Achieve Latency-Efficient Temporal-Coding Spiking LLMs via Discretization-Aware Conversion

Jinjie Fang[1]* **Tianxing Man**[2]* **Chengxun Jin**[1] **Xiao Du**[1]
Haozhen Zhang[1] **Yi Chang**[1,3,4]† **Bin Gu**[1]†
[1]School of Artificial Intelligence, Jilin University, China
[2]College of Computer Science, Inner Mongolia University, China
[3]International Center of Future Science, Jilin University, China
[4]Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, MOE, China
{fangjj24, jincx1521, haozhen23}@mails.jlu.edu.cn
{mantx, duxiao18, yichang, gubin}@jlu.edu.cn

Abstract

Large language models (LLMs) have achieved remarkable success while introducing critical energy bottlenecks that challenge sustainable deployment. Spiking neural networks (SNNs) provide a promising approach for energy-efficient spiking LLMs via ANN-to-SNN (A2S) conversion. Among various spike coding methods, time-to-first-spike (TTFS) coding is particularly appealing as it conveys information with a single spike, further reducing energy consumption. However, existing TTFS-based A2S conversion relies on continuous-time assumptions, requiring prohibitively large latencies (e.g., 4096 time steps) to approximate ANN's continuous values. This dependency leads to unacceptable inference delays in LLMs, posing significant challenges for developing practical temporal-coding spiking LLMs. In this paper, we propose a discrete analysis framework for TTFS-based LLMs, reformulating TTFS-based A2S conversion from continuous lossless conversion into discrete error-aware representation. Building on this analysis, we establish an equivalence between discrete TTFS-based SNNs and static quantized ANNs, allowing TTFS conversion to be handled through quantization techniques. Motivated by this equivalence, we introduce Quantization-Consistent ANN-to-SNN (QC-A2S) conversion, which combines static quantization with discretization-compatible TTFS neurons to achieve latency-efficient and high-performance temporal-coding spiking LLMs. Extensive evaluations on representative LLMs across multiple benchmarks demonstrate competitive performance with substantially reduced inference latency.

1 Introduction

Large Language Models (LLMs) represent a paradigm shift in artificial intelligence, leveraging deep learning architectures trained on massive text corpora to capture intricate linguistic patterns, syntactic structures, and semantic relationships, thereby achieving remarkable capabilities in natural language understanding and generation [66, 53, 1, 15]. Most LLMs are built upon the Transformer architecture, which relies heavily on multi-head attention mechanisms and dense matrix multiplications, resulting in cubic computational complexity and substantial energy consumption during both training and inference [54, 68]. Moreover, following the "scaling law", LLMs have grown from billions to trillions of parameters to achieve better performance, which further increases computational demand [5, 21]. Consequently, the critical challenge facing the LLM community is developing approaches to reduce energy consumption while preserving model performance capabilities.

*Equal Contribution.
†Corresponding Authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Spiking Neural Networks (SNNs) are biologically plausible computational models inspired by the mechanisms of neurons and synapses in the human brain [32, 41]. SNNs transmit and compute information asynchronously through discrete spike events rather than continuous-valued activation functions, demonstrating remarkable energy efficiency when implemented on specialized neuromorphic hardware [61, 71, 13, 34]. Consequently, developing **Spiking LLMs** has emerged as a promising solution to address the substantial energy consumption challenges of LLMs. Currently, two primary approaches are used to develop spiking LLMs: direct training methods that incorporate surrogate gradients to address non-differentiability [61, 36, 70], and ANN-to-SNN (A2S) conversion methods that transfer pre-trained weights while preserving approximate equivalence through carefully designed techniques [24, 6, 20]. Given the enormous computational requirement of direct training for LLMs, practical spiking LLMs are predominantly achieved through A2S conversion for energy-efficient intelligent applications in resource-constrained environments [58].

Beyond the rate coding commonly used in A2S conversion methods, recent neuroscience research has highlighted temporal-based spike coding that offer superiors energy efficiency advantages [38, 65, 49]. Temporal coding represents continuous values through precise spike timing rather than spike counts, suggesting that the representation of information depends on when the spikes occur [19]. Among various temporal codings, **time-to-first-spike (TTFS) coding** is particularly noteworthy, as it encodes information in the latency of a single spike, which substantially reduces energy consumption by minimizing spike counts [39, 42]. Specifically, the spike firing rate of TTFS coding is inversely proportional to the time window T (denoted as $1/T$, as illustrated in Table 1).

Table 1: Firing rates of TTFS coding across different time windows T

Time window (T)	2	4	8	16	32	64	128	256	512
Firing rate (%)	50.00	25.00	12.50	6.250	3.125	1.563	0.7813	0.3906	0.1953

Existing TTFS-based conversion methods are primarily built upon continuous-time assumptions, under which ANN values can be represented with infinitely fine temporal precision, thereby enabling theoretically lossless ANN-to-SNN conversion [67, 49]. However, applying this paradigm to LLMs requires extremely large time windows to preserve such fine-grained temporal representation, resulting in prohibitively high inference latency (Figure 1) and making the idealized continuous correspondence impractical. Consequently, the excessive inference latency introduced by continuous-time assumptions severely limits the practicality of **temporal-coding spiking LLMs**.

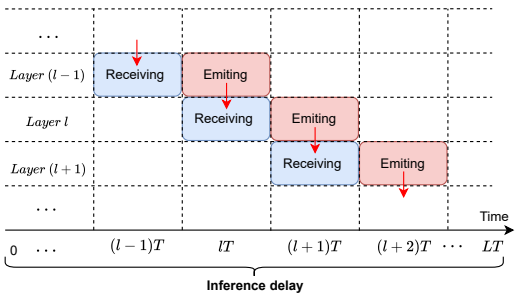


Figure 1: Inference delay across network layers.

To address this challenge, we reformulate TTFS-based LLM construction as a discrete error analysis problem rather than as an ideal continuous lossless conversion problem. We propose a discretization-aware theoretical framework that explicitly characterizes the practical errors introduced by finite-latency spike-time representation and reveals their dependence on inference latency. Building on this framework, we further establish an equivalence between discrete TTFS-based SNNs and static quantized ANNs, which transforms TTFS conversion into a quantization-compatible problem. Based on this equivalence, we introduce Quantization-Consistent ANN-to-SNN Conversion (QC-A2S), which combines static quantization with discretization-compatible TTFS neurons to align quantized activations with discrete spike dynamics. As a result, QC-A2S effectively mitigates conversion errors and enables high-performance temporal-coding spiking LLMs under low inference latency (Figure 2).

The key contributions are summarized as follows:

- We propose a discrete error analysis framework for TTFS-based LLMs, shifting TTFS conversion from ideal continuous lossless conversion to practical error-aware discrete representation, and theoretically characterizing finite-latency conversion errors.

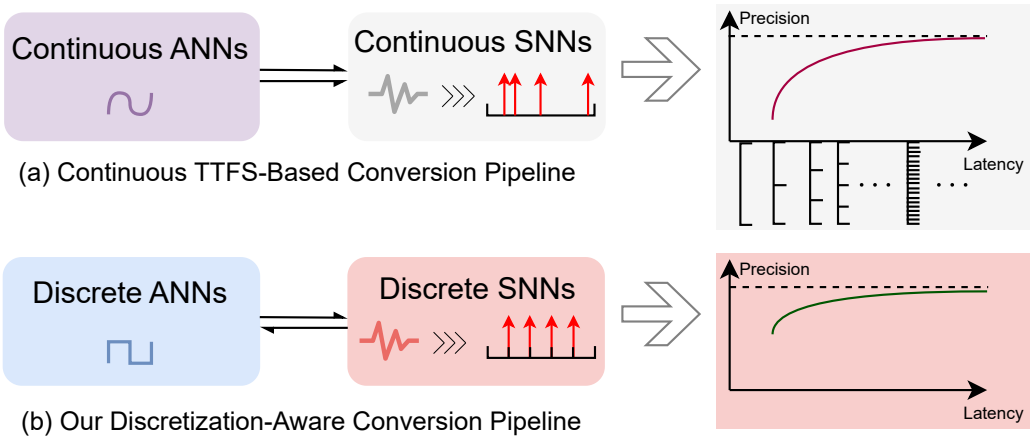


Figure 2: Overview of the QC-A2S framework.

- We establish an equivalence between discrete TTFS-based SNNs and static quantized ANNs, based on which we introduce QC-A2S to leverage quantization techniques and discretization-compatible TTFS neurons for low-latency, high-performance spiking LLMs.
- Extensive evaluations on representative LLMs and diverse benchmarks show that QC-A2S achieves competitive performance while substantially reducing inference latency.

2 Related Works

Spiking LLMs. The success of LLMs has motivated the development of SNN counterparts (spiking LLMs) that maintain energy efficiency while achieving comparable capabilities. Several approaches have emerged for creating spiking variants of transformer-based models [62, 71, 69]. SpikeGPT replaces traditional self-attention with Spiking RWKV mechanisms [72]. SpikingBERT employs a two-stage knowledge distillation method that utilizes pre-trained BERT models as teachers to train spiking student architectures [31]. Similarly, SpikingMiniLM builds upon BERT with parameter initialization and A2S distillation methods to achieve faster convergence during training. Recent work introduced SpikeLLM, scaling to 70 billion parameters through spike-driven quantization [59, 58]. However, existing spiking LLMs rely exclusively on rate coding, where information is encoded through spike frequency. This leaves unexplored the potential of temporal-based spiking LLMs, which could achieve substantially lower energy consumption.

Temporal-based A2S Conversions. While rate-based conversion methods have dominated ANN-to-SNN conversion research, temporal-based encoding approaches offer compelling advantages in terms of energy efficiency by leveraging precise spike timing rather than spike frequency. These methods include time-to-first spike [50], reverse coding [65, 39], phase coding [35] and burst coding [38]. Among temporal coding schemes, TTFS coding has emerged as particularly promising, where each neuron emits at most one spike per time window with information encoded in the spike latency. Early TTFS-based conversion methods were developed by [42] and further improved by [65] and [39], but these approaches introduced conversion errors across layers. A breakthrough came with [48, 49], who demonstrated exact mapping from ReLU-based networks to SNNs using TTFS coding through a two-stage neuron activation process, achieving lossless conversion while maintaining energy benefits. Recently, [67] proposed TTFSFormer, the first TTFS-based conversion framework for Transformer architectures. However, existing TTFS-based conversion methods require extremely high latency to match continuous-time assumptions, preventing their implementation on large-scale models.

Model Quantization. Quantization has emerged as a critical technique for reducing model size and memory consumption, enabling efficient deployment of LLMs on resource-constrained devices [45], falling into two primary categories: quantization-aware training (QAT) [29] and post-training quantization (PTQ) [57]. QAT optimizes quantized weights during training using techniques like the straight-through estimator [10, 14] but is computationally impractical for LLMs. PTQ has

thus become the preferred approach, requiring only minimal calibration data while using dynamic activation quantization to address outlier-induced accuracy degradation [16]. Recent PTQ advances address outlier-induced errors using orthogonal transformations (QuaRot [2], SpinQuant [30]) or dual transformations (DuQuant [28]) to redistribute outliers across channels. However, these methods require computationally expensive per-token dynamic computation during inference. PrefixQuant [8] offers an alternative by isolating token-wise outliers to enable efficient per-tensor static quantization, achieving comparable performance.

3 Revisiting Continuous TTFS-based ANN-to-SNN Conversion

3.1 Continuous TTFS-based Lossless Conversion

Existing TTFS-based SNN methods, exemplified by TTFSFormer [67], are primarily built upon continuous-time assumptions, aiming to **establish a theoretically lossless ANN-to-SNN conversion** under ideal conditions. Their key idea is to reinterpret ANN computation as computation in a temporal-coordinate space. Specifically, an ANN activation is represented by the timing of a single spike, and each numerical operation is implemented by spiking neurons whose inputs and outputs are encoded as spike-time coordinates.

Let T_{ref} be the zero-reference time, τ be time constant, and $T = T_{\mathrm{end}} - T_{\mathrm{emit}}$ be time window. The time-to-value mapping and SNN's representable output range are determined by:

$$x = \frac{1}{\tau}(T_{\mathrm{ref}} - t), \quad a = \frac{1}{\tau}(T_{\mathrm{ref}} - T_{\mathrm{end}}), \quad b = \frac{1}{\tau}(T_{\mathrm{ref}} - T_{\mathrm{emit}}). \tag{1}$$

For a fixed output range $[a, b]$, the numerical resolution is approximately

$$\Delta x = \frac{\tau}{T}(b - a). \tag{2}$$

This lossless equivalence relies on two idealized assumptions. First, it requires **continuous temporal precision**, whereby spike-time coordinates can be represented with arbitrarily fine resolution. This condition typically demands a sufficiently large time window, such that the induced value interval satisfies $\Delta x \to 0$. Second, it assumes **complete output-range coverage**, meaning that the SNN's representable interval fully contains the corresponding ANN output, $\alpha \in [a, b]$, thereby ensuring that clipping remains inactive.

3.2 Empirical Observation: TTFSFormer Struggles in LLMs

As shown in Figure 3(a), directly applying TTFSFormer to LLMs results in severe performance degradation. This finding highlights a fundamental mismatch between the idealized continuous-time formulation of TTFS-based conversion and the discrete, finite-latency constraints of practical LLM inference. We attribute this mismatch to two dominant sources of error, each corresponding to the violation of one key assumption in continuous TTFS-based lossless conversion (Figure 3(b)):

Low-latency discretization error occurs when the assumption of continuous temporal precision breaks down. Although ideal TTFS encoding can represent ANN values with arbitrarily fine spike-time coordinates, practical TTFS-based LLMs must use finite time windows to maintain low inference latency. A smaller time window T yields a coarser temporal grid, thereby increasing discretization error. **Outlier-induced range error** occurs when complete output-range coverage is not satisfied. TTFSFormer relies on predefined SNN output intervals to map ANN activations to spike times, with these intervals typically calibrated on a small subset before conversion. However, LLM activations often follow heavy-tailed distributions with prominent outliers. Consequently, the calibrated bounds may not cover the activation values encountered during inference, as shown in Figure 3(d). When an ANN activation α lies outside the representable interval, i.e., $\alpha \notin [a, b]$, the TTFS representation is clipped, leading to non-negligible conversion error.

