BiSpikCLM: A Spiking Language Model integrating Softmax-Free Spiking Attention and Spike-Aware Alignment Distillation

Anonymous ACL submission

Abstract

Spiking Neural Networks (SNNs) offer promising energy-efficient alternatives to large language models (LLMs) due to their event-driven nature and ultra-low power consumption. However, to preserve capacity, most existing spiking LLMs still incur intensive floating-point matrix multiplication (MatMul) and nonlinearities, or training difficulties arising from the complex spatiotemporal dynamics. To address these challenges, we propose BiSpikCLM, the first fully binary spiking MatMul-free causal language model. BiSpikCLM introduces Softmax-Free Spiking Attention (SFSA), eliminating softmax and floating-point operations in autoregressive language modeling. For efficient training, we introduce Spike-Aware Alignment Distillation (SpAD), which aligns ANN teacher and SNN student across embeddings, attention maps, intermediate features, and output logits. SpAD framework allows BiSpikCLM to reach comparable performance to ANN counterparts using substantially fewer training tokens (e.g., only 5.6% of the tokens for the 1.3B model). As a result, BiSpikCLM achieves competitive performance at only 4.16%–5.87% of the computational cost on natural language generation tasks. Our results highlight the feasibility and effectiveness of fully binary spike-driven LLMs and establish the distillation as a promising pathway for brain-inspired spiking NLP.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language processing, powering a wide range of applications from conversational agents to code generation (Brown et al., 2020; OpenAI et al., 2024). However, these models typically require extensive computational resources and energy consumption during both training and inference. For example, GPT-3 was trained with 175 billion parameters using hundreds of petaflop/s-days of compute (Brown et al., 2020). In addition, inference also incurs substantial energy costs, as serving a single query can involve billions of operations and significant GPU utilization (Strubell et al., 2020; Schwartz et al., 2020), raising concerns about their scalability and environmental impact (Strubell et al., 2020).

Unlike ANN-based LLMs, the human brain achieves superior intelligence with far lower energy consumption, operating on just 20 watts to power approximately 86 billion neurons (Izhikevich, 2003; Gerstner et al., 2014). Inspired by this energy-efficiency, Spiking Neural Networks (SNNs) (Maass, 1997; Gerstner et al., 2014) communicate through binary spike events, enabling event-driven and low-power computation (Yin et al., 2021; Schuman et al., 2022), particularly on neuromorphic hardware, making SNNs a promising alternative to traditional ANNs.

While recent efforts have shown promising results of SNNs in computer vision tasks (Zhou et al., 2024; Li et al., 2024; Luo et al., 2024), extending SNNs to natural language processing (NLP), especially LLMs, remains largely underexplored. A central challenge is the design of spiking attention mechanisms. In contrast to vision models, where representations are often bidirectional and spatially local, autoregressive LLMs require **causal attention** to ensure that each token prediction depends only on its preceding context. However, conventional causal attention relies on floating-point matrix multiplications and the softmax operation, both of which are computationally intensive and fundamentally incompatible with spike-based processing. Existing attempts either retain these components (Zhu et al., 2023) or introduce multi-threshold neurons and integer activations (Xing et al., 2024a), which still incur substantial floating-point overhead. Designing a spike-driven causal attention mechanism is therefore critical: it must eliminate softmax while preserving the autoregressive representational capacity of binary spike trains.

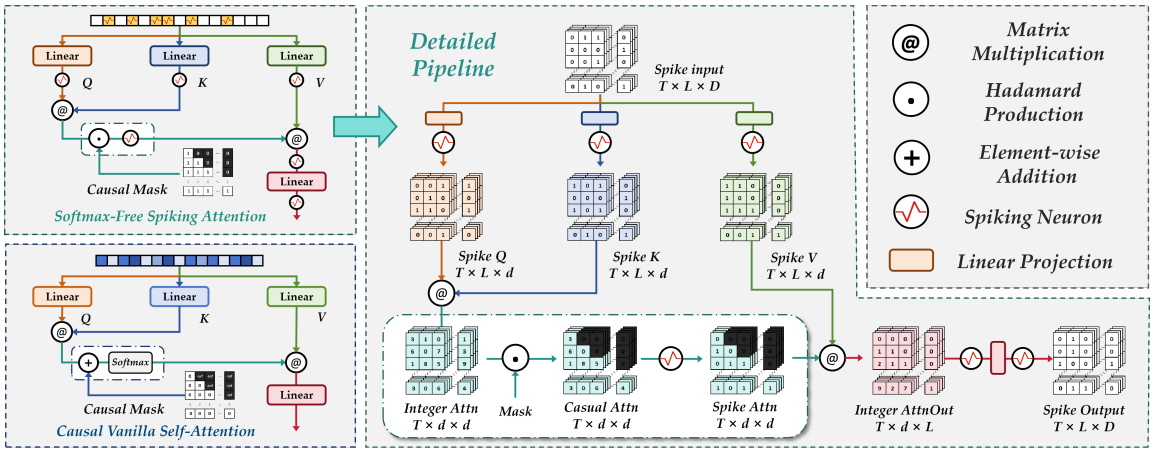


Figure 1: Overview of **Softmax-Free Spiking Attention (SFSA)**. Left: Comparison between Vanilla Causal Self-Attention (CSA) (bottom) and SFSA (top). CSA uses softmax and additive masks, while SFSA employs spike-based activation and binary causal masking. Right: Detailed SFSA pipeline, showing spike-form Q, K, V computation, masked integer attention, spiking activation, and spike-based output, enabling fully discrete and energy-efficient attention modeling.

This challenge directly motivates our **Softmax-Free Spiking Attention (SFSA)**, which enables spike-based language modeling for spiking LLMs.

Moreover, training spiking LLMs introduces additional difficulties beyond those in vision tasks. The inherent temporal dynamics of SNNs already leads to complex computational graphs and high computational cost during backpropagation. Scaling up the architecture further exacerbates this, making full end-to-end training inefficient. Consequently, prior works mostly resort to ANN-to-SNN conversions (Xing et al., 2024a; Schmidgall et al., 2024). However, such methods typically require large time steps to approximate ANN activations, resulting in high inference cost. Integer-based conversions further scale the operations by $T \times N$, which compromises the potential energy benefits of event-driven spiking computation.

To address these challenges, we propose BiSpikCLM, a binary spiking causal language model built on two key components: a spike-driven attention mechanism (SFSA), schematically depicted in Figure 1, and a spike-aware knowledge distillation scheme (SpAD), presented in Figure 3. Overall, our contributions can be summarized as follows:

- We propose **BiSpikCLM**, the first binary spiking MatMul-free causal language model equipped with a fully spike-driven attention mechanism. Our **SFSA** replaces the causal self-attention, which relies on floating-point operations and softmax, enabling efficient autoregressive language modeling with binary spikes. The overall design follows the OPT-family architecture (Zhang et al., 2022), adapted to the spiking domain.
- We introduce **SpAD**, a novel training framework that enables BiSpikCLM to be directly trained from random initialization. SpAD distills hierarchical knowledge covering embeddings, attention maps, intermediate features, and output logits from the teacher model, thereby accelerating convergence and reducing the amount of training data required for large-scale spiking LLMs.
- With only 10B training tokens, significantly fewer than the 180B tokens used to train OPT-1.3B, our SpAD framework enables BiSpikCLM-1.3B to achieve 42.19% zero-shot accuracy on common reasoning benchmarks using 4 time steps, approaching the 49.73% of OPT-1.3B, while consuming just 10.6% of the energy per inference. Remarkably, even at 2 time steps, the model maintains 41.33% accuracy with only 5.88% of the energy cost.

2 Related Work

2.1 SNNs in Downstream Tasks

Recent works show SNNs achieving competitive performance in vision tasks with lower computational consumption. In image classification, advances in surrogate gradients, attention have boosted accuracy and efficiency on CIFAR-10/100 and ImageNet (Rathi et al., 2020; Zhou et al., 2022, 2023, 2024; Li et al., 2024). In object detection,

Model	Params	Spike Form	Training Method	Architecture	Softmax-Free	Exp-Free	FP-Mul-Free
SpikeBERT (Lv et al., 2023)	109M	Binary	Knowledge distillation	Encoder-only	✓	✓	×
SpikingBERT (Bal and Sengupta, 2024)	50M	Binary	Knowledge distillation	Encoder-only	×	×	×
SpikeLM (BERT) (Xing et al., 2024b)	110M	Ternary	Train from scratch	Encoder-only	×	×	×
SpikeLM (BART) (Xing et al., 2024b)	139M	Ternary	Train from scratch	Encoder–Decoder	×	×	×
SpikeLM (mBART) (Xing et al., 2024b)	680M	Ternary	Train from scratch	Encoder–Decoder	×	×	×
SpikeGPT (Zhu et al., 2023)	216M	Binary	Train from scratch	Decoder-only	✓	×	×
SpikeLLM (Xing et al., 2024a)	7-70B	Integer	Post-training quantization	Decoder-only	×	×	×
BiSpikCLM (ours)	1.3B	Binary	SpAD	Decoder-only	✓	✓	✓

Table 1: Comparison of spiking language models. The properties "Softmax-Free", "Exp-Free (exponential-function-free)", and "FP-Mul-Free (floating-point multiplication-free)" are assessed at the attention-module level.

models like SFOD and SpikeYOLO reduce energy cost while closing the gap with ANNs (Su et al., 2023; Bodden et al., 2024; Fan et al., 2024; Luo et al., 2024). For event-based vision, architectures such as MG-SNN effectively handle gesture, motion, and optical flow tasks (Lee et al., 2020; Gehrig et al., 2021; Qiu et al., 2025; Zheng et al., 2025).

In contrast, the application of SNNs in natural language processing (NLP) is still largely underexplored, with only a few attempts adapting language models to spike-based computation. For example, Lv et al. (2023) employs a two-stage distillation pipeline to train an encoder-only spiking student from a pre-trained BERT teacher; similarly, Bal and Sengupta (2024) distills a compact spiking encoder via two-stage KD. However, both still retain floating-point operations and are evaluated at modest scales. Xing et al. (2024b) proposes a spike-driven language model with bi-directional encoding, yet relies on floating-point spikes and retains softmax operations, undermining event-driven efficiency. Zhu et al. (2023) replaces attention with a linear-complexity Spiking RWKV module, but still depends on dense floating-point computation and remains modest in size (216M parameters). Xing et al. (2024a) scales to 7B–70B with the GIF neuron and OBSpiking framework, but substitutes binary spikes with quantized integer signals and retains softmax operation, losing event-driven benefits and fine-grained temporal dynamics. These prior approaches differ substantially in spike representation, training strategy, and especially attention design. A detailed comparison at the attention-module level is provided in Table 1.

2.2 Knowledge Distillation

Knowledge distillation is a widely adopted approach for compressing large-scale language models into smaller, more efficient ones, as demonstrated by models like DistilBERT and TinyBERT (Sanh et al., 2019; Jiao et al., 2019). In the context of SNNs, early distillation efforts have primarily targeted small-scale vision tasks, using spike-based student networks guided by soft targets from ANN teachers (Xu et al., 2023; Qiu et al., 2024; Xu et al., 2024).

In contrast, spike-based distillation for language modeling remains underexplored. Existing methods often overlook the temporal dynamics of SNNs. For example, SpikeBERT (Lv et al., 2023) maps spike activations into continuous representations via an additional MLP. However, this introduces extra trainable parameters and computational overhead, while bypassing the native spike representation, thus limiting the preservation of spike-driven semantics. To address this, we propose the Spike-Aware Alignment Distillation framework, featuring spike-attention and spike-feature alignment modules that enable hierarchical knowledge transfer while preserving the discrete and temporal nature of spiking computation.

3 Methods

We propose BiSpikCLM, a binary spike-based causal language model that integrates a spike-native architectural design with an efficient training paradigm tailored for large-scale SNNs. Specifically, we design a fully spike-driven attention mechanism, Softmax-Free Spiking Attention (SFSA), which replaces conventional softmax-based attention with spike-compatible computation, supporting autoregressive language modeling using binary spikes. Building upon this architecture, we further develop Spike-Aware Alignment Distillation (SpAD), a hierarchical distillation framework that enables stable and scalable training from random initialization by transferring rich supervision signals from a frozen ANN teacher to the SNN student. The overall model architecture is shown in Figure 2, and the training strategy is illustrated in Figure 3.

3.1 Problem Statement

We consider the task of autoregressive generation using a decoder-only causal language model

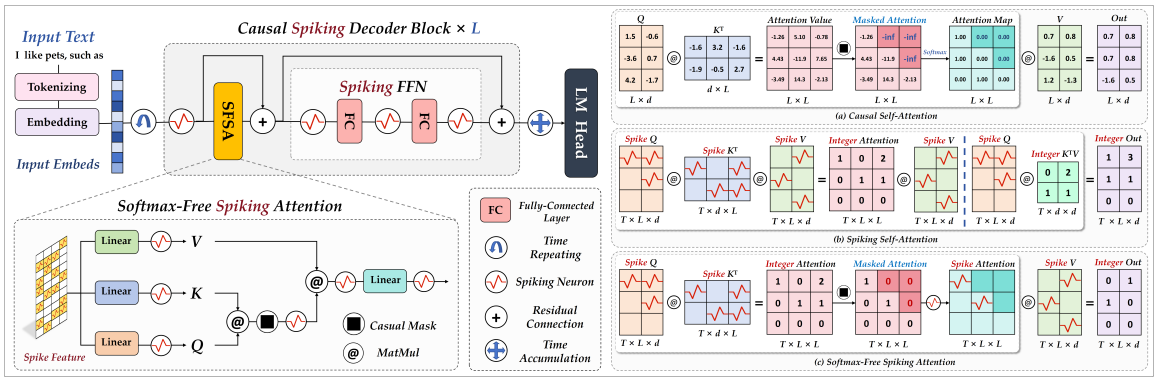


Figure 2: Left depicts the BiSpikCLM framework, detailing the operations of the Softmax-Free Spiking Attention (SFSA) module and the Spiking Feed-Forward Network (SFFN) module. Right compares the computational process of vanilla Causal Self-Attention (CSA), Spiking Self-Attention (SSA), and SFSA, where red spikes represent binary values of 1 and all other values are 0.

(CLM). Formally, given a sequence of tokens $x_1, x_2, \ldots, x_n$, the model is trained to predict the next token x_{n+1} conditioned on the previous n tokens. This can be expressed as maximizing the likelihood $P(x_{n+1} \mid x_1, x_2, \ldots, x_n)$.

During the pre-training stage, the ground-truth label for each autoregressive generation step τ is the token $x_{\tau+1}$, and the model is optimized using the standard cross-entropy loss. The goal is to learn a function that maps token sequences to probability distributions over the vocabulary, employing causal attention under temporal constraints.

