SpikeZIP-TF: Conversion is All You Need for Transformer-based SNN

Kang You [*1] **Zekai Xu** [*1] **Chen Nie** [1] **Zhijie Deng** [1] **Qinghai Guo** [2] **Xiang Wang** [2] **Zhezhi He** [1]

Abstract

Brain-inspired spiking neural network (SNN) has attracted great attention because of its high efficiency over the traditional artificial neural network (ANN). Currently, ANN to SNN conversion methods can produce SNNs using convolution neural network as backbone architecture which achieves on-par accuracy w.r.t ANNs with ultra-low latency (*e.g.*, 8 time-steps) on computer vision (CV) tasks. Although Transformer-based networks have achieved the prevailing precision on both CV and natural language processing (NLP) tasks, Transformer-based SNNs are still lagging behind their ANN counterparts. In this work, we introduce a novel ANN-to-SNN conversion method, called SpikeZIP-TF, through which ANN and the converted SNN are exactly equivalent thus incurring no accuracy degradation. SpikeZIP-TF achieves 83.82% Top-1 accuracy on the CV image classification task with ImageNet dataset and 93.79% accuracy on the NLP dataset (SST-2), which both are higher than SOTA Transformer-based SNNs. The code is publicly available at: https://github.com/Intelligent-Computing-Research-Group/SpikeZIP-TF

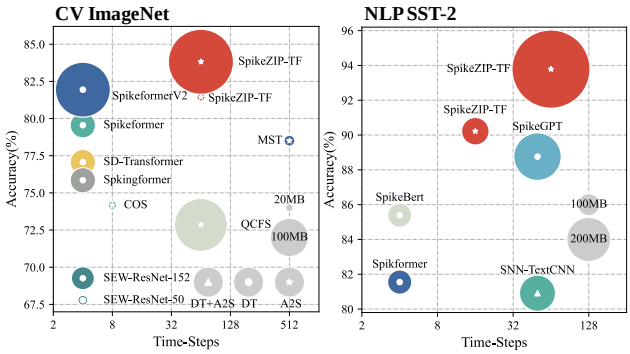


Figure 1. **Comparison of Transformer-based SNNs.** The markers, represented by circles, star, and triangle shapes, denote the direct learning (DT) method, ANN-to-SNN (A2S) conversion method and using both the DT and A2S methods, respectively, where the area of the scatter corresponds to the model size. Results show that the pikeZIP-TF generated SNN achieves higher accuracy with greater model size than the other recent SNNs. The largest model size of SpikeZIP-TF on ImageNet is 304.33 MB.

1. Introduction

Spiking neural network (SNN) (Maass, 1997) is a type of biologically plausible neural network inspired by brains of living organisms. Unlike modern ANNs (LeCun et al., 2015) using continuous activation value to propagate information between neurons synchronously, SNNs utilize discrete events or "spikes" for asynchronous neuron-to-neuron communication and processing (Merolla et al., 2014; Davies et al., 2018). Meanwhile, in the field of deep learning, Transformers (Vaswani et al., 2017) have made significant strides and revolutionized various applications. Inspired by the architecture of the ANN Transformer, introducing the Transformer structure to SNN to improve the SNN accuracy is an emerging trend (Zhou et al., 2022; Lv et al., 2023).

Currently, methods to train Transformer-based SNN come in twofold: *directly training (DT)* and *ANN-to-SNN Conversion (A2S)* (Roy et al., 2019). The DT methods leverage back-propagation through time (BPTT) to update the synaptic weights of SNN. Unfortunately, due to the inaccurate gradient approximation (Neftci et al., 2019) for the non-differential SNN neuron, *e.g.*, integrate and fire (IF) neuron, an accuracy gap persists between SNN and its ANN counterpart (Zhou et al., 2024; Lv et al., 2023; Zhou et al., 2023).

Rather than directly training an SNN, the A2S methods transfer the parameters of the pre-trained ANN to its SNN counterpart (Cao et al., 2015) that yields close-to-ANN accuracy. The previous A2S algorithm achieves the on-par accuracy to ANN with ultra-low latency (*e.g.*, 8 time-steps) on convolution-based SNN (Hu et al., 2023). However, when leveraging the existing A2S algorithms to produce Transformer-based SNNs, it is difficult to build the equivalence between SNN operators and ANN operators,

[*]Equal contribution [1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China [2]Huawei Technologies, Shenzhen, China. Correspondence to: Zhezhi He <zhezhi.he@sjtu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

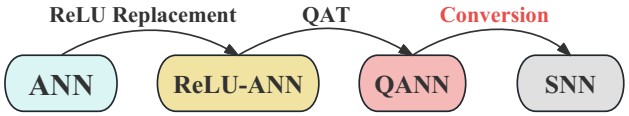


Figure 2. The conversion pipeline of SpikeZIP-TF.

like softmax, layer normalization (Ba et al., 2016), and attention (*i.e.*, dot product with two non-stationary matrix). The inequivalence hinders the development of A2S algorithm for Transformer-based SNN. Correspondingly, we propose SpikeZIP Transformer (*aka.* SpikeZIP-TF), which introduces the spike-equivalent self-attention (SESA), Spike-Softmax and Spike-LayerNorm, while maintaining the equivalence between the operators of ANN and SNN. Figure 1 shows the overall results of SpikeZIP-TF on CV (ImageNet) and NLP task (SST-2).

Contributions of SpikeZIP-TF are summarized as follows:

- We propose an ANN-SNN conversion method called SpikeZIP-TF that builds the equivalence between the activation-quantized Transformer-based ANN and SNN by supporting the SNN-unfriendly operators of ANN (*e.g.*, softmax and layernorm) in converted SNN.
- SpikeZIP-TF deals with both the CV and NLP tasks by converting the quantized vision Transformer (ViT) (Dosovitskiy et al., 2020) and Roberta (Devlin et al., 2018) to SNN and achieves the state-of-the-art accuracy than competing Transformer-based SNNs.

2. Background and Related Works

Spiking Neurons. Integrate and fire (IF) neuron is widely utilized in the A2S methods due to the mathematical similarity between the IF neuron and ReLU (Bu et al., 2023). Nevertheless, the error in the accumulated output of the IF neuron to the ReLU persists, which hampers the accuracy of SNNs. To deal with the error, a recently emerged SNN neuron, which we name it as bipolar integrate and fire with spike tracer (ST-BIF) neuron, are introduced to further approaching the equivalence to the quantized-ReLU (Q-ReLU) function (Li et al., 2022; Hu et al., 2023). Unfortunately, the native ST-BIF neuron is merely equivalent to the quantized-ReLU function rather than the quantized function for activation in attention which is widely used in self-attention in Transformer.

