Async-SNN: EFFICIENT SPIKE-DRIVEN LARGE LANGUAGE MODELS WITH ASYNCHRONOUS COMPUTING PARADIGM

Anonymous authors
Paper under double-blind review

ABSTRACT

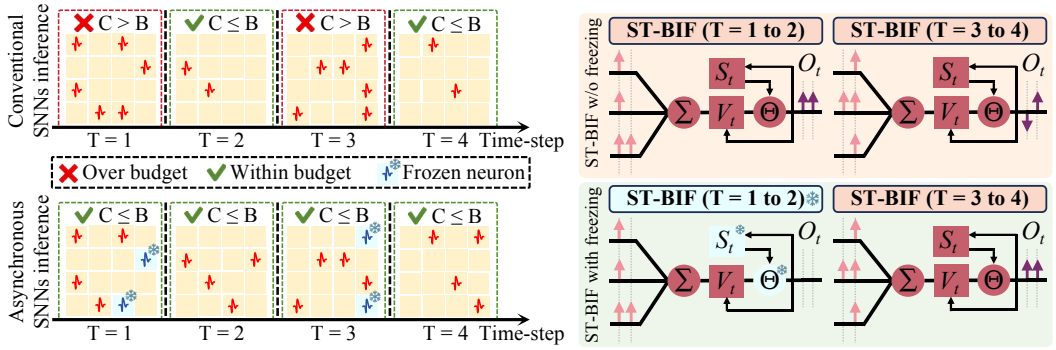
Large language models (LLMs) are increasingly deployed in edge scenarios where power supply can be strict, intermittent, and time-varying. Spiking neural networks (SNNs), with their event-driven and asynchronous computation paradigm, offer a promising path toward power-adaptive LLM inference. However, existing spiking LLMs largely inherit synchronization-intensive operators from conventional Transformers, such as RMSNorm and Softmax Attention, whose global statistical dependencies fundamentally prevent true neuron-level asynchronous execution. In this work, we propose *Async-SNN*, a framework for constructing fully spiking LLMs with asynchronous computing capability at the billion-parameter scale. We first build a ready-to-convert *Async-LLM* by replacing RMSNorm with Hardtanh Dynamic Normalization (HDN) and Softmax Attention with Normalized Linear Attention (NLA), thereby eliminating major sources of synchronization. We then introduce a coarse-to-fine ANN-to-SNN conversion pipeline that combines calibration, supervised fine-tuning based quantization-aware training, and equivalent ST-BIF neuron conversion to preserve task performance. Finally, we propose greedy adaptive firing (GAF), which dynamically reallocates neuron firing budgets across layers and time-steps under instantaneous power constraints. Experiments on 0.6B and 1.7B models show that *Async-SNN* consistently outperforms existing spiking LLM baselines and maintains strong perplexity and zero-shot accuracy even when only 10% of neurons are allowed to be active per time-step.

1 INTRODUCTION

With the rapid progress of large language models (LLMs) (Brown & Mann, 2020; Touvron & Lavril, 2023; Zhang et al., 2022; Le Scao et al., 2023; Yang et al., 2025), the demand for efficient inference has grown substantially. Unlike cloud-based LLM serving, where power is typically stable and continuously available, many edge platforms operate under stringent and time-varying power budgets, such as AIoT devices powered by energy-harvesting systems (Montanari et al., 2020; Lv & Xu, 2022; Sabovic et al., 2023) and intermittent computing platforms (Lucia & Ransford, 2015; Gobieski et al., 2019). In these settings, it remains challenging for LLM inference to adapt to the available power budget while maintaining strong task performance. Brain-inspired spiking neural networks (SNNs), with their *asynchronous and event-driven computation paradigm* (Yao et al., 2024; Koopman et al., 2025; Zhang et al., 2024; Deng et al., 2025; Rathi et al., 2023), offer a promising solution: neuronal activities can be dynamically suspended or resumed according to the instantaneous power budget, enabling power-adaptive execution without severely compromising task accuracy. Recent advances in spiking LLMs further suggest the potential of deploying large language models efficiently on neuromorphic hardware (Xing et al., 2024; Zheng et al., 2026; Hwang et al., 2024).

Nevertheless, constructing asynchronous spike-driven LLMs faces three key challenges. **1) Limited asynchronous capability in existing spiking LLMs.** Asynchronous computing (Koopman et al., 2025) allows activations or spikes to be processed upon arrival without layer-wise synchronization. However, existing spiking LLMs (Hwang et al., 2024; Xing et al., 2024; Zheng et al., 2026) largely inherit synchronization-intensive operators from conventional LLMs, including RMSNorm (Zhang & Sennrich, 2019), layer normalization, and Softmax Attention (Vaswani et al., 2017), which introduce global dependencies and hinder neuron-level asynchronous execution. **2) Difficulty in preserving**

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



(a) Asynchronous computing paradigm. (*e.g.*, B = 4) (b) Asynchronous paradigm of ST-BIF neuron.

Figure 1: Left: The asynchronous computing paradigm adaptively reschedules spike firing to satisfy the per-timestep budget constraint $C \leq B$, where C denotes the number of active neurons and B denotes the hardware budget (*e.g.*, $B = 4$). Right: Asynchronous execution of the ST-BIF neuron. Without freezing, ST-BIF emits three positive spikes and one negative spike, yielding a net output of two positive spikes. With freezing, ST-BIF reduces the number of emitted spikes from 4 to 2 while preserving the same cumulative output.

task performance. Removing synchronization operators changes the model architecture, while ANN-to-SNN conversion introduces discretization and temporal approximation errors. Thus, asynchronous spiking LLMs must preserve language modeling and reasoning performance with limited training overhead. **3) Lack of adaptive neuron-activity control under strict power budgets.** Naively suppressing spikes to meet a power budget may discard informative neuron activity. A practical solution must dynamically allocate firing opportunities across layers and time-steps, fully utilize the available budget, and reduce synaptic operations without degrading accuracy.

We develop *Async-SNN* to address these challenges. To the best of our knowledge, Async-SNN is the first framework that scales asynchronous SNNs to billion-parameter LLMs while supporting true neuron-level asynchronous execution, strong NLP performance, and strict power-budget compliance. **Our key idea is to pretrain a synchronization-free, ready-to-convert LLM, and convert it into an asynchronous SNN with budget-aware neuron firing control.** Our contributions are:

- **Synchronization-free ready-to-convert LLM.** We construct *Async-LLM* by replacing synchronization-intensive operators with asynchronous-conversion-friendly alternatives: RMSNorm is replaced by Hardtanh Dynamic Normalization (HDN), and Softmax Attention is replaced by Normalized Linear Attention (NLA). These replacements remove major global dependencies and make LLMs compatible with asynchronous SNN conversion.
- **Coarse-to-fine ANN-to-SNN conversion.** We introduce a coarse-to-fine conversion pipeline that preserves task performance by progressively applying *coarse calibration*, *supervised fine-tuning based quantization-aware training* (SFT QAT), and *equivalent ST-BIF conversion* on the ready-to-convert *Async-LLM*.
- **Greedy adaptive firing strategy.** We propose *greedy adaptive firing* (GAF), which exploits the asynchronism of *Async-SNN* to dynamically allocate spike firing across layers and time-steps, achieving a favorable budget–performance trade-off under instantaneous power constraints.

2 Related Works and Preliminary

Asynchronous Computing. Asynchronous computing enables SNNs to dynamically regulate spike firing, which is important for energy efficiency and hardware utilization in resource-constrained edge environments (M Ferreira et al., 2025; Yao et al., 2024; Zhang et al., 2024). As shown in Fig. 1(a), spiking activity exhibits inherent dynamic imbalance (Yao et al., 2024): the number of active neurons C at each time-step can vary substantially, either exceeding the hardware budget B or leaving available resources under-utilized. Such budget constraints are common in energy-harvesting systems (Montanari et al., 2020; Lv & Xu, 2022; Sabovic et al., 2023) and intermittent computing platforms (Lucia & Ransford, 2015; Gobieski et al., 2019), where peak-power operation cannot

always be sustained. The asynchronous paradigm addresses this issue by adaptively rescheduling spike firing over time, ensuring $C \leq B$ at every time-step while preserving reliable task execution.

