

Rethinking the Tradeoff Between Temporal Encoding and Nonlinear Computation in Spiking Language Models

Hanfei Liu^{1,2} Shuchang Feng^{1,2} Yanxia Chen^{1,2}
Changzeng Fu^{1,2} Shiqi Zhao^{1,2*}

¹Northeastern University, 110819, Shenyang, China

²Hebei Key Laboratory of Marine Perception Network
and Data Processing, 066004, Qinhuangdao, China

liuhf@mails.neu.edu.cn fengshuchang@mails.neu.edu.cn
chenyanxia@mails.neu.edu.cn fuchangzeng@qhd.neu.edu.cn
zhaoshiqi@qhd.neu.edu.cn

Abstract

Spiking language models face a tradeoff between representing continuous semantic features over short temporal windows and retaining costly nonlinear attention operations. We introduce Spora, which jointly designs spike encodings and attention operators. Binary temporal weights let T spikes represent compositional values with up to T bits of capacity, compared with $O(\log_2 T)$ bits for spike-count readout. Unipolar Binary Spiking (UBS) uses thresholds and spike-triggered residual decay to produce non-negative integer codes; Bipolar Binary Spiking (BBS) separates sign and magnitude and learns a scale for signed activations. These representations support accumulation-and-shift dot products and integer-exponent mappings in attention. With four time steps, Spora achieves 76.6 average GLUE score and 44.1 CoLA MCC, improving over SpikeLM by 1.2 and 6.2 points, respectively. Extending BBS to six steps raises these scores to 78.2 and 47.4. Conditional-decay analysis, matched-budget activation-quantization comparisons, event-workload statistics, and fixed-point evaluation further characterize the connection between encoding fidelity and computational cost.

Keywords: Spiking neural networks ; Language model ; Temporal encoding ; Approximate attention

1 Introduction

Spiking Neural Networks (SNNs) process information through discrete events and membrane-potential dynamics inspired by biological neurons (Mehonic and Kenyon, 2022; Zhang et al., 2020; Maass, 1997). Their sparse communication and event-driven computation offer a route to energy-efficient neural processing (Li et al., 2021). Recent advances in spiking vision (Wu et al., 2022; Hu et al., 2023), particularly Transformer-based architectures (Vaswani et al., 2017; Zhou et al., 2023a,b; Li et al., 2024; Yao et al., 2023), have achieved competitive performance with four or fewer time steps.

Motivated by the generalization capabilities of large-scale language models (Du et al., 2022), SpikeLM (Xing et al., 2024) and SpikeGPT (Zhu et al., 2023) demonstrate the feasibility of spike-driven language processing. Language representations, however, are dense, signed, and context-dependent. Transformer predictions depend on both the precision of these features and their inter-

*Corresponding author.

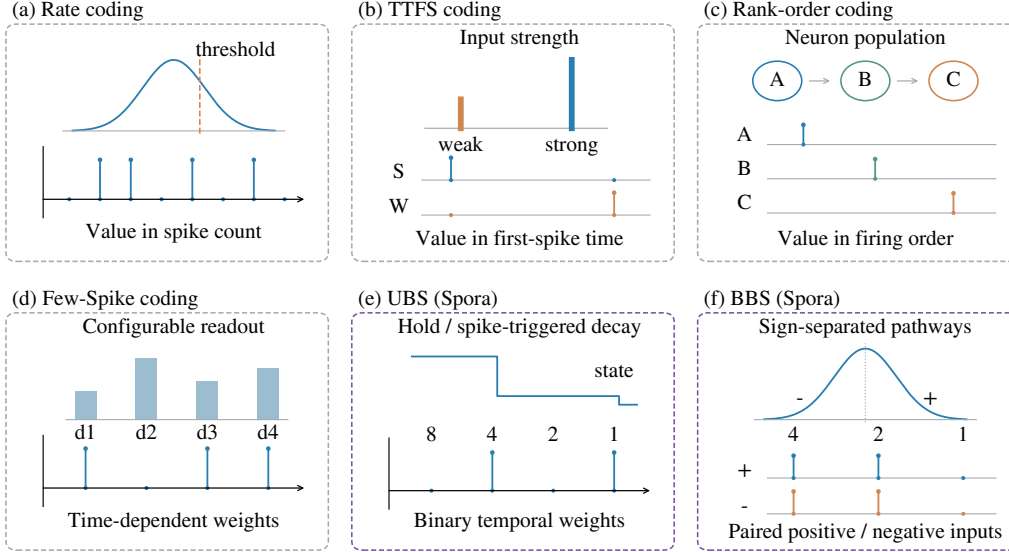


Figure 1: Spike encodings: (a) rate, (b) TTFS, (c) rank-order, (d) Few-Spike, (e) UBS, and (f) BBS. Event labels denote readout weights; BBS traces show separate positive and negative inputs.

actions through attention. Mapping continuous activations into short spike sequences can therefore affect not only individual feature values but also the attention distributions that combine information across tokens.

Existing spiking language models approach this problem from two directions. Models that introduce spikes into selected linear modules retain floating-point Softmax to preserve attention fidelity (Xing et al., 2024). Hardware-oriented alternatives replace nonlinear operations with approximations (Tang et al., 2025; Zhou et al., 2026), while temporal budgets remain important for representation precision. Increasing the number of steps provides more encoding opportunities but also increases temporal processing. *Although these two categories differ in implementation, they face the same fundamental challenge: preserving fine-grained semantic information within a short temporal window while avoiding costly nonlinear attention computation.*

Figure 1 illustrates the relevant encoding choices. Rate coding counts events, TTFS uses first-spike latency (Yu et al., 2023), and rank-order coding uses relative firing order across neurons (Rufin Van Rullen, 2001). Weighted-spike and Few-Spike methods assign numerical readout weights to temporal events (Kim et al., 2018; Stöckl and Maass, 2021). Our approach uses binary place values: an event’s time index identifies its numerical contribution. Unlike a spike-count estimate, this readout forms an integer operand that can enter subsequent arithmetic directly.

Following this principle, Spora combines Unipolar Binary Spiking (UBS) and Bipolar Binary Spiking (BBS) encodings. UBS configures input partitions through thresholds and spike-triggered residual decay while retaining fixed binary readout weights. BBS separates sign from magnitude and uses a learnable scale to represent signed activations. These complementary codes support accumulation-and-shift dot products and integer-exponent attention mappings. The design follows two requirements: encoding should adapt to activation distributions, and the resulting values should preserve the arithmetic structure needed by the next operator. Conditional decay and learned scales serve the first requirement; binary readout serves the second. This connects the temporal representation to the attention computation that consumes it.

Our contributions are:

(1) Unipolar encoding. UBS distinguishes the numerical values specified by binary readout from the input intervals determined by thresholds and conditional decay. This provides short-window integer representations for shift-based exponential approximation.

(2) **Bipolar encoding.** BBS applies a shared magnitude process to both signs, with a learnable scale that adapts its finite signed codebook to the feature range.

(3) **Spiking attention and evaluation.** The combined design replaces major attention multiplications and exponential evaluations with accumulation and shifts, yielding a theoretical 98.53% reduction in core attention energy. Four-step Spora reaches 76.6 average GLUE score and 44.1 CoLA MCC, exceeding SpikeLM by 1.2 and 6.2 points. Conditional-decay analysis, matched-budget Static-QAT comparisons, event workloads, and fixed-point experiments connect these task results to encoding fidelity and computational cost.