Learning Methods of SNN comes in twofolds: direct training (DT) and ANN-to-SNN conversion (A2S). The DT algorithm employs the back-propagation through time (BPTT) (Lee et al., 2016; Shrestha & Orchard, 2018) with surrogate gradient (Neftci et al., 2019) to train an SNN for a fixed time-step. However, due to the errors of esitamted gradient during training, a loss gap exists between the SNN obtained through DT and its ANN counterpart. In the realm of A2S algorithm, the ReLU/Q-ReLU layers in ANN are substituted with spiking neurons, resulting in an equivalent SNN model that achieves comparable accuracy to the ANN (Bu et al., 2023; Hu et al., 2023; Li et al., 2022; Hao et al., 2023; Wang et al., 2023; Rueckauer et al., 2017). Compared to DT, A2S-based SNNs not only achieve higher accuracy, but also consume lower training cost in terms of time and memory. Such characteristic makes the SNN more amenable to model scaling and deployment on neuromorphic hardware.

Table 1. **Summary of Transformer-based SNNs.** SSA: spike self-attention; SDSA: spike-driven self-attention; ASR-SA: average spiking rate self-attention; S-RWKV denotes spiking-RWKV; VSA: vanilla self-attetion; QVAS: quantized vanilla self-attetion.

Methods	Algorithm	Neuron	Attention	NAS	Pretrain	Distill
SpikformerV1	BPTT	LIF	SSA			
SD-Transformer	BPTT	LIF	SDSA			
Spikingformer	BPTT	LIF	SSA			
Auto-Spikformer	BPTT	LIF	SSA	✓		
SPIKEBERT	BPTT	LIF	SSA			✓
SpikingBERT	Implicit Diff	LIF	ASR-SA			✓
SpikeGPT	BPTT	LIF	S-RWKV		✓	
SpikformerV2	BPTT	LIF	SSA		✓	
MST	Conversion	IF	VSA		✓	

As depicted in Figure 2, SpikeZIP-TF adheres to the conversion pipeline established by prior A2S methods (Bu et al., 2023; Li et al., 2022; Hu et al., 2023). The process initiates with the replacement of activation functions in the ANN to ensure that only ReLUs are present. Subsequently, quantization-aware training (QAT) (Gholami et al., 2022; He & Fan, 2019) is applied to acquire a low bit-width and high-accuracy QANN. Finally, the QANN undergoes conversion to an SNN by substituting the ReLU activation(neuron) with specific spiking neuron, without accuracy degradation.

Transformer-based SNNs. The Transformer-based ANN (Vaswani et al., 2017; Dosovitskiy et al., 2020) comprises three key components: 1) an embedding layer designed to convert image patches or words in sentences to tokens for enhanced learning; 2) cascaded Transformer encoders, incorporating several self-attention and multi-layer perceptron blocks, aimed at learning spatial-temporal features within the tokens; 3) shallow head responsible for executing specific tasks. Notably, the Transformer-based ANNs attain state-of-the-art accuracy in both the CV and NLP tasks, thereby catalyzing the advancement of Transformer-based SNNs, as tabulated in Table 1.

Transformer-based SNNs are initially pursued through DT methods, such as Spikformer v1/v2 (Zhou et al., 2022; 2024), SD-Transformer (Yao et al., 2023) and SpikeGPT

(Zhu et al., 2023). Spikingformer (Zhou et al., 2023) tackles *non-spike computation* challenges by swapping the positions of convolution and batch normalization. To further improve the accuracy of the SNN, SPIKEBERT (Lv et al., 2023) and SpikingBERT (Bal & Sengupta, 2023) employ knowledge distillation algorithms to transfer information from ANN. Furthermore, SpikeGPT (Zhu et al., 2023) and spikformer V2 (Zhou et al., 2024) leverage the pre-train algorithm for better-initiated model weights, while Auto-Spikformer (Che et al., 2023) leverages the network architecture search (NAS) to identify a spiking Transformer with high accuracy and low energy consumption. Besides, SpikingBERT (Bal & Sengupta, 2023) adopts an implicit differentiation algorithm, distinct from previous works using BPTT, for SNN training. In contrast, for A2S methods, MST (Wang et al., 2023) replaces the QCFS (Bu et al., 2023) activation function to ReLU function in ANN and converts the Transformer-based ANN to SNN.

The limited adoption of A2S methods in Transformer-based SNNs stems from the challenge of establishing mathematical equivalence between operators in quantized Transformer-based ANNs and SNNs. Existing A2S methods have yet to tackle the equivalence issues associated with the following operators: *self-attention, softmax, and layer normalization* (Ba et al., 2016). In SpikeZIP-TF, we address the operator equivalence challenge by introducing a novel spiking equivalence self-attention (*aka*. SESA). Additionally, for softmax and layer-norm, we employ a differential algorithm to design their equivalent spiking forms. By integrating our spiking operators, SpikeZIP-TF establishes equivalence between quantized Transformer-based ANNs and SNNs.

3. Methods

3.1. Dynamics of ST-BIF+ Neuron

In SpikeZIP-TF, to address the inequivalence between ST-BIF neuron and quantized function in Transformer, we propose the ST-BIF+ whose dynamics can be expressed as:

$$\begin{aligned} V_t &= V_{t-1} + V_t^{\text{in}} - V_{\text{thr}} \cdot \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \\ S_t &= S_{t-1} + \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \\ \Theta(V, V_{\text{thr}}, S) &= \begin{cases} 1; & V \geq V_{\text{thr}} \ \& \ S < S_{\text{max}} \\ 0; & \text{other} \\ -1; & V < 0 \ \& \ S > S_{\text{min}} \end{cases} \end{aligned} \tag{1}$$

where the notations are specified in Table 2. Compared to the ST-BIF neuron, the ST-BIF+ neuron expands the minimum value of the spike tracer from zero to the lower clamp bound in the quantized function as follows:

$$\text{Quantize}(x) = s \cdot \text{clamp}(\text{round}(x/s), \alpha, \beta) \tag{2}$$

where s represents the quantization scale size, α, β represent the minimum and maximum of clamp range in the quantizer.

Table 2. Summary of mathematical notations used in this paper.

