TASK-GUIDED ANN-TO-SNN CONVERSION FOR LARGE LANGUAGE MODELS VIA SPARSE-BURST PHASE ENCODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Converting pretrained large language models (LLMs) from artificial neural networks (ANNs) to spiking neural networks (SNNs) provides an efficient way to reduce inference energy without costly training from scratch. However, accurate ANN-to-SNN conversion with few timesteps remains challenging, as activation outliers in a small number of ANN channels expand the dynamic range and require finer resolution in SNNs for accurate spike-based representation. Existing studies have made progress, but still face two limitations: some improve representation fidelity by adding timesteps or neurons, increasing inference overhead, while others optimize task-agnostic reconstruction, limiting task performance. To tackle this, we propose a novel conversion method that combines an efficient spike encoding scheme with a task-guided conversion framework. First, we introduce Sparse-Burst Phase (SBP) encoding to improve representational efficiency. SBP uses an integer resolution parameter together with signed burst spikes to decouple activation resolution from timestep count, enabling flexible representation of wide and nonuniform LLM activations without additional timesteps or neurons. Building on SBP, we further develop Task-Guided Adaptive-Resolution Conversion (TARC) framework to improve task performance. TARC first allocates the representation budget across channels according to their task sensitivity and then aligns the converted SNN with the pretrained ANN through channel-wise threshold and offset calibration. Extensive evaluations are conducted on seven LLaMA and Qwen models ranging from 7B-70B, covering language modeling, commonsense reasoning, and knowledge-intensive tasks. With only 4 timesteps, TARC achieves state-of-the-art performance among converted SNNs while closely matching the original ANN models. These results demonstrate that TARC enables accurate and efficient spike-driven LLMs under limited temporal and hardware budgets.

1 INTRODUCTION

Large language models (LLMs), exemplified by GPT-4 (OpenAI et al., 2023), Qwen3 (Yang et al., 2025), and LLaMA3 (Grattafiori et al., 2024), have demonstrated strong capabilities in language understanding, reasoning, and code generation. However, inference in these artificial neural network (ANN)-based models relies heavily on dense multiply-accumulate (MAC) operations, resulting in substantial arithmetic cost (Horowitz, 2014). In contrast, spiking neural networks (SNNs) transmit information through binary spikes (Maass, 1997; Roy et al., 2019), thereby replacing costly MAC operations with more efficient accumulation (AC) operations. Motivated by this potential, growing research efforts have focused on developing spike-based LLMs (Zhu et al., 2024; Xing et al., 2025).

Current studies mainly develop spike-based LLMs through two paradigms: direct training and ANN-to-SNN conversion (Zhu et al., 2024; Xing et al., 2025). Direct training typically uses surrogate gradients to address the non-differentiability of spiking neurons, but requires backpropagation through time, resulting in substantial memory and computational overhead (Neftci et al., 2019; Wei et al., 2024). In contrast, ANN-to-SNN conversion maps ANN activations to spiking neuronal dynamics while reusing pretrained weights (Rueckauer et al., 2017; Deng & Gu, 2021). It therefore avoids costly training from scratch and provides a practical route to constructing spike-based LLMs.

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.

However, ANN-to-SNN conversion is particularly challenging for LLMs, where activation outliers in a few channels enlarge the activation range and demand finer representational resolution under limited timesteps (Xiao et al., 2023; Wei et al., 2023). Existing methods therefore adapt spike encoding to this nonuniform LLM activations. SpikeLLM (Xing et al., 2025) assigns additional timesteps to selected channels, while LAS (Chen et al., 2026) introduces subneurons with different thresholds for normal and outlier activations. Although both methods reduce conversion error, they introduce additional temporal or neuronal overhead, leaving room for further efficiency improvement. Distribution-aware phase coding (Zheng et al., 2026) instead learns multiple bases to reduce local activation reconstruction error. However, its activation-level reconstruction objective is task-agnostic and does not distinguish channels according to their impact on language-modeling loss, potentially leading to suboptimal task-level performance. Therefore, existing LLM conversion methods still leave substantial room for improvement in both efficiency and performance.

In this paper, we propose a novel task-guided conversion method for LLMs, aimed at improving both conversion efficiency and task performance under a limited timestep. Specifically, we introduce Sparse-Burst Phase (SBP) encoding to decouple activation resolution from the number of timesteps. By allowing signed burst spikes at the first timestep, SBP increases activation resolution and enables flexible representation of wide and nonuniform LLM activations without additional timesteps or neurons. Building on SBP, we develop a Task-guided Adaptive-Resolution Conversion (TARC) framework, which first allocates the representation budget across channels according to task sensitivity and then aligns the converted SNN with the pretrained ANN through channel-wise threshold and offset calibration. Together, SBP and TARC enable efficient and accurate conversion of pretrained LLMs into spike-driven models under short time steps. The main contributions are summarized as:

- We introduce Sparse-Burst Phase Encoding (SBP), which introduces a resolution parameter to control representational capacity, decoupling activation resolution from timestep count. SBP uses signed burst spikes at the first timestep and signed unit spikes or silence thereafter. This design enables SBP to support a wider range of activation levels without additional timesteps or neurons, enabling flexible encoding of wide and nonuniform LLM activations.
- Building on SBP, we develop a novel ANN-to-SNN conversion framework, termed Task-Guided Adaptive-Resolution Conversion (TARC). TARC allocates activation resolutions across channels under a global budget based on their task sensitivity, followed by channel-wise threshold and offset calibration. Together, the design of SBP and TARC enables efficient and accurate conversion of pretrained LLMs under a limited number of timesteps.
- Extensive evaluations are conducted on seven LLaMA and Qwen models ranging from 7B to 70B, covering language modeling, commonsense reasoning, and knowledge-intensive tasks. With only 4 timesteps, TARC attains state-of-the-art performance among converted SNNs, closely matching the accuracy of the original ANNs while reducing spike activity by about 40% on LLaMA-3-8B compared with conventional 4-timestep phase coding.

2 Related work

ANN-to-SNN conversion ANN-to-SNN conversion transfers pretrained ANN parameters to SNNs, enabling energy-efficient inference without training from scratch. Early studies approximate ReLU activations in ANNs using firing rates in SNNs (Cao et al., 2015) and reduce the resulting mismatch through weight and threshold normalization (Diehl et al., 2015; Sengupta et al., 2019). However, with few timesteps, limited firing-rate levels restrict representation resolution and lead to large conversion errors. To address this issue, QCFS reduces quantization mismatch during ANN training (Bu et al., 2022), while post-conversion calibration adjusts neuron parameters to correct spike-count errors (Hao et al., 2023a;b; Cao et al., 2026). Another line of work improves the spike representation itself. Temporal weighting assigns different contributions to spikes across timesteps (Kim et al., 2018; Gu et al., 2026), burst spiking allows multiple spikes within each timestep (Li & Zeng, 2022), and TTFSFormer represents activations through first-spike timing (Zhao et al., 2025). Despite these advances, converting pretrained LLMs remains challenging because their activations often exhibit wide and highly nonuniform distributions, requiring finer spike-based representations (Zheng et al., 2026).Recent methods improve representation fidelity by adding timesteps or neurons (Xing et al., 2025; Jiang et al., 2025), but incur temporal or spatial overhead. Thus, accurate and efficient LLM conversion with limited budgets remains challenging.

Spiking language models Early studies developed spiking language models through direct training (Bal & Sengupta, 2024; Zhu et al., 2024; Xing et al., 2024), but the training cost grows rapidly with model scale. Later studies turned to ANN-to-SNN conversion to construct spiking language models from pretrained ANNs, as demonstrated by SpikeZIP-TF on RoBERTa (You et al., 2024) and SpikedAttention on BERT (Hwang et al., 2024). At the LLM scale, FAS combines model finetuning with SNN calibration (Chen et al., 2025), while SpikingBrain modifies the architecture and performs continued pretraining (Pan et al., 2025). Despite advances, both methods still require additional weight training. Recent studies improve LLM conversion mainly in two main ways. One line increases the representation capacity of spiking neurons. LAS uses separate subneurons for normal and outlier activations (Chen et al., 2026), while SpikeLLM assigns more encoding timesteps to important channels (Xing et al., 2025). Although effective, these methods introduce additional spatial or temporal overhead. Another line focuses on adaptive representation and reconstruction. SpikeQuant allocates resolution based on activation outlier statistics (Wang et al., 2026), while distribution-aware phase coding learns multiple encoding bases through local reconstruction (Zheng et al., 2026). However, these methods mainly optimize local statistics or reconstruction errors rather than the global language-modeling objective, leaving room for further performance improvement.

3 PRELIMINARY

Phase coding scheme ANN-to-SNN conversion approximates pretrained ANN activations with the accumulated outputs of spiking neurons (Cao et al., 2015). To realize this mapping, common encoding schemes include rate coding (Cao et al., 2015), time-to-first-spike coding (Zhao et al., 2025), and phase coding (Zheng et al., 2026). Among them, phase coding assigns timestep-dependent weights to spikes, providing higher representation resolution with only a few timesteps (Kim et al., 2018). This makes it well suited for low-latency conversion and has motivated its use in LLMs.

A common form of phase coding is binary phase coding, which assigns exponentially decaying weights to different timesteps. Specifically, the phase weight at timestep t is defined as $\omega_t = 2^{T-t}$, where T is the number of timesteps and the final timestep has unit weight (Hwang et al., 2024). The phase weight determines the firing threshold, reset magnitude, and output contribution at each timestep. Given a base threshold $V_\theta > 0$, the neuron dynamics are

$$v_1 = \sum_{t=1}^{T} I_t, \quad s_t = H(v_t - V_\theta \omega_t), \quad v_{t+1} = v_t - V_\theta \omega_t s_t, \tag{1}$$

where v_t is the membrane potential, $H(\cdot)$ is the Heaviside step function, and s_t is the binary spike. The weighted output is $o_t = V_\theta \omega_t s_t$, so earlier spikes represent larger values while later spikes provide finer resolution. As described above, T timesteps provide 2^T distinct activation levels; it can be extended to $2^{T+1} - 1$ levels by introducing negative spikes (Hwang et al., 2024), i.e., $s_t \in \{0, \pm 1\}$. However, the representable range remains limited by the timestep budget, and further expansion requires more timesteps or parallel neurons.

