eng

MDPI

Review

A Practical Tutorial on Spiking Neural Networks: Comprehensive Review, Models, Experiments, Software Tools, and Implementation Guidelines

Bahgat Ayasi [1,2,*], Cristóbal J. Carmona [3], Mohammed Saleh [4] and Angel M. García-Vico [3]

[1] Computer Science Department, University of Jaén, Campus Las Lagunillas s/n, Andalucía, 23071 Jaén, Spain
[2] Computer Science Department, Arab American University (AAUP), Jenin P.O. Box 240, Palestine
[3] Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Jaén, 23071 Jaén, Spain; ccarmona@ujaen.es (C.J.C.); agvico@ujaen.es (A.M.G.-V.)
[4] Information Technology Center, Al-Istiqlal University, Jericho P.O. Box 10, Palestine; msaleh@pass.ps
* Correspondence: ba000034@red.ujaen.es

Abstract

Spiking neural networks (SNNs) provide a biologically inspired, event-driven alternative to artificial neural networks (ANNs), potentially delivering competitive accuracy at substantially lower energy. This tutorial-study offers a unified, practice-oriented assessment that combines critical review and standardized experiments. We benchmark a shallow fully connected network (FCN) on MNIST and a deeper VGG7 architecture on CIFAR-10 across multiple neuron models (leaky integrate-and-fire (LIF), sigma–delta, etc.) and input encodings (direct, rate, temporal, etc.), using supervised surrogate-gradient training implemented in Intel Lava, SLAYER, SpikingJelly, Norse, and PyTorch. Empirically, we observe a consistent but tunable trade-off between accuracy and energy. On MNIST, sigma–delta neurons with rate or sigma–delta encodings achieve 98.1% accuracy (ANN baseline: 98.23%). On CIFAR-10, sigma–delta neurons with direct input reach 83.0% accuracy at just two time steps (ANN baseline: 83.6%). A GPU-based operation-count energy proxy indicates that many SNN configurations operate below the ANN energy baseline; some frugal codes minimize energy at the cost of accuracy, whereas accuracy-oriented settings (e.g., sigma–delta with direct or rate coding) narrow the performance gap while remaining energy-conscious—yielding up to threefold efficiency compared with matched ANNs in our setup. Thresholds and the number of time steps are decisive factors: intermediate thresholds and the minimal time window that still meets accuracy targets typically maximize efficiency per joule. We distill actionable design rules—choose the neuron–encoding pair according to the application goal (accuracy-critical vs. energy-constrained) and co-tune thresholds and time steps. Finally, we outline how event-driven neuromorphic hardware can amplify these savings through sparse, local, asynchronous computation, providing a practical playbook for embedded, real-time, and sustainable AI deployments.

Keywords: spiking neural networks; artificial neural networks; energy efficiency; supervised learning; implementation guidelines; software tools

check for updates

Academic Editor: Fei Teng

Received: 22 September 2025
Revised: 6 October 2025
Accepted: 24 October 2025
Published: 2 November 2025

Citation: Ayasi, B.; Carmona, C.J.; Saleh, M.; García-Vico, A.M. A Practical Tutorial on Spiking Neural Networks: Comprehensive Review, Models, Experiments, Software Tools, and Implementation Guidelines. *Eng* **2025**, *6*, 304. https://doi.org/10.3390/eng6110304

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Modern AI systems—particularly deep neural networks—have achieved remarkable accuracy across vision, language, and control tasks, but at rapidly growing computational and energy costs [1,2]. Mitigation strategies such as pruning and quantization can reduce

multiply-accumulate (MAC) counts and memory traffic [3–5]. Yet, the overall footprint of state-of-the-art models continues to raise sustainability concerns [6]. This tension motivates the exploration of approaches that are both accurate and power-aware. Recent studies demonstrate that conventional machine learning techniques are increasingly being applied to embedded and industrial tasks under tight resource and latency constraints [7–9], underscoring the need for more event-driven, energy-efficient paradigms.

Spiking neural networks (SNNs) offer a biologically inspired, event-driven paradigm in which discrete spikes convey information over time [10,11]. Their sparse, asynchronous computation and temporal coding can translate into lower energy consumption on neuromorphic substrates while naturally capturing temporal structure [12–15]. At the same time, practical deployment remains challenging; spikes are non-differentiable, complicating gradient-based optimization [16–18]; performance depends critically on the choice of encoding scheme [19,20]; and toolchains and benchmarks for fair SNN–ANN comparisons are still maturing.

Gap. The key gap lies in the lack of a unified, practice-oriented analysis. Existing surveys often treat neuron models, encodings, learning rules, and software stacks in isolation, providing limited *apples-to-apples* evidence on accuracy–energy trade-offs against equivalent ANN baselines across both shallow and deep regimes. A comprehensive perspective that links design choices to measurable performance and energy efficiency remains scarce [18–33].

This work. We address this gap by combining a comprehensive, critical review with a hands-on tutorial and standardized benchmarking:

- We systematize SNN *components*, as follows: neuron models (integrate-and-fire (IF)/leaky integrate-and-fire (LIF), adaptive leaky integrate-and-fire (ALIF), exponential integrate-and-fire/adaptive exponential integrate-and-fire (EIF/AdEx), resonate-and-fire (RF), Hodgkin–Huxley (HH), Izhikevich (IZH), resonate-and-fire–Izhikevich (RF–IZH) hybrid, current-based neuron (CUBA), sigma–delta (ΣΔ)); neural *encodings* (direct/single-value encoding, rate coding, temporal variants including time-to-first-spike (TTFS), rank-order with number-of-spikes (R–NoM), population coding, phase-of-firing coding (PoFC), burst coding, sigma–delta encoding (ΣΔ)); and *learning paradigms*: supervised (backpropagation-through-time (BPTT) with surrogate gradients, e.g., SLAYER, SuperSpike, EventProp), unsupervised (spike-timing–dependent plasticity (STDP) and variants), reinforcement (reward-modulated STDP (R-STDP), e-prop), hybrid supervised STDP (SSTDP), and ANN→SNN conversion. The practical pipeline used in this study relies on supervised training via BPTT with surrogate gradients (aTan by default; SLAYER/SuperSpike-style updates) for the tutorial and benchmark experiments [16–19].
- We provide a practical tutorial (with reference to a representative neuromorphic software stack, e.g., Intel Lava) covering model construction, encoding choices, and training–inference workflows suitable for resource-constrained deployment.
- We establish a side-by-side evaluation protocol that compares SNNs with architecturally matched ANNs on a *shallow* setting (MNIST) and a *deeper* convolutional setting (CIFAR-10 with VGG-style backbones [34]). Metrics include task accuracy, time steps, spike activity, and power-oriented proxies to highlight accuracy–efficiency trade-offs.
- We distill design guidelines that map application goals—accuracy targets and per-inference energy budgets—onto actionable choices of neuron model, encoding scheme, number of time steps, and supervised surrogate-gradient training (e.g., SLAYER, SuperSpike, aTan).

Scope and structure. Section 2 reviews the fundamentals of SNNs, including encoding strategies, neuron models, and learning paradigms. Section 3 presents the datasets, experi-

mental setup, model architectures, and software tools. Section 4 reports the comparative performance and energy analyses on MNIST and CIFAR-10 and provides an integrated discussion of accuracy–energy trade-offs and design implications. Finally, Section 5 summarizes the key findings and offers recommendations for future research. Our aim is to provide a coherent pathway from principles to practice, guiding the design of SNNs that balance accuracy with energy efficiency in real-world deployments. An overview of the article's logical flow and interconnections among its main sections is illustrated in Figure 1.

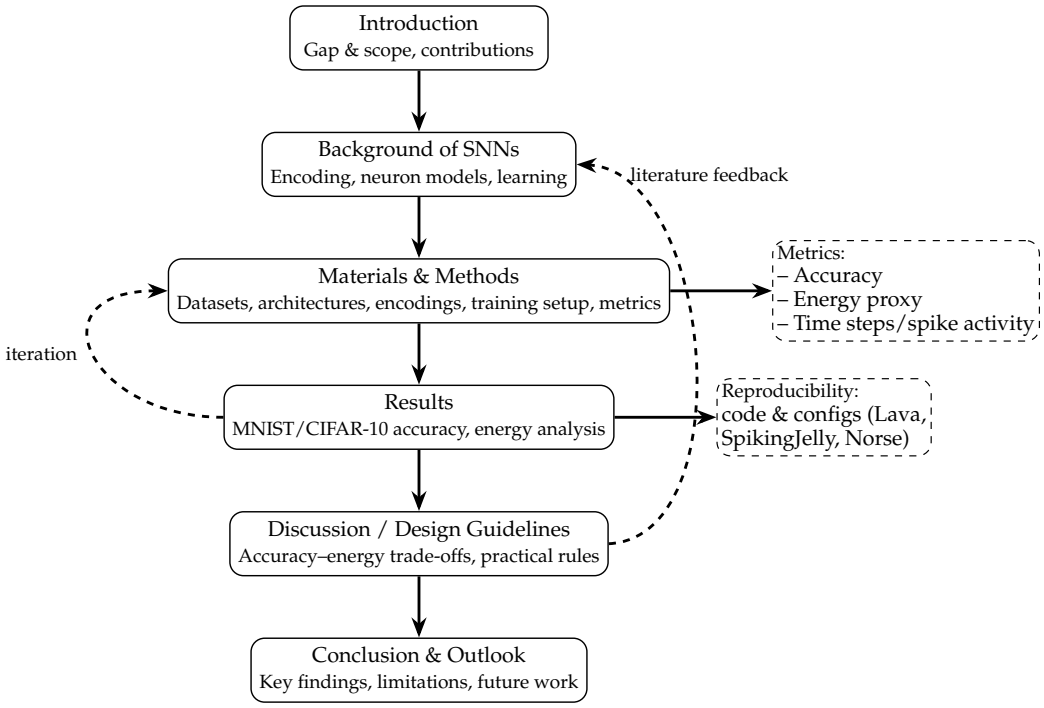


Figure 1. Organizational chart of the article. The flow illustrates how the *Background* informs *Materials and Methods*, leading to *Results*, analyzed in *Discussion*, and concluding with future perspectives. Dashed arrows show iterative refinement and reference feedback.

2. Background of Spiking Neural Networks

SNNs are widely regarded as the "third generation" of neural models, narrowing the gap between ANNs and biological computation by representing information through discrete spike events over time [10,35]. Whereas ANNs operate with continuous activations and synchronous MAC operations [36], SNNs exploit sparse, event-driven accumulation updates. This temporal, asynchronous processing aligns with neural physiology and can yield substantial energy savings—particularly on neuromorphic hardware—while natively handling time-dependent signals.

2.1. Key Aspects of Spiking Neural Networks

2.1.1. Processing Pipeline

Figure 2 outlines a generic SNN workflow comprising *encoding*, *network processing*, *decoding*, and *learning*. In the encoding stage, external signals are transformed into spike trains. Common strategies include rate codes, time-based codes (e.g., TTFS or inter-spike intervals), and population codes, selected according to signal statistics and latency–energy constraints. For instance, image intensities can be converted into Poisson spike trains whose rates are proportional to pixel values [37]. During network processing, spikes propagate through layers of model neurons—e.g., LIF, AdEx, IZH, or HH—whose subthreshold dynamics and thresholds determine temporal integration and spike generation [38,39]. Decoding then maps output spike activity to decisions using spike counts (rate-based), precise

timing (temporal), or pooled population activity, depending on the task requirements [37]. Learning rules—supervised, unsupervised, reinforcement, or hybrid—adjust synapses to meet specific behavioral or optimization goals.

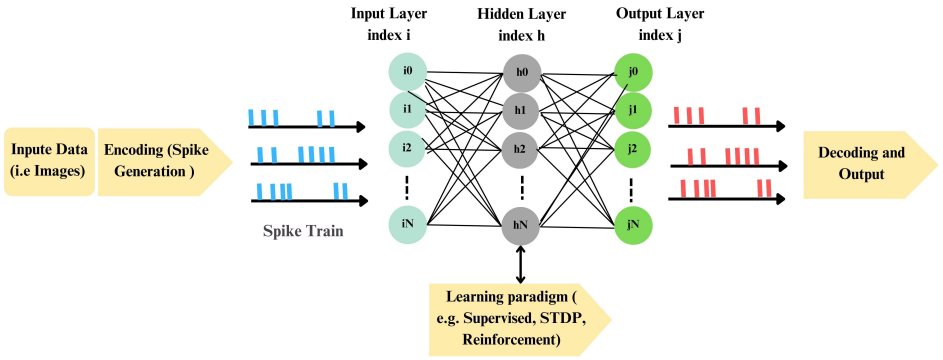


Figure 2. SNN processing schematic. Inputs are encoded as spike trains, processed by layers of spiking neurons, adapted via learning rules, and decoded into task outputs.

2.1.2. Power Efficiency: Mechanisms and Practice

The energy efficiency of SNNs stems from *sparsity* (computation occurs only upon spike events), *lower-cost accumulate (AC) updates* in place of dense MACs, and *event-driven memory traffic*, which reduces data movement—often the dominant energy term in modern systems [40,41]. On neuromorphic substrates (e.g., Intel Loihi, IBM TrueNorth, SpiNNaker), these properties translate into significant system-level gains through fine-grained parallelism, on-chip spike routing, and local memory near compute units [42–47]. Empirically, energy per inference scales with the total number of synaptic events and the average firing rate. Consequently, design parameters—encoding sparsity (e.g., TTFS, $\Sigma\Delta$), neuron/leak constants, thresholds, rate regularizers, and time-window length—directly trade accuracy for energy [28,37,48]. Comparative studies report multi$\times$ efficiency improvements for SNNs in event-rich settings and on dedicated hardware [42,45,49], while highlighting the importance of maintaining low spike rates and co-optimizing algorithms with hardware constraints.

2.1.3. Advantages and Challenges

By design, SNNs capture temporal dependencies with high timing resolution. They can compute efficiently in sparse regimes, enabling low-latency, low-power inference in settings such as event-based sensing and edge computing [25,28,48]. However, three obstacles hinder their widespread adoption. First, spike non-differentiability complicates gradient-based training; practical solutions rely on surrogate gradients, exact adjoints, local plasticity, or ANN-to-SNN conversion, each with trade-offs in accuracy, stability, or hardware compatibility [16–18]. Second, performance can lag ANN baselines if encoders, decoders, and neuron models are not co-designed for the specific task [50]. Third, software and hardware ecosystems for SNNs are still maturing, and standardized, energy-aware benchmarks remain limited.

2.1.4. Learning and Encoding

Unsupervised learning in SNNs often relies on STDP to uncover structure from temporal correlations [51]. Supervised training employs BPTT with surrogate gradients to approximate spike derivatives, enabling deep SNN optimization [16]. Reinforcement learning adjusts synaptic weights based on reward signals, supporting closed-loop control and robotics [52]. Encoding strategies strongly influence latency and energy—rate codes are robust but can be spike-heavy; time-based codes reduce spikes and latency but demand pre-

cise timing; and population codes improve separability and noise tolerance at the expense of added computational cost.

2.1.5. Real-World Applications

SNNs have been validated across multiple domains where temporal precision and energy constraints are critical. In *event-based vision*, SNNs implemented on neuromorphic hardware achieve real-time gesture recognition on Dynamic Vision Sensor (DVS) datasets with milliwatt-scale power budgets [42,53], demonstrate robust object and gesture processing on mobile platforms [54,55], and enable low-latency tracking and control [56,57]. Beyond human-centric vision, embedded deep models are applied to animal affect recognition [9], where SNNs offer low-latency, low-power inference for on-animal or field-deployed sensors. In *robotics and closed-loop control*, while conventional ANN-based controllers remain prevalent in practice [58,59], spike-based policies can operate on-chip, enabling responsive and power-aware navigation and manipulation [53,56]. In *biomedical signal processing*, SNNs support wearable ECG/EEG analytics and brain–computer interfaces under stringent energy and latency constraints [60–63]. In *industrial monitoring and tribology*, neural models estimate lubrication parameters from sensor data [7], where event-driven SNN inference can reduce both power consumption and latency at the edge. For *time-series forecasting*, SNNs model nonstationary environmental and energy signals, with traditional ML baselines from subsurface and energy domains providing context for accuracy–efficiency comparisons [8] (e.g., wind and solar forecasting), while maintaining low inference costs [64–68]. In *finance and IoT/edge analytics*, event-driven SNNs process sparse, asynchronous streams for anomaly detection and prediction under tight power budgets [69–72]. These applications underscore SNNs' ability to transform biological inspiration into practical, energy-efficient intelligence—particularly when learning rules, encoding schemes, and neuron models are co-optimized with the target hardware and workload.

2.2. Encoding in SNNs

Encoding is a pivotal process in SNNs, transforming continuous-valued inputs into discrete spike events and thereby bridging the gap between external stimuli and spike-based information processing [73]. This transformation is central to leveraging the distinctive advantages of SNNs, including temporal dynamics, event-driven computation, and energy efficiency [74]. The choice of encoding strategy directly determines how effectively information is represented, how temporal patterns are captured, and how power-efficiently the network operates—properties that are crucial for real-time processing and deployment on neuromorphic hardware [75].

Despite their importance, encoding schemes face several design challenges. These include balancing representational accuracy with computational complexity, maintaining biological plausibility, and ensuring compatibility with neuromorphic circuits [76,77]. Furthermore, encoding decisions strongly influence system robustness, as some strategies offer greater resilience to noise or adversarial perturbations than others [78]. Consequently, research on encoding has increasingly focused on systematically exploring and optimizing strategies to enhance scalability and efficiency, thereby positioning SNNs as viable alternatives to traditional ANNs in energy-constrained and real-time environments [79].

This subsection reviews the main categories of encoding schemes employed in SNNs, emphasizing their respective strengths and limitations. Table 1 provides a comparative overview, while Figures 3 and 4 illustrate representative examples. The first figure highlights three fundamental approaches—rate, TTFS, and burst coding—while the second figure expands the view to include inter-spike interval, N-of-M, population, and PoFC coding. Together, these visualizations and the summary table offer an integrated perspective on how information can be encoded in SNNs.

Table 1. Summary Of SNN encoding schemes reflecting the critical analysis in Section 2.2. Figures 3 and 4 illustrate representative examples.

Encoding Type	Main Applications	Complexity	Biological Plausibility	Advantages	Challenges
Rate Coding [38,75,78]	Image and signal processing; ANN-to-SNN conversion; resource-constrained inference	Low	High	Simple implementation; noise–adversarial robustness; hardware-friendly mapping from activations	Loses fine temporal structure; window-length–latency sensitivity; can require high spike counts that raise energy [80]
Direct Input Encoding [73,74,76,78,81]	Deep vision and real-time pipelines with large datasets; accuracy/latency–critical use	Moderate–high	Low	Fewer time steps; preserves input fidelity; simplifies front end; fast inference	Not event driven; multi-bit input raises compute–energy; lower biological realism
Temporal Coding [14,24,38,76,82]	Rapid sensory processing; real-time decisions; fine temporal discrimination–patterns	High	High	High information per spike; low-latency responses; potentially energy efficient with sparse spiking	Sensitive to jitter/noise; complex decoding; training with precise timings is challenging
Population Coding [83,84]	Speech–audio; noisy environments; improving separability with simple classifiers	High	High	Noise robustness via redundancy; improved linear separability	More neurons increase energy; decoding large populations adds computational overhead
ΣΔ Encoding [74,78,85–88]	Dynamic signals (wearables, biomedical, streaming sensors); energy-aware neuromorphic platforms	Moderate–high	Moderate	Encodes changes (fewer spikes) with good fidelity; noise shaping; strong energy savings	Requires feedback-loop–circuit tuning; trade-offs among fidelity, latency, and energy
Burst Coding [13,75,89,90]	Biologically realistic simulations; temporally complex signals; long-activity tasks	Moderate–high	High	Rapid information transfer in spike packets; can be energy saving when bursts are well managed	Synchronizing bursts complicates decoding; scalability and parameter tuning on hardware
PoFC [87,91–93]	Spatial navigation/sensory processing with oscillations; high-fidelity temporal representation	High	High	Dense information per spike via phase; strong discriminability; potential spike-count/energy reduction	Requires precise global phase reference; sensitive to timing noise; complex decoding and STDP integration

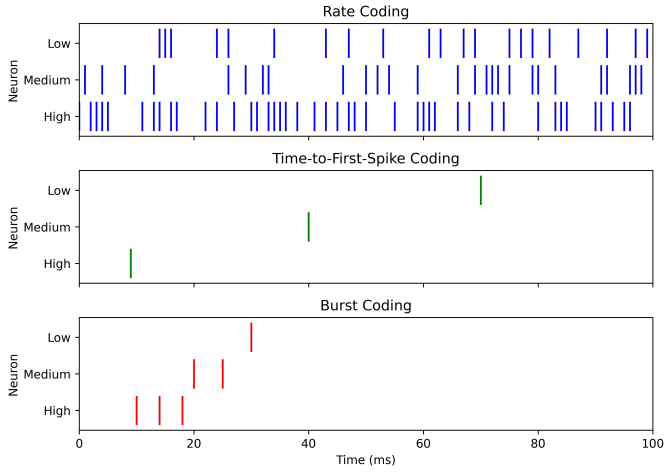


Figure 3. Illustration of encoding using rate, TTFS, and burst coding. (1) Rate coding shows different firing rates for low, medium, and high stimulus intensities. (2) TTFS coding represents intensity by the latency to the first spike. (3) Burst coding shows variations in the number and duration of spikes. Note: The figure is a single composite illustration, and the numbers (1)–(3) are not subfigure labels.

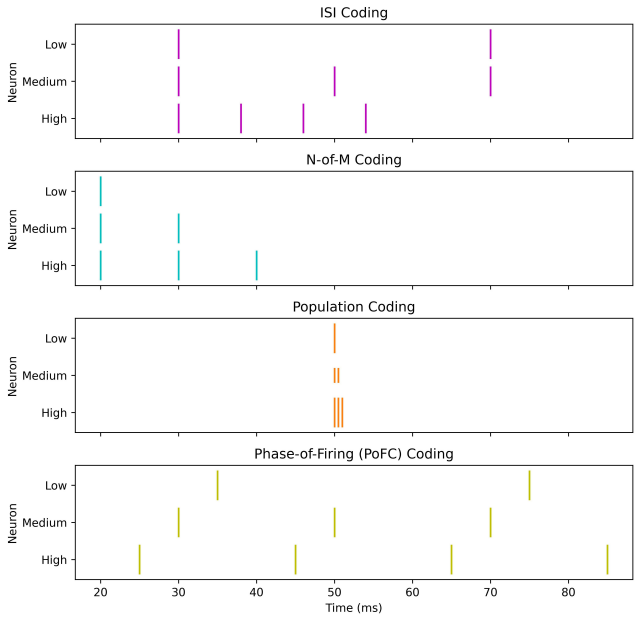


Figure 4. Illustration of encoding using inter-spike interval (ISI), N-of-M, population, and PoFC coding. (1) SIS coding shows differences in inter-spike intervals across stimulus intensities. (2) N-of-M coding illustrates the number of spikes from a fixed set. (3) Population coding indicates the number of active neurons. (4) PoFC coding aligns spikes with specific phases of an oscillatory cycle. Note: The figure is a single composite illustration, and the numbers (1)–(4) are not subfigure labels.

2.2.1. Rate Coding

Rate coding represents stimulus intensity by the number of spikes emitted within a time window:

$$V = \frac{N_{\text{spike}}}{T}. \tag{1}$$

Its simplicity, biological plausibility, and the clear correspondence between firing rates and ANN activations make it a common choice for image and signal processing, as well as for embedded deployments where robustness and ease of implementation are priorities [78]. Because it averages activity over a window T, it discards fine timing information. Consequently, achieving high accuracy often requires longer windows or higher spike counts, which increases latency and energy consumption, making this method less suitable for rapid, event-driven scenarios [78,80].

2.2.2. Direct Input Encoding

Direct input encoding feeds continuous-valued signals (e.g., pixel intensities) directly into the input layer without converting them into spike trains [74,78]. By bypassing stochastic spike generation, it can reduce the number of time steps and improve latency and accuracy—making it useful for deep architectures, large-scale vision applications, and real-time decision tasks—while preserving input fidelity and simplifying the preprocessing stage [76]. However, this convenience comes at the cost of event-driven sparsity: multi-bit input activity increases computational load and energy consumption compared to spike-based schemes and reduces biological plausibility, making it a weaker fit for resource- or power-constrained neuromorphic deployments [73,74,78,81].

2.2.3. Temporal Coding

Temporal coding emphasizes the precise timing of spikes rather than their average rate, aligning with biological evidence that spike timing conveys rich and rapidly accessible information [24,38,75]. By leveraging the timing of spikes, this family of methods offers

faster and potentially more energy-efficient information transmission than rate coding. Several variants exist:

- *Time-to-first-spike (TTFS):* Encodes stimulus strength in the latency of the first spike:

$$t_{\text{spike}} = t_{\text{onset}} + \Delta t, \tag{2}$$

  where t_{onset} is the stimulus onset time and Δt the delay before firing [14]. TTFS is highly power-efficient, as minimal spiking activity can support rapid decisions, although it requires precise timing and complicates learning.
- *Inter-spike interval (ISI):* Encodes information in the time gap between consecutive spikes:

$$ISI_i = t_{i+1} - t_i, \tag{3}$$

  providing richer temporal detail at the cost of increased spiking and energy use [14].
- *N-of-M (NoM) coding:* Transmits only the first N out of M possible spikes, improving hardware efficiency but discarding spike-order information [82].
- *Rank order coding (ROC):* Utilizes the sequence of spike arrivals according to synaptic weights, providing high discriminability but at the cost of computational intensity and sensitivity to precise timing [24].
- *Ranked-N-of-M (R-NoM):* Combines ROC and NoM by propagating the first N spikes while weighting their order:

$$\text{Activation} = \sum_{i=1}^{N} \text{Weight}_i \times f(i) \times \text{Spike}_i, \tag{4}$$

  where $f(i)$ is a decreasing modulation function with spike order [82].

