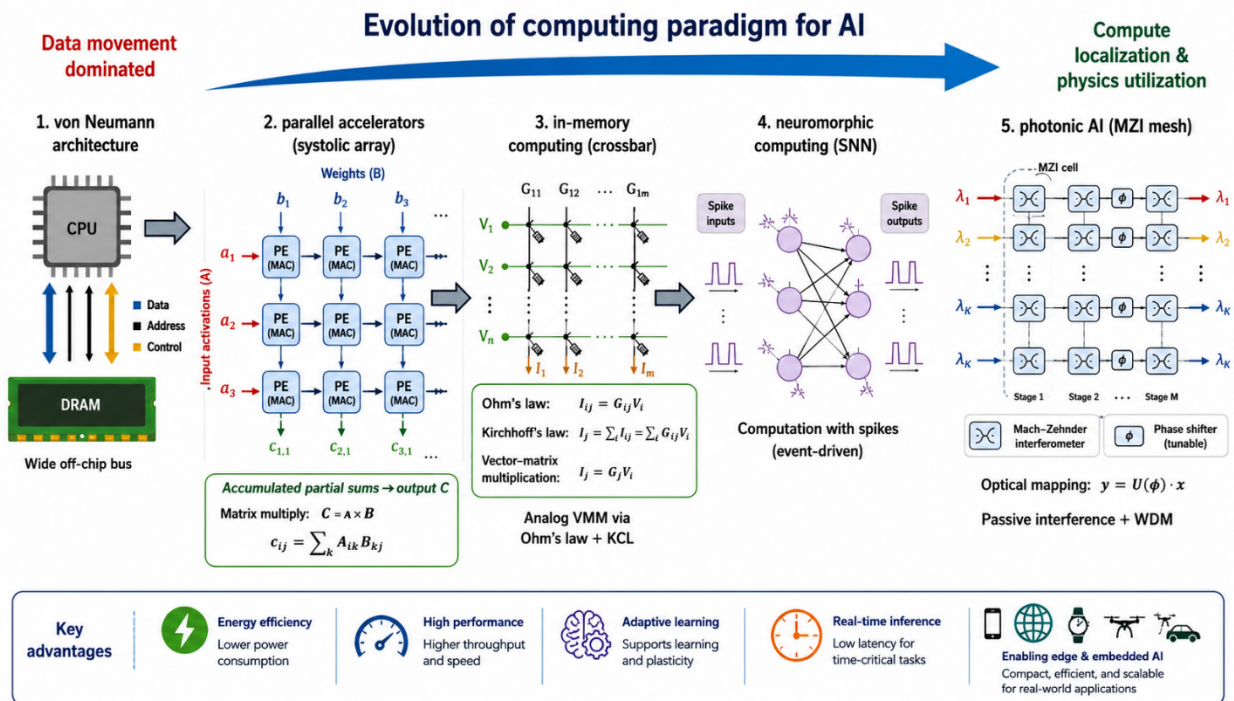


Figure 2. Evolution of artificial intelligence (AI) computing paradigms from Von Neumann processors to physics-driven platforms. The trajectory progresses from CPU-based systems bottlenecked by off-chip memory access, through massively parallel graphics processing unit (GPU) and tensor processing unit (TPU) accelerators, toward architectures that increasingly embed computation within physical substrates, including in-memory computing, neuromorphic spiking systems, and photonic processors, each motivated by the need to reduce data movement and improve energy efficiency.

AI workloads are dominated by dense linear algebra operations and have high data-level parallelism, which is well-suited to architectures that can perform a large number of arithmetic operations in parallel.

Graphics processing units were initially developed for graphics rendering, an application in which thousands of pixels must be processed simultaneously. Their single instruction, multiple data (SIMD) execution model, which executed thousands of lightweight threads in parallel, was highly effective for deep learning, enabling the processing of large batches of data simultaneously. With the advent of programming frameworks such as the Compute Unified Device Architecture (CUDA) and optimized libraries such as CUDA deep neural network (cuDNN), researchers were able to efficiently map neural network workloads to GPUs, and the training of large-scale AI models became the default platform of choice for researchers during the 2010s.²⁴ In 2012, the AlexNet result showed that GPU-accelerated deep learning could outperform the traditional computer vision methods by a large margin², marking a pivotal moment toward the dominance of GPU deep learning. Later generations of GPUs fused dedicated GPU

Tensor cores for mixed-precision matrix operations, such as the NVIDIA A100, which provides up to 312 TFLOPS of FP16 Tensor performance²³ to accelerate the training of large-scale AI models.

An advancement that followed was the Google TPUs, domain-specific accelerators designed for neural network operations, introduced in 2016. These were called systolic array architectures and involved repeatedly pumping data through arrays of multiply-accumulate units, resulting in lower control overhead than general-purpose processors and better data reuse.²³ With later versions of TPUs, additional on-chip memory was added to reduce off-chip memory accesses; in TPU v4, approximately 32 MB of on-chip static RAM (SRAM) is available.²⁵ They are reported to achieve up to 15–30 times faster inference performance than the CPU-based system and 2–5 times superior energy efficiency for image classification tasks (such as ResNet-50)²⁴ and, in some cases, 3.1 times faster inference performance than the A100-based system.²⁶

Nonetheless, both GPUs and TPUs have continued to adhere to the Von Neumann architectural model, in which computational and memory resources remain physically

distinct, and neural network weights must be repeatedly moved between them. With the shift to billions and trillions of parameters in AI models, the energy and latency of data transfer are now major performance factors, driving the investigation of paradigms that fundamentally rethink the memory–compute relationship.

2.2. In-memory and analog computing

Given the size limitation of the persistent memory in GPU and TPU architectures, an alternative approach was conceived: compute directly on the data where it is stored, thus avoiding the energy and latency costs of persistent memory. This concept, coined in-memory computing, was born from the realization that the weighted addition inherent in neural network inference can be naturally implemented using the physical characteristics of specific memory devices, without moving data to a separate processor.²¹

However, the significance of this paradigm shift is not so much in the specific circuit design as in the fundamental shift in conceptual thinking in how memory and computation are done now: memory aids computation; memory and computation are the same physical operation. This idea was widely explored in the experiments, and it was observed that this is not only possible but also quite feasible; for example, Diehl and Cook²⁷ demonstrated the feasibility of directly training and inferring a neural network within the memory array (digit recognition) using a memristor crossbar. Later demonstrations of competitive classification accuracy for some phase-change memory (PCM) array implementations, compared with digital implementations, further stimulated this direction of interest.²⁵

The basic concept of in-memory computing is to create a dense array of new non-volatile memory technologies, such as resistive RAM (RRAM), PCM, and magnetoresistive RAM (MRAM), that can store analog values, with multiply-accumulate operations directly derived from electrical laws. These devices and their organization into computing arrays via physical circuit architectures are discussed in Sections 3 and 4, respectively.

2.3. Brain-inspired (neuromorphic) computing

A new paradigm emerged from another source of inspiration: the extraordinary energy efficiency of the human brain. The brain is many orders of magnitude more efficient than conventional digital processors at doing cognitively intensive work—using just tens of watts to perform equivalent functions. This inspired the development of approaches to emulate the brain's computational methods in hardware. Concepts are rooted

in low-level mathematical models of neuronal dynamics. The integrate-and-fire model of the neuron was proposed by Lapicque²⁸ to describe how electrical signals build up in a neuron until they reach a threshold, at which a spike is produced. Later, McCulloch and Pitts²⁹ showed that a simplified version of the logical neuron could perform computation by being networked together, thus laying the foundations for neural computation. The modern idea of neuromorphic engineering was first explicitly stated by Mead²⁴, who advocated the design of analog very large-scale integration circuits that would directly emulate the neural processing of biological circuits, rather than simulating neural function on conventional processors.

Neuromorphic systems compute in fundamentally different ways from conventional processors. They process asynchronously rather than following a global clock, so computation can only occur when neural spike events are generated. In this SNN model, described by Maass²⁵ as the third generation of neural network models, information is encoded in the timing and rate of spikes, enabling efficient processing of signals that vary over time. With the advent of large-scale processors, the neuromorphic paradigm was put to the test. IBM TrueNorth achieved 1 million programmable neurons and 256 million synapses with low power usage (in the milliwatt range)³⁰, and Intel Loihi showed on-chip learning capabilities using local plasticity mechanisms.³¹ These milestones set the stage for integrating millions of neurons with energy consumption hundreds of times lower than that of traditional digital implementations. In Sections 4 and 5, the circuit architectures and learning algorithms that implement these systems are described.

2.4. Photonic and optical computing

The discovery of substrates for computing beyond electronics led to a new computing paradigm that leverages a different physical substrate: light. In the photonic computing approach, computation is carried out by photons in optical media rather than by electrons in pathways with resistance. Optical systems can, in principle, be extremely high-bandwidth, low-latency, and low-energy-consumption systems for certain computational tasks since photons have no resistive losses and travel at the speed of light.

Optical computing has been of interest for a long time, even before the era of deep learning. Early research showed that optical interference and diffraction could be used to perform convolution and Fourier transforms, the basic signal-processing operations.¹⁸ Later, theoretical work demonstrated that VMM could be performed via optical interference in an electro-optical system³², which laid the groundwork for optical neural computation. This

direction received further impetus when Shen *et al.*¹⁴ showed, for the first time, deep learning inference on a coherent nanophotonic circuit, laying the foundation for a viable photonic computing system to accelerate AI. The important point is that the weighted summation is performed passively, that is, without having to switch active components, and across many channels simultaneously, without needing a large number of optical components. An additional multiplexing property of wavelength-division multiplexing (WDM) is the ability to send a number of signals at different wavelengths through the same waveguide without disturbing each other, which is naturally missing in electronics. The structures of photonic devices and the photonic circuit architectures that enable these systems are described in Sections 3 and 4.

2.5. Other emerging approaches

In addition to in-memory, neuromorphic, and photonic computing, there are a number of other physics-based paradigms that emerge based on the intrinsic physics dynamics for computation. These methods are not deterministic digital logic, but rather utilize nonlinear dynamics, randomness, and quantum phenomena.

