SAT-LLM: CONVERSION-AWARE FINE-TUNING OF LARGE LANGUAGE MODELS FOR LOW-LATENCY SPIKING DEPLOYMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking large language models (LLMs) offer a promising path to energy-efficient deployment on neuromorphic hardware, while directly training such models remains prohibitively expensive. ANN-to-SNN conversion provides a practical alternative by reusing pretrained ANN weights, but existing approaches typically perform ANN training and SNN conversion as separate stages. This separation neglects the spiking dynamics during deployment in ANN optimization, thus causing substantial conversion error, particularly at low latency. To address this challenge, we propose SAT-LLM, a conversion-aware fine-tuning framework that incorporates the spiking behavior into ANN optimization. Specifically, SAT-LLM first constructs a Hadamard-rotated ANN to suppress activation outliers and introduces spiking-aware replacement sites across the Transformer blocks. At each replacement site, Phase or GIF neuronal dynamics are locally realized and aggregated, and common clipping constrains activations to the jointly representable range of candidate deployment neurons, with both operations performed during ANN fine-tuning, while temporal signals are not propagated across ANN layers. After fine-tuning, the ANN is converted into a full-temporal SNN with Phase, GIF, or MTN neurons, where temporal signals are propagated through the corresponding spiking operators. Theoretically, we characterize the approximation error between the hard and surrogate formulations and establish convergence guarantees for Phase-aware and GIF-aware fine-tuning. Furthermore, extensive experiments on the Qwen3-Base family and LLaMA3-8B show that SAT-LLM effectively improves the performance of low-latency spiking LLMs across diverse model architectures.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks (Yang et al., 2025; Grattafiori et al., 2024). However, Transformer based LLMs rely heavily on dense floating point multiply accumulate operations (Vaswani et al., 2017), leading to substantial computational and energy costs as model scale grows (De Vries, 2023). Inspired by biological neural systems, spiking neural networks (SNNs) communicate through sparse discrete spikes, enabling energy efficient computation on neuromorphic hardware (Merolla et al., 2014; Davies et al., 2018). This has motivated recent efforts to scale spiking computation to LLMs (Zheng et al., 2026). Yet directly training spiking LLMs remains prohibitively expensive, as their temporal dynamics require computation to unfold over multiple timesteps (Huang et al., 2025).

Research on SNNs has primarily followed two paradigms: direct training and ANN to SNN conversion (Chen et al., 2026a; Cao et al., 2015). Direct training learns the spiking network from scratch, but the associated time and computational costs become prohibitive at the scale of modern LLMs (Chen et al., 2026a). In contrast, ANN to SNN conversion starts from a pretrained ANN, reuses its learned weights, and replaces or approximates ANN activations and operators with their spiking counterparts (Rueckauer et al., 2017). To reduce the mismatch introduced during conversion, prior methods further employ techniques such as weight and threshold normalization, membrane correction, and calibration of converted neurons (Diehl et al., 2015; Hao et al., 2023a;b). Nevertheless, current pipelines generally optimize the ANN first and perform SNN construction or calibration afterward (You et al., 2024), leaving ANN optimization and SNN conversion as separate stages.

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.

Such a separated pipeline creates a mismatch between ANN optimization and subsequent spiking deployment. During ANN training, activations are optimized as continuous values, whereas after conversion they must be represented by discrete spike sequences under the dynamics of spiking neurons (Datta & Beerel, 2022; Bu et al., 2023). This discrepancy introduces conversion errors such as quantization error, unevenness error, and residual membrane error, which have been widely observed in ANN to SNN conversion (Bu et al., 2023; Hao et al., 2023a;b). These errors become particularly pronounced under a small timestep budget, where the limited temporal representation makes it more difficult for the converted SNN to faithfully reproduce ANN activations (Datta & Beerel, 2022; Hu et al., 2023). Consequently, optimizing the ANN without considering its subsequent spiking dynamics can lead to substantial performance degradation after conversion.

To address this issue, we propose SAT-LLM, a conversion-aware fine tuning framework that aligns ANN optimization with subsequent low timestep spiking deployment. Specifically, SAT-LLM first constructs a Hadamard-rotated ANN to suppress activation outliers (Liu et al., 2025; Ashkboos et al., 2024) and introduces spiking-aware replacement sites across the Transformer blocks. At these sites, Phase or GIF neuronal dynamics are locally realized and aggregated during ANN fine tuning, while common clipping constrains activations to the jointly representable range of candidate deployment neurons without propagating temporal signals across ANN layers. After fine tuning, the ANN is converted into a full-temporal SNN with Phase, GIF, or MTN neurons, where temporal signals are propagated through the corresponding spiking operators.

The main contributions of our work are summarized below:

- We propose SAT-LLM, a conversion-aware fine tuning framework for low latency spiking LLM deployment. SAT-LLM combines Hadamard rotation, spiking aware replacement sites, and common clipping to incorporate Phase or GIF neuronal dynamics into ANN fine tuning without propagating temporal signals across ANN layers, while supporting subsequent full-temporal SNN deployment with Phase, GIF, or MTN neurons.

- We provide a theoretical analysis of the proposed conversion-aware fine tuning process. We characterize the approximation error between the hard and surrogate formulations for Phase and GIF, and derive convergence guarantees for Phase aware and GIF aware fine tuning under standard local assumptions.

- We conduct extensive experiments on the Qwen3-Base family and LLaMA3-8B. The results demonstrate that SAT-LLM consistently improves the performance of converted spiking LLMs under low latency deployment, validating the effectiveness and generality of the proposed framework.

2 Related Work

2.1 ANN-to-SNN Conversion

ANN to SNN conversion provides an efficient route to obtain high performance SNNs by transferring pretrained ANN parameters rather than training spiking models from scratch. Early studies established effective mappings between ANN activations and spiking neurons (Deng & Gu, 2021), while subsequent work improved conversion fidelity through normalization (Sengupta et al., 2019), calibration (Li et al., 2021), and optimized potential initialization (Bu et al., 2022). To further reduce latency, quantization based approaches align discrete ANN activations with finite timestep spike representations (Li et al., 2022), whereas residual membrane mechanisms improve the utilization of information remaining after spike generation (Han et al., 2020). Beyond conventional rate coding, temporal coding schemes have been explored to increase representational efficiency under limited timesteps, including weighted spike coding and logarithmic temporal coding (Zhang et al., 2020). More recent methods further reduce conversion error through single spike phase coding (Hwang & Kung, 2024), differential coding with activation dependent threshold optimization (Huang et al., 2025), and distribution aware phase coding that adapts discrete representations to nonuniform activation distributions (Zheng et al., 2026).

2.2 Spiking Transformers and Large Language Models

Extending spiking computation from convolutional networks to Transformers introduces additional challenges due to self attention and nonlinear operators that are not directly compatible with conventional spiking dynamics. Early spiking Transformers primarily relied on direct training and redesigned spike driven attention or residual structures (Zhou et al., 2022; 2023), while language oriented models further explored knowledge distillation and autoregressive spiking architectures (Bal & Sengupta, 2024; Zhu et al., 2023). Conversion based approaches subsequently sought to preserve pretrained Transformer knowledge, with STA approximating nonlinear operators for training free conversion (Jiang et al., 2024), ECMT improving the conversion of Transformer components (Huang et al., 2024), and SpikedAttention developing spike based mechanisms for attention and Softmax (Hwang et al., 2024). At the LLM scale, SpikeLLM combines saliency aware spiking with quantized representations to scale spiking models to billions of parameters (Xing et al., 2025), while SpikeZIP-TF (You et al., 2024), FAS (Chen et al., 2025), and LAS (Chen et al., 2026a) further improve Transformer or LLM conversion through specialized conversion pipelines, calibration, spiking operators, and adaptive coding strategies. Despite these advances, most existing methods still treat ANN optimization and SNN conversion as separate stages. In contrast, SAT-LLM incorporates deployment oriented neuronal behavior into ANN fine tuning itself through local spiking realization and common clipping, while avoiding interlayer temporal propagation during optimization.

3 Preliminary

3.1 Hadamard Rotation

A Hadamard matrix is a square matrix with entries in $\{+1, -1\}$ and mutually orthogonal rows. We use its normalized form $\mathbf{H} \in \mathbb{R}^{d\times d}$ such that $\mathbf{H}\mathbf{H}^\top = \mathbf{I}$, which provides an orthogonal change of basis that helps redistribute channel-wise activation outliers (Ashkboos et al., 2024). For $d = 2^n$, the normalized Walsh–Hadamard matrix is recursively defined as

$$\mathbf{H}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \mathbf{H}_{2^n} = \mathbf{H}_2 \otimes \mathbf{H}_{2^{n-1}}, \tag{1}$$

where $\otimes$ denotes the Kronecker product.

For dimensions that are not powers of two, we use the catalog of known Hadamard matrices (Sloane, 1999). Specifically, for $d = 2^n m$, where a Hadamard matrix of order m is known, we construct

$$\mathbf{H}_d = \mathbf{H}_{2^n} \otimes \mathbf{H}_m. \tag{2}$$

Following Tseng et al. (2024), we further randomize the transform using $\mathbf{R} = \mathbf{H}\mathbf{D}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{s})$ and $\mathbf{s} \in \{-1, +1\}^d$ contains random signs. In SAT-LLM, these rotations are fixed during fine-tuning and, whenever possible, absorbed into adjacent linear projections, preserving the original ANN function while suppressing activation outliers for subsequent spiking conversion.

3.2 Spiking Neurons

In this subsection, we introduce the three spiking neuron models adopted in this work, namely Phase, Multi-Threshold Neuron (MTN), and GIF, which serve as the candidate neuronal dynamics for conversion-aware fine-tuning and subsequent low-latency SNN deployment.

Phase Neuron. Following phase coding (Kim et al., 2018; Zhang et al., 2020), given a temporal input sequence $\{I(t)\}_{t=1}^{T}$, the membrane potential is initialized from the accumulated temporal input and a temporally coded spike sequence is generated according to

$$\begin{aligned} v(1) &= \sum_{t=1}^{T} I(t), \qquad s(t) = \Theta\left(v(t) - \theta(t)\right), \\ v(t+1) &= v(t) - h(t)s(t), \qquad O(t) = d(t)s(t). \end{aligned} \tag{3}$$

Here, $\theta(t)$, $h(t)$, and $d(t)$ denote the timestep-dependent firing threshold, membrane subtraction, and output scale, respectively. In this work, we adopt

$$d(t) = \tau 2^{-t}, \qquad t = 1, \ldots, T, \tag{4}$$

where τ denotes the Phase scale.

