A NEW PERSPECTIVE ON ANN-TO-SNN CONVERSION: HARDWARE-ALIGNED MULTI-THRESHOLD NEURON DYNAMICS FOR NEUROMORPHIC INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

By accumulating spikes over multiple timesteps, rate coding approximates the real-valued activations of an ANN, a principle that underpins numerous ANN-to-SNN conversion methods. However, the converted SNNs typically incur loss in accuracy compared to the original ANN. Existing methods either increase a number of timesteps or introduce more complex neuronal dynamics to reduce this accuracy gap. Yet, both approaches ultimately lead to the same outcome: redundancy either in information or in computation, thereby suffering from efficiency degradation.

To address this issue, we propose **PRIME** (Potential-Residual Integration for Multi-threshold Encoding), a **training-free** multi-threshold concurrent spiking neuron framework redesigned from a hardware data representation perspective: a) We introduce the Residual Membrane Potential Amplification (RMPA) mechanism, which replaces rate coding with Integration Coding by actively amplifying the residual membrane potential. RMPA enables the binary representation of spikes to carry substantially more information, achieving near-lossless conversion with as few as two timesteps. b) Complementarily, we implement concurrent spike firing in multi-threshold neurons, where spikes are decoded directly from the binary representation of floating-point data, eliminating membrane potential–threshold comparisons and replacing multiplications with lightweight bit-shift and accumulation operations that align with hardware computation primitives. At $T = 2$ and $D = 3$, we achieve competitive performance across both vision and language benchmarks with only 40.6% of the FP32 bit-transmission cost (81.3% of FP16) and 10.5% of the FP32 multiplication energy consumption (35.5% of FP16). The PRIME framework establishes a new design principle for efficient ANN-to-SNN conversion, bridging the gap between algorithmic efficiency and hardware constraints.

1 INTRODUCTION

Human brains execute high-level cognitive tasks at remarkably low energy cost Gerstner et al. (2014); Izhikevich (2003), offering a compelling biological blueprint for energy-efficient hardware design. Spiking neural networks (SNNs) emulate spike-based biological communication, inheriting high sparsity and event-driven processing Maass (1997); Davies et al. (2018). While direct training via surrogate gradients suffers from approximation errors and expensive spatio-temporal backpropagation Gygax & Zenke (2024); Zhu et al. (2024), ANN-to-SNN conversion has emerged as the dominant cost-effective paradigm for obtaining high-performance SNNs Cao et al. (2015).

Existing conversion methods rely predominantly on rate coding Rueckauer et al. (2017) and fall into two categories: (1) **extending timesteps**, where fidelity asymptotically improves with more timesteps Fang et al. (2023), and (2) **introducing auxiliary neuronal dynamics**, such as differential encoding or multi-threshold mechanisms that store multiple pre-set thresholds Huang et al. (2025a); Xing et al. (2025). The former incurs *information redundancy*, as a single activation value requires sizable spikes across many timesteps. The latter introduces *computational redundancy* through complex computations imposed by auxiliary neuronal dynamics. Both pathways share a common root: neither exploits the intrinsic alignment between SNN spike representations and the binary data formats native to modern hardware.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We identify two complementary insights that address this gap. The conversion error stems from residual membrane potential — the unencoded input remaining after all timesteps. Conventional approaches passively wait for this residual to dissipate, yielding only marginal per-step gains. Actively amplifying the residual instead enables exponential suppression of conversion error. From a hardware perspective, binary spike sequences share the same underlying representation as stored binary data, suggesting that spike thresholds and amplitudes can be read directly from floating-point bit fields, tightly coupling neuronal dynamics with the native hardware data format.

Guided by these insights, we propose **PRIME** (Potential-Residual Integration for Multi-threshold Encoding), a **training-free** conversion framework. **At the encoding level**, the Residual Membrane Potential Amplification (RMPA) mechanism actively amplifies residual membrane potential per step, enabling near-lossless conversion at $T \leq 2$ without auxiliary compensation. **At the inference level**, spike thresholds and amplitudes are read directly from the sign, exponent, and mantissa bits of floating-point values, eliminating membrane potential accumulation and multi-threshold comparison. Reading D consecutive mantissa bits enables concurrent multi-bit spike emission per step; when $T \leq 2$, inter-step independence allows fully parallel inference with multiplications replaced by shift-and-add operations. PRIME simultaneously resolves the information redundancy of time-step extension and the computational redundancy of threshold storage. Extensive experiments on CV and NLP tasks validate its effectiveness. Our main contributions are summarized as follows:

- **At the encoding level**, we propose the RMPA mechanism and Integration Coding, which actively amplify residual membrane potential to maximize per-step information density, achieving near-lossless conversion at $T \leq 2$ without auxiliary compensation.
- **At the inference level**, we propose a hardware-native spike-driven inference scheme that couples neuronal dynamics with floating-point bit representations, replacing multiplications with shift-and-add operations. When $T \leq 2$, inter-step independence enables fully parallel inference; when $D \leq 3$, each intra-step computation requires at most a single adder.
- Building upon these two designs, we propose **PRIME**, a unified ANN-to-SNN conversion framework that jointly addresses encoding inefficiency and hardware deployment overhead.
- Extensive experiments demonstrate that under $T = 2$, $D = 3$, PRIME achieves near-lossless conversion accuracy on representative CV and NLP benchmarks, with energy consumption at 10.5% of FP32 multiplication (35.5% of FP16) and bit-transmission cost between layers at 40.6% of FP32 (81.3% of FP16), providing a strong foundation for energy-efficient neuromorphic hardware design.

2 Related Work

2.1 Training-Dependent Conversion

Training-dependent methods retrain the source ANN to align activations with spiking neuron dynamics before conversion. Under rate coding, representative works optimize membrane potential initialization Bu et al. (2022), replace ReLU with quantization-aware activations Bu et al. (2023b), or apply dual-phase pipelines for weight and threshold co-calibration Ding et al. (2021); Wang et al. (2022d). For Transformers, SpikeZIP-TF You et al. (2024) fine-tunes pretrained weights with spike-compatible surrogates, and SpikeLLM Xing et al. (2024) extends spiking mechanisms to billion-parameter LLMs via saliency-aware encoding. While effective, these methods require training pipeline access, limiting applicability when the source ANN is fixed or proprietary.

2.2 Training-Free Conversion

In contrast, Training-free approaches convert pretrained ANNs directly without fine-tuning. The dominant paradigm is rate coding Rueckauer et al. (2017); Diehl et al. (2015), which requires at least $T = N$ timesteps to distinguish N activation levels. To mitigate this, prior works calibrate residual and initial membrane potentials Hao et al. (2023b;c), extend neuron outputs to ternary values Wang et al. (2022b), or refine threshold calibration strategies Wang et al. (2025b); Bu et al. (2025). Beyond rate coding, phase coding Kim et al. (2018); Wang et al. (2022c) supports 2^T representable values but requires all timesteps to elapse before decoding, while differential coding (DCGS) Huang et al. (2025b) eliminates temporal decay at the cost of hardware-unfriendly division operations. For

Transformer conversion, ECMT Huang et al. (2024b) employs multi-threshold neurons but retains floating-point multiplication; STA Jiang et al. (2024b) approximates arbitrary nonlinearities but requires hundreds of timesteps; and SpikedAttention Hwang et al. (2024) achieves fully spike-driven conversion via trace-driven matrix multiplication. Despite these advances, all existing methods rely on iterative accumulation and gradual dissipation of residual membrane potential, encoding redundant information across timesteps — a fundamental bottleneck that neither algorithmic nor hardware optimizations have yet resolved.

In contrast, PRIME belongs to the training-free category and requires no fine-tuning or retraining: the conversion is performed in a single forward pass on the pretrained weights, directly parsing the binary representation of floating-point data to generate spikes.

3 METHOD

In this section, we propose the **P**otential-**R**esidual **I**ntegration for **M**ulti-threshold **E**ncoding (**PRIME**) framework, which optimizes the multi-threshold paradigm along two complementary dimensions: a more expressive encoding strategy and an inference-oriented hardware design tailored to it.

At the encoding level, PRIME introduces Integration Coding together with its companion Residual Membrane Potential Amplification (RMPA) mechanism, which progressively doubles the residual membrane potential so that MTNs deterministically recover previously unencoded information, achieving rapid convergence toward the corresponding ANN activation values.

At the inference level, PRIME co-designs spike generation and multiplication around the bit-level structure of FP16 operands: Direct FP16 Binary Parsing maps the sign, exponent, and mantissa fields directly to thresholds and spikes via a leading-one detector, while spike-driven multiplication reduces the mantissa product to trace-aligned shift-add operations, eliminating the usage of multipliers.

3.1 PRELIMINARY

Following the successful practice of many prior works You et al. (2024); Wang et al. (2022b); Huang et al. (2025b; 2024b), in this paper, we adopt ternary-spike-based MTNs with outputs in $\{-1, 0, +1\}$ as the core computational unit of the SNN. Throughout this paper, we use the following notation: $t \in \{1, \ldots, T\}$ denotes the timestep index, where T is the total number of timesteps; i indexes the threshold within an MTN and also the amplification loop in RMPA; and D denotes the intra-timestep spike dimension, i.e., the number of consecutive mantissa bits read per timestep. Each MTN in layer ℓ is characterized by a base threshold θ^ℓ and $2n$ thresholds, comprising n positive and n negative ones. The i-th threshold in layer ℓ, denoted λ_i^ℓ, is defined as $\lambda_i^\ell = \theta^\ell / 2^{i-1}$ for the positive branch ($i = 1, \ldots, n$) and $\lambda_{n+i}^\ell = -\theta^\ell / 2^{i-1}$ for the negative branch ($i = 1, \ldots, n$), forming a symmetric set of geometrically decaying values around zero.