ANN-to-SNN conversion For phase coding-based conversion, an ANN activation x is first quantized to an integer q, and then represented by a spike train $\mathbf{S} = \{s_t\}_{t=1}^{T}$ with phase weights:

$$q = \mathrm{clip}\left(\mathrm{round}\left(\frac{x + V_{\mathrm{off}}}{V_\theta}\right), q_{\min}, q_{\max}\right) = \sum_{t=1}^{T} \omega_t s_t, \tag{2}$$

where V_θ controls the representation resolution, V_{off} denotes the membrane-potential offset, and $q_{\min}$ and $q_{\max}$ specify the representable range over T timesteps. Since this discrete mapping introduces ANN-SNN mismatch, further calibration is generally needed to reduce conversion error.

In practice, calibration adjusts the channel-wise threshold $V_{\theta,c}$ and offset $V_{\mathrm{off},c}$ to better match ANN activations (Cao et al., 2026). At layer l, these parameters can be optimized over a calibration set $\mathcal{D}_{\mathrm{cal}}$ by minimizing the output mismatch between the ANN and SNN (Li et al., 2021):

$$\min_{\{V_{\theta,c}, V_{\mathrm{off},c}\}_c} \underset{\mathcal{D}_{\mathrm{cal}}}{\mathrm{mean}} \left\| y_{\mathrm{SNN}}^{(l)} - y_{\mathrm{ANN}}^{(l)} \right\|_2^2. \tag{3}$$

Although calibration improves layer-wise alignment and reduces conversion error, it remains a local optimization and may limit task-level performance. This gap motivates global task-guided conversion, enabling direct optimization of the final task objective for improved performance.

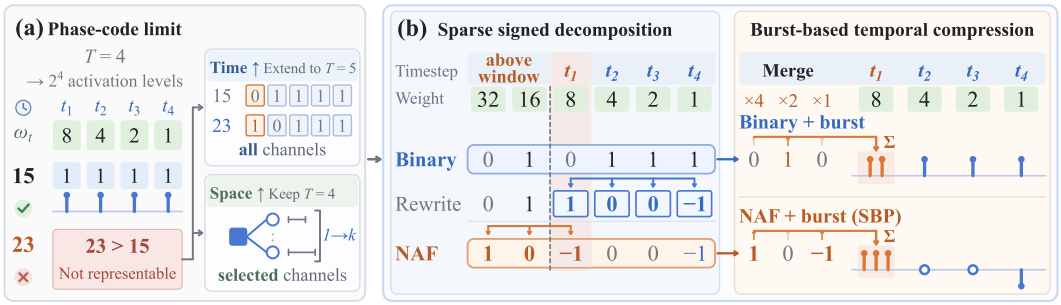


Figure 1: (a) Limitation of conventional binary phase coding under a fixed timestep budget. (b) SBP consists of two stages: sparse signed decomposition to produce sparse spike trains, and burst-based temporal compression to fit it within the T -timestep window.

4 METHOD

In this section, we present our LLM conversion method. First, we design sparse-burst phase encoding, which decouples activation resolution from the number of timesteps, enabling high-resolution spike representation without increasing timesteps or neurons. Building on this encoding, we develop a task-guided adaptive-resolution conversion framework that allocates representation budgets to channels based on their task sensitivity, followed by channel-wise threshold and offset calibration.

4.1 SPARSE-BURST PHASE ENCODING

Conventional phase coding couples activation resolution to the number of timesteps. To increase resolution without extending the temporal window, SBP introduces a resolution parameter $b \geq T$. This allows the quantized activation q in Eq. (2) to span $[-2^{b-1}, 2^{b-1} - 1]$, providing 2^b representable levels within a fixed T timestep. To realize this higher-resolution representation without additional timesteps or neurons, SBP includes two stage: sparse signed decomposition to produce sparse spike trains, and burst-based temporal compression to fit it within the T-timestep window.

Sparse signed decomposition In this stage, we focus on obtaining a sparse spike train at resolution b, temporarily ignoring the T-timestep constraint. Given b, the quantized value q is encoded using signed binary phase coding with the phase-weight vector $\boldsymbol{\omega} = (2^{b-1}, 2^{b-2}, \ldots, 2^0)$. Under this coding scheme, the same value of q may correspond to multiple spike trains. We adopt the non-adjacent form (NAF), a canonical signed-digit representation with minimum Hamming weight over the binary phase weights (Reitwiesner, 1960; Arno & Wheeler, 1993), to obtain a unique and sparse representation. Formally, the NAF representation of q is defined as $\mathbf{S}_q^{\mathrm{NAF}} = \{s_1, \ldots, s_b\}$, it satisfies:

$$s_t s_{t+1} = 0, \quad q = \sum_{t=1}^{b} s_t 2^{b-t}. \tag{4}$$

The constraint $s_t s_{t+1} = 0$ removes consecutive active spikes while preserving the exact value of q, yielding a uniquely defined sparse signed spike train. For a spike train obtained by standard signed binary phase coding, its NAF representation can be constructed through recursive carry-and-compensate rewriting. In particular, each run of adjacent same-sign spikes can be rewritten as

$$(0, \underbrace{\pm 1, \ldots, \pm 1}_{\text{adjacent same-sign spikes}}) \xrightarrow{\text{NAF rewrite}} (\underbrace{\pm 1, 0, \ldots, 0, \mp 1}_{\text{rewritten spikes}}), \tag{5}$$

which moves the accumulated contribution to a higher weight while preserving the value. This operation is repeated until no adjacent nonzero spikes remain, yielding the unique NAF spike train.

We then use an example in Fig. 1(b) to illustrate the sparse signed decomposition. Consider $q = 23$ with $b = 6$. Under the binary phase weights $\boldsymbol{\omega} = (2^5, 2^4, \ldots, 2^0)$, the standard binary spike train for $q = 23$ is $\mathbf{S} = (0, 1, 0, 1, 1, 1)$. Using the recursive rewriting rule in Eq. (5), we obtain

$$\mathbf{S}_{23}^{\mathrm{NAF}}\colon (0, 1, 0, \underbrace{1, 1, 1}_{\text{rewrite}}) \longrightarrow (0, \underbrace{1, 1}_{\text{rewrite}}, 0, 0, -1) \longrightarrow (1, 0, -1, 0, 0, -1). \tag{6}$$

Clearly, the represented value remains unchanged throughout the rewriting, while the number of active spikes is reduced from four to three. When $T = b = 6$, the resulting NAF spike train can be directly emitted. However, when $T < b$, the leading $b - T$ components fall outside the available timestep window and require additional burst-based temporal compression.

Burst-based temporal compression In this stage, we fit the extended spike train into the fixed T-timestep window when $b > T$. To achieve this, SBP aggregates the out-of-window contributions into a first-timestep burst consisting of an integer number of unit spikes. Since their binary weights are integer multiples of $\omega_1 = 2^{T-1}$, this burst operation preserves the represented value exactly.

For each quantized activation q, we denote its final SBP representation as $\mathbf{S}_q^{\mathrm{SBP}} = \{s_t\}_{t=1}^{T}$, with the T-timestep phase weights $\boldsymbol{\omega} = (2^{T-1}, 2^{T-2}, \ldots, 2^0)$. Given the NAF spike train $\mathbf{S}_q^{\mathrm{NAF}} = \{\tilde{s}_t\}_{t=1}^{b}$, we construct each s_t in $\mathbf{S}_q^{\mathrm{SBP}}$ as follows

$$s_t = \begin{cases} \dfrac{1}{\omega_1} \displaystyle\sum_{k=1}^{b-T+1} \tilde{s}_k 2^{b-k}, & t = 1, \\ \tilde{s}_{b-T+t}, & 2 \le t \le T. \end{cases} \tag{7}$$

At the first timestep, $s_1 \in \mathbb{Z}$ specifies a signed burst of $|s_1|$ unit spikes, with its sign determining the spike polarity. For the remaining timesteps, $s_t \in \{0, \pm 1\}$ denotes a signed unit spike or silence. By construction, the resulting spike train preserves the represented value exactly, i.e., $q = \sum_{t=1}^{T} \omega_t s_t$.

We continue with the example illustrated in Fig. 1(b), where $q = 23$, $b = 6$, and $T = 4$. Given the NAF spike train $\mathbf{S}_{23}^{\mathrm{NAF}} = (1, 0, -1, 0, 0, -1)$, the two highest-order components $(1, 0)$ lie outside the valid $T = 4$ timestep window. SBP therefore merges these out-of-window contributions into the first-timestep burst, yielding $s_1 = 3$. Consequently, the resulting spike train is $\mathbf{S}_{23}^{\mathrm{SBP}} = (3, 0, 0, -1)$, where $s_1 = 3$ denotes three positive unit spikes emitted at the first timestep rather than a single spike of amplitude three. As described above, SBP preserves spike-driven computation while supporting b-resolution representation within T timesteps, without additional timesteps or neurons.

Representation and sparsity of SBP SBP first sparsifies the signed phase representation using NAF and then compresses the high-order components into a first-timestep burst, preserving the encoded value efficiently and exactly. Compared with conventional binary phase coding, SBP extends the representational range to b-bit activations within a fixed T timesteps, without additional timesteps or neurons. Proposition 1 further shows that SBP minimizes the number of active timesteps among all feasible encodings, with its proof provided in Appendix C.1.

Proposition 1 (Temporal Sparsity of SBP). *For every integer $q \in [-2^{b-1}, 2^{b-1} - 1]$, consider length-$T$ spike trains $\mathbf{S} = (s_1, \ldots, s_T)$ satisfying $q = \sum_{t=1}^{T} 2^{T-t} s_t$, where $s_1 \in \mathbb{Z}$, $|s_1| \le 2^{b-T}$, and $s_t \in \{-1, 0, +1\}$ for $2 \le t \le T$. SBP minimizes the active-timestep count $\|\mathbf{S}\|_0 := \sum_t \mathbb{1}[s_t \neq 0]$ among all feasible spike trains. Moreover, its unit spike count $\Gamma(\mathbf{S}) := \sum_{t=1}^{T} |s_t|$ is at most one larger than the minimum achievable unit spike count under the same constraints.*

4.2 Task-guided adaptive-resolution conversion framework

SBP allows each channel c to adopt an individual resolution b_c. Thus, given a fixed global budget B, allocating it across channels to maximize conversion performance is crucial for SBP-based ANN-to-SNN conversion. This motivates us to propose a task-guided adaptive-resolution conversion framework, which assigns channel-wise resolutions according to their task relevance.