Moreover, the depth of LLMs further exacerbates these errors. Because the output spike times of each layer become the input temporal coordinates of the next, discretization and range-mismatch errors accumulate across layers, as illustrated in Figure 3(c). This motivates us to move beyond idealized, continuous-time lossless conversion and reformulate TTFS-based LLM construction as a discrete error analysis problem, in which errors induced by finite latency and activation outliers can be explicitly characterized and mitigated.

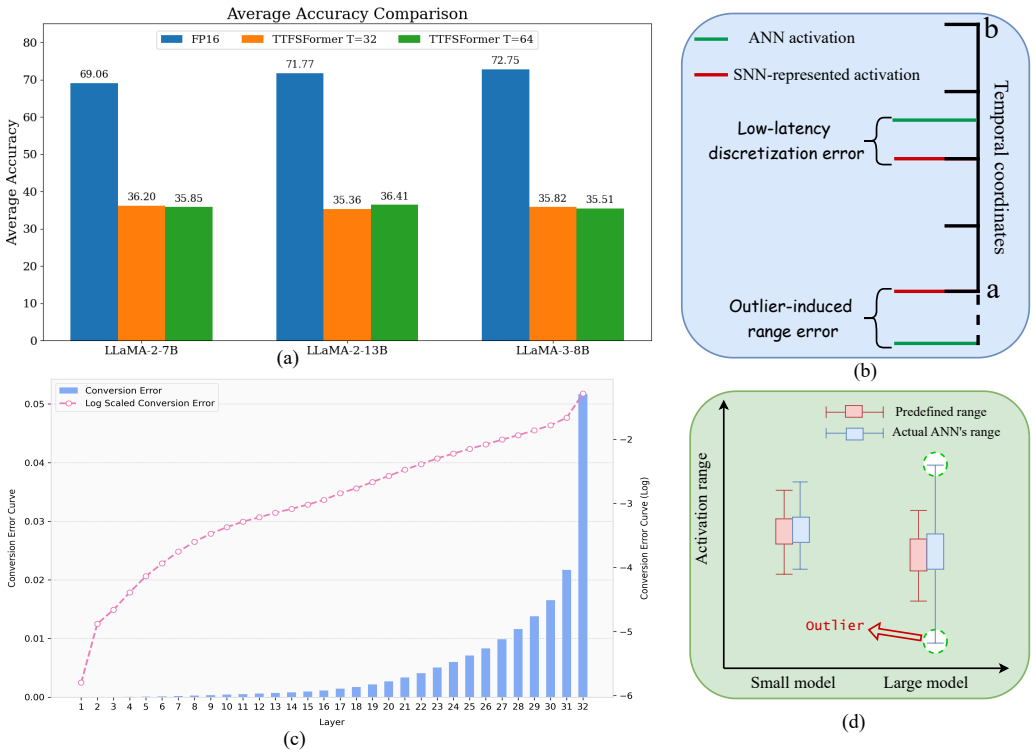


Figure 3: (a) Average accuracy comparison between FP16 LLMs and TTFSFormer; (b) Discretization and range mismatch errors in finite-latency TTFS coding; (c) Layer-wise accumulation of TTFS conversion errors; (d) Mismatch between predefined ranges and actual ANN activation ranges.

Challenge: *How can we develop a discrete theoretical framework that characterizes and mitigates conversion errors in finite-latency TTFS-based LLMs, thereby enabling accurate, low-latency temporal-coding spiking LLMs?*

4 Quantization-Aware Conversion via Discrete Error Analysis

To address this challenge, we shift from continuous-time idealization to a discrete-time formulation grounded in finite-latency constraints. Rather than pursuing theoretically lossless conversion under unrealistic assumptions, we explicitly account for the limitations of discrete temporal encoding. Our key idea is to systematically characterize and control conversion-induced discrepancies, particularly discretization errors from finite temporal resolution and range errors caused by quantization and bounded representable intervals. This perspective provides the foundation for a quantization-aware conversion framework that enables accurate, low-latency TTFS-based spiking LLMs. The detailed proofs of all theoretical results are deferred to Appendix F.

4.1 Discrete Error Analysis Framework for TTFS-based Conversion

First, we present a discretized version of TTFS-based neurons. At the i-th neuron in l-th layer, $i = 1, 2, ..., I$ and $l = 1, 2, ..., L$. We denote the time range of the receiving phase as $\{t^{(l)}_{\text{recv}}, \ldots, t^{(l)}_{\text{emit}}\}$, and the emitting phase as $\{t^{(l)}_{\text{emit}}, \ldots, t^{(l)}_{\text{end}}\}$. With the initial membrane potential $V(t^{(l)}_{\text{recv}}) = 0$, the discrete membrane potential dynamics are given by:

$$V(t+1) - V(t) = \begin{cases} \frac{1}{\tau_i^{(l)}} \left(\sum_j w_{ij}^{(l)} \eta_{ij}^{(l)} \left(t - t_j^{(l-1)} \right) + C_i^{(l)} \right), t \in \{t^{(l)}_{\text{recv}}, \ldots, t^{(l)}_{\text{emit}} - 1\}, \\ \psi_i^{(l)} \left(t - t^{(l)}_{\text{emit}} \right), \ t \in \{t^{(l)}_{\text{emit}}, \ldots, t^{(l)}_{\text{end}} - 1\}. \end{cases} \tag{3}$$

Once the potential exceeds the threshold $\theta_i^{(l)}$, the neuron emits a spike. The relation between the spike time $t_i^{(l)}$ and the corresponding activation value $x_i^{(l)}$ of ANNs is:

$$x_i^{(l)}\,\tau_i^{(l)} = t_{\text{ref}}^{(l)} - t_i^{(l)}, \tag{4}$$

Therefore, the output range $[a_i^{(l)}, b_i^{(l)}]$, whose length is $d_i^{(l)} = b_i^{(l)} - a_i^{(l)}$, can be expressed as:

$$a_i^{(l)} = \frac{1}{\tau_i^{(l)}}\left(t_{\text{ref}}^{(l)} - t_{\text{end}}^{(l)}\right), \qquad b_i^{(l)} = \frac{1}{\tau_i^{(l)}}\left(t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)}\right). \tag{5}$$

Based on the discrete membrane potential dynamics and the resulting time-to-value mapping defined above, we establish the following theorem to provide a formal error bound for TTFS-based spiking LLMs to rigorously quantify how discrete-time operations impact the global conversion fidelity.

Theorem 4.1. ***Error analysis for TTFS-based conversion***. *Let Ω be the corresponding clock time constant, G_1 and G_2 be two bounded constants associated with the output transformation. For the i-th neuron in the l-th layer, let $\alpha_i^{(l)}$ denote the corresponding ANN output. We define $\rho = \max_{\{i,l\}}\left\{\left|\alpha_i^{(l)} - \frac{a_i^{(l)}+b_i^{(l)}}{2}\right|\right\}$, which measures the maximum deviation of ANN outputs from the center of the SNN representable range. Then, the conversion error of TTFS-based SNNs can be bounded as:*

$$\mathcal{E} \le LI \cdot \left(\max\left(\rho - \frac{T}{2\tau}, 0\right) + \frac{G_1 G_2 \Omega}{T}\right). \tag{6}$$

Remark 4.2. *Theorem 4.1 explicitly decomposes the practical conversion error in TTFS-based SNNs. The first term corresponds to the outlier-induced range error: output intervals estimated from limited calibration data may not cover activation outliers encountered during inference, causing clipping or distorted spike-time representations. The second term captures the low-latency discretization error: because continuous ANN values are encoded by discrete spike-time coordinates, the converted SNN can only approximate ANN outputs within the temporal resolution determined by the time window T. The dependence on L and I further indicates that these errors accumulate across layers and neurons.*

4.2 Equivalence between TTFS-based SNNs and Discrete ANNs

We theoretically establish the equivalence between TTFS-based SNNs and discrete ANNs. Given a TTFS-based neuron with fixed parameters, we can derive its corresponding ANN function:

Theorem 4.3. *For arbitrary fixed $\eta_{ij}^{(l)}$, $\psi_i^{(l)}$, $C_i^{(l)}$ and $\theta_i^{(l)}$ in SNNs with time window $T^{(l)}$, if we define $S(t) = \sum_{v=0}^{t-t_{emit}^{(l)}-1} \psi_i^{(l)}(v)$ with $t \in \left\{t_{emit}^{(l)}, \ldots, t_{end}^{(l)}\right\}$, then the corresponding activation value of discrete ANNs is given by:*

$$\begin{aligned} x_i^{(l)} &= F^{(l)}(\mathbf{W}^{(l)}; \mathbf{X}^{(l-1)}) \\ &= \frac{1}{\tau_i^{(l)}}\left(t_{ref}^{(l)} - S^{-1}\left(\theta_i^{(l)} + \Delta_i^{(l)} - T^{(l)}C_i^{(l)} - \frac{1}{\tau_i^{(l)}}\sum_{j=1}^{I}\sum_{t=t_j^{(l-1)}}^{T_{emit}^{(l)}-1} w_{ij}^{(l)}\,\eta_{ij}^{(l)}\left(\psi_{ij}(t)\right)\right)\right), \end{aligned} \tag{7}$$

where $\Delta_i^{(l)} \ge 0$ is the difference between $\theta_i^{(l)}$ and the membrane potential at the spike time.

Next, we determine the corresponding transform kernel functions and parameters in the TTFS-based neuron using the ANN function. For any given fixed ANN function, the TTFS-based neuron with the corresponding transform kernel functions and parameters can be identified:

Theorem 4.4. *Let f_{ij} be a function with input set of discrete points between $a_i^{(l-1)}$ and $b_i^{(l-1)}$, and h be a monotone increasing function with output set of discrete points between $a_i^{(l)}$ and $b_i^{(l)}$. We denote $u = t - t_j^{(l-1)}$ with $t \in \left\{t_{recv}^{(l)}, \ldots, t_{emit}^{(l)}\right\}$, and $v = t - t_{emit}^{(l)}$ with $t \in \left\{t_{emit}^{(l)}, \ldots, t_{end}^{(l)}\right\}$. To represent the corresponding activation value of discrete ANNs:*

$$x_i^{(l)} = F^{(l)}(\mathbf{W}^{(l)}; \mathbf{X}^{(l-1)}) = \text{clip}\left(h\left(\sum_j w_{ij}^{(l)} f_{ij}(x_j^{(l-1)})\right), a_i^{(l)}, b_i^{(l)}\right), \tag{8}$$

we need to configure the SNN as follows:

$$\eta_{ij}^{(l)}(u) = \begin{cases} \tau_i^{(l-1)} \left(f_{ij} \left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)} \right) - f_{ij} \left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)} \right) \right), & u \geq 0, \\ 0, & u < 0. \end{cases} \quad \theta_i^{(l)} = h^{-1}(b_i^{(l)}) + \Delta_i^{(l)},$$

$$\psi_i^{(l)}(v) = h^{-1}\left(b_i^{(l)} - \frac{v}{\tau_i^{(l)}} \right) - h^{-1}\left(b_i^{(l)} - \frac{v+1}{\tau_i^{(l)}} \right), C_i^{(l)} = \frac{\sum_j w_{ij}^{(l)} f_{ij}\left(a_i^{(l-1)}\right)}{d_i^{(l-1)}}. \tag{9}$$

Based on the above representational equivalence, we demonstrate the equivalence between the discrete TTFS-based neuron and the quantization function:

Corollary 4.5. *We define the processes of quantization and dequantization as follows:*

$$\hat{\mathbf{X}}_q^{(l)} = \lambda^{(l)} \cdot \text{clip}(\lfloor \frac{\mathbf{X}_q^{(l)}}{\lambda^{(l)}} \rfloor + z^{(l)}, 0, N) - \lambda^{(l)} \cdot z^{(l)}, \tag{10}$$

where $\lambda^{(l)} = \frac{\max(\mathbf{X}_q^{(l)}) - \min(\mathbf{X}_q^{(l)})}{N}$ *and* $z^{(l)} = -\lfloor \frac{\min(\mathbf{X}_q^{(l)})}{\lambda^{(l)}} \rfloor$ *are scale and zero point values, respectively;* $N = 2^n - 1$ *denotes the quantization level and* n *denotes the quantization bits;* $\hat{\mathbf{X}}_q^{(l)}$ *and* $\mathbf{X}_q^{(l)}$ *are the quantized and original tensor, respectively.*

For a TTFS-based SNN defined in (3), $\mathcal{H}$ *is the Heaviside step function, if we set the* $\eta_{ij}^{(l)}$, $\psi_i^{(l)}$, $C_i^{(l)}$ *and* $\theta_i^{(l)}$ *as follow:*

$$\eta_{ij}^{(l)}(u) = \mathcal{H}\left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)} \right), \quad \psi_i^{(l)}(v) = \frac{1}{\tau_i^{(l)}}, \quad C_i^{(l)} = \sum_j \frac{a_i^{(l-1)}}{d_i^{(l-1)}} w_{ij}, \quad \theta_i^{(l)} = b_i^{(l)}, \tag{11}$$

and let $t_{emit}^{(l)} = 0$, $t_{end}^{(l)} = N$, $\tau_i^{(l)} = \frac{1}{\lambda^{(l)}}$, $t_{end}^{(l)} - t_{ref}^{(l)} = z^{(l)}$, *and* $\mathbf{X}_q^{(l)} = \sum_{j=1}^{I} w_{ij}^{(l)} x_j^{(l-1)}$. *The output of the spiking neuron and quantization function are equivalent, i.e.,* $x_i^{(l)} = \hat{\mathbf{X}}_q^{(l)}$.

