GRADIENT-GUIDED FINITE-PRECISION TEMPORAL CODING FOR EFFICIENT SPIKING MULTIMODAL LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) have achieved remarkable progress, but their inference incur huge energy consumption. ANN-to-SNN conversion offers a promising pathway for the energy-efficient deployment of MLLMs by replacing complex floating-point arithmetic operations with sparse spike operations. However, dominant rate coding methods incur substantial memory access, data movement, and state updates due to frequent multi-spike propagation, preventing SNNs from fully realizing their potential for energy efficiency. In contrast, Time-to-First-Spike (TTFS) coding permits each neuron to fire at most a single spike, thereby even more significantly reducing energy consumption. Nevertheless, this temporal coding schemes rely on the ideal assumption of infinite time precision, suffering performance degradation when deployed on real hardware with finite time precision. To address this dilemma, we propose GFTC, a Gradient-guided Finite-precision Temporal Coding framework. Our key insight is that representation errors across different activation regions exert markedly different impacts on the overall conversion loss. With minimizing conversion error as the optimization objective, GFTC dynamically optimizes the temporal codebook under a fixed time budget by prioritizing more time steps for gradient-sensitive activation regions. Furthermore, to mitigate deep-layer error accumulation, we introduce an iterative calibration mechanism combining SNN execution trajectories with ANN teacher targets, accompanied by theoretical convergence guarantees. Evaluated across representative MLLM architectures and benchmarks, GFTC achieves nearly lossless conversion under extremely low time budgets, matching full-precision ANN accuracy while reducing inference energy consumption by up to 64.68%.

1 INTRODUCTION

Multimodal Large Language Models (MLLMs) have demonstrated exceptional capabilities across diverse vision language reasoning tasks (Liu et al., 2023b; Bai et al., 2025). However, their continuous inference over long multimodal sequences relies on dense floating point operations within multilayer Transformers (Vaswani et al., 2017), leading to severe latency and energy bottlenecks during deployment. Overcoming this efficiency barrier urgently requires exploring low power computing paradigms to sustain the ever expanding scale of LLMs. Inspired by low-power biological neural systems, Spiking Neural Networks (SNNs) offer a promising pathway for the energy-efficient deployment of models (Maass, 1997; Roy et al., 2019). They process information through sparse spike operations rather than complex floating-point arithmetic operations.

As an efficient paradigm that circumvents training SNNs from scratch, ANN-to-SNN conversion constructs SNNs by reusing pre-trained ANN weights followed by conversion and calibration (Rueckauer et al., 2017). Existing ANN-to-SNN conversion approaches predominantly rely on rate coding, which requires neurons to fire multiple spikes across a time window and approximates continuous ANN activations through firing frequencies (Diehl et al., 2015). However, because each spike triggers an independent forward execution, the repetitive spike propagation induced by rate coding incurs substantial memory access, data movement, and state updates (Dampfhoffer et al., 2023). In practice, these non-computational system-level overheads dominate the overall execution

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.

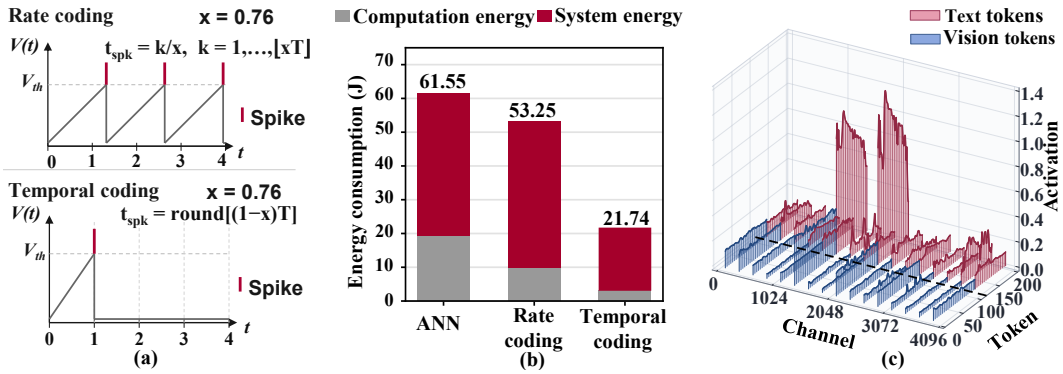


Figure 1: (a) An example of encoding a continuous activation x using rate coding and temporal coding. $V(t)$ denotes membrane potential, t_{spk} denotes spike time, and T denotes the coding window length. (b) Energy estimates for an ANN and SNNs using rate and temporal coding on LLaVA. Computation energy covers multiply–accumulate and accumulate operations, system energy covers memory access, data movement, and state updates. (c) MLLM activations across channels and tokens, highlighting channel outliers and distribution differences between text and vision modalities.

cost, particularly in large-scale architectures like MLLMs, preventing SNNs from fully realizing their theoretical energy-efficiency potential at the hardware level.

In contrast, temporal coding used in ANN-to-SNN conversion, such as Time-to-First-Spike (TTFS) coding, permits each neuron to fire at most a single spike (Kheradpisheh & Masquelier, 2020), thereby more significantly reducing energy consumption, as shown in Figure 1(a) and Figure 1(b). Nevertheless, this temporal coding schemes rely on the ideal assumption of infinite time precision, often suffering from performance degradation under finite precision. This poses severe challenges for their deployment on real hardware, which is inherently constrained by limited clock resolution. Increasing the number of time steps introduces significant inference latency, whereas reducing time steps degrades model accuracy due to coarse-grained representations.

Furthermore, MLLMs display distinct activation characteristics, certain channels exhibit pronounced activation outliers, and activation distributions differ vastly across tokens from different modalities, as illustrated in Figure 1(c). This highly non-uniform activation distribution causes traditional discrete temporal coding to suffer from massive conversion errors, severely hindering the application of TTFS to large models.

To address this issue, we propose GFTC, a Gradient-guided Finite-precision Temporal Coding framework. Our key insight is that representation errors across different activation regions exert markedly different impacts on the overall conversion loss. With minimizing conversion error as the optimization objective, GFTC dynamically optimizes the temporal coding codebook under a fixed time budget by prioritizing more time steps for gradient-sensitive activation regions. Furthermore, to alleviate error accumulation in deep networks, we introduce an iterative calibration mechanism that combines SNN execution trajectories with validation against ANN teacher targets. Theoretically, we provide convergence guarantees for this iterative calibration mechanism to ensure stable and effective optimization. Applying this hardware-friendly temporal coding scheme to multimodal large language models, we evaluate GFTC across multiple benchmarks and architectures. Extensive experiments demonstrate that GFTC achieves nearly lossless conversion under low time precision, matching ANN accuracy while reducing inference energy consumption by up to 64.68%.

Our main contributions are summarized as follows:

- **Gradient-Guided Temporal Coding.** We propose a finite-precision temporal coding scheme that optimizes the decoding codebook based on gradient sensitivity, drastically reducing conversion errors under strict time budgets.
- **Iterative Calibration in Deep Networks.** To address multi-layer error accumulation in large models, we align SNN trajectories with frozen ANN targets via an iterative calibration framework, backed by a theoretical convergence guarantee.

- **Efficient Spiking MLLM Conversion.** We present an efficient spiking conversion scheme for MLLMs that addresses multimodal activation distributions and outliers, achieving accuracy comparable to the original ANN with up to 64.68% energy reduction.

2 Related Works

2.1 ANN-to-SNN Conversion

Inspired by low power biological neural systems, Spiking Neural Networks achieve energy efficient inference via sparse spike operations by reusing pretrained ANN weights. Early rate coding methods, including RMP-SNN (Han et al., 2020), QCFS (Bu et al., 2022), SlipReLU (Jiang et al., 2023), SNNC (Li et al., 2021), and FTBC (Wu et al., 2024), established ReLU to firing rate mappings using threshold balancing, activation redesign, and calibration. However, because approximating continuous activations requires repeated forward propagation, rate coding incurs substantial memory access, data movement, and state updates. To address this, Time-to-First-Spike (TTFS) coding permits each neuron to fire at most a single spike per window (Kheradpisheh & Masquelier, 2020), substantially reducing spike transmissions. Recent Transformer extensions like LC-TTFS (Yang et al., 2024), SpikedAttention (Hwang et al., 2024), T-SpikeFormer (Sun et al., 2025), and TTFSFormer (Zhao et al., 2025) established continuous time TTFS mappings for attention mechanisms and nonlinear operators. Nevertheless, these approaches rely on the ideal assumption of infinite temporal precision. Under the fixed clock resolution of real hardware, increasing time steps introduces additional inference latency, whereas reducing time steps severely degrades model accuracy.

2.2 Spiking Large Language Models

Transformer based Large Language Models (LLMs) achieve outstanding performance, yet their inference faces massive energy bottlenecks. Converting LLMs into SNNs provides a promising avenue for low power edge deployment. Early methods like STA (Jiang et al., 2024) and SpikeZIP-TF (You et al., 2024) addressed Transformer attention and nonlinearities, while SpikeLLM (Xing et al., 2025), FAS (Chen et al., 2025), and LAS (Chen et al., 2026) extended conversion to pretrained LLMs via salience aware coding, calibration, and dedicated neurons. Subsequent works reduced spike redundancy via SDLLM (Xu et al., 2026b) and improved activation approximation using distribution aware phase coding (Zheng et al., 2026). For multimodal scenarios, FAS and LAS evaluated vision language models, and SpikeMLLM (Xu et al., 2026a) introduced modality specific time scales and spike compression. These methods predominantly rely on rate coding and multispike schemes. SpikeQuant (Wang et al., 2026) introduced mixed-precision TTFS coding for activation outliers, its uniform quantization causes sub-optimal errors under non-uniform distributions. Moreover, it retains floating-point operations for attention matrix multiplication, Softmax, and normalization, lacking complete operator-level conversion. While TTFSFormer (Zhao et al., 2025) achieves full operator-level conversion, its performance degrades severely when applied to MLLMs under finite temporal precision. Consequently, cross-modal discrepancies, activation outliers, and deep-layer error accumulation remain key challenges for finite-precision temporal coding in MLLMs.