Multi-Threshold Neuron. Following the Multi-Threshold Neuron (MTN) in Huang et al. (2025), let $\{\lambda_j\}_{j=1}^{2n}$ denote a set of n positive and n negative thresholds. The thresholds are initialized symmetrically as

$$\begin{aligned} \lambda_1 = \theta, \quad \lambda_2 = \frac{\theta}{2}, \quad \dots \quad \lambda_n = \frac{\theta}{2^{n-1}}, \\ \lambda_{n+1} = -\theta, \quad \lambda_{n+2} = -\frac{\theta}{2}, \quad \dots \quad \lambda_{2n} = -\frac{\theta}{2^{n-1}}, \end{aligned} \tag{5}$$

where $\theta > 0$ denotes the base threshold. The corresponding charge-and-fire dynamics are

$$\begin{gathered} m[t] = v[t-1] + I[t], \qquad s_j[t] = \mathrm{MTH}\left(m[t], j\right), \\ u[t] = \sum_{j=1}^{2n} s_j[t]\lambda_j, \quad v[t] = m[t] - u[t], \quad \mathrm{MTH}(x,i) = \begin{cases} 0, \text{if } \lambda_{2n} < x < \lambda_n \\ 1, \text{elif } i = \arg\min_p |x - \lambda_p| \\ 0, \text{else} \end{cases}. \end{gathered} \tag{6}$$

At each timestep, MTN integrates the incoming temporal signal into its membrane potential and activates at most one positive or negative threshold according to $\mathrm{MTH}(\cdot)$. In contrast to Phase coding's preload-and-discharge dynamics, MTN is intrinsically driven by temporally propagated inputs from preceding operators.

GIF Neuron. Following SpikeLLM (Xing et al., 2025), the Generalized Integrate-and-Fire (GIF) neuron improves the encoding efficiency of conventional IF neurons by merging multiple binary spiking steps into a single multi-level step. Specifically, every L consecutive IF steps are aggregated into one GIF step:

$$s_{\mathrm{GIF}}(t') = \sum_{t=1}^{L} s_{\mathrm{IF}}(t), \quad s_{\mathrm{GIF}}(t') = \begin{cases} k, & kV_{\mathrm{th}} \le Lv(t') < (k+1)V_{\mathrm{th}}, \quad k = 0, 1, \dots, L-1, \\ L, & v(t') \ge V_{\mathrm{th}}. \end{cases} \tag{7}$$

The merging reduces the recursive temporal length from T to $T' = T/L$, while allowing each GIF step to encode a multi-level output rather than a binary spike. The membrane state remains recursively propagated across successive GIF steps. Combined with saliency-aware spiking, salient channels can be assigned multiple GIF steps while the remaining channels use a one-step representation, providing additional temporal representation capacity where needed.

Additionally, the full-temporal conversion rules for LLM operations used during SNN deployment are detailed in Appendix C.

4 SAT-LLM

4.1 Conversion-Aware ANN Fine-Tuning

Spiking-aware ANN fine-tuning. SAT-LLM fine-tunes a functionally equivalent Hadamard-rotated ANN with spiking-aware replacement sites in the self-attention and MLP blocks (Figure 1). At each site, spiking behavior is locally realized while the surrounding ANN computation remains static.

Among the neurons in Section 3.2, Phase and GIF admit site-local realizations for a static ANN activation $\mathbf{Z}$. For Phase, we initialize $\mathbf{V}(1) = \mathbf{Z}$, unfold the neuronal dynamics for T timesteps, and replace $\mathbf{Z}$ with the aggregated output $\widehat{\mathbf{Z}} = \sum_{t=1}^{T} \mathbf{O}(t)$. For GIF, we use the quantization-equivalent realization of its merged multi-level representation. For an N-bit affine representation,

$$\begin{gathered} \Delta = \frac{x_{\max} - x_{\min}}{2^N - 1}, \qquad z = \mathrm{Round}\left(-\frac{x_{\min}}{\Delta}\right), \\ q(x) = \mathrm{clamp}\left(\mathrm{Round}\left(\frac{x}{\Delta}\right) + z, 0, 2^N - 1\right), \qquad \mathcal{Q}(x) = \left(q(x) - z\right)\Delta. \end{gathered} \tag{8}$$

Here, Δ and z are the quantization step size and zero point, respectively. During GIF-aware fine-tuning, $\mathbf{Z}$ is replaced by $\mathcal{Q}(\mathbf{Z})$, with rounding handled by a straight-through estimator. Thus, neither Phase nor GIF propagates temporal signals between ANN layers.

MTN instead requires temporally propagated upstream inputs and is therefore used only after conversion to the full-temporal SNN. The corresponding temporal LLM operators are detailed in Appendix C, and Algorithm 1 summarizes the overall procedure.

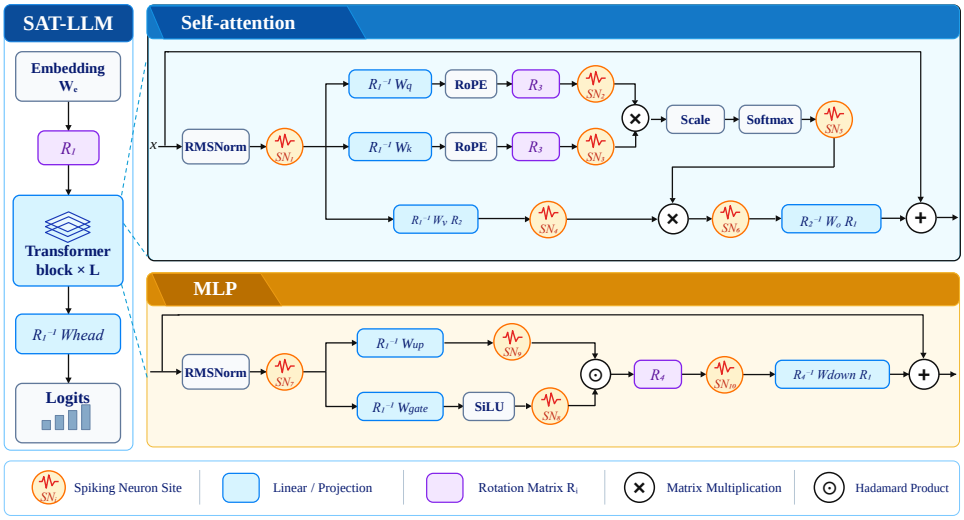


Figure 1: Overview of SAT-LLM. The left column summarizes the full model pipeline, while the right panels expand a Transformer block into its Self-attention and MLP submodules. The diagram highlights ten layer-local spiking neuron replacement sites, SN_1 through SN_{10}, used in conversion-aware fine-tuning and temporal deployment.

Neuron-independent clipping. To reduce the range mismatch between ANN fine-tuning and SNN deployment, we constrain eligible activations to the common representable range of Phase, MTN, and GIF.

For Phase, since $d(t) = \tau 2^{-t}$, its T-step signed representable interval is

$$\mathcal{I}^{\mathrm{Phase}} = \left[-\tau(1-2^{-T}),\ \tau(1-2^{-T})\right]. \tag{9}$$

For MTN, according to Eq. 5 and the definition of $\mathrm{MTH}(\cdot)$ in Eq. 6, at most one threshold is activated at each timestep. Hence, the corresponding T-step accumulated representable interval is

$$\mathcal{I}^{\mathrm{MTN}} = [-T\theta,\ T\theta]. \tag{10}$$

For GIF, with Δ and z from Eq. 8, define

$$\epsilon = z - \left(-\frac{x_{\min}}{\Delta}\right), \qquad -\frac{1}{2} \le \epsilon \le \frac{1}{2}. \tag{11}$$

The exact dequantized representable interval is

$$\mathcal{I}^{\mathrm{GIF}} = [x_{\min} - \epsilon\Delta,\ x_{\max} - \epsilon\Delta]. \tag{12}$$

The common clipping interval is

$$\begin{aligned} \mathcal{I}^{\mathrm{common}} &= \mathcal{I}^{\mathrm{Phase}} \cap \mathcal{I}^{\mathrm{MTN}} \cap \mathcal{I}^{\mathrm{GIF}} = [a, b], \\ a &= \max\left\{-\tau(1-2^{-T}), -T\theta, x_{\min} - \epsilon\Delta\right\}, \\ b &= \min\left\{\tau(1-2^{-T}), T\theta, x_{\max} - \epsilon\Delta\right\}. \end{aligned} \tag{13}$$

During fine-tuning, eligible activations are clipped element-wise to $[a, b]$ to reduce the range mismatch between ANN fine-tuning and subsequent SNN deployment.

5 Convergence Analysis

We analyze the optimization behavior of the conversion-aware ANN fine-tuning stage for the Phase-aware and GIF-aware training modes. Let $m \in \{\mathrm{P}, \mathrm{G}\}$ denote the Phase-aware and GIF-aware

Algorithm 1 SAT-LLM: Conversion-aware ANN fine-tuning and full-temporal SNN deployment

Require: Pretrained ANN weights $\mathcal{W}$; downstream dataset $\mathcal{D}$; calibration subset $\mathcal{D}_{\mathrm{cal}}$; learning rate η; epochs K; timestep T; rotation types $\mathcal{P} = \{1, 2, 3, 4\}$; training mode $m \in \{\mathrm{Phase}, \mathrm{GIF}\}$; active replacement sites $\mathcal{A}_m$; deployment neuron $\nu \in \{\mathrm{Phase}, \mathrm{GIF}, \mathrm{MTN}\}$
Ensure: Fine-tuned weights $\mathcal{W}^\star$ and a full-temporal SNN

Hadamard rotation.
for all layers ℓ and rotation types $p \in \mathcal{P}$ **do**
  Initialize and freeze the randomized Hadamard rotation R_p^ℓ.
  Absorb each eligible rotation into the adjacent linear projections.
end for

Conversion-aware ANN fine-tuning.
Calibrate and freeze the neuron parameters on $\mathcal{D}_{\mathrm{cal}}$, and compute the common clipping bounds by Eq. 13.
for epoch $k = 1, \ldots, K$ **do**
  for all mini-batches $\mathcal{B} \subset \mathcal{D}$ **do**
    for all active replacement sites $i \in \mathcal{A}_m$ encountered in the forward pass **do**
      Obtain the actual site input activation $\mathbf{Z}_i$.
      if $m = \mathrm{Phase}$ **then**
        Initialize $\mathbf{V}_i(1) \leftarrow \mathbf{Z}_i$, unfold the Phase dynamics in Eq. 3, and set $\widehat{\mathbf{Z}}_i \leftarrow \sum_{t=1}^{T} \mathbf{O}_i(t)$.
      else if $m = \mathrm{GIF}$ **then**
        $\widehat{\mathbf{Z}}_i \leftarrow \mathcal{Q}(\mathbf{Z}_i)$ using Eq. 8.
      end if
      $\widetilde{\mathbf{Z}}_i \leftarrow \mathrm{clip}(\widehat{\mathbf{Z}}_i; \mathbf{a}_i, \mathbf{b}_i)$ if clipping applies at site i; otherwise, $\widetilde{\mathbf{Z}}_i \leftarrow \widehat{\mathbf{Z}}_i$.
      Use $\widetilde{\mathbf{Z}}_i$ as the site output and continue the ANN forward pass.
    end for
    Compute the downstream task loss $\mathcal{L}(\mathcal{W}; \mathcal{B})$.
    Backpropagate using the sigmoid surrogate gradient for Phase or the rounding STE for GIF.
    $\mathcal{W} \leftarrow \mathcal{W} - \eta \nabla_{\mathcal{W}} \mathcal{L}$.
  end for
end for
$\mathcal{W}^\star \leftarrow \mathcal{W}$. No temporal signals are propagated between ANN layers.