Spiking Neuron for Async-SNN and A2S Conversion. To reconcile ANN-to-SNN conversion fidelity with event-driven asynchronous execution, we adopt the *bipolar integrate-and-fire with spike tracer* (ST-BIF) neuron (You et al., 2024). Rather than acting merely as a conventional spiking neuron, ST-BIF provides an equivalent mapping from an activation-quantized ANN to an asynchronous SNN. Specifically, ST-BIF is mathematically equivalent to a quantized rectified linear unit (Q-ReLU), enabling lossless conversion from quantized ANNs (QANNs) to SNNs. Its dynamics, illustrated in Fig. 1(b), are defined as:

$$\Theta(V, \mathrm{V_{thr}}, S) = \begin{cases} 1; & V \geq \mathrm{V_{thr}} \;\&\; S < S_{\max} \\ 0; & \text{other} \\ -1; & V < 0 \;\&\; S > S_{\min} \end{cases}; \quad O_t = \Theta(V_{t-1} + V_t^{\mathrm{in}}; \mathrm{V_{thr}}; S_{t-1}) \times \mathrm{V_{thr}} \tag{1}$$

$$V_t = V_{t-1} + V_t^{\mathrm{in}} - \mathrm{V_{thr}} \cdot \Theta(V_{t-1} + V_t^{\mathrm{in}}, \mathrm{V_{thr}}; S_{t-1}), S_t = S_{t-1} + \Theta(V_{t-1} + V_t^{\mathrm{in}}; \mathrm{V_{thr}}; S_{t-1}) \tag{2}$$

Here, V_t and V_t^{in} denote the membrane potential and input voltage at time-step t, respectively; $\mathrm{V_{thr}}$ is the firing threshold; and S_t is the spike tracer that records cumulative spikes within $[S_{\min}, S_{\max}]$. The function $\Theta(\cdot)$ determines the firing event and produces the output spike O_t, while Eq. (2) updates the membrane potential and spike tracer.

Beyond conversion fidelity, ST-BIF satisfies the *Law of Charge Conservation* (LoCC) (You et al., 2025), where cumulative output spikes are determined solely by the total accumulated input. As shown in Fig. 1(b), this property enables neuron-level temporal decoupling: **freezing a neuron at the current time-step only changes future firing schedules, without altering the total output spikes**. This temporal invariance provides the theoretical basis for asynchronous and intermittent execution.

Bottlenecks in Scaling Asynchronous SNNs to LLMs. Although ST-BIF provides neuron-level asynchronous equivalence, it does not remove synchronization induced by LLM operators such as RMSNorm and Softmax:

$$\mathrm{RMSNorm}(x) = \left(x / \sqrt{\tfrac{1}{d} \textstyle\sum_{i=1}^{d} x_i^2 + \epsilon} \right) \odot \gamma, \quad [\mathrm{Softmax}(x)]_i = \tfrac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}. \tag{3}$$

As shown in Eq. (3), both operators depend on global statistics: RMSNorm requires the root mean square across channels, while Softmax requires normalization over all entries. Prior spiking LLMs (You et al., 2024; Zheng et al., 2026) adopt differential-based spiking operators, such as Spiking RMSNorm and Spiking Softmax, to emulate these functions during SNN inference:

$$\boldsymbol{X}_t^{\mathrm{acc}} = \boldsymbol{X}_{t-1}^{\mathrm{acc}} + \boldsymbol{X}_t, \quad \boldsymbol{O}_t^{\mathrm{acc}} = \sigma(\boldsymbol{X}_t^{\mathrm{acc}}), \quad \boldsymbol{O}_t = \boldsymbol{O}_t^{\mathrm{acc}} - \boldsymbol{O}_{t-1}^{\mathrm{acc}}, \tag{4}$$

where σ denotes Softmax or RMSNorm, $\boldsymbol{X}_t$ is the input at time-step t, $\boldsymbol{X}_t^{\mathrm{acc}}$ is the accumulated input, and $\boldsymbol{O}_t$ is the emitted output. However, since $\sigma(\boldsymbol{X}_t^{\mathrm{acc}})$ still requires global statistics at every time-step, these operators impose layer-wise synchronization and prevent true asynchronous execution.

3 METHOD

To scale asynchronous SNNs to LLMs, our method proceeds in three steps. **Step-①: Ready-to-convert LLM construction.** We remove synchronization-intensive operators by replacing RMSNorm with Hardtanh Dynamic Normalization (HDN) and Softmax Attention with Normalized Linear Attention (NLA), yielding *Async-LLM*, a synchronization-free architecture compatible with asynchronous SNN conversion. **Step-②: Coarse-to-fine ANN-to-SNN conversion.** We convert *Async-LLM* into its equivalent *Async-SNN* through a coarse-to-fine pipeline, including *coarse calibration*, *fine-grained SFT-based QAT*, and *equivalent ST-BIF conversion*, to preserve task performance after conversion. **Step-③: Greedy adaptive firing.** We propose *greedy adaptive firing* (GAF), which exploits the asynchronism of *Async-SNN* to dynamically regulate spike firing and achieve a favorable budget–performance trade-off under instantaneous power constraints.

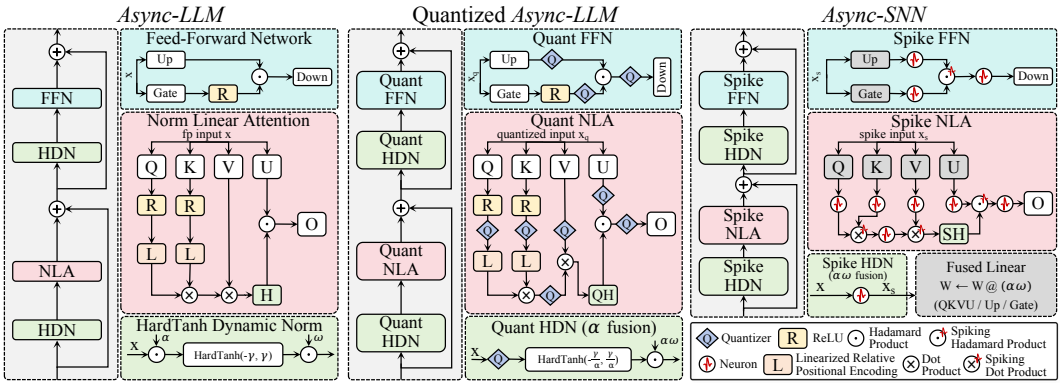


Figure 2: *Async-LLM* removes synchronization-intensive operators and provides a ready-to-convert architecture for asynchronous SNN conversion (Section 3.1). Through the coarse-to-fine ANN-to-SNN conversion pipeline (Section 3.2), the quantized *Async-LLM* is transformed into an equivalent *Async-SNN*. H, QH, and SH denote HDN, Quant HDN, and Spike HDN, respectively.

3.1 Ready-to-convert LLM construction

Normalization Replacement. We replace RMSNorm with an asynchronous-conversion-friendly normalization to enable asynchronous spiking LLM conversion. The replacement should satisfy two requirements: *1)* it should avoid synchronous statistics (*e.g.*, channel-wise mean/variance) and differential-based operators during SNN inference; and *2)* its quantized form should be mathematically equivalent to the ST-BIF neuron, enabling lossless conversion from quantized ANNs to asynchronous SNNs. Inspired by (Zhu et al., 2025), we adopt Hardtanh Dynamic Normalization (HDN):

$$\mathrm{HDN}(x) = w \cdot \mathrm{HardTanh}(\alpha x, -\gamma, \gamma) = w \cdot \mathbf{C}(\alpha x, -\gamma, \gamma), \tag{5}$$

where $x \in \mathbb{R}^{n\times d}$, $\alpha \in \mathbb{R}^1$ and $w \in \mathbb{R}^d$ are learnable parameters, $\gamma \in \mathbb{R}^1$ is fixed, and $\mathbf{C}(\cdot)$ denotes the clipping function. Since HDN is element-wise and independent of channel-wise statistics, it satisfies the constraint *1)* and allows converted spiking neurons to operate independently.

To verify that HDN satisfies the constraint *2)*, we use the proven equivalence between the ST-BIF neuron and the quantization function $\mathbf{Q}$ (You et al., 2024; Yan et al., 2026):

$$\sum_t^T \text{ST-BIF}(x_t) = \mathbf{Q}(\textstyle\sum_t^T x_t) = \mathbf{C}\Big(\mathbf{R}\Big(\sum_t^T x_t/s\Big), Z_{\min}, Z_{\max}\Big) \cdot s, \tag{6}$$

where $\mathbf{R}(\cdot)$ is the rounding function, s is the quantization scale, and $Z_{\min}, Z_{\max} \in \mathbb{Z}$ are the lower and upper integer bounds, respectively. Combining Eq. (5) and Eq. (6), the quantized HDN is:

