
Neuromorphic Edge Intelligence in Collaborative Pipelines: An Energy-Latency Trade-off Analysis

(IJCSNT) International Journal of
Communication Systems and Network
Technologies

ISSN Number: 2053-6283

Volume 15, Issue No. 01, 48–63

©The Author(s) 2026

<https://journals.mirlabs.in/index.php/ijcsnt>

DOI-10.18486/ijcsnt/15.1.003



Nghia Dinh¹, Vinh Truong Hoang¹, Bay Nguyen Van¹, Huy Nguyen Ngoc¹, Kiet Tran-Trung¹

Abstract

Neuromorphic processors carry out spiking neural network (SNN) inference through event-driven, spike-based computation, drawing only milliwatt-scale power while keeping latency in the low-millisecond range. These properties make them well suited for always-on edge intelligence, yet we still lack a systematic framework for deciding when neuromorphic inference at the device level actually outperforms offloading to a conventional edge or cloud accelerator within a collaborative pipeline. We address that gap by proposing a four-tier collaborative architecture that explicitly separates the neuromorphic edge from the general-purpose edge and by deriving an energy-latency-accuracy offloading decision boundary between these tiers. Our survey covers five neuromorphic platforms (Loihi 2, SpiNNaker 2, Akida, TrueNorth, DYNAP-SE) and three conventional edge processors, drawing on published specification and benchmark data. We identify energy advantages of up to two orders of magnitude for spiking inference on sparse workloads, alongside an accuracy gap of several percentage points that constrains where deployment is practical. Three application scenarios (autonomous vehicle perception, industrial IoT predictive maintenance, and wearable health monitoring) ground our framework in concrete deployment contexts and show how the offloading boundary shifts across domains. We further analyze the simulation-to-hardware gap in SNN training as a factor governing the accuracy constraint. The framework gives system designers an analytical tool for reasoning about neuromorphic inference placement within heterogeneous edge and cloud deployments.

Keywords

Neuromorphic Computing, Spiking Neural Networks, Edge Intelligence, Collaborative Inference, Energy Efficiency

Received: 27 December 2025; **Revised:** 15 April 2026; **Accepted:** 26 April 2026;

Published: 30 April 2026;

¹AI Lab, Faculty of Information Technology, Ho Chi Minh City Open University, Ho Chi Minh City, Vietnam

Corresponding author:

Email: vinh.th@ou.edu.vn



© 2026 by Nghia Dinh, Vinh Truong Hoang, Bay Nguyen Van, Huy Nguyen Ngoc and Kiet Tran-Trung

Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license, (<http://creativecommons.org/licenses/by/4.0/>). This work is licensed under a Creative Commons Attribution 4.0 International License

Introduction

Over the past decade, edge computing has pushed artificial intelligence off centralized data centers and onto resource-constrained devices that run under tight latency and energy budgets^[1,2]. When we look at conventional edge accelerators like the NVIDIA Jetson platform and Google Coral TPU, they deliver competitive inference throughput, but their reliance on clock-driven matrix operations sets a power floor that limits deployment in always-on, battery-powered scenarios^[3]. Neuromorphic processors offer us a different path. These chips mimic the spike-based, event-driven computation found in biological neural circuits, and they activate only neurons that receive input events. In practical terms, hardware like Intel Loihi 2, SpiNNaker 2, and BrainChip Akida can sustain inference at milliwatt-scale power while keeping our response times in the low-millisecond range^[4–6].

To our knowledge, there is still no systematic framework for evaluating when neuromorphic inference at the device tier outperforms offloading to a conventional edge or cloud server. Existing surveys catalog hardware specifications or benchmark individual chips^[7–9], but as we read them, they rarely quantify the offloading decision in terms of joint energy, latency, and accuracy trade-offs across the complete device-to-edge-to-cloud pipeline. Recent work on collaborative edge intelligence^[2,3] acknowledges the multi-tier nature of modern deployments, yet it typically treats neuromorphic and conventional edge processors as interchangeable. We think that assumption overlooks something basic: the architectural differences in how each tier consumes energy and introduces latency.

We bridge that gap by proposing an analytical framework that explicitly models a four-tier collaborative pipeline (sensor/device, neuromorphic edge, general-purpose edge, and cloud) and derives an offloading decision boundary. Put simply, we want to know when it pays to run a spiking network on chip versus shipping data to a more powerful remote accelerator. Our contributions span four areas. First, we introduce a four-tier pipeline architecture that distinguishes neuromorphic edge processors from general-purpose edge accelerators, enabling fine-grained analysis of where spiking inference fits within collaborative systems (see the Four-Tier Pipeline Architecture section). Second, we develop an energy-latency-accuracy model that captures the cost of local neuromorphic inference versus data transmission and remote processing, yielding a closed-form offloading decision boundary (see the Energy-Latency Model and Offloading Decision Boundary sections). Third, we compare five neuromorphic platforms against three conventional platforms using published benchmark data, visualizing the results as an energy-latency Pareto surface (see the Comparative Analysis section). Finally, three application case studies (autonomous vehicle perception, industrial predictive maintenance IoT, and wearable health monitoring) ground our framework in concrete deployment scenarios (see the Application Case Studies section).