Full-temporal SNN deployment.
Instantiate deployment neuron ν at the designated spiking sites with its corresponding temporal dynamics.
Propagate temporal signals end-to-end using the full-temporal conversion rules in Appendix C.

modes, respectively, and let $\mathbf{W}$ denote all trainable ANN parameters. The auxiliary constructions, assumptions, and proofs used below are deferred to Appendix B.

For a training sample ξ, let $f_m(\mathbf{W}; \xi)$ denote the actual ANN forward pass under mode m, and define the hard training objective

$$\mathcal{L}_m(\mathbf{W}) = \mathbb{E}_{\xi \sim \mathcal{D}} \left[\ell \left(f_m(\mathbf{W}; \xi), y_\xi \right) \right]. \tag{14}$$

Because Phase contains a hard firing function and GIF contains a non-differentiable rounding operation, the backward pass uses surrogate derivatives. We therefore introduce a smooth proxy network $\widetilde{f}_m$ and its objective

$$\widetilde{\mathcal{L}}_m(\mathbf{W}) = \mathbb{E}_{\xi \sim \mathcal{D}} \left[\ell \left(\widetilde{f}_m(\mathbf{W}; \xi), y_\xi \right) \right]. \tag{15}$$

For Phase, the proxy replaces the hard firing function by a sigmoid function; for GIF, it removes rounding while retaining the same affine scale, zero point, and saturation range. Common clipping, whenever enabled, is identical in the hard and proxy networks.

The analysis proceeds in two stages. We first quantify the approximation error between the proxy and hard operators at individual replacement sites, and then propagate these local errors to the overall objective before combining them with the optimization behavior of the proxy objective.

For Phase, this local approximation error is characterized by the firing margin, which measures the distance of the hard membrane trajectory from the firing threshold.

Lemma 5.1 (Phase surrogate approximation)**.** *For every Phase site i, with the maximum norm defined in Eq. 20 and the hard map P_i and its corresponding proxy map $\widetilde{P}_i$ defined in Eq. 21, there exists a*

finite scalar constant $K_i^{\mathrm{P}} > 0$ defined in Eq. 34, such that

$$\left\| P_i(\mathbf{X}) - \widetilde{P}_i(\mathbf{X}) \right\|_{\max} \leq K_i^{\mathrm{P}} \Psi_{\mathrm{P}} \left(\gamma_i^{\mathrm{P}}(\mathbf{X}) \right), \tag{16}$$

where γ_i^{P} is the firing margin defined in Eq. 23, and Ψ_{P} is the scalar error function defined in Eq. 24.

For GIF, the hard and proxy operators share the same affine scale, zero point, saturation range, and dequantization; their only difference is the rounding operation. Consequently, the approximation error admits a simpler bound that depends directly on the effective quantization scale.

Lemma 5.2 (GIF surrogate approximation)**.** *For every GIF replacement site i, with the hard quantizer Q_i and its corresponding proxy quantizer $\widetilde{Q}_i$ defined in Eqs. 36–37, we have*

$$\left\| Q_i(\mathbf{X}) - \widetilde{Q}_i(\mathbf{X}) \right\|_{\max} \leq \frac{1}{2} \|\mathbf{s}_i\|_\infty, \tag{17}$$

where $\mathbf{s}_i$ is the effective scale vector defined in Eq. 35.

Together, Lemmas 5.1 and 5.2 establish local approximation guarantees at individual replacement sites. To lift these local errors to the training loss, Appendix B.4 constructs hybrid networks that replace one active branch at a time and quantify the resulting downstream sensitivity. Aggregating these contributions yields the mode-specific term $\Delta_m(\mathbf{W})$ used below.

Theorem 5.3 (Objective gap between the hard and proxy formulations)**.** *Let $\Delta_m(\mathbf{W})$ be the mode-specific approximation term defined in Eqs. 50 and 51 of Appendix B.4. Under Assumption B.1, for either $m \in \{\mathrm{P}, \mathrm{G}\}$ and any $\mathbf{W}$, we have*

$$\left| \mathcal{L}_m(\mathbf{W}) - \widetilde{\mathcal{L}}_m(\mathbf{W}) \right| \leq \Delta_m(\mathbf{W}). \tag{18}$$

Theorem 5.3 connects the local neuronal approximation errors to the global training objective. Under the optimization conditions in Appendix B, combining this objective gap with the convergence behavior of the proxy objective yields the convergence guarantee for the actual hard objective.

Theorem 5.4 (Convergence of spiking-aware ANN fine-tuning)**.** *Suppose Assumptions B.1, B.2, and B.3 hold, and assume that the hard objective admits a minimizer $\mathbf{W}_m^\star$. Let $0 < \eta \leq 1/M_m$ and $\mu_m \leq M_m$. Then*

$$\begin{aligned} \mathbb{E}\left[\mathcal{L}_m(\mathbf{W}_K) - \mathcal{L}_m^\star\right] \leq (1 - \eta\mu_m)^K \left(\widetilde{\mathcal{L}}_m(\mathbf{W}_0) - \widetilde{\mathcal{L}}_m^\star \right) \\ + \frac{B_m^2}{2\mu_m} + \frac{M_m \eta \sigma_m^2}{2\mu_m} + \mathbb{E}\left[\Delta_m(\mathbf{W}_K)\right] + \Delta_m(\mathbf{W}_m^\star), \end{aligned} \tag{19}$$

where M_m, μ_m, B_m, and σ_m^2 are defined in Assumptions B.2–B.3, $\widetilde{\mathcal{L}}_m^\star$ is defined in Eq. 57, $\mathbf{W}_m^\star$ and $\mathcal{L}_m^\star$ are defined in Eq. 104, and Δ_m is defined in Eqs. 50 and 51 of Appendix B.4.

Theorem 5.4 decomposes the overall convergence behavior into distinct sources of error. The first term decreases geometrically with the number of optimization steps, the next two terms capture the residual effects of surrogate-gradient bias and stochastic noise, and the final two terms transfer the proxy guarantee to the actual hard objective through Δ_m. The full derivation of all four results is provided in Appendix B.

6 Experiments

6.1 Experimental Setup

Datasets and Models. We consider two standard sets of open-source models and datasets. First, we fine-tune the pre-trained Qwen3-1.7B-Base and Qwen3-8B-Base models (Yang et al., 2025) on the Reddit TL;DR summarization dataset (Stiennon et al., 2020), and evaluate the fine-tuned models on the TL;DR test set using the ROUGE metric (Lin, 2004). Second, we fine-tune the pre-trained Llama3-8B model (Grattafiori et al., 2024) on the Tulu 3 SFT mixture dataset (Lambert et al., 2024), and evaluate its downstream performance using the lm-evaluation-harness (Gao et al., 2021). The evaluation includes six widely used benchmarks, namely AGIEval (Zhong et al., 2024), ARC-Challenge (ARC-C) (Clark et al., 2018), BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), TruthfulQA (TQA) (Lin et al., 2022), and WinoGrande (WinoG) (Sakaguchi et al., 2021).

Table 1: Performance of Qwen3-1.7B on the Reddit TL;DR dataset under different ANN fine-tuning and SNN conversion settings.

Training Mode	Evaluated Model	R-1	R-2	R-L	R-LSum	ROUGE (Avg.)
Base	Pre-trained ANN	0.2935	0.0974	0.2234	0.2245	0.2097
Vanilla	Fine-tuned ANN	0.3320	0.1169	0.2577	0.2578	0.2411
	Phase SNN	0.0383	0.0000	0.0314	0.0309	0.0252
	GIF SNN	0.0368	0.0003	0.0326	0.0339	0.0259
	MTN SNN	0.0236	0.0000	0.0208	0.0207	0.0163
Unaware	Fine-tuned ANN	0.3393	0.1182	0.2604	0.2607	0.2447
	Phase SNN	0.1392	0.0290	0.1127	0.1123	0.0983
	GIF SNN	0.0891	0.0023	0.0683	0.0689	0.0572
	MTN SNN	0.0582	0.0027	0.0497	0.0498	0.0401
Phase-aware	Fine-tuned ANN	0.2951	0.0899	0.2264	0.2266	0.2095
	Phase SNN	0.2943	0.0904	0.2235	0.2221	0.2076
	GIF SNN	0.1994	0.0404	0.1565	0.1571	0.1383
	MTN SNN	0.2413	0.0711	0.1934	0.1928	0.1746
GIF-aware	Fine-tuned ANN	0.3050	0.0967	0.2346	0.2345	0.2177
	Phase SNN	0.3143	0.0991	0.2388	0.2393	0.2229
	GIF SNN	0.3047	0.0959	0.2363	0.2368	0.2184
	MTN SNN	0.3011	0.0945	0.2259	0.2258	0.2118

Baselines. We consider four ANN fine-tuning modes for every model-task pair. As spiking-unaware baselines, vanilla performs conventional ANN fine-tuning without Hadamard rotation or incorporating any spiking neuronal behavior, whereas unaware applies Hadamard rotation before ANN fine-tuning but remains unaware of the subsequent spiking deployment. Therefore, unaware also serves as an ablation to isolate the effect of Hadamard rotation. Our proposed SAT-LLM corresponds to the other two modes, Phase-aware and GIF-aware, which incorporate the local neuronal behavior of Phase and GIF, respectively, into ANN fine-tuning together with the common clipping mechanism.

Experiment Configurations. All experiments are conducted on a single server equipped with four NVIDIA A100 GPUs, Intel Xeon CPUs, and 512 GB of system memory. Our implementation is based on Python 3.10, PyTorch 2.5.1, Transformers 4.53.2, and DeepSpeed 0.16.9. We use lm-evaluation-harness v0.4.8 for downstream evaluation and the Dao-AILab Fast Hadamard Transform implementation for efficient Hadamard rotations. For reproducibility, the hyperparameter configurations used in all experiments are summarized in Appendix D.1, with the common settings and learning rates reported in Tables 5 and 6, respectively. Additionally, to further mitigate activation outliers in LLMs, we adopt the prefixed outlier tokens technique proposed in Chen et al. (2026b).

6.2 Main Results

The results in Tables 1–3 show that conventional ANN fine-tuning is poorly suited for subsequent low-timestep SNN conversion. Despite the strong performance of the fine-tuned ANNs, direct conversion to Phase, GIF, or MTN SNNs results in substantial performance degradation. Hadamard rotation alone partially mitigates this degradation, but a considerable ANN-to-SNN gap remains. In contrast, Phase-aware and GIF-aware fine-tuning substantially reduce the conversion gap, demonstrating the effectiveness of incorporating spiking neuronal behavior into ANN optimization.

Additionally, both Phase-aware and GIF-aware fine-tuning exhibit cross-neuron generalization, in the sense that models optimized with one neuronal behavior can still be converted using other deployment neurons while retaining competitive performance. This generalization is more pronounced for GIF-aware fine-tuning, whose converted Phase and MTN SNNs generally retain performance close to that of the corresponding fine-tuned ANNs across different model-task settings. In comparison, Phase-aware models exhibit larger degradation when converted to GIF or MTN neurons, indicating that GIF-aware fine-tuning provides stronger generalization across different spiking neuron dynamics.

