SpikeLM: Towards General Spike-Driven Language Modeling via Elastic Bi-Spiking Mechanisms

Xingrun Xing [1 2 3] Zheng Zhang [3] Ziyi Ni [1 2] Shitao Xiao [3] Yiming Ju [3] Siqi Fan [3] Yequan Wang [3]
Jiajun Zhang [1 2] Guoqi Li [1]

Abstract

Towards energy-efficient artificial intelligence similar to the human brain, the bio-inspired spiking neural networks (SNNs) have advantages of biological plausibility, event-driven sparsity, and binary activation. Recently, large-scale language models exhibit promising generalization capability, making it a valuable issue to explore more general spike-driven models. However, the binary spikes in existing SNNs fail to encode adequate semantic information, placing technological challenges for generalization. This work proposes the first fully spiking mechanism for general language tasks, including both discriminative and generative ones. Different from previous spikes with {0,1} levels, we propose a more general spike formulation with bi-directional, elastic amplitude, and elastic frequency encoding, while still maintaining the addition nature of SNNs. In a single time step, the spike is enhanced by direction and amplitude information; in spike frequency, a strategy to control spike firing rate is well designed. We plug this elastic bi-spiking mechanism in language modeling, named SpikeLM. It is the first time to handle general language tasks with fully spike-driven models, which achieve much higher accuracy than previously possible. SpikeLM also greatly bridges the performance gap between SNNs and ANNs in language modeling. Our code is available at https://github.com/Xingrun-Xing/SpikeLM.

[1]Institute of Automation, Chinese Academy of Sciences [2]School of Artificial Intelligence, University of Chinese Academy of Sciences [3]Beijing Academy of Artificial Intelligence. Correspondence to: Zheng Zhang <zhangz.goal@gmail.com>, Jiajun Zhang <jjzhang@nlpr.ia.ac.cn>, Guoqi Li <guoqi.li@ia.ac.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Creating artificial general intelligence by simulating the human brain has always been a human dream, which is known as Brain-Inspired Computing (BIC) (Mehonic & Kenyon, 2022; Zhang et al., 2020b). Although artificial neural networks (ANNs) (Touvron et al., 2023; Kirillov et al., 2023) have achieved tremendous success, the working ways are still so different from the human brain. The biological neurons communicate with spikes (Roy et al., 2019) and only activate when the membrane potential exceeds a certain threshold. Spiking neural networks (SNNs) (Maass, 1997) are designed by simulating biological neuron dynamics (Gerstner et al., 2014), and have distinctive attributions of biological plausiblity, event-driven sparsity, and binary activation. Given event-driven computation, high sparsity is dynamically achieved by event occurrence. Given binary activations, matrix multiplications convert to accumulate (AC) operations. These characteristics make bio-inspired SNNs a significantly energy-efficient alternative (Yin et al., 2021; Schuman et al., 2022) to traditional ANNs. Deepening our understanding of spiking neurons (Fang et al., 2021b) and expanding the usage scope of SNNs (Kim et al., 2020; Zhou et al., 2024) have become increasingly valuable issues.

Previous SNNs mainly focus on computer vision (Wu et al., 2021; Hu et al., 2021) due to relatively simple tasks and smaller model sizes. Recently, large-scale language models (Touvron et al., 2023; Du et al., 2022) exhibit much more advanced generalization ability (Brown et al., 2020) than other fields in machine learning, which motivates us to pursue more general spike-driven models with language modeling. However, this objective is no-trial due to the technological challenges in spike representation (Deng & Gu, 2020; Guo et al., 2023b) and optimization (Neftci et al., 2019; Wu et al., 2018; Guo et al., 2023c). In representation, binary spike leads to severe information loss (Deng & Gu, 2020), making it difficult to generalize across language tasks. In optimization, large-scale language models require stable and highly efficient gradient calculation, while neuronal dynamics in SNNs are non-differentiable. Therefore, there are very limited language-oriented SNNs.

This work focuses on fully spike-driven language modeling

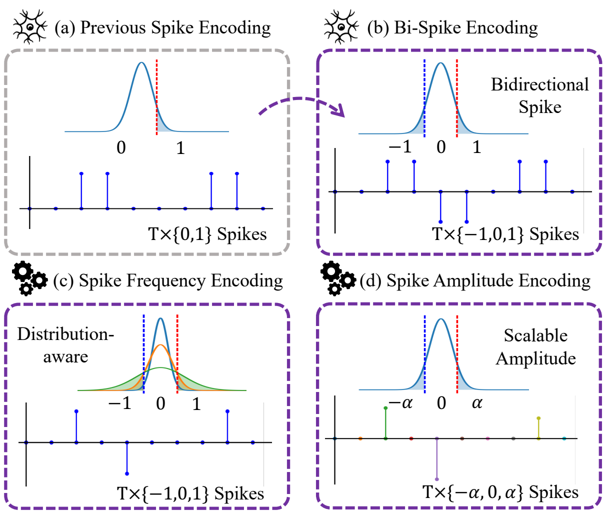


Figure 1. Comparisons between previous spike encoding (a) and our elastic bidirectional encodings (b, c, d). The bidirectional, frequency and amplitude encodings are sequentially applied .

in general tasks, including both discriminative and generative ones, which is not addressed in the previous SNN studies. Notably, fully spike-driven indicates replacing all matrix multiplications as spike operations, except the last regression. To explore the capabilities and limitations of current SNNs, we initially apply existing SNN technologies (Hu et al., 2021; Gerstner et al., 2014) to construct fully spike-driven baselines. Basically, there is a large performance gap between language-oriented ANNs and SNNs. Moreover, the fixed spike firing rate in SNNs makes a suboptimal trade-off between performance and energy efficiency.

To address the aforementioned issues, we focus on boosting modeling capabilities of SNNs through generalized spike encoding methods. To extend semantic information, we sequentially generalize spike formulations as shown in Fig.1:
(i) Bi-directional spike encoding. Different from previous binary spike levels $\{0, 1\}$, we propose bidirectional spikes with ternary levels $\{-1, 0, 1\}$. Bidirectional encoding doubles semantic information and maintains the addition nature of SNNs at the same time.
(ii) Elastic spike frequency encoding. Different from previous empirical spike firing rates, we encode spike frequency according to input distributions, achieving a controllable firing rate for better performance and energy trade-off.
(iii) Elastic spike amplitude encoding. To retain membrane potential intensity, we encode spike with amplitude information as $\{-\alpha, 0, \alpha\}$. A layerwise α is used, which can be merged with weights after training. Therefore, the addition nature of SNNs is still maintained.
Given a multi-step spike, these encoding methods jointly extend spike capabilities by direction and amplitude in each time step and frequency across time steps. We plug this elastic bi-spiking mechanism in language modeling, termed

Table 1. Comparisons with SpikeBERT (Lv et al., 2023) and SpikeGPT (Zhu et al., 2023). The "+" sign indicates the level of capability, with SpikeGPT only utilized for basic language modeling, lacking applications in sentence-level generation tasks. AC and MAC indicate ACcumulate and Multiply-ACcumulate.

Task/Operation	SpikeBERT	SpikeGPT	SpikeLM
Discrimination	+++	+++	+++
Generation	−	+	+++
Spike-Driven Matrix Mul.	Fully ACs	Partly MACs	Fully ACs

Table 2. Comparisons between ANN and SNNs in generative and discriminative tasks in the same BERT or BART architecture.

Dataset	ANN	LIF-SNN	SpikeLM
MNLI$_{\text{-m/mm}}$ (Acc.)	83.8/83.4	56.8/55.2	77.1/77.2
MRPC (F1)	89.8	82.3	85.7
STS-2 (Acc)	92.3	80.6	87.0
RTE (Acc.)	69.3	53.8	69.0
STS-B (SP.)	89.4	20.0	84.9
XSUM (R-L)	34.7	28.3	32.9
CNN-DM (R-L)	31.7	28.1	29.1
WMT16 (BLEU)	26.8	19.0	23.0
Average	66.8	47.1	62.9

SpikeLM. Thanks to improved spikes, as Table 1, it achieves the first fully spiking mechanism in general language tasks, by replacing all matrix multiplications in ANNs. Our contributions are summarised as follows:

- We propose SpikeLM, the first general fully spike-driven language modeling, significantly broadening the usage scope of language-oriented SNNs. As Table 2, SpikeLM achieves much higher accuracy than what was possible previously and largely bridges the performance gap between SNNs and ANNs.
- We propose an elastic bi-spiking mechanism. At the same time, it maintains the addition nature of SNNs. A controllable spike firing rate is also achieved.
- We introduce the dynamic isometry (Chen et al., 2020), and theoretically prove that the training stability of the elastic bi-spiking function surpasses the ReLU function in ANNs, ensuring stable optimization for SpikeLMs.

2. Related Work

Bio-inspired SNNs. BIC field (Mehonic & Kenyon, 2022; Zhang et al., 2020b) is boosted by both advanced neuroscience and deep learning. Recent BIC field gets inspiration from learning rules (Payeur et al., 2021), structures (Pham et al., 2021), and energy-efficient computation (Schuman et al., 2022; Yao et al., 2023b) in the nervous system. As a BIC algorithm, SNN (Maass, 1997) also takes advantages of deep learning, for example, the spike-driven residual learning (Fang et al., 2021a), normalization (Zheng et al., 2021; Guo et al., 2023d), self-attention (Yao et al., 2023a; Zhou et al., 2023; Yao et al., 2024a), backpropagation (Meng et al., 2022; Li et al., 2021; Su et al., 2023; Guo et al., 2023a), and ANN-SNN conversion (Bu et al., 2021; Deng & Gu, 2020) technologies. One of the recent works also introduces ternary spikes (Guo et al., 2024) in the computer vision field, while this work is in parallel with it and has different frequency and amplitude encoding to reduce average firing rate in language tasks. Inspired by generalization capability in both the spike-driven human brain (Gerstner et al., 2014; Izhikevich, 2003) and recent large language models (Touvron et al., 2023; Brown et al., 2020), we are the first time to explore fully spike-driven models in general language tasks.

