

Neuromorphic Computing for Ultra-Low-Power Intelligent Systems: Architectures, Learning Paradigms, Applications, and Emerging Challenges

Naveen Kumar, Assistant Professor, Department of Electronics and Communication Engineering, Supaul College of Engineering, under the Department of Science, Technology and Technical Education, Bihar, India naveen6064@gmail.com

Dr. Achyuta Nand Mishra, Associate Professor, Department of Electronics and Communication Engineering, Supaul College of Engineering, under the Department of Science, Technology and Technical Education, Bihar, India an.mishra53@gmail.com

Ravi Ranjan, Assistant Professor, Department of Electronics and Communication Engineering, Supaul College of Engineering, under the Department of Science, Technology and Technical Education, Bihar, India ranjan92.ravi@gmail.com

Pavan Kumar Shukla, School of Engineering and Technology, CGC University Mohali, Punjab, India pavanfec@gmail.com

Abstract: The rapid expansion of artificial intelligence (AI) into edge devices, autonomous systems, intelligent sensors, robotics, healthcare platforms, and pervasive Internet-of-Things environments has exposed fundamental limitations in conventional computing architectures. Although contemporary deep neural networks achieve remarkable performance, their dependence on high-volume memory transfers, dense numerical operations, and centralized acceleration creates substantial energy, latency, and bandwidth costs. Neuromorphic computing offers an alternative paradigm in which computation is organized around sparse, asynchronous, event-driven information processing inspired by biological nervous systems. Spiking neural networks (SNNs), specialized memory and synaptic structures, event-driven communication, and tightly coupled sensing and computation constitute the principal elements of this emerging paradigm. This review critically examines neuromorphic computing as a foundation for ultra-low-power intelligent systems, with particular emphasis on architectures, learning paradigms, algorithms, applications, implementation technologies, and unresolved research challenges. The evolution from early neuromorphic circuits to contemporary large-scale platforms such as Loihi, Loihi 2, SpiNNaker 2, BrainScaleS-2, and event-driven sensing-computing systems is analyzed. SNN learning mechanisms, including spike-timing-dependent plasticity, surrogate-gradient optimization, ANN-to-SNN conversion, temporal coding, local learning, and hybrid learning, are compared from algorithmic and hardware perspectives. Emerging technologies involving memristive devices, spintronic elements, two-dimensional materials, analog computation, and in-memory processing are also examined. Particular attention is given to the relationship among spike sparsity, memory locality, communication overhead, accuracy, latency, and energy consumption. Applications in event-based vision, robotics, autonomous navigation, speech recognition, healthcare, industrial monitoring, smart agriculture, edge intelligence, and brain-machine interfaces are reviewed. The analysis identifies training complexity, benchmark inconsistency, hardware heterogeneity, limited software interoperability, device variability, scalability, security, interpretability, and sustainability as persistent barriers. The review concludes that practical progress will depend less on reproducing the brain literally and more on systematic algorithm-architecture-device co-design, standardized evaluation methodologies, adaptive learning, interoperable software abstractions, and application-specific neuromorphic system design.

Keywords: Neuromorphic computing, spiking neural networks, ultra-low-power AI, event-driven computing, edge intelligence, neuromorphic hardware, spike-based learning, memristors, in-memory computing, event-based vision, TinyML, brain-inspired computing.

1. Introduction

Artificial intelligence has progressively migrated from centralized cloud infrastructures toward edge devices that acquire, process, and act upon information in real time. Cameras, wearable sensors, autonomous robots, industrial controllers, environmental monitors, intelligent vehicles, medical devices, and Internet-of-Things nodes increasingly require local intelligence rather than continuous communication with remote servers. This transition creates a fundamental systems problem. A model may achieve excellent predictive accuracy while remaining unsuitable for deployment when its computational requirements exceed the energy, memory, thermal, or latency budgets of the target device. The growth of deep learning has therefore shifted attention from accuracy alone toward the joint optimization of computation, communication, memory movement, latency, and energy consumption.

Conventional von Neumann computing is poorly aligned with many of these requirements because computation and memory are physically separated. Repeated movement of model parameters and intermediate activations between memory and processing units can consume substantial energy, particularly when neural networks contain millions or billions of parameters. Graphics processing units and neural processing accelerators mitigate some computational limitations through massive parallelism, but dense matrix multiplication remains fundamentally expensive when the input stream contains substantial redundancy or when decisions must be made from sparse temporal events. The energy problem becomes especially pronounced in always-on applications, where the computational workload continues even when little useful information is present.

Neuromorphic computing approaches the problem from a different direction. Instead of treating computation as a sequence of dense arithmetic operations over synchronous numerical tensors, neuromorphic systems represent and process information through events, commonly encoded as spikes. A neuron integrates incoming signals over time and generates an output event when its internal state crosses a threshold. Consequently, computation can be performed only when relevant events occur. The resulting combination of sparse communication, distributed memory, asynchronous execution, and temporal information processing provides an attractive basis for ultra-low-power intelligence [1].

The modern field extends beyond simply implementing biological neurons in silicon. Contemporary neuromorphic computing encompasses digital many-core architectures, analog and mixed-signal circuits, event-based sensors, memristive devices, spintronic systems, reservoir computers, in-memory computing, and hybrid architectures capable of supporting both conventional artificial neural networks and SNNs. The evolution of the field is consequently characterized by increasing algorithm–hardware interaction. Neuromorphic architectures are no longer merely execution platforms for predefined neural models; hardware constraints increasingly influence network topology, learning rules, numerical precision, temporal resolution, and even the representation of information.

A central motivation is the observation that biological nervous systems perform complex perception and control under exceptionally constrained energy budgets. This observation does not imply that biological computation can be reproduced directly with present-day semiconductor technologies. Rather, biological principles provide design abstractions—sparsity, locality, asynchronous communication, adaptation, temporal coding, hierarchical processing, and co-location of memory and computation—that may be useful for engineering. Roy *et al.* emphasized that spike-based computation represents a transition from conventional dense neural processing toward event-driven machine intelligence [1]. More recent work has reinforced the importance of scaling these concepts beyond laboratory prototypes and toward complete computational ecosystems.

The current research landscape is therefore best understood as a convergence of several disciplines. Computational neuroscience provides neuron and synapse models; machine learning provides optimization techniques; computer architecture provides scalable execution mechanisms; semiconductor engineering provides energy-efficient circuits; emerging materials provide nonvolatile and analog device functionality; and application domains define the latency, accuracy, and energy constraints that ultimately determine practical usefulness.

This review examines that convergence systematically. It distinguishes neuromorphic computing from conventional accelerator-based AI, develops a taxonomy of architectures and learning paradigms, examines mathematical and algorithmic foundations, evaluates representative hardware platforms, compares implementation technologies, and analyzes application-level evidence. Particular attention is given to the distinction between theoretical energy efficiency and complete system-level efficiency, because an SNN that reduces arithmetic operations may still lose its advantage through excessive spike traffic, memory access, encoding overhead, or host-device communication.

The review subsequently examines implementation tools and benchmark methodologies, followed by a detailed case study illustrating an event-driven intelligent edge system. The final sections address unresolved challenges, ethical and societal implications, and research directions likely to shape neuromorphic computing between 2026 and the following decade.

2. Background and Motivation

2.1 From conventional computing to brain-inspired computation

The computational structure of modern AI has historically evolved around deterministic digital arithmetic. Classical processors execute instructions sequentially or exploit instruction-level parallelism, whereas GPUs and tensor accelerators execute large collections of arithmetic operations simultaneously. Although this architecture is extremely effective for dense workloads, neural intelligence often exhibits properties that are not naturally represented by dense computation. Sensory streams are temporally sparse, many neural states change only occasionally, and useful information may be concentrated around transitions rather than static values.