Table 2: Performance of Qwen3-8B on the Reddit TL;DR dataset under different ANN fine-tuning and SNN conversion settings.

Training Mode	Evaluated Model	R-1	R-2	R-L	R-LSum	ROUGE (Avg.)
Base	Pre-trained ANN	0.3216	0.1186	0.2508	0.2550	0.2365
Vanilla	Fine-tuned ANN	0.3526	0.1349	0.2781	0.2784	0.2610
	Phase SNN	0.1295	0.0156	0.1019	0.1015	0.0871
	GIF SNN	0.0569	0.0004	0.0498	0.0488	0.0390
	MTN SNN	0.1038	0.0069	0.0834	0.0845	0.0697
Unaware	Fine-tuned ANN	0.3531	0.1343	0.2785	0.2785	0.2611
	Phase SNN	0.2398	0.0690	0.1832	0.1827	0.1687
	GIF SNN	0.1382	0.0179	0.1096	0.1096	0.0938
	MTN SNN	0.1734	0.0398	0.1353	0.1348	0.1208
Phase-aware	Fine-tuned ANN	0.3212	0.1075	0.2464	0.2466	0.2305
	Phase SNN	0.3337	0.1071	0.2562	0.2547	0.2379
	GIF SNN	0.2285	0.0645	0.1865	0.1865	0.1665
	MTN SNN	0.2499	0.0715	0.1985	0.1976	0.1794
GIF-aware	Fine-tuned ANN	0.3384	0.1181	0.2596	0.2600	0.2440
	Phase SNN	0.3279	0.1088	0.2461	0.2458	0.2321
	GIF SNN	0.3407	0.1203	0.2639	0.2631	0.2470
	MTN SNN	0.3221	0.1071	0.2450	0.2445	0.2297

Table 3: Downstream performance of Llama3-8B fine-tuned on the Tulu 3 SFT mixture under different ANN fine-tuning and SNN conversion settings.

Training Mode	Evaluated Model	AGIEval	ARC-C	BoolQ	PIQA	TQA	WinoG	Avg.
Base	Pre-trained ANN	0.2610	0.3882	0.7012	0.7633	0.2632	0.6235	0.5001
Vanilla	Fine-tuned ANN	0.3420	0.4812	0.8376	0.8063	0.3586	0.6851	0.5851
	Phase SNN	0.2640	0.2927	0.5376	0.5968	0.2326	0.4909	0.4024
	GIF SNN	0.2500	0.2722	0.4367	0.5103	0.2375	0.5162	0.3705
	MTN SNN	0.2600	0.3114	0.4988	0.6273	0.2632	0.4980	0.4098
Unaware	Fine-tuned ANN	0.3120	0.4940	0.8483	0.8101	0.3672	0.7127	0.5907
	Phase SNN	0.2450	0.2995	0.6260	0.6317	0.3023	0.5312	0.4393
	GIF SNN	0.2470	0.2312	0.4266	0.5234	0.2570	0.4964	0.3636
	MTN SNN	0.2670	0.2577	0.5691	0.5941	0.2448	0.5122	0.4075
Phase-aware	Fine-tuned ANN	0.2660	0.3729	0.7446	0.7258	0.3219	0.5675	0.4998
	Phase SNN	0.2600	0.3643	0.7456	0.7073	0.3121	0.5556	0.4908
	GIF SNN	0.2710	0.2423	0.4193	0.5604	0.2472	0.4783	0.3698
	MTN SNN	0.2690	0.2927	0.5911	0.6425	0.3072	0.5304	0.4388
GIF-aware	Fine-tuned ANN	0.2580	0.3558	0.7446	0.7133	0.2938	0.5343	0.4833
	Phase SNN	0.2790	0.3754	0.7031	0.7018	0.3207	0.5485	0.4881
	GIF SNN	0.2600	0.3490	0.7453	0.6997	0.2889	0.5257	0.4781
	MTN SNN	0.2710	0.3370	0.7569	0.6741	0.3452	0.5359	0.4867

7 Conclusion

In this work, we proposed SAT-LLM, a conversion-aware fine-tuning framework for low-latency spiking deployment of large language models. SAT-LLM incorporates spiking neuron behavior into ANN fine-tuning through Hadamard rotation, spiking-aware replacement sites, and neuron-independent common clipping, rather than treating ANN optimization and SNN conversion as separate stages. Phase and GIF dynamics are locally realized during fine-tuning without interlayer temporal propagation, while the resulting model can be converted into a full-temporal SNN with Phase, GIF, or MTN neurons. We further established convergence guarantees for Phase-aware and GIF-aware fine-tuning. Experiments across different models and downstream tasks show that SAT-LLM mitigates the performance degradation caused by low-timestep ANN-to-SNN conversion.

AI USE STATEMENT

In this work, we used generative AI tools to assist in proof writing, provide feedback on research methodology and experimental design, support the implementation of methods, and facilitate the interpretation of results. We have not used generative AI tools for synthetic dataset generation or dataset cleaning and reformatting, and human-subject data collection and participant interaction are not applicable to this work. Additionally, we used generative AI tools for drafting and editing parts of the manuscript to improve readability, suggesting experimental parameters, and refining the presentation of scientific figures. We have reviewed all AI-assisted work. All AI-assisted mathematical derivations, methodological suggestions, implementation details, and interpretations of results were carefully reviewed and verified by the authors for correctness and consistency. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

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.

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

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 11–20, 2022.

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

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

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

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, 2026a.

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. Prefixquant: Eliminating outliers by prefixed tokens for large language models quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026b.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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

Gourav Datta and Peter A Beerel. Can deep neural networks be converted to ultra low-latency spiking neural networks? In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 718–723. IEEE, 2022.

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.

Alex De Vries. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194, 2023.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

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

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

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.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13555–13564. IEEE, 2020.

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.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023b.

Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):14546–14562, 2023.

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

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

Sangwoo Hwang and Jaeha Kung. One-spike snn: Single-spike phase coding with base manipulation for ann-to-snn conversion loss minimization. *IEEE Transactions on Emerging Topics in Computing*, 13(1):162–172, 2024.

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.

Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. Spatio-temporal approximation: A training-free snn conversion for transformers. In *International Conference on Learning Representations*, volume 2024, pp. 38769–38790, 2024.

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

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.

Chen Li, Lei Ma, and Steve Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pp. 6316–6325. PMLR, 2021.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

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

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.

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

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.

Neil JA Sloane. A library of hadamard matrices. *Manuscript available at http://research. att. com/njas/hadamard/index. html*, 1999.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024.

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.

Xingrun Xing, Boyan Gao, Zheng Liu, David 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*, volume 2025, pp. 32880–32899, 2025.

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. *arXiv preprint arXiv:2406.03470*, 2024.

Ming Zhang, Zonghua Gu, Nenggan Zheng, De Ma, and Gang Pan. Efficient spiking neural networks with logarithmic temporal coding. *IEEE access*, 8:98156–98167, 2020.

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.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the association for computational linguistics: NAACL 2024*, pp. 2299–2314, 2024.

Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv e-prints*, pp. arXiv–2304, 2023.

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

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

APPENDIX

A NOTATION AND MATRIX CONVENTION

The convergence analysis uses a single matrix abstraction for all replacement sites. A site activation is written as $\mathbf{X}$, and neuron parameters are represented as feature-wise vectors that are broadcast over the row-like axes of the activation. For a matrix $\mathbf{A}$, we use the entrywise maximum norm

$$\|\mathbf{A}\|_{\max} := \max_{n,d} |A_{nd}|. \tag{20}$$

For convenience, Table 4 summarizes the core notation used throughout Sec. 5 and Appendix B. Symbols introduced in Sec. 5 retain the same meaning in the proofs, while auxiliary quantities used only in the proofs are defined when first introduced.

Table 4: Core notation used in Sec. 5 and Appendix B.

Symbol	Meaning
$m \in \{\mathrm{P}, \mathrm{G}\}$	Phase-aware or GIF-aware ANN fine tuning mode.
$\mathbf{W}, \mathbf{W}_k$	Trainable ANN parameters and the iterate at step k.
$f_m, \widetilde{f}_m$	Hard ANN forward map and its corresponding proxy map.
$\mathcal{L}_m, \widetilde{\mathcal{L}}_m$	Hard and proxy expected objectives.
$P_i, \widetilde{P}_i$	Hard Phase map and its corresponding sigmoid relaxed map at site i.
$Q_i, \widetilde{Q}_i$	Hard GIF quantizer and its corresponding proxy without rounding.
C_i	Common clipping map, or the identity when clipping is inactive.
$\gamma_i^{\mathrm{P}}(\mathbf{X})$	Minimum absolute pre-threshold margin along the hard Phase trajectory at site i.
K_i^{P}	Local Phase approximation constant defined in Eq. 34.
$\gamma_r^{\mathrm{P}}(\mathbf{W})$	Uniform Phase firing margin of branch r over the data support.
$\varepsilon_r^m(\mathbf{W}; \xi)$	Per sample maximum norm difference between the hard and proxy outputs after branch r.
G_r	Local downstream sensitivity constant in Assumption B.1.
$\Delta_m(\mathbf{W})$	Mode specific bound on the objective gap between the hard and proxy formulations.
$\widehat{\mathbf{g}}_k^m$	Stochastic surrogate gradient at iteration k.
$M_m, \mu_m, B_m, \sigma_m^2$	Constants governing proxy smoothness, PL regularity, bias in the surrogate gradient, and variance of the stochastic gradient.
$\widetilde{\mathcal{L}}_m^\star$	Infimum of the proxy objective, defined in Eq. 57.
$\mathbf{W}_m^\star, \mathcal{L}_m^\star$	Minimizer and minimum value of the hard objective, defined in Eq. 104.

B PROOFS FOR THE CONVERGENCE ANALYSIS

B.1 PHASE APPROXIMATION AND PROOF OF LEMMA 5.1

Consider a Phase replacement site i with matrix input $\mathbf{X}$. Let $\boldsymbol{\theta}_{i,t}$, $\mathbf{h}_{i,t}$, and $\mathbf{d}_{i,t}$ denote the firing threshold, membrane subtraction, and output scale at timestep t, respectively. With $\mathbf{V}_1 = \widetilde{\mathbf{V}}_1 = \mathbf{X}$, the hard Phase dynamics and their corresponding smooth proxy are

$$\begin{aligned}
\mathbf{S}_t &= H\big(\mathbf{V}_t - \mathbf{1}_N \boldsymbol{\theta}_{i,t}^\top\big), & \widetilde{\mathbf{S}}_t &= \sigma_\beta\Big(\widetilde{\mathbf{V}}_t - \mathbf{1}_N \boldsymbol{\theta}_{i,t}^\top\Big), \\
\mathbf{V}_{t+1} &= \mathbf{V}_t - \mathbf{S}_t \odot \big(\mathbf{1}_N \mathbf{h}_{i,t}^\top\big), & \widetilde{\mathbf{V}}_{t+1} &= \widetilde{\mathbf{V}}_t - \widetilde{\mathbf{S}}_t \odot \big(\mathbf{1}_N \mathbf{h}_{i,t}^\top\big), \\
P_i(\mathbf{X}) &= \sum_{t=1}^{T} \mathbf{S}_t \odot \big(\mathbf{1}_N \mathbf{d}_{i,t}^\top\big), & \widetilde{P}_i(\mathbf{X}) &= \sum_{t=1}^{T} \widetilde{\mathbf{S}}_t \odot \big(\mathbf{1}_N \mathbf{d}_{i,t}^\top\big).
\end{aligned} \tag{21}$$

Here, $H(u) = \mathbb{1}\{u > 0\}$ and $\sigma_\beta(u) = (1 + \exp(-\beta u))^{-1}$ are applied element-wise, and the maximum norm is defined in Eq. 20.