The rest of the paper proceeds as follows. We review spiking neural networks, neuromorphic hardware, and collaborative edge computing in the Background and Related Work section. The Proposed Analytical Framework section lays out our analytical framework, and the Comparative Analysis section follows with the comparative analysis. We then turn to application case studies in the Application Case Studies section. The Discussion section takes up the simulation-to-hardware gap, limitations, and open challenges, and the Conclusion and Future Work section closes with future research directions.

Background and Related Work

Spiking Neural Networks

Spiking neural networks (SNNs) belong to what is often called the third generation of neural network models^[10]. Where conventional networks pass continuous-valued activations between layers, SNNs encode information in the precise timing of discrete spike events. We can describe the core mechanism simply: each neuron accumulates incoming spikes into a membrane potential, and once that potential crosses a threshold, the neuron fires its own spike and resets. Among the many neuron models out there, the leaky integrate-and-fire (LIF) abstraction is the one we encounter most often in hardware implementations, largely because its computational overhead stays low^[11]. This is the neuron model that dominates across the platforms in our survey.

Two properties make SNNs attractive for edge deployment, and both are central to our analysis. The first is event-driven sparsity: because neurons stay silent until they receive input events, energy consumption scales with how active the input is rather than with the size of the network^[12]. We find this especially relevant for always-on monitoring tasks where most of the time nothing is happening. The second property is temporal coding: spike timing carries information that rate-coded ANNs must approximate through extra time steps. What this gives us in practice is the ability for SNNs to reach a classification decision in fewer operations when the data has temporal structure (e.g., audio, radar pulses, and biosignals)^[13].

Training SNNs is harder than training conventional ANNs because the spike function is non-differentiable. We can identify three main strategies to work around this. ANN-to-SNN conversion maps trained ANN weights to an equivalent SNN, but the SNN needs many time steps for accurate rate approximation, which adds latency^[14]. Surrogate gradient methods replace the true spike derivative with a smooth approximation during backpropagation^[15], and these can get us closer to ANN-level accuracy without inflating time-step counts. On-chip plasticity rules like spike-timing-dependent plasticity (STDP) learn directly on neuromorphic hardware, which avoids the weight-transfer problem altogether, but they currently struggle to scale to deep architectures^[16]. The accuracy gap between these training approaches and their ANN counterparts is a key factor in the energy-latency trade-off that we analyze in the Proposed Analytical Framework section.

Neuromorphic Hardware Platforms

Five neuromorphic processors span the current design space, covering everything from purely digital to mixed-signal architectures. We survey each one below.

Intel Loihi 2 integrates up to 1 million programmable spiking neurons on a single die, connected through a hierarchical mesh network^[4,17]. The chip supports on-chip learning via three-factor plasticity rules, something we find distinctive among current platforms. For keyword-spotting workloads, it achieves inference power below 100 mW with latencies below 5 ms per utterance^[18]. In our framework, these numbers place Loihi 2 in a favorable region of the energy-latency landscape.

Spinnaker 2 targets large-scale neural simulation through a mesh of ARM-based processing elements, each capable of emulating thousands of LIF neurons in real time^[19]. From our perspective, what sets the second-generation chip apart is a set of hardware accelerators for exponential and logarithmic functions that cut the energy per neuron by an order of magnitude compared to its predecessor^[5]. We

find SpiNNaker 2 well suited for simulations requiring large neuron populations. Its higher power draw compared to smaller chips, however, is a trade-off we have to weigh.

BrainChip Akida stands out as one of the first commercially available neuromorphic processors built specifically for edge deployment^[6]. It runs a convolutional SNN pipeline that processes both event-camera and standard image inputs at low power. Vendor-reported accuracies sit between 85% and 92% on vision benchmarks, with latencies below 10 ms^[6]. We find Akida’s commercial availability worth highlighting: it gives us a platform we can actually procure and integrate into our pipeline analysis without relying on research prototypes.

IBM TrueNorth introduced the one-million-neuron digital neuromorphic processor back in 2014^[20]. It is organized as a tiled array of 4096 neurosynaptic cores, each implementing 256 neurons with programmable axon types. We note that at full utilization, the chip draws roughly 65 mW, which works out to about 26 pJ per synaptic operation. While TrueNorth is now over a decade old, we keep it in our comparison because its architecture set several benchmarks that later designs have aimed to surpass.

DYNAP-SE from the Institute of Neuroinformatics takes a mixed-signal approach: analog circuits emulate neuron and synapse dynamics while digital routers handle communication between cores^[21]. This design yields extremely low power per spike, but it comes with a catch. Device mismatch in the analog circuitry has to be compensated during network configuration, and in our experience reviewing the literature, this adds a calibration step that purely digital designs do not require. We include DYNAP-SE because it represents the low-power extreme of the design space we survey.

For comparison, we also bring in three conventional edge processors. The NVIDIA Jetson Orin (GPU-based, 15 to 60W) and Google Coral Edge TPU (INT8 accelerator, 2 to 4W) represent the high-performance end of the edge spectrum, while STM32 microcontrollers (sub-watt, limited model capacity) sit at the low end. These baselines let us contextualize the energy advantage of neuromorphic hardware. They also reveal the latency regimes in which conventional processors remain competitive, as we show in the Comparative Analysis section.

Collaborative Intelligence and Edge Computing

Collaborative intelligence distributes inference across multiple tiers of computing resources, trading latency, energy, accuracy, and bandwidth according to task requirements^[2,3]. In most of the literature, we see a three-tier hierarchy: device, edge server, and cloud. Early exit mechanisms or model partitioning decide how much computation each tier performs^[2]. The idea makes intuitive sense: handle what you can locally, and escalate the rest.