Remark 4.6. *The connection between TTFS-based SNNs and quantized ANNs is fundamentally rooted in parameter mapping. Our framework treats the quantized ANN as a robust baseline, directly translating its quantization parameters into the SNN's firing thresholds and time constants. The primary distinction, however, lies in the domains of representation and computation:* ***while quantized ANNs reduce numerical bit-width to compress spatial data, TTFS-based SNNs transpose these discrete values into the temporal domain as precise spike timings****.*

4.3 Quantization-Consistent ANN-to-SNN Conversion

Building upon the equivalence established in the previous section, we further propose a novel quantization-consistent ANN-to-SNN conversion framework. Algorithm 1 outlines the proposed QC-A2S pipeline for converting a vanilla LLM into a low-latency, high-accuracy temporal-coding spiking LLM. Our approach involves discretization-aware TTFS neurons by systematically processing both linear and nonlinear components:

Linear computations. We employ PTQ, specifically utilizing PrefixQuant [9] to establish a robust, low-bit baseline. Unlike traditional methods, PrefixQuant effectively suppresses the impact of activation outliers pervasive in LLMs, ensuring that these extreme values do not distort the subsequent temporal mapping. We explicitly build a mapping relationship between this outlier-aware quantization function and our TTFS formulation: for each linear layer and neuron, the retrieved quantization parameters are directly translated into the spiking neuron's time constant, emission time, and ending time (Corollary 4.5).

Nonlinear operators. For nonlinear operations, which constitute only a tiny fraction of the overall computation, their continuous activations are systematically replaced with discrete spiking equivalents as derived in Appendix G. For instance, SiLU and GELU are expressed through combinations of thresholding, weighted summation, and the nonlinearity of single neurons Corollarys (G.1 and G.2); Softmax is implemented via a log-sum-exp neuron, allowing the computation of normalized probabilities in a single spiking neuron Corollary (G.3); and RMSNorm is decomposed into spike-based operations for mean-square computation and scaling Corollary (G.4).

Algorithm 1 Implementation of QC-A2S

Input: Vanilla LLM
Output: Temporal-coding Spiking LLM
Obtaining well-quantized LLM by PTQ
Mapping Quantized LLMs to Temporal-coding Spiking LLMs:
for layer $l = 1, 2, ..., L$ **do**
    for neuron $i = 1, 2, ..., I$ **do**
        Get quantization parameters N, $\lambda_i^{(l)}$ and $z^{(l)}$
        Construct TTFS-based spiking neural neurons:
        Calculate $t_{\text{emit}}^{(l)}, t_{\text{ref}}^{(l)}, t_{\text{end}}^{(l)}, \tau_i^{(l)}$ by Corollary 4.5.
    end for
    if l is SiLU, GELU, Softmax or RMSNorm **then**
        Replace the original nonlinear activation with spiking counterpart in the Appendix G.
    end if
end for

5 Experiment

5.1 Implement Details

Datasets and Models. For accuracy evaluation, five representative reasoning datasets are adopted, namely PIQA [3], HellaSwag (Hella.) [64], WinoGrande (Wino.) [43], ARC-Easy, and ARC-Challenge [12]. For perplexity evaluation, we additionally use five widely adopted language modeling datasets, including C4 [40], The Pile [17], WikiText-2 [33], and RedPajama (RedP.) [51]. We consider a range of representative models, including LLaMA-2-7B, LLaMA-2-13B, LLaMA-2-70B, LLaMA-3-8B, LLaMA-3-70B, Qwen2-7B and Mistral-7B v0.3. Our code is available at https://github.com/fangjj2000/TTFS-based-LLM.

Baselines. We compare our approach against several representative baselines that adapt LLMs through either quantization or A2S method: PrefixQuant [9], SpikeLLM [58], TTFSFormer [67], and DuQuant [28]. Details regarding the baselines are provided in the Appendix D. We define the correspondence between the time window of QC-A2S and the quantization bit-width as: $T = 2^n$.

5.2 Main Results

Table 2 reports the accuracy and PPL metrics of all methods on the LLaMA models. The results indicate that: (i) QC-A2S achieves performance comparable to quantized LLMs across all LLaMA models, providing further empirical evidence for the equivalence between TTFS-based SNNs and quantized ANNs; (ii) our method substantially outperforms TTFSFormer under low-latency settings, and surpasses the state-of-the-art spiking LLM, further validating the effectiveness of the proposed approach. Table 3 reports the performance on other models (Qwen2-7B and Mistral-7B v0.3), and Table 12 reports the performance comparison between our method and DuQuant.

In Table 4, we conduct a detailed study of how latency influences the performance of temporally coded spiking LLMs using the LLaMA-2-7B and LLaMA-2-13B models. The results reveal a clear trend: increasing latency consistently improves accuracy across all evaluated benchmarks. This indicates that longer time windows allow TTFS-based SNNs to better approximate the activations of ANNs, thereby reducing discretization-induced errors and enhancing representational fidelity. Moreover, deeper models do not necessarily yield monotonically higher accuracy.

5.3 Energy Analysis for QC-A2S

We estimate the inference energy consumption using a theoretical energy model adopted in previous SNN literature [71, 6, 7]. The overall energy cost is formulated as

$$E_{\text{total}} = E_{\text{MAC}} \cdot \text{Count}_{\text{MAC}} + E_{\text{AC}} \cdot \text{Count}_{\text{AC}}, \tag{12}$$

where E_{MAC} and E_{AC} are the unit energy costs of MAC and AC operations, and $\text{Count}_{\text{MAC}}$ and Count_{AC} correspond to their operation counts during inference. Consistent with the 45 nm CMOS

Table 2: Comprehensive comparison of Accuracy (↑) and Perplexity (↓) across various LLaMA models. The entries highlighted in gray correspond to our QC-A2S method.

			Accuracy ↑						Perplexity↓				
Model	**Method**	**Prec.**	**Wino.**	**Hella.**	**ArcC**	**ArcE**	**PiQA**	**Avg.**	**C4**	**Pile**	**Wiki2**	**RedP.**	**Avg.**
2-7B	Baseline	FP16	69.22	76.00	46.33	74.62	79.11	69.06	6.97	4.63	5.47	5.61	5.67
	TTFSFormer	T=32	51.07	26.32	28.84	25.84	48.91	36.20	>100	>100	>100	>100	>100
	TTFSFormer	T=64	48.86	26.40	28.67	26.09	49.24	35.85	>100	>100	>100	>100	>100
	SpikeLLM	W8A6	65.51	73.61	42.49	70.16	75.41	65.44	7.89	5.14	6.43	6.21	6.42
	PrefixQ	W4A4	66.77	73.62	42.83	70.88	76.93	66.21	7.72	5.00	6.12	6.28	6.28
	PrefixQ	W5A5	69.06	75.53	43.94	73.06	77.86	67.89	7.20	4.74	5.67	5.82	5.86
	PrefixQ	W6A6	70.48	76.22	45.48	73.86	78.35	68.88	7.06	4.67	5.54	5.70	5.74
	QC-A2S	T=16	66.77	73.08	41.21	70.45	77.64	65.83	7.73	5.00	6.12	6.29	6.29
	QC-A2S	T=32	69.38	75.49	44.28	72.85	78.02	68.00	7.20	4.74	5.67	5.82	5.86
	QC-A2S	T=64	69.38	76.23	45.99	73.57	78.13	68.66	7.07	4.67	5.54	5.70	5.75
3-8B	Baseline	FP16	72.69	79.19	53.41	77.69	80.79	72.75	8.88	5.52	6.14	7.44	7.01
	TTFSFormer	T=32	49.88	26.40	26.54	24.71	51.58	35.82	>100	>100	>100	>100	>100
	TTFSFormer	T=64	50.59	26.19	26.11	24.49	50.16	35.51	>100	>100	>100	>100	>100
	SpikeLLM	W8A6	58.25	59.28	32.34	53.37	68.66	54.38	>100	>100	>100	>100	>100
	PrefixQ	W4A4	71.03	74.51	48.72	75.88	77.80	69.59	11.22	6.62	7.82	9.69	8.84
	PrefixQ	W5A5	71.74	77.59	53.41	78.45	79.11	72.06	9.75	5.94	6.79	8.30	7.70
	PrefixQ	W6A6	72.77	78.52	53.07	78.58	79.22	72.43	9.29	5.74	6.47	7.85	7.34
	QC-A2S	T=16	70.09	74.36	48.29	75.97	77.86	69.31	11.23	6.61	7.82	9.69	8.84
	QC-A2S	T=32	70.01	77.64	53.50	78.07	79.71	71.79	9.75	5.94	6.79	8.29	7.69
	QC-A2S	T=64	72.06	78.26	54.95	77.90	79.43	72.52	9.28	5.74	6.47	7.86	7.34
2-13B	Baseline	FP16	72.38	79.38	49.06	77.53	80.52	71.77	6.47	4.34	4.88	5.19	5.22
	TTFSFormer	T=32	48.15	26.40	28.24	25.00	49.02	35.36	>100	>100	>100	>100	>100
	TTFSFormer	T=64	48.54	26.15	30.03	26.30	51.03	36.41	>100	>100	>100	>100	>100
	SpikeLLM	W8A6	68.03	76.76	44.88	73.32	77.48	68.09	7.16	4.74	5.81	5.53	5.81
	PrefixQ	W4A4	69.69	75.76	47.53	73.78	78.62	69.08	8.46	5.33	6.38	7.05	6.81
	PrefixQ	W5A5	72.38	78.30	49.91	76.26	79.76	71.32	10.89	6.99	7.28	8.66	8.46
	PrefixQ	W6A6	72.53	79.07	49.23	75.76	79.71	71.26	6.62	4.44	5.03	5.32	5.35
	QC-A2S	T=16	70.32	75.62	46.42	73.95	77.97	68.86	8.36	5.29	6.36	7.02	6.76
	QC-A2S	T=32	71.35	78.24	49.23	75.00	79.87	70.74	11.09	7.13	7.39	8.82	8.61
	QC-A2S	T=64	72.53	79.07	49.23	75.76	79.71	71.26	6.63	4.45	5.04	5.33	5.36
2-70B	Baseline	FP16	77.98	83.82	57.42	80.98	82.70	76.58	5.52	3.74	3.32	4.31	4.22
	TTFSFormer	T=32	48.15	26.40	28.24	25.00	49.02	35.36	>100	>100	>100	>100	>100
	SpikeLLM	W8A6	75.06	81.42	52.82	75.29	80.58	73.03	6.28	4.18	4.97	4.11	4.89
	PrefixQ	W4A4	75.45	74.32	46.08	72.47	77.53	69.17	7.31	4.76	5.02	5.85	5.74
	QC-A2S	T=16	73.95	79.22	51.96	76.94	80.09	72.43	7.31	4.76	5.03	5.85	5.74
3-70B	Baseline	FP16	80.35	84.88	64.33	85.86	84.55	79.99	6.73	4.38	2.85	5.32	4.82
	TTFSFormer	T=32	49.01	26.32	26.88	25.25	50.05	35.50	>100	>100	>100	>100	>100
	SpikeLLM	W8A6	52.09	28.40	26.45	31.36	53.59	38.38	>100	>100	>100	>100	>100
	PrefixQ	W4A4	77.98	82.84	59.98	81.73	83.35	77.18	8.18	5.16	4.66	6.92	6.23
	QC-A2S	T=16	77.27	82.67	59.04	82.15	83.13	76.85	8.16	5.15	4.64	6.90	6.21

measurements in [22], we use $E_{\mathrm{MAC}} \approx 4.6$ pJ and $E_{\mathrm{AC}} \approx 0.9$ pJ. For operation-counting procedure, linear transformations are counted according to the equivalent firing rate of the SNN, thereby capturing the effective spike-based computational workload. For nonlinear operators, their costs are estimated following the operation-count formulation in Table 13 in Appendix D [67]. Table 5 reports the calculation counts and energy consumption of QC-A2S compared with the ANN baseline.

6 Conclusion

In this paper, we present QC-A2S, a quantization-consistent ANN-to-SNN conversion framework for building low-latency temporal-coding spiking LLMs. We first identify the limitations of existing TTFS-based conversion methods, whose continuous-time assumptions require prohibitively large time windows and thus hinder practical deployment in LLMs. To address this issue, we reformulate TTFS-based conversion as a discrete error analysis problem and derive a formal error bound that characterizes the effects of finite temporal resolution and bounded representable ranges. Based on

Table 3: Comparison of Accuracy (↑) and Perplexity (↓) on Qwen2-7B and Mistral-7B v0.3.

			Accuracy↑					Perplexity↓			
Model	**Method**	**Prec.**	**Wino.**	**Hella.**	**ArcC**	**PiQA**	**Avg.**	**C4**	**Pile**	**Wiki2**	**Avg.**
Qwen-2-7B	ANN	FP16	72.45	78.78	49.91	81.23	71.42	9.90	5.53	7.14	7.52
	PrefixQ	W4A4	68.19	74.67	49.66	79.00	69.51	11.44	6.24	8.26	8.65
	QC-A2S	T=16	68.59	74.56	48.12	78.13	68.91	11.43	6.24	8.26	8.64
	PrefixQ	W5A5	71.51	77.60	48.46	79.87	70.24	10.34	5.73	7.44	7.84
	QC-A2S	T=32	71.51	77.51	49.40	80.20	70.31	10.34	5.74	7.43	7.84
	PrefixQ	W6A6	71.51	78.68	51.19	80.36	71.45	10.05	5.60	7.23	7.63
	QC-A2S	T=64	71.82	78.41	51.45	81.18	71.79	10.05	5.60	7.23	7.63
Mistral-7B v0.3	ANN	FP16	73.88	80.44	52.30	82.26	73.42	7.84	4.46	5.32	5.87
	PrefixQ	W4A4	71.90	77.55	49.91	79.54	71.05	8.58	4.84	5.93	6.45
	QC-A2S	T=16	71.59	77.38	50.77	80.36	71.26	8.58	4.84	5.93	6.45
	PrefixQ	W5A5	72.53	79.74	52.90	80.85	73.05	8.12	4.61	5.54	6.09
	QC-A2S	T=32	73.09	79.63	52.82	81.23	73.11	8.12	4.61	5.54	6.09
	PrefixQ	W6A6	72.14	80.41	53.24	82.37	73.58	7.99	4.55	5.45	6.00
	QC-A2S	T=64	73.56	80.32	53.67	81.66	73.71	7.99	4.55	5.46	6.00

Table 4: Results under different latency configurations on LLaMA models.