Oscillatory computing relies on networks of coupled oscillators that compute via their synchronization dynamics. Theoretical and experimental work initially showed that physical oscillator systems can support associative memory models similar to those of Hopfield networks, in which stable synchronization states represent patterns or optimal solutions stored in the network.^{33,34} More generally, coupled-oscillator dynamics has been recognized as a computational mechanism in nonlinear systems.^{35,36} This paradigm goes beyond the rate-based representation of information in neuromorphic computations by coding information in oscillators' phases rather than their rates, and by computing using phase synchronization and dynamical convergence.

Recognizing that many machine learning algorithms are robust to noise and that the noise generated by nanoscale devices could be used rather than ignored, stochastic and probabilistic computing emerged. In stochastic computing, the numbers are encoded as probabilistic bit streams, so that arithmetic is possible using only simple logic gates.³⁶ In a more general context, probabilistic computing is a way of dealing with uncertainty as a computational resource, and there is a natural substrate for Bayesian inference, probabilistic graphical models, and generative modeling in this space.^{36,37}

It was suggested that the nonlinear dynamics of a

recurrent system be used to process temporal information using reservoir computing.³⁸ The reservoir is a fixed dynamical system that capitalizes on the input signal to generate high-dimensional representations, which are read out by a simple linear output layer and classified afterward. This idea has been realized in a variety of physical media, including photonic circuits, analog electronic systems, and spintronic oscillators. Quantum-inspired optimization algorithms can be combined with physical devices, including Ising spin models, a field known as quantum-inspired computing that links quantum-inspired algorithms and physics-driven devices.^{14,39} This direction includes the quantum approximate optimization algorithm, which is one of the hybrid quantum-classical computation frameworks for combinatorial optimization.³⁹ Although a large-scale fault-tolerant quantum computer is still being developed, quantum-inspired techniques are effective in solving optimization and sampling problems in AI.

Common problems across these emerging paradigms include maintaining intrinsic noise and variability, scaling nonlinear dynamics to large problem sizes, and benchmarking against conventional accelerators for deep learning. Together, they show how computation can be integrated into the actual dynamics of the hardware, which is what all physics-based approaches to AI have in common. In the next section, the physical technologies that make each of these paradigms possible are analyzed.

3. Device-level foundations for physics-driven artificial intelligence

The new generation of nanoscale devices, which go beyond conventional binary CMOS switches, enables the evolution of computing paradigms described in Section 2. Emerging electronic and photonic devices have complex physical processes, including resistive switching, magnetic spin interactions, phase transitions, and optical interference, which can be directly implemented to implement computational primitives, unlike digital transistors, which need to act as deterministic on/off switches. This allows physics-native computation, that is, the material dynamics are used directly for AI computation rather than being emulated by digital logic circuits.^{14,40} The physical mechanisms, material properties, and device-level computational primitives of five major classes of devices: memristive, phase change, spintronic, photonic, and neuromorphic CMOS devices are explored. The circuit topology used to organize functional computing systems is discussed separately in Section 4. These device types are discussed in detail in the subsections below, and their physical structures and operating mechanisms are summarized in [Figure 3](#).

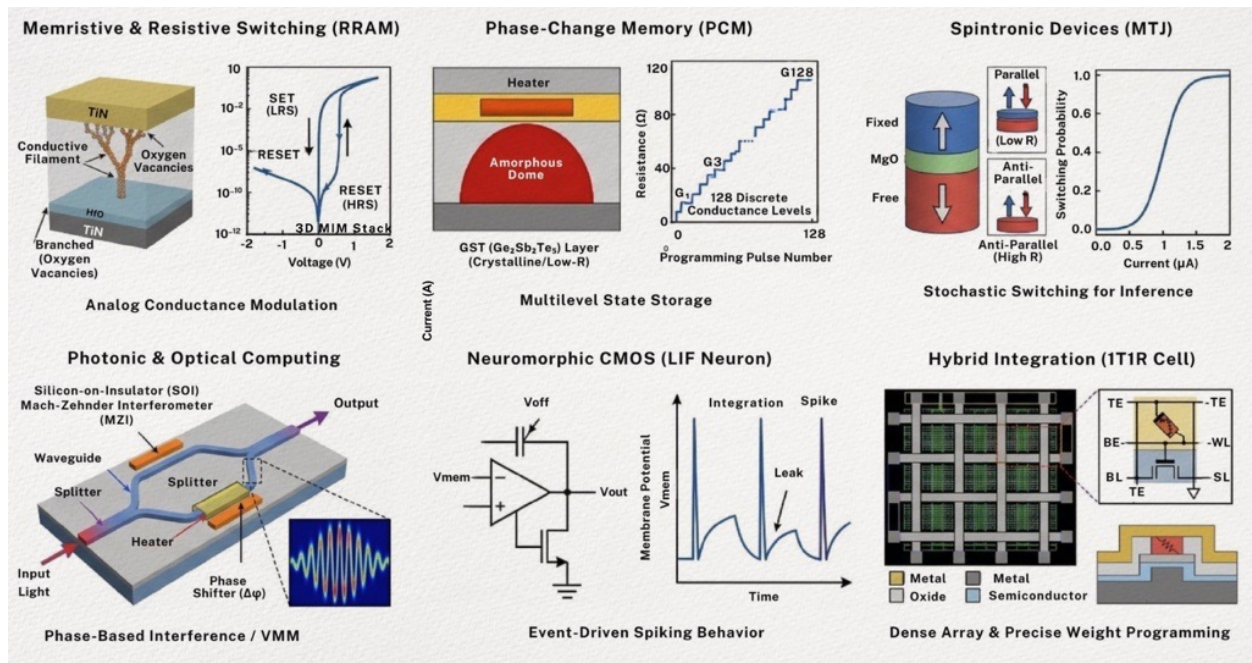


Figure 3. An overview of device-level implementation of emerging platforms in physics-inspired computing, including neuromorphic computing and analog in-memory computing, consisting of resistive and phase-change memory devices, spintronic magnetic tunnel junctions (MTJs), photonic circuits, complementary metal oxide semiconductor (CMOS), and hybrid one-transistor integration schemes.

Abbreviations: BE: Bottom electrode; BL: Bit line; RRAM: Resistive random-access memory; SL: Select line; TE: Top electrode; VVM: Vector-matrix multiplication; WL: Word line.

3.1. Memristive and resistive switching devices

The memristor is a theoretically proposed fourth fundamental circuit element related to charge and magnetic flux.⁴¹ Later, non-volatile modulation of the resistance state of such metal-oxide thin-film devices was practically demonstrated, showing that internal physical processes can be used to modulate and retain the resistance state non-volatily.⁴⁰

In most of the experimental manifestations, the memristive switching is triggered by one of three physical effects: either by the movement of metal ions or oxygen vacancies (ionic migration) under an electric field, and the formation of a conducting bridge (filament) between the electrodes; by modulation of the resistance at the interface between the metal and the oxide (interfacial resistance modulation); or by modulation of the bulk electronic effects of a strongly correlated material (bulk electronic modulation). The conductive filament formed by ionic migration has been demonstrated to be stable and reproducible in oxide-based devices.⁴² The materials explored range from the transition metal oxides (HfO and TaO_x) to the chalcogenides and new two-dimensional

materials, with different switching kinetics and retention properties.

One physical characteristic unique to memristive devices is that their conductance can be continuously controlled at many intermediate states, rather than switching between two binary states. This multilevel behavior is due to partial filament formation/dissolution, enabling the device resistance to be programmed to a desired analog value. This property allows memristive devices to physically store conductance states that can represent synaptic weights, a graded value that can be stored non-volatily.^{40,43} At the single-device level, the relationship between the device current, applied voltage, and conductance is as follows:

$$I_{ij} = G_{ij} V_i \quad (1)$$

where G_{ij} is the conductance of the programmable device, and V_i is the applied voltage. This Ohmic property of the device is the physical basis for analog multiply-accumulate operations; how multiple devices can be configured into computing arrays to exploit this property is discussed in Section 4.

There are several physical non-idealities related to the stochastic nature of filament formation and ion migration. The variation from device to device is due to the random nucleation of the conductive filament, and the variation from cycle to cycle is due to irreproducible filament rupture and nucleation. Nonlinear and asymmetric conductance update characteristics, in which the conductance increase per programming pulse is not the same based on the conductance state, make it difficult to program a precise weight. Limited endurance is due to progressive damage to the switching layer from repeated cycling, and conductance drift with cycling time is due to progressive relaxation of the filaments.¹⁴ The reliability of circuits and systems designed based on these physical non-idealities can be crucially dependent on the accuracy of the compact models.⁷ These effects can be minimized by device engineering and compact modeling, but they can never be eliminated, and the analog precision vs. device endurance trade-off will continue to be a challenge for widespread use.¹⁴ The effects of these can be reduced by engineering the devices and by using compact models, but cannot be fully removed, and there is an opportunity for research in large-scale deployment for a trade-off between analog precision and device endurance. Although these can be mitigated by various device engineering and compact modeling methods, they cannot be completely avoided, and there is an opportunity for research on large-scale deployment to balance analog precision and device endurance.

3.2. Phase-change memory devices

Phase-change memory devices exploit reversible structural phase transitions in chalcogenide materials, typically $\text{Ge}_2\text{Sb}_2\text{Te}_5$, that result in the amorphous and crystalline phases.⁴⁴ Thermally induced phase transitions are achieved via electrical heating pulses using an adjacent resistive heater element. When the chalcogenide material is rapidly melted and quenched to the disordered amorphous state by a high-amplitude, short-duration pulse called RESET, it has a high electrical resistance. A lower-amplitude, longer-duration pulse (SET) will anneal the material above the crystallization temperature, and the ordered crystalline phase will revert to a lower electrical resistance. The resistance difference of the two phases is several orders of magnitude, resulting in a wide read margin and reliable state detection.⁴⁴

Phase-change memory devices allow intermediate conductance with finely graded intensity, rather than binary switching, through controlled partial crystallization. Carefully varying the amplitude and duration of the SET pulses allows crystallization of a range of volume fractions within the chalcogenide film and, hence, the

attainment of intermediate resistance values between the fully amorphous and fully crystalline extremes. PCM devices can have over 100 distinct conductance levels per cell⁷, which is significantly more than the typical range of a resistive memory device and the level of precision needed for gradient-based training of neural networks.

