SCIENCE CHINA
Information Sciences

October 2024, Vol. 67, Iss. 10, 200406:1–200406:14
https://doi.org/10.1007/s11432-024-4101-6

• RESEARCH PAPER •
Special Topic: AI Chips and Systems for Large Language Models

SpikingMiniLM: energy-efficient spiking transformer for natural language understanding

Jiayu ZHANG[1,2], Jiangrong SHEN[1,2*], Zeke WANG[1,5], Qinghai GUO[6], Rui YAN[3], Gang PAN[1,2,4] & Huajin TANG[1,2,4*]

[1]*College of Computer Science and Technology, Zhejiang University, Hangzhou* 310000*, China;*
[2]*The State Key Lab of Brain-Machine Intelligence, Zhejiang University, Hangzhou* 310000*, China;*
[3]*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou* 310000*, China;*
[4]*MOE Frontier Science Center for Brain Science and Brain-Machine Integration, Zhejiang University, Hangzhou* 310000*, China;*
[5]*Collaborative Innovation Center of Artificial Intelligence, Zhejiang University, Hangzhou* 310000*, China;*
[6]*Advanced Computing and Storage Laboratory, Huawei Technologies Co., Ltd., Shenzhen* 518000*, China*

Received 3 January 2024/Revised 23 April 2024/Accepted 9 July 2024/Published online 20 September 2024

Abstract In the era of large-scale pretrained models, artificial neural networks (ANNs) have excelled in natural language understanding (NLU) tasks. However, their success often necessitates substantial computational resources and energy consumption. To address this, we explore the potential of spiking neural networks (SNNs) in NLU—a promising avenue with demonstrated advantages, including reduced power consumption and improved efficiency due to their event-driven characteristics. We propose the SpikingMiniLM, a novel spiking Transformer model tailored for natural language understanding. We first introduce a multi-step encoding method to convert text embeddings into spike trains. Subsequently, we redesign the attention mechanism and residual connections to make our model operate on the pure spike-based paradigm without any normalization technique. To facilitate stable and fast convergence, we propose a general parameter initialization method grounded in the stable firing rate principle. Furthermore, we apply an ANN-to-SNN knowledge distillation to overcome the challenges of pretraining SNNs. Our approach achieves a macro-average score of 75.5 on the dev sets of the GLUE benchmark, retaining 98% of the performance exhibited by the teacher model MiniLMv2. Our smaller model also achieves similar performance to $\text{BERT}_{\text{MINI}}$ with fewer parameters and much lower energy consumption, underscoring its competitiveness and resource efficiency in NLU tasks.

Keywords spiking neural networks, natural language understanding, spiking Transformer, spike-based attention, multi-step encoding, ANN-to-SNN distillation

1 Introduction

The effectiveness of pretrained models, such as BERT [1] and GPT [2], has been emphasized in recent studies across various natural language processing tasks. Built upon the Transformer architecture, both BERT and GPT exhibit a remarkable capacity for efficiently capturing contextual information and addressing long-range dependencies in language, leading to significant breakthroughs in various natural language understanding (NLU) tasks. Despite these achievements, the extensive energy consumption associated with large-scale models, primarily driven by resource-intensive multiplication operations, presents a notable impediment to their widespread deployment on low-power devices. In contrast, spiking neural networks (SNNs) capitalize on discrete spikes for interneuronal communication, establishing spike-based computation as a more energy-efficient alternative compared to artificial neural networks (ANNs). This paper explores the potential integration of SNNs with the Transformer architecture for NLU tasks, unveiling a promising avenue for energy-efficient and biologically inspired language processing. The investigation delves into the distinctive advantages of SNNs, examining their role in mitigating energy consumption challenges while simultaneously ensuring high-performance outcomes in NLU tasks.

* Corresponding author (email: jrshen@zju.edu.cn, htang@zju.edu.cn)

Recent investigations have explored the efficacy of spike-based Transformer models in computer vision tasks, exemplified by studies such as Spikformer [3] and spike-driven Transformer [4]. These models aim to address challenges in spatiotemporal pattern recognition and object tracking [5]. To improve energy efficiency, these spike-based Transformer models replace floating-point multiplications with accumulation operations utilizing spiking neurons. For instance, Spikformer implements the spiking self attention (SSA) component through the spike-form Query, Key, and Value without softmax normalization to reduce the energy consuming. While this approach narrows the performance gap between spike-based Transformers and conventional Transformers, it limits the potential for deployment of the partial spike-based Transformer on neuromorphic chips, thereby forfeiting the benefits of low energy consumption. New developments in spike-based Transformer research have been trying to address this limitation by exclusively incorporating mask and sparse addition operations with completely circumventing multiplication [4]. These studies suggest that the spike-based Transformer holds promise for significantly reducing computational energy while maintaining performance comparable to the Transformer baseline in computer vision tasks.

Despite these strides in computer vision, there remains a notable gap in research investigating the applicability of pure spike-based Transformer models to NLU tasks. Many existing SNN architectures are not directly applicable to natural language tasks. Although there have been several models for NLU tasks, such as SpikingBERT [6] and SpikeBERT [7], the performance of these models still lags far behind ANN models with similar parameter scales. Hence, adapting Transformer architectures to SNNs entails a comprehensive overhaul of the computation components that are not compatible with SNNs, such as attention mechanisms and layer normalization. Meanwhile, due to the discreteness and non-differentiable nature of spike propagation, the feasibility of conducting large-scale language pretraining from scratch for SNNs is even more difficult. How to bring the current remarkable ANN's knowledge into SNNs remains challenging to employ pretrained ANN to improve SNN's training. In addition, unlike image data, where pixel intensity can be easily mapped to firing rates, designing an effective encoding method for text data to maintain information precision and integrity is challenging. As mentioned above, this paper seeks to fill the void by exploring the potential of pure spike-based Transformer models for NLU tasks, considering the unique challenges and opportunities that arise in the intersection of pure spike-based computation and language processing tasks.

To this end, we propose a novel spiking Transformer model tailored for the intricacies of natural language understanding tasks, named SpikingMiniLM. The contributions are summarized as follows:

- The proposed SpikingMiniLM model implements a pure spiking Transformer architecture, liberating itself from the reliance on incompatible computation components with SNNs like layer normalization and original self-attention. The neurologically plausible spike-based attention, and residual connections without normalization are redesigned to fully exploit the energy-efficiency potential of spike-based computation.
- To facilitate fast convergence of SpikingMiniLM model on NLU task, the multi-step encoding and parameter initialization according to the stable firing rate principle are proposed. The ANN-to-SNN knowledge distillation method compositing logits and attention distillation is proposed to harness the expertise of well-pretrained ANN models by transferring their knowledge into SNNs, circumventing the great challenge of training SNNs from scratch through language pretraining.
- The experiments are conducted on the GLUE benchmark. Our model retains 98% performance of the teacher MiniLMv2. And smaller models achieve comparable performance to $\text{BERT}_{\text{MINI}}$, with fewer parameters and less energy consumption. We also verify the effectiveness of the proposed methods by ablation studies. Our work demonstrates that SNNs could have the competitive capability to ANNs in natural language understanding.