Neuromorphic engineering attempts to exploit these properties. A neuromorphic system typically combines a neuron model, synaptic state, event-routing mechanism, and learning or adaptation mechanism. Instead of repeatedly evaluating every neuron at every discrete time step, an event-driven architecture can update only those computational elements affected by incoming spikes. This difference becomes increasingly important when activity sparsity is high.

The significance of event-driven computation can be expressed through a simplified computational model. Consider a conventional layer requiring $[C_{\mathrm{ANN}} \approx T N_{\mathrm{in}} N_{\mathrm{out}}]$ multiply-accumulate operations for (T) temporal steps. An SNN may instead require approximately $[C_{\mathrm{SNN}} \approx \rho T N_{\mathrm{in}} N_{\mathrm{out}}]$, where (ρ) represents the average spike activity ratio. When ($\rho \ll 1$), the number of active synaptic operations can be substantially reduced. However, this theoretical advantage only translates into energy savings when the hardware can exploit the sparsity efficiently. If the architecture executes dense operations despite sparse spikes, the expected benefit largely disappears.

This observation explains why neuromorphic computing cannot be evaluated solely from the neural-network algorithm. The interaction among encoding, neuron dynamics, synaptic operations, routing, memory organization, and physical implementation determines actual efficiency. The same SNN can therefore have radically different energy characteristics on a GPU, FPGA, digital neuromorphic processor, analog neuromorphic circuit, or memristive accelerator.

2.2 Temporal information as a computational resource

Another fundamental motivation arises from the temporal dimension. Conventional artificial neural networks often convert temporal signals into frames, windows, or sequences before processing them. This conversion can discard fine-grained timing information and introduce redundant computation. SNNs inherently represent information through sequences of discrete events, allowing spike timing, spike frequency, population activity, and temporal relationships to encode information.

This capability is particularly relevant to event-based cameras, auditory sensors, tactile systems, and asynchronous physiological signals. An event camera, for example, does not necessarily transmit a complete image at a fixed frame rate. Instead, individual pixels generate events when their measured intensity changes beyond a specified threshold. A neuromorphic processor can therefore respond to changes rather than repeatedly processing an entire image.

The resulting architecture is not merely an energy-efficient neural-network accelerator. It can be viewed as an alternative information-processing pipeline in which sensing, representation, computation, and decision-making are temporally aligned. Recent sensing-computing systems demonstrate the significance of this approach by integrating asynchronous sensing with spike-based computation rather than transferring large volumes of conventional sensor data to a separate processor [2].

2.3 Motivation for ultra-low-power intelligence

Ultra-low-power intelligence is increasingly important in systems where energy availability is limited or where heat dissipation is difficult. Battery-operated sensors, implantable medical devices, wearable systems, autonomous micro-robots, agricultural sensor nodes, and remote monitoring platforms cannot assume access to continuous mains power or high-performance computing resources.

The concept of TinyML has consequently emerged around machine learning under strict memory, computational, and energy constraints. The MLPerf Tiny benchmark was introduced to provide standardized measurements for accuracy, latency, and energy across representative ultra-low-power machine-learning workloads [3]. Although MLPerf Tiny is not exclusively a neuromorphic benchmark, its methodology is highly relevant because it emphasizes the importance of measuring complete system behavior rather than reporting model accuracy in isolation.

Neuromorphic systems introduce additional dimensions into this evaluation. The number of synaptic operations, spike rate, temporal simulation length, memory accesses, routing events, and energy per event can all influence performance. Consequently, a meaningful comparison must report not only watts or operations per second but also energy per inference, latency-to-decision, accuracy, event rate, memory footprint, and workload characteristics.

2.4 Why neuromorphic computing remains challenging

The motivation for neuromorphic computing is compelling, but the field should not be interpreted as a straightforward replacement for GPUs or conventional AI accelerators. Deep learning has benefited from mature optimization algorithms, large datasets, standardized frameworks, high-performance libraries, and highly optimized hardware. SNNs do not yet possess an equivalent ecosystem.

Training remains particularly difficult because the spike-generation function is discontinuous. A threshold neuron can be expressed as $S(t) = H(V(t) - V_{\text{th}})$, where $H(\cdot)$ is the Heaviside function. The derivative of this function is zero almost everywhere and undefined at the threshold, which prevents straightforward application of conventional backpropagation. Surrogate-gradient methods address this problem by replacing the exact derivative with a smooth approximation during training [4].

This creates a fundamental distinction between biological plausibility and engineering practicality. Local learning rules such as spike-timing-dependent plasticity (STDP) are attractive because they can be implemented locally and asynchronously, but they may be less effective than global gradient-based optimization for complex classification tasks. Conversely, surrogate-gradient training can produce strong accuracy but may require substantial computational resources during training and may not map directly onto hardware-friendly learning mechanisms.

The field therefore faces a central systems question: **How can the advantages of sparse, temporal, event-driven computation be retained without sacrificing the programmability, accuracy, scalability, and development ecosystem of modern deep learning?** The remainder of this review examines this question from architectural, algorithmic, application, and technological perspectives.

3. Taxonomy / Classification

Neuromorphic computing is sufficiently broad that a single classification based only on hardware architecture is inadequate. A useful taxonomy must simultaneously consider physical implementation, computation model, learning mechanism, communication strategy, and application scale. Table 1 summarizes the principal categories.

Table 1. Taxonomy of neuromorphic computing systems

Classification dimension	Major categories	Principal characteristics
Computational representation	Spike-based, rate-based, analog state, reservoir state	Determines how information is represented
Hardware realization	Digital, analog, mixed-signal	Determines precision, programmability, speed, and variability
Memory organization	SRAM, local register state, eNVM, memristive crossbar	Determines data movement and synaptic storage
Communication	Synchronous, asynchronous, packet-based, multicast	Determines routing overhead and scalability
Learning	STDP, reinforcement, surrogate-gradient, ANN-to-SNN, local plasticity, hybrid	Determines adaptability and training requirements
Architecture	Many-core, processor-centric, crossbar, reservoir, SoC	Determines mapping and workload suitability
Sensor integration	Conventional sensors, event cameras, silicon cochleae, tactile sensors	Determines input representation
Deployment	Cloud, edge server, embedded edge, wearable, implantable	Determines power and latency constraints
Temporal processing	Rate coding, temporal coding, latency coding, population coding	Determines information density and spike activity

3.1 Digital neuromorphic systems

Digital neuromorphic systems implement neuron and synapse dynamics using conventional semiconductor logic while reorganizing the architecture around event-driven communication. Intel Loihi and Loihi 2, IBM TrueNorth, and SpiNNaker are representative examples. Digital implementations offer high programmability, reproducibility, and compatibility with established semiconductor processes. Their major limitation is that digital circuits can require more transistors and switching energy than highly specialized analog circuits for some neuron and synapse operations.

Loihi introduced programmable spiking neurons and on-chip learning mechanisms, while Loihi 2 extended the flexibility of neuron models and improved the ability to represent more complex temporal dynamics [5]. Loihi 2 experiments have demonstrated that specialized neuron dynamics can reduce computational and communication requirements for streaming signal-processing tasks [5].

SpiNNaker follows a different architectural philosophy. Instead of constructing a dedicated accelerator for every neural primitive, it uses many general-purpose processor cores connected through a specialized communication fabric. SpiNNaker 2 extends this approach through heterogeneous processing, near-threshold operation, dynamic voltage and frequency scaling, and dedicated neural acceleration [6].