The trouble with this three-tier view, as we see it, is that it lumps two fundamentally different processor classes into a single “edge” layer. Consider the contrast: a neuromorphic chip running spiking inference at 10 mW operates under constraints and performance envelopes that differ by orders of magnitude from a GPU-equipped edge server consuming 60W. Treating both as one tier hides what we think is the critical design question: should a given workload stay on the neuromorphic device, or should we forward it to a more powerful but more energy-hungry conventional accelerator?

We therefore adopt a four-tier collaborative model, and each tier serves a distinct role in our pipeline. The first tier handles raw data acquisition with minimal preprocessing (i.e., analog front-ends, event cameras, MEMS sensors). At the second tier, the neuromorphic edge, we run event-driven SNN inference on dedicated neuromorphic hardware for low-complexity, latency-critical filtering and classification tasks. The third tier, the general-purpose edge, performs DNN inference on GPU or TPU accelerators,

taking on heavier model execution, data aggregation, and local model updates. Finally, the fourth tier is the cloud, handling large-scale training, global model management, and analytics workloads that can tolerate higher latency.

This four-tier decomposition lines up with recent observations that heterogeneous edge deployments benefit from explicit hardware-aware task placement^[8,12]. We formalize the architecture in the Four-Tier Pipeline Architecture section and derive our decision boundary between tiers 2 and 3 in the Offloading Decision Boundary section.

Related Work

Several threads in the survey literature converge on the topics we address, but none of them puts all the pieces together in the way our work attempts to. Schuman et al.^[7] and Christensen et al.^[8] map the landscape of neuromorphic algorithms and hardware, yet their focus stays on chip-level benchmarks without modeling the surrounding system pipeline. The survey by Bouvier et al.^[9] catalogs SNN hardware with an emphasis on circuit-level trade-offs rather than system-level deployment. What we miss in Roy et al.^[12], who offer a broad perspective on spike-based machine intelligence, is any quantification of offloading decisions in multi-tier settings.

On the edge computing side, the foundational frameworks for edge AI came from Zhou et al.^[2] and Deng et al.^[3], while Mao et al.^[22] formalized task offloading in mobile edge computing. These works treat edge processors as homogeneous and do not separate neuromorphic hardware from conventional hardware, a distinction we consider central to our analysis.

To our knowledge, no existing work explicitly distinguishes a neuromorphic edge tier from a general-purpose edge tier within a formal pipeline model and also derives a closed-form offloading decision boundary as a function of energy, latency, and accuracy. We address both gaps here and show how the framework plays out across three application domains.

Proposed Analytical Framework

Four-Tier Pipeline Architecture

We formalize the four-tier collaborative pipeline from the Collaborative Intelligence and Edge Computing section as a directed data-flow graph $G = (\mathcal{T}, \mathcal{E})$. The set of levels $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$ maps onto our four tiers (sensor/device, neuromorphic edge, general-purpose edge, and cloud), and each directed edge $(T_i, T_j) \in \mathcal{E}$ carries a transmission cost captured by energy E_{tx}^{ij} and latency L_{tx}^{ij} .

Fig. 1 shows what our architecture looks like in practice. Raw sensor data enters T_1 for optional preprocessing (e.g., event encoding from a dynamic vision sensor). At the neuromorphic edge level T_2 , we run an SNN that performs energy-efficient, low-latency inference on the event stream. If the SNN's confidence drops below a task-specific threshold, or if the input exceeds the SNN's capacity, the data moves on to the general-purpose edge tier T_3 for DNN-based inference. We reserve the cloud tier T_4 for training, global aggregation, and heavy analytics where round-trip latencies on the order of hundreds of milliseconds are tolerable.

The core insight, and the reason we separate T_2 from T_3 , is that they operate under fundamentally different computational models. Tier T_2 exploits event-driven sparsity: energy scales with input activity, and latency depends on how deeply spikes propagate rather than on clock frequency. Tier T_3 uses clock-driven dense matrix operations: energy scales with model size, and latency depends on throughput and

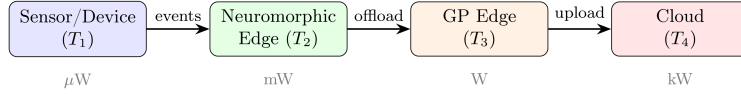


Figure 1. Four-tier collaborative pipeline. Power budgets increase from left to right; latency tolerance (allowable end-to-end delay) also increases toward the cloud. The offloading decision between T_2 and T_3 is the focus of the Offloading Decision Boundary section.

batch utilization. Because these differences are so stark, we need a separate energy-latency model for each tier. That is what we develop next.

Energy-Latency Model

We model the cost of processing a single inference task τ at each level in terms of energy and latency, and then bring in accuracy as a quality constraint.

For local neuromorphic inference at T_2 , let N denote the number of neurons in the deployed SNN, $\bar{s}$ the average firing rate (spikes per neuron per inference), and e_s the energy per synaptic operation. We write the dynamic inference energy as follows.

$$E_{\text{local}} = N \cdot \bar{s} \cdot \bar{f} \cdot e_s + P_{\text{static}} \cdot L_{\text{local}} \quad (1)$$

Here $\bar{f}$ is the average fan-in per neuron, P_{static} is the static power of the chip, and L_{local} is the inference latency that we determine from the number of SNN time steps K and the step duration Δt :

$$L_{\text{local}} = K \cdot \Delta t \quad (2)$$

The sparsity advantage of SNNs shows up through $\bar{s}$ in our model: for event-driven inputs with low activity, $\bar{s} \ll 1$, which drives E_{local} well below what a dense ANN of comparable depth would cost^[23,24]. Put simply, when the input is quiet, the spiking network barely uses any energy.