Neuromorphic chips. Neuromorphic chips are inspired by the brain with non-von Neumann architectures (Yao et al., 2024b; Schuman et al., 2022; Roy et al., 2019; Merolla et al., 2014). Owning the high sparsity and event-driven SNNs, their energy consumption can be tens to hundreds of mWs (Basu et al., 2022) in SNNs workloads by compute gating or clock gating techniques (Narayanan et al., 2020).

Language-oriented SNNs. SpikeBERT (Lv et al., 2023) distills the spikingformer (Zhou et al., 2023) in some discriminative tasks. However, performance drops to 59.7% on the GLUE. SpikeGPT (Zhu et al., 2023) introduces spike propagation between transformer blocks, but overall blocks are still ANNs. Compared with recent weight-quantized language models, BitNet (Wang et al., 2023) and ternary BitNet (Ma et al., 2024), SNNs more concentrate on bio-plausible activation spike encoding.

3. Problem Formulation

3.1. Language Modeling with Vanilla SNNs

We start by developing the first general baseline for fully spike-driven language modeling. Without loss of generality, we apply the most popular Leaky Integrate-and-Fire (LIF) neurons (Gerstner et al., 2014) to encode real-valued activations into spike sequences. Fully spike-driven transformers are achieved through LIF neurons in linear layers and the key and value of self-attention (Vaswani et al., 2017).

Spike encoding in linear. LIF neurons are neuronal dynamics added before linear layers, which output binary spikes with $\{0, 1\}$ levels. The following matrix multiplication is converted to additions. By simulating the charging and firing of biological neurons, LIF neurons can be governed by:

$$\boldsymbol{m}^l(t) = \boldsymbol{v}^l(t-1) + \boldsymbol{x}^{l-1}(t), \tag{1}$$

$$\boldsymbol{s}^l(t) = \begin{cases} 0, & \text{if } \boldsymbol{m}^l(t) < \boldsymbol{\theta}^l \\ 1, & \text{if } \boldsymbol{m}^l(t) \geq \boldsymbol{\theta}^l \end{cases}, \tag{2}$$

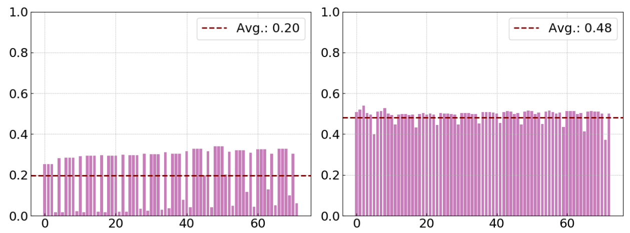


Figure 2. Spike firing rate in LIF-BERT (left) and activated rate in Binary BERT (right) in every linear layer.

$$\boldsymbol{v}^l(t) = \beta \boldsymbol{m}^l(t)(1 - \boldsymbol{s}^l(t)) + v_{reset}\boldsymbol{s}^l(t). \tag{3}$$

At each time step t, the LIF neuron performs a spike encoding until a certain spike length T. $\boldsymbol{m}^l(t)$ and $\boldsymbol{v}^l(t)$ indicate the membrane potential before and after spike encoding respectively. To simulate the charging process, $\boldsymbol{m}^l(t)$ adds the inputs $\boldsymbol{x}^{l-1}(t)$ at the current moment to the membrane potential $\boldsymbol{v}^l(t-1)$ from the last moment. When the membrane potential $\boldsymbol{m}^l(t)$ exceeds the firing threshold $\boldsymbol{\theta}^l$, the neuron is triggered and the spike $\boldsymbol{s}^l(t)$ is encoded as 1; otherwise, it is 0. After spike encoding, the membrane potential $\boldsymbol{v}^l(t)$ is reset to a certain potential v_{reset} if the spike is 1; otherwise, it will decay by a factor $\beta(< 1)$.

Spike encoding in the key and value. The matrix multiplications in self-attention include the multiplication between the key and query, and the multiplication between the attention map and value. By encoding the key and value as spikes by Eq.1,2,3, all matrix multiplications are converted to additions. We set $\beta = 0$ in key and value. Notably, the time step of SNNs is an additional dimension.

We construct LIF-based transformers for both BERT (Devlin et al., 2018) and BART (Lewis et al., 2019) architectures, termed LIF-BERT and LIF-BART, for discriminative and generative tasks respectively. For optimization, we propose a straight-through estimator (STE) (Bengio et al., 2013) based backpropagation in Appendix A.1, which achieves a strong baseline in general language tasks.

3.2. Performance & Energy Efficiency in SNNs

Previous SNNs directly encode spikes, leading to an ill-posed problem: when the spike firing rate is low, it leads to a reduced information entropy in the Bernoulli-distributed spikes, limiting model capability. When the spike firing rate is high, it decreases the sparsity of the spikes, resulting in increased energy consumption.

Performance drop. As shown in Table 2, compared with ANNs, the LIF-based SNNs are driven by sparse binary spike, resulting in the average 19.7% performance drop. We analyze the spike firing rate in LIF-BERT in Fig.2. Although the LIF-BERT has a low firing rate, the sparse and binary encoded spike is too simple without much capability to represent semantic information.

Energy efficiency. We consider a case of the high firing rate. We directly replace the LIF neuron with a binary quantization function for one-step spike encoding following BiPFT (Xing et al., 2024), which is a Binary BERT with the {-1, +1} binarization level. For binary neural networks (BNNs) (Courbariaux et al., 2016; Xing et al., 2022), the binary activations can map to $\{0, 1\}$ in inference. As shown in Fig.2, the proportion of 1 in Binary-BERT is close to 50%. Compared with SNNs, BNNs demonstrate a much higher activation rate. With equally probable 0 and 1, information entropy in the Bernoulli distribution approaches maximum. However, the reduced sparsity leads to significantly increased energy consumption.

4. General Spike Language Modeling

We divide and conquer the problem of effective spike encoding into three aspects: spike direction encoding, spike frequency encoding, and spike amplitude encoding. From these perspectives, we present general and advanced spike encoding strategies, significantly enhancing the overall representational capacity of the spike signals. Finally, we theoretically confirm the effectiveness of our spike encoding methods in general language-oriented SNNs.

4.1. Bi-Directional Spike Encoding

As shown in Eq.2, the previous spike encoding binarizes the current membrane potential into $\{0, 1\}$, overlooking all negative membrane potentials with half of the information. We propose a bidirectional spike encoding with ternary levels $\{-1, 0, 1\}$, considering both positive and negative membrane potentials. Since the spike encoding of the membrane potential is non-differentiable, we first define stochastic spike encoding to relax spikes as random variables. Then, we calculate the expectation of gradient based on the distribution of the spikes for backward propagation.

We first define positive stochastic spikes $\widetilde{\boldsymbol{s}}^{+}(t)$ and negative stochastic spikes $\widetilde{\boldsymbol{s}}^{-}(t)$ to encode positive and negative membrane potentials respectively. And then, the bidirectional stochastic spike $\widetilde{\boldsymbol{s}}^{\pm}(t)$ can be defined as the summarization of positive and negative spikes:

$$\widetilde{\boldsymbol{s}}^{+}(t) \overset{\text{def}}{=} \begin{cases} 0, & \boldsymbol{p}^{0} = \text{clip}(1-\boldsymbol{m}(t), 0, 1) \\ +1, & \boldsymbol{p}^{+} = \text{clip}(\boldsymbol{m}(t), 0, 1) \end{cases}, \tag{4}$$

$$\widetilde{\boldsymbol{s}}^{-}(t) \overset{\text{def}}{=} \begin{cases} 0, & \boldsymbol{p}^{0} = \text{clip}(1+\boldsymbol{m}(t), 0, 1) \\ -1, & \boldsymbol{p}^{-} = \text{clip}(-\boldsymbol{m}(t), 0, 1) \end{cases}, \tag{5}$$

$$\widetilde{\boldsymbol{s}}^{\pm}(t) \overset{\text{def}}{=} \widetilde{\boldsymbol{s}}^{+}(t) + \widetilde{\boldsymbol{s}}^{-}(t), \tag{6}$$

where $\boldsymbol{p}^{+}$, $\boldsymbol{p}^{0}$, and $\boldsymbol{p}^{-}$ indicate the probability of +1, 0, and -1 respectively. We define the $\boldsymbol{p}^{+}$, $\boldsymbol{p}^{0}$, and $\boldsymbol{p}^{-}$ according to their distance to the value of +1, 0 and -1, and the clip(.) operations confirm the probability in the range of [0,1], so that, the definitions of $\widetilde{\boldsymbol{s}}^{+}(t)$ and $\widetilde{\boldsymbol{s}}^{-}(t)$ confirm $\boldsymbol{p}^{+}+\boldsymbol{p}^{0} = 1$ and $\boldsymbol{p}^{-} + \boldsymbol{p}^{0} = 1$ respectively.