Temporal coding can yield more information per spike and support low-latency decision-making, potentially reducing overall energy consumption if efficiently implemented. However, it also introduces challenges in robustness: decoding can be complex, the schemes are sensitive to noise and spike-timing variability, and training deep SNNs with precise temporal codes remains difficult [76]. These trade-offs position temporal coding as a powerful yet demanding alternative, best suited for applications where rapid response and high temporal precision outweigh implementation simplicity and energy stability.

2.2.4. Population Coding

Population coding distributes information across an ensemble of neurons, enhancing robustness and separability—useful for noisy, time-varying signals such as speech, audio, or sensor fusion—but at the cost of greater neuron count, decoding overhead, and increased energy and memory traffic [83,84]. A common readout is the weighted population response:

$$R = \sum_{i=1}^{n} w_i \, r_i,$$

where w_i and r_i denote the weight and firing rate of the i-th neuron, respectively. This approach complements latency- and phase-based schemes by pooling precise temporal cues, thereby trading efficiency for improved reliability [83,84].

2.2.5. ΣΔ Encoding

ΣΔ encoding transmits *changes* rather than absolute values through a simple feedback loop:

$$\Delta x_t = x_t - x_{t-1}, \tag{5}$$

$$y_t = \text{Threshold}(\Sigma_t + \Delta x_t), \tag{6}$$

$$\Sigma_{t+1} = \Sigma_t + \Delta x_t - y_t, \tag{7}$$

where x_t is the input, Δx_t the change in input, y_t the emitted spike, and Σ_t an accumulated reconstruction error. By encoding differences instead of absolute values, ΣΔ encoding produces sparse spike trains with high temporal fidelity and reduced energy consumption, making it particularly effective for dynamic, streaming signals on neuromorphic and wearable/biomedical platforms [74,78]. Noise shaping supports robust signal reconstruction [85–87], though effective deployment requires careful tuning of thresholds and feedback parameters, as well as circuit-aware trade-offs among fidelity, latency, and power [88].

2.2.6. Burst Coding

Burst coding represents information using short, high-frequency spike packets, mirroring rhythmic firing patterns observed in cortical and subcortical regions [13,75,89,90]. A burst for neuron i over a window $[t_{\text{start}}, t_{\text{end}}]$ can be expressed as

$$B_i = \sum_{t=t_{\text{start}}}^{t_{\text{end}}} s_i(t), \tag{8}$$

where $s_i(t) \in \{0, 1\}$ indicates a spike at time t. Grouping spikes into short, dense packets facilitates rapid information transfer and provides strong temporal cues with fewer decision windows—an advantage for event-rich sensing and closed-loop control. However, synchronizing burst onset and network-wide timing increases decoding complexity and hardware demands, while poorly tuned burst parameters may negate energy savings [13,75,89,90]. In practice, burst coding is most effective when biological realism and efficient processing of temporally complex signals are priorities, provided that synchronization and parameter tuning are carefully managed.

2.2.7. PoFC Coding

Phase-of-firing coding (PoFC) encodes information through the phase of each spike relative to an ongoing oscillation (e.g., theta or gamma rhythms) [87,91–93]. The phase of the i-th spike is computed as

$$\phi_i = \text{mod}\Big(t_{\text{spike}_i} - t_{\text{osc}},\, T_{\text{osc}}\Big), \tag{9}$$

where t_{osc} is the oscillation reference time and T_{osc} its period. By exploiting phase locking observed in the hippocampus and other brain regions, PoFC can encode more information per spike than rate coding—yielding rich, low-count representations and potentially lower energy consumption. However, it requires precise synchronization and is sensitive to timing jitter; accurate phase extraction and integration with plasticity mechanisms (e.g., STDP) further increase implementation complexity [91–93]. PoFC is most effective for high-fidelity temporal discrimination and navigation or sensory tasks shaped by oscillatory dynamics, though resource-constrained neuromorphic deployments must balance its representational richness against robustness and hardware simplicity [87].

2.2.8. Impact of Encoding Schemes on SNN Performance and Power Efficiency

Encoding serves as a primary lever shaping both information representation and the computational or energy footprint of SNNs. Foundational work includes ΣΔ modulation for efficient analog-to-spike conversion [85] and temporally rich schemes such as rank-order coding [24]. Subsequent surveys emphasize that the "best" encoding scheme is application-dependent, balancing accuracy, latency, energy, robustness, and hardware constraints [73].

Recent developments highlight several trends: (i) ΣΔ interfaces adapted to SNNs maintain signal fidelity over wide dynamic ranges while reducing spike counts, though they require careful feedback tuning and add circuit complexity [86,87]; (ii) unified analyses of TTFS, ROC, NoM, and R–NoM clarify discriminability under realistic noise and timing variability, and differentiable training methods help narrow the gap with gradient-based optimization [76,82]; and (iii) hybrid or dynamic approaches—such as layer-wise burst+phase schemes or attention-gated temporal encodings—enhance throughput-per-watt without sacrificing accuracy [13,94,95].

Comparative trends across studies remain consistent: Direct-input (analog) encoding reduces time steps and can improve accuracy and latency for deep architectures but sacrifices event-driven sparsity at the input stage [74]; rate coding is simple, conversion-friendly, and robust but loses fine temporal resolution and may require longer windows or higher spike counts [78]; temporal encodings (TTFS/ISI) provide high information density and fast decision-making yet are sensitive to jitter and more challenging to train or deserialize at scale [14,82]; and population coding improves separability and noise tolerance in temporally rich signals but increases neuron count and decoding cost [84]. Table 1 and Figures 3–5 summarize these trade-offs.

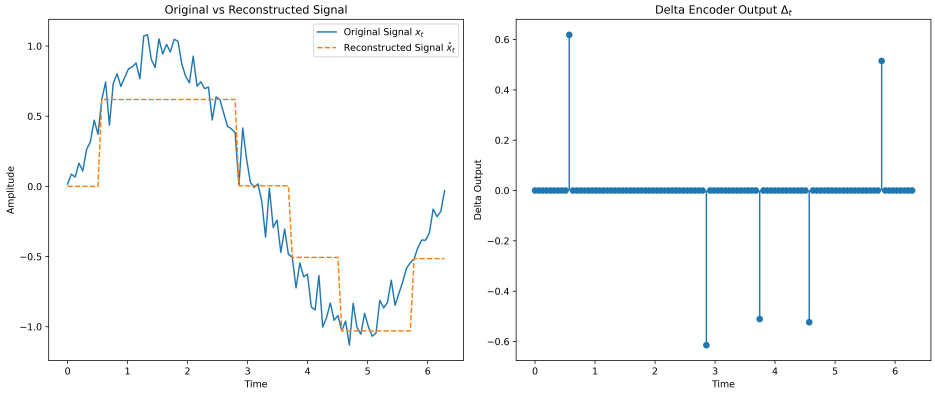


Figure 5. ΣΔ encoding of a dynamic signal. The original noisy input x_t and its reconstruction $\hat{x}_t$ from spikes y_t are shown. The delta stream Δx_t highlights thresholded variations, illustrating high fidelity with reduced spike counts.

Guidelines for Selecting an Encoding Scheme

- *Primary objective:* For *latency-critical* workloads, prefer TTFS, ROC, or layer-wise hybrids that emphasize fast temporal cues [13,14]. For *energy-constrained* continuous sensing, use ΣΔ or sparse rate coding [78,86,87].
- *Noise and non-idealities:* In noisy sensors or non-ideal hardware, population or burst coding can enhance robustness. Fine temporal encodings, however, may require precise clocking or signal filtering [13,84].
- *Model and hardware complexity:* Choose rate or direct-input encoding for simplicity, ANN→SNN conversion, or resource-limited devices. Reserve ΣΔ and temporal hybrid schemes for platforms that support feedback loops or high-precision timing [74,78,87].
- *Scalability and training:* When end-to-end gradient optimization is essential, select encodings with proven surrogate-gradient compatibility (rate, TTFS, or selected hybrids) and validated noise tolerance [76,82].

In summary, no single encoding scheme dominates across all use cases. Effective designs tailor—and often *hybridize*—encodings according to the target application's accuracy–latency–energy requirements and the hardware's timing and circuit capabilities. These relationships are visualized in Figures 3–5, and summarized in Table 1.

2.3. Spiking Neuron Models

Neuron models are the computational primitives of SNNs: they define how synaptic events are integrated, how spikes are generated, and how signals propagate through a network. The choice of model governs representational power, energy consumption, latency, and hardware suitability [10,38,96]. Broadly, neuron models lie on a spectrum that trades biological fidelity for computational efficiency. At one end, biophysical formulations such as HH accurately reproduce membrane dynamics but are computationally expensive at scale [97,98]. At the other end, families of integrate-and-fire models (IF, LIF, and extensions such as ALIF, QIF/EIF, SRM) abstract spiking as filtered integration followed by thresholding, enabling large networks and low-power deployment [38]. Intermediate phenomenological models (IZH, AdEx, RF) capture rich firing behaviors with moderate cost, offering a pragmatic balance for many practical applications [39,99].

Selecting a neuron model is, therefore, an application-driven decision that weighs fidelity, efficiency, and implementation complexity. Detailed conductance-based models can be advantageous for tasks requiring precise subthreshold or adaptive dynamics, while simplified IF variants often suffice for rate-based processing and fast, energy-efficient inference. Intermediate models are attractive when diverse temporal patterns are important, but computational resources are limited. Additional factors include the availability of stable learning rules (e.g., surrogate-gradient training for LIF/AdEx/IZH), robustness to noise and device non-idealities, and the target hardware's support for state variables and synaptic dynamics [96,98,99]. A concise, per-model summary of mechanisms, advantages, and challenges is provided in Table 2.

2.3.1. Spike Response Neuron (SRM)

The SRM provides a compact, kernel-based description of a spiking neuron, in which the membrane potential is modeled as the superposition of postsynaptic response kernels and a spike-triggered afterpotential [38,100]:

$$V(t) \;=\; \eta(t-\hat{t}) \;+\; \int_{-\infty}^{t} \kappa(t-\hat{t},s)\, I(t-s)\, ds, \tag{10}$$

where $\hat{t}$ denotes the last spike time, $\eta(\cdot)$ is a refractory/after-spike kernel, $\kappa(\cdot,\cdot)$ is a synaptic filter, and $I(\cdot)$ represents the input drive. A spike is generated when $V(t)$ exceeds a dynamic threshold:

$$\theta(t-\hat{t}) = \begin{cases} \infty, & t-\hat{t} \le \gamma_{\text{ref}}, \\ \theta_0 + \theta_1\, e^{-(t-\hat{t})/\tau_\theta}, & \text{otherwise}, \end{cases} \tag{11}$$

where γ_{ref} is the absolute refractory period and the threshold decays exponentially back to θ_0. The SRM encompasses fixed-kernel variants (SRM_0) and forms with spike-triggered threshold adaptation or state-dependent kernels (SRM^+), unifying integrate-and-fire dynamics while remaining analytically tractable and efficient for event-driven simulation. This separation of synaptic and refractory effects makes SRM convenient for modeling STDP, analyzing computational properties, and hardware-efficient implementations. Being phenomenological, however, it lacks rich subthreshold biophysics (e.g., voltage-gated conductances or resonance) unless extended, and practical use requires careful calibration of η, κ, and $\theta(\cdot)$ [38].

2.3.2. Integrate-and-Fire Neuron Family (IF, LIF)

The integrate-and-fire family models a neuron as a linear integrator with threshold-and-reset dynamics [35,38,101–103]. When $V(t)$ crosses ϑ from below, a spike is emitted and $V \leftarrow V_{\text{reset}}$ (optionally followed by a refractory period). Their minimal state, analytical

tractability, and event-driven nature make IF/LIF models a cornerstone for large-scale SNNs and neuromorphic hardware deployment.

Perfect IF (PIF)

The ideal, non-leaky integrator accumulates input without passive decay:

$$C\,\frac{dV(t)}{dt} \;=\; I(t), \tag{12}$$

where C is the membrane capacitance and $I(t)$ is the input current. When $V(t) \geq \vartheta$, the neuron emits a spike and resets to V_{reset}. PIF is computationally minimal but overestimates temporal integration by neglecting leak effects.

Leaky IF (LIF)

Illustrated in Figure 6, the LIF model incorporates passive membrane leakage:

$$\tau_m\,\frac{dV(t)}{dt} \;=\; -\big(V(t) - V_{\text{rest}}\big) + R\,I(t), \tag{13}$$

where $\tau_m = RC$ is the membrane time constant, R the membrane resistance, and V_{rest} the resting potential. The LIF model captures membrane decay, supports stochastic analysis, and enables efficient event-driven updates [38,102].

Both IF and LIF provide simplicity, stability, and compatibility with surrogate-gradient training, but their linear subthreshold dynamics lack conductance-based nonlinearities such as adaptation or resonance. They are frequently extended with adaptive mechanisms (ALIF) or nonlinear thresholds (EIF/AdEx) when richer temporal behavior is required.

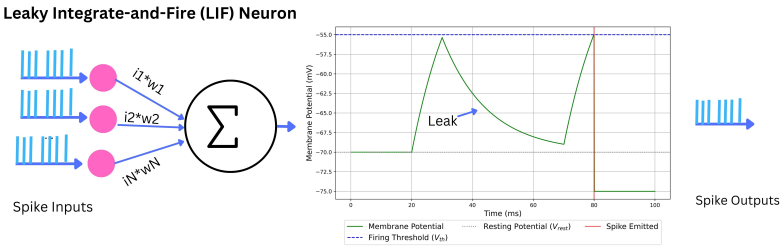


Figure 6. Illustrative membrane trajectory of the leaky integrate-and-fire (LIF) neuron. $V(t)$ integrates synaptic input with passive leak toward V_{rest}, emits a spike upon reaching the threshold ϑ, and resets to V_{reset}.

2.3.3. Adaptive Leaky Integrate-and-Fire Neuron (ALIF)

ALIF augments LIF with spike-frequency adaptation, reproducing reduced firing under sustained drive via either a *dynamic threshold* or a *spike-triggered hyperpolarizing current* [38,100,104]. This adds history dependence while retaining efficient, event-driven simulation and surrogate-gradient trainability.

Dynamic-threshold ALIF

$$\tau_m \frac{dv(t)}{dt} = -\big(v(t) - V_{\text{rest}}\big) + R\,I(t), \qquad \text{spike if } v(t) \geq \theta(t), \tag{14}$$

$$\theta(t) = \theta_0 + \beta \sum_{t_s<t} \exp\!\big(-(t - t_s)/\tau_\theta\big), \tag{15}$$

where $\tau_m{=}RC$, θ_0 is the baseline threshold, β the adaptation strength, and τ_θ its decay constant. On a spike at t_s, $v(t_s^+){=}V_{\text{reset}}$.

Current-based ALIF

$$\tau_m \frac{dv(t)}{dt} = -\big(v(t) - V_{\text{rest}}\big) + R\, I(t) - w(t), \qquad \text{spike if } v(t) \geq \vartheta, \tag{16}$$

$$\tau_w \frac{dw(t)}{dt} = -w(t) + b \sum_{t_s < t} \delta(t - t_s), \tag{17}$$

with adaptation current $w(t)$, jump b per spike, and decay τ_w. After a spike, $v(t_s^+)=V_{\text{reset}}$ and $w(t_s^+) \leftarrow w(t_s^-)+b$.

ALIF improves sequence processing and temporal credit assignment with modest overhead but introduces additional state variables and parameters that require calibration. Very slow or multi-timescale adaptation may call for extended variants (e.g., AdEx or multi-τ_w ALIF) [35,102].

2.3.4. Exponential Integrate-and-Fire Neuron (EIF)

The EIF neuron sharpens spike initiation by adding an exponential term to the leaky membrane dynamics, providing a smooth, biophysically motivated threshold [38,105]. In current-based form,

$$\tau_m \frac{dV(t)}{dt} = -\big(V(t) - V_{\text{rest}}\big) + \Delta_T \exp\left(\frac{V(t) - V_T}{\Delta_T}\right) + R\, I(t), \tag{18}$$

where V_{rest} is the resting potential, V_T the rheobase (effective spike-initiation) voltage, Δ_T the slope factor controlling onset sharpness, and $I(t)$ the input current. A spike is emitted when $V(t)$ diverges past a set peak (e.g., V_{spike}), after which $V \leftarrow V_{\text{reset}}$ and an absolute refractory period τ_{ref} may be imposed.

Compared with LIF, EIF reproduces the rapid upstroke of action potentials and more accurate responses to fast, fluctuating inputs (gain, phase response, and f–I curves) while remaining far cheaper than conductance-based models [103,105]. The added nonlinearity and parameters improve fidelity but require calibration (notably Δ_T and V_T) and can stiffen numerical integration near threshold. EIF serves as a practical compromise for studies of high-frequency synaptic integration, spike initiation, and neuromorphic implementations that need a smooth, differentiable surrogate of threshold dynamics [38].

2.3.5. Adaptive Exponential Integrate-and-Fire Neuron (AdEx)

AdEx extends LIF with an exponential spike-initiation term and a spike-triggered adaptation variable, yielding a compact neuron that reproduces regular/fast-spiking, adapting, and bursting behaviors [106]:

$$C_m \frac{dV_m}{dt} = -G_L\,(V_m - E_L) + G_L\,\Delta_T \exp\left(\frac{V_m - V_T}{\Delta_T}\right) - w + I_{\text{syn}}, \tag{19}$$

$$\tau_w \frac{dw}{dt} = a\,(V_m - E_L) - w, \tag{20}$$

where V_m is membrane potential, C_m capacitance, G_L leak conductance, E_L leak reversal, V_T spike-initiation voltage, Δ_T slope factor, w the adaptation current, a subthreshold adaptation, τ_w its time constant, and I_{syn} synaptic current. A spike is registered when V_m exceeds V_{spike}, after which $V_m \leftarrow V_{\text{reset}}$ and $w \leftarrow w + b$ (optional absolute refractory).

EIF-like sharp onset supports realistic spike initiation, while (a, τ_w, b) captures spike-frequency adaptation and bursting at modest cost, enabling efficient numerical integration and event-driven simulation. Hardware-oriented variants (fixed-point/high-accuracy arithmetic, power-of-two linearizations, and CORDIC exponentials) further reduce runtime without sacrificing fidelity [107–109]. Careful calibration of Δ_T, V_T, and adaptation parameters is essential; near-threshold stiffness may require smaller steps or dedicated solvers. In

practice, AdEx is a good default when richer dynamics than LIF are needed with only a slight complexity increase.

2.3.6. Resonate-and-Fire Neuron (RF)

The RF neuron captures subthreshold oscillations and frequency selectivity—complementing integrator-style IF/LIF models—via a complex state with linear dynamics [110]:

$$\dot{z}_i(t) \;=\; \big(b_i + i\,\omega_i\big)\, z_i(t) \;+\; \sum_{j=1}^{n} c_{ij}\,\delta\big(t - t'_j\big), \tag{21}$$

where $z_i \in \mathbb{C}$ encodes the oscillatory state, ω_i is the intrinsic angular frequency, $b_i < 0$ sets damping (stability requires $\mathrm{Re}\{b_i\} < 0$), c_{ij} are synaptic couplings, and $\delta(\cdot)$ are presynaptic spike impulses at times t'_j. A spike is emitted when a readout (e.g., $\mathrm{Im}\{z_i\}$ or a linear projection of $(\mathrm{Re}\{z_i\}, \mathrm{Im}\{z_i\})$) crosses threshold ϑ_{RF} from below, followed by a reset $z_i \leftarrow z_{\mathrm{reset}}$. Writing $z_i = x_i + iy_i$ reveals a damped resonator with eigenvalues $b_i \pm i\omega_i$, producing natural selectivity to inputs near ω_i and to spike phase—useful for tasks rich in temporal structure, resonance, and phase/PoFC-style coding.

In practice, RF offers lightweight oscillatory dynamics that simulate efficiently and admit analysis; balanced RF (BRF) improves stability in recurrent SNNs by regulating excitation–inhibition [111], and analog realizations demonstrate direct signal-to-spike conversion for edge sensing without explicit A/D front ends [112]. As a phenomenological model, however, spiking is threshold-based (not biophysical), channel nonlinearities are implicit, parameters (frequency, damping, reset) require calibration, and the second-order state increases per-neuron cost versus IF/LIF; for strongly nonlinear spiking such as complex bursting, AdEx or IZH variants can be preferable.

2.3.7. Hodgkin–Huxley Neuron (HH)

The HH model gives a biophysical account of spike generation via voltage-gated Na^+ and K^+ channels plus leak, fit to voltage-clamp data from squid axon [97]. The membrane equation balances capacitive and ionic currents:

$$C_m \frac{dV}{dt} = -\,g_L\,(V - E_L) \;-\; g_{\mathrm{Na}}\,m^3 h\,(V - E_{\mathrm{Na}}) \;-\; g_{\mathrm{K}}\,n^4\,(V - E_{\mathrm{K}}) \;+\; I_{\mathrm{syn}} \;+\; I_{\mathrm{ext}}, \tag{22}$$

with gating kinetics

$$\dot{x} \;=\; \alpha_x(V)\,\big(1 - x\big) - \beta_x(V)\,x, \qquad x \in \{m, h, n\}. \tag{23}$$

As the gold standard for single-cell fidelity, HH reproduces subthreshold dynamics, spike upstroke, refractoriness, and pharmacological effects, making it ideal for mechanism studies and validating reduced models. Its computational cost (stiff ODEs, many parameters) limits large network use, so EIF/AdEx/LIF variants are typically preferred for SNN simulation [38]. Hardware-aware implementations (e.g., CORDIC-based exponentials/fractions on FPGAs) improve tractability [113], and conductance-based synapse extensions clarify information transfer under realistic inputs [114].

Table 2. Summary of spiking neuron models: key mechanisms, advantages, and limitations. Complexity and plausibility are qualitative: very low (V. Low), low, moderate (Mod.), moderate–high (Mod.–High), high, very high (V. High).

Neuron Model	Main Applications	Complexity	Biological Plausibility	Advantages	Challenges
SRM [38,100]	STDP studies; analytical probes of SNNs; event-driven/hardware simulation	Low	Mod.	Kernel-based, tractable; separates synaptic and refractory effects; efficient	Limited subthreshold nonlinearities; requires kernel/threshold calibration
IF [101,103]	Baselines; large-scale SNNs; theoretical analysis	V. Low	Low	Extremely simple; fast; closed-form insights	Unrealistic integration (no leak); no adaptation/resonance
LIF [38,102]	Neuromorphic inference; brain–computer interfaces; large-scale simulations	Low	Mod.	Good accuracy–efficiency balance; event-driven; well-studied statistics	No intrinsic adaptation/bursting; linear subthreshold
ALIF [100,104]	Temporal/sequential tasks; speech-like streams; robotics control	Mod.	Mod.	Spike-frequency adaptation; better temporal credit assignment	Extra state and parameters; tuning sensitivity
EIF [103,105]	Fast fluctuating inputs; spike initiation studies; neuromorphic surrogates	Mod.	High	Sharp, smooth onset; improved gain/phase vs. LIF	Parameter calibration; stiffer near threshold
AdEx [106]	Cortical pattern repertoire; adapting/bursting cells; efficient yet rich neurons	Mod.–High	High	Diverse firing patterns with compact model; efficient integration	More parameters; careful numerical and hardware calibration
RF [110]	Resonance/phase codes; frequency-selective processing; edge sensing prototypes	Mod.	High	Captures subthreshold oscillations and resonance; phase selectivity	Phenomenological spike; added second-order state; parameter tuning
HH [97]	Biophysical mechanism studies; channelopathies; pharmacology; single-cell fidelity	V. High	V. High	Gold-standard fidelity; reproduces ionic mechanisms and refractoriness	Computationally expensive; stiff; many parameters
IZH [39,89]	Large-scale networks with rich firing; cortical microcircuits; plasticity studies	Mod.	High	Wide repertoire at low cost; simple 2D form with reset	Lower biophysical interpretability; heuristic fitting
RF–IZH [110,115]	Phase-aware resonance with lightweight reset; recurrent SNNs; neuromorphic stacks	Mod.	High	Preserves phase; efficient event rules; toolchain support (Lava)	Phenomenological; calibration of (ω, b, ϑ); still > IF/LIF cost
CUBA [98]	Large-scale SNNs; theory; fast prototyping; hardware with current-mode synapses	Low	Low	Very fast; analytically convenient; event-driven synapses independent of V	Ignores reversal/shunt; biases in high-conductance regimes vs. COBA
ΣΔ [88,115,116]	Energy-constrained streaming (audio/vision); edge sensing; ΣΔ–LIF layers	Mod.	Mod.	Sparse, error-driven spikes; low switching energy; good reconstruction	Feedback/threshold tuning; integration details for stability and latency

2.3.8. Izhikevich Neuron (IZH)

The IZH neuron is a two-state hybrid model reproducing a wide repertoire of cortical firing patterns at very low computational cost [39,89]. With membrane potential V and recovery variable U,

$$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - U + I(t), \tag{24}$$

$$\frac{dU}{dt} = a\left(bV - U\right), \tag{25}$$

and after-spike reset

$$\text{if } V \geq V_{\text{peak}} \text{ (typically 30 mV)}, \quad V \leftarrow c, \qquad U \leftarrow U + d. \tag{26}$$

Here $I(t)$ is the synaptic/injected current; a, b, c, d control time scale, sensitivity, and reset. Proper tuning yields tonic/fast spiking, bursting, chattering, rebound, and Class I/II excitability while avoiding explicit action-potential integration, thus enabling large-scale simulations and neuromorphic emulation with good accuracy–efficiency trade-offs [39,89]. As a phenomenological model, parameters have limited biophysical interpretability and often require heuristic fitting; numerical care near V_{peak} is useful. In practice, it serves as a pragmatic middle ground—richer dynamics than IF/LIF at modest cost, though for detailed channel mechanisms HH/AdEx may be preferred, and for strict minimalism or gradient training pipelines LIF/ALIF remain common [38].