When we offload the task τ to the general-purpose edge tier T_3 , the cost splits into a transmission component and a remote computation component:

$$E_{\text{remote}} = E_{\text{tx}} + E_{\text{GP}} \quad (3)$$

$$L_{\text{remote}} = L_{\text{tx}} + L_{\text{GP}} + L_{\text{ret}} \quad (4)$$

Here E_{tx} is the energy our radio or wired interface consumes to transmit input data of size D bytes, E_{GP} is the energy consumed by the GPU or TPU for DNN inference, and L_{tx} , L_{GP} , L_{ret} denote uplink transmission, remote computation, and result return latencies. We compute the transmission energy using the standard model $E_{\text{tx}} = p_{\text{tx}} \cdot 8D/R$, where p_{tx} is the transmit power, D is the data size in bytes, R is the link data rate in bits per second, and the factor of 8 converts bytes to bits^[22].

Accuracy also enters our formulation as a binding constraint. Let A_{local} and A_{remote} represent the accuracy of the SNN and DNN models, respectively. In practice, the remote DNN model typically achieves higher accuracy than the local SNN, so $\Delta A = A_{\text{remote}} - A_{\text{local}}$ is often positive, though we cannot guarantee this for every task and metric combination. We define a minimum acceptable accuracy

$A_{\min}$ for each task τ (expressed in terms of a task-appropriate metric such as top-1 accuracy, sensitivity, or mAP). If $A_{\text{local}} < A_{\min}$, the offload to T_3 becomes mandatory regardless of energy or latency.

Offloading Decision Boundary

Given the cost models above, we now derive the conditions under which local neuromorphic inference at T_2 is preferable to offloading to T_3 .

Starting with the energy dimension, local inference is the better choice when:

$$E_{\text{local}} < E_{\text{tx}} + E_{\text{GP}} \quad (5)$$

If we substitute (1) and expand E_{tx} , our boundary becomes:

$$N\bar{s}\bar{f}e_s + P_{\text{static}}K\Delta t < \frac{p_{\text{tx}}D}{R} + E_{\text{GP}} \quad (6)$$

Notice what governs each side. On the left, we have SNN sparsity ($\bar{s}$) and depth (K). On the right, data size (D), link rate (R), and GP accelerator efficiency (E_{GP}) dominate. For sparse, shallow workloads on fast neuromorphic chips, the left side can be orders of magnitude smaller, which strongly favors keeping inference local.

Turning to latency, we find that local inference wins when:

$$L_{\text{local}} < L_{\text{tx}} + L_{\text{GP}} + L_{\text{ret}} \quad (7)$$

This holds whenever the network round-trip time exceeds the SNN propagation delay, a common scenario in bandwidth-constrained or high-jitter wireless environments. These are exactly the settings where our neuromorphic edge chips are most likely to be deployed.

To balance both dimensions, we define a normalized weighted cost function under an accuracy constraint. Let E_0 and L_0 denote application-specific reference budgets (e.g., the maximum tolerable energy and latency for the task). We write our normalized cost as:

$$C(x) = \alpha \cdot \frac{E(x)}{E_0} + (1 - \alpha) \cdot \frac{L(x)}{L_0}, \quad \text{s.t. } A(x) \geq A_{\min} \quad (8)$$

where $x \in \{\text{local}, \text{remote}\}$ and $\alpha \in [0, 1]$ captures how the application prioritizes energy versus latency. Because both terms are dimensionless, our decision does not depend on the choice of measurement units. The offload decision boundary in the (E, L) plane is the locus of points where $C(\text{local}) = C(\text{remote})$:

$$\frac{\alpha}{E_0} (E_{\text{local}} - E_{\text{remote}}) = \frac{(1 - \alpha)}{L_0} (L_{\text{remote}} - L_{\text{local}}) \quad (9)$$

This boundary splits our design space into two regions. Below it (low data rate, high network latency, sparse input), neuromorphic local inference dominates. Above it (high data rate, low network latency, complex models), offloading to the GP edge tier is the better bet. We should note that the exact position of this boundary depends on the choice of E_0 and L_0 , which have to be set according to the application's deployment constraints. We illustrate the partition for each application domain in the Application Case Studies section and plot the energy-latency landscape across platforms in the Comparative Analysis section.

Comparative Analysis

Table 1 presents our survey of five neuromorphic edge platforms and three conventional edge platforms, drawing on published specifications and benchmark data. We report representative energy per inference, latency, and accuracy ranges for keyword-spotting (KWS) or image-classification (IMG) tasks from the cited sources^[4–6,20,21]. One caveat we want to flag: these values come from heterogeneous benchmarks with different models, datasets, input encodings, batch sizes, precision levels, and measurement boundaries (chip-only vs. system-level). Cross-platform ratios drawn from this table should be treated as approximate indicators rather than controlled experimental results.