Backward propagation. Eq.4 and 5 are non-differentiable. To enable backpropagation in the entire SNN, we calculate the expectation of the stochastic gradient of $\widetilde{\boldsymbol{s}}^{\pm}(t)$. We use the gradient expectation $\mathbb{E}_{\widetilde{\boldsymbol{s}}^{\pm}(t)}$ in place of the deterministic gradient to complete the backpropagation:

$$\begin{aligned} \mathbb{E}_{\widetilde{\boldsymbol{s}}^{\pm}(t)}[\frac{\partial \widetilde{\boldsymbol{s}}^{\pm}(t)}{\partial \boldsymbol{m}(t)}] &= \frac{\partial}{\partial \boldsymbol{m}(t)} \mathbb{E}[\widetilde{\boldsymbol{s}}^{\pm}(t)] \\ &= \frac{\partial}{\partial \boldsymbol{m}(t)} (-1 \times \boldsymbol{p}^{-} + 0 \times \boldsymbol{p}^{0} + 1 \times \boldsymbol{p}^{+}), \\ &= \frac{\partial}{\partial \boldsymbol{m}(t)} \text{clip}(\boldsymbol{m}(t), -1, 1) \end{aligned} \tag{7}$$

where the gradient expectation $\mathbb{E}_{\widetilde{\boldsymbol{s}}^{\pm}(t)}$ can be derived as the straight-through estimator (STE) (Bengio et al., 2013), which is widely applied to relax non-differentiable operations. The backpropagation can achieve high efficiency, which only performs gradient identity between +1 and -1.

Forward propagation. Eq.4 and 5 involve random sampling. In practice, we convert stochastic spike encoding into deterministic by setting fixed thresholds for efficiency:

$$\boldsymbol{s}^{\pm}(t) = \begin{cases} -1, & \text{if } \boldsymbol{m}(t) < -1 \\ 0, & \text{if } \boldsymbol{m}(t) \in (-1, +1) \\ +1, & \text{if } \boldsymbol{m}(t) > +1 \end{cases}, \tag{8}$$

which is derived by setting the probability condition $\boldsymbol{p}^{+} = 1$ in Eq.4 and $\boldsymbol{p}^{-} = 1$ in Eq.5 for +1 and -1 spike encoding respectively. After bidirectional spike encoding, the membrane potentials are encoded by sequences of {+1,0,-1}. Notably, matrix multiplications between bidirectional spikes and real-valued weights can be converted to pure addition and subtraction operations.

Under the same firing rate r, a bidirectional spike can, at most, increases information entropy r bits for each time compared to the unidirectional spike. This is achieved by directly calculating $\mathcal{H}(\boldsymbol{s}_i^{\pm}(t)) - \mathcal{H}(\boldsymbol{s}_i^{+}(t))$, where the information entropy of the original and bidirectional spikes are formulated by Eq.9:

$$\begin{aligned} &\mathcal{H}(\boldsymbol{s}_i^{+}(t)) = -r\log(r) - (1-r)\log(1-r), \\ &\mathcal{H}(\boldsymbol{s}_i^{\pm}(t)) = -2 \times \frac{r}{2}\log(\frac{r}{2}) - (1-r)\log(1-r). \end{aligned} \tag{9}$$

According to Eq.4,5, information entropy achieves maximum, as long as the positive and negative spikes have the same probability.

4.2. Spike Frequency Encoding

As shown in Eq.2, previous spike encoding disregards input distributions, directly using a fixed threshold $\boldsymbol{\theta}^{l}$ to binarize

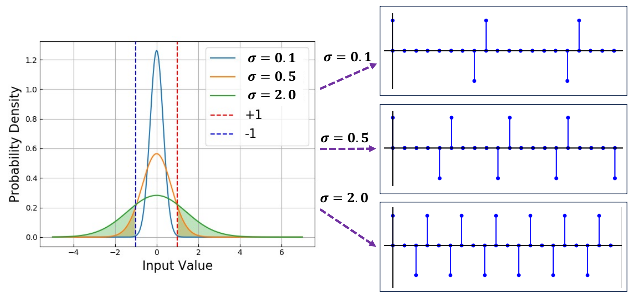


Figure 3. Relationship between the variance of input distributions and the spike firing frequencies.

inputs. However, the input distributions vary across different neurons. A key issue with previous spike encoding is its failure to perceive the variance of input distributions, resulting in difficulties to maintain a reasonable firing rate.

Distribution-aware frequency encoding. We introduce a distribution-aware frequency encoding, which adjusts the input distribution for each neural layer. We achieve stable and manually controllable spike firing frequencies by elasticating membrane potentials with a scaling factor $\boldsymbol{\alpha}(t)$, and the elastic membrane potential is $\hat{\boldsymbol{m}}(t) = \boldsymbol{m}(t)/\boldsymbol{\alpha}(t)$. As illustrated in Fig. 3, by adjusting the variance of the input distribution, we can encode spikes in different frequencies, where a larger variance distribution results in a higher spike firing rate. We define the scaling factor as k times the mean of membrane potential amplitude, $\frac{1}{n}\sum_{i=1}^{n}|\boldsymbol{m}_i(t)|$:

$$\boldsymbol{\alpha}^l(t) \overset{\text{def}}{=} \frac{k}{n}\sum_{i=1}^{n}\left|\boldsymbol{m}_i^{l,(1)}(t)\right|, \tag{10}$$

where $\boldsymbol{m}_i^{l,(1)}(t)$ indicates the membrane potential under the first batch of training data. For stable training, we determine every $\boldsymbol{\alpha}^l(t)$ by the first batch, and then freeze $\boldsymbol{\alpha}^l(t)$ in training. We replace the $\boldsymbol{m}^l(t)$ in Eq.8 with the elastic membrane potential $\hat{\boldsymbol{m}}^l(t)$ and obtain the frequency encoding:

$$\boldsymbol{s}^{\pm}(t) = \begin{cases} -1, & \text{if } \boldsymbol{m}(t) < -\boldsymbol{\alpha}(t) \\ 0, & \text{if } \boldsymbol{m}(t) \in (-\boldsymbol{\alpha}(t), +\boldsymbol{\alpha}(t)) \\ +1, & \text{if } \boldsymbol{m}(t) > +\boldsymbol{\alpha}(t) \end{cases}. \tag{11}$$

In Eq.10, we define k as an adjustable hyperparameter. When reducing k, it equally reduces the spike threshold in Eq.11, leading to an increased spike frequency; conversely, increasing k reduces the spike frequency.

Then, we will understand $\boldsymbol{\alpha}^l(t)$ from the perspective of the variance of input, which distribution is widely believed as roughly zero-mean Gaussian or Laplacian (Lin et al., 2022; Banner et al., 2019). For simplification, we also use this assumption; for more complex distributions, it can be proved similarly as follows.

Lemma 1. *Given a zero-mean Gaussion or Laplacian membrane potential m, i.e., $m \sim \mathcal{N}(0,\sigma^2)$ or $m \sim La(0,b)$, the scaling factor $\boldsymbol{\alpha}^l(t)$ is $\sqrt{\frac{2}{\pi}}\sigma k$ or bk.*

This is proved by calculating the expectation of $\alpha^l(t)$, and $\boldsymbol{\alpha}^l(t) = k\mathbb{E}[|\boldsymbol{m}^l(t)|] = k\int_{-\infty}^{\infty}|m|f(m)\,dm$, where $f(.)$ is zero-mean Gaussian or Laplacian distribution. As *Lemma 1*, $\boldsymbol{\alpha}^l(t)$ is linearly related to the standard deviation σ.

4.3. Spike Amplitude Encoding

Eq.2 also neglects the intensity of membrane potential intensity. To preserve the intensity information, we encode the expectation of membrane potentials into spikes amplitude. This is achieved by scaling spike amplitude to $\boldsymbol{\alpha}^l(t)$, which formulates an identity transformation for membrane potentials in expectation:

$$\boldsymbol{s}^{\pm}(t) = \begin{cases} -\boldsymbol{\alpha}(t), & \text{if } \boldsymbol{m}(t) < -\boldsymbol{\alpha}(t) \\ 0, & \text{if } \boldsymbol{m}(t) \in (-\boldsymbol{\alpha}(t), +\boldsymbol{\alpha}(t)) \\ +\boldsymbol{\alpha}(t), & \text{if } \boldsymbol{m}(t) > +\boldsymbol{\alpha}(t) \end{cases}. \tag{12}$$

In this case, the backpropagation is the same as Eq.7, since Eq.12 is equivalent to dividing and then multiplying Eq.8 by $\boldsymbol{\alpha}(t)$ before and after respectively. Due to the spike amplitude becoming $\boldsymbol{\alpha}^l(t)$, we accordingly revise the membrane potential update formula Eq.3 as:

$$\boldsymbol{v}^l(t) = \boldsymbol{m}^l(t)(\boldsymbol{\alpha}(t) - \boldsymbol{s}^l(t)) + v_{reset}\boldsymbol{s}^l(t). \tag{13}$$

Notably, at each time step t, we employ a layerwise amplitude encoding. Due to the commutative property of multiplication, we can first conduct matrix multiplications with unit amplitude spikes, and then reweight the results using spike amplitude. After training, the spike amplitude can merge with the weights in this layer. Amplitude encoding does not change the addition property of spike-driven operations.

4.4. Conquering Language Modeling with SNNs

We refer to the proposed spike encoding in Eq.1,12,13 as elastic bi-spiking mechanisms, which jointly encodes extended direction, frequency, and amplitude information of membrane potentials. We replace the traditional LIF neurons with elastic bi-spiking mechanisms and construct directly trainable language-oriented SNNs, termed SpikeLM.

We theoretically prove that elastic bi-spiking mechanisms ensure high optimization stability in SpikeLM, guaranteeing its performance in general language tasks. This is achieved by dynamical isometry: if a neural network achieves dynamical isometry, it prevents gradients from vanishing or exploding, maintaining nearly all values of its input-output Jacobian matrixes around one. A neural network can generally be viewed as a series of blocks $f_{\boldsymbol{\theta}^j}^j$ with parameters $\boldsymbol{\theta}^j$:

$$f(x_0) = f_{\boldsymbol{\theta}^L}^L \circ f_{\boldsymbol{\theta}^{L-1}}^{L-1} \circ \cdots \circ f_{\boldsymbol{\theta}^1}^1(x_0), \tag{14}$$

where the Jacobian matrix $\frac{\partial f^j}{\partial f^{j-1}}$ is $\boldsymbol{J}_j$. It can be defined $\phi(\boldsymbol{J}) \overset{\text{def}}{=} \mathbb{E}[tr(\boldsymbol{J})]$, and $\varphi(\boldsymbol{J}) \overset{\text{def}}{=} \phi(\boldsymbol{J}^2) - \phi(\boldsymbol{J})^2$.