3.2 Analog and mixed-signal systems

Analog neuromorphic systems exploit physical circuit dynamics to represent neuron membrane potentials, synaptic currents, adaptation, and other state variables. Because the computation is represented directly in electrical quantities, analog implementations can be highly efficient for certain operations. Their disadvantages include process variation, temperature sensitivity, limited precision, calibration requirements, and greater difficulty in reproducing arbitrary computational functions.

BrainScaleS-2 represents a significant mixed-signal approach. It combines accelerated analog neuron and synapse circuits with digital control and event routing. Importantly, it supports configurable neuron dynamics and hybrid plasticity mechanisms, enabling experiments that are difficult to perform on fixed-function accelerators [7].

The classification of analog neuromorphic computing should therefore distinguish between **analog emulation**, where biological dynamics are approximated physically, and **analog in-memory computing**, where physical device properties are used to perform neural arithmetic. These approaches overlap but have different design objectives.

3.3 Emerging device-based neuromorphic systems

A third category operates closer to the device level. Memristors, resistive switching devices, phase-change materials, spintronic elements, ferroelectric devices, two-dimensional materials, and organic or perovskite devices have all been investigated as artificial synapses and neurons. Such devices can combine non-volatile storage with analog conductance modulation, reducing the distance between memory and computation.

Memristive systems are particularly attractive because a crossbar can theoretically perform vector-matrix multiplication according to Kirchhoff's laws. If input voltages (V_i) are applied to rows and conductance (G_{ij}) represent synaptic weights, the column current can be approximated by $I_j = \sum_i G_{ij} V_i$. This physical computation can reduce explicit data movement. However, device nonlinearity, conductance drift, limited endurance, write variability, sneak currents, analog-to-digital conversion overhead, and programming precision remain significant engineering constraints.

Recent research has also explored 2D materials and spintronic devices as candidates for dense, energy-efficient neuromorphic systems [8], [9]. These technologies may ultimately enable computing paradigms that are fundamentally different from transistor-only architectures.

3.4 Learning-based taxonomy

From a learning perspective, neuromorphic systems can be divided into offline-trained, online-adaptive, and hybrid systems. Offline-trained networks are optimized on conventional computing infrastructure and subsequently deployed on neuromorphic hardware. This is currently the most practical pathway for many applications.

Online systems modify their synaptic weights after deployment. Such adaptation is valuable when environmental statistics change, as in autonomous robotics or personalized healthcare. STDP and related local learning mechanisms are particularly suitable because synaptic updates can depend only on pre- and postsynaptic activity. Hybrid systems combine global optimization with local adaptation. For example, a network may be pretrained using surrogate gradients and subsequently employ local plasticity to adapt to changing environments. This approach attempts to combine the representational power of deep learning with the adaptability of biological learning.

3.5 Application-oriented taxonomy

Neuromorphic workloads can also be classified according to their temporal characteristics. Static image classification is a relatively weak use case because conventional CNN accelerators are highly optimized for this task. In contrast, event-based perception, always-on sensing, auditory processing, gesture recognition, anomaly detection, adaptive control, and closed-loop robotics naturally generate sparse temporal workloads.

This distinction is essential. Neuromorphic computing should not be evaluated according to whether it can outperform conventional hardware on every machine-learning benchmark. Its strongest value proposition lies in workloads where temporal sparsity, asynchronous events, continuous adaptation, and strict energy constraints coexist.

4. Core Concepts and Methodologies

4.1 Spiking neuron models

The spiking neuron is the fundamental computational primitive of most neuromorphic systems. The leaky integrate-and-fire (LIF) model provides a useful compromise between biological interpretability and hardware simplicity. Its membrane potential can be represented as $\tau_m \frac{dV(t)}{dt} = -V(t) + V_{\text{rest}} + R_m I(t)$, where $V(t)$ is membrane potential, V_{rest} is resting potential, τ_m is membrane time constant, R_m is membrane resistance, and $I(t)$ represents synaptic input. When $V(t) \geq V_{\text{th}}$, the neuron emits a spike and the membrane potential is reset.

The LIF model is attractive because integration, thresholding, and reset can be implemented efficiently. More sophisticated models such as adaptive LIF, exponential integrate-and-fire, resonate-and-fire, and adaptive exponential integrate-and-fire introduce additional state variables. These mechanisms enable richer temporal computation but increase implementation complexity.

The importance of stateful neuron dynamics has received renewed attention with programmable neuromorphic processors. Loihi 2, for example, supports a wider range of neuron dynamics than earlier architectures, enabling computation to be performed by temporal filters rather than by static activation functions [5]. This expands the role of the neuron from a simple binary threshold element into a programmable dynamical system.

4.2 Synaptic models and plasticity

Synapses determine how neural activity influences downstream neurons. A simple current-based synapse can be expressed as $\sum_k w_k s_k(t)$, where w_k is synaptic weight and $s_k(t)$ is the presynaptic spike train.

Plasticity modifies w_k according to neural activity. STDP is a classical temporal learning mechanism in which weight changes depend on the relative timing of pre- and postsynaptic spikes $[\Delta w = \begin{cases} A_+ \exp(-\Delta t / \tau_+) & \Delta t > 0 \\ -A_- \exp(\Delta t / \tau_-) & \Delta t < 0 \end{cases}]$

The biological appeal of STDP lies in its locality. The synapse can update using information available at or near the synaptic site. This makes STDP attractive for hardware implementations with limited memory and communication bandwidth.

However, STDP alone does not automatically produce the hierarchical feature learning observed in modern deep networks. Its performance depends strongly on input representation, network structure, homeostatic mechanisms, and task formulation. Consequently, recent research increasingly considers hybrid plasticity, reward-modulated STDP, supervised local rules, and gradient-assisted local learning.

4.3 Spike encoding

Input encoding strongly influences SNN performance. Common approaches include rate coding, temporal coding, latency coding, population coding, and direct event representation.

Rate coding represents a numerical value through spike frequency. It is relatively robust but may require many spikes and therefore longer inference windows. Temporal coding conveys information through precise spike timing and can reduce the number of events required. Latency coding maps input magnitude to the timing of an individual spike, potentially reducing computation but increasing sensitivity to temporal noise.

For event-based sensors, direct event representation is often preferable because the sensor already generates asynchronous events. Transforming such events into dense frames may eliminate the primary advantage of event-driven sensing.

4.4 Surrogate-gradient learning

The central difficulty in gradient-based SNN training is the non-differentiability of spike generation. Surrogate-gradient methods replace the zero or undefined derivative with a smooth approximation $[\frac{\partial S}{\partial V}] \approx \sigma'(V - V_{\text{th}})$, where σ' may be a sigmoid derivative, piecewise-linear function, fast-sigmoid approximation, or another surrogate.

During the forward pass, the actual spike function remains discrete. During the backward pass, a differentiable surrogate is used. This seemingly simple modification has transformed SNN training by enabling the use of backpropagation with deep architectures.

Its principal limitation is that the optimization procedure is not itself a biological learning mechanism. Furthermore, the training process may require storing membrane states across many time steps, generating substantial memory traffic. Consequently, a model can be computationally efficient during inference while being expensive to train.

4.5 ANN-to-SNN conversion

Another major methodology converts pretrained artificial neural networks into SNNs. The conventional ANN is trained using standard backpropagation and then transformed so that activations are represented through spike rates or temporal events.

Conversion is attractive because it separates optimization from neuromorphic deployment. However, conversion often requires sufficiently long simulation windows to approximate continuous activations accurately. Excessive spike rates can also reduce the energy advantage. More recent conversion strategies attempt to minimize firing rates, reduce temporal steps, and directly optimize hardware-relevant parameters.