2 Related work

Transformer models have emerged as focal points in deep learning research, exhibiting exceptional performance across various NLU tasks. To harness the biological plausibility and energy efficiency of SNNs, researchers have seamlessly integrated the spike-based paradigm into the Transformer.

2.1 Spiking transformer models with hybrid operations

These spiking Transformer models incorporate multiply-and-accumulate (MAC) operations, predominantly influenced by conventional Transformer components, and accumulate (AC) operations driven by spiking neurons. For instance, the Spikingformer model with the spike-driven residual learning architecture is designed in [8], and the ConvBN-Maxpooling-LIF and the SNN-optimized downsampling are proposed to implement the precise gradient backpropagation for the spike-based Transformer [9]. The spike-based Transformers are employed to solve the regression task of event-based human pose tracking [10] and single object tracking [5]. The Spikformer V2 is designed in [11] with the SSA and Spiking Convolutional Stem to enhance the original structure of the Spikformer. To solve the fast prosthetic hand control problem, the Transformers with spiking neurons with the sliding window attention mechanism are designed in [12] to process sequences element-by-element and implement online processing of sEMG signals, which suggests the energy efficiency on the online temporal signal processing with the co-integration of Transformer and SNNs. Furthermore, the neural architecture search method for spiking Transformers named AutoST is developed to release the fixed architecture constraint [13].

2.2 Spiking transformer models with only spike-driven computation

Several spiking Transformer models have endeavored to realize the vision of a pure spike-based Transformer, aiming for exclusive AC operations driven solely by spiking neurons, aligning with the primary objective of our paper. Notably, the spike-driven Transformer introduced in [4] achieves this goal by leveraging only mask and addition operations. The model introduces the spike-driven self-attention mechanism, eliminating multiplication operations, thereby highlighting the inherent potential for low computational energy in spike-based Transformers. The Meta-SpikeFormer is proposed in [14] to extend the spike-driven Transformer into a meta structure to support classification, detection, and segmentation. However, it is worth noting that most of the spike-based Transformers mentioned above cannot be directly applied to NLU tasks. This challenge arises due to the intricate nature of textual information, coupled with the discreteness and non-differentiable characteristics inherent in spike propagation. This complexity necessitates careful consideration and exploration in the development of spike-based Transformers for NLU applications.

2.3 Spiking neural networks for natural language processing

Several prior studies have delved into the domain of natural language processing using SNNs. In the realm of generative language models, SpikeGPT has been introduced in [15], leveraging binary spiking activation units adapted from the receptance weighted key value (RWKV) language model [16]. However, it is noteworthy that the SpikeGPT model incorporates hybrid operations, retaining the conventional Transformer components of RWKV. Beyond generative models, systematic spiking-based encoders have been proposed in prior research. Ref. [17] employed stacked bi-directional SNN layers to encode input text, demonstrating the potential of SNNs to significantly reduce computation energy while maintaining performance comparable to a bi-GRU baseline. Ref. [18] introduced a convolutional SNN model for text classification through converting and fine-tuning TextCNN. Nevertheless, these studies primarily focus on straightforward text sentiment classification tasks and have not explored more intricate and general NLU tasks such as natural language inference (NLI). The newest models of SpikingBERT [6] and SpikeBERT [7] proposed in recent months successfully apply spike-based Transformer into NLU task by virtue of distilling knowledge from pretrained BERT model to train the spiking architecture. The SpikeBERT employs a two-stage, "pretraining + task-specific" knowledge distillation training method to improve the performance of spiking Transformer. The SpikingBERT leverages the average spiking rate of neurons at equilibrium to train a spiking Transformer using an implicit differentiation technique to guarantee the steady-state convergence of the spiking neurons. Despite the above improvements, these models still rely on normalization techniques and their performance trails much behind the ANN models with the similar parameter scale.

To address these gaps, our work emphasizes the exploration of a pure spike-based Transformer for NLU tasks. It is crucial to highlight that the hybrid computing approach adopted by most of the aforementioned spike Transformers makes them challenging to deploy on neuromorphic hardware, limiting their potential to harness the high energy efficiency characteristic of SNNs. Meanwhile, the performance gap between

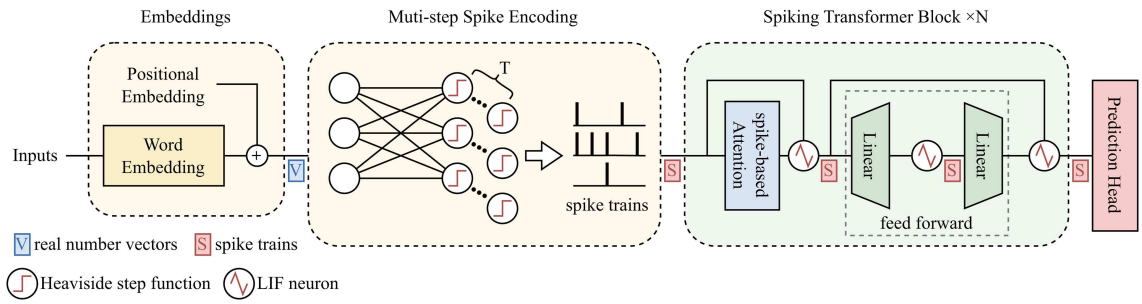

Figure 1 (Color online) Overall architecture of the proposed spiking Transformer. We use the same embeddings as BERT. Then the embedding vectors are converted into spike trains using a multi-step spike encoding approach. The Transformer blocks, including self-attention, feed forward, and residual connections, have been redesigned to facilitate spike-based computation, using spike trains as hidden states. Following the final transformer layer, a task-specific prediction head is used to generate the desired outputs.

ANNs and SNNs on NLU tasks should be narrowed by improving the pure spike-based Transformer training.

3 Method

In this section, we first introduce the detailed implementation of our SpikingMiniLM model. Then we propose a parameter initialization based on stable firing rates, and an ANN-to-SNN knowledge distillation method specifically for the Transformer-based models.

3.1 Spiking transformer

The overall architecture of our SpikingMiniLM model is illustrated in Figure 1. Inspired by BERT, our model is a multi-layer Transformer encoder. We use the same tokenizer and embeddings as BERT. Then we convert the embedding vectors into spike trains using a proposed multi-step spike encoding method. All the Transformer blocks are redesigned for SNN computation, using spike trains as the hidden states. After the last transformer layer, we use a task-specific prediction head to generate the outputs.