Conductance drift, which is the change in the resistance of the amorphous phase with time, is the main physical issue unique to PCM that arises because of the tendency of the disordered structure of the amorphous phase to relax toward a more stable configuration. These are inherent material properties that cannot be optimized solely at the programming level and should also be considered when designing at the system level.⁴⁵ Other physical problems include cycle-to-cycle variations in programming, limited endurance due to structural damage from repeated phase transitions, and thermal crosstalk between neighboring cells in close-packed arrays. Especially in the context of PCM-based training, one of the most important yet unanswered questions is the time over which stored weights can be guaranteed to remain accurate, which is directly constrained by conductance drift. Maintaining reliable weights on the PCM over the long term is one of the most important unsolved problems for such systems⁴⁴, due to conductance drift.

3.3. Spintronic devices

Spintronic devices exploit the spin degree of freedom of electrons, in addition to charge, to enable a new class of non-volatile devices with high endurance and low switching energy.⁴⁶ The basic element of spintronic devices is the magnetic tunnel junction (MTJ), composed of two ferromagnetic layers that are separated by a thin insulating tunnel barrier, usually MgO. The electrons tunnel through the barrier quantum mechanically, and the probability of tunneling and hence the device resistance depend on the relative orientation of the magnetization vectors of the two ferromagnetic layers. Low resistance is observed when the magnetizations are parallel, and high resistance when they are antiparallel. This effect, called tunnel magnetoresistance, is the physical mechanism responsible for non-volatile information storage in spintronic devices.⁴⁷

Spin-transfer torque (STT) devices switch in response to the flow of a spin-polarized current through the MTJ stack. As the spin-polarized electrons flow through the free ferromagnetic layer, they transfer angular momentum to the free layer, and the torque is great enough to switch the direction of its magnetization when the current is larger than a critical switching current. Spin-orbit torque (SOT) devices separate the paths of the read and write currents by relying on the giant spin Hall effect in an adjacent heavy-

metal layer, which generates a transverse spin current that switches the free-layer magnetization. The advantage of SOT switching over STT is that the oxide tunnel barrier is not strained during writing^{19,48}, thereby improving energy efficiency and endurance.

In addition to binary switching, some spintronic structures can sustain self-oscillation. Spin-torque nano-oscillators (STNOs) are devices that can produce magnetization precession at microwave frequencies when operated by a direct current above a threshold, resulting in an oscillating resistance and a microwave voltage output. The oscillation frequency and phase can be controlled by the current and magnetic field applied to the device, offering a single-device-level electrically controllable oscillatory degree of freedom. This is a physical property that is the device-level basis for oscillatory and phase-based neural computation.^{40,47}

When the magnetic free layer is only a few nanometers in size, thermal energy fluctuations can approach the energy barrier between the parallel and antiparallel states. This renders the switching process probabilistic instead of deterministic, and the switching probability can be controlled by applying current, pulse duration, and temperature. The randomness of spintronic devices can be precisely described and, as a physical resource for probabilistic computing, differs from that of memristive devices. Hardware-native random number generation with tunable probability is possible using stochastic MTJs operating in the thermally unstable regime, which can function as physical probabilistic neurons.⁴⁹ This can be used in combination with the deterministic multilevel analog states of the PCM devices described in Section 3.2.

3.4. Photonic devices

Photonic devices exploit the physical properties of light for computation and offer a fundamentally new hardware substrate compared with charge-based electronic devices. Photonics devices can enable ultra-high bandwidth, low latency, and energy-efficient signal processing, as photons can travel at the speed of light with no resistive losses in optical waveguides.

The information used in computing is encoded in the physical properties of the optical fields, including the amplitude, phase, or wavelength of the signal. The physical process that allows for linear computation in photonic systems is optical interference, namely the constructive or destructive addition of the electric fields of two coherent optical fields when they interact in the same space. This is a passive physical process that does not require any active switching energy per operation, instead performing a weighted summation.

Photonic computing platforms are made of several building blocks: photonic device components. Optical waveguides can confine and guide photons by total internal reflection at the interface between a high-refractive-index core and a lower-index cladding. Programmable optical interference can be achieved using phase shifters that modulate the optical phase of a guided signal based on the thermo-optic or electro-optic effect. Micro-ring resonators are resonant structures that can be wavelength-selective, with transmission characteristics controlled by varying temperature and/or carrier injection rate. The conversion of optical signals back to electrical currents is performed by photodetectors at the output of a photonic circuit. The physical difference between photonic devices is that more than one optical signal at different wavelengths can travel simultaneously in the same waveguide without interfering, since they are in orthogonal modes of the electromagnetic field. This wavelength orthogonality allows the wave to exhibit inherent spatial parallelism at the device level without any electronic analog.¹⁴

Linear operations are easily and efficiently performed in the optical domain using optical interference; however, introducing nonlinear optical effects at the device level is quite difficult, as such effects generally require high optical intensities and/or long interaction lengths. This fundamental limit of physics renders the use of nonlinear functions in photonic AI systems, in general, optoelectronic.⁵⁰ The manufacturing tolerances for photonic devices are also very tight, typically 1% or less, to ensure good inference accuracy. In addition to fabrication tolerances, refractive index drift due to thermal crosstalk between optical components continuously disturbs phase settings during operation, requiring correction by active circuits that incur power and area overheads.

3.5. Neuromorphic complementary metal-oxide-semiconductor and hybrid devices

Neuromorphic CMOS devices use standard semiconductor technology to implement brain-like computation primitives, whereas hybrid approaches use emerging device technologies. The leaky integrate-and-fire (LIF) neuron model is the simplest model that mimics the essential dynamics of biological neurons in two coupled physical processes. A passive, physical integration process (no switching energy required) first occurs: a capacitor charges to the neuron's membrane potential when synaptic input currents flow into the neuron. Second, a threshold comparator determines the capacitor's voltage; when it exceeds a firing threshold, a spike is generated, and the capacitor is discharged. This implementation does not consume active energy during the integration phase, except for rare spike activations.⁴⁹ This is the source of the energy efficiency of this implementation.

A hybrid CMOS–memristor architecture can directly encode the synaptic weight in a memristive device, storing the physical state of resistance, without using a separate SRAM cell to store the weight value. Similarly, PCM devices can be used for high-precision multilevel weight storage, while spintronic MTJs can be used for fast switching with high endurance for frequently updated synaptic connections.²³ An important integration scheme at the device level is the one-transistor-one-resistor (1T1R) cell, in which a select transistor is connected in series with each memory device. The transistor enables accurate current adjustment during programming and prevents sneak-path leakage, enabling individual devices to be addressed. Memory stacks can be co-fabricated on top of the CMOS layer using back-end-of-line processing and provide high-density vertical integration without increasing chip area.²³ This is a physical integration-based approach, at the device level, for the crossbar computing arrays described in Section 4. The physical mechanisms of the devices described in this section lay the groundwork for the circuit architecture described in Section 4 and for how these devices are organized, interconnected, and controlled to form functional computing systems. Table 1 offers a summary comparison of the five device platforms covered in this section.

4. Circuit-level implementations

In the case of physics-driven AI, the actual physical mechanisms of the device are described at the device level in Section 3 and serve as the basic computational primitives upon which the AI can operate. Turning these primitives into practical and scalable computing systems necessitates circuit architectures that organize, connect, and control devices to create hardware platforms that can perform valuable computation. Circuit-level design addresses practical issues that do not arise at the single-device level, such as analog-to-digital and digital-to-analog signal conversion, asynchronous communication across large arrays of circuits, and the integration of different device technologies into a single design. Four key circuit-level architectures: in-memory computing arrays, neuromorphic and event-driven circuits, oscillatory and stochastic circuits, and photonic circuits are explored. For each, the discussion emphasizes the circuit-level organization and control of devices, while referring to the discussion of device physics in Section 3 and the co-design of algorithms and hardware in Section 5.

4.1. In-memory computing circuits

In-memory computing circuits can directly perform VMM operation in memory arrays, avoiding the continual copies and movement of data between the data

Table 1. Comparative overview of physics-driven AI hardware platforms

Platform	Computational principle	Scalability	Precision	Energy efficiency	Primary non-ideality	Application targets
Memristive (RRAM)	Analog conductance for in-memory VMM	High	4–8 bits effective	Tens of TMACS/W	Variability, endurance, drift	Edge inference, on-chip learning
Phase-change memory	Multilevel amorphous–crystalline transitions	Moderate	Up to 100 levels/cell	High for inference	Conductance drift, thermal cross-talk	Synaptic storage, analog training
Spintronic (MTJ)	TMR for memory; STNOS for oscillatory computation	High endurance	Binary to low-precision	Sub-pJ switching energy	Thermal stability, variability	Non-volatile memory, probabilistic computing
Photonic	Optical interference for passive linear VMM	Limited by fabrication tolerances	6–8 bits effective	Ultra-low latency; laser overhead limits net efficiency	Phase calibration, thermal drift	High-speed inference, data-center AI
Neuromorphic CMOS	Event-driven spike integration; hybrid CMOS-memory synapses	High (millions of neurons)	Spike-based temporal encoding	Milliwatt-range for sparse workloads	SNN training difficulty, interface overhead	Edge AI, healthcare, robotics

Abbreviations: AI: Artificial intelligence; CMOS: Complementary metal-oxide-semiconductor; MTJ: Magnetic tunnel junction; RRAM: Resistive random-access memory; SNN: Spiking neural network; STNO: Spin-torque nano-oscillator; TMACS/W: Tera multiply-accumulate operations per second per watt; TMR: Tunnel magnetoresistance; VMM: Vector–matrix multiplication.

processor and memory, which are responsible for most energy consumption in conventional Von Neumann accelerators.^{7,51}

The basic structure of circuits in in-memory computing systems is the crossbar array, where a physical switching device, such as a memristor, a PCM cell, or a ferroelectric field-effect transistor (FeFET), as described in Section 3, is crossed by horizontal word and vertical bit lines. Analog voltages (V_i) are given as inputs row-wise. As per Ohm's law, the currents flowing through each of the devices are as in **Equation 1**; the currents thus add up along each column as per Kirchhoff's current law:

$$J_j = \sum_i G_{ij} V_i \quad (2)$$

This column current J_j corresponds directly to the dot product of the input voltage vector with the j -th column of the conductance matrix, implementing VMM in a single passive electrical step across the entire array simultaneously, without any data movement.²¹ Digital-to-analog converters (DACs) are needed to convert the digital input activation values to analog voltages applied to the rows of the crossbar, and analog-to-digital converters (ADCs) are needed to convert the analog output currents from the columns into digital values for use in downstream processing. ADC and DAC circuits are often a significant contributor to system-level energy consumption and latency, and in some cases, their power consumption accounts for most of the total system power, partially eliminating the benefits of analog in-array computation.^{21,30}

There are several circuit-level effects that do not occur at the single-device level and affect the accuracy of crossbar VMM. The voltage supplied to devices farther from the driver on the word or bit line will differ from the desired input voltage due to current-resistance product drop across the word or bit line resistance, resulting in systematic computation errors that increase with the array size. In passive crossbar arrays, sneak-path currents cause a false loading on column currents, and circuit-level modeling has demonstrated the need for correct sneak-path characterization to guarantee reliable array design for large arrays.³¹ The 1T1R cell structure has been proposed to solve the sneak-path currents problem by adding a select transistor (1T) for every memory device, thus individual cell addressing is achieved at the expense of increased cell area.^{21,31} The hierarchical array tiling approach is to divide large logical arrays into small physical sub-arrays with their respective local peripheral circuits to limit IR drop and the accumulation of sneak-paths to manageable levels.

