TOWARD ACCURATE FEW-TIMESTEP ANN TO SNN CONVERSION OF LLMS THROUGH LEARNED ROTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

ANN-to-SNN conversion offers a practical route to transferring pretrained large language models (LLMs) to spiking neural networks, but preserving model accuracy with only a few timesteps remains challenging. Multi-threshold neurons (MTNs) have enabled nearly lossless conversion of smaller-scale models with only a few timesteps. However, directly applying MTNs to large language models at similarly low timestep budgets leads to substantial accuracy degradation. We identify a mismatch between pretrained activation distributions and the nonuniform output levels of MTNs, with pronounced conversion errors concentrated in channels containing activation outliers. We propose a task-driven rotation calibration framework that adapts activation distributions to MTN conversion. The framework learns orthogonal rotations through the converted model's next-token loss, redistributing activations across channels to substantially mitigate the impact of activation outliers on MTN conversion. To enable efficient and accurate calibration, we introduce the equivalent parallel multi-threshold neuron (EPMTN). EPMTN requires only a single forward and backward pass, reducing calibration cost while mitigating the accumulation of surrogate-gradient approximation errors across timesteps. Experiments on LLaMA2-7B and LLaMA3-8B across four language-modeling corpora and four reasoning benchmarks demonstrate SOTA conversion performance with only four timesteps. These results highlight the potential of learned rotations and EPMTN calibration as a path toward accurate, low-timestep spiking LLMs.

1 INTRODUCTION

Spiking neural networks (SNNs) provide an event-driven computing paradigm in which neurons communicate through sparse spikes (Maass, 1997; Rueckauer et al., 2017). Advances in spiking self-attention and autoregressive language modeling have extended the applicability of SNNs to Transformer architectures and language generation (Zhou et al., 2023; Yao et al., 2023; Zhu et al., 2023). SpikingBrain further integrates spiking mechanisms into 7B- and 76B-parameter architectures, bringing this paradigm to the scale of large language models (LLMs) (Pan et al., 2025). At this scale, the cost of pretraining makes transferring existing models particularly attractive. ANN-to-SNN conversion offers a practical route by transferring pretrained artificial neural network (ANN) parameters and approximating continuous activations with neuronal outputs (Cao et al., 2015; Diehl et al., 2015; Rueckauer et al., 2017). Recent conversion methods have improved the preservation of Transformer computations and pretrained model accuracy through nonlinear compensation and conversion calibration (Jiang et al., 2024; Huang et al., 2024; You et al., 2024; Chen et al., 2025; 2026). Their practical utility nevertheless depends on the temporal cost of this approximation (Bu et al., 2022b; Hao et al., 2023). Therefore, preserving model accuracy with only a few timesteps remains a central requirement for spiking LLM conversion (Chen et al., 2026; Zheng et al., 2026).

Multi-threshold neurons (MTNs) improve the representational capacity of each timestep through multiple signed firing amplitudes (Huang et al., 2024), enabling nearly lossless conversion of smaller-scale models within two to four timesteps (Huang et al., 2025). However, applying MTNs directly to pretrained LLaMA2-7B and LLaMA3-8B still causes substantial degradation at small timestep budgets. Our channel-wise analysis reveals pronounced conversion-error peaks in chan-

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.

nels containing activation outliers (Figure 1). Pretrained activation distributions are not optimized for MTN conversion, leaving a mismatch between the nonuniform output levels of MTNs and the activation distributions to be represented.

Inspired by rotation-based LLM quantization (Ashkboos et al., 2024; Liu et al., 2025), we develop a rotation-based ANN-to-SNN conversion framework that adapts pretrained activation distributions to multi-threshold neurons. Orthogonal transformations redistribute activations across channels, while corresponding weight transformations preserve the original ANN function. However, although orthogonal rotations can mitigate activation outliers to some extent, they are not optimized for the nonuniform output levels of MTNs, leaving substantial scope for improving conversion accuracy. We therefore learn the rotation matrices through the converted model's next-token loss and introduce **EPMTN** to support this optimization. EPMTN computes the same time-averaged output as an MTN simulated over multiple timesteps in a single evaluation. EPMTN requires only a single forward and backward pass, reducing calibration cost while mitigating the accumulation of surrogate-gradient approximation errors across timesteps. By combining the rotation calibration framework with EPMTN, our method achieves SOTA conversion performance with only four timesteps. Our main contributions are as follows:

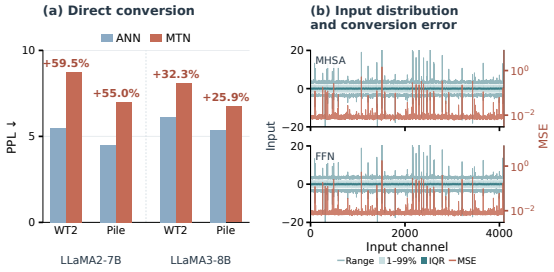

Figure 1: **Few-timestep conversion loss.** (a) PPL increases at $T = 4$. (b) Input distributions and conversion error. Range: min–max range. 1–99%: 1–99% band. IQR: interquartile range. MSE: per-channel input MSE (warm-colored curve, log scale), over 8,192 tokens.

- We develop a rotation-based ANN-to-SNN conversion framework for LLMs. Within this framework, a task-driven rotation calibration method uses the converted model's task loss to learn activation rotations adapted to MTN outputs at a low timestep budget.
- We introduce EPMTN to replace multi-timestep MTN simulation with a single forward and backward pass during calibration. This formulation reduces calibration cost and mitigates the accumulation of surrogate-gradient approximation errors across timesteps, enabling efficient and accurate rotation calibration.
- We evaluate the framework on LLaMA2-7B and LLaMA3-8B across four language-modeling corpora and four reasoning benchmarks. Our method achieves SOTA conversion performance with only four timesteps, including WikiText-2 perplexities of 6.21 and 6.99 on the two models, respectively.

2 Related Work

Conversion of Transformers and LLMs. ANN-to-SNN conversion provides a way to transfer pretrained Transformer capabilities into spiking computation. Spatio-temporal approximation addresses attention and nonlinear operations through temporal and spatial constructions (Jiang et al., 2024), whereas expectation compensation leverages accumulated temporal information to preserve nonlinear computations (Huang et al., 2024). To prevent precision degradation in normalization and continuous representations during spike transmission, SpikeZIP-TF establishes an equivalence between quantized Transformers and their spiking counterparts using a quantization-aware source model (You et al., 2024). These approaches establish the computational foundations for converting Transformer architectures. At language-model scale, conversion additionally needs to preserve pretrained behavior with an affordable adaptation budget. FAS combines source-model fine-tuning with coarse-to-fine conversion calibration (Chen et al., 2025), whereas LAS develops specialized neurons and spike-equivalent Transformer components to accommodate activation outliers and nonlinear operations (Chen et al., 2026). Distribution-aware multi-granularity phase coding further demonstrates that neuron parameters can be learned from activation distributions while keeping pretrained weights

frozen (Zheng et al., 2026). Together, these methods make the representation of pretrained activations a central design choice in LLM conversion.

Neurons for conversion. The conversion neuron determines the values that can be represented within a finite temporal window. Classical integrate-and-fire conversion approximates ANN activations through spike rates. Reset by subtraction, scale normalization, and membrane-potential initialization improve the correspondence between firing rates and continuous activations (Rueckauer et al., 2017; Bu et al., 2022a). Few-spike neurons optimize temporal dynamics to approximate activation functions with weighted spikes (Stöckl & Maass, 2021), and multi-basis exponential decay neurons extend this strategy to nonlinear Transformer operations (Wang et al., 2026). Distribution-aware multi-granularity phase coding learns multiple bases to allocate representable values according to activation density (Zheng et al., 2026). Unlike the aforementioned neurons, multi-threshold neurons support conversion with fewer timesteps by offering multiple firing amplitudes at each step. These neurons have been used for Transformer conversion (Huang et al., 2024) and combined with differential coding and distribution-based threshold optimization for training-free conversion (Huang et al., 2025). We adopt multi-threshold neurons and further optimize them.