3.1.1 *Neuron model*

As the fundamental units in SNNs, the leaky integrate-and-fire (LIF) neuron model is employed in our spiking Transformer. The LIF model is suitable for building large-scale complex SNN models for its simplification and high efficiency. The postsynaptic neuron receives the resultant current produced by afferent spike trains from presynaptic neurons and accumulates membrane potential, then emits a spike once its membrane potential exceeds the firing threshold followed by the membrane potential reset. The dynamics of the LIF models is shown as follows:

$$U_l^t = \tau U_l^{t-1}(1 - S_l^{t-1}) + W S_{l-1}^t, \tag{1}$$

$$S_l^t = \mathcal{F}(U_l^t - u_{\text{th}}), \tag{2}$$

where U_l^t represents the membrane potential of the neurons in the l-th layer at time step t, and S_l^t represents the output spikes. W denotes the synaptic weights, and u_{th} is the firing threshold. $\mathcal{F}(u)$ is the Heaviside step function which gives 0 when $u < 0$ otherwise 1. As $\mathcal{F}$ is non-differentiable, we define the surrogate gradient as follows:

$$\frac{\partial \mathcal{F}}{\partial u} = \exp(-|2u|). \tag{3}$$

We use a learnable membrane time constant τ to enrich the diversity of neurons [19], while maintaining a fixed firing threshold u_{th} to mitigate potential instability during training.

3.1.2 *Multi-step spike encoding*

Rate coding (e.g., Poisson coding) is widely used for image inputs in the previous SNNs work. Input pixels are converted into spike trains proportionally to their intensity. Similarly, Refs. [18,20] used Poisson coding to convert word embeddings into spike trains. However, rate coding cannot utilize the temporal information, and thus has limited capacity for word representation. In this study, we propose a multi-step spike encoding method. At each time step, we perform a linear transformation and then apply the firing function:

$$S^t = \mathcal{F}(XW^t + b^t), \tag{4}$$

where $X \in \mathbb{R}^{s\times d}$ is the original embedding vector, $S^t \in \{0,1\}^{s\times d}$ is the generated spike at time step t, W^t and b^t are learnable parameters. s denotes the length of input sequence, and d denotes the hidden size of the model. By employing distinct parameters at different time steps, the temporal dimension representation capability of the spike trains is enhanced. In our case, the number of time steps is small ($T \leqslant 16$), and thus the resulting parameter count growth is acceptable. During training, we can apply a simple constraint to control the firing rate of the encoding:

$$\mathcal{L}_{\text{fr}} = \left| \frac{1}{T}\sum_{t}^{T} S^t - \text{fr} \right|, \tag{5}$$

where fr is the target firing rate we need. In our experiments, we set fr = 0.1.

3.1.3 *Spike-based attention*

The original self-attention mechanism [21] uses query (Q) and key (K) to compute the attention map, which needs float-point matrix multiplication and softmax function. However, these operations are not compatible with the computational rules of SNNs. We modify the calculation of the attention map. Using spike-form query and key, we change the float-point matrix multiplication into logical AND operation and masked accumulation operation. Given a spike-form input $X \in \{0,1\}^{s\times d}$, our self-attention mechanism can be described as

$$Q = \text{LIF}(XW^Q),\ K = \text{LIF}(XW^K),\ V = XW^V, \tag{6}$$

$$\text{Attention} = QK^\top V/d, \tag{7}$$

where W^Q, W^K, and W^V are learnable parameters. To scale the values of the attention map ($QK^\top$) between 0 and 1, we divide it by d (which can be absorbed into W^V). As we discard the use of the softmax function, the order of calculation between Q, K, and V is changeable. We can calculate $QK^\top$ first (Figure 2(a)) or $K^\top V$ first (Figure 2(b)). As shown in Figure 2, only binary dot product operation and masked accumulation operation are required in the calculation process. The binary dot product operation can be described by the following behavior of neurons: the attention neuron fires spikes only when it receives stimuli from both query and key neuron, which is equivalent to the logical AND operation. Also, the masked accumulation operation can be described by the disinhibition behavior of neurons: the key neuron and query neuron inhibit the signal transmission between the input neuron and output neuron. When it fires spikes, a disinhibition is triggered, allowing the input neuron to send signals to the output neuron.

In practice, we can simply extend the spiking attention to h-heads by splitting $Q, K \in \{0,1\}^{s\times d}, V \in \mathbb{R}^{s\times d}$ into $(q_1,\ldots,q_h),(k_1,\ldots,k_h),(v_1,\ldots,v_h)$, where $q_i, k_i \in \{0,1\}^{s\times d/h}, v_i \in \mathbb{R}^{s\times d/h}$. Then after each calculation of the attention head, they are concatenated and once again projected:

$$\text{MultiHeadAttention} = \text{LIF}\left(\frac{\text{Concat}(q_1k_1^\top v_1,\ldots,q_hk_h^\top v_h)}{d/h}\right)W^O. \tag{8}$$

3.1.4 *Residual connection*

ANN Transformers use layer normalization (LayerNorm) to maintain stability in residual connections. Unfortunately, there is no equivalent operation in SNNs. To address this, we redesigned the residual

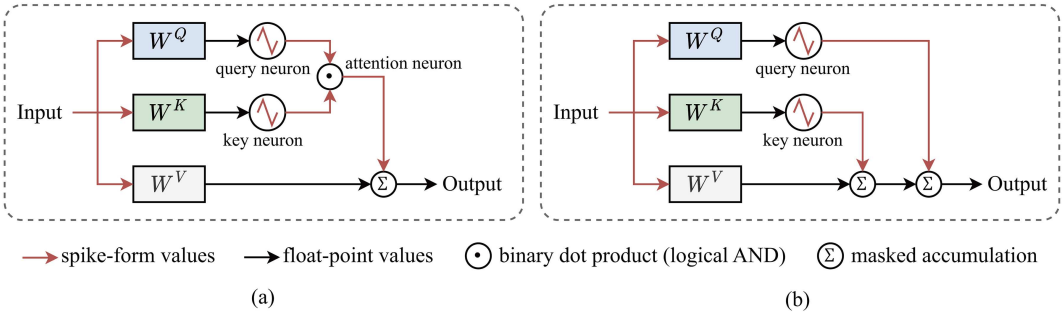

Figure 2 (Color online) Calculation process of spike-based attention. (a) $QK^\top$ first: The spike-form $Q, K \in \{0,1\}^{s\times d}$ are sent to the attention neuron, performing binary dot product (logical AND operation). The outputs of attention neuron are used to do the masked accumulation of V. (b) $K^\top V$ first: Use $K^\top$ and then Q to do the masked accumulation of V.

connection in the Transformer Block. LIF neurons are used to receive stimuli from both direct inputs and sub-layer (attention or feed forward layer) outputs:

$$X' = \mathrm{LIF}(X + \alpha \mathrm{Sublayer}(X)), \tag{9}$$

where $X, X' \in \{0,1\}^{s\times d}$ are in spike form, while the outputs of the sub-layer are current stimuli (float-point). As the firing threshold $u_{\mathrm{th}} \leqslant 1$, an identity mapping of X onto X' is achieved. The surrogate gradient we used for the LIF neurons ensures that the gradients will not be completely truncated in the residual connections. We also rescale the sub-layer outputs by a small factor α to make the network easier to train [22], in conjunction with the parameter initialization method introduced later, enabling our network to work without normalization.