2.3.9. Resonate-and-Fire Izhikevich Neuron (RF–IZH)

RF–IZH augments the RF oscillator with a minimal hybrid spike/reset that preserves phase while simplifying post-spike dynamics [110]. The subthreshold state follows RF (cf. (21)):

$$\dot{z}(t) = (b + i\omega)z(t) + \sum_j c_j \, \delta(t - t'_j), \tag{27}$$

with $z \in \mathbb{C}$ the oscillatory state, ω preferred frequency, $b < 0$ damping, and c_j synaptic couplings. Spiking is phase-sensitive, with a lightweight reset that retains the imaginary (phase) component:

$$\text{spike if } \operatorname{Im}\{z(t)\} \geq \vartheta, \tag{28}$$

$$\text{after spike: } \operatorname{Re}\{z(t)\} \leftarrow 0, \quad z(t) \leftarrow i \operatorname{Im}\{z(t)\}, \tag{29}$$

optionally followed by an absolute refractory period. This preserves resonance and phase continuity, yielding low-cost frequency selectivity and PoFC-style coding that is more faithful to oscillatory timing than IF/LIF yet far cheaper than conductance-based neurons. As a phenomenological model, spike generation is thresholded and channel nonlinearities are implicit; parameters (ω, b, ϑ) require calibration, and for complex bursting, AdEx/IZH or HH may be preferable. Efficient discrete-time updates with event-driven checks are available in neuromorphic toolchains (e.g., Lava) for large-scale simulation and deployment [115].

2.3.10. Current-Based Neuron (CUBA)

In the current-based (CUBA) formulation, synaptic events enter the membrane equation as additive currents independent of voltage, in contrast to conductance-based (COBA) synapses that scale with the driving force [98]. The membrane potential obeys

$$\frac{dV(t)}{dt} = -\frac{V(t) - V_{\text{rest}}}{\tau_m} + I_{\text{syn}}(t), \tag{30}$$

with resting potential V_{rest}, membrane time constant τ_m, and total synaptic current $I_{\text{syn}}(t)$ (typically a weighted sum of presynaptic spike kernels). A spike is emitted when $V(t) \geq V_{\text{thresh}}$, followed by a reset $V \leftarrow V_{\text{reset}}$ and an optional absolute refractory period.

CUBA is computationally efficient and analytically convenient for large-scale SNNs because synaptic drive does not depend on V. Its main limitation is reduced biological realism relative to COBA (e.g., no reversal potentials or shunting), which can bias gain and dynamics in high-conductance regimes [98].

2.3.11. Sigma–Delta Neuron (ΣΔ)

The ΣΔ neuron realizes asynchronous pulsed ΣΔ modulation (APSDM), emitting spikes only when the discrepancy between the instantaneous drive $S(t)$ (e.g., filtered synaptic current) and its internal reconstruction $\hat{S}(t)$ exceeds a dynamic threshold [116]. With the error variable

$$u(t) = S(t) - \hat{S}(t), \tag{31}$$

a spike is produced when

$$|u(t)| \geq \vartheta(t) \;\Rightarrow\; y(t) \in \{-1, +1\}, \quad \hat{S}(t^{+}) \leftarrow \hat{S}(t^{-}) + y(t)\,\vartheta(t), \tag{32}$$

and the adaptive threshold evolves via

$$\vartheta(t) = \vartheta_0 + \sum_{t_i < t} \beta \exp\bigl(-(t - t_i)/\tau_\vartheta\bigr). \tag{33}$$

Thus, spikes convey only significant changes, yielding sparse activity and low switching energy.

A practical discrete-time form (common in neuromorphic stacks) uses a dead-zone delta encoder with sigma reconstruction [115]:

$$\Delta x_t = x_t - \hat{x}_{t-1}, \qquad y_t = \operatorname{sgn}(\Delta x_t)\,\mathbb{1}\{|\Delta x_t| > \theta\}, \qquad \hat{x}_t = \hat{x}_{t-1} + y_t\,\theta, \tag{34}$$

optionally combined with LIF subthreshold dynamics ("ΣΔ–LIF"). In dynamic sensing (audio/vision streams), this event-driven compressor achieves high-fidelity reconstructions with few spikes and strong energy efficiency [88,116].

Effective use hinges on feedback/threshold tuning and stability of the reconstruction. Poorly chosen θ or kernels can increase quantization noise or latency, and deployment may require careful calibration (step sizes, refractory handling). Reference implementations and layer abstractions are available in Lava for large-scale simulation and deployment [115].

2.3.12. Trade-Offs in Neuron Model Selection for SNNs

Model choice balances biological fidelity, compute–energy, trainability, and hardware fit (see Table 2).

IF/LIF—Minimal state and event-driven efficiency make them defaults for large, low-power systems; linear subthreshold dynamics limit adaptation and resonance [38,101–103].

ALIF—Adds spike-frequency adaptation (dynamic threshold or current) for better sequence processing with modest overhead; extra parameters require calibration [100,104].

EIF/AdEx—Smooth spike onset (exponential term) and adaptation reproduce diverse cortical patterns at moderate cost; accuracy depends on careful tuning of $\Delta_T, V_T, a, b, \tau_w$ and numerics can stiffen near threshold [105,106].

SRM—Kernel superposition with spike-triggered refractoriness is analytically tractable and efficient; phenomenological nature limits rich subthreshold nonlinearities unless extended [38,100].

IZH—Two-dimensional hybrid yields rich firing at low cost; parameters are less biophysically interpretable and typically fit heuristically [39,89].

RF/RF–IZH—Resonator neurons capture phase/resonance for PoFC-like codes; lightweight but with added second-order state and calibration; BRF improves recurrent stability [110,111].

HH—Gold-standard ion-channel fidelity for mechanism studies; too costly/stiff for large or ultra-low-power networks; useful as a reference [97].
CUBA—Current-based synapses are simple and fast for scaling and analysis; reduced realism vs. COBA grows in high-conductance regimes [98].
ΣΔ *neuron*—Event-driven, error-based spiking transmits only significant changes for sparse, energy-efficient operation; performance hinges on feedback/threshold tuning; supported in Lava and low-power circuits [88,115,116].

Practical Guidelines

- *Energy/scale:* LIF/ALIF or ΣΔ; use CUBA when speed > realism.
- *Temporal richness:* ALIF, EIF/AdEx, or RF/RF–IZH for adaptation, resonance, or phase coding.
- *Mechanistic fidelity:* HH for channel-level questions; validate reduced models against HH.
- *Trainability:* Prefer models with robust surrogate-gradient practice; constrain parameters to avoid stiffness/instability.
- *Hardware fit:* Match state and nonlinearities to the target fabric (fixed-point, exponentials/CORDIC, event-driven kernels); layer-wise hybrids (e.g., LIF front ends + AdEx/ALIF deeper) often optimize accuracy–efficiency.

Overall, no single model is optimal; combine or stack models to meet the task's accuracy–latency–energy envelope and hardware constraints.

2.4. Learning Paradigms in SNNs

Learning endows SNNs with the capacity to adapt and generalize while exploiting event-driven, temporally precise computation—an advantage over ANNs with static, continuous activations [18]. The same spiking discreteness, however, introduces two central challenges: (i) the non-differentiability of spikes, which complicates gradient-based optimization, and (ii) temporal credit assignment across membrane dynamics and spike times [16,17]. Contemporary training strategies address these challenges through several complementary paradigms.

Supervised methods treat SNNs as recurrent dynamical systems and apply BPTT, typically using surrogate gradients that replace the intractable spike derivative with smooth approximations. Recent extensions learn surrogate shapes or widths to mitigate gradient vanishing and instability while improving convergence [117–120].

Unsupervised methods—most prominently STDP—implement biologically plausible synaptic adaptation driven by spike-timing correlations, though they often require additional supervisory or architectural signals to reach competitive accuracy [51,121].

Reinforcement learning leverages reward signals to optimize spiking policies in interactive or sequential tasks, whereas *hybrid* approaches combine supervision, self-organization, and reward modulation to exploit their respective strengths.

Finally, *ANN-to-SNN conversion* transfers pre-trained ANN weights to spiking equivalents, circumventing non-differentiability during training and enabling efficient deployment on neuromorphic hardware [16,18].

No single paradigm dominates across all applications; the optimal choice depends on the target task's accuracy–latency–energy trade-offs, biological plausibility, data regime, and hardware constraints. A consolidated summary of representative algorithms, their mechanisms, advantages, and limitations is presented in Table 3.

2.4.1. Supervised Learning

Supervised learning trains SNNs on labeled datasets by associating each input with a desired output (e.g., class labels or target spike statistics). Because spikes are discrete events,

standard backpropagation is hindered by the non-differentiability of spike generation. Therefore, practical methods unroll the network in time—analogous to recurrent neural networks—and optimize losses defined on spike-based readouts, such as spike counts, rates, or latencies, while replacing the non-differentiable spike function with a smooth *surrogate gradient* [16].

This approach enables gradient-based optimization using BPTT and its variants, achieving high accuracy on both classification and regression tasks. However, it incurs increased memory and computational demands from temporal unrolling and requires careful tuning of surrogate functions, membrane time constants, and firing thresholds.

SpikeProp

SpikeProp [122] was one of the earliest backpropagation-style algorithms for SNNs, extending temporal error backpropagation to learn precise spike times—particularly useful for temporal pattern recognition. The synaptic update is

$$\Delta w_{ij} = -\eta \sum_d \frac{\partial E}{\partial t_i^d} \frac{\partial t_i^d}{\partial w_{ij}}, \tag{35}$$

where η is the learning rate, E measures the mismatch between actual and desired spike times t_i^d, and $\partial t_i^d / \partial w_{ij}$ is the spike-time sensitivity. SpikeProp showed that backpropagation concepts can be carried into the temporal domain, enabling accurate nonlinear classification with fewer units than purely rate-based networks and foreshadowing modern surrogate-gradient and locality-aware rules.

SuperSpike

SuperSpike [123] generalizes SpikeProp by introducing surrogate gradients and eligibility traces for deep, multilayer SNNs trained on spatiotemporal inputs. The update takes the form

$$\Delta w_{ij} = \eta \int_{t_b}^{t_{b+1}} e_i(t) \left[\alpha * \left(\sigma'(U_i(t)) \cdot (\epsilon * S_j)(t) \right) \right] dt, \tag{36}$$

with error signal $e_i(t)$, surrogate derivative $\sigma'(U_i(t))$ of the spike function, and presynaptic trace $(\epsilon * S_j)(t)$. SuperSpike enables end-to-end training despite spike non-differentiability by combining gradient approximation with local eligibility.

SLAYER

SLAYER (Spike Layer Error Reassignment in Time) [119] tackles temporal credit assignment by backpropagating errors through time and reassigning them to causal spike events. The generic update is

$$\Delta w = -\eta \sum_t \frac{\partial E}{\partial s(t)} \frac{\partial s(t)}{\partial w}, \tag{37}$$

where $\partial s(t) / \partial w$ is a surrogate gradient of the spike train concerning the weight. SLAYER is effective for tasks that hinge on precise spike timing, such as sequence prediction and temporal classification.

EventProp

EventProp [124] computes *exact* gradients by explicitly handling derivative discontinuities at spike times via an adjoint method, avoiding surrogate approximations. The loss is

$$L = l_p(t_{\text{post}}) + \int_0^T l_V(V(t), t)\, dt, \tag{38}$$

and its gradient w.r.t. a synapse w_{ji} is

$$\frac{dL}{dw_{ji}} = -\tau_{\text{syn}} \sum_{\text{spikes from } i} (\lambda_I)_j, \tag{39}$$

where $(\lambda_I)_j$ is the adjoint for the synaptic current and τ_{syn} a synaptic constant. The event-driven treatment lowers memory and compute overheads, making EventProp appealing for neuromorphic execution.

2.4.2. Unsupervised Learning in SNNs

Unsupervised learning discovers structure without labels, typically through local rules that exploit spike timing. The most prominent is STDP, with numerous extensions improving stability, hardware efficiency, and biological plausibility.

STDP

Classical STDP [51,121] modifies synapses by the relative timing of pre/post-spikes:

$$\Delta w = \begin{cases} A^+ \exp\left(\frac{t_{\text{pre}} - t_{\text{post}}}{\tau^+}\right), & t_{\text{pre}} \leq t_{\text{post}}, \\ A^- \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau^-}\right), & t_{\text{pre}} > t_{\text{post}}, \end{cases} \tag{40}$$

with (A^+, A^-) the potentiation/depression amplitudes and (τ^+, τ^-) the time constants.

Adaptive STDP (aSTDP)

aSTDP extends classical STDP by dynamically adjusting the parameters governing synaptic updates, thereby improving stability and robustness—particularly in neuromorphic hardware with limited synaptic resolution [125]. The variant proposed by Gautam and Kohno simplifies the exponential weight-update function into a rectangular learning window, improving hardware efficiency. The update rule is as follows:

$$\Delta w_j = \begin{cases} +1 \text{ bit}, & \text{if } t_j \leq t_i \text{ and } t_i - t_j < t_{\text{pre}} \text{ (LTP)}, \\ -1 \text{ bit}, & \text{if } t_j > t_i \text{ and } t_j - t_i < t_{\text{post}} \text{ (LTD)}, \end{cases} \tag{41}$$

where t_{pre} is the maximum delay between a presynaptic spike followed by a postsynaptic spike that induces *long-term potentiation* (LTP); t_{post} maximum delay between a postsynapticsynaptic spike followed by a presynaptic spike that induces *long-term depression* (LTD); adaptively increased during learning, t_i and t_j postsynaptic and presynaptic spike times, respectively.

Alternative aSTDP formulations, such as that proposed by Li et al. [126], enhance biological plausibility using perturbation-based approximations of postsynaptic derivatives. These approaches facilitate biologically realistic, local unsupervised learning without global supervision.

Multiplicative STDP

Multiplicative STDP [127] incorporates the current synaptic weight into the learning rule, improving biological plausibility and preventing unbounded weight growth or decay. The update dynamics are as follows:

$$\Delta w = A^+ \cdot x_{\text{pre}} \cdot \delta_{\text{post}} - A^- \cdot x_{\text{post}} \cdot \delta_{\text{pre}}, \tag{42}$$

$$\frac{dx_{\text{pre}}}{dt} = -\frac{x_{\text{pre}}(t)}{\tau^+} + \delta(t), \tag{43}$$

$$\frac{dx_{\text{post}}}{dt} = -\frac{x_{\text{post}}(t)}{\tau^-} + \delta(t), \tag{44}$$

where A^+ and A^- Learning rate parameters for potentiation and depression,x_{pre} and x_{post} Pre- and postsynaptic spike traces,δ_{pre} and δ_{post} Indicators for pre- and postsynaptic spike events,τ^+ and τ^- Time constants governing trace decay.

Triplet STDP

Triplet STDP [128] extends pair-based STDP by incorporating triplet interactions, such as two presynaptic spikes and one postsynaptic spike or vice versa. This extension accounts for the frequency dependence of synaptic changes, where high-frequency spiking produces stronger potentiation or depression due to cumulative intracellular calcium effects. The update rule is as follows:

$$\Delta w = \eta \left(x_{\text{pre}} - x_{\text{tar}}\right) \left(w_{\max} - w\right) u, \tag{45}$$

where η learning rate, x_{pre} is the presynaptic trace value, and x_{tar} is the target presynaptic trace at the time of a postsynaptic spike. $w_{\max}$ Maximum allowable synaptic weight,w current synaptic weight,u modulation term controlling dependence on the current weight. By integrating multi-spike interactions, triplet STDP captures nonlinear dependencies and better reflects the complexity of biological synaptic plasticity.

2.4.3. Reinforcement Learning in SNNs

Reinforcement Learning modulates plasticity with evaluative feedback, enabling SNNs to learn action policies from rewards—well-suited to closed-loop, real-time settings.

R-STDP

R-STDP [52] extends classical STDP by incorporating a reward signal $R(t)$ that modulates synaptic weight changes according to whether the received feedback is positive or negative. The weight update rule is as follows:

$$\Delta w = R(t) \cdot \begin{cases} A^+ \exp\left(\frac{-\Delta t}{\tau^+}\right), & \text{if } \Delta t > 0, \\ A^- \exp\left(\frac{\Delta t}{\tau^-}\right), & \text{if } \Delta t < 0, \end{cases} \tag{46}$$

where $R(t)$ Reward signal at time t, Δw Change in synaptic weight, A^+ and A^- Learning rates for potentiation and depression, respectively, $\Delta t = t_{\text{post}} - t_{\text{pre}}$ Time difference between postsynaptic and presynaptic spikes, τ^+ and τ^- Time constants defining the potentiation and depression windows.

By integrating temporal spike relationships with reward feedback, R-STDP enables adaptive learning in environments where the network's behavior must evolve based on environmental cues, such as maze navigation or game playing, where purely supervised or unsupervised methods may be less effective.

ReSuMe (Rewarded Subspace Method)

ReSuMe [129] combines supervised learning principles with reinforcement signals, enabling synaptic weight adjustments that align target outputs with environmental rewards. The update rule is as follows:

$$\Delta w = \eta \cdot (r - y) \cdot x \tag{47}$$

where Δw change in synaptic weight, η learning rate, r reward signal, y actual neuron output, and x input signal.

By leveraging reinforcement-modulated weight updates, ReSuMe bridges the gap between biologically inspired learning and computational efficiency, making it suitable for tasks requiring both supervised target guidance and environmental adaptation.

Eligibility Propagation (e-prop)

e-prop [130] provides a biologically plausible alternative to backpropagation by using eligibility traces to capture the influence of past synaptic activity on current outputs. This

approach is particularly effective for tasks involving long temporal dependencies. The eligibility trace is updated according to the following:

$$E_{t+1} = \lambda E_t + \frac{\partial y_t}{\partial w} \tag{48}$$

where E_t Eligibility trace at time t, λ trace decay factor, y_t output at time t, and w synaptic weight.

e-prop allows error information to propagate backward through time without storing the entire history of network states, significantly reducing memory requirements while maintaining the ability to learn both synaptic weights and temporal dependencies in parallel.

2.4.4. Hybrid Learning Paradigms

Hybrid schemes combine global error signals with local spike-based plasticity or reward, aiming for better accuracy–efficiency trade-offs and hardware compatibility.

(SSTDP)

SSTDP [131] is a supervised learning rule designed to enhance training efficiency and accuracy in SNNs by uniting backpropagation-based global optimization with the local temporal dynamics of STDP. This hybrid approach bridges the gap between gradient-based learning and biologically plausible spike-based plasticity.

The synaptic weight update is as follows:

$$\Delta w_{ij}^l = -\alpha \frac{\partial E}{\partial w_{ij}^l}, \quad \text{where} \quad \frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial t_j^l} \frac{\partial t_j^l}{\partial w_{ij}^l}, \tag{49}$$

where α Learning rate, $\frac{\partial E}{\partial t_j^l}$ Error signal propagated back to the firing time of the postsynaptic neuron, $\frac{\partial t_j^l}{\partial w_{ij}^l}$ Partial derivative of postsynaptic firing time with respect to the synaptic weight.

The derivative $\frac{\partial t_j^l}{\partial w_{ij}^l}$ is defined as:

$$\frac{\partial t_j^l}{\partial w_{ij}^l} = \begin{cases} \epsilon_1 \left(e^{-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau}} - \delta \right) (w_{\max} - w)^\mu, & t_{\text{post}} > t_{\text{pre}}, \\ \epsilon_2 \left(e^{-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau}} - \delta \right) (w_{\max} - w)^\mu, & t_{\text{post}} < t_{\text{pre}}, \end{cases} \tag{50}$$

where t_{pre}, t_{post} Pre- and postsynaptic firing times,ϵ_1, ϵ_2 Scaling factors for potentiation and depression, τ Time constant for exponential decay of STDP effects,δ Temporal window parameter defining effective STDP update intervals,$w_{\max}$ Maximum allowable synaptic weight,w Current synaptic weight,μ Weight-dependence factor controlling the influence of $w_{\max}$ on update magnitude.

SSTDP achieves reduced inference latency, lower computational overhead, and improved energy efficiency for neuromorphic deployment by combining global error feedback with local spike-timing-based plasticity.

ANN-to-SNN Conversion

ANN-to-SNN conversion reduces the need to train SNNs from scratch by transforming pre-trained ANNs into equivalent spiking architectures [132,133]. This approach maps continuous neuron activations and synaptic weights to spike-based counterparts, minimizing performance degradation while preserving learned representations [31,134–137].

The general process involves:

1. Training a conventional ANN using backpropagation.
2. Converting neuron activations to spike rates or spike times.

3. Adjusting weights, thresholds, and normalization parameters to match the target SNN framework.

Two main strategies exist:

- *Rate-based conversion:* Maps ANN activations to SNN firing rates using normalization and threshold adaptation, ensuring the spike rate approximates the ANN output [132,133].
- *Temporal coding conversion:* Encodes information in spike timing to capture temporal patterns, reducing latency and improving performance on dynamic datasets [89].

Enhancements such as Max Normalization for pooling layers and reset-by-subtraction mechanisms further mitigate performance loss, maintaining high accuracy on datasets like MNIST [138] and CIFAR-10 [139] while improving energy efficiency [136,140]. ANN-to-SNN conversion thus combines the mature training capabilities of ANNs with the deployment advantages of SNNs on neuromorphic hardware.

Table 3. Overview of learning algorithms in SNNs, summarizing their paradigms, typical applications, computational complexity, biological plausibility, key advantages, and main challenges.

Algorithm Name	Learning Paradigm	Main Applications	Complexity	Biological Plausibility	Advantages	Challenges
SpikeProp [122]	Supervised	Temporal pattern recognition	Low	Low	Enables precise spike timing learning; first event-based backpropagation for SNNs	Limited to shallow networks; affected by spike non-differentiability
SuperSpike [123]	Supervised	Temporal pattern recognition; deep SNN training	Medium	Medium	Uses surrogate gradients; enables multilayer training	Requires careful surrogate design; added computational cost
SLAYER [119]	Supervised	Complex temporal data; sequence prediction	High	Medium	Addresses temporal credit assignment; handles sequences well	Computationally intensive; complex to implement
EventProp [124]	Supervised	Exact gradient computation; neuromorphic hardware	High	High	Computes exact gradients; efficient for event-driven processing	Complex discontinuity handling; implementation challenges
STDP [51,121]	Unsupervised	Pattern recognition; feature extraction	Low	High	Biologically plausible; local weight updates	Limited scalability; lower accuracy for large-scale tasks
aSTDP [125,126]	Unsupervised	Adaptive feature learning	Medium	High	Dynamically adapts learning parameters; robust	Parameter tuning complexity; additional computation
Multiplicative STDP [127]	Unsupervised	Weight updates scaled by current weight; prevents unbounded growth/decay	High	Medium	Improves biological plausibility; stabilizes learning	Requires careful parameter tuning
Triplet-STDP [128]	Unsupervised	Frequency-dependent learning	Medium	Medium	Captures multi-spike interactions; models frequency effects	Complex spike attribution; higher computation
R-STDP [52]	RL	Adaptive learning; decision making	Medium	High	Integrates reward signals; adaptive to reinforcement tasks	Requires carefully designed reward schemes
ReSuMe [141]	RL	Temporal precision learning	Medium	High	Combines supervised targets with reinforcement feedback	Dependent on reward design; non-gradient-based
e-prop [130]	RL	Temporal dependencies; complex dynamics	High	High	Tracks synaptic influence with eligibility traces	Computationally intensive; eligibility tracking complexity
SSTDP [131]	Hybrid	High temporal precision; visual recognition	Medium	High	Merges backpropagation and STDP; energy-efficient	Requires precise timing data; integration complexity
ANN-to-SNN Conversion [134,135]	Hybrid	Neuromorphic deployment	Medium	Low	Leverages pre-trained ANNs; fast deployment	Accuracy loss in conversion; parameter mapping issues

2.5. Evolution of Supervised Learning in SNNs and Broader Context

Supervised training for SNNs has evolved from early adaptations of backpropagation to methods that explicitly handle spike timing and discontinuities. Initial work by [142] applied backpropagation-like updates with adaptive learning rates, demonstrating viability on small datasets (e.g., Iris) but highlighting the difficulty of optimizing through non-differentiable spikes. ReSuMe [143] bridged supervised objectives with STDP-like timing rules, while perceptron-style temporal learners [144] and multi-spike gradient approaches [145] improved efficiency and supported richer temporal patterns. Meta-heuristic variants—such as SpikeProp with PSO [146]—achieved gains in accuracy and convergence, and biologically inspired models like the tempotron [147] underscored the utility of precise temporal coding.

A major inflection point came with surrogate gradients, which replace the intractable spike derivative with smooth approximations, enabling BPTT in deep SNNs [16,148]. Temporal-coding supervision via direct gradient descent [76] aligned learning with event-driven dynamics; SuperSpike [123] combined surrogate gradients with eligibility traces for multilayer training; and SLAYER [119] reassigned errors across space and time to address temporal credit assignment. EventProp [124] later introduced exact gradients for continuous-time SNNs using an adjoint-state formulation, eliminating the need for surrogates. Hybrid rules such as SSTDP [131] merged global error signals with local timing windows, while system-level innovations—including single-spike hybrid input encodings [149], threshold-dependent batch normalization for deep SNNs [150], and spiking Transformers [151,152]—expanded the accuracy–latency–efficiency frontier on large-scale benchmarks.