Model	**Latency**	**Wino.**	**Hella.**	**ArcC**	**ArcE**	**PIQA**	**Avg.**
2-7B	T=16	66.77	73.08	41.21	70.45	77.64	65.83
	T=32	69.38	75.49	44.28	72.85	78.02	68.00
	T=64	69.38	76.23	45.99	73.57	78.13	68.66
	T=256	70.56	76.69	46.42	73.57	78.40	69.13
	T=1024	70.32	76.62	46.16	73.78	78.18	69.01
2-13B	T=16	70.32	75.62	46.42	73.95	77.97	68.86
	T=32	71.35	78.24	49.23	75.00	79.87	70.74
	T=64	72.53	79.07	49.23	75.76	79.71	71.26
	T=256	72.85	79.71	49.32	76.64	80.36	71.78
	T=1024	72.38	79.75	48.72	76.43	80.52	71.56

Table 5: Calculation count and energy cost of ANN (FP16) and our spiking LLM ($T = 16$).

Model	**Method**	**Avg. ACC**	**Avg. PPL**	**Calculation Count**	**Energy Cost** (J)
2-7B	ANN	69.06	5.67	2354.512G MACs + 1.399G ACs	10.83
	QC-A2S	65.83	6.29	67.11G MACs + 5587.09G ACs	5.34
3-8B	ANN	72.75	7.01	3226.94G MACs + 1.78G	14.85
	QC-A2S	69.31	8.84	268.97G MACs + 6258.85G ACs	6.87
2-13B	ANN	71.77	5.22	4543.80G MACs + 2.19G ACs	20.90
	QC-A2S	68.86	6.76	83.89G MACs, 9599.97G ACs	9.04

the established equivalence between discrete TTFS-based SNNs and quantized ANNs, QC-A2S directly maps quantization parameters to TTFS neuron parameters, enabling discretization-compatible temporal encoding. Extensive experiments on representative LLMs demonstrate that QC-A2S achieves competitive accuracy and perplexity under substantially reduced latency, while also lowering the estimated energy consumption compared with ANN baselines.

Acknowledgments

Dr. Yi Chang was supported by the National Key R&D Program of China (Grant No. 2023YFF0905400), the National Natural Science Foundation of China (Grant No. U2341229), the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM903), and the New Cornerstone Science Foundation through the XPLORER PRIZE.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] James A. Boyle, Mark Plagge, Suma G. Cardwell, Frances S. Chance, and Andreas Gerstlauer. Performance and energy simulation of spiking neuromorphic architectures for fast exploration. In *Proceedings of the International Conference on Neuromorphic Systems (ICONS)*. ACM, 2023.

[5] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems. *arXiv preprint arXiv:2403.02419*, 2024.

[6] Long Chen, Xiaotian Song, Andy Song, BaDong Chen, Jiancheng Lv, and Yanan Sun. Fas: Fast ann-snn conversion for spiking large language models. *arXiv preprint arXiv:2502.04405*, 2025.

[7] Long Chen, Xiaotian Song, and Yanan Sun. Las: Loss-less ann-snn conversion for fully spike-driven large language models. *arXiv preprint arXiv:2505.09659*, 2025.

[8] Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. Prefixquant: Static quantization beats dynamic through prefixed outliers in llms. *arXiv preprint arXiv:2410.05265*, 2024.

[9] Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. Prefixquant: Eliminating outliers by prefixed tokens for large language models quantization, 2025.

[10] Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*, 2024.

[11] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 27–39. IEEE, 2016.

[12] 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*, abs/1803.05457, 2018.

[13] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[14] Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *arXiv preprint arXiv:2402.10631*, 2024.

[15] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[16] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan-Adrian Alistarh. Optq: Accurate post-training quantization for generative pre-trained transformers. In *11th International Conference on Learning Representations*, 2023.

[17] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2021.

[18] Julian Göltz, Laura Kriener, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Andreas F. Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A. Petrovici. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature Machine Intelligence*, 3:823–835, 2021.

[19] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420–428, 2006.

[20] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11–21, 2023.

[21] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[22] Mark Horowitz. Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[23] Shubham Jain, Sumeet Kumar Gupta, and Anand Raghunathan. TiM-DNN: Ternary in-memory accelerator for deep neural networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(7):1567–1577, 2020.

[24] 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*, 2024.

[25] Edgar Lemaire, Loïc Cordone, Andrea Castagnetti, Pierre-Emmanuel Novac, Jonathan Courtois, and Benoît Miramond. An analytical estimation of spiking neural networks energy efficiency. In *International Conference on Neural Information Processing (ICONIP)*, pages 574–587, 2023.

[26] Dongwoo Lew, Kyungchul Lee, and Jongsun Park. A time-to-first-spike coding and conversion aware training for energy-efficient deep spiking neural network processor design. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 265–270, 2022.

[27] Fanfan Li, Dingwei Li, Chuanqing Wang, Guolei Liu, Rui Wang, Huihui Ren, Yingjie Tang, Yan Wang, Yitong Chen, Kun Liang, Qi Huang, Mohamad Sawan, Min Qiu, Hong Wang, and Bowen Zhu. An artificial visual neuron with multiplexed rate and time-to-first-spike coding. *Nature Communications*, 15:3689, 2024.

[28] Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800, 2024.

[29] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[30] Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024.

[31] Changze Lv, Tianlong Li, Jianhan Xu, Chenxi Gu, Zixuan Ling, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Spikebert: A language spikformer trained with two-stage knowledge distillation from bert. 2023.

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

[33] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[34] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[35] Marcelo A Montemurro, Malte J Rasch, Yusuke Murayama, Nikos K Logothetis, and Stefano Panzeri. Phase-of-firing coding of natural visual stimuli in primary visual cortex. *Current biology*, 18(5):375–380, 2008.

[36] Bhaskar Mukhoty, Velibor Bojkovic, William de Vazelhes, Xiaohan Zhao, Giulia De Masi, Huan Xiong, and Bin Gu. Direct training of snn using local zeroth order method. *Advances in Neural Information Processing Systems*, 36:18994–19014, 2023.

[37] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. Timeloop: A systematic approach to DNN accelerator evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 304–315. IEEE, 2019.

[38] Seongsik Park, Seijoon Kim, Hyeokjun Choe, and Sungroh Yoon. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[39] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE design automation conference (DAC)*, pages 1–6. IEEE, 2020.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[41] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[42] Bodo Rueckauer and Shih-Chii Liu. Conversion of analog to spiking neural networks using sparse temporal coding. In *2018 IEEE international symposium on circuits and systems (ISCAS)*, pages 1–5. IEEE, 2018.

[43] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[44] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 14–26. IEEE, 2016.

[45] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. 2024.

[46] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Joon Kyung Kim, Vikas Chandra, and Hadi Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018.

[47] Shrihari Sridharan, Jacob R. Stevens, Kaushik Roy, and Anand Raghunathan. X-Former: In-memory acceleration of transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 31(8):1223–1233, 2023.

[48] Ana Stanojevic, Stanisław Woźniak, Guillaume Bellec, Giovanni Cherubini, Angeliki Pantazi, and Wulfram Gerstner. An exact mapping from relu networks to spiking neural networks. *Neural Networks*, 168:74–88, 2023.

[49] Ana Stanojevic, Stanisław Woźniak, Guillaume Bellec, Giovanni Cherubini, Angeliki Pantazi, and Wulfram Gerstner. High-performance deep spiking neural networks with 0.3 spikes per neuron. *Nature Communications*, 15(1):6793, 2024.

[50] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural networks*, 14(6-7):715–725, 2001.

[51] Together Computer. Redpajama: Reproducible pretraining data. `https://www.together.xyz/blog/redpajama`, 2023.

[52] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[53] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[55] Weier Wan, Rajkumar Kubendran, Clemens Schaefer, Sukru Burc Eryilmaz, Wenqiang Zhang, Dabin Wu, Stephen Deiss, Priyanka Raina, He Qian, Bin Gao, Siddharth Joshi, Huaqiang Wu, H.-S. Philip Wong, and Gert Cauwenberghs. A compute-in-memory chip based on resistive random-access memory. *Nature*, 608:504–512, 2022.

[56] Yannan Nellie Wu, Joel S. Emer, and Vivienne Sze. Accelergy: An architecture-level energy estimation methodology for accelerator designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.

[57] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[58] 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.

[59] Xingrun Xing, Zheng Zhang, Ziyi Ni, Shitao Xiao, Yiming Ju, Siqi Fan, Yequan Wang, Jiajun Zhang, and Guoqi Li. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms. In *International Conference on Machine Learning*, pages 54698–54714. PMLR, 2024.

[60] Zhanglu Yan, Jiayi Mao, Qianhui Liu, Fanfan Li, Tao Luo, Gang Pan, Bowen Zhu, and Weng-Fai Wong. Otters: An energy-efficient spiking transformer via optical time-to-first-spike encoding. In *International Conference on Learning Representations*, 2026.

[61] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36:64043–64058, 2023.

[62] 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 *Proceedings of the 41st International Conference on Machine Learning*, pages 57367–57383, 2024.

[63] Xinzhe Yuan, Xiang Peng, Bin Gu, and Huan Xiong. Plug-and-play spiking operators: Breaking the nonlinearity bottleneck in spiking transformers. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 151938–151954. PMLR, 2026.

[64] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[65] Lei Zhang, Shengyuan Zhou, Tian Zhi, Zidong Du, and Yunji Chen. Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1319–1326, 2019.

[66] 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.

[67] Lusen Zhao, Zihan Huang, Jianhao Ding, and Zhaofei Yu. TTFSFormer: A TTFS-based lossless conversion of spiking transformer. In *Forty-second International Conference on Machine Learning*, 2025.

[68] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

[69] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, 2023.

[70] Chenlin Zhou, Han Zhang, Liutao Yu, Yumin Ye, Zhaokun Zhou, Liwei Huang, Zhengyu Ma, Xiaopeng Fan, Huihui Zhou, and Yonghong Tian. Direct training high-performance deep spiking neural networks: a review of theories and methods. *Frontiers in Neuroscience*, 18:1383844, 2024.

[71] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

[72] Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

A Notations

We summarize the notations used in this paper in Table 6.

Table 6: Symbol Definitions

Symbol	Definition	Symbol	Definition
l	Layer index	$\mathbf{X}_q$	Inputs of QANN
i, j	Neuron index	$\hat{\mathbf{X}}_q$	Output of QANN
W	Weight matrix	$a_i^{(l)}$	Output lower bound of TTFS-based neuron
t_{recv}	Receiving time step	$b_i^{(l)}$	Output upper bound of TTFS-based neuron
t_{emit}	Emitting time step	I	The number of neurons in each layer
t_{end}	End time-step	θ	Threshold
$\mathcal{H}$	Heaviside function	C	Bias term in TTFS-based neuron
t	Time step index	τ	Time constant in TTFS-based neuron
η	Input transform kernel	N	Quantization level
ψ	Output transform kernel	n	Quantization bits

B Discussion: From Quantized LLMs to Event-Driven Execution

A natural question is why a quantized LLM should be further converted into a spiking model rather than directly deployed as a low-precision ANN. The practical motivation of QC-A2S is not to introduce an alternative numerical representation for its own sake, but to transform a quantized LLM into a functionally equivalent form that can expose *event-driven execution* to suitable hardware. In this sense, quantization and TTFS conversion play complementary roles: quantization constructs a compact discrete numerical model, whereas QC-A2S translates the same discrete activation values into temporal events that can be directly mapped to event-driven hardware primitives.

From value-domain computation to event-triggered execution. Conventional quantization reduces numerical precision, model storage, and arithmetic cost, but a quantized ANN still represents activations as explicit multi-bit values and executes them through conventional value-domain tensor operations. In contrast, QC-A2S represents each quantized activation using the timing of at most one binary spike. The corresponding temporal kernels, thresholding dynamics, and neuron states preserve the original layer-wise computation while changing how it is executed.

On event-driven hardware, the arrival of a spike can trigger the accumulation of the associated synaptic weight and its communication to downstream neurons. Conversely, when no event is emitted, the corresponding activation-dependent computation and communication can be skipped. This execution model enables hardware behaviors that are not directly exposed by conventional quantization alone, including spike-triggered accumulation, event-gated computation and communication, single-spike temporal representation, and asynchronous propagation between upstream emission and downstream accumulation. Prior TTFS and neuromorphic studies have demonstrated that such sparse temporal representations can be efficiently executed on dedicated hardware [26, 18, 39, 49, 27, 41].

Hardware evidence for TTFS execution. Existing hardware-oriented studies provide concrete evidence that TTFS representations can offer practical benefits beyond simply executing the corresponding ANN or QANN. For example, Otters [60] compares a TTFS Transformer with a structurally matched quantized BERT and reports lower estimated energy for an attention block under a hardware-aware model that accounts for computation, memory access, data movement, and static energy. Lew et al. [26] further implement a TTFS-oriented processor in a 28-nm CMOS process and demonstrate energy-efficient inference on CIFAR-10, CIFAR-100, and Tiny-ImageNet. Together with first-spike neuromorphic demonstrations [18], these results indicate that spike-time conversion is not merely an alternative numerical encoding: it enables discrete activations to be executed through sparse, event-driven hardware primitives.

Complementarity with memory-centric architectures. For large language models, data movement can dominate arithmetic energy. This limitation has motivated near-memory and compute-in-

memory architectures that colocate model weights with computation and thereby reduce repeated off-chip weight transfers. For example, X-Former [47] accelerates Transformer inference through hybrid in-memory processing, while RRAM-based compute-in-memory systems directly perform matrix operations close to stored weights [55]. Similarly, TiM-DNN [23] demonstrates substantial reductions in system-level energy through in-memory ternary computation. Such techniques address the movement of relatively stationary weights, but they do not eliminate activation-dependent computation and communication. After repeated weight movement is reduced, TTFS provides a complementary opportunity: an arriving spike selectively activates the corresponding stored weights and triggers accumulation, whereas a silent activation requires no corresponding synaptic event. Consequently, temporal sparsity can potentially reduce not only arithmetic operations, but also wordline/bitline activity, peripheral readout, event routing, neuron-state updates, and other activation-dependent hardware activity. Therefore, memory-centric computing and event-driven TTFS execution address different sources of inefficiency and can be naturally combined rather than viewed as competing alternatives.