3.1.5 *Prediction head*

The prediction head is a simple fully connected layer, with LIF neurons to generate output spike trains. For a classification task with N categories, we use N output neurons and calculate their firing rates as the model's logits output:

$$z_i = \frac{1}{T}\sum_t^T s_i^t, \tag{10}$$

where s_i^t is the output spike of the i-th neuron at time step t.

However, for regression tasks, generating continuous value is not straightforward for spiking neural networks. If we still use the firing rate as the output value, it will result in limited numerical representation and training instability. Here we introduce a general method for regression tasks. First, we divide the continuous range of target values into N discrete labels $\{L_1, \ldots, L_N\}$. Then we use N output neurons, each of which predicts the probability of the corresponding label. Finally, we calculate the probability-weighted average term:

$$y = \frac{\sum_i^N \exp(kz_i)\cdot L_i}{\sum_i^N \exp(kz_i)}, \tag{11}$$

where the scale factor k controls the smoothness of the softmax output.

3.2 Parameter initialization

In the early stages of training spiking neural networks, the stability of firing rates holds particular significance. The firing rate of a neuron is determined by the magnitude of synaptic weights. Small weights may lead to spike vanishing problems, while large weights may result in spike explosion (fire spikes continually). On one hand, spike vanishing or explosion leads to inefficient information transmission. On the other hand, it implies that the membrane potential is likely to deviate far from the firing threshold, consequently causing gradient vanishing problems. As a result, SNNs may suffer from slow convergence. Therefore, a proper initialization is essential for training SNNs.

Based on the idea of keeping firing rates stable, we propose a general parameter initialization method. Consider the membrane potential change of a neuron within a single time step:

$$\Delta u_i = \sum_j^N w_{ij} s_j, \tag{12}$$

where w_{ij} denotes the synaptic weight from the j-th neuron in presynaptic layer to the i-th neuron in the postsynaptic layer. s_j is the output spike of the j-th neuron. Suppose that the process of firing spikes is independently stochastic, with a firing rate of p, which means it follows a Bernoulli distribution: $P(s_j = 1) = p, P(s_j = 0) = 1 - p$. Let the mean of w_{ij} be 0 and the variance be $\mathrm{Var}[w_{ij}]$. We could have

$$\mathrm{Var}[\Delta u_i] = \sum_j^N \mathrm{Var}[w_{ij} s_j] \tag{13}$$

$$= N(\mathrm{E}[w_{ij}^2 s_j^2] - \mathrm{E}[w_{ij} s_j]^2) \tag{14}$$

$$= Np\,\mathrm{Var}[w_{ij}]. \tag{15}$$

When the number of neurons N is large (> 100), by the central limit theorem, the distribution of Δu_i can be approximated by a normal distribution. Thus we assume that $\Delta u_i \sim \mathcal{N}(0, \mathrm{Var}[\Delta u_i])$. To ensure that the firing rate of the neuron in the postsynaptic layer remains consistent with that of the neuron in the presynaptic layer, we need

$$P(\Delta u_i > u_{\mathrm{th}}) = p \Rightarrow \Phi\left(\frac{u_{\mathrm{th}}}{\sqrt{\mathrm{Var}[\Delta u_i]}}\right) = 1 - p. \tag{16}$$

By a logistic approximation to the cumulative normal distribution [23]:

$$\Phi(x) \approx \frac{1}{1 + \mathrm{e}^{-kx}},\ k = 1.702, \tag{17}$$

we can have

$$\frac{u_{\mathrm{th}}}{\sqrt{\mathrm{Var}[\Delta u_i]}} = \frac{\ln(1-p) - \ln p}{k}, \tag{18}$$

$$\sqrt{\mathrm{Var}[w_{ij}]} = \frac{1}{\sqrt{Np}} \cdot \frac{k u_{\mathrm{th}}}{\ln(1-p) - \ln p}. \tag{19}$$

It is important to note that the above approach only takes into account the neural activity within a single time step. In the multi-step activity of neurons, due to the reset mechanism, the actual behavior will be more complex. In fact, our experiments have shown that, in the case of a small number of time steps ($T \leqslant 16$), this approach still provides a well-performing initialization.

3.3 ANN-to-SNN distillation

Using large-scale pretraining to enhance model capabilities has become a foundational principle in modern natural language processing (NLP) research and applications. However, as SNNs are generally considered to be more challenging to train compared to ANNs because of the non-differentiable spike trains, how to do effective large-scale pretraining on SNNs remains a problem for further exploration. Fortunately, there are many well-pretrained ANN models available today. We can take full advantage of these models by transferring their knowledge into our SNN models. In this study, we propose an ANN-to-SNN knowledge distillation method specifically for the Transformer-based models. As shown in Figure 3, our approach consists of two components: logits distillation and attention distillation.

3.3.1 *Logits distillation*

In logits distillation, the student SNN model learns to mimic the output distribution of the teacher ANN model. The cross-entropy loss (CE) between the student's logits and the teacher's logits is used as the training objective:

$$\mathcal{L}_{\mathrm{logits}} = \mathrm{CE}(z^T/t^T, z^S/t^S), \tag{20}$$

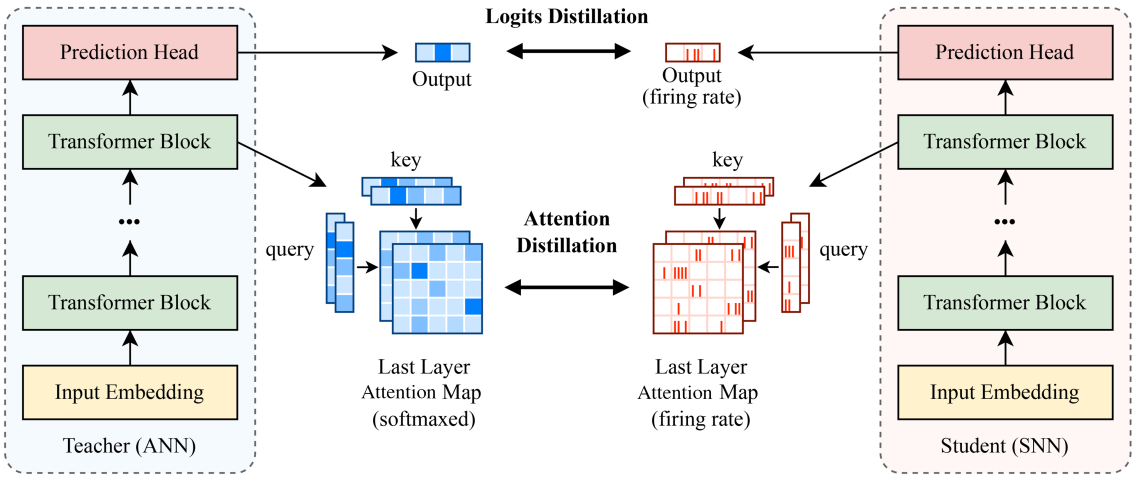

Figure 3 (Color online) Proposed ANN-to-SNN knowledge distillation for SpikingMiniLM, consisting of logits distillation and attention distillation.

where z^T and z^S are the logits output by the teacher and student, respectively. t^T and t^S are the temperatures for distillation. Unlike distillation from ANNs to ANNs, the numerical range of logits output from the SNN model differs from that of the ANN model. Therefore, we apply different temperature factors for the teacher and the student to align the smoothness of their output distributions. In our case, $t^T = 4$ and $t^S = 1$ work well.