Most ANN-to-SNN methods are built upon rate coding You et al. (2024); Wang et al. (2022b; 2025a). The neuronal dynamics of rate-coding-based MTNs can be described as the following equations:

$$\begin{aligned} \mathbf{m}^\ell[t] &= \mathbf{V}^\ell[t-1] + \mathbf{I}^\ell[t], \qquad & \mathbf{S}_i^\ell[t] &= \mathrm{MTH}_{\theta,n}(\mathbf{m}^\ell[t], i), \\ \mathbf{X}^\ell[t] &= \sum_i \mathbf{S}_i^\ell[t]\, \mathbf{W}^\ell \lambda_i^\ell, & \mathbf{V}^\ell[t] &= \mathbf{m}^\ell[t] - \mathbf{X}^\ell[t], \end{aligned} \tag{1}$$

$$\mathrm{MTH}_{\theta,n}(\mathbf{m}^\ell[t], i) = \begin{cases} 1, & \text{if } i = i^*(t) \text{ and } \mathbf{m}^\ell[t] \notin (\lambda_{2n}, \lambda_n), \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $\mathbf{m}^\ell[t]$, $\mathbf{V}^\ell[t]$, $\mathbf{I}^\ell[t]$, $\mathbf{S}_i^\ell[t]$, $\mathbf{W}^\ell$ and $\mathbf{X}^\ell[t]$ denote the pre-spike membrane potential, post-spike membrane potential, input current, output spike of the i-th threshold, weight, and total output signal of layer ℓ at timestep t, respectively. Here $i^*(t) \triangleq \arg\min_{p \in \{1,\ldots,2n\}} |\mathbf{m}^\ell[t] - \lambda_p|$ denotes the index of the threshold closest to the current membrane potential, so that the MTN fires a spike on the i-th channel only when i matches this nearest threshold and $\mathbf{m}^\ell[t]$ lies outside the dead zone $(\lambda_{2n}, \lambda_n)$. Under rate coding, the quantization error between the ANN activation and the SNN output arises from the residual membrane potential that remains below the firing threshold. Existing methods mitigate this error by increasing the number of timesteps or thresholds, both of which incur additional

memory, computation, and energy overhead, thereby impeding edge deployment. To address this, we propose integration coding-based MTNs, which directly exploit the residual membrane potential without requiring extensive time-step iterations or a large number of thresholds.

3.2 Multi-Threshold Neuron Based on Integration Coding

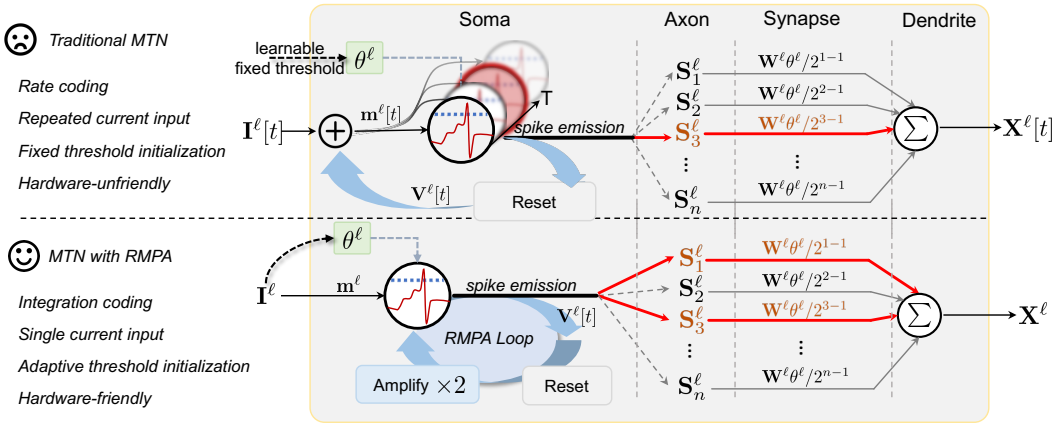


Figure 1: Comparison between the traditional MTN and the proposed MTN with RMPA. **Upper**: For the traditional MTN, the input current is injected at each timestep, necessitating repeated accumulation operations of the input. The neuron maintains $2n$ pre-set thresholds and compares its membrane potential against all of them to determine which spike to fire. This results in redundant storage and computational overhead for threshold comparison per time step. **Lower**: For the proposed MTN with RMPA, the input current is injected only once at the beginning, requiring no additional accumulation operation. Instead of comparing against multiple thresholds, the neuron adaptively sets the only threshold $\theta^\ell = 2^p$ based on the input strength (Eq. 3). At each timestep, a spike is fired if the residual potential after doubling exceeds θ^ℓ. Thus, only one threshold comparison is required, greatly reducing the computational overhead and the storage overhead from $\mathcal{O}(n)$ to $\mathcal{O}(1)$.

The dynamics of the integration coding-based MTN are illustrated in Figure 1. Conventional rate coding-based neurons accumulate input current at every timestep, whereas the integration coding-based MTN receives input only once, so the input current $\mathbf{I}^\ell$ directly initializes the membrane potential $m^\ell = \mathbf{I}^\ell$. Afterwards, the neuron identifies the largest threshold that triggers spike emission, aligned with the binary structure of floating-point numbers (the detailed description is in Subsection 3.3.1). At the beginning, the base threshold θ^ℓ is set and followed by a soft reset according to Eq. equation 3:

$$\theta^\ell = 2^p \text{ where } p = \max\{p \mid 2^p \le \mathbf{m}^\ell,\ p \in \mathbb{Z}\}, \quad \mathbf{V}^\ell = \mathbf{m}^\ell - \frac{\mathbf{S}_i^\ell \theta^\ell}{2^{i-1}}, \tag{3}$$

where i indicates the loop index of RMPA. In each subsequent iteration, $\mathbf{m}^\ell$ is set to the post-reset membrane potential $\mathbf{V}^\ell$.

The RMPA-based MTN iteratively generates sub-threshold spikes from the residual membrane potential, progressively reducing the quantization error. Each spike generation requires only a bit-shift and a single comparison against the base threshold θ^ℓ, eliminating storing a large number of thresholds. Formally, the firing rule is defined as Eq. equation 4:

$$\mathbf{S}_i^\ell = \mathrm{MTH}_{\theta,n}(\mathbf{m}^\ell, i) = \begin{cases} 1, & \text{if } \mathbf{m}^\ell \cdot 2^{i-1} - \theta^\ell \ge 0 \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

A soft reset is then applied as defined in Eq. equation 3. Once the membrane potential reaches zero or the prescribed number of iterations is exhausted, the neuron decodes the emitted spike sequence to produce the total output signal of layer ℓ:

$$\mathbf{X}^\ell = \sum_i \mathbf{S}_i^\ell \mathbf{W}^\ell \frac{\theta^\ell}{2^{i-1}}. \tag{5}$$

To further illustrate the dynamics of RMPA, a concrete example is provided in Appendix B.

Our proposed neurons offer two key advantages: (i) threshold storage is reduced from $2n$ to 1, with a corresponding reduction in I/O operations; (ii) unlike rate coding, integration coding-based MTNs actively exploit the residual membrane potential to suppress quantization error, which is the fundamental source of the accuracy gain.

Proposition 1 (Error Bound of RMPA-based MTN)**.** *The encoding error of rate coding is bounded by $\varepsilon_{rate} < \theta/2T$, which decays linearly with T. In contrast, the proposed PRIME encoding attains a bound $\varepsilon_{PRIME} < \theta/2^{T\cdot D+1}$, which decays exponentially with T, where θ denotes the spike threshold, T is the number of inter-timesteps, and D is the number of concurrent intra-timestep spikes.*

The full proof is given in Appendix A. Accordingly, the error bound of our proposed MTN is significantly tighter than that of conventional MTN: for instance with $T = 2$ and $D = 3$, the error bound of PRIME is $\theta/2^7 = \theta/128$, which is $32\times$ tighter than the rate coding bound of $\theta/4$.

3.3 Hardware-efficient Inference

3.3.1 Spike Generation via Direct FP16 Binary Parsing

Here we describe how the spike generation process is bound to the binary representation of FP16 data, eliminating conventional membrane potential computation. The key is that each field of the FP16 format: sign, exponent, and mantissa can be mapped to the threshold and the spike corresponding to Eq. equation 3 and equation 4, enabling a purely read-based spike emission mechanism.

A normalized FP16 floating-point number is represented as:

$$x_{\mathbf{10}} = (-1)^S \times 2^{E-15} \times \left(1 + \sum_{j=1}^{10} M_j \times 2^{-j}\right), \tag{6}$$

where S is the 1-bit sign, E is the 5-bit exponent ($1 \le E \le 30$), and $M_1 M_2 \cdots M_{10}$ is the 10-bit mantissa with an implicit leading 1. In the first cycle, the implicit leading 1 of the FP16 representation is directly emitted as the first spike, and the sign bit S together with the exponent E determines the corresponding threshold $\theta_1 = (-1)^S \times 2^{E-15}$. Starting from the second cycle, the position of the first 1 in the remaining mantissa is located using a leading-one detector (LOD), a well-established combinatorial logic module in digital circuit design. If the first 1 is found at position k (counting from the most significant bit, $k = 1, 2, \ldots, 10$), the RMPA loop runs k times and the corresponding threshold for the emitted spike is $\theta_k = (-1)^S \times 2^{E-15-k}$. That is, position k directly encodes the scaling factor 2^{-k} relative to the initial threshold, without any explicit computation. After the spike is emitted, the bit at position k is cleared, and the LOD is applied to the updated mantissa in the next cycle to find the next 1. This process repeats until all cycles are exhausted or the mantissa becomes zero. Unlike the traditional ANN-to-SNN approaches that require the full subtract-compare-conditional updates, this method simplifies each cycle to LOD lookups followed by a bit-clear operation, significantly reducing computational cost.