To achieve a higher effective number of bits of precision

than is possible with a single analog device, a bit-slicing architecture stores each synaptic weight in a bunch of low-precision memory elements and then combines the output of the memory elements in the digital domain with the correct binary weighting. Open and closed loops of write-and-verify circuits repeatedly program each device and check its conductance state to account for device variability and achieve more accurate weight placement.^{21,31} Implementing in-memory computing circuits based on FeFETs has reached power efficiency of more than 800 TOPS/W⁷, and large-scale systems based on memristors have been able to reach energy efficiency of tens of TMACS/W with mixed precision.^{7,31} One such example of the necessity of cross-layer co-design is the consideration of memristive conductance variability in the system's accuracy. The variability across memristive device cycles is 10–20%, which can cause systematic errors in each crossbar's output in the VMM and reduce inference performance by a few percentage points when using the VMM without hardware-aware optimization. When trains are noise-aware, and circuits include closed-loop write-and-verify circuits, this error rate can be reduced to less than 1% for standard classification circuits without sacrificing algorithmic or circuit fixes.

4.2. Neuromorphic and event-driven circuits

The components of neurons and synapses are arranged into asynchronous, event-driven systems called neuromorphic circuits that compute only when they receive a spike event and do not use a global clock. This paradigm at the circuit level enables high savings in power consumption and data transfers for sparse, temporally encoded workloads.²³

The LIF neuron model is implemented at the circuit level by using three functional blocks. The passive integration of membrane potential over time is achieved through a membrane capacitor, C_m , which stores incoming synaptic currents. A leak resistor provides a continuous discharge path, allowing the membrane potential to decay toward the resting voltage. When the capacitor voltage V_m reaches a threshold, the threshold firing voltage V_{th} is detected, generating an output spike pulse and quickly resetting the capacitor voltage to a specific level, thereby quickly entering the refractory period. The analog CMOS version of this circuit consumes power only during spikes but is vulnerable to process variations.³¹ In recent years, mixed-signal solutions that integrate analog and generate digital spikes in the neuromorphic processor are gaining more traction.⁵²

In hybrid neuromorphic designs, synaptic weights are physically stored in a non-volatile memory device, and read-current sensing and spike-triggered weight-update

operations are performed by CMOS circuits. Coincidence detection circuits can implement spike-timing-dependent plasticity (STDP) locally at the synaptic level, enabling on-chip weight adaptation without relying on an external training infrastructure.¹⁴

Communication in large-scale neuromorphic systems uses asynchronous communication (address-event representation [AER]), where spikes are sent as digital packets containing the address of the firing neuron. AER-based interconnects transmit information sparsely at the circuit communication level by implementing event-driven sparsity.⁵² IBM TrueNorth is a fully digital, event-driven, massively parallel architecture with 1 million programmable neurons and 256 million synapses, consuming a few milliwatts of power when performing inference workloads.³⁰ Intel Loihi has on-chip learning circuits that enable local plasticity rules, such as STDP³¹, and programmable spiking neuron models.

4.3. Oscillatory, stochastic, and probabilistic circuits

Oscillatory, stochastic, and probabilistic circuits leverage the inherent dynamics and randomness of physical systems as computing resources, rather than treating them as sources of error to be suppressed.⁵³ Networks of coupled oscillators are realized in the circuit domain as CMOS ring oscillators, STNOs, or memristive relaxation oscillators. Individual oscillators are coupled to one another via electrical resistors, capacitors, or mutual inductance, and the values of these components determine the strength of the coupling. Injection locking, in which a reference signal is injected into an oscillator, locks the oscillator's phase to the injected reference, and mutual phase coupling, in which two oscillators interact to establish stable phase relationships, are the two main circuit-level mechanisms that underlie computation. The collective circuit dynamics converge toward phase configurations that minimize a global energy function, physically implementing Ising machine optimization:

$$E = -\sum_{i,j} J_{ij} S_i S_j \quad (3)$$

where S_{ij} is the state of the phase of oscillator i/j , and J_{ij} is the coupling strength between oscillators i and j . This directly maps to the physical dynamics of the circuit, the combinatorial optimization problems.⁵⁴

Stochastic circuits use the natural noise from devices, such as thermal noise, random telegraph noise, and stochastic MTJ switching, to produce random bitstreams at the circuit output. The numbers $x \in [0, 1]$ are represented in the stream by the probability that a particular bit in the stream is set to logic 1. Then the minimal logic circuits are

used to carry out arithmetic operations: the multiplication is performed using a single AND gate:

$$y = x_1 \cdot x_2 \quad (4)$$

and the scaled addition by a multiplexer. This significantly reduces hardware complexity compared with conventional binary arithmetic hardware units.^{52,53} Tunable probability distributions are wired into circuit configurations of stochastic MTJs and phase change devices, which have intrinsic randomness. By using voltage-controlled biasing circuits, the switching probabilities of stochastic devices can be programmed to produce random outputs that can be used as physical samplers in Markov Chain Monte Carlo (MCMC) sampling and Boltzmann machine (BM) inference, enabling hardware-efficient probabilistic AI without software-based random number generation.^{44,52}

4.4. Photonic and optical circuits

For use in computing systems, the waveguides, interferometers, modulators, and detectors described in Section 3.4 need to be organized and interconnected into a photonic circuit. The basic programmable circuit element in photonic computing is the Mach-Zehnder interferometer (MZI), which consists of a pair of phase shifters and a pair of 50/50 beam splitters and can be used to perform a programmable unitary 2×2 transformation. An arbitrary $N \times N$ unitary matrix transformation can be implemented by arranging $N(N-1)/2$ MZIs in a triangular or rectangular mesh topology, thus forming the circuit-level basis for optical VMM.¹⁹

By using wavelength-selective components such as arrayed waveguide grating or banks of micro-ring resonators, multiple wavelength channels can be routed through a common waveguide network in a WDM circuit architecture, and many independent VMM operations can use the same physical MZI mesh using different wavelength channels. With WDM technology, more than 100 parallel computational channels can be accommodated on a single waveguide, far more than can be achieved with electronic interconnects.⁷ Further increasing computational throughput is temporal multiplexing, where multiple data streams are interleaved in time using optical delay lines and fast switches.

The current photonic AI circuits are in a hybrid configuration. In the optical domain, high-speed, passive linear operations are performed using MZI meshes. Nonlinear activation functions, memory of the weight, scheduling, and generation of control signals for the phase shifters are generated in the electronic domain. Interface circuits include electro-optic modulators and photodetectors and are used to convert signals between the

electrical and optical domains at the circuit boundaries. One of the most important circuit-level specifications is active phase calibration, which can be used to correct static phase offsets due to waveguide geometry variations that degrade VMM accuracy. Closed-loop correction circuits keep the phase error below 1%.²³ Fabrication-induced phase errors from 1–2% have been demonstrated to decrease the inference accuracy of a neural network by 5–15% compared to the ideal case (dependent on both the depth of the neural network and the number of MZI stages). The power and area costs of these active calibration circuits are intrinsic to any system-level efficiency analysis, as they are directly driven by these circuits.^{14,50} A summary of the circuit-level architecture discussed in this section is presented in Table 2, which cross-correlates the platform with the important circuit-level architectures in each, the primary operation performed by each architecture, and the major problem at the circuit level.

5. Algorithm-hardware co-design

A major paradigm to enable efficient AI systems on new computing substrates is the co-design of algorithms and hardware. AI paradigms such as in-memory computing systems, neuromorphic processors, and photonic accelerators deviate from the traditional digital computing paradigm because their algorithms depend on the devices' physical properties, including finite precision, device variability, stochastic switching, and other nonlinear response characteristics.

Algorithms assume a certain kind of relationship that

may not hold in hardware, leading to a massive loss of accuracy when directly mapped onto it.⁵⁵ Thus, co-design methods are becoming more and more in demand, where the learning algorithm is specifically designed for the physical properties of the underlying hardware, especially in the context of analog and mixed-signal systems, in which the amount of computation is limited by the physics of the hardware.⁷

Algorithms and hardware design go hand in hand, and, most importantly, hardware defects should not be overcome solely by the algorithm but leveraged as a computational resource. The stochastic nature inherent in nanoscale devices can be used for probabilistic inference and sampling-based learning⁸, and analog dynamics in a physical system are an efficient way to implement an optimization process by minimizing the energy of the system.⁵² Since computation in neuromorphic systems is event-driven, it offers sparse and asynchronous information processing with much less energy consumption than synchronous digital ones.⁴⁵ Co-design has been reported to bring about significant benefits in terms of energy efficiency, latency, and scalability, specifically for edge and resource-constrained workloads²¹ by co-optimizing the algorithm design with device and circuit properties.