3 Preliminaries

Rate coding represents activation magnitude via the spike count in a window, whereas time-to-first-spike (TTFS) coding utilizes the timing of at most one spike per neuron. Larger activations trigger earlier spikes, thereby reducing redundant computation and memory access.

At the scalar level, a monotone TTFS mapping corresponds to a set of decreasing positive thresholds. Given an ANN activation $x \geq 0$, the membrane potential is initialized as $v(1) = x$ without subsequent input. For $t \in \{1, \ldots, T\}$, the neuron dynamics are:

$$s(t) = \Theta(v(t) - \theta(t)), \quad v(t+1) = v(t)\,(1 - s(t)), \tag{1}$$

where $\theta(t) > 0$ is the firing threshold, $s(t) \in \{0, 1\}$ is the binary spike, and $\Theta(z) = \mathbb{I}(z \geq 0)$. The output is $o(t) = d(t)s(t)$ with decoding weight $d(t)$. Resetting $v(t)$ after firing prevents further spikes, enforcing $\sum_{t=1}^{T} s(t) \leq 1$.

Under a finite budget $T \leq T_{\text{budget}}$, let $\mathcal{C} = \{c_1, \ldots, c_T\}$ $(c_1 > \cdots > c_T > 0)$ denote the temporal decoding codebook and $q_0 = +\infty > q_1 > \cdots > q_T > 0$ be the decision boundaries. Setting $\theta(t) = q_t$ and $d(t) = c_t$ yields:

$$s(t) = \mathbb{I}(x \in [q_t, q_{t-1}))\,. \tag{2}$$

The reconstructed activation is:

$$\hat{x} = \sum_{t=1}^{T} o(t) = \sum_{t=1}^{T} c_t\, \mathbb{I}(x \in [q_t, q_{t-1}))\,. \tag{3}$$

Activations $x < q_T$ yield no spikes and reconstruct to zero. Thus, finite-resolution TTFS implements scalar quantization with at most $T + 1$ reconstruction levels (including silence), forming the basis of our framework in Section 4.

4 METHODOLOGY

To reduce conversion errors under finite temporal precision, we present the proposed Gradient-guided Finite-precision Temporal Coding (**GFTC**) framework, which includes gradient-guided temporal codebook optimization and a trajectory-aware iterative calibration mechanism, as illustrated in Figure 2 and Figure 3. We first present the gradient-guided temporal codebook optimization under finite precision (§4.1), which aims to prioritize the allocation of time steps to regions with higher conversion loss gradients, thereby achieving adaptive temporal resolution within a limited time budget. We then describe the trajectory-aware iterative calibration mechanism to suppress multi-layer error accumulation after converting MLLMs to SNNs (§4.2). Finally, we establish the theoretical convergence guarantees for the iterative optimization process (§4.3) and provide the pseudocode for the complete conversion algorithm.

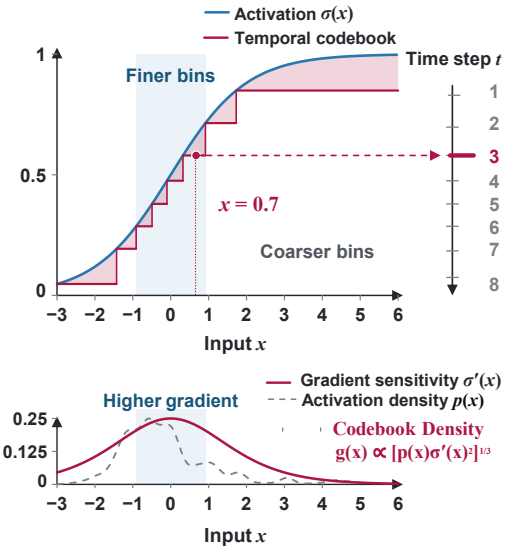


Figure 2: Gradient-guided finite-precision temporal coding illustrated using a sigmoid activation. The temporal codebook assigns finer bins to regions with higher activation sensitivity and coarser bins elsewhere.

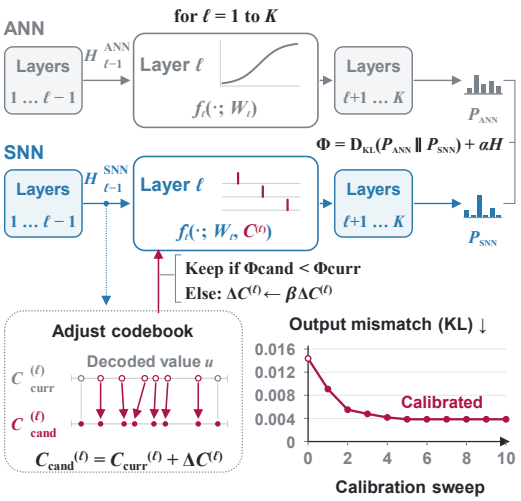


Figure 3: Trajectory-aware iterative calibration. Within each sweep, temporal codebooks are updated layer by layer using current SNN trajectories and validated against ANN teacher outputs, while pretrained weights remain frozen. Repeated sweeps reduce conversion errors accumulated across layers.

4.1 GRADIENT-GUIDED TEMPORAL CODEBOOK OPTIMIZATION

In ANN-to-SNN conversion for Multimodal Large Language Models (MLLMs), significant outliers exist in certain channels, and different modality tokens exhibit distinct distribution patterns. Under a strict latency budget $T \leq T_{\text{budget}}$, traditional uniform temporal coding struggles to handle such

complex activation behaviors, as representation errors across different activation ranges exert vastly unequal impacts on the final conversion loss. To address this issue, we propose a gradient-guided temporal coding scheme. By evaluating the gradient sensitivity of activation values with respect to the conversion loss, our framework dynamically allocates more temporal steps to high-sensitivity activation regions and fewer to low-sensitivity ones, thereby minimizing the conversion error.

Definition 1 (Finite-Precision SNN Conversion Loss)**.** *Let $u_i \in [0, u_{\max}]$ denote the continuous activation at index i, and $Q_{\mathcal{C}}(u_i) = \mathcal{D}(u_i; \mathcal{C})$ be its reconstructed value under temporal codebook $\mathcal{C}$. Given the task loss $\mathcal{L}$, the sensitivity-weighted conversion loss $\mathcal{E}(\mathcal{C})$ is defined as:*

$$\mathcal{E}(\mathcal{C}) = \sum_{i=1}^{N} s_i \left(u_i - Q_{\mathcal{C}}(u_i)\right)^2, \tag{4}$$

where $s_i = \left(\frac{\partial \mathcal{L}}{\partial u_i}\right)^2$ represents the first-order gradient sensitivity weight of activation u_i, and N is the total number of elements in the activation tensor.

Based on Definition 1, we formulate the continuous limit of the optimal codebook density allocation under a finite time budget T.

Theorem 1 (Optimal Codebook Density Distribution)**.** *Let $p(u)$ denote the empirical probability density function of continuous activations, and let $\bar{s}(u) = \mathbb{E}[s_i \mid u_i = u]$ be the conditional expectation of gradient sensitivity given activation magnitude u. Under asymptotic conditions ($T \gg 1$), the continuous codebook density function $g(u)$ that minimizes $\mathcal{E}(\mathcal{C})$ satisfies:*

$$g^*(u) = \frac{\left(p(u)\bar{s}(u)\right)^{1/3}}{\int_0^{u_{\max}} \left(p(x)\bar{s}(x)\right)^{1/3} dx}, \tag{5}$$

such that the minimum expected weighted conversion loss scales as $\mathcal{O}(T^{-2})$. The detailed proof is provided in the Appendix A.2.

Discrete Codebook Optimization. Theorem 1 provides the optimal continuous density function $g^*(u)$. To construct the discrete codebook $\mathcal{C} = \{c_1, c_2, \ldots, c_T\}$ in practice, we compute the inverse cumulative distribution function of $g^*(u)$ to partition the activation space into T intervals with equal integrated mass, setting each codeword c_t to the weighted centroid of its corresponding region. Because different token sequences exhibit dynamic range variations across modalities, we dynamically measure the activation range $[0, u_{\max}]$ per layer and token during calibration. Furthermore, to account for cross-modal distribution discrepancies, image tokens and text tokens maintain modality-specific codebooks, while multi-modal fusion layers utilize a shared codebook structure.

4.2 Trajectory-Aware SNN Iterative Calibration

While local codebook optimization minimizes conversion errors within isolated layers, feedforward propagation through deep MLLM architectures causes these local errors to accumulate sequentially. This progressive drift leads to severe misalignment between SNN spike trajectories and the target ANN activation states across deeper layers. To control this accumulated trajectory deviation, we propose an offline trajectory-aware iterative calibration mechanism. During the offline calibration stage, it updates each layer's codebook using SNN execution trajectories generated on calibration samples rather than static ANN activations.