4.6 Reservoir computing

Reservoir computing represents another important methodology. Instead of training all synaptic connections, a recurrent dynamical system generates a rich temporal representation, while only a readout layer is trained. Physical reservoir computing uses material dynamics directly as computational resources [10].

The attraction of reservoir computing is its compatibility with nonlinear dynamical substrates. The reservoir need not be explicitly programmed to perform every transformation. Instead, its natural dynamics provide a high-dimensional temporal state. The major challenge is controlling reservoir stability, reproducibility, and trainability.

4.7 Hardware-aware learning

A growing methodological trend is to include hardware constraints during network optimization. Instead of minimizing only classification loss, $\mathcal{L}_{\text{task}}$, the optimization objective can incorporate energy, latency, spike count, memory, or hardware utilization $\mathcal{L}_{\text{task}} + \lambda_E E + \lambda_S S + \lambda_L L + \lambda_M M$.

Here (E), (S), (L), and (M) represent energy, spike activity, latency, and memory cost, while the (λ) coefficients determine their relative importance.

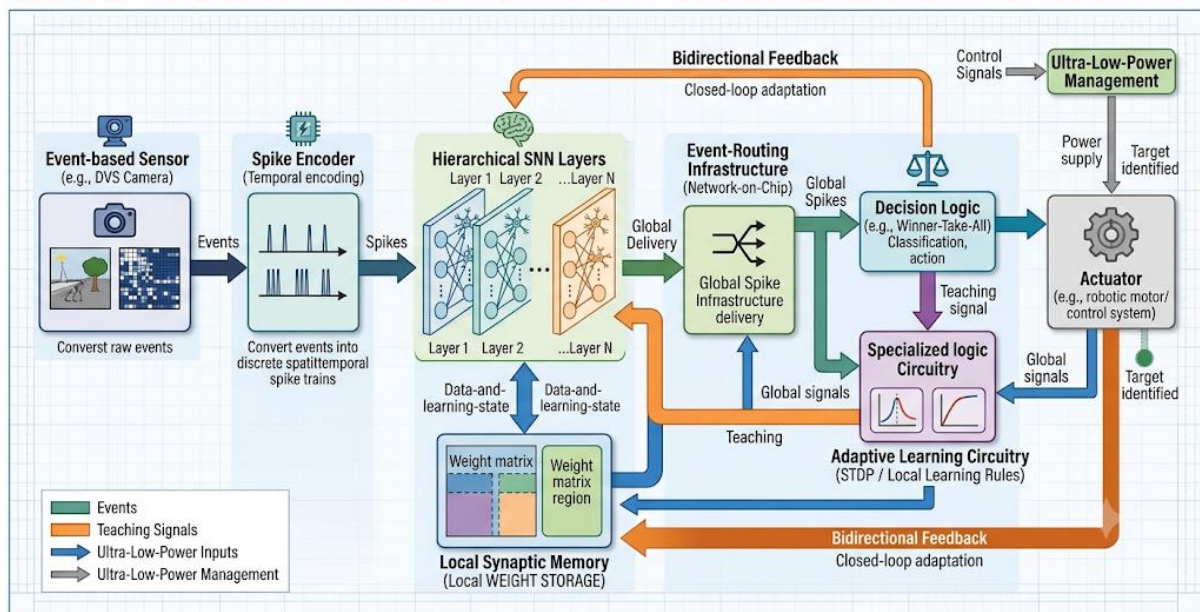
This formulation reflects a major conceptual transition. Rather than designing an SNN first and attempting to map it to hardware afterward, the hardware becomes part of the optimization problem itself.

5. System Architecture / Algorithms

5.1 General neuromorphic system architecture

A practical neuromorphic intelligent system can be represented as a pipeline consisting of sensing, event encoding, neuromorphic computation, learning/adaptation, decision-making, and actuation. Unlike conventional pipelines, the boundaries between these components may be blurred.

FIGURE 1: GENERAL ARCHITECTURE OF AN ULTRA-LOW-POWER NEUROMORPHIC INTELLIGENT SYSTEM



The first stage acquires environmental signals. Event cameras, microphones, inertial sensors, tactile arrays, biosensors, and conventional sensors can provide input. When conventional sensors are used, an encoding stage converts measurements into spike streams.

The second stage performs hierarchical processing. Early layers typically detect local temporal or spatial features. Intermediate layers integrate information over longer time constants, while later layers perform classification, prediction, or decision-making.

The event-routing fabric is central to scalability. In packet-based systems, each spike can be represented by a destination address and transported through an asynchronous network-on-chip. Multicast mechanisms are particularly important because one presynaptic event may influence many postsynaptic neurons.

5.2 Many-core architectures

Many-core neuromorphic processors divide computation into small processing units. Each core stores local neuron and synaptic state and communicates through an interconnect. This approach minimizes global memory traffic and supports large networks.

The principal architectural problem is mapping. A network may contain more neurons than available local memory, requiring partitioning across cores. Excessive communication between partitions can eliminate energy savings. Therefore, graph partitioning, topology-aware placement, and routing optimization are critical.

SpiNNaker exemplifies a processor-centric many-core approach. SpiNNaker 2 retains general-purpose processing while adding accelerators and dynamic power-management mechanisms [6]. This design improves flexibility but may not achieve the energy efficiency of highly specialized digital or analog architectures for every workload.

5.3 Crossbar-based architectures

Crossbar architectures exploit device physics to perform weighted summation. If conductance represents synaptic weight, analog currents naturally implement accumulation. This arrangement reduces explicit memory access because weights reside where computation occurs.

The principal challenge is precision. Deep neural networks increasingly benefit from low-precision arithmetic, but analog crossbars introduce device-dependent errors that can accumulate across layers. Calibration, differential encoding, redundancy, and algorithmic noise tolerance are therefore essential.

Emerging memristive devices can support multiple synaptic mechanisms in compact physical structures. Recent experimental work has demonstrated devices capable of reproducing several synaptic behaviors while targeting energy-efficient neural computation [11].

5.4 Hybrid neuromorphic architectures

Hybrid architectures combine conventional deep-learning computation with spike-based components. Tianjic demonstrated this philosophy by integrating artificial neural-network and brain-inspired computation within a common many-core architecture [12]. The significance of this approach lies in architectural flexibility. Real applications may not benefit from converting every computational layer into spikes.

A hybrid system can assign dense static computation to conventional accelerators while using neuromorphic components for event-driven perception, temporal filtering, or adaptive control. This approach may be particularly valuable in edge systems where a heterogeneous SoC can dynamically select computational modes.

5.5 Algorithms for event-driven perception

Event-driven vision commonly uses convolutional SNNs, recurrent SNNs, spatiotemporal graph structures, and hybrid architectures. Unlike conventional image processing, event streams contain tuples such as $e_i = (x_i, y_i, t_i, p_i)$, where (x_i) and (y_i) denote spatial coordinates, (t_i) denotes timestamp, and (p_i) represents polarity.

A network can process these events directly. A spatiotemporal feature may then be computed using a temporal kernel $[\sum_i w_i K(x - x_i, y - y_i, t - t_i)]$. Such processing avoids constructing a dense frame and can exploit temporal locality.

5.6 Recurrent and adaptive algorithms

Recurrent SNNs are particularly suitable for temporal prediction, speech, gesture recognition, and control. Their internal state provides memory without requiring an explicit external sequence buffer.

However, long temporal dependencies introduce training difficulties. Backpropagation through time can consume substantial memory because intermediate states must be retained. Eligibility propagation and related local-gradient approaches seek to reduce this burden.

Experiments using eligibility propagation on SpiNNaker 2 demonstrate the possibility of online learning with reduced memory requirements relative to conventional backpropagation through time [13]. Such approaches are important for edge devices that must learn continuously without storing complete training sequences.