We present an example of parsing the FP16 data of 1.4 in Figure 2. We refer to D as the intra-timestep spike dimension, which controls how many bits are emitted within a single timestep, and T as the number of inter-timesteps, which controls how many successive timesteps are used to encode a given value. In the basic case ($D = 1$), each timestep emits a 1-bit spike indicating whether the threshold is met. This paradigm further supports multi-bit spike emission by reading D consecutive bits at a time, rather than a single bit. When $D > 1$, each spike is a D-bit unsigned integer ranging from 0 to $2^D - 1$, representing 0 to $2^D - 1$ times the corresponding threshold. For example, when $D = 2$, 2 bits are read per timestep, encoding spike values in $\{0, 1, 2, 3\}$; when $D = 3$, 3 bits are read, encoding values in $\{0, 1, \ldots, 7\}$. In the last timestep, an additional bit (i.e., the (D+1)-th bit) is read beyond the D-bit spike. This extra bit is then added to the D-bit spike value, treated as an unsigned integer, to perform a carry operation, which is equivalent to a round-half-up operation, and further improves encoding accuracy without introducing a separate rounding stage.

At the hardware level, this intra-timestep expansion only requires widening the read window from 1 bit to D bits; the LOD-based position finding and threshold determination remain unchanged. This multi-bit emission significantly increases the information density per timestep and is difficult to

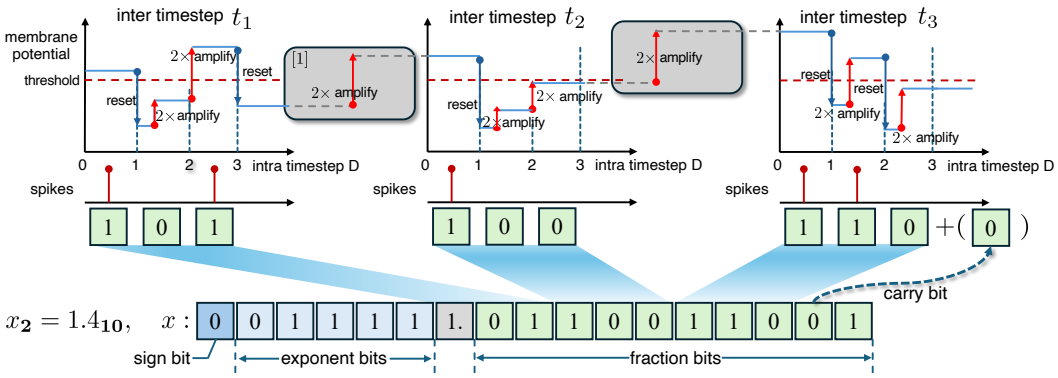


Figure 2: Parsing the FP16 representation of 1.4 into spikes with D=T=3. The first inter-timestep emits the implicit 1 concatenated with the leading 2 mantissa bits at threshold $\theta = (-1)^S \times 2^{E-15-(D-1)}$; the second uses the LOD to locate the first 1 in the remaining 8 bits, reads D bits from that position as the spike value, and shifts the threshold accordingly. [1] The number of amplifications is not strictly one. It equals the bit-offset of the next leading 1 from the end of the current T-bit group.

achieve in traditional ANN-to-SNN conversion algorithms, where each timestep is inherently limited to a binary fire-or-not decision.

A key advantage of this binary-parsing approach is that when $T \leq 2$, the two timesteps are data-independent: the first reads the implicit 1 and leading mantissa bits, while the second operates on the remaining bits through a LOD. Since the second timestep depends only on a predetermined bit range, both can be executed in parallel, effectively halving the latency.

3.3.2 Efficient Multiplication Under Spike Traces

The spike generation mechanism described above decomposes an FP16 value into a sequence of weighted spikes over T inter-timesteps. We now show how this decomposition naturally induces a hardware-efficient multiplication scheme: instead of performing a full 11×11-bit mantissa multiplication, the product $x \times w$ is computed incrementally through spike-driven conditional shift-add operations across timesteps.

Sign and Exponent. Given two FP16 operands x and w, the sign and exponent of the product are computed in the standard manner. The output sign is obtained via a single XOR gate: $S_o = S_x \oplus S_w$, and the output exponent is $E_o = E_x + E_w - \text{bias}$, where $\text{bias} = 01111_2 = 15_{10}$. This requires only a 5-bit adder and a 6-bit subtractor, identical to a conventional FP16 multiplier. The key innovation lies entirely in how the mantissa product is computed, as described below.

Mantissa Decomposition via Spike Traces. Let $F_x = 1.M_{x,1}M_{x,2}\cdots M_{x,10}$ denote the full mantissa of x (including the implicit leading 1), and similarly F_w for w. The spike generation process of the previous subsection decomposes F_x into T spike values $\{s_t\}_{t=1}^{T}$, each being a D-bit unsigned integer, with an associated trace value Tr_t that records the bit-position offset of the t-th spike relative to the binary point. Specifically, the first spike s_1 is composed of the implicit 1 together with the first $D-1$ mantissa bits, and the corresponding trace is $\text{Tr}_1 = 1$. For subsequent timesteps $t \geq 2$, the LOD identifies the starting position of the next group, and D consecutive bits are read as s_t. The trace value satisfies: $\text{Tr}_t = 1 + (t-1) \times D$, when the mantissa bits are densely packed (i.e., no intervening zeros between groups). Under this decomposition, the mantissa can be reconstructed as $F_x = \sum_{t=1}^{T} s_t \times 2^{-(\text{Tr}_t - 1)}$. Substituting into the mantissa product yields:

$$F_o = F_x \times F_w = \sum_{t=1}^{T} \underbrace{(s_t \times F_w)}_{\triangleq\, F_o|_t} \times 2^{-(\text{Tr}_t - 1)}. \tag{7}$$

This factorization reveals that the 11×11-bit mantissa multiplication is decomposed into T smaller multiplications of a D-bit spike s_t by the 11-bit mantissa F_w, followed by trace-aligned accumulation. Each per-timestep partial product $F_o|_t = s_t \times F_w$ is a D-bit $\times$ 11-bit multiplication, which can

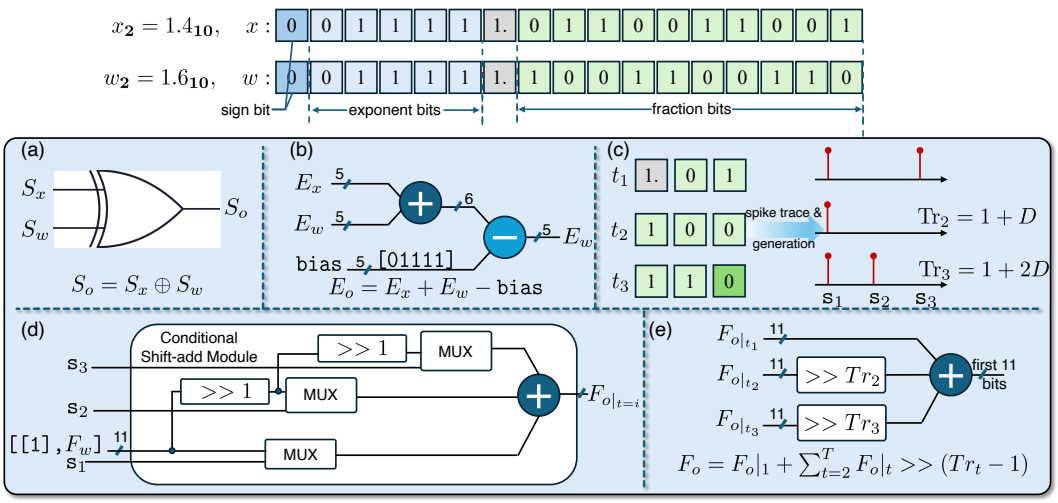


Figure 3: Spike-driven FP16 multiplication. Sign, exponent, and mantissa of x and w are processed in parallel: **(a)** $S_o = S_x \oplus S_w$ via XOR; **(b)** $E_o = E_x + E_w - \text{bias}$ via a 5-bit adder and 6-bit subtractor; **(c)** the mantissa F_x is decomposed into T spikes $\{s_t\}$ with traces $\text{Tr}_t = 1 + (t-1)D$; **(d)** each partial product $F_o|_t = s_t \times F_w$ is computed by a Conditional Shift-add Module of MUXes and an adder tree, replacing the 11×11 multiplier array; **(e)** partial products are trace-aligned via right shifts and summed to form F_o, truncated to 11 bits.

be efficiently realized by a Conditional Shift-add Module as illustrated in Figure 2(d). Let $s_t = \sum_{j=1}^{D} b_j \times 2^{D-j}$, where $b_j \in \{0, 1\}$ are the individual bits of the D-bit spike. Then:

$$F_o|_t = s_t \times F_w = \sum_{j=1}^{D} b_j \times (F_w \gg (j-1)), \tag{8}$$

where $\gg$ denotes logical right shift. Each term $b_j \times (F_w \gg (j-1))$ is implemented by a multiplexer (MUX) controlled by b_j: when $b_j = 1$, the shifted value $F_w \gg (j-1)$ is selected; when $b_j = 0$, zero is selected. The D selected values are then summed by an adder tree. This conditional shift-add structure replaces the combinatorial multiplier array with D MUXes and a $(D-1)$-stage adder tree, significantly reducing the hardware area. Note that when spike $= 111$, the multiplication can be computed more efficiently by rewriting it as $1000 - 0001$, requiring only a single subtraction. A detailed derivation is provided in Appendix D.

Trace-aligned Temporal Accumulation. As shown in Figure 3(e), the final mantissa is the trace-aligned sum of the T partial products $F_o|_t$:

$$F_o = F_o|_1 + \sum_{t=2}^{T} F_o|_t \gg (\text{Tr}_t - 1). \tag{9}$$

The right shift $\gg (\text{Tr}_t - 1)$ aligns the t-th partial product to the correct bit position relative to the first partial product, and the final sum is truncated to the first 11 bits to produce the output mantissa. This temporal accumulation directly mirrors the dynamics of synaptic integration in SNNs: each incoming spike s_t triggers a weighted response $F_o|_t$, and the trace Tr_t governs the temporal decay (realized here as a right shift) before the responses are accumulated at the output neuron.

The Spike Generation via Direct FP Binary Parsing, together with the Spike-driven Multiplication, yields significant energy and data-transmission savings. The detailed energy analysis is given in Appendix C.

4 Experiments

In this section, we evaluate the proposed PRIME framework on both computer vision (CV) and natural language processing (NLP) tasks. All the experiments are executed on a single NVIDIA A100 GPU with PyTorch 2.10.0 and CUDA 12.8.