2 Related Work

Spiking neural encoding. Rate coding uses equally weighted events to approximate continuous values. Spike Firing Approximation (Yao et al., 2025) and memory-efficient training methods (Xiao et al., 2022; Meng et al., 2023) reduce training costs, but spike-count readout still limits the number of representable levels. Temporal alternatives include TTFS (Yu et al., 2023; Zhao et al., 2025) and rank-order coding (Rufin Van Rullen, 2001), with tradeoffs in training stability and decoding dynamics (Wei et al., 2023). Weighted-spike and Few-Spike methods introduce temporal numerical weights (Kim et al., 2018; Stöckl and Maass, 2021). Spora uses fixed binary weights and conditional residual decay to obtain integer codes suited to signed features and attention operators.

Transformer-based SNNs. Spikformer (Zhou et al., 2023b) replaces Softmax with spiking self-attention and reaches 74.81% ImageNet-1k accuracy (Deng et al., 2009) in four steps. Spike-driven Transformer (Yao et al., 2023) uses additions and masking, while QKformer (Zhou et al., 2024) introduces linear-complexity Q-K attention. MaxFormer (Fang et al., 2026) and SpikePool (Lee et al., 2025) use max-pooling in patch embedding and attention, respectively, to improve signal transmission. Language models such as BERT (Devlin et al., 2019) additionally require precise interactions between context-dependent token features, motivating joint treatment of representation and attention approximation.

Hardware-adapted language SNNs. SpikeBERT (Lv et al., 2026) uses Not-XOR spiking attention, and WE-SpikingFormer (Zhou et al., 2026) uses winner-take-all attention. Sorbet (Tang et al., 2025) combines shift-based PTsoftmax and PowerNorm to avoid multiplication, division, and square roots, using sixteen time steps. SpikeLM (Xing et al., 2024) improves information flow through ternary spikes and adaptive thresholds, but retains exponential attention operations, as does SpikeLLM (Xing et al., 2025). Spora instead links binary temporal readout to integer-exponent attention, addressing short-window precision and nonlinear computation within one encoding design.

3 Method

Spora combines BBS-encoded signed features with UBS-based nonlinear mappings (Figure 2). Binary temporal weights give both representations explicit numerical values for accumulation-and-shift computation.

3.1 Representational Capacity and Numerical Roles

We measure the size of a decoded alphabet by $C = \log_2 |\mathcal{C}|$, where $\mathcal{C}$ is the set of distinct readout values. For a binary sequence of length U , count-based decoding retains only $K = \sum_{t=1}^U S(t)$, whereas binary-weight decoding retains its place values:

$$\begin{aligned} \mathcal{C}_{\text{count}} &= \{0, \dots, U\}, \\ C_{\text{count}} &= \log_2(U + 1), \\ \mathcal{C}_{\text{binary}} &= \{0, \dots, 2^U - 1\}, \\ C_{\text{binary}} &= U. \end{aligned} \tag{1}$$

For example, at $U = 4$, count readout maps the sixteen possible spike patterns to five values, while binary readout assigns each pattern a distinct integer. This is an alphabet-capacity comparison: the thresholds and residual dynamics determine which patterns an input distribution actually occupies.

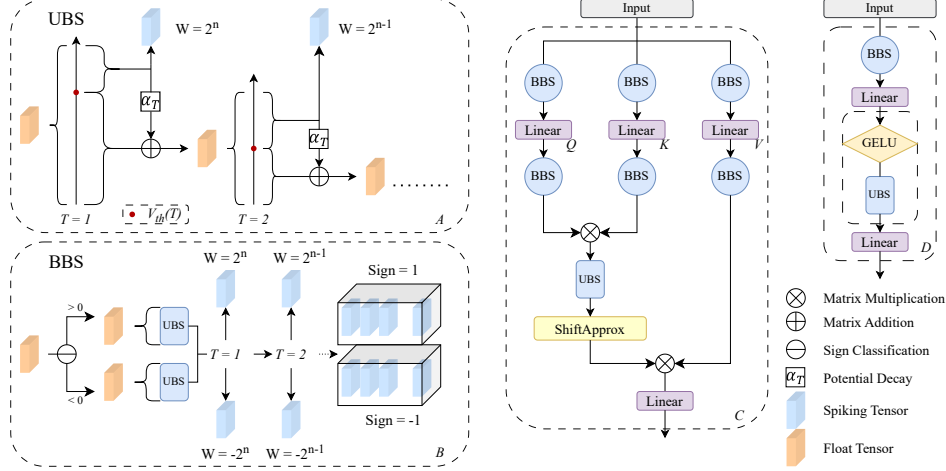


Figure 2: Spora architecture. (A) UBS temporal weights; (B) sign-separated BBS; (C) approximate attention, with ShiftApprox denoting shift-based Softmax approximation; (D) feed-forward block, where UBS implements the GELU approximation during inference.

UBS uses the larger alphabet for nonlinear mappings whose outputs must have explicit integer meaning.

BBS extends this construction to signed features. With M magnitude decisions, the signed alphabet contains at most $2(2^M - 1) + 1 = 2^{M+1} - 1$ states. Throughout the experiments, $T = 2M$ denotes the BBS configuration with M positive and M negative bit planes. Thus, $T = 4$ and $T = 6$ correspond to seven and fifteen signed states. UBS uses $U = 4$ stages in both configurations. This separates the BBS temporal budget from the UBS exponent-code budget.

3.2 Unipolar Binary Spiking Encoding

UBS processes larger-weight decisions before lower-weight refinements. We use U for its number of stages, reserving T for the BBS configuration reported in the experimental tables.

For non-negative input x , UBS initializes $V(0) = x$ and uses U temporal weights $w(t) = 2^{U-t}$. At each step, a threshold comparison generates a binary spike:

$$\begin{aligned} S(t) &= \Theta[V(t-1) - V_{\text{th}}(t)], \\ S(t) &\in \{0, 1\}, \end{aligned} \quad (2)$$

where Θ is the Heaviside function. When $S(t) = 1$, the threshold is subtracted and the remaining potential is multiplied by α . When $S(t) = 0$, the potential is passed unchanged to the next comparison. This spike-triggered update is

$$V(t) = \begin{cases} \alpha [V(t-1) - V_{\text{th}}(t)], & S(t) = 1, \\ V(t-1), & S(t) = 0. \end{cases} \quad (3)$$

The decoded value is $\text{UBS}(x) = \sum_{t=1}^U w(t)S(t)$. **How conditional decay shapes the code.** Binary readout specifies output values, while the residual dynamics determine their input regions. Consider the first two decisions, writing $\theta_t = V_{\text{th}}(t)$. The second spike satisfies

$$\begin{aligned} S(2) = 1 &\iff \\ \begin{cases} x \geq \theta_2, & S(1) = 0, \\ x \geq \theta_1 + \theta_2/\alpha, & S(1) = 1. \end{cases} \end{aligned} \quad (4)$$

Thus, the effective threshold for a later event depends on whether an earlier event fired. Changing α moves this boundary without changing either event’s readout weight. Over multiple steps, this mechanism shapes input partitions through the preceding spike sequence. The no-decay comparison in Section 4.2 tests the effect of this partitioning on attention fidelity while retaining binary readout.