3 PRELIMINARIES

3.1 MULTI-THRESHOLD NEURONS

Multi-threshold neurons (MTNs) represent signals with multiple signed firing thresholds, enabling ANN-to-SNN conversion over short temporal windows (Huang et al., 2024; 2025). Let $\theta > 0$ be the base threshold and n the number of thresholds per sign. We write the threshold ratios as

$$\lambda_i = 2^{-i}, \quad i = 0, \dots, n-1, \qquad \lambda_{\min} = 2^{-(n-1)}, \tag{1}$$

where λ_i is the ith threshold ratio and $\lambda_{\min}$ is the smallest ratio. The physical firing thresholds are $\pm\theta\lambda_i$. We express currents and membrane potentials in units of θ. Let $I[t]$ be the input current, and $m[t]$ and $v[t]$ the membrane potentials before and after firing. The neuron selects the largest threshold not exceeding $|m[t]|$ and emits on the corresponding positive or negative channel. With binary channel spikes $s_i^{\pm}[t]$, its dynamics are

$$\begin{aligned} m[t] &= v[t-1] + I[t], & v[t] &= m[t] - \xi[t], \\ \xi[t] &= \sum_{i=0}^{n-1} \lambda_i \big(s_i^+[t] - s_i^-[t]\big), & s_i^{\pm}[t] &\in \{0,1\}. \end{aligned} \tag{2}$$

where $\xi[t]$ is the signed emitted signal in threshold units. $s_i^+[t]$ and $s_i^-[t]$ indicate firing on the positive and negative channels associated with λ_i, respectively. At most one channel fires per step. Equality at a threshold triggers a spike, and no channel fires when $|m[t]| < \lambda_{\min}$. Subtracting the emitted signal retains the residual membrane for the next step. This channel-based description follows the established MTN representation (Huang et al., 2025).

3.2 ANN-TO-SNN CONVERSION WITH MTNS

The conversion principle is to match an ANN activation with the time-averaged response of its corresponding spiking neuron (Rueckauer et al., 2017; Bu et al., 2022a; Huang et al., 2025). For a continuous activation x, an MTN receives $I[t] = x/\theta$ for T steps, starting from $v[0] = 0$. Its response in the original activation units is

$$r_T^{\mathrm{MTN}}(x;\theta) = \frac{\theta}{T}\sum_{t=1}^{T}\xi[t], \qquad x - r_T^{\mathrm{MTN}}(x;\theta) = \frac{\theta v[T]}{T}. \tag{3}$$

where $r_T^{\mathrm{MTN}}(x;\theta)$ denotes the time-averaged MTN output in the original activation units and $v[T]$ is the final residual membrane potential. The second identity follows by summing the membrane updates. It connects conversion fidelity directly to the residual charge after T steps. The base threshold sets the firing range, while the threshold levels and temporal window determine the attainable neuronal responses. Each new input starts from zero membrane, and its base threshold remains fixed

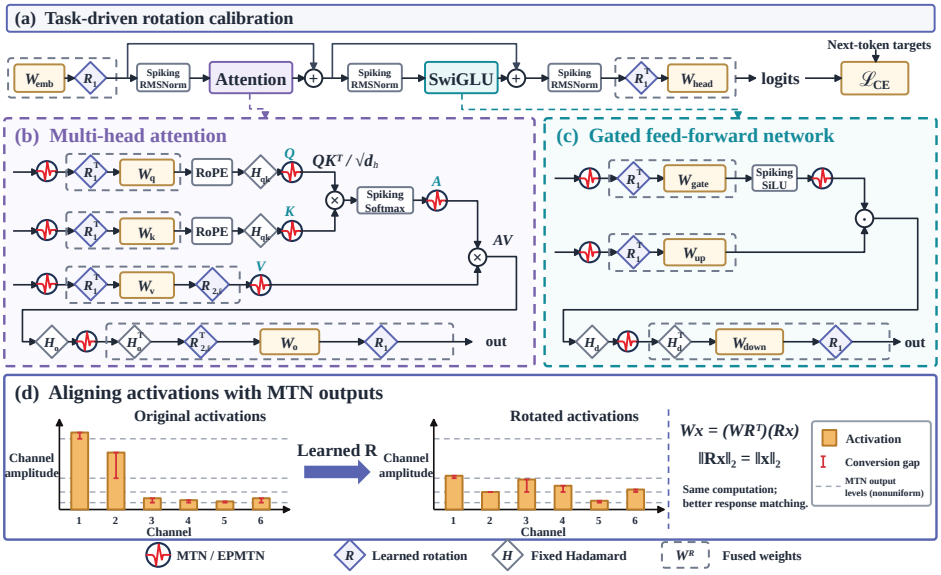

Figure 2: **Rotation-based conversion framework.** (a) Overall network architecture. (b) Attention block. (c) GLU block. (d) Illustration of the effect of rotation.

during the T steps. Vector activations follow the same rule coordinatewise within the designated threshold-sharing groups. Linear transformations preserve this temporal correspondence:

$$\frac{\theta}{T}\sum_{t=1}^{T} W\xi[t] = W r_T^{\mathrm{MTN}}(x;\theta). \tag{4}$$

where W is the synaptic weight matrix of the linear transformation. Thus the same synaptic weights act on the emitted signals and their average response. Nonlinear Transformer operations require a treatment of the accumulated activation. Expectation compensation and differential graded units supply temporal constructions for such operations (Huang et al., 2024; 2025).

4 Task-Driven Rotation Calibration

Our channel-wise analysis reveals pronounced conversion-error peaks in channels containing activation outliers (Figure 1), suggesting a mismatch between pretrained activation distributions and the nonuniform output levels of MTNs at small timestep budgets. Motivated by this observation, we develop a function-preserving rotation framework and optimize the rotations through the converted model's next-token loss, using EPMTN to compute MTN outputs and support calibration without temporal unrolling. Appendix D provides the error definition and additional channel-wise analyses.

4.1 Orthogonal rotations for MTN conversion

Orthogonal rotations redistribute activations across channels while preserving the pretrained ANN function through corresponding weight transformations. For a linear layer $y = Wx$, an orthogonal matrix R gives

$$z = Rx, \qquad \widetilde{W} = WR^{\top}, \qquad R^{\top}R = I, \tag{5}$$

so that $\widetilde{W}z = Wx$. After MTN conversion, the linear output becomes

$$\widehat{y}_R = WR^{\top} r_T(Rx), \tag{6}$$

where $\widehat{y}_R$ is the converted linear output and r_T denotes the time-averaged MTN output over T timesteps, with the threshold argument omitted for brevity. Because MTNs operate on individual activation components, different rotations yield different conversion errors despite preserving the same ANN computation. We introduce a shared residual-stream rotation R_1 and a per-layer value

rotation $R_{2,\ell}$, as illustrated in Figure 2. The shared rotation represents each hidden state as $\widetilde{h} = R_1 h$, where h and $\widetilde{h}$ are the original and rotated hidden-state vectors, respectively. After absorbing RMSNorm gains into adjacent projection weights, normalization commutes with the orthogonal rotation. We apply the corresponding weight transformations to the projections, token embedding, and language-model head:

$$\begin{aligned} \widetilde{W}_p &= W_p R_1^\top, \qquad p \in \{q, k, \text{gate}, \text{up}, \text{head}\}, \\ \widetilde{W}_{\text{down}} &= R_1 W_{\text{down}}, \quad \widetilde{W}_{\text{emb}} = R_1 W_{\text{emb}}. \end{aligned} \tag{7}$$