Table 1: Performance comparison of different methods on the ImageNet-1K dataset. "Bits" denotes the total number of bits transmitted across all timesteps (i.e., the cumulative bit-width of the spike over T steps), reflecting the inter-layer data transmission cost during inference.

Method	Arch.	Param.(M)	ANN Acc.(%)	T	Bits	SNN Acc.(%)
TSDeng & Gu (2021)	VGG-16	138.0	72.40	64	64	70.97 (-1.43)
SNMWang et al. (2022a)	VGG-16	138.0	73.18	64	128	71.50 (-1.68)
QCFSBu et al. (2023a)	ResNet-34	21.8	74.32	64	64	72.35 (-1.97)
	VGG-16	138.0	74.29	64	64	72.85 (-1.44)
SRPHao et al. (2023a)	ResNet-34	21.8	74.32	64	64	68.61 (-5.71)
	VGG-16	138.0	74.29	64	64	69.43 (-4.86)
MSTWang et al. (2023)	Swin-T	28.5	80.51	512	512	78.51 (-2.00)
STAJiang et al. (2024a)	ViT-B/32	86.0	83.60	256	256	82.79 (-0.81)
SpikeZIP-TFkang you et al. (2024)	SViT-S-32	22.1	81.59	64	128	81.45 (-0.14)
	SViT-B-32	86.6	82.83	64	128	82.71 (-0.12)
	SViT-L-32	304.3	83.86	64	128	83.82 (-0.04)
ECMTHuang et al. (2024a)	ViT-S/16	22.0	78.04	10	40	77.07 (-0.97)
	EVA-G	1074.0	89.62	8	32	89.40 (-0.22)
Adaptive fissionJiang et al. (2025) + QCFSBu et al. (2023a)	ResNet-34	21.8	74.32	32	64	69.37 (-4.95)
DCGSHuang et al. (2025a)	VGG-4/1	138.0	73.25	8	24	73.17 (-0.08)
	ResNet34-4/1	21.8	76.42	8	24	76.04 (-0.38)
	ViT-L-8/4	304.3	85.84	4	16	85.45 (-0.39)
	EVA02-L-8/4	305.1	90.05	4	16	89.72 (-0.33)
PRIME (ours)	VGG-16	138.0	71.12	**2** (D=3)	**13**	**71.08 (-0.04)**
	ResNet-34	21.8	76.09	**2** (D=3)	**13**	**76.13 (-0.00)**
	ViT-S	22.1	78.04	**2** (D=3)	**13**	**78.06 (-0.00)**
	ViT-B	86.6	82.91	**2** (D=3)	**13**	**82.87 (-0.04)**
	ViT-L	304.3	84.88	**2** (D=3)	**13**	**84.89 (-0.00)**
	EVA02-T	5.8	80.62	**2** (D=3)	**13**	**80.63 (-0.00)**
	EVA02-S	22.1	85.73	**2** (D=3)	**13**	**85.70 (-0.03)**
	EVA02-B	87.1	88.26	**2** (D=3)	**13**	**88.27 (-0.00)**
	EVA02-L	305.1	89.63	**2** (D=3)	**13**	**89.63 (-0.00)**
	EVA02-G	1013.0	89.62	**2** (D=3)	**13**	**89.58 (-0.04)**

4.1 Evaluation on Computer Vision Tasks

For CV tasks, we validate the effectiveness of PRIME on the classic CNN and Transformer architectures, including VGG, ResNet, ViT, and EVA, evaluated on the ImageNet-1K Deng et al. (2009) and CIFAR Krizhevsky (2009) (provided in Appendix H, Figure 5) datasets. We follow the experimental settings of Huang et al. (2024a).

The experimental results on ImageNet-1K are presented in Table 1. The results show that PRIME consistently achieves lower conversion error across both CNN and Transformer architectures. Compared with the classic baseline DCGS (Huang et al. (2025a)), PRIME uses only two timesteps to achieve lower accuracy degradation on VGG and ResNet (0.04% vs. 0.08%, 0.00% vs. 0.38%). It demonstrates more pronounced advantages on ViT and EVA architectures (0.00% vs. 0.39%, 0.00% vs. 0.33%). In summary, PRIME is the closest to lossless ANN-to-SNN conversion among the existing frameworks in Table 1, while further reducing the inter-layer bit-width during hardware deployment (13 bits vs. 16 bits). (For additional analysis, the full results across all (T, D) configurations are provided in Appendix G (Table 9).)

4.2 Evaluation on Natural Language Tasks

For NLP tasks, we conduct the evaluations on Qwen, Llama, and OPT series models Yang et al. (2025); Touvron et al. (2023); Zhang et al. (2022). The evaluations perform on the primary NLP benchmarks, including MMLU, PIQA, ARC-C, ARC-E, Winogrande, and Hellaswag Hendrycks et al. (2020); Bisk et al. (2020); Tandon et al. (2019); Clark et al. (2018). The evaluation process is implemented based on the work of Gao et al. (2024). We evaluate our method on the common sense reasoning tasks by converting LLAMA2-7b and LLAMA2-13b models with ultra-low latency ($T = 2$). Following common practices in SNN conversion, we apply the same architectural configurations for our method. The results are presented in Table 2 and indicate that our method maintains an accuracy

loss of less than 0.3% on the aforementioned NLP benchmarks. To the best of our knowledge, this is the first ANN-to-SNN conversion framework that consistently attains <0.3% accuracy degradation on common-sense reasoning benchmarks, with only 2 timesteps. More results under different settings and datasets are shown in Appendix I (Figure 6), Appendix J (Table 10), and Appendix K (Table 11). Notably, for other bit precision, our proposed method can also be applied effectively. The conversion results for FP8- and INT4-quantized LLMs are provided in Appendix L.

Table 2: Performance Comparison on Common Sense Reasoning Tasks (%). "Bits" denotes the total bits transmitted across all timesteps, reflecting the data transmission cost during inference.

Method	Model	Spike	T	Bits	PIQA	ARC-c_n	ARC-e_n	Winog.	HellS._n
LLAMA	LLAMA2-7b	×	N/A	16	78.45	45.73	74.87	69.38	76.28
	LLAMA2-13b	×	N/A	16	80.52	48.89	76.56	72.22	79.71
SpikeLLMXing et al. (2025)	LLAMA2-7b	✓	2	32	65.67	28.41	41.88	52.80	49.87
	LLAMA2-13b	✓	2	32	67.68	30.38	44.53	57.22	58.02
FASChen et al. (2025a)	OPT-7B	✓	16	32	73.23	—	64.73	60.38	—
LASChen et al. (2025b)	OPT-7B	✓	16	64	76.22	—	65.95	65.75	—
	OPT-13B	✓	16	64	76.28	—	67.38	65.51	—
	OPT-66B	✓	16	64	78.67	—	71.25	68.27	—
PRIME (ours)	LLAMA2-7b	✓	**2** (D=3)	**13**	**78.67**	**45.65**	**74.75**	**69.06**	**76.18**
	LLAMA2-13b	✓	**2** (D=3)	**13**	**80.58**	**49.15**	**76.47**	**72.14**	**79.69**

4.3 Ablation Study

By default, we adopt the full FP16 exponent bit-width (5 bits) to set the threshold range, which preserves conversion accuracy. To evaluate the trade-off between efficiency and accuracy, we reduce the exponent bit-width to 4 and report the results in Table 3. Across all (T, D) configurations, the accuracy drop is marginal, while the number of active thresholds and the inter-layer bit-transmission cost is substantially reduced. This confirms that the proposed threshold-range optimization offers a practical knob for resource-constrained deployment with negligible performance sacrifice. More ablation results on the query spiking strategy in Transformer attention are provided in Appendix F.

Table 3: Ablation study on lightweight threshold-range optimization via exponent-bit width reduction (`width=4`) on ImageNet-1K, evaluated across different (T, D) settings. `Default` refers to the full fp16 exponent-bit width , with spike conversion applied to k and v only in Transformer blocks.

Model	ANN	D	Default			Exponent-bit width = 4		
			$T=1$	$T=2$	$T=3$	$T=1$	$T=2$	$T=3$
ViT-S (patch16_224)	78.04	1	48.20	76.27	77.80	55.26	76.29	77.47
		2	73.78	77.94	78.03	73.94	77.66	77.65
		3	77.01	78.06	78.03	76.92	77.66	77.65
ViT-B (patch16_224)	82.91	1	74.00	82.21	82.74	75.80	81.30	81.77
		2	80.89	82.77	82.89	80.30	81.74	81.88
		3	82.41	82.87	82.91	81.38	81.87	81.87
ViT-L (patch16_224)	84.88	1	80.89	84.46	84.79	81.14	83.81	84.31
		2	84.03	84.80	84.89	83.25	84.31	84.32
		3	84.45	84.89	84.88	84.01	84.30	84.30

5 Conclusion

In this paper, we propose PRIME, a unified **training-free** ANN-to-SNN conversion framework that tackles both information redundancy in time-step extension and computational redundancy in auxiliary neuronal dynamics. At the encoding level, the proposed MTNs with RMPA tighten the error bound from $\theta/2T$ of rate coding to $\theta/2^{T\cdot D+1}$, enabling near-lossless conversion with $T \leq 2$; at the inference level, thresholds and values are read directly from FP binary representation, eliminating membrane potential accumulation and multi-threshold comparison. These results establish a hardware-aligned design principle for efficient ANN-to-SNN conversion.

AI USE DISCLOSURE

Required disclosure: We did not use generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proving mathematical claims, proposing or refining hypotheses, designing research methodology or experiments, data cleaning and reformatting, qualitative and thematic data analysis, or interpreting results.

Recommended disclosure: We used `Claude Code` (Anthropic, 2025) to assist with the following tasks:

- **Editing the paper for readability:** Claude Code assisted in editing portions of the manuscript to improve clarity and language. All AI-edited text was reviewed and approved by the authors.
- **Formatting tables:** Claude Code assisted in formatting LaTeX tables.