Channel-wise heterogeneity Before allocating the representation budget, we first examine how conversion errors are distributed across channels. Conversion error mainly arises from rounding and clipping, with rounding error accounting for 99.12% of the total squared conversion error before allocation and calibration, as shown in Fig. 3(a). Therefore, we focus on rounding error when determining channel-wise resolutions in our TARC. We further analyze rounding errors across individual layers and channels. As shown in Fig. 3(b), some channels exhibit rounding errors that are orders of magnitude larger than others. Fig. 3(c) further shows that the top 1% of site-channel pairs contribute 19.7% of the total rounding error.

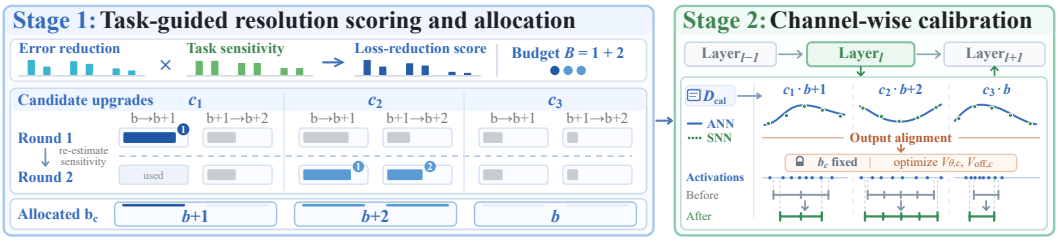


Figure 2: Overview of the proposed task-guided adaptive-resolution conversion framework.

These results reveal a highly nonuniform error distribution and pronounced channel-wise heterogeneity. Therefore, a simple budget allocation strategy is to allocate more resolution to channels with larger conversion errors. However, such error-based allocation remains local, since reducing a channel's reconstruction error does not necessarily translate into improved task performance. To better preserve the task performance, we jointly consider the conversion error and its impact on the global task objective.

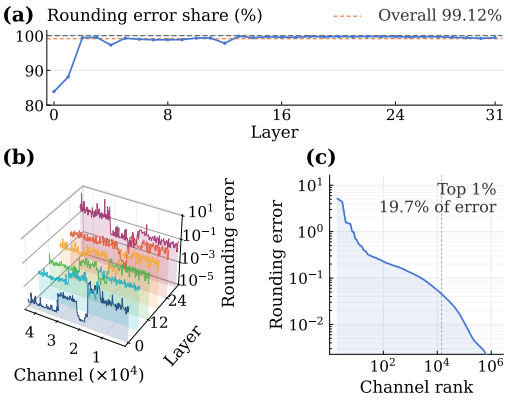


Figure 3: Channel-wise heterogeneity of rounding error in LLaMA-3-8B on WikiText-2.

Task-guided resolution scoring and allocation To jointly quantify these two factors, we estimate the task-level benefit of increasing each channel's resolution. As illustrated in Fig. 2, the score combines the rounding error reduction with the task sensitivity of the corresponding channel. Consider increasing channel c from resolution b to $b+1$. Let r_c denote the width of its activation range. Under the uniform-noise model (Bennett, 1948), the rounding-error variance at resolution b is $\mathbb{E}[\epsilon_{c,b}^2] = \frac{r_c^2}{12(2^b-1)^2}$. We further measure the task sensitivity of channel c using the empirical Fisher $F_c = \mathbb{E}_{\mathcal{D}_{\text{cal}}}\left[\left(\frac{\partial \mathcal{L}}{\partial a_c}\right)^2\right]$, where $\mathcal{L}$ denotes the task loss and a_c is an activation in channel c. Under a second-order approximation of the task loss (LeCun et al., 1990), the estimated loss reduction from increasing the resolution from b to $b+1$ is:

$$\widehat{\Delta\mathcal{L}}_{c,b} = \frac{F_c r_c^2}{24}\left[\frac{1}{(2^b-1)^2} - \frac{1}{(2^{b+1}-1)^2}\right]. \tag{8}$$

The detailed derivation is provided in Appendix C.2. This estimate jointly captures the reducible rounding error and the task sensitivity of each channel. Since the factor $1/24$ is common to all candidates, we define $g_{c,b} = 24\,\widehat{\Delta\mathcal{L}}_{c,b}$ as the allocation score.

Given the task-guided scores, we allocate the global budget B across candidate resolution increases. Each channel starts from $b_c = b_0$, with $b_0 = T$ by default, and each increase $b \to b+1$ uses one budget unit. As shown in Fig. 2, all candidates are ranked by $g_{c,b}$, and the highest-scoring ones are selected until the budget is exhausted, yielding the final channel-wise resolutions $\{b_c\}$. We constrain $b_c \le b_{\max}$ to limit the first-timestep burst amplitude. Since channel sensitivities may change after resolution updates, the allocation is performed in two rounds, with F_c re-estimated after the first round. Within each round, F_c and r_c are fixed, and successive resolution increases of the same channel have diminishing gains. The resulting allocation is optimal for the estimated task-loss reduction within each round, as stated below; the proof is provided in Appendix C.2.

Proposition 2 (Per-Round Optimality of Resolution Allocation)**.** *Within an allocation round, fix the channel sensitivities F_c, activation-range widths r_c, and starting resolutions. For any fixed number of one-level increases, selecting the feasible candidates with the highest $g_{c,b}$ scores maximizes the total estimated task-loss reduction.*

Channel-wise calibration After determining the channel-wise resolutions, TARC further aligns the converted SNN with the pretrained ANN following the calibration scheme in Sec. 3. The preceding scoring and budget allocation mainly reduce rounding error, whereas this calibration adjusts the channel-wise threshold $V_{\theta,c}$ and offset $V_{\mathrm{off},c}$ to reduce the remaining conversion error. This calibration is performed layer by layer while keeping the pretrained weights fixed.

5 EXPERIMENTS

Models and tasks We evaluate TARC on seven pretrained LLMs across different model families and scales: LLaMA-2 (7B, 13B, and 70B), LLaMA-3 (8B and 70B), Qwen2-7B, and Qwen3-8B. We evaluate three task types: language modeling, commonsense reasoning, and knowledge-intensive reasoning. Specifically, we report perplexity (PPL) on WikiText-2 and C4, mean zero-shot accuracy on six commonsense benchmarks (CS-Avg.), and 5-shot accuracy on MMLU.

Baselines We compare TARC with five ANN-to-SNN conversion methods: TTFSFormer (Zhao et al., 2025), SpikedAttention (Hwang et al., 2024), LAS (Chen et al., 2026), SpikeLLM (Xing et al., 2025), and SpikingLLM (Zheng et al., 2026). Each is evaluated on the pretrained models supported by its original formulation. For each model, all methods use the same data and evaluation protocol.

Implementation settings TARC first allocates channel-wise resolutions and then calibrates each converted SNN layer. For intuitive comparison with baselines, we use $\rho \in \{0.2, 0.5, 1.0\}$ to denote the average number of one-level resolution increases per channel. We set $T = b_0 = 4$ and $b_{\max} = 6$ by default. We use language-modeling cross-entropy as the task loss $\mathcal{L}$ for resolution allocation. Both allocation and layer-wise calibration use the same 128 WikiText-2 training sequences, disjoint from the evaluation data. The pretrained weights remain fixed in FP16 during conversion. Ablation studies use LLaMA-3-8B unless otherwise stated. Additional details are provided in Appendix B.

5.1 PERFORMANCE COMPARISON

Table 1: PPL (↓) comparison of TARC on WikiText-2 and C4. Top three results are highlighted as first, second, and third. † indicates self-reproduced results using open-source implementations.

Dataset	Method	T	LLaMA-2-7B	LLaMA-2-13B	LLaMA-2-70B	LLaMA-3-8B	LLaMA-3-70B	Qwen2 7B	Qwen3 8B
Wiki-2	FP16	–	5.47	4.88	3.33	6.14	2.85	7.14	9.72
	TTFSFormer†	64	6.69	5.58	4.20	20.30	6.11	–	–
	SpikedAttention†	20	8.85	7.54	5.45	14.43	11.89	118.18	15.87
	LAS†	8	13.81	21.97	4114.33	42.68	4813.87	183.87	5890.56
	SpikeLLM†	4	8.22	6.94	6.08	20.02	10.59	18.72	21.55
	SpikingLLM†	6	17.10	14.92	65.79	7.16	6.37	9.41	11.25
	TARC$_{\rho=0.2}$	4	5.76	5.11	3.54	6.72	3.69	7.53	10.20
	TARC$_{\rho=0.5}$	4	5.69	5.02	3.48	6.54	3.42	7.44	10.07
	TARC$_{\rho=1.0}$	4	5.57	4.95	3.43	6.35	3.13	7.38	9.82
C4	FP16	–	6.84	6.41	5.55	8.62	6.81	9.77	13.46
	TTFSFormer†	64	8.56	7.23	6.60	39.38	11.43	–	–
	SpikedAttention†	20	10.55	9.26	7.38	17.97	17.82	151.21	20.30
	LAS†	8	17.92	45.59	6017.93	44.22	1919.02	218.52	4095.93
	SpikeLLM†	4	10.45	8.78	8.18	24.43	15.07	22.51	27.53
	SpikingLLM†	6	24.21	22.73	65.09	10.61	12.14	13.71	18.79
	TARC$_{\rho=0.2}$	4	7.17	6.65	5.69	9.58	7.34	11.07	14.10
	TARC$_{\rho=0.5}$	4	7.07	6.57	5.64	9.35	7.17	10.81	13.88
	TARC$_{\rho=1.0}$	4	6.95	6.49	5.61	8.97	7.00	10.64	13.65

We first evaluate TARC on WikiText-2 and C4 across 7 pretrained LLMs using PPL. As shown in Table 1, TARC consistently outperforms prior ANN-to-SNN conversion methods across model families and scales. With $\rho = 0.2$, corresponding to an effective timestep budget of about 4.2 under conventional phase coding, TARC surpasses baselines using 6-64 timesteps in most settings. Increasing ρ further closes the gap to FP16 ANN models, with $\rho = 1.0$ achieving the best converted PPL. This establishes TARC as the state-of-the-art conversion method for language modeling, featuring high accuracy with low inference cost. Additional task-level evaluations are provided in Appendix B.