Define the hard pre-threshold gap at timestep t and entry (n, d) by

$$\delta_{i,t,nd}^{\mathrm{P}}(\mathbf{X}) := [\mathbf{V}_t]_{nd} - \theta_{i,t,d}, \tag{22}$$

and the site-level hard firing margin by

$$\gamma_i^{\mathrm{P}}(\mathbf{X}) := \min_{1\le t\le T} \min_{n,d} \left|\delta_{i,t,nd}^{\mathrm{P}}(\mathbf{X})\right|. \tag{23}$$

Thus, $\gamma_i^{\mathrm{P}}(\mathbf{X})$ is the smallest absolute distance to any firing threshold encountered along the hard Phase trajectory generated from $\mathbf{X}$. Define

$$\Psi_{\mathrm{P}}(d) := \frac{1}{1+\exp(\beta d)}, \qquad d \ge 0. \tag{24}$$

For scalar u,

$$|H(u) - \sigma_\beta(u)| = \frac{1}{1+\exp(\beta|u|)} = \Psi_{\mathrm{P}}(|u|). \tag{25}$$

Moreover,

$$|\sigma'_\beta(u)| = \beta\sigma_\beta(u)(1-\sigma_\beta(u)) \le \frac{\beta}{4}, \tag{26}$$

so entrywise sigmoid is $\beta/4$-Lipschitz in the maximum norm.

Define the scalar membrane discrepancy

$$e_t := \|\mathbf{V}_t - \widetilde{\mathbf{V}}_t\|_{\max}, \tag{27}$$

and

$$\bar{h}_{i,t} := \|\mathbf{h}_{i,t}\|_\infty, \qquad \bar{d}_{i,t} := \|\mathbf{d}_{i,t}\|_\infty. \tag{28}$$

The two recursions share the same initial membrane matrix, so $e_1 = 0$. For timestep t,

$$\begin{aligned}\|\mathbf{S}_t - \widetilde{\mathbf{S}}_t\|_{\max} &\le \left\|H\left(\mathbf{V}_t - \mathbf{1}_N\boldsymbol{\theta}_{i,t}^\top\right) - \sigma_\beta\left(\mathbf{V}_t - \mathbf{1}_N\boldsymbol{\theta}_{i,t}^\top\right)\right\|_{\max} \\ &\quad + \left\|\sigma_\beta\left(\mathbf{V}_t - \mathbf{1}_N\boldsymbol{\theta}_{i,t}^\top\right) - \sigma_\beta\left(\widetilde{\mathbf{V}}_t - \mathbf{1}_N\boldsymbol{\theta}_{i,t}^\top\right)\right\|_{\max} \\ &\le \Psi_{\mathrm{P}}\left(\gamma_i^{\mathrm{P}}(\mathbf{X})\right) + \frac{\beta}{4}e_t. \end{aligned} \tag{29}$$

Using the membrane updates,

$$\begin{aligned} e_{t+1} &\le e_t + \bar{h}_{i,t}\|\mathbf{S}_t - \widetilde{\mathbf{S}}_t\|_{\max} \\ &\le \left(1 + \frac{\beta\bar{h}_{i,t}}{4}\right) e_t + \bar{h}_{i,t}\Psi_{\mathrm{P}}\left(\gamma_i^{\mathrm{P}}(\mathbf{X})\right). \end{aligned} \tag{30}$$

Define the scalar recurrence

$$c_{i,1} = 0, \qquad c_{i,t+1} = \left(1 + \frac{\beta\bar{h}_{i,t}}{4}\right) c_{i,t} + \bar{h}_{i,t}. \tag{31}$$

Induction gives

$$e_t \le c_{i,t}\Psi_{\mathrm{P}}\left(\gamma_i^{\mathrm{P}}(\mathbf{X})\right). \tag{32}$$

Therefore,

$$\begin{aligned}\left\|P_i(\mathbf{X}) - \widetilde{P}_i(\mathbf{X})\right\|_{\max} &\le \sum_{t=1}^{T} \bar{d}_{i,t}\|\mathbf{S}_t - \widetilde{\mathbf{S}}_t\|_{\max} \\ &\le \sum_{t=1}^{T} \bar{d}_{i,t}\left(1 + \frac{\beta c_{i,t}}{4}\right)\Psi_{\mathrm{P}}\left(\gamma_i^{\mathrm{P}}(\mathbf{X})\right). \end{aligned} \tag{33}$$

Defining

$$K_i^{\mathrm{P}} := \sum_{t=1}^{T} \bar{d}_{i,t}\left(1 + \frac{\beta c_{i,t}}{4}\right) \tag{34}$$

proves Lemma 5.1. □

B.2 GIF Approximation and Proof of Lemma 5.2

Consider a GIF replacement site i with matrix input $\mathbf{X}$. Let $\mathbf{x}_i^{\min}, \mathbf{x}_i^{\max}$ denote the calibrated feature-wise lower and upper bounds, respectively. For an N-bit asymmetric affine quantizer, define the effective scale and zero-point vectors as

$$\mathbf{s}_i = \frac{\mathbf{x}_i^{\max} - \mathbf{x}_i^{\min}}{2^N - 1}, \qquad \mathbf{z}_i = \mathrm{Round}\left(-\mathbf{x}_i^{\min} \oslash \mathbf{s}_i\right), \tag{35}$$

where $\oslash$ denotes element-wise division. The hard affine quantizer is

$$Q_i(\mathbf{X}) = \left[\Pi_i\left(\mathrm{Round}(\mathbf{X} \oslash \mathbf{s}_i) + \mathbf{z}_i\right) - \mathbf{z}_i\right] \odot \mathbf{s}_i, \tag{36}$$

where row-wise broadcasting is implicit and Π_i is the element-wise projection onto the integer range. The no-rounding proxy is

$$\widetilde{Q}_i(\mathbf{X}) = \left[\Pi_i\left(\mathbf{X} \oslash \mathbf{s}_i + \mathbf{z}_i\right) - \mathbf{z}_i\right] \odot \mathbf{s}_i. \tag{37}$$

Let X_{nd} denote a scalar entry of $\mathbf{X}$. Since projection onto an interval is non-expansive, for every row n and feature coordinate d,

$$\begin{aligned}\left|[Q_i(\mathbf{X})]_{nd} - [\widetilde{Q}_i(\mathbf{X})]_{nd}\right| &\le s_{i,d}\left|\mathrm{Round}\left(\frac{X_{nd}}{s_{i,d}}\right) - \frac{X_{nd}}{s_{i,d}}\right| \\ &\le \frac{s_{i,d}}{2} \le \frac{1}{2}\|\mathbf{s}_i\|_\infty.\end{aligned}$$

Taking the maximum over all matrix entries immediately gives

$$\left\|Q_i(\mathbf{X}) - \widetilde{Q}_i(\mathbf{X})\right\|_{\max} = \max_{n,d}\left|[Q_i(\mathbf{X})]_{nd} - [\widetilde{Q}_i(\mathbf{X})]_{nd}\right| \le \frac{1}{2}\|\mathbf{s}_i\|_\infty. \tag{38}$$

Thus the local GIF hard-to-proxy error does not introduce an explicit factor depending on either the row count N or the feature dimension D_i. For an identity GIF site, both maps equal the identity matrix map, so the approximation error is exactly zero. □

B.3 Non-expansiveness of Common Clipping

For every scalar coordinate, interval projection satisfies

$$|\,\mathrm{clip}(x; a, b) - \mathrm{clip}(y; a, b)| \le |x - y|. \tag{39}$$

Applying this inequality entrywise to two matrices and then taking the maximum over all entries gives

$$\|C_i(\mathbf{X}) - C_i(\mathbf{Y})\|_{\max} \le \|\mathbf{X} - \mathbf{Y}\|_{\max}. \tag{40}$$

Thus common clipping cannot enlarge either local approximation error in the entrywise maximum norm. □

B.4 Hybrid Construction and Uniform Local Replacement Errors

Let $\mathcal{A}_m = \{r_1, \ldots, r_{S_m}\}$ be the active replacement branches in reverse topological order, with r_1 the most downstream branch. For $q = 0, \ldots, S_m$, let $f_m^{(q)}$ be the hybrid network in which $r_1, \ldots, r_q$ use proxy operators and all remaining branches use hard operators:

$$f_m^{(0)} = f_m, \qquad f_m^{(S_m)} = \widetilde{f}_m. \tag{41}$$

For $r = r_q$, adjacent hybrids have the same branch input $\mathbf{X}_r(\mathbf{W}; \xi) \in \mathbb{R}^{N_r(\xi) \times D_r}$. Let Φ_r^m and $\widetilde{\Phi}_r^m$ denote the pre-clipping hard and proxy branch maps. Their post-branch outputs are

$$\begin{aligned}\mathbf{U}_r^{\mathrm{H}}(\mathbf{W}; \xi) &= C_r\left(\Phi_r^m(\mathbf{X}_r(\mathbf{W}; \xi))\right), \\ \mathbf{U}_r^{\mathrm{S}}(\mathbf{W}; \xi) &= C_r\left(\widetilde{\Phi}_r^m(\mathbf{X}_r(\mathbf{W}; \xi))\right).\end{aligned} \tag{42}$$

Here, C_r is the common clipping map when clipping is active at branch r and the identity map otherwise; its non-expansiveness follows from Eq. 40.