Definition 1. Block Dynamical Isometry (Definition 3.1 in (Chen et al., 2020)). *Consider a neural network that can be represented as Eq. 14 and the j-th block's Jacobian matrix is denoted as $\boldsymbol{J}_j$. If $\forall$ j, $\phi(\boldsymbol{J}_j\boldsymbol{J}_j^T) \approx 1$ and $\varphi(\boldsymbol{J}_j\boldsymbol{J}_j^T) \approx 0$, the network achieves block dynamical isometry.*

Lemma 2. *Given the probability of the input greater than 0 is p, the values of $\phi(\boldsymbol{J})$ and $\varphi(\boldsymbol{J})$ are p and $p - p^2$ for the ReLU function.* (Proof in A.6 (Chen et al., 2020))

Lemma 3. *Given the spike fire rate is r, the values of $\phi(\boldsymbol{J})$ and $\varphi(\boldsymbol{J})$ are $1 - r$ and $r - r^2$ respectively for the elastic bi-spiking function in Eq.12.*

Proof. For clarity, we denote the elastic spike encoding (Eq.12) as $\boldsymbol{s}(\boldsymbol{m})$, where $\boldsymbol{m}$ is the membrane potential, and donate the Jacobian matrix as $\boldsymbol{s}_{\boldsymbol{m}}$. Because $\boldsymbol{s}(\boldsymbol{m})$ is the element-wise operation, $\boldsymbol{s}_{\boldsymbol{m}}$ is a diagonal matrix. According to Eq.7, the gradient of $\boldsymbol{s}(\boldsymbol{m})$ is the STE between -1 and +1, so that, the value of $\boldsymbol{s}_{\boldsymbol{m}}$ is 0 or 1. Given the spike firing rate r, the probability in [-1,+1] is $1 - r$. Therefore, the spectral density of $\boldsymbol{s}_{\boldsymbol{m}}$ is: $\rho_{\mathbf{s_m}}(z) = r\delta(z) + (1-r)\delta(z-1)$. And we have $\rho_{\mathbf{s_m}\mathbf{s_m}^T}(z) = \rho_{\mathbf{s_m}}(z)$ because of the $\{0,1\}$ matrix value. Accordingly, we have:

$$\begin{aligned} \phi(\mathbf{s_m}\mathbf{s_m}^T) &= \int_{\mathbb{R}} z\rho_{\mathbf{s_m}\mathbf{s_m}^T}(z)dz = 1 - r, \\ \varphi(\mathbf{s_m}\mathbf{s_m}^T) &= \int_{\mathbb{R}} z^2\rho_{\mathbf{s_m}\mathbf{s_m}^T}(z)dz - \phi^2(\mathbf{s_m}\mathbf{s_m}^T) \\ &= r - r^2. \end{aligned} \tag{15}$$

□

Theorem 1. *In a deep neural network, the elastic bi-spiking function achieves better dynamical isometry than the ReLU: $\phi(\mathbf{s_m}\mathbf{s_m}^T) > \phi(\mathbf{f_x}\mathbf{f_x}^T)$, $\varphi(\mathbf{s_m}\mathbf{s_m}^T) < \varphi(\mathbf{f_x}\mathbf{f_x}^T)$.*

Proof. In *Lemma 2* and *Lemma 3*, p is usually believed 0.5 for zero-mean input distribution and r is roughly 0.1 to 0.3 in SNNs. Accordingly, $\phi(\mathbf{s_m}\mathbf{s_m}^T) > \phi(\mathbf{f_x}\mathbf{f_x}^T)$ is achieved. Moreover, the function $f(x) = x - x^2$ achieves maximum given $x = 0.5$, so that, $\varphi(\mathbf{s_m}\mathbf{s_m}^T) < \varphi(\mathbf{f_x}\mathbf{f_x}^T)$ is achieved. □

Based on *Theorem 1*, the Jacobian matrix of Eq.12 is closer to $\mathbf{I}$ than the ReLU function in ANNs. As a result, the elastic bi-spiking function has better optimization stability than ReLU at least. The training stability of SpikeLM is confirmed accordingly.

5. Experiments

We evaluate previous SNNs and SpikeLMs on a range of general language tasks, including discriminative and generative. We mainly explore three key issues: **(i)** the baseline performance of traditional SNNs in general language tasks; **(ii)** the effectiveness of elastic bidirectional spike encoding in SpikeLM; and **(iii)** how to achieve controllable spike firing rate for energy efficiency.

5.1. Settings

Language tasks. For discriminative tasks, we evaluate SNNs on the standard GLUE benchmark(Wang et al., 2018), which includes 8 subsets for classification and regression in different scenes. For generative tasks, we evaluate text summarization benchmarks: XSUM (Narayan et al., 2018) and CNN-DailyMail (Nallapati et al., 2016). Additionally, we evaluate the machine translation task on the WMT16 English-Romanian dataset (Bojar et al., 2016).

Architectures. We develop SNN baselines and SpikeLM for discriminative and generative tasks using BERT and BART architectures respectively. For frequency encoding, we set $k = 2$. As Section 3.1, we implement SNNs by replacing all matrix multiplications in ANNs with spike operations, maintaining the same architectures. Specifically, we use: **(i)** the BERT base (Devlin et al., 2018) for discriminative tasks, which is a 12-layer encoder transformer with 110M parameters; **(ii)** the BART base (Lewis et al., 2019) for text summarization, which is a encoder-decoder transformer with 6 layers for each and totally 139M parameters; and **(iii)** the mBART large model (Liu et al., 2020) for translation tasks, which is pretrained on 25 languages and has 680M parameters.

5.2. Discriminative Tasks

We follow the standard ANN-based BERT to develop SNN-based LIF-BERT and SpikeLM, which include two stages: pretraining and finetuning. In pretraining, we use the BooksCorpus (Zhu et al., 2015) and English Wikipedia (Devlin et al., 2018) as training data, including 800M and 2500M words respectively. In finetuning, we use the GLUE benchmark training with the common settings of ANNs. Training details are reported in Appendix A.2.

Results of GLUE benchmark. As shown in Table 3, we compare SpikeLM with both ANNs and SNNs. Our ANN baselines include BERTs (Devlin et al., 2018), ELMo (Peters et al., 2018), and Q2BERT with 2-bit weights and 8-bit activations (Zhang et al., 2020a), while the SNN baselines include SpikeBERT (Lv et al., 2023) and directly training SpikeingFormer (Zhou et al., 2023). We additionally implement spike-driven BERTs with the PSN (Fang et al., 2023b) and LIF (Gerstner et al., 2014) neurons with original neuron settings (Fang et al., 2023a; Lv et al., 2023). The difference between LIF-BERT and LIF-BERT* is in Appendix A.1. Compared with BERT$_{\texttt{base}}$, SpikeLM reduces the performance gap to 6.7%, while the original gap is 28.3% in

Table 3. Comparisons between SpikeLM and ANNs or other SNNs on the GLUE dev set. The energy consumption is evaluated by FP32 operations and more results about FP16 energy consumption are in Table 5. We report the evaluation metric in Appendix A.3. We implement LIF-BERT* and PSN-BERT* with LIF and PSN spiking neurons in Spikingjelly (Fang et al., 2023a), where the hyperparameters of spiking neurons follow previous SpikeBERT (Lv et al., 2023). $\text{BERT}_{\texttt{3L}}$ and $\text{SpikeLM}_{\texttt{6L}}$ indicate the 3-layer BERT model and 6-layer SpikeLM respectively. We transfer the vision-oriented SpikingFormer (Zhou et al., 2023) to language tasks by removing spiking neurons in the query and attention weight for a fair comparison with LIF-BERT and SpikeLM. Except for SpikeBERT (Lv et al., 2023), all spiking models apply the same training scheme.

Model	**Energy** (mJ)	**Time**	**MNLI**$_{\text{-m/mm}}$	**QQP**$_{\text{F1}}$	**QNLI**	**SST-2**	**CoLA**	**STS-B**	**MRPC**$_{\text{F1}}$	**RTE**	**Avg.**
$\text{BERT}_{\texttt{base}}$	51.41	–	83.8/83.4	90.5	90.7	92.3	60.0	89.4	89.8	69.3	83.2
$\text{BERT}_{\texttt{3L}}$	12.9	–	77.1/77.1	85.2	85.8	88.1	31.7	85.7	86.4	66.4	75.9
Q2BERT	–	–	47.2/47.3	67.0	61.3	80.6	0.0	4.7	81.2	52.7	49.1
ELMo	–	–	68.6/–	86.2	71.1	91.5	44.1	70.4	76.6	53.4	70.2
SpikeBERT	14.30	4	71.4/71.0	68.2	66.4	85.4	16.9	18.7	82.0	57.5	59.7
LIF-BERT*	–	4	35.4/35.2	0.0	50.5	50.9	0.0	0.0	81.2	52.7	34.6
PSN-BERT*	–	4	35.4/35.2	0.0	50.5	50.9	0.0	6.8	81.2	52.7	34.7
LIF-BERT	7.98	4	56.8/55.2	70.0	60.6	80.6	14.6	20.0	82.3	53.8	54.9
SpikingFormer	–	1	67.8/68.6	79.3	74.6	82.7	16.7	72.3	83.0	58.8	67.1
SpikingFormer	–	4	70.2/70.6	80.9	79.5	83.9	12.8	77.0	83.0	62.1	68.9
$\text{SpikeLM}_{\texttt{6L}}$	2.05	1	73.9/75.3	83.2	84.2	86.2	30.7	83.7	85.7	66.8	74.4
$\text{SpikeLM}_{\texttt{6L}}$	7.06	4	75.1/75.3	83.5	84.6	87.4	33.7	84.5	86.5	64.3	75.0
SpikeLM	3.98	1	76.0/76.9	84.0	84.9	86.5	37.9	84.3	85.6	65.3	75.7
SpikeLM	13.74	4	77.1/77.2	83.9	85.3	87.0	38.8	84.9	85.7	69.0	76.5

Table 4. Comparisons between SpikeLM, SpikingFormer (Zhou et al., 2023), and ultra-low bit quantization methods on the GLUE dev set. Weight and Act. indicate the bit-width of weights and activations respectively, where ter indicates the ternary quantization level. For SpikeLM and SpikingFormer, we set the time step as 1. The 1-bit weight SpikeLM has similar operations with binary BERTs, which is because the binary BERTs have about 0.5× equivalent sparsity (Fig. 2) while SpikeLM has less activation firing rate in the BERT architecture.