In parallel, unsupervised and reinforcement-based paradigms emphasize locality and energy efficiency. Classical and adaptive STDP variants [51,121,125,126] excel at feature discovery but typically benefit from auxiliary supervision to achieve state-of-the-art accuracy. R-STDP and e-prop [52,130] enable closed-loop learning with reduced memory requirements. ANN-to-SNN conversion [31,132–137] leverages mature ANN training pipelines, mapping activation rates or spike timings to spiking equivalents for neuromorphic deployment. These methods sustain strong performance on MNIST [138] and CIFAR-10 [139] while achieving improved efficiency [18,140]. Energy-aware objectives and normalization techniques (e.g., Spike-Norm, rate normalization) further stabilize training and reduce power consumption [153,154].

Overall, supervised approaches deliver the highest accuracy but at increased computational and energy cost; unsupervised and reinforcement learning methods prioritize locality and efficiency, often at the expense of scalability; and hybrid or conversion-based frameworks increasingly reconcile these objectives. Continued progress will depend on integrating temporal coding, normalization, and hardware-aware optimization to achieve energy-efficient, high-performance SNNs at scale.

The preceding sections reviewed encoding strategies, neuron models, and learning paradigms that distinguish SNNs from ANNs, emphasizing their potential for event-driven efficiency alongside persistent challenges of trainability, robustness, and hardware compatibility. Building on these foundations, the following section presents a controlled empirical study that couples matched SNN and ANN architectures with standardized encodings and training procedures. The aim is to quantify accuracy–energy trade-offs under comparable conditions and to derive actionable design guidelines. Accordingly, the next section details the datasets, model backbones, encoding and neuron configurations, training setup, and evaluation metrics used in our analysis.

3. Materials and Methods

This section describes the datasets and preprocessing steps, model architectures and spiking configurations, training and inference procedures, and evaluation metrics used in the comparative study.

3.1. Experimental Design

Objective. The goal is to evaluate the predictive efficacy (accuracy) and energy efficiency (power consumption) of SNNs relative to architecturally matched ANNs for image classification tasks. To ensure fair comparison, each SNN is paired with an ANN using the same network backbone. Experiments are conducted on two standard benchmarks—MNIST and CIFAR-10. SNN variants span multiple combinations of encoding schemes and neuron models under supervised learning, along with different surrogate-gradient functions. Energy per inference is estimated following [155] using the `KerasSpiking` framework [156].

We design and train both shallow and deep architectures suited to MNIST and CIFAR-10:

- *Fully connected network (FCN):* two hidden fully connected layers and a classification head, applied to MNIST; approximately 118,288 trainable parameters.
- *Deep convolutional network (VGG7):* five convolutional layers, two max-pooling layers, one hidden fully connected layer, and a classifier, applied to CIFAR-10; approximately 548,554 trainable parameters.

Each architecture is instantiated with diverse neuron models and encoding schemes to generate spike trains from input data. The encoding and neuron model details are summarized in Tables 1 and 2.

- *Neuron models:* IF, LIF, ALIF, CUBA, ΣΔ, RF, RF–IZH, EIF, and AdEx.
- *Encoding schemes:* Direct encoding, rate encoding, temporal TTFS, sigma–delta (ΣΔ) encoding, burst coding, PoFC, and R–NoM. Each method transforms continuous pixel intensities into discrete spike trains distributed across specific time steps.

The number of time steps is limited to maintain tractable inference: $T \in \{4, 6, 8\}$ for MNIST and $T \in \{2, 4, 6\}$ for CIFAR-10, given the latter's higher complexity and computational cost. Every SNN variant is trained and evaluated across these configurations, and performance is assessed using two primary metrics:

- *Predictive efficacy (accuracy):* proportion of correctly classified samples.
- *Energy efficiency (power consumption):* theoretical power usage estimated per inference.

The evaluation procedure includes:

- Training models on MNIST and CIFAR-10 under varying time-step configurations;
- Encoding input images using the predefined schemes;
- Measuring predictive accuracy and estimating energy consumption; and
- Comparing SNN performance against equivalent ANN baselines to quantify accuracy–energy trade-offs.

These experiments identify SNN configurations that best balance accuracy and energy consumption, providing insights for both research and neuromorphic deployment.

3.2. Data Collection and Preprocessing

The evaluation uses MNIST [138] and CIFAR-10 [139]. *MNIST* contains 70,000 grayscale images (28×28 pixels) of handwritten digits (60,000 for training and 10,000 for testing across ten classes). Images are normalized to mean 0.5, standard deviation 0.5, and range $[-1, 1]$, then converted into spike trains using multiple encoding schemes with time steps $T \in \{4, 6, 8\}$. *CIFAR-10* comprises 60,000 RGB images (32×32 pixels) across ten

classes (50,000 training and 10,000 testing). Data augmentation includes random cropping (padding of four) and horizontal flipping, followed by per-channel normalization with means $[0.4914, 0.4822, 0.4465]$ and standard deviations $[0.2023, 0.1994, 0.2010]$. Normalized images are encoded as spike trains with $T \in \{2, 4, 6\}$. Encoding methods follow Table 1, and neuron models (IF, LIF, ALIF, CUBA, ΣΔ, RF, RF–IZH, EIF, AdEx) are implemented as listed in Table 2.

3.3. Implementation Frameworks and Tools

All preprocessing, encoding, and model development were implemented in Python using specialized SNN frameworks—*Lava* [115], *SpikingJelly* [157], and *Norse* [158]—integrated within the *PyTorch* environment. These toolchains supported both MNIST (FCN) and CIFAR-10 (VGG7) experiments.

3.3.1. Lava

Intel's neuromorphic framework provides modular neuron and synapse models, learning rules, and deployment utilities. The Lava-DL module includes SLAYER 2.0 for efficient surrogate-gradient training, while NetX facilitates compilation to neuromorphic targets such as Loihi [42]. It supports rate and temporal coding, ANN→SNN conversion (e.g., Bootstrap), PyTorch interoperability, and HDF5-based, platform-independent model exchange [115].

3.3.2. SpikingJelly

A PyTorch-native SNN library offering IF/LIF neurons, advanced surrogate gradients (e.g., ATan), and multiple encoding options (rate, temporal, and phase). CuPy-based GPU acceleration enables efficient large-scale training [157].

3.3.3. Norse

A lightweight PyTorch extension emphasizing biologically realistic neuron models (e.g., AdEx) and efficient simulation. It supports surrogate-gradient training (e.g., SuperSpike), just-in-time compilation, and GPU acceleration [158].

3.3.4. PyTorch

Used to construct both ANN and SNN backbones, manage training loops, and optimize parameters (Adam optimizer). It provides standard deep-learning infrastructure while integrating seamlessly with SNN-specific modules.

Together, these frameworks enabled efficient experimentation, reproducibility, and fair comparison across ANN and SNN configurations incorporating diverse neuron models and encoding strategies.

3.4. Neural Network Architectures

In this study, both ANNs and SNNs were employed on the MNIST and CIFAR-10 datasets to enable direct, architecture-level comparisons. The ANN models were implemented in *PyTorch*, using an *FCN* (two fully connected layers with 128 units, 5% dropout, and ReLU activations) for MNIST, and a *VGG7*-inspired convolutional network (multiple convolutional layers interleaved with batch normalization, ReLU, max-pooling, and a 1024-unit fully connected layer with 20% dropout) for CIFAR-10. MNIST images were normalized to a mean of 0.5 and a standard deviation of 0.5. CIFAR-10 images underwent data augmentation—random cropping with 4-pixel padding and horizontal flipping—followed by per-channel normalization using dataset-specific statistics.

The SNN counterparts were developed using the *Lava*, *SpikingJelly*, and *Norse* frameworks integrated with PyTorch. These models adopted the same architectural depth and width as their ANN equivalents to ensure a fair comparison. Each SNN variant incorpo-

rated a specific neuron model and encoding scheme as described in Section 3.1. The detailed configurations of the two architectures are summarized in Table 4, and their structural overviews are illustrated in Figures 7 and 8.

FCN architecture. This shallow network, containing approximately 118,288 parameters, processes spike trains derived from the 784 input pixels of MNIST. It consists of two fully connected layers with 128 spiking neurons each (5% dropout), followed by an output layer of ten neurons corresponding to the digit classes.

VGG7 architecture. This deeper network, with a total of approximately 548,554 parameters, is based on the VGG family design and operates on CIFAR-10 images of size 32×32 pixels in RGB format. It comprises sequential convolutional layers with varying strides and 64 or 128 filters, interspersed with max-pooling operations. The extracted feature maps are flattened and passed through a 1024-neuron fully connected layer (20% dropout) before the final output layer of ten neurons corresponding to the image classes.

Table 4. ANN and SNN architectures for FCN and VGG7.

Layer	**FCN Architecture**	**VGG7 Architecture**
Input Layer	784 input neurons	3 channels, 32×32 pixels
Layer 1	Dense, 128 neurons, Dropout($p = 0.05$)	Conv, 64 filters, 3×3 kernel, stride 1, padding 1
Layer 2	Dense, 128 neurons, Dropout($p = 0.05$)	Conv, 64 filters, 3×3 kernel, stride 2, padding 1
–	Output, 10 neurons	Max Pooling, 2×2 kernel, stride 2
Layer 3	—	Conv, 128 filters, 3×3 kernel, stride 1, padding 1
Layer 4	—	Conv, 128 filters, 3×3 kernel, stride 2, padding 1
Layer 5	—	Conv, 128 filters, 3×3 kernel, stride 2, padding 1
–	—	Max Pooling, 2×2 kernel, stride 2
Flatten Layer	—	Flatten feature maps
Layer 6	—	Dense, 1024 neurons, Dropout($p = 0.2$)
Layer 7	Output, 10 neurons	Output, 10 neurons
Special Components	Weight Normalization, Weight Scaling	Weight Normalization, Weight Scaling
Total Parameters	∼118,016	∼548,554
Frameworks Used	PyTorch for ANNs; Lava, Norse, SpikingJelly for SNNs	PyTorch for ANNs; Lava, Norse, SpikingJelly for SNNs

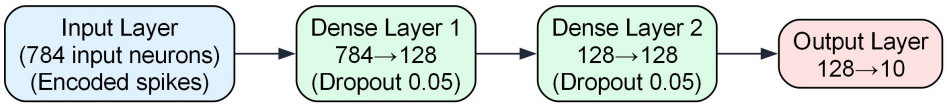


Figure 7. Fully connected network (FCN) architecture for the MNIST dataset.

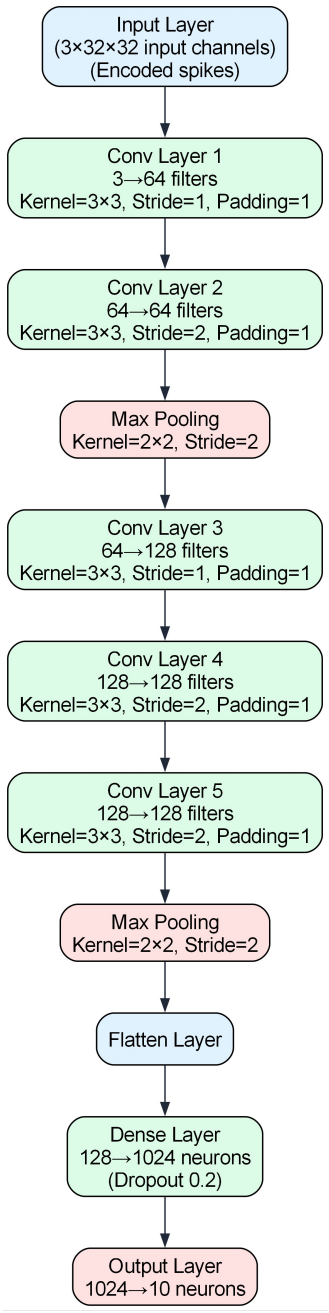


Figure 8. VGG7 Network architecture for the CIFAR-10 dataset. Abbreviations used K for kernel size, S for stride, and P for padding.

3.5. Training Configuration and Procedures

All experiments were conducted on Google Colab with GPU acceleration. ANNs were implemented in *PyTorch*, while SNNs were trained using *PyTorch* in conjunction with the *Lava*, *SpikingJelly*, and *Norse* frameworks, employing surrogate-gradient backpropagation through time (BPTT). A consistent training–validation split and identical data loaders were used across all models, encoding schemes, and neuron types to ensure fair, reproducible comparisons for both the FCN and VGG7 architectures.

Hyperparameters: Unless stated otherwise, all models used the Adam optimizer (learning rate 0.001, weight decay 1×10^{-5}) with `CrossEntropyLoss`. FCN models were trained for 100 epochs, and VGG7 models for 150 epochs, with a batch size of 64 in all cases. Dropout was 5% for FCN and 20% for VGG7, with batch normalization applied where appropriate. ANNs used ReLU activations and the default PyTorch weight initialization.

SNNs relied on spiking activations, with neuron parameters matched across datasets except for threshold voltage: FCN used $V_{\text{th}} = 1.25$, and VGG7 used $V_{\text{th}} = 0.5$. Shared SNN settings were as follows: current decay 0.25, voltage decay 0.03, tau gradient 0.03, scale gradient 3, and refractory decay 1. A consolidated summary of parameters is provided in Table 5.

Preprocessing and encoding: ANN pipelines applied standard normalization (and augmentation for CIFAR-10). For SNNs, inputs were converted into spike trains using the evaluated encoding schemes (e.g., rate, temporal, and $\Sigma\Delta$) and processed by the selected neuron models.

Training loop: Each iteration comprised a forward pass, temporal aggregation of SNN outputs (time-averaged logits or readouts), computation of cross-entropy loss, gradient backpropagation (BPTT with surrogate gradients for SNNs, standard backpropagation for ANNs), and parameter updates using Adam. This unified training procedure ensured consistent optimization across all ANN and SNN configurations.

Table 5. Hyperparameters for ANN and SNN models on FCN and VGG7 architectures.

Parameter	ANN (FCN/VGG7)	SNN (FCN)	SNN (VGG7)
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Weight decay	1×10^{-5}	1×10^{-5}	1×10^{-5}
Loss function	Cross-Entropy	CrossEntropyLoss	CrossEntropyLoss
Number of epochs	FCN: 100; VGG7: 150	100	150
Batch size	64	64	64
Dropout rate	FCN: 5%; VGG7: 20%	5%	20%
Activation function	ReLU	—	—
Weight initialization	PyTorch defaults	—	—
Neuron parameters	—	Threshold (V_{th}): 1.25 Current decay: 0.25 Voltage decay: 0.03 Tau gradient: 0.03 Scale gradient: 3 Refractory decay: 1	Threshold (V_{th}): 0.5 Current decay: 0.25 Voltage decay: 0.03 Tau gradient: 0.03 Scale gradient: 3 Refractory decay: 1

3.6. Evaluation Metrics

Two metrics were used for evaluation: (i) classification accuracy and (ii) a per-inference energy estimate.

Accuracy: The fraction of correctly classified test samples (single-label, top-1), computed from logits using `CrossEntropyLoss`. For SNNs, outputs were temporally aggregated (e.g., time-averaged logits or spike counts across T steps) before applying the final `argmax`. Identical data splits, preprocessing, batch size, and, for SNNs, time step settings were used across FCN and VGG7 models to ensure fairness.

Energy estimate: The energy per inference was estimated using the KerasSpiking methodology [155,156], which models total energy as the sum of synaptic (multiply-accumulate, MAC) operations and neuron-state updates:

$$E_S = E_o \, S, \tag{51}$$

$$E_N = E_u \, U, \tag{52}$$

$$E_{\text{total}} = E_S + E_N, \tag{53}$$

where E_o and E_u are hardware-dependent energy constants, and S and U represent operation counts collected during inference. For ANNs, S is the number of MACs and U the number of activation evaluations; for SNNs, S counts synaptic events (spike transmissions) and U counts membrane or state updates across T time steps. Constants E_o and E_u were instantiated using published hardware values (e.g., GPU figures reported in [155], with typical Titan-class estimates $E_o \approx E_u \approx 0.3\,\text{nJ}$), and E_{total} is reported in nJ/inference.

Rationale and limitations of KerasSpiking: KerasSpiking integrates with TensorFlow/Keras to provide energy estimates by pairing measured or literature-based device constants with operation counts [156]. Prior studies have applied this approach to benchmark SNNs against ANNs [155,159,160]. However, these constants represent generalized device characteristics rather than chip-specific, peer-validated calibrations; thus, they serve as approximations sensitive to architecture, memory traffic, and data movement. In this work, KerasSpiking is used as a transparent, hardware-pluggable proxy—appropriate for relative comparisons and adaptable to other platforms (e.g., neuromorphic chips such as Intel Loihi)—by substituting platform-specific per-operation constants. It should not be regarded as a definitive measure of absolute energy consumption.

3.7. Algorithms

This subsection enumerates the exact encodings, neuron models, and learning rules used in the experiments. Unless otherwise specified, SNNs employed surrogate-gradient BPTT. Time steps were set to $T \in \{4, 6, 8\}$ for FCN and $T \in \{2, 4, 6\}$ for VGG7.

Encodings used:

- Direct input: the image is duplicated across T time steps without spike synthesis.
- Rate (Poisson): intensities $\in [0, 1]$ mapped to firing rates (maximum 100 Hz); Bernoulli sampling applied at each step.
- TTFS: a single spike latency monotonically mapped from intensity across T bins.
- $\Sigma\Delta$: dead-zone delta encoder with feedback; spikes emitted when $|\Delta x| > \theta$ (here $\theta = 0.5$), reconstructed by $\pm\theta$.
- R–NoM: rank-modulated top-N spiking from sorted intensities; N tuned by validation.
- PoFC: phase derived from normalized intensity over a $T_{\text{osc}} = T$ cycle.
- Burst: intensity-dependent bursts capped at $B_{\max}$ (selected via validation) within T.

Neuron models instantiated:

- IF and LIF (SpikingJelly; ATan surrogate gradient).
- ALIF (Lava; dynamic threshold or current adaptation).
- CUBA (Lava; current-based neuron).
- $\Sigma\Delta$ neuron (Lava; event-driven, error-based spiking).
- RF and RF–IZH (Lava; resonance and phase sensitivity).
- EIF and AdEx (Norse; exponential onset and adaptive dynamics).

Learning rules:

- SLAYER surrogate gradient (Lava/SLAYER 2.0).
- SuperSpike surrogate gradient (Norse).
- ATan surrogate function (SpikingJelly).

Algorithm 1: Training/evaluation for VGG7 on CIFAR-10 and FCN on MNIST with operation counting for energy estimation.

Data: VGG7/CIFAR-10 with $T \in \{2, 4, 6\}$; FCN/MNIST with $T \in \{4, 6, 8\}$
Result: Trained models, accuracy/loss, per-inference energy proxy
Define encodings $\mathcal{E}$ and neuron models $\mathcal{N}$; set lr, batch size, epochs;
foreach *dataset* $D \in \{CIFAR\text{-}10, MNIST\}$ **do**
  set T per D; load and preprocess D;
  foreach $t \in T$ **do**
    foreach $e \in \mathcal{E}$ **do**
      foreach $n \in \mathcal{N}$ **do**
        encode data with e over t steps; define architecture with neuron n;
        initialize params; reset energy counters $S \leftarrow 0$, $U \leftarrow 0$;
        for *epoch* $= 1$ **to** *epochs* **do**
          foreach *batch* (X, y) **do**
            forward pass (aggregate over time for SNNs); compute cross-entropy loss;
            backprop; update (Adam);
            accumulate operation counts: $S+=$ synaptic/MAC ops; $U+=$ neuron/state updates;
          end
          evaluate accuracy/loss on validation/test set;
        end
        save model and metrics; compute $E_{\text{total}}=E_o S + E_u U$;
      end
    end
  end
end

Notes: thresholds were dataset-specific ($V_{\text{th}} = 1.25$ for FCN/MNIST; 0.5 for VGG7/CIFAR-10). Shared SNN settings: current decay 0.25, voltage decay 0.03, tau gradient 0.03, scale gradient 3, refractory decay 1. Frameworks: PyTorch (ANNs); SpikingJelly/Lava/Norse (SNNs). Energy is reported via $E_{\text{total}} = E_o S + E_u U$ using hardware constants cited in Section 3.6.

4. Results

This section presents the outcomes of the experiments conducted to evaluate the predictive performance and energy efficiency of all SNN configurations in comparison with their ANN counterparts. Experiments were performed on the MNIST and CIFAR-10 datasets using different network architectures, encoding schemes, and neuron models. The primary metrics include classification accuracy, inference energy consumption, and computational efficiency, quantified in terms of synaptic operations (SynOps) and event sparsity.

4.1. Performance on the MNIST Dataset

A shallow FCN consisting of two dense layers was used for the MNIST experiments, as described in Section 3.1. The objective was to assess the predictive performance of different SNN configurations relative to the baseline ANN model. Various encoding schemes and neuron models were tested to investigate their influence on classification accuracy. Furthermore, the models were evaluated at different numbers of time steps (4, 6, and 8) to examine the impact of temporal dynamics on performance.

4.1.1. Classification Accuracy Results

Table 6 and Figure 9 summarize the maximum MNIST classification accuracies (in percent) obtained across neuron models, encoding schemes, and time steps ($T = 4, 6, 8$). The baseline ANN achieved an accuracy of 98.23%. For each neuron-encoding combination, the best-performing result is highlighted in the table. Figure 9 panels (a–c) present the corresponding bar plots for 4, 6, and 8 time steps, respectively.

Table 6. Maximum classification accuracies (%) on the FCN with MNIST dataset.

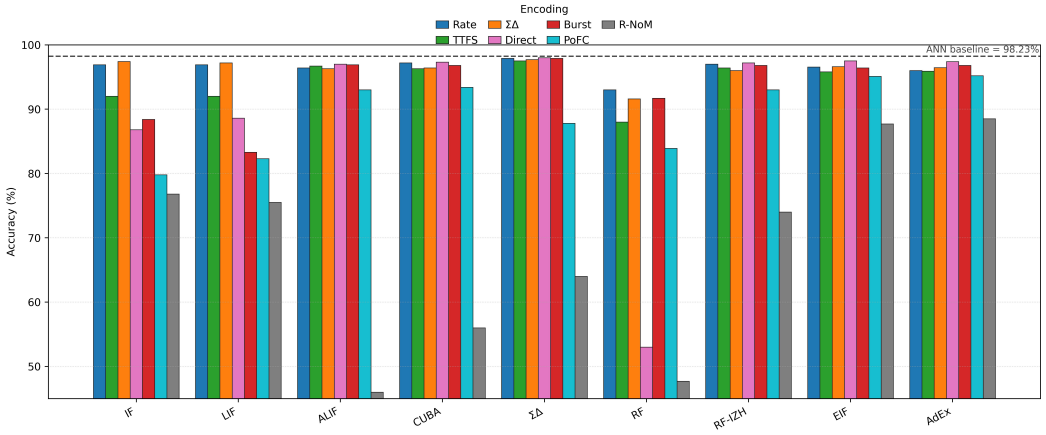
Neuron Type	Time Steps	Rate Encoding	TTFS	ΣΔ	Direct Coding	Burst Coding	PoFC	R-NoM
ANN (Baseline)	–				**98.23%**			
IF	8	97.20	86.60	**97.70**	78.90	80.00	71.20	72.20
	6	97.10	90.00	97.60	88.10	91.00	78.50	74.00
	4	96.90	92.00	97.40	86.80	88.40	79.80	76.80
LIF	8	97.00	86.50	97.40	81.90	80.06	72.10	71.00
	6	96.90	90.00	**97.50**	90.00	91.10	77.70	73.00
	4	96.90	92.00	97.20	88.60	83.30	82.30	75.50
ALIF	8	**97.30**	96.40	96.50	97.10	97.14	92.70	58.00
	6	96.60	96.10	96.30	97.00	97.00	92.90	57.00
	4	96.40	96.70	96.30	97.00	96.90	93.00	46.00
CUBA	8	**97.66**	97.10	97.40	97.60	79.50	94.10	68.00
	6	97.56	97.00	97.18	97.50	79.30	94.10	66.50
	4	97.20	96.30	96.40	97.30	96.80	93.40	56.00
ΣΔ	8	**98.10**	97.90	98.00	88.00	88.30	95.00	86.70
	6	97.80	97.50	98.00	88.50	97.90	87.40	86.00
	4	97.90	97.50	97.70	98.00	97.90	87.80	64.00
RF	8	93.00	86.80	90.05	**97.20**	92.20	85.80	52.00
	6	92.60	88.70	92.00	79.00	92.00	84.90	50.00
	4	93.00	88.00	91.60	53.00	91.70	83.90	47.70
RF-IZH	8	**97.70**	47.00	79.60	**97.70**	50.00	94.90	48.00
	6	97.55	87.70	97.40	97.70	97.40	94.80	69.70
	4	97.00	96.40	96.00	97.20	96.80	93.00	74.00
EIF	8	96.70	94.60	96.50	97.50	96.30	95.20	88.10
	6	96.20	94.90	96.20	**97.60**	96.00	94.50	86.70
	4	96.55	95.80	96.60	97.50	96.40	95.10	87.70
AdEx	8	96.50	94.70	96.50	97.40	96.70	95.40	89.50
	6	96.40	95.00	96.40	**97.50**	96.80	95.30	89.20
	4	96.00	95.90	96.44	97.39	96.80	95.20	88.50

Observation

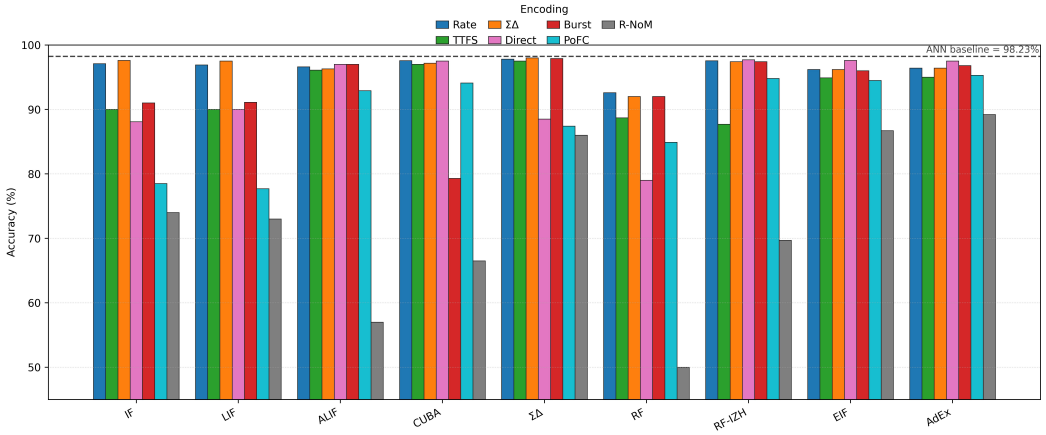
The results indicate that SNNs can achieve classification accuracies comparable to traditional ANNs on the MNIST dataset. The choice of neuron model and encoding scheme significantly influences performance.