Table 2: Comparison of CS-Avg./5-shot MMLU accuracy (%). Top three results are highlighted as first, second, and third; † denotes self-reproduced results with official code.

Method	T	LLaMA-2 7B	LLaMA-2 13B	LLaMA-2 70B	LLaMA-3 8B	LLaMA-3 70B	Qwen2 7B	Qwen3 8B
FP16	–	70.40/45.93	73.20/54.91	77.73/68.72	74.19/65.20	80.87/78.81	73.63/70.77	74.05/75.13
TTFSFormer†	64	65.80/36.22	69.99/47.73	74.82/57.83	61.50/29.87	74.32/53.35	–/–	–/–
SpikedAttention†	20	62.56/31.18	66.50/41.00	70.62/55.57	60.88/37.49	60.56/50.48	50.26/39.81	61.70/53.57
LAS†	8	63.64/28.18	65.08/30.46	50.46/25.32	54.59/26.94	36.27/22.95	38.12/25.44	35.88/23.69
SpikeLLM†	4	60.82/32.18	66.00/41.32	67.08/49.06	51.63/29.65	60.33/33.39	54.45/41.70	55.10/43.31
SpikingLLM†	6	69.48/32.87	72.86/35.63	75.64/45.71	72.99/62.66	74.20/61.97	72.08/66.19	72.76/69.82
TARC$_{\rho=0.2}$	4	68.24/41.27	71.43/52.51	76.66/66.84	73.16/61.87	78.57/76.54	73.50/66.55	72.57/72.33
TARC$_{\rho=0.5}$	4	69.19/42.19	72.13/53.08	77.23/67.69	73.54/62.94	79.89/77.60	73.21/67.56	73.42/73.15
TARC$_{\rho=1.0}$	4	69.68/43.97	72.63/54.32	77.51/68.03	73.89/63.91	79.83/78.09	73.71/68.67	73.58/74.15

We further evaluate TARC on commonsense reasoning and MMLU across seven pretrained LLMs. As shown in Table 2, TARC consistently achieves higher CS-Avg. and 5-shot MMLU accuracy than prior ANN-to-SNN conversion methods across model families and scales. With $\rho = 0.2$, corresponding to an effective timestep budget of about 4.2, TARC outperforms baselines using 6-64 timesteps in most settings. Increasing ρ further improves task accuracy, with $\rho = 1.0$ achieving the best converted MMLU results on all seven models and the best CS-Avg. on most models. Overall, TARC achieves state-of-the-art downstream performance with a low inference budget.

5.2 Ablation and Validation Studies

Sparse-burst phase encoding We study SBP from three aspects: encoding sparsity, the effect of its resolution budget b, and burst overhead. First, we isolate the effect of SBP by comparing it with binary phase coding at $T = b_c = 4$, without budget allocation or layer-wise calibration. As shown in Fig. 4(a), SBP increases the silent-timestep ratio from 50.01% to 76.52%, while maintaining similar PPL and CS-Avg. This proves its efficiency and accuracy. Second, we integrate SBP into TARC and vary the resolution budget to study the effect of b. As shown by SBP ($\rho > 0$), increasing ρ reduces WikiText-2 PPL from 7.69 to 6.54 and increases CS-Avg., while the mean firing rate remains below that of binary phase coding even at $\rho = 1.0$. Third, we examine burst activity to study its overhead. As shown in Fig. 4(b), burst timesteps remain rare, increasing only from 0.26% at $\rho = 0.2$ to 1.30% at $\rho = 1.0$, indicating that the additional burst cost is limited. Overall, SBP improves representation capacity while preserving sparse spike activity, achieving a favorable accuracy-efficiency trade-off.

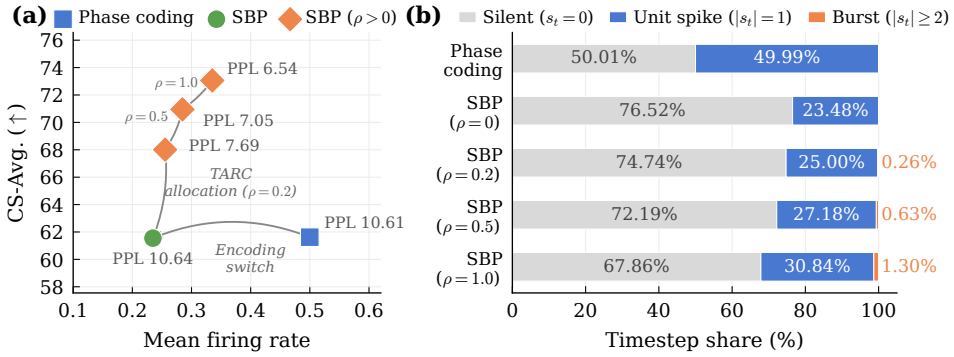


Figure 4: Ablation study of SBP on LLaMA-3-8B at $T = 4$. (a) Accuracy-efficiency trade-off of SBP. (b) Distribution of silent, unit spike, and burst timesteps under binary phase coding and SBP.

TARC conversion framework TARC consists of two stages: task-guided resolution scoring and allocation, followed by channel-wise calibration. To validate Stage 1, we compare binary phase coding without resolution allocation, random resolution allocation, and our task-guided resolution allocation under the same budget. As shown in Table 3, both random and task-guided allocation

improve over binary phase coding, while the task-guided strategy further reduces PPL to 7.05 and increases CS-Avg. to 70.95% at $\rho = 0.5$. We then validate Stage 2 through channel-wise calibration, which consistently reduces PPL and improves CS-Avg. across no upgrades, random allocation, and task-guided allocation, reaching 7.17/71.04%, 6.78/72.31%, and 6.54/73.54%, respectively.

Table 3: Ablation study of the two stages of TARC on LLaMA-3-8B at $T = 4$. "w/o Calib." and "w/ Calib." denote results without and with channel-wise calibration, respectively.

Allocation strategy	ρ	**Wiki-2 PPL ↓**		**CS-Avg. (%) ↑**	
		w/o Calib.	**w/ Calib.**	**w/o Calib.**	**w/ Calib.**
w/o resolution allocation	0	$10.64_{\textbf{Base}}$	$7.17_{\textbf{Base}}$	$61.55_{\textbf{Base}}$	$71.04_{\textbf{Base}}$
Random resolution allocation	0.5	$8.26_{\textbf{-2.38}}$	$6.78_{\textbf{-0.39}}$	$67.63_{\textbf{+6.08}}$	$72.31_{\textbf{+1.27}}$
Task-guided resolution allocation	0.5	$7.05_{\textbf{-3.59}}$	$6.54_{\textbf{-0.63}}$	$70.95_{\textbf{+9.40}}$	$73.54_{\textbf{+2.50}}$

Resolution score function in TARC We examine the two components of the TARC score function: task sensitivity and rounding-error reduction. As shown in Fig. 5(a), the full score consistently outperforms Fisher-only, range-only, and random allocation in held-out WikiText-2 cross-entropy reduction. At $\rho = 0.05$, it already reaches 57% of its $\rho = 1$ reduction relative to uniform b_0, showing that it identifies important channels more effectively under tight budgets. Fig. 5(b) further shows a selective allocation: at $\rho = 0.5$, 63.7% of channels remain at the base resolution, while only about 11% receive two or more upgrades. These results show that combining task sensitivity and rounding-error reduction enables more effective resolution allocation.

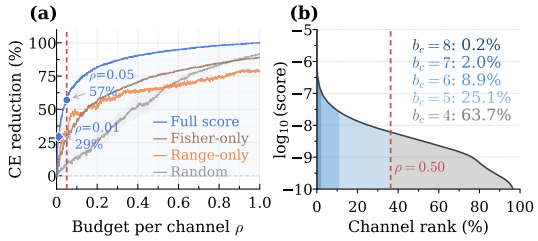


Figure 5: Ablation of the score function in TARC on LLaMA-3-8B. (a) Cross-entropy reduction under different scoring strategies. (b) Ranked full scores and the resulting resolution allocation at $\rho = 0.5$.

5.3 Energy Analysis

We evaluate the efficiency of TARC using the energy estimation in Appendix D. As shown in Table 4, at $T = 4$, TARC with $\rho = 0.2$ reduces firing rate from 0.50 to 0.30 and energy from 29.391 to 18.626 mJ/token, while improving both PPL and CS-Avg. With $\rho = 1.0$, four-step TARC further outperforms five-step phase coding in both PPL (6.35 vs. 6.43) and CS-Avg. (73.89% vs. 73.10%), while using 38.7% less energy. Increasing ρ improves task performance with moderate increases in firing rate and energy, and all TARC settings remain substantially more efficient than ANN. Overall, TARC achieves a favorable performance-energy trade-off through sparse spike activity and adaptive resolution allocation.

Table 4: Energy analysis of TARC on LLaMA-3-8B.

Method	T	**Performance**		**Efficiency**		
		PPL ↓	**CS-Avg. ↑**	**FR ↓**	**Energy ↓**	**Ratio ↓**
Dense ANN	–	6.14	74.19	–	35.783	1.000
Phase coding	4	7.21	71.30	0.50	29.391	0.821
Phase coding	5	6.43	73.10	0.50	36.430	1.018
$\text{TARC}_{\rho=0.2}$	4	6.72	73.16	0.30	18.626	0.521
$\text{TARC}_{\rho=0.5}$	4	6.54	73.54	0.33	20.004	0.559
$\text{TARC}_{\rho=1.0}$	4	6.35	73.89	0.37	22.347	0.625

6 Conclusion

This work presents a task-guided ANN-to-SNN conversion framework for efficient spike-driven LLMs under limited timesteps. SBP encoding decouples representation resolution from timestep count and provides a sparse representation of wide and nonuniform activations without additional timesteps or neurons. Building on SBP, TARC allocates channel-wise resolution according to task relevance and further reduces conversion mismatch through channel-wise calibration. Across seven LLaMA and Qwen models, the proposed method achieves state-of-the-art conversion performance at $T = 4$ and closely approaches the original ANN on language modeling and reasoning tasks. Thus, TARC provides an effective solution for accurate and efficient LLM conversion under tight budgets.