5.1. Co-design framework and cross-layer optimization

The algorithm–hardware co-design can be considered a multi-layer optimization framework that accounts for the constraints and opportunities of devices, circuits,

Table 2. Summary of circuit-level architectures for physics-driven AI

Circuit type	Key circuit structures	Primary computation	Device basis	Dominant challenge
In-memory computing	Crossbar arrays, 1T1R cells, bit-slicing, ADC/DAC peripherals	Analog VMM via Ohm's and Kirchhoff's laws	Memristors, PCM, FeFET	ADC/DAC overhead, current-resistance product drop, sneak-path currents
Neuromorphic and event-driven	LIF neuron circuits, synaptic weight circuits, AER interconnects	Event-driven spike integration and communication	Neuromorphic CMOS, hybrid memory synapses	Analog-digital trade-off, SNN training mapping
Oscillatory and stochastic	Coupled ring oscillators, STNOs, stochastic bitstream generators	Phase synchronization, Ising energy minimization, MCMC sampling	CMOS oscillators, spintronic MTJs, PCM	Synchronization stability, noise control, digital interface
Photonic and optical	MZI meshes, WDM multiplexers, AWGs, hybrid electro-optical interfaces	Optical VMM via interference, wavelength-parallel computation	Waveguides, phase shifters, micro-ring resonators	Phase calibration, nonlinear activation, CMOS integration

Abbreviations: 1T1R: One-transistor-one-resistor; ADC: Analog-to-digital converter; AER: Address-event representation; AI: Artificial intelligence; AWG: Arrayed waveguide grating; CMOS: Complementary metal-oxide-semiconductor; DAC: Digital-to-analog converter; FeFET: Ferroelectric field-effect transistor; LIF: Leaky integrate-and-fire; MCMC: Markov Chain Monte Carlo; MTJ: Magnetic tunnel junction; MZI: Mach–Zehnder interferometer; PCM: Phase-change memory; SNN: Spiking neural network; STNO: Spin-torque nano-oscillator; VMM: Vector–matrix multiplication; WDM: Wavelength-division multiplexing.

architectures, and algorithms. Unlike classical design flows, where hardware and software design are regarded as largely independent layers, co-design is a correlation in which the possibility to use a given hardware influences the design of the algorithm, and the need for a given algorithm influences the choice and configuration of the hardware used in its implementation.¹⁰

Co-design is, at heart, a way to cope with the mismatch between idealized algorithmic models and non-ideal hardware implementations. Unlike conventional deep learning, which is assumed to use high-precision arithmetic with exact weight updates, emerging platforms support low- or medium-precision arithmetic and are inherently nondeterministic due to noise, variability, and the nonlinearity of the platform.^{16,56}

The co-design framework of D-C-A is shown in Figure 4. Algorithm-level methods limit precision, tolerance to variability, and sparsity. Top-down and physics-level primitives drive hardware capabilities and constraints top-down from device/circuit levels. At the circuit-architecture level, these two directions converge to provide co-optimized systems that leverage, rather than merely tolerate, physical dynamics.

Physically, the variability, noise, nonlinearity, and limited precision of the devices add constraints on the behavior of the circuits and performance of the

algorithms. Conductance drift and stochastic switching in memristive devices affect the stability and accuracy of the weights that, in turn, affect training dynamics and inference reliability.^{18,56} Developing effective precision, latency, and energy efficiency at the circuit level is directly tied to algorithmic requirements, such as quantization levels, training strategies, and model architectures^{7,57}, which are influenced by design choices such as the size of the crossbar array, the resolution of the ADC/DAC, and peripheral overhead. Algorithms such as hardware-aware training, quantization-aware learning, and noise-aware optimization modify models to account for hardware constraints, achieving robust, efficient performance in the presence of hardware nonidealities.^{16,56} The main enablers for effective cross-layer co-optimization are hardware-aware algorithms with explicit consideration for the characteristics of the devices and circuits in the system during training^{16,57}; algorithm-architecture co-design where the algorithm structure is tailored to the hardware capabilities of the system^{16,21} mixed precision computing, where different layers of the system are allocated different number of bits for precision⁵⁷; and calibration and adaptive tuning, which compensate for device variability and aging during the system lifetime.¹³ With the increasing trend of using modern AI accelerators that feature a mix of digital CMOS cores, analog in-memory computing blocks, neuromorphic processors, and photonic interconnects,

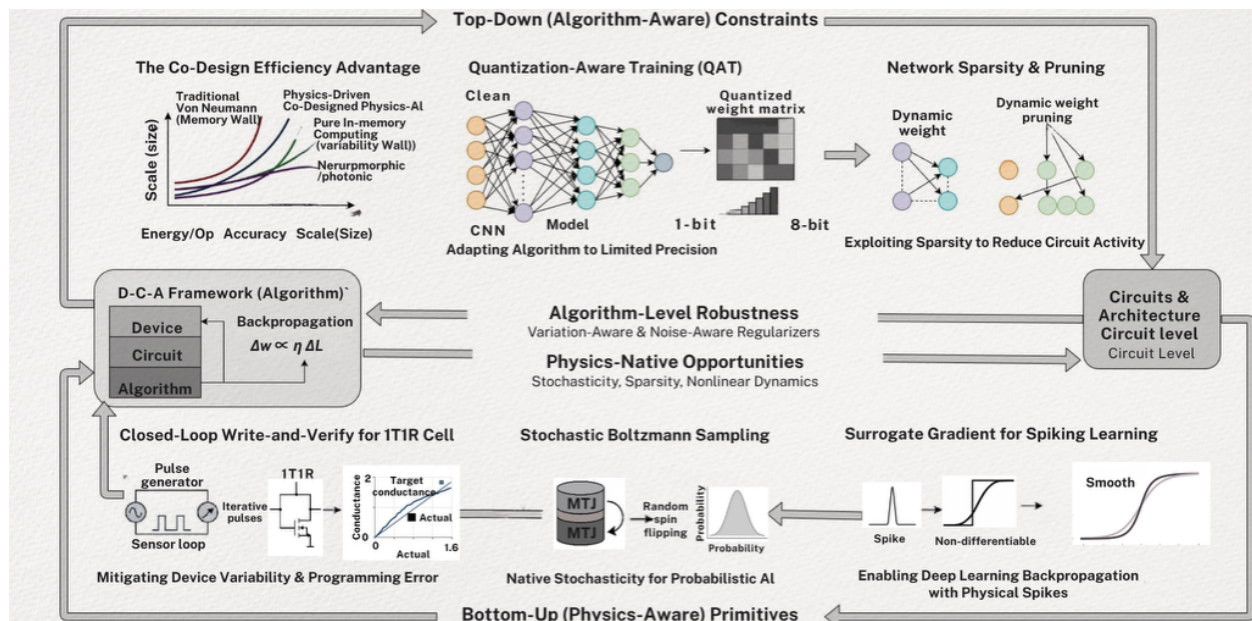


Figure 4. Cross-layer device-circuit-algorithm (D-C-A) co-design framework combining top-down algorithm-constrained and bottom-up physics-constrained primitives to neuromorphic and physics-driven artificial intelligence (AI) systems. Abbreviations: 1T1R: One-transistor-one-resistor; CNN: Convolutional neural network; MTJ: Magnetic tunnel junction.

careful co-design of communication protocols, dataflow, and workload partitioning is required.^{16,21}

5.2. Precision-aware and hardware-aware training

The software–hardware mismatches are associated with the non-ideal behavior of emerging hardware platforms and are the subject of the research area known as hardware-aware training. Conventional deep neural networks are trained under the assumption of high-precision numerical computation, fixed weight updates, and precise linear operations. However, when implemented in analog or mixed-signal systems such as resistive crossbar arrays, PCM systems, and neuromorphic processors, the above assumptions are no longer valid because of device variability, limited precision, noise, and nonlinear programming characteristics.^{16,56,57}

Weight updates are often nonlinear and asymmetric in analog in-memory computing systems because of the physics of the devices described in **Section 3.1**, and inference is subject to conductance drift, IR drops, and read noise.^{18,56} Noise-aware training adds stochastic perturbations to simulate the device variability, thus enhancing generalization and robustness when deployed in hardware.⁵⁶ Reports of device-aware training frameworks that explicitly model conductance quantization and a restricted dynamic range enable community networks to be tailored to discrete, restricted weight representations.⁵⁸

Quantization-aware training (QAT) bounds weights and activations to low-bit representations, such as 4-bit or 8-bit, and mimics the quantization effect during training to maintain accuracy at inference time.⁵⁹ It has been demonstrated that QAT can be used with reduced precision to achieve near-floating-point precision while reducing the energy required to deploy it on a platform with limited resources.⁵⁸ A good compromise between precision and efficiency is given by mixed-precision strategies, where important layers have high precision.⁵⁷

In a memristive crossbar array, the incremental conductance modification is also a nonlinear function of the programming pulses; training algorithms that account for this nonlinearity can model the weight evolution during training to match the actual device programming behavior.⁵⁶ The presence of non-ideal synaptic devices further boosts the training accuracy and convergence in closed-loop write-and-verify schemes and pulse-based update models.¹³ These stochastic perturbations, such as dropout or adversarial noise injection, are regularization techniques that can also improve generalization in low-precision and analog computing regimes^{16,56} while also

helping to create deployment-robust networks.

5.3. Spiking neural networks and learning rules

Spiking neural networks are a fundamentally new paradigm of computation that propagate information and compute in the form of spike events rather than continuous-valued activations, as in conventional artificial neural networks (ANNs). This representation is time-rich and naturally aligns with neuromorphic hardware (Section 4.2), allowing for sparse, asynchronous, and energy-efficient computation.^{25,45}

One of the main problems in SNNs is the design of an efficient learning algorithm; such algorithms struggle to be directly applied to conventional backpropagation because spike events are non-differentiable. The first works were biologically inspired local learning rules, notably STDP²⁷, which adjust synaptic weights based on the relative timing of pre-synaptic and post-synaptic spikes. Although very scalable to shallow architectures, STDP is not as suitable for unsupervised feature extraction with high hardware compatibility as it is for deep architectures.^{27,31} While biologically realistic models and current machine learning methods have been connected in the past via gradient-based optimization, this concept was later extended to spiking networks by Bohte *et al.*⁶⁰, who showed how error back-propagation can be adapted for use with temporally encoded spiking networks.