3.2 BiSpikCLM Architecture

To enable efficient language modeling with SNNs, we propose BiSpikCLM, which integrates binary spiking neurons with causal attention for softmax-free, energy-efficient computation. Unlike prior works (Zhu et al., 2023; Xing et al., 2024b; Schmidgall et al., 2024; Xing et al., 2024a), BiSpikCLM is fully spike-driven and employs a Hadamard-masked dot product followed by spiking neuron to implement causal attention without softmax. The architecture consists of three main components: (1) Spiking Neuron Modules, (2) Softmax-Free Spiking Attention (SFSA), and (3) Spiking Feed-Forward Network (SFFN). The overall design is built upon the OPT-family architecture (Zhang et al., 2022), chosen for its open-source nature, simplicity, and proven effectiveness.

3.2.1 Spiking Neuron Modules

We primarily instantiate our spike-driven causal language model with the standard Leaky Integrate-and-Fire (LIF) neuron (Wu et al., 2018), which strikes a favorable balance between computational efficiency and modeling capacity. This design results in our core model, **BiSpikCLM**, where neurons emit binary spikes in $\{0, 1\}$. The LIF implementation follows SpikingJelly (Fang et al., 2023), a widely adopted framework for efficient spiking neural networks.

Concretely, the LIF neuron generates a spike $S_t \in \{0, 1\}$ when the membrane potential U_t exceeds a firing threshold U_{thr}, followed by a reset:

$$S_t = \begin{cases} 1, & \text{if } U_t \geq U_{\text{thr}}, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$
$$U_t = I_t + \beta U_{t-1} - S_{t-1} U_{\text{thr}},$$

where $I_t = W X_t$ denotes the input current and β controls the temporal decay of the membrane potential. This mechanism induces sparse, event-driven computation and avoids dense activation patterns in ANN counterparts.

Importantly, our architecture is neuron-agnostic and can be readily extended to different spiking formulations. To demonstrate this flexibility, we further introduce TriSpikCLM, which replaces the binary LIF neuron with a ternary spiking neuron inspired by (Xing et al., 2024b). In this variant, features are extended to ternary values $\{-\alpha, 0, +\alpha\}$, allowing neurons to encode both the sign and magnitude of membrane potential deviations.

Specifically, the ternary neuron produces:

$$S_t = \begin{cases} -\alpha, & \text{if } U_t < -\alpha, \\ 0, & \text{if } |U_t| \leq \alpha, \\ +\alpha, & \text{if } U_t > +\alpha, \end{cases} \tag{2}$$

with the membrane potential updated as:

$$U_t = U_t(\alpha - S_t) + U_{\text{reset}} S_t. \tag{3}$$

While TriSpikCLM provides richer signal representation by relaxing binary constraints, it incurs additional computational overhead and weakens the strict sparsity and event-driven efficiency. We therefore include TriSpikCLM as a comparative extension to assess the trade-off between representational expressiveness and efficiency.

3.2.2 Softmax-Free Spiking Attention

To enable attention mechanisms in spike-based neural networks while preserving computational efficiency, we propose the Softmax-Free Spiking Attention (SFSA) module, presented in Figure 1. SFSA reformulates the self-attention mechanism using spike-based representations, constrained by causality and spiking dynamics.

Specifically, input spike sequences are projected to queries, keys, and values, which are then discretized via spiking neurons. Spike-based dot products between queries and keys yield integer-valued attention scores, followed by a causal mask to ensure autoregressive flow. The masked scores are passed through a spiking neuron to produce sparse attention weights, which are used to compute the weighted sum over value spikes. A final projection and spiking activation generate the output. Details are provided in Appendix B.

3.2.3 Spiking Feed-Forward Network

The Spiking Feed-Forward Network (SFFN) replaces nonlinear activations in the standard Transformer FFN with temporal spiking neurons:

$$\mathcal{FC}(x) = \mathrm{SN}\big(\mathcal{W}x + b\big), \tag{4}$$

$$\mathrm{SFFN}(x) = \mathcal{FC}_2\big(\mathcal{FC}_1(x)\big), \tag{5}$$

where SN denotes a spiking neuron. While binary SFFN preserves strict event-driven sparsity, the ternary extension enables richer signed encoding via multi-level discrete states.

3.3 Spike-Aware Alignment Distillation

To enable effective knowledge transfer from an ANN teacher to an SNN student, we propose Spike-Aware Alignment Distillation (SpAD), consisting of five key components targeting different levels (Figure 3). To address structural mismatches between teacher and student (e.g., embedding size, depth), we employ lightweight alignment techniques such as linear projection and layer skipping.

Among the proposed alignments, we focus on **Spike-Attention Alignment** and **Spike-Feature Alignment** to bridge the core ANN–SNN gap: continuous-valued representations versus discrete, temporally extended spike activities. Detailed formulations are provided in Appendix C.

3.3.1 Spike-Attention Alignment

Prior works (Ji et al., 2021; Wang and Yoon, 2021; Gou et al., 2023) have shown that aligning intermediate representations, especially attention maps and hidden features, is critical for transferring knowledge, beyond matching output distributions alone.

However, directly aligning ANN and SNN attention maps is non-trivial due to two key mismatches. First, the ANN teacher produces continuous-valued attention scores, whereas the SNN student operates with discrete spike activities. Second, the SNN introduces an additional temporal dimension, leading to a dimensional mismatch between teacher and student attention representations. To bridge this gap, we design two alignment strategies.

Temporal Replication and Spike Encoding. We first replicate the attention map $A_{\text{ANN}} \in \mathbb{R}^{L\times L}$ across T simulation steps:

$$\tilde{A}_{\text{ANN}} = \text{Repeat}(A_{\text{ANN}}, T) \in \mathbb{R}^{T\times L\times L}. \tag{6}$$

We then convert $\tilde{A}_{\text{ANN}}$ into spike-form signals using the same spiking neuron as the SNN student, denoted by σ_{spike}. For each attention entry (i, j), we treat $a_{ij} = A_{\text{ANN}}(i, j)$ as a constant input current and run the spiking neuron over T time steps:

$$\begin{aligned} U_t^{(i,j)} &= a_{ij} + \beta U_{t-1}^{(i,j)} - S_{t-1}^{(i,j)} U_{\text{thr}}, \\ S_t^{(i,j)} &= \begin{cases} 1, & \text{if } U_t^{(i,j)} \geq U_{\text{thr}}, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{7}$$

where $\beta \in (0, 1)$ is the leak factor and U_{thr} is the firing threshold.

The spike-form teacher attention is defined as

$$\hat{A}_{\text{spike}}^{\text{ANN}}(t) = \sigma_{\text{spike}}\left(\tilde{A}_{\text{ANN}}(t)\right), \quad t = 1, \ldots, T. \tag{8}$$

This strategy is motivated by rate coding in spiking neurons: larger input currents tend to induce higher firing activities over time. Therefore, spike encoding provides a spike-domain proxy of the ANN attention map, making teacher-student alignment compatible with the temporal spike dynamics of the SNN. We align the time-averaged teacher and student spike attention using Rate-MSE:

$$\mathcal{L}_{\text{attn}}^{\text{RateMSE}} = \text{MSE}\left(\frac{1}{T}\sum_{t=1}^{T} \hat{A}_{\text{spike}}^{\text{ANN}}(t), \frac{1}{T}\sum_{t=1}^{T} A_{\text{SNN}}(t)\right). \tag{9}$$

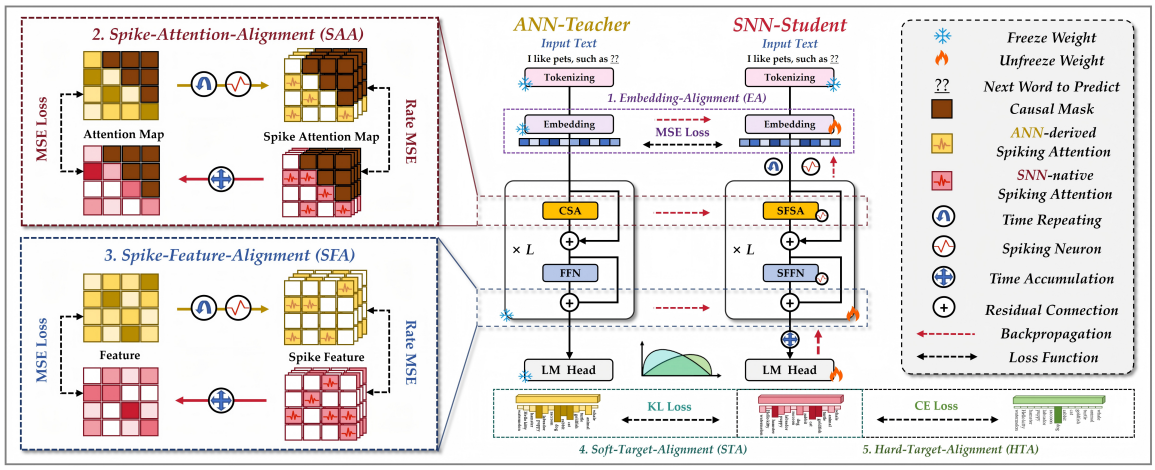


Figure 3: Overview of our **Spike-Aware Alignment Distillation (SpAD)** framework. Knowledge is transferred from a frozen ANN teacher to a trainable SNN student via five alignment modules: (1) **Embedding Alignment** (EA); (2) **Spike-Attention Alignment** (SAA); (3) **Spike-Feature Alignment** (SFA); (4) **Soft-Target Alignment** (STA); and (5) **Hard-Target Alignment** (HTA). Losses include MSE, CE, and spike-aware temporal strategies. In particular, our proposed *Rate-MSE* loss in Eq. (9) aligns the attention dynamics between ANN and SNN models over time. Dashed arrows indicate loss paths; spike-related operations are denoted with icons.

Temporal Fusion and Distribution Matching. Alternatively, we align the overall attention structure by fusing the SNN attention maps. Specifically, we average the SNN attention over time:

$$\bar{A}_{\text{SNN}} = \frac{1}{T}\sum_{t=1}^{T} A_{\text{SNN}}(t). \tag{10}$$

We then match the temporally averaged SNN attention to the static ANN attention map:

$$\mathcal{L}_{\text{attn}}^{\text{MSE}} = \text{MSE}(A_{\text{ANN}}, \bar{A}_{\text{SNN}}). \tag{11}$$

This temporal averaging converts spike activities into a rate-coded representation, which smooths spike noise while preserving the cumulative token-to-token attention structure. Thus, this loss encourages the SNN student to attend to similar semantic cues as the ANN teacher.

The overall spike-attention alignment loss is:

$$\mathcal{L}_{\text{attn}} = \gamma_{\text{attn}}\mathcal{L}_{\text{attn}}^{\text{RateMSE}} + (1-\gamma_{\text{attn}})\,\mathcal{L}_{\text{attn}}^{\text{MSE}}, \tag{12}$$

where $\gamma_{\text{attn}} \in (0,1)$ balances spike-domain alignment and temporally averaged attention matching.

3.3.2 Spike-Feature Alignment

Similarly, intermediate hidden states are aligned using a combination of rate-based and temporally averaged MSE:

$$\mathcal{L}_{\text{feat}} = \gamma_{\text{feat}}\mathcal{L}_{\text{feat}}^{\text{RateMSE}} + (1-\gamma_{\text{feat}})\,\mathcal{L}_{\text{feat}}^{\text{MSE}}, \tag{13}$$

where $\gamma_{\text{feat}} \in (0,1)$ balances the contribution of rate-based and temporally averaged alignment about intermediate features.

Detailed analysis is provided in Appendix C.

4 Experiments

4.1 Experimental Settings

Training details. We use FineWeb-Edu (Penedo et al., 2024), a subset of the FineWeb corpus curated for factual and educational content. We pretrain BiSpikCLM on 1–10B tokens, depending on model scale. The model sizes range from 0.125B to 1.3B parameters. Detailed training settings are provided in Appendix D.

Evaluation benchmarks. We evaluate all models using zero-shot accuracy on a diverse set of commonsense reasoning and question-answering benchmarks, including ARC-Easy (ARC-e), ARC-Challenge (ARC-c) (Clark et al., 2018), Winogrande (WG) (Sakaguchi et al., 2021), BoolQ (BQ) (Clark et al., 2019), PIQA (Bisk et al., 2020), HellaSwag (HS) (Zellers et al., 2019), OpenBookQA (OBQA) (Mihaylov et al., 2018), and HeadQA (HQA) (Vilares and Gómez-Rodríguez, 2019). These tasks measure the generalization and reasoning abilities without task-specific finetuning.

Baselines. We compare BiSpikCLM with scale-matched ANN language models, including Pythia (Biderman et al., 2023) and OPT (Zhang et al., 2022), as well as SpikeGPT (Zhu et al., 2023),

Model	Params (B)	Tokens (B)	Spike Form	Time Step	OPs (G)	Firing Rate	Energy (mJ)	Zero - shot Accuracy (%) ↑								
								ARC-e	ARC-c	WG	BQ	PIQA	HS	OBQA	HQA	Avg.
OPT	0.125	180	×	—	125.6	—	125.95	43.6	19.3	52.3	54.6	62.4	32.1	20.2	23.7	38.60
Pythia	0.160	300	×	—	125.7	—	126.01	43.7	19.8	52.8	55.1	62.7	33.6	20.1	24.2	39.00
SpikeGPT	0.046	16.5	Binary	—	3.66	0.174	3.29	32.3	16.2	50.2	45.7	54.6	25.3	15.7	20.6	32.58
SpikeGPT	0.216	16.5	Binary	—	18.3	0.168	16.53	35.2	17.7	50.7	47.3	55.1	27.6	17.3	23.1	34.25
BiSpikCLM	0.125	**1.0**	Binary	2	12.1	0.196	**5.24**	39.1	18.9	50.3	52.7	56.7	28.1	19.8	22.9	36.05
TriSpikCLM	0.125	**1.0**	Ternary	2	13.7	0.412	**10.74**	38.5	18.3	51.3	52.3	57.7	29.1	19.2	22.5	36.11
BiSpikCLM	0.125	**1.0**	Binary	4	23.1	0.173	**9.43**	39.4	19.0	51.2	53.0	57.5	29.2	19.7	23.1	36.50
TriSpikCLM	0.125	**1.0**	Ternary	4	25.8	0.386	**19.92**	38.9	18.5	51.5	52.9	58.0	28.3	19.2	22.9	36.27
OPT	0.350	180	×	—	360.8	—	197.57	47.5	22.2	55.3	57.2	66.1	40.7	25.7	26.6	42.68
Pythia	0.410	300	×	—	360.9	—	197.71	48.7	24.8	56.8	58.1	66.7	41.6	26.1	26.2	43.63
BiSpikCLM	0.350	**2.0**	Binary	2	43.4	0.182	**9.31**	41.5	21.7	52.3	55.1	59.7	32.7	21.2	23.8	38.48
TriSpikCLM	0.350	**2.0**	Ternary	2	47.7	0.404	**18.61**	41.5	21.7	52.4	54.9	59.1	31.6	20.8	23.1	38.14
BiSpikCLM	0.350	**2.0**	Binary	4	84.2	0.178	**16.75**	42.1	21.4	52.1	56.1	60.5	33.1	21.9	23.5	38.84
TriSpikCLM	0.350	**2.0**	Ternary	4	88.3	0.377	**35.35**	41.8	21.9	52.8	55.7	60.4	34.0	21.3	23.1	38.87
OPT	1.300	180	×	—	1237.1	—	632.22	57.8	30.4	60.4	60.8	71.7	52.6	33.4	30.7	49.73
Pythia	1.400	300	×	—	1237.4	—	632.48	60.5	31.2	61.3	61.1	71.1	53.6	33.2	31.9	50.49
BiSpikCLM	1.300	**10.0**	Binary	2	66.7	0.192	**37.16**	45.7	23.5	54.2	56.3	62.3	40.2	24.5	24.0	41.33
TriSpikCLM	1.300	**10.0**	Ternary	2	74.2	0.426	**74.32**	44.5	23.7	54.2	55.3	62.3	40.4	24.6	23.6	41.08
BiSpikCLM	1.300	**10.0**	Binary	4	131.9	0.184	**66.89**	46.3	24.3	55.6	56.8	63.4	41.7	25.2	24.3	42.19
TriSpikCLM	1.300	**10.0**	Ternary	4	141.6	0.411	**141.21**	45.8	24.5	55.6	56.1	63.4	41.3	25.5	24.8	42.12

Table 2: Comparison of performance and estimated energy efficiency between BiSpikCLM, TriSpikCLM and conventional ANN baselines on the ACC benchmark. BiSpikCLM adopts classic LIF neurons (see Eq. (1)) implemented via SpikingJelly (Fang et al., 2023), while TriSpikCLM employs ternary-valued spiking neurons with amplitude encoding (see Eq. (2)). All energy estimates are calculated under a uniform FP32-based energy model for fair comparison. Time Step indicates the number of discrete simulation steps during SNN inference.

the most comparable decoder-only SNN trained from scratch. We do not directly compare against Lv et al. (2023); Bal and Sengupta (2024); Xing et al. (2024b), as they are encoder-based or target different downstream tasks rather than decoder-only language modeling. Similarly, Xing et al. (2024a) is obtained via quantization and spiking conversion from pretrained ANN LLMs, which differs from our training-from-scratch setting; we provide additional discussion of such conversion-based approaches in Appendix G.