AI USE STATEMENT

In this work, we used generative AI tools to assist with manuscript writing, including language editing and improving the clarity and conciseness of the presentation. Generative AI was also used to assist in developing the proof of Proposition 1. The proof was independently checked and revised multiple times by the authors for mathematical correctness. Generative AI was not used to generate experimental results or make scientific conclusions. All AI-assisted content was carefully reviewed by the authors, who take full responsibility for the final content of this work.

REFERENCES

Steven Arno and Ferrell S. Wheeler. Signed digit representations of minimal hamming weight. *IEEE Transactions on Computers*, 42(8):1007–1010, 1993. doi: 10.1109/12.238495.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

Malyaban Bal and Abhronil Sengupta. SpikingBERT: Distilling BERT to train spiking language models using implicit differentiation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10998–11006, 2024. doi: 10.1609/aaai.v38i10.28975.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

W. R. Bennett. Spectra of quantized signals. *The Bell System Technical Journal*, 27(3):446–472, 1948. doi: 10.1002/j.1538-7305.1948.tb01340.x.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7B3IJMM1k_M.

Honglin Cao, Shuai Wang, Zijian Zhou, Ammar Belatreche, Wenjie Wei, Yu Liang, Yu Yang, Rui Xi, Malu Zhang, and Haizhou Li. Towards training-free and accurate ANN-to-SNN conversion via activation-aware redistribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 1703–1711, 2026. doi: 10.1609/aaai.v40i3.37148. URL https://ojs.aaai.org/index.php/AAAI/article/view/37148.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015. doi: 10.1007/s11263-014-0788-3.

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. URL https://arxiv.org/abs/2502.04405.

Long Chen, Xiaotian Song, and Yanan Sun. LAS: Loss-less ANN-SNN conversion for fully spike-driven large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 1730–1738, 2026. doi: 10.1609/aaai.v40i3.37151.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=FZ1oTwcXchK.

Peter U. Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, pp. 1–8, 2015. doi: 10.1109/IJCNN.2015.7280696.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: Hessian AWare quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 293–302, 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yiwen Gu, Junchuan Gu, Haibin Shen, and Kejie Huang. Temporal weighted encoding: Towards maximal-capacity spike coding for ANN–SNN conversion. In *International Conference on Machine Learning*, 2026.

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, pp. 11–21, 2023a. doi: 10.1609/aaai.v37i1.25071.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between ANNs and SNNs by calibrating offset spikes. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=PFbzoWZyZRX.

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.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7c9341ad0263428b5057d92f4d88dfa0-Abstract-Conference.html.

Yizhou Jiang, Feng Chen, Yihan Li, Yuqian Liu, Haichuan Gao, Tianren Zhang, and Ying Fang. Adaptive fission: Post-training encoding for low-latency spike neural networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e2e0a9004132490db1be58bffa419268-Abstract-Conference.html.

Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018. doi: 10.1016/j.neucom.2018.05.087.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pp. 598–605, 1990.

Yang Li and Yi Zeng. Efficient and accurate conversion of spiking neural network with burst spikes. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2485–2491, 2022. doi: 10.24963/ijcai.2022/345. URL https://doi.org/10.24963/ijcai.2022/345.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6316–6325. PMLR, 2021. URL https://proceedings.mlr.press/v139/li21d.html.

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. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9febda1c8344cc5f2d51713964864e93-Abstract-Conference.html.

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

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019. doi: 10.1109/MSP.2019.2931595.

OpenAI, 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.

Yuqi Pan, Yupeng Feng, Jinghao Zhuang, Siyu Ding, Han Xu, Zehao Liu, Bohan Sun, Yuhong Chou, Xuerui Qiu, Anlin Deng, et al. SpikingBrain: Spiking brain-inspired large models. *arXiv preprint arXiv:2509.05276*, 2025. URL https://arxiv.org/abs/2509.05276.

George W. Reitwiesner. Binary arithmetic. In *Advances in Computers*, volume 1, pp. 231–308. Academic Press, 1960. doi: 10.1016/S0065-2458(08)60610-5.

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

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682, 2017. doi: 10.3389/fnins.2017.00682.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 13:95, 2019. doi: 10.3389/fnins.2019.00095.

Chenyu Wang, Zhanglu Yan, Zhi Zhou, Xu Chen, and Weng-Fai Wong. Energy-efficient and dequantization-free quantization of LLMs: A spiking neural network approach to salient value mitigation. In *Proceedings of the ACM Web Conference 2026*, pp. 5334–5345, 2026. doi: 10.1145/3774904.3792337.

Wenjie Wei, Malu Zhang, Jilin Zhang, Ammar Belatreche, Jibin Wu, Zijing Xu, Xuerui Qiu, Hong Chen, Yang Yang, and Haizhou Li. Event-driven learning for spiking neural networks. *arXiv preprint arXiv:2403.00270*, 2024.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1648–1665, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.102. URL https://aclanthology.org/2023.emnlp-main.102/.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54698–54714. PMLR, 2024. URL https://proceedings.mlr.press/v235/xing24d.html.

Xingrun Xing, Boyan Gao, Zheng Liu, David A. Clifton, Shitao Xiao, Wanpeng Zhang, Li Du, Zheng Zhang, Guoqi Li, and Jiajun Zhang. SpikeLLM: Scaling up spiking neural network to large language models via saliency-based spiking. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/510e7d39fce008a3e31de54b8f5be9ac-Abstract-Conference.html.

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.

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*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57367–57383. PMLR, 2024. URL https://proceedings.mlr.press/v235/you24b.html.

Lusen Zhao, Zihan Huang, Jianhao Ding, and Zhaofei Yu. TTFSFormer: A TTFS-based lossless conversion of spiking transformer. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhao25n.html.

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*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/81858886d6a6f22c2e555e9d8f5172d9-Abstract-Conference.html.

Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K. Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=gcf1anBL9e.

A CONVERSION ALGORITHM AND PARAMETER UPDATES

Algorithm 1 summarizes TARC with R allocation rounds ($R = 2$ by default). The parameter updates are specified below; Appendix D describes SNN inference and its operation counts.

Algorithm 1 Task-guided adaptive-resolution conversion (TARC) with SBP.

Require: f_{ANN}, $\mathcal{D}_{\mathrm{cal}}$, T, b_0, $b_{\max}$, $\alpha_{\min}$, and nonnegative round budgets $\{B^{(r)}\}_{r=1}^{R}$ totaling B.
1: Initialize f_{SNN} from f_{ANN} with SBP at $b_c = b_0$, setting $(V_{\theta,c}, V_{\mathrm{off},c})$ from ANN calibration extrema (m_c^-, m_c^+) (Appendix A.1).
2: **Stage 1: allocate channel resolutions.**
3: **for** $r = 1$ to R **do**
4: Estimate channel sensitivities F_c on the current SNN.
5: Score all candidate increases (c, b) with $m_c^- < m_c^+$ and $b_c \le b < b_{\max}$ using $g_{c,b} = 24\,\widehat{\Delta\mathcal{L}}_{c,b}$ from Eq. (8).
6: Select up to $B^{(r)}$ highest-scoring increases.
7: For each upgraded channel, raise b_c by its selected count and update $(V_{\theta,c}, V_{\mathrm{off},c})$ once from its pre-round values by Eq. (12).
8: **end for**
9: **Stage 2: calibrate layers with $\{b_c\}$ fixed.**
10: **for** each transformer layer l in input-to-output order **do**
11: Initialize this layer's endpoint ratios at $\alpha_c^- = \alpha_c^+ = 1$.
12: Minimize $\mathcal{L}_{\mathrm{rec}}^{(l)}$ over $\alpha_c^{\pm} \in [\alpha_{\min}, 1]$, with thresholds and offsets set by Eq. (13) and the same propagated input for its ANN and SNN versions.
13: Freeze this layer's thresholds and offsets; pass its SNN output to both versions of the next layer.
14: **end for**
15: **return** f_{SNN}.

A.1 PARAMETER INITIALIZATION AND CALIBRATION

Initialization. Each channel shares $(b_c, V_{\theta,c}, V_{\mathrm{off},c})$ across its scalar activations. For bounds $l_c < u_c$, set

$$V_{\theta,c} = \frac{u_c - l_c}{2^{b_c} - 1}, \qquad V_{\mathrm{off},c} = -l_c - 2^{b_c - 1} V_{\theta,c}. \tag{9}$$

We take the observed ANN activation extrema (m_c^-, m_c^+) as the initial candidate bounds. At $b_c = b_0$, apply the threshold guard $\varepsilon_q = 10^{-5}$ and $M_q = 10^4$ during initialization:

$$V_{\theta,c} = \mathrm{clip}\left(\frac{m_c^+ - m_c^-}{2^{b_0} - 1}, \varepsilon_q, M_q\right), \qquad l_c = m_c^-, \qquad u_c = l_c + (2^{b_0} - 1) V_{\theta,c}. \tag{10}$$

Equation (9) then gives $V_{\mathrm{off},c}$. For channels with $m_c^- < m_c^+$, the effective width $r_c = u_c - l_c$ is used for allocation; channels constant on the calibration set remain at b_0 and are not scored for upgrades. If the threshold is not clipped, $(l_c, u_c) = (m_c^-, m_c^+)$. For an activation x_c, the signed integer encoded by SBP and its reconstructed value are

$$\begin{aligned} q_c &= \mathrm{clip}\left(\mathrm{round}\left(\frac{x_c + V_{\mathrm{off},c}}{V_{\theta,c}}\right), -2^{b_c - 1}, 2^{b_c - 1} - 1\right), \\ \widehat{x}_c &= V_{\theta,c} q_c - V_{\mathrm{off},c}. \end{aligned} \tag{11}$$

This is Eq. (2) for channel c. In the $b_c = T$ binary phase-coding comparison, the unsigned label is $\ell_c = q_c + 2^{b_c - 1}$; signed spike codes encode q_c. Throughout, round uses nearest-integer rounding with ties to even.