Thresholds and decay are fitted to the target mapping while keeping the readout weights fixed; Section 3.5 gives the search procedure and training gradients.

3.3 Bipolar Binary Spiking Encoding

BBS separates sign and magnitude to encode signed activations, such as layer-normalized features, building on bidirectional spike representations in SpikeLM (Xing et al., 2024). For $x \in \mathbb{R}$, it initializes $V(0) = |x|/s$ with learnable scale $s > 0$ and applies Equations 2–3 to the magnitude. With $M = T/2$ magnitude steps and binary weights $w_M(t) = 2^{M-t}$, the output is

$$Q(|x|) = \sum_{t=1}^M w_M(t) S(t), \quad (5)$$

$$\text{BBS}(x) = s \text{sign}(x) Q(|x|).$$

The T binary spike tensors comprise M positive and M negative pathways; negative-path events contribute with a minus sign. The default $T = 4$ configuration uses magnitude thresholds and weights $[2, 1]$, while $T = 6$ uses $[4, 2, 1]$. The scale adapts the fixed-threshold range to activation distributions and can be fused into the corresponding linear weights for accumulation-based inference.

Separating polarity from magnitude gives positive and negative inputs the same magnitude-encoding rule. At $T = 4$, the decoded magnitudes range from 0 to 3, giving seven signed states including zero; the $T = 6$ configuration extends this to magnitudes from 0 to 7 and fifteen signed states. The scale controls their spacing in the original activation domain. This provides the representation budgets used in the workload and Static-QAT experiments.

BBS therefore combines uniformly spaced reconstruction values with input regions determined by the spike recurrence. A static uniform quantizer can use the same signed values while assigning inputs through uniformly spaced thresholds. Matching the number of states compares equal-sized output alphabets; the downstream experiment examines how these different assignments perform within the complete model.

3.4 Spiking Attention Mechanism

Spiking QKV representation. BBS encodes the outputs of the Query, Key, and Value projections. For $T = 4$, define the signed integer tensor $\tilde{Q}$ by

$$\text{BBS}(Q) = s_q \tilde{Q}, \quad \tilde{Q} = Q_{P1} + 2Q_{P2} - Q_{N1} - 2Q_{N2}, \quad (6)$$

where Q_{Pj} and Q_{Nj} are binary positive and negative tensors with magnitude weight j . Defining $\tilde{K}$ and $\tilde{V}$ analogously gives

$$\text{BBS}(Q) \text{BBS}(K)^T = s_q s_k \tilde{Q} \tilde{K}^T. \quad (7)$$

Expanding the binary planes expresses the product through binary interactions, signed accumulations, and power-of-two shifts, followed by the scale factor. In particular, multiplying two binary-weight contributions produces another power-of-two coefficient. Their magnitudes can therefore be handled by shifts, while the positive and negative pathways determine the accumulation sign. The same signed representation is retained for the Value projection.

Spiking approximation of Softmax. The identity $e^{x/\sqrt{d}} = 2^{x/(\sqrt{d} \ln 2)}$ converts exponential evaluation into an integer-exponent mapping. Using the threshold search above, UBS approximates the exponent by

$$c(x) = \text{UBS}(x) = \sum_{t=1}^U 2^{U-t} S(t) \approx \left\lfloor \frac{x}{\sqrt{d} \ln 2} \right\rfloor, \quad (8)$$

$$e^{x/\sqrt{d}} \approx 2^{c(x)} = \prod_{t=1}^U 2^{2^{U-t} S(t)}.$$

Each active bit contributes a power-of-two shift, and the integer exponent determines the combined shift amount. In this construction, UBS approximates the exponent code; the subsequent power-of-two mapping supplies the nonlinearity used for attention scores. Let $C = \text{UBS}(s_q s_k \tilde{Q} \tilde{K}^U)$ and $E = 2^C$, with UBS and exponentiation applied elementwise. The normalized attention output is

$$\text{Softmax}\left(\frac{QK^U}{\sqrt{d}}\right) V \approx \text{diag}(E\mathbf{1})^{-1} E s_v \tilde{V}, \quad (9)$$

where $\mathbf{1}$ sums over keys. Integer codes support shift-based exponentiation and accumulation with BBS-encoded values; row normalization and scale application remain floating-point operations. At inference, $s_q s_k$ can be absorbed into the UBS thresholds, leaving s_v on the output.

Why evaluate attention-output fidelity? For a reference attention row $p_j \propto 2^{u_j}$, exponent-code errors δ_j yield

$$\hat{p}_j = \frac{p_j 2^{\delta_j}}{\sum_k p_k 2^{\delta_k}}, \quad (10)$$

where the sum is over valid keys. A common exponent offset cancels during normalization, whereas differences between code errors redistribute attention across tokens. The effect on the output also depends on the associated Value vectors. This motivates measuring $\hat{P}V$ against reference attention outputs in addition to fitting the exponent codes: the former evaluates the quantity passed to the next layer.

For training, we use a straight-through estimator (STE) (Bengio et al., 2013) for spike firing. UBS and BBS return decoded integer representations without explicitly unrolling spike propagation through subsequent layers; the trained parameters are used for the corresponding temporal binary encoding at inference.

3.5 Threshold Search and Differentiable Training

To approximate a target function $f(x)$, we optimize four thresholds $\theta = [\theta_1, \dots, \theta_4]$ and α for the decoded output $P(x; \theta, \alpha)$:

$$\min_{\theta, 0 < \alpha \leq 1} \mathcal{L}(\theta, \alpha) = \frac{1}{N} \sum_{i=1}^N [P(x_i; \theta, \alpha) - f(x_i)]^2 + \lambda \mathbb{I}. \quad (11)$$

Inputs x_i are uniformly sampled from the target domain; $\mathbb{I}$ indicates violations of threshold ordering and range constraints, penalized by λ . Differential evolution (Storn and Price, 1997) provides a global approximate solution, followed by local grid refinement. The resulting parameters are used for the selected activation and exponential mappings. The global stage searches across threshold configurations, while local refinement adjusts the best candidate’s partitions. Readout weights remain powers of two throughout the search, so fitting the target mapping does not change the arithmetic structure of the resulting code.

Offline search. For a chosen target f and domain $[a, b]$, offline search uses 1,201 uniformly spaced inputs, a population of 125 candidates, and 120 generations. Differential mutation samples $F \sim \mathcal{U}(0.5, 1)$ and binomial crossover uses probability 0.7, with seed 42. Threshold bounds are $[1, 60]$ and decay bounds are $[0.01, 1]$. After differential evolution, thresholds are rounded to integers and locally refined. The target domain and threshold–weight association are specified for each mapping. This offline function fit supplies encoder parameters; the data-based recalibration in Section 4.2 is a separate mechanism experiment.

Surrogate gradients. Network training uses straight-through or smooth surrogate derivatives for discrete firing (Bengio et al., 2013). For the attention UBS code, the clipped surrogate is $Q_{\text{att}}(x) = \text{clip}(x/(\sqrt{d} \ln 2), 0, n)$; for post-activation FFN UBS, it is $Q_{\text{ffn}}(x) = \text{clip}(x, 0, n)$, where $n = 2^U - 1$ is the maximum unsigned code. These surrogates propagate gradients through the corresponding code range while the forward computation retains discrete values.