We have reviewed all AI-assisted work and take full responsibility for the final content of this work, including text, claims produced with the aid of generative AI.

ETHICS STATEMENT

This work does not raise ethical concerns that require special disclosure.

REPROCIBILITY STATEMENT

This work is fully reproducible. All experiments use publicly available datasets and benchmarks. The complete source code will be made publicly available upon acceptance of this work.

REFERENCES

P. Marikkannu B. Annapoorani. Performance measurement of energy efficient and highly scalable hybrid adder. Computer Systems Science and Engineering, 45(3):2659–2672, 2023. doi: 10.32604/csse.2023.025075. URL http://www.techscience.com/csse/v45n3/50715.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. Proceedings of the AAAI Conference on Artificial Intelligence, 36(1):11–20, Jun. 2022. doi: 10.1609/aaai.v36i1.19874. URL https://ojs.aaai.org/index.php/AAAI/article/view/19874.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. arXiv preprint arXiv:2303.04347, 2023a.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. arXiv preprint arXiv:2303.04347, 2023b.

Tong Bu, Maohua Li, and Zhaofei Yu. Inference-scale complexity in ann-snn conversion for high-performance and low-power applications. In Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 24387–24397, 2025.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. International Journal of Computer Vision, 113(1):54–66, 2015.

Long Chen, Xiaotian Song, Andy Song, BaDong Chen, Jiancheng Lv, and Yanan Sun. Fas: Fast ann-snn conversion for spiking large language models, 2025a. URL https://arxiv.org/abs/2502.04405.

Long Chen, Xiaotian Song, and Yanan Sun. Las: Loss-less ann-snn conversion for fully spike-driven large language models, 2025b. URL https://arxiv.org/abs/2505.09659.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. arXiv:1803.05457v1, 2018.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In International Conference on Learning Representations, 2021.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In 2015 International joint conference on neural networks (IJCNN), pp. 1–8. ieee, 2015.

Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. arXiv preprint arXiv:2105.11654, 2021.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. Science Advances, 9(40):eadi1480, 2023. doi: 10.1126/sciadv.adi1480. URL https://www.science.org/doi/abs/10.1126/sciadv.adi1480.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press, 2014.

Julia Gygax and Friedemann Zenke. Elucidating the theoretical underpinnings of surrogate gradient learning in spiking neural networks, 2024.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. Proceedings of the AAAI Conference on Artificial Intelligence, 37(1):11–21, Jun. 2023a. doi: 10.1609/aaai.v37i1.25071.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential, 2023b. URL https://arxiv.org/abs/2302.02091.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between ANNs and SNNs by calibrating offset spikes. In The Eleventh International Conference on Learning Representations, 2023c.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. CoRR, abs/2009.03300, 2020. URL https://arxiv.org/abs/2009.03300.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp. 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

Zihan Huang, Xinyu Shi, Zecheng Hao, Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Towards high-performance spiking transformers from ANN to SNN conversion. In Jianfei Cai, Mohan S. Kankanhalli, Balakrishnan Prabhakaran, Susanne Boll, Ramanathan Subramanian, Liang Zheng, Vivek K. Singh, Pablo César, Lexing Xie, and Dong Xu (eds.), Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024, pp. 10688–10697. ACM, 2024a. doi: 10.1145/3664647.3680620.

Zihan Huang, Xinyu Shi, Zecheng Hao, Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Towards high-performance spiking transformers from ann to snn conversion. In Proceedings of the 32nd ACM international conference on multimedia, pp. 10688–10697, 2024b.

Zihan Huang, Wei Fang, Tong Bu, Peng Xue, Zecheng Hao, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. Differential coding for training-free ann-to-snn conversion, 2025a. URL https:%arxiv.org/abs/2503.00301.

Zihan Huang, Wei Fang, Tong Bu, Peng Xue, Zecheng Hao, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. Differential coding for training-free ann-to-snn conversion. arXiv preprint arXiv:2503.00301, 2025b.

Sangwoo Hwang, Seunghyun Lee, Dahoon Park, Donghun Lee, and Jaeha Kung. Spikedattention: Training-free and fully spike-driven transformer-to-snn conversion with winner-oriented spike shift for softmax operation. Advances in Neural Information Processing Systems, 37:67422–67445, 2024.

Eugene M Izhikevich. Simple model of spiking neurons. IEEE Transactions on neural networks, 14 (6):1569–1572, 2003.

Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. Spatio-temporal approximation: A training-free SNN conversion for transformers. In The Twelfth International Conference on Learning Representations, 2024a.

Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. Spatio-temporal approximation: A training-free snn conversion for transformers. In The twelfth international conference on learning representations, 2024b.

Yizhou Jiang, Feng Chen, Yihan Li, Yuqian Liu, Haichuan Gao, Tianren Zhang, and Ying Fang. Adaptive fission: Post-training encoding for low-latency spike neural networks. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=2zZzdAMyYi.

kang you, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. SpikeZIP-TF: Conversion is all you need for transformer-based SNN. In Forty-first International Conference on Machine Learning, 2024.

Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. Neurocomputing, 311:373–386, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Zecheng Liu, Wu Luo, Baoyuan Wu, Xi Liu, Jiwen Lim, et al. Bi-real net: Binarizing deep network towards real-network performance. volume 128, pp. 202–219. Springer, 2020.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. Neural networks, 10(9):1659–1671, 1997.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Binary neural networks: A survey. Pattern Recognition, 105:107281, 2020.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Frontiers in Neuroscience, 11, 2017. ISSN 1662-453X. doi: 10.3389/fnins.2017.00682.

Niket Tandon, Bhavana Dalvi Mishra, Keisuke Sakaguchi, Antoine Bosselut, and Peter Clark. Wiqa: A dataset for "what if ..." reasoning over procedural text. arXiv:1909.04739v1, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Jingya Wang, Xin Deng, Wenjie Wei, Dehao Zhang, Shuai Wang, Qian Sun, Jieyuan Zhang, Hanwen Liu, Ning Xie, and Malu Zhang. Training-free ann-to-snn conversion for high-performance spiking transformer, 2025a. URL https://arxiv.org/abs/2508.07710.

Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In Proceedings of the International Joint Conference on Artificial Intelligence, pp. 2501–2508, 2022a.

Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In IJCAI, pp. 2501–2508, 2022b.

Zhehui Wang, Xiaozhe Gu, Rick Siow Mong Goh, Joey Tianyi Zhou, and Tao Luo. Efficient spiking neural networks with radix encoding. IEEE Transactions on Neural Networks and Learning Systems, 35(3):3689–3701, 2022c.

Ziming Wang, Shuang Lian, Yuhao Zhang, Xiaoxin Cui, Rui Yan, and Huajin Tang. Towards lossless ann-snn conversion under ultra-low latency with dual-phase optimization. arXiv preprint arXiv:2205.07473, 2022d.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. Masked spiking transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1761–1771, 2023.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Hongwei Ren, and Renjing Xu. Adaptive calibration: A unified conversion framework of spiking neural networks. Proceedings of the AAAI Conference on Artificial Intelligence, 39(2):1583–1591, Apr. 2025b. doi: 10.1609/aaai.v39i2.32150. URL https://ojs.aaai.org/index.php/AAAI/article/view/32150.

Xingrun Xing, Boyan Gao, Zheng Zhang, David A Clifton, Shitao Xiao, Li Du, Guoqi Li, and Jiajun Zhang. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. arXiv preprint arXiv:2407.04752, 2024.

Xingrun Xing, Boyan Gao, Zheng Zhang, David A. Clifton, Shitao Xiao, Li Du, Guoqi Li, and Jiajun Zhang. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking, 2025. URL https://arxiv.org/abs/2407.04752.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui

Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Kang You, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. Spikezip-tf: Conversion is all you need for transformer-based snn. arXiv preprint arXiv:2406.03470, 2024.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068, 2022.

Yaoyu Zhu, Jianhao Ding, Tiejun Huang, Xiaodong Xie, and Zhaofei Yu. Online stabilization of spiking neural networks. In The Twelfth International Conference on Learning Representations, 2024.

A PROOF OF PROPOSITION 1

Proof. The conversion error in ANN-to-SNN algorithms arises primarily from residual membrane potential below the threshold that fails to generate corresponding spikes.

CONVENTIONAL RATE CODING

In conventional ANN-to-SNN conversion, the neuron accumulates the input signal $\mathbf{I}$ at each timestep:

$$\mathbf{m} \leftarrow \mathbf{m} + \mathbf{I}.$$

The membrane potential is then compared against the threshold θ. If $\mathbf{m} > \theta$, a spike is emitted and the membrane potential is updated as:

$$\mathbf{m} \leftarrow \mathbf{m} - \theta.$$

After T accumulation steps:

$$T \cdot \mathbf{I} = \theta \cdot N_{\text{spike}} + \mathbf{m}_{\text{rest}},$$

where N_{spike} denotes the total number of emitted spikes. The ideal encoded value is $\mathbf{I}$, whereas the actual decoded value is:

$$\frac{\theta \cdot N_{\text{spike}}}{T} = \mathbf{I} - \frac{\mathbf{m}_{\text{rest}}}{T}.$$

The encoding error over T steps is therefore:

$$\varepsilon = \frac{\mathbf{m}_{\text{rest}}}{T}, \quad \mathbf{m}_{\text{rest}} < \theta \implies \varepsilon < \frac{\theta}{T}.$$

With optimized initial membrane potential initialization, $\mathbf{m}_{\text{rest}} < \theta/2$ can be guaranteed Hao et al. (2023c), yielding a tighter bound:

$$\varepsilon < \frac{\theta}{2T}.$$

PROPOSED INTEGRATION CODING

The method proposed in this paper adopts integration coding, where the input signal $\mathbf{I}$ is applied only at the first timestep:

$$\mathbf{m} = \mathbf{I}.$$

In subsequent steps, the residual membrane potential is amplified without further input injection. Let n_i denote the amplification index at which the i-th spike is emitted; the corresponding spike amplitude is $\theta/2^{n_i}$. After n amplification operations, the input is decomposed as:

$$\mathbf{I} = \sum_i \frac{\theta}{2^{n_i}} + \frac{\mathbf{m}_{\text{rest}}}{2^n},$$

where the second term represents the residual encoding error. The encoding error is thus:

$$\varepsilon = \frac{\mathbf{m}_{\text{rest}}}{2^n}, \quad \mathbf{m}_{\text{rest}} < \theta \implies \varepsilon < \frac{\theta}{2^n}.$$