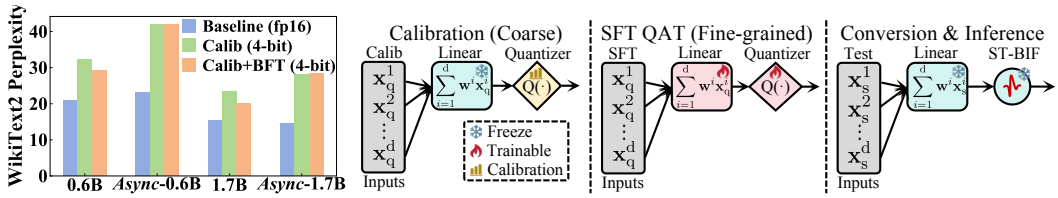
$$\mathrm{HDN}(\mathbf{Q}(\textstyle\sum_t^T x_t)) = \alpha w \cdot \mathbf{C}\Big(\mathbf{C}\Big(\mathbf{R}\Big(\sum_t^T x_t/s\Big), Z_{\min}, Z_{\max}\Big) \cdot s, -\gamma/\alpha, \gamma/\alpha\Big). \tag{7}$$

By setting $|Z|_{\max} = \max(|Z_{\min}|, |Z_{\max}|)$ and constraining $|s|_{\max} \le \gamma/(\alpha|Z|_{\max})$, the outer clipping is inactive, yielding:

$$\mathrm{HDN}(\mathbf{Q}(\textstyle\sum_t^T x_t)) = \overbrace{(\alpha w)}^{\text{fuse to linear}} \cdot \underbrace{\mathbf{C}\Big(\mathbf{R}\Big(\sum_t^T x_t/s\Big), Z_{\min}, Z_{\max}\Big) \cdot s}_{\text{equal to ST-BIF}}; \quad \text{s.t.} \quad |s|_{\max} \le \tfrac{\gamma}{\alpha|Z|_{\max}}. \tag{8}$$

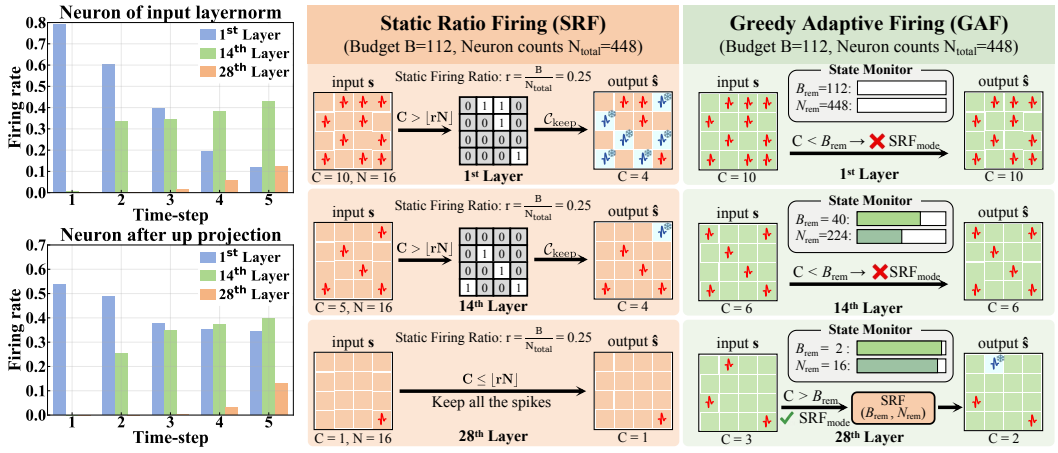
As shown in Fig. 2, the factor (αw) can be equivalently fused into the subsequent linear projection, while the remaining term matches the ST-BIF-equivalent quantization function in Eq. (6). Therefore, HDN satisfies Constraint-2 and enables lossless conversion to the asynchronous ST-BIF neuron.

Attention Replacement. We further replace Softmax Attention with Normalized Linear Attention (NLA) to remove the synchronization-intensive softmax operator. As shown in Fig. 2, NLA projects the input $\mathbf{X}$ into query Q, key K, value V, and gate U using $\mathbf{W}_\mathrm{q}$, $\mathbf{W}_\mathrm{k}$, $\mathbf{W}_\mathrm{v}$, and $\mathbf{W}_\mathrm{u}$. Unlike Softmax Attention, which requires global normalization over the token dimension, NLA applies ReLU to Q and K to obtain non-negative attention features, and incorporates Linearized Relative Positional Encoding (LRPE) (Qin et al., 2024) to preserve positional information. The attention response is computed as $(\mathrm{Q_l K_l^T})\mathrm{V}$, where $\mathrm{Q_l}$ and $\mathrm{K_l}$ are the LRPE-encoded query and key. The



(a) Comparison of various precision. (b) Coarse-to-fine conversion pipeline.

Figure 3: **Left: WikiText2 perplexity comparison across different precision settings.** 0.6B/1.7B and *Async*-0.6B/1.7B denote Qwen3-0.6B/1.7B and *Async-LLM*-0.6B/1.7B, respectively. **Right: Coarse-to-fine conversion pipeline**, consisting of *coarse calibration* → *fine-grained SFT QAT* → *conversion*, which converts *Async-LLM* into its corresponding *Async-SNN* counterpart.



(a) Firing rate visualization. (b) Visualization on static ratio firing (SRF) and greedy adaptive firing (GAF).

Figure 4: **Left: Firing-rate visualization of *Async-SNN***, showing that shallow-layer neurons dominate spike firing at early time-steps. **Right: Illustration of static ratio firing (SRF) and greedy adaptive firing (GAF)**, where GAF dynamically reallocates the remaining firing budget across layers to preserve informative spikes while satisfying the global budget constraint.

resulting feature is normalized by HDN, modulated by the gate U, and projected by $\mathbf{W}_o$ to produce O. The NLA computation is summarized as:

$$\begin{aligned} &\mathrm{Q} = \mathrm{ReLU}\left(\mathbf{X}\mathbf{W}_q\right), \quad \mathrm{K} = \mathrm{ReLU}\left(\mathbf{X}\mathbf{W}_k\right), \quad \mathrm{V} = \mathbf{X}\mathbf{W}_v, \quad \mathrm{U} = \mathbf{X}\mathbf{W}_u; \\ &\mathrm{Q}_l = \mathrm{LRPE}\left(\mathrm{Q}\right), \quad \mathrm{K}_l = \mathrm{LRPE}\left(\mathrm{K}\right), \quad \mathrm{O} = \mathbf{W}_o\big(\mathrm{HDN}\big((\mathrm{Q}_l\mathrm{K}_l^{\mathrm{T}})\mathrm{V}\big) \odot \mathrm{U}\big). \end{aligned} \tag{9}$$

By replacing RMSNorm and Softmax Attention with HDN and NLA, *Async-LLM* removes synchronous operators and becomes ready for conversion into its asynchronous spiking counterpart.

3.2 Coarse-to-fine ANN-to-SNN Conversion Pipeline

Given the equivalence between quantization and ST-BIF neurons, an effective pipeline for converting quantized *Async-LLM* into SNNs is essential. Recent studies (Shao et al., 2024; Chen et al., 2024; 2025) show that calibration (Calib) combined with block-wise fine-tuning (BFT) improves low-bit quantization by sequentially optimizing transformer blocks with a mean-squared-error objective. However, as shown in Fig. 3(a), 4-bit Calib+BFT performs well on open-source Qwen3-0.6B/1.7B but severely degrades on *Async-LLM*. We attribute this degradation to the *architectural divergence between Async-LLM and vanilla Qwen3*. Async-LLM replaces synchronization-inducing operators with HDN and NLA for asynchronous SNN conversion, making conventional Calib+BFT insufficient for preserving task performance.

To address this limitation, we propose a *coarse-to-fine ANN-to-SNN conversion pipeline*, as illustrated in Fig. 3(b). In the coarse stage, we insert quantizers into *Async-LLM* to obtain the quantized *Async-LLM* shown in Fig. 2, and perform layer-wise calibration to initialize quantization parameters. In the fine-grained stage, we apply supervised fine-tuning based quantization-aware training (SFT QAT) to jointly optimize model weights and quantization parameters on SFT data. Finally, we replace the quantizers with ST-BIF neurons, converting the quantized *Async-LLM* into an equivalent *Async-SNN*.