For BBS, write $u = |x|/s$ and $y = s \text{sign}(x)q(u)$, where q is the decoded magnitude from the complete residual recurrence. If $q'_{\text{sur}}(u)$ denotes its surrogate derivative, the scale gradient follows the chain rule:

$$\frac{\partial \mathcal{L}}{\partial s} \approx \frac{\partial \mathcal{L}}{\partial y} \text{sign}(x) [q(u) - u q'_{\text{sur}}(u)]. \quad (12)$$

The first term accounts for rescaling the decoded magnitude, and the second accounts for moving the normalized input relative to the thresholds. Smooth spike derivatives provide gradients near decision boundaries and are propagated through the residual updates. A smaller scale increases the normalized magnitude and threshold crossings; a larger scale widens the represented activation range. Learning s therefore adapts the fixed magnitude codebook to each encoder’s feature distribution.

Algorithm 1 Differential evolution and local refinement for UBS

Require: Target f , sampled inputs $\{x_k\}$, feasible set $\mathcal{B}$, population size N_P , generations G

Ensure: Thresholds and decay $z^* = (\theta^*, \alpha^*)$

- 1: Initialize $z_1, \dots, z_{N_P} \in \mathcal{B}$; evaluate the loss in Equation 11
 - 2: **for** $g = 1, \dots, G$ **do**
 - 3: Let z_{best} be the lowest-loss candidate
 - 4: **for** each candidate z_i **do**
 - 5: Select distinct $r_1, r_2 \neq i$ and sample $F \sim \mathcal{U}(0.5, 1)$
 - 6: Form $v_i = z_{\text{best}} + F(z_{r_1} - z_{r_2})$
 - 7: Cross v_i with z_i at rate 0.7, retaining at least one coordinate from v_i
 - 8: Repair the trial to satisfy $\mathcal{B}$ and evaluate its loss
 - 9: Retain the trial if its loss is no larger than that of z_i
 - 10: **end for**
 - 11: **end for**
 - 12: Round the best thresholds to integers and refine locally under the same constraints
 - 13: **return** the best feasible z^*
-

4 Results

Benchmarks and metrics. We evaluate eight GLUE datasets spanning grammatical acceptability (CoLA), sentiment classification (SST-2), paraphrase identification (QQP and MRPC), semantic similarity (STS-B), and inference (MNLI, QNLI, and RTE). MNLI provides matched and mismatched development splits, giving nine reported task columns. Classification tasks use accuracy, CoLA uses Matthews correlation, and STS-B uses Spearman correlation, all on a 0–100 scale. The all-task average is the arithmetic mean of these nine columns. The reduced average excludes CoLA and follows the MNLI-split convention marked in Table 1.

Training setup. We evaluate Spora on GLUE (Wang et al., 2019) after pre-training on English Wikipedia (Devlin et al., 2019) and BookCorpus (Zhu et al., 2015) at maximum sequence length 128. Training uses AdamW (Loshchilov and Hutter, 2019), a learning rate of 5×10^{-5} , 5,000 warmup steps, and mixed precision (Micikevicius et al., 2018). Multi-level BERT MLM distillation (Sun et al., 2019) transfers the BERT-base teacher’s output distribution, hidden representations, and attention matrices, followed by GLUE fine-tuning. Training uses four 24GB RTX4090 GPUs; layer-local mechanism evaluation uses one RTX4090 D.

4.1 Performance on GLUE

Table 1 compares Spora with full-precision BERT (Devlin et al., 2019); quantized models Q2BERT (Shen et al., 2020), BinaryBERT (Bai et al., 2021), BiBERT (Qin et al., 2022), Ternary-BERT (Zhang et al., 2020), BiT (Liu et al., 2022), and BiPET (Xing et al., 2024); and spiking language models. Among the quantized and spiking baselines, all except Q2BERT, WE-SpikingFormer, and SpikingFormer employ knowledge distillation.

With four time steps, Spora reaches 76.6 average score across all reported GLUE columns and 80.9 under the table’s reduced averaging convention. Relative to SpikeLM (Xing et al., 2024), it improves QQP by 1.3 points and MNLI-m/mm by 3.2/2.5 points. The largest gain is on CoLA: 44.1 MCC versus 37.9 for SpikeLM, 27.9 for WE-SpikingFormer (Zhou et al., 2026), and 16.7 for SpikingFormer (Zhou et al., 2023a). On SST-2, Spora reaches 89.6%, approaching Sorbet’s (Tang et al., 2025) 90.4% with four rather than sixteen steps. It also exceeds SpikingBERT (Bal and Sengupta, 2024) on QQP, MNLI-m, and SST-2 with four rather than 125 steps. Spora also reaches 85.4% on QNLI and 80.2% on MRPC, compared with 84.9% and 78.7% for SpikeLM. On QQP, its 89.2% accuracy is within 2.1 points of full-precision BERT. These improvements span sentence-pair classification, sentiment analysis, and grammatical acceptability, showing that the short-window design benefits several language-understanding tasks rather than a single benchmark.

4.2 Encoding and Attention Analysis

Conditional-decay protocol. We evaluate $\text{Spora}_{T=6}$ on CoLA at maximum length 128, comparing original four-step UBS, no-decay UBS ($\alpha = 1$) with recalibrated thresholds, jointly calibrated UBS,

Table 1: GLUE development results (0–100). QQP/MRPC: accuracy; STS-B: Spearman’s correlation; CoLA: MCC. Avg. excludes CoLA; Avg._{all} includes it. Asterisks indicate averages using only one MNLI split. Time denotes time steps. Spora_{T=4}’s QQP/MRPC F1 scores are 85.9/86.6; the reduced average for Spora_{T=6} uses MNLI-m.

Model	Time	QQP	MNLI _{m/mm}	SST-2	QNLI	RTE	MRPC	STS-B	Avg.	CoLA	Avg. _{all}
<i>ANN</i>											
BERT	–	91.3	83.8/83.4	92.3	90.7	69.3	88.2	89.4	86.1	60.0	83.2
Q2BERT	–	67.0	47.2/47.3	80.6	61.3	52.7	68.4	4.4	53.6	0.0	47.7
BinaryBERT	–	79.9	62.7/63.9	82.5	52.6	52.7	68.3	6.5	58.6	14.6	53.7
BiBERT	–	83.7	66.1/67.5	88.7	72.6	57.4	72.5	33.6	67.8	25.4	63.0
TernaryBERT	–	63.1	40.3/40.0	80.7	50.0	54.5	68.3	12.4	51.2	0.0	45.5
BiT	–	82.9	77.1/77.5	87.7	85.7	58.8	79.7	71.1	77.6	25.1	71.7
BiPET	–	83.7	69.5/70.6	86.2	81.7	66.1	76.2	80.2	76.8	22.9	70.8
<i>SNN</i>											
SpikingFormer	4	83.8	67.8/68.6	82.7	74.6	58.8	74.0	72.3	73.4	16.7	66.6
SpikeLM	4	87.9	76.0/76.9	86.5	84.9	65.3	78.7	84.3	80.1	37.9	75.4
Sorbet	16	86.5	77.3/–	90.4	86.1	60.3	79.9	78.1	79.8*	–	–
SpikingBERT	125	86.8	78.1/–	88.2	85.2	66.1	79.2	82.2	80.8*	–	–
WE-SpikingFormer	4	85.1	–/70.1	89.0	77.5	55.9	81.6	42.8	71.7	27.9	66.2
Spora _{T=4}	4	89.2	79.2/79.4	89.6	85.4	60.0	80.2	82.5	80.9*	44.1	76.6
Spora _{T=6}	6	89.4	80.1/80.6	89.9	86.9	63.2	82.2	84.3	82.3*	47.4	78.2

and static uniform integer encoding. Calibration minimizes exponent-code MSE on 256 training examples under the same maximum search budget; evaluation uses all 1,043 development examples in FP32, excluding padding.