- ΣΔ *neurons achieving the highest accuracy:* ΣΔ neurons attained the highest accuracy of 98.10% using rate encoding at eight time steps. They also performed strongly across other encodings, reaching 98.00% with ΣΔ encoding at both eight and six time steps. These results highlight their effectiveness in precise spike-based computation and suitability for tasks requiring accurate temporal processing and efficient encoding.
- *Adaptive neuron models showing competitive performance:* Adaptive neuron models such as ALIF and AdEx achieved competitive accuracy, particularly with direct and burst encodings. ALIF reached a maximum of 97.30% with rate encoding at eight time steps, while AdEx achieved 97.50% with direct encoding at six time steps. Their adaptation mechanisms enable better capture of temporal dynamics, providing a favorable trade-off between accuracy and computational efficiency.

(**a**) 4 time steps



(**b**) 6 time steps



(**c**) 8 time steps

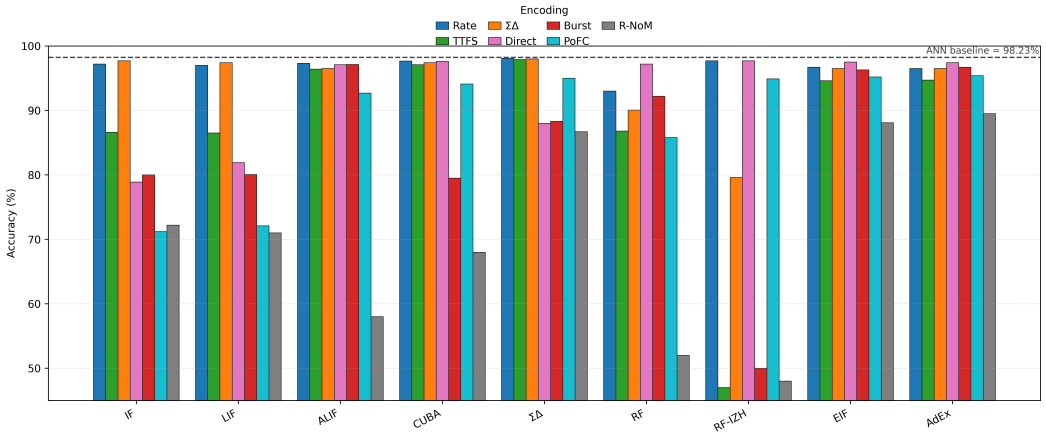


Figure 9. MNIST accuracy by neuron and encoding—4, 6, and 8 time steps.

- *Solid performance of simpler neuron models:* Simpler models such as IF and LIF also demonstrated strong performance with rate and ΣΔ encodings. IF achieved 97.70% accuracy with ΣΔ encoding at eight time steps, while LIF reached 97.50% at six time steps. Although slightly below the ANN baseline of 98.23%, these results confirm that simpler neuron models can still perform effectively when combined with appropriate encoding schemes.
- *Performance of RF neurons:* The standard RF neuron achieved lower accuracies compared with other models, with a maximum of 97.20% using direct encoding at eight

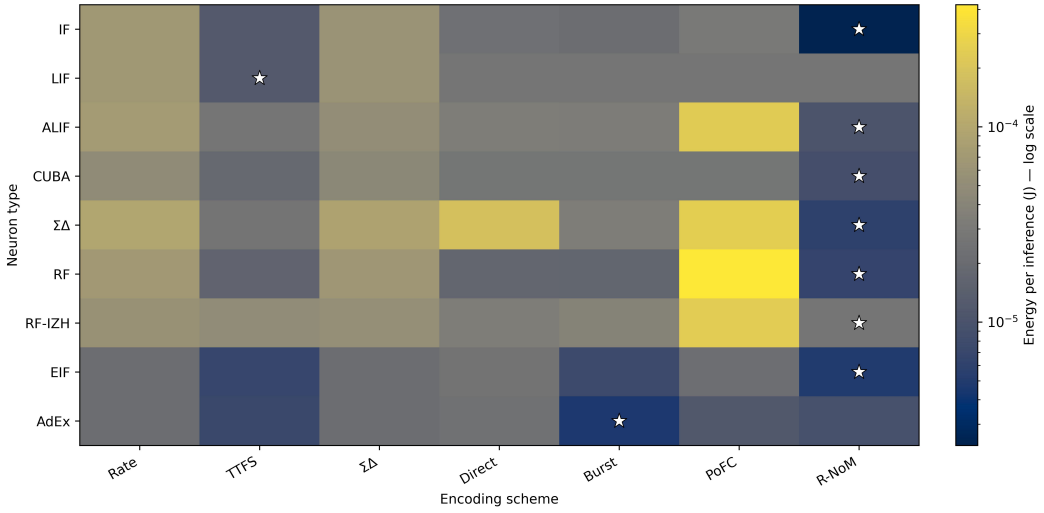
time steps. The RF–IZH variant performed better, reaching 97.70% with rate and direct encodings at eight time steps.

- *Robustness of CUBA neurons:* CUBA neurons achieved consistent results, with a maximum accuracy of 97.66% using rate encoding at eight time steps and 97.60% using direct encoding. Their stability across encoding schemes highlights robustness and adaptability.
- *EIF neurons' performance:* EIF neurons achieved 97.60% accuracy using direct encoding at six time steps, reflecting their capability to process direct input representations efficiently. Models such as CUBA and EIF that perform well across multiple encoding strategies demonstrate strong generalization and are suitable for applications requiring flexibility in encoding.
- *Effectiveness of encoding schemes:* Direct and ΣΔ encodings emerged as the most effective across multiple neuron types, frequently producing the highest accuracies. Burst encoding also performed well, particularly when paired with adaptive neuron models such as ALIF and AdEx. Temporal encodings (TTFS and PoFC) improved with increasing time steps but generally did not surpass direct or ΣΔ encoding.
- *Effect of time steps on accuracy:* Increasing the number of time steps generally improved accuracy for most neuron and encoding combinations, especially for temporal encodings such as TTFS and PoFC. However, models such as ALIF and ΣΔ neurons maintained high accuracy even with fewer time steps, demonstrating their efficiency in capturing temporal dynamics while minimizing computational and energy costs.
- *Advanced versus simpler neuron models:* Advanced neuron models such as ΣΔ, ALIF, and AdEx consistently achieved higher accuracies than simpler models like IF and LIF. This underscores the importance of mechanisms such as spike-frequency adaptation and precise spike timing in enhancing classification performance. Although some SNN configurations slightly lagged behind the ANN baseline, several achieved near-equivalent accuracy while maintaining superior energy efficiency and temporal processing capabilities.
- *Variable performance of R–NoM encoding:* R–NoM encoding showed variable performance across neuron models. ALIF neurons achieved a maximum of 58.00%, whereas EIF neurons reached 88.10%. This variability suggests that R–NoM's effectiveness depends heavily on the underlying neuron dynamics.

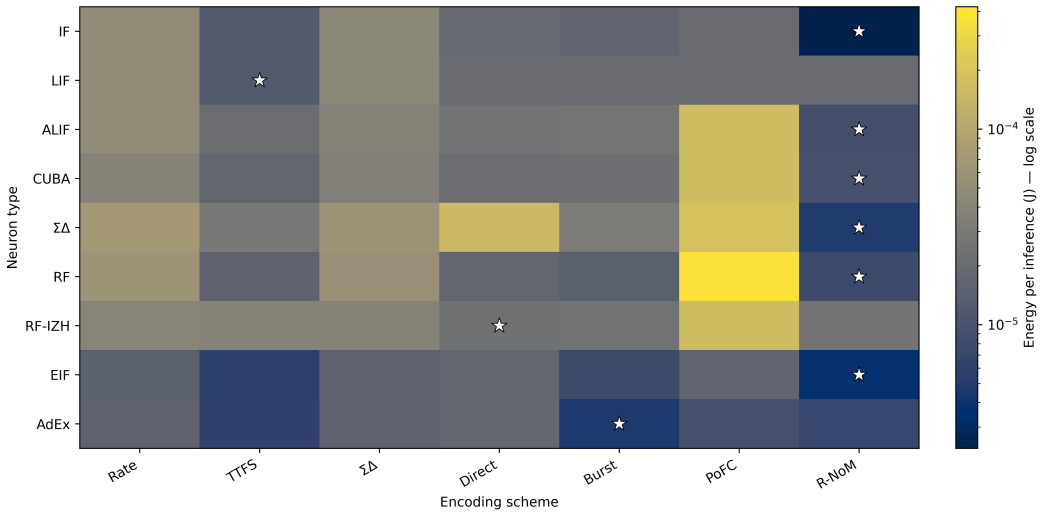
4.1.2. Energy Consumption

Table 7 reports the total energy per inference (joules) on MNIST across SNN configurations, measured on a GPU, with the ANN baseline consuming $\mathbf{1.1355 \times 10^{-3}}$ J per sample. The lower energy consumption for each neuron type and encoding are highlighted in bold Figure 10 complements the table with heatmaps for 8, 6, and 4 time steps (panels a–c), where rows are neuron types and columns are encoding schemes; color denotes energy on a logarithmic scale, and a star marks the lowest-energy encoding within each neuron type.

(**a**) 8 time steps



(**b**) 6 time steps



(**c**) 4 time steps

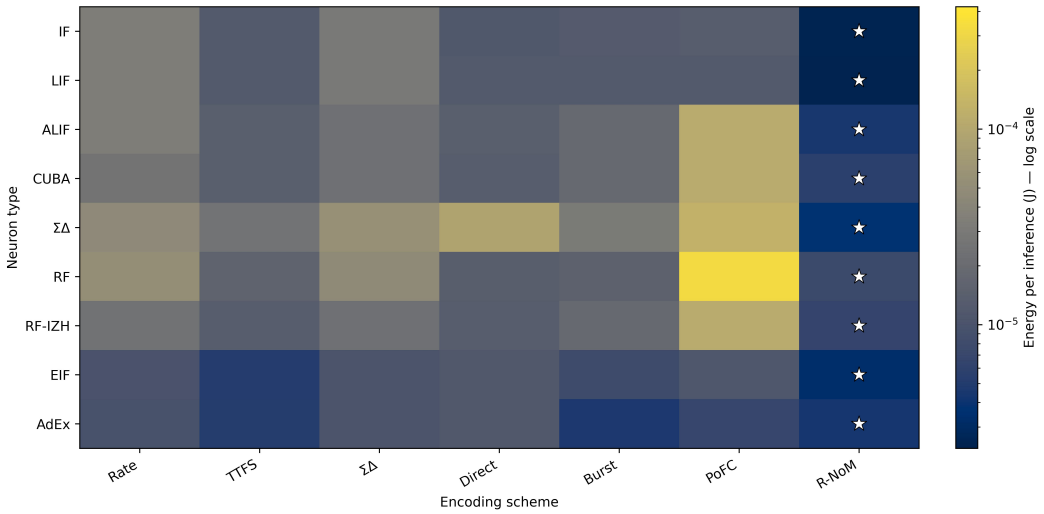


Figure 10. Energy per sample on MNIST for each neuron (rows) and encoding (columns). The best cell per neuron is marked with a star.

Table 7. Total energy consumption during inference on the MNIST dataset (joules per sample).

Neuron Type	Time Steps	Rate Encoding	TTFS	ΣΔ	Direct Coding	Burst Coding	PoFC	R-NoM
IF	8	6.69581×10^{-5}	1.23361×10^{-5}	5.99763×10^{-5}	2.28804×10^{-5}	2.12915×10^{-5}	3.00687×10^{-5}	2.40915×10^{-6}
	6	5.02249×10^{-5}	1.22764×10^{-5}	4.48093×10^{-5}	1.89740×10^{-5}	1.66753×10^{-5}	1.96420×10^{-5}	2.33240×10^{-6}
	4	3.34828×10^{-5}	1.22051×10^{-5}	2.95766×10^{-5}	1.15835×10^{-5}	1.27242×10^{-5}	1.32330×10^{-5}	2.50741×10^{-6}
LIF	8	6.69782×10^{-5}	1.23361×10^{-5}	6.00020×10^{-5}	2.69682×10^{-5}	2.69682×10^{-5}	2.69682×10^{-5}	2.69682×10^{-5}
	6	5.02339×10^{-5}	1.22746×10^{-5}	4.48206×10^{-5}	1.94391×10^{-5}	1.94391×10^{-5}	1.94391×10^{-5}	1.94391×10^{-5}
	4	3.34905×10^{-5}	1.22085×10^{-5}	2.95881×10^{-5}	1.23209×10^{-5}	1.23209×10^{-5}	1.23209×10^{-5}	2.44273×10^{-6}
ALIF	8	7.3510×10^{-5}	2.6871×10^{-5}	5.1945×10^{-5}	3.40339×10^{-5}	3.24837×10^{-5}	2.33675×10^{-4}	1.03986×10^{-5}
	6	4.92296×10^{-5}	2.10553×10^{-5}	3.81021×10^{-5}	2.50982×10^{-5}	2.70057×10^{-5}	1.72831×10^{-4}	8.96057×10^{-6}
	4	3.28536×10^{-5}	1.44922×10^{-5}	2.33582×10^{-5}	1.41680×10^{-5}	1.86396×10^{-5}	1.10753×10^{-4}	4.41252×10^{-6}
CUBA	8	4.90000×10^{-5}	1.90000×10^{-5}	4.50000×10^{-5}	2.64009×10^{-5}	2.66573×10^{-5}	2.66573×10^{-5}	8.95950×10^{-6}
	6	3.81410×10^{-5}	1.74550×10^{-5}	3.48430×10^{-5}	2.17564×10^{-5}	2.35472×10^{-5}	1.69526×10^{-4}	9.35265×10^{-6}
	4	2.55850×10^{-5}	1.43430×10^{-5}	2.31800×10^{-5}	1.33601×10^{-5}	1.85833×10^{-5}	1.11244×10^{-4}	5.78706×10^{-6}
ΣΔ	8	9.57181×10^{-5}	2.57815×10^{-5}	8.88477×10^{-5}	1.86487×10^{-4}	3.38357×10^{-5}	2.50818×10^{-4}	5.86194×10^{-6}
	6	6.94061×10^{-5}	2.85522×10^{-5}	6.02046×10^{-5}	1.55357×10^{-4}	3.23769×10^{-5}	1.88630×10^{-4}	5.02764×10^{-6}
	4	4.70307×10^{-5}	2.47350×10^{-5}	5.58219×10^{-5}	9.05406×10^{-5}	3.03132×10^{-5}	1.28965×10^{-4}	3.66343×10^{-6}
RF	8	7.04992×10^{-5}	1.64224×10^{-5}	6.49873×10^{-5}	1.77361×10^{-5}	1.71799×10^{-5}	4.22585×10^{-4}	6.52145×10^{-6}
	6	6.15202×10^{-5}	1.59444×10^{-5}	5.60981×10^{-5}	1.75526×10^{-5}	1.49569×10^{-5}	3.67283×10^{-4}	7.72182×10^{-6}
	4	5.28138×10^{-5}	1.57942×10^{-5}	4.79926×10^{-5}	1.38899×10^{-5}	1.52142×10^{-5}	3.30481×10^{-4}	7.46545×10^{-6}
RF-IZH	8	5.74303×10^{-5}	4.94997×10^{-5}	5.39998×10^{-5}	3.38775×10^{-5}	3.84832×10^{-5}	2.41357×10^{-4}	2.69479×10^{-5}
	6	4.09763×10^{-5}	3.86407×10^{-5}	3.84926×10^{-5}	2.36274×10^{-5}	2.48624×10^{-5}	1.72002×10^{-4}	2.54921×10^{-5}
	4	2.43212×10^{-5}	1.33653×10^{-5}	2.27467×10^{-5}	1.31812×10^{-5}	1.88247×10^{-5}	1.10943×10^{-4}	6.54786×10^{-6}
EIF	8	2.09544×10^{-5}	6.89162×10^{-6}	2.10879×10^{-5}	2.54309×10^{-5}	7.57636×10^{-6}	2.18263×10^{-5}	4.88905×10^{-6}
	6	1.50407×10^{-5}	5.82237×10^{-6}	1.57690×10^{-5}	1.73112×10^{-5}	7.84863×10^{-6}	1.61259×10^{-5}	3.81949×10^{-6}
	4	9.98311×10^{-6}	5.11222×10^{-6}	1.03737×10^{-5}	1.17131×10^{-5}	7.73232×10^{-6}	1.11675×10^{-5}	3.30303×10^{-6}
AdEx	8	2.10624×10^{-5}	7.36623×10^{-6}	2.10755×10^{-5}	2.37614×10^{-5}	4.60625×10^{-6}	1.17924×10^{-5}	9.51151×10^{-6}
	6	1.55792×10^{-5}	6.13887×10^{-6}	1.56530×10^{-5}	1.81244×10^{-5}	4.65618×10^{-6}	9.51664×10^{-6}	6.74905×10^{-6}
	4	9.70576×10^{-6}	5.24387×10^{-6}	1.04670×10^{-5}	1.17669×10^{-5}	4.59669×10^{-6}	6.84008×10^{-6}	4.31047×10^{-6}

Trade-Off Between Accuracy and Power Consumption

As summarized in Table 7, R–NoM consistently yields the lowest energy consumption for most neuron types. However, Table 6 shows that its classification accuracy remains notably below that of the best-performing schemes such as ΣΔ and direct coding. Conversely, the highest-accuracy configurations (for example, ΣΔ neurons with rate or ΣΔ encoding) still operate at energy levels below the ANN baseline (1.1355×10^{-3} J), highlighting the ability of SNNs to outperform traditional ANNs in power efficiency without substantial loss in accuracy.

- *R–NoM: minimal energy, lower accuracy.* R–NoM encoding achieves exceptionally low power usage, for instance 2.33×10^{-6} J for the IF neuron at six time steps. However, these configurations typically exhibit weaker accuracies (approximately 70–75% for IF and considerably lower for ALIF), illustrating a clear trade-off between energy savings and predictive performance.
- *High-accuracy configurations remain energy-efficient.* Several neuron types—including ΣΔ, ALIF, and CUBA—achieve accuracies near or above 97–98% while consuming far less energy than the ANN baseline. For example, ΣΔ neurons with rate encoding at eight time steps yield 98.10% accuracy (Table 6) and require approximately 9.57×10^{-5} J per inference (Table 7), representing about a tenfold improvement in energy efficiency relative to the ANN baseline.
- *AdEx neurons: best energy with burst coding, highest accuracy with direct coding.* AdEx neurons achieve their minimum energy consumption under burst coding (as reported in Table 7) but attain their best accuracy, around 97.4–97.5%, using direct coding (Table 6). This illustrates that the most energy-efficient configuration does not necessarily coincide with the highest-accuracy one, even within the same neuron model.

- *Fewer time steps reduce energy but may lower accuracy.* Most neuron types consume less power at four or six time steps than at eight. However, for temporal encodings such as TTFS and PoFC, reducing the number of steps can lower accuracy by several percentage points. Models with adaptive dynamics—ALIF, AdEx, and ΣΔ—maintain relatively high accuracy at lower time steps, making them favorable for energy-constrained applications.
- *Overall SNN advantage.* Almost all SNN configurations, including those achieving 97–98% accuracy, consume substantially less energy than the ANN baseline. This confirms the suitability of SNNs for edge and low-power deployments, where minor accuracy losses may be acceptable in exchange for significant energy savings.
- *Balancing encoding and neuron type.* Although R–NoM leads in power reduction, it consistently trails in accuracy. Encodings such as ΣΔ or direct coding achieve near-ANN accuracy with only moderate energy overhead compared with R–NoM, while still maintaining far lower energy usage than ANNs. Selecting the optimal combination of a neuron model and encoding scheme, therefore, requires balancing accuracy requirements against available power budgets, as each pairing exhibits distinct performance–efficiency characteristics.

4.2. Performance on CIFAR-10 Dataset

In this subsection, we utilized a VGG7 network architecture as described in Section 3.1 and evaluated it on the CIFAR-10 dataset using various neuron models and encoding schemes to show classification accuracies achieved. We aimed to assess the performance of different SNN configurations compared to a baseline ANN. Additionally, we evaluated the models at varying time steps (2, 4, and 6) to evaluate the effect of temporal dynamics.

4.2.1. Classification Accuracy Results

Table 8 and Figure 11 summarize the maximum classification accuracies (%) achieved by various neuron models, encoding schemes, and time steps on CIFAR-10. The dashed horizontal line in Figure 11 marks the ANN baseline (83.6%), and panels (a)–(c) correspond to 2, 4, and 6 time steps, respectively. Note that accuracies near 10% indicate ineffective learning at those settings (roughly random guessing over 10 classes). For clarity, the highest accuracies for each neuron type and encoding scheme are highlighted in bold in Table 8.

Observation

The results in Table 8 demonstrate that SNNs can approach the classification accuracy of traditional ANNs on complex datasets such as CIFAR-10. The choice of neuron model, encoding scheme, and number of time steps has a significant impact on performance.

- ΣΔ *neurons achieving the highest accuracy.* ΣΔ neurons attained the highest SNN accuracy of 83.00% with direct coding at two time steps, closely matching the ANN baseline of 83.60%. This strong performance with a minimal number of time steps highlights the efficiency of ΣΔ neurons in capturing complex spatial–temporal patterns. In addition, TTFS encoding with ΣΔ neurons achieved accuracies up to 72.50%, confirming their versatility across different encoding schemes.
- *Performance of IF and LIF neurons.* IF and LIF neurons achieved comparable performance, with maximum accuracies of 74.50% each using direct coding at four time steps. These findings suggest that even simpler neuron models can perform effectively on complex datasets when combined with appropriate encoding methods. Accuracy improved modestly with additional time steps, indicating that temporal dynamics contribute positively to their classification capability.

Table 8. Maximum classification accuracies (%) on VGG7 with CIFAR-10 dataset.