Energy estimation. To evaluate the efficiency of SNNs, we report both the firing rate and the theoretical energy consumption. The firing rate is defined as the average proportion of active spikes, where lower values indicate higher sparsity and potentially lower energy consumption. Following Horowitz (2014); Kundu et al. (2021a); Yin et al. (2021); Kim and Panda (2021), we estimate energy under a 45nm hardware setting by counting spike operations (SOPs) for SNN models and floating-point operations (FLOPs) for ANN baselines. Detailed procedures are provided in Appendix F.

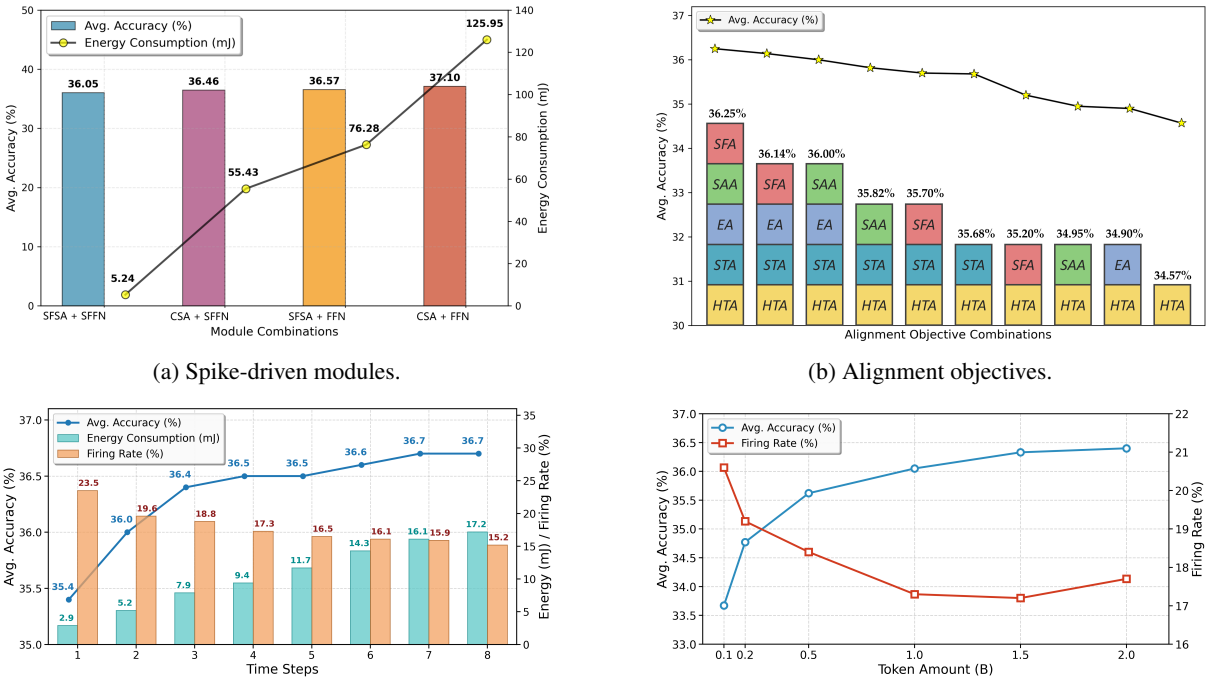
4.2 Main Results

Table 2 summarizes zero-shot accuracy, firing rate, operation count, and estimated per-sample inference energy. Overall, BiSpikCLM achieves a strong accuracy–energy trade-off across model scales. Compared with scale-matched OPT baselines, it retains 83.11–94.56% of the zero-shot accuracy while consuming only **4.16–10.58%** of the inference energy. For instance, BiSpikCLM-1.3B with 4 time steps achieves 42.19% average accuracy versus 49.73% for OPT-1.3B, while reducing energy from 632.22 mJ to 66.89 mJ per sample. BiSpikCLM also outperforms prior decoder-only SNN language models. At the 125M scale, it achieves 36.05–36.50% average accuracy using only 1B training tokens, surpassing SpikeGPT-216M, which reaches 34.25% accuracy after training on 16.5B tokens. Increasing the number of simulation time steps improves accuracy but increases energy consumption, showing a controllable trade-off. TriSpikCLM achieves comparable accuracy to BiSpikCLM, but its higher firing rates lead to substantially higher energy consumption. Thus, binary spiking provides the best overall energy efficiency, while ternary spiking offers an alternative operating point when richer spike representations are desired.

4.3 Ablation Studies

We conduct ablation studies on BiSpikCLM-125M to examine the effects of spike-driven modules, distillation alignment, simulation time steps, and training token volume.

Spike-driven Modules. We replace SFSA and SFFN with their ANN counterparts to assess the



(a) Spike-driven modules.

(b) Alignment objectives.

(c) Varying time steps.

(d) Training data volume.

Figure 4: Visualization of ablation experiments.

contribution of fully spike-driven computation. As shown in Figure 4a, the full spike-driven design achieves 36.05% accuracy with only 5.24 mJ energy. Replacing either module slightly improves accuracy but increases energy by more than 10×, while replacing both modules raises accuracy to 37.10% at a 24× energy cost. These results show that SFSA and SFFN are essential for preserving the energy efficiency of BiSpikCLM.

Distillation Alignment. We evaluate each SpAD component by progressively adding it to the base HTA model (Figure 4b). STA provides the largest individual gain, while EA alone brings a smaller improvement. Higher-level constraints SFA and SAA further improve performance (+0.63% and +0.38%), and their combination yields stronger gains due to complementary supervision signals. The full SpAD objective achieves the best accuracy, improving the base model by 1.68%, which confirms the effectiveness of hierarchical alignment for knowledge transfer.

Time Steps. We investigate simulation steps from 1 to 8 (Figure 4c). Increasing time steps improves accuracy by refining temporal resolution, but gains saturate beyond 4 steps while energy cost rises sharply. Firing rates gradually decline with longer steps, indicating increased sparsity. Overall, 2–4 steps provide a good trade-off between efficiency and performance, while more steps yield marginal accuracy gains at higher energy cost.

Training Token Volume. We study the effect of training data by varying the token budget from 0.1B to 2.0B (Figure 4d). Accuracy improves consistently, with the largest gains in the low-data regime and saturation beyond 1.0B tokens. The accuracy increases from 33.67% to 36.40%, reaching 94.3% of the ANN teacher's performance. We also observe a gradual decrease in firing rate as the token budget increases, suggesting that more data also improves temporal sparsity.

5 Conclusion

We introduce BiSpikCLM, the first fully binary spiking MatMul-free causal language model. Its Softmax-Free Spiking Attention (SFSA) supports spike-based autoregressive modeling, reducing computational cost by over 10× compared to ANNs. The Spike-Aware Alignment Distillation (SpAD) framework further improves performance by aligning representations across multiple levels. We further introduce TriSpikCLM as a ternary extension and comparison baseline. Results show that the fully binary design already achieves competitive accuracy with consistent energy advantages, suggesting a promising pathway for energy-efficient, brain-inspired NLP.

Limitations

While BiSpikCLM significantly reduces computational cost, its accuracy still lags behind large-scale ANNs on benchmarks, and training larger models requires careful tuning of time steps and distillation schedules. Future work could explore improved spike-based architectures and more effective distillation strategies to further close the gap with ANN performance.

Ethical Considerations

This work focuses on improving the energy efficiency of causal language models through binary spiking architectures. We do not introduce new datasets involving personal or sensitive information, nor do we conduct human-subject studies. The proposed method may help reduce the computational and environmental cost of language model deployment. However, as with other language models, BiSpikCLM may inherit biases, toxic content, or other harmful behaviors from its training corpus and ANN teacher models. Future deployment should therefore include appropriate safety evaluation, bias analysis, and usage safeguards.

References

Malyaban Bal and Abhronil Sengupta. 2024. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 10998–11006.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Lennard Bodden, Duc Bach Ha, Franziska Schwaiger, Lars Kreuzberg, and Sven Behnke. 2024. Spiking centernet: A distillation-boosted spiking neural network for object detection. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.

Richard C Bradley. 2005. Basic properties of strong mixing conditions. a survey and some open questions. *Probability Surveys*, 2:107–144.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

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

Paul Doukhan. 1995. Mixing. In *Mixing: Properties and Examples*, pages 15–23. Springer.

Yimeng Fan, Wei Zhang, Changsong Liu, Mingyang Li, and Wenrui Lu. 2024. Sfod: Spiking fusion object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17191–17200.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. 2023. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480.

Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. 2021. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. 2014. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Jianping Gou, Liyuan Sun, Baosheng Yu, Shaohua Wan, and Dacheng Tao. 2023. Hierarchical multi-attention transfer for knowledge distillation. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(2):1–20.

Mark Horowitz. 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*, pages 10–14. IEEE.

Yifan Hu, Yujie Wu, Lei Deng, and Guoqi Li. 2021. Advancing residual learning towards powerful deep spiking neural networks. *arXiv preprint arXiv:2112.08954*, 7:7.

Eugene M Izhikevich. 2003. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572.

Mingi Ji, Byeongho Heo, and Sungrae Park. 2021. Show, attend and distill: Knowledge distillation via attention-based feature matching. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 7945–7952.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Ayush Kaushal, Tejas Vaidhya, Arnab Kumar Mondal, Tejas Pandey, Aaryan Bhagat, and Irina Rish. 2024. Spectra: Surprising effectiveness of pretraining ternary language models at scale. *arXiv preprint arXiv:2407.12327*.

Youngeun Kim and Priyadarshini Panda. 2021. Optimizing deeper spiking neural networks for dynamic vision sensing. *Neural Networks*, 144:686–698.

Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. 2021a. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3953–3962.

Souvik Kundu, Massoud Pedram, and Peter A Beerel. 2021b. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5209–5218.

Chankyu Lee, Adarsh Kumar Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. 2020. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks. In *European conference on computer vision*, pages 366–382. Springer.

David A Levin and Yuval Peres. 2017. *Markov chains and mixing times*, volume 107. American Mathematical Soc.

Yudong Li, Yunlin Lei, and Xu Yang. 2024. Spikeformer: Training high-performance spiking neural network with transformer. *Neurocomputing*, 574:127279.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. 2024. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pages 253–272. Springer.

Changze Lv, Tianlong Li, Jianhan Xu, Chenxi Gu, Zixuan Ling, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2023. Spikebert: A language spikformer learned from bert with knowledge distillation. *arXiv preprint arXiv:2308.15122*.

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

Florence Merlevède, Magda Peligrad, and Emmanuel Rio. 2009. Bernstein inequality and moderate deviations under strong mixing conditions. In *High dimensional probability V: the Luminy volume*, volume 5, pages 273–293. Institute of Mathematical Statistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849. Curran Associates, Inc.

Haonan Qiu, Munan Ning, Zeyin Song, Wei Fang, Yanqi Chen, Tao Sun, Zhengyu Ma, Li Yuan, and Yonghong Tian. 2024. Self-architectural knowledge distillation for spiking neural networks. *Neural Networks*, 178:106475.

Xuerui Qiu, Man Yao, Jieyuan Zhang, Yuhong Chou, Ning Qiao, Shibo Zhou, Bo Xu, and Guoqi Li. 2025. Efficient 3d recognition with event-driven spike sparse convolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20086–20094.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. 2020. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Samuel Schmidgall, Rojin Ziaei, Jascha Achterberg, Louis Kirsch, S Hajiseyedrazi, and Jason Eshraghian. 2024. Brain-inspired learning in artificial neural networks: a review. *APL Machine Learning*, 2(2).

Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. 2022. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Communications of the ACM*, 63(12):54–63.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13693–13696.

Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. 2023. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Joel A. Tropp. 2015. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230.

David Vilares and Carlos Gómez-Rodríguez. 2019. Head-qa: A healthcare dataset for complex reasoning. *arXiv preprint arXiv:1906.04701*.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Lin Wang and Kuk-Jin Yoon. 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331.

Xingrun Xing, Boyan Gao, Zheng Liu, David A Clifton, Shitao Xiao, Wanpeng Zhang, Li Du, Zheng Zhang, Guoqi Li, and Jiajun Zhang. 2024a. Spikellm: Scaling up spiking neural network to large language models via saliency-based spiking. In *The Thirteenth International Conference on Learning Representations*.

Xingrun Xing, Zheng Zhang, Ziyi Ni, Shitao Xiao, Yiming Ju, Siqi Fan, Yequan Wang, Jiajun Zhang, and Guoqi Li. 2024b. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms. *arXiv preprint arXiv:2406.03287*.

Qi Xu, Yaxin Li, Xuanye Fang, Jiangrong Shen, Qiang Zhang, and Gang Pan. 2024. Reversing structural pattern learning with biologically inspired knowledge distillation for spiking neural networks. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3431–3439.

Qi Xu, Yaxin Li, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. 2023. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7886–7895.

Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. 2021. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10221–10230.

Bojian Yin, Federico Corradi, and Sander M Bohté. 2021. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10):905–913.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Jiayi Zheng, Yaping Wan, Xin Yang, Hua Zhong, Minghua Du, and Gang Wang. 2025. Motion feature extraction using magnocellular-inspired spiking neural networks for drone detection. *Frontiers in Computational Neuroscience*, 19:1452203.

Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. 2023. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*.

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. 2024. Qkformer: Hierarchical spiking transformer using qk attention. *Advances in Neural Information Processing Systems*, 37:13074–13098.

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

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

A Artifact and Data Use

We use publicly available scientific artifacts, including the FineWeb-Edu dataset, OPT-family teacher models, Pythia and OPT baselines, SpikeGPT, and standard zero-shot evaluation benchmarks. We cite the original creators of these artifacts in Sections 2 and 4.1. Our use of these artifacts is limited to research purposes, including language model training, evaluation, and comparison, and is consistent with their intended research use. We follow the licenses and terms of use provided by the original artifact creators and do not redistribute raw training data. The FineWeb-Edu 10BT sample is used directly from the public release without additional data collection. Although FineWeb-Edu is curated for factual and educational content, it is derived from web data and may still contain residual personally identifying or offensive content. We therefore do not introduce new sensitive datasets, do not conduct human-subject studies, and report only aggregate model-level results.

B Algorithm Procedure of SFSA

Algorithm 1 Softmax-Free Spiking Attention

Input: Spike-based input X

Output: Spiking attention output

// Step 1: Input Projection ($FP \leftarrow Spike \,@\, FP$)
$q, k, v \leftarrow \text{Linear}_{Q,K,V}(X)$
// Step 2: Spiking Neuron Encoding ($Spike \leftarrow FP$)
$\text{spike}_q \leftarrow SN_Q(q)$
$\text{spike}_k \leftarrow SN_K(k)$
$\text{spike}_v \leftarrow SN_V(v)$
// Step 3: Attention ($Integer \leftarrow Spike \,@\, Spike$)
$\text{attn_int} \leftarrow \text{spike}_q @ \text{spike}_k^T$
// Step 4: Causal Masking and Spiking
$\text{causal_mask} \leftarrow \text{causal_mask} \odot \text{attn_mask}$
$\text{attn_causal} \leftarrow \text{causal_mask} \odot \text{attn_int}$
$spike_{attn} \leftarrow SN_{Attn}(\text{attn_causal})$
// Step 5: Summation ($Integer \leftarrow Spike \,@\, Spike$)
$\text{attn_out} \leftarrow spike_{attn} @ \text{spike}_v^T$
$spike_{attn_out} \leftarrow SN_{AttnOut}(\text{attn_out})$
// Step 6: Output Projection ($FP \leftarrow Spike \,@\, FP$)
$\text{fp_out} \leftarrow \text{Linear}_{\text{out}}(spike_{attn_out})$
// Step 7: Spiking ($Spike \leftarrow FP$)
$\text{spike_out} \leftarrow SN_{Out}(\text{fp_out})$
return spike_out

C Detailed design of SpAD

We present **Spike-Aware Alignment Distillation (SpAD)**, a framework that distills a frozen ANN teacher into a trainable SNN student. SpAD consists of five losses: Embedding Alignment (EA), Spike-Attention Alignment (SAA), Spike-Feature Alignment (SFA), Soft-Target Alignment (STA), and Hard-Target Alignment (HTA). EA/STA/HTA follow standard distillation practice, while SAA and SFA explicitly address the two principal mismatches between ANN and SNN: (i) *continuous vs. spiking* representations and (ii) *static vs. temporal* representations.

Notation. Let T be the number of spiking time steps, L the sequence length, and d the hidden dimension. Teacher (ANN) representations are time-independent, while student (SNN) representations are time-indexed. We denote attention maps by A (shape $L \times L$ for teacher; $T \times L \times L$ for student) and hidden features by H (shape $L \times d$ for teacher; $T \times L \times d$ for student). If teacher and student differ in dimension (or heads/layers), we use lightweight projections and layer/head mapping to align shapes.

C.1 Embedding Alignment (EA)

We align the embedding outputs to ease early optimization. Let $E_{\text{ANN}} \in \mathbb{R}^{L \times d_T}$ and $E_{\text{SNN}}(t) \in \mathbb{R}^{L \times d_S}$ be teacher and student embeddings. We first apply a linear projection $\Pi_{\text{emb}} : \mathbb{R}^{d_S} \to \mathbb{R}^{d_T}$ if $d_S \neq d_T$ and then use temporal fusion:

$$\begin{aligned} \bar{E}_{\text{SNN}} &= \frac{1}{T} \sum_{t=1}^{T} \Pi_{\text{emb}}(E_{\text{SNN}}(t)) \,, \\ \mathcal{L}_{\text{emb}} &= \left\| E_{\text{ANN}} - \bar{E}_{\text{SNN}} \right\|_F^2 . \end{aligned} \tag{14}$$

C.2 Spike-Attention Alignment (SAA)

We align self-attention since it encodes relational structure (token-to-token interactions), which is difficult to transfer via logits alone. Let $A_{\text{ANN}} \in \mathbb{R}^{L \times L}$ be the teacher attention map and $A_{\text{SNN}}(t) \in \{0,1\}^{L \times L}$ be the student spike-attention at time step t.

(a) Temporal Replication and Spike Encoding (Teacher → Student). To align representations in the spiking domain, we construct a spike proxy of the teacher attention. First, we replicate the static teacher attention map across time:

$$\tilde{A}_{\text{ANN}}(t) = A_{\text{ANN}}, \qquad t = 1, \ldots, T. \tag{15}$$

We then apply the spiking neuron σ_{spike} entry-wise to obtain the teacher spiking attention:

$$\hat{A}^{\text{ANN}}_{\text{spike}}(t) = \sigma_{\text{spike}}\Big(\tilde{A}_{\text{ANN}}(t)\Big), \qquad t = 1, \dots, T, \tag{16}$$

and match time-averaged firing rates:

$$\begin{aligned} \bar{A}^{\text{ANN}}_{\text{spike}} &= \frac{1}{T}\sum_{t=1}^{T} \hat{A}^{\text{ANN}}_{\text{spike}}(t), \\ \mathcal{L}^{\text{RateMSE}}_{\text{attn}} &= \big\|\bar{A}^{\text{ANN}}_{\text{spike}} - \bar{A}_{\text{SNN}}\big\|_F^2 . \end{aligned} \tag{17}$$

(b) Temporal Fusion and Distribution Matching (student → teacher). We temporally average the student attention to remove the extra temporal dimension:

$$\begin{aligned} \bar{A}_{\text{SNN}} &= \frac{1}{T}\sum_{t=1}^{T} A_{\text{SNN}}(t), \\ \mathcal{L}^{\text{MSE}}_{\text{attn}} &= \big\|A_{\text{ANN}} - \bar{A}_{\text{SNN}}\big\|_F^2 . \end{aligned} \tag{18}$$

Total attention alignment.

$$\mathcal{L}_{\text{attn}} = \gamma_{\text{attn}}\mathcal{L}^{\text{RateMSE}}_{\text{attn}} + (1-\gamma_{\text{attn}})\,\mathcal{L}^{\text{MSE}}_{\text{attn}}. \tag{19}$$

C.2.1 Theoretical justification

The two strategies can be justified by interpreting spiking attention as a rate-coded random variable, and temporal averaging as an estimator of its expectation.

Lemma 1 (Entry-wise concentration under geometric mixing)**.** *Fix x and an entry (i,j). Let $X_t = A_{\text{SNN}}(t)[i,j] \in \{0,1\}$ and $\mu = \mathbb{E}[X_t \mid x]$. Assume $\{X_t\}_{t\geq 1}$ is (conditionally on x) stationary and geometrically α_{mix} with*

$$\alpha_{\text{mix}}(k) \leq c_0 \exp(-k/\tau_{\text{mix}}) \qquad (k \geq 1), \tag{20}$$

for some constants $c_0 > 0$ and mixing time $\tau_{\text{mix}} \geq 1$ (see, e.g., Doukhan (1995); Bradley (2005)). Then for any $\epsilon > 0$,

$$\Pr(|\bar{X} - \mu| \geq \epsilon \mid x) \;\leq\; 2\exp\Big(-c\,\frac{T\epsilon^2}{\tau_{\text{mix}}}\Big), \tag{21}$$

for a universal constant $c > 0$ (see, e.g., Merlevède et al. (2009)). Moreover,

$$\begin{aligned} \mathbb{E}\big[(\bar{X}-\mu)^2 \mid x\big] &= \mathcal{O}\Big(\frac{\tau_{\text{mix}}}{T}\Big), \\ \mathbb{E}\big[|\bar{X}-\mu| \mid x\big] &= \mathcal{O}\Big(\sqrt{\frac{\tau_{\text{mix}}}{T}}\Big). \end{aligned} \tag{22}$$

The above concentration result applies to any stationary geometrically mixing spike process, and will be instantiated for both the teacher-side spike proxy (method a) and the student-side temporal fusion (method b).

Teacher-side spike proxy is consistent. In the main text, the spiking map σ_{spike} is implemented by driving a leaky integrate-and-fire (LIF) neuron with a constant input current (BiSpikCLM). Accordingly, the teacher spiking attention is *not* an i.i.d. Bernoulli process, but a rate-coded spike train generated by LIF dynamics with reset.

Concretely, for each attention entry (i,j), we treat $A_{\text{ANN}}[i,j]$ as a constant input current to a LIF neuron, and define

$$\hat{A}^{\text{ANN}}_{\text{spike}}(t)[i,j] = \text{LIF}\big(A_{\text{ANN}}[i,j]\big)_t \in \{0,1\}, \quad t = 1, \dots, T. \tag{23}$$

The resulting spike process is stationary after a short transient and admits a well-defined firing rate

$$p_{ij} = \mathbb{E}\big[\hat{A}^{\text{ANN}}_{\text{spike}}(t)[i,j]\big] = g\big(A_{\text{ANN}}[i,j]\big), \tag{24}$$

where $g(\cdot)$ is the implicit rate–current transfer function of the LIF neuron, which is monotone increasing.

Although $\{\hat{A}^{\text{ANN}}_{\text{spike}}(t)[i,j]\}_{t\geq 1}$ exhibits temporal correlations due to membrane integration and reset, the leak factor $\beta < 1$ induces exponential forgetting of past states. Under standard regularity assumptions for LIF dynamics, the spike process is geometrically α_{mix}-mixing with

$$\begin{aligned} \alpha_{\text{mix}}(k) &\leq c_0 \exp(-k/\tau^{\text{ANN}}_{\text{mix}}), \\ \tau^{\text{ANN}}_{\text{mix}} &= \mathcal{O}((1-\beta)^{-1}). \end{aligned} \tag{25}$$

Let

$$\bar{A}^{\text{ANN}}_{\text{spike}} = \frac{1}{T}\sum_{t=1}^{T} \hat{A}^{\text{ANN}}_{\text{spike}}(t). \tag{26}$$

Then, by the same mixing-based concentration argument as in Lemma 1, we have

$$\begin{aligned} \mathbb{E}\big[\bar{A}^{\text{ANN}}_{\text{spike}}\big] &= g(A_{\text{ANN}}), \\ \mathbb{E}\Big[\big\|\bar{A}^{\text{ANN}}_{\text{spike}} - g(A_{\text{ANN}})\big\|_F^2\Big] &= \mathcal{O}\Big(\frac{\tau^{\text{ANN}}_{\text{mix}}\,L^2}{T}\Big). \end{aligned} \tag{27}$$

Therefore, $\bar{A}^{\text{ANN}}_{\text{spike}}$ can be interpreted as a consistent estimator of the rate-coded teacher attention $g(A_{\text{ANN}})$ under geometric mixing. This provides a principled spiking-domain representation of the teacher attention that is directly comparable to temporally averaged student spiking attention.

Student-side temporal fusion concentrates to the rate-coded attention. Fix an input x and an entry (i,j) of the attention matrix. Let

$$X_t = A_{\text{SNN}}(t)[i,j] \in \{0,1\} \tag{28}$$

be the spike at time step t, and define its conditional mean

$$\mu = \mathbb{E}[X_t \mid x]. \tag{29}$$

The temporally averaged spiking attention is

$$\bar{X} = \frac{1}{T}\sum_{t=1}^{T} X_t. \tag{30}$$

For LIF-type dynamics, the leak factor $\beta < 1$ induces exponential forgetting of past states. Under mild regularity, the induced spike process is geometrically mixing with $\tau_{\text{mix}} = \mathcal{O}((1-\beta)^{-1})$, so temporal averaging achieves an effective sample size $T_{\text{eff}} \approx T/\tau_{\text{mix}}$.

Extending to the full attention matrix, define

$$\begin{aligned} \bar{A}_{\text{SNN}} &= \frac{1}{T}\sum_{t=1}^{T} A_{\text{SNN}}(t), \\ A^\star &= \mathbb{E}[A_{\text{SNN}}(t) \mid x]. \end{aligned} \tag{31}$$

Proposition 1 (Matrix concentration and loss consistency under mixing)**.** *Assume each entry process $\{A_{\text{SNN}}(t)[i,j]\}_{t\geq 1}$ satisfies the condition in Lemma 1 with the same τ_{mix} (conditionally on x). Then*

$$\begin{aligned} &\mathbb{E}\big[\|\bar{A}_{\text{SNN}} - A^\star\|_F^2 \mid x\big] \\ &\quad = \sum_{i,j} \mathbb{E}\Big[\big(\bar{A}_{\text{SNN}}[i,j] - A^\star[i,j]\big)^2 \mid x\Big] \\ &\quad = \mathcal{O}\Big(\frac{\tau_{\text{mix}}\, L^2}{T}\Big). \end{aligned} \tag{32}$$

Furthermore, for $\hat{\mathcal{L}} = \|A_{\text{ANN}} - \bar{A}_{\text{SNN}}\|_F^2$ and $\mathcal{L}^\star = \|A_{\text{ANN}} - A^\star\|_F^2$,

$$\begin{aligned} &\mathbb{E}\big[|\hat{\mathcal{L}} - \mathcal{L}^\star| \mid x\big] \\ &\quad \leq 2\|A_{\text{ANN}} - A^\star\|_F\, \mathbb{E}\big[\|\bar{A}_{\text{SNN}} - A^\star\|_F \mid x\big] \\ &\quad + \mathbb{E}\big[\|\bar{A}_{\text{SNN}} - A^\star\|_F^2 \mid x\big]. \end{aligned} \tag{33}$$

which vanishes as $T \to \infty$ provided $T \gg \tau_{\text{mix}}$ (e.g., Tropp (2015); Levin and Peres (2017)).

Alignment of methods (a) and (b). By Proposition 1, $\bar{A}_{\text{SNN}}$ is a consistent estimator of the student rate-coded attention $A^\star$ when $T \gg \tau_{\text{mix}}$. Combined with the teacher-side result that $\bar{A}_{\text{spike}}^{\text{ANN}}$ consistently estimates $g(A_{\text{ANN}})$, the rate-based alignment loss in method (a),

$$\big\|\bar{A}_{\text{spike}}^{\text{ANN}} - \bar{A}_{\text{SNN}}\big\|_F^2, \tag{34}$$

can be viewed as an empirical approximation of the ideal population loss $\|g(A_{\text{ANN}}) - A^\star\|_F^2$. The gap between the two vanishes as $T \gg \max\{\tau_{\text{mix}}, \tau_{\text{mix}}^{\text{ANN}}\}$.