Surrogate gradient methods, based on a smooth surrogate, are used in the backpropagation process for SNNs to optimize the network using gradient-based methods when the spike activation function is non-differentiable.⁶¹ The energy efficiency (on neuromorphic hardware⁵⁸) and the fact that surrogate gradient learning has been a powerful method to improve the accuracy of SNNs on benchmark tasks with performance near ANN-level have motivated the development of this method. However, the use of a surrogate gradient function brings its own problems (sensitivity to the specific choice of surrogate function, convergence instability in deep architectures, and lower accuracy than conventional ANNs on more complex tasks), which remain open questions and active areas of research.^{58,61} SNNs utilize spike timing, spike rate, or spike phase to encode information, with no computation occurring in between spikes and thus exhibiting naturally sparse activity²⁵, which can directly lead to lower energy use on neuromorphic hardware. A few promising proof-of-concept implementations of SNNs, such as IBM TrueNorth and Intel Loihi, have demonstrated that they can be a viable alternative to digital processors and are significantly more energy efficient.^{30,31}

5.4. Oscillatory and phase-based algorithms

Oscillatory and phase-based computing paradigms are a class of physics-inspired paradigms in which computation arises from the collective dynamics of coupled nonlinear oscillators. In contrast to traditional neural networks, which store information in the weighted sum and nonlinear activation of a set of static neurons, these networks store information in the phase, frequency, or synchronization state of the oscillatory elements, and use the synchronization, phase locking, and energy minimization of these elements to solve optimization and inference problems.⁵² While Section 4.3 discusses circuit-level implementations of oscillator networks, this section focuses on the algorithmic principles and computational mappings.

The most notable use of oscillatory computing has been for Ising machines for solving combinatorial optimization problems. Oscillator phases code for binary variables, and the intensity of the interactions between variables is coded as coupling strengths.⁵² The dynamics of the system are based on phase synchronization to minimize the Ising energy function of **Equation 3**, which helps to solve non-deterministic polynomial-time (NP) hard problems such as graph partitioning, traveling salesman, and constraint satisfaction efficiently.⁵² The computation in the phase-based approach is based on the relative differences in phases of information and not absolute amplitude. Coupled oscillator systems can display phase transitions, frequency entrainment, and collective synchronization, which can be utilized to achieve powerful information processing even in the presence of noise and variability.^{52,54} While SNNs can be used to represent temporal dynamics, hybrid solutions, involving the combination of SNNs and oscillatory dynamics, have been developed to enable richer temporal representations, while oscillator networks coupled with probabilistic devices can be used for sampling and inference in probabilistic computing.⁶²

5.5. Probabilistic and stochastic inference

Probabilistic and stochastic inference is not merely a way of dealing with uncertainty or randomness as sources of error, but as a source of computational resources. Probabilistic models, unlike deterministic neural networks, which always output the same result for a given input, explicitly model uncertainty and allow reasoning with incomplete or noisy information, which is crucial for applications such as Bayesian inference, generative modeling, and reasoning under uncertainty.⁶³

Stochastic computing is a way of representing numerical values as a stochastic bitstream, where each bit encodes a probability. A value $x \in [0, 1]$ is represented by

a series of bits with a probability of getting a logic 1 of x . Minimal logic circuits are then used to execute arithmetic operations: Multiplication is carried out by multiplying the two inputs x_1 and x_2 with a single AND gate, and scaled addition uses a multiplexer, which greatly simplifies the hardware implementation.⁵² New devices that naturally produce random fluctuations enable hardware-based sampling for MCMC methods and Bayesian inference.^{52,62}

Probabilistic inference can be conveniently expressed in terms of energy models, in which the probability of a state of the system is:

$$P(s) = \frac{1}{Z} \exp(-E(s)) \quad (5)$$

The partition function is a function Z that is used to calculate the energy function $E(s)$. This formulation is very similar to the Ising model in **Equation 3**, thereby establishing a direct link between stochastic inference and physics-based optimization.⁶³ This framework is exploited by BMs to learn and perform inferences by sampling from the energy distribution; hardware implementations of the BMs can take advantage of the intrinsic stochasticity of the device, allowing this sampling to be performed efficiently, with much reduced computational overheads as compared to digital implementations.⁵² Thermally activated switching spintronic devices enable tunable probability distributions for stochastic neurons and memristive devices designed to operate in the probabilistic switching regime, enabling stochastic learning and inference.^{57,62}

A summary of the algorithm-hardware co-design strategies discussed in this section is provided in **Table 3**, which indicates the extent of hardware properties exploited by the algorithmic technique, the representative hardware platform, and the device basis for each paradigm.

6. System-level applications

The D-C-A innovations described in Sections 3–5 collectively enable a new generation of intelligent hardware systems with capabilities that extend well beyond conventional digital accelerators. This section examines how physics-driven AI platforms translate into tangible system-level impact across five representative application domains: edge AI and wearable devices, autonomous vehicles and drones, healthcare monitoring, robotics, and space electronics.

6.1. Edge artificial intelligence and wearable devices

Edge AI is the deployment of intelligent inference and learning on resource-constrained devices that are not dependent on cloud connectivity. Some applications of this include keyword detection in smart speakers,

Table 3. Summary of algorithm–hardware co-design strategies across physics-driven AI paradigms

Co-design dimension	Precision-aware training	Spiking neural networks	Oscillatory and phase-based	Probabilistic and stochastic	References
Device basis	Memristive and PCM analog conductance	Neuromorphic CMOS LIF circuits	Spintronic and CMOS oscillators	Stochastic MTJ, memristive noise	7, 31, 47, 58
Primary algorithm technique	QAT, noise-aware training, device-aware backprop	STDP, surrogate gradient learning	Ising energy minimization, phase encoding	MCMC sampling, Boltzmann inference	27,47,59,63
Energy efficiency mechanism	Eliminated data movement via IMC	Temporal sparsity, event-driven processing	Physics-native convergence	Simple logic gates replace arithmetic units	7,25,47,48
Key challenge	Accuracy loss under device variability	Training convergence, scaling to deep SNNs	Synchronization stability at scale	Precision vs. convergence speed trade-off	47–49,56

Abbreviations: AI: Artificial intelligence; CMOS: Complementary metal-oxide-semiconductor; IMC: In-memory computing; LIF: Leaky integrate-and-fire; MCMC: Markov Chain Monte Carlo; MTJ: Magnetic tunnel junction; PCM: Phase-change memory; QAT: Quantization-aware training; SNN: Spiking neural network; STDP: Spike-timing-dependent plasticity.

gesture recognition in smart watches, anomaly detection in industrial sensors, and activity classification in fitness trackers. All these applications have similar requirements: low power consumption, real-time response, small size, and operation without a power cord, all powered by batteries.²³

Sensory inference workloads are power-efficient with neuromorphic processors such as IBM TrueNorth and Intel Loihi, which consume only milliwatts of power and process only when relevant input events occur. SNNs (Section 5.3) are naturally sparse and intermittent, similar to the nature of data streams from wearable sensors, thereby allowing energy consumption to be proportional to the amount of input activity, rather than a fixed, clock-driven overhead. Further efficiencies in inference energy come from analog in-memory computing circuits (Section 4.1) that execute multiply-accumulate operations within memory arrays, making it possible to achieve hundreds of TOPS/W.⁷ When it comes to analog devices, hardware-aware training techniques (Section 5.2) ensure that the models used on these low-precision devices are still accurate enough for deployment, despite the variability in the devices.¹⁶

6.2. Autonomous vehicles and drones

To make safety-critical decisions based on high-bandwidth streams of data from various types of sensors, such as light detection and ranging (LiDAR) point clouds, camera frames, and radar returns, autonomous vehicles and unmanned aerial vehicles require real-time processing under strict latency and power constraints.²³ In this field, photonic accelerators (Section 4.4) are especially suited since the optical interference can be used to perform

matrix multiplications at the speed of light with very low energy consumption per operation, which is necessary for the very low latency demanded for real-time navigation.^{14,19} WDM also allows multiple sensor streams to be processed in parallel on the same physical circuit. A natural uncertainty-estimation substrate for a perception system would be spintronic probabilistic computing (Section 5.5), which would allow vehicles to estimate the confidence of sensor interpretations and take conservative actions when they are not confident.

6.3. Healthcare monitoring

Healthcare monitoring involves continuous cardiac rhythm analysis, epileptic seizure detection, glucose level prediction, sleep staging, and the diagnosis of the initial stage of a disease from biosignal streams. These applications need to be sustained with low-power, real-time anomaly detection of the patient's physiology and on-device learning to adapt to each patient's physiology.⁸ Neuromorphic computing platforms are very suitable for healthcare monitoring due to the time-varying nature, sparsity, and event-driven nature of biosignals such as electroencephalography, electrocardiography, and electromyography. SNNs (Section 5.3) take these signals and detect clinically pertinent patterns while consuming a low amount of energy resources.^{31,45} Memristive and PCM synaptic devices can adaptively update synaptic weights in hardware, rather than via external server communication, to better suit individual patient data, and can be performed continuously through online learning^{7,12}. In clinical applications, one of the most interesting frameworks is probabilistic inference (Section 5.5), as it is important to quantify uncertainty to make a reliable diagnosis.⁶³

6.4. Robotics

The perception, planning, and actuation elements of robotic systems need to be tightly coupled with short time delays, low latencies, and high adaptability. Event-based vision sensors with neuromorphic processing have tremendous potential for revolutionizing robotic perception. Event cameras are not traditional frame-based cameras; instead, they emit a sparse stream of asynchronous events when pixel intensities change, greatly reducing data bandwidth.²⁶ Real-time optical flow estimation and object tracking can be done orders of magnitude more efficiently in terms of energy consumption using neuromorphic processors natively processing AER spike streams.^{30,31} Oscillatory neural networks (Section 5.4) are a natural framework to model coordinated rhythmic motor patterns similar to those found in biological central pattern generators for motor control.^{52,53} Stochastic policy gradient methods have been used in robotic control^{39,49}, including for spintronic devices that exhibit intrinsic stochastic dynamics.