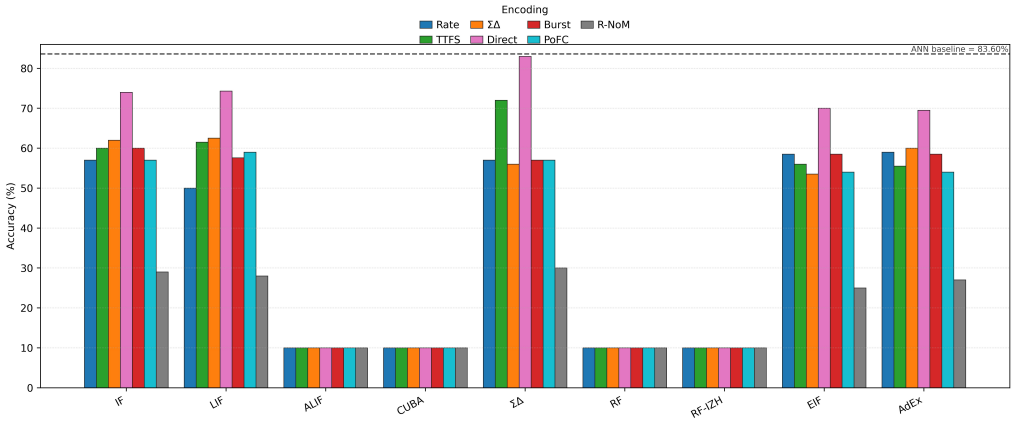
Neuron Type	Time Steps	Rate Encoding	TTFS	ΣΔ	Direct Coding	Burst Coding	PoFC	R-NoM
ANN (Baseline)	–				**83.60%**			
IF	2	57.00	60.00	62.00	74.00	60.00	57.00	29.00
	4	56.50	62.00	65.00	**74.50**	64.00	65.00	27.00
	6	57.00	62.50	65.00	74.50	64.50	68.00	30.00
LIF	2	50.00	61.50	62.50	74.30	57.60	59.00	28.00
	4	51.00	62.00	64.50	**74.50**	61.00	63.00	23.00
	6	50.00	59.50	65.00	74.00	61.50	67.00	21.00
ALIF	2	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	4	39.00	34.00	20.00	46.00	20.00	27.00	14.00
	6	**51.00**	38.00	28.00	49.00	27.00	29.00	24.00
CUBA	2	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	4	34.00	**50.00**	33.50	40.00	30.00	24.00	25.00
	6	45.00	43.00	40.00	50.00	30.00	31.00	27.00
ΣΔ	2	57.00	72.00	56.00	**83.00**	57.00	57.00	30.00
	4	61.00	72.00	66.00	78.00	66.00	62.00	27.00
	6	60.00	72.50	68.00	79.00	66.00	67.00	24.00
RF	2	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	4	22.00	10.00	10.00	42.00	10.00	10.00	10.00
	6	37.00	36.00	32.00	**47.00**	39.00	37.00	31.00
RF-IZH	2	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	4	10.00	10.00	10.00	33.00	10.00	10.00	10.00
	6	**45.00**	37.00	30.00	39.00	32.00	32.00	27.00
EIF	2	58.50	56.00	53.50	**70.00**	58.50	54.00	25.00
	4	59.50	64.00	60.00	69.00	57.00	57.00	22.50
	6	60.00	65.00	61.00	68.50	53.50	60.00	24.00
AdEx	2	59.00	55.50	60.00	69.50	58.50	54.00	27.00
	4	60.00	62.00	61.50	70.00	56.50	59.00	29.00
	6	60.50	63.00	60.50	**70.10**	52.00	61.00	25.50

- *Performance of ALIF neurons.* The adaptive LIF (ALIF) model achieved a maximum accuracy of 51.00% with rate encoding at 6 time steps. The relatively lower accuracy compared with other neuron types suggests that ALIF may require further tuning or more advanced encoding strategies to fully exploit its adaptation mechanisms on CIFAR-10.
- *Performance of CUBA neurons.* CUBA neurons reached a peak accuracy of 50.00% with TTFS encoding at 4 time steps. Although this exceeds random performance, it indicates that CUBA neurons alone may not effectively capture the complex features in CIFAR-10 without additional optimization.
- *Performance of RF and RF–IZH neurons.* RF and its Izhikevich variant (RF–IZH) achieved peak accuracies of 47.00% and 45.00%, respectively. These results suggest that resonate-and-fire dynamics are less suited for complex vision tasks such as CIFAR-10 image classification.
- *Performance of EIF and AdEx neurons.* EIF neurons reached an accuracy of 70.00% with direct coding at 2 time steps, while AdEx achieved 70.10% with direct coding at 6 time steps. These findings suggest that exponential spike initiation and adaptive dynamics enhance the processing of complex input patterns.
- *Effectiveness of direct coding and TTFS encoding.* Across neuron models, direct coding proved the most effective scheme, consistently yielding higher accuracies. This is likely because it preserves detailed spatial information without depending heavily on temporal representation. TTFS also performed well, particularly with ΣΔ neurons, indicating that precise spike timing benefits certain configurations.
- *Impact of time steps.* The number of time steps affected performance, though less strongly than in the MNIST experiments. Some models, such as ΣΔ, achieved high

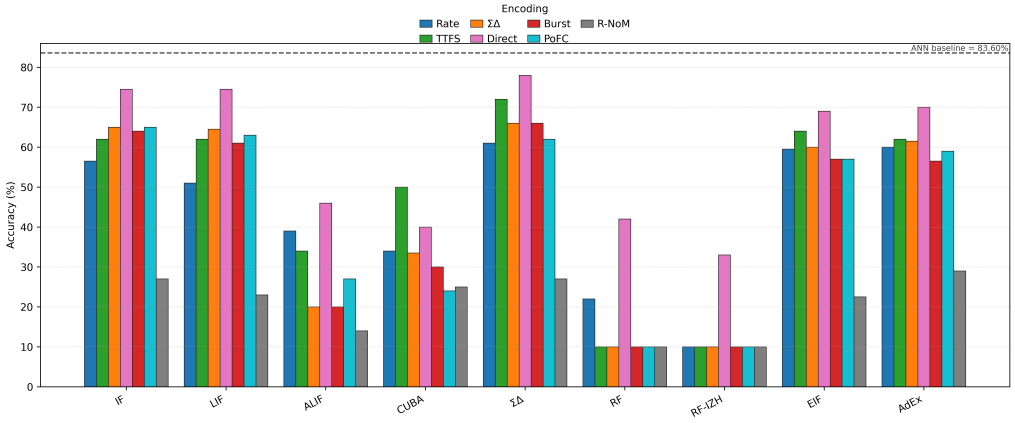
accuracy even at 2 time steps, highlighting computational efficiency. Increasing time steps generally provided modest accuracy gains, suggesting a balance between temporal resolution and computational cost.

- *Comparison with ANN baseline.* Although several SNN configurations approached the ANN baseline, many remained slightly below. Careful selection of neuron models and encoding schemes is, therefore, crucial to achieving high performance on complex datasets. In particular, ΣΔ neurons combined with direct coding exhibit strong potential, offering high accuracy at low time steps while maintaining energy and computational efficiency.

(**a**) 2 time steps



(**b**) 4 time steps



(**c**) 6 time steps

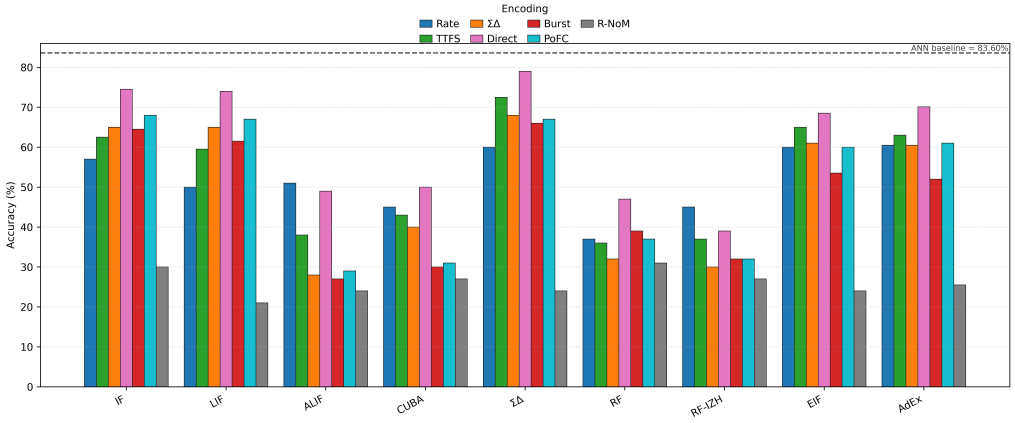


Figure 11. CIFAR-10 accuracy by neuron and encoding—2, 4, and 6 time steps.

4.2.2. Energy Consumption on the CIFAR-10 Dataset

Table 9 reports the total energy per inference (joules) on CIFAR-10 across all SNN configurations, measured on a GPU. The ANN baseline consumed 1.1355×10^{-3} J per sample. For each neuron-encoding pair, the configuration with the lowest energy consumption is indicated in the table. Figure 12 complements these results with heatmaps for 6, 4, and 2 time steps (panels a–c), where rows correspond to neuron types and columns to encoding schemes. Color intensity represents energy consumption on a logarithmic scale, and a star denotes the lowest-energy encoding within each neuron category.

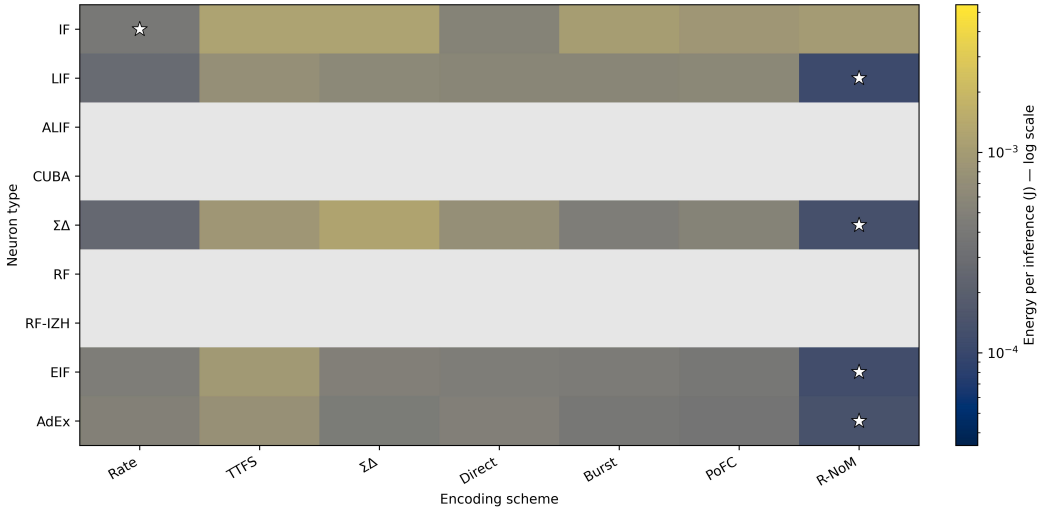
Observation

- *Overall energy trends.* SNNs consistently consume less energy than the baseline ANN across neuron types and encoding schemes. Models achieving higher classification accuracy—such as those using direct or temporal encodings—typically exhibit higher energy usage than simpler schemes. Nevertheless, even the most energy-demanding SNN configurations remain below the 1.1355×10^{-3} J reference value set by the ANN. This finding reinforces SNNs' potential for substantial energy savings in complex tasks such as CIFAR-10 classification.

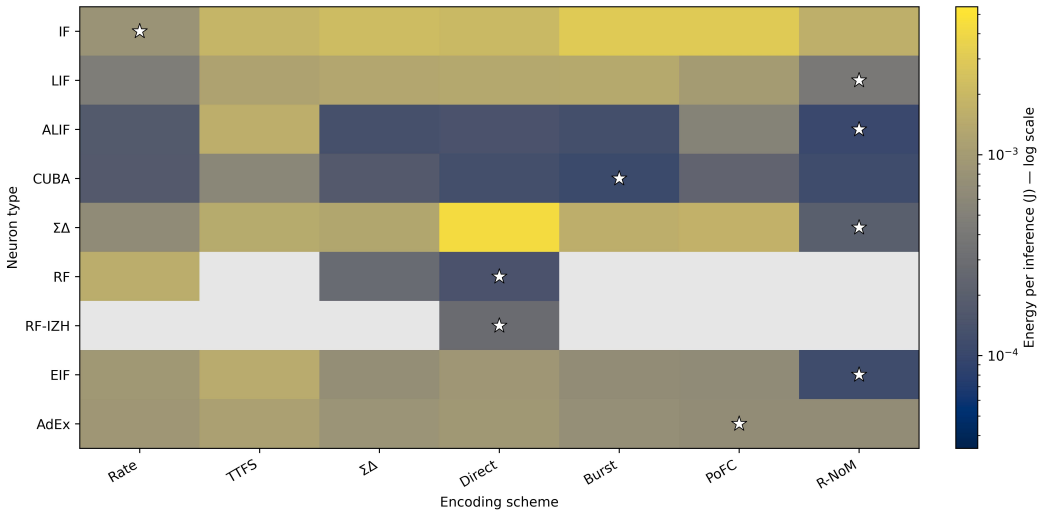
Table 9. Total energy consumption during inference on the CIFAR-10 dataset (joules per sample).

Neuron Type	Time Steps	Rate Encoding	TTFS	ΣΔ	Direct Coding	Burst Coding	PoFC	R-NoM
IF	2	$\mathbf{3.94852 \times 10^{-4}}$	1.16663×10^{-3}	1.17000×10^{-3}	5.22092×10^{-4}	1.04445×10^{-3}	8.80562×10^{-4}	9.83376×10^{-4}
	4	8.27214×10^{-4}	1.83994×10^{-3}	2.20152×10^{-3}	2.05982×10^{-3}	3.00816×10^{-3}	2.99366×10^{-3}	1.63505×10^{-3}
	6	1.08961×10^{-3}	2.65824×10^{-3}	2.94586×10^{-3}	2.94252×10^{-3}	4.02648×10^{-3}	4.26378×10^{-3}	1.70701×10^{-3}
LIF	2	2.78448×10^{-4}	7.26634×10^{-4}	6.23768×10^{-4}	5.80780×10^{-4}	5.80780×10^{-4}	6.11386×10^{-4}	$\mathbf{1.08714 \times 10^{-4}}$
	4	4.61336×10^{-4}	1.20903×10^{-3}	1.32926×10^{-3}	1.38078×10^{-3}	1.41197×10^{-3}	1.01033×10^{-3}	4.01296×10^{-4}
	6	6.68692×10^{-4}	1.42619×10^{-3}	1.84539×10^{-3}	1.90760×10^{-3}	1.63428×10^{-3}	1.50135×10^{-3}	8.81364×10^{-4}
ALIF	2	0	0	0	0	0	0	0
	4	1.74226×10^{-4}	1.60722×10^{-3}	1.29974×10^{-4}	1.41245×10^{-4}	1.25511×10^{-4}	5.29554×10^{-4}	$\mathbf{1.03862 \times 10^{-4}}$
	6	2.60075×10^{-4}	1.70157×10^{-3}	1.85570×10^{-4}	2.79213×10^{-4}	1.70813×10^{-4}	2.28347×10^{-3}	5.30497×10^{-4}
CUBA	2	0	0	0	0	0	0	0
	4	1.78050×10^{-4}	6.03918×10^{-4}	1.75168×10^{-4}	1.26776×10^{-4}	$\mathbf{1.07812 \times 10^{-4}}$	2.34300×10^{-4}	1.15656×10^{-4}
	6	3.63035×10^{-4}	1.46932×10^{-3}	2.48938×10^{-4}	1.60722×10^{-4}	1.83053×10^{-4}	5.15800×10^{-3}	2.10588×10^{-4}
ΣΔ	2	2.56910×10^{-4}	8.69356×10^{-4}	1.23596×10^{-3}	7.30794×10^{-4}	4.56246×10^{-4}	5.34300×10^{-4}	1.30238×10^{-4}
	4	6.60378×10^{-4}	1.43843×10^{-3}	1.29967×10^{-3}	4.38806×10^{-3}	1.60582×10^{-3}	1.76199×10^{-3}	1.96755×10^{-4}
	6	9.75392×10^{-4}	1.45020×10^{-3}	2.06100×10^{-3}	4.73134×10^{-3}	1.37457×10^{-3}	2.10540×10^{-3}	$\mathbf{4.66853 \times 10^{-5}}$
RF	2	0	0	0	0	0	0	0
	4	1.54216×10^{-3}	0	2.75743×10^{-4}	1.42578×10^{-4}	0	0	0
	6	4.95305×10^{-3}	5.45528×10^{-3}	8.55122×10^{-4}	5.99531×10^{-4}	1.45782×10^{-3}	3.74125×10^{-3}	$\mathbf{3.45218 \times 10^{-5}}$
RF-IZH	2	0	0	0	0	0	0	0
	4	0	0	0	$\mathbf{2.85883 \times 10^{-4}}$	0	0	0
	6	2.60712×10^{-4}	1.12258×10^{-4}	6.43347×10^{-4}	5.21456×10^{-4}	2.23209×10^{-4}	4.21825×10^{-4}	2.45245×10^{-4}
EIF	2	4.51992×10^{-4}	9.22080×10^{-4}	4.80420×10^{-4}	4.62498×10^{-4}	4.36462×10^{-4}	3.92722×10^{-4}	$\mathbf{1.18310 \times 10^{-4}}$
	4	9.11080×10^{-4}	1.45713×10^{-3}	7.15536×10^{-4}	8.86820×10^{-4}	6.84824×10^{-4}	6.64914×10^{-4}	1.15656×10^{-4}
	6	1.30268×10^{-3}	1.81481×10^{-3}	1.06204×10^{-3}	1.34335×10^{-3}	9.94740×10^{-4}	1.01460×10^{-3}	2.10588×10^{-4}
AdEx	2	4.98822×10^{-4}	7.65386×10^{-4}	$\mathbf{4.27554 \times 10^{-4}}$	4.79906×10^{-4}	3.92988×10^{-4}	3.56532×10^{-4}	1.38236×10^{-4}
	4	8.93400×10^{-4}	1.12146×10^{-3}	8.35974×10^{-4}	9.07006×10^{-4}	7.49690×10^{-4}	6.93842×10^{-4}	6.97172×10^{-4}
	6	1.27278×10^{-3}	1.45338×10^{-3}	1.19859×10^{-3}	1.36141×10^{-3}	1.03748×10^{-3}	9.92686×10^{-4}	1.89960×10^{-4}

(**a**) 2 time steps



(**b**) 4 time steps



(**c**) 6 time steps

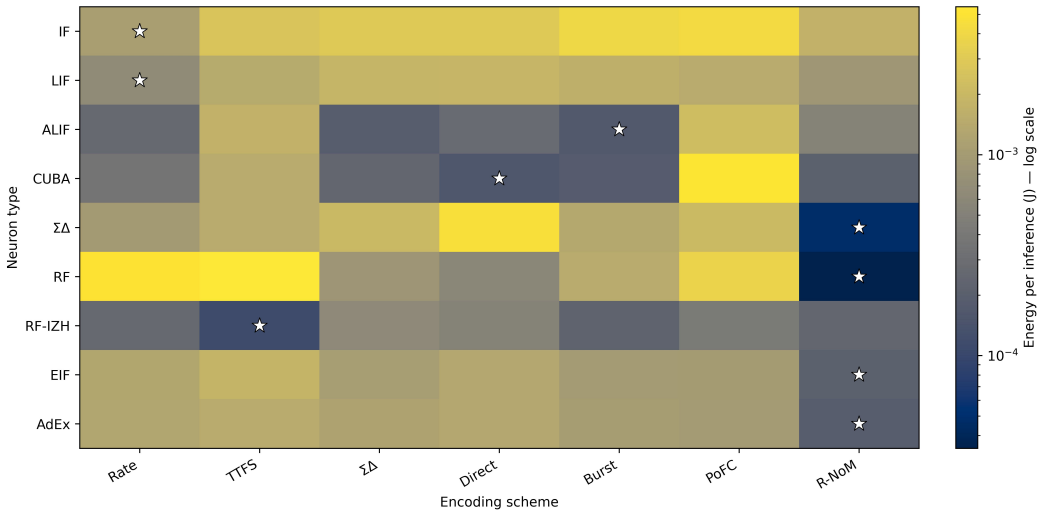


Figure 12. Energy per sample on **CIFAR-10** for each neuron (rows) and encoding (columns). The lowest energy per neuron (among measured entries) is marked with a star; zero entries indicate configurations not evaluated.

- *Influence of time steps.* A consistent pattern emerges in which increasing the number of time steps leads to higher energy consumption. For example, IF neurons increase from 3.95×10^{-4} J at two time steps to 1.09×10^{-3} J at six time steps under rate encoding. Although additional time steps can enhance classification accuracy, they also raise energy cost. Consequently, applications requiring real-time performance or low power consumption may favor configurations with fewer time steps, provided accuracy remains within acceptable limits.
- *Neuron models and encoding schemes.* Simpler neuron models (e.g., IF, LIF) generally exhibit moderate energy usage, but their efficiency depends strongly on the encoding strategy. For instance, IF neurons with rate encoding at two time steps require as little as 3.95×10^{-4} J, whereas LIF neurons combined with temporal encodings consume more energy but often achieve better accuracy. Advanced models such as ALIF and AdEx may improve classification performance but do not always minimize energy consumption. Their adaptive dynamics can reduce spike activity under certain conditions, although this benefit varies depending on the encoding scheme and the number of time steps.
- *Balancing accuracy and efficiency.* Encoding schemes with low energy requirements often exhibit lower accuracy. Thus, selecting an optimal combination of neuron model, encoding method, and time steps is essential for balancing performance and efficiency. Overall, the results confirm that SNNs consistently maintain lower energy consumption than conventional ANNs, even when configured for higher accuracy. This balance between accuracy and energy efficiency underscores SNNs' suitability for edge devices and other power-sensitive applications.

4.3. Effect of Thresholding and Encoding Schemes on Model Performance and Energy Consumption

This subsection investigates how variations in neuronal firing threshold and encoding scheme influence both classification accuracy and energy consumption of SNNs. The analysis focuses on the EIF neuron model applied to the CIFAR-10 dataset. The experimental setup includes evaluation at time steps of 2, 4, and 6, threshold values of 0.1, 0.5, and 0.75, and a range of encoding schemes—rate encoding, TTFS, ΣΔ, direct coding, burst coding, PoFC, and R–NoM. These experiments provide insight into how threshold tuning interacts with encoding dynamics to affect SNN performance and efficiency.

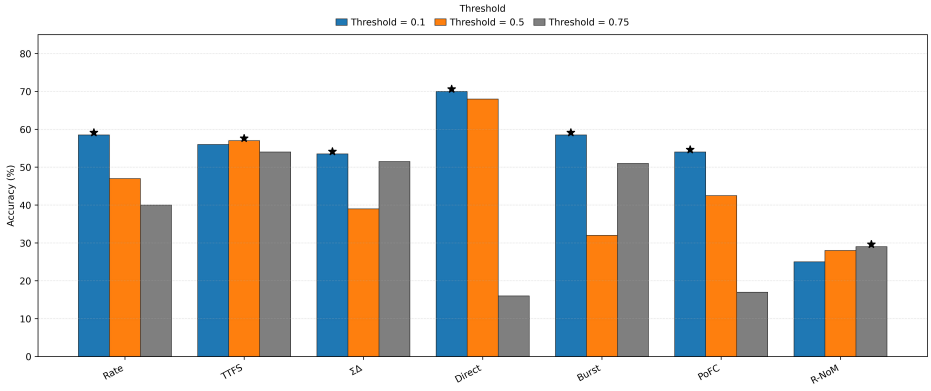
4.3.1. Impact of Threshold Values on Classification Accuracy

As summarized in Table 10 and visualized in Figure 13, the neuronal firing threshold across encoding schemes strongly influences classification accuracy. In Figure 13, a ★ marks the best threshold for each encoding at each time step. Lower thresholds, such as 0.1, tend to facilitate easier spiking and thus higher accuracy in schemes like Rate Encoding and Direct Coding, where performance noticeably declines as the threshold increases. In contrast, temporal TTFS encoding achieves optimal accuracy at an intermediate threshold of 0.5, which appears to balance the timing of the first spike effectively. ΣΔ encoding similarly benefits from lower thresholds, particularly at longer time steps, underscoring its sensitivity to threshold adjustments. Meanwhile, burst coding and PoFC exhibit variable performance across thresholds and time steps, indicating that their optimal operation requires careful tuning of threshold values. Finally, R-NoM consistently produces lower accuracy levels, with only marginal improvements at higher thresholds, suggesting that threshold variations limit its capability to capture complex patterns in the CIFAR-10 dataset.

Table 10. Maximum classification accuracies (%) on CIFAR-10 dataset with varying thresholds. The highest accuracy for each encoding scheme is highlighted in bold.

Encoding Scheme	Threshold	2 Steps	4 Steps	6 Steps
Rate encoding	0.1	**58.50**	**59.50**	**60.00**
	0.5	47.00	53.00	47.50
	0.75	40.00	41.00	27.00
TTFS encoding	0.1	56.00	64.00	65.00
	0.5	**57.00**	**65.50**	**67.50**
	0.75	54.00	65.00	55.00
ΣΔ encoding	0.1	**53.50**	**60.00**	**61.00**
	0.5	39.00	38.50	54.00
	0.75	51.50	30.00	51.00
Direct coding	0.1	**70.00**	**69.00**	**68.50**
	0.5	68.00	67.50	66.00
	0.75	16.00	17.00	15.00
Burst Coding	0.1	**58.50**	57.00	53.50
	0.5	32.00	**56.00**	**60.00**
	0.75	51.00	57.00	55.00
PoFC	0.1	**54.00**	**57.00**	**60.00**
	0.5	42.50	53.50	56.00
	0.75	17.00	39.50	20.00
R-NoM	0.1	25.00	22.50	24.00
	0.5	**28.00**	**28.00**	**29.50**
	0.75	29.00	28.50	28.00

(**a**) 2 time steps



(**b**) 4 time steps

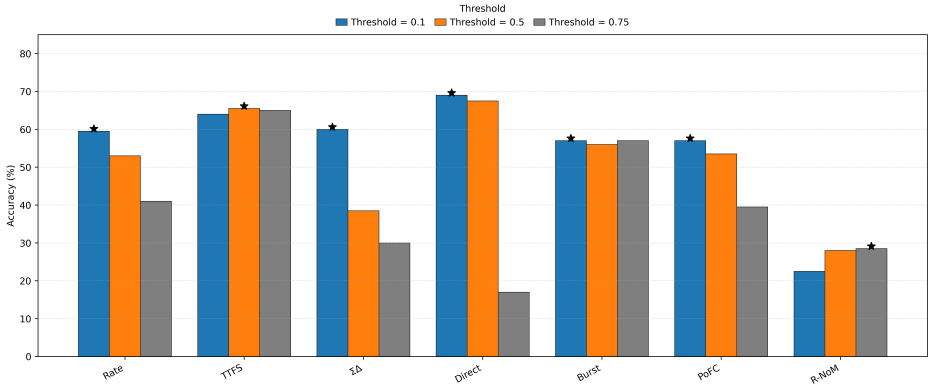


Figure 13. *Cont.*

(**c**) 6 time steps

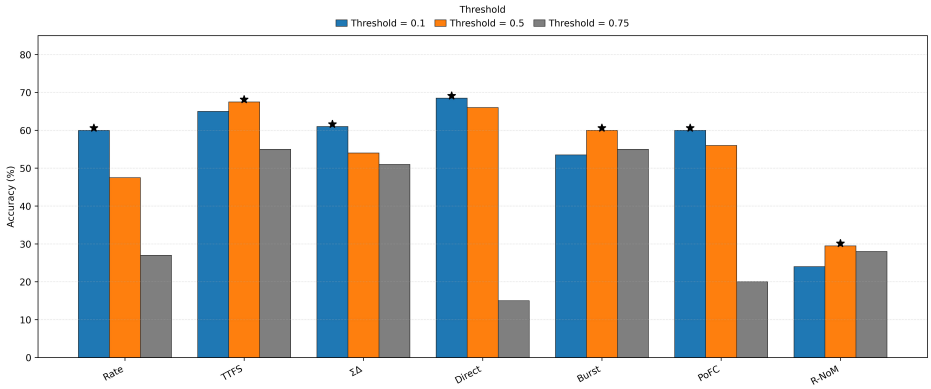


Figure 13. CIFAR-10 accuracy vs. threshold by encoding—2, 4, and 6 time steps, on EIF neuron. A ★ above a bar marks the best threshold for that encoding at the given time step.