When the carry mechanism is applied, the error is further reduced by half:

$$\varepsilon < \frac{\theta}{2^{n+1}}.$$

Here, n denotes the total number of amplification operations. When $T = 1$, we have $n = D$. When $T > 1$, additional amplification operations are required between timesteps to satisfy the condition $\mathbf{m}_{\text{rest}} > \theta$ prior to spike emission, yielding:

$$n \geq T \cdot D,$$

and the error is bounded by:

$$\varepsilon < \frac{\theta}{2^{T \cdot D+1}}.$$

□

B RMPA EXAMPLE

We illustrate the internal dynamics of RMPA with a concrete example. Consider an input $\mathbf{I}^\ell = 1.4$. According to Eq. equation 3, the threshold θ^ℓ is set to 1 (2^p, $p = 0$). After a spike of $\theta^\ell = 1$ is emitted, it leaves $\mathbf{m}^\ell = 1.4 - 1$ as 0.4 and the index i of RMPA loop as 1. In the second iteration, $\mathbf{m}^\ell$ is doubled to 0.8 and i advances to 2; since $\mathbf{m}^\ell = 0.8 < \theta^\ell = 1$, no spike is emitted. In the third iteration, $\mathbf{m}^\ell$ is doubled to 1.6 and i advances to 3; since $\mathbf{m}^\ell = 1.6 > \theta^\ell = 1$, a spike of $\theta^\ell / 2^{i-1} = 0.25$ is fired and $\mathbf{m}^\ell$ is reset to $1.6 - 1 = 0.6$. At this point, the total encoded value is $1 + 0.25 = 1.25$, with residual membrane potential $\mathbf{m}^\ell = 0.6$. The remaining encoding error is $1.4 - 1.25$, the same as the result of $\mathbf{m}^\ell / 2^{i-1} = 0.6/4$, which decays exponentially with increasing iterations.

C ENERGY CONSUMPTION ANALYSIS

The energy savings achieved by ANN-to-SNN conversion stem from two primary sources: (1) replacing floating-point multiplication with lightweight addition operations, and (2) reducing bit-transmission cost between layers through the proposed spike encoding scheme. This section provides a detailed quantitative analysis of both aspects.

C.1 COMPUTATIONAL ENERGY CONSUMPTION

Based on the proposed spike acquisition method, all multiplication operations $w \times x$ are fully decomposed into integer-level additions and bit-shift operations. For reference, prior work has established that FP32 multiplication consumes 3.7 pJ and FP16 multiplication consumes 1.1 pJ Horowitz (2014), as summarized in Table 4.

Table 4: Energy Consumption of Common Hardware Operations

Integer Operation		**Floating-Point Operation**		**Memory Access (64-bit)**	
Operation	Energy	Operation	Energy	Memory	Energy
8-bit Add	0.03 pJ	16-bit FAdd	0.4 pJ	8 KB Cache	10 pJ
32-bit Add	0.1 pJ	32-bit FAdd	0.9 pJ	32 KB Cache	20 pJ
8-bit Mult	0.2 pJ	16-bit FMult	1.1 pJ	1 MB Cache	100 pJ
32-bit Mult	3.1 pJ	32-bit FMult	3.7 pJ	DRAM	1.3–2.6 nJ

Taking $T = 2$, $D = 3$ as a representative configuration (illustrated in Fig. 3 with $x = 1.4$, $w = 1.6$), the computation of $x \times w$ proceeds as follows.

Step 1: Spike extraction from x.

- *Cycle 1*: The spike is formed by the implicit leading 1 of the mantissa concatenated with the first $D - 1 = 2$ fractional bits.
- *Cycle 2*: The position Tr_2 of the first set bit among the remaining 8 mantissa bits is identified (here $\mathrm{Tr}_2 = 1$), yielding a raw spike of `100`. A carry operation is then applied: the bit immediately following `100` is 1, so the spike is corrected to $\texttt{100} + 1 = \texttt{101}$.
- *Remark*: Since the leading bit of each spike is always 1 by construction, it need not be transmitted explicitly; only the trailing $D - 1$ bits are communicated, further reducing bit-transmission cost between layers.

Step 2: Exponent and sign processing. The sign bit of the result is obtained by XOR-ing the sign bits of w and x. The exponent of the result is computed by summing the exponent fields of w and x and subtracting the bias of 15. The fractional part of w is then shifted and accumulated in parallel according to the spike value of each cycle:

- `spike = 101`: add the mantissa of w and its right-shift by 2 bits.
- `spike = 100`: retain the mantissa of w unchanged.
- `spike = 110`: add the mantissa of w and its right-shift by 1 bit.

- `spike = 111`: compute the difference between the left-shifted mantissa and its right-shift by 2 bits (realizable via two's complement addition).

Step 3: Cross-cycle accumulation. The fractional result from Cycle 1 is added to that of Cycle 2 right-shifted by $(\mathrm{Tr}_2 - 1)$ bits, accounting for any threshold scaling introduced by the carry operation. The final product $w \times x$ is assembled by concatenating the sign bit, the exponent, and the leading 11 bits of the accumulated mantissa.

Energy Estimation Basis

Following established hardware scaling rules, energy consumption scales sub-linearly with operand width: doubling the bit-width typically increases energy by less than $2\times$ B. Annapoorani (2023). On this basis, INT16 addition is estimated at 0.05–0.06 pJ (conservatively taken as 0.06 pJ), and sign-bit XOR is upper-bounded by INT8 addition at 0.03 pJ.

First-Cycle Energy

(1) **Sign-bit XOR**: 0.03 pJ.

(2) **Exponent summation**: summing exponent bits and subtracting the bias consumes 0.06 pJ.

(3) **Spike–mantissa multiplication**:

D=1: Retain mantissa directly; 0 pJ.

D=2: With probability $1/2$ (`spike=10`): retain directly (0 pJ); with probability $1/2$ (`spike=11`): one INT16 addition. Average: $(0 + 0.06)/2 = 0.03$ pJ. Total with sign/exponent: 0.045 pJ.

D=3: With probability $1/4$ (`spike=100`): 0 pJ; with probability $1/2$ (`spike=101,110`): one INT14 addition (≈ 0.06 pJ); with probability $1/4$ (`spike=111`): complement-based subtraction (estimated at $1.5\times$ INT addition). Average: 0.075 pJ.

(4) **Rounding**: one rounding operation on the mantissa at the end of each multiplication, 0.03 pJ, overlapped with Cycle 1.

Subsequent-Cycle Energy ($T \geq 2$)

(1) **Leading-one detection**: locating the first set bit and recording offset p is negligible; upper-bounded by INT8 addition at 0.03 pJ.

(2) **Spike–mantissa multiplication**: identical rules as Cycle 1; average energy is 0 pJ, 0.045 pJ, and 0.075 pJ for $D = 1, 2, 3$, respectively.

(3) **Cross-cycle accumulation**: right-shift by $(D + Tr_2 - 1)$ bits followed by addition. The maximum operand widths are INT24 ($D = 2$) and INT25 ($D = 3$), both truncated to INT16 addition: $0.03 + 0.06 = 0.09$ pJ.

The energy requirements for all subsequent cycles ($T > 2$) are identical to those of Cycle 2. The total energy consumption in representative configurations (T, D) is reported in Table 5.

C.2 Bit-transmission cost between layers Evaluation

In traditional schemes, the bit-transmission cost between layers depends on the data type: if the FP32 format is adopted, 32-bit data needs to be transmitted for each multiplication calculation; if the FP16 or BF16 format is adopted, 16-bit data needs to be transmitted for each multiplication calculation. The proposed bit-transmission cost between layers consists of three parts: 1 bit sign information, 5 bit threshold information, $(D - 1)$ bit spike information transmitted in the first cycle (the first bit defaults to 1 and can be omitted), and $(D - 1 + 3)$ bit spike information transmitted in subsequent cycles (where the additional 3 bits are used to represent the threshold offset Tr_t, with an upper limit of 8 bits: 000 for 1 and 111 for 8, and offsets exceeding 8 bits are directly ignored). The detailed transmission volume in different configurations is summarized in Table 7. As can be seen in Table 7, the transmission bit width of the proposed scheme is significantly lower than that of traditional ANN schemes, thus further saving energy consumption in the data transmission process.