Nonlinear operators and practical scope. A complete spiking Transformer must also account for nonlinear operators such as Softmax, normalization, and SiLU, which are not naturally implemented by standard integrate-and-fire dynamics. Recent work has begun to address this limitation through spike-compatible implementations of nonlinear Transformer operators [63]. In our current arithmetic-energy analysis, the contribution of Softmax, RMSNorm, and SiLU is small relative to the dominant linear Transformer operations; nevertheless, their implementation remains an important consideration for fully event-driven deployment. Overall, the motivation for QC-A2S is therefore not that spiking representations universally outperform quantized models on conventional hardware. Rather, QC-A2S serves as an algorithm–architecture bridge: PrefixQuant first constructs an accurate low-bit model, and QC-A2S subsequently transforms its discrete activations into an equivalent TTFS representation that exposes conditional computation, spike-based communication, and asynchronous execution to event-driven and memory-centric hardware.

C Architecture-Level Energy and Latency Analysis

To evaluate the practical hardware implications of QC-A2S beyond arithmetic-operation counts, we conduct a matched architecture-level analysis against a W4A4 quantized-ANN (QANN) backend. The comparison considers two architecture-specific backends under matched resource assumptions, rather than measurements on the same fabricated chip. We adopt an action-count-based energy model following established accelerator-level estimation methodologies [22, 56, 37]. For the spiking backend, we follow the event-driven synapse/soma abstraction of SANA-FE [4], while explicitly accounting for memory, communication, and membrane-state traffic, as motivated by prior analytical SNN studies [25]. The TTFS execution model is consistent with prior first-spike coding systems [39, 18].

Hardware-mapping setup. Table 7 summarizes the architectural mapping of the two backends. The W4A4 QANN backend employs conventional low-bit MAC execution, whereas QC-A2S uses event-driven synaptic accumulation with $T = 16$ first-spike temporal codes. The latter retains membrane states locally and uses a receiving–emitting pipeline to overlap inter-layer communication and computation.

Table 7: Hardware-mapping comparison between the W4A4 QANN and QC-A2S TTFS backends.

Aspect	W4A4 QANN backend	QC-A2S TTFS backend
Numerical representation	4-bit weights and 4-bit activations	4-bit weights and $T = 16$ first-spike temporal codes
Main execution unit	Low-bit multiply-accumulate array	Event-driven synaptic-addition units
Activation communication	4-bit activation tensors	16-bit event packets
State	Conventional activation buffers	4-bit membrane state retained locally
Scheduling	Dense tensor/GEMM scheduling	Event queue and receiving–emitting pipeline
Completion condition	Completion of scheduled tensor operations	A neuron becomes final after its first spike
Hardware precedent	BitFusion-style low-bit composable arithmetic [46]	SANA-FE-style synapse/soma separation [4]
Critical assumption	Low-bit MAC throughput under matched area/resources	Membrane state remains local during synaptic accumulation

The matched hardware parameters are summarized in Table 8. We use a 45-nm reference process, consistent with commonly used architecture-level energy models [22]. The $4\times$ AC-throughput

operating point is an equal-resource analytical point motivated by the throughput scaling of bit-level composable arithmetic [46].

Table 8: Matched hardware parameters used in the architecture-level analysis.

Parameter	Matched value
Process	45 nm
Clock frequency	500 MHz
Weight precision	4 bit
QANN activation precision	4 bit
QC-A2S state precision	4 bit
Event-packet width	16 bit
DRAM bandwidth	256 GB/s
Aggregate SRAM bandwidth	16 TB/s
Aggregate NoC bandwidth	4 TB/s
Average NoC distance	2 hops
QANN MAC throughput	32.768 TMAC/s
QC-A2S AC throughput	131.072 TAC/s
Base static power	1 W/backend
Control energy	2% of dynamic energy
Scheduling overhead	5% of critical-path latency
Residency amortization	$M = 1000$ workloads

Energy model and weight residency. Following the action-count methodology [56], the total energy is modeled as

$$E_{\text{total}} = \sum_a N_a e_a + E_{\text{control}} + E_{\text{static,infer}} + \overline{E}_{\text{static,init}}, \tag{13}$$

where N_a and e_a denote the action count and corresponding unit energy, respectively. The dynamic component includes computation, weight transfer, activation/event communication, local storage, and state-update costs:

$$\begin{aligned} E_{\text{dynamic}} = {} & E_{\text{MAC}} + E_{\text{AC}} + E_{\text{DRAM,w}} + E_{\text{local-write,w}} \\ & + E_{\text{NoC,w}} + E_{\text{SRAM/queue}} + E_{\text{NoC,act/event}} + E_{\text{state}}. \end{aligned} \tag{14}$$

We further set

$$E_{\text{control}} = 0.02 E_{\text{dynamic}}, \qquad E_{\text{static,infer}} = P_{\text{static}} t_{\text{infer}}. \tag{15}$$

To account for different memory organizations, we define $r \in [0, 1]$ as the fraction of weights retained locally across workloads. Specifically, $r = 0$ corresponds to no cross-workload residency, $r = 0.5$ retains half of the weights, and $r = 1$ represents full residency after initialization. If resident weights are loaded once and reused across M workloads, the average weight-fetch factor is

$$g(r, M) = 1 - r + \frac{r}{M}. \tag{16}$$

With $M = 1000$, this gives $g(0, 1000) = 1$, $g(0.5, 1000) = 0.5005$, and $g(1, 1000) = 0.001$. Letting B_w denote the total number of weight bits, the corresponding average weight-transfer costs can be written as

$$\overline{E}_{\text{DRAM,w}} = g(r, M) B_w e_{\text{DRAM/bit}}, \tag{17}$$

$$\overline{E}_{\text{local-write,w}} = g(r, M) B_w e_{\text{SRAM/bit}}, \tag{18}$$

and

$$\overline{E}_{\text{NoC,w}} = g(r, M) B_w e_{\text{NoC/bit/hop}}. \tag{19}$$

Table 9 reports the resulting system-level energy sensitivity across LLaMA-2 and LLaMA-3 model scales [52, 15]. Without cross-workload residency ($r = 0$), both backends repeatedly incur substantial off-chip weight-transfer costs, partially masking the compute-side advantage of QC-A2S. Nevertheless, QC-A2S already reduces total system energy by 10.31%–15.96%. With partial residency

Table 9: System-energy sensitivity to matched weight residency.

Model	$r = 0$			$r = 0.5$			$r = 1$		
	QANN	QC-A2S	Reduction	QANN	QC-A2S	Reduction	QANN	QC-A2S	Reduction
LLaMA-2-7B	808.057	724.763	**10.31%**	526.307	443.013	**15.83%**	244.557	161.263	**34.06%**
LLaMA-3-8B	1004.926	875.727	**12.86%**	669.251	540.052	**19.30%**	333.575	204.376	**38.73%**
LLaMA-2-13B	1557.354	1360.234	**12.66%**	1013.919	816.799	**19.44%**	470.484	273.364	**41.90%**
LLaMA-2-70B	8259.307	7160.104	**13.31%**	5375.899	4276.696	**20.45%**	2492.490	1393.287	**44.10%**
LLaMA-3-70B	8939.462	7581.898	**15.19%**	5990.128	4632.564	**22.66%**	3040.794	1683.230	**44.65%**
LLaMA-3.1-405B	51457.459	43242.875	**15.96%**	34491.700	26277.116	**23.82%**	17525.942	9311.358	**46.87%**

($r = 0.5$), the reduction increases to 15.83%–23.82%, and under the full-residency steady-state bound ($r = 1$), it reaches 34.06%–46.87%.

The residency analysis is particularly relevant to accelerator designs with large on-package memories, 3D-stacked memory, near-memory processing, and processing-/compute-in-memory architectures, which reduce repeated weight movement. ISAAC [44] and PRIME [11], for example, demonstrate architectures in which weights are stored and processed close to or inside memory arrays. We emphasize that $r = 1$ should be interpreted as an idealized multi-chip or memory-integrated steady-state bound rather than an assumption that all model weights fit on a conventional single chip. For the models considered here, the required 4-bit weight capacity ranges from approximately 3.37 GB for LLaMA-2-7B to 202.93 GB for LLaMA-3.1-405B; $r = 0.5$ therefore serves as an informative intermediate operating point.

Latency model. For QANN (Q) and QC-A2S (S), the compute times are modeled as

$$t^{Q}_{\text{compute}} = \frac{N^{Q}_{\text{MAC}}}{\Theta_{\text{MAC}}} + \frac{N^{Q}_{\text{AC}}}{\Theta_{\text{AC}}}, \tag{20}$$

and

$$t^{S}_{\text{compute}} = \frac{N^{S}_{\text{MAC}}}{\Theta_{\text{MAC}}} + \frac{N^{S}_{\text{AC}}}{\Theta_{\text{AC}}}, \tag{21}$$

where Θ_{MAC} and Θ_{AC} denote the matched MAC and AC throughputs.

An important property of QC-A2S is that the emitting phase of layer l overlaps with the receiving phase of layer $l + 1$. Consequently, the TTFS causal depth is

$$t_{\text{TTFS,causal}} = \frac{(L + 1)T}{f}, \tag{22}$$

rather than the fully serialized $2LT/f$. This receiving–emitting pipeline follows the event-driven principle that message reception and neuron processing can proceed concurrently [4].

The resulting end-to-end latencies are determined by the slowest overlapping path:

$$t_Q = 1.05 \max\left(t^{Q}_{\text{compute}}, t^{Q}_{\text{DRAM}}, t^{Q}_{\text{SRAM}}, t^{Q}_{\text{NoC}}\right), \tag{23}$$

and

$$t_S = 1.05 \max\left(t^{S}_{\text{compute}}, t^{S}_{\text{DRAM}}, t^{S}_{\text{SRAM/queue}}, t^{S}_{\text{NoC}}, t_{\text{TTFS,causal}}\right). \tag{24}$$

Table 10 reports the component-wise critical-path latency at $r = 0$. The compute path remains the critical path for all six models. QC-A2S consistently reduces the end-to-end latency, from 75.458 to 46.908 ms on LLaMA-2-7B and from 5505.827 to 2902.573 ms on LLaMA-3.1-405B. Importantly, the TTFS causal component remains negligible because inter-layer receiving and emitting are pipelined rather than serialized across the full temporal window.

Scaling behavior. Table 11 summarizes the scaling trend. Across the evaluated model sizes, the latency speedup increases from $1.61\times$ on LLaMA-2-7B to $1.90\times$ on LLaMA-3.1-405B. The system-energy advantage also generally becomes more pronounced with increasing model scale, particularly when weight residency reduces the shared off-chip memory cost. For example, under $r = 1$, the estimated system-energy reduction grows from 34.06% to 46.87%. These results suggest that QC-A2S benefits increasingly from larger compute-dominated workloads, while memory residency further exposes the energy advantage by amortizing common weight-transfer costs.

Table 10: Component-wise critical-path latency under matched hardware conditions ($r = 0$). All values are in ms per matched workload.

Model	Backend	Compute	DRAM	SRAM/queue	NoC	TTFS causal	Total
LLaMA-2-7B	QANN	71.865	6.589	0.387	0.563	–	75.458
	QC-A2S	44.674	6.589	0.598	0.986	0.001	**46.908**
LLaMA-3-8B	QANN	98.492	7.850	0.474	0.698	–	103.417
	QC-A2S	55.960	7.850	0.767	1.283	0.001	**58.758**
LLaMA-2-13B	QANN	138.682	12.708	0.719	1.032	–	145.616
	QC-A2S	75.802	12.708	1.047	1.689	0.001	**79.592**
LLaMA-2-70B	QANN	738.170	67.428	3.603	5.054	–	775.078
	QC-A2S	398.882	67.428	4.711	7.268	0.003	**418.826**
LLaMA-3-70B	QANN	905.967	68.969	3.778	5.354	–	951.266
	QC-A2S	485.132	68.969	5.188	8.174	0.003	**509.389**
LLaMA-3.1-405B	QANN	5243.645	396.737	20.442	28.214	–	5505.827
	QC-A2S	2764.356	396.737	24.676	36.682	0.004	**2902.573**

Table 11: Scaling trends in system energy and latency.

Model	System-energy reduction			Latency speedup
	$r = 0$	$r = 0.5$	$r = 1$	
LLaMA-2-7B	10.31%	15.83%	34.06%	1.61×
LLaMA-3-8B	12.86%	19.30%	38.73%	1.76×
LLaMA-2-13B	12.66%	19.44%	41.90%	1.83×
LLaMA-2-70B	13.31%	20.45%	44.10%	1.85×
LLaMA-3-70B	15.19%	22.66%	44.65%	1.87×
LLaMA-3.1-405B	15.96%	23.82%	46.87%	1.90×

D Experimental Supplementary

Experiment Configurations. All experiments were conducted on a server equipped with NVIDIA A100 GPUs (80 GB of memory), Intel Xeon CPUs, and 512 GB of RAM. The models were implemented in PyTorch 2.6 with CUDA 12.4 support. For fair comparison, all baseline methods were re-implemented or run using their officially released code under the same environment and hyperparameter settings whenever possible. In addition to the hardware information mentioned in the main text, we provide further details about the reproduction of baselines here.

Baselines. We compare our approach against several representative baselines that adapt LLMs through either quantization or ANN-to-SNN conversion:

- **PrefixQuant** [9] is a weight activated quantization method that addresses token-wise outliers in the KV cache and employs lightweight blockwise training, achieving strong conversion performance
- **TTFSFormer** [67] applies time-to-first-spike coding to Transformers under continuous assumptions, extending TTFS neurons to handle nonlinear layers and achieving competitive accuracy with reduced energy consumption.
- **SpikeLLM** [58] presents the first spiking LLMs by incorporating bio-inspired spiking mechanisms with generalized integrate-and-fire neurons, yielding improvements in perplexity and reasoning accuracy compared to quantized LLMs
- **DuQuant** [28] introduces a method that uses rotation and permutation transformations to redistribute outlier activations, effectively improving the performance of low-bit quantized LLMs

Supplementary Experiment Results.

Table 12: Comparison with **DuQuant** of Accuracy (↑) and Perplexity (↓) across various LLaMA models. The entries highlighted in gray correspond to our QC-A2S method.