3.3.2 *Attention distillation*

Since there are structural gaps between SNNs and ANNs, employing only logits distillation cannot provide a sufficient knowledge transfer. We found that the transfer of attention maps from ANNs to SNNs is highly suitable for our model. In our SNN model, the attention maps have the same function, and numerical range as those in the ANN model. Therefore, we directly minimize the mean squared error (MSE) between the last layer attention maps of the teacher and the student:

$$\mathcal{L}_{\text{attention}} = \frac{1}{h}\sum_{i}^{h} \text{MSE}(A_i^T, A_i^S), \tag{21}$$

where $A_i^T, A_i^S \in \mathbb{R}^{s\times s}$ are attention maps of the i-th head from the last Transformer block of the teacher and student, respectively. Note that the attention map of the teacher is computed by $\text{softmax}(Q_i K_i^\top/\sqrt{d})$, while in the student model, the attention map is computed by the firing rates of the attention neurons:

$$A_i^S = \frac{1}{T}\sum_{t}^{T} Q_i^t K_i^{t\top}/d. \tag{22}$$

Finally, the training objective is the conjunction of the logits distillation loss and the attention distillation loss:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{\text{logits}} + \alpha\mathcal{L}_{\text{attention}}, \tag{23}$$

where α controls the weight of the two distillation components.

4 Experiments

We go beyond the confines of simple text sentiment classification tasks and instead evaluate our approach on the general language understanding evaluation (GLUE) benchmark [24], which consists of 9 tasks for evaluating natural language understanding systems.

We first build a student model with the number of transformer layers $l = 4$ and the hidden size $d = 192$, using MiniLMv2 [25] as the teacher model ($l = 6, d = 384$). As the hidden size of the teacher and student

Table 1 Performance results of our models on the dev sets of GLUE benchmark. We report Matthews correlation for CoLA, F1, and accuracy for MRPC and QQP, the accuracy of both matched and mismatched data for MNLI, Pearson and Spearman correlations for STS-B, and accuracy for other tasks[a)]

Model	#Param	Avg[b)]	CoLA	SST-2	MRPC	QQP	QNLI	RTE	WNLI	MNLI	STS-B
MiniLMv2 (teacher)	22.7M	76.8	35.3	90.5	90.1/86.3	86.1/89.5	90.1	72.2	56.3	82.4/83.0	88.1/88.0
CBoW	–	61.4	4.6	79.5	83.7/75.0	65.5/75.0	62.5	71.9	56.3	57.1	70.6/71.1
BiLSTM	–	66.7	17.6	87.5	85.1/77.9	82.0/85.3	77.0	58.5	56.3	66.7	71.6/72.0
$\text{BERT}_{\text{TINY}}$[c)]	4.4M	67.4	11.9	83.3	85.2/77.5	80.5/84.2	80.0	62.5	56.3	69.3/69.8	77.9/80.6
$\text{BERT}_{\text{MINI}}$[c)]	11.3M	72.4	32.2	85.3	85.4/78.2	83.1/87.2	84.5	64.6	56.3	75.3/76.4	85.4/85.9
SpikeBERT	110M	–	16.9	85.4	82.0/–	68.2/–	66.4	57.5	–	71.4/71.0	–/18.7
SpikingBERT	50M	–	–	86.8	85.2/79.2	–/86.8	85.2	66.1	–	78.1/–	82.2/81.9
Ours ($T = 4$)	7.9M	70.2	19.0	84.4	**85.9/80.2**	81.2/85.7	82.2	63.9	60.6[d)]	71.7/72.5	83.1/83.0
Ours ($T = 16$)	8.3M	**71.9**	21.7	85.3	**86.2/80.1**	**83.2/87.7**	83.8	64.6	63.4[d)]	73.4/74.6	**86.1/85.8**
Ours ($T = 16$)	25.0M	**75.5**	28.1	**89.2**	**89.2/84.8**	**85.2/88.9**	**86.7**	**70.4**	64.8[d)]	**78.8/79.5**	**86.9/86.9**

a) Bold indicates scores notably higher than other models of the similar scale.
b) The GLUE score is the macro-average score over all tasks.
c) $\text{BERT}_{\text{TINY}}$ and $\text{BERT}_{\text{MINI}}$ are fine-tuned from the HuggingFace implementation [28].
d) The results on WNLI are trained from scratch without distillation.

Table 2 Energy consumption estimation

Model	Size	#Param	Computational cost (mJ)	Memory cost (mJ)	Total energy
$\text{BERT}_{\text{TINY}}$	$l = 2, d = 128$	4.4M	1.55	1.80	3.35
$\text{BERT}_{\text{MINI}}$	$l = 4, d = 256$	11.3M	9.91	7.72	17.63
MiniLMv2	$l = 6, d = 384$	22.7M	30.68	24.54	55.22
Ours ($T = 4$)	$l = 4, d = 192$	7.9M	0.40	0.83	1.23
Ours ($T = 16$)	$l = 4, d = 192$	8.3M	1.58	3.33	4.91
Ours ($T = 16$)	$l = 6, d = 384$	25.0M	8.72	12.91	21.63

differs, we use a simple linear auto-encoder to compress the embeddings. For data-rich tasks the Quora question pairs (QQP) and the multi-genre natural language inference corpus (MNLI), we perform the distillation for 10 epochs. For other tasks on small datasets, we initialize the model with the pretrained model from QQP or MNLI task [26], and then proceed train process with more epochs. Furthermore, we build a larger model with the same size as the teacher model.

4.1 Performance comparison on GLUE

In this section, we evaluate the performance of the proposed model on GLUE datasets by comparing it with current ANN and SNN models with similar parameter scales. On one hand, we directly compare our models with ANN models: CBoW and BiLSTM baselines from the original GLUE paper, and similar scale pretrained transformer model $\text{BERT}_{\text{TINY}}$ and $\text{BERT}_{\text{MINI}}$ from [27]. On the other hand, we compare the performance of our model with SpikingBERT [6] and SpikeBERT [7] models that were proposed in recent months. As illustrated in Table 1 [28], our model with 16-time steps achieves the highest average GLUE score of 75.5 with about 25M parameters among these models. We trained our 4-layer models with 4 time steps and 16 time steps. The two models obtain GLUE scores of 70.2 and 71.9, respectively, and the model with 16-time steps retains about 94% performance of its ANN teacher model. The 6-layer model with 16 time steps obtains a GLUE score of 75.5, retaining 98% performance of its ANN teacher model. This model also outperforms SpikingBERT and SpikeBERT models with much fewer parameters. Meanwhile, we also give a comparison of theoretical energy consumption between ANNs and SNNs in Table 2. In contrast to prior studies [3, 4, 7] that solely considered computational energy consumption, we extend our analysis to incorporate the energy overhead associated with memory access. A detailed analysis of the theoretical energy consumption is provided in Appendix A. The results show that even the performance of the proposed 4-layer model with only 4 time steps outperforms the smaller $\text{BERT}_{\text{TINY}}$, and achieves comparable performance to the larger $\text{BERT}_{\text{MINI}}$, with much lower energy consumption.