Model	**Weight**	**Act.**	**MNLI**$_{\text{-m/mm}}$	**QQP**$_{\text{acc}}$	**QNLI**	**SST-2**	**CoLA**	**STS-B**	**MRPC**$_{\text{acc}}$	**RTE**	**Avg.**
Q2BERT	2	8	47.2/47.3	67.0	61.3	80.6	0.0	4.4	68.4	52.7	47.7
TernaryBERT	Ter	Ter	40.3/40.0	63.1	50.0	80.7	0.0	12.4	68.3	54.5	45.5
BinaryBERT	1	1	62.7/63.9	79.9	52.6	82.5	14.6	6.5	68.3	52.7	53.7
BiBERT	1	1	66.1/67.5	84.8	72.6	88.7	25.4	33.6	72.5	57.4	63.2
BiT	1	1	77.1/77.5	82.9	85.7	87.7	25.1	71.1	79.7	58.8	71.0
BiPFT	1	1	69.5/70.6	83.7	81.7	86.2	22.9	80.2	76.2	66.1	70.8
SpikingFormer	32	1	67.8/68.6	83.8	74.6	82.7	16.7	72.3	74.0	58.8	66.6
SpikeLM	32	Ter	76.0/76.9	87.9	84.9	86.5	37.9	84.3	78.7	65.3	75.4
SpikeLM	1	Ter	74.9/75.1	87.2	84.5	86.6	36.0	83.9	78.9	65.7	74.8

Table 5. Energy consumption of the ANN-based BERT, LIF-BERT, and SpikeLM. We evaluate energy with both the FP16 and FP32 Multiply-ACcumulate (MAC) or ACcumulate (AC) operations.

Model	$\text{BERT}_{\texttt{base}}$	LIF-BERT	SpikeLM	SpikeLM
Steps	1	4	1	4
FP16 (mJ)	15.21	3.55	1.77	6.10
FP32 (mJ)	51.41	7.98	3.98	13.74

the LIF-BERT baseline. We compare SpikeLM with SpikeBERT (Lv et al., 2023), which is distilled from ANN-based BERT. SpikeLM exceeds 16.8% average performance without any distillation, indicating the overall improvements in stand-alone learning capabilities of SNNs. Compared with original LIF-BERT* and PSN-BERT*, SpikeLM dramatically improves 41.9% and 41.8% respectively. Results show that previous {0,1} spikes can not successfully model standard discriminative tasks. By leveraging elastic bi-spike encoding, their performance drop is effectively addressed.

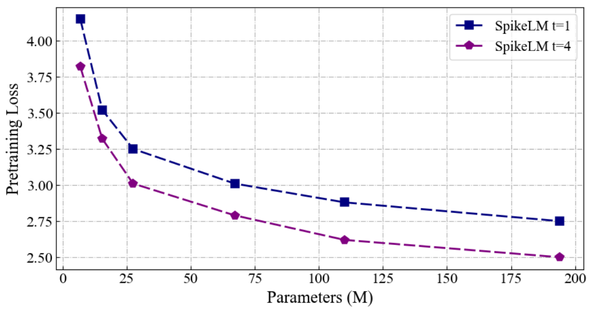
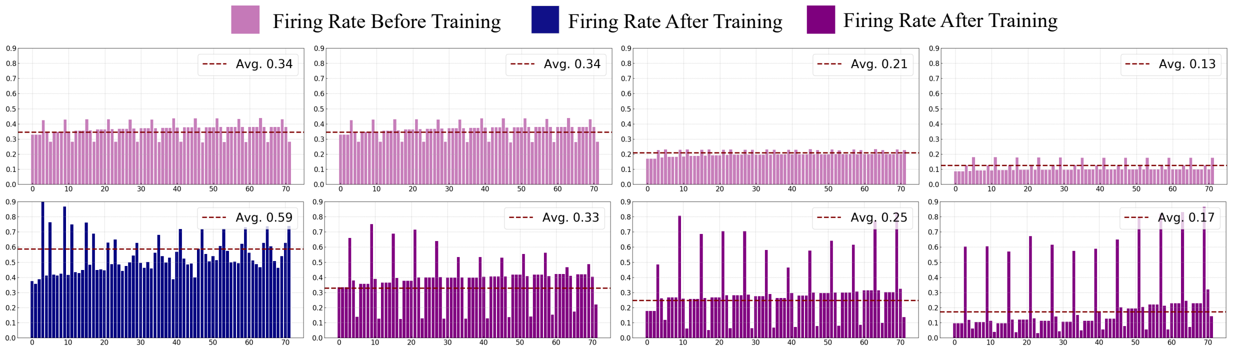


Figure 4. SNN scaling law of SpikeLM (T=1,4).

As shown in Table 4, we also compare SpikeLMs with ultra-low bit quantization BERTs including Q2BERT (Zhang et al., 2020a), TernaryBERT (Qin et al., 2022; Zhang et al.,



(a) Learnable Threshold (b) Frequency Encoding, k=2 (c) Frequency Encoding, k=3 (d) Frequency Encoding, k=4

Figure 5. Spiking firing rate in linear layers under 2 settings: the learnable thresholds $\boldsymbol{\alpha}(t)$ (a) and spike frequency encoding (b, c, d).

Table 6. Ablation study on bidirectional spike encoding and spike frequency/amplitude encoding.

Method	**Spikes**	**GLUE Avg.**
LIF-BERT* (T=4)	$\{0,1\}$	34.6
LIF-BERT (T=4)	$\{0,1\}$	54.9
+ Bidirect. encoding	$\{-1,0,1\}$	55.7
+ Freq./Amp. encoding	$\{-\alpha,0,\alpha\}$	76.5
LIF-BERT (T=4)	$\{0,1\}$	54.9
+ Freq./Amp. encoding	$\{0,\alpha\}$	71.8
+ Bidirect. encoding	$\{-\alpha,0,\alpha\}$	76.5

2020a), BinaryBERT (Qin et al., 2022; Bai et al., 2020), BiBERT (Qin et al., 2022), BiT (Liu et al., 2022), and BiPFT (Xing et al., 2024). Because of the sparse encoding in SpikeLM, the 1-bit weight SpikeLM (T=1) has similar operations to BERTs with both binary weights and activations. Specifically, we view the sparsity of BNNs as 0.5 according to Fig. 2, because value levels are able to map to {0,1} in inference. Compared with binary BERTs, SpikeLM also achieves higher performance.

Energy efficiency. As Table 3, compared with BERT$_{\texttt{base}}$, SpikeLM saves 12.9× and 3.7× energy consumption with spike time steps 1 and 4 respectively. Compared with SpikeBERT (Lv et al., 2023), SpikeLM (T=1) exceeds 16.0% average performance and also saves 3.6× energy. In Table 5, it is shown that FP16 and FP32 operations have a similar tendency.

SNN scaling law. As shown in Fig. 4, we explore the scalability of language-oriented SNNs by adjusting the parameter number with different model widths. We pretrain SpikeLMs from 6.9M to 194M parameters and use pretraining loss, including the mask language modeling and next sentence prediction, as the evaluation metric. For larger models, Wikipedia and BooksCorpus may be insufficient for pretraining, and larger-scale datasets are needed. The experiments show that the elastic bi-spiking mechanism follows the scaling law, supporting SpikeLM's scalability to

Table 7. Trade-off between performance and efficiency. N/A indicates directly setting spike thresholds as -1 and +1 as Eq.8.

Settings	N/A	Learnable	k=2	k=3	k=4
GLUE Avg. (%)	55.7	76.8	**76.5**	75.1	60.8
Erengy (mJ)	17.3	23.8	**13.7**	10.4	7.4

some extent.

Ablation study. The improvements of SpikeLM are attributed to elastic bi-spiking mechanisms, by encoding the direction, frequency, and amplitude of spikes. Notably, the frequency and amplitude encoding are coupled in Eq. 12. Therefore, we analyze the individual contributions of bidirectional and frequency/amplitude encoding in Table 6. When implementing backpropagation by Straight-Through Estimator (STE), adding spike frequency/amplitude encoding and bidirectional spike encoding improves the GLUE performance by 16.9% and 4.7%, demonstrating the enhanced modeling capacity of elastic bi-spiking mechanism. As a result, our visualization in Appendix A.4 demonstrates the extended bidirectional and amplitude information in spikes, and a proper firing rate is maintained.