Notation	Description
V_t	potential of neuron membrane at time-step t
V_{thr}	threshold voltage for neuron to fire a spike
$V^{\text{in}}, V^{\text{out}}$	input or output voltage of neuron
S_t	spike tracer at time-step t
$S_{\text{max}}, S_{\text{min}}$	maximum and minimum bound of spike tracer
$\text{clip}(x, \alpha_{\text{min}}, \alpha_{\text{max}})$	clip function that limits x within α_{min} and α_{max}
$\Theta(V, V_{\text{thr}}, S)$	output spike decision function of ST-BIF+
T_{eq}	time-step of SNN enters the equilibrium state

By setting $V_{\text{thr}} = s, S_{\text{min}} = \alpha, S_{\text{max}} = \beta$, the accumulated output of ST-BIF+ is equivalence to the quantized function.

3.2. Transformer-based SNN in SpikeZIP-TF

In this section, we first elaborate on the network topology of Transformer-based SNN in SpikeZIP-TF. Then, for equivalent ANN-SNN conversion, we introduce SNN-friendly operators in SpikeZIP-TF including SESA, Spike-Softmax and Spike-Layernorm.

3.2.1. Architecture Overview

As shown in Figure 3, the architecture of Transformer-based SNN generated by SpikeZIP-TF is almost identical to the vanilla Transformer, which consists of an embedded layer, a Transformer encoder, and a shallow head. Given a target Transformer-based ANN and to obtain its SNN counterpart, we coduct the following procedures:

1. Quantizers are inserted both ahead of and behind the matrix multiplication operators to acquire a quantized Transformer, which is consistent with prior quantized Transformer work, *e.g.*, I-BERT (Kim et al., 2021);

2. Once the model with quantization function is trained with QAT, quantized functions are replaced with ST-BIF+ neuron to ensure the inputs and outputs of matrix multiplication are in the form of spike trains;

3. SNN-unfriendly operators in the quantized Transformer (*e.g.*, Softmax, LayerNorm and dot product) are subsituted with SNN-friendly operators (*e.g.*, Spike-Softmax, Spike-LayerNorm and spiking dot product).

3.2.2. Embedding & Head for SNN

Compared to the embedding layer in QANN, an ST-BIF+ neuron layer is introduced following the embedding layer to facilitate the conversion of analog input charge into spike trains for Transformer encoding. For the head of SNN, following the previous SNN works (Wang et al., 2023; Zhou et al., 2024; Yao et al., 2023), we use membrane potential in the ST-BIF+ rather than spike trains as the output.

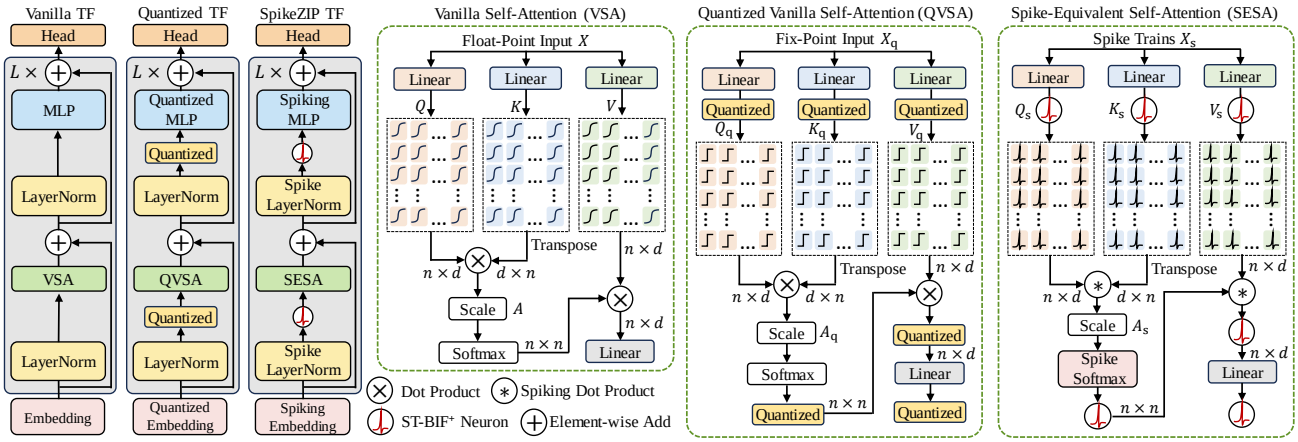
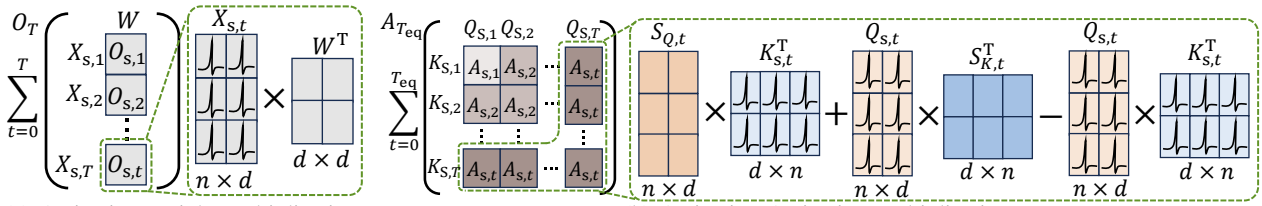


Figure 3. **Architecture of Transformer-based SNN in SpikeZIP-TF.** Compared to the vanilla Transformer, SpikeZIP-TF inserts the ST-BIF$^+$ neuron ahead of and behind the matrix multiplication operations and substitutes SNN-unfriendly operators (dot product, Softmax and LayerNorm) with SNN-friendly ones (spiking dot product, Spike-Softmax and Spike-LayerNorm). TF: Transformer; n: sequence length; d: token dimension; $\{\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{A}\}$, $\{\boldsymbol{Q}_\mathrm{q}, \boldsymbol{K}_\mathrm{q}, \boldsymbol{V}_\mathrm{q}, \boldsymbol{A}_\mathrm{q}\}$, $\{\boldsymbol{Q}_\mathrm{s}, \boldsymbol{K}_\mathrm{s}, \boldsymbol{V}_\mathrm{s}, \boldsymbol{A}_\mathrm{s}\}$: {query, key, value, attention array}, {theirs quantized form} and {spike form}.



(a) Activation-Weight Multiplication (b) Activation-Activation Multiplication

Figure 4. **The process of matrix multiplication in SESA.** The bracket part in (a) and (b) corresponds to Equation (3) and Equation (4) respectively. (a) $\boldsymbol{X}_{\mathrm{s},t}, \boldsymbol{O}_{\mathrm{s},t}$ represent the input and output spike trains in SNN at time-step t. (b) $\boldsymbol{Q}_{\mathrm{s},t}, \boldsymbol{K}_{\mathrm{s},t}$ denote the query and key in SNN at time-step t; $\boldsymbol{S}_{\mathrm{Q},t}, \boldsymbol{S}_{\mathrm{K},t}$ are the spike tracers in the neuron layers, which store the accumulated output for query and key. At each time-step, we utilize the accumulated output in the spike tracer to perform AA multiplication via three matrix multiplications.