Definition 2 (SNN Trajectory Drift)**.** *Let $\mathbf{H}_l^{ANN}$ be the target activation tensor of the l-th layer in the pre-trained ANN teacher. Let $\mathbf{H}_l^{SNN}(\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(l)})$ denote the actual output tensor produced by executing the offline SNN up to layer l using the sequence of calibrated temporal codebooks $\{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(l)}\}$. The cumulative trajectory deviation $\mathbf{R}_l$ at layer l is defined as:*

$$\mathbf{R}_l = \mathbf{H}_l^{SNN}(\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(l)}) - \mathbf{H}_l^{ANN}. \tag{6}$$

To control the accumulation of $\mathbf{R}_l$ across deep Transformer layers, we calibrate each layer's codebook $\mathcal{C}^{(l)}$ under empirical SNN trajectories rather than static ANN teacher activations.

Algorithm 1 Gradient-Guided Finite-Precision Temporal Coding (GFTC)

Input: Pre-trained MLLM $\mathcal{M}_{\text{ANN}}$, K layers, calibration set $\mathcal{D}_{\text{calib}}$, budget T, tolerance η, and decay factor $\beta \in (0, 1)$.

Output: Optimized multi-layer codebooks $\mathbf{C}^* = \{\mathcal{C}^{(1)*}, \ldots, \mathcal{C}^{(K)*}\}$.

// Phase 1: Codebook Initialization
Compute layerwise activation sensitivities $s_i^{(l)}$ from gradients on $\mathcal{D}_{\text{calib}}$.
for $l = 1$ to K **do**
    Estimate density $p_l(u)$, sensitivity $\bar{s}_l(u)$, and per-token activation ranges .
    Construct $g_l^*(u)$ and its CDF $G_l^*(u)$.
    Partition the ranges into T intervals via G_l^* and use weighted centroids to initialize $\mathcal{C}_{\text{curr}}^{(l)}$.
end for
// Phase 2: Iterative Calibration
Initialize $\mathbf{C}_{\text{curr}} \leftarrow \{\mathcal{C}_{\text{curr}}^{(1)}, \ldots, \mathcal{C}_{\text{curr}}^{(K)}\}$ and $\Phi_{\text{prev}} \leftarrow \Phi(\mathbf{C}_{\text{curr}})$.
repeat
    for $l = 1$ to K **do**
        Obtain upstream trajectory $\mathbf{H}_{l-1}^{\text{SNN}}$ and generate a sensitivity-driven perturbation $\Delta\mathcal{C}^{(l)}$.
        Form $\mathbf{C}_{\text{cand}}$ by replacing $\mathcal{C}_{\text{curr}}^{(l)}$ with $\mathcal{C}_{\text{curr}}^{(l)} + \Delta\mathcal{C}^{(l)}$ while preserving valid codebook ordering.
        if $\Phi(\mathbf{C}_{\text{cand}}) < \Phi(\mathbf{C}_{\text{curr}})$ **then**
            $\mathbf{C}_{\text{curr}} \leftarrow \mathbf{C}_{\text{cand}}$
        else
            $\Delta\mathcal{C}^{(l)} \leftarrow \beta\Delta\mathcal{C}^{(l)}$; recompute and re-evaluate $\mathbf{C}_{\text{cand}}$.
        end if
    end for
    $\Delta\Phi \leftarrow |\Phi_{\text{prev}} - \Phi(\mathbf{C}_{\text{curr}})|$ and $\Phi_{\text{prev}} \leftarrow \Phi(\mathbf{C}_{\text{curr}})$.
until $\Delta\Phi < \eta$
return $\mathbf{C}^* \leftarrow \mathbf{C}_{\text{curr}}$

Theorem 2 (Multi-Layer Error Boundedness)**.** *Assume that for any layer $l \in \{1, \ldots, K\}$, the feedforward mapping function $f_l(\cdot)$ is L_f-Lipschitz continuous over the bounded activation domain $[0, u_{\max}]$. If $\mathbf{H}_l^{ANN} = f_l(\mathbf{H}_{l-1}^{ANN})$ and $\mathbf{H}_l^{SNN} = f_l(\mathbf{H}_{l-1}^{SNN}) + \boldsymbol{\delta}_l$ with $\|\boldsymbol{\delta}_l\|_2 \leq \epsilon_l$, then the global trajectory deviation at the output layer K satisfies:*

$$\|\mathbf{R}_K\|_2 \leq \sum_{l=1}^{K} L_f^{K-l} \epsilon_l. \tag{7}$$

The bound is conditional on these local perturbation assumptions, a teacher-validation threshold alone does not imply $\epsilon_l \leq \epsilon_{threshold}$. The detailed proof is provided in the Appendix A.3.

Offline Iterative Calibration Protocol. Theorem 2 provides a conditional output-error bound based on the local conversion errors. We execute a layer-wise offline calibration procedure while keeping all pre-trained ANN parameters strictly frozen. At layer l, the current empirical trajectory $\mathbf{H}_{l-1}^{\text{SNN}}$ is fed into the candidate codebook generator to produce a perturbation $\mathcal{C}_{\text{cand}}^{(l)} = \mathcal{C}_{\text{curr}}^{(l)} + \Delta\mathcal{C}^{(l)}$, where $\Delta\mathcal{C}^{(l)}$ represents a step search driven by gradient sensitivity. We evaluate the candidate configuration $\mathbf{C}_{\text{cand}} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}_{\text{cand}}^{(l)}, \ldots, \mathcal{C}^{(K)}\}$ against the ANN teacher logits over calibration tokens using the validation objective Φ. An update step is accepted ($\mathcal{C}_{\text{curr}}^{(l)} \leftarrow \mathcal{C}_{\text{cand}}^{(l)}$) if and only if $\Phi(\mathbf{C}_{\text{cand}}) < \Phi(\mathbf{C}_{\text{curr}})$. Otherwise, the candidate is rejected, and an inner backtracking loop updates $\Delta\mathcal{C}^{(l)} \leftarrow \beta\Delta\mathcal{C}^{(l)}$ and re-evaluates the candidate until acceptance or the prescribed trial limit is reached.

4.3 Objective Monotonicity and Value Convergence

In this section, we analyze the validation objective induced by the accept/reject rule. Specifically, we establish monotonic objective-value reduction and convergence of the objective values under a lower-bound assumption, and subsequently summarize the complete operational procedure in Algorithm 1.

Definition 3 (Validation Objective Function)**.** *For a fixed network topology, the overall validation objective $\Phi(\mathbf{C})$ over all layer codebooks $\mathbf{C} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(K)}\}$ is defined as:*

$$\Phi(\mathbf{C}) = \mathcal{D}_{KL}\left(P_{ANN}(Y \mid X) \,||\, P_{SNN}(Y \mid X; \mathbf{C})\right) + \alpha \mathcal{H}\left(P_{ANN}(Y \mid X), P_{SNN}(Y \mid X; \mathbf{C})\right), \quad (8)$$

where $\mathcal{D}_{KL}$ is the Kullback-Leibler divergence, $\mathcal{H}$ is the cross-entropy loss against teacher tokens, and $\alpha > 0$ is a balancing hyperparameter.

Theorem 3 (Monotonicity and Objective-Value Convergence)**.** *Let $\{\mathbf{C}^{(k)}\}_{k=0}^{\infty}$ denote the sequence of multi-layer temporal codebooks generated by the iterative calibration protocol at iteration step k. Under the accept/reject mechanism, the validation objectives satisfy*

$$\Phi(\mathbf{C}^{(k+1)}) \leq \Phi(\mathbf{C}^{(k)}), \qquad \forall k \geq 0.$$

Since $\Phi(\mathbf{C}) \geq 0$ is bounded from below, the scalar sequence $\{\Phi(\mathbf{C}^{(k)})\}$ converges to a finite limit

$$\Phi_\infty = \lim_{k \to \infty} \Phi(\mathbf{C}^{(k)}).$$

Our verification mechanism guarantees monotonic decrease and convergence of the validation objective.

The complete execution workflow of the proposed Gradient-guided Finite-precision Temporal Coding (GFTC) framework is summarized in Algorithm 1.

5 Experiments

5.1 Experimental Setup

Models and benchmarks. We evaluate GFTC on LLaVA-v1.5-7B (Liu et al., 2023a) and LLaVA-NeXT-v1.6-Vicuna-7B (Liu et al., 2024a) across five multimodal benchmarks: TextVQA (Singh et al., 2019), GQA (Hudson & Manning, 2019), POPE (Li et al., 2023), ScienceQA-IMG (Lu et al., 2022), and MM-Vet (Yu et al., 2024). To validate the generalizability of our method across model architectures and scales, we evaluate the 4B, 8B, and 32B variants of Qwen3-VL (Bai et al., 2025) on MMMU (Yue et al., 2024), MMStar (Chen et al., 2024), RealWorldQA (xAI, 2024) and MM-Bench (Liu et al., 2024b). The original FP32 ANN serves as the accuracy reference.

Baselines. We extend four baseline methods to MLLMs for comparative experiments. **SpikeLLM** (Xing et al., 2025) is a rate-coding-based approach that combines saliency-aware quantization with generalized integrate-and-fire neurons, utilizing additional spiking steps to represent salient activations. **PhaseSNN** (Zheng et al., 2026) employs phase coding, incorporating spike encoding into attention computation while evaluating nonlinear functions on accumulated inputs and propagating their output differences. **TTFSFormer** (Zhao et al., 2025) extends single-spike temporal computation to Transformer operators including attention, softmax, and normalization, providing a foundation for neuromorphic execution. **SpikeQuant** (Wang et al., 2026) also utilizes temporal coding with mixed-precision TTFS encoding for quantized linear projections but retains floating-point computation for attention matrix products, softmax, and normalization.