For attention scores Z , let $r = \text{round}(Z)$. The encoder input is $x = \text{clip}(r, 0, 60)$, and the calibration target is $x/(8 \ln 2)$. We compare $\hat{P}V$ with two reference outputs:

$$\begin{cases} P_{\text{raw}}V = \text{Softmax}(Z/8)V, \\ P_{\text{shared}}V = \text{Softmax}(x/8)V. \end{cases} \quad (13)$$

The raw-score reference measures total error, while the shared-preprocessing reference measures coding error after the same preprocessing. For either reference P , output NMSE is

$$\text{NMSE} = \frac{\sum \|(\hat{P} - P)V\|_F^2}{\sum \|PV\|_F^2}, \quad (14)$$

where both sums are aggregated over valid queries.

Attention-output fidelity. Original UBS lowers error relative to recalibrated no-decay UBS in all 12 layers (Figure 3). Aggregate NMSE decreases from 0.02100 to 0.01834 under the raw-score reference and from 0.01277 to 0.01066 under the shared-preprocessing reference, corresponding to reductions of 12.69% and 16.58%, respectively. Static uniform encoding achieves raw-score NMSE of 0.01805 and shared-preprocessing NMSE of 0.01063. Joint UBS recalibration improves the original by less than 0.2% relative.

Conditional decay thus refines input partitions while retaining binary readout, bringing temporal encoding close to direct integer encoding in attention fidelity. The raw and shared references distinguish errors introduced before encoding from those of the integer mapping itself. Improvement under both references connects the residual update to the attention output, while the small recalibration gain indicates that the checkpoint’s UBS configuration is already close to the jointly calibrated setting.

Comparison with static activation quantization. We next compare the complete BBS/UBS design against Static-QAT on CoLA.

Spora_{T=6} uses three magnitude steps [4, 2, 1] with 15 signed states; Static-QAT uses symmetric 4-bit quantization with the same 15 states. Both have a four-bit UBS representation budget. Static-QAT replaces the BBS and UBS encoders with uniform quantizers with learnable scales, retaining the architecture and base-2 attention normalization. It follows the four-epoch MLM distillation schedule with the same BERT-base teacher, temperature 2.0, and MLM/hidden/attention loss weights of 1.0/0.5/5.0, followed by CoLA fine-tuning and development-MCC checkpoint selection. Its best checkpoint is evaluated alongside the existing fine-tuned Spora_{T=6} checkpoint.

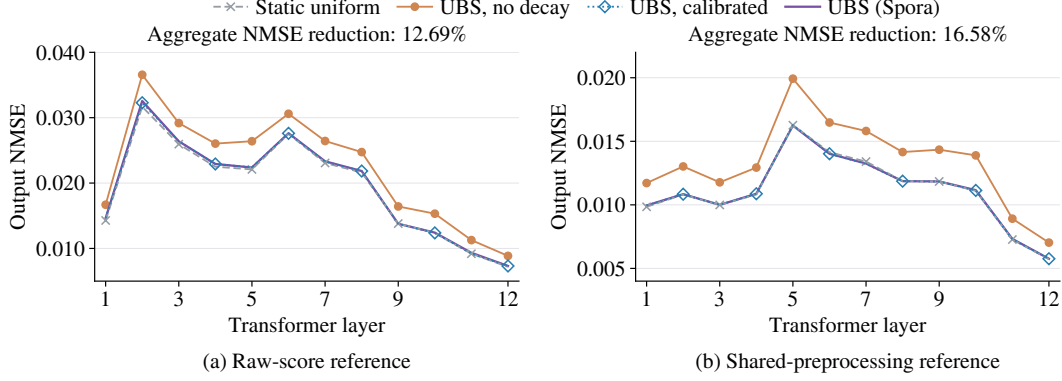


Figure 3: Attention-output NMSE for four-step integer encoders: (a) raw-score and (b) shared-preprocessing references. Annotations report aggregate reductions relative to recalibrated no-decay UBS.

Table 2: CoLA comparison under matched representation budgets: 15 signed states and four-bit UBS.

Method	Acc. (%)	MCC $\times 100$
Static-QAT	76.61	40.97
Spora _{T=6}	79.00	47.40

Table 2 reports 47.40 MCC and 79.00% accuracy for Spora_{T=6}, exceeding Static-QAT by 6.43 MCC points and 2.39 accuracy points. The two comparisons address successive parts of the design. The layer-local study measures how UBS partitions affect a specific attention operator and shows fidelity close to static integer encoding. The downstream comparison includes signed BBS features, UBS mappings, and their interaction with training. Its 6.43-point MCC gain establishes a task-level advantage for the complete design under the matched representation budget.

4.3 Computational Cost and Numerical Precision

Arithmetic energy accounting. We estimate energy as operation counts multiplied by technology-level unit costs (Horowitz, 2014). For FP32 arithmetic, the reference costs are 4.6 pJ per MAC, 3.7 pJ per multiplication, and 0.9 pJ per addition. In the FP32 linear-layer accounting, we use the addition cost as a proxy for a shift. The low-bit attention model uses 0.03 pJ for an 8-bit integer addition and 0.1 pJ for a 32-bit integer addition; exponential evaluation uses the 1.7 pJ estimate adopted in Sorbet (Tang et al., 2025). These are arithmetic energy estimates for the specified operation models.

For a linear map with dense MAC count N_{dense} , let r_t be the active fraction in signed bit plane t and w_t its numerical weight. Event-wise accounting gives

$$\begin{aligned}
 N_{\text{AC}} &= N_{\text{dense}} \sum_t r_t, \\
 N_{\text{shift}} &= N_{\text{dense}} \sum_{t: |w_t| > 1} r_t, \\
 E_{\text{event}} &= N_{\text{AC}} E_{\text{AC}} + N_{\text{shift}} E_{\text{shift}}, \\
 E_{\text{dense}} &= N_{\text{dense}} E_{\text{MAC}}.
 \end{aligned} \tag{15}$$

The shifted planes are identified by weight magnitude rather than their storage order. For weights $[2, 1]$, only magnitude-two events contribute to the shift term. Sign selects addition or subtraction. This notation connects the energy estimate to the measured event counts in Figure 4.

Attention is accounted for component by component: binary-plane products and shifts for QK^T , UBS code generation and shifts for exponentiation, row normalization, accumulation with V , and output scaling. The energy of a component is $\sum_o N_o E_o$ over its arithmetic operations. Normalization

Table 3: Left: theoretical attention energy in μJ (Conv.: conventional; Approx.: proposed). Right: fixed-point emulation on CoLA. State and Prob. denote BBS membrane-state and attention-probability formats; Sat. is the percentage of clipped state values.