Controllable spike firing rate. As Section 3.2, a key issue with existing SNNs is the trade-off between energy and performance. Spike frequency encoding can achieve a controllable firing rate, thereby enabling a manageable balance between performance and energy consumption. To estimate this, we compare two settings: **(i)** setting $\boldsymbol{\alpha}(t)$ as learnable parameters in each spike layer, as shown in Fig.5(a), and **(ii)** setting $\boldsymbol{\alpha}(t)$ as Eq.10 with $k = 2, 3, 4$, as shown in Fig.5(b, c, d). In Fig.5, we compare the distributions of spike firing rate in each linear layer of SpikeLM BERT models. We have the following results: **(i)** In Fig.5(a), the freely trainable $\boldsymbol{\alpha}(t)$ leads to excessively high spike firing rates due to maximizing the entropy of the multinomial distribution. **(ii)** Spike frequency can be effectively controlled by hyperparameter k in Eq.12. By increasing k from 2 to 4, the average firing rate changes from 33% to 17%. **(iii)** Spike

Table 8. Comparisons between SpikeLM and ANNs or LIF-BARTs on generative tasks. We evaluate text summarization on XSUM and CNN-DailyMail with rouge-{1,2,L} metrics; and evaluate translation on the WMT16 En-Ro dataset with the BLEU metric. For XSUM and CNN-DailyMail, we use the base BART architecture; for WMT16, we use the large mBART architecture. We evaluate average energy consumption on the XSUM dataset for the base-sized BART, LIF-BART, and SpikeLM. Following other SNN works (Zhou et al., 2023; Yao et al., 2023a), we also apply FP32 operations for energy evaluation.

	XSUM	**XSUM**			**CNN-DailyMail**			**WMT16**
Model	**Energy** (mJ)	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L	BLEU
$\text{BART}_{\texttt{base}}/\text{mBART}_{\texttt{large}}$	378.22	42.75	19.80	34.72	44.88	22.24	31.75	26.82
BERTSUMABS	–	38.76	16.33	31.15	41.72	19.39	38.76	–
PTGEN	–	29.70	9.21	23.24	36.44	15.66	33.42	–
GPT-2	–	–	–	–	29.34	8.27	26.58	–
LIF-BART (T=1)	6.59	33.17	12.30	26.56	39.88	16.77	26.15	14.72
LIF-BART (T=4)	28.92	35.47	13.84	28.32	41.39	18.33	28.05	19.04
SpikeLM (T=1)	25.68	39.40	17.13	31.97	41.56	18.74	28.51	20.93
SpikeLM (T=4)	115.79	40.63	17.97	32.92	42.19	19.27	29.06	22.95

frequency encoding controls the firing rates without much performance drop. Compared to learnable thresholds, the frequency encoding at $k = 2$ has almost the same performance and saves 42.4% of energy in Table 7. **(iv)** The firing rate remains similar before and after training with spike frequency encoding, allowing to predict firing rate at the beginning of training.

5.3. Generative Tasks

We evaluate fully spike-driven language models in generative tasks for the first time. Generation tasks require extended input and output sequence lengths, which necessitates advanced language modeling capabilities in SNNs. Therefore, we introduce the distillation strategy, which involves initializing and employing knowledge distillation from a pretrained ANN teacher. In detail, we select the pretrained BART-base and mBART-large models as ANN teachers for summarization and translation. Training details are shown in Appendix A.2.

Results of summarization. In Table 8, we compare SpikeLM with both ANN and SNN baselines. Compared with the BART base model, the ROUGE-L of SpikeLM (T=4) only drops 1.80% and 2.69% on XSUM and CNN-DailyMail, despite replacing all the matrix multiplications as spikes-driven. Compared with GPT-2 and other ANNs, SpikeLMs can be also competitive. This is the first time to verify fully spike-driven models achieve competitive performance with ANNs in challenging generative tasks.

Compared with LIF-BARTs on XSUM, SpikeLMs exceed 5.41% with one-step spikes and 4.60% with 4-step spikes. Similar results are also shown on CNN-DailyMail. This indicates the omnidirectionally improved bidirectional spikes with levels $\{-\alpha, 0, \alpha\}$ achieve much more capabilities in language modeling than previous $\{0, 1\}$ spikes.

Results of translation. In Table 8, we go further to evaluate SpikeLMs on multilingual tasks and large-sized mBART architecture. We observe that, even with the large-sized model, the performance of LIF-BART (T=4) remains inferior to SpikeLM (T=1). Therefore, improving spike capacity by generalized encodings is more effective than increasing spike time steps in previous SNNs.

6. Conclusion

This work proposes a fully spiking mechanism for general language tasks, demonstrating the potential generalization capacity of SNNs at a higher level. Unlike previous binary spikes, spike capabilities are significantly extended from bi-direction, amplitude, and frequency encodings, while maintaining the addition nature of SNNs. Inspired by advanced neuroscience, it would be great potential to develop efficient and environmentally friendly large language models with spike-driven methods in the future.

Limitations

The limitations of this work include the activation spiking in traditional SNNs and the model scale. Compared with weight quantization, activation quantization in SNNs is more challenging. Moreover, recent large language models are memory-bounded, and leveraging spiking neuronal dynamics to quantize weights may achieve higher performance and efficiency in the future.

Acknowledgements

This work is supported by National Key R&D Program of China 2022ZD0160602, Natural Science Foundation of China 62122088, National Science Foundation of China under grant No.62206150, Distinguished Young Scholars (62325603), National Natural Science Foundation of China (62236009,62441606), and Beijing Natural Science Foundation for Distinguished Young Scholars (JQ21015).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Bai, H., Zhang, W., Hou, L., Shang, L., Jin, J., Jiang, X., Liu, Q., Lyu, M., and King, I. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*, 2020.

Banner, R., Nahshan, Y., and Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019.

Basu, A., Deng, L., Frenkel, C., and Zhang, X. Spiking neural network integrated circuits: A review of trends and future directions. In *2022 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8. IEEE, 2022.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Logacheva, V., Monz, C., et al. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*, pp. 131–198. Association for Computational Linguistics, 2016.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

Chen, Z., Deng, L., Wang, B., Li, G., and Xie, Y. A comprehensive and modularized statistical framework for gradient norm equality in deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (1):13–31, 2020.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Deng, S. and Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021a.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2661–2671, 2021b.

Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., Huang, L., Zhou, H., Li, G., and Tian, Y. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40): eadi1480, 2023a.

Fang, W., Yu, Z., Zhou, Z., Chen, D., Chen, Y., Ma, Z., Masquelier, T., and Tian, Y. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Guo, Y., Huang, X., and Ma, Z. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023a.

Guo, Y., Liu, X., Chen, Y., Zhang, L., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17391–17401, 2023b.

Guo, Y., Peng, W., Chen, Y., Zhang, L., Liu, X., Huang, X., and Ma, Z. Joint a-snn: Joint training of artificial and spiking neural networks via self-distillation and weight factorization. *Pattern Recognition*, 142:109639, 2023c.

Guo, Y., Zhang, Y., Chen, Y., Peng, W., Liu, X., Zhang, L., Huang, X., and Ma, Z. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19420–19430, 2023d.

Guo, Y., Chen, Y., Liu, X., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12244–12252, 2024.

Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

Hu, Y., Tang, H., and Pan, G. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

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

Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11270–11277, 2020.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021.

Li, Z., Wang, Z., Tan, M., Nallapati, R., Bhatia, P., Arnold, A., Xiang, B., and Roth, D. Dq-bart: Efficient sequence-to-sequence model via joint distillation and quantization. *arXiv preprint arXiv:2203.11239*, 2022.

Lin, M., Ji, R., Xu, Z., Zhang, B., Chao, F., Lin, C.-W., and Shao, L. Siman: Sign-to-magnitude network binarization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6277–6288, 2022.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8: 726–742, 2020.

Liu, Z., Oguz, B., Pappu, A., Xiao, L., Yih, S., Li, M., Krishnamoorthi, R., and Mehdad, Y. Bit: Robustly binarized multi-distilled transformer. *Advances in neural information processing systems*, 35:14303–14316, 2022.

Lv, C., Li, T., Xu, J., Gu, C., Ling, Z., Zhang, C., Zheng, X., and Huang, X. Spikebert: A language spikformer trained with two-stage knowledge distillation from bert. *arXiv preprint arXiv:2308.15122*, 2023.

Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., and Wei, F. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

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

Mehonic, A. and Kenyon, A. J. Brain-inspired computing needs a master plan. *Nature*, 604(7905):255–260, 2022.

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., and Luo, Z.-Q. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12444–12453, 2022.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*, 2018.

Narayanan, S., Taht, K., Balasubramonian, R., Giacomin, E., and Gaillardon, P.-E. Spinalflow: An architecture and dataflow tailored for spiking neural networks. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 349–362. IEEE, 2020.

Neftci, E. O., Mostafa, H., and Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., and Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, 24(7):1010–1019, 2021.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In Walker, M., Ji, H., and Stent, A. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Pham, Q., Liu, C., and Hoi, S. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34:16131–16144, 2021.

Qin, H., Ding, Y., Zhang, M., Yan, Q., Liu, A., Dang, Q., Liu, Z., and Liu, X. Bibert: Accurate fully binarized bert. *arXiv preprint arXiv:2203.06390*, 2022.

Roy, K., Jaiswal, A., and Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Schuman, C. D., Kulkarni, S. R., Parsa, M., Mitchell, J. P., Kay, B., et al. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

Su, Q., Chou, Y., Hu, Y., Li, J., Mei, S., Zhang, Z., and Li, G. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6555–6565, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

Wu, J., Xu, C., Han, X., Zhou, D., Zhang, M., Li, H., and Tan, K. C. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (11):7824–7840, 2021.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331, 2018.

Xing, X., Li, Y., Li, W., Ding, W., Jiang, Y., Wang, Y., Shao, J., Liu, C., and Liu, X. Towards accurate binary neural networks via modeling contextual dependencies. In *European Conference on Computer Vision*, pp. 536–552. Springer, 2022.

Xing, X., Du, L., Wang, X., Zeng, X., Wang, Y., Zhang, Z., and Zhang, J. Bipft: Binary pre-trained foundation transformer with low-rank estimation of binarization residual polynomials. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16094–16102, 2024.