Precision and implementation. To ensure a fair comparison across different encoding paradigms, namely rate, phase, and temporal coding, in Table 1, we uniformly use the bit depth B corresponding to 2^B distinguishable states (Gray & Neuhoff, 1998) to quantify the encoding capacity. Since the temporal period T relates differently to information capacity across schemes, direct comparison of T is unfair. Therefore, Table 1 aligns methods by fixing the bit depth B and evaluating 6-bit and 8-bit precisions, alongside higher bit depths of 10 and 13 for TTFSFormer to accommodate its severe accuracy drop under limited temporal precision. In contrast, for the evaluation on Qwen3-VL in Table 2, both the baseline TTFSFormer and our method adopt temporal coding where T corresponds to the inference latency, allowing us to report both the bit depth B and the temporal period T.

5.2 Main Results

The main experimental results on LLaVA-1.5-7B are presented in Table 1. While maintaining the high efficiency of single-spike coding, our proposed method achieves complete spike-driven conversion, including spike attention, and exhibits extraordinary robustness under low temporal precision.

Table 1: Results on LLaVA-1.5-7B. "Max Spikes" indicates the maximum number of spikes a neuron can emit per temporal window. "Spike Attention" indicates whether attention is fully spike-driven. "Bits" denotes the bit depth B representing the information capacity encodable within the temporal window. Avg. Acc. denotes the average score across five multimodal benchmarks.

Method	Spike Attention	Max Spikes	Bits	TextVQA	GQA	POPE	SQA-IMG	MM-Vet	Avg. Acc.
ANN	—	—	32	58.18	61.94	85.92	69.46	31.30	61.36
SpikeLLM	✓	256	8	56.92	60.64	84.38	67.76	29.91	59.92
SpikeLLM	✓	64	6	52.18	55.73	79.24	64.06	25.68	55.38
PhaseSNN	✓	8	8	58.08	60.87	85.16	69.54	29.28	60.59
PhaseSNN	✓	6	6	56.71	60.12	84.73	67.82	29.24	59.72
TTFSFormer	✓	1	13	44.73	48.26	76.82	63.12	24.68	51.52
TTFSFormer	✓	1	10	31.64	40.37	73.42	61.49	23.47	46.08
SpikeQuant	×	1	8	57.41	60.38	85.91	67.92	29.04	60.13
SpikeQuant	×	1	6	56.53	59.82	84.57	67.31	28.46	59.34
GFTC (Ours)	✓	**1**	**8**	**58.06**	**61.67**	**85.75**	**69.26**	**31.14**	**61.18**
GFTC (Ours)	✓	**1**	**6**	**56.96**	**61.08**	**85.28**	**68.47**	**30.62**	**60.48**

At 8-bit temporal precision, our method achieves virtually lossless conversion, matching the FP32 ANN baseline. Even under an ultra-low temporal budget of 6-bit, our method maintains strong performance, consistently outperforming multi-spike encoding approaches such as SpikeLLM and PhaseSNN. Although TTFSFormer attempts operator-level temporal conversion for the attention mechanism, its performance heavily relies on near-infinite temporal precision. When evaluated under realistic, finite temporal precision (13-bit and 10-bit), TTFSFormer suffers catastrophic performance degradation. Meanwhile, although SpikeQuant delivers reasonable performance, its attention mechanism still relies on non-spiking floating-point operations, failing to achieve true end-to-end spike-driven execution and thus leaving the energy efficiency potential of SNNs unfulfilled. In contrast, our method achieves superior conversion accuracy under complete temporal spike conversion. As shown in Table 2, we evaluate our method on Qwen3-VL variants of multiple scales, adopting TTFSFormer, which also utilizes fully spike-driven temporal coding, as the baseline. The results demonstrate that our method significantly outperforms the baseline across various bit depth and temporal step configurations. Ablation studies, evaluation on LLaVA-NeXT-7B, and further experimental results are provided in the Appendix.

5.3 Energy Analysis

We estimate the total energy (E_{total}) required to process one image–question pair with LLaVA-1.5-7B. We divide the energy required for one inference into computation energy (E_{comp}) and system energy (E_{sys}). Computation energy consists of multiply–accumulate (MAC) operations and accumulate (AC) operations. System energy accounts for memory accesses, data movement, and neuron-state updates. We follow the hardware-aware energy accounting model proposed in SpikedAttention (Hwang et al., 2024). Under the 45-nm CMOS reference, one MAC costs $4.6\,\text{pJ}$, one AC costs $0.9\,\text{pJ}$, and one 32-bit memory access costs $5\,\text{pJ}$ (Horowitz, 2014). Both rate coding and temporal coding use 32-bit weights and 8-bit-depth activations with $T = 256$. The computation and system energies are calculated as

$$E_{\text{comp}} = e_{\text{MAC}} N_{\text{MAC}} + e_{\text{AC}} N_{\text{AC}}, \tag{9}$$

$$E_{\text{sys}} = e_{32}(R_{32} + W_{32}) + e_1(R_1 + W_1) + e_{\text{cmp}} N_{\text{cmp}}. \tag{10}$$

The total energy is given by $E_{\text{total}} = E_{\text{comp}} + E_{\text{sys}}$. Here, $e_{\text{MAC}} = 4.6\,\text{pJ}$ and $e_{\text{AC}} = 0.9\,\text{pJ}$ denote the energy costs of one MAC and one AC. The parameter $e_{32} = 5\,\text{pJ}$ denotes the energy cost of one 32-bit memory access, $e_1 = \frac{5}{32}\,\text{pJ}$ denotes the energy cost of one one-bit spike transfer,

Table 2: Results on Qwen3-VL of multiple scales across four multimodal benchmarks. "Bits" is the bit depth B representing the information capacity encodable within the temporal window. Under temporal coding, "T" denotes the temporal period, corresponding to 2^B time steps and the resulting inference latency. "Avg. Acc." denotes the average score across four multimodal benchmarks.

Model	Method	Bits	T	MMMU	MMStar	RWQA	MMBench-EN	Avg. Acc.
Qwen3-VL-4B	ANN	—	—	67.4	69.8	70.9	85.1	73.30
	TTFSFormer	13	8192	48.9	55.8	60.7	72.1	59.38
	TTFSFormer	10	1024	47.8	53.1	59.9	70.7	57.88
	GFTC (Ours)	**8**	**256**	**65.4**	**68.5**	**69.5**	**83.1**	**71.63**
	GFTC (Ours)	**6**	**64**	**63.6**	**66.9**	**68.4**	**81.6**	**70.13**
Qwen3-VL-8B	ANN	—	—	69.6	70.9	71.5	85.0	74.25
	TTFSFormer	13	8192	53.1	57.8	62.2	73.6	61.68
	TTFSFormer	10	1024	51.8	56.4	61.0	72.2	60.35
	GFTC (Ours)	**8**	**256**	**68.3**	**69.4**	**70.5**	**83.5**	**72.93**
	GFTC (Ours)	**6**	**64**	**66.4**	**67.9**	**69.2**	**82.0**	**71.38**
Qwen3-VL-32B	ANN	—	—	76.0	77.7	79.0	88.9	80.40
	TTFSFormer	13	8192	57.6	64.3	66.9	77.4	66.55
	TTFSFormer	10	1024	56.3	63.0	65.8	76.0	65.28
	GFTC (Ours)	**8**	**256**	**73.8**	**76.4**	**77.4**	**87.5**	**78.78**
	GFTC (Ours)	**6**	**64**	**70.8**	**73.1**	**75.9**	**85.0**	**76.20**

and $e_{\mathrm{cmp}} = 0.9\,\mathrm{pJ}$ denotes the energy cost of one threshold comparison. N_{MAC} and N_{AC} denote the numbers of MAC and AC operations, respectively, R_{32} and W_{32} denote 32-bit read and write operations for weights, activations, and neuron states, R_1 and W_1 denote one-bit spike transfers, and N_{cmp} denotes the number of threshold comparisons.

We select SpikeLLM as the representative rate-coding baseline and compare it with the full-precision ANN. As shown in Table 3, our method achieves the lowest energy consumption, reducing the total energy by 64.68% compared with the conventional artificial neural network (ANN). This advantage results from replacing energy-intensive MAC operations with low-power AC operations and limiting each neuron to at most one spike within a coding window, thereby reducing synaptic operations and dynamic memory accesses.

Table 3: The calculation count and energy cost of ANN, rate coding and ours.

Method	Calculation Count	E_{comp} (J)	Spike Count	E_{sys} (J)	E_{total} (J)	Energy Reduction
ANN	4210.52 G MACs + 5.81 G ACs	19.37	—	42.18	61.55	—
Rate coding	95.98 G MACs + 10468.73 G ACs	9.86	1.9558 G	43.39	53.25	13.48%
Ours	**0.13 G MACs + 3377.82 G ACs**	**3.04**	**0.5681 G**	**18.70**	**21.74**	**64.68%**

6 Conclusion

We propose a gradient-guided finite-precision temporal coding framework for ANN-to-SNN MLLM conversion. By integrating adaptive temporal codebooks with layer-wise calibration, our method achieves near-lossless performance with frozen pretrained weights, substantially reducing computation and system energy to enable efficient multimodal inference.

AI USE STATEMENT

In this work, we used generative AI tools for language editing and basic code debugging. We did not use generative AI tools for any other tasks requiring disclosure. We reviewed and edited all AI-assisted content to ensure its accuracy and take full responsibility for the final content.

ETHICS STATEMENT

All participants in this work, as well as the paper submission, adhere to the ICLR Code of Ethics (`https://iclr.cc/public/CodeOfEthics`).