5.7 Compiler and intermediate representation architecture

Hardware heterogeneity creates another major problem. A model trained using one software framework may require extensive rewriting before deployment on another neuromorphic processor.

The Neuromorphic Intermediate Representation (NIR) addresses this problem by providing a hardware-independent representation for neuromorphic computational graphs. Experimental demonstrations have shown mappings across multiple neuromorphic platforms, including digital and mixed-signal systems [14]. This represents an important step toward a compiler ecosystem analogous to ONNX in conventional machine learning.

6. Comparative Analysis

Neuromorphic computing must be compared against conventional CPU, GPU, FPGA, and AI-accelerator architectures using workload-specific criteria rather than generalized claims of superior efficiency.

Table 2. Comparison of computing paradigms

Property	CPU	GPU	FPGA	AI accelerator	Neuromorphic processor
General programmability	Very high	High	High	Moderate	Moderate-high
Dense matrix operations	Moderate	Excellent	High	Excellent	Low-moderate
Sparse event processing	Moderate	Moderate	High	Moderate	Excellent
Temporal computation	Software-based	Software-based	Configurable	Limited	Native
Asynchronous operation	Limited	Limited	Possible	Limited	Native
On-chip learning	Rare	Rare	Possible	Limited	Strong potential
Energy efficiency for dense AI	Moderate	Moderate	High	Very high	Workload-dependent
Energy efficiency for sparse temporal workloads	Low-moderate	Moderate	High	High	Potentially very high
Software maturity	Excellent	Excellent	High	High	Emerging
Hardware ecosystem	Mature	Mature	Mature	Growing	Fragmented
Best suited workloads	General computing	Large-scale AI	Custom pipelines	Dense AI inference	Event-driven temporal intelligence

The most important comparison concerns energy rather than raw throughput. A GPU can process enormous numbers of operations per second, but this throughput is unnecessary when the workload consists of infrequent sensor events. Conversely, a neuromorphic processor may perform poorly when required to emulate dense matrix operations with little temporal sparsity.

Table 3. Representative neuromorphic platforms

Platform	Architecture	Primary strength	Principal limitation	Typical research focus
IBM TrueNorth	Digital many-core	Large-scale event-driven processing	Limited programmability and learning flexibility	Low-power inference
Intel Loihi	Digital many-core	On-chip learning and programmable SNNs	Research-oriented ecosystem	Adaptive intelligence
Intel Loihi 2	Advanced digital	Flexible neuron dynamics	Limited commercial availability	Temporal processing
SpiNNaker	Processor-centric	Flexible large-scale simulation	Processor energy overhead	Brain-scale simulation
SpiNNaker 2	Hybrid many-core	Dynamic power control and acceleration	Mapping complexity	Edge learning
BrainScaleS-2	Mixed-signal	Fast physical neuron emulation	Analog variability	Neuroscience and adaptive computing
Tianjic	Hybrid	ANN/SNN integration	Specialized ecosystem	Multimodal intelligence
Speck	Sensing-computing SoC	Integrated event sensing and processing	Application-specific scope	Edge vision

Memristive crossbar	In-memory	Compute-memory integration	Device variability	Analog neural acceleration
---------------------	-----------	----------------------------	--------------------	----------------------------

The relationship can be represented conceptually as $[E_{\text{compute}} + E_{\text{memory}} + E_{\text{communication}} + E_{\text{sensor}} + E_{\text{control}}]$.

Neuromorphic architectures primarily target reductions in the first three terms through sparse computation, local state, and event-driven communication. However, these advantages can be offset if event routing becomes excessive or if the sensor itself consumes substantial energy.

Loihi and related platforms have produced evidence that neuromorphic architectures can provide meaningful gains for selected workloads rather than merely theoretical benefits [15]. However, published measurements frequently differ in what they include. Some report core energy only, whereas others include memory, communication, sensors, preprocessing, and host processors. Such differences can make direct comparisons misleading.

The problem has been recognized in the broader neuromorphic community, which increasingly emphasizes system-level evaluation and scaling. Large-scale neuromorphic research identifies architecture, programming models, communication, benchmarks, and application ecosystems as interconnected requirements rather than independent components [16].

Table 4. Learning paradigm comparison

Learning paradigm	Training cost	Biological plausibility	Online adaptation	Hardware suitability	Main limitation
STDP	Low-moderate	High	Excellent	Excellent	Task-dependent accuracy
Reward-modulated STDP	Moderate	High	Excellent	High	Reward design
ANN-to-SNN conversion	Low deployment cost	Low	Limited	High	Conversion loss
Surrogate-gradient	High	Moderate	Moderate	High inference suitability	Training memory
BPTT	High	Low	Low	Moderate	Temporal memory cost
Eligibility propagation	Moderate	Higher	High	High	Optimization complexity
Reservoir computing	Low	Moderate-high	High	High	Reservoir design
Hybrid learning	Moderate-high	Moderate	High	High	Increased system complexity

The comparison indicates that no learning paradigm simultaneously optimizes biological plausibility, accuracy, training simplicity, online adaptability, and hardware efficiency. The likely long-term solution is therefore heterogeneous learning rather than a universally dominant algorithm.

7. Applications

7.1 Event-based vision

Event-based vision is arguably the most natural application domain for neuromorphic computing. Conventional cameras generate complete frames at predetermined intervals, even when most pixels have not changed. Event cameras instead generate asynchronous events associated with changes in luminance.

This representation is naturally sparse and temporal. Neuromorphic processors can therefore operate directly on events without reconstructing complete images. Applications include object detection, optical flow, gesture recognition, tracking, depth estimation, and autonomous navigation.

Loihi 2 studies have demonstrated the usefulness of programmable temporal neurons for signal-processing tasks such as optical flow, illustrating that specialized temporal dynamics can replace computationally expensive conventional pipelines [5].

7.2 Autonomous robotics

Robotics requires perception and control under strict latency constraints. A robot must continuously observe its environment, update its internal state, and generate motor commands. The event-driven nature of SNNs is therefore compatible with closed-loop control.

Neuromorphic control architectures can process sensor events while simultaneously updating recurrent neural states. Such systems may reduce latency because decisions do not necessarily wait for complete frames.

However, robotics introduces safety requirements absent from many benchmark datasets. A small classification error may be acceptable in an academic benchmark but unacceptable when it results in collision or unsafe actuation. Consequently, robustness, uncertainty estimation, and fail-safe mechanisms are essential.

7.3 Speech and auditory processing

Speech is intrinsically temporal. Conventional speech-recognition systems typically convert audio into spectrograms or feature sequences. Neuromorphic auditory systems can instead represent acoustic events using temporal spikes.

Resonate-and-fire neurons are particularly attractive because their temporal dynamics resemble band-pass filtering. Loihi 2 experiments demonstrated that resonant neuron models can implement signal-processing functions with substantially reduced output event bandwidth compared with conventional approaches [5].

7.4 Healthcare and biomedical monitoring

Neuromorphic computing can support continuous monitoring of physiological signals such as electrocardiograms, electromyograms, electroencephalograms, and inertial measurements. These signals often contain sparse diagnostic events embedded in long periods of relatively stable activity. An always-on neuromorphic detector could therefore remain in a low-power state and activate higher-level processing only when an abnormal temporal pattern is detected.

Implantable applications impose particularly stringent requirements. Energy dissipation, thermal effects, reliability, electromagnetic compatibility, and long-term device stability must be considered. In these environments, energy efficiency is not merely a performance metric; it can directly affect system safety and battery life.