where W_{emb} is the token embedding matrix and W_{head} maps hidden states to vocabulary logits. Sharing R_1 across Transformer blocks preserves identity skip connections and allows these transformations to be absorbed into the model weights. Within attention layer ℓ, we apply $R_{2,\ell}$ along the value head dimension and share it across heads. This rotation commutes with attention aggregation:

$$(R_{2,\ell} V) A^\top = R_{2,\ell}(V A^\top), \tag{8}$$

where V contains the value vectors and A is the attention matrix. Let $B_{v,\ell}$ and $B_{o,\ell}$ denote block-diagonal repetitions of $R_{2,\ell}$ over the value and output heads, respectively. The corresponding weight transformations are

$$\widetilde{W}_{v,\ell} = B_{v,\ell} W_{v,\ell} R_1^\top, \qquad \widetilde{W}_{o,\ell} = R_1 W_{o,\ell} B_{o,\ell}^\top. \tag{9}$$

where $W_{v,\ell}$ and $W_{o,\ell}$ are the value and attention-output projection weights in layer ℓ. Together, these rotations adjust the activation distributions along the residual stream and within the value and attention-output pathways. We additionally use fixed Hadamard transforms on Q/K after RoPE and before the O and down projections. Applying the same orthogonal transform to Q and K preserves attention scores. The inverse transforms before the O and down projections are absorbed into their weights. The activation-side transforms are computed online using the Hadamard structure, which requires additions and subtractions apart from normalization. MTNs are placed at the activation interfaces shown in Figure 2, and the learned rotations are folded into the pretrained weights after calibration.

However, fixed rotations do not explicitly account for the nonuniform output levels of MTNs at the target timestep budget. We therefore optimize the rotations through the converted model's next-token loss. To see why a task-level objective matters, consider the activation conversion error and its effect on the subsequent linear output:

$$\begin{aligned} e_R(x) &= r_T(Rx) - Rx, \qquad \widehat{y}_R - y = W R^\top e_R(x), \\ \|\widehat{y}_R - y\|_2^2 &= e_R(x)^\top R W^\top W R^\top e_R(x). \end{aligned} \tag{10}$$

where $e_R(x)$ is the activation conversion error in the rotated coordinates and $\|\cdot\|_2$ denotes the Euclidean norm. The resulting output error depends on both the magnitude of the activation conversion error and its direction relative to the projection weights. Minimizing activation conversion error alone therefore does not fully account for its effect on model predictions. Calibrating rotations through the next-token loss incorporates error propagation through the converted network.

4.2 EPMTN FOR EFFICIENT AND ACCURATE ROTATION CALIBRATION

Learning rotation matrices through the converted model's task loss requires backpropagating through MTN computations. Temporal simulation maintains membrane states across T timesteps and propagates surrogate gradients through repeated firing and reset operations, increasing calibration cost. We introduce EPMTN to compute the time-averaged MTN outputs directly and construct their surrogate gradients without temporal unrolling. Under the input conversion setting in Section 3, EPMTN preserves the MTN outputs at the prescribed timestep budget and removes recurrent membrane updates during calibration.

For the threshold ratios $\lambda_i = 2^{-i}$, $i = 0, \ldots, n-1$, let $\lambda_{\min} = 2^{-(n-1)}$ and $q = |x|/\theta$. We define

$$\begin{aligned} &c_0 = 0, && \Delta_0 = \lambda_{\min}/T, \\ &c_k = 2^{k-1}\lambda_{\min}, && \Delta_k = c_k/T \quad (1 \le k < n), \end{aligned} \qquad \beta_{k,j} = c_k + j\Delta_k. \tag{11}$$

where $k = 0, \ldots, n-1$ indexes threshold groups, c_k is the starting threshold of group k, Δ_k is its threshold spacing and output increment, and $\beta_{k,j}$ is its jth comparison threshold for $j = 1, \ldots, T$.

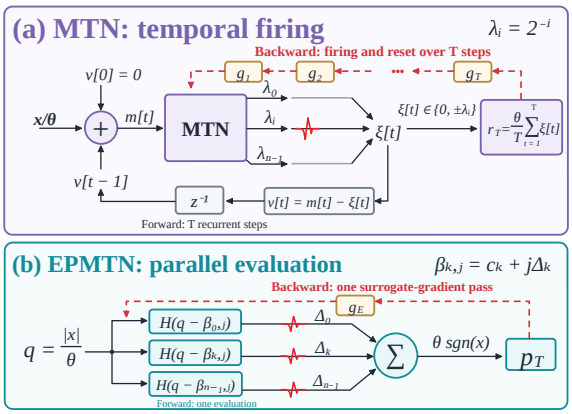

Figure 3: **Comparison of MTN and EPMTN.**

Method	Fwd.	Bwd.	δ_F	δ_G
EPMTN	Direct	Direct	0	0
MTN	Unroll	Temporal	0	0.3729
Control	Unroll	Direct	0	0

Table 1: Relative output and rotation-gradient differences with respect to EPMTN. Direct backward uses a single-step surrogate gradient.

Metric	MTN	EPMTN
Network passes	4	1
Time (min)	59.67	**17.85**
Memory (GiB)	58.26	**16.59**
PPL	6.8457	**6.21**

Table 2: LLaMA2-7B calibration at $T = 4$. Time: training only. Memory: peak memory usage.

EPMTN expresses the MTN output as a sum of parallel threshold comparisons:

$$p_T(x;\theta) = \theta\,\mathrm{sgn}(x)\sum_{k=0}^{n-1}\sum_{j=1}^{T}\Delta_k\mathcal{H}(q-\beta_{k,j}), \qquad \mathcal{H}(z) = \mathbf{1}\{z \geq 0\}. \tag{12}$$

where $p_T(x;\theta)$ is the EPMTN output, sgn is the sign function, and $\mathcal{H}$ is the Heaviside step function with $\mathcal{H}(0) = 1$. $\mathbf{1}\{\cdot\}$ equals one when its condition holds and zero otherwise. Each active term contributes an output increment determined by Δ_k, allowing the accumulated effect of T MTN timesteps to be evaluated without updating membrane states.

Proposition 1 (exact equivalence of time-averaged outputs). *Consider the MTN in Section 3.1 with input x/θ, fixed threshold $\theta > 0$, zero initial membrane potential, subtractive reset, and largest-eligible firing with inclusive threshold comparisons. For every $x \in \mathbb{R}$ and positive integers n, T,*

$$p_T(x;\theta) = r_T^{\mathrm{MTN}}(x;\theta) = \theta\,\mathrm{sgn}(x)\min\left\{1, \frac{b(q)}{T}\left\lfloor\frac{Tq}{b(q)}\right\rfloor\right\}, \tag{13}$$

where $b(q) = \max\big(\{\lambda_{\min}\} \cup \{\lambda_i : \lambda_i \leq q\}\big)$ selects the largest threshold ratio not exceeding q, with a lower bound of $\lambda_{\min}$, and $\lfloor\cdot\rfloor$ denotes the floor function. The proof is provided in Appendix A. This equivalence preserves the activation conversion error and its effect on subsequent linear operations. Consequently, replacing each input MTN with EPMTN in an otherwise identical calibration network preserves its logits and next-token loss. EPMTN therefore provides the same forward calibration objective through direct evaluation of MTN outputs, as illustrated in Figure 3.

However, identical forward outputs do not imply identical surrogate gradients. Temporal MTN calibration composes approximate firing and reset derivatives across timesteps, whereas EPMTN defines a surrogate gradient directly from its parallel expression. The logistic surrogate and its derivative are specified in Appendix A.1. Thresholds are detached during backpropagation. This formulation evaluates the gradient without propagating derivatives through recurrent membrane states. Table 1 isolates this distinction: at $T = 4$, temporal MTN and EPMTN produce identical outputs but different rotation gradients. The control uses the temporal MTN forward pass and the EPMTN surrogate for backward propagation, reproducing both the EPMTN outputs and rotation gradients. Table 2 compares the two calibration implementations at $T = 4$. EPMTN substantially reduces calibration cost. When evaluated using the same EPMTN inference implementation, perplexity decreases from 6.85 to 6.21, suggesting that EPMTN mitigates the effects of surrogate-gradient approximation errors accumulating across timesteps.