4.2 Ablation studies

We verify the effectiveness of the proposed method by conducting the ablation studies on our 4-layer models. As illustrated in Table 3, on the two data-rich tasks QQP and MNLI, we ablate the contributions

Table 3 Ablation studies[a)]

Ablation	QQP	MNLI	Avg drop
Our baseline ($l = 4, T = 16$)	**85.4**	**74.0**	–
W/o attention distillation	76.8	67.0	−7.8
Rate coding	82.8	70.8	−2.9
BERT initialization	82.3	70.7	−3.2

a) The variants are validated on data-rich tasks QQP and MNLI, with the 4-layer model using 16 time steps. QQP scores are the average of F1 and accuracy. MNLI scores are the average of matched and mismatched accuracy. Bold indicates the best performance over the ablation settings.

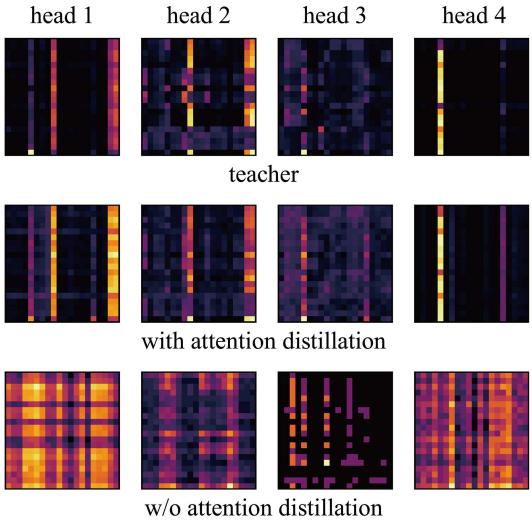

Figure 4 (Color online) Visualization of the attention maps (first 4 heads) from the last transformer layer, during the evaluation of a QQP task.

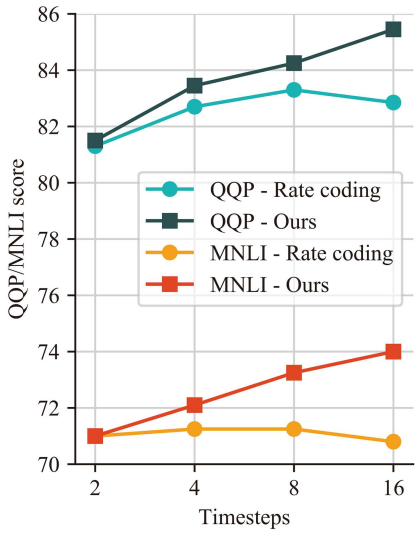

Figure 5 (Color online) QQP/MNLI scores comparison of SpikingMiniLM with the proposed multi-step spike encoding and rate coding methods under different numbers of time steps.

of three components of our method: attention distillation, multi-step spike encoding, and parameter initialization.

4.2.1 *Effectiveness of attention distillation*

We compare the performance of the model using full distillation objective (Baseline) with the model using only logit distillation. As illustrated in Table 3, the results show that the performance without attention distillation decreases significantly. And among the three ablation components, attention distillation contributes the highest score improvement. We visualized the attention maps of the last layer at the end of training to investigate the impact of attention distillation. As illustrated in Figure 4, attention distillation effectively transfers the semantic knowledge from the ANN model to the SNN model, while the model without attention distillation remains comparatively semantically ambiguous.

4.2.2 *Effectiveness of multi-step spike encoding method*

To investigate the effectiveness of the spike encoding method, we replace the proposed multi-step encoding in our model with the commonly used rate coding. Performance degradation is observed in both two tasks as shown in Table 3. We further investigate the performance of these two encoding methods at different numbers of time steps. Figure 5 shows the score variation trend as the time steps in the proposed models with the above two encoding methods. Our method outperforms rate coding across all numbers of time steps. Furthermore, as the number of time steps increases, multi-step encoding tends to preserve more effective information, resulting in a larger performance gap compared with rate coding.

4.2.3 *Effectiveness of parameter initialization method*

We compare our proposed parameter initialization method with the original BERT initialization which uses a normal distribution of std = 0.02. As shown in Table 3, the score of our model with BERT

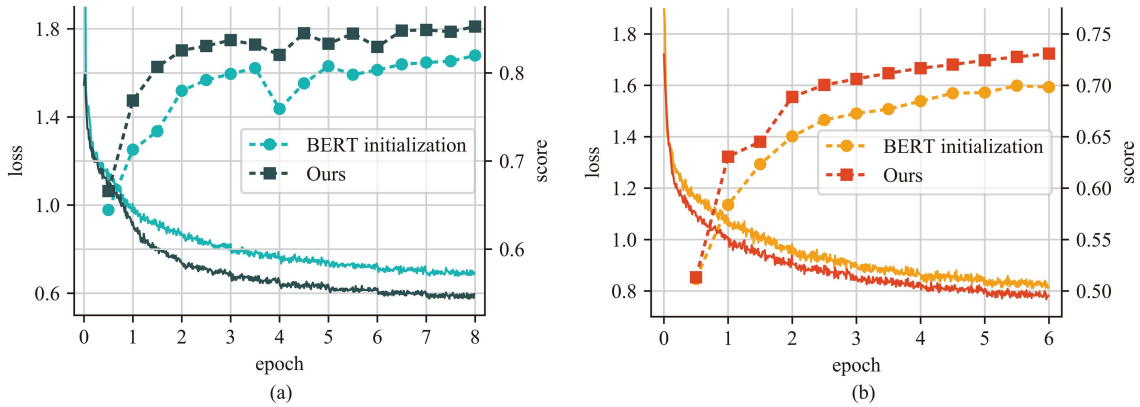

Figure 6 (Color online) Loss and score curves of (a) QQP and (b) MNLI task using different parameter initialization methods for SpikingMiniLM.

initialization drops by about 3 macro scores compared with the baseline using our proposed parameter initialization method on QQP and MNLI tasks. The results suggest the essential role of parameter initialization in training SNNs. Furthermore, as shown in Figure 6, with the proposed parameter initialization method, our model achieves faster convergence with higher scores.