6.5. Space electronics

Space computers are extremely power-constrained, exposed to high radiation levels (which can flip bits and degrade device performance), subject to strict thermal control, and are not physically maintainable once deployed.²³ Another benefit of PCM devices (Section 3.2) is that they are inherently radiation-hard, unlike SRAM-based digital memories, which depend on stored charge, and hence do not suffer from single-event upsets caused by high-energy particles hitting them.^{7,35} A power-efficient way to operate neuromorphic processors is to use them in event-driven mode, where they consume power only when processing relevant data, thereby exhibiting duty-cycled operation,

significantly reducing average power consumption, and being very well suited for Earth observation satellites that have to process hyperspectral image streams on board.^{30,31} Spintronic MRAM offers non-volatility, high endurance, and radiation tolerance, ensuring weight retention across temperature extremes and radiation doses in low Earth orbit (LEO) and geostationary orbits, respectively.³⁹

A summary of the mapping of application-domain requirements to physics-driven hardware platforms discussed in this review is given in Table 4. One theme that emerges in all five of these application domains is that the physical characteristics of emerging hardware platforms, such as event-driven sparsity, analog in-memory computation, stochastic dynamics, and optical parallelism, are natural matches to the computational requirements of real-world intelligent systems. This alignment drives the design philosophy of the D-C-A, the focus of this review.

7. Challenges, open problems, and future directions

Despite rapid advances in physics-based AI, several challenges remain in implementing these new paradigms into a scalable, reliable, and widely deployable system. There are problems at every level—device physics, algorithms, and system integration—and there is still a need for innovation and cross-disciplinary innovation.

7.1. Device-level challenges: variability, reliability, and scalability

With the advent of new technologies such as memristors, PCM, and spintronic devices, there is inherent variability in switching mechanisms across devices due to stochastic switching, fabrication defects, and material variations.

Table 4. Mapping of system-level application requirements to physics-driven AI hardware platforms

Application domain	Key requirements	Primary challenge	Suited hardware platform	References
Edge AI and wearables	Sub-mW power, real-time inference, small footprint	Always-on operation with minimal energy	Neuromorphic processors, analog IMC arrays	7,30,31
Autonomous vehicles and drones	Ultra-low latency, high bandwidth, real-time perception	Processing dense sensor streams within ms windows	Photonic accelerators, hybrid analog-digital IMC	7,14,19
Healthcare monitoring	Low power, on-device learning, uncertainty quantification	Adapting to individual patient physiology	Neuromorphic SNNs, memristive and PCM synapses, probabilistic circuits	7,31,63
Robotics	Real-time sensorimotor processing, adaptive learning	Event-driven perception and motor control	Neuromorphic event cameras, oscillatory networks, spintronic devices	31,43,47
Space electronics	Radiation hardness, thermal tolerance, self-calibration	Device degradation under radiation exposure	PCM-based IMC, spintronic MRAM, neuromorphic processors	7,30,33

Abbreviations: AI: Artificial intelligence; IMC: In-memory computing; MRAM: Magnetoresistive random-access memory; mW: milliwatt; PCM: Phase-change memory; SNN: Spiking neural network.

Such non-idealities include conductance drift, cycle-to-cycle variations, endurance limitations, and retention degradation, all of which affect computational accuracy and long-term reliability.

At scale, other effects such as IR drop, sneak-path currents, and thermal cross-talk further impact performance in dense arrays. These effects can be exploited in stochastic computation applications, but are a serious limitation for deterministic and precision applications. However, a key challenge for practical deployment is to obtain characteristics that are reliable, reproducible, and manufacturable.

7.2. Circuit-level constraints and energy bottlenecks

Analog and mixed-signal circuits are subject to inherent precision limitations at the circuit level due to noise, quantization, and non-ideal analog circuit components. Peripheral circuits, such as ADCs and DACs, can consume significant system energy, canceling out the efficiency of in-memory/analog computation.

Moreover, parasitic effects will compromise signal integrity during the scaling up of crossbar arrays, and the stability of oscillatory and neuromorphic circuits becomes increasingly complicated as they scale up. These restrictions make it clear that circuit architecture is needed that combines accuracy, low power consumption, and scalability.

7.3. Algorithmic robustness and learning paradigms

The algorithms used in classical digital hardware cannot be directly transferred to physics-based hardware because of imprecision, stochasticity, and nonlinear device properties. Hardware-aware training and quantization techniques have shown some promise, but consistent performance across different hardware substrates remains an open question.

Moreover, new issues are arising in training and optimization due to paradigms such as SNNs, stochastic inference, and oscillatory computing, such as non-differentiability, convergence instability, and sensitivity to noise. There is a research direction toward developing learning frameworks intrinsically tied to the physics of the hardware, rather than working around it.

7.4. System integration and cross-layer complexity

One of the biggest problems is integrating different computing paradigms, such as analog in-memory computing, digital logic, neuromorphic processors, and photonic interconnects. Efficiently coordinating the activities of these elements is essential, since a large portion of the energy consumed in the system is often devoted to

moving data and communicating among components.

Efficient workload partitioning, scheduling, and interface design are necessary to balance the computation among the different domains. Additionally, the lack of common design methodologies, interactions, or interfaces with other systems in a large-scale integration reduces the scalability of existing solutions.

7.5. Benchmarking, standardization, and evaluation

One of the biggest challenges to advancing the field is the lack of common benchmarking initiatives for physics-driven AI systems. However, it is difficult to make performance comparisons because of differences in databases, levels of accuracy, and evaluation methods. Measures of robustness, adaptability, and reliability under real-world conditions must be included alongside traditional metrics, e.g., throughput and energy efficiency. To facilitate fair comparisons and accelerate technology adoption, it is necessary to establish unified evaluation frameworks.

7.6. Software-hardware interface and programmability

A lack of well-developed programming models and software tool chains is another major challenge. Whereas in traditional computing platforms, there are well-defined software abstractions, programming physics-driven hardware requires consideration of device and circuit behavior. The fundamental challenge of bridging high-level machine learning frameworks and low-level hardware implementations remains. Compiler, simulation, and hardware-aware programming abstractions will play a key role in enabling these systems to be accessible to a wider research and engineering community.

7.7. Emerging directions and research opportunities

Amidst these difficulties, there are also some encouraging steps forward that are poised to impact the future of physics-based AI:

- Hybrid computing paradigms: Several of the aforementioned computing paradigms (analog, digital, neuromorphic, photonic) will be combined into a single architecture in the future, and the best of each domain will be used to achieve better trade-offs between accuracy, speed, and energy.
- Physics-native learning algorithms: There is growing interest in developing algorithms that directly exploit device physics rather than merely compensating for non-idealities, including oscillatory optimization, stochastic sampling-based learning, and analog gradient computation.

- Self-adaptive and self-calibrating systems: AI hardware could also become self-adaptive and self-calibrating, dynamically adjusting to device variations, aging, and environmental factors to ensure long-term reliability and robustness.
- AI for hardware co-design: Automated exploration of complex, high-dimensional design spaces can be achieved by optimizing hardware components, circuit parameters, and system architectures using machine learning techniques.
- Edge AI and autonomous systems: Physics-driven AI is well-suited for edge applications, especially for low latency and energy efficiency, enabling real-time, on-device intelligence to monitor healthcare, robots, and autonomous vehicles.
- Beyond Von Neumann computing: Finally, physics-driven AI is a step toward fundamentally new computing paradigms that overcome the limitations of traditional Von Neumann architectures by offering tightly integrated memory, computation, and communication.

Physics-driven AI represents a paradigm shift in computing, promising routes toward energy-efficient, scalable, and adaptable intelligent systems. There are still significant challenges at the device, circuit, algorithm, and system integration levels, but cross-layer co-design, new materials, and novel computational approaches are making rapid strides to close the gap between theoretical potential and practical fruition. As these technologies evolve and mature, they are poised to revolutionize the building blocks of computing and enable the development of more efficient, resilient, adaptive, and physically inspired AI systems.

8. Conclusion

This paper provides an extensive overview of physics-inspired AI, focusing on moving from traditional digital computing architectures to those that leverage physical phenomena directly for computation. It began by explaining the Von Neumann architecture and the memory wall paradigm, then focused on emerging paradigms for fundamentally new approaches to energy-efficient and scalable AI, such as in-memory computing, neuromorphic systems, oscillatory and stochastic computing, and photonic architectures.

Technologies such as memristive, spintronic, and phase-change devices enable the integration of memory and computation at the device level and form the foundation of a physics-native approach to computing. New circuit and architectural implementations, such as crossbar arrays, neuromorphic cores, and oscillatory networks, illustrate

how one can exploit physical dynamics to perform efficient computation. In terms of the algorithmic approach, methods such as hardware-aware training, SNNs, stochastic inference, and oscillatory optimization demonstrate the need for algorithmic and hardware co-design.

An interesting idea that recurs throughout this work is co-optimization: devices, circuits, architectures, and algorithms are designed together to optimize the system as a whole. The limitations of traditional computing and the full exploitation of emerging hardware capabilities can only be overcome through this holistic approach.

Although there has been considerable progress, issues remain in device reliability, circuit scalability, algorithm robustness, system integration, and standardization. The challenges will need to be addressed through ongoing interdisciplinary work across materials science, device engineering, circuit design, computer architecture, and machine learning.

Physics-driven AI is a transformative path for future computing systems overall. It goes beyond abstraction layers and exploits the inherent characteristics of physical systems to create innovative, energy-efficient, adaptive, and scalable intelligent machines. These paradigms are poised to be pivotal in the development of future AI technologies as research progresses.

Acknowledgments

None.

Funding

None.

Conflict of interest

Sonal Shreya is an Editorial Board Member of this journal, but was not in any way involved in the editorial and peer-review process conducted for this paper, directly or indirectly. The authors declare they have no competing interests.

Author contributions

Conceptualization: All authors

Writing—original draft: All authors

Writing—review & editing: All authors

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data

Not applicable.