Algorithm 1 Static Ratio Firing (SRF)

Require: Input spikes $\mathbf{s}$, Budget B, Neuron counts $\mathrm{N}_{\mathrm{total}}$
Ensure: Output spikes $\hat{\mathbf{s}}$

$\mathcal{C}=\{i:\mathbf{s}^i\neq 0\}$, $\mathrm{N}=|\mathbf{s}|$, $\mathrm{C}=|\mathcal{C}|$
$\mathrm{r}=\mathrm{B}/\mathrm{N}_{\mathrm{total}}, \mathrm{K}=\lfloor \mathrm{rN} \rfloor$
if $\mathrm{C}\leq \mathrm{K}$ **then**
  $\hat{\mathbf{s}}=\mathbf{s}$
else
  $\mathcal{C} \xrightarrow{\text{Partition}} \{\mathcal{C}_{\text{keep}}, \mathcal{C}_{\text{drop}}\}$
  $|\mathcal{C}_{\text{keep}}|=\mathrm{K}$
  $|\mathcal{C}_{\text{drop}}|=\mathrm{C}-\mathrm{K}$
  for $i\in\mathcal{C}_{\text{keep}}$ **do**
    $\hat{\mathbf{s}}^i=\mathbf{s}_i$
  end for
  for $i\in\mathcal{C}_{\text{drop}}$ **do**
    $\hat{\mathbf{s}}^i=0$
  end for
end if

Algorithm 2 Greedy Adaptive Firing (GAF)

Require: Budget B, Module counts M, Neuron counts $\{\mathrm{N}_m^{\Theta}\}_{m=1}^{\mathrm{M}}$, Input spikes $\{\mathbf{s}_m\}_{m=1}^{\mathrm{M}}$
Ensure: Output spikes $\{\hat{\mathbf{s}}_m\}_{m=1}^{\mathrm{M}}$

$B_{\mathrm{rem}}=\mathrm{B},\ N_{\mathrm{rem}}=\sum_{m=1}^{\mathrm{M}}\mathrm{N}_m^{\Theta}$
$\mathrm{SRF}_{\mathrm{mode}}=\texttt{False}$
for $m=1$ **to** M **do**
  $\mathrm{C}_m=|\{i:\mathbf{s}_m^i\neq 0\}|$
  if $\neg\mathrm{SRF}_{\mathrm{mode}}$ **and** $\mathrm{C}_m\leq B_{\mathrm{rem}}$ **then**
    # Sufficient budget → Greedy mode
    $\hat{\mathbf{s}}_m=\mathbf{s}_m$
  else
    # Insufficient budget → SRF mode
    $\mathrm{SRF}_{\mathrm{mode}}=\texttt{True}$
    $\hat{\mathbf{s}}_m=\mathrm{SRF}(\mathbf{s}_m, B_{\mathrm{rem}}, N_{\mathrm{rem}})$
  end if
  $B_{\mathrm{rem}}=\max(0, B_{\mathrm{rem}}-|\{i:\hat{\mathbf{s}}_m^i\neq 0\}|)$
  $N_{\mathrm{rem}}=\max(0, N_{\mathrm{rem}}-\mathrm{N}_m^{\Theta})$
end for

3.3 Greedy Adaptive Firing Strategy

Given the asynchronous computing paradigm in Fig. 1(a), a global firing budget B constrains the total number of active neurons C at each time-step. A simple solution is Static Ratio Firing (SRF) in Algorithm 1, which assigns each layer the same global budget ratio $r=\mathrm{B}/\mathrm{N}_{\mathrm{total}}$ and preserves at most $\lfloor rN \rfloor$ active neurons, where N is the number of neurons in the current layer. However, SRF ignores the temporal and layer-wise imbalance of spike activity. As shown in Fig. 4(a), shallow layers (*e.g.*, the 1st layer) dominate firing at early time-steps, making static suppression likely to discard informative early spikes and hinder subsequent propagation.

To better exploit the asynchronous computing paradigm of *Async-SNN*, we propose Greedy Adaptive Firing (GAF), as illustrated in Fig. 4(b) and Algorithm 2. Unlike SRF, which enforces a fixed per-layer budget, GAF dynamically allocates the remaining budget during layer-wise inference. Specifically, GAF maintains the remaining budget B_{rem} and the remaining neuron count N_{rem}. For each layer m, if the current spike count C_m does not exceed B_{rem}, GAF greedily preserves all active spikes, allowing informative shallow-layer responses to pass through without unnecessary suppression. When the remaining budget becomes insufficient, GAF switches to SRF mode and applies $\mathrm{SRF}(\mathbf{s}_m, B_{\mathrm{rem}}, N_{\mathrm{rem}})$ to the current and subsequent layers, ensuring that the global budget constraint is still satisfied. In this way, GAF dynamically redistributes the firing budget across layers, preserving critical shallow-layer spikes while constraining total activity, thus achieving a favorable budget–performance trade-off under instantaneous power budget constraints.

4 Experiments

4.1 Setups

Training Details. We build *Async-LLM*-0.6B/1.7B by replacing RMSNorm, softmax attention, and SiLU in Qwen3-0.6B/1.7B (Yang et al., 2025) with HDN, NLA, and ReLU, respectively. The models are pretrained on 100B tokens from UltraFineWeb (Wang et al., 2025; Team et al., 2025) using a Warmup-Stable scheduler (Hu et al., 2024). We then apply the coarse-to-fine ANN-to-SNN conversion pipeline: *1) Calibration*: insert 4-bit activation quantizers into *Async-LLM* (Fig. 2) and calibrate their parameters on UltraFineWeb samples; *2) SFT QAT*: optimize the 4-bit quantized model with supervised fine-tuning based quantization-aware training on 10B SFT tokens using a decay scheduler (Hu et al., 2024); and *3) Conversion*: replace activation quantizers with equivalent ST-BIF neurons to obtain *Async-SNN*. More training details are specified in Section A1.

Table 1: Experimental results. Best results are highlighted in **bold**. B/N_{total} denotes the firing budget ratio, which limits the fraction of neurons allowed to be active at each time-step. Asyn indicates whether the SNN supports asynchronous execution. T, WT2, HS, and WG denote time-steps, WikiText2, HellaSwag, and WinoGrande, respectively.

Method	Category	T	Asyn	$\frac{B}{N_{\text{total}}}$	Perplexity(↓)		Zero-shot Accuracy(↑)					
					WT2	C4	PIQA	ARC-e	ARC-c	HS	WG	Avg.
Qwen3-0.6B	ANNs	N/A	N/A	N/A	19.15	23.81	67.68	56.02	34.13	47.29	56.04	52.23
Async-LLM-0.6B					23.09	22.68	69.86	58.21	31.91	43.84	49.80	50.72
SpikedAttention	SNNs	16	✗	N/A	32.48	36.78	65.76	44.48	29.69	44.24	50.97	47.03
SpikeLLM			✗		29.12	33.08	65.32	48.56	29.43	45.24	49.87	47.68
SpikingLLM			✗		29.94	34.46	64.09	42.68	**31.40**	**46.06**	**54.78**	47.80
Async-SNN (ours)			✓		**28.37**	**27.62**	**66.76**	**55.30**	30.29	40.34	51.99	**48.94**
SpikedAttention	SNNs	16	✗	10%	>100	>100	51.58	25.34	26.88	25.35	51.44	36.06
SpikeLLM			✗		>100	>100	51.25	25.75	25.89	24.76	49.83	35.50
SpikingLLM			✗		>100	>100	52.39	25.25	25.94	26.28	49.57	35.89
Async-SNN (ours)			✓		**28.46**	**27.67**	**66.70**	**55.64**	**30.20**	**40.30**	**51.67**	**48.90**
Qwen3-1.7B	ANNs	N/A	N/A	N/A	15.63	18.12	72.20	69.70	43.17	60.44	61.09	61.32
Async-LLM-1.7B					14.56	16.14	74.92	69.65	39.42	59.70	57.22	60.18
SpikedAttention	SNNs	16	✗	N/A	21.64	24.12	66.73	62.46	35.82	48.73	49.84	52.72
SpikeLLM			✗		21.32	24.23	67.01	63.52	35.12	47.04	50.67	52.67
SpikingLLM			✗		21.39	25.14	68.55	59.76	34.81	48.81	53.51	53.09
Async-SNN (ours)			✓		**18.92**	**21.18**	**70.42**	**64.56**	**35.97**	**51.46**	**57.32**	**55.95**
SpikedAttention	SNNs	16	✗	10%	>100	>100	51.80	25.76	26.11	25.24	50.20	35.82
SpikeLLM			✗		>100	>100	51.42	25.48	25.97	25.13	49.82	35.56
SpikingLLM			✗		>100	>100	50.76	25.04	25.85	26.07	49.49	35.44
Async-SNN (ours)			✓		**18.97**	**21.26**	**70.35**	**64.39**	**36.20**	**51.49**	**57.24**	**55.93**

Baselines. We compare *Async-SNN* with three representative spiking LLM baselines: **SpikedAttention** (Hwang et al., 2024), which enables fully spike-driven self-attention through trace-driven matrix multiplication and spike shifting; **SpikeLLM** (Xing et al., 2024), which combines generalized integrate-and-fire neurons with saliency-based spike encoding; and **SpikingLLM** (Zheng et al., 2026), which adopts distribution-aware multi-granularity phase coding to reduce conversion errors. For fair comparison, we extend SpikedAttention, SpikeLLM, and SpikingLLM to the open-source Qwen3-0.6B/1.7B, and apply 4-bit activation quantization to all baselines to maintain consistency.