REPRODUCIBILITY STATEMENT

We confirm that the algorithms and results of this work are reproducible, and the experimental settings are described in the paper. The source code will be released upon acceptance.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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`.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, and Feng Zhao. Are we on the right way for evaluating large vision-language models? In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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, 2026. doi: 10.1609/AAAI.V40I3.37151.

Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Are snns really more energy-efficient than anns? an in-depth hardware-aware study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(3):731–741, 2023. doi: 10.1109/TETCI.2022.3214509.

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, 2015. doi: 10.1109/IJCNN.2015.7280696.

Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. doi: 10.1109/18.720541.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13558–13567, 2020.

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, San Francisco, CA, USA, 2014. IEEE. doi: 10.1109/ISSCC.2014.6757323.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702. IEEE, 2019.

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. doi: 10.52202/079017-2152.

Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: Towards optimal mapping from activation values to firing rates. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 14945–14974. PMLR, 2023.

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*, 2024.

Saeed Reza Kheradpisheh and Timothée Masquelier. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(6):2050027, 2020. doi: 10.1142/S0129065720500276.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.20.

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, pp. 6316–6325. PMLR, 2021.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023b.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, pp. 216–233. Springer, 2024b. doi: 10.1007/978-3-031-72658-3_13. URL https://arxiv.org/abs/2307.06281.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2507–2521, 2022. doi: 10.52202/068431-0182.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. doi: 10.1016/S0893-6080(97)00050-7.

Kaushik Roy, Abhishek Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019. doi: 10.1038/s41586-019-1677-2.

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.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8317–8326, 2019.

Qian Sun, Chengzhuo Lu, Wenyu Chen, Wenjie Wei, Jingya Wang, Jieyuan Zhang, Xiaoli Liu, Yalan Ye, Yang Yang, and Malu Zhang. Temporal-coded spiking transformer. In *MM '25: Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 2616–2624, 2025. doi: 10.1145/3746027.3754545.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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.

Xiaofeng Wu, Velibor Bojkovic, Bin Gu, Kun Suo, and Kai Zou. Ftbc: Forward temporal bias correction for optimizing ann-snn conversion. In *European Conference on Computer Vision*, 2024.

xAI. Realworldqa: A benchmark for real-world spatial understanding. `https://huggingface.co/datasets/xai-org/RealworldQA`, 2024. Dataset.

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.

Han Xu, Zhiyong Qin, Di Shang, Jiahong Zhang, Xuerui Qiu, Bo Lei, Tiejun Huang, Bo Xu, and Guoqi Li. Spikemllm: Spike-based multimodal large language models via modality-specific temporal scales and temporal compression. *arXiv preprint arXiv:2604.18610*, 2026a. doi: 10.48550/arXiv.2604.18610.

Han Xu, Xuerui Qiu, Baiyu Chen, Xinhao Luo, Xingrun Xing, Jiahong Zhang, Bo Lei, Tiejun Huang, Bo Xu, and Guoqi Li. Spike-driven large language model. *arXiv preprint arXiv:2604.16475*, 2026b. doi: 10.48550/arXiv.2604.16475.

Qu Yang, Malu Zhang, Jibin Wu, Kay Chen Tan, and Haizhou Li. Lc-ttfs: Toward lossless network conversion for spiking neural networks with ttfs coding. *IEEE Transactions on Cognitive and Developmental Systems*, 16(5):1626–1639, 2024. doi: 10.1109/TCDS.2023.3334010.

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, pp. 57367–57383. PMLR, 2024.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 57730–57754. PMLR, 2024.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

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, pp. 77558–77571. PMLR, 2025.

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.

APPENDIX

This appendix gives the proofs of the three theoretical statements in the main text, operator-level conversion details, implementation details, ablations, and additional model results.

A ADDITIONAL METHOD DETAILS AND PROOFS

A.1 FINITE-PRECISION CODEBOOK NOTATION

For an activation $u \in [0, u_{\max}]$, a finite temporal codebook is represented by positive reconstruction values $C = \{c_1, \dots, c_T\}$ and decision boundaries $q_0 = +\infty > q_1 > \cdots > q_T > 0$. The codebook decoder is

$$Q_C(u) = \sum_{t=1}^{T} c_t \mathbb{I}\{u \in [q_t, q_{t-1})\}. \tag{11}$$

The silent region $u < q_T$ is reconstructed as zero. Thus, a window with T possible firing times has at most $T + 1$ reconstruction levels, including silence. For an activation u_i, we define the sensitivity weight $s_i = (\partial \mathcal{L}/\partial u_i)^2$ and the sensitivity-weighted local distortion

$$\mathcal{E}(C) = \sum_{i=1}^{N} s_i \big(u_i - Q_C(u_i)\big)^2. \tag{12}$$

The empirical activation density is denoted by $p(u)$, and $\bar{s}(u) = \mathbb{E}[s_i \mid u_i = u]$ denotes the conditional mean sensitivity.

A.2 PROOF OF THEOREM 1

Proof. We use the standard high-resolution scalar-quantization approximation, in which the contribution of a sufficiently small cell of width $\Delta(u)$ to squared reconstruction error is $\Delta(u)^2/12$ up to higher-order terms (Gray & Neuhoff, 1998). Let $g(u)$ denote the normalized density of codebook cells, so that $\int_0^{u_{\max}} g(u)\,\mathrm{d}u = 1$. With T cells, the local cell width satisfies

$$\Delta(u) = \frac{1}{T g(u)} + o(T^{-1}). \tag{13}$$

Consequently, the leading term of the expected sensitivity-weighted distortion is

$$\mathcal{E}_T(g) = \frac{1}{12T^2} \int_0^{u_{\max}} \frac{p(u)\bar{s}(u)}{g(u)^2}\,\mathrm{d}u + o(T^{-2}). \tag{14}$$

Let $a(u) = p(u)\bar{s}(u)$. Minimizing the leading term subject to the density constraint is equivalent to minimizing

$$J[g] = \int_0^{u_{\max}} a(u) g(u)^{-2}\,\mathrm{d}u + \lambda \left(\int_0^{u_{\max}} g(u)\,\mathrm{d}u - 1 \right). \tag{15}$$

The pointwise stationarity condition is

$$-2a(u)g(u)^{-3} + \lambda = 0, \tag{16}$$

which gives $g(u) \propto a(u)^{1/3}$. Normalizing the density yields

$$g^*(u) = \frac{\big(p(u)\bar{s}(u)\big)^{1/3}}{\displaystyle\int_0^{u_{\max}} \big(p(x)\bar{s}(x)\big)^{1/3}\,\mathrm{d}x}. \tag{17}$$

Substitution into the high-resolution distortion expression gives

$$\mathcal{E}_T(g^*) = \frac{1}{12T^2} \left(\int_0^{u_{\max}} \big(p(u)\bar{s}(u)\big)^{1/3}\,\mathrm{d}u \right)^3 + o(T^{-2}), \tag{18}$$

which proves both the optimal density and the $O(T^{-2})$ scaling. The approximation assumes that the density and sensitivity are integrable, the occupied cells are sufficiently fine, and boundary cells contribute only higher-order terms. □

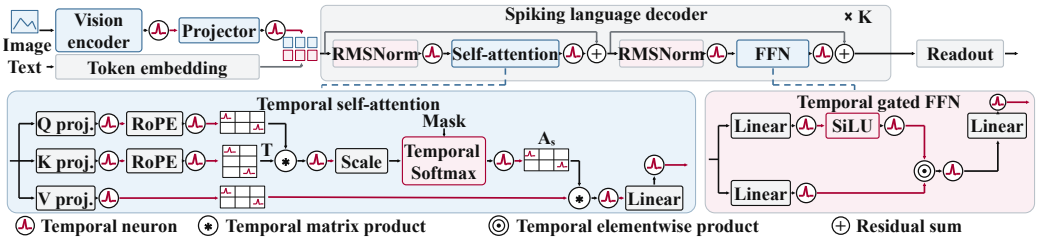


Figure 4: Architecture-level temporal conversion of the multimodal language model. Image and text streams are mapped to temporal representations, and the visual encoder, projector, Transformer attention, gated feed-forward blocks, residual paths, and readout are converted into event-driven operators.

A.3 Proof of Theorem 2

Proof. Write the ANN layer mapping as $\mathbf{H}_l^{\mathrm{ANN}} = f_l(\mathbf{H}_{l-1}^{\mathrm{ANN}})$ and the converted layer as

$$\mathbf{H}_l^{\mathrm{SNN}} = f_l(\mathbf{H}_{l-1}^{\mathrm{SNN}}) + \boldsymbol{\delta}_l, \tag{19}$$

where $\boldsymbol{\delta}_l$ is the local conversion residual induced by the codebook and satisfies $\|\boldsymbol{\delta}_l\|_2 \le \epsilon_l$. Define $\mathbf{R}_l = \mathbf{H}_l^{\mathrm{SNN}} - \mathbf{H}_l^{\mathrm{ANN}}$. If f_l is L_f-Lipschitz on the bounded activation domain, then

$$\|\mathbf{R}_l\|_2 \le \left\| f_l(\mathbf{H}_{l-1}^{\mathrm{SNN}}) - f_l(\mathbf{H}_{l-1}^{\mathrm{ANN}}) \right\|_2 + \|\boldsymbol{\delta}_l\|_2 \tag{20}$$

$$\le L_f \|\mathbf{R}_{l-1}\|_2 + \epsilon_l. \tag{21}$$

Because the input representation is shared, $\mathbf{R}_0 = 0$. Repeated substitution gives

$$\|\mathbf{R}_K\|_2 \le \epsilon_K + L_f \epsilon_{K-1} + \cdots + L_f^{K-1}\epsilon_1 = \sum_{l=1}^{K} L_f^{K-l}\epsilon_l. \tag{22}$$