4.3 Task-driven rotation calibration

With EPMTN, we calibrate the rotation matrices through the converted model's next-token loss while keeping the pretrained parameters Θ fixed. Let $\mathcal{R} = \{R_1, R_{2,1}, \ldots, R_{2,L}\}$ denote the learnable rotations and $\mathcal{D}_{\mathrm{cal}}$ the calibration dataset. At the target timestep budget T, we optimize

$$\min_{\mathcal{R}}\ \mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{D}_{\mathrm{cal}}}\left[\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\tau};\Theta,\mathcal{R},T)\right], \qquad R^\top R = I, \quad R \in \mathcal{R}. \tag{14}$$

where $\boldsymbol{\tau}$ is a calibration sequence and $\mathcal{L}_{\mathrm{CE}}$ is the next-token cross-entropy loss. L is the number of Transformer layers. The rotations are therefore jointly calibrated for the complete converted model at the prescribed T. We initialize the rotations with orthogonal Hadamard matrices and optimize them using Cayley updates. For a rotation matrix R and its surrogate-gradient update direction G, the update is

$$\begin{aligned} \Omega &= GR^\top - RG^\top, \\ R^+ &= \left(I + \frac{\eta}{2}\Omega\right)^{-1}\left(I - \frac{\eta}{2}\Omega\right)R, \qquad \eta > 0. \end{aligned} \tag{15}$$

where Ω is the skew-symmetric update generator, η is the learning rate, and R^+ is the updated rotation matrix. Since $\Omega^\top = -\Omega$, the exact Cayley update preserves $(R^+)^\top R^+ = I$, as shown in Appendix A.2. This constraint maintains the function-preserving transformations described in Section 4.1 while allowing the activation distributions to adapt to MTN conversion. In practice, we backpropagate the next-token loss through EPMTN and update the rotations using the iterative Cayley implementation of SGDG. After calibration, we fold the selected rotations into the corresponding projection, embedding, and language-model head weights. The fixed online Hadamard transforms remain at their designated interfaces. We retain the timestep budget, firing amplitudes, and threshold policy used during calibration, so that the folded model preserves the calibrated EPMTN outputs. For inference, each EPMTN can be replaced by its equivalent MTN implementation unrolled over T timesteps, preserving its time-averaged output under the same input and neuron settings. Algorithm 1 in Appendix E summarizes the procedure.

5 Experiments

We evaluate conversion accuracy on language-modeling and zero-shot reasoning tasks, and examine the effects of rotation calibration on model performance, calibration efficiency, and activation distributions. For consistent comparison, we use EPMTN for calibration and its equivalent MTN for evaluation. Experimental settings, evaluation protocols, and baselines are provided in Appendix B.1. An operation-count-based energy estimate is provided in Appendix F.

5.1 Short-window conversion performance

Language modeling. Table 3 reports language-modeling perplexity across four corpora. At four timesteps, our method achieves average perplexities of 6.94 on LLaMA2-7B and 8.20 on LLaMA3-8B, substantially outperforming the four-step Phase Coding results reported in the table. Increasing the timestep budget consistently improves perplexity on all four corpora for both models, with eight-step performance approaching the FP16 baselines. These results demonstrate that calibrated rotations support accurate conversion across different language-modeling distributions. The strongest gains over prior conversion methods occur at the low timestep budgets of $T = 4$ and $T = 6$, where our method achieves state-of-the-art average perplexity on both model families. At $T = 8$, the larger temporal budget allows competing methods to approximate ANN activations more accurately, narrowing the performance gap.

Zero-shot reasoning. Table 4 compares our method with Phase Coding at the same timestep budgets. At $T = 4$, our method achieves average accuracies of 63.43% on LLaMA2-7B and 69.67% on LLaMA3-8B, substantially exceeding the corresponding Phase Coding results of 41.40% and 40.80%. As the timestep budget increases, the gap narrows. At $T = 8$, our method achieves average accuracy comparable to the FP16 ANN baselines. This comparison highlights the advantage of our method at four timesteps while showing competitive performance at larger timestep budgets. The eight-step results are broadly comparable to those of existing methods. The main benefit of our framework is therefore to retain strong predictive performance with fewer timesteps, rather than to require a larger temporal budget to approach ANN accuracy.

5.2 Effect of rotation calibration

Figure 4(a) compares neuron types and rotation settings using the same WikiText-2 evaluation protocol. At $T = 16$, signed IF neurons without rotation yield a perplexity of 58,815.89. Fixed and learned rotations reduce it to 16.35 and 8.55, respectively. This demonstrates the effectiveness of our

Model	Method	T	WikiText-2	C4	RedPajama	Pile	Avg ↓
LLaMA2-7B	FP16	N/A	5.47	6.97	5.61	4.63	5.67
	TTFSFormer	128	11.88	16.47	13.18	9.32	12.71
	SpikeLLM	8	5.86	7.51	6.08	4.97	6.10
	LAS	8	34.26	40.39	32.18	20.10	31.73
	SpikedAttention	8	19.02	25.05	20.77	14.80	19.91
	Phase Coding (G=2)	8	6.71	8.96	7.23	5.74	7.16
	Ours	8	5.50	7.17	5.79	4.79	5.81
	Phase Coding (G=2)	6	24.45	27.51	25.32	15.19	23.12
	Ours	6	5.60	7.39	6.00	4.95	5.99
	Phase Coding (G=2)	4	> 100	> 100	> 100	> 100	> 100
	Ours	4	6.21	8.68	7.18	5.70	6.94
LLaMA3-8B	FP16	N/A	6.14	8.88	7.44	5.52	7.00
	TTFSFormer	128	6.72	9.82	8.12	6.03	7.67
	SpikeLLM	8	> 100	> 100	> 100	> 100	> 100
	LAS	8	7.06	10.49	8.57	6.51	8.16
	SpikedAttention	8	9.46	14.02	12.08	8.17	10.93
	Phase Coding (G=2)	8	6.32	9.14	7.69	5.74	7.22
	Ours	8	6.34	9.25	7.83	5.59	7.25
	LAS	6	93.13	> 100	> 100	> 100	> 100
	SpikedAttention	6	> 100	> 100	> 100	83.95	> 100
	Phase Coding (G=2)	6	8.04	12.25	10.14	7.23	9.42
	Ours	6	6.51	9.57	8.12	5.75	7.49
	Phase Coding (G=2)	4	> 100	> 100	> 100	> 100	> 100
	Ours	4	6.99	10.57	9.02	6.23	8.20

Table 3: Language-modeling perplexity (↓) across two model families.

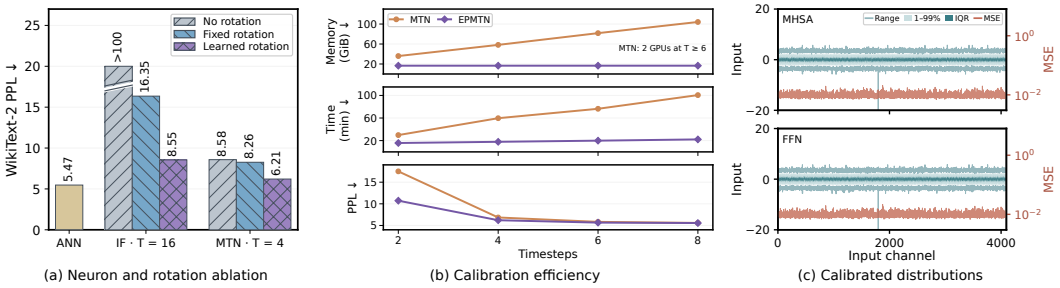

Figure 4: Ablation studies and efficiency analysis on LLaMA2-7B. (a) WikiText-2 perplexity for the FP16 ANN, signed IF neurons at $T = 16$, and MTNs at $T = 4$, with no rotation, fixed rotation, or learned rotation. (b) Peak calibration memory, calibration time, and WikiText-2 perplexity (top to bottom) for MTN and EPMTN at $T = 2, 4, 6, 8$. (c) Input distributions and per-channel conversion MSE after calibrated rotation, using the same configuration as Figure 1.

learnable rotation framework. At $T = 4$, MTNs achieve perplexities of 8.58 without rotation, 8.26 with fixed rotation, and 6.21 with learned rotation. Learning the rotations reduces four-step MTN perplexity by 24.8% relative to fixed rotation. The limited improvement from fixed rotations indicates that suppressing outliers alone is insufficient. Adapting the rotations to MTN outputs through task-driven calibration provides an additional accuracy benefit. The final four-step MTN model also outperforms the calibrated 16-step IF model, demonstrating the combined benefit of multi-threshold neurons and task-driven rotation calibration.