Evaluation Metrics. We evaluate language modeling with perplexity (PPL) on WikiText2 (Merity et al., 2016) and C4 (Raffel et al., 2023). For zero-shot reasoning, we evaluate *Async-SNN* on PIQA (Bisk et al., 2019), ARC-easy/challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2019). We report accuracy (*acc*) for WinoGrande and normalized accuracy (*acc_norm*) for the other datasets (Lin et al., 2024). To emulate strict power budgets, we set $B/N_{\text{total}} = 10\%$ in Table 1, allowing at most 10% of neurons to be active at each time-step. Following (Zhou et al., 2023), we estimate per-token multiply-accumulate operations ($\#\text{MACs}$) for *Async-LLM*, synaptic operations ($\#\text{SOPs}$) for SNNs, and the corresponding operation-level energy $\mathbf{E}_{\text{est}}$. We further report hardware-simulated energy $\mathbf{E}_{\text{sim}}$, leveraging Eyeriss (Chen et al., 2016) for 4-bit *Async-LLM* and SANA-FE (Boyle et al., 2025) with a 32-chip Loihi configuration (Davies et al., 2018) for *Async-SNN*. More evaluation details are specified in Section A1.

4.2 Main Results

We report the main results in Table 1. *Async-SNN* achieves the best overall performance in both unconstrained and budget-constrained settings, yielding lower perplexity while maintaining competitive zero-shot accuracy. Without budget constraints, *Async-SNN* consistently outperforms Qwen3-based spiking baselines, including SpikedAttention, SpikeLLM, and SpikingLLM, showing that the adopted asynchronous operators (*e.g.*, HDN and NLA) preserve representational fidelity after ANN-to-SNN conversion. More importantly, under a stringent 10% firing budget, where at most 10% of neurons can be active at each time-step, all baselines collapse with perplexity exceeding 100 and zero-shot accuracy approaching near-random levels. In contrast, *Async-SNN* maintains stable performance. This robustness comes from enforcing asynchronism at both the neuronal and architectural levels: unlike prior spiking LLMs that retain synchronization-intensive operators such as RMSNorm and softmax, *Async-SNN* eliminates these operators by design. Additional comparisons with 4-bit quantized Mamba (Gu & Dao, 2023) and RWKV (Peng et al., 2023) in Section A7 further demonstrate a competitive performance–efficiency trade-off against efficient quantized ANN baselines.

Table 2: #MACs, #SOPs, $\mathbf{E}_{\text{est}}$ and $\mathbf{E}_{\text{sim}}$ of *Async-LLM*† and *Async-SNN*. Best results are marked in **bold**, and runner-up results are shaded in gray. WT2 PPL and Avg Acc. are WikiText2 perplexity and average zero-shot accuracy, respectively. † denotes 4-bit activation quantization.

Model	PS(B)	Category	B/N_{total}	WT2 PPL(↓)	Avg Acc.(↑)	#MACs(↓) (G / Token)	#SOPs(↓) (G / Token)	$\mathbf{E}_{\text{est}}$(↓) (mJ / Token)	$\mathbf{E}_{\text{sim}}$(↓) (mJ / Token)
Async-LLM†	0.6	ANN	N/A	28.53	48.75	0.76	N/A	3.50	6.98
Async-SNN		SNN	N/A	**28.37**	**48.94**	N/A	3.77	3.39	6.47
+ SRF			10%	66.48	42.27		**1.49**	**1.34**	**3.76**
+ GAF				28.46	48.90		2.73	2.46	4.08
Async-LLM†	1.7	ANN	N/A	19.14	55.67	2.35	N/A	10.81	14.56
Async-SNN		SNN	N/A	**18.92**	**55.95**	N/A	10.55	9.50	11.83
+ SRF			10%	45.42	46.74		**3.95**	**3.56**	**6.53**
+ GAF				18.97	55.93		7.72	6.95	9.76

Table 3: Ablation on operators replacement. Baseline denotes Qwen3-0.6B.

Method	HDN	NLA	Loss(↓)	PPL(↓)
Baseline	✗	✗	**2.49**	**16.89**
Norm-replaced	✓	✗	**2.49**	16.97
Attn-replaced	✗	✓	2.54	17.53
Async-LLM	✓	✓	2.52	17.16

Table 4: Ablation on coarse-to-fine ANN-to-SNN conversion pipeline. *AS* denotes *Async-SNN*.

Model	Calib	BFT	SFT QAT	WT2 PPL(↓)	Avg Acc.(↑)
AS-0.6B	✓	✗	✗	41.95	46.07
	✓	✓	✗	42.06	45.94
	✓	✗	✓	**28.37**	**48.94**
AS-1.7B	✓	✗	✗	28.22	50.92
	✓	✓	✗	28.41	50.73
	✓	✗	✓	**18.92**	**55.95**

We further report the estimated #MACs, #SOPs, operation-level energy $\mathbf{E}_{\text{est}}$, and hardware-simulated energy $\mathbf{E}_{\text{sim}}$ in Table 2. *Async-SNN* replaces dense #MACs with spike-driven #SOPs, reducing energy while maintaining performance comparable to *Async-LLM*†, with consistent trends under both energy estimation and hardware simulation. With a 10% firing budget, GAF further reduces #SOPs and energy with negligible performance degradation; for example, at 1.7B scale, it reduces $\mathbf{E}_{\text{sim}}$ from 11.83 to 9.76 mJ/token relative to unconstrained *Async-SNN*, and by 33.0% relative to *Async-LLM*†. In contrast, SRF achieves lower energy but incurs substantial degradation in perplexity and accuracy. These results demonstrate that GAF provides a favorable budget–performance trade-off by selectively constraining spike activity while preserving model performance.

Furthermore, we demonstrate the advantages of *Async-SNN* in Fig. 5 and Fig. 6(a). As shown in Fig. 5, under GAF, *Async-SNN* maintains convergent normalized accuracy under the firing-budget constraint, whereas SpikingLLM fails to do so. Fig. 6(a) compares the layer-wise cosine similarity between outputs without GAF (y) and with GAF (y_G) for Spike RMSNorm in SpikingLLM and Spike HDN in *Async-SNN*. Spike HDN maintains near-identity similarity across layers, confirming its compatibility with asynchronous execution, while Spike RMSNorm does not. Additional hardware simulated results in Section A6 further demonstrate that *Async-SNN* reduces both energy consumption and inference latency under the 10% GAF budget.

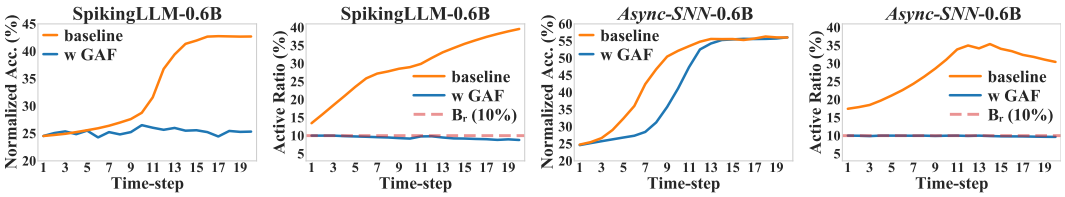
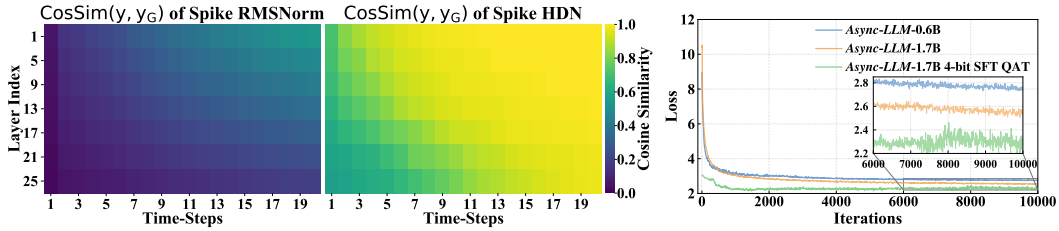


Figure 5: Normalized accuracy and active ratio on ARC-easy (Clark et al., 2018) across time-steps for SpikingLLM-0.6B and *Async-SNN*-0.6B, evaluated under two settings: baseline without firing control (w/o strategy) and with GAF. B_r denotes the firing budget ratio, which is set to 10%.

4.3 Ablation Study

Ablation on Operator Replacement. We ablate operator replacement using 15B tokens sampled from SlimPajama (Soboleva et al., 2023), comparing Qwen3-0.6B with Norm-replaced, Attn-replaced, and *Async-LLM*-0.6B variants. Norm-replaced, Attn-replaced, and *Async-LLM* achieve performance comparable to the Qwen3-0.6B baseline, validating the effectiveness of HDN and NLA as asynchronous operators. The loss curves in Fig. 6(b) further show that *Async-LLM* converges stably and



(a) Layer-wise $\mathrm{CosSim}(\mathrm{y}, \mathrm{y_G})$ of Spike RMSNorm and Spike HDN. (b) Loss curves of different models.