Table 5: Total Energy Consumption and Energy Ratio Relative to FP32 for Different $(T,\ D)$ Configurations (Unit: pJ)

Energy (pJ)	D=1	D=2	D=3
T=1	0.090	0.210	0.330
T=2	0.165	0.330	0.495
T=3	0.195	0.390	0.585
FP16 (ANN)		1.100	
FP32 (ANN)		3.700	
Energy Ratio (SNN/FP32)	T=1	T=2	T=3
D=1	2.43%	5.68%	8.92%
D=2	4.46%	8.92%	13.38%
D=3	5.27%	10.54%	15.81%

Table 6: Total Energy Consumption and Energy Ratio for Different $(T,\ D)$ Configurations (Unit: pJ)

Energy (pJ)	D=1	D=2	D=3
T=1	0.090	0.210	0.330
T=2	0.165	0.330	0.495
T=3	0.195	0.390	0.585
FP16 (ANN)		1.100	
FP32 (ANN)		3.700	
Energy Ratio (SNN/FP16)	T=1	T=2	T=3
D=1	8.18%	19.09%	30.00%
D=2	15.00%	30.00%	45.00%
D=3	17.73%	35.45%	53.18%

Table 7: Bit-transmission Cost Between Layers Under Different Exponent Bit Widths and Parameter Configurations

exp_dim	$\boldsymbol{D}$	$\boldsymbol{T}$=1	$\boldsymbol{T}$=2	$\boldsymbol{T}$=3
5	1	6	9	15
	2	7	11	18
	3	8	13	21
4	1	5	8	14
	2	6	10	17
	3	7	12	20
FP32 (ANN)			32	
FP16 or BF16 (ANN)			16	

D SPECIAL CASE: SPIKE = 111

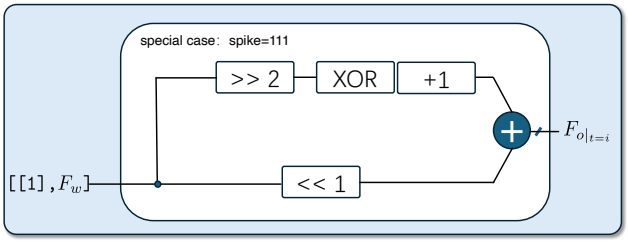


Figure 4: A special case for input spikes equals "111"

When $D = 3$ and spike $= 111$, the naive shift-add formulation requires three terms:

$$[1, F_w] \cdot 111_2 = [1, F_w] + ([1, F_w] \gg 1) + ([1, F_w] \gg 2), \tag{10}$$

which would naively require a two-stage adder tree. However, this case admits a more efficient decomposition. Observing that $111_2 = 1000_2 - 0001_2$, and noting that multiplication by 1000_2 and 0001_2 correspond to a left shift by 1 bit and a right shift by 2 bits, respectively (relative to the implicit leading bit), we obtain:

$$[1, F_w] \cdot 111_2 = ([1, F_w] \ll 1) - ([1, F_w] \gg 2),$$

which requires only a single subtraction. In hardware, subtraction is implemented via two's complement addition: negating $[1, F_w] \gg 2$ is achieved by a bitwise complement followed by adding 1 (absorbed into the carry-in of the adder), yielding:

$$[1, F_w] \cdot 111_2 = ([1, F_w] \ll 1) + \overline{([1, F_w] \gg 2)},$$

where $\ll$ and $\gg$ denote left and right logical shift respectively, and $\overline{(\cdot)}$ denotes the two's complement negation. As illustrated in Fig. 4, the datapath consists of a left-shift-by-1 unit on the lower path, a right-shift-by-2 unit followed by a two's complement unit on the upper path, and a single adder combining both results.

This decomposition ensures that, for all spike patterns with $D \leq 3$, each intra timestep computation requires **at most one adder**, with no case requiring more than a single addition or subtraction operation. The hardware overhead is therefore strictly bounded regardless of the spike value, confirming the area efficiency of the proposed design.

E SINGLE-STEP VS. MULTI-STEP EXECUTION MODES

In ANN-to-SNN conversion, two execution modes can be distinguished based on the temporal scheduling strategy: *single-step mode* and *multi-step mode*.

Single-Step Mode. In single-step mode, the input data at each time step propagates through the entire network sequentially, layer by layer, before the next time step begins. That is, at time step t, the spike train is fully processed across all layers—from input to output—prior to the commencement of computation at time step $t + 1$.

Multi-Step Mode. In multi-step mode, the execution order is transposed: for each layer, computations across all T time steps are completed before proceeding to the next layer. Specifically, layer ℓ processes inputs for all time steps $t = 1, 2, \ldots, T$ and only then forwards the accumulated results to layer $\ell + 1$.

Recommendation for RMPA. Even when RMPA is deployed without its hardware-level optimizations, we recommend adopting multi-step mode for the following reasons:

- **Hardware compatibility.** Multi-step mode aligns more naturally with the dataflow patterns of contemporary hardware accelerators, where minimizing memory access overhead is critical for sustained throughput.

- **Reduced weight reloading.** By completing all time steps within a given layer before advancing to the next, multi-step mode eliminates the need to repeatedly reload the same synaptic weights for each time step, thereby significantly reducing memory bandwidth pressure.
- **Improved parallelism.** Multi-step mode naturally exposes intra-layer parallelism across time steps, enabling more efficient utilization of vectorized or batched computation resources.

Adapting RMPA for Single-Step Mode. If single-step mode is required by the deployment scenario, a minor modification to RMPA is necessary. Specifically, for the second and all subsequent time steps ($t \geq 2$), when newly generated spike-driven data is forwarded to downstream layers, the membrane potential m_ℓ of layer ℓ must be updated by incorporating the incoming contribution scaled by 2^{i-1}:

$$\mathbf{m}_\ell(t) \leftarrow V_\ell(t) + 2^{i-1} \cdot \mathbf{I}_\ell(t), \qquad t \geq 2, \tag{11}$$

where $\mathbf{I}_\ell(t)$ denotes the input increment arriving at layer ℓ at time step t, and i is the number of RMPA loop. This scaling ensures that the accumulated membrane potential correctly reflects the appropriately amplified value.

F Ablation Study on Query Spiking Strategy

For the spiking conversion of Transformer models, this paper investigates the trade-off between computational efficiency and model accuracy in the treatment of the query (q) representation. By default, q is retained in floating-point form and excluded from spike conversion, while spiking conversion is applied exclusively to the key (k) and value(v). This design treats q as a weight-like parameter, which eliminates floating-point multiply-accumulate operations in the attention mechanism while preserving model accuracy. To assess the impact of this design choice, we conduct ablation experiments on whether to convert q into spikes, with results reported in Table 8.

Table 8: Experiments on ViT models under different (T, D) settings. `q_include` denotes the variant that also converts the query (q) into spikes (in addition to k and v), while `Default` converts only k and v.

Task	Model	ANN	D	Default			q_include		
				T=1	T=2	T=3	T=1	T=2	T=3
ImageNet-1K	vit_small_patch16_224	78.04	1	48.20	76.27	77.80	42.34	76.25	77.87
			2	73.78	77.94	78.03	73.26	77.93	78.04
			3	77.01	78.06	78.03	76.94	78.06	78.03
	vit_base_patch16_224	82.91	1	74.00	82.21	82.74	72.37	82.17	82.73
			2	80.89	82.77	82.89	80.71	82.76	82.87
			3	82.41	82.87	82.91	82.43	82.87	82.92
	vit_large_patch16_224	84.88	1	80.89	84.46	84.79	80.26	84.43	84.79
			2	84.03	84.80	84.89	83.96	84.81	84.89
			3	84.45	84.89	84.88	84.41	84.89	84.88
MMLU	qwen3_0.6b_base	52.41	1	29.06	50.29	52.11	25.58	49.47	52.31
			2	45.85	52.55	52.56	43.40	52.24	52.73
			3	51.08	52.60	52.49	50.11	52.60	52.60
	qwen3_1.7b_base	62.61	1	41.59	60.80	62.37	31.94	60.12	62.47
			2	57.83	62.42	62.53	55.62	62.34	62.78
			3	60.87	62.26	62.50	60.93	62.63	62.56
	qwen3_4b_base	72.90	1	57.58	72.34	72.70	43.85	71.57	72.72
			2	69.29	72.75	73.10	67.40	72.62	73.07
			3	72.21	72.92	73.04	71.85	73.00	73.04
	qwen3_8b_base	76.85	1	65.26	76.13	76.86	54.42	75.97	76.51
			2	74.88	76.71	76.78	73.06	76.59	76.81
			3	76.45	76.90	76.81	75.99	76.81	76.73
	qwen3_14b_base	80.47	1	70.70	79.73	80.39	62.08	79.58	80.37
			2	78.63	80.47	80.45	77.43	80.42	80.65
			3	80.37	80.49	80.49	79.83	80.40	80.44

G Additional Experimental Results on ImageNet-1K

This section presents comprehensive experimental results of our SNN conversion framework on the ImageNet-1K dataset across various models, including EVA02, ViT, ResNet, and VGG. We report accuracy in different timesteps T and spike dimensions D in table 9. The ANN baseline accuracy is provided for reference.

Table 9: Accuracy (%) on ImageNet-1K Dataset under Different T and D Settings

Model	$T=1$	$T=2$	$T=3$	ANN
eva02_tiny_patch14_336				
$d=1$	73.26	80.11	80.53	80.62
$d=2$	78.82	80.60	80.65	
$d=3$	80.19	80.63	80.61	
eva02_small_patch14_336				
$d=1$	80.18	85.22	85.68	85.73
$d=2$	84.54	85.74	85.75	
$d=3$	85.45	85.70	85.73	
eva02_base_patch14_448				
$d=1$	81.53	87.85	88.25	88.26
$d=2$	86.88	88.28	88.26	
$d=3$	88.00	88.27	88.26	
eva02_large_patch14_448				
$d=1$	85.66	89.32	89.57	89.63
$d=2$	88.88	89.60	89.63	
$d=3$	89.46	89.63	89.63	
eva_giant_patch14_336				
$d=1$	88.65	89.41	89.57	89.62
$d=2$	89.27	89.56	89.60	
$d=3$	89.47	89.58	89.62	

Model	$T=1$	$T=2$	$T=3$	ANN
vit_small_patch16_224				
$d=1$	48.20	76.27	77.80	78.04
$d=2$	73.78	77.94	78.03	
$d=3$	77.01	78.06	78.03	
vit_base_patch16_224				
$d=1$	74.00	82.21	82.74	82.91
$d=2$	80.89	82.77	82.89	
$d=3$	82.41	82.87	82.91	
vit_large_patch16_224				
$d=1$	80.89	84.46	84.79	84.88
$d=2$	84.03	84.80	84.89	
$d=3$	84.45	84.89	84.88	
resnet34				
$d=1$	20.23	72.45	75.83	76.09
$d=2$	66.59	76.08	76.12	
$d=3$	74.81	76.13	76.09	
vgg16				
$d=1$	49.82	68.86	70.97	71.12
$d=2$	65.25	70.94	71.10	
$d=3$	69.72	71.08	71.12	

H Experimental Results on CIFAR datasets

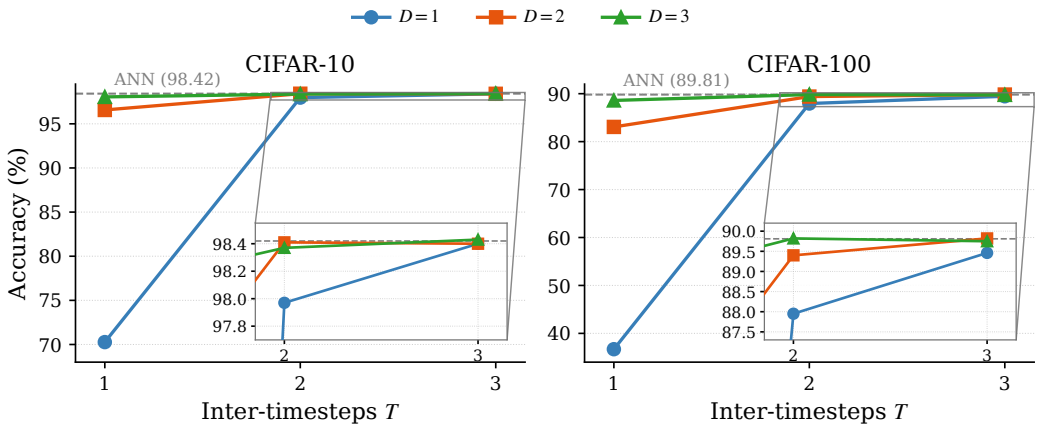


Figure 5: Test accuracy of ViT-S on CIFAR-10 (left) and CIFAR-100 (right) under different settings of T and D.