3.2.3. Spike-Equivalent Self-Attention (SESA)

As a pivotal component within the SpikeZIP-TF, the design of spike-equivalent self-attention (SESA) adheres to two fundamental principles: 1) ensuring that the accumulated output remains *equivalent to quantized vanilla self-attention*, and 2) *aligning with the computing paradigm in SNN*.

SESA is implemented as described in Figure 3. There are two types of activation matrix multiplication in SESA, *i.e.*, 1) *Activation-Weight* (*aka.* AW) multiplication in the linear layer, where one operand is stationary while the other is dynamically generated; 2) *Activation-Activation* (*aka.* AA) multiplication, where both operands are generated on-the-fly. It occurs between the query $\boldsymbol{Q}$ and key $\boldsymbol{K}$, as well as attention array $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{K}$ and value $\boldsymbol{V}$. The AA multiplication is also presented as the spiking dot product in Figure 3, *i.e.*, both tensors are composed by spikes.

AW Multiplication. Thanks to the equivalence between ST-BIF$^+$ and the quantized function, we can conclude that:

$$\boldsymbol{O}_{T_\mathrm{eq}} = \boldsymbol{W} \cdot \boldsymbol{X}_\mathrm{q} = \sum_{t=0}^{T_\mathrm{eq}} (\boldsymbol{W} \cdot \boldsymbol{X}_{\mathrm{s},t}); x_{\mathrm{s},t} \in \{0, \pm 1\} \tag{3}$$

where $\boldsymbol{W}$ denotes the weight of linear layer, while other notations are summarized in Table 2 and Figure 3. Note that, when $t > T_\mathrm{eq}$, all neurons enter the *equilibirum state*, during which they neither receive nor fire any spike.

AA Multiplication. Taking the multiplication between query and key as an example, it can be written as:

$$\begin{aligned} \boldsymbol{A}_{T_\mathrm{eq}} &= \boldsymbol{Q}_\mathrm{q} \cdot \boldsymbol{K}_\mathrm{q} = \sum_{t_1=0}^{T_\mathrm{eq}} \boldsymbol{Q}_{\mathrm{s},t_1} \cdot \sum_{t_2=0}^{T_\mathrm{eq}} \boldsymbol{K}_{\mathrm{s},t_2} \\ &= \sum_{t=0}^{T_\mathrm{eq}} \boldsymbol{S}_{\mathrm{Q},t} \cdot \boldsymbol{K}_{\mathrm{s},t}^T + \boldsymbol{Q}_{\mathrm{s},t} \cdot \boldsymbol{S}_{\mathrm{K},t}^T - \boldsymbol{Q}_{\mathrm{s},t} \cdot \boldsymbol{K}_{\mathrm{s},t}^T \end{aligned} \tag{4}$$

Table 3. **Complexity analysis of SpikeZIP-TF** for its operators of AW(AA)-Multiplication (*abbr*. Mult), Spike-Softmax (SSoftmax), and Spike-LayerNorm (SLayerNorm). n: sequence length; d: token demension; T: # time-steps; γ: Performance ratio of one operation in ANN versus SNN.

	Network	AW-Mult	AA-Mult	SSoftmax	SLayerNorm
Spatial	SNN	$O(nd+d^2)$	$O(nd)$	$O(n^2)$	$O(nd)$
Temporal		$O(Tnd^2)$	$O(Tnd^2)$	$O(Tn^2)$	$O(Tnd)$
	Network	AW-Mult	AA-Mult	Softmax	LayerNorm
Spatial	(Q)ANN	$O(nd+d^2)$	$O(nd)$	$O(1)$	$O(1)$
Temporal		$O(\gamma nd^2)$	$O(\gamma nd^2)$	$O(\gamma n^2)$	$O(\gamma nd)$

where $\boldsymbol{Q}_{\mathrm{s},t_1}, \boldsymbol{K}_{\mathrm{s},t_2}$ represent the spike trains of query and key at time-step t_1 and t_2 respectively, $\boldsymbol{S}_{\mathrm{Q},t}, \boldsymbol{S}_{\mathrm{K},t}$ represent the accumulated spike trains of query and key which are stored in ST-BIF$^+$ neuron, while $\boldsymbol{A}_{T_{\mathrm{eq}}}$ denotes the attention array accumulated during T_{eq}.

To compute Equations (3) and (4) in SESA, the matrix multiplication must be decomposed into sub-operations for each time-step. To address this, we propose a novel calculation method for matrix multiplication. The detailed processes of AW and AA multiplication are outlined in Figure 4. For AW multiplication, as depicted in Figure 4(a), we perform matrix multiplication on the input spike trains per time-step to generate the output. In contrast, for AA multiplication, we utilize the accumulated output in the spike tracer to compute the attention array through three matrix multiplications, as illustrated in Figure 4(b). With the computation depicted in Figure 4, we achieve a lossless conversion from QVSA to SESA while adhering to the computing paradigm in SNN.

3.2.4. Spike-Softmax & Spike-LayerNorm

To enable Softmax and LayerNorm operations in SNN, we introduce Spike-Softmax and Spike-LayerNorm, which are equivalent to their ANN counterparts. The process of Spike-Softmax and Spike-LayerNorm can be expressed as:

$$\begin{aligned} \boldsymbol{X}_T = \textstyle\sum_{t=0}^{T} \boldsymbol{X}_{\mathrm{s},t}; \quad \boldsymbol{O}_T = \sigma(\boldsymbol{X}_T) \\ \boldsymbol{O}_{\mathrm{s},t} = \boldsymbol{O}_t - \boldsymbol{O}_{t-1} \end{aligned} \tag{5}$$

where σ represents the function of Softmax or LayerNorm. $\boldsymbol{X}_{\mathrm{s},t}$ and $\boldsymbol{O}_{\mathrm{s},t}$ are the input and output of the operator at time-step t respectively, $\boldsymbol{X}_T$ is the summation of the input during T time-steps, which is stored in the operator. $\boldsymbol{O}_T$ is the output of the function σ with input $\boldsymbol{X}_T$. The Spike-Softmax(LayerNorm) can be made equivalent to Softmax(LayerNorm) by summing up $\boldsymbol{O}_{\mathrm{s},t}$ through time.