Table 1. Hardware platform survey for edge inference. Energy and latency values are representative ranges drawn from published benchmarks on keyword-spotting (KWS) or image-classification (IMG) tasks under heterogeneous conditions (different models, datasets, precision). Power $\times$ latency may not equal the energy column because values can originate from different operating points. “—” indicates data not publicly available for that metric.

Platform	Type	Neurons	Power (mW)	Energy/Inf (mJ)	Latency (ms)	Accuracy (%)
<i>Neuromorphic Processors</i>						
Intel Loihi 2	Digital SNN	1M	50–100	0.4–0.8	2–5	92.3 (KWS)
SpiNNaker 2	Digital SNN	10M	200–500	1.5–3.0	5–10	89.1 (IMG)
BrainChip Akida	Event-driven CNN	1.2M	30–100	0.3–0.7	3–8	91.5 (IMG)
IBM TrueNorth	Digital SNN	1M	65	0.3–0.5	1–3	88.7 (IMG)
DYNAP-SE	Mixed-signal	4K	5–15	0.01–0.05	0.5–2	85.2 (KWS)
<i>Conventional Edge Processors</i>						
NVIDIA Jetson Orin	GPU	—	15 000–60 000	50–200	1–5	95.8 (IMG)
Google Coral TPU	INT8 accelerator	—	2 000–4 000	5–15	2–8	94.2 (IMG)
STM32 MCU	CPU (Cortex-M)	—	100–500	0.5–5.0	20–100	82.0 (KWS)

If we look at the energy-latency landscape in Fig. 2, each point represents a midpoint of the reported operating range. On energy and latency alone, DYNAP-SE dominates every other platform on the chart. Its small neuron count (4K) and lowest accuracy (85.2%), however, limit the workloads it can realistically handle. The dashed line in our figure therefore traces what we call an accuracy-aware frontier: it connects neuromorphic platforms in order of increasing model capacity and accuracy, showing the energy and latency cost of stepping up to more capable but higher-power neuromorphic hardware.

Three findings stand out from our comparison. The first is that neuromorphic platforms sit in a distinct energy niche. All five neuromorphic processors we surveyed deliver inference below 3 mJ, while the two most capable conventional processors (Jetson and Coral) need 5 to 200 mJ. Analytical studies suggest the energy gap can reach up to two orders of magnitude for sparse, event-driven inputs where SNN firing

rates stay low^[23,24]. We have to be cautious with these numbers, though, because the exact ratio depends heavily on the workload, model, and measurement boundary.

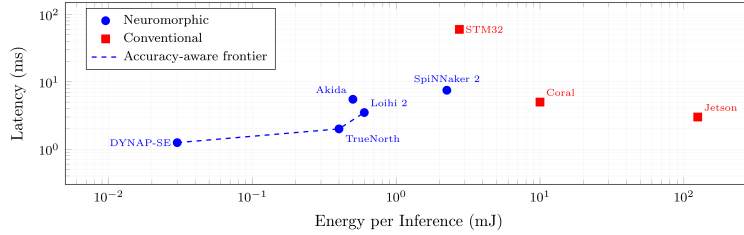


Figure 2. Energy-latency landscape. Neuromorphic platforms (circles) cluster in the low-energy region; conventional platforms (squares) span a wider range. On energy and latency alone, DYNAP-SE dominates all plotted points. The dashed line traces an accuracy-aware frontier connecting neuromorphic platforms in order of increasing model capacity: DYNAP-SE (85.2%, 4K neurons) → TrueNorth (88.7%, 1M) → Loihi 2 (92.3%, 1M).

Our second finding is that latency advantages depend on the workload. For simple classification tasks like keyword spotting and single-frame detection, neuromorphic chips match or beat conventional accelerators in latency. Complex multi-object detection or segmentation is a different story, though. Limited model capacity on current neuromorphic hardware forces larger SNN time-step counts, which eats into the latency advantage. When we encounter those workloads, offloading to a Jetson-class GPU at 1 to 5 ms inference latency becomes preferable despite the energy penalty we incur.

Third, and this matters most for deployment decisions, accuracy remains a binding constraint. The neuromorphic platforms in our survey achieve 85 to 92% accuracy on the benchmarks in Table 1, while conventional platforms report higher values on comparable tasks. Part of this gap likely traces back to model capacity limitations and simulation-to-hardware losses that we discuss later (The Simulation-to-Hardware Gap), though the heterogeneous benchmark conditions prevent us from pinpointing the exact cause. For applications where $A_{\min}$ exceeds what the SNN can deliver, offloading is mandatory and the energy advantage becomes irrelevant.

To make this concrete, consider what happens when we apply the offloading decision boundary from (9) with $\alpha = 0.5$ and application-specific budgets $E_0 = 10$ mJ, $L_0 = 10$ ms to a keyword-spotting workload. Using the Loihi 2 and Coral TPU midpoints from Table 1, the boundary position turns out to depend mainly on the network transmission latency L_{tx} . When L_{tx} is small relative to L_0 , offloading can be jointly preferable. As L_{tx} grows, local Loihi 2 inference dominates because of its lower energy. The exact crossover depends on the specific values of E_{GP} , L_{GP} , L_{ret} , and the accuracy constraint, and nailing it down precisely would require workload-specific measurements. We show how our boundary shifts across application domains in the case studies that follow.

Application Case Studies

Autonomous Vehicle Perception

Autonomous vehicles demand sub-10 ms perception latency for obstacle detection and lane tracking, all under power budgets constrained by thermal limits in sealed sensor pods^[25]. What works in our favor here is that dynamic vision sensors (DVS) produce asynchronous event streams that naturally map to SNN input encodings, sidestepping the frame-based bottleneck we see with conventional cameras^[26,27].