5 Conclusion

In this study, we proposed the SpikingMiniLM, a novel approach that enables SNNs to achieve comparable natural language understanding capabilities to pretrained ANNs with similar parameter scale. We introduced an effective multi-step spike encoding method to convert the continuous text data into spike trains. We redesigned the spike-based attention mechanism and residual connections within the transformer block, achieving a pure spike-based Transformer model. Based on the redesigned residual connections and with the proposed parameter initialization method, we successfully trained our models without normalization techniques. The proposed ANN-to-SNN distillation method enables us to transfer the knowledge from well-pretrained ANN models into our SNN models, circumventing the great challenge of training SNNs from scratch through large-scale language pretraining. The results on GLUE benchmark demonstrated the effectiveness of our approach. This work provides a promising instance for the application of SNNs in NLU.

For future work, we are exploring the application of the proposed spike-based transformer block to generative language models, as well as its scalability to larger language models. By incorporating an attention mask within the attention computation, our model can be readily transformed into a generative language model. And the methods we propose, such as multi-step spike coding, attention mechanisms, and parameter initialization methods, also possess a degree of generality and versatility. However, efficient training strategies are crucial for enabling spike-based transformers to handle the vast amounts of data needed for large language modeling. Future work could involve incorporating additional techniques to ensure stable and efficient learning for these large models. Firstly, the stable training and fast convergence of deep spike-based Transformer models with deeper layers and larger parameters must be solved. The layer-wised techniques such as local training might alleviate the unstable training and speed up convergence speed. Secondly, more efficient ANN-to-SNN distillation approach would be helpful to distill the knowledge of larger pretrained ANN generative language models to SNNs. Thirdly, the more effective temporal feature extraction, such as the flexible time step in large generative language models of SNNs could be helpful to leverage the advantages of temporal dynamics of spike-based Transformer blocks. In addition, the current spike-based Transformer block also adapts to other kinds of tasks such as image processing (e.g., object detection and tracking) and speech processing. For instance, by augmenting our model with an image tokenizer and detection heads, it can be utilized for more efficient and low-power consumption object detection tasks.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62236007, 62306274, 61925603) and Huawei and Zhejiang University Brain-Inspired Computing Joint Innovation Research Project (Grant No. FA201911-1021-2023-01).

References

1 Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. 2018. ArXiv:1810.04805

2 Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 33: 1877–1901

3 Zhou Z, Zhu Y, He C, et al. Spikformer: when spiking neural network meets transformer. In: Proceedings of the 11th International Conference on Learning Representations, 2022

4 Yao M, Hu J, Zhou Z, et al. Spike-driven transformer. 2023. ArXiv:2307.01694

5 Zhang J, Dong B, Zhang H, et al. Spiking transformers for event-based single object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022. 8801–8810

6 Bal M, Sengupta A. SpikingBERT: distilling BERT to train spiking language models using implicit differentiation. 2023. ArXiv:2308.10873

7 Lv C, Li T, Xu J, et al. SpikeBERT: a language Spikformer trained with two-stage knowledge distillation from BERT. 2023. ArXiv:2308.15122

8 Zhou C, Yu L, Zhou Z, et al. Spikingformer: spike-driven residual learning for transformer-based spiking neural network. 2023. ArXiv:2304.11954

9 Zhou C, Zhang H, Zhou Z, et al. Enhancing the performance of transformer-based spiking neural networks by improved downsampling with precise gradient backpropagation. 2023. ArXiv:2305.05954

10 Zou S, Mu Y, Zuo X, et al. Event-based human pose tracking by spiking spatiotemporal transformer. 2023. ArXiv:2303.09681

11 Zhou Z, Che K, Fang W, et al. Spikformer V2: join the high accuracy club on ImageNet with an SNN ticket. 2024. ArXiv:2401.02020

12 Leroux N, Finkbeiner J, Neftci E. Online transformers with spiking neurons for fast prosthetic hand control. 2023. ArXiv:2303.11860

13 Wang Z, Zhao Q, Cui J, et al. AutoST: training-free neural architecture search for spiking transformers. 2023. ArXiv:2307.00293

14 Yao M, Hu J, Hu T, et al. Spike-driven transformer V2: meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In: Proceedings of the 12th International Conference on Learning Representations, 2024

15 Zhu R J, Zhao Q, Eshraghian J K. SpikeGPT: generative pre-trained language model with spiking neural networks. 2023. ArXiv:2302.13939

16 Peng B, Alcaide E, Anthony Q, et al. RWKV: reinventing RNNs for the transformer era. 2023. ArXiv:2305.13048

17 Xiao R, Wan Y, Yang B S, et al. Towards energy-preserving natural language understanding with spiking neural networks. IEEE ACM Trans Audio Speech Lang Process, 2023, 31: 439–447

18 Lv C, Xu J, Zheng X. Spiking convolutional neural networks for text classification. In: Proceedings of the 11th International Conference on Learning Representations, 2022

19 Fang W, Yu Z, Chen Y, et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. 2661–2671

20 Huang J, Serb A, Stathopoulos S, et al. Text classification in memristor-based spiking neural networks. Neuromorph Comput Eng, 2023, 3: 014003

21 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 30

22 Bachlechner T, Majumder B P, Mao H, et al. ReZero is all you need: fast convergence at large depth. In: Proceedings of Uncertainty in Artificial Intelligence, 2021. 1352–1361

23 Bowling S R, Khasawneh M T, Kaewkuekool S, et al. A logistic approximation to the cumulative normal distribution. J Indust Eng Manag, 2009, 2: 114–127

24 Wang A, Singh A, Michael J, et al. GLUE: a multi-task benchmark and analysis platform for natural language understanding. 2018. ArXiv:1804.07461

25 Wang W, Bao H, Huang S, et al. MiniLMv2: multi-head self-attention relation distillation for compressing pretrained transformers. 2020. ArXiv:2012.15828

26 Phang J, Févry T, Bowman S R. Sentence encoders on stilts: supplementary training on intermediate labeled-data tasks. 2018. ArXiv:1811.01088

27 Turc I, Chang M W, Lee K, et al. Well-read students learn better: on the importance of pre-training compact models. 2019. ArXiv:1908.08962

28 Bhargava P, Drozd A, Rogers A. Generalization in NLI: ways (not) to go beyond simple heuristics. In: Proceedings of the 2nd Workshop on Insights from Negative Results in NLP, 2021. 125–135

Appendix A Energy consumption estimation

ANNs mainly perform multiply-accumulate (MAC) operations for matrix multiplication, while SNNs mainly perform synaptic operations (SOPs) when spikes are received, which can be viewed as spike-based accumulate (AC) operations. This distinction has been leveraged in previous work, such as Spikformer [3] and spike-driven Transformer [4], to estimate the computational power consumption of their models. However, the sparse computing characteristics of SNNs also introduce distinct memory access patterns. Therefore, in addition to computational cost, we also incorporate an analysis of memory access into our energy consumption estimation.