3.3. Complexity Analysis

The spatial and temporal complexity analysis of the operations in SpikeZIP-TF is presented in Table 3. Note that, the synaptic operations in SNN are addition or subtraction for AW multiplication and binary operation for AA multiplication, while ANN is integer or floating-point multiplication (Horowitz, 2014). Therefore, we introduce an ratio of $\gamma >> 1$ to indicate the higher operation cost of (Q)ANN *w.r.t* SNN. For spatial complexity, AW multiplication and AA multiplication do not bring additional cost, but Spike-Softmax and Spike-LayerNorm have n^2 and nd times more complexity than ANN. It is resulted from that Spike-Softmax and Spike-LayerNorm requires extra memory to store the accmulated input.

4. Experiments

4.1. Experimental Setup

Vision Benchmarks. Various vision datasets are adopted for evaluation. **1) static vision datasets**, including CIFAR10/100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). **2) neuromorphic vision dataset**: We evaluate SpikeZIP-TF on CIFAR10-DVS (Hongmin et al., 2017). CIFAR10-DVS is a neuromorphic event-stream dataset with 10 distinct classes, which is created by leveraging the dynamic vision sensor (DVS) to convert 10k frame-based images from CIFAR10 (Krizhevsky et al., 2009) dataset into 10k event streams. For ImageNet, we apply the pre-trained Vision Transformer-Small/Base/Large (*aka*. ViT-S/B/L) (Dosovitskiy et al., 2020) as the source ANN. For CIFAR-10/100 and CIFAR10-DVS, we utilize the pre-trained Vision Transformer-Small (*aka*. ViT-S) as the source ANN.

NLP Benchmarks. Various natural language understanding (NLU) datasets are evaluated, including English (MR (Pang & Lee, 2004), Subj (Pang & Lee, 2004), SST-2, SST-5 (Socher et al., 2013)) and Chinese (ChnSenti, Waimai). For NLP tasks, the Roberta-Base/Large (Liu et al., 2019) (*aka*. Roberta-B/L) is chosen as source ANN owing to its high accuracy in NLP benchmarks.

4.2. Results Comparison

Comparison on CIFAR-10/100 of SpikeZIP-TF and previous methods are elaborated in Table 4, revealing SpikeZIP-TF's superiority over prior approaches across both CIFAR-10 and CIFAR-100 datasets. Notably, with ViT-S as the backbone, SpikeZIP-TF surpasses MST (Wang et al., 2023) by 1.4% on CIFAR-10 and 2.8% on CIFAR-100 with 8× less time-steps. Compared with direct training methods, SpikeZIP-TF exhibits a 2.7% and 9.3% improvement over Spikingformer+CML on CIFAR-10 and CIFAR-100, respectively.

Comparison on CIFAR10-DVS is reported in Table 4 as well. To expedite the training convergence via leveraging the pre-trained weights, we adopt the pre-processing approach outlined in (Wang et al., 2023). This involves adding

Table 4. **Experimental results on CIFAR-10, CIFAR-100 and CIFAR10-DVS.** *CF* is the abbreviation of CIFAR. The best results are in **bold**, the runner-up results are in gray.

Methods	Category	Param(M)	*CF*10-DVS		*CF*-10		*CF*-100	
			Acc	T	Acc	T	Acc	T
ViT-S	ANN	21.70	90.4	1	99.2	1	91.9	1
QViT-S-8Level	QANN	21.70	-	-	98.0	1	87.2	1
QViT-S-16Level			88.4	1	98.7	1	89.5	1
QViT-S-32Level			90.2	1	-	-	-	-
tdBN	SNN (Direct Training)	/	67.8	10	93.2	6	-	-
ASpikformer		8.46	/	/	96.4	4	78.2	4
Spikformer)		9.32	80.9	16	95.5	4	78.2	4
SDformer		/	80.0	16	95.6	4	78.4	4
Sformer+CML		9.32	81.4	16	96.0	4	80.4	4
MST	SNN (A2S)	27.60	88.1	512	97.3	256	86.9	256
SpikeZIP-TF (ours)		21.70	87.6	32	97.7	16	87.3	16
			90.5	64	**98.7**	32	**89.7**	32

an additional reduction layer to reduce the channel dimension of neuromorphic data to 3. The experimental results in Table 4 underscore the effectiveness of SpikeZIP-TF in processing neuromorphic datasets. Compared to A2S-based MST (Wang et al., 2023), SpikeZIP-TF achieves a 2.4% higher accuracy with fewer time-steps. Despite the lower time-step requirement of the previous SOTA direct training method (Spikingformer+CML), SpikeZIP-TF delivers a remarkable 9.1% accuracy boost on CIFAR10-DVS.

Comparison on ImageNet of SpikeZIP-TF and previous methods is tabulated in Table 5. As anticipated, SpikeZIP-TF surpasses previous SOTA methods. Compared to the SOTA A2S conversion method (MST (Wang et al., 2023)), SpikeZIP-TF achieves 2.94% higher top-1 accuracy while utilizing fewer time-steps and a more lightweight model (with 6.4M parameter reduction). Although direct training methods such as Spikformer (Zhou et al., 2022), Spikingformer (Zhou et al., 2023) and Spikformer V2 (Zhou et al., 2024) require lower time-step, they demand relatively high training cost to achieve compatible performance with ANN-to-SNN conversion-based methods. In contrast, compared to previous SOTA on direct training methods (Spikformer V2 (Zhou et al., 2024)), SpikeZIP-TF incurs significantly lower computational cost while maintaining a more lightweight model and achieving SOTA top-1 accuracy. For large-scale models (ViT-L), after simply fine-tuning and quantizing the publicly available pre-trained ANN, SpikeZIP-TF yields promising performance (83.28% on ImageNet).

Comparison on NLP Benchmarks We conduct a comparative analysis of SpikeZIP-TF with other SOTA works, including SNN-TextCNN (Lv et al., 2022), SpikeGPT (Zhu et al., 2023) and SpikeBERT (Lv et al., 2023). The results are summarized in Table 6. SpikeZIP-TF outperforms SpikeGPT (Zhu et al., 2023) and SpikeBERT (Lv et al., 2023) in terms of accuracy across both the English datasets and Chinese datasets. The improvements in accuracy are particularly notable in the MR (3.65% increase) and SST-5 (5.24% increase) datasets. Moreover, SpikeZIP-TF achieves the highest accuracy despite having a greater model size (355M) compared to SpikeGPT (216 M). It is noteworthy that, as shown in Table 6, SpikeZIP-TF converted from Roberta-L exhibits lower performance compared to the SpikeZIP-TF converted from Roberta-B. This difference can be attributed to the relatively lower accuracy of the pre-trained Roberta-L model compared to the pre-trained Roberta-B model.