7.5 Smart agriculture

Smart agriculture provides a representative ultra-low-power deployment environment. Distributed sensor nodes may monitor soil moisture, temperature, humidity, crop conditions, acoustic signals, and images. Sending raw measurements continuously to a cloud server consumes communication energy and introduces latency.

A neuromorphic edge node could perform local anomaly detection and transmit only meaningful events. For example, a node could learn the normal temporal pattern of soil moisture and generate a spike when measurements deviate significantly. This would reduce communication and enable local irrigation decisions.

The main challenge is that agricultural environments are highly variable. Models must adapt to seasonal changes, sensor aging, crop type, and environmental conditions. Online learning and robust calibration are therefore important.

7.6 Industrial monitoring

Industrial systems generate continuous streams from vibration sensors, acoustic sensors, motors, pumps, and machinery. Faults often appear as transient temporal patterns rather than persistent changes.

Neuromorphic systems are well suited to this scenario because anomaly detection can be performed using temporal dynamics. A reservoir or recurrent SNN can learn the normal state and respond strongly when incoming activity deviates from the learned manifold.

7.7 Brain-computer interfaces

Brain-computer interfaces require processing neural signals with low latency and often under strict power constraints. Spiking architectures can provide a natural representation for neural events and can potentially process signals locally.

The key challenge is robustness. Neural signals vary across individuals and over time, requiring adaptation. Neuromorphic hardware with on-chip learning could potentially update models without transferring raw neural data to external processors.

7.8 Edge intelligence and TinyML

The intersection between neuromorphic computing and TinyML is particularly significant. MLPerf Tiny demonstrated the importance of evaluating accuracy, latency, and energy jointly [3]. Neuromorphic systems can add another dimension: event sparsity.

The future edge-AI landscape is therefore likely to be heterogeneous. Some tasks will remain on conventional microcontrollers; others will use DSPs, NPUs, FPGAs, or GPUs; and event-driven workloads may increasingly be assigned to neuromorphic processors.

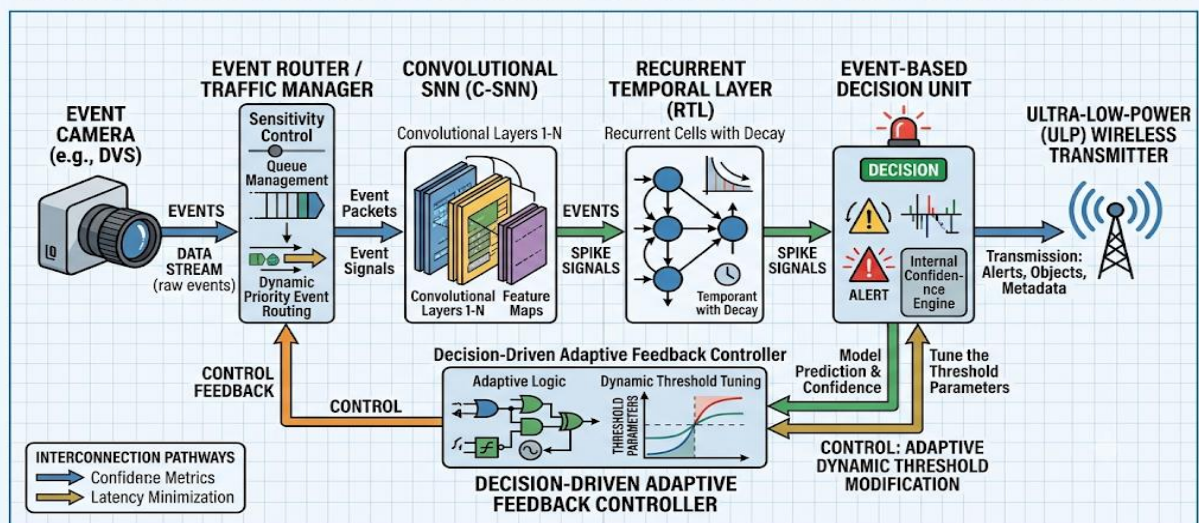
8. Case Study: Ultra-Low-Power Event-Based Vision for Autonomous Edge Monitoring

8.1 Problem formulation

Consider an autonomous edge device deployed for continuous visual monitoring in a battery-powered environment. The system must detect objects or abnormal movement while operating continuously for several months. A conventional camera-CPU pipeline captures frames at a fixed rate, performs preprocessing, executes a CNN, and transmits detections. The energy requirement can be approximated by $[N_f(E_{\text{capture}} + E_{\text{preprocess}} + E_{\text{inference}}) + E_{\text{communication}}]$, where N_f is the number of frames processed per day. A neuromorphic alternative uses an event camera and SNN $[N_e(E_{\text{event}} + E_{\text{SNN}}) + E_{\text{communication,event}}]$, where N_e represents the number of events rather than frames. The fundamental opportunity arises when N_e is sufficiently smaller than the equivalent dense pixel-processing workload.

8.2 Architecture

FIGURE 2: REFINED FEEDBACK-DRIVEN EVENT-BASED AUTONOMOUS MONITORING ARCHITECTURE



The system uses four computational stages. The first stage receives events directly from the sensor. The second stage extracts spatial features using convolutional spiking layers. The third stage integrates temporal context using recurrent neurons. The final stage generates a detection event.

8.3 Hardware mapping

The network can be trained using surrogate gradients on a conventional GPU. After training, the model is converted into a hardware-compatible representation and mapped onto a neuromorphic processor. NIR provides an emerging interoperability layer that can reduce dependence on platform-specific implementations [14].

The case study demonstrates a crucial design principle: the objective should not simply be to maximize classification accuracy. A better optimization objective combines accuracy with spike count, latency, and energy.

9. Tools, Technologies and Implementation

9.1 Software frameworks

The neuromorphic software ecosystem has expanded substantially. snnTorch provides SNN components within the PyTorch ecosystem and supports surrogate-gradient training. Norse provides SNN primitives integrated with PyTorch and targets deep-learning workflows [17]. SpikingJelly provides a broader infrastructure covering dataset processing, model development, optimization, simulation, and deployment [18].

Table 5. Major SNN and neuromorphic software ecosystems

Framework/tool	Main ecosystem	Principal use	Deployment orientation
snnTorch	PyTorch	SNN training and experimentation	Research/deployment
SpikingJelly	PyTorch	Full-stack SNN development	Research/deployment
Norse	PyTorch	SNN modeling and training	Research
Lava	Intel ecosystem	Neuromorphic application development	Loihi-oriented
NIR	Platform-independent	Model interoperability	Cross-platform
Brian2	Python	Computational neuroscience	Simulation
Nengo	Python	Neural modeling and cognitive systems	Simulation/hardware
Rockpool	Python	Neuromorphic hardware development	Xylo-oriented
Sinabs	PyTorch	SNN conversion/deployment	Edge neuromorphic
FPGA toolchains	HDL/HLS	Custom SNN accelerators	Embedded hardware

The emergence of NIR is particularly important because software fragmentation has historically limited neuromorphic adoption. A common intermediate representation can allow researchers to develop models without committing to a particular hardware platform at the beginning of the workflow [14].

9.2 Hardware technologies

Neuromorphic implementation technologies span conventional CMOS, analog CMOS, mixed-signal circuits, FPGAs, resistive memories, phase-change devices, spintronic devices, ferroelectric devices, and two-dimensional materials.

CMOS remains dominant because of its maturity and scalability. Emerging devices are attractive because they may combine memory and computation more naturally.

Spintronic neuromorphic systems are being investigated for synaptic and neuronal functions because magnetic dynamics can provide nonlinear, stochastic, and temporal behaviors [9]. Two-dimensional materials are attractive for vertically stacked architectures and high device density [8].