Module	Conv.	Approx.	Format	State	Prob.	Acc. (%)	MCC $\times 100$	Sat. (%)
QK^T	57.881	0.303	FP32	Float	Float	77.95	44.39	—
$e^x/2^{\text{UBS}}$	0.334	0.028	16-bit	Q6.10	U16	77.56	43.44	0.0194
Denominator	0.176	0.005	12-bit	Q6.6	U12	76.51	40.64	0.0193
AV	57.881	1.008	8-bit	Q6.2	U8	77.28	42.90	0.0184
s_v scaling	—	0.364						
Total	116.272	1.7083						

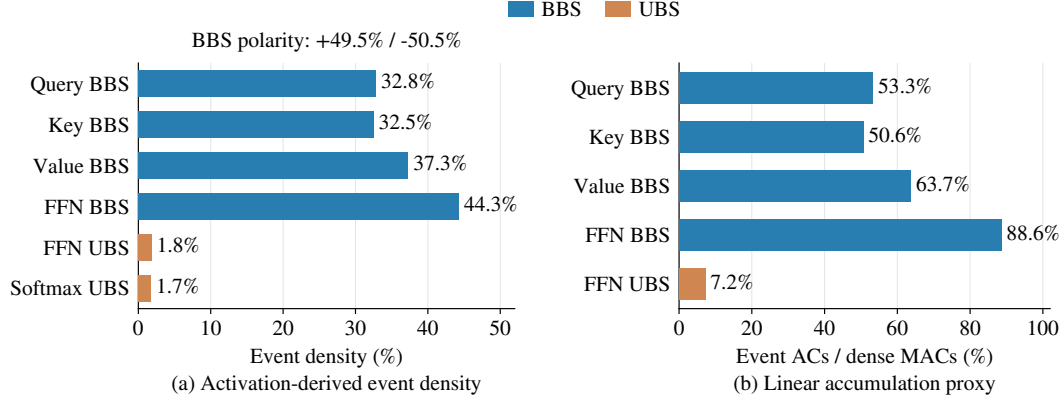


Figure 4: Activation-derived workload on CoLA at padded length 128. Left: event density by component. Right: estimated event-driven accumulations relative to dense MACs in spike-to-linear paths.

and value scaling remain explicit in Table 3; the total reduction therefore refers to the complete core-attention accounting shown there.

Theoretical attention energy. For BERT-base at sequence length 128, with 12 heads of dimension 64, estimated core attention energy decreases from 116.272 to 1.7083 μJ per layer, a 98.53% reduction (Table 3, left). The savings arise from accumulation-and-shift computation for QK^T and AV and integer-exponent mappings for Softmax. Query/key scales are absorbed into UBS thresholds; value scaling is applied to the output.

Workload and precision protocol. We use the 1,043-example CoLA development set at maximum length 128 and the fine-tuned four-step model: two BBS magnitude steps [2, 1], seven signed states $\{-3, -2, -1, 0, 1, 2, 3\}$, and four UBS bit-planes.

Event workload. Figure 4 aggregates active events and temporal slots across layers. BBS densities range from 32.5% to 44.3%, with balanced positive/negative events (49.5%/50.5%); FFN and attention UBS densities are 1.8% and 1.7%. For spike-to-linear paths, active-event counts multiplied by output fanout estimate event-driven accumulations. Their aggregate ratio to dense MACs is 50.1%, falling to 7.2% for FFN UBS, which offers the sparsest accumulation path. The contrast with FFN BBS’s 88.6% ratio shows where event sparsity contributes most. Binary weights determine how an event is processed, while activation-dependent firing determines how many such operations are needed; these are complementary sources of computational savings.

Fixed-point emulation. We quantize BBS membrane potentials to $Q6.(b-6)$, round the decay coefficient, and uniformly quantize attention probabilities to $b \in \{8, 12, 16\}$ bits. Integer thresholds remain exact; learned scales and other operators remain floating point, and automatic mixed precision is disabled. Table 3 (right) reports 43.44/42.90 MCC at 16/8 bits, decreases of 0.95/1.48 points from FP32’s 44.39. The 12-bit configuration reaches 40.64 MCC; saturation remains below 0.02% in all cases. The 16- and 8-bit formats therefore retain most downstream accuracy for low-precision states and probabilities.

Table 4: Whole-model arithmetic energy estimates at the reported configurations. Starred values are inferred from energy ratios in the corresponding papers. The accounting uses FP32 MAC/AC reference costs.

Model	BERT-base	LIF-BERT _{T=4}	SpikeLM _{T=4}	SpikingBERT	SpikeBERT _{T=4}	Sorbet _{T=16}	Spora _{T=4}
Energy (mJ)	51.41	7.98	13.74	12.85*	14.74	24.14*	11.17

Table 5: Ablation study on GLUE. Spora₁ removes BBS encoding from the Query-layer input. Spora_{Softmax} restores standard scaled dot-product attention while retaining BBS-encoded Q/K/V. Spora w/o KD removes knowledge distillation.

Model	QQP	MNLI _{m/mm}	SST-2	QNLI	RTE	MRPC	STS-B	CoLA	Avg.
Spora	89.2	79.2/79.4	89.6	85.4	60.0	80.2	82.5	44.1	76.6
Spora ₁	88.9	79.3/79.4	89.4	85.2	58.6	80.9	83.2	46.8	76.8
Spora _{Softmax}	88.8	78.5/79.3	89.7	85.0	62.1	81.2	80.7	45.2	76.7
Spora w/o KD	87.1	76.4/76.7	87.2	83.2	57.1	79.4	79.7	38.5	73.9

Whole-model comparison. The FP32 MAC/AC accounting summarized in Table 4 estimates 11.17 mJ for four-step Spora, compared with 51.41 mJ for BERT-base, a ratio of approximately $4.60\times$. This whole-model estimate includes computation outside the core attention block and therefore has a different scope from the 98.53% core-attention reduction.

4.4 Ablation Study

All variants in Table 5 are pre-trained on the same corpus and fine-tuned on GLUE. Replacing approximate attention with standard Softmax changes the average score from 76.6 to 76.7; removing Query-input BBS changes it to 76.8. The complete model thus retains comparable average accuracy with both encoding and approximate attention enabled. These ablations answer complementary questions: Query-input BBS tests the effect of encoding features before the projection, whereas the Softmax variant tests the nonlinear attention approximation with BBS-encoded Q/K/V retained. The small average differences show that both choices preserve task performance within this architecture. Under the same time-step and training budgets, knowledge distillation increases the average from 73.9 to 76.6 and CoLA MCC from 38.5 to 44.1.

Increasing the temporal budget. As shown in Table 1, Spora_{T=6} improves the all-task average from 76.6 to 78.2, a gain of 1.6 points, and improves every reported task column. CoLA rises by 3.3 points, RTE by 3.2, MRPC by 2.0, and STS-B by 1.8. The additional magnitude bit expands the signed codebook from seven to fifteen states while keeping four-stage UBS attention. These results expose a controllable accuracy–representation tradeoff: four-step Spora provides the shorter configuration, while six-step Spora uses finer signed representations to improve task performance.

Attention distribution. Figure 5 compares BERT and Spora across 12 layers. Spora exhibits more dispersed attention than BERT on this example, with further dispersion at $T = 6$. Extending magnitude weights from $[1, 2]$ to $[1, 2, 4]$ increases representational resolution; the associated task improvements are quantified in Table 1.