Computational cost. In ANNs, the multiplication of matrices $A \in \mathbb{R}^{a\times b}, B \in \mathbb{R}^{b\times c}$ requires $a \times b \times c$ MAC operations. Operations such as softmax, activation, and LayerNorm take one MAC operation per input. Additionally, residual connection requires an extra AC operation. While in SNNs, the multiplication of spike $A \in \{0,1\}^{a\times b}$ and weight $B \in \mathbb{R}^{b\times c}$ degrade into a masked accumulation operation. Due to the event-driven nature of SNNs, no calculation is triggered when the input spike is zero. Hence, the multiplication of A and B requires only $Tp \times a \times b \times c$ AC operations. Additionally, the update of membrane potentials for each neuron requires one MAC operation at each time step.

Table A1 shows the MACs and ACs of a single layer of Transformer and SpikingMiniLM. During inference, SpikingMiniLM can compute $K^\top V$ first, which requires $Tp \times (s \times d/h \times d/h) \times h$ AC operations. Then, performing masked accumulation with Q requires $Tp \times (d/h \times d/h \times s) \times h$ AC operations. Thus, the attention mechanism requires a total of $Tp \times 2sd^2/h$ AC operations.

Table A1 MACs and ACs of a single layer of Transformer and SpikingMiniLM[a)]

		MACs	ACs
Transformer	Q, K, V	$3sd^2$	–
	Attention	$2s^2d + s^2h$	–
	Multi-head	$sd^2 + sd$	sd
	FeedForward	$8sd^2 + 5sd$	sd
SpikingMiniLM	Q, K, V	$T \times 2sd$	$Tp \times 3sd^2$
	Attention	$T \times sd$	$Tp \times 2sd^2/h$
	Multi-head	$T \times sd$	$Tp \times sd^2$
	FeedForward	$T \times 5sd$	$Tp \times 8sd^2$

a) s denotes the sequence length of input data. d is the hidden size of the model and h is the number of heads in multi-head attention. T is the number of time steps and p (≈ 0.1) represents the average firing rate of spiking neurons. Residual connections are aggregated into the corresponding Multi-head and FeedForward sub-layers.

Memory cost. In ANNs, each matrix multiplication $AB = C$ involves reading $(a \times b + b \times c)\mathcal{F}$ bits and writing $(a \times c)\mathcal{F}$ bits, where $\mathcal{F}$ is the number of bits for floating-point numbers, typically 32 or 16. We treat softmax, activation, and LayerNorm as optimized in-place operations, and thus they do not entail read or write operations to RAM. Residual connection requires an extra read operation. In SNNs, just like in the computational process, no memory access is triggered when the input spike is zero. The activity of neurons' receiving spike trains $A \in \{0,1\}^{T\times a\times b}$ and generating spike trains $C \in \{0,1\}^{T\times a\times c}$ require reading $Tp \times ab$ bits of input spikes and $Tp \times bc\mathcal{F}$ bits of synaptic weights, writing $Tp \times ac$ bits of output spikes.

Table A2 Memory access of a single layer of Transformer and SpikingMiniLM

		Read	Write
Transformer	Q, K, V	$(3sd + 3d^2)\mathcal{F}$	$3sd\mathcal{F}$
	Attention	$(3sd + s^2h)\mathcal{F}$	$(s^2h + sd)\mathcal{F}$
	Multi-head	$(2sd + d^2)\mathcal{F}$	$sd\mathcal{F}$
	FeedForward	$(6sd + 8d^2)\mathcal{F}$	$5sd\mathcal{F}$
SpikingMiniLM	Q, K, V	$Tp \times (3sd + 3d^2\mathcal{F})$	$Tp \times 2sd + T \times sd\mathcal{F}$
	Attention	$Tp \times (2sd + sd\mathcal{F} + d^2/h\mathcal{F})$	$Tp \times sd + T \times d^2/h\mathcal{F}$
	Multi-head	$Tp \times (2sd + d^2\mathcal{F})$	$Tp \times sd$
	FeedForward	$Tp \times (6sdS + 8d^2\mathcal{F})$	$Tp \times 5sd$

Table A2 shows the memory access of a single layer of Transformer and SpikingMiniLM. In our spike-based attention mechanism, V requires the writing of $T \times sd\mathcal{F}$ bits. During the calculation of attention, storing the computed results of $K\top V$ requires writing $T \times d^2/h\mathcal{F}$ bits. Then, computation with Q requires a corresponding read of $Tp \times d^2/h\mathcal{F}$ bits. Finally, the theoretical energy consumption of a single Transformer layer is calculated as

$$E_{\text{Transformer}} = E_{\text{MAC}} \times (12sd^2 + 2s^2d + s^2h + 6sd) \tag{A1}$$
$$+ E_{\text{AC}} \times 2sd \tag{A2}$$
$$+ E_{\text{Read}} \times (14sd + 12d^2 + s^2h)\mathcal{F} \tag{A3}$$
$$+ E_{\text{Write}} \times (10sd + s^2h)\mathcal{F}. \tag{A4}$$

And for a single SpikingMiniLM layer,

$$E_{\text{SpikingMiniLM}} = E_{\text{MAC}} \times T \times 9sd \tag{A5}$$
$$+ E_{\text{AC}} \times Tp \times (12sd^2 + 2sd^2/h) \tag{A6}$$
$$+ E_{\text{Read}} \times T \times (13psd + (12pd^2 + psd + pd^2/h)\mathcal{F}) \tag{A7}$$

Table A3 Energy consumption estimation

Model	Size	MACs	ACs	Read (Mb)	Write (Mb)	Computational cost (mJ)	Memory cost (mJ)	Total energy
$\text{BERT}_{\text{TINY}}$	$l=2, d=128$	337M	0.26M	105	75.5	1.55	1.80	3.35
$\text{BERT}_{\text{MINI}}$	$l=4, d=256$	2155M	1.05M	470	302	9.91	7.72	17.63
MiniLMv2	$l=6, d=384$	6670M	2.36M	1472	981	30.68	24.54	55.22
Ours ($T=4$)	$l=4, d=192$	14.2M	367M	29.9	53.3	0.40	0.83	1.23
Ours ($T=16$)	$l=4, d=192$	56.6M	1470M	120	213	1.58	3.33	4.91
Ours ($T=16$)	$l=6, d=384$	170M	8818M	632	659	8.72	12.91	21.63

$$+ E_{\text{Write}} \times T \times (9psd + (sd + d^2/h)\mathcal{F}). \tag{A8}$$

Therefore, we could conduct the energy consumption estimation by considering both the computation cost and memory cost. In line with previous studies, we use the data from the reference[1)], where $E_{\text{MAC}} = 4.6$ pJ, $E_{\text{AC}} = 0.9$ pJ, $E_{\text{Read/Write}} \approx 10$ pJ/bit. For an input sequence with a length of $s = 512$, we estimate the energy consumption in Table A3.

1) Horowitz M. 1.1 computing's energy problem (and what we can do about it). In: Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014. 10–14.