Figure 6: Left: Layer-wise cosine similarity $\mathrm{CosSim}(\mathrm{y}, \mathrm{y_G})$ of Spike RMSNorm and Spike HDN, y and $\mathrm{y_G}$ denote output without GAF and with GAF, respectively. Right: Loss curves of Qwen3-0.6B-baseline, Attn-replaced, Norm-replaced, and *Async-LLM*-0.6B.

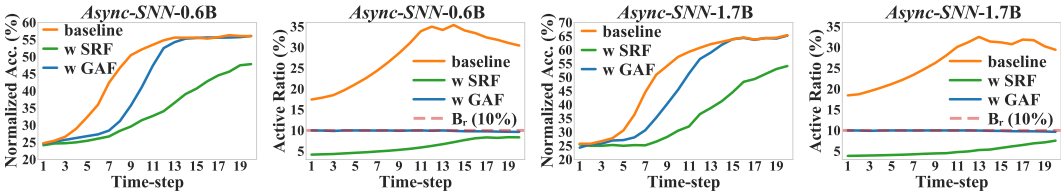


Figure 7: Visualization on normalized accuracy and active ratio on ARC-easy (Clark et al., 2018) over time-steps of *Async-SNN*-0.6B/1.7B under three settings: baseline (w/o strategy), SRF, and GAF. Note that $\mathrm{B_r}$ denotes the budget ratio, which is set to 10%.

remains close to the baseline throughout training. Additional in-context and long-context evaluations in Section A3 further verify that *Async-LLM* maintains performance comparable to Qwen3 baseline.

Ablation on Coarse-to-Fine ANN-to-SNN Conversion. We ablate the coarse-to-fine conversion pipeline in Table 4. Coarse calibration alone yields poor performance, while adding block-wise fine-tuning (BFT) brings no improvement and can even degrade performance, indicating that conventional Calib+BFT is ineffective for *Async-LLM*. In contrast, incorporating SFT QAT substantially reduces perplexity (*e.g.*, from 41.95 to 28.37 on *Async-SNN*-0.6B) and improves average zero-shot accuracy. These results validate the necessity of our coarse-to-fine pipeline, especially the fine-grained SFT QAT stage, for converting *Async-LLM* into an effective *Async-SNN*.

Ablation on Firing Strategies. We evaluate *Async-SNN*-0.6B/1.7B on ARC-easy under three settings: without strategy, with static ratio firing (SRF), and with greedy adaptive firing (GAF). As shown in Fig. 7, both SRF and GAF substantially reduce the active ratio and enforce the firing budget. However, SRF applies a fixed per-layer firing ratio, which indiscriminately suppresses shallow-layer spikes and under-utilizes the budget, leading to slower convergence in normalized accuracy. In contrast, GAF dynamically allocates the firing budget across layers: it greedily preserves informative shallow-layer spikes and switches to SRF only when the remaining budget becomes tight. As reported in Table 2, this adaptive mechanism achieves a favorable budget–performance trade-off, maintaining strong accuracy while reducing energy consumption. Additional firing-budget sweeps from 5% to 25% in Section A4 further verify that 10% is the lowest evaluated budget that preserves comparable performance. We also verify in Section A5 that random spike selection in GAF achieves comparable performance to membrane-state-based selection.

5 Conclusion

Asynchronous computing is critical for deploying SNNs under stringent and fluctuating power budgets. In this work, we propose *Async-SNN*, the first framework that scales asynchronous SNNs to billion-parameter LLMs. We construct *Async-LLM* with asynchronous-friendly operators, develop a coarse-to-fine ANN-to-SNN conversion pipeline to preserve task performance, and introduce greedy adaptive firing to dynamically regulate spike activity under constrained power budgets. Extensive experiments show that *Async-SNN* achieves strong performance while strictly satisfying the given firing budget. We further expect *Async-SNN* to benefit from learning-based SNN methods, potentially improving performance and energy efficiency.

AI USE STATEMENT

Large Language Models (LLMs) were used solely for language polishing, including sentence rephrasing, grammar checking, and improving readability. All research ideas, methodological design, code implementation, experiments, and analyses were independently developed and conducted by the authors. The authors take full responsibility for the content of the manuscript.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

James A Boyle, Mark Plagge, Suma George Cardwell, Frances S Chance, and Andreas Gerstlauer. Sana-fe: Simulating advanced neuromorphic architectures for fast exploration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 44(8):3165–3178, 2025.

Tom Brown and Benjamin et al. Mann. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models, 2024. URL https://arxiv.org/abs/2407.11062.

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. URL https://arxiv.org/abs/2410.05265.

Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138, 2016.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Liu, Yonghao Zhu, Jiantao Ni, Xianzhi Du, Min Jiao, E. P. Raman, Shuo Zhang, Qi Dan, Jian Zeng, Ce Zhang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Sharegpt: A repository of conversations with chatgpt and gpt-4. https://github.com, 2023.

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, 2018. URL https://arxiv.org/abs/1803.05457.

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.

Shuiguang Deng, Di Yu, Changze Lv, Xin Du, Linshan Jiang, Xiaofan Zhao, Wentao Tong, Xiaoqing Zheng, Weijia Fang, Peng Zhao, et al. Edge intelligence with spiking neural networks. *arXiv preprint arXiv:2507.14069*, 2025.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

Graham Gobieski, Brandon Lucia, and Nathan Beckmann. Intelligence beyond the edge: Inference on intermittent embedded systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 199–213, 2019.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

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

Roel Koopman, Amirreza Yousefzadeh, Mahyar Shahsavari, Guangzhi Tang, and Manolis Sifalakis. Exploring the limitations of layer synchronization in spiking neural networks. *Transactions on Machine Learning Research*, 2025.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving, 2024. URL https://arxiv.org/abs/2405.04532.

Brandon Lucia and Benjamin Ransford. A simpler, safer programming and execution model for intermittent systems. *ACM SIGPLAN Notices*, 50(6):575–585, 2015.

Mingsong Lv and Enyu Xu. Deep learning on energy harvesting iot devices: Survey and future challenges. *IEEE Access*, 10:124999–125014, 2022.

Pietro M Ferreira, Siqi Wang, Yueyuan Gao, and Aziz Benlarbi-Delai. A comparative review of deep and spiking neural networks for edge ai neuromorphic circuits. *Frontiers in Neuroscience*, 19: 1676570, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Alessandro Montanari, Manuja Sharma, Dainius Jenkus, Mohammed Alloulah, Lorena Qendro, and Fahim Kawsar. eperceptive: energy reactive embedded intelligence for batteryless sensors. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pp. 382–394, 2020.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pp. 14048–14077, 2023.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and Yiran Zhong. Transnormerllm: A faster and better large language model with improved transnormer, 2024. URL https://arxiv.org/abs/2307.14995.

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, 2023. URL https://arxiv.org/abs/1910.10683.

Nitin Rathi, Indranil Chakraborty, Adarsh Kosta, Abhronil Sengupta, Aayush Ankit, Priyadarshini Panda, and Kaushik Roy. Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware. *ACM Computing Surveys*, 55(12):1–49, 2023.

Adnan Sabovic, Michiel Aernouts, Dragan Subotic, Jaron Fontaine, Eli De Poorter, and Jeroen Famaey. Towards energy-aware tinyml on battery-less iot devices. *Internet of Things*, 22:100736, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

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. URL https://arxiv.org/abs/2308.13137.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023.

MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, et al. Minicpm4: Ultra-efficient llms on end devices. *arXiv preprint arXiv:2506.07900*, 2025.

Hugo Touvron and Thibaut et al. Lavril. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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.

Yudong Wang, Zixuan Fu, Jie Cai, Peijun Tang, Hongya Lyu, Yewei Fang, Zhi Zheng, Jie Zhou, Guoyang Zeng, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Ultra-FineWeb: Efficient data filtering and verification for high-quality llm training data, 2025.

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.

Jing Yan, Kang You, Zhezhi He, and Yaoyu Zhang. Determinism in the undetermined: Deterministic output in charge-conserving continuous-time neuromorphic systems with temporal stochasticity. *arXiv preprint arXiv:2603.15987*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature communications*, 15(1):4464, 2024.

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 Forty-First International conference on Machine Learning (ICML)*, 2024.

Kang You, Ziling Wei, Jing Yan, Boning Zhang, Qinghai Guo, Yaoyu Zhang, and Zhezhi He. Vistream: Improving computation efficiency of visual streaming perception via law-of-charge-conservation inspired spiking neural network. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 8796–8805, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *CoRR*, abs/1905.07830, 2019. URL http://arxiv.org/abs/1905.07830.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Guanlei Zhang, Lei Feng, Fanqin Zhou, Zhixiang Yang, Qiyang Zhang, Alaa Saleh, Praveen Kumar Donta, and Chinmaya Kumar Dehury. Spiking neural networks in intelligent edge computing. *IEEE Consumer Electronics Magazine*, 14(4):66–75, 2024.