While the above analysis is stated in the asymptotic regime $T \to \infty$, in practice we employ a small number of spiking time steps (e.g., $T = 2$ or 4). Importantly, the Rate-MSE objective does not require exact rate convergence, but only an approximately order-preserving correspondence between input drives and spike activities, i.e., larger attention values induce higher spike probabilities even with few time steps under LIF dynamics. Moreover, stochastic averaging across tokens, attention heads, and training iterations further mitigates the variance introduced by finite-T sampling.

As a result, the mapping $a \mapsto \hat{r}_T(a)$ implemented by the LIF-based spiking map σ_{spike} provides an effective spiking-domain proxy of ANN attention signals. Minimizing the Rate-MSE loss in Eq. (9) therefore encourages the student SNN to match the relative structure of the teacher's attention, bridging method (a) and method (b) in both theory and practice.

C.3 Spike-Feature Alignment (SFA)

We align intermediate features to transfer compositional and hierarchical knowledge. Let $H_{\text{ANN}} \in \mathbb{R}^{L\times d_T}$ be the teacher hidden state at a layer and $H_{\text{SNN}}(t) \in \{0,1\}^{L\times d_S}$ be the student spike feature at time step t. Following prior feature distillation in spiking Transformers (e.g., Lv et al. (2023)), we map teacher/student features into the same content space:

$$\begin{aligned} H'_{\text{ANN}} &= H_{\text{ANN}}, \\ H'_{\text{SNN}} &= \text{LayerNorm}\left(\text{MLP}\left(\frac{1}{T}\sum_{t=1}^{T} H_{\text{SNN}}(t)\right)\right), \end{aligned} \tag{35}$$

where the MLP (or linear projection) matches dimensionality ($d_S \to d_T$) and LayerNorm stabilizes scale. When teacher and student have different depths, we align layers using uniform skipping (e.g., every $\lceil B/M \rceil$ layers when $B > M$). The feature loss combines rate-based and temporally fused MSE:

$$\begin{aligned} \bar{H}_{\text{SNN}} &= \frac{1}{T}\sum_{t=1}^{T} H_{\text{SNN}}(t), \\ \mathcal{L}_{\text{feat}}^{\text{MSE}} &= \big\|H'_{\text{ANN}} - H'_{\text{SNN}}\big\|_F^2, \\ \mathcal{L}_{\text{feat}}^{\text{RateMSE}} &= \big\|g(H_{\text{ANN}}) - \bar{H}_{\text{SNN}}\big\|_F^2, \end{aligned} \tag{36}$$

and

$$\mathcal{L}_{\text{feat}} = \gamma_{\text{feat}} \mathcal{L}_{\text{feat}}^{\text{RateMSE}} + (1 - \gamma_{\text{feat}}) \mathcal{L}_{\text{feat}}^{\text{MSE}}. \quad (37)$$

C.4 Soft-Target Alignment (STA)

We distill output distributions using softened logits:

$$\mathcal{L}_{\text{soft}} = \tau^2 \, \text{KL}\Big(\text{softmax}\Big(\frac{z_{\text{ANN}}}{\tau}\Big) \,\Big\|\, \text{softmax}\Big(\frac{z_{\text{SNN}}}{\tau}\Big)\Big) \quad (38)$$

where z_{ANN} and z_{SNN} are teacher/student logits and τ is the temperature.

C.5 Hard-Target Alignment (HTA)

We include standard cross-entropy with the ground-truth next token y:

$$\mathcal{L}_{\text{hard}} = \text{CE}(\text{softmax}(z_{\text{SNN}}),\, y)\,. \quad (39)$$

C.6 Total Loss

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{emb}} + \lambda_2 \mathcal{L}_{\text{attn}} + \lambda_3 \mathcal{L}_{\text{feat}} + \lambda_4 \mathcal{L}_{\text{soft}} + \lambda_5 \mathcal{L}_{\text{hard}}. \quad (40)$$

Each λ_i balances the corresponding objective. The coefficients $\{\lambda_i\}_{i=1}^{5}$ are listed in Table 3 and satisfy $\sum_{i=1}^{5} \lambda_i = 1$ for a stable loss scale; we set them based on preliminary sweeps, with larger weights on distillation terms improving convergence and accuracy.

D Training Details

Training Paradigm The training of BiSpikCLM follows a teacher–student paradigm, where the teacher model is a pre-trained open-source ANN-based large language model from the OPT family, and the student is our spike-based BiSpikCLM. To ensure consistency between the teacher and student models, we align both the vocabulary and tokenizer with those used in OPT.

Optimization Setup All experiments are conducted using a batch size of 16 and a gradient accumulation factor of 16, effectively yielding a total batch size of 256 tokens. The optimizer used is Adam, with a learning rate set to 5×10^{-4}. A cosine learning rate scheduler with a warm-up ratio of 0.2 is employed to stabilize early training. Gradient clipping is applied with a threshold of 0.7 to avoid exploding gradients in the early training phases, which can be particularly pronounced in spiking models. All models are trained on 8 NVIDIA RTX 4090 GPUs (24GB each).

Spike-Aware Alignment Distillation For Spike-Aware Alignment Distillation (SpAD), we adopt a temperature of $\tau = 2.0$ for the teacher's soft targets. The overall loss is computed as a weighted combination of multiple alignment objectives defined in Method Section. The corresponding loss weights are: $\lambda_1 = 0.2$, $\lambda_2 = 0.1$, $\lambda_3 = 0.1$, $\lambda_4 = 0.3$, and $\lambda_5 = 0.3$. These coefficients are summarized in Table 3. We set $\sum_{i=1}^{5} \lambda_i = 1$ to keep the overall loss scale stable across experiments. The weights are selected based on preliminary sweeps: placing more emphasis on token-level attention/output alignment improves convergence and accuracy, whereas assigning too much weight to auxiliary terms can hurt training stability.

Inference Time Steps To investigate the trade-off between accuracy and energy efficiency, BiSpikCLM is trained with varying numbers of inference time steps, specifically $T = 2$ and $T = 4$. A higher number of steps improves temporal resolution and accuracy at the cost of increased energy consumption, enabling flexible deployment depending on the application constraints.

Training Data Selection The subset of training data used in our experiments was drawn from the fineweb-edu dataset (Penedo et al., 2024), specifically the 10BT sample accessible at `https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu/tree/main/sample/10BT`. Since our claim of using a lower training token volume is a core contribution, it is crucial to detail how the data was sampled to ensure reproducibility. This subset was selected directly from the publicly available sample without additional filtering or preprocessing, providing other researchers with a clear and reproducible training set.

E Surrogate Gradient and Backpropagation Through Time

Surrogate gradient. Training spiking neural networks (SNNs) presents a significant challenge due to the non-differentiable nature of spike generation functions, such as the Heaviside step function used in the spiking neuron model. To enable end-to-end optimization with backpropagation, we adopt a surrogate gradient approach introduced by Fang et al. (2023).

Specifically, the discrete spiking activation S is approximated by a continuous and differentiable

Hyperparameter	Value
Teacher ANN model	OPT-family
Student SNN model	BiSpikCLM
Tokenizer / Vocabulary	Aligned with OPT
Batch size (per GPU)	16
Gradient accumulation steps	16
Effective batch size	256
Optimizer	Adam
Learning rate	5×10^{-4}
Scheduler	Cosine decay
Warm-up ratio	0.2
Gradient clipping threshold	0.7
Temperature τ (for SpAD)	2.0
Distillation weights (λ_1 to λ_5)	0.2, 0.1, 0.1, 0.3, 0.3
Inference time steps (T)	2 and 4
Hardware	8× NVIDIA RTX 4090 (24GB)

Table 3: Summary of training hyperparameters and configurations used for BiSpikCLM, including optimization settings, distillation parameters, and hardware specifications.

function using an arctangent-based surrogate:

$$S(U) \approx \frac{1}{\pi} \arctan\Big(\frac{\pi}{2}\alpha U\Big) + \frac{1}{2}, \qquad (41)$$

where U is the membrane potential and α is a tunable hyperparameter controlling the sharpness of the transition. In our experiments, we set $\alpha = 2$ by default, balancing gradient magnitude and smoothness.

Taking the derivative of Equation (41) yields the surrogate gradient used during backpropagation:

$$\frac{\partial S}{\partial U} = \frac{\alpha}{2} \cdot \frac{1}{1 + \big(\frac{\pi}{2}\alpha U\big)^2}. \qquad (42)$$

This surrogate formulation enables stable and effective gradient-based optimization for BiSpikCLM. It allows error signals to be backpropagated through spike-generating layers without requiring exact gradients, thus making the training pipeline compatible with standard deep learning frameworks.

Boundedness. A useful property of the arctangent surrogate in Eq. (42) is that the gradient is bounded:

$$0 \le \frac{\partial S}{\partial U} \le \frac{\alpha}{2}. \qquad (43)$$

This bounded surrogate derivative mitigates gradient explosion originating from spike nonlinearity and contributes to stable optimization in practice.

E.1 Backpropagation Through Time

We further provide the gradient derivation of spiking neurons trained with backpropagation through time (BPTT), following the standard formulation in spiking deep networks. Let the total loss be a sum of per time step losses:

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_t. \qquad (44)$$

Since the synaptic weights are shared across time, the global gradient is

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial W}. \qquad (45)$$

Cross-time dependency. Although (45) sums per time step contributions, each $\mathcal{L}_t$ depends on the shared weight W *both directly* through the instantaneous synaptic term WX_t and *indirectly* through the recurrent state propagation $U_t \leftarrow U_{t-1} \leftarrow \cdots$. To make this dependence explicit, we can write

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \sum_{j=1}^{t} \frac{\partial \mathcal{L}_t}{\partial W_j} \frac{\partial W_j}{\partial W}, \qquad (46)$$

where W_j denotes the (conceptual) copy of the shared parameter applied at time step j. Since weights are shared across time, $W_1 = \cdots = W_T = W$, hence $\partial W_j / \partial W = 1$ and

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \sum_{j=1}^{t} \frac{\partial \mathcal{L}_t}{\partial W_j}. \qquad (47)$$

The inner sum captures the cross-time influence of W on $\mathcal{L}_t$ via all earlier states, which will be compactly represented by the eligibility trace recursion below.

Neuron dynamics. Consider a discrete-time spiking neuron with membrane potential U_t and spike output S_t. A commonly used update takes the form

$$U_t = \beta U_{t-1} + W X_t, \tag{48}$$

where X_t denotes the pre-synaptic input at time step t, and $\beta \in (0,1)$ is the decay factor. (If a reset term is used, see the remark below.) The spike is generated by a thresholding function $S_t = \mathbf{1}[U_t > \theta]$, whose derivative is approximated by the surrogate gradient in Eq. (42).

Local error term. We derive the gradient of the per time step loss with respect to the shared synaptic weights. By the chain rule, for each time step t,

$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}_t}{\partial S_t} \cdot \frac{\partial S_t}{\partial U_t} \cdot \frac{\partial U_t}{\partial W}. \tag{49}$$

The spiking nonlinearity operates independently on each neuron; therefore, its Jacobian $\frac{\partial S_t}{\partial U_t}$ is diagonal. In practice, this Jacobian-vector product reduces to an element-wise (Hadamard) product. Accordingly, we define the local error term as

$$\delta_t = \frac{\partial \mathcal{L}_t}{\partial S_t} \odot \frac{\partial S_t}{\partial U_t}, \tag{50}$$

where $\odot$ denotes element-wise multiplication and $\frac{\partial S_t}{\partial U_t}$ is given by the surrogate gradient in Eq. (42).

With this definition, the gradient with respect to the shared weights can be compactly written as

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \delta_t \cdot \frac{\partial U_t}{\partial W}. \tag{51}$$

Eligibility trace recursion. A key step in BPTT for SNNs is that $\frac{\partial U_t}{\partial W}$ admits an efficient recurrence. From Eq. (48), we have

$$\frac{\partial U_t}{\partial W} = X_t + \beta \frac{\partial U_{t-1}}{\partial W}. \tag{52}$$

Define the eligibility trace $e_t = \frac{\partial U_t}{\partial W}$ with $e_0 = 0$. Then

$$e_t = X_t + \beta e_{t-1}, \qquad \frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \delta_t \, e_t. \tag{53}$$

This form makes explicit that the gradient is the accumulation over time of a local error term δ_t weighted by the eligibility trace e_t.

E.2 Stability of the eligibility trace

We analyze the stability of the eligibility trace in Eq. (53). Consider the recurrence

$$e_t = X_t + \beta e_{t-1}, \qquad t = 1, \dots, T, \tag{54}$$

where $\beta \in (0,1)$ and $e_0 = 0$. Let $\|\cdot\|$ denote any norm that satisfies (i) the triangle inequality and (ii) positive homogeneity (e.g., ℓ_2 norm for vectors or Frobenius norm for matrices). Assume the pre-synaptic input is uniformly bounded, i.e., $\|X_t\| \le M$ for all t.

Lemma 2 (Uniform bound on e_t)**.** *Under the above assumptions and $\beta \in (0,1)$, the eligibility trace satisfies*

$$\|e_t\| \le \frac{M}{1-\beta}, \qquad \forall t \ge 1. \tag{55}$$

Proof. Unrolling the recursion in (53) gives

$$e_t = \sum_{k=0}^{t-1} \beta^k X_{t-k}. \tag{56}$$

Taking norms on both sides and applying the triangle inequality gives

$$\|e_t\| = \left\| \sum_{k=0}^{t-1} \beta^k X_{t-k} \right\| \le \sum_{k=0}^{t-1} \left\| \beta^k X_{t-k} \right\|. \tag{57}$$

By positive homogeneity of the norm and $\beta^k \ge 0$, we have

$$\begin{aligned} \sum_{k=0}^{t-1} \left\| \beta^k X_{t-k} \right\| &= \sum_{k=0}^{t-1} \beta^k \|X_{t-k}\| \\ &\le M \sum_{k=0}^{t-1} \beta^k, \end{aligned} \tag{58}$$

where the last inequality uses the bounded-input assumption $\|X_{t-k}\| \le M$. Finally, since $\beta \in (0,1)$, the geometric series satisfies

$$\sum_{k=0}^{t-1} \beta^k = \frac{1-\beta^t}{1-\beta} \le \frac{1}{1-\beta}, \tag{59}$$

which concludes the proof of Eq. (55). □

Implication for gradient stability. Using (43), we have $\|\partial S_t / \partial U_t\|_\infty \le \alpha/2$. For any vector norm compatible with element-wise multiplication,

$$\begin{aligned} \|\delta_t\| &= \left\| \frac{\partial \mathcal{L}_t}{\partial S_t} \odot \frac{\partial S_t}{\partial U_t} \right\| \\ &\le \left\| \frac{\partial \mathcal{L}_t}{\partial S_t} \right\| \left\| \frac{\partial S_t}{\partial U_t} \right\|_\infty \\ &\le \frac{\alpha}{2} \left\| \frac{\partial \mathcal{L}_t}{\partial S_t} \right\|. \end{aligned} \tag{60}$$

Combining this with Lemma 2 yields the bound

$$\|\delta_t e_t\| \leq \|\delta_t\| \, \|e_t\| \leq \frac{\alpha M}{2(1-\beta)} \left\| \frac{\partial \mathcal{L}_t}{\partial S_t} \right\|. \quad (61)$$

Therefore, under bounded inputs, the temporal accumulation through the eligibility trace does not introduce gradient explosion by itself; its magnitude is controlled by β and the surrogate slope parameter α.