Yao, M., Hu, J., Zhou, Z., Yuan, L., Tian, Y., Bo, X., and Li, G. Spike-driven transformer. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Yao, M., Zhang, H., Zhao, G., Zhang, X., Wang, D., Cao, G., and Li, G. Sparser spiking activity can be better: Feature refine-and-mask spiking neural network for event-based visual recognition. *Neural Networks*, 166:410–423, 2023b.

Yao, M., Hu, J., Hu, T., Xu, Y., Zhou, Z., Tian, Y., Xu, B., and Li, G. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. *arXiv preprint arXiv:2404.03663*, 2024a.

Yao, M., Richter, O., Zhao, G., Qiao, N., Xing, Y., Wang, D., Hu, T., Fang, W., Demirci, T., De Marchi, M., Deng, L., Yan, T., Nielsen, C., Sheik, S., Wu, C., Tian, Y., Xu, B., and Li, G. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications*, 15(1):4464, May 2024b. ISSN 2041-1723. URL https://doi.org/10.1038/s41467-024-47811-6.

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

Zhang, W., Hou, L., Yin, Y., Shang, L., Chen, X., Jiang, X., and Liu, Q. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*, 2020a.

Zhang, Y., Qu, P., Ji, Y., Zhang, W., Gao, G., Wang, G., Song, S., Li, G., Chen, W., Zheng, W., et al. A system hierarchy for brain-inspired computing. *Nature*, 586(7829): 378–384, 2020b.

Zhang, Y., Zhang, Z., and Lew, L. Pokebnn: A binary pursuit of lightweight accuracy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12475–12485, 2022.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11062–11070, 2021.

Zhou, C., Yu, L., Zhou, Z., Zhang, H., Ma, Z., Zhou, H., and Tian, Y. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, 2023.

Zhou, Z., Che, K., Fang, W., Tian, K., Zhu, Y., Yan, S., Tian, Y., and Yuan, L. Spikformer v2: Join the high accuracy club on imagenet with an snn ticket. *arXiv preprint arXiv:2401.02020*, 2024.

Zhu, R.-J., Zhao, Q., and Eshraghian, J. K. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A. Appendix

A.1. Implementation Details of the LIF-based Transformers

For comparison, we implement Leaky integrate-and-fire (LIF) neurons (Gerstner et al., 2014) with 2 backpropagation algorithms:

- the original arctangent-like surrogate gradient function. We implement LIF-BERT* by Spikingjelly (Fang et al., 2023a), which is a popular open-source SNN framework with previous spike neurons.
- To make a strict comparison with our method in Section 4, we also propose a straight-through estimator (STE) (Bengio et al., 2013) based backpropagation for LIF neurons. We implement our LIF-BERT/BART baselines in this way by PyTorch.

Forward propagation. The same as Eq.2, the membrane potential is binarized by Eq.16, the $\boldsymbol{\theta}^l$ is 1 in usual:

$$\boldsymbol{s}^l(t) = \begin{cases} 0, & \text{if } \boldsymbol{m}^l(t) < \boldsymbol{\theta}^l \\ 1, & \text{if } \boldsymbol{m}^l(t) \geq \boldsymbol{\theta}^l \end{cases} . \tag{16}$$

Both of our and the original implementations are the same in forward propagation. The difference is how to relax this non-differentiable function for gradient calculation.

Backward propagation. In the original LIF neurons, a gradient surrogate function is used:

$$\boldsymbol{s}^l(t) \approx \frac{1}{\pi}\arctan(\frac{\pi}{2}\alpha\boldsymbol{m}^l(t)) + \frac{1}{2}, \tag{17}$$

which is arctangent-like to simulate Eq.16. The following gradient estimation is used accordingly:

$$\frac{\partial \boldsymbol{s}^l(t)}{\partial \boldsymbol{m}^l(t)} = \frac{\alpha}{2}\frac{1}{(1+(\frac{\pi}{2}\alpha\boldsymbol{m}^l(t))^2)}. \tag{18}$$

To make a comparison with our method in Section 4, we also implement a similar STE-based backpropagation and evaluate the performance. Similar to Section 4.1, we relax the LIF neuron output as stochastic variables $\widetilde{\boldsymbol{s}}(t)$:

$$\widetilde{\boldsymbol{s}}(t) \overset{\text{def}}{=} \begin{cases} 0, & \boldsymbol{p}^0 = \text{clip}(1-\boldsymbol{m}(t), 0, 1) \\ 1, & \boldsymbol{p}^+ = \text{clip}(\boldsymbol{m}(t), 0, 1) \end{cases} , \tag{19}$$

where $\boldsymbol{p}^+$ and $\boldsymbol{p}^0$ indicate the probability of 1 and 0. We define the $\boldsymbol{p}^+$ and $\boldsymbol{p}^0$ according to their distance to 1 and 0, and the $\text{clip}(.)$ operation confirms the probability in [0,1]. As a result, we can use the gradient expectation of the stochastic variable $\widetilde{\boldsymbol{s}}(t)$ for backpropagation, similar to Eq.7:

$$\begin{aligned} \mathbb{E}_{\widetilde{\boldsymbol{s}}(t)}[\frac{\partial \widetilde{\boldsymbol{s}}(t)}{\partial \boldsymbol{m}(t)}] &= \frac{\partial}{\partial \boldsymbol{m}(t)}\mathbb{E}[\widetilde{\boldsymbol{s}}(t)] \\ &= \frac{\partial}{\partial \boldsymbol{m}(t)}(0\times\boldsymbol{p}^0 + 1\times\boldsymbol{p}^+) \; . \\ &= \frac{\partial}{\partial \boldsymbol{m}(t)}\text{clip}(\boldsymbol{m}(t), 0, 1) \end{aligned} \tag{20}$$

In our LIF-BERT/BART baselines, we use Eq.20 in backward to compare the elastic bi-spiking mechanisms with Eq.7. For the spike neurons in linear layers, both our LIF-BERT/BART baseline and SpikeLM set β in Eq.3 as 0.25 the same as other works; for the spike neurons in the key-value cache of self-attention, the β is set to 0, leading to not considering the results of last time step. This would confirm the parallelism of self-attention operations. For the original LIF-BERT* implementation, we apply the same hyperparameters of spiking neurons as the previous SpikeBERT (Lv et al., 2023). As shown in Table 9, our implemented LIF-BERT baseline has higher accuracy.

Table 9. Comparisons between our implemented baseline LIF-BERT and the original LIF-BERT*.

Model	Time	MNLI$_{\text{m/mm}}$	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
LIF-BERT* (Original)	4	35.4/35.2	0.0	50.5	50.9	0.0	0.0	81.2	52.7	34.6
LIF-BERT (Ours)	4	56.8/55.2	70.0	60.6	80.6	14.6	20.0	82.3	53.8	54.9

A.2. Experiment Details

A.2.1. GLUE Benchmark

In the pretraining phase, we keep the settings of the SNN baselines and SpikeLM similar to BERTs. As shown in Table 3, our trained baselines include the original LIF-BERT*, PSN-BERT*, and our implemented LIF-BERT. We utilize the BooksCorpus (Zhu et al., 2015) and English Wikipedia (Devlin et al., 2018) as our training datasets, which include 800M and 2500M words respectively. The same as the approach taken in BERT (Devlin et al., 2018), lists, tables, and headers are ignored in Wikipedia. In the preprocessing stage, our approach aligns with BERT's methodology, employing the WordPiece tokenizer (Devlin et al., 2018) with a 30522 vocabulary size. We set the maximum length of each sentence as 128 tokens. The batch size is set to 512 in training. The entire pretraining encompasses a total of 10^5 steps. The same as ANN conditions, we train SNNs with an AdamW optimizer with a 2×10^{-4} peak learning rate and 0.01 weight decay. We adapt the learning rate by a linear schedule with 5000 warm-up steps. Our experiments show the commonly used pretraining hyperparameters for ANN-based BERTs are general and robust enough for SNNs.

We apply the standard GLUE benchmark (Wang et al., 2018) to evaluate the natural language understanding performance of LIF-BERT and SpikeLM. We follow previous works and use the 8 subsets, including CoLA, STS-B, MRPC, RTE, QQP, MNLI, and QNLI for classification or regression in different scenes. For evaluation, we follow BERT (Devlin et al., 2018) and report F1 scores for QQP and MRPC datasets; Spearman correlations for the STS-B dataset; and accuracy scores for other datasets. In the finetuning phase, we maintain commonly used hyperparameters for ANN-based BERTs. Specifically, we maintain a constant learning rate of 2×10^{-5} and a batch size of 32 for all subsets. We keep the same training epochs as previous BiPFTs (Xing et al., 2024) for different datasets. It's important to note that we do not adapt to the best learning rate or batchsize for GLUE subsets. This can improve performance a lot but may potentially overestimate performance when applied to new tasks.

A.2.2. Generative Benchmarks

For generative tasks, we use the XSUM and CNN-DailyMail summarization benchmarks, and the WMT16 En-Ro dataset as a translation benchmark. XSUM is sampled from the BBC news website, including 226k documents and their one-sentence summarizations. CNN-DailyMail has longer documents and multi-sentence summarizations, and there are 300k data pairs.

For XSUM, CNN-DailyMail, and WMT16 datasets, we use the AdamW optimizer and train 20 epochs with a 128 batch size, and a peak learning rate of 3.5×10^{-4}, 7×10^{-4}, or 1×10^{-4} respectively. We adapt the learning rate by a linear schedular with 0.05× total steps' warm-up. All SNN models are trained on a single node with 8 A800 GPUs.

We distill the SNN-based BART models following the model compression methods (Li et al., 2022). As training objectives, we jointly use the cross-entropy loss and additionally the distillation loss including K-L divergence for the last-layer logits, and L2 loss for the hidden states and attention map in each layer.