We evaluate the converted ViT-S model in CIFAR tasks under different combinations of parameters and report the post-conversion accuracy in figure 5. For CIFAR-10, with $D=1$, our method achieves **98.40%** accuracy at $T=3$, closely matching the ANN baseline (98.42%) while requiring only 3 timesteps. Remarkably, even at $T=2$, the accuracy reaches 97.97%, with a negligible drop of 0.45%.

For CIFAR-100, our method attains **89.46%** at $T = 3$ and 87.95% at $T = 2$, again approaching the baseline of the ANN of 89.81%.

We further observe that deeper networks enable faster convergence. With $D = 3$, the SNN on CIFAR-10 already reaches 98.05% at $T = 1$ and saturates at 98.37% for $T = 3$, effectively matching the ANN. Similarly, on CIFAR-100, $D = 3$ yields 88.59% at $T = 1$ and 89.82% at $T = 2$, narrowing the gap to within 0.06% of the ANN. These results demonstrate that our method achieves almost lossless conversion with as few as $T = 2$ timesteps, and deeper architectures further reduce the required latency.

I Experimental Results on MMLU Benchmark

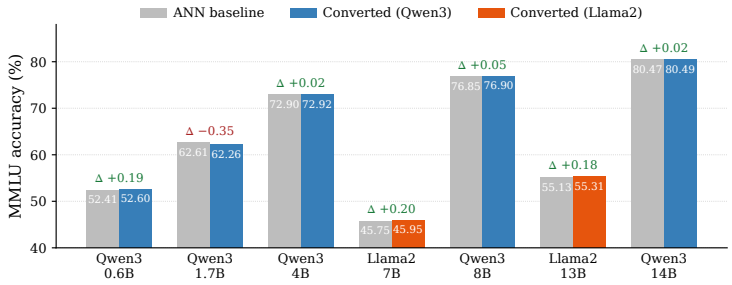


Figure 6: Conversion results on MMLU for different models

The Qwen3Yang et al. (2025) and Llama2Touvron et al. (2023) series models are evaluated on the MMLU benchmark in this work. We perform comprehensive evaluations on this key NLP benchmark, and the corresponding results are shown in figure 6.

J ADDITIONAL EXPERIMENTAL RESULTS OF LLAMA2-7B AND LLAMA2-13B

This section presents comprehensive experimental results of our SNN conversion framework on various NLP tasks for the LLaMA2-7B and LLaMA2-13B models in table 10. We report accuracy in different timesteps T and spike dimensions D. The ANN baseline accuracy is provided for reference.

Table 10: Accuracy (%) of LLaMA2-7B and LLaMA2-13B on NLP Tasks under Different T and D Settings

	LLaMA2-7B				**LLaMA2-13B**			
Task	T=1	T=2	T=3	ANN	T=1	T=2	T=3	ANN
MMLU								
d=1	26.90	43.88	45.93	45.75	33.76	53.62	54.99	55.13
d=2	39.97	45.67	45.73		51.40	55.05	55.23	
d=3	44.92	45.95	45.93		54.15	55.31	55.14	
PIQA-norm								
d=1	72.14	79.11	78.56	78.45	74.32	80.09	80.52	80.52
d=2	77.37	78.73	78.84		79.60	80.69	80.25	
d=3	78.78	78.67	78.51		80.25	80.58	80.47	
ARC-Challenge norm								
d=1	33.19	44.88	45.48	45.73	39.76	48.55	48.55	48.89
d=2	43.60	45.48	45.65		47.78	48.72	48.38	
d=3	46.16	45.65	45.99		49.40	49.15	48.55	
ARC-Easy norm								
d=1	58.33	73.61	74.75	74.87	66.79	75.93	76.60	76.56
d=2	72.26	73.99	74.54		75.51	76.26	76.47	
d=3	73.82	74.75	74.66		75.80	76.47	76.43	
Winogrande norm								
d=1	60.38	69.38	69.46	69.38	62.67	71.90	71.90	72.22
d=2	67.01	69.53	69.30		71.43	72.53	72.22	
d=3	67.56	69.06	69.46		70.56	72.14	72.53	
HellaSwag norm								
d=1	65.61	75.72	76.30	76.28	66.05	79.04	79.55	79.71
d=2	74.65	76.25	76.18		78.57	79.63	79.71	
d=3	75.77	76.18	76.25		79.23	79.69	79.72	

K MMLU RESULTS OF QWEN3 SERIES MODELS

This section supplements the accuracy of the MMLU task of the Qwen3 series with different timesteps T and spike dimensions D in table 11, with the original ANN accuracy provided as the baseline reference.

Table 11: MMLU Accuracy (%) of Qwen3 Series Models under Different T and D Settings

Model	T=1	T=2	T=3	**ANN**
qwen3-0.6b-base				
d=1	29.06	50.29	52.11	52.41
d=2	45.85	52.55	52.56	
d=3	51.08	52.60	52.49	
qwen3-1.7b-base				
d=1	41.59	60.80	62.37	62.61
d=2	57.83	62.42	62.53	
d=3	60.87	62.26	62.50	
qwen3-4b-base				
d=1	57.58	72.34	72.70	72.90
d=2	69.29	72.75	73.10	
d=3	72.21	72.92	73.04	

Model	T=1	T=2	T=3	**ANN**
qwen3-8b-base				
d=1	65.26	76.13	76.86	76.85
d=2	74.88	76.71	76.78	
d=3	76.45	76.90	76.81	
qwen3-14b-base				
d=1	70.70	79.73	80.39	80.47
d=2	78.63	80.47	80.45	
d=3	80.37	80.49	80.49	

L SUPPORT FOR MORE FORMATS

We emphasize that based on ANN2SNN, this framework can further combine with model quantization strategies, bringing additional benefits in data movement and energy consumption. The two complement each other, making it more friendly for edge AI. For this reason, we explored the efficiency and performance comparison of models under extreme quantization conditions (INT4). Following the energy estimation method for ultra-low bit operations described in Liu et al. (2020); Qin et al. (2020), we set the energy consumption of single-bit logic gate operations to 3.75 fJ and calculated the efficiency of the quantized model under INT4 or FP8 conditions and the conversion model under the PRIME framework (as shown in Table 12,14).

L.1 FP8 FORMAT

Table 12 summarizes the energy consumption and bit-transmission cost after PRIME conversion on FP8 operands.Table 13 reports the MMLU accuracy of Qwen3-0.6B in FP8 format before and after PRIME conversion.

Table 12: Energy and bit-transmission cost of FP8 format after PRIME conversion.

	Energy (pJ)	Energy Ratio (vs. FP8)	Transmission Bits	Transmission Ratio (vs. FP8)
T1D1	0.0375	13.39%	5	62.50%
T1D2	0.04875	17.41%	6	75.00%
T2D1	0.07125	25.45%	6	75.00%
T2D2	0.09375	33.48%	8	100.00%
FP8	0.28	100.00%	8	100.00%

Table 13: MMLU accuracy of Qwen3-0.6B-FP8 before and after PRIME conversion.

Config	MMLU Acc (%)
Pure FP8 (Baseline_a8w8)	26.78
T1D1	16.38
T1D2	23.84
T2D1	25.76
T2D2	26.66

L.2 INT4 FORMAT

Table 14 summarizes the energy and bit-transmission analysis for INT4 operands.Table 15 reports the accuracy of LLaMA2-7B under FlatQuant W4A4 quantization and further PRIME conversion.

Table 14: Energy and bit-transmission cost of INT4 format after PRIME conversion.

	Energy (pJ)	Energy Ratio (vs. INT4)	Transmission Bits	Transmission Ratio (vs. INT4)
T1D1	0.01125	22.50%	3	75.00%
T1D2	0.016875	33.75%	4	100.00%
T2D1	0.0375	75.00%	4	100.00%
INT4	0.05	100.00%	4	100.00%

Table 15: Accuracy comparison of FlatQuant W4A4 and further PRIME conversion on LLaMA2-7B.

	PPL(wiki)	PPL(c4)	ARC_c	ARC_e	Hella	LAMB	PIQA	Wino
FlatQuant W4A4	5.69	7.62	44.37	73.11	74.80	72.27	77.86	67.88
PRIME T1D1	9.50	13.79	35.49	63.68	64.62	62.62	75.19	62.04
PRIME T1D2	6.15	8.33	40.44	69.70	72.52	70.64	77.20	66.46
PRIME T2D1	6.17	8.38	43.00	70.03	72.58	70.06	76.44	64.25

M LIMITATIONS

Our current evaluation is simulation-based and does not include silicon measurements. However, since all underlying primitives (bit-shift, integer addition, and leading-one detection) are naively supported by standard digital logic, the proposed neuron dynamics require no custom analog or mixed-signal circuitry. Physical implementation is therefore a natural next step, which we leave to our future work.