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

Hanyuan Zheng, Haozhen Zhang, Tianshuo Chen, Zhaogeng Liu, Yi Chang, and Bin Gu. Distribution-aware multi-granularity phase coding: Towards lower conversion error for spike-driven large language models. In *International Conference on Learning Representations*, volume 2026, pp. 80051–80076, 2026.

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

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization, 2025. URL https://arxiv.org/abs/2503.10622.

Appendix

A1 Implementation Details

Table A1: **Pretraining setups on *Async-LLM*-0.6B/1.7B.**

config	*Async-LLM*-0.6B	*Async-LLM*-1.7B
optimizer	AdamW	AdamW
base learning rate	4e-3	1e-3
warmup iterations	1000	1000
training sequence length	4096	4096
batch size per device	4	4
global batch size	1024	1024
gradient accumulation steps	4	4
training iterations	24000	24000
lr scheduler	warmup-stable	warmup-stable

***Async-LLM* pretraining.** We pretrain our *Async-LLM*-0.6B/1.7B on 100B tokens sampled from UltraFineWeb (Wang et al., 2025) with Megatron-LM (Shoeybi et al., 2019). The pretraining setups are specified in Table A1.

Table A2: **SFT QAT setups on *Async-LLM*-0.6B/1.7B.**

config	*Async-LLM*-0.6B	*Async-LLM*-1.7B
optimizer	AdamW	AdamW
weight learning rate	1e-3	1e-3
quant learning rate	1e-4	1e-4
warmup iterations	0	0
training sequence length	4096	4096
batch size per device	1	1
global batch size	256	256
gradient accumulation steps	4	4
training iterations	10000	10000
lr scheduler	decay	decay
min lr ratio	0.1	0.1

SFT QAT on *Async-LLM*. Table A2 summarizes the SFT QAT configurations for *Async-LLM*-0.6B/1.7B. Both models share the same hyperparameters: AdamW optimizer, weight learning rate of $1e-3$, training sequence length of 4096, global batch size of 256, gradient accumulation steps of 4, 10k training iterations, and a decay scheduler with a minimum learning rate ratio of 0.1. Notably, the quantization parameters are more sensitive during training; therefore, their learning rate is set one order of magnitude lower than that of the weights, *i.e.*, $1e-4$, to ensure stable fine-tuning. For SFT datasets, we construct a SFT datasets with 10B tokens consisting of UltraFineWeb (Wang et al., 2025), sharegpt-gpt4 (Chiang et al., 2023), ultrachat-200k (Ding et al., 2023) and tulu-3-sft-mixture (Lambert et al., 2024).

MACs, SOPs and energy estimation. We follow the procedure from Spikingformer (Zhou et al., 2023) to conduct energy evaluation between LLM and SNN, which is concluded as follows:

$$\begin{aligned} \mathrm{E_{LLM}} &= \#\mathrm{MACs} \times \mathrm{E_{MAC}} \\ \mathrm{E_{SNN}} &= \#\mathrm{SOPs} \times \mathrm{E_{SOP}} \end{aligned} \tag{A1}$$

$\#\mathrm{MACs}$ and $\#\mathrm{SOPs}$ refer to the total number of Multiply-ACcumulate and Synaptic OPerations, respectively. We assume that the MAC and AC operations are implemented on the 45nm hardware (Horowitz, 2014), where $\mathrm{E_{MAC}} = 4.6\mathrm{pJ}$ and $\mathrm{E_{SOP}} = 0.9\mathrm{pJ}$. We follow the procedure from SpikeZIP-TF (You et al., 2024) to calculate $\#\mathrm{MACs}$ and $\#\mathrm{SOPs}$, which is concluded as follows:

Algorithm 3 #MACs in Linear and Attention Layers

Input: Number of tokens N; Input feature dimension of linear layer D_{in}; Output feature dimension of linear layer D_{out}; Token dimension in attention layer D; Total layers L.
Output: #MACs

#MACs $\leftarrow 0$
for l = 1 to L **do**
  `# Calculate #MACs for linear layer`
  #MACs $\leftarrow$ #MACs + $N \times D_{in} \times D_{out}$
  `# Calculate #MACs for attention layer`
  #MACs $\leftarrow$ #MACs + $2 \times N \times N \times D$
end for
return #MACs

Algorithm 4 #SOPs in Linear and Attention Layers

Input: Number of tokens N; Input feature dimension of linear layer D_{in}; Output feature dimension of linear layer D_{out}; Token dimension in attention layer D; Spike count per time-step C_t; Total inference time-steps T; Total layers L.
Output: #SOPs

#SOPs $\leftarrow 0$
for l = 1 to L **do**
  for t = 1 to T **do**
    for k = 1 to C_t **do**
      `# Linear layer: Dout synapses activated per spike and 2 operations per spike for ST-BIF neuron`
      #SOPs $\leftarrow$ #SOPs + $D_{out} + 2$
      `# Attention layer: 2*N synapses activated per spike (dual matrix product) and 2 operations per spike for ST-BIF neuron`
      #SOPs $\leftarrow$ #SOPs + $2 \times N + 2$
    end for
  end for
end for
return #SOPs

A2 Visualization on Loss Curves

We visuaulize the loss curves of *Async-LLM*-0.6B/1.7B pretrain and *Async-LLM*-1.7B 4-bit SFT QAT in Fig. A1. As illustrated, during the pre-training phase, both *Async-LLM*-0.6B and *Async-LLM*-1.7B exhibit smooth and consistently decreasing loss curves without significant oscillations. This verifies the numerical stability of the proposed asynchronous-friendly operators (*e.g.*, HDN and NLA). Furthermore, we perform supervised fine-tuning with quantization-aware training (SFT QAT) on 4-bit activation quantization. The loss curve of *Async-LLM*-1.7B under 4-bit SFT QAT demonstrates a stable transition from the pre-trained state, with the loss continuing to decrease and converging to a relatively low value. This indicates that the training strategy effectively mitigates the accuracy degradation caused by activation quantization.

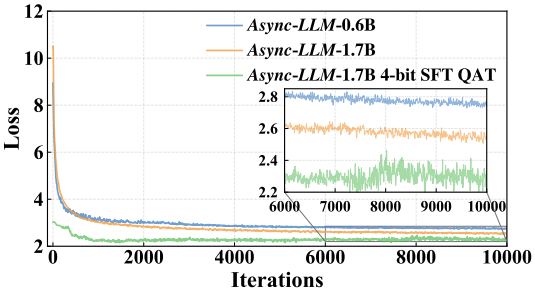


Figure A1: Loss curves on *Async-LLM*-0.6B/1.7B pretrain and *Async-LLM*-1.7B 4-bit SFT QAT.

Table A3: In-context and long-context evaluation of Qwen3 and *Async-LLM*. Best results within each model scale are in **bold**.

Model	0-shot ARC-Easy	0-shot HellaSwag	5-shot ARC-Easy	5-shot HellaSwag	BABILong Avg.
Qwen3-0.6B	55.51	**45.43**	61.47	43.14	33.40
Async-LLM-0.6B	**62.92**	43.84	**64.60**	**43.16**	**34.40**
Qwen3-1.7B	68.73	58.92	70.21	59.98	**45.40**
Async-LLM-1.7B	**70.21**	**59.70**	**71.31**	**60.12**	43.80

Table A4: Firing-budget sweep for the 0.6B models. † denotes 4-bit activation quantization. FR refers to firing ratio.

Method	Category	GAF	$\frac{B}{N_{\text{total}}}$	T	Avg. FR	WT2 PPL(↓)	Avg. Acc.(↑)	#MACs(↓) (G / Token)	#SOPs(↓) (G / Token)	$\mathbf{E}_{\text{est}}$(↓) (mJ / Token)
Async-LLM-0.6B†	QANN	N/A	N/A	N/A	N/A	28.53	48.75	0.76	N/A	3.50
SpikingLLM-0.6B	SNN	✗	N/A	16	35%	29.94	47.80	0.04	4.28	4.04
		✓	10%	16	10%	> 100	35.89	0.04	2.68	2.60
Async-SNN-0.6B	SNN	✗	N/A	16	27%	28.37	**48.94**	N/A	3.77	3.39
		✓	25%	16	25%	**28.35**	48.92	N/A	3.62	3.26
		✓	20%	16	20%	28.39	48.92	N/A	3.37	3.03
		✓	15%	16	15%	28.43	48.90	N/A	3.06	2.75
		✓	10%	16	10%	28.46	48.90	N/A	2.73	2.46
		✓	5%	16	5%	89.48	39.62	N/A	**2.32**	**2.09**

A3 In-Context and Long-Context Evaluation

To further evaluate whether replacing RMSNorm and Softmax attention with HDN and NLA preserves general language-modeling capabilities, we pretrain *Async-LLM* and the corresponding Qwen3 baselines under the same setting. We evaluate both 0-shot and 5-shot performance on ARC-Easy (Clark et al., 2018) and HellaSwag Zellers et al. (2019), together with long-context performance on BABILong (Kuratov et al., 2024). As shown in Table A3, *Async-LLM* achieves comparable performance to the Qwen3 baseline across both model scales. Averaged over the four in-context settings, *Async-LLM* obtains 53.63 and 65.34 accuracy at 0.6B and 1.7B, compared with 51.39 and 64.46 for Qwen3, respectively. On BABILong, *Async-LLM* achieves 34.40 versus 33.40 at 0.6B and 43.80 versus 45.40 at 1.7B. Together with the stable loss curves in Fig. 6(b), these results indicate that HDN and NLA preserve in-context and long-context capabilities at the evaluated model scales.