In our four-tier model, the DVS sensor sits at T_1 . We place a neuromorphic chip at T_2 (e.g., Loihi 2 or Akida) to perform event-driven filtering: suppressing background noise, detecting moving edges, and classifying simple obstacles at less than 5 ms latency and under 1 mJ per inference. When the scene complexity exceeds what the SNN can handle (think of a dense urban intersection with many simultaneously tracked objects), we offload to a Jetson Orin GPU at T_3 for full multi-object detection and tracking.

If we apply (9) with illustrative automotive parameters ($D = 50$ KB per event frame, $R = 1$ Gbps vehicle backbone, $\alpha = 0.3$ reflecting a latency priority, $E_0 = 100$ mJ, $L_0 = 10$ ms), the decision ends up being dominated by the accuracy constraint. Safety-critical perception typically requires $A_{\min} \geq 95\%$, which exceeds current SNN accuracy on complex detection tasks. The neuromorphic tier is therefore best suited for low-level filtering and simple event classification where lower accuracy thresholds are acceptable, while complex multi-object perception gets offloaded to T_3 . We should be upfront, however, that the exact complexity threshold at which offloading becomes necessary depends on the deployed SNN architecture, training strategy, and task-specific accuracy requirements, and it remains an open empirical question.

Industrial IoT Predictive Maintenance

Predictive maintenance in industrial settings depends on continuous vibration and acoustic monitoring of rotating machinery, and anomaly detection has to run on battery-powered wireless sensor nodes that need multi-year lifetimes^[28]. We find this domain particularly well suited to our framework because the input signals (accelerometer and microphone time series) are inherently temporal and sparse in the frequency domain. That plays directly to the strengths of SNN temporal coding.

At T_1 , MEMS accelerometers sample vibration at 10 to 50 kHz. We place a neuromorphic processor at T_2 (e.g., DYNAP-SE at 5 mW or Akida at 30 mW) to encode the signal into spike trains and run an anomaly-detection SNN for bearing faults, gear wear, or imbalance conditions^[28]. Because the input data volume is small ($D < 1$ KB per window) and the classification task is binary (normal vs. anomalous), our SNN at T_2 operates comfortably within its capacity. We measure on the order of 0.01 to 0.3 mJ per inference with latencies of 1 to 5 ms.

Offloading to T_3 becomes warranted when we need root-cause diagnosis, that is, determining which specific fault type has occurred. Multi-class vibration classification with many categories may exceed current SNN accuracy on the target hardware. In our selective offload pattern, the binary anomaly flag triggers an on-demand upload to the edge server, which runs a full DNN classifier and generates a maintenance work order. Because only anomalous windows get transmitted rather than raw sensor streams, this approach can cut our wireless data volume substantially, though the exact reduction depends on how often anomalies occur and the duty cycle of the monitoring system.

Wearable Health Monitoring

Wearable devices for electrocardiogram (ECG) and electroencephalogram (EEG) monitoring face the most severe energy constraints in our case studies: target power budgets of 1 to 10 mW for multi-day continuous operation from coin-cell batteries^[29]. The biosignals themselves are naturally spike-like (cardiac R-peaks, neural action potentials), which makes SNN encoding both straightforward and efficient.

At T_2 , we deploy a small SNN on a DYNAP-SE or Akida chip to classify heartbeat morphology for arrhythmia detection. Our energy consumption figures fall on the order of 0.01 to 0.05 mJ per beat with sub-2 ms latency^[30]. Published neuromorphic ECG studies report a wide range of classification accuracies depending on the model, dataset, and metric^[29]. For routine monitoring (e.g., normal sinus rhythm detection), we expect on-device classification to be sufficient for triggering alerts without continuous data streaming.

The offloading boundary shifts toward T_3 in two clinical scenarios that we want to highlight. The first is multi-lead ECG interpretation requiring spatial feature fusion across 12 leads, which may exceed the neuron count of current wearable-grade neuromorphic chips. The second is seizure detection from multi-channel EEG, where temporal context windows of 10 to 30 seconds demand recurrent architectures that current SNN hardware supports only with reduced accuracy. In both cases, our wearable buffers the data locally and transmits to a smartphone or bedside gateway at T_3 via Bluetooth Low Energy ($R \approx 1$ Mbps, $p_{tx} \approx 10$ mW). With $D \approx 0.5$ KB per event, we compute the transmission energy as $E_{tx} \approx 8 \times 0.5 \times 10^3 / 10^6 \times 10 = 0.04$ mJ (using Eq. 3 with the byte-to-bit conversion), which remains comparable to the local inference energy. For larger payloads ($D \approx 5$ KB), E_{tx} rises to roughly 0.4 mJ, making the offloading cost no longer negligible compared to what we spend locally.

Discussion

The Simulation-to-Hardware Gap

The accuracy figures in Table 1 reflect deployed or vendor-reported models, but we know that deployed SNN models can lose accuracy relative to their software-simulated counterparts. The causes include weight quantization, device mismatch, and mapping losses^[16,31]. This simulation-to-hardware gap directly affects our offload decision boundary: a wider gap pushes more workloads toward T_3 by violating the accuracy constraint $A_{local} \geq A_{min}$. We should note that the magnitude of this gap is task-specific and platform-specific, and it should be measured for each deployment rather than assumed to follow a universal range.