5 Discussion and Conclusion

Spora gives temporal spike sequences a numerical structure that can be used directly by attention operators. Binary weights preserve place values; conditional decay shapes input partitions; sign-separated BBS and learned scales accommodate the feature distributions of language models. The resulting design links representation to accumulation, shifts, and integer-exponent mappings rather than treating the spike encoder as an isolated discretization step.

The experiments connect these mechanisms at several levels. Conditional-decay analysis measures local attention fidelity, the matched-budget Static-QAT comparison evaluates the complete encoding design after training, and GLUE ablations characterize the accuracy cost of the selected projection and attention approximations. Extending BBS from four to six steps raises the all-task average from 76.6 to 78.2 and CoLA MCC from 44.1 to 47.4. Event workloads, arithmetic energy estimates, and fixed-point emulation further describe the computational properties of the representation.

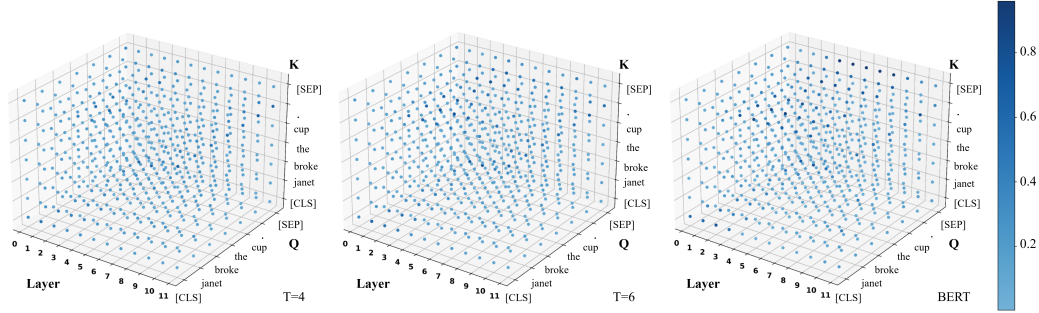


Figure 5: Attention score visualization of BERT and Spora with time steps 4 and 6 on the CoLA sample “Janet broke the cup.” across 12 layers. Q denotes the query token and K denotes the key token.

An immediate extension is to combine activation encoding with weight quantization so that parameter storage and event arithmetic can be optimized together. Implementations that exploit signed bit planes and event sparsity would also allow the operation-level savings to be studied alongside memory traffic and execution overhead. The present results support a joint design principle: short-window codes should preserve useful semantic distinctions and expose numerical structure that makes the next operation simpler.

References

- Mehonic, A., Kenyon, A.J., 2022. Brain-inspired computing needs a master plan. *Nature* 604, 255–260. URL: <https://doi.org/10.1038/s41586-021-04362-w>, doi:10.1038/s41586-021-04362-w.
- Zhang, Y., Qu, P., Ji, Y., Zhang, W., Gao, G., Wang, G., Song, S., Li, G., Chen, W., Zheng, W., Chen, F., Pei, J., Zhao, R., Zhao, M., Shi, L., 2020. A system hierarchy for brain-inspired computing. *Nature* 586, 378–384. doi:10.1038/s41586-020-2782-y.
- Maass, W., 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks* 10, 1659–1671. URL: <https://www.sciencedirect.com/science/article/pii/S0893608097000117>, doi:https://doi.org/10.1016/S0893-6080(97)00011-7.
- Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., Gu, S., 2021. Differentiable spike: Rethinking gradient-descent for training spiking neural networks, in: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (Eds.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., pp. 23426–23439. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/c4ca4238a0b923820dcc509a6f75849b-Paper.pdf.
- Wu, J., Xu, C., Han, X., Zhou, D., Zhang, M., Li, H., Tan, K.C., 2022. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7824–7840. doi:10.1109/TPAMI.2021.3114196.
- Hu, Y., Tang, H., Pan, G., 2023. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems* 34, 5200–5205. doi:10.1109/TNNLS.2021.3119238.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- Zhou, C., Yu, L., Zhou, Z., Zhang, H., Ma, Z., Zhou, H., Tian, Y., 2023a. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *CoRR* abs/2304.11954. URL: <https://doi.org/10.48550/arXiv.2304.11954>.

- Zhou, Z., Zhu, Y., He, C., Wang, Y., YAN, S., Tian, Y., Yuan, L., 2023b. Spikformer: When spiking neural network meets transformer, in: The Eleventh International Conference on Learning Representations. URL: https://openreview.net/forum?id=frE4fUwz_h.
- Li, Y., Lei, Y., Yang, X., 2024. Spikeformer: Training high-performance spiking neural network with transformer. *Neurocomputing* 574, 127279. URL: <https://www.sciencedirect.com/science/article/pii/S092523122400050X>, doi:<https://doi.org/10.1016/j.neucom.2024.127279>.
- Yao, M., Hu, J., Zhou, Z., Yuan, L., Tian, Y., XU, B., Li, G., 2023. Spike-driven transformer, in: Thirty-seventh Conference on Neural Information Processing Systems. URL: <https://openreview.net/forum?id=9Fmoly0Hi5>.
- Du, N., Huang, Y., Dai, A.M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A.W., Firat, O., Zoph, B., Fedus, L., Bosma, M.P., Zhou, Z., Wang, T., Wang, E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q., Wu, Y., Chen, Z., Cui, C., 2022. GLaM: Efficient scaling of language models with mixture-of-experts, in: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (Eds.), *Proceedings of the 39th International Conference on Machine Learning*, PMLR. pp. 5547–5569. URL: <https://proceedings.mlr.press/v162/du22c.html>.
- Xing, X., Zhang, Z., Ni, Z., Xiao, S., Ju, Y., Fan, S., Wang, Y., Zhang, J., Li, G., 2024. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms, in: *Proceedings of the 41st International Conference on Machine Learning*, PMLR. URL: <https://icml.cc/virtual/2024/poster/35024>.
- Zhu, R.J., Zhao, Q., Eshraghian, J.K., 2023. Spikegpt: Generative pre-trained language model with spiking neural networks. *CoRR* abs/2302.13939. URL: <https://doi.org/10.48550/arXiv.2302.13939>.
- Tang, K., Yan, Z., Wong, W.F., 2025. Sorbet: A neuromorphic hardware-compatible transformer-based spiking language model, in: Forty-second International Conference on Machine Learning. URL: <https://openreview.net/forum?id=5dFJukfj4y>.
- Zhou, C., Guo, S., Wang, J., Ma, D., Che, K., Chen, B., Meng, Q., Ma, Z., Tian, Y., 2026. Winner-take-all spiking transformer for language modeling. URL: <https://arxiv.org/abs/2604.11321>, arXiv:2604.11321.
- Yu, M., Xiang, T., P., S., Chu, K.T.N., Amornpaisannon, B., Tavva, Y., Miriyala, V.P.K., Carlson, T.E., 2023. A tfts-based energy and utilization efficient neuromorphic cnn accelerator. *Frontiers in Neuroscience* Volume 17 - 2023. URL: <https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2023.1121592>, doi:10.3389/fnins.2023.1121592.
- Rufin Van Rullen, S.J.T., 2001. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation* 13, 1255–1283. URL: <https://doi.org/10.1162/08997660152002852>.
- Kim, J., Kim, H., Huh, S., Lee, J., Choi, K., 2018. Deep neural networks with weighted spikes. *Neurocomputing* 311, 373–386. URL: <https://www.sciencedirect.com/science/article/pii/S0925231218306726>, doi:<https://doi.org/10.1016/j.neucom.2018.05.087>.
- Stöckl, C., Maass, W., 2021. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence* 3, 230–238.
- Yao, M., Qiu, X., Hu, T., Hu, J., Chou, Y., Tian, K., Liao, J., Leng, L., Xu, B., Li, G., 2025. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, 2973–2990. URL: <http://dx.doi.org/10.1109/TPAMI.2025.3530246>, doi:10.1109/tpami.2025.3530246.
- Xiao, M., Meng, Q., Zhang, Z., He, D., Lin, Z., 2022. Online training through time for spiking neural networks, in: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (Eds.), *Advances in Neural Information Processing Systems*, Curran Associates, Inc.. pp. 20717–20730. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/82846e19e6d42ebfd4ace4361def29ae-Paper-Conference.pdf.

- Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., Luo, Z.Q., 2023. Towards Memory- and Time-Efficient Backpropagation for Training Spiking Neural Networks , in: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE Computer Society, Los Alamitos, CA, USA. pp. 6143–6153. URL: <https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.00567>, doi:10.1109/ICCV51070.2023.00567.
- Zhao, L., Huang, Z., Ding, J., Yu, Z., 2025. TTFSFormer: A TTFS-based lossless conversion of spiking transformer, in: Forty-second International Conference on Machine Learning. URL: <https://openreview.net/forum?id=mJAa823xKu>.
- Wei, W., Zhang, M., Qu, H., Belatreche, A., Zhang, J., Chen, H., 2023. Temporal-Coded Spiking Neural Networks with Dynamic Firing Threshold: Learning with Event-Driven Backpropagation , in: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Los Alamitos, CA, USA. pp. 10518–10528.
- Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. doi:10.1109/CVPR.2009.5206848.
- Zhou, C., Zhang, H., Zhou, Z., Yu, L., Huang, L., Fan, X., Yuan, L., Ma, Z., Zhou, H., Tian, Y., 2024. Qkformer: Hierarchical spiking transformer using q-k attention, in: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 13074–13098. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/179f5dcdeedc149443ebd3ba70811dbd-Paper-Conference.pdf, doi:10.52202/079017-0416.
- Fang, Y., Zhou, D., Wang, Z., Ren, H., Zeng, Z., Li, L., shibo zhou, Xu, R., 2026. Spiking neural networks need high-frequency information, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: <https://openreview.net/forum?id=owNPA17LNK>.
- Lee, D., Sima, A., Li, Y., Stinis, P., Panda, P., 2025. Spikepool: Event-driven spiking transformer with pooling attention. URL: <https://arxiv.org/abs/2510.12102>, arXiv:2510.12102.
- Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding, in: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 4171–4186. URL: <https://aclanthology.org/N19-1423/>, doi:10.18653/v1/N19-1423.
- Lv, C., Li, T., Qiao, W., Wang, X., Wu, M., Liu, W., Dou, S., Zheng, X., Huang, X., 2026. Spikebert: A language spikformer learned from bert with knowledge distillation. Neural Networks 197, 108482. URL: <https://www.sciencedirect.com/science/article/pii/S0893608025013632>, doi:<https://doi.org/10.1016/j.neunet.2025.108482>.
- Xing, X., Gao, B., Zhang, Z., Clifton, D.A., Xiao, S., Du, L., Li, G., Zhang, J., 2025. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. URL: <https://arxiv.org/abs/2407.04752>, arXiv:2407.04752.
- Bengio, Y., Léonard, N., Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. URL: <https://arxiv.org/abs/1308.3432>, arXiv:1308.3432.
- Storn, R., Price, K., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359. doi:10.1023/A:1008202821328.
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R., 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding, in: International Conference on Learning Representations, pp. 353–355. URL: <https://openreview.net/forum?id=rJ4km2R5t7>, doi:10.18653/v1/w18-5446.

- Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S., 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 19–27. URL: https://openaccess.thecvf.com/content_iccv_2015/html/Zhu_Aligning_Books_and_ICCV_2015_paper.html.
- Loshchilov, I., Hutter, F., 2019. Decoupled weight decay regularization, in: International Conference on Learning Representations. URL: <https://openreview.net/forum?id=Bkg6RiCqY7>.
- Miciekevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., Wu, H., 2018. Mixed precision training, in: International Conference on Learning Representations. URL: <https://openreview.net/forum?id=r1gs9JgRZ>.
- Sun, S., Cheng, Y., Gan, Z., Liu, J., 2019. Patient knowledge distillation for BERT model compression, in: Inui, K., Jiang, J., Ng, V., Wan, X. (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China. pp. 4323–4332. URL: <https://aclanthology.org/D19-1441/>, doi:10.18653/v1/D19-1441.
- Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K., 2020. Q-bert: Hessian based ultra low precision quantization of bert. Proceedings of the AAAI Conference on Artificial Intelligence 34, 8815–8821. URL: <http://dx.doi.org/10.1609/aaai.v34i05.6409>, doi:10.1609/aaai.v34i05.6409.
- Bai, H., Zhang, W., Hou, L., Shang, L., Jin, J., Jiang, X., Liu, Q., Lyu, M., King, I., 2021. BinaryBERT: Pushing the limit of BERT quantization, in: Zong, C., Xia, F., Li, W., Navigli, R. (Eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online. pp. 4334–4348. URL: <https://aclanthology.org/2021.acl-long.334/>, doi:10.18653/v1/2021.acl-long.334.
- Qin, H., Ding, Y., Zhang, M., Yan, Q., Liu, A., Dang, Q., Liu, Z., Liu, X., 2022. Bibert: Accurate fully binarized bert. URL: <https://arxiv.org/abs/2203.06390>, arXiv:2203.06390.
- Zhang, W., Hou, L., Yin, Y., Shang, L., Chen, X., Jiang, X., Liu, Q., 2020. TernaryBERT: Distillation-aware ultra-low bit BERT, in: Webber, B., Cohn, T., He, Y., Liu, Y. (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online. pp. 509–521. URL: <https://aclanthology.org/2020.emnlp-main.37/>, doi:10.18653/v1/2020.emnlp-main.37.
- Liu, Z., Oguz, B., Pappu, A., Xiao, L., Yih, S., Li, M., Krishnamoorthi, R., Mehdad, Y., 2022. Bit: Robustly binarized multi-distilled transformer, in: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., pp. 14303–14316. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/5c1863f711c721648387ac2ef745facb-Paper-Conference.pdf.
- Xing, X., Du, L., Wang, X., Zeng, X., Wang, Y., Zhang, Z., Zhang, J., 2024. Bipft: Binary pre-trained foundation transformer with low-rank estimation of binarization residual polynomials. Proceedings of the AAAI Conference on Artificial Intelligence 38, 16094–16102. URL: <https://ojs.aaai.org/index.php/AAAI/article/view/29542>, doi:10.1609/aaai.v38i14.29542.
- Bal, M., Sengupta, A., 2024. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. Proceedings of the AAAI Conference on Artificial Intelligence 38, 10998–11006. URL: <https://ojs.aaai.org/index.php/AAAI/article/view/28975>, doi:10.1609/aaai.v38i10.28975.
- Horowitz, M., 2014. 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. doi:10.1109/ISSCC.2014.6757323.