This bound is conditional on the assumed local residual bounds. The teacher-validation accept/reject rule alone does not imply $\epsilon_l \le \epsilon_{\mathrm{threshold}}$. For layer-specific Lipschitz constants, the corresponding factor is $\prod_{j=l+1}^{K} L_j$ rather than L_f^{K-l}, with an empty product equal to one. □

A.4 Proof of Theorem 3

Proof. Let $\mathbf{C}^{(k)}$ denote the codebooks after calibration iteration k, with Φ evaluated on fixed calibration data and teacher targets. If a candidate is accepted, the strict verification rule requires $\Phi(\mathbf{C}^{(k+1)}) < \Phi(\mathbf{C}^{(k)})$; if it is rejected, the current codebook is retained and $\Phi(\mathbf{C}^{(k+1)}) = \Phi(\mathbf{C}^{(k)})$. Thus,

$$\Phi(\mathbf{C}^{(k+1)}) \le \Phi(\mathbf{C}^{(k)}), \qquad k \ge 0. \tag{23}$$

Both the KL divergence and the teacher cross-entropy are non-negative for valid probability distributions, and $\alpha > 0$, so $\Phi(\mathbf{C}) \ge 0$. Assuming $\Phi(\mathbf{C}^{(0)}) < \infty$, the non-increasing sequence $\{\Phi(\mathbf{C}^{(k)})\}$ is bounded below and therefore converges to a finite limit Φ_∞. For a finite run, the sequence can be extended by retaining the final codebook after termination. □

B Architecture-Level Temporal Conversion

Figure 4 summarizes the architecture-level conversion used in our MLLM setting. The visual encoder, multimodal projector, token embedding, self-attention, gated feed-forward network, residual paths, and readout are converted using TTFS-compatible Transformer primitives from TTFS-Former (Zhao et al., 2025). GFTC provides the finite-precision, gradient-guided codebooks and activation ranges used at the converted boundaries.

For a scalar activation $x \in [a, b]$, the continuous TTFS reference encoding and decoding are defined as

$$t_x = T_{\mathrm{ref}} - \tau x, \qquad \tau = \frac{\Delta t}{b - a}, \qquad T_{\mathrm{ref}} = T_{\mathrm{emit}} + b\tau, \tag{24}$$

and

$$\widehat{x} = \mathrm{clip}\left(\frac{T_{\mathrm{ref}} - t_x}{\tau}, a, b\right). \tag{25}$$

In the finite-precision GFTC implementation, each converted boundary and token index uses a codebook $C = \{c_1, \ldots, c_T\}$ with $c_1 > \cdots > c_T > 0$ and decision thresholds $q_0 = +\infty > q_1 > \cdots > q_T > 0$. The finite-precision spike representation and reconstruction are

$$s_t^C(x) = \mathbb{I}(q_t \le x < q_{t-1}), \qquad \mathcal{E}_C(x) = \big(s_1^C(x), \ldots, s_T^C(x)\big), \tag{26}$$

$$\mathcal{D}_C(\mathbf{s}) = \sum_{t=1}^{T} c_t s_t, \qquad \mathcal{Q}_C(x) = \mathcal{D}_C(\mathcal{E}_C(x)) = \sum_{t=1}^{T} c_t \mathbb{I}(q_t \le x < q_{t-1}). \tag{27}$$

Values below q_T produce the silent code and are reconstructed as zero. For readability, boundary- and token-specific subscripts are suppressed below. The continuous maps above provide the reference TTFS parameterization, whereas the finite maps $\mathcal{E}_C$ and $\mathcal{D}_C$ are used during GFTC inference.

For a generic operator $\mathcal{O}$ with input and output codebooks, its finite-precision temporal realization is written as

$$\mathcal{O}^{\mathrm{TTFS}}_{C_{\mathrm{in}} \to C_{\mathrm{out}}}(\mathbf{s}) = \mathcal{E}_{C_{\mathrm{out}}}\left(\mathcal{O}(\mathcal{D}_{C_{\mathrm{in}}}(\mathbf{s}))\right). \tag{28}$$

For linear and product modules, the inner sums are implemented as event-weighted accumulations over the nonzero spike indicators.

Patch projection and token embedding. A visual patch projection is converted as

$$\mathbf{X}_{\mathrm{img}} = \mathrm{Patch}(I)\mathbf{W}_p + \mathbf{b}_p, \qquad \mathbf{T}_{\mathrm{img}} = \mathcal{E}_{C_{\mathrm{img}}}(\mathbf{X}_{\mathrm{img}}). \tag{29}$$

For text tokens, the embedding lookup is retained and its output is encoded temporally:

$$\mathbf{X}_{\mathrm{text}}[i,:] = \mathbf{E}[\mathrm{id}_i], \qquad \mathbf{T}_{\mathrm{text}} = \mathcal{E}_{C_{\mathrm{text}}}(\mathbf{X}_{\mathrm{text}}). \tag{30}$$

Other linear projections in the visual encoder and multimodal projector use the same codeword-weighted accumulation construction.

Linear projections and self-attention. A linear or affine operator is converted as

$$\mathrm{Linear}_{\mathrm{TTFS}}(\mathbf{s}_x) = \mathcal{E}_{C_y}\left(\mathbf{W}\mathcal{D}_{C_x}(\mathbf{s}_x) + \mathbf{b}\right), \qquad y_i = \sum_j W_{ij} x_j + b_i. \tag{31}$$

The query, key, and value projections therefore become

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q + \mathbf{b}_Q, \qquad \mathbf{K} = \mathbf{X}\mathbf{W}_K + \mathbf{b}_K, \qquad \mathbf{V} = \mathbf{X}\mathbf{W}_V + \mathbf{b}_V, \tag{32}$$

where the displayed affine forms describe the decoded arithmetic and each output is re-encoded using the codebook of the corresponding boundary.

The finite-precision multiplication module is

$$\mathrm{Mul}_{\mathrm{TTFS}}(\mathbf{s}_u, \mathbf{s}_v) = \mathcal{E}_{C_o}\left(\mathcal{D}_{C_u}(\mathbf{s}_u)\mathcal{D}_{C_v}(\mathbf{s}_v)\right). \tag{33}$$

Thus, the scaled attention scores are computed as

$$S_{ij} = \frac{1}{\sqrt{d_k}} \sum_{r=1}^{d_k} \mathrm{Mul}_{\mathrm{TTFS}}(Q_{ir}, K_{jr}), \tag{34}$$

and the attention–value product is

$$\mathbf{O}_i = \sum_j \mathrm{Mul}_{\mathrm{TTFS}}(A_{ij}, \mathbf{V}_j). \tag{35}$$

When rotary position encoding is used, fixed position-dependent rotations are applied to the decoded query and key values and the results are re-encoded:

$$\mathbf{T}^{\mathrm{rot}}_{Q_i} = \mathcal{E}_{C_Q}\left(R_i \mathcal{D}_{C_Q}(\mathbf{T}_{Q_i})\right), \qquad \mathbf{T}^{\mathrm{rot}}_{K_j} = \mathcal{E}_{C_K}\left(R_j \mathcal{D}_{C_K}(\mathbf{T}_{K_j})\right). \tag{36}$$

Softmax. For each attention row, the temporal Softmax is decomposed into max, subtraction, log-sum-exp, and exponential modules:

$$m_i = \max_{j:M_{ij}=1} S_{ij}, \qquad \widetilde{S}_{ij} = S_{ij} - m_i \quad \text{for } M_{ij} = 1, \tag{37}$$

$$z_i = \log \sum_{j:M_{ij}=1} \exp(\widetilde{S}_{ij}), \qquad A_{ij} = \exp(\widetilde{S}_{ij} - z_i) \quad \text{for } M_{ij} = 1. \tag{38}$$

For masked entries with $M_{ij} = 0$, the corresponding terms are omitted from the temporal Softmax and A_{ij} is set to zero before the attention–V accumulation. The max, log-sum-exp, exponential, and intermediate attention values are represented by their corresponding finite codebooks and re-encoded before being passed to the next temporal module.

Nonlinearities and normalization. For SiLU and GELU, the value-domain functions are

$$\mathrm{SiLU}(u) = u\sigma(u), \qquad \sigma(u) = \frac{1}{1+\exp(-u)}, \tag{39}$$

$$\mathrm{GELU}(u) = u\Phi(u), \qquad \Phi(u) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{u}{\sqrt{2}}\right)\right]. \tag{40}$$

Following the TTFS input-transform construction, the corresponding kernels use the derivatives

$$\eta_{\mathrm{SiLU}}(s) = \mathbf{1}_{s\geq 0}\, \sigma(u_C)\left[u_C + 1 - u_C\sigma(u_C)\right], \tag{41}$$

$$\eta_{\mathrm{GELU}}(s) = \mathbf{1}_{s\geq 0}\left[\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left(\frac{u_C}{\sqrt{2}}\right) + \frac{u_C}{\sqrt{2\pi}}\exp\left(-\frac{u_C^2}{2}\right)\right], \tag{42}$$

where the finite-precision input is

$$u_C = \mathcal{D}_C(\mathcal{E}_C(u)) = \sum_{t=1}^{T} c_t s_t^C(u), \qquad s_t^C(u) = \mathbb{I}(q_t \leq u < q_{t-1})\,. \tag{43}$$

The output of each nonlinear kernel is re-encoded with the codebook of the next converted boundary. During calibration, the straight-through estimator defined in Appendix C.2 is used for the discrete assignment.