			Accuracy ↑						Perplexity↓		
Model	**Method**	**Prec.**	**Wino.**	**Hella.**	**ArcC**	**ArcE**	**PiQA**	**Avg.**	**C4**	**Wiki2**	**Avg.**
2-7B	Baseline	FP16	69.22	76.00	46.33	74.62	79.11	69.06	6.97	5.47	6.22
	DuQuant	W4A4	62.12	69.54	36.77	51.89	75.24	59.11	7.90	6.28	7.09
	DuQuant	W6A6	67.32	72.49	40.87	52.99	76.99	62.13	7.03	5.53	6.28
	QC-A2S	T=16	66.77	73.08	41.21	70.45	77.64	65.83	7.73	6.12	6.93
	QC-A2S	T=64	69.38	76.23	45.99	73.57	78.13	68.66	7.07	5.54	6.30
3-8B	Baseline	FP16	72.69	79.19	53.41	77.69	80.79	72.75	8.88	6.14	7.51
	DuQuant	W4A4	66.22	73.07	41.81	68.48	75.68	65.05	11.98	8.56	10.27
	DuQuant	W6A6	72.77	79.14	52.05	77.27	80.20	72.29	8.38	6.27	7.33
	QC-A2S	T=16	70.09	74.36	48.29	75.97	77.86	69.31	11.23	7.82	9.53
	QC-A2S	T=64	72.06	78.26	54.95	77.90	79.43	72.52	9.28	6.47	7.88
2-13B	Baseline	FP16	72.38	79.38	49.06	77.53	80.52	71.77	6.47	4.88	5.67
	DuQuant	W4A4	66.06	73.68	41.55	55.60	77.31	62.84	7.05	5.42	6.23
	DuQuant	W6A6	69.22	76.19	43.43	56.94	78.62	64.88	6.50	4.92	5.71
	QC-A2S	T=16	70.32	75.62	46.42	73.95	77.97	68.86	8.36	6.36	7.36
	QC-A2S	T=64	72.53	79.07	49.23	75.76	79.71	71.26	6.63	5.04	5.83
2-70B	Baseline	FP16	77.98	83.82	57.42	80.98	82.70	76.58	5.52	3.32	4.42
	DuQuant	W4A4	74.19	79.21	46.07	58.16	79.27	67.38	5.87	3.79	4.83
	QC-A2S	T=16	73.95	79.22	51.96	76.94	80.09	72.43	7.31	5.03	6.17

Table 13: Nonlinear operation count.

Layer	**Operand Size**	**Number of Neurons**	**Operation Counts**
$\mathrm{MatMul}(A, B)$	$A \in \mathbb{R}^{M\times K},\ B \in \mathbb{R}^{K\times N}$	$MN(K+1)$	$MN(3K+2)$
$\mathrm{Softmax}(x)$	$x \in \mathbb{R}^N$	$2(N+1)$	$6N$
$\mathrm{LayerNorm}(x)$	$x \in \mathbb{R}^N$	$3N+3$	$9N+1$
Activation (GELU/SiLU)	$x \in \mathbb{R}^N$	N	N

E Conversion Error

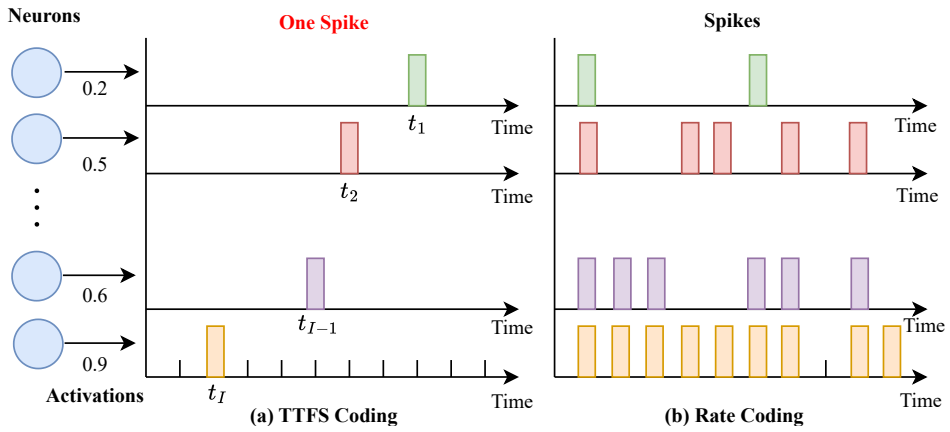


Figure 4: TTFS Coding vs. Rate Coding.

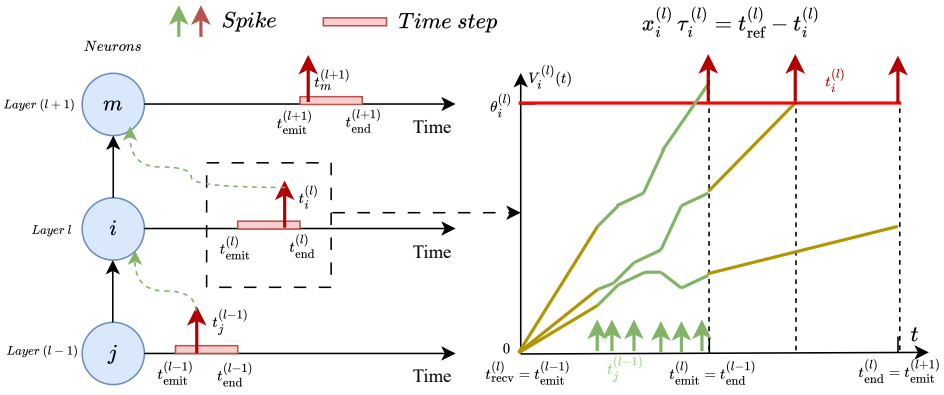


Figure 5: The process of TTFS-based spiking neural neurons.

In this section, we provide a detailed analysis of the conversion error between the ANN and the converted TTFS-based SNN across layers. We assume that both the ANN and SNN receive the same input from layer $l-1$, i.e., $\alpha^{(l-1)} = x^{(l-1)}$, and then analyze the error in layer l.

ANN neurons. For ANNs, the output α^l of neurons in layer l is realized by a linear weighting $W^{(l)}$ and nonlinear mappings $F(\cdot)$:

$$\alpha^{(l)} = F\left(W^{(l)}\alpha^{(l-1)}\right), \tag{25}$$

SNN neurons. For TTFS-based SNNs, we consider the relation between the spike time $t^{(l)}$ of SNN and the corresponding activation value $x^{(l)}$ of ANN:

$$x^{(l)} = \frac{1}{\tau^{(l)}}\left(t_{\text{ref}}^{(l)} - t^{(l)}\right). \tag{26}$$

From Eqs.(25) and (26), along with the conditions $V\left(t_{\text{emit}}^{(l)}\right) = W^{(l)} \cdot f\left(x^{(l-1)}\right)$ and $V(t^{(l)} - 1) < \theta^{(l)} \leq V(t^{(l)})$, $t^{(l)} \in \{t_{\text{emit}}^{(l)}, \ldots, t_{\text{end}}^{(l)}\}$, it follows that a transformation between the temporal domain (relate to $t^{(l)}$) and the numerical domain (relate to $x^{(l)}$) enables the activation value $a^{(l)}$ of analog neurons in the ANN to be mapped onto $x^{(l)}$ in the TTFS-based SNN. Because the output ranges and types of SNNs and ANNs differ, conversion errors are generally unavoidable. During the ANN-to-SNN conversion, two primary sources of error, clipping error $\mathcal{E}_{\text{c}}^{(l)}$ and quantization error $\mathcal{E}_{\text{q}}^{(l)}$, both of which contribute to the performance gap between ANNs and SNNs.

For layer l, the total error decomposes as:

$$\mathcal{E}^{(l)} = \mathcal{E}_{\mathrm{c}}^{(l)} + \mathcal{E}_{\mathrm{q}}^{(l)} \tag{27}$$

Clipping error. Clipping error denotes the error caused by different value ranges of ANNs and SNNs. For an temporal coding spiking neural neuron, when the time steps $T^{(l)}$ are fixed, the output of SNN: $x^{(l)}$ is in the range of $\left[a^{(l)}, b^{(l)}\right]$, where $a^{(l)} = \frac{t_{\mathrm{ref}}^{(l)} - t_{\mathrm{end}}^{(l)}}{\tau^{(l)}}$ and $b^{(l)} = \frac{t_{\mathrm{ref}}^{(l)} - t_{\mathrm{emit}}^{(l)}}{\tau^{(l)}}$. We define the α_{max} as the maximum value in $\alpha^{(l)}$, α_{min} is the minimum value in in $\alpha^{(l)}$. Then the output $\alpha \in \left[\alpha_{min}, a^{(l)}\right]$ of ANNs will be mapped to the same value $a^{(l)}$, the output $\alpha \in \left[b^{(l)}, \alpha_{max}\right]$ of ANNs will be mapped to the same value $b^{(l)}$, which will cause conversion error named clipping error $\mathcal{E}_{\mathrm{c}}$.

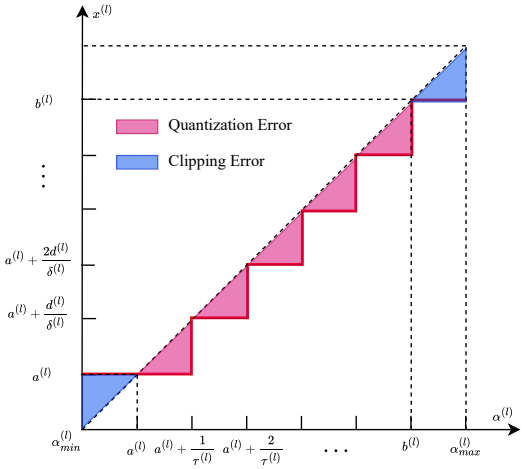


Figure 6: Clipping error and quantization error.

Quantization error. The output spike time $t^{(l)}$ is discrete, so the final output $x^{(l)} = \frac{1}{\tau^{(l)}}\left(t_{\mathrm{ref}}^{(l)} - t^{(l)}\right)$ is also discrete, while the output activation value α of the ANNs is continuous. Therefore, when mapping $\alpha^{(l)}$ to $x^{(l)}$, there will be unavoidable error related to temporal resolution, named quantization error $\mathcal{E}_{\mathrm{q}}$. For example, when the output of ANNs satisfies $\alpha \in \left[\frac{t^{(l)}d^{(l)}}{T^{(l)}}, \frac{(t^{(l)}+1)d^{(l)}}{T^{(l)}}\right), t^{(l)} = t_{\mathrm{ref}}^{(l)} - t_{\mathrm{end}}^{(l)}, t_{\mathrm{ref}}^{(l)} - t_{\mathrm{end}}^{(l)} + 1, ..., t_{\mathrm{ref}}^{(l)} - t_{\mathrm{emit}}^{(l)} - 1$, the corresponding mapped value of SNN will be $\frac{t^{(l)}d^{(l)}}{T^{(l)}}$.

Lemma E.1. ***Analysis for clipping error****. For a target ANN's output $\alpha_i^{(l)}$, the clipping error between the output of ANN and SNN is:*

$$\mathcal{E}_c^{(l)} = \begin{cases} \left\| \alpha_i^{(l)} - b_i^{(l)} \right\| & \text{if } \alpha_i^{(l)} > b_i^{(l)} \\ 0 & \text{if } \alpha_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}] \\ \left\| a_i^{(l)} - \alpha_i^{(l)} \right\| & \text{if } \alpha_i^{(l)} < a_i^{(l)}, \end{cases} \tag{28}$$

where $a_i^{(l)} = \frac{t_{ref}^{(l)} - t_{end}^{(l)}}{\tau_i^{(l)}}$ and $b_i^{(l)} = \frac{t_{ref}^{(l)} - t_{emit}^{(l)}}{\tau_i^{(l)}}$.

Lemma E.2. ***Upper bound for quantization error:*** *In the theoretical analysis under the continuous setting of Theorem 4.1 and Theorem 4.3 in [67], we denote the output under continuous coding as $y_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}]$ corresponds to the ANN with continuous outputs, while in the practically deployable discrete coding scenario, the SNN output is denoted as $x_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}]$ corresponds to the ANN with discretized outputs. That is, there exists quantization error $\mathcal{E}_q^{(l)}$ between TTFS coding in the continuous setting and its practical deployment. Let $T^{(l)}$ be the time window and Ω be the corresponding clock time, and the derivatives of the function h and its inverse are bounded by G_1 and G_2. Then, the quantization error can be bounded as:*

$$\mathcal{E}_q^{(l)} = \left\| x_i^{(l)} - y_i^{(l)} \right\| \leq \frac{G_1 G_2 \Omega}{T^{(l)}}. \tag{29}$$

F Proofs

Proof. Proof of Lemma E.2:
According to Theorem 4.3 in [67]: in the continuous setting, if the spike is emitted at time $t_i^{(l)} \in [t_{\text{emit}}^{(l)}, t_{\text{end}}^{(l)}]$, i.e. the corresponding value $y_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}]$. Then

$$\theta_i^{(l)} = V(t_{\text{emit}}^{(l)}) + \int_0^{t_i^{(l)} - t_{\text{emit}}^{(l)}} \psi_i^{(l)}(v)\, dv \tag{30}$$

$$= V(t_{\text{emit}}^{(l)}) + \int_0^{t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)} - \tau_i^{(l)} y_i^{(l)}} \frac{1}{\tau_i^{(l)}} (h^{-1})' \left(b_i^{(l)} - \tfrac{v}{\tau_i^{(l)}} \right) dv \tag{31}$$

$$= V(t_{\text{emit}}^{(l)}) - h^{-1}\left(b_i^{(l)} - \tfrac{v}{\tau_i^{(l)}} \right) \Big|_0^{\tau_i^{(l)}(b_i^{(l)} - y_i^{(l)})} \tag{32}$$

$$= V(t_{\text{emit}}^{(l)}) - h^{-1}(y_i^{(l)}) + h^{-1}(b_i^{(l)}). \tag{33}$$

Because $\theta_i^{(l)} = h^{-1}(b_i^{(l)})$:

$$h^{-1}(y_i^{(l)}) = V(t_{\text{emit}}^{(l)}), \tag{34}$$

which indicates that:

$$y_i^{(l)} = h\Big(V(t_{\text{emit}}^{(l)}) \Big). \tag{35}$$

In the discrete setting, the spike is emitted at time $t_i^{(l)} \in \{t_{\text{emit}}^{(l)}, t_{\text{emit}}^{(l)} + 1, \ldots, t_{\text{end}}^{(l)}\}$, the corresponding value $x_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}]$. Let $\Delta_i^{(l)} \geq 0$ is a compensation constant, which is actually the difference between the $\theta_i^{(l)}$ and the membrane potential at the spike time. The following equation satisfies:

$$\theta_i^{(l)} = V(t_{\text{emit}}^{(l)}) + \int_0^{t_i^{(l)} - t_{\text{emit}}^{(l)}} \psi_i^{(l)}(v)\, dv - \Delta_i^{(l)} \tag{36}$$

$$= V(t_{\text{emit}}^{(l)}) + \int_0^{t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)} - \tau^{(l)} x_i^{(l)}} \frac{1}{\tau_i^{(l)}} (h^{-1})' \left(b_i^{(l)} - \tfrac{v}{\tau_i^{(l)}} \right) dv - \Delta_i^{(l)} \tag{37}$$

$$= V(t_{\text{emit}}^{(l)}) - h^{-1}\left(b_i^{(l)} - \tfrac{v}{\tau_i^{(l)}} \right) \Big|_0^{\tau_i^{(l)}(b_i^{(l)} - x_i^{(l)})} - \Delta_i^{(l)} \tag{38}$$

$$= V(t_{\text{emit}}^{(l)}) - h^{-1}(x_i^{(l)}) + h^{-1}(b_i^{(l)}) - \Delta_i^{(l)}. \tag{39}$$