Define the sample-wise and uniform local replacement errors by

$$\begin{aligned} \varepsilon_r^m(\mathbf{W};\xi) &:= \left\|\mathbf{U}_r^{\mathrm{H}}(\mathbf{W};\xi) - \mathbf{U}_r^{\mathrm{S}}(\mathbf{W};\xi)\right\|_{\max}, \\ \varepsilon_r^m(\mathbf{W}) &:= \sup_{\xi\in\mathrm{supp}(\mathcal{D})} \varepsilon_r^m(\mathbf{W};\xi). \end{aligned} \tag{43}$$

For a Phase branch r, define the uniform firing margin over the data support by

$$\gamma_r^{\mathrm{P}}(\mathbf{W}) := \inf_{\xi\in\mathrm{supp}(\mathcal{D})} \gamma_r^{\mathrm{P}}\left(\mathbf{X}_r(\mathbf{W};\xi)\right). \tag{44}$$

For a Phase branch r and sample ξ, the local input is $\mathbf{X}_r(\mathbf{W};\xi)$. Lemma 5.1 therefore gives

$$\left\|P_r(\mathbf{X}_r) - \widetilde{P}_r(\mathbf{X}_r)\right\|_{\max} \le K_r^{\mathrm{P}}\Psi_{\mathrm{P}}\left(\gamma_r^{\mathrm{P}}\left(\mathbf{X}_r(\mathbf{W};\xi)\right)\right). \tag{45}$$

Because common clipping is non-expansive in the same norm,

$$\varepsilon_r^{\mathrm{P}}(\mathbf{W};\xi) \le K_r^{\mathrm{P}}\Psi_{\mathrm{P}}\left(\gamma_r^{\mathrm{P}}\left(\mathbf{X}_r(\mathbf{W};\xi)\right)\right). \tag{46}$$

By Eq. 44 and the fact that Ψ_{P} is monotonically non-increasing,

$$\varepsilon_r^{\mathrm{P}}(\mathbf{W}) \le K_r^{\mathrm{P}}\Psi_{\mathrm{P}}\left(\gamma_r^{\mathrm{P}}(\mathbf{W})\right). \tag{47}$$

For a quantized GIF branch, Lemma 5.2 and the same clipping argument give

$$\varepsilon_r^{\mathrm{G}}(\mathbf{W};\xi) \le \frac{1}{2}\|\mathbf{s}_r\|_\infty. \tag{48}$$

Since $\mathbf{s}_r$ is fixed before fine-tuning, the right-hand side is independent of ξ and $\mathbf{W}$, hence

$$\varepsilon_r^{\mathrm{G}}(\mathbf{W}) \le \frac{1}{2}\|\mathbf{s}_r\|_\infty. \tag{49}$$

Identity GIF branches have zero error exactly.

Let G_r denote the local downstream sensitivity constant from Assumption B.1. The mode-specific hard-to-proxy approximation terms are

$$\Delta_{\mathrm{P}}(\mathbf{W}) := \sum_{r\in\mathcal{A}_{\mathrm{P}}} G_r K_r^{\mathrm{P}}\Psi_{\mathrm{P}}\left(\gamma_r^{\mathrm{P}}(\mathbf{W})\right), \tag{50}$$

and, with $\mathcal{A}_{\mathrm{G}}$ denoting the non-identity GIF branches,

$$\Delta_{\mathrm{G}}(\mathbf{W}) := \frac{1}{2}\sum_{r\in\mathcal{A}_{\mathrm{G}}} G_r\|\mathbf{s}_r\|_\infty. \tag{51}$$

We write Δ_m for the corresponding mode-specific quantity.

B.5 Proof of Theorem 5.3

For a fixed parameter vector $\mathbf{W}$, sample ξ, and branch $r = r_q$, let $\mathcal{D}_{r,m}^{\mathbf{W},\xi}(\mathbf{U})$ denote the scalar downstream computation from the output matrix $\mathbf{U}$ of branch r to the sample loss. Because the two adjacent hybrids differ only at branch r, their downstream map is shared and

$$\begin{aligned} &\ell\left(f_m^{(q-1)}(\mathbf{W};\xi), y_\xi\right) - \ell\left(f_m^{(q)}(\mathbf{W};\xi), y_\xi\right) \\ &\quad = \mathcal{D}_{r,m}^{\mathbf{W},\xi}\left(\mathbf{U}_r^{\mathrm{H}}\right) - \mathcal{D}_{r,m}^{\mathbf{W},\xi}\left(\mathbf{U}_r^{\mathrm{S}}\right). \end{aligned} \tag{52}$$

Assumption B.1 (Local downstream sensitivity)**.** For every active branch r, there exists a finite constant $G_r > 0$, uniform over $\mathbf{W}$ and $\xi \in \mathrm{supp}(\mathcal{D})$, such that

$$\left|\mathcal{D}_{r,m}^{\mathbf{W},\xi}(\mathbf{U}) - \mathcal{D}_{r,m}^{\mathbf{W},\xi}(\mathbf{V})\right| \le G_r\|\mathbf{U}-\mathbf{V}\|_{\max} \tag{53}$$

for $\mathbf{U}$ and $\mathbf{V}$ on the line segment joining the hard and proxy post-branch outputs in Eq. 42.

Using the reverse-topological hybrid sequence,

$$\begin{aligned}
&\left|\ell\left(f_m(\mathbf{W};\xi),y_\xi\right)-\ell\left(\widetilde{f}_m(\mathbf{W};\xi),y_\xi\right)\right| \\
&\le \sum_{q=1}^{S_m}\left|\ell\left(f_m^{(q-1)}(\mathbf{W};\xi),y_\xi\right)-\ell\left(f_m^{(q)}(\mathbf{W};\xi),y_\xi\right)\right| \\
&\le \sum_{r\in\mathcal{A}_m} G_r\varepsilon_r^m(\mathbf{W};\xi) \le \sum_{r\in\mathcal{A}_m} G_r\varepsilon_r^m(\mathbf{W}) \le \Delta_m(\mathbf{W}),
\end{aligned} \tag{54}$$

where the last inequality follows from Eqs. 47–49 and the mode-specific definitions in Eqs. 50 and 51. Taking expectation over ξ and using $|\mathbb{E}[X]| \le \mathbb{E}[|X|]$ gives

$$\left|\mathcal{L}_m(\mathbf{W})-\widetilde{\mathcal{L}}_m(\mathbf{W})\right| \le \Delta_m(\mathbf{W}). \tag{55}$$

□

B.6 Proof of Theorem 5.4

The optimization argument uses the following regularity conditions along the optimization trajectory.

Assumption B.2 (Local regularity of the proxy objective)**.** For each $m \in \{\mathrm{P},\mathrm{G}\}$, $\widetilde{\mathcal{L}}_m$ satisfies:

1. $\widetilde{\mathcal{L}}_m$ is differentiable and M_m-smooth on every line segment joining two consecutive iterates generated by the update rule. In particular, for any $\mathbf{W}_1$ and $\mathbf{W}_2$ on such a segment,
$$\left\|\nabla_{\mathbf{W}}\widetilde{\mathcal{L}}_m(\mathbf{W}_1)-\nabla_{\mathbf{W}}\widetilde{\mathcal{L}}_m(\mathbf{W}_2)\right\|_2 \le M_m\|\mathbf{W}_1-\mathbf{W}_2\|_2. \tag{56}$$
2. The proxy objective is lower bounded, with
$$\widetilde{\mathcal{L}}_m^\star := \inf_{\mathbf{W}}\widetilde{\mathcal{L}}_m(\mathbf{W}) > -\infty. \tag{57}$$
3. The proxy objective satisfies a Polyak–Łojasiewicz condition at every iterate: for some $\mu_m > 0$ and every $\mathbf{W}_k$ generated by the update rule,
$$\widetilde{\mathcal{L}}_m(\mathbf{W}_k)-\widetilde{\mathcal{L}}_m^\star \le \frac{1}{2\mu_m}\left\|\nabla_{\mathbf{W}}\widetilde{\mathcal{L}}_m(\mathbf{W}_k)\right\|_2^2. \tag{58}$$

Assumption B.3 (Surrogate-gradient bias and variance)**.** Let $\mathcal{F}_k$ denote the natural filtration up to iteration k. For every iterate $\mathbf{W}_k$, the stochastic surrogate gradient satisfies

$$\mathbb{E}\left[\widehat{\mathbf{g}}_k^m \mid \mathcal{F}_k\right] = \nabla_{\mathbf{W}}\widetilde{\mathcal{L}}_m(\mathbf{W}_k)+\mathbf{b}_k^m, \tag{59}$$

with

$$\|\mathbf{b}_k^m\|_2^2 \le B_m^2, \tag{60}$$

and

$$\mathbb{E}\left[\|\widehat{\mathbf{g}}_k^m-\mathbb{E}[\widehat{\mathbf{g}}_k^m \mid \mathcal{F}_k]\|_2^2 \,\middle|\, \mathcal{F}_k\right] \le \sigma_m^2. \tag{61}$$

For conciseness, we omit the mode superscript m throughout this proof and write

$$\widetilde{\mathcal{L}} := \widetilde{\mathcal{L}}_m, \qquad M := M_m, \qquad \mu := \mu_m, \qquad B := B_m, \qquad \sigma := \sigma_m. \tag{62}$$

The update is

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \eta\widehat{\mathbf{g}}_k. \tag{63}$$

Define the exact gradient of the proxy objective at iteration k by

$$\mathbf{g}_k := \nabla_{\mathbf{W}}\widetilde{\mathcal{L}}(\mathbf{W}_k), \tag{64}$$

and define the conditional mean of the stochastic surrogate gradient as

$$\overline{\mathbf{g}}_k := \mathbb{E}\left[\widehat{\mathbf{g}}_k \mid \mathcal{F}_k\right]. \tag{65}$$

By Assumption B.3,

$$\overline{\mathbf{g}}_k = \mathbf{g}_k + \mathbf{b}_k, \tag{66}$$

where $\mathbf{b}_k$ is the conditional surrogate-gradient bias and

$$\|\mathbf{b}_k\|_2^2 \le B^2. \tag{67}$$

We further decompose the stochastic surrogate gradient as

$$\widehat{\mathbf{g}}_k = \overline{\mathbf{g}}_k + \boldsymbol{\zeta}_k, \tag{68}$$

where

$$\boldsymbol{\zeta}_k := \widehat{\mathbf{g}}_k - \overline{\mathbf{g}}_k. \tag{69}$$

Conditioned on $\mathcal{F}_k$, the noise has zero mean:

$$\mathbb{E}\left[\boldsymbol{\zeta}_k \mid \mathcal{F}_k\right] = \mathbf{0}, \tag{70}$$

and its conditional second moment is bounded by

$$\mathbb{E}\left[\|\boldsymbol{\zeta}_k\|_2^2 \mid \mathcal{F}_k\right] \le \sigma^2. \tag{71}$$

Step 1: One-step descent from smoothness. By Assumption B.2, $\widetilde{\mathcal{L}}$ is M-smooth along the line segment joining $\mathbf{W}_k$ and $\mathbf{W}_{k+1}$, and therefore

$$\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \le \widetilde{\mathcal{L}}(\mathbf{W}_k) + \left\langle \nabla_{\mathbf{W}} \widetilde{\mathcal{L}}(\mathbf{W}_k), \mathbf{W}_{k+1} - \mathbf{W}_k \right\rangle + \frac{M}{2}\|\mathbf{W}_{k+1} - \mathbf{W}_k\|_2^2. \tag{72}$$

Equivalently,

$$\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \le \widetilde{\mathcal{L}}(\mathbf{W}_k) + \langle \mathbf{g}_k, \mathbf{W}_{k+1} - \mathbf{W}_k \rangle + \frac{M}{2}\|\mathbf{W}_{k+1} - \mathbf{W}_k\|_2^2. \tag{73}$$

Using the update in Eq. 63,

$$\mathbf{W}_{k+1} - \mathbf{W}_k = -\eta \widehat{\mathbf{g}}_k, \tag{74}$$

we obtain

$$\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \le \widetilde{\mathcal{L}}(\mathbf{W}_k) - \eta \langle \mathbf{g}_k, \widehat{\mathbf{g}}_k \rangle + \frac{M\eta^2}{2}\|\widehat{\mathbf{g}}_k\|_2^2. \tag{75}$$