Remark (with reset). If a soft reset is used, e.g.,

$$U_t = \beta U_{t-1}(1 - S_{t-1}) + W X_t, \quad (62)$$

then the eligibility trace obeys

$$e_t = X_t + \beta(1 - S_{t-1}) e_{t-1}. \quad (63)$$

Since $0 \leq 1 - S_{t-1} \leq 1$, the effective contraction factor is at most β at each time step, yielding an equal or tighter bound than Lemma 2.

F Theoretical Synaptic Operation and Energy Consumption Calculation

The theoretical energy consumption of *BiSpikCLM* is estimated by first calculating the synaptic operations (SOPs). For each block or layer l, we have:

$$\text{SOPs}(l) = f_r(l) \times T \times \text{FLOPs}(l), \qquad (64)$$

where l indexes a block in *BiSpikCLM*, $f_r(l)$ is the average firing rate of the input spike train to block l (measured as spikes per neuron per time step), and T is the simulation time steps of the spiking neuron. $\text{FLOPs}(l)$ denotes the number of multiply-and-accumulate (MAC) operations of block l in the equivalent ANN. $\text{SOPs}(l)$ thus represents the spike-based accumulate (AC) operations performed in the SNN.

Following (Horowitz, 2014), we assume the energy per operation on a 45 nm process as

$$E_{\text{MAC}} = 4.6\,\text{pJ}, \qquad E_{\text{AC}} = 0.9\,\text{pJ}.$$

For ANNs, the theoretical energy consumption of a block b is

$$\text{Power}_{\text{ANN}}(b) = E_{\text{MAC}} \times \text{FLOPs}(b). \qquad (65)$$

For SNNs, the theoretical energy consumption of block b is

$$\text{Power}_{\text{SNN}}(b) = E_{\text{AC}} \times \text{SOPs}(b). \qquad (66)$$

Following (Horowitz, 2014; Kundu et al., 2021a,b; Hu et al., 2021; Yin et al., 2021; Kim and Panda, 2021; Yao et al., 2021), the total energy consumption of *BiSpikCLM* can be decomposed into three parts: (1) the embedding stage, which is executed with dense MAC operations, (2) the L stacked transformer blocks, each of which is spiking and therefore counted using AC operations, and (3) the language-model head that maps hidden states to vocabulary logits (dense MACs). We write:

$$\begin{aligned} E_{\text{BiSpikCLM}} = {} & E_{\text{MAC}} \cdot \big(\text{FLOPs}_{\text{Emb}} + \text{FLOPs}_{\text{Head}}\big) \\ & + E_{\text{AC}} \cdot \sum_{l=1}^{L} \Big(\text{SOP}_{\text{SFSA}}(l) \\ & \qquad + \text{SOP}_{\text{SFFN}}(l)\Big) \qquad (67) \end{aligned}$$

where $\text{FLOPs}_{\text{Embed}}$ and $\text{FLOPs}_{\text{LM-head}}$ denote the MAC operations of the embedding stage and the output projection to vocabulary logits, respectively; $\text{SOP}_{\text{SFSA}}(l)$ and $\text{SOP}_{\text{SFFN}}(l)$ represent the spike-accumulate operations of the Spiking Causal Self-Attention and Spiking Feed-Forward Network modules in block l; E_{MAC} and E_{AC} are the energy costs per MAC and AC operation; L is the number of transformer blocks; and f_r and T denote the average firing rate and the number of simulation time steps.

G Comparison with SpikeLLM

Table 4 compares BiSpikCLM with SpikeLLM across different model scales, quantization methods, spike forms, and simulation time steps. Overall, BiSpikCLM achieves competitive zero-shot accuracy with smaller models and lower-precision spike forms. Notably, the SNN/ANN ratio of BiSpikCLM is consistently higher (82–95%) than that of SpikeLLM, indicating that our spike-based models retain more of the original ANN performance. This improvement is largely attributed to our spike-aware knowledge distillation framework, which effectively transfers information from ANN teachers to spiking students. These results not only highlight the efficiency and effectiveness of our approach but also provide a promising pathway for further improving SNN-LLMs through advanced distillation strategies.

Model	Params (B)	Tokens (B)	Spike Form	Time Step	Avg. Acc.(%) ↑	SNN/ANN Ratio (%) ↑
BiSpikCLM	0.125	**1.0**	Binary	2	36.05	93.39
TriSpikCLM	0.125	**1.0**	Ternary	2	36.11	93.55
BiSpikCLM	0.125	**1.0**	Binary	4	36.50	94.56
TriSpikCLM	0.125	**1.0**	Ternary	4	36.27	93.96
BiSpikCLM	0.350	**2.0**	Binary	2	38.48	90.16
TriSpikCLM	0.350	**2.0**	Ternary	2	38.14	89.36
BiSpikCLM	0.350	**2.0**	Binary	4	38.84	91.00
TriSpikCLM	0.350	**2.0**	Ternary	4	38.87	91.07
BiSpikCLM	1.300	**10.0**	Binary	2	41.33	83.11
TriSpikCLM	1.300	**10.0**	Ternary	2	41.08	82.60
BiSpikCLM	1.300	**10.0**	Binary	4	42.19	84.84
TriSpikCLM	1.300	**10.0**	Ternary	4	42.12	84.70
SpikeLLM	7.000	—	Integer (W2A16)	2	49.92	78.17
SpikeLLM	7.000	—	Integer (W2A8)	4	41.77	65.41
SpikeLLM	13.00	—	Integer (W2A16)	2	53.76	81.34
SpikeLLM	13.00	—	Integer (W2A8)	4	50.12	75.78
SpikeLLM	70.00	—	Integer (W2A16)	2	60.47	82.55

Table 4: Comparison of BiSpikCLM and SpikeLLM across different model scales, neuron/spike formats, and time steps. Avg. Acc. reports zero-shot accuracy (%), and SNN/ANN Ratio shows the performance of spiking models relative to their ANN counterparts.

H Weight Visualization

In this subsection, we analyze the weight distributions of Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs) across different layers and components. The weight visualization provides valuable insights into the fundamental differences between these two types of neural networks and highlights the unique characteristics of SNNs. The weight distributions of ANNs and SNNs exhibit distinct patterns across various layers and components.

Our approach differs from traditional ANN-to-SNN conversion methods in that we do not passively fit SNN weights to match those of ANNs. Instead, we actively capture and adapt to the unique characteristics of SNNs. This active adaptation is crucial for leveraging the full potential of SNNs, which operate on spiking dynamics rather than continuous activation values. By focusing on the inherent properties of SNNs, such as their broader weight distribution and dynamic spiking behavior, our method ensures that the network is optimized for spiking neural computation. This approach allows SNNs to maintain their distinct advantages, such as energy efficiency and biological plausibility, while still achieving high performance.

In summary, the weight visualization demonstrates the differences between ANNs and SNNs. Our method capitalizes on these differences by actively adapting to the unique characteristics of SNNs, rather than forcing them to conform to the weight and activation patterns of ANNs. This approach is essential for developing effective and efficient SNNs that can leverage their spiking dynamics.

I Outlier Analysis of Weight Distributions

To further investigate the differences between Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs), we analyze the number of outliers in the weight distributions across various layers and components. Outliers are defined as weights that significantly deviate from the mean, potentially indicating instability or overparameterization in specific components.

I.1 Total Number of Outliers

As summarized in Table 5, ANNs exhibit significantly more outliers, with over three times as many compared to SNNs across all evaluated layers.

Model	Total Outliers
ANN	18,663
SNN	5,834

Table 5: Total number of weight outliers across all layers and components.

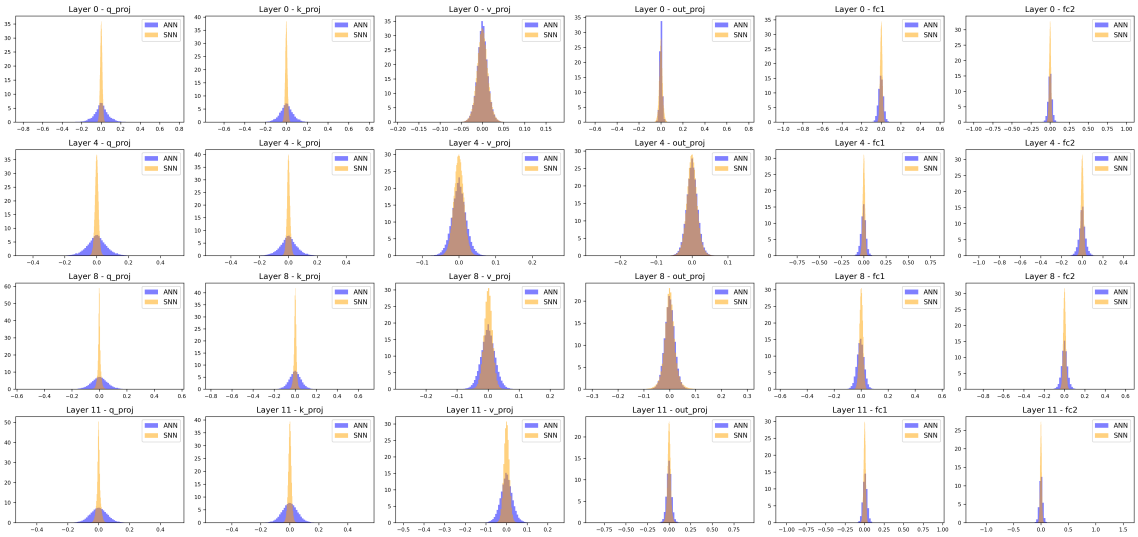


Figure 5: Weight distribution comparison between Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs). ANNs typically exhibit a more concentrated weight distribution around zero, especially in early layers (e.g., Layer 0). In deeper layers (e.g., Layer 11), their weight distribution becomes slightly more spread out but remains relatively compact, indicating tightly clustered weights that contribute to stability and ease of training. In contrast, SNNs show a broader and more dispersed weight distribution, with weights less tightly clustered around zero. This broader spread is particularly notable in deeper layers (e.g., Layer 11), reflecting the dynamic and diverse weight updates characteristic of their spiking nature.

I.2 Layer-wise Comparison

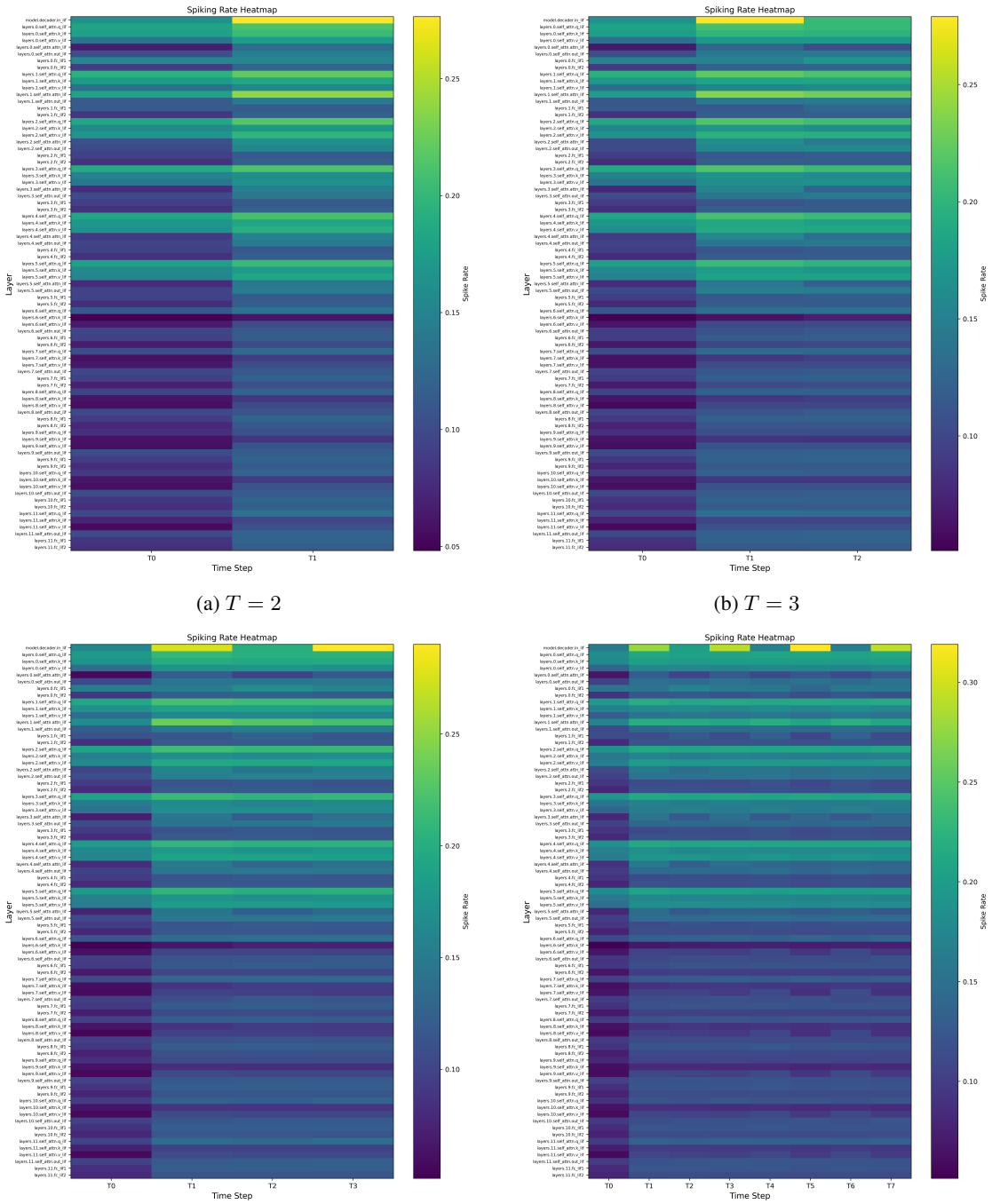
To gain deeper insights, we present a detailed layer-wise and component-wise comparison in Table 6, showing the number of outliers for both models. **ANNs exhibit a significantly higher number of outliers**, especially in deeper fully connected layers (`fc1`, `fc2`), which may be due to larger weight magnitudes and higher variance. While **SNNs show fewer outliers overall**, reflecting their more compact and tightly regulated weight distributions. Interestingly, in some components (e.g., `q_proj`, `k_proj` at Layer 7), SNNs have more outliers than ANNs. This suggests local spikes in weight variability, possibly due to the intrinsic dynamics of spiking updates. The standard deviation of weights (not shown here) is consistently lower in SNNs, reinforcing the observation that they operate within a narrower, more stable range. These findings highlight a fundamental difference in the behavior of ANNs and SNNs: while ANNs may rely on larger weight magnitudes and are more prone to extreme values, SNNs exhibit smoother, biologically plausible weight distributions that reduce the risk of instability.