Model	Method	T	Winogrande	ARC-c	ARC-e	PiQA	Avg ↑
LLaMA2-7B	FP16	N/A	69.06	46.33	74.54	79.05	67.25
	Phase Coding (G=2)	8	70.56	46.16	73.99	77.97	67.17
	Ours	8	68.43	44.80	74.24	78.56	66.51
	Phase Coding (G=2)	6	66.54	44.97	72.77	77.53	65.45
	Ours	6	68.51	43.69	72.64	77.86	65.67
	Phase Coding (G=2)	4	50.67	23.55	34.05	57.34	41.40
	Ours	4	65.11	41.81	69.07	77.75	63.43
LLaMA3-8B	FP16	N/A	72.85	53.33	77.74	80.85	71.19
	Phase Coding (G=2)	8	72.69	54.35	78.11	80.14	71.32
	Ours	8	74.35	53.58	79.76	80.30	72.00
	Phase Coding (G=2)	6	73.16	47.87	73.74	77.64	68.10
	Ours	6	72.77	52.39	78.87	79.33	70.84
	Phase Coding (G=2)	4	51.46	24.83	32.32	54.57	40.80
	Ours	4	70.88	51.28	77.31	79.22	69.67

Table 4: Zero-shot accuracy across two model families. Complete comparisons appear in Appendix C. ARC-c, ARC-e, and PiQA use normalized accuracy. Winogrande uses accuracy.

5.3 Calibration efficiency

Figure 4(b) shows that MTN calibration requires increasing memory and time as the timestep budget grows. In contrast, EPMTN maintains nearly constant memory usage and exhibits only a modest increase in calibration time. At $T = 4$, EPMTN reduces calibration time from 59.67 to 17.85 minutes and peak memory from 58.26 to 16.59 GiB, while improving perplexity from 6.85 to 6.21. EPMTN also achieves lower perplexity across all four timestep budgets, with the largest improvement at $T = 2$. Together with the gradient comparison in Table 1, these results support the effectiveness of its directly constructed surrogate gradients in mitigating approximation errors accumulated across timesteps, enabling resource-efficient and accurate rotation calibration.

5.4 Activation distributions after calibration

Figure 4(c) applies the calibrated residual-stream rotation to the same ANN activations used in Figure 1, isolating the effect of the change in coordinates. The largest per-channel conversion MSE decreases from 1.45 to 0.021 in attention inputs and from 1.77 to 0.022 in FFN inputs. Thus, the rotation redistributes concentrated conversion errors across channels. The perplexity ablation in Figure 4(a) measures the resulting benefit to model predictions.

6 Conclusion

We present a rotation-based ANN-to-SNN conversion framework for accurate few-timestep conversion of LLMs. Task-driven rotation calibration reduces the impact of activation outliers and aligns activation distributions with the nonuniform output levels of MTNs. To enable efficient and accurate calibration, EPMTN computes equivalent time-averaged MTN outputs with a single forward pass and uses a single backward pass, reducing calibration overhead and mitigating the accumulation of surrogate-gradient approximation errors across timesteps. Experiments on LLaMA2-7B and LLaMA3-8B demonstrate state-of-the-art conversion performance with only four timesteps across language modeling and zero-shot reasoning tasks. These results highlight the value of jointly optimizing activation distributions and neuronal gradient approximations, providing a practical path toward high-accuracy, low-timestep conversion of LLMs.

REPRODUCIBILITY

The experiments can be reproduced using the provided code and the complete experimental settings documented in the appendix.

AI USE STATEMENT

AI tools assisted with manuscript writing and language editing. The authors take full responsibility for the manuscript and its results.

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.

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, 2022a.

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*, 2022b. URL https://openreview.net/forum?id=7B3IJMM1k_M.

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, 2026.

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.

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, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 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, 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. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=OxBWTFSGcv.

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.

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.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

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

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

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.

Christoph Stöckl and Wolfgang Maass. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 3(3):230–238, 2021.

Jingya Wang, Xin Deng, Wenjie Wei, Dehao Zhang, Shuai Wang, Qian Sun, Jieyuan Zhang, Hanwen Liu, Ning Xie, and Malu Zhang. Training-free ann-to-snn conversion for high-performance spiking transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2128–2136, 2026.

Maurice Weber, Daniel Y Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37: 116462–116492, 2024.

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

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 Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57367–57383. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/you24b.html.

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

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=frE4fUwz_h.

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

A TIME-AVERAGED MTN OUTPUTS AND SURROGATE GRADIENTS

Proof. By sign symmetry, it suffices to consider $u = x/\theta \geq 0$. Let $S_t = \sum_{s=1}^{t} \xi[s]$ be the cumulative emitted amplitude. Summing the membrane updates gives

$$v[t] = tu - S_t. \tag{16}$$

For $\lambda_i \leq u < 2\lambda_i$, $i \geq 1$, induction gives $0 \leq v[t] < \lambda_i$: if $v[t-1] \in [0, \lambda_i)$, then $m[t] = v[t-1] + u \in [\lambda_i, 3\lambda_i)$, so the neuron emits either λ_i or $2\lambda_i$, leaving the residual in the same interval. Since S_T is an integer multiple of λ_i, Equation 16 uniquely determines

$$0 \leq Tu - S_T < \lambda_i \quad \Longrightarrow \quad S_T = \lambda_i \left\lfloor \frac{Tu}{\lambda_i} \right\rfloor. \tag{17}$$

For $0 \leq u < \lambda_{\min}$, the same argument applies with emissions 0 and $\lambda_{\min}$, yielding Equation 17 with $\lambda_i = \lambda_{\min}$. For $u \geq 1$, every step emits 1, giving $S_T = T$. Thus the temporal average equals the closed form in Equation 13.

It remains to verify the parallel expression. Below $\lambda_{\min}$, only group $k = 0$ can fire, producing $(\lambda_{\min}/T)\lfloor Tu/\lambda_{\min} \rfloor$. For $c_k \leq u < 2c_k$, $k \geq 1$, all preceding groups contribute $\sum_{k'=0}^{k-1} T\Delta_{k'} = c_k$, while the current group contributes $(c_k/T)\lfloor T(u-c_k)/c_k \rfloor$. Their sum is

$$c_k + \frac{c_k}{T}\left\lfloor \frac{T(u-c_k)}{c_k} \right\rfloor = \frac{c_k}{T}\left\lfloor \frac{Tu}{c_k} \right\rfloor, \tag{18}$$

matching the temporal response. For $u \geq 1$, all events fire and their increments sum to 1. Inclusive threshold comparisons establish the same equality at interval boundaries. Restoring the sign and scale θ completes the proof. □

The residual invariant established in Proposition 1 gives a response-error bound: for $|x| < \theta$,

$$|x - r_T(x;\theta)| < \frac{\theta b(|x|/\theta)}{T}. \tag{19}$$

Proposition 1 establishes exact equivalence of time-averaged outputs and preservation of conversion errors in Section 4.2. These statements concern exact arithmetic.