Step 2: Conditional expectation and stochastic-gradient variance. Conditioned on $\mathcal{F}_k$, both $\mathbf{W}_k$ and $\mathbf{g}_k$ are fixed. Taking conditional expectation of Eq. 75 therefore gives

$$\begin{aligned}
&\mathbb{E}\left[\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \mid \mathcal{F}_k\right] \\
&\le \widetilde{\mathcal{L}}(\mathbf{W}_k) - \eta \langle \mathbf{g}_k, \mathbb{E}[\widehat{\mathbf{g}}_k \mid \mathcal{F}_k] \rangle + \frac{M\eta^2}{2}\mathbb{E}\left[\|\widehat{\mathbf{g}}_k\|_2^2 \mid \mathcal{F}_k\right] \\
&= \widetilde{\mathcal{L}}(\mathbf{W}_k) - \eta \langle \mathbf{g}_k, \overline{\mathbf{g}}_k \rangle + \frac{M\eta^2}{2}\mathbb{E}\left[\|\widehat{\mathbf{g}}_k\|_2^2 \mid \mathcal{F}_k\right].
\end{aligned} \tag{76}$$

We next expand the conditional second moment. From Eq. 68,

$$\begin{aligned}
\|\widehat{\mathbf{g}}_k\|_2^2 &= \|\overline{\mathbf{g}}_k + \boldsymbol{\zeta}_k\|_2^2 \\
&= \|\overline{\mathbf{g}}_k\|_2^2 + 2\langle \overline{\mathbf{g}}_k, \boldsymbol{\zeta}_k \rangle + \|\boldsymbol{\zeta}_k\|_2^2.
\end{aligned} \tag{77}$$

Taking conditional expectation and using Eq. 70,

$$\begin{aligned}
\mathbb{E}\left[\|\widehat{\mathbf{g}}_k\|_2^2 \mid \mathcal{F}_k\right] &= \|\overline{\mathbf{g}}_k\|_2^2 + 2\langle \overline{\mathbf{g}}_k, \mathbb{E}[\boldsymbol{\zeta}_k \mid \mathcal{F}_k] \rangle + \mathbb{E}\left[\|\boldsymbol{\zeta}_k\|_2^2 \mid \mathcal{F}_k\right] \\
&= \|\overline{\mathbf{g}}_k\|_2^2 + \mathbb{E}\left[\|\boldsymbol{\zeta}_k\|_2^2 \mid \mathcal{F}_k\right] \\
&\le \|\overline{\mathbf{g}}_k\|_2^2 + \sigma^2.
\end{aligned} \tag{78}$$

Step 3: Separating the exact-gradient and bias terms. Since $\overline{\mathbf{g}}_k = \mathbf{g}_k + \mathbf{b}_k$, we have

$$\mathbf{b}_k = \overline{\mathbf{g}}_k - \mathbf{g}_k. \tag{79}$$

Therefore,

$$\begin{aligned}\|\mathbf{b}_k\|_2^2 &= \|\overline{\mathbf{g}}_k - \mathbf{g}_k\|_2^2 \\ &= \|\overline{\mathbf{g}}_k\|_2^2 + \|\mathbf{g}_k\|_2^2 - 2\langle \mathbf{g}_k, \overline{\mathbf{g}}_k\rangle .\end{aligned} \tag{80}$$

Rearranging yields the identity

$$2\langle \mathbf{g}_k, \overline{\mathbf{g}}_k\rangle = \|\mathbf{g}_k\|_2^2 + \|\overline{\mathbf{g}}_k\|_2^2 - \|\mathbf{b}_k\|_2^2. \tag{81}$$

Hence

$$-\eta\langle \mathbf{g}_k, \overline{\mathbf{g}}_k\rangle = -\frac{\eta}{2}\|\mathbf{g}_k\|_2^2 - \frac{\eta}{2}\|\overline{\mathbf{g}}_k\|_2^2 + \frac{\eta}{2}\|\mathbf{b}_k\|_2^2. \tag{82}$$

Substituting Eqs. 78 and 82 into Eq. 76 gives

$$\begin{aligned}&\mathbb{E}\left[\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \mid \mathcal{F}_k\right] \\ &\le \widetilde{\mathcal{L}}(\mathbf{W}_k) - \frac{\eta}{2}\|\mathbf{g}_k\|_2^2 - \frac{\eta}{2}\|\overline{\mathbf{g}}_k\|_2^2 + \frac{\eta}{2}\|\mathbf{b}_k\|_2^2 + \frac{M\eta^2}{2}\left(\|\overline{\mathbf{g}}_k\|_2^2 + \sigma^2\right) \\ &= \widetilde{\mathcal{L}}(\mathbf{W}_k) - \frac{\eta}{2}\|\mathbf{g}_k\|_2^2 - \frac{\eta}{2}(1 - M\eta)\|\overline{\mathbf{g}}_k\|_2^2 + \frac{\eta}{2}\|\mathbf{b}_k\|_2^2 + \frac{M\eta^2}{2}\sigma^2.\end{aligned} \tag{83}$$

By the uniform bias bound $\|\mathbf{b}_k\|_2^2 \le B^2$,

$$\begin{aligned}&\mathbb{E}\left[\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \mid \mathcal{F}_k\right] \\ &\le \widetilde{\mathcal{L}}(\mathbf{W}_k) - \frac{\eta}{2}\|\mathbf{g}_k\|_2^2 - \frac{\eta}{2}(1 - M\eta)\|\overline{\mathbf{g}}_k\|_2^2 + \frac{\eta}{2}B^2 + \frac{M\eta^2}{2}\sigma^2.\end{aligned} \tag{84}$$

Because the stepsize satisfies

$$0 < \eta \le \frac{1}{M}, \tag{85}$$

we have

$$1 - M\eta \ge 0. \tag{86}$$

Consequently,

$$-\frac{\eta}{2}(1 - M\eta)\|\overline{\mathbf{g}}_k\|_2^2 \le 0. \tag{87}$$

Dropping this non-positive term gives the weaker but simpler upper bound

$$\mathbb{E}\left[\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) \mid \mathcal{F}_k\right] \le \widetilde{\mathcal{L}}(\mathbf{W}_k) - \frac{\eta}{2}\|\mathbf{g}_k\|_2^2 + \frac{\eta}{2}B^2 + \frac{M\eta^2}{2}\sigma^2. \tag{88}$$

Step 4: Using the local PL condition. The local Polyak–Łojasiewicz condition states that

$$\widetilde{\mathcal{L}}(\mathbf{W}_k) - \widetilde{\mathcal{L}}^\star \le \frac{1}{2\mu}\|\mathbf{g}_k\|_2^2. \tag{89}$$

Equivalently,

$$\|\mathbf{g}_k\|_2^2 \ge 2\mu\left(\widetilde{\mathcal{L}}(\mathbf{W}_k) - \widetilde{\mathcal{L}}^\star\right). \tag{90}$$

Substituting Eq. 90 into Eq. 88 gives

$$\begin{aligned}&\mathbb{E}\left[\widetilde{\mathcal{L}}(\mathbf{W}_{k+1}) - \widetilde{\mathcal{L}}^\star \mid \mathcal{F}_k\right] \\ &\le (1 - \eta\mu)\left(\widetilde{\mathcal{L}}(\mathbf{W}_k) - \widetilde{\mathcal{L}}^\star\right) + \frac{\eta}{2}B^2 + \frac{M\eta^2}{2}\sigma^2.\end{aligned} \tag{91}$$

Now define the scalar expected proxy-objective suboptimality

$$\Gamma_k := \mathbb{E}\left[\widetilde{\mathcal{L}}(\mathbf{W}_k) - \widetilde{\mathcal{L}}^\star\right]. \tag{92}$$

Taking full expectation on both sides of Eq. 91 and applying the tower property,

$$\mathbb{E}\left[\mathbb{E}[X \mid \mathcal{F}_k]\right] = \mathbb{E}[X], \tag{93}$$

we obtain the deterministic scalar recurrence

$$\Gamma_{k+1} \le (1 - \eta\mu)\Gamma_k + \frac{\eta}{2}B^2 + \frac{M\eta^2}{2}\sigma^2. \tag{94}$$

Step 5: Unrolling the contraction recursion. Let

$$\rho := 1 - \eta\mu, \qquad c := \frac{\eta}{2}B^2 + \frac{M\eta^2}{2}\sigma^2. \tag{95}$$

Because $\eta > 0$, $\mu > 0$, $\eta \le 1/M$, and $\mu \le M$, we have

$$0 \le \rho < 1. \tag{96}$$

Equation 94 becomes

$$\Gamma_{k+1} \le \rho\Gamma_k + c. \tag{97}$$

Applying this relation repeatedly,

$$\Gamma_1 \le \rho\Gamma_0 + c, \tag{98}$$

$$\Gamma_2 \le \rho^2\Gamma_0 + c(1+\rho), \tag{99}$$

$$\Gamma_3 \le \rho^3\Gamma_0 + c(1+\rho+\rho^2), \tag{100}$$

and, by induction,

$$\Gamma_K \le \rho^K\Gamma_0 + c\sum_{j=0}^{K-1}\rho^j. \tag{101}$$

Since

$$\sum_{j=0}^{K-1}\rho^j = \frac{1-\rho^K}{1-\rho} = \frac{1-\rho^K}{\eta\mu} \le \frac{1}{\eta\mu}, \tag{102}$$

Eq. 101 implies

$$\begin{aligned}\Gamma_K &\le (1-\eta\mu)^K\Gamma_0 + \left(\frac{\eta}{2}B^2 + \frac{M\eta^2}{2}\sigma^2\right)\frac{1}{\eta\mu} \\ &= (1-\eta\mu)^K\left(\widetilde{\mathcal{L}}(\mathbf{W}_0) - \widetilde{\mathcal{L}}^\star\right) + \frac{B^2}{2\mu} + \frac{M\eta\sigma^2}{2\mu}.\end{aligned} \tag{103}$$

Thus, the proxy objective converges geometrically up to a neighborhood determined by the surrogate-gradient bias and stochastic-gradient variance.