4.3.2. Impact of Threshold Values on Energy Consumption

The results presented in Table 11 show that energy consumption generally decreases as the neuronal firing threshold increases from 0.1 to 0.75. This trend is consistent across most encoding schemes, reflecting the fact that higher thresholds reduce the number of spikes generated during inference and thereby lower the computational load. For example, both rate and TTFS encodings show substantial energy reductions at elevated thresholds, while direct coding exhibits a marked drop in energy usage at a threshold of 0.75—though at the expense of classification accuracy. ΣΔ encoding also benefits from higher thresholds, although the reduction in energy consumption is less pronounced, likely due to its inherent reliance on representing fine-grained variations in the input. Overall, these results highlight the delicate balance between minimizing energy consumption and maintaining predictive accuracy, as the threshold directly modulates the trade-off between energy efficiency and inference performance.

Table 11. Energy consumption (joules) on CIFAR-10 dataset with varying thresholds. The lowest energy consumption for each encoding scheme is highlighted in bold.

Encoding Scheme	Threshold	2 Steps	4 Steps	6 Steps
Rate encoding	0.1	4.51992×10^{-4}	9.11080×10^{-4}	1.30268×10^{-3}
	0.5	1.81350×10^{-4}	4.17308×10^{-4}	4.60568×10^{-4}
	0.75	8.03670×10^{-5}	1.67766×10^{-4}	6.69786×10^{-5}
TTFS encoding	0.1	9.22080×10^{-4}	1.45713×10^{-3}	1.81481×10^{-3}
	0.5	2.55394×10^{-4}	4.67294×10^{-4}	6.54078×10^{-4}
	0.75	1.78473×10^{-4}	3.65690×10^{-4}	2.76786×10^{-4}
ΣΔ encoding	0.1	4.80420×10^{-4}	7.15536×10^{-4}	1.06204×10^{-3}
	0.5	8.77776×10^{-5}	1.77862×10^{-4}	4.00548×10^{-4}
	0.75	1.13804×10^{-4}	7.55347×10^{-5}	2.59425×10^{-4}
Direct coding	0.1	4.62498×10^{-4}	8.86820×10^{-4}	1.34335×10^{-3}
	0.5	3.14614×10^{-4}	6.51242×10^{-4}	9.66956×10^{-4}
	0.75	9.16112×10^{-6}	1.70082×10^{-5}	3.43297×10^{-5}
Burst Coding	0.1	4.36462×10^{-4}	6.84824×10^{-4}	9.94740×10^{-4}
	0.5	9.06143×10^{-5}	3.29403×10^{-4}	3.35664×10^{-4}
	0.75	1.68229×10^{-4}	2.16433×10^{-4}	2.27624×10^{-4}
PoFC	0.1	3.92722×10^{-4}	6.64914×10^{-4}	1.01460×10^{-3}
	0.5	1.51961×10^{-4}	4.15849×10^{-4}	5.35901×10^{-4}
	0.75	2.53021×10^{-5}	1.48193×10^{-4}	4.42032×10^{-5}
R-NoM	0.1	1.18310×10^{-4}	1.15656×10^{-4}	2.10588×10^{-4}
	0.5	7.57243×10^{-5}	7.66446×10^{-5}	2.48482×10^{-4}
	0.75	6.81089×10^{-5}	5.90738×10^{-5}	6.24119×10^{-5}

4.3.3. Trade-Off Between Accuracy and Energy Consumption

The analysis of Tables 10 and 11 indicates that threshold values play a key role in determining the balance between classification accuracy and energy efficiency in SNNs. Lower thresholds increase neuronal activity and typically yield higher accuracy but result in greater energy consumption due to elevated spike rates. Conversely, higher thresholds suppress spiking activity and reduce energy use but may hinder the network's ability to capture complex temporal or spatial patterns. Intermediate thresholds (e.g., 0.5) often provide a balanced compromise between accuracy and energy, as observed for TTFS and direct encodings. These findings emphasize the importance of optimal threshold selection, as it directly affects both network efficiency and representational fidelity. Moreover, the degree of threshold sensitivity varies among encoding schemes, underscoring the need for task-specific tuning of this hyperparameter to maximize the EIF model's performance on complex datasets such as CIFAR-10.

4.3.4. Comparison with Related Studies

Recent studies have reached complementary conclusions. Takaghaj and Sampson (2024) introduced Rouser, a training method that adaptively learns neuronal thresholds, effectively mitigating the "dead neuron" problem and improving accuracy and convergence across neuromorphic datasets [161]. Their adaptive mechanism aligns with our observation that threshold optimization is crucial for performance stability. Similarly, Sengupta et al. [154] emphasized the importance of threshold balancing in ANN-to-SNN conversion for deep architectures (e.g., VGG and ResNet), showing that appropriate threshold tuning preserves accuracy and reduces hardware overhead by limiting redundant spiking. Diehl et al. [37] also demonstrated that optimal weight and threshold balancing minimizes performance degradation during ANN-to-SNN conversion, supporting our findings on the sensitivity of performance to threshold parameters.

4.4. Comparative Analysis and Discussion

Overall trade-off. Across all experimental conditions, a clear trade-off is evident: configurations optimized for maximum accuracy typically incur higher per-inference energy consumption and SynOps, whereas energy-efficient encodings achieve lower accuracy.

FCN (MNIST). From Table 6, ΣΔ neurons with rate encoding at eight time steps achieved an accuracy of 98.10%, closely matching the ANN baseline of 98.23%. However, Table 7 shows that this configuration consumes more energy than simpler IF/LIF models with energy-efficient encodings such as burst or R–NoM. ALIF and AdEx also performed competitively, while direct and ΣΔ encodings consistently yielded strong results across neuron types. Increasing the number of time steps generally improved accuracy—particularly for temporal encodings such as TTFS and PoFC—but at the cost of higher energy consumption.

VGG7 (CIFAR-10). From Table 8, ΣΔ neurons with direct coding achieved 83.00% accuracy at two time steps, close to the ANN baseline of 83.60%. This demonstrates effective temporal dynamics even at low T. However, this configuration required more energy than several lower-accuracy SNNs (Table 9). IF and LIF neurons with direct coding peaked at approximately 74.5%, whereas ALIF and CUBA underperformed without further tuning. Direct coding proved the most effective scheme overall for CIFAR-10, while TTFS paired particularly well with ΣΔ neurons.

Thresholds and time steps. Lower thresholds increased spike rates, typically improving accuracy but at the expense of higher energy usage. Higher thresholds reduced spiking and energy consumption but occasionally limited the model's capacity to learn complex patterns. Intermediate thresholds offered the best trade-offs, especially for direct and TTFS

encodings. Similarly, additional time steps improved accuracy in many cases yet also elevated energy demands, reinforcing the need for task-dependent tuning of T.

Energy. Across datasets, per-inference energy consumption is primarily influenced by encoding sparsity and the number of time steps. On MNIST, nearly all SNN configurations consumed less energy than the ANN baseline (1.1355×10^{-3} J); R–NoM yielded the lowest values (e.g., IF at $T = 6$: 2.33×10^{-6} J; LIF at $T = 4$: 2.44×10^{-6} J). Higher-accuracy configurations—such as $\Sigma\Delta$ neurons with rate or $\Sigma\Delta$ encoding—required more energy than R–NoM or burst but remained well below the ANN reference. On CIFAR-10, energy generally increased with T and denser encodings; several high-T configurations exceeded the ANN baseline, while low-T direct or rate encodings often remained below it. In practice, using fewer time steps and sparse encodings (R–NoM or burst) minimizes energy, whereas $\Sigma\Delta$ or direct coding improves accuracy with a moderate energy overhead.

Application guidance.

- *Accuracy-critical tasks:* $\Sigma\Delta$ neurons with direct (CIFAR-10) or rate/$\Sigma\Delta$ encodings (MNIST) achieve near-ANN accuracy, albeit with higher energy costs than ultra-sparse alternatives, yet still well below ANN levels.
- *Energy-constrained or edge deployments:* IF or LIF neurons combined with burst or R–NoM encoding and fewer time steps provide strong energy savings. Thresholds can be tuned to meet power constraints, accepting minor accuracy trade-offs.

Neuromorphic potential. Event-driven accelerators can leverage SNN properties such as data-dependent spiking, temporal asynchrony, and localized memory access to achieve system-level energy gains, reinforcing SNNs' suitability for low-power and real-time applications.

Positioning in the literature. These findings are consistent with prior studies on the accuracy–efficiency trade-off and the challenges of training and converting deep spiking networks for complex vision tasks [133,153,154]. Our results highlight the promise of $\Sigma\Delta$ neurons for achieving high accuracy while confirming the substantial energy advantages of simpler neurons combined with sparse encodings.

Key takeaways. (i) $\Sigma\Delta$ neurons with suitable encodings achieve near-ANN accuracy on MNIST and reduce the performance gap on CIFAR-10; (ii) R–NoM and burst encodings minimize energy but lower accuracy; (iii) thresholds and time steps are the primary parameters governing the accuracy–energy balance; and (iv) direct and $\Sigma\Delta$ encodings are broadly effective, whereas temporal encodings benefit from larger T and careful tuning.

5. Conclusions and Outlook

This work paired a comprehensive review with a standardized, hands-on benchmarking of SNNs against architecturally matched ANNs on MNIST (shallow FCN) and CIFAR-10 (VGG7). We varied neuron models, input encodings, thresholds, and time steps, and we reported accuracy alongside a transparent per-inference energy proxy. Three consistent conclusions emerge:

(1) Accuracy–energy trade-off is real but tunable.

Across both datasets, higher accuracy typically coincided with higher energy, yet many SNN settings still undercut the ANN energy baseline:

- **MNIST (FCN):** $\Sigma\Delta$ neurons with rate/$\Sigma\Delta$ encodings reached up to **98.1%** (vs. 98.23% ANN) while remaining energetically below the ANN proxy. ALIF/AdEx and even LIF/IF also performed strongly when paired with effective encodings.
- **CIFAR-10 (VGG7):** $\Sigma\Delta$ neurons with **Direct** input achieved **83.0%** at **2** time steps (vs. 83.6% ANN), indicating that well-chosen neuron–encoding pairs can approach ANN performance even on a deeper, more complex task.

- Encodings that are extremely frugal (e.g., **R–NoM**) minimized the energy proxy but incurred the largest accuracy drops; conversely, settings that closed the accuracy gap (e.g., ΣΔ with direct/rate) used more energy—but still generally below the ANN reference on our GPU-targeted model.

(2) Practical configuration rules.

The most useful knobs in practice were the encoding, the neuron model, the threshold, and the number of time steps T:

- **Accuracy-critical:** Prefer ΣΔ neurons with **Direct** (CIFAR-10) or with **Rate/ΣΔ** (MNIST); AdEx/EIF are solid fallbacks. Keep *T as low as possible* once the accuracy target is met.
- **Energy-constrained:** Favor simpler neurons (**IF/LIF**) with **burst** or **R–NoM** encoding and small T; expect some accuracy loss. Intermediate thresholds typically balance activity with correctness better than very low or very high ones.
- **General tip:** Tune thresholds jointly with the encoding; moderate T and carefully chosen thresholds often deliver the best *accuracy-per-joule*.

(3) Neuromorphic potential.

Event-driven accelerators can exploit data-dependent spiking, temporal asynchrony, and local memory access to translate our per-inference energy reductions into larger system-level savings, reinforcing SNNs' suitability for edge and real-time deployments.

Limitations

While this study provided a standardized comparative analysis of SNNs and ANNs, several limitations remain. First, energy efficiency was assessed through an operation-based GPU proxy using literature-derived constants rather than direct measurements on neuromorphic or low-power hardware, which may cause deviations from real device performance. Second, the experimental scope was restricted to two benchmark datasets (MNIST and CIFAR-10) and two network topologies (FCN and VGG7), leaving other domains—such as event-based vision or temporal signal processing—unexplored. Third, hyperparameter and threshold tuning were manually configured, limiting the exploration of automated optimization strategies that could further improve energy–accuracy balance. Finally, while surrogate-gradient training was applied effectively, cross-paradigm comparisons (e.g., hybrid STDP-supervised or reinforcement-based learning) were not systematically analyzed. These constraints outline important directions for expansion and validation.

Future work

Future research should address the above limitations through specific methodological strategies: (i) *Hardware-in-the-loop optimization:* Incorporate direct power metering and latency profiling on neuromorphic and embedded devices (e.g., Loihi 2, SpiNNaker, TrueNorth, or Edge-TPUs) to calibrate theoretical energy models. (ii) *Algorithm–hardware co-design:* Develop reinforcement- or evolutionary-based search frameworks that jointly tune neuron thresholds, time windows, and precision levels to achieve optimal energy–accuracy trade-offs. (iii) *Dynamic spiking policies:* Introduce adaptive thresholding and time step schedules that adjust spiking activity in real time based on task difficulty or input sparsity. (iv) *Architecture and encoding search:* Employ automated architecture discovery (e.g., neural architecture search or Bayesian optimization) to identify optimal neuron-encoding pairs across tasks. (v) *Cross-domain and on-device learning:* Expand SNN applications to event-based sensors, biomedical, and audio datasets while investigating local plasticity and continual learning mechanisms for adaptive, low-power deployment. Collectively, these

research pathways define a concrete roadmap toward robust, scalable, and energy-aware SNN systems that bridge algorithmic theory and neuromorphic practice.

Distinct contribution and comparison with existing literature

Unlike previous surveys and reviews on spiking neural networks [19–23,27–30,33], which primarily focus on summarizing neuron models, encoding strategies, or learning algorithms in isolation, This article integrates *both theoretical synthesis and experimental benchmarking* into a unified, tutorial-style framework. Specifically, our study is the first to conduct a standardized, side-by-side evaluation of diverse neuron models (e.g., LIF, ALIF, AdEx, ΣΔ, RF) and encoding schemes (direct, rate, temporal, ΣΔ, burst, PoFC, R–NoM) on both shallow (MNIST/FCN) and deep (CIFAR-10/VGG7) architectures using common surrogate-gradient pipelines (SLAYER, SpikingJelly, Norse). While prior works such as [18,26,45] discuss energy efficiency conceptually, our contribution lies in providing a *quantitative energy–accuracy analysis* that links algorithmic parameters (thresholds, time steps, encodings) to measurable efficiency trends under reproducible experimental conditions. Furthermore, this article distills actionable design rules and optimization guidelines—bridging survey-style synthesis and empirical validation—thereby positioning it as a practical tutorial and benchmarking reference for new and advanced SNN researchers.

Takeaway

With careful choices of neuron model, encoding, threshold, and time steps, SNNs can *consistently* approach ANN accuracy on both shallow and deep settings while offering meaningful energy advantages. The recipe is straightforward. Pick the encoding and neuron to match the task's accuracy target, set T to the minimum that meets it, and calibrate thresholds for energy-aware operation—then map to event-driven hardware to compound the gains.

Author Contributions: B.A.: resources, software, methodology, data curation, investigation, writing—original draft; A.M.G.-V.: writing—review and editing, writing—original draft, validation, supervision, software, resources, methodology, investigation, funding acquisition, formal analysis, conceptualization; C.J.C.: writing—review and editing, writing—original draft, visualization, validation, supervision, software, resources, project administration, methodology, investigation, funding acquisition, conceptualization; M.S.: resources, investigation, formal analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Thedatasets used in this study are publicly available. MNIST can be accessed at http://yann.lecun.com/exdb/mnist/ (accessed on 25 October 2025), and CIFAR-10 can be accessed at https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 25 October 2025). No new data were generated in this study.

Acknowledgments: We thank the DaSCI Institute and the University of Jaén for support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. *arXiv* **2019**, arXiv:1906.02243.
2. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *arXiv* **2020**, arXiv:2007.10792.
3. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both weights and connections for efficient neural networks. In Proceedings of the 28th Conference on Neural Information Processing Systems (NeurIPS 2015), Montreal, QC, Canada, 7–12 December 2015; Volume 28.

4. Shi, Y.; Nguyen, L.; Oh, S.; Liu, X.; Kuzum, D. A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. *Front. Neurosci.* **2019**, *13*, 405. https://doi.org/10.3389/fnins.2019.00405.
5. Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **2017**, *18*, 6869–6898.
6. Patterson, D.A.; Gonzalez, J.; Le, Q.V.; Liang, P.; Munguia, L.M.; Rothchild, D.; So, D.R.; Texier, M.; Dean, J. Carbon emissions and large neural network training. *arXiv* **2021**, arXiv:2104.10350. Available online: https://arxiv.org/abs/2104.10350 (accessed on 26 October 2025).
7. Paschek, S.; Förster, F.; Kipfmüller, M.; Heizmann, M. Probabilistic Estimation of Parameters for Lubrication Application with Neural Networks. *Eng* **2024**, *5*, 2428–2440. https://doi.org/10.3390/eng5040127.
8. Nashed, S.; Moghanloo, R. Replacing Gauges with Algorithms: Predicting Bottomhole Pressure in Hydraulic Fracturing Using Advanced Machine Learning. *Eng* **2025**, *6*, 73. https://doi.org/10.3390/eng6040073.
9. Sumon, R.I.; Ali, H.; Akter, S.; Uddin, S.M.I.; Mozumder, M.A.I.; Kim, H.C. A Deep Learning-Based Approach for Precise Emotion Recognition in Domestic Animals Using EfficientNetB5 Architecture. *Eng* **2025**, *6*, 9. https://doi.org/10.3390/eng6010009.
10. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. https://doi.org/10.1016/s0893-6080(97)00011-7.
11. Adrian, E.D.; Zotterman, Y. The impulses produced by sensory nerve endings: Part 3. impulses set up by touch and pressure. *J. Physiol.* **1926**, *61*, 465–483.
12. Rueckauer, B.; Liu, S.C. Conversion of analog to spiking neural networks using sparse temporal coding. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2018), Florence, Italy, 27–30 May 2018; pp. 1–5.
13. Park, S.; Kim, S.; Choe, H.; Yoon, S. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In Proceedings of the 56th Annual Design Automation Conference (DAC 2019), Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6. https://doi.org/10.1145/3316781.3317809.
14. Gollisch, T.; Meister, M. Rapid neural coding in the retina with relative spike latencies. *Science* **2008**, *319*, 1108–1111.
15. Yamazaki, K.; Vo-Ho, V.K.; Bulsara, D.; Le, N. Spiking neural networks and their applications: A review. *Brain Sci.* **2022**, *12*, 863.
16. Neftci, E.O.; 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*, 51–63.
17. Lee, J.; Delbrück, T.; Pfeiffer, M. Enabling Gradient-Based Learning in Spiking Neural Networks with Surrogate Gradients. *Front. Neurosci.* **2020**, *14*, 123. https://doi.org/10.3389/fnins.2020.00123.
18. Zhou, C.; Zhang, H.; Yu, L.; Ye, Y.; Zhou, Z.; Huang, L.; Tian, Y. Direct training high-performance deep spiking neural networks: A review of theories and methods. *arXiv* **2024**, arXiv:2405.04289. Available online: https://arxiv.org/abs/2405.04289 (accessed on 26 October 2025).
19. Auge, D.; Hille, J.; Mueller, E.; Knoll, A. A survey of encoding techniques for signal processing in spiking neural networks. *Neural Process. Lett.* **2021**, *53*, 4693–4710.
20. Zhou, X.; Hanger, D.P.; Hasegawa, M. DeepSNN: A Comprehensive Survey on Deep Spiking Neural Networks. *Front. Neurosci.* **2020**, *14*, 456. https://doi.org/10.3389/fnins.2020.00456.
21. Nguyen, D.A.; Tran, X.T.; Iacopi, F. A review of algorithms and hardware implementations for spiking neural networks. *J. Low Power Electron. Appl.* **2021**, *11*, 23.
22. Dora, S.; Kasabov, N. Spiking neural networks for computational intelligence: An overview. *Big Data Cogn. Comput.* **2021**, *5*, 67.
23. Pietrzak, P.; Szczęsny, S.; Huderek, D.; Przyborowski, Ł. Overview of spiking neural network learning approaches and their computational complexities. *Sensors* **2023**, *23*, 3037.
24. Thorpe, S.; Gautrais, J. Rank order coding. In *Computational Neuroscience: Trends in Research*; Springer: Boston, MA, USA, 1998; pp. 113–118.
25. Paugam-Moisy, H. *Spiking Neuron Networks: A Survey*; Technical Report EPFL-REPORT-83371; École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2006.
26. Davies, M.; Wild, A.; Orchard, G.; Sandamirskaya, Y.; Guerra, G.A.F.; Joshi, P.; Risbud, S.R. Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proc. IEEE*, 2021, **109**, 911–934.
27. Paul, P.; Sosik, P.; Ciencialova, L. A survey on learning models of spiking neural membrane systems and spiking neural networks. *arXiv* **2024**, arXiv:2403.18609. Available online: https://arxiv.org/abs/2403.18609 (accessed on 26 October 2025).
28. Rathi, N.; Chakraborty, I.; Kosta, A.; Sengupta, A.; Ankit, A.; Panda, P.; Roy, K. Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware. *ACM Comput. Surv.* **2023**, *55*, 1–49.
29. Dampfhoffer, M.; Mesquida, T.; Valentian, A.; Anghel, L. Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *35*, 11906–11921.
30. Nunes, J.D.; Carvalho, M.; Carneiro, D.; Cardoso, J.S. Spiking neural networks: A survey. *IEEE Access* **2022**, *10*, 60738–60764.
31. Schliebs, S.; Kasabov, N. Evolving spiking neural network—A survey. *Evol. Syst.* **2013**, *4*, 87–98.

32. Martinez, F.S.; Casas-Roma, J.; Subirats, L.; Parada, R. Spiking neural networks for autonomous driving: A review. *Eng. Appl. Artif. Intell.* **2024**, *138*, 109415. https://doi.org/10.1016/j.engappai.2024.109415.
33. Wu, J.; Wang, Y.; Li, Z.; Lu, L.; Li, Q. A review of computing with spiking neural networks. *Comput. Mater. Contin.* **2024**, *78* .
34. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556. Available online: https://arxiv.org/abs/1409.1556 (accessed on 26 October 2025).
35. Dayan, P.; Abbott, L.F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*; MIT Press: Cambridge, MA, USA, 2001.
36. Nielsen, M.A. *Neural Networks and Deep Learning*; Determination Press: San Francisco, CA, USA, 2015; Volume 25, pp. 15–24.
37. Diehl, P.U.; Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* **2015**, *9*, 99.
38. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press: Cambridge, UK, 2002.
39. Izhikevich, E.M. Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* **2004**, *15*, 1063–1070. https://doi.org/10.1109/TNN.2004.832719.
40. Horowitz, M. Computing's energy problem (and what we can do about it). In Proceedings of the 2014 IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers, San Francisco, CA, USA, 9–13 February 2014; pp. 10–14. https://doi.org/10.1109/ISSCC.2014.6757323.
41. Sze, V.; Chen, Y.-H.; Yang, T.-J.; Emer, J.S. How to Evaluate Deep Neural Network Processors: TOPS/W (Alone) Considered Harmful. *IEEE Solid-State Circuits Mag.* **2020**, *12*, 28–41. https://doi.org/10.1109/MSSC.2020.3002140.
42. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **2018**, *38*, 82–99. https://doi.org/10.1109/MM.2018.112130359.
43. Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Imam, N.; Nakamura, Y.; Datta, P.; Nam, G.J.; et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1537–1557. https://doi.org/10.1109/TCAD.2015.2474396.
44. Furber, S.B.; Galluppi, F.; Temple, S.; Plana, L.A. The spinnaker project. *Proc. IEEE* **2014**, *102*, 652–665. https://doi.org/10.1109/JPROC.2014.2304638.
45. Roy, K.; Jaiswal, A.; Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature* **2019**, *575*, 607–617. https://doi.org/10.1038/s41586-019-1677-2.
46. Plank, J.S.; Rizzo, C.P.; Gullett, B.; Dent, K.E.M.; Schuman, C.D. Alleviating the Communication Bottleneck in Neuromorphic Computing with Custom-Designed Spiking Neural Networks. *J. Low Power Electron. Appl.* **2025**, *15*, 50. https://doi.org/10.3390/jlpea15030050.
47. Wang, X.; Zhu, Y.; Zhou, Z.; Chen, X.; Jia, X. Memristor-Based Spiking Neuromorphic Systems Toward Brain-Inspired Perception and Computing. *Nanomaterials* **2025**, *15*, 1130. https://doi.org/10.3390/nano15141130.
48. Panda, P.; Aketi, S.A.; Roy, K. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Front. Neurosci.* **2020**, *14*, 653.
49. Lemaire, Q.; Cordone, L.; Castagnetti, A.; Novac, P.E.; Courtois, J.; Miramond, B. An Analytical Estimation of Spiking Neural Networks Energy Efficiency. In Proceedings of the International Conference on Artificial Neural Networks (ICANN), Heraklion, Greece, 26–29 September 2023; pp. 585–597. https://doi.org/10.1007/978-3-031-30105-6_48.
50. Shen, S.; Zhang, R.; Wang, C.; Huang, R.; Tuerhong, A.; Guo, Q.; Lu, Z.; Zhang, J.; Leng, L. Evolutionary Spiking Neural Networks: A Survey. *Memetic Comput.* **2024**, *16*, 123–145. https://doi.org/10.1007/s41965-024-00156-x.
51. Bi, G.Q.; Poo, M.M. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* **1998**, *18*, 10464–10472.
52. Mozafari, M.; Ganjtabesh, M.; Nowzari-Dalini, A.; Thorpe, S.J.; Masquelier, T. Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognit.* **2019**, *94*, 87–95.
53. Amir, A.; Taba, B.; Berg, D.; Melano, T.; McKinstry, J.; Nolfo, C.D.; Nayak, T.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; et al. A low power, fully event-based gesture recognition system. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 7388–7397. IEEE: Piscataway, NJ, USA. https://doi.org/10.1109/CVPR.2017.781.
54. Massa, R.; Marchisio, A.; Martina, M.; Shafique, M. An efficient spiking neural network for recognizing gestures with a DVS camera on the Loihi neuromorphic processor. *arXiv* **2020**, arXiv:2006.09985. Available online: https://arxiv.org/abs/2006.09985 (accessed on 26 October 2025). https://doi.org/10.48550/arXiv.2006.09985.
55. Ma, S.; Pei, J.; Zhang, W.; Wang, G.; Feng, D.; Yu, F.; Song, C.; Qu, H.; Ma, C.; Lu, M.; et al. Neuromorphic Computing Chip with Spatiotemporal Elasticity for Multi-Intelligent-Tasking Robots. *Sci. Robot.* **2022**, *7*, eabk2948. https://doi.org/10.1126/scirobotics.abk2948.