LayerNorm is decomposed into mean, centering, variance, reciprocal square-root, multiplication, and affine operators:

$$\mu = \frac{1}{d}\sum_{r=1}^{d} x_r, \qquad v = \frac{1}{d}\sum_{r=1}^{d}(x_r - \mu)^2, \tag{44}$$

$$\mathrm{LN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x}-\mu}{\sqrt{v+\varepsilon}} + \beta. \tag{45}$$

For RMSNorm-based blocks, we use the corresponding decomposition

$$\mathrm{RMSNorm}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x}}{\sqrt{d^{-1}\sum_{r=1}^{d} x_r^2 + \varepsilon}}, \tag{46}$$

which omits mean subtraction and bias. The intermediate mean, variance or mean-square, reciprocal square-root, and elementwise products are computed from decoded codeword values and re-encoded at their respective output boundaries.

Gated feed-forward network. A gated feed-forward block is converted as

$$\mathbf{g} = \mathrm{SiLU}(\mathbf{X}\mathbf{W}_g + \mathbf{b}_g), \qquad \mathbf{u} = \mathbf{X}\mathbf{W}_u + \mathbf{b}_u, \tag{47}$$

$$\mathrm{FFN}(\mathbf{X}) = (\mathbf{g} \odot \mathbf{u})\mathbf{W}_d + \mathbf{b}_d. \tag{48}$$

The two projections use the finite-precision TTFS linear module, the activation uses the corresponding re-encoded nonlinear module, and $\odot$ is implemented by the finite-precision multiplication module. GELU is used in the same way when specified by the original architecture.

Table 4: Operator correspondence used in the architecture-level conversion.

ANN operator	TTFS realization
Linear and patch projection	$\mathcal{E}_{C_y}(\mathbf{W}\mathcal{D}_{C_x}(\mathbf{s}) + \mathbf{b})$
Q, K, and V projections	Finite-precision TTFS affine projections
$QK^\top$	TTFS multiplication and event-weighted accumulation
Attention–V product	TTFS multiplication and event-weighted accumulation
Softmax	Finite-precision max, log-sum-exp, subtraction, and exponential modules
LayerNorm/RMSNorm	Mean or mean-square, reciprocal square-root, and multiplication modules
SiLU/GELU and gated FFN	Finite-precision nonlinear kernels and TTFS products
Residual addition and readout	Event-driven addition and finite-precision TTFS decoding

Residual paths and readout. A residual path is implemented by event-driven addition:

$$\mathbf{Y} = \mathbf{X} + \mathbf{F}(\mathbf{X}), \qquad \mathbf{T}_Y = \mathcal{E}_{C_Y}\left(\mathcal{D}_{C_X}(\mathbf{T}_X) + \mathcal{D}_{C_F}(\mathbf{T}_F)\right). \tag{49}$$

The final readout is converted as

$$\mathbf{Z} = \mathbf{H}\mathbf{W}_o + \mathbf{b}_o, \qquad \widehat{\mathbf{Z}} = \mathcal{D}_{C_Z}\left(\mathcal{E}_{C_Z}(\mathbf{Z})\right) = \mathcal{Q}_{C_Z}(\mathbf{Z}), \tag{50}$$

with token prediction given by

$$\widehat{y} = \arg\max_j \widehat{Z}_j. \tag{51}$$

Thus, the final prediction is obtained from decoded finite-precision logits rather than from an unquantized floating-point bypass.

All operator correspondences in Table 4 use the finite-precision codebook interface defined above. The continuous equations are value-domain descriptions of the corresponding spike-level modules and do not represent an unquantized inference bypass.

The pretrained weights remain fixed during conversion. For each converted boundary, GFTC estimates activation ranges and gradient sensitivities, initializes a finite-precision codebook using the density rule in Theorem 1, and updates the codebook along the executed SNN trajectory. Image and text tokens use separate codebooks when their activation statistics differ, while fusion boundaries use a shared codebook when the modalities are jointly processed.

C Experimental Protocol and Reproducibility Details

C.1 Experimental Setup and Conversion Strategy

We evaluate LLaVA-1.5-7B and LLaVA-NeXT-v1.6-Vicuna-7B on TextVQA, GQA, POPE, ScienceQA-IMG, and MM-Vet. The Qwen3-VL experiments use the 4B, 8B, and 32B variants and the four benchmarks reported in the main text. The original full-precision ANN is used as the accuracy reference. All rate, phase, and temporal configurations are compared under the stated information capacity. For the LLaVA experiments, the 6-bit and 8-bit settings correspond to $T = 64$ and $T = 256$, respectively. The weights remain at 32-bit precision in the energy reference model, while temporal activations use the stated coding precision. To ensure a fair comparison, we additionally quantize intermediate activations to enforce identical activation precision, while preserving the original attention implementation of SpikeQuant.

We distinguish the temporal period T from the encoding-capacity label B. The value of B is defined according to the corresponding coding scheme when comparing different methods. For GFTC and TTFS temporal coding, we report the nominal label $B = \log_2 T$, while rate and phase coding follow their respective scheme-specific mappings. The silent output is treated as an implicit zero code and is not counted as an additional temporal slot.

The conversion uses two offline stages. First, a fixed calibration subset of 384 VQA examples is passed through the frozen ANN to collect activation values and task gradients. For each layer and token index, we estimate the empirical density, sensitivity density, and activation range. The inverse CDF of the density in Theorem 1 partitions each range into T bins, and each bin is represented by its sensitivity-weighted centroid. The same calibration subset is used for all temporal settings and calibration ablations, while task evaluation uses the corresponding held-out benchmark split.

Second, the candidate codebook is inserted into the SNN, and the upstream converted trajectory is executed on the calibration examples. The candidate is evaluated against the ANN teacher objective. Accepted candidates replace the current codebook, whereas rejected candidates retain the current codebook and reduce the perturbation magnitude. Inference uses only the frozen model weights and the resulting codebooks.

C.2 Implementation of Discrete Codebook Calibration

Algorithm 1 summarizes the calibration procedure at a high level. The hard codebook assignment for an activation $u_i^{(l)}$ is

$$Q_{\mathcal{C}^{(l)}}(u_i^{(l)}) = c_t^{(l)}, \quad q_t^{(l)} \le u_i^{(l)} < q_{t-1}^{(l)}. \tag{52}$$

Because the assignment index is discrete, gradients are computed with the straight-through estimator

$$\widetilde{Q}_{\mathcal{C}^{(l)}}(u_i^{(l)}) = Q_{\mathcal{C}^{(l)}}(u_i^{(l)}) + u_i^{(l)} - \mathrm{stopgrad}(u_i^{(l)}). \tag{53}$$

The activation sensitivity used during codebook construction is

$$s_i^{(l)} = \left(\frac{\partial \mathcal{L}_{\mathrm{ANN}}}{\partial u_i^{(l)}} \right)^2. \tag{54}$$

During trajectory calibration, the codebook gradient is obtained by back-propagating the validation objective through the straight-through quantizer:

$$g_t^{(l)} = \frac{\partial \Phi}{\partial c_t^{(l)}}. \tag{55}$$

For the local sensitivity-weighted distortion, the corresponding surrogate gradient is

$$g_t^{(l)} = 2 \sum_{i \in \mathcal{B}_t^{(l)}} s_i^{(l)} \left(c_t^{(l)} - u_i^{(l)} \right), \tag{56}$$

where $\mathcal{B}_t^{(l)}$ denotes the activations assigned to codeword $c_t^{(l)}$.

The sensitivity-driven perturbation $\Delta C^{(l)}$ is chosen in the negative gradient direction and initialized such that

$$\max_t \left| \Delta c_t^{(l)} \right| = 0.065. \tag{57}$$

After a rejected candidate, the current perturbation is multiplied by

$$\beta = 0.5. \tag{58}$$

The candidate codebook is then obtained as

$$C_{\mathrm{cand}}^{(l)} = \Pi_{\downarrow,\xi} \left(C_{\mathrm{curr}}^{(l)} + \Delta C^{(l)} \right), \tag{59}$$

where

$$\Pi_{\downarrow,\xi}(\mathbf{z}) = \arg\min_{\mathbf{c}} \|\mathbf{c} - \mathbf{z}\|_2^2 \tag{60}$$

subject to

$$c_t - c_{t+1} \ge \xi, \qquad c_T \ge \xi, \qquad \xi = 0.006. \tag{61}$$

This projection preserves the strict codebook ordering $c_1 > \cdots > c_T > 0$. The decision boundaries are fixed after initialization, so the ordering $q_0 > q_1 > \cdots > q_T > 0$ is also preserved.

A candidate is accepted only when

$$\Phi(C_{\mathrm{cand}}) < \Phi(C_{\mathrm{curr}}) - 10^{-10}. \tag{62}$$

For each layer, at most nine candidate trials are performed before the next layer is visited. As illustrated in Figure 3, the measured objective usually changes little after roughly ten complete layer sweeps. We therefore stop calibration when either one complete sweep accepts no update or the number of complete sweeps reaches 20. All teacher outputs, calibration examples, activation sensitivities, and pretrained weights remain fixed throughout calibration.

Table 5: Cumulative ablation of GFTC components on LLaVA-1.5-7B TextVQA at 8-bit precision ($T = 256$).

Gradient-guided optimization	Iterative calibration	Token-wise dynamic ranges	Modality-specific codebooks	Acc. (%)
×	×	×	×	47.63
✓	×	×	×	52.48
✓	✓	×	×	56.31
✓	✓	✓	×	57.76
✓	✓	✓	✓	**58.06**

Table 6: Scope ablation of GFTC on a fully converted SNN derived from LLaVA-1.5-7B, evaluated on TextVQA at 8-bit precision ($T = 256$). All configurations convert the complete model to an SNN; the checkmarks indicate the components to which GFTC is applied.