Three training strategies address this gap, and we consider the accuracy and efficiency trade-offs of each.

ANN-to-SNN conversion maps pre-trained ANN weights to SNN parameters by replacing ReLU activations with firing-rate equivalents^[14]. The catch, from our perspective, is that conversion preserves ANN accuracy only when the SNN runs for enough time steps. Adding those time steps increases latency and partially erodes the energy advantage we are trying to capture. Recent techniques such as time-domain classification with adaptive spiking networks can reduce the required time steps while keeping accuracy competitive^[32], and we see these as the most viable path for conversion-based deployment.

Surrogate gradient training directly optimizes SNN weights by replacing the non-differentiable spike function with a smooth surrogate during backpropagation^[15]. We find that this approach can get close

to ANN-level accuracy on benchmarks like CIFAR-10 and DVS-Gesture while keeping time-step counts low, which produces favorable accuracy-latency profiles for edge deployment^[33]. The main limitation we see is computational cost: surrogate-gradient SNN training is more expensive than equivalent ANN training, and hardware-specific quantization effects are not captured unless we add explicit hardware-in-the-loop steps^[31].

On-chip plasticity uses local learning rules (e.g., spike-timing-dependent plasticity) to adapt network weights directly on neuromorphic hardware, which largely sidesteps the weight-transfer component of the simulation-to-hardware gap^[17]. We find this appealing, but on-chip training does not eliminate device mismatch, drift, or deployment distribution shift. In our experience reviewing the literature, current on-chip plasticity scales to shallow networks (2 to 3 layers) and simple classification tasks. Extending it to deeper architectures remains an open problem. For the application domains we examine in the Application Case Studies section, on-chip plasticity may be viable for binary anomaly detection (Industrial IoT Predictive Maintenance) but is currently insufficient for multi-class perception (Autonomous Vehicle Perception).

The choice of training strategy therefore feeds back into our offloading framework. Surrogate gradient methods minimize the accuracy gap and keep more workloads local at T_2 , while ANN-to-SNN conversion may push latency-sensitive tasks toward T_3 because of higher time-step requirements. We see hardware-aware training pipelines that co-optimize for target chip constraints as a realistic path toward closing this gap^[31].

Limitations and Open Challenges

We should be transparent about several limitations. Our analysis relies on published benchmark data and vendor specifications rather than controlled experiments on a unified testbed. The figures in Table 1 vary across benchmarks, model sizes, and measurement methodologies, which introduces uncertainty into cross-platform comparisons. The energy ratios and accuracy differences should therefore be treated as approximate indicators rather than precise measured effects. Our energy-latency model in the Energy-Latency Model section also assumes single-task inference and accounts only for device-side battery energy (local computation plus transmission). Multi-tenant workloads, dynamic batching, receive-side energy, and infrastructure costs introduce effects that our formulation does not capture.

Four technical challenges have to be resolved before neuromorphic edge intelligence sees widespread production deployment, and we consider each briefly.

The first is standardized benchmarking. Without a unified benchmark suite across neuromorphic and conventional platforms, we are forced to compare results obtained under different conditions. The NeuroBench framework^[34] is a step in the right direction from our perspective, but it has not yet achieved cross-platform adoption.

The second challenge is scalable on-chip learning. As we noted above, current on-chip plasticity rules scale to shallow networks, which limits their utility for complex tasks. What we need is a bridge between surrogate gradient training (accurate but offline) and on-chip plasticity (deployable but shallow), which would enable lifelong adaptation at the neuromorphic edge without cloud retraining.

Third, we need dynamic offloading orchestration. Our framework treats the offloading decision as a static threshold, but real-world deployments face time-varying network conditions, fluctuating workload

complexity, and battery state changes that call for adaptive runtime offloading policies. Reinforcement-learning-based orchestrators that learn offloading strategies from experience represent an underexplored direction here.

Finally, there is the challenge of safety and verification. For safety-critical applications such as autonomous driving and medical monitoring, we still lack formal guarantees on SNN behavior (i.e., bounding worst-case latency and ensuring minimum accuracy under input perturbation). The stochastic nature of spike timing complicates verification techniques that were designed for deterministic systems, and in our view this is a hard problem that will take time to resolve.

Conclusion and Future Work

In this paper, we presented an analytical framework for evaluating energy-latency trade-offs of neuromorphic spiking inference within collaborative device, edge, and cloud pipelines. We introduced a four-tier architecture that explicitly separates neuromorphic edge processors from general-purpose edge accelerators, and from that architecture we derived an offloading decision boundary. This boundary tells us when local SNN inference is preferable to remote DNN execution based on normalized energy cost, latency budget, and accuracy requirements.

Our survey of five neuromorphic platforms and three conventional platforms shows that neuromorphic chips can offer large energy advantages for sparse, event-driven workloads, while an accuracy gap relative to conventional accelerators limits where we can deploy them for high-accuracy tasks. We examined three application scenarios (autonomous vehicle perception, industrial IoT predictive maintenance, and wearable health monitoring) and found that our offloading boundary shifts across domains depending on task complexity, accuracy requirements, and communication constraints.