J Firing Visualization

To investigate the temporal dynamics of spiking activity in BiSpikCLM, we visualize layer-wise firing rate distributions under different inference time steps, namely $T = 2$, $T = 3$, $T = 4$, and $T = 8$, as shown in Fig. 6(a–d). In these heatmaps, the vertical axis corresponds to model layers, the horizontal axis represents discrete time steps, and the color intensity encodes the normalized firing rate from low (purple) to high (yellow).

As the number of time steps increases, the firing patterns exhibit a clear transition from sparse, layer-localized activation to more distributed and pervasive spiking activity. Under a limited temporal budget (e.g., $T = 2$), high firing rates are concentrated in a small subset of layers, indicating that only essential components are activated for early-stage processing. With increased temporal resolution ($T = 3$ and $T = 4$), elevated firing activity gradually propagates across more layers, reflecting enhanced temporal integration and deeper hierarchical interactions. When the temporal budget is sufficiently large ($T = 8$), the firing distribution becomes relatively uniform across most layers, suggesting a temporally saturated regime in which nearly all layers participate in the computation.

Overall, these observations indicate that BiSpikCLM dynamically adapts its internal computation to the available temporal budget, scaling from sparse and efficient activation to more expressive and distributed processing as additional time steps



(a) $T = 2$ (b) $T = 3$

(c) $T = 4$ (d) $T = 8$

Figure 6: Layer-wise firing rate heatmaps under different temporal resolutions. (a) At $T = 2$, firing activity is sparse and concentrated in a small subset of layers. (b) At $T = 3$, elevated firing rates begin to propagate across more layers, indicating increased temporal integration. (c) With $T = 4$, spiking activity becomes more distributed, engaging a larger portion of the network. (d) At $T = 8$, firing patterns are largely uniform across layers, suggesting a temporally saturated regime with broad layer participation.

are provided. This temporal adaptability highlights the potential of spiking neural architectures for scalable and energy-aware language modeling.

While the layer-wise heatmaps reveal how spiking activity propagates across depth, the token-level visualizations in Figures 7–10 further show how temporal integration shapes fine-grained, token-specific activation patterns.

In addition to the layer-wise temporal spiking visualization, we further examine the firing patterns of BiSpikCLM at the level of individual token positions. Figures 7–10 visualize token-wise

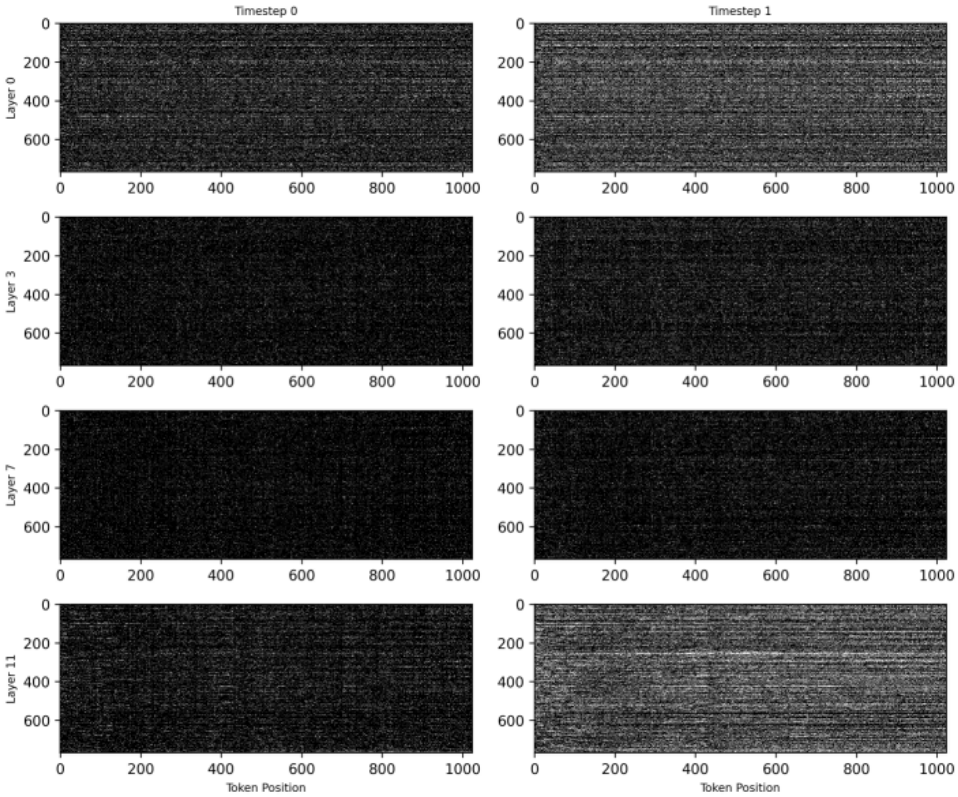


Figure 7: This figure presents the spiking activity across token positions for four representative layers (Layer 0, Layer 3, Layer 7, and Layer 11) at an early inference time step. The firing patterns are relatively sparse and uniformly distributed, particularly in the lower layers. This reflects the initial stage of neuronal processing, where the model begins encoding input signals with limited temporal context. Notably, deeper layers such as Layer 11 exhibit subdued activation, suggesting that higher-level abstractions have not yet emerged.

spiking activity for selected layers (Layer 0, Layer 3, Layer 7, and Layer 11) under different inference time steps. In each figure, rows correspond to the selected layers, while columns represent discrete inference time steps. Within each subplot, the horizontal axis indicates token positions and the vertical axis indexes neurons (or channels) within the corresponding layer; color intensity encodes the firing magnitude, with lighter colors representing stronger activity.

These token-level visualizations provide a more fine-grained view of the internal computational dynamics of BiSpikCLM. At early inference stages (e.g., $T = 2$), spiking activity remains relatively sparse and diffuse across token positions, particularly in intermediate layers, reflecting an initial encoding phase with limited temporal context. As additional time steps are introduced ($T = 3$ and $T = 4$), firing patterns become increasingly structured, with deeper layers exhibiting emerging selectivity over specific token regions, suggesting progressive temporal refinement and hierarchical processing. When the temporal budget is sufficiently large ($T = 8$), deeper layers display pronounced and focused activation patterns, indicating refined internal representations and enhanced contextual integration.

Overall, these results demonstrate that BiSpikCLM dynamically allocates its computational resources across both layers and token positions as inference time increases, transitioning from broad and shallow activation to more selective and semantically rich processing. This behavior highlights the model's capacity for temporally adaptive computation and underscores the suitability of spiking neural architectures for efficient and scalable language modeling.

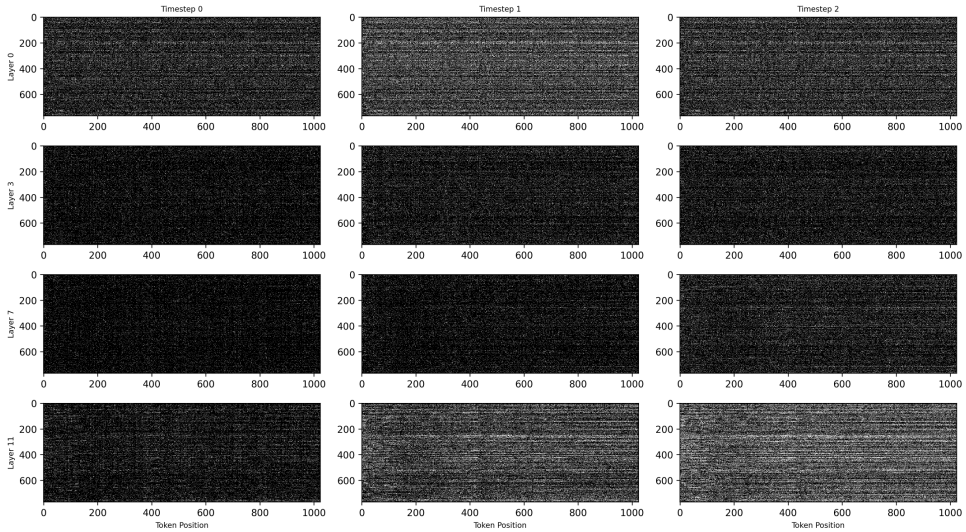


Figure 8: At $T = 3$, the firing distributions become slightly more structured across token positions and layers. While lower layers maintain broadly distributed activity, deeper layers begin to display early signs of selective activation. Compared to $T = 2$, this figure reveals the onset of temporal refinement, indicating that additional time steps allow the model to initiate more context-sensitive computation, particularly in the upper layers.

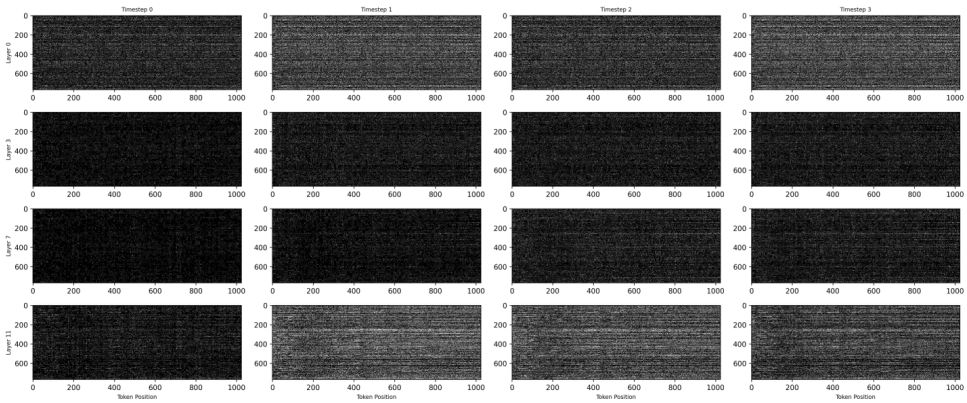


Figure 9: With four time steps, the model exhibits more pronounced spatiotemporal differentiation in firing behavior. Activity becomes more variable across token positions, and certain regions in deeper layers start to display concentrated firing. This suggests that the network is engaging in increasingly specialized processing, distributing its computation more selectively based on both input semantics and accumulated temporal evidence.

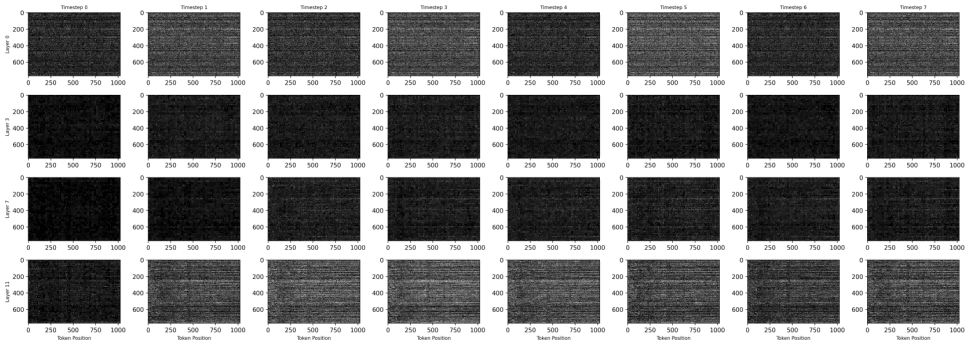


Figure 10: By $T = 8$, the firing patterns exhibit substantial temporal evolution and structural complexity. Deeper layers, in particular, show heightened and more focused activation for specific token regions, reflecting refined internal representations. This level of activity suggests that the model has transitioned into a more stable and semantically rich encoding phase. The marked increase in firing diversity and intensity across layers highlights the model's capacity to utilize extended temporal windows for deeper contextual integration and task-specific computation.

Layer	Component	Model	Mean	Std	Max	Min	Num_Outliers
0	q_proj	ANN	-3.15E-05	7.87E-02	0.77	-0.80	596
0	q_proj	SNN	1.30E-04	1.14E-02	0.06	-0.06	**6**
0	k_proj	ANN	6.12E-05	7.26E-02	0.77	-0.70	1154
0	k_proj	SNN	-3.36E-05	1.10E-02	0.06	-0.06	**24**
0	v_proj	ANN	-1.55E-05	1.28E-02	0.17	-0.19	163
0	v_proj	SNN	2.97E-05	1.31E-02	0.08	-0.08	**34**
0	out_proj	ANN	-8.64E-06	1.30E-02	0.78	-0.62	702
0	out_proj	SNN	-2.16E-04	1.58E-02	0.12	-0.11	**73**
0	fc1	ANN	-3.17E-03	2.93E-02	0.56	-1.00	5657
0	fc1	SNN	4.58E-04	1.16E-02	0.06	-0.06	**2**
0	fc2	ANN	-1.23E-05	2.62E-02	1.00	-1.00	716
0	fc2	SNN	2.74E-04	1.24E-02	0.06	-0.11	**189**
3	q_proj	ANN	-2.93E-04	6.02E-02	0.56	-0.48	105
3	q_proj	SNN	1.98E-04	1.15E-02	0.07	-0.10	**57**
3	k_proj	ANN	1.19E-04	6.16E-02	0.45	-0.46	116
3	k_proj	SNN	1.34E-04	1.10E-02	0.10	-0.10	**199**
3	v_proj	ANN	1.71E-05	2.06E-02	0.23	-0.27	60
3	v_proj	SNN	-2.36E-04	1.39E-02	0.09	-0.09	**63**
3	out_proj	ANN	2.38E-06	1.74E-02	0.50	-0.36	183
3	out_proj	SNN	-2.75E-04	1.48E-02	0.10	-0.10	**79**
3	fc1	ANN	-1.81E-03	2.49E-02	0.86	-0.73	2045
3	fc1	SNN	2.39E-05	1.30E-02	0.07	-0.08	**38**
3	fc2	ANN	-2.44E-06	2.67E-02	0.64	-1.06	2053
3	fc2	SNN	4.87E-05	1.30E-02	0.08	-0.10	**137**
7	q_proj	ANN	-1.80E-04	5.98E-02	0.53	-0.54	147
7	q_proj	SNN	-4.13E-04	1.10E-02	0.13	-0.13	**1484**
7	k_proj	ANN	-2.34E-05	6.03E-02	0.74	-0.68	185
7	k_proj	SNN	5.25E-04	1.35E-02	0.17	-0.19	**1004**
7	v_proj	ANN	1.22E-05	2.02E-02	0.13	-0.14	20
7	v_proj	SNN	-2.11E-04	1.35E-02	0.12	-0.12	**207**
7	out_proj	ANN	1.25E-05	1.76E-02	0.19	-0.19	93
7	out_proj	SNN	-1.91E-04	2.03E-02	0.19	-0.17	**609**
7	fc1	ANN	-4.40E-03	2.73E-02	0.59	-0.59	315
7	fc1	SNN	6.75E-05	1.30E-02	0.15	-0.13	**224**
7	fc2	ANN	-3.44E-05	3.00E-02	0.38	-1.00	1315
7	fc2	SNN	-8.44E-05	1.24E-02	0.10	-0.12	**204**
11	q_proj	ANN	-2.52E-04	5.57E-02	0.50	-0.49	153
11	q_proj	SNN	-1.89E-04	1.19E-02	0.15	-0.12	**181**
11	k_proj	ANN	5.30E-05	5.44E-02	0.53	-0.53	366
11	k_proj	SNN	2.98E-04	1.17E-02	0.19	-0.11	**225**
11	v_proj	ANN	4.60E-05	2.95E-02	0.20	-0.50	68
11	v_proj	SNN	-5.92E-05	1.35E-02	0.24	-0.19	**268**
11	out_proj	ANN	-2.00E-05	3.15E-02	0.89	-0.88	478
11	out_proj	SNN	-1.07E-05	1.96E-02	0.19	-0.16	**325**
11	fc1	ANN	4.87E-03	2.82E-02	0.92	-1.04	473
11	fc1	SNN	-4.00E-05	1.37E-02	0.07	-0.09	**52**
11	fc2	ANN	3.81E-05	3.23E-02	1.56	-1.23	1500
11	fc2	SNN	2.15E-05	1.50E-02	0.12	-0.11	**150**

Table 6: Comparison of weight statistics between ANN and SNN across various layers and components.