4.3. Training Cost Analysis

One of the key advantages of SpikeZIP-IF is its low training cost, as illustrated in Table 7. SpikeZIP-TF exhibits lower training hours and consumes less energy compared to SpikeGPT and Spikformer V2. This efficiency stems from SpikeZIP-TF's ability to skip the pre-training stage by leveraging the pre-trained ANN accessed accessed in open sources (*e.g.*, Pytorch Hub, huggingface, etc.) to initialize the quantization-aware training in QANN fine-tuning.

4.4. Power Estimation on Neuromorphic Hardware

To assess the efficiency of SpikeZIP-TF, we employ the energy model proposed by Cao et al. (2015), which has been utilized in prior works such as Wang et al. (2023); Ding et al. (2021). The model can be expressed as:

$$P = \frac{\text{\#total-spikes}}{1 \times 10^{-3}} \times \alpha \tag{6}$$

where #total-spikes is the number of spike activities occurring in SNN during one time-step which takes 1ms in Cao et al. (2015) and 1 spike activity consumes α Joules. Unit of P is Watt. According to the 45nm hardware (Horowitz, 2014), we take α as 0.9pJ. As summarized in Table 8, we use the above power model to compare SpikeZIP-TF with Spikformer (Zhou et al., 2022), Spikformer V2 (Zhou et al., 2024), Spikingformer (Zhou et al., 2023) on ImageNet, as well as SpikeBERT (Lv et al., 2023) and SpikeGPT (Zhu et al., 2023) on SST-2. During ImageNet inference, although SpikeZIP-TF has higher power consumption than Spikingformer and Spikformer V2 with similar parameters, it achieves lower power with higher accuracy, compared to Spikformer V2. This suggests that SpikeZIP-TF can achieve a better power-accuracy trade-off than Spikformer V2. For SST-2 inference, SpikeZIP-TF exhibits lower power consumption and higher accuracy compared to SpikeBERT, indicating a better power-accuracy trade-off as well.

4.5. Ablation Study

Accuracy vs. Time-Steps To achieve a better trade-off between accuracy and time-steps in SpikeZIP-TF, we conduct

Table 5. **Comparison on ImageNet.** ♢: For ViT-S, we finetune the pretrained model from AugReg (Steiner et al., 2021) with ReLU activation to achieve corresponding ANN. †: For ViT-B/L, we finetune the pretrained models from MAE (He et al., 2022) with ReLU activation to achieve corresponding ANN. ⋆: LSQ (Esser et al., 2019) quantization results of corresponding MAE pre-trained models. Our SpikeZIP-TF results are the equivalent conversion from corresponding quantization results. The prefix **Level** in architecture column means the quantization level. The best results are in **bold**, and the runner-up results are in gray.

Category	Methods	Architecture	Param(M)	Time-Step	Acc(%)
ANN	T2T-ViT	T2T-ViT-24	64.10	1	82.30
	PVT	PVT-Large	61.40	1	81.70
	Swin Transformer	SWIN-Base	88.00	1	83.50
	AugReg♢	ViT-S	22.05	1	82.34
	MAE (ReLU)†	ViT-B	86.57	1	83.75
		ViT-L	304.33	1	85.41
QANN	LSQ⋆	QViT-S-32Level	22.05	1	81.59
		QViT-B-32Level	86.57	1	82.83
		QViT-L-32Level	304.33	1	83.86
SNN (Direct Training)	TET	Spiking-ResNet-34	21.79	6	64.79
		SEW-ResNet-34	21.79	4	68.00
	STBP-tdBN	Spiking-ResNet-34	21.79	6	63.72
	SEW ResNet	SEW-ResNet-34	21.79	4	67.04
		SEW-ResNet-50	25.56	4	67.78
		SEW-ResNet-101	44.55	4	68.76
		SEW-ResNet-152	60.19	4	69.26
	Attention-SNN	ResNet-104	78.37	4	77.08
	Spike-driven Transfromer	Spiking Transformer-8-768	66.34	4	77.07
	Spikingformer	Spiking Transformer-8-768	66.34	4	75.85
	CML	Spiking Transformer-8-768	66.34	4	77.64
	Spikformer	Spikformer-6-512	23.37	4	77.26
		Spikformer-8-768	66.34	4	79.55
	Spikformer V2	Spikformer V2-8-384	29.11	4	78.80
		Spikformer V2-8-512	51.55	4	80.38
		Spikformer V2-16-768	172.70	4	82.35
SNN (A2S)	Hybrid training	ResNet-34	21.79	250	61.48
	Spiking ResNet	ResNet-34	21.79	350	71.61
		ResNet-50	25.56	350	72.75
	QCFS	VGG-16	138.42	64	72.85
	Fast-SNN	VGG-16	138.42	7	72.95
	COS	ResNet-34	21.79	8	74.17
	MST	Swin-T (BN)	28.5	512	78.51
	SpikeZIP-TF (ours)	SViT-S-32Level	22.05	64	81.45
		SViT-B-32Level	86.57	64	82.71
		SViT-L-32Level	304.33	64	**83.82**

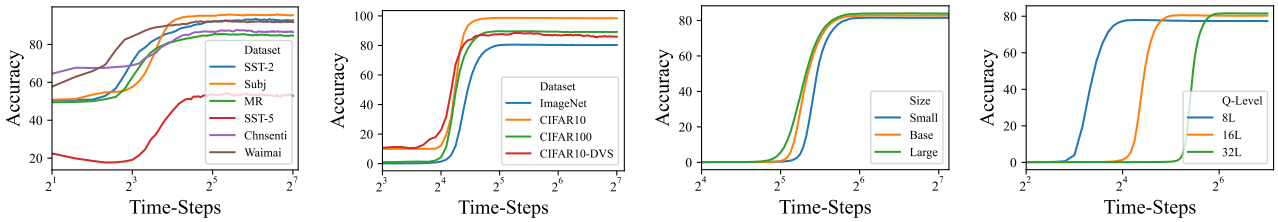
an investigation into various configurations, including the impact of different datasets, model sizes, and quantization levels, with their curves plotted in Figure 5. Overall, among all the curves in Figure 5, there exists a specific time-step called T_{up}, beyond which the model's accuracy increases drastically. This phenomenon occurs because it requires several time-steps for SpikeZIP-TF to accumulate its output. **1) Dataset:** T_{up} increases when the dataset becomes harder and more complex. As shown in Figure 5(a) and Figure 5(b), the T_{up} of SpikeZI-TF with Roberta-B on Chnsenti is much smaller than SST-5 and T_{up} of SpikeZIP-TF with ViT-B on CIFAR10-DVS is smaller than ImageNet. **2) Model Size:** Larger model size leads to better trade-off of accuracy versus time-steps. T_{up} of SpikeZIP-TF with ViT on ImageNet decreases when model size becomes large. The curve of ViT-L in Figure 5.(c) is above the curve of ViT-B and ViT-S. **3) Quantization Level:** A reduction in quantication level leads to smaller T_{up} values but also results in lower accuracy. For SpikeZIP-TF with ViT-B, T_{up} is proportional to quantization level of QViT-B, indicating that the model requires fewer time-steps to complete the inference. However, the accuracy of SpikeZIP ViT-S is lower than ViT-L due to the increase of quantization error in QAT. Therefore, choosing a suitable quantization level is crucial to strike a balance between accuracy and time-steps in A2S conversion.