A.3. Energy Consumption Metric

Following previous works, we evaluate the overall energy consumption of a neural network by the energy consumption of accumulate operations E_{AC} and multiply-accumulate operations E_{MAC}. Under the 45nm process technology, the 32-bit floating point has an energy consumption of $E_{AC} = 0.9pJ$ and $E_{MAC} = 4.6pJ$ (Horowitz, 2014; Zhang et al., 2022). Moreover, for FP16 operations, we apply the energy consumption of $E_{AC} = 0.4pJ$ and $E_{MAC} = 1.5pJ$ respectively.

For ANNs, the overall energy consumption can be directly evaluated by their MACs. For example, given a linear layer with input dimension m and output dimension n, its energy consumption can be:

$$E_{Linear} = m \times n \times E_{MAC}. \tag{21}$$

For SNNs, the energy consumption is also determined by the spike firing rate r in a certain layer, and also the time step T of the whole SNN. Given the same example as the ANN case, the energy consumption of the linear layer can be:

$$E_{Linear} = m \times n \times E_{AC} \times T \times r, \tag{22}$$

because of the r times sparsity in this spike neuron and T times calculation of the SNN. Different from multiply-accumulate operations (MACs) in ANNs, SNNs convert matrix multiplications to pure accumulate operations (ACs). In Table 3, 5, we evaluate energy by sampling the same 64 pretraining datas from the first batch. In Table 8, we evaluate energy on the XSUM test set.

A.4. Visualization of Spike Neurons

We compare three settings of spike neurons and visualize their input distillations, membrane potentials, and generated spikes in every SNN time step. For comparison, we select the first feed-forward linear layer in the first transformer block. We randomly sample 64 pretraining data of the BERT-architectured models and acquire the input, membrane potential, and output spike of the selected spike neurons.

- As shown in Fig. 6, we visualize the previous LIF neuron with binary spike levels $\{0, 1\}$.
- As shown in Fig. 7, we visualize the LIF neuron with the bidirectional spike encoding proposed in Section 4.1. So that, the generated spikes have the ternary spike levels $\{-1, 0, 1\}$. However, this setting leads to a much higher spike firing rate, causing the energy consumption problem.
- As shown in Fig. 8, we visualize the elastic bi-spiking mechanism in SpikeLMs, which includes bidirectional spiking encoding, spike frequency encoding, and spike amplitude encoding. It is shown that the spikes not only have ternary levels $\{-\alpha, 0, \alpha\}$, but also have the amplitude α. In the frequency aspect, compared with Fig. 7, the elastic bi-spiking mechanism achieves a lower spiking firing rate, demonstrating the effectiveness of spike frequency encoding.

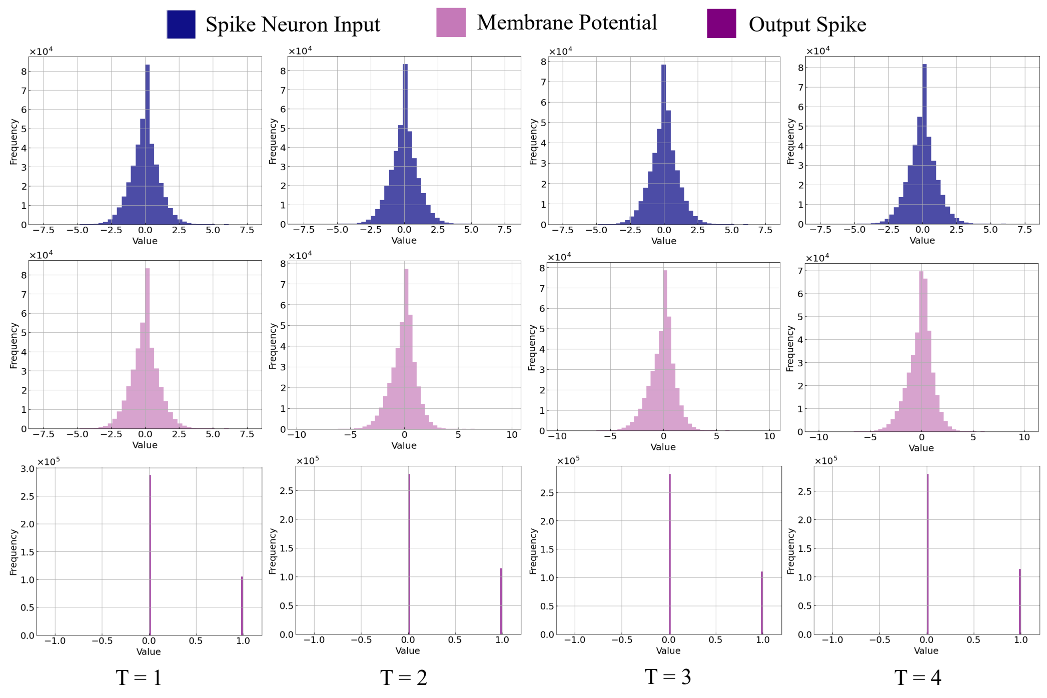


Figure 6. Visualization of the Leaky Integrate-and-Fire (LIF) neuron with $\{0, 1\}$ spike levels.

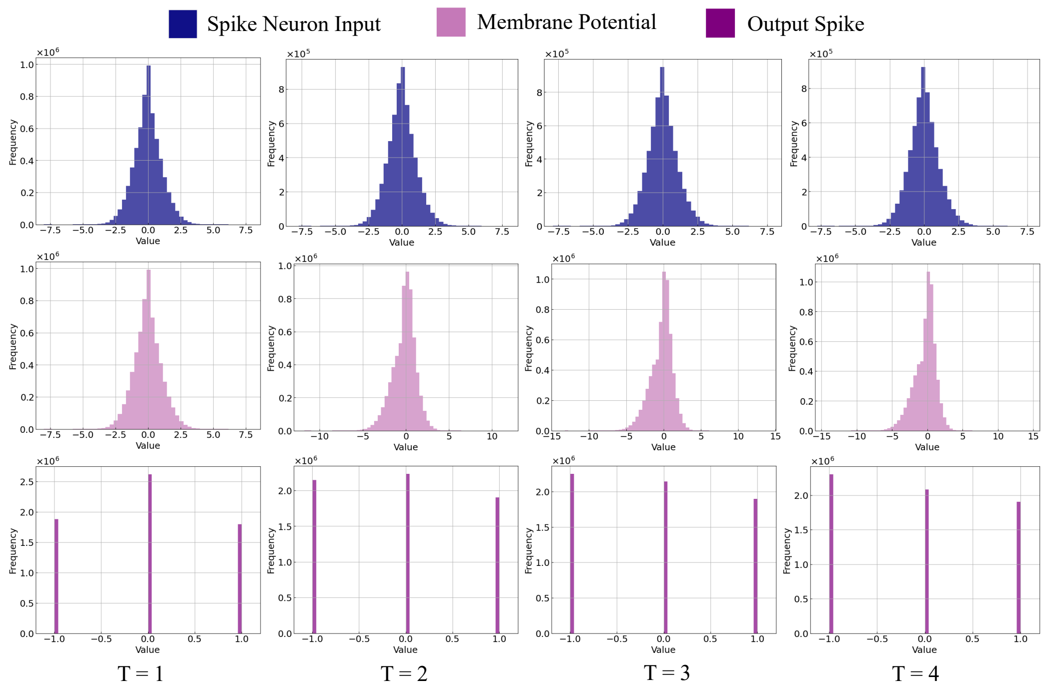


Figure 7. Visualization of the bidirectional spike encoding with $\{-1, 0, 1\}$ spike levels.

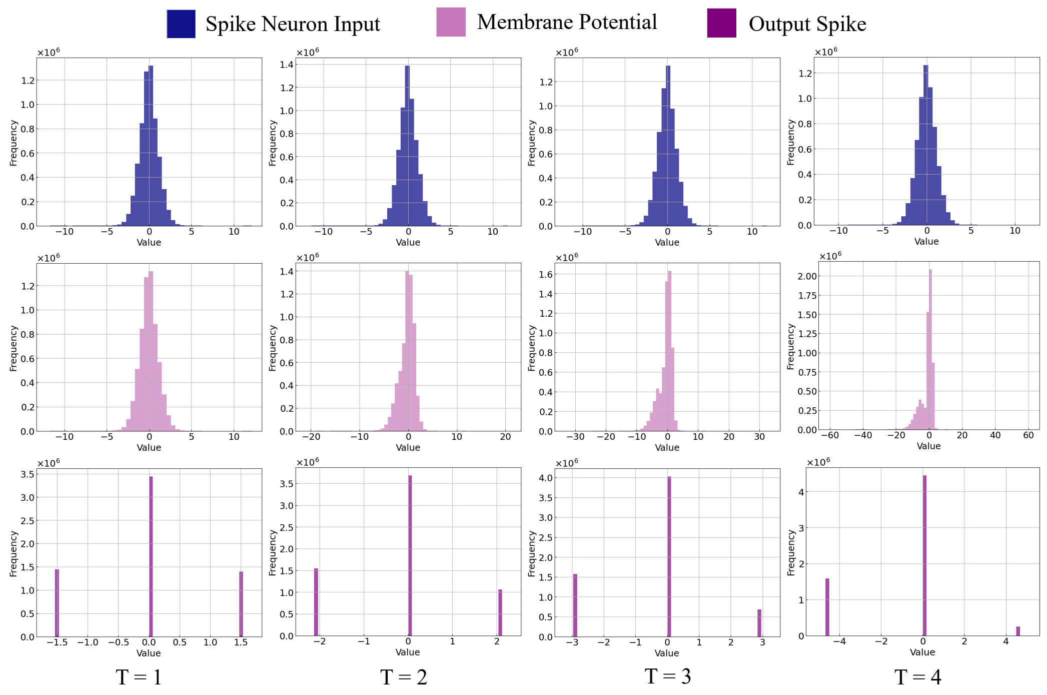


Figure 8. Visualization of the elastic bi-spiking mechanism with $\{-\alpha, 0, \alpha\}$ spike levels.