A4 Firing Budget Analysis

The budget ratio B/N_{total} specifies the fraction of neurons allowed to emit spikes under the instantaneous firing-budget constraint. To characterize the efficiency–performance trade-off, we sweep B/N_{total} over {5%, 10%, 15%, 20%, 25%}. Results for the 0.6B and 1.7B models are reported in Tables A4 and A5. As the budget decreases, the average firing ratio, #SOPs, and $\mathbf{E}_{\text{est}}$ decrease accordingly. A 10% budget is the lowest setting that retains performance close to unconstrained *Async-SNN*: reducing the budget to 5% increases WT2 perplexity from 28.46 to 89.48 at 0.6B and from 18.97 to 55.32 at 1.7B, with corresponding drops in average accuracy. We therefore use 10% as the operating point that balances efficiency gains and performance retention. Applying the same 10% GAF constraint to SpikingLLM substantially degrades performance, further showing the importance of the synchronization-free *Async-LLM* architecture under strict firing budgets.

A5 Ablation on Retained-Spike Selection Strategy

When the number of active spikes exceeds the remaining budget, GAF switches to SRF for the current and subsequent layers. In the default implementation, the required number of spikes is randomly sampled from the active candidates. We additionally compare this strategy with membrane-state-based selection, which ranks active neurons by their membrane states before retaining spikes. As shown in Table A6, the two strategies yield nearly identical results at both model scales. Random selection therefore avoids membrane-state ranking while preserving performance.

Table A5: Firing-budget sweep for the 1.7B models. † denotes 4-bit activation quantization. FR refers to firing ratio.

Method	Category	GAF	$\frac{B}{N_{\text{total}}}$	T	Avg. FR	WT2 PPL(↓)	Avg. Acc.(↑)	#MACs(↓) (G / Token)	#SOPs(↓) (G / Token)	$\mathbf{E}_{\text{est}}$(↓) (mJ / Token)
Async-LLM-1.7B†	QANN	N/A	N/A	N/A	N/A	19.14	55.67	2.35	N/A	10.81
SpikingLLM-1.7B	SNN	✗	N/A	16	38%	21.39	53.09	0.11	11.07	10.47
		✓	10%	16	10%	> 100	35.44	0.11	7.92	7.63
Async-SNN-1.7B	SNN	✗	N/A	16	31%	**18.92**	**55.95**	N/A	10.55	9.50
		✓	25%	16	25%	18.89	56.01	N/A	9.94	8.95
		✓	20%	16	20%	18.92	55.99	N/A	9.22	8.30
		✓	15%	16	15%	18.94	55.97	N/A	8.45	7.61
		✓	10%	16	10%	18.97	55.93	N/A	7.72	6.95
		✓	5%	16	5%	55.32	42.68	N/A	**5.68**	**5.11**

Table A6: Comparison of retained-spike selection strategies under a 10% GAF budget.

Method	GAF	Selection strategy	WT2 PPL(↓)	Avg. Acc.(↑)
Async-SNN-0.6B	✓	Random	28.46	48.90
		Membrane state	28.43	48.91
Async-SNN-1.7B	✓	Random	18.97	55.93
		Membrane state	18.95	55.96

A6 Hardware Simulation

In addition to the operation-level energy estimate $\mathbf{E}_{\text{est}}$, we evaluate hardware-level energy consumption and end-to-end inference latency. For 4-bit activation-quantized *Async-LLM*-1.7B, we use the Eyeriss simulation framework (Chen et al., 2016). For *Async-SNN*-1.7B, we use SANA-FE (Boyle et al., 2025) with a 32-chip Loihi configuration (Davies et al., 2018). The use of $T = 16$ simulation time-steps does not directly imply a $16\times$ increase in latency on neuromorphic hardware, where execution cost depends on the synaptic activity and hardware scheduling. As shown in Table A7, *Async-SNN*-1.7B + GAF (10%) reduces $\mathbf{E}_{\text{sim}}$ from 14.56 to 9.76 mJ/token (32.97%) and latency from 4.62 to 2.23 ms/token (51.73%) relative to *Async-LLM*-1.7B†, while retaining comparable task performance. Relative to unconstrained *Async-SNN*-1.7B, the corresponding reductions are 17.50% in simulated energy and 17.71% in latency.

Table A7: Hardware simulation of *Async-LLM*-1.7B† with Eyeriss and *Async-SNN*-1.7B with SANA-FE on a 32-chip Loihi configuration. † denotes 4-bit activation quantization. Reductions in parentheses are relative to *Async-LLM*-1.7B†.

Method	Category	GAF	$\frac{B}{N_{\text{total}}}$	WT2 PPL(↓)	Avg. Acc.(↑)	$\mathbf{E}_{\text{sim}}$(↓) (mJ / Token)	Latency(↓) (ms / Token)
Async-LLM-1.7B†	QANN	N/A	N/A	19.14	55.67	14.56	4.62
Async-SNN-1.7B	SNN	✗	N/A	**18.92**	**55.95**	11.83 (18.75%↓)	2.71 (41.34%↓)
		✓	20%	18.92	55.99	11.18 (23.21%↓)	2.56 (44.59%↓)
		✓	15%	18.94	55.97	10.49 (27.95%↓)	2.40 (48.05%↓)
		✓	10%	18.97	55.93	9.76 (32.97%↓)	2.23 (51.73%↓)
		✓	5%	55.32	42.68	**8.95 (38.53%↓)**	**2.03 (56.06%↓)**

A7 Comparison with Quantized ANN Baselines

We further compare *Async-SNN* with 4-bit activation-quantized Mamba (Gu & Dao, 2023) and RWKV (Peng et al., 2023) baselines in Table A8. At the smaller scale, *Async-SNN*-0.6B + GAF achieves comparable task performance to Mamba-0.8B† while reducing $\mathbf{E}_{\text{est}}$ from 4.28 to 2.46 mJ/token. At the larger scale, *Async-SNN*-1.7B achieves lower WT2 perplexity and higher average accuracy than RWKV-1.5B†, while GAF further reduces $\mathbf{E}_{\text{est}}$ from 10.03 to 6.95 mJ/token with negligible performance degradation. These comparisons provide complementary evidence for the accuracy–efficiency trade-off of the synchronization-free architecture against efficient quantized ANN baselines.

Table A8: Comparison with 4-bit activation-quantized ANN baselines. † denotes 4-bit activation quantization.

Method	GAF	$\frac{B}{N_{\text{total}}}$	WT2 PPL(↓)	Avg. Acc.(↑)	#MACs(↓) (G / Token)	#SOPs(↓) (G / Token)	$\mathbf{E}_{\text{est}}$(↓) (mJ / Token)
Mamba-0.8B†	N/A	N/A	**26.72**	**49.82**	0.93	N/A	4.28
Async-LLM-0.6B†	N/A	N/A	28.53	48.75	**0.76**	N/A	3.50
Async-SNN-0.6B	✗	N/A	28.37	48.94	N/A	3.77	3.39
	✓	10%	28.46	48.90	N/A	**2.73**	**2.46**
RWKV-1.5B†	N/A	N/A	21.32	55.13	**2.18**	N/A	10.03
Async-LLM-1.7B†	N/A	N/A	19.14	55.67	2.35	N/A	10.81
Async-SNN-1.7B	✗	N/A	**18.92**	**55.95**	N/A	10.55	9.50
	✓	10%	18.97	55.93	N/A	**7.72**	**6.95**

The architectural and conversion-stage contributions are complementary. The synchronization-free *Async-LLM* architecture enables asynchronous execution and robustness under strict firing budgets, whereas SFT QAT preserves accuracy during low-bit quantization and ANN-to-SNN conversion. In particular, the conversion ablation in Table 4 shows that SFT QAT substantially improves both WT2 perplexity and average accuracy over calibration alone or calibration with block-wise fine-tuning at both model scales.