A.1 SURROGATE-GRADIENT CONTROL

Using logistic surrogates with widths $\varepsilon_k = \rho\Delta_k$, we obtain

$$g_{\mathrm{E}}(x) = \sum_{k=0}^{n-1}\sum_{j=1}^{T} \frac{\Delta_k}{\varepsilon_k}\,\sigma'\!\left(\frac{q-\beta_{k,j}}{\varepsilon_k}\right), \qquad \rho = 0.25, \tag{20}$$

where $g_{\mathrm{E}}(x)$ is the surrogate derivative of the EPMTN output with respect to x, ε_k is the smoothing width for group k, ρ sets its scale relative to Δ_k, and $\sigma(z) = (1+e^{-z})^{-1}$ is the logistic sigmoid with derivative $\sigma'(z) = \sigma(z)(1-\sigma(z))$.

The equality of forward outputs leaves the surrogate backward rule unspecified. Let g_t be the surrogate derivative of $\xi[t]$ with respect to $m[t]$, and let $D_t = \widetilde{\partial} v[t]/\partial u$, with the base threshold held fixed. Differentiating through both the membrane and subtractive reset gives

$$D_0 = 0, \qquad D_t = (1-g_t)(D_{t-1}+1), \qquad \frac{\widetilde{\partial} r_T^{\mathrm{MTN}}}{\partial x} = 1 - \frac{D_T}{T}. \tag{21}$$

The tilde denotes differentiation using the prescribed surrogate. The resulting gradient depends on the composition of firing and reset derivatives across time. Expanding Equation 21 gives

$$D_T = \sum_{k=1}^{T}\prod_{j=k}^{T}(1-g_j), \qquad \frac{\widetilde{\partial} r_T^{\mathrm{MTN}}}{\partial x} = 1 - \frac{1}{T}\sum_{k=1}^{T}\prod_{j=k}^{T}(1-g_j). \tag{22}$$

This composition generally differs from g_{E} in Equation 20, although both implementations return the same hard response. For example, at $T = 2$ the temporal derivative is $[g_1 + g_2(2-g_1)]/2$. The

difference concerns the assigned surrogate derivatives, rather than the true derivatives of two equal functions.

We isolate this distinction in the matched $T = 4$ CE100 study with a control that retains the temporal MTN forward pass but assigns the complete-response surrogate g_{E} to its backward pass. At the recorded pre-training probe, direct and temporal implementations have zero relative forward difference, while their rotation gradients have relative difference 0.3729 and cosine similarity 0.9282. Replacing only the temporal backward rule with g_{E} reduces the gradient difference to zero, with cosine similarity 1.

A.2 Orthogonality of the Cayley update

Proof of orthogonality preservation. By construction, $\Omega^\top = -\Omega$. Set $\alpha = \eta/2$, $C_+ = I + \alpha\Omega$, and $C_- = I - \alpha\Omega = C_+^\top$. For any nonzero real test vector d, $d^\top C_+ d = \|d\|_2^2 > 0$, so C_+ is invertible. The matrices C_+ and C_- commute, as do their inverses. Consequently, the Cayley transform matrix $C_{\mathrm{cay}} = C_+^{-1} C_-$ satisfies

$$C_{\mathrm{cay}}^\top C_{\mathrm{cay}} = C_+ C_-^{-1} C_+^{-1} C_- = I, \qquad (R^+)^\top R^+ = R^\top C_{\mathrm{cay}}^\top C_{\mathrm{cay}} R = I. \tag{23}$$

This proves the claim. □

The argument applies to any skew-symmetric generator, including one formed from a momentum-adjusted surrogate-gradient direction. Our implementation uses SGDG's iterative Cayley solver with numerical orthogonality checks and QR correction. The proof establishes the exact-arithmetic property of the Cayley transform. The iterative solver and correction implement this constraint numerically.

B Calibration Settings and Evaluation Provenance

B.1 Experimental setup

Models and calibration. We evaluate our framework on LLaMA2-7B and LLaMA3-8B at timestep budgets of $T \in \{4, 6, 8\}$. The pretrained weights remain frozen in FP16, and MTNs use eight firing amplitudes per sign. Conversion covers linear-layer inputs, key and value outputs, post-RoPE queries, attention probabilities, and SiLU outputs. We optimize the residual-stream and value rotations using next-token cross-entropy for 200 updates. Calibration uses 128 WikiText-2 (Merity et al., 2016) sequences of 2,048 tokens, with 16 additional sequences held out for checkpoint selection. The global batch size is four, and the learning rate starts at 0.1 and follows a cosine decay schedule.

Evaluation and baselines. We evaluate language modeling using perplexity on WikiText-2 (Merity et al., 2016), C4 (Raffel et al., 2020), RedPajama (Weber et al., 2024), and Pile (Gao et al., 2020), and zero-shot reasoning using accuracy on ARC-Challenge, ARC-Easy (Clark et al., 2018), Winogrande (Sakaguchi et al., 2021), and PiQA (Bisk et al., 2020). ARC-Challenge, ARC-Easy, and PiQA use normalized accuracy, whereas Winogrande uses standard accuracy. C4, RedPajama, and Pile are excluded from calibration. We compare against FP16 models and representative conversion methods, including TTFSFormer, SpikeLLM, LAS, SpikedAttention, and Phase Coding.

C Extended Conversion Comparisons

Model	Method	T	Winogrande	ARC-c	ARC-e	PiQA	Avg ↑
LLaMA2-7B	FP16	N/A	69.06	46.33	74.54	79.05	67.25
	TTFSFormer	128	70.56	44.88	73.11	78.94	66.87
	SpikeLLM	8	67.40	42.58	71.46	77.75	64.80
	LAS	8	69.46	45.56	73.65	77.97	66.66
	SpikedAttention	8	68.19	41.38	68.77	77.20	63.89
	Phase Coding (G=2)	8	70.56	46.16	73.99	77.97	67.17
	Ours	8	68.43	44.80	74.24	78.56	66.51
	Phase Coding (G=2)	6	66.54	44.97	72.77	77.53	65.45
	Ours	6	68.51	43.69	72.64	77.86	65.67
	Phase Coding (G=2)	4	50.67	23.55	34.05	57.34	41.40
	Ours	4	65.11	41.81	69.07	77.75	63.43
LLaMA3-8B	FP16	N/A	72.85	53.33	77.74	80.85	71.19
	TTFSFormer	128	72.69	52.90	77.65	79.33	70.64
	SpikeLLM	8	69.38	49.23	73.11	78.67	67.60
	LAS	8	74.82	54.52	77.48	80.74	71.89
	SpikedAttention	8	69.53	49.91	74.33	77.20	67.74
	Phase Coding (G=2)	8	72.69	54.35	78.11	80.14	71.32
	Ours	8	74.35	53.58	79.76	80.30	72.00
	LAS	6	71.19	51.88	73.48	79.33	68.97
	SpikedAttention	6	63.30	32.25	54.00	66.21	53.94
	Phase Coding (G=2)	6	73.16	47.87	73.74	77.64	68.10
	Ours	6	72.77	52.39	78.87	79.33	70.84
	Phase Coding (G=2)	4	51.46	24.83	32.32	54.57	40.80
	Ours	4	70.88	51.28	77.31	79.22	69.67

Table 5: Zero-shot accuracy in percent (↑) across two model families. Gray rows denote our full-scope conversion. ARC-c, ARC-e, and PiQA use normalized accuracy. Winogrande uses accuracy.