Stage 1 updates. Within an allocation round, upgrading a channel changes its resolution while preserving its range. From the parameters saved before the round, the selected final resolution b_c gives

$$\begin{aligned} V_{\theta,c} &= V_{\theta,c}^{\mathrm{in}} \frac{2^{b_c^{\mathrm{in}}} - 1}{2^{b_c} - 1}, \\ V_{\mathrm{off},c} &= V_{\mathrm{off},c}^{\mathrm{in}} + 2^{b_c^{\mathrm{in}} - 1} V_{\theta,c}^{\mathrm{in}} - 2^{b_c - 1} V_{\theta,c}. \end{aligned} \tag{12}$$

This preserves both range endpoints. The threshold guard in Eq. (10) is not reapplied during Stage 1, since doing so could shift them. Before each subsequent round, we re-estimate F_c on the updated model to score the remaining resolution increases. Appendix B specifies the default round budgets.

Stage 2 calibration. With b_c fixed, endpoint ratios $\alpha_c^{\pm} \in [\alpha_{\min}, 1]$ scale the observed channel minimum m_c^- and maximum m_c^+. They determine candidate bounds, then the threshold and offset:

$$\begin{aligned} \tilde{l}_c &= \alpha_c^- m_c^-, \qquad \tilde{u}_c = \alpha_c^+ m_c^+, \\ \bar{\Delta}_c &= \frac{\tilde{u}_c - \tilde{l}_c}{2^{b_c} - 1}, \qquad V_{\theta,c} = \mathrm{clip}(\bar{\Delta}_c, \varepsilon_q, M_q), \\ & \qquad\qquad V_{\mathrm{off},c} = -\tilde{l}_c - 2^{b_c - 1} V_{\theta,c}. \end{aligned} \tag{13}$$

The candidate endpoints are not projected to enforce $\tilde{l}_c < \tilde{u}_c$. When they are ordered and threshold clipping is inactive, Eq. (13) realizes both bounds exactly. If clipping is active, the lower endpoint remains $\tilde{l}_c$, while the upper endpoint becomes $\tilde{l}_c + (2^{b_c} - 1)V_{\theta,c}$. Because initialization may clip the threshold, starting Stage 2 with $\alpha_c^- = \alpha_c^+ = 1$ need not reproduce the final Stage 1 grid. We use the layer-output mismatch in Eq. (3) as the calibration loss $\mathcal{L}_{\mathrm{rec}}^{(l)}$ and minimize it over the bounded endpoint ratios; Appendix B gives the optimization settings.

For both Fisher estimation and Stage 2 calibration, the forward pass uses $\widehat{x}_c = V_{\theta,c} q_c - V_{\mathrm{off},c}$. The backward pass uses an identity straight-through estimator (STE) for rounding (Bengio et al., 2013) and the standard clipping derivative, zero outside the clipping interval.

B Experimental details

Evaluation protocol. We load each model from its public pretrained checkpoint and use its tokenizer; converted models are evaluated in a GPU-based simulator. Within each target model, all methods use the same evaluation text. Perplexity is computed on the complete, consecutive 2048-token blocks of the WikiText-2 test split and 256 fixed, unpadded 2048-token windows from C4 validation. For the LLaMA-3-8B spike-activity and energy analyses, we pool the 141 WikiText-2 blocks with these C4 windows. The held-out cross-entropy reductions in Fig. 5(a) use the WikiText-2 blocks alone. With lm-evaluation-harness v0.4.2 and its standard task definitions, we report zero-shot `acc_norm` for ARC-e, ARC-c, HellaSwag, and PIQA, zero-shot `acc` for BoolQ and WinoGrande, and 5-shot `acc` for MMLU. CS-Avg. is the mean of the six commonsense task accuracies.

Data for allocation and calibration. Resolution allocation and layer-wise calibration use the same 128 sequences of 2048 tokens sampled from the WikiText-2 training split. This set is disjoint from the perplexity evaluation text and labeled benchmark examples.

Fixed method settings. Across C converted channels, $\rho \in [0, b_{\max} - b_0]$ sets the target average number of one-level resolution increases, giving $B = \mathrm{round}(\rho C)$ and $\sum_c (b_c - b_0) \le B$. By default, the two allocation rounds receive $B^{(1)} = \mathrm{round}(0.4B)$ and $B^{(2)} = B - B^{(1)}$. If eligible capacity is smaller than B, we leave the remainder unassigned rather than upgrading channels constant on the calibration set. Random allocation uses the same b_0, $b_{\max}$, and round budgets as TARC. Within each round, it samples uniformly among the next available one-level increases, allowing a channel's later increase only after its earlier one has been selected. Once resolutions are assigned, we calibrate each layer with three passes over the calibration set using AdamW, one sequence per update, a fixed learning rate of $5{\times}10^{-3}$, and $\alpha_{\min} = 0.5$. We sample each blockwise randomized Hadamard transform once and hold it fixed. Within each converted Transformer layer, these transforms precede SNN encoding at linear-layer inputs and post-RoPE Q/K activation sites; post-projection V activations are encoded without a Hadamard transform (Ashkboos et al., 2024; Lin et al., 2024). All rows in Fig. 4 use the same transforms.

Baseline implementation. For LLaMA models, we use released configurations for the target architecture when available and otherwise adapt the reported conversion rule. For Qwen models, we include only methods with released implementations supporting the target model. Each baseline uses the encoding timesteps listed in the main tables, without per-benchmark retuning. For each target model, all baselines use that model's tokenizer and share the sequence length, random seed, and evaluation harness. The FP16 rows evaluate the original ANN checkpoints under the same protocol.

Per-task results. Table 5 breaks down CS-Avg. into its six tasks for the five LLaMA models and reports their MMLU accuracy.

Table 5: Per-task accuracy (%) for the five LLaMA models in Table 2. Top three converted results per model and metric are highlighted as first, second, and third.

Model	Method	T	ARC-e	ARC-c	BoolQ	Hella Swag	PIQA	Wino Grande	CS-Avg.	MMLU
LLaMA-2-7B	FP16	–	74.49	46.25	77.77	75.99	79.05	68.82	70.40	45.93
	TTFSFormer	64	69.61	41.38	69.05	72.15	76.71	65.90	65.80	36.22
	SpikedAttention	20	62.63	37.20	67.06	70.53	74.65	63.30	62.56	31.18
	LAS	8	69.07	39.51	69.91	66.16	76.12	61.09	63.64	28.18
	SpikeLLM	4	62.04	36.43	67.34	68.07	71.38	59.67	60.82	32.18
	SpikingLLM	6	72.85	45.22	77.68	76.41	77.42	67.32	69.48	32.87
	$\text{TARC}_{\rho=0.2}$	4	72.77	43.09	75.84	74.11	77.86	65.75	68.24	41.27
	$\text{TARC}_{\rho=0.5}$	4	72.90	44.11	76.51	74.97	78.24	68.43	69.19	42.19
	$\text{TARC}_{\rho=1.0}$	4	73.65	45.05	77.03	75.38	78.94	68.03	69.68	43.97
LLaMA-2-13B	FP16	–	77.48	49.15	80.55	79.39	80.52	72.14	73.20	54.91
	TTFSFormer	64	74.66	45.56	74.71	76.01	78.35	70.64	69.99	47.73
	SpikedAttention	20	67.17	43.86	70.70	75.41	76.77	65.11	66.50	41.00
	LAS	8	71.80	45.48	68.13	70.00	76.77	58.33	65.08	30.46
	SpikeLLM	4	67.42	42.32	72.81	73.96	74.70	64.80	66.00	41.32
	SpikingLLM	6	75.93	49.74	81.35	78.19	80.74	71.19	72.86	35.63
	$\text{TARC}_{\rho=0.2}$	4	75.38	47.01	78.53	77.68	79.82	70.17	71.43	52.51
	$\text{TARC}_{\rho=0.5}$	4	76.56	47.44	80.31	78.41	79.43	70.64	72.13	53.08
	$\text{TARC}_{\rho=1.0}$	4	76.52	48.89	80.55	78.87	79.54	71.43	72.63	54.32
LLaMA-2-70B	FP16	–	80.85	57.34	83.67	83.78	82.86	77.90	77.73	68.72
	TTFSFormer	64	78.75	54.35	80.24	79.75	80.85	74.98	74.82	57.83
	SpikedAttention	20	74.45	50.17	71.96	76.50	77.86	72.77	70.62	55.57
	LAS	8	61.20	36.52	43.64	38.32	70.46	52.64	50.46	25.32
	SpikeLLM	4	66.41	44.54	74.40	74.23	76.12	66.77	67.08	49.06
	SpikingLLM	6	77.69	56.23	78.96	82.53	81.18	77.27	75.64	45.71
	$\text{TARC}_{\rho=0.2}$	4	79.50	56.57	82.51	82.80	82.64	75.93	76.66	66.84
	$\text{TARC}_{\rho=0.5}$	4	80.30	56.48	82.97	83.17	83.08	77.35	77.23	67.69
	$\text{TARC}_{\rho=1.0}$	4	80.35	57.51	83.03	83.31	82.97	77.90	77.51	68.03
LLaMA-3-8B	FP16	–	77.74	53.41	81.38	79.13	80.79	72.69	74.19	65.20
	TTFSFormer	64	68.98	41.30	58.93	61.35	74.92	63.54	61.50	29.87
	SpikedAttention	20	59.13	37.63	68.20	65.84	70.62	63.85	60.88	37.49
	LAS	8	53.03	34.13	60.24	59.08	66.76	54.30	54.59	26.94
	SpikeLLM	4	50.34	32.94	52.57	56.55	61.97	55.41	51.63	29.65
	SpikingLLM	6	76.68	51.19	80.28	78.18	79.33	72.30	72.99	62.66
	$\text{TARC}_{\rho=0.2}$	4	77.23	50.51	81.22	77.73	79.87	72.38	73.16	61.87
	$\text{TARC}_{\rho=0.5}$	4	77.90	51.45	80.80	77.82	80.09	73.16	73.54	62.94
	$\text{TARC}_{\rho=1.0}$	4	78.54	52.22	80.37	78.66	80.08	73.48	73.89	63.91
LLaMA-3-70B	FP16	–	85.86	64.33	85.23	84.89	84.55	80.35	80.87	78.81
	TTFSFormer	64	79.84	54.78	78.32	78.74	79.98	74.27	74.32	53.35
	SpikedAttention	20	61.11	42.49	64.31	63.54	73.18	58.72	60.56	50.48
	LAS	8	28.49	24.15	38.84	26.17	49.95	50.04	36.27	22.95
	SpikeLLM	4	64.35	41.13	54.07	71.25	74.21	56.99	60.33	33.39
	SpikingLLM	6	77.19	53.07	83.06	79.71	79.54	72.61	74.20	61.97
	$\text{TARC}_{\rho=0.2}$	4	83.04	59.90	84.71	83.84	82.37	77.58	78.57	76.54
	$\text{TARC}_{\rho=0.5}$	4	84.85	61.52	85.20	84.45	83.73	79.56	79.89	77.60
	$\text{TARC}_{\rho=1.0}$	4	84.64	60.84	85.44	84.65	84.17	79.24	79.83	78.09

Seed robustness. We repeat the LLaMA-3-8B conversion at $\rho = 0.5$ with seeds 0, 1, and 2. Within each seed, Random and TARC share the calibration inputs and fixed randomized Hadamard transforms, differing in their resolution-allocation rule. TARC has lower PPL on both WikiText-2 and C4 in all three paired runs (Table 6).