9.3 FPGA implementation

FPGAs occupy an important intermediate position between conventional processors and dedicated neuromorphic ASICs. They offer parallelism and reconfigurability without requiring fabrication of a custom chip.

A 2025 survey of SNN implementations on FPGAs identified parallel processing, low latency, and configurable architecture as important advantages while emphasizing challenges involving memory, implementation methodology, and network scalability [19].

The main disadvantage is that FPGA implementations often use conventional digital logic to emulate neuron dynamics. Consequently, the energy efficiency may not match that of purpose-built neuromorphic ASICs.

9.4 Neuromorphic sensors

Sensor–processor co-design is increasingly important. Event cameras, silicon cochleas, and tactile sensors can produce spike-like signals directly. This removes the need for expensive analog-to-digital conversion followed by dense preprocessing.

The most efficient architecture may therefore be one in which the sensor, encoder, processor, memory, and communication network are designed jointly. This approach can reduce the energy spent converting between incompatible representations.

10. Performance Evaluation

Performance evaluation in neuromorphic computing is complicated because traditional metrics are insufficient. Throughput, FLOPS, and average power do not fully describe event-driven computation.

Table 6. Recommended neuromorphic evaluation metrics

Metric	Definition	Importance
Accuracy	Task-specific correctness	Measures intelligence
Energy/inference	Total energy per decision	Measures efficiency
Power	Energy/time	Measures thermal and battery demand
Latency	Time to decision	Measures real-time capability
Spike rate	Events per neuron per time	Measures sparsity
SynOps	Number of synaptic events	Measures computational workload
Memory footprint	Model + state storage	Determines deployability
Communication energy	Routing and I/O energy	Important for distributed systems
Adaptation time	Time to learn new patterns	Measures online learning
Robustness	Performance under perturbation	Measures reliability

10.1 Benchmark datasets

Representative datasets include MNIST and Fashion-MNIST for basic classification; CIFAR-10 and CIFAR-100 for image recognition; N-MNIST for event-based digit recognition; DVS Gesture for event-based gesture classification; SHD for speech-related temporal classification; and Spiking Heidelberg Digits/Words-type datasets for auditory temporal learning.

The limitation of many traditional datasets is that they do not exercise the defining strengths of neuromorphic systems. Static image datasets can be converted into spikes, but this does not necessarily reflect real event-driven sensing. Consequently, event-native datasets should receive greater emphasis in future benchmarking.

10.7 Benchmarking methodology

A complete benchmark should report:

1. identical dataset and preprocessing;
2. identical accuracy target;

3. complete inference energy;
4. hardware configuration;
5. clock frequency;
6. precision;
7. batch size;
8. simulation time;
9. spike rate;
10. memory and communication overhead.

This requirement is consistent with the broader emphasis on standardized TinyML evaluation [3] and the field's increasing recognition that scaling neuromorphic systems requires reproducible system-level benchmarks [16].

11. Challenges and Limitations

Neuromorphic computing faces several challenges that limit widespread adoption. **Training complexity** remains significant because surrogate gradients, temporal credit assignment, and simulation steps increase computational and memory requirements. **Accuracy and task dependence** indicate that SNNs are not universally superior to DNNs; advantages are strongest for workloads exploiting temporal sparsity. **Hardware heterogeneity** across platforms such as Loihi, SpiNNaker, FPGAs, and memristive systems complicates portability and optimization. **Memory and communication bottlenecks** can arise from excessive spike traffic and connectivity. **Device variability** requires noise-aware algorithms, calibration, redundancy, and adaptive learning. **Benchmark fragmentation** makes energy and performance comparisons difficult, highlighting the need for standardized protocols. **Software maturity** remains behind conventional AI ecosystems despite progress from SpikingJelly, snnTorch, Norse, Lava, and NIR [17], [18]. **Scaling** introduces routing, synchronization, thermal, and manufacturing challenges [16]. Additionally, **security vulnerabilities** and **reproducibility issues** require stronger protection, standardized configurations, transparent measurements, and comprehensive reporting.

12. Future Research Directions

Future neuromorphic computing should emphasize **algorithm–hardware co-design**, integrating accuracy, energy, latency, memory, spike rate, and routing costs during training. **Adaptive and continual learning** can enable systems to adjust to changing environments, while self-calibrating architectures can compensate for device variability. Research into **neuromorphic foundation models, transformers, and state-space models** should explore event-driven attention and temporal representations. **3D integration** may combine sensing, memory, and processing, while **memristive and spintronic devices** offer opportunities for non-volatile and in-memory computation. **Standardized benchmarks** covering accuracy, latency, energy, memory, communication, and system-level efficiency are essential for meaningful comparisons. **Interoperable software frameworks** can improve portability across heterogeneous hardware. Future architectures may also combine **neuromorphic edge processing with cloud AI** to reduce communication overhead. Finally, **explainable, certifiable, secure, and sustainable SNNs** should address reliability, transparency, lifecycle energy consumption, and deployment in safety-critical applications.

14. Conclusion

Neuromorphic computing offers an alternative paradigm for intelligent information processing by integrating computation, memory, communication, and learning through sparse, event-driven activity. Its greatest potential lies in ultra-low-power applications involving continuous sensing, temporal information, low latency, and constrained energy budgets. Current platforms demonstrate diverse approaches, including digital, mixed-signal, hybrid, and emerging-device architectures. However, practical efficiency depends on complete system factors such as communication, memory access, encoding, and data movement. Future progress requires algorithm–hardware–device co-design, adaptive learning, standardized benchmarks, and interoperable software. Neuromorphic processors are unlikely to universally replace CPUs, GPUs, or NPU; instead, they can complement existing accelerators for specialized workloads. Ultimately, neuromorphic computing may expand the design space for heterogeneous, adaptive, distributed, and energy-efficient intelligent systems.

References

- [1] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, pp. 607–617, 2019.