D CHANNEL-WISE ACTIVATION DISTRIBUTIONS AND MTN CONVERSION ERROR

We extend the layer-15 diagnostic in Figure 1 to nine layers spanning LLaMA2-7B: 0, 4, 8, 12, 15, 20, 24, 28, and 31 (zero-based). All panels use the same 8,192 WikiText-2 training tokens, arranged as four 2,048-token sequences. Inputs are collected from the ANN with RMSNorm gains absorbed into the adjacent projection weights, without rotation. Each input is then evaluated with the explicit input MTN recurrence at $T = 4$, eight firing levels per sign, and the per-token threshold $\theta = 0.9 \max_j |x_j|$. The orange curves report the channel-wise conversion error defined below. Channels remain in their original order, with no smoothing. Only the activation display is clipped to $[-20, 20]$. Errors use the complete inputs. The logarithmic MSE axes are scaled per panel.

We measure channel-wise conversion error before the linear projection as the mean squared difference between ANN activations and time-averaged MTN outputs:

$$\mathcal{E}_{T,j} = \mathbb{E}_x \left[\left(r_T(x_j; \theta(x)) - x_j \right)^2 \right]. \tag{24}$$

where $\mathcal{E}_{T,j}$ is the mean squared conversion error for channel j at timestep budget T, x_j is the input activation in that channel, r_T denotes the time-averaged MTN output over T timesteps, and $\theta(x)$ is the threshold for input x. The expectation is estimated over the sampled tokens. This metric measures activation conversion error without weighting by the subsequent projection matrix.

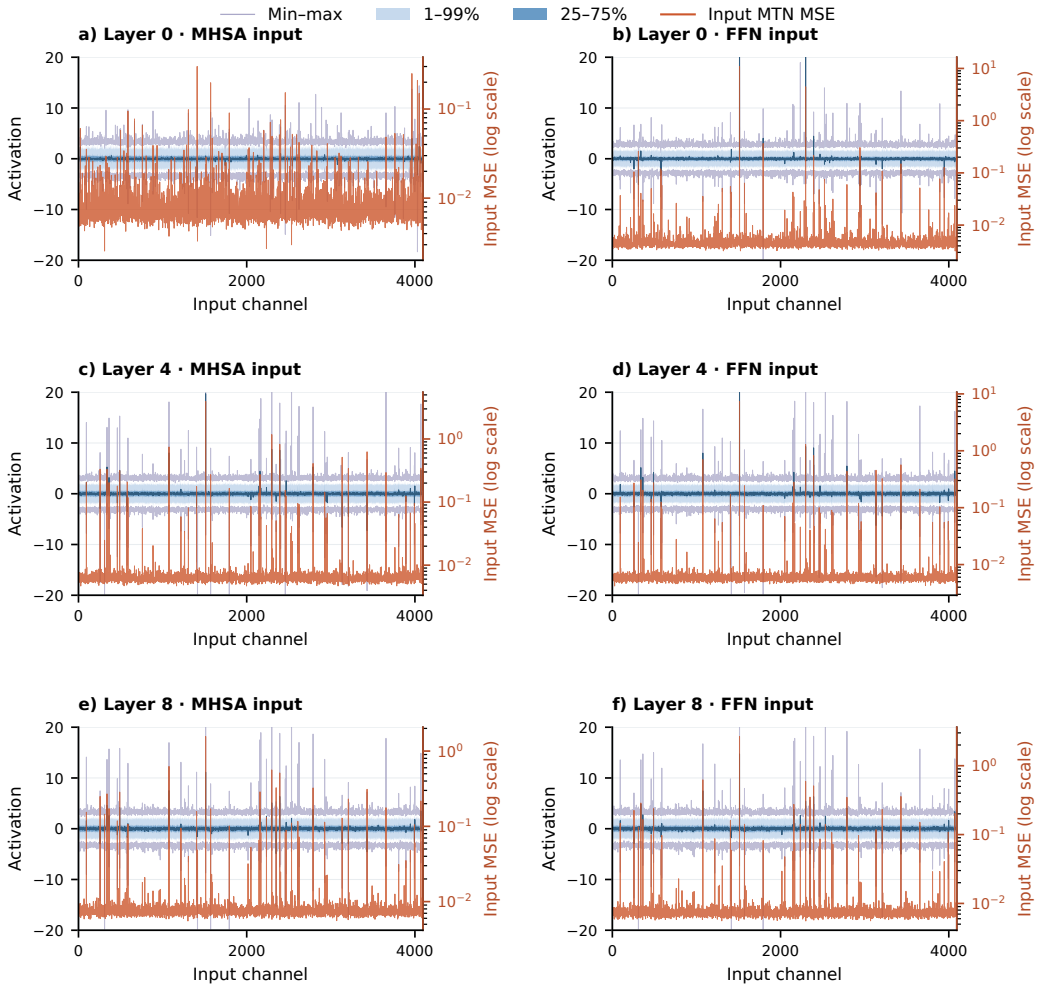

Figure 5: **Early layers: 0, 4, and 8.** Left: MHSA inputs. Right: FFN inputs. Blue bands show the 25–75% and 1–99% activation intervals. Light purple lines show extrema. Orange curves show input conversion MSE after MTN conversion.

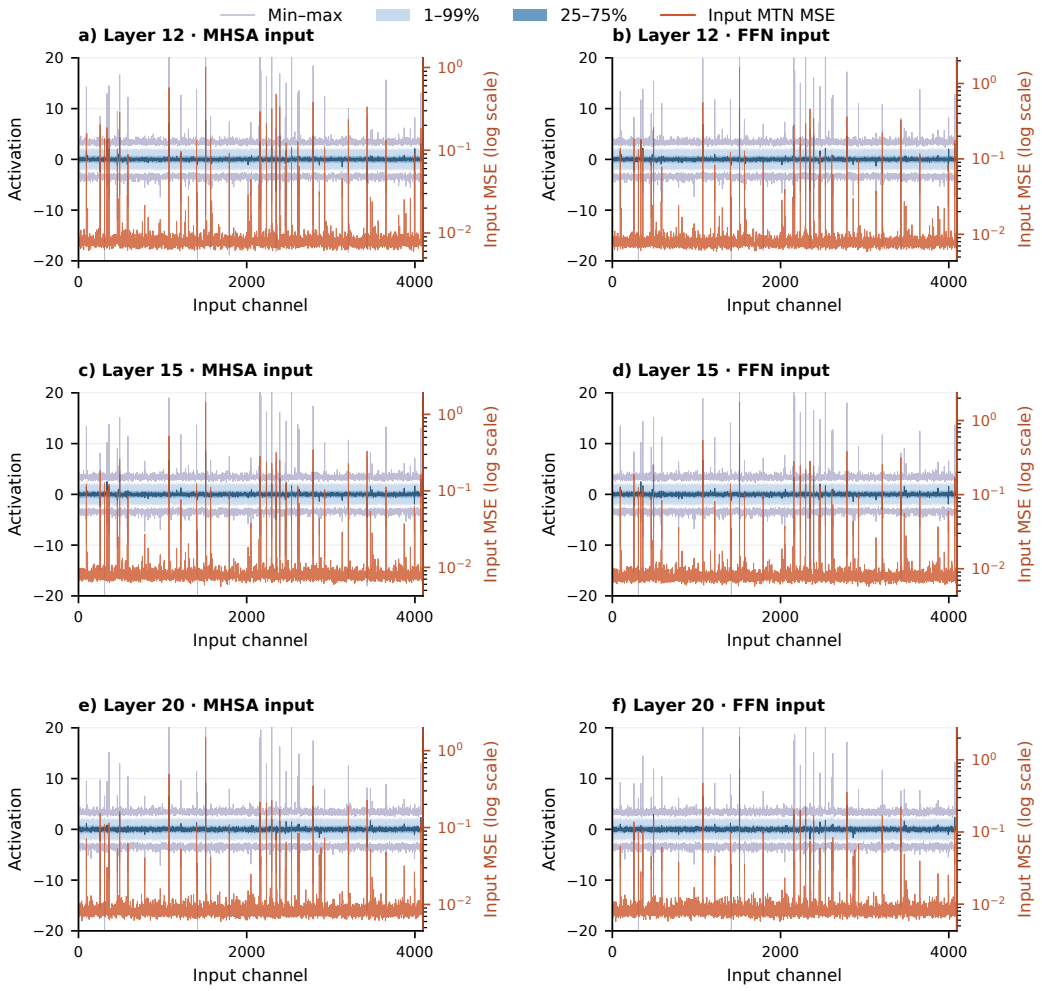

Figure 6: **Middle layers: 12, 15, and 20.** Signed activation distributions and per-channel MTN input conversion MSE, using the same tokens and settings as Figure 5.