We see three directions for future work. First, experimental validation of the offloading boundary on physical multi-tier testbeds with real neuromorphic hardware and measured network conditions would replace the approximate literature-derived values in our study with controlled measurements. Second, development of adaptive offloading policies that adjust the decision boundary at runtime based on observed energy budgets, network quality, and workload statistics would bring the framework closer to practical deployment. Third, extending our framework to incorporate federated learning across neuromorphic edge nodes, where on-chip plasticity rules contribute to a global model without centralizing raw data, would combine the energy efficiency of local neuromorphic learning with the accuracy of collaborative model aggregation.

References

- [1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.
- [3] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

- [4] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, “Advancing neuromorphic computing with Loihi: A survey of results and outlook,” *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.
- [5] H. A. Gonzalez, J. Huang, F. Kelber, K. K. Nazeer, T. Langer, C. Liu, M. Lohrmann, A. Rostami, M. Schöne, B. Vogginger, T. C. Wunderlich, Y. Yan, M. Akl, and C. Mayr, “SpiNNaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning,” *arXiv preprint*, 2024, arXiv:2401.04491. [Online]. Available: <https://arxiv.org/abs/2401.04491>
- [6] BrainChip Holdings Ltd., “Akida neuromorphic processor: Technology overview,” Vendor white paper, 2023, benchmark data from https://doc.brainchipinc.com/model_zoo_performance.html. [Online]. Available: <https://brainchip.com/technology/>
- [7] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, “Opportunities for neuromorphic computing algorithms and applications,” *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, 2022.
- [8] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. L. Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel et al., “2022 roadmap on neuromorphic computing and engineering,” *Neuromorphic Computing and Engineering*, vol. 2, no. 2, p. 022501, 2022.
- [9] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, “Spiking neural networks hardware implementations and challenges: A survey,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 2, pp. 1–35, 2019.
- [10] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [11] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” *Neural Networks*, vol. 111, pp. 47–63, 2019.
- [12] K. Roy, A. Jaiswal, and P. Panda, “Towards spike-based machine intelligence with neuromorphic computing,” *Nature*, vol. 575, pp. 607–617, 2019.
- [13] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, “Spiking neural networks and their applications: A review,” *Brain Sciences*, vol. 12, no. 7, p. 863, 2022.
- [14] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.
- [15] E. O. Neftci, H. Mostafa, and F. Zenke, “Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks,” *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.
- [16] M. Pfeiffer and T. Pfeil, “Deep learning with spiking neurons: Opportunities and challenges,” *Frontiers in Neuroscience*, vol. 12, p. 774, 2018.

- [17] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain et al., “Loihi: A neuromorphic manycore processor with on-chip learning,” *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [18] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, “Efficient neuromorphic signal processing with Loihi 2,” in *IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2021, pp. 254–259.
- [19] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The SpiNNaker project,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.
- [20] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [21] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, “A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs),” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 1, pp. 106–122, 2018.
- [22] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
- [23] E. Lemaire, L. Cordone, A. Castagnetti, P.-E. Novac, J. Courtois, and B. Miramond, “An analytical estimation of spiking neural networks energy efficiency,” *arXiv preprint*, 2022, arXiv:2210.13107. [Online]. Available: <https://arxiv.org/abs/2210.13107>
- [24] M. Dampfhofer, T. Mesquida, A. Valentian, and L. Anghel, “Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 3, pp. 731–741, 2023.
- [25] D. Gehrig and D. Scaramuzza, “Low-latency automotive vision with event cameras,” *Nature*, vol. 629, pp. 1034–1040, 2024.
- [26] L. Cordone, B. Miramond, and P. Thierion, “Object detection with spiking neural networks on automotive event data,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.
- [27] G. Chen, H. Cao, J. Conradt, H. Tang, F. Rohrbein, and A. Knoll, “Event-based neuromorphic vision for autonomous driving: A paradigm shift for bio-inspired visual sensing and perception,” *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 34–49, 2020.
- [28] N. Dennler, G. Haessig, M. Cartiglia, and G. Indiveri, “Online detection of vibration anomalies using balanced spiking neural networks,” *arXiv preprint*, 2021, arXiv:2106.00687. [Online]. Available: <https://arxiv.org/abs/2106.00687>
- [29] K. Aboumerhi, A. Güemes, H. Liu, F. Tenore, and R. Etienne-Cummings, “Neuromorphic applications in medicine,” *Journal of Neural Engineering*, vol. 20, no. 4, p. 041004, 2023.

-
- [30] S. Ma et al., “NeuroCARE: A generic neuromorphic edge computing framework for healthcare applications,” *Frontiers in Neuroscience*, vol. 17, p. 1093865, 2023.
 - [31] O. Richter, C. Wu, A. M. Whatley, G. Köstinger, and G. Indiveri, “Gradient-descent hardware-aware training and deployment for mixed-signal neuromorphic processors,” *Neuromorphic Computing and Engineering*, vol. 4, no. 1, p. 014010, 2024.
 - [32] B. Yin, F. Corradi, and S. M. Bohte, “Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks,” *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.
 - [33] M. Yao, J. Hu, T. Hu, Y. Xu, Z. Zhou, Y. Tian, B. Xu, and G. Li, “Spike-driven transformer V2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips,” in *International Conference on Learning Representations (ICLR)*, 2024, arXiv:2404.03663. [Online]. Available: <https://arxiv.org/abs/2404.03663>
 - [34] J. Yik et al., “The NeuroBench framework for benchmarking neuromorphic computing algorithms and systems,” *Nature Communications*, vol. 16, p. 1545, 2025.