Table 6: Seed robustness on LLaMA-3-8B at ρ=0.5 with layer-wise calibration. The final row gives the mean $\pm$ standard deviation of paired PPL reductions (Random minus TARC) over three seeds.

Seed	Wiki-2 PPL ↓		C4 PPL ↓	
	Random	TARC	Random	TARC
0	6.80	**6.58**	9.65	**9.39**
1	6.80	**6.54**	9.66	**9.33**
2	6.78	**6.54**	9.64	**9.35**
Mean PPL reduction	$\mathbf{0.24 \pm 0.02}$		$\mathbf{0.29 \pm 0.04}$	

C Proofs

C.1 Encoding validity and spike activity

Proof of Proposition 1. We first verify that NAF fits the b phase weights for every $q \in [-2^{b-1}, 2^{b-1}-1]$. If its highest nonzero digit had weight 2^h with $h \geq b$, lower nonadjacent digits could cancel less than 2^{h-1}, giving $|q| > 2^{h-1} \geq 2^{b-1}$, a contradiction. Thus we can write the NAF digits from Eq. (4) as $\mathbf{S}_q^{\mathrm{NAF}} = (\tilde{s}_1, \ldots, \tilde{s}_b)$.

Encoding validity. With $\omega_t = 2^{T-t}$, the first timestep combines the $b - T$ out-of-window NAF contributions with the first in-window contribution; later timesteps retain their NAF digits:

$$s_1 = \frac{1}{\omega_1} \sum_{j=1}^{b-T+1} \tilde{s}_j 2^{b-j}, \qquad s_t = \tilde{s}_{b-T+t} \quad (2 \leq t \leq T). \tag{14}$$

Every weight in the sum for s_1 is a multiple of ω_1, so s_1 is an integer. Grouping these NAF contributions leaves the represented value unchanged: $q = \sum_{t=1}^{T} \omega_t s_t$.

Let r be the total contribution of timesteps $2, \ldots, T$. Since $|s_t| \leq 1$ and $\sum_{t=2}^{T} \omega_t = \omega_1 - 1$, we have $|r| \leq \omega_1 - 1$. Writing $q = \omega_1 s_1 + r$ gives

$$|s_1| \leq \frac{|q| + |r|}{\omega_1} < 2^{b-T} + 1.$$

Since s_1 is an integer, it satisfies $|s_1| \leq 2^{b-T}$. Hence the construction is feasible under the amplitude constraints in Section 4.1. Reducing the first-step bound to $2^{b-T} - 1$ would limit the total magnitude to $2^{b-1} - 1$, excluding the required endpoint -2^{b-1}.

For active-timestep optimality, we temporarily allow any integer first-step amplitude. Since SBP already meets the required bound, optimality in this larger class implies the constrained claim in Proposition 1.

Lemma 1 (Minimum active-timestep count)**.** *The SBP construction minimizes $\|S\|_0$ among length-T spike trains representing q, with integer s_1 and $s_t \in \{-1, 0, +1\}$ for $t \geq 2$.*

Proof. Let $C_L(m)$ be the minimum active-timestep count for representing m over L timesteps, with weights 2^{L-t}, an unrestricted integer first amplitude, and later amplitudes in $\{-1, 0, +1\}$. Then $C_1(m) = \mathbb{1}[m \neq 0]$. For $L > 1$, an even m requires a zero last amplitude, while an odd m requires $+1$ or -1. Therefore,

$$\begin{aligned} C_L(2a) &= C_{L-1}(a), \\ C_L(2a+1) &= 1 + \min\{C_{L-1}(a), C_{L-1}(a+1)\}. \end{aligned} \tag{15}$$

These recurrences imply

$$C_L(2a) \leq C_L(2a+1) \leq C_L(2a) + 1, \qquad C_L(2a+2) \leq C_L(2a+1) \leq C_L(2a+2) + 1.$$

Indeed, adjacent values of C_1 differ by at most one. If the same holds at level $L-1$, the odd recurrence lies between each neighboring even value and that value plus one, establishing the bounds and the induction.

To connect this minimum to SBP, we apply the block rewrite in Eq. (5) to the ordinary binary expansion of $|q|$ with q's sign, always choosing the rightmost maximal run of adjacent same-sign digits. The equivalent digit-wise NAF recurrence starts from $m = q$ and selects each digit u from lowest to highest weight before updating the residual:

$$u = \begin{cases} 0, & m \equiv 0 \pmod 2, \\ 1, & m \equiv 1 \pmod 4, \\ -1, & m \equiv 3 \pmod 4, \end{cases} \qquad m \leftarrow (m-u)/2.$$

For odd m, either sign incurs one active timestep; the NAF rule chooses the even remaining quotient: $2a$ rather than $2a+1$ for $m = 4a+1$, and $2a+2$ rather than $2a+1$ for $m = 4a+3$. The bounds above show that this choice never increases the minimum cost of the remaining positions. After $T-1$ lower digits, the remaining quotient is the first-step amplitude in Eq. (14), so SBP attains $C_T(q)$ and proves the first claim of Proposition 1. □

Unit spike count. For the SBP train, write $a_{\mathrm{SBP}} = s_1$, $k_{\mathrm{SBP}} = \sum_{t=2}^{T} |s_t|$, and $\Gamma_{\mathrm{SBP}} = |a_{\mathrm{SBP}}| + k_{\mathrm{SBP}}$. For any other feasible train S representing q, denote the corresponding quantities by a and k. Both residuals $q - \omega_1 a_{\mathrm{SBP}}$ and $q - \omega_1 a$ have magnitude at most $\omega_1 - 1$, so $|a_{\mathrm{SBP}} - a| \le 1$. Lemma 1 also gives

$$\mathbb{1}[a_{\mathrm{SBP}} \neq 0] + k_{\mathrm{SBP}} \le \mathbb{1}[a \neq 0] + k.$$

At the first timestep, $|a| - \mathbb{1}[a \neq 0]$ counts unit spikes beyond the first. Since the first amplitudes differ by at most one, we also have

$$|a_{\mathrm{SBP}}| - \mathbb{1}[a_{\mathrm{SBP}} \neq 0] \le |a| - \mathbb{1}[a \neq 0] + 1.$$

Adding this to the active-timestep inequality gives $\Gamma_{\mathrm{SBP}} \le \Gamma(S) + 1$. Taking the minimum over feasible S proves the second claim of Proposition 1. The constant is tight: for $T = 3$, $b = 4$, and $q = 6$, SBP gives $(2, -1, 0)$ with $\Gamma = 3$, whereas $(1, 1, 0)$ has $\Gamma = 2$ under the same constraints. □

C.2 Derivation and optimality of resolution allocation

Score derivation. Consider one allocation round, in which only channels with nonconstant calibration activations are eligible. Index each eligible channel at its activation site by c, and write $\mathbf{b} = (b_c)_c$ for their resolutions. Hold each channel's activation range, of width $r_c > 0$, and sensitivity F_c fixed during the round. All sums below run over these channels. At resolution b, write $V_{\theta,c}^{(b)} := r_c/(2^b - 1)$ for channel c's base threshold.

Fix a calibration example and an activation site. Let $\mathbf{a}$ be the clipped activations before rounding, and let $\mathcal{L}(\mathbf{a})$ denote the task loss as a function of these activations. The modeled rounding error $\boldsymbol{\epsilon}$ changes them to $\mathbf{a} + \boldsymbol{\epsilon}$. For channel c at resolution b, the uniform-noise model (Bennett, 1948) gives

$$\epsilon_{c,b} \mid \mathbf{a} \sim \mathcal{U}\left[-\frac{V_{\theta,c}^{(b)}}{2}, \frac{V_{\theta,c}^{(b)}}{2}\right], \qquad \mathbb{E}_\epsilon[\epsilon_{c,b} \mid \mathbf{a}] = 0,$$

$$\mathbb{E}_\epsilon[\epsilon_{c,b}^2 \mid \mathbf{a}] = \frac{1}{V_{\theta,c}^{(b)}} \int_{-V_{\theta,c}^{(b)}/2}^{V_{\theta,c}^{(b)}/2} u^2 \, du = \frac{(V_{\theta,c}^{(b)})^2}{12} = \frac{r_c^2}{12(2^b-1)^2}.$$

For scoring, treat $\mathcal{L}$ as locally twice differentiable in these activations. Its second-order expansion around $\mathbf{a}$ gives (LeCun et al., 1990)

$$\mathcal{L}(\mathbf{a} + \boldsymbol{\epsilon}) - \mathcal{L}(\mathbf{a}) \approx \nabla_{\mathbf{a}} \mathcal{L}(\mathbf{a})^\top \boldsymbol{\epsilon} + \frac{1}{2} \boldsymbol{\epsilon}^\top \nabla_{\mathbf{a}}^2 \mathcal{L}(\mathbf{a}) \boldsymbol{\epsilon}.$$

Taking the conditional expectation over $\boldsymbol{\epsilon}$ removes the linear term but leaves the Hessian-weighted quadratic term. To obtain an additive per-channel score, we retain its diagonal terms and use the nonnegative empirical Fisher F_c (Section 4.2) as a proxy for their curvature coefficients (Kirkpatrick

et al., 2017; Dong et al., 2019; 2020). We estimate F_c on the current SNN with the backward rule in Appendix A.1 and hold it fixed within the round. Since the modeled noise variance does not depend on $\mathbf{a}$, averaging over calibration examples and summing over channels gives the loss-increase surrogate

$$\widehat{\Delta\mathcal{L}}_{\mathrm{inc}}(\mathbf{b}) := \frac{1}{2}\sum_c F_c\,\mathbb{E}_\epsilon[\epsilon^2_{c,b_c}] = \frac{1}{24}\sum_c \frac{F_c r_c^2}{(2^{b_c}-1)^2}. \tag{16}$$

For a candidate with $b_c = b$, let $\mathbf{b}'$ match $\mathbf{b}$ except that $b'_c = b+1$. All other terms in Eq. (16) cancel:

$$\begin{aligned}\widehat{\Delta\mathcal{L}}_{c,b} &:= \widehat{\Delta\mathcal{L}}_{\mathrm{inc}}(\mathbf{b}) - \widehat{\Delta\mathcal{L}}_{\mathrm{inc}}(\mathbf{b}') \\ &= \frac{F_c r_c^2}{24}\left[\frac{1}{(2^b-1)^2} - \frac{1}{(2^{b+1}-1)^2}\right] = \frac{g_{c,b}}{24}.\end{aligned}$$

This is the estimated reduction in Eq. (8); multiplying by the common positive factor 24 gives the allocation score $g_{c,b}$ without changing the ranking.