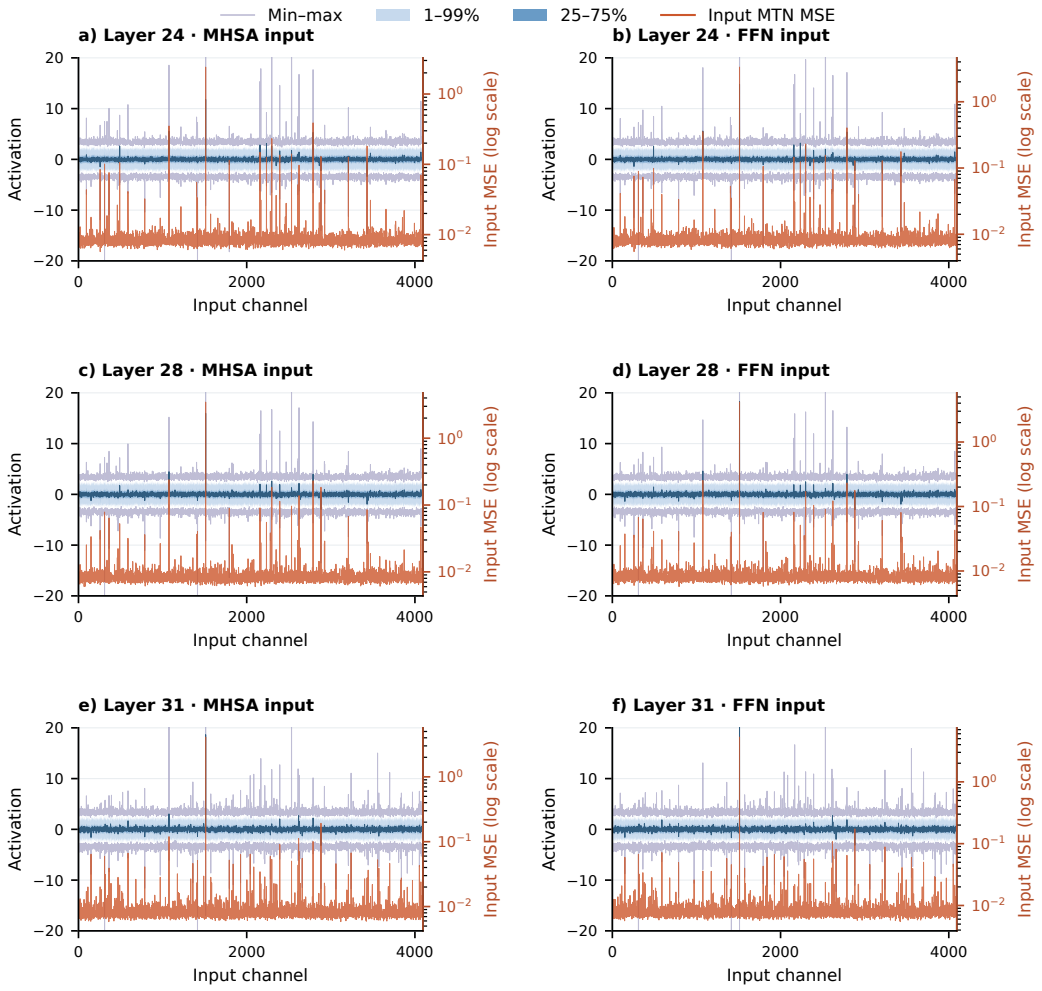

Figure 7: **Late layers: 24, 28, and 31.** Signed activation distributions and per-channel MTN input conversion MSE, using the same tokens and settings as Figure 5.

Method	PPL ↓	MAC (G)	AC (G)	Relative energy ↓
ANN	**5.67**	3451.54	1.12	100.00%
Ours $T = 2$	14.52	175.44	6532.41	**42.11%**
Ours $T = 4$	6.94	106.72	13248.87	78.19%

Table 6: Perplexity, computational cost, and relative energy consumption. MAC and AC denote multiply–accumulate and accumulation operations, respectively. G denotes billions of operations.

E Rotation Calibration Algorithm

Algorithm 1: Task-driven rotation calibration

Input: Frozen Θ, datasets $\mathcal{D}_{\text{cal}}, \mathcal{D}_{\text{val}}$, MTN settings (T, n, θ), N_{cal}, η.

1 Fuse RMSNorm gains and initialize rotations $\mathcal{R}^{(0)}$ with Hadamard matrices.
2 **for** each calibration step $\kappa = 1, \ldots, N_{\text{cal}}$ **do**
3 Sample tokens $\mathcal{B}_\kappa \sim \mathcal{D}_{\text{cal}}$ and compute EPMTN next-token loss $\mathcal{L}_\kappa$.
4 Compute surrogate gradients: $G_R \leftarrow \widetilde{\nabla}_R \mathcal{L}_\kappa$ for each rotation R.
5 Form the skew-symmetric matrix: $\Omega_R \leftarrow G_R R^\top - R G_R^\top$.
6 Apply the Cayley update: $R^{(\kappa)} \leftarrow (I + \frac{\eta}{2}\Omega_R)^{-1}(I - \frac{\eta}{2}\Omega_R)R$.
7 **end for**
8 Select $\kappa^\star$ with the lowest validation loss on $\mathcal{D}_{\text{val}}$.
9 Fold $\mathcal{R}^{(\kappa^\star)}$ into $\widetilde{\Theta}$ and retain the fixed online Hadamard transforms.

Output: Converted model $f_{\widetilde{\Theta},T}$ with the calibrated MTN settings.

where $\mathcal{D}_{\text{val}}$ is the held-out validation dataset, N_{cal} is the number of calibration updates, and $\mathcal{R}^{(\kappa)}$ is the set of rotations after update κ. The sampled batch $\mathcal{B}_\kappa$ defines loss $\mathcal{L}_\kappa$. $\widetilde{\nabla}_R$ denotes differentiation using the EPMTN surrogate, yielding direction G_R and generator Ω_R for rotation R. The selected index $\kappa^\star$ identifies the best validation checkpoint, $\widetilde{\Theta}$ denotes the weights after folding its rotations, and $f_{\widetilde{\Theta},T}$ is the resulting converted model.

F Computational Cost and Energy Estimation

Following the operation-count-based energy model used for Phase Coding (Zheng et al., 2026), we estimate inference energy as

$$E_{\text{ops}} = N_{\text{MAC}} e_{\text{MAC}} + N_{\text{AC}} e_{\text{AC}}, \qquad E_{\text{rel}} = \frac{E_{\text{ops}}}{E_{\text{ops}}^{\text{ANN}}} \times 100\%. \tag{25}$$

Here, N_{MAC} and N_{AC} are the numbers of multiply–accumulate and accumulation operations, respectively. We use per-operation costs $e_{\text{MAC}} = 4.6$ pJ and $e_{\text{AC}} = 0.9$ pJ based on 45 nm CMOS estimates. $E_{\text{ops}}^{\text{ANN}}$ is the corresponding ANN estimate, and E_{rel} is the relative energy consumption. This estimate covers MAC and AC operations and excludes memory-access and other hardware overheads.

Table 6 reports the operation counts and relative energy consumption. At $T = 4$, our method achieves a perplexity of 6.94 with 78.19% of the ANN's estimated operation energy. At $T = 2$, the relative energy decreases to 42.11%, illustrating the trade-off between conversion accuracy and energy consumption.