K Comparison with Quantized ANNs

To contextualize our contributions, it is crucial to distinguish Spiking Neural Networks (SNNs) from quantized Artificial Neural Networks (ANNs), a common point of comparison. Quantized ANNs achieve efficiency through **spatial discretization**, converting continuous floating-point weights or activations into low-bit, fixed-point formats (e.g., 2-bit, 4-bit). While this approach facilitates model compression and acceleration, the underlying computation remains fundamentally dependent on dense Multiply-Accumulate (MAC) operations. Although 1-bit networks eliminate MACs, they are notoriously difficult to train effectively.

In stark contrast, SNNs leverage **temporal sparse encoding** via binary spikes. A spike (1) or its absence (0) at a given time step encodes information, enabling event-driven and asynchronous computation. This paradigm allows for substantial energy savings on neuromorphic hardware by exploiting sparsity, a benefit difficult for quantized ANNs to replicate (Horowitz, 2014). Our work harnesses this inherent property of SNNs to significantly reduce inference energy while preserving language generation capabilities.

We provide a direct performance comparison between our 1.3B/1.2B BiSpikCLM models and state-of-the-art quantization methods in Table 7. While quantized ANNs like Shao et al. (2023), Wang et al. (2023), and Kaushal et al. (2024) may exhibit a marginal edge in accuracy, we emphasize that these represent fundamentally different methodological paradigms.

Therefore, the primary contribution of our work is not to surpass quantization methods in accuracy, but to pioneer and validate a new, energy-efficient pathway for large language models. Our significance is threefold: (1) We introduce the first train-from-scratch binary-spike-based LLM; (2) We propose the Softmax-Free Spiking Attention (SFSA) mechanism to enable effective causal modeling in SNNs; and (3) We demonstrate the fundamental feasibility of SNN-LLMs, filling a critical gap in the field and establishing a foundation for future research into scalable, energy-conscious language models.

L Extending SFSA to Llama Architecture

To demonstrate the versatility of our proposed SFSA (Softmax-Free Spiking Attention) mechanism, we extend its application beyond the original OPT-based BiSpikCLM architecture. Specifically, we implement SFSA on the Llama architecture (Touvron et al., 2023), training two new models: BiSpikCLM-Llama with 165M and 1.2B parameters. This expansion validates that SFSA is designed fundamentally for spike-based language modeling, independent of specific architectural choices.

The results show consistent performance improvements when increasing time steps from T=2 to T=4 across all model variants. Notably, the Llama-based models achieve comparable or slightly better results than their OPT-based counterparts, particularly in the 1.2B parameter range where BiSpikCLM-Llama (T=4) reaches an average accuracy of 42.33%. This demonstrates that SFSA effectively captures spiking dynamics across different transformer architectures while maintaining competitive language modeling capabilities.

M Effect of Training Scale and Conversational Ability

To validate the scalability of our BiSpikCLM, we conducted experiments to assess the impact of increased training data. Our initial 1B and 10B token training runs were conducted under constrained GPU resources, primarily serving as a proof-of-concept. To further probe the potential of our model, we scaled the training for the 125M and 1.3B models to 5B and 25B tokens, respectively. While this scale is still modest compared to standard pre-training regimens, the use of knowledge distillation from a teacher model allows the student BiSpikCLM to learn more effectively, mitigating the extensive data requirements typically associated with training from scratch.

As shown in Table 9, increasing the training tokens yields significant performance gains. The average accuracy of the 125M model improved by 1.05% when trained on 5B tokens compared to 1B tokens. More impressively, the 1.3B model's average accuracy increased by 3.06% when scaling from 10B to 25B tokens. These results demonstrate that BiSpikCLM is not merely a small-scale proof-of-concept but a model architecture that responds positively and effectively to increased training data, suggesting strong potential for further scaling.

To further investigate the training dynamics, we plot the training loss curves for our models. Figure 11 illustrates the loss progression for the 1.3B model trained on both 10B and 25B tokens. The

Model	Params (B)	Zero-shot Accuracy (%) ↑								
		ARC-e	ARC-c	WG	BQ	PIQA	HS	OBQA	HQA	Avg.
BiSpikCLM (T=2)	1.3	45.7	23.5	54.2	56.3	62.3	40.2	24.5	24.0	41.33
BiSpikCLM (T=4)	1.3	46.3	24.3	55.6	56.8	63.4	41.7	25.2	24.3	42.19
BiSpikCLM-Llama (T=2)	1.2	45.8	23.9	54.7	56.1	63.0	40.6	24.7	24.1	41.61
BiSpikCLM-Llama (T=4)	1.2	46.5	24.4	55.9	56.6	63.6	41.8	25.4	24.4	42.33
BitNet (1.58-bit)	1.3	48.7	24.1	56.8	57.4	64.2	40.6	24.6	24.9	42.66
SmoothQuant (W4A4)	1.3	44.3	23.5	54.2	55.6	62.3	40.0	23.9	23.2	40.88
OmniQuant (W4A4)	1.3	49.7	25.5	58.1	58.3	66.0	41.7	26.2	25.7	43.90
TriLM (1.58-bit)	1.1	46.2	23.9	55.5	56.5	64.1	39.2	24.7	24.5	41.83
TriLM (1.58-bit)	1.5	49.1	25.2	57.3	57.2	64.5	41.1	25.6	25.3	43.16

Table 7: Performance comparison of BiSpikCLM with quantized LLMs.

Model	Params (B)	Time Step	Zero-shot Accuracy (%) ↑								
			ARC-e	ARC-c	WG	BQ	PIQA	HS	OBQA	HQA	Avg.
BiSpikCLM-OPT	0.125	2	39.1	18.9	50.3	52.7	56.7	28.1	19.8	22.9	36.05
BiSpikCLM-OPT	0.125	4	39.4	19.0	51.2	53.0	57.5	29.2	19.7	23.1	36.50
BiSpikCLM-Llama	0.165	2	39.3	19.2	51.3	52.8	56.8	28.3	20.3	23.2	36.40
BiSpikCLM-Llama	0.165	4	39.6	19.6	51.7	53.2	57.3	29.5	20.7	23.7	36.91
BiSpikCLM-OPT	1.300	2	45.7	23.5	54.2	56.3	62.3	40.2	24.5	24.0	41.33
BiSpikCLM-OPT	1.300	4	46.3	24.3	55.6	56.8	63.4	41.7	25.2	24.3	42.19
BiSpikCLM-Llama	1.200	2	45.8	23.9	54.7	56.1	63.0	40.6	24.7	24.1	41.61
BiSpikCLM-Llama	1.200	4	46.5	24.4	55.9	56.6	63.6	41.8	25.4	24.4	42.33

Table 8: Performance comparison of BiSpikCLM models across different architectures and time steps.

Model	Params (B)	Tokens (B)	Zero-shot Accuracy (%) ↑								
			ARC-e	ARC-c	WG	BQ	PIQA	HS	OBQA	HQA	Avg.
BiSpikCLM (T=2)	0.125	1.0	39.1	18.9	50.3	52.7	56.7	28.1	19.8	22.9	36.05
BiSpikCLM (T=2)	0.125	5.0	41.4	19.2	51.4	53.4	58.2	30.2	19.9	23.1	37.10
BiSpikCLM (T=2)	1.300	10.0	45.7	23.5	54.2	56.3	62.3	40.2	24.5	24.0	41.33
BiSpikCLM (T=2)	1.300	25.0	48.3	26.4	57.8	58.6	65.1	44.9	27.3	26.7	44.39

Table 9: Performance of BiSpikCLM with varying training token counts.

curves exhibit a smooth and consistent downward trend, indicating that our training process is stable and converges effectively. Notably, the model trained on 25B tokens continues to decrease its loss to a lower final value, corroborating the quantitative performance gains observed in Table 9. This stable convergence behavior across different training scales demonstrates the robustness of our proposed BiSpikCLM architecture and the effectiveness of the SFSA mechanism in facilitating the optimization of spike-based language models.

Beyond quantitative metrics, we qualitatively evaluated the conversational abilities of our BiSpikCLM-1.3B model trained with 10B and 25B tokens. Table 10 presents sample responses to a set of prompts. The 10B model exhibits basic conversational skills and can maintain a simple dialogue. However, it fails to answer factual questions correctly, likely due to a lack of exposure to that specific knowledge during its limited training. In contrast, the 25B model provides more accurate and detailed responses, correctly answering the factual question about the U.S. capital and generating more coherent and contextually relevant text. This qualitative improvement aligns with the quantitative gains, underscoring the importance of training scale for developing the factual knowledge and reasoning capabilities of spiking language models.

N Analysis of Long-Context Dependency and Sparsity

In the Softmax-Free Spiking Attention (SFSA) mechanism, information propagation relies on spike-triggered updates of the attention state. Consequently, a higher sparsity level, achieved by increasing the firing threshold or reducing the number of time steps, can theoretically weaken the trans-

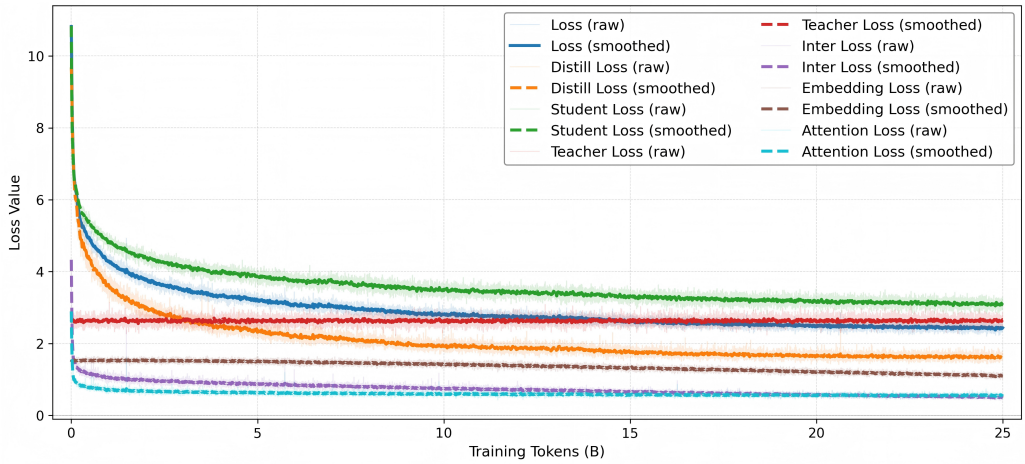


Figure 11: Training loss curves for the BiSpikCLM-1.3B model, comparing the 10B and 25B token training regimes. The smooth downward trend confirms stable convergence.

Prompt	**BiSpikCLM (10B Tokens)**	**BiSpikCLM (25B Tokens)**
Hi there! How are you doing today?	Hi! How are you? I'm doing well. How about you?	I'm doing great! I'm just tired as hell. I went to bed at 11pm last night, and now I'm up at 4am. I have a class tomorrow morning at 7:30, so I have to get up and go to class.
Where is the capital of the United States?	It's in the middle of the country.	Washington, D.C.
How about the cat?	I think it's a good idea to have a cat in the house.	I have a cat and he doesn't like the camera. I'll send you a picture when he is in the mood.

Table 10: Comparison of conversational abilities for BiSpikCLM-1.3B trained on different token counts.

Model	**Params (B)**	**Firing Threshold**	**Context Length** 512	1024	2048	4096	8192
OPT	1.300	-	16.26	13.58	11.13	-	-
Llama-3.2	1.200	-	12.93	10.96	9.76	9.02	8.54
BiSpikCLM	1.300	0.70	36.72 (0.184)	32.49 (0.183)	29.34 (0.181)	-	-
		0.85	39.75 (0.177)	36.30 (0.175)	32.18 (0.174)	-	-
		1.00	43.17 (0.163)	39.88 (0.162)	34.62 (0.160)	-	-
BiSpikCLM-Llama	1.200	0.70	33.53 (0.198)	29.82 (0.195)	27.59 (0.194)	25.32 (0.192)	23.77 (0.190)
		0.85	37.14 (0.186)	33.37 (0.185)	31.26 (0.185)	28.89 (0.183)	25.42 (0.181)
		1.00	40.88 (0.174)	37.25 (0.172)	34.08 (0.171)	30.76 (0.171)	27.19 (0.168)

Table 11: Perplexity (PPL) on WikiText-2 with varying context lengths and firing thresholds. The firing rate (sparsity) is shown in parentheses.

mission of long-range signals, potentially impairing the model's ability to model long-distance dependencies.

To empirically evaluate this, we conducted a sliding-window evaluation on the WikiText-2 dataset. We compared our BiSpikCLM-1.3B (T=4) and BiSpikCLM-Llama-1.2B (T=4) models with varying firing thresholds against the full-precision OPT-1.3B and Llama-3.2-1.2B baselines. The models were tested with context lengths of 512, 1024, 2048, 4096, and 8192 tokens, and the performance was measured using Perplexity (PPL). For reference, the maximum supported context length for OPT-1.3B is 2048. The results are presented in Table 11.

The results lead to several key observations. First, as the sparsity increases (i.e., the threshold rises from 0.70 to 1.00), the PPL performance degrades across all context lengths. This confirms our intuition that higher sparsity can impede the flow

of information. However, the magnitude of this degradation is moderate, suggesting that increasing sparsity within a certain range does not cause a catastrophic collapse in long-context dependency handling.

And more intriguingly, for a fixed threshold, the PPL consistently decreases as the context length increases. For instance, the BiSpikCLM-Llama model with a threshold of 0.70 improves from a PPL of 33.53 at 512 tokens to 23.77 at 8192 tokens. This indicates that the natural increase in sparsity caused by longer sequences has a negligible negative impact on the model's capability. The model retains its ability to extract global information from longer contexts.

Finally, we acknowledge that the current design of SFSA lacks specialized mechanisms for handling extremely long contexts. We are actively working to address this limitation through future research directions, such as adaptive spike scheduling and the integration of long-term memory neurons. We believe these are engineering frontier challenges rather than fundamental obstacles and can be progressively overcome in future work.