Vision encoder	Projector	LLM	Configuration	Acc. (%)
×	×	×	SNN baseline (w/o GFTC)	47.63
✓	×	×	GFTC on vision encoder	48.06
✓	✓	×	GFTC on vision encoder + projector	47.82
✓	✓	✓	GFTC on the full MLLM	**58.06**

D Ablation Studies

D.1 Cumulative Component Ablation

Table 5 reports the cumulative ablation on LLaVA-1.5-7B TextVQA at 8-bit precision. The first row uses none of the four proposed components, and each subsequent row adds one component.

The largest improvement appears when the gradient-guided codebook is introduced, showing that allocating finite temporal resolution according to task sensitivity is more important than uniformly increasing the number of representable values. Iterative calibration then reduces the trajectory mismatch accumulated across layers. Token-wise ranges provide a further correction for local dynamic-range variation, while modality-specific codebooks provide a smaller but consistent improvement by separating the statistics of visual and textual tokens. The cumulative pattern supports the design in which codebook allocation and trajectory calibration address complementary sources of conversion error.

D.2 Conversion Scope Ablation

Table 6 isolates the effect of applying the conversion to different parts of the multimodal architecture.

The scope ablation starts from a fully converted SNN without any GFTC component. GFTC is then enabled progressively on the vision encoder, the multimodal projector, and the language model. Applying GFTC only to the vision encoder or to the vision encoder and projector produces limited changes, whereas enabling it throughout the language model yields the largest improvement. This result indicates that the language decoder is the dominant source of recoverable conversion error, consistent with the repeated propagation of approximation errors through decoder blocks and the sensitivity of decoder activations to channel outliers. It also supports applying GFTC to the language pathway rather than restricting it to the visual components.

D.3 Temporal-Budget Ablation

Increasing the temporal budget from 6 to 8 bits gives a clear improvement, while additional temporal states yield only a small change and then saturate. This indicates that the proposed codebook and calibration procedure recover most of the useful precision within the 8-bit setting, which is the setting used for the energy comparison.

Table 7: Temporal-budget ablation on LLaVA-1.5-7B TextVQA with all four GFTC components enabled.

Bits	T	Acc. (%)
6	64	56.96
8	256	**58.06**
10	1024	58.24
12	4096	58.24

Table 8: Sensitivity to calibration hyperparameters on LLaVA-1.5-7B TextVQA at 8-bit precision ($T = 256$).

Initial perturbation $\max_t \lvert\Delta c_t\rvert$	β	TextVQA Acc. (%)
0.065	0.5	**58.06**
0.0325	0.5	57.82
0.13	0.5	57.71
0.065	0.25	57.84
0.065	0.75	57.96

D.4 Calibration Hyperparameter Ablation

Table 8 studies the sensitivity of the calibration procedure to the initial perturbation magnitude and the decay factor β. The experiment uses LLaVA-1.5-7B on TextVQA with the 8-bit temporal setting ($T = 256$). The default configuration initializes the perturbation with $\max_t |\Delta c_t^{(l)}| = 0.065$ and uses $\beta = 0.5$.

The default setting achieves the highest accuracy among the tested configurations. Both substantially smaller and larger initial perturbations lead to slightly lower accuracy, while changing the decay factor produces a smaller variation. These results support using an initial perturbation magnitude of 0.065 and $\beta = 0.5$ as the default calibration settings.

E Energy Accounting Details

We estimate the energy of one LLaVA-1.5-7B image–question prefill using the event-driven accounting structure of SpikedAttention (Hwang et al., 2024). The arithmetic reference follows the 45-nm CMOS operation costs used in the main text and in TTFSFormer (Zhao et al., 2025). The workload contains 604 sequence positions and one final vocabulary readout. We use 32-bit weights, 8-bit activation coding, $T = 256$ for both rate and temporal coding, and a repeat weight-reuse factor of $R = 1.5$.

The computation and system terms are

$$E_{\text{comp}} = e_{\text{MAC}} N_{\text{MAC}} + e_{\text{AC}} N_{\text{AC}}, \tag{63}$$

$$E_{\text{sys}} = e_{32}(R_{32} + W_{32}) + e_1(R_1 + W_1) + e_{\text{cmp}} N_{\text{cmp}}, \tag{64}$$

with $e_{\text{MAC}} = 4.6\,\text{pJ}$, $e_{\text{AC}} = 0.9\,\text{pJ}$, $e_{32} = 5\,\text{pJ}$, $e_1 = \frac{5}{32}\,\text{pJ}$, and $e_{\text{cmp}} = 0.9\,\text{pJ}$. Here, R_{32} and W_{32} include 32-bit weight, activation, and state reads and writes, R_1 and W_1 count one-bit spike-payload transfers, and N_{cmp} counts threshold comparisons. Repeated weight reads receive the reuse credit $R = 1.5$, while first reads and event payload transfers remain fully charged.

All operation and access counts are obtained from the operator-level execution ledger of a single LLaVA-1.5-7B image–question prefill with batch size one. Each input contains 604 positions, including 576 visual tokens and 28 text tokens. The workload includes the visual encoder, multimodal projector, language-model Transformer blocks, and one vocabulary projection at the final position; autoregressive answer generation is excluded. For each linear, projection, attention, and feed-forward operator l, let A_l denote the number of input activations, O_l the number of output activations, W_l the number of weight elements, and F_l the operator fan-out. The ANN MAC count is accumulated as

$$N_{\text{MAC}}^{\text{ANN}} = \sum_l A_l F_l, \tag{65}$$

Table 9: Results on LLaVA-NeXT-7B. "Max spikes" indicates the maximum number of spikes a neuron can emit per temporal window. "Spike attention" indicates whether attention is fully spike-driven. "Bits" denotes the information capacity encodable within the temporal window.

Method	Spike attention	Max spikes	Bits	TextVQA	GQA	POPE	SQA-IMG	MM-Vet	Avg. Acc.
ANN	—	—	32	60.33	62.48	86.82	67.47	40.02	63.42
SpikeLLM	✓	256	8	58.76	60.94	85.17	65.83	37.91	61.72
SpikeLLM	✓	64	6	54.12	56.48	80.76	62.91	33.74	57.60
PhaseSNN	✓	8	8	58.92	61.38	86.82	66.17	38.34	62.33
PhaseSNN	✓	6	6	58.44	60.83	85.37	65.71	37.82	61.63
TTFSFormer	✓	1	13	45.18	48.94	77.63	61.82	29.47	52.61
TTFSFormer	✓	1	10	34.76	42.31	74.28	61.16	25.93	47.69
SpikeQuant	×	1	8	58.93	61.36	86.74	65.92	37.96	62.18
SpikeQuant	×	1	6	58.26	60.44	85.48	65.13	37.09	61.28
GFTC (Ours)	✓	**1**	**8**	**60.26**	**62.57**	**86.77**	**67.54**	**40.06**	**63.44**
GFTC (Ours)	✓	**1**	**6**	**59.38**	**61.86**	**86.08**	**66.71**	**39.21**	**62.65**

which includes the linear projections, $QK^\top$ products, attention–V products, feed-forward projections, and the final vocabulary projection. For spike-based execution, the event ledger records the number of emitted input events S_l and their fan-out. The corresponding AC count is

$$N_{\mathrm{AC}} = \sum_l S_l F_l, \tag{66}$$

while residual dense operators that do not receive spike inputs contribute to N_{MAC}. Thus, the reported rate-coding and temporal-coding rows include the same operator set as the ANN reference, with only the event-dependent arithmetic replaced by AC operations.

For system energy, R_{32} and W_{32} are accumulated from weight, activation, intermediate-output, KV-cache, and neuron-state reads and writes recorded for every operator invocation. The prefill KV cache is written when keys and values are produced and read by subsequent attention products within the same forward pass; no additional autoregressive decoding or generated-token cache access is included. Weight accesses are counted once for the first invocation of each matrix and repeated accesses are charged using the declared weight-reuse factor $R = 1.5$. Event payload accesses are counted separately: R_1 is the number of one-bit spike payloads delivered to receiving operators, and W_1 is the number of emitted spike payloads written to the next operator or event buffer. Finally, N_{cmp} counts the threshold comparisons performed for the corresponding temporal neuron updates. All counts are summed over the 604 input positions and all converted layers before substitution into the computation- and system-energy equations.

The spike count in Table 3 is the measured input-event scope for the replayed Linear and patch interfaces. The Rate coding row uses the multi-spike schedule and the visual attention conversion scope described in the main text, while the temporal row uses the single-spike operator construction. The lower system cost of the temporal row follows from fewer synaptic events, fewer payload transfers, and fewer state checkpoints under the same $T = 256$ window.

F Additional Results on LLaVA-NeXT-7B

Table 9 extends the comparison to LLaVA-NeXT-7B across five multimodal benchmarks. "Max spikes" denotes the maximum number of spikes emitted by one neuron within a temporal window, and "Spike attention" indicates whether the attention computation is fully spike-driven. "Bits" denotes the information capacity of the coding window, and "Avg. Acc." is the arithmetic mean over the five displayed benchmarks.

The additional LLaVA-NeXT results show the same qualitative behavior as the LLaVA-1.5 experiments. GFTC maintains the advantage of fully spike-driven attention while using a single spike per neuron, and the advantage remains under the lower 6-bit budget. The result supports the transfer of the codebook and trajectory-calibration design to a model with a different visual-token processing path.