The surrogate estimates only the reduction due to rounding error at fixed ranges. It omits clipping changes and later range calibration; Stage 2 uses the layer-output mismatch in Eq. (3), which reflects both errors.

Proof of Proposition 2. Consider the eligible channels with nonconstant calibration activations. Fix their $\{F_c, r_c\}$ and starting resolutions $\bar{\mathbf{b}} = (\bar{b}_c)_c$, where $\bar{b}_c \in \{b_0, \dots, b_{\max}\}$. Let $0 \le K \le \sum_c (b_{\max} - \bar{b}_c)$ be an integer budget. A feasible final allocation $\mathbf{b} = (b_c)_c$ has integer b_c satisfying $\bar{b}_c \le b_c \le b_{\max}$ for every channel and $\sum_c (b_c - \bar{b}_c) = K$. From Eq. (16), its estimated loss reduction is exactly the sum of its successive one-level scores:

$$\begin{aligned}\widehat{\Delta\mathcal{L}}_{\mathrm{inc}}(\bar{\mathbf{b}}) - \widehat{\Delta\mathcal{L}}_{\mathrm{inc}}(\mathbf{b}) &= \frac{1}{24}\sum_c F_c r_c^2\left[\frac{1}{(2^{\bar{b}_c}-1)^2} - \frac{1}{(2^{b_c}-1)^2}\right] \\ &= \frac{1}{24}\sum_c \sum_{j=\bar{b}_c}^{b_c-1} g_{c,j}.\end{aligned}$$

Thus each channel contributes a prefix of the candidates (c, j), from $j = \bar{b}_c$ through $j = b_c - 1$. To show that the K largest scores form such prefixes, factor their resolution-dependent term as

$$\frac{1}{(2^b-1)^2} - \frac{1}{(2^{b+1}-1)^2} = \frac{1}{(2^b-1)^2}\left[1 - \left(\frac{2^b-1}{2^{b+1}-1}\right)^2\right].$$

For $b \ge 1$, the ratio $(2^b-1)/(2^{b+1}-1) = 1/2 - 1/[2(2^{b+1}-1)]$ increases with b. Both positive factors on the right therefore decrease with b, making the difference positive and strictly decreasing. Whenever $F_c r_c^2 > 0$, a later increase has a lower score than every earlier increase of the same channel. A top-K selection then respects their order. If $F_c r_c^2 = 0$, all of that channel's scores are zero. Replacing any selected positions by the same number of earliest positions makes the selection a prefix without changing its gain or budget.

The K largest scores maximize the sum among all K-candidate selections. As the preceding argument makes this selection prefix-feasible, it attains the maximum over feasible allocations and hence minimizes Eq. (16), proving the claim. □

D SNN INFERENCE AND ENERGY ESTIMATION

D.1 SNN INFERENCE

Execution schedule. Converted sites follow the ANN graph. Each consumes complete input spike trains over its own T-timestep window, then quantizes and clips its output as in Eq. (2) and encodes the resulting integer as an SBP train using Eq. (7). Linear maps accumulate weighted spikes; nonlinear operators and attention use the updates below.

Nonlinear operators. Let $Z^{(t)}$ be the represented input accumulated through timestep t, including channel thresholds and offsets. As in prior Transformer conversions (You et al., 2024; Zheng et al.,

2026), RMSNorm and SwiGLU compute $Y^{(t)} = f(Z^{(t)})$ and accumulate the difference $Y^{(t)} - Y^{(t-1)}$, with $Y^{(0)}$ defined as zero for the first update. These differences sum to $Y^{(T)}$.

Attention. Q and V remain spike trains; K is accumulated once and held fixed as a centered-integer tensor. Q spikes update the logits against K; masked Softmax is then evaluated once per timestep. The K offset adds the same constant to all valid key logits and so leaves Softmax unchanged; the Q offset is retained in the logits. Logits, Softmax probabilities, represented V values, and context remain in floating point.

Let $P^{(t)}$ be the masked probabilities and $\widehat{V}^{(t)}$ the represented V values, including their offsets. New V spikes and probability changes update the context together:

$$\begin{aligned} \Delta C^{(1)} &= P^{(1)}\widehat{V}^{(1)}, \\ \Delta C^{(t)} &= P^{(t)}(\widehat{V}^{(t)} - \widehat{V}^{(t-1)}) \\ &\quad + (P^{(t)} - P^{(t-1)})\widehat{V}^{(t-1)}, \quad t \geq 2. \end{aligned} \tag{17}$$

The updates sum to $C^{(t)} = P^{(t)}\widehat{V}^{(t)}$. Because each probability row sums to one, the V offset contributes only at the first timestep. A V spike therefore contributes on arrival and to every later probability-change update, as counted below.

In exact arithmetic, the accumulated outputs at $t = T$ equal the ANN operator outputs on represented activations. Applying the same output quantization at each site preserves this equality through the network.

D.2 Operation counts and energy estimation

We estimate arithmetic energy per nonpadding token for batch-1 causal prefill from each configuration's frozen evaluation trace (Table 4). The evaluation pool described in Appendix B contains $N_{\text{tok}} = 813{,}056$ nonpadding tokens. A spike-train entry with signed count k represents $|k|$ same-sign unit spikes. The reported mean firing rate uses this count, $T^{-1}\mathbb{E}[\sum_{t=1}^{T} |s_t|]$, pooled over site–token–channel trains.

In the event path, each unit spike incurs one addition per valid destination. Let $\mathcal{E}$ index nonzero spike entries at operator inputs, s_e be the entry's amplitude, and f_e the number of valid destinations for one unit spike. The spike-driven additions per token are

$$n_{\text{spike}}^{\text{add}} = \frac{1}{N_{\text{tok}}} \sum_{e \in \mathcal{E}} |s_e| f_e. \tag{18}$$

The fan-out is the output width for a fixed-weight linear map, one for K accumulation, and the number of valid keys for QK. For a V spike at timestep t, it is the number of query–head pairs that attend to its key position and share its KV head, multiplied by $T - t + 1$. The $T - t + 1$ factor counts the spike's arrival and all later probability-change updates in Eq. (17).

For QK and PV, this rule counts only additions to floating-point logits and context. Coefficient formation, attention scaling, timestep-wise Softmax, probability differences, affine corrections, and value scaling are counted separately. The event-path totals also include affine quantization and encoding, fixed blockwise Hadamard transforms, timestep-wise RMSNorm/SwiGLU output differences, RoPE, residual additions, the final RMSNorm, and the LM head.

Per encoded scalar, phase coding uses $T + 1$ shifts, $T - 1$ logic operations, and one subtraction; SBP at $T = 4$ uses 9 shifts, 17 logic operations, 3 additions, 7 subtractions, and one comparison for Eq. (14). One power-of-two shift per active source–destination pair is reused across its $|k|$ unit additions. The Dense ANN row counts the original model's operations in the same classes.

The spike-driven additions in Eq. (18) and all other additions enter n_{low}, together with subtractions, comparisons, shifts, and logic operations. n_{mul} and n_{MAC} count multiplications and MACs; n_{func} counts exponential, reciprocal, reciprocal-square-root, SiLU, and rounding evaluations. Using 45-nm, 32-bit costs (Horowitz, 2014; You et al., 2024; Hwang et al., 2024; Zheng et al., 2026), charging every low-cost operation as an addition and each function evaluation as one MAC, we estimate

$$\widehat{E}_{\text{comp}} = 10^{-9}\big[0.9 n_{\text{low}} + 3.7 n_{\text{mul}} + 4.6\big(n_{\text{MAC}} + n_{\text{func}}\big)\big] \ \text{mJ/token}. \tag{19}$$

This arithmetic-only estimate excludes memory traffic, interconnect, controller/clocking, static energy, latency, and throughput.

Table 7: Per-token operation counts and analytical compute energy under 45-nm, 32-bit costs, with each function evaluation charged as one MAC. Phase/SBP use the event path; Dense ANN is the original pretrained model.

Configuration	n_{low}	n_{mul}	n_{MAC}	n_{func}	$\widehat{E}_{\text{comp}}$
	10^6 operations/token				mJ/token
Dense ANN	3.837	4.142	7773.225	1.509	35.783
Phase, $T = 4$	29858.336	17.646	525.861	7.478	29.391
Phase, $T = 5$	37654.275	21.579	525.992	8.986	36.430
SBP, $\rho = 0.2$	17898.993	17.646	525.468	7.478	18.626
SBP, $\rho = 0.5$	19429.700	17.646	525.468	7.478	20.004
SBP, $\rho = 1.0$	22033.601	17.646	525.468	7.478	22.347

At $T = 4$, the reduction in n_{low} accounts for most of SBP's lower estimated compute energy relative to phase coding; multiplication, MAC, and function counts change little. The ratios in Table 4 divide each configuration's estimate by that of Dense ANN.