4.6. Equivalence Inspection via Experiments

To further demonstrate that SNN generated by SpikeZIP is functionally equivalent to QANN and takes fewer time-steps

Table 6. **Comparison on NLU datasets.** The source ANN of SpikeZIP-TF with 125M param and 355M param are Roberta-B and Roberta-L. Cat. is short for Catogery. ‡: results taken from the SpikeBERT (Lv et al., 2023). †: Results of Roberta with ReLU activation. ⋆: LSQ quantization results of corresponding Roberta pre-trained models. The best results are in **bold**, the runner-up results are in gray.

Methods	Param (M)	Cat.	English Dataset					Avg.	Chinese Dataset			Avg.
			MR	SST-2	Subj	SST-5	T.		ChnSenti	Waimai	T.	
TextCNN	n/a	ANN	77.41	83.25	94.00	45.48	1	75.04	86.74	88.49	1	87.62
Roberta-B†	125	ANN	87.16	94.15	96.30	54.57	1	83.05	88.22	92.05	1	90.14
Roberta-L†	355	ANN	91.33	96.21	97.25	57.42	1	85.55	86.90	92.91	1	89.91
Roberta-B-32Level⋆	125	QANN	85.76	92.81	95.55	52.71	1	81.71	88.36	91.88	1	90.12
Roberta-L-64Level⋆	355	QANN	88.77	93.24	96.70	56.11	1	83.71	87.03	91.80	1	89.42
SNN-TextCNN	-	Direct Training	75.45	80.91	90.60	41.63	50	72.15	85.02	86.66	40	85.84
Spikformer‡	110	Direct Training	76.38	81.55	91.80	42.02	4	72.94	85.45	86.93	4	86.19
SpikeBERT	109	Direct Training	80.69	85.39	93.00	46.11	4	76.30	86.36	89.66	4	88.01
SpikeGPT	45	Direct Training	69.23	80.39	88.45	37.69	50	68.94	n/a	n/a	n/a	n/a
	216	Direct Training	85.63	88.76	95.30	51.27	50	80.24	n/a	n/a	n/a	n/a
SpikeZIP-TF (ours)	125	A2S	86.13	92.81	95.55	52.71	64	81.80	86.77	**91.88**	64	**89.33**
	355	A2S	**89.28**	**93.79**	**96.70**	**56.51**	128	**84.07**	**87.16**	91.29	128	89.23



(a) NLU datasets (b) CV datasets (c) model size (d) quantization level

Figure 5. **Curves of accuracy versus time-step with different settings.** (a) SpikeZIP-TF uses Roberta architecture (QANN is quantized with 32 levels); (b) SpikeZIP-TF uses ViT-small (QANN is quantized with 16 levels); (c) SpikeZIP-TF use ViT small/base/large as architecture on ImageNet; (d) Architecture is ViT-B on ImageNet, where QANNs (Figure 3) are quantized with different levels.

Table 7. **Comparison of training cost.** The SpikeZIP-TF consumes fewer training hours and less energy than SpikeGPT and Spikformer V2. **Acc.**: short for accuracy. **N.**: short for Nvidia

Method	Params	Dataset	GPU	Time(h)	Energy(kw/h)	Acc.(%)
SpikeGPT	216	SST-2	4 N	48.0	57.6	88.76
SpikeZIP-TF	355	SST-2	1 N	**1.03**	0.36	**93.79**
SpikformerV2	172.7	Image Net	8 N	196.7	472.08	81.10
SpekeZIP-TF	304.3	Image Net	8 N	**30.0**	108.00	**83.82**

Table 8. **The power consumption of SpikeZIP-TF and other works.** The time-step of SpikeGPT used in power estimation is larger than 50, therefore the power is less than 0.234.

Method	Params(M)	Dataset	Time-Steps	Power(W)	Acc(%)
SpikeBERT	109	SST-2	4	7.135	85.39
SpikeZIP-TF	355	SST-2	64	4.320	93.79
Spikformer	66.34	ImageNet	4	8.02	74.81
Spikingformer	66.34	ImageNet	4	3.42	75.85
Spikformer V2	64.18	ImageNet	4	3.67	81.17
Spikformer V2	172.70	ImageNet	4	6.39	82.35
SpikeZIP-TF	86.57	ImageNet	64	6.30	82.71
SpikeZIP-TF	304.33	ImageNet	64	19.85	83.82

for SNN inference, we visualize the evolution of feature maps *w.r.t* different time-steps T, as depicted in Figure 6. The feature maps are obtained by accumulating spiking attention array in SESA and masking the input with the accumulated spiking attention array. We can draw the following observations from Figure 6: 1) With the spiking accumulation, the feature map of SpikeZIP ViT is gradually close to the corresponding feature map in QANN and final equal to it when T_{eq}. 2) When the model size becomes larger, the feature map of SpikeZIP ViT concentrates on the target object at an earlier time-step, which is consistent with the results in Figure 5(c).