Because $\theta_i^{(l)} = h^{-1}(b_i^{(l)})$:

$$h^{-1}(x_i^{(l)}) = V(t_{\text{emit}}^{(l)}) - \Delta_i^{(l)}, \tag{40}$$

which indicates that:

$$x_i^{(l)} = h\Big(V(t_{\text{emit}}^{(l)}) - \Delta_i^{(l)} \Big). \tag{41}$$

The error of discrete coding in the continuous setting can be expressed as:

$$\left\| y_i^{(l)} - x_i^{(l)} \right\| = \left\| h\Big(V(t_{\text{emit}}^{(l)}) \Big) - h\Big(V(t_{\text{emit}}^{(l)}) - \Delta_i^{(l)} \Big) \right\| \tag{42}$$

By the mean value theorem, we obtain:

$$\left\| y_i^{(l)} - x_i^{(l)} \right\| = \left\| |h'(\xi)| \cdot \Delta_i^{(l)} \right\|, \tag{43}$$

where $\xi \in \left[V(t_{\text{emit}}^{(l)}) - \Delta_i^{(l)}, V(t_{\text{emit}}^{(l)})\right]$.

Furthermore, we examine $\Delta_i^{(l)}$ to provide a more in-depth analysis of the error. We assume that the spike firing time corresponding precisely to the ANN output is denoted as $[t]_i^{(l)}$. Based on the characteristics of TTFS encoding, it follows that:

$$t_i^{(l)} - 1 \leq [t]_i^{(l)} \leq t_i^{(l)}. \tag{44}$$

Then $\Delta_i^{(l)}$ can be represented as:

$$\Delta_i^{(l)} = \int_{[t]_i^{(l)} - t_{\text{emit}}^{(l)}}^{t_i^{(l)} - t_{\text{emit}}^{(l)}} \psi_i(s)\, ds = h^{-1}(t_i^{(l)} - t_{\text{emit}}^{(l)}) - h^{-1}([t]_i^{(l)} - t_{\text{emit}}^{(l)}). \tag{45}$$

By the mean value theorem, we obtain:

$$\left\|\Delta_i^{(l)}\right\| = \left\|\left|(h^{-1})'(\hat{t}_i)\right| \cdot (t_i^{(l)} - [t]_i^{(l)})\right\|, \tag{46}$$

where $\hat{t}_i \in \left[[t]_i^{(l)} - t_{\text{emit}}^{(l)}, t_i^{(l)} - t_{\text{emit}}^{(l)}\right]$.

Then the error $\epsilon_i^{(l)}$ can be bounded by the following inequality:

$$\left\|y_i^{(l)} - x_i^{(l)}\right\| \leq |h'(\xi)| \cdot \left|(h^{-1})'(\hat{t}_i)\right| \cdot |[t]_i^{(l)} - t_i^{(l)}| \tag{47}$$

By the definition of clock precision: $\Delta t_{real} = t_{real}(t+1) - t_{real}(t)$, where $t_{real}(t) = t \cdot \Delta t_{real}$, we obtain:

$$\begin{aligned}
\left\|y_i^{(l)} - x_i^{(l)}\right\| &\leq |h'(\xi)| \cdot \left|(h^{-1})'(\hat{t}_i)\right| \cdot \Delta t_{real} \\
&= |h'(\xi)| \cdot \left|(h^{-1})'(\hat{t}_i)\right| \cdot \frac{\Omega}{t_{\text{end}}^{(l)} - t_{\text{emit}}^{(l)}} \\
&= |h'(\xi)| \cdot \left|(h^{-1})'(\hat{t}_i)\right| \cdot \frac{\Omega}{T^{(l)}} \\
&\leq \frac{G_1 G_2 \Omega}{T^{(l)}}
\end{aligned} \tag{48}$$

□

Proof. Proof of Theorem 4.1:
For clipping error, according to Lemma E.1, we can get:

$$\mathcal{E}_{\text{c}}^{(l)} = \begin{cases} \left\|\alpha_i^{(l)} - b_i^{(l)}\right\| & \text{if } \alpha_i^{(l)} > b_i^{(l)} \\ 0 & \text{if } \alpha_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}] \\ \left\|a_i^{(l)} - \alpha_i^{(l)}\right\| & \text{if } \alpha_i^{(l)} < a_i^{(l)}, \end{cases} \tag{49}$$

We define the center of the output interval of SNN as:

$$c_i^{(l)} = \frac{a_i^{(l)} + b_i^{(l)}}{2} \tag{50}$$

The clipping error can then be restated as follows:

$$\mathcal{E}_{\text{c}}^{(l)}(T^{(l)}) = \max\left(\left|\alpha_i^{(l)} - c_i^{(l)}\right| - \frac{T^{(l)}}{2\tau_i^{(l)}}, 0\right) \tag{51}$$

We take the derivative of $T^{(l)}$ to get the sensitivity of the error $\mathcal{E}_{\text{c}}^{(l)}(T^{(l)})$ to $T^{(l)}$:

$$\frac{d}{dt}\mathcal{E}_{\text{c}}^{(l)}(T^{(l)}) = \begin{cases} -\dfrac{1}{2\tau_i^{(l)}}, & \left|\alpha_i^{(l)} - c_i^{(l)}\right| > \dfrac{T^{(l)}}{2\tau_i^{(l)}}, \\ 0, & \left|\alpha_i^{(l)} - c_i^{(l)}\right| < \dfrac{T^{(l)}}{2\tau_i^{(l)}}. \end{cases} \tag{52}$$

Once clipping occurs, increasing $T^{(l)}$ will reduce the error linearly with a constant slope of $-\frac{1}{2\tau_i^{(l)}}$; within the valid interval, the error is unaffected by $T^{(l)}$.

For quantization error, according to Lemma E.2, we can get:

$$\mathcal{E}_{\mathrm{q}}^{(l)} \leq \frac{G_1 G_2 \Omega}{T^{(l)}}. \tag{53}$$

For an L-layer network with I neurons in each layer, we can get:

$$\begin{aligned}\mathcal{E} &= \sum_{i=1}^{I}\sum_{l=1}^{L}\left(\mathcal{E}_{\mathrm{c}}^{(l)} + \mathcal{E}_{\mathrm{q}}^{(l)}\right) \\ &\leq \sum_{i=1}^{I}\sum_{l=1}^{L}\left(\max\left(\left|\alpha_i^{(l)} - c_i^{(l)}| - \frac{T^{(l)}}{2\tau_i^{(l)}}\right|, 0\right) + \frac{G_1 G_2 \Omega}{T^{(l)}}\right)\end{aligned} \tag{54}$$

Let $T = \min\left\{T^{(l)}\right\}_{l=1}^{L}$ and $\tau = \max\left\{\left\{\tau_i^{(l)}\right\}_{i=1}^{I}\right\}_{l=1}^{L}$:

$$\mathcal{E} \leq LI \cdot \max\left(\left|\alpha_i^{(l)} - \frac{a_i^{(l)} + b_i^{(l)}}{2}\right| - \frac{T^{(l)}}{2\tau_i^{(l)}}, 0\right) + \frac{LIG_1G_2\Omega}{T} \tag{55}$$

Proof. Proof of Theorem 4.3:

For arbitrary fixed $\eta_{ij}^{(l)}$, $\psi_i^{(l)}$, $C_i^{(l)}$ and $\theta_i^{(l)}$ in SNNs with time window $T^{(l)}$, in the receiving phase:

$$\begin{aligned}&V\left(t_{\text{emit}}^{(l)}\right) \\ &= \frac{1}{\tau_i^{(l)}} \sum_{t=T_{\text{recv}}^{(l)}}^{T_{\text{emit}}^{(l)}-1} \left(\sum_{j=1}^{I} w_{ij}^{(l)}\, \eta_{ij}^{(l)}\left(t - t_j^{(l-1)}\right) - C_i^{(l)}\right) \\ &= \frac{1}{\tau_i^{(l)}} \sum_{j=1}^{I} \sum_{t=t_j^{(l-1)}}^{T_{\text{emit}}^{(l)}-1} w_{ij}^{(l)}\, \eta_{ij}^{(l)}\left(x_j^{(l-1)}\tau_j^{(l-1)} + t - t_{\text{ref}}^{(l-1)}\right) + T^{(l)} C_i^{(l)}.\end{aligned} \tag{56}$$

In the emitting phase, let $\Delta_i^{(l)} \geq 0$ is a compensation constant, which is actually the difference between the $\theta_i^{(l)}$ and the membrane potential at the spike time. We can get:

$$V\left(t_{\text{emit}}^{(l)}\right) + \sum_{v=0}^{t_i^{(l)} - t_{\text{emit}}^{(l)} - 1} \psi_i^{(l)}(v) - \Delta_i^{(l)} = \theta_i^{(l)}. \tag{57}$$

We denote $S(t) = \sum_{v=0}^{t - t_{\text{emit}}^{(l)} - 1} \psi_i^{(l)}(v)$, then:

$$S(t_i^{(l)}) = \theta_i^{(l)} + \Delta_i^{(l)} - V\left(t_{\text{emit}}^{(l)}\right). \tag{58}$$

Then:

$$t_i^{(l)} = S^{-1}\left(\theta_i^{(l)} + \Delta_i^{(l)} - V\left(t_{\text{emit}}^{(l)}\right)\right). \tag{59}$$

According to the relationship between $x_i^{(l)}$ and $t_i^{(l)}$, we can get:

$$x_i^{(l)} = \frac{1}{\tau_i^{(l)}}\left(t_{\text{ref}}^{(l)} - S^{-1}\left(\theta_i^{(l)} + \Delta_i^{(l)} - V\left(t_{\text{emit}}^{(l)}\right)\right)\right). \tag{60}$$

let $W^{(l)} = (w_{ij}^{(l)})_{I\times I}$ is the weight matrix:

$$\begin{aligned}x_i^{(l)} &= F^{(l)}(W^{(l)}; x_1^{(l-1)}, ..., x_I^{(l-1)}) \\ &= \frac{1}{\tau_i^{(l)}}\left(t_{\text{ref}}^{(l)} - S^{-1}\left(\theta_i^{(l)} + \Delta_i^{(l)} - \frac{1}{\tau_i^{(l)}}\sum_{j=1}^{I}\sum_{t=t_j^{(l-1)}}^{T_{\text{emit}}^{(l)}-1} w_{ij}^{(l)}\,\eta_{ij}^{(l)}\left(x_j^{(l-1)}\tau_j^{(l-1)} + t - t_{\text{ref}}^{(l-1)}\right) - T^{(l)}C_i^{(l)}\right)\right),\end{aligned} \tag{61}$$

□

Proof. Proof of Theorem 4.4:
Consider the potential change in the receiving stage.

$$
\begin{aligned}
&V_i\Big(t_{\text{emit}}^{(l)}\Big) \\
&= \frac{1}{\tau_i^{(l-1)}} \sum_{t=t_{\text{emit}}^{(l-1)}}^{t_{\text{end}}^{(l-1)}-1} \sum_j w_{ij}^{(l)}\, \eta_{ij}^{(l)}\Big(t - t_j^{(l-1)}\Big) + C_i^{(l)} \\
&= \frac{1}{\tau_i^{(l-1)}} \sum_j w_{ij}^{(l)} \sum_{u=0}^{t_{\text{end}}^{(l-1)}-t_j^{(l-1)}-1} \eta_{ij}^{(l)}(u) + d_i^{(l-1)} \cdot C_i^{(l)} \\
&= \sum_j w_{ij}^{(l)}\, \frac{1}{\tau_i^{(l-1)}} \sum_{u=0}^{t_{\text{end}}^{(l-1)}-t_{\text{ref}}^{(l-1)}+\tau_i^{(l-1)}x_j^{(l-1)}-1} \eta_{ij}^{(l)}(u) + d_i^{(l-1)} \cdot C_i^{(l)} \\
&= \sum_j w_{ij}^{(l)} \sum_{u=0}^{\tau_i^{(l-1)}\left(x_j^{(l-1)}-a_i^{(l-1)}\right)-1} (f_{ij}\left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) - f_{ij}\left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right)) + d_i^{(l-1)} \cdot C_i^{(l)} \\
&= \sum_j w_{ij}^{(l)} \Big(f_{ij}(x_j^{(l-1)}) - f_{ij}(a_i^{(l-1)})\Big) + d_i^{(l-1)} \cdot C_i^{(l)} \\
&= \sum_j w_{ij}^{(l)}\, f_{ij}(x_j^{(l-1)}).
\end{aligned} \tag{62}
$$

where the second equation uses $u = t - t_j^{(l-1)}$; third equation uses $x_j^{(l-1)}\,\tau_i^{(l-1)} = t_{\text{ref}}^{(l-1)} - t_j^{(l-1)}$; fourth equation uses $a_i^{(l-1)} = \frac{t_{\text{ref}}^{(l-1)} - t_{\text{end}}^{(l-1)}}{\tau_i^{(l-1)}}$.