Step 6: Transferring the proxy-objective bound to the hard objective. It remains to bound the suboptimality of the actual hard ANN objective $\mathcal{L}$. Assume that the hard objective admits a minimizer. Restoring the mode index for this definition, let

$$\mathbf{W}_m^\star \in \arg\min_{\mathbf{W}} \mathcal{L}_m(\mathbf{W}), \qquad \mathcal{L}_m^\star := \mathcal{L}_m(\mathbf{W}_m^\star). \tag{104}$$

Within the mode-suppressed proof notation, we write these quantities as $\mathbf{W}^\star$ and $\mathcal{L}^\star$, respectively. For any iterate $\mathbf{W}_K$, add and subtract $\widetilde{\mathcal{L}}(\mathbf{W}_K)$ and $\widetilde{\mathcal{L}}^\star$:

$$\mathcal{L}(\mathbf{W}_K) - \mathcal{L}^\star = \underbrace{\widetilde{\mathcal{L}}(\mathbf{W}_K) - \widetilde{\mathcal{L}}^\star}_{T_1} + \underbrace{\mathcal{L}(\mathbf{W}_K) - \widetilde{\mathcal{L}}(\mathbf{W}_K)}_{T_2} + \underbrace{\widetilde{\mathcal{L}}^\star - \mathcal{L}^\star}_{T_3}. \tag{105}$$

For T_1, Eq. 103 gives

$$\mathbb{E}[T_1] \le (1-\eta\mu)^K\left(\widetilde{\mathcal{L}}(\mathbf{W}_0) - \widetilde{\mathcal{L}}^\star\right) + \frac{B^2}{2\mu} + \frac{M\eta\sigma^2}{2\mu}. \tag{106}$$

For T_2, Theorem 5.3 gives

$$\left|\mathcal{L}(\mathbf{W}_K) - \widetilde{\mathcal{L}}(\mathbf{W}_K)\right| \le \Delta(\mathbf{W}_K). \tag{107}$$

Hence

$$T_2 \le \Delta(\mathbf{W}_K), \tag{108}$$

and therefore

$$\mathbb{E}[T_2] \le \mathbb{E}[\Delta(\mathbf{W}_K)]. \tag{109}$$

For T_3, since $\widetilde{\mathcal{L}}^\star$ is the infimum of the proxy objective,

$$\widetilde{\mathcal{L}}^\star \le \widetilde{\mathcal{L}}(\mathbf{W}^\star). \tag{110}$$

Therefore,

$$\begin{aligned} T_3 &= \widetilde{\mathcal{L}}^\star - \mathcal{L}(\mathbf{W}^\star) \\ &\le \widetilde{\mathcal{L}}(\mathbf{W}^\star) - \mathcal{L}(\mathbf{W}^\star) \\ &\le \left|\widetilde{\mathcal{L}}(\mathbf{W}^\star) - \mathcal{L}(\mathbf{W}^\star)\right| \\ &\le \Delta(\mathbf{W}^\star), \end{aligned} \tag{111}$$

where the final inequality again follows from Theorem 5.3.

Taking expectation in Eq. 105 and combining Eqs. 106, 109, and 111 yields

$$\begin{aligned} \mathbb{E}\left[\mathcal{L}(\mathbf{W}_K) - \mathcal{L}^\star\right] \le (1-\eta\mu)^K \left(\widetilde{\mathcal{L}}(\mathbf{W}_0) - \widetilde{\mathcal{L}}^\star\right) \\ + \frac{B^2}{2\mu} + \frac{M\eta\sigma^2}{2\mu} + \mathbb{E}\left[\Delta(\mathbf{W}_K)\right] + \Delta(\mathbf{W}^\star). \end{aligned} \tag{112}$$

Restoring the mode index m gives

$$\begin{aligned} \mathbb{E}\left[\mathcal{L}_m(\mathbf{W}_K) - \mathcal{L}_m^\star\right] \le (1-\eta\mu_m)^K \left(\widetilde{\mathcal{L}}_m(\mathbf{W}_0) - \widetilde{\mathcal{L}}_m^\star\right) \\ + \frac{B_m^2}{2\mu_m} + \frac{M_m\eta\sigma_m^2}{2\mu_m} + \mathbb{E}\left[\Delta_m(\mathbf{W}_K)\right] + \Delta_m(\mathbf{W}_m^\star), \end{aligned} \tag{113}$$

which is exactly Eq. 19. □

Interpretation of the convergence bound. The first term in Eq. 19 contracts geometrically. The next two terms quantify surrogate-gradient bias and stochastic mini-batch noise, while the final two terms transfer the proxy guarantee to the actual hard ANN objective through the Phase/GIF approximation error. Thus, the bound separates the optimization error of the proxy objective from the error introduced by replacing hard neuronal operations with their analytical proxies.

C Full-Temporal Conversion of LLM Operations

After conversion-aware ANN fine-tuning, fully temporal SNN deployment requires the operators along the LLM forward path to process temporally propagated signals. Following the temporal conversion scheme in You et al. (2024), we distinguish linear operations, whose temporal computation can be obtained directly from linearity, from activation–activation multiplication and nonlinear operations, which require dedicated temporal computation rules.

Linear operation. Let $\{\mathbf{X}_t\}_{t=1}^T$ denote the temporal representation of an activation matrix $\mathbf{X}$. For an activation–weight multiplication, the linearity of the projection gives

$$\mathbf{X} = \sum_{t=1}^{T} \mathbf{X}_t, \qquad \mathbf{O} = \mathbf{X}\mathbf{W} = \sum_{t=1}^{T} \mathbf{X}_t\mathbf{W}. \tag{114}$$

Therefore, the linear transformation can be evaluated independently at each timestep, while summing the resulting temporal outputs recovers the corresponding accumulated ANN output.

Spiking attention. For self-attention, the query, key, and value matrices are represented by the sum of their temporal components:

$$\mathbf{Q} = \sum_{t=1}^{T} \mathbf{Q}_t, \qquad \mathbf{K} = \sum_{t=1}^{T} \mathbf{K}_t, \qquad \mathbf{V} = \sum_{t=1}^{T} \mathbf{V}_t,$$

where $\mathbf{Q}_t$, $\mathbf{K}_t$, and $\mathbf{V}_t$ denote the query, key, and value at timestep t, respectively.

Unlike activation–weight multiplication, self-attention additionally contains activation–activation (AA) multiplication. Define

$$\mathbf{S}_{Q,t} = \sum_{\tau=1}^{t} \mathbf{Q}_\tau, \qquad \mathbf{S}_{K,t} = \sum_{\tau=1}^{t} \mathbf{K}_\tau,$$

as the accumulated query and key outputs up to timestep t. Taking the multiplication between query and key as an example, the temporal computation can be written as

$$\begin{aligned} \mathbf{A} = \mathbf{Q} \cdot \mathbf{K} &= \left(\sum_{t=1}^{T} \mathbf{Q}_t \right) \cdot \left(\sum_{t=1}^{T} \mathbf{K}_t \right) \\ &= \sum_{t=1}^{T} \left(\mathbf{S}_{Q,t} \cdot \mathbf{K}_t + \mathbf{Q}_t \cdot \mathbf{S}_{K,t} - \mathbf{Q}_t \cdot \mathbf{K}_t \right). \end{aligned} \tag{115}$$

Hence, the AA contribution generated at timestep t is $\mathbf{S}_{Q,t} \cdot \mathbf{K}_t + \mathbf{Q}_t \cdot \mathbf{S}_{K,t} - \mathbf{Q}_t \cdot \mathbf{K}_t$. The same temporal AA construction is applied to the subsequent multiplication between the attention matrix $\mathbf{A}$ and the value matrix $\mathbf{V}$.

Spiking Softmax, RMSNorm, and SiLU. Softmax, RMSNorm, and SiLU cannot in general be applied independently to each temporal input while preserving the corresponding ANN computation. We therefore first accumulate the temporally propagated inputs, evaluate the nonlinear operator on the accumulated value, and then recover the output associated with the current timestep through a difference operation.

Specifically, let $\mathbf{X}_t$ denote the input arriving at timestep t and define its accumulated value as

$$\begin{aligned} \mathbf{S}_{X,t} &= \mathbf{S}_{X,t-1} + \mathbf{X}_t, \\ \mathbf{Y}_t &= \phi\left(\mathbf{S}_{X,t}\right), \\ \mathbf{O}(t) &= \mathbf{Y}_t - \mathbf{Y}_{t-1}, \end{aligned} \tag{116}$$

where $\phi \in \{\mathrm{Softmax}, \mathrm{RMSNorm}, \mathrm{SiLU}\}$. Here, $\mathbf{S}_{X,t}$ represents the accumulated input up to timestep t, while $\mathbf{Y}_t$ denotes the output of the corresponding nonlinear operator evaluated on this accumulated input. The temporally propagated output at timestep t is then given by $\mathbf{O}(t)$.

Spiking MLP. Besides its linear projections and SiLU activation, the gated MLP contains an activation–activation Hadamard product. Let $\mathbf{A}_t$ denote the temporal output of the up-projection branch and $\mathbf{B}_t$ the temporal output of the Spiking SiLU branch at timestep t. Their accumulated activations are

$$\mathbf{A} = \sum_{t=1}^{T} \mathbf{A}_t, \qquad \mathbf{B} = \sum_{t=1}^{T} \mathbf{B}_t.$$

The Hadamard product can then be expressed as

$$\begin{aligned} \mathbf{A} \odot \mathbf{B} &= \left(\sum_{t=1}^{T} \mathbf{A}_t \right) \odot \left(\sum_{t=1}^{T} \mathbf{B}_t \right) \\ &= \sum_{t=1}^{T} \left(\mathbf{A}_t \odot \mathbf{B}_t + \sum_{\substack{i=1 \\ i \neq t}}^{T} \frac{\mathbf{A}_t \odot \mathbf{B}_i + \mathbf{A}_i \odot \mathbf{B}_t}{2} \right). \end{aligned}$$

Accordingly, the Hadamard-product contribution at timestep t is

$$\mathbf{A}_t \odot \mathbf{B}_t + \sum_{i=1, i \neq t}^{T} \frac{\mathbf{A}_t \odot \mathbf{B}_i + \mathbf{A}_i \odot \mathbf{B}_t}{2}.$$

These temporal operator realizations are introduced only after ANN fine-tuning, when the model is converted into a fully temporal SNN. During conversion-aware ANN fine-tuning, temporal dynamics are confined to individual spiking-aware replacement sites, and temporal signals are not propagated between ANN layers.

D More Experimental Results

D.1 Experimental Configurations

For reproducibility, we summarize the hyperparameter configurations used in our experiments. Unless otherwise specified, the common configurations in Table 5 are shared across all model-task pairs and ANN fine-tuning modes. The mode- and model-specific learning rates are reported separately in Table 6.

Table 5: Common hyperparameter configurations used in all experiments.

Configuration	**Value**
Epochs	1
Batch size per GPU	1
Gradient accumulation steps	16
Optimizer	AdamW
Learning-rate schedule	Cosine
Warmup ratio	0
Maximum sequence length	2048
Training samples	10,000
Random seed	42
Phase timesteps T	4
MTN timesteps T	4
MTN threshold number K	6
Calibration samples	128
Calibration group size	128

Table 6: Learning rates used for different ANN fine-tuning modes and models.

Mode	**Qwen3-1.7B**	**Qwen3-8B**	**Llama3-8B**
Vanilla	1×10^{-6}	1×10^{-6}	1×10^{-6}
Unaware	1×10^{-6}	1×10^{-6}	1×10^{-6}
Phase-aware	2×10^{-5}	2×10^{-5}	5×10^{-6}
GIF-aware	5×10^{-5}	2×10^{-6}	5×10^{-6}

For the sigmoid surrogate used for the Phase firing function, the slope parameter β is set to 1, 2.5, and 12 for Qwen3-1.7B, Qwen3-8B, and Llama3-8B, respectively.