- [2] Y. Liu et al., "Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip," *Nature Communications*, vol. 15, Art. no. 4464, 2024.
- [3] C. Banbury et al., "MLPerf Tiny Benchmark," *Proc. NeurIPS Datasets and Benchmarks*, 2021.
- [4] J. H. Lee and T. Delbruck, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, Art. no. 508, 2016.
- [5] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, "Efficient neuromorphic signal processing with Loihi 2," *arXiv:2111.03746*, 2021.
- [6] S. Höppner et al., "The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing," *arXiv:2103.08392*, 2021.
- [7] C. Pehle et al., "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity," *Frontiers in Neuroscience*, vol. 16, Art. no. 795876, 2022.
- [8] S. J. Kim, H.-J. Lee, C.-H. Lee, and H. W. Jang, "2D materials-based 3D integration for neuromorphic hardware," *npj 2D Materials and Applications*, vol. 8, Art. no. 70, 2024.
- [9] C. H. Marrows, J. Barker, T. A. Moore, and T. Moorsom, "Neuromorphic computing with spintronics," *npj Spintronics*, vol. 2, Art. no. 12, 2024.
- [10] X. Liang, J. Tang, Y. Zhong, B. Gao, and H. Qian, "Physical reservoir computing with emerging electronics," *Nature Electronics*, vol. 7, pp. 193–206, 2024.
- [11] C. Weilenmann et al., "Single neuromorphic memristor closely emulates multiple synaptic mechanisms for energy efficient neural networks," *Nature Communications*, 2024.
- [12] J. Pei et al., "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, pp. 106–111, 2019.
- [13] A. Rostami, B. Vogginger, Y. Yan, and C. G. Mayr, "E-prop on SpiNNaker 2: Exploring online learning in spiking RNNs on neuromorphic hardware," *Frontiers in Neuroscience*, vol. 16, Art. no. 1018006, 2022.
- [14] J. E. Pedersen et al., "Neuromorphic intermediate representation: A unified instruction set for interoperable brain-inspired computing," *Nature Communications*, vol. 15, Art. no. 8122, 2024.
- [15] M. Davies et al., "Advancing neuromorphic computing with Loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.
- [16] D. Kudithipudi et al., "Neuromorphic computing at scale," *Nature*, vol. 637, pp. 801–812, 2025.
- [17] B. C. Cramer et al., "Norse: A deep learning library for spiking neural networks," *software framework*, 2021–2025.
- [18] W. Fang et al., "SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence," *Science Advances*, vol. 9, 2023.
- [19] M. Karamimanes, E. Abiri, M. Shahsavari, K. Hassanli, A. van Schaik, and J. Eshraghian, "Spiking neural networks on FPGA: A survey of methodologies and recent advancements," *Neural Networks*, vol. 186, Art. no. 107256, 2025.
- [20] C. Ganguly et al., "Spike frequency adaptation: Bridging neural models and neuromorphic applications," *Communications Engineering*, vol. 3, Art. no. 22, 2024.
- [21] A. Gebregiorgis et al., "Spike-based neuromorphic computing: An overview from bio-inspiration to hardware architectures and learning mechanisms," *Microprocessors and Microsystems*, vol. 124, Art. no. 105240, 2026.
- [22] M. P. Ferreira, S. Wang, Y. Gao, and A. Benlarbi-Delai, "A comparative review of deep and spiking neural networks for edge AI neuromorphic circuits," *Frontiers in Neuroscience*, vol. 19, Art. no. 1676570, 2025.
- [23] H. He and S. Gao, "A four-stage structural evolution framework for spiking neural networks: A review and perspective from binary ANN to event-driven models," *Neural Processing Letters*, vol. 58, Art. no. 14, 2026.
- [24] O. Y. L. Imanov, D. U. Kulali, T. Yilmaz, D. Erisken, and R. I. Turhan, "Energy-efficient neuromorphic computing for edge AI: A framework with adaptive spiking neural networks and hardware-aware optimization," *arXiv:2602.02439*, 2026.
- [25] R. V. Wicaksana Putra, P. Wickramasinghe, and M. Shafique, "Enabling efficient processing of spiking neural networks with on-chip learning on commodity neuromorphic processors for edge AI systems," *arXiv:2504.00957*, 2025.
- [26] S. Wang et al., "Ternary spike-based neuromorphic signal processing system," *Neural Networks*, vol. 187, Art. no. 107333, 2025.
- [27] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers in Neuroscience*, vol. 12, Art. no. 774, 2018.
- [28] J. Eshraghian et al., "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, 2023.
- [29] T. Delbruck et al., "Activity-driven event-based vision and neuromorphic sensing," in *Proc. IEEE*, 2021.
- [30] G. Indiveri and T. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, 2015.
- [31] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, 2015.
- [32] F. Akopyan et al., "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.
- [33] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [34] S. Furber, "Large-scale neuromorphic computing systems," *Journal of Neural Engineering*, vol. 13, no. 5, Art. no. 051001, 2016.
- [35] J. Schemmel et al., "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE International Symposium on Circuits and Systems*, 2010.
- [36] G. Indiveri et al., "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, 2011.
- [37] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, 2017.
- [38] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks," *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018.
- [39] T. C. Pfeil et al., "Is a 300× reduction in synaptic operations sufficient for energy-efficient neuromorphic computing?" *Frontiers in Neuroscience*, 2022.
- [40] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018.
- [41] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Computation*, 2021.
- [42] G. Bellec et al., "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Communications*, vol. 11, 2020.
- [43] F. Zenke, S. M. Bohtë, and T. Delbruck, "The remarkable robustness of surrogate gradient learning," *Neural Computation*, 2021.

- [44] Y. Wu et al., "Direct training for spiking neural networks: Faster, larger, better," in Proc. AAAI Conference on Artificial Intelligence, 2019.
- [45] W. Fang et al., "Deep residual learning in spiking neural networks," Advances in Neural Information Processing Systems, 2021.
- [46] Q. Zhang et al., "Spiking transformers: A promising approach for energy-efficient intelligent processing," IEEE Transactions on Neural Networks and Learning Systems, 2023.
- [47] J. K. Eshraghian et al., "Attention meets spiking neural networks: A survey and perspective," IEEE Transactions on Neural Networks and Learning Systems, 2024.
- [48] C. Schuman et al., "Opportunities for neuromorphic computing algorithms and applications," Nature Computational Science, 2022.
- [49] G. Orchard, A. Jayawant, G. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets," Frontiers in Neuroscience, vol. 9, 2015.
- [50] A. Amir et al., "A low power, fully event-based gesture recognition system," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2017.
- [51] A. Sironi et al., "HATS: Histograms of averaged time surfaces for robust event-based object classification," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.
- [52] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 7, pp. 3227–3239, 2018.
- [53] Y. Chen et al., "Deep learning for event-based vision," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.
- [54] C. Pehle and J. Schemmel, "Neuromorphic computing systems and applications," in Handbook of Neuroengineering, 2021.
- [55] S. Dey et al., "A review of neuromorphic vision systems," Proceedings of the IEEE, 2023.
- [56] K. Roy et al., "Towards brain-inspired computing with spintronic and emerging devices," Nature Reviews Electrical Engineering, 2024.
- [57] C. H. Bennett et al., "The physical limits of computation," International Journal of Theoretical Physics, 1982.
- [58] M. Horowitz, "1.1 Computing's energy problem," in ISSCC Digest of Technical Papers, 2014.
- [59] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks," Synthesis Lectures on Computer Architecture, vol. 15, no. 2, 2020.
- [60] D. Kudithipudi et al., "Brain-inspired computing: Opportunities and challenges," Nature Electronics, 2021.
- [61] A. Das et al., "A survey on neuromorphic computing and spiking neural networks," ACM Computing Surveys, 2024.
- [62] A. Sebastian et al., "Memory devices and applications for neuromorphic computing," Nature Nanotechnology, 2020.
- [63] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," Nature Electronics, 2018.
- [64] Q. Wang et al., "Neuromorphic computing with memristive devices," Nature Reviews Materials, 2024.
- [65] K. Roy, "Brain-inspired computing with emerging technologies," IEEE Design & Test, 2022.
- [66] J. Schemmel et al., "BrainScaleS: A wafer-scale accelerated neuromorphic system," IEEE International Joint Conference on Neural Networks, 2012.
- [67] S. Furber et al., "The SpiNNaker project," Proceedings of the IEEE, 2014.
- [68] T. Wunderlich et al., "Demonstrating advantages of brain-inspired architectures with Loihi," IEEE Micro, 2019.
- [69] E. Stamatias et al., "Scalable energy-efficient neuromorphic systems," Frontiers in Neuroscience, 2021.
- [70] D. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," Neural Computation, 2007.
- [71] G. Bellec et al., "Long short-term memory and eligibility traces for spiking neural networks," Neural Computation, 2020.
- [72] A. N. Burkitt, "A review of the integrate-and-fire neuron model," Biological Cybernetics, vol. 95, pp. 1–19, 2006.
- [73] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" IEEE Transactions on Neural Networks, vol. 15, no. 5, pp. 1063–1070, 2004.
- [74] W. Maass, "Networks of spiking neurons: The third generation of neural network models," Neural Networks, vol. 10, no. 9, pp. 1659–1671, 1997.
- [75] C. Mead, "Neuromorphic electronic systems," Proceedings of the IEEE, vol. 78, no. 10, pp. 1629–1636, 1990.