If the spike is emitted at time $t_i^{(l)} \in \{t_{\text{emit}}^{(l)}, t_{\text{emit}}^{(l)}+1, \ldots, t_{\text{end}}^{(l)}\}$, i.e. the corresponding value $x_i^{(l)} \in [a_i^{(l)}, b_i^{(l)}]$. Then:

$$
\begin{aligned}
\theta^{(l)} &= V(t_{\text{emit}}^{(l)}) + \sum_{v=0}^{t_i^{(l)}-t_{\text{emit}}^{(l)}-1} \psi_i^{(l)}(v) - \Delta_i^{(l)} \\
&= V(t_{\text{emit}}^{(l)}) + \sum_{v=0}^{t_{\text{ref}}^{(l)}-t_{\text{emit}}^{(l)}-\tau_i^{(l)}x_i^{(l)}-1} \left(h^{-1}(b_i^{(l)} - \frac{v}{\tau_i^{(l)}}) - h^{-1}(b_i^{(l)} - \frac{v+1}{\tau_i^{(l)}})\right) - \Delta_i^{(l)} \\
&= V(t_{\text{emit}}^{(l)}) + \sum_{v=0}^{\tau_i^{(l)}(b_i^{(l)}-x_i^{(l)})-1} \left(h^{-1}(b_i^{(l)} - \frac{v}{\tau_i^{(l)}}) - h^{-1}(b_i^{(l)} - \frac{v+1}{\tau_i^{(l)}})\right) - \Delta_i^{(l)} \\
&= V(t_{\text{emit}}^{(l)}) + h^{-1}(b_i^{(l)}) - h^{-1}(x_i^{(l)}) - \Delta_i^{(l)}.
\end{aligned} \tag{63}
$$

where the first equation uses $v = t_i^{(l)} - t_{\text{emit}}^{(l)}$, the second equation uses $x_i^{(l)}\,\tau_i^{(l)} = t_{\text{ref}}^{(l)} - t_i^{(l)}$, the third equation uses $b_i^{(l)} = \frac{t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)}}{\tau_i^{(l)}}$.
Thus

$$h^{-1}(x_i^{(l)}) = V(t_{\text{emit}}^{(l)}).$$

which indicates that

$$x_i^{(l)} = h(V(t_{\text{emit}}^{(l)})).$$

If $h(V(t_{\text{emit}}^{(l)})) > b_i^{(l)}$, then $V(t_{\text{emit}}^{(l)}) > h^{-1}(b_i^{(l)}) = \theta_i^{(l)}$, which means that a spike is emitted once at $t_{\text{emit}}^{(l)}$, representing the value $\frac{t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)}}{\tau_i^{(l)}} = b_i^{(l)}$.

If $h(V(t_{\text{emit}}^{(l)})) < a_i^{(l)}$, then the potential at time $t_{\text{end}}^{(l)}$ is:

$$\begin{aligned}
&V(t_{\text{emit}}^{(l)}) + \sum_{v=0}^{T^{(l)}} \psi_i^{(l)}(v) \\
&= V(t_{\text{emit}}^{(l)}) + \sum_{v=0}^{T^{(l)}} \left(h^{-1}(b_i^{(l)} - \frac{v}{\tau_i^{(l)}}) - h^{-1}(b_i^{(l)} - \frac{v+1}{\tau_i^{(l)}}) \right) \\
&= V(t_{\text{emit}}^{(l)}) + h^{-1}(b_i^{(l)}) - h^{-1}(a_i^{(l)}) \\
&< h^{-1}(b_i^{(l)}) = \theta_i^{(l)}.
\end{aligned} \tag{64}$$

which means that there will be no spike, representing the value $a_i^{(l)}$. □

□

Proof. Proof of Corollary 4.5:

Input transform: The input of QANN at i-th neuron of l-th layer is $\mathbf{X}_q^{(l)} = \sum_j w_{ij}^{(l)} x_j^{(l-1)} \in [a_i^{(l-1)}, b_i^{(l-1)}]$. In order to approximate the input of QANN, based on Theorem 4.4, we set the kernel function $\eta_{ij}^{(l)}$ and $C_i^{(l)}$ as follows:

$$\eta_{ij}^{(l)}(u) = \mathcal{H}\left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)} \right),$$

$$C_i^{(l)} = \sum_j \frac{a_i^{(l-1)}}{d_i^{(l-1)}} w_{ij},$$

Then, the membrane potential after reception is completed can be expressed as:

$$V(T_{\text{emit}}) = \sum_j w_{ij}^{(l)} x_j^{(l-1)} = \mathbf{X}_q^{(l)}. \tag{65}$$

Output transform: In order to approximate the output of QANN at l-th layer of i-th neuron: $\hat{\mathbf{X}}_q^{(l)} \in [a_i^{(l)}, b_i^{(l)}]$, based on Theorem 4.4, we set the kernel function $\psi_i^{(l)}$ and threshold as follows:

$$\psi_i^{(l)}(v) = \frac{1}{\tau_i^{(l)}}, \quad \theta_i^{(l)} = b_i^{(l)} \tag{66}$$

If the spike is emitted at time $t \in \{t_{\text{emit}}^{(l)}, t_{\text{emit}}^{(l)} + 1, \ldots, t_{\text{end}}^{(l)}\}$:

$$X_i^{(l-1)} + \frac{1}{\tau_i^{(l)}} \cdot t \geq \theta_i^{(l)}. \tag{67}$$

According to the definition of t, we can get:

$$t = \left\lceil (\theta_i^{(l)} - \mathbf{X}_q^{(l)}) \tau_i^{(l)} \right\rceil \tag{68}$$

According to the rounding range of t, we add the clip function to get:

$$t = \text{clip}\left(\left\lceil (\theta_i^{(l)} - \mathbf{X}_q^{(l)}) \tau_i^{(l)} \right\rceil, t_{\text{emit}}^{(l)}, t_{\text{end}}^{(l)} \right) \tag{69}$$

According to the relation between spike time and corresponding activation value and $\theta_i^{(l)} = b_i^{(l)}$:

$$x_i^{(l)} = \frac{1}{\tau_i^{(l)}} \left(t_{\text{ref}}^{(l)} - \text{clip}(\left\lceil (b_i^{(l)} - \mathbf{X}_q^{(l)}) \tau_i^{(l)} \right\rceil, t_{\text{emit}}^{(l)}, t_{\text{end}}^{(l)}) \right) \tag{70}$$

According to $b_i^{(l)} = \frac{t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)}}{\tau_i^{(l)}}$, we can get:

$$x_i^{(l)} = \frac{1}{\tau_i^{(l)}} \left(t_{\text{ref}}^{(l)} - \text{clip}(\left\lceil t_{\text{ref}}^{(l)} - t_{\text{emit}}^{(l)} - \mathbf{X}_q^{(l)} \tau_i^{(l)} \right\rceil, t_{\text{emit}}^{(l)}, t_{\text{end}}^{(l)}) \right) \tag{71}$$

Based on the relationship between the ceiling function and the floor function, we can derive the following:

$$x_i^{(l)} = \frac{1}{\tau_i^{(l)}}\text{clip}(\left\lfloor \mathbf{X}_q^{(l)}\tau_i^{(l)} \right\rfloor - t_{\text{ref}}^{(l)} + t_{\text{emit}}^{(l)}, -t_{\text{end}}^{(l)}, -t_{\text{emit}}^{(l)}) + \frac{1}{\tau_i^{(l)}} t_{\text{ref}}^{(l)} \tag{72}$$

Based on the properties of the floor function, we can conclude that:

$$x_i^{(l)} = \frac{1}{\tau_i^{(l)}}\text{clip}(\left\lfloor \mathbf{X}_q^{(l)}\tau_i^{(l)} \right\rfloor + t_{\text{end}}^{(l)} - t_{\text{ref}}^{(l)} + t_{\text{emit}}^{(l)}, 0, t_{\text{end}}^{(l)} - t_{\text{emit}}^{(l)}) + \frac{1}{\tau_i^{(l)}}\left(t_{\text{ref}}^{(l)} - t_{\text{end}}^{(l)}\right) \tag{73}$$

Let $t_{\text{emit}}^{(l)} = 0$, $t_{\text{end}}^{(l)} = N$, $\tau_i^{(l)} = \frac{1}{\lambda^{(l-1)}}$, $t_{\text{end}}^{(l)} - t_{\text{ref}}^{(l)} = z^{(l)}$, we can get:

$$x_i^{(l)} = \lambda^{(l)} \cdot \text{clip}(\left\lfloor \frac{\mathbf{X}_q^{(l)}}{\lambda^{(l)}} \right\rfloor + z^{(l)}, 0, N) - \lambda^{(l)} z^{(l)} = \hat{\mathbf{X}}_q^{(l)} \tag{74}$$

□

G Nonlinear Operations in QC-A2S

Corollary G.1 (Construction of SiLU)**.** *A TTFS-based neuron can be made equivalent to a discrete SiLU function with through the following configuration:*

$$\eta_{ij}^{(l)}(u) = \mathbb{I}[u \geq 0] \cdot \tau_i^{(l-1)} \cdot \left(\left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) \cdot \sigma\left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) - \left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) \cdot \sigma\left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right)\right)$$

$$C_i^{(l)} = \sum_j w_{ij}^{(l)} \frac{a_i^{(l-1)} \cdot \sigma(a_i^{(l-1)})}{d_i^{(l-1)}},\ \psi_i^{(l)}(v) = \frac{1}{\tau_i^{(l)}},\ \sigma(x) = \frac{1}{1+e^{-x}}. \tag{75}$$

Corollary G.2 (Construction of GELU)**.** *A TTFS-based neuron can be made equivalent to a discrete GELU function with*

$$\eta_{ij}^{(l)}(u) = \mathbb{I}[u \geq 0] \cdot \tau_i^{(l-1)} \cdot \left(\left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) \cdot \Phi\left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) - \left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) \cdot \Phi\left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right)\right)$$

$$\psi_i^{(l)}(v) = \frac{1}{\tau_i^{(l)}},\ \theta_i^{(l)} = b_i^{(l)},\ C_i^{(l)} = \sum_j w_{ij}^{(l)}\ \frac{a_i^{(l)}\Phi(a_i^{(l)})}{d_i^{(l-1)}}, \Phi(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \quad \text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}$$

Corollary G.3 (Construction of Softmax)**.** *The log-sum-exp of I inputs $x_1, x_2, \cdots, x_I$, i.e.,*

$$\log \sum_{j=1}^{I} e^{x_j}, \tag{76}$$

can be calculated in a single neuron with

$$\eta_{ij}^{(l)}(u) = \tau_i^{(l-1)} \cdot \left(\exp\left(\frac{u+1}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right) - \exp\left(\frac{u}{\tau_i^{(l-1)}} + a_i^{(l-1)}\right)\right). \tag{77}$$

$$C_i^{(l)} = \frac{I}{d_i^{(l-1)}} e^{a_i^{(l-1)}},\ \psi_i^{(l)}(v) = \frac{1}{\tau_i^{(l)}} \exp\left(b_i^{(l)} - \frac{v}{\tau_i^{(l)}}\right). \tag{78}$$

With the log-sum-exp neuron, we can obtain the softmax operator. We can calculate the logarithm of softmax, i.e.

$$\log\left(\frac{e^{x_i}}{\sum_{j=1}^{I} e^{x_j}}\right) = x_i - \log \sum_{j=1}^{I} e^{x_j}, \tag{79}$$

by subtracting the log-sum-exp from x_i. Finally, we can obtain the output after an exponential layer.

Corollary G.4 (Construction of RMSNorm)**.** *RMSNorm is a normalization method widely used in LLaMA architecture, which is a linear operation. RMSNorm is defined as:*

$$\mathrm{RMSNorm}(x_i) = \frac{x_i}{\sqrt{\frac{1}{I}\sum_{i=1}^{I} x_i^2}} \cdot \gamma + \beta. \tag{80}$$

We first can obtain the $\frac{1}{I}\sum_{i=1}^{I} x_i^2$ *by a single neuron with*

$$\eta_{ij}^{(1)}(u) = \tau_{ij}^{(0)} \left[\left(\frac{u+1}{\tau_i^{(0)}} + a_i^{(0)} \right)^2 - \left(\frac{u}{\tau_i^{(0)}} + a_i^{(0)} \right)^2 \right],\ C_i^{(1)} = \frac{(a_i^{(0)})^2}{Id_i^{(0)}},\ w^{(1)} = \frac{1}{I} \tag{81}$$

$$\psi_i^{(1)}(v) = \frac{1}{\tau_i^{(1)}},\ \theta_i^{(1)} = b_i^{(1)}. \tag{82}$$

Then, we can get $\frac{1}{\sqrt{\frac{1}{I}\sum_{i=1}^{I} x_i^2}}$ *with:*

$$\eta_{ij}^{(2)}(u) = \tau_{ij}^{(1)} \left[\left(\frac{u+1}{\tau_i^{(1)}} + a_i^{(1)} \right)^{-\frac{1}{2}} - \left(\frac{u}{\tau_i^{(1)}} + a_i^{(1)} \right)^{-\frac{1}{2}} \right],\ C_i^{(2)} = \frac{1}{Id_i^{(1)}(a_i^{(1)})^{\frac{1}{2}}}, \tag{83}$$

$$w^{(2)} = 1\ \psi_i^{(l)}(v) = \frac{1}{\tau_i^{(l)}},\ \theta_i^{(l)} = b_i^{(l)}. \tag{84}$$

Finally, multiply x_i *with* $\frac{1}{\sqrt{\frac{1}{I}\sum_{i=1}^{I} x_i^2}}$.

H Limitation

To construct a fully spike-based LLM, the design of the nonlinear components in this work follows TTFSFormer [67]. This design incurs complexity in both formulation and implementation. Simplifying and generalizing the nonlinear module construction while maintaining performance remains an important direction for future research.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction sections accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The assumptions are provided in Section 4, and the complete proofs are provided in the Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Code link and experimental details are provided in Section 5 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Data and code are provided in the code link of Section 5

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental details are provided in Appendix D

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Similar studies do not conduct statistical significance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments are conducted on a machine equipped with a cluster of NVIDIA A100-SXM4-80GB GPUs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conform the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our method does not involve any privacy or security concerns.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: All the datasets and models used in this paper are publicly available.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the codes, datasets, and models used in this paper are publicly available.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [No]

Justification: We utilize the LLM only for grammar checking.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.