References

1. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7533):436-444.
doi: 10.1038/nature14539
2. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst*. 2012;25:1097-1105.
3. Mnih V, Kavukcuoglu K, Silver D, *et al*. Human-level control through deep reinforcement learning. *Nature*. 2015;518(7540):529-533.
doi: 10.1038/nature14236
4. Jouppi NP, Young C, Patil N, *et al*. In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture. 2017:1-12.
doi: 10.1145/3079856.3080246
5. Horowitz M. Computing's energy problem (and what we can do about it). *IEEE Int Solid-State Circuits Conf Dig Tech Pap*. 2014:10-14.
doi: 10.1109/ISSCC.2014.6757323
6. Shalf J, Leland R. Computing beyond Moore's law. *Computer*. 2015;48(12):14-23.
doi: 10.1109/MC.2015.374
7. Sebastian A, Le Gallo M, Khaddam-Aljameh R, Eleftheriou E. Memory devices and applications for in-memory computing. *Nat Nanotechnol*. 2020;15(7):529-544.
doi: 10.1038/s41565-020-0655-z
8. Indiveri G, Liu SC. Memory and information processing in neuromorphic systems. *Proc IEEE*. 2015;103(8):1379-1397.
doi: 10.1109/JPROC.2015.2444094
9. Burr GW, Shelby RM, Sebastian A, *et al*. Neuromorphic computing and engineering. *Adv Phys X*. 2017;2(1):89-124.
doi: 10.1080/23746149.2016.1259585
10. Moitra A, Bhattacharjee A, Li Y, Kim Y, Panda P. When in-memory computing meets spiking neural networks—A perspective on device-circuit-system-and-algorithm co-design. *Appl Phys Rev*. 2024;11(3):031325.
doi: 10.1063/5.0211040
11. Li C, Hu M, Li Y, *et al*. Analogue signal and image processing with large memristor crossbars. *Nat Electron*. 2018;1:52-59.
doi: 10.1038/s41928-017-0002-z
12. Prezioso M, Merrih-Bayat F, Hoskins BD, Adam GC, Likharev KK, Strukov DB. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*. 2015;521(7550):61-64.
doi: 10.1038/nature14441
13. Ambrogio S, Narayanan P, Tsai H, *et al*. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*. 2018;558(7708):60-67.
doi: 10.1038/s41586-018-0180-5
14. Shen Y, Harris NC, Skirlo S, *et al*. Deep learning with coherent nanophotonic circuits. *Nat Photonics*. 2017;11(7):441-446.
doi: 10.1038/nphoton.2017.93
15. Wang Z, Joshi S, Savel'ev SE, *et al*. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nat Mater*. 2017;16:101-108.
doi: 10.1038/nmat4756
16. Rasch MJ, Mackin C, Le Gallo M, *et al*. Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators. *Nat Commun*. 2023;14:5282.
doi: 10.1038/s41467-023-40770-4
17. Wang Z, Wu H, Burr GW, *et al*. Resistive switching materials for information processing. *Nat Rev Mater*. 2020;5(3):173-195.
doi: 10.1038/s41578-019-0159-3
18. Yu S. Neuro-inspired computing with emerging nonvolatile memories. *Proc IEEE*. 2018;106(2):260-285.
doi: 10.1109/JPROC.2018.2790840
19. Feldmann J, Youngblood N, Karpov M, *et al*. Parallel convolutional processing using an integrated photonic tensor core. *Nature*. 2021;589(7840):52-58.
doi: 10.1038/s41586-020-03070-1
20. Wulf WA, McKee SA. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH Comput Archit News*. 1995;23(1):20-24.
doi: 10.1145/216585.216588
21. Sze V, Chen YH, Yang TJ, Emer JS. Efficient processing of deep neural networks: A tutorial and survey. *Proc IEEE*. 2017;105(12):2295-2329.
doi: 10.1109/JPROC.2017.2761740
22. Jouppi NP, Yoon DH, Ashcraft M, *et al*. Ten lessons from three generations shaped Google's TPUs. In: Proceedings of the 2021 ACM/IEEE 48th Annu Int Symp Comput Archit (ISCA). 2021:1-14.
doi: 10.1109/ISCA52012.2021.00010
23. Chen YH, Krishna T, Emer JS, Sze V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J Solid-State Circuits*. 2017;52(1):127-138.

- doi: 10.1109/JSSC.2016.2616357
24. Mead C. Neuromorphic electronic systems. *Proc IEEE*. 1990;78(10):1629-1636.
doi: 10.1109/5.58356
25. Maass W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw*. 1997;10(9):1659-1671.
doi: 10.1016/S0893-6080(97)00011-7
26. Boahen KA. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans Circuits Syst II Analog Digit Signal Process*. 2000;47(5):416-434.
doi: 10.1109/82.842110
27. Diehl PU, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front Comput Neurosci*. 2015;9:99.
doi: 10.3389/fncom.2015.00099
28. Lapicque L. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J Physiol Pathol Gén*. 1907;9:620-634.
29. McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys*. 1943;5(4):115-133.
doi: 10.1007/BF02478259
30. Merolla PA, Arthur JV, Alvarez-Icaza R, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*. 2014;345(6197):668-673.
doi: 10.1126/science.1254642
31. Davies M, Srinivasa N, Lin TH, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*. 2018;38(1):82-99.
doi: 10.1109/MM.2018.112130359
32. Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. *Nature*. 2008;453(7191):80-83.
doi: 10.1038/nature06932
33. Tuma T, Pantazi A, Le Gallo M, Sebastian A, Eleftheriou E. Stochastic phase-change neurons. *Nat Nanotechnol*. 2016;11(8):693-699.
doi: 10.1038/nnano.2016.70
34. Kent AD, Worledge DC. A new spin on magnetic memories. *Nat Nanotechnol*. 2015;10(3):187-191.
doi: 10.1038/nnano.2015.24
35. Locatelli N, Cros V, Grollier J. Spin-torque building blocks. *Nat Mater*. 2014;13(1):11-20.
doi: 10.1038/nmat3823
36. Grollier J, Querlioz D, Camsari KY, Everschor-Sitte K, Fukami S, Stiles MD. Neuromorphic spintronics. *Nat Electron*. 2020;3(7):360-370.
doi: 10.1038/s41928-019-0360-9
37. Torrejon J, Riou M, Araujo FA, et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*. 2017;547(7664):428-431.
doi: 10.1038/nature23011
38. Liu L, Pai CF, Li Y, Tseng HW, Ralph DC, Buhrman RA. Spin-torque switching with the giant spin Hall effect of tantalum. *Science*. 2012;336(6081):555-558.
doi: 10.1126/science.1218197
39. Aadit NA, Grimaldi A, Carpentieri M, et al. Massively parallel probabilistic computing with sparse Ising machines. *Nat Electron*. 2022;5(7):460-468.
doi: 10.1038/s41928-022-00774-2
40. Brunner D, Soriano MC, Mirasso CR, Fischer I. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nat Commun*. 2013;4:1364.
doi: 10.1038/ncomms2368
41. Chua LO. Memristor — the missing circuit element. *IEEE Trans Circuit Theory*. 1971;18(5):507-519.
doi: 10.1109/TCT.1971.1083337
42. Soliman T, Chatterjee S, Laleni N, et al. First demonstration of in-memory computing crossbar using multi-level Cell FeFET. *Nat Commun*. 2023;14:6348.
doi: 10.1038/s41467-023-42110-y
43. Feldmann J, Youngblood N, Wright CD, Bhaskaran H, Pernice WHP. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature*. 2019;569(7755):208-214.
doi: 10.1038/s41586-019-1157-8
44. Le Gallo M, Nandakumar SR, Ciric L, et al. Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars. *Neuromorph Comput Eng*. 2022;2(1):014009.
doi: 10.1088/2634-4386/ac4fb7
45. Gerstner W, Kistler WM, Naud R, Paninski L. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press; 2014.
doi: 10.1017/CBO9781107447615
46. Indiveri G, Linares-Barranco B, Hamilton TJ, et al. Neuromorphic silicon neuron circuits. *Front Neurosci*. 2011;5:73.
doi: 10.3389/fnins.2011.00073
47. Csaba G, Porod W. Coupled oscillators for computing: A review and perspective. *Appl Phys Rev*. 2020;7(1):011302.

- doi: 10.1063/1.5120412
48. Xu X, Tan M, Corcoran B, *et al.* 11 TOPS photonic convolutional accelerator for optical neural networks. *Nature*. 2021;589(7840):44-51.
doi: 10.1038/s41586-020-03063-0
 49. Borders WA, Pervaiz AZ, Fukami S, Camsari KY, Ohno H, Datta S. Integer factorization using stochastic magnetic tunnel junctions. *Nature*. 2019;573(7774):390-393.
doi: 10.1038/s41586-019-1557-9
 50. Shastri BJ, Tait AN, Ferreira de Lima T, *et al.* Photonics for artificial intelligence and neuromorphic computing. *Nat Photonics*. 2021;15(2):102-114.
doi: 10.1038/s41566-020-00754-y
 51. Le Gallo M, Khaddam-Aljameh R, Stanisavljevic M, *et al.* A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat Electron*. 2023;6(9):680-693.
doi: 10.1038/s41928-023-01010-1
 52. Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci USA*. 1982;79(8):2554-2558.
doi: 10.1073/pnas.79.8.2554
 53. Esser SK, Merolla PA, Arthur JV, *et al.* Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc Natl Acad Sci USA*. 2016;113(41):11441-11446.
doi: 10.1073/pnas.1604850113
 54. Wang T, Wu L, Nobel P, Roychowdhury J. Solving combinatorial optimisation problems using oscillator based Ising machines. *Nat Comput*. 2021;20(2):287-306.
doi: 10.1007/s11047-021-09845-3
 55. Zenke F, Ganguli S. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Comput*. 2018;30(6):1514-1541.
doi: 10.1162/neco_a_01086
 56. Le Gallo M, Sebastian A, Mathis R, *et al.* Mixed-precision in-memory computing. *Nat Electron*. 2018;1:246-253.
doi: 10.1038/s41928-018-0054-8
 57. Suri M, Bichler O, Querlioz D, *et al.* Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Trans Electron Devices*. 2013;60(7):2402-2409.
doi: 10.1109/TED.2013.2263000
 58. Neftci EO, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process Mag*. 2019;36(6):51-63.
doi: 10.1109/MSP.2019.2931595
 59. Jacob B, Kligys S, Chen B, *et al.* Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE; 2018:2704-2713.
doi: 10.1109/cvpr.2018.00286
 60. Bohte SM, Kok JN, La Poutre H. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*. 2002;48(1-4):17-37.
doi: 10.1016/S0925-2312(01)00658-0
 61. Alibart F, Zamanidoost E, Strukov DB. Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nat Commun*. 2013;4:2072.
doi: 10.1038/ncomms3072
 62. Gupta S, Agrawal A, Gopalakrishnan K, Narayanan P. Deep learning with limited numerical precision. In: *International Conference on Machine Learning*. 2015:1737-1746.
 63. Hinton GE, Sejnowski TJ. Learning and relearning in Boltzmann machines. In: Rumelhart DE, McClelland JL, eds. *Parallel Distributed Processing*, Vol 1. MIT Press; 1986:282-317.