56. Stewart, K.; Orchard, G.; Shrestha, S.B.; Neftci, E. Online few-shot gesture learning on a neuromorphic processor. *arXiv* **2020**, arXiv:2008.01151. Available online: https://arxiv.org/abs/2008.01151 (accessed on 26 October 2025). https://doi.org/10.48550/arXiv.2008.01151.
57. Bartolozzi, C.; Indiveri, G.; Donati, E. Embodied Neuromorphic Intelligence. *Nat. Commun.* **2022**, *13*, 1–14. https://doi.org/10.1038/s41467-022-28487-2.
58. Leitão, D.; Cunha, R.; Lemos, J.M. Adaptive Control of Quadrotors in Uncertain Environments. *Eng* **2024**, *5*, 544–561. https://doi.org/10.3390/eng5020030.
59. Velarde-Gomez, S.; Giraldo, E. Nonlinear Control of a Permanent Magnet Synchronous Motor Based on State Space Neural Network Model Identification and State Estimation by Using a Robust Unscented Kalman Filter. *Eng* **2025**, *6*, 30. https://doi.org/10.3390/eng6020030.
60. Tan, C.; Šarlija, M.; Kasabov, N. NeuroSense: Short-Term Emotion Recognition and Understanding Based on Spiking Neural Network Modelling of Spatio-Temporal EEG Patterns. *Neurocomputing* **2021**, *434*, 137–148. https://doi.org/10.1016/j.neucom.2020.12.098.
61. Yang, G.; Kang, Y.; Charlton, P.H.; Kyriacou, P.A.; Kim, K.K.; Li, L.; Park, C. Energy-Efficient PPG-Based Respiratory Rate Estimation Using Spiking Neural Networks. *Sensors* **2024**, *24*, 3980. https://doi.org/10.3390/s24123980.
62. Kumar, N.; Tang, G.; Yoo, R.; Michmizos, K.P. Decoding EEG with Spiking Neural Networks on Neuromorphic Hardware. Transactions on Machine Learning Research. 2022; pp. 1–15. Available online: https://openreview.net/forum?id=ZPBJPGX3Bz (accessed on 26 October 2025).
63. Garcia-Palencia, O.; Fernandez, J.; Shim, V.; Kasabov, N.K.; Wang, A.; the Alzheimer's Disease Neuroimaging Initiative. Spiking Neural Networks for Multimodal Neuroimaging: A Comprehensive Review of Current Trends and the NeuCube Brain-Inspired Architecture. *Bioengineering* **2025**, *12*, 628. https://doi.org/10.3390/bioengineering12060628.
64. Ayasi, B.; Vázquez, I.X.; Saleh, M.; Garcia-Vico, A.M.; Carmona, C.J. Application of Spiking Neural Networks and Traditional Artificial Neural Networks for Solar Radiation Forecasting in Photovoltaic Systems in Arab Countries. *Neural Comput. Appl.* **2025**, *37*, 9095–9127. https://doi.org/10.1007/s00521-025-11066-z.
65. Sopeña, J.M.G.; Pakrashi, V.; Ghosh, B. A Spiking Neural Network Based Wind Power Forecasting Model for Neuromorphic Devices. *Energies* **2022**, *15*, 7256. https://doi.org/10.3390/en15197256.
66. Thangaraj, V.K.; Nachimuthu, D.S.; Francis, V.A.R. Wind Speed Forecasting at Wind Farm Locations with a Unique Hybrid PSO-ALO Based Modified Spiking Neural Network. *Energy Syst.* **2023**, *16*, 713–741. https://doi.org/10.1007/s12667-023-00607-x.
67. AbouHassan, I.; Kasabov, N.; Bankar, T.; Garg, R.; Sen Bhattacharya, B. PAMeT-SNN: Predictive Associative Memory for Multiple Time Series based on Spiking Neural Networks with Case Studies in Economics and Finance. *TechRxiv* **2023**. https://doi.org/10.36227/techrxiv.24063975.v1.
68. Joseph, G.V.; Pakrashi, V. Spiking Neural Networks for Structural Health Monitoring. *Sensors* **2022**, *22*, 9245. https://doi.org/10.3390/s22239245.
69. Reid, D.; Hussain, A.J.; Tawfik, H. Financial Time Series Prediction Using Spiking Neural Networks. *PLoS ONE* **2014**, *9*, e103656. https://doi.org/10.1371/journal.pone.0103656.
70. Du, X.; Tong, W.; Jiang, L.; Yu, D.; Wu, Z.; Duan, Q.; Deng, S. SNN-IoT: Efficient Partitioning and Enabling of Deep Spiking Neural Networks in IoT Services. *IEEE Trans. Serv. Comput.* 2025, *in press*. https://doi.org/10.1109/TSC.2025.3592380.
71. Li, H.; Tu, B.; Liu, B.; Li, J.; Plaza, A. Adaptive Feature Self-Attention in Spiking Neural Networks for Hyperspectral Classification. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 1–15. https://doi.org/10.1109/TGRS.2024.3516742.
72. Chunduri, R.K.; Perera, D.G. Neuromorphic Sentiment Analysis Using Spiking Neural Networks. *Sensors* **2023**, *23*, 7701. https://doi.org/10.3390/s23187701.
73. Schuman, C.D.; Plank, J.S.; Bruer, G.; Anantharaj, J. Non-Traditional Input Encoding Schemes for Spiking Neuromorphic Systems. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2019), Budapest, Hungary, 14-19 July 2019; pp. 1–10. https://doi.org/10.1109/IJCNN.2019.8852139.
74. Datta, G.; Liu, Z.; Abdullah-Al Kaiser, M.; Kundu, S.; Mathai, J.; Yin, Z.; Jacob, A.P.; Jaiswal, A.R.; Beerel, P.A. In-sensor and neuromorphic computing are all you need for energy-efficient computer vision. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10094830.
75. Guo, W.; Fouda, M.E.; Eltawil, A.M.; Salama, K.N. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Front. Neurosci.* **2021**, *15*, 638474.
76. Mostafa, H. Supervised learning based on temporal coding in spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 1–9. https://doi.org/10.1109/TNNLS.2017.2726060.
77. Sakemi, Y.; Morino, K.; Morie, T.; Aihara, K. A supervised learning algorithm for multilayer spiking neural networks based on temporal coding toward energy-efficient VLSI processor design. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 394–408. https://doi.org/10.1109/TNNLS.2021.3095068.

78. Kim, Y.; Park, H.; Moitra, A.; Bhattacharjee, A.; Venkatesha, Y.; Panda, P. Rate coding or direct coding: Which one is better for accurate, robust, and energy-efficient spiking neural networks? In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2022), Singapore, 23–27 May 2022; pp. 71–75. https://doi.org/10.1109/ICASSP43922.2022.9747906.
79. Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S.T.; Sanyal, A. Temporal-coded deep spiking neural network with easy training and robust performance. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2021), Virtual Event, 2–9 February 2021; Volume 35, pp. 11143–11151. https://doi.org/10.1609/aaai.v35i12.17329.
80. Gautrais, J.; Thorpe, S. Rate coding versus temporal order coding: A theoretical approach. *Biosystems* **1998**, *48*, 57–65.
81. Duarte, R.C.; Uhlmann, M.; Van Den Broek, D.; Fitz, H.; Petersson, K.; Morrison, A. Encoding symbolic sequences with spiking neural reservoirs. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2018), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8. https://doi.org/10.1109/IJCNN.2018.8489224.
82. Bonilla, L.; Gautrais, J.; Thorpe, S.; Masquelier, T. Analyzing time-to-first-spike coding schemes: A theoretical approach. *Front. Neurosci.* **2022**, *16*, 971937.
83. Averbeck, B.B.; Latham, P.E.; Pouget, A. Neural correlations, population coding and computation. *Nat. Rev. Neurosci.* **2006**, *7*, 358–366. https://doi.org/10.1038/nrn1888.
84. Pan, Z.; Wu, J.; Zhang, M.; Li, H.; Chua, Y. Neural population coding for effective temporal classification. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2019), Budapest, Hungary, 14–19 July 2019; pp. 1–8. https://doi.org/10.1109/IJCNN.2019.8852348.
85. Cheung, K.F.; Tang, P.Y. Sigma-delta modulation neural networks. In Proceedings of the IEEE International Conference on Neural Networks (ICNN 1993), San Francisco, CA, USA, 28 March–1 April 1993; pp. 489–493. https://doi.org/10.1109/ICNN.1993.298629.
86. Yousefzadeh, A.; Hosseini, S.; Holanda, P.; Leroux, S.; Werner, T.; Serrano-Gotarredona, T.; Simoens, P. Conversion of synchronous artificial neural network to asynchronous spiking neural network using sigma-delta quantization. In Proceedings of the IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS 2019), Hsinchu, Taiwan, 18–20 March 2019; pp. 81–85. https://doi.org/10.1109/AICAS.2019.8771510.
87. Nasrollahi, S.A.; Syutkin, A.; Cowan, G. Input-layer neuron implementation using delta-sigma modulators. In Proceedings of the 2022 20th IEEE Interregional NEWCAS Conference (NEWCAS 2022), Québec City, QC, Canada, 19–22 June 2022; pp. 533–537. https://doi.org/10.1109/NEWCAS52662.2022.9842225.
88. Nair, M.V.; Indiveri, G. An ultra-low power sigma-delta neuron circuit. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2019), Sapporo, Japan, 26–29 May 2019; pp. 1–5. https://doi.org/10.1109/ISCAS.2019.8702338.
89. Izhikevich, E.M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **2003**, *14*, 1569–1572. https://doi.org/10.1109/TNN.2003.820440.
90. Eyherabide, H.G.; Rokem, A.; Herz, A.V.; Samengo, I. Bursts generate a non-reducible spike-pattern code. *Front. Neurosci.* **2009**, *3*, 490.
91. O'Keefe, J.; Recce, M.L. Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus* **1993**, *3*, 317–330.
92. Montemurro, M.A.; Rasch, M.J.; Murayama, Y.; Logothetis, N.K.; Panzeri, S. Phase-of-firing coding of natural visual stimuli in primary visual cortex. *Curr. Biol.* **2008**, *18*, 375–380.
93. Masquelier, T.; Hugues, E.; Deco, G.; Thorpe, S.J. Oscillations, Phase-of-Firing Coding, and Spike Timing-Dependent Plasticity: An Efficient Learning Scheme. *J. Neurosci.* **2009**, *29*, 13484–13493.
94. Wang, Z.; Yu, N.; Liao, Y. Activeness: A Novel Neural Coding Scheme Integrating the Spike Rate and Temporal Information in the Spiking Neural Network. *Electronics* **2023**, *12*, 3992.
95. Qiu, X.; Zhu, R.J.; Chou, Y.; Wang, Z.; Deng, L.J.; Li, G. Gated attention coding for training high-performance and efficient spiking neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2024), Vancouver, BC, Canada, 20–27 February 2024; Volume 38, pp. 601–610. https://doi.org/10.1609/aaai.v38i1.27816.
96. Paugam-Moisy, H.; Bohte, S.M. Computing with Spiking Neuron Networks. In *Handbook of Natural Computing*; Rozenberg, G., Back, T., Kok, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 335–376. https://doi.org/10.1007/978-3-540-92910-9_10.
97. Hodgkin, A.L.; Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **1952**, *117*, 500.
98. Brette, R.; Rudolph, M.; Carnevale, T.; Hines, M.; Beeman, D.; Bower, J.M.; Diesmann, M.; Morrison, A.; Goodman, P.H.; Harris, F.C., Jr.; et al. Simulation of networks of spiking neurons: A review of tools and strategies. *J. Comput. Neurosci.* **2007**, *23*, 349–398.
99. Indiveri, G.; Liu, S.C. Neuromorphic VLSI circuits for spike-based computation. *Proc. IEEE* **2011**, *99*, 2414–2435.
100. Gerstner, W.; van Hemmen, J.L. Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns. *Biol. Cybern.* **1993**, *69*, 503–515.
101. Lapicque, L. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Générale* **1907**, *9*, 620–635.

102. Burkitt, A.N. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biol. Cybern.* **2006**, *95*, 1–19.
103. Brunel, N.; Van Rossum, M.C.W. Lapicque's 1907 paper: From frogs to integrate-and-fire. *Biol. Cybern.* **2007**, *97*, 337–339. https://doi.org/10.1007/s00422-007-0190-0.
104. Benda, J.; Herz, A.V.M. A universal model for spike-frequency adaptation. *Neural Comput.* **2003**, *15*, 2523–2564.
105. Fourcaud-Trocmé, N.; Hansel, D.; Van Vreeswijk, C.; Brunel, N. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *J. Neurosci.* **2003**, *23*, 11628–11640.
106. Gerstner, W.; Brette, R. Adaptive exponential integrate-and-fire model. *Scholarpedia* **2009**, *4*, 8427. https://doi.org/10.4249/scholarpedia.8427.
107. Makhlooghpour, A.; Soleimani, H.; Ahmadi, A.; Zwolinski, M.; Saif, M. High-accuracy implementation of adaptive exponential integrate-and-fire neuron model. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2016), Vancouver, BC, Canada, 24–29 July 2016; pp. 192–197. https://doi.org/10.1109/IJCNN.2016.7727255.
108. Haghiri, S.; Ahmadi, A. A Novel Digital Realization of AdEx Neuron Model. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 1444–1451. https://doi.org/10.1109/TCSII.2019.2938180.
109. Ahmadi, A.; Zwolinski, M. A modified Izhikevich model for circuit implementation of spiking neural networks. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2010), Paris, France, 30 May–2 June 2010; pp. 4253–4256. https://doi.org/10.1109/ISCAS.2010.5537139.
110. Izhikevich, E.M. Resonate-and-fire neurons. *Neural Netw.* **2001**, *14*, 883–894.
111. Higuchi, S.; Kairat, S.; Bohte, S.M.; Otte, S. Balanced resonate-and-fire neurons. *arXiv* **2024**, arXiv:2402.14603. Available online: https://arxiv.org/abs/2402.14603 (accessed on 26 October 2025).
112. Lehmann, H.M.; Hille, J.; Grassmann, C.; Issakov, V. Direct Signal Encoding with Analog Resonate-and-Fire Neurons. *IEEE Access* **2023**, *11*, 71985–71995. https://doi.org/10.1109/ACCESS.2023.3278098.
113. Leigh, A.J.; Heidarpur, M.; Mirhassani, M. A Resource-Efficient and High-Accuracy CORDIC-Based Digital Implementation of the Hodgkin–Huxley Neuron. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2023**, *31*, 1377–1388.
114. Devi, M.; Choudhary, D.; Garg, A.R. Information processing in extended Hodgkin–Huxley neuron model. In Proceedings of the 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE 2020), Jaipur, India, 7–8 February 2020; pp. 176–180. https://doi.org/10.1109/ICETCE48199.2020.9091772.
115. Intel Neuromorphic Computing Lab. Lava: A Software Framework for Neuromorphic Computing. 2021. Available online: https://lava-nc.org (accessed on 26 October 2025).
116. Zambrano, D.; Bohte, S.M. Fast and efficient asynchronous neural computation with adapting spiking neural networks. *arXiv* **2016**, arXiv:1609.02053. Available online: https://arxiv.org/abs/1609.02053 (accessed on 26 October 2025).
117. Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Shi, L. Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. *Front. Neurosci.* **2018**, *12*, 331. https://doi.org/10.3389/fnins.2018.00331.
118. Lee, J.H.; Delbruck, T.; Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Front. Neurosci.* **2016**, *10*, 508.
119. Shrestha, S.B.; Orchard, G. SLAYER: Spike layer error reassignment in time. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018; Volume 31.
120. Lian, S.; Shen, J.; Liu, Q.; Wang, Z.; Yan, R.; Tang, H. Learnable surrogate gradient for direct training spiking neural networks. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23), Macao, SAR, China, 19–25 August 2023; pp. 3002–3010. https://doi.org/10.24963/IJCAI.2023/335.
121. Sjöström, J.; Gerstner, W. Spike-timing-dependent plasticity. *Scholarpedia* **2010**, **5**, 1362. https://doi.org/10.4249/scholarpedia.1362.
122. Bohte, S.; Kok, J.; Poutré, J. SpikeProp: Backpropagation for Networks of Spiking Neurons. In Proceedings of the 8th European Symposium on Artificial Neural Networks, ESANN 2000, Bruges, Belgium, 26–28 April 2000; Volume 48, pp. 419–424.
123. Zenke, F.; Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput.* **2018**, *30*, 1514–1541.
124. Wunderlich, T.C.; Pehle, C. Event-based backpropagation can compute exact gradients for spiking neural networks. *Sci. Rep.* **2021**, *11*, 12829.
125. Gautam, A.; Kohno, T. Adaptive STDP-Based On-Chip Spike Pattern Detection. *Front. Neurosci.* **2023**, *17*, 1203956. https://doi.org/10.3389/fnins.2023.1203956.
126. Li, S. aSTDP: A more biologically plausible learning. *arXiv* **2022**, arXiv:2206.14137. Available online: https://arxiv.org/abs/2206.14137 (accessed on 26 October 2025).
127. Paredes-Vallès, F.; Scheper, K.Y.; Croon, G.C.D. Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2051–2064. https://doi.org/10.1109/TPAMI.2019.2903179.
128. Caporale, N.; Dan, Y. Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annu. Rev. Neurosci.* **2008**, *31*, 25–46. https://doi.org/10.1146/annurev.neuro.31.060407.125639.

129. Ponulak, F. *ReSuMe—New Supervised Learning Method for Spiking Neural Networks*; Technical Report; Institute of Control and Information Engineering, Poznań University of Technology: Poznań, Poland, 2005.
130. Bellec, G.; Scherr, F.; Hajek, E.; Salaj, D.; Legenstein, R.; Maass, W. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv* **2019**, arXiv:1901.09049. Available online: https://arxiv.org/abs/1901.09049 (accessed on 26 Octobe
131. Liu, F.; Zhao, W.; Chen, Y.; Wang, Z.; Yang, T.; Jiang, L. SSTDP: Supervised Spike Timing Dependent Plasticity for Efficient Spiking Neural Network Training. *Front. Neurosci.* **2021**, *15*, 756876.
132. Diehl, P.U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.C.; Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2015), Killarney, Ireland, 12–17 July 2015; pp. 1–8. https://doi.org/10.1109/IJCNN.2015.7280696.
133. Rueckauer, B.; Lungu, I.A.; Hu, Y.; Pfeiffer, M.; Liu, S.C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* **2017**, *11*, 682.
134. Cao, Y.; Chen, Y.; Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* **2015**, *113*, 54–66. https://doi.org/10.1007/s11263-014-0788-3.
135. Hunsberger, E.; Eliasmith, C. Spiking deep networks with LIF neurons. *arXiv* **2015**, arXiv:1510.08829. Available online: https://arxiv.org/abs/1510.08829 (accessed on 26 October 2025). https://doi.org/10.48550/arXiv.1510.08829.
136. Han, B.; Srinivasan, G.; Roy, K. RMP-SNN: Residual membrane potential neuron for enabling deeper, high-accuracy, and low-latency spiking neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020), Seattle, WA, USA, 14–19 June 2020; pp. 13558–13567. https://doi.org/10.1109/CVPR42600.2020.01358.
137. Bu, T.; Fang, W.; Ding, J.; Dai, P.; Yu, Z.; Huang, T. Optimal ANN–SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In Proceedings of the 10th International Conference on Learning Representations (ICLR 2022), Virtual Conference, 25–29 April 2022; OpenReview: Online. Available online: https://openreview.net/forum?id=7B3IJMM1kXq (accessed on 26 October 2025).
138. LeCun, Y.; Cortes, C.; Burges, C.J. *The MNIST Database of Handwritten Digits*; Technical Report; AT&T Labs: Florham Park, NJ, USA, 1998. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 26 October 2025).
139. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report UTML TR 2009; University of Toronto: Toronto, ON, Canada, 2009.
140. Gaurav, R.; Tripp, B.; Narayan, A. Spiking approximations of the MaxPooling operation in deep SNNs. *arXiv* **2022**, arXiv:2205.07076. Available online: https://arxiv.org/abs/2205.07076 (accessed on 26 October 2025). https://doi.org/10.48550/arXiv.2205.07076.
141. Ponulak, F.; Kasinski, A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiol. Exp.* **2011**, *71*, 409–433.
142. Xin, J.; Embrechts, M.J. Supervised learning with spiking neural networks. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2001), Washington, DC, USA, 15–19 July 2001; Volume 3, pp. 1772–1777. https://doi.org/10.1109/IJCNN.2001.938438.
143. Ponulak, F.; Kasiński, A. Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting. *Neural Comput.* **2010**, *22*, 467–510.
144. Xu, Y.; Zeng, X.; Zhong, S. A New Supervised Learning Algorithm for Spiking Neurons. *Neural Comput.* **2013**, *25*, 1472–1511.
145. Xu, Y.; Zeng, X.; Han, L.; Yang, J. A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Netw.* **2013**, *43*, 99–113. https://doi.org/10.1016/j.neunet.2013.02.003.
146. Ahmed, F.Y.; Shamsuddin, S.M.; Hashim, S.Z.M. Improved spikeprop for using particle swarm optimization. *Math. Probl. Eng.* **2013**, *2013*, 257085.
147. Yu, Q.; Tang, H.; Tan, K.C.; Yu, H. A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing* **2014**, *138*, 3–13. https://doi.org/10.1016/j.neucom.2013.06.052.
148. Huh, D.; Sejnowski, T.J. Gradient descent for spiking neural networks. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, QC, Canada, 3–8 December 2018; Volume 31.
149. Datta, G.; Kundu, S.; Beerel, P.A. Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2021), Shenzhen, China, 18–22 July 2021; pp. 1–8. https://doi.org/10.1109/IJCNN52387.2021.9534121.
150. Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; Li, G. Going deeper with directly trained larger spiking neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2021), Virtual Event, 2–9 February 2021; Volume 35, pp. 11062–11070. https://doi.org/10.1609/aaai.v35i12.17320.
151. Shi, X.; Hao, Z.; Yu, Z. SpikingResFormer: Bridging ResNet and Vision Transformer in spiking neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024), Seattle, WA, USA, 17–21 June 2024; pp. 5610–5619. https://doi.org/10.1109/CVPR52733.2024.00541.

152. Zhou, C.; Zhang, H.; Zhou, Z.; Yu, L.; Huang, L.; Fan, X.; Yuan, L.; Ma, Z.; Zhou, H.; Tian, Y. Qkformer: Hierarchical spiking transformer using qk attention. *arXiv* **2024**, arXiv:2403.16552. Available online: http://arxiv.org/abs/2403.16552 (accessed on).
153. Sorbaro, M.; Liu, Q.; Bortone, M.; Sheik, S. Optimizing the Energy Consumption of Spiking Neural Networks for Neuromorphic Applications. *Front. Neurosci.* **2020**, *14*, 516916. https://doi.org/10.3389/fnins.2020.00662.
154. Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; Roy, K. Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* **2019**, *13*, 95. https://doi.org/10.3389/fnins.2019.00095.
155. Kucik, A.S.; Meoni, G. Investigating spiking neural networks for energy-efficient on-board AI applications: A case study in land cover and land use classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2021), Virtual Event, 19–25 June 2021; pp. 2020–2030. https://doi.org/10.1109/CVPRW53098.2021.00208.
156. Applied Brain Research. 2024. KerasSpiking—Estimating Model Energy. Available online: https://www.nengo.ai/keras-spiking/examples/model-energy.html (accessed on 12 April 2024).
157. Fang, W.; Chen, Y.; Ding, J.; Yu, Z.; Masquelier, T.; Chen, D.; Yu, Z.; Zhou, H.; Tian, Y. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Sci. Adv.* **2023**, *9*, eadi1480. https://doi.org/10.1126/sciadv.adi1480.
158. Pehle, C.G.; Pedersen, J.E. Norse—A Deep Learning Library for Spiking Neural Networks. 2021. Available online: https://doi.org/10.5281/zenodo.4422025 (accessed on 26 October 2025).
159. Ali, H.A.H.; Dabbous, A.; Ibrahim, A.; Valle, M. Assessment of recurrent spiking neural networks on neuromorphic accelerators for naturalistic texture classification. In Proceedings of the 2023 18th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME 2023), Valencia, Spain, 18–21 June 2023; pp. 145–148. https://doi.org/10.1109/PRIME58259.2023.10161772.
160. Herbozo Contreras, L.F.; Huang, Z.; Yu, L.; Nikpour, A.; Kavehei, O. Biologically plausible algorithm for seizure detection: Toward AI-enabled electroceuticals at the edge. *APL Mach. Learn.* **2024**, *2*, 026113. https://doi.org/10.1063/5.0192875.
161. Takaghaj, S.M.; Sampson, J. Rouser: Robust SNN training using adaptive threshold learning. *arXiv* **2024**, arXiv:2407.19566. Available online: https://arxiv.org/abs/2407.19566 (accessed on 26 October 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.