5. Conclusion

SpikeZIP-TF constructs an ANN-to-SNN conversion method that establishes the equivalence between quantized Transformer-based ANN and its SNN counterpart. To *make the equivalence framework applicable, we introduce the Spike-Equivalent Self-Attention, Spike-Softmax and Spike-LayerNorm to support the SNN-unfriendly operaters of Transformer-based ANN. Our SpikeZIP-TF leads to state-of-the-art performance on both computer vision,

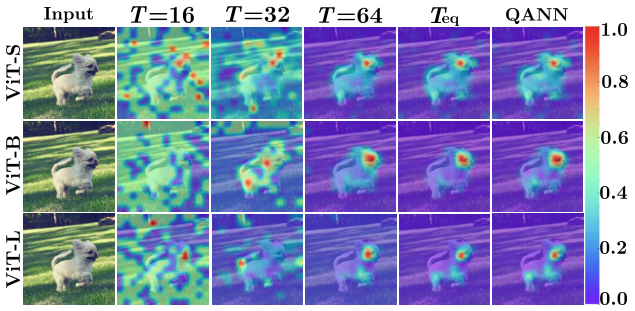


Figure 6. **Feature map visualization in SpikeZIP-TF (ViT-S/B/L) at different time-steps.** We mask the input with intermediate attention array to visualize the feature map. With the improvement of T, feature map in SpikeZIP-TF is more identical to its QANN counterpart.

neuromorphic, and natural language understanding tasks. In this work, we mainly focus on ANN-to-SNN conversion method due to its low training cost and nearly loss-less performance between SNN and ANN. We anticipate to extend our SpikeZIP-TF on direct learning methods, which is expected to reduce training cost and achieve promising performance under ultra-low inference time-step.

Acknowledgments

This work is partially supported by National Key R&D Program of China (2022YFB4500200), National Natural Science Foundation of China (Nos.62102257, 62306176), Biren Technology–Shanghai Jiao Tong University Joint Laboratory Open Research Fund, Microsoft Research Asia Gift Fund, Lingang Laboratory Open Research Fund (No.LG-QS-202202-11), Natural Science Foundation of Shanghai (No. 23ZR1428700), and CCF-Baichuan-Ebtech Foundation Model Fund.

Impact Statement

This work is a fundamental research in bridging the state-of-the-art deep neural network and spiking neural network in the community of neuromorphic community. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bal, M. and Sengupta, A. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. *arXiv preprint arXiv:2308.10873*, 2023.

Bao, H., Dong, L., Piao, S., and Wei, F. Beit: Bert pre-training of image transformers, 2022.

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. *arXiv preprint arXiv:2303.04347*, 2023.

Cao, Y., Chen, Y., and Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113: 54–66, 2015.

Che, K., Zhou, Z., Ma, Z., Fang, W., Chen, Y., Shen, S., Yuan, L., and Tian, Y. Auto-spikformer: Spikformer architecture search. *arXiv preprint arXiv:2306.00807*, 2023.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. Generative pretraining from pixels. In *International conference on machine learning*, pp. 1691–1703. PMLR, 2020.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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.

Ding, J., Yu, Z., Tian, Y., and Huang, T. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv preprint arXiv:2105.11654*, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. *CoRR*, abs/1902.08153, 2019. URL http://arxiv.org/abs/1902.08153.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Hao, Z., Ding, J., Bu, T., Huang, T., and Yu, Z. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, June 2022.

He, Z. and Fan, D. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11438–11446, 2019.

Hongmin, L., Hanchao, L., Xiangyang, J., Guoqi, L., and Luping, S. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 2017.

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., Zheng, Q., Jiang, X., and Pan, G. Fast-snn: Fast spiking neural network by converting quantized ann. *arXiv preprint arXiv:2305.19868*, 2023.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.

Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pp. 5506–5518. PMLR, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *technical report*, 2009.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lee, J. H., Delbruck, T., and Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

Li, C., Ma, L., and Furber, S. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lv, C., Xu, J., and Zheng, X. Spiking convolutional neural networks for text classification. In *The Eleventh International Conference on Learning Representations*, 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.

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

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.

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.

Pang, B. and Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *arXiv preprint cs/0409058*, 2004.

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

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Shrestha, S. B. and Orchard, G. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. How to train your vit? data, augmentation, and regularization in vision transformers. *CoRR*, abs/2106.10270, 2021. URL https://arxiv.org/abs/2106.10270.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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, Z., Fang, Y., Cao, J., Zhang, Q., Wang, Z., and Xu, R. Masked spiking transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1761–1771, 2023.

Yao, M., Hu, J., Zhou, Z., Yuan, L., Tian, Y., Xu, B., and Li, G. Spike-driven transformer. *arXiv preprint arXiv:2307.01694*, 2023.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

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., Zhu, Y., He, C., Wang, Y., Yan, S., Tian, Y., and Yuan, L. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

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.

A1. Experimental Results on Swin-Transformer

In Table A1, we conduct comprehensive experiments with Swin-Transformer Tiny network, on ImageNet, CIFAR-100, CIFAR-10 and CIFAR-10-DVS. The power consumption is calculated by Equation (6) in the manuscript, which is adopted in MST as well. Compared with the digital Transformers (QANN) counterpart, our SpikeZIP-TF achieves on-par accuracy with lower power consumption. Note that, the minor accuracy difference between QANN and SNN in SpikeZIP-TF are resulted from the GPU numeric error. For MST (the closest peer of SpikeZIP-TF), SNN suffers not only the distinguishable accuracy degradation from its QANN, but also the power reduction is lower than that of SpikeZIP-TF.

A2. Calculation of Total Spikes

The *total spikes* means the total spike activity during one time-step, whose type includes the pre-synaptic (*i.e.*, delivered by synapases) and post-synaptic (*i.e.*, generated by neurons). The number of post-synaptic spikes N_{post} is calculated by Equation (A1).

$$N_{\text{post}} = \sum_{i=0}^{l} R^i \times N_{\text{neu}}^i \tag{A1}$$

where R is the firing rate of the i-th layer and N_{neu}^i is the number of neurons of the i-th layer. N_{pre} reflects the update operation in neurons. Then for the number of pre-synaptic spikes, the calculation depends on the synaptic connection structure, which is modeled as: Equation (A2).

$$N_{\text{pre}} = \sum_{i=0}^{l} R^{i-1} \times f_{\text{in}}^i \times N_{\text{neu}}^i \tag{A2}$$

where f_{in}^i is the number of fan-in operation of the i-t layer. f_{in}^i depends on the connection struture of between neuron layers. For the convolution layer, $f_{\text{in,conv}}^i$ is calculated by:

$$f_{\text{in,conv}} = C_{\text{in}} \times K_{\text{H}} \times K_{\text{W}} \tag{A3}$$

where C_{in} is the input channel; K_{H} and K_{W} are the height and width of kernel; Therefore, for convolution layer, the pre-synaptic spikes N_{pre} is:

$$N_{\text{pre,conv}} = C_{\text{in}} \times K_{\text{H}} \times K_{\text{W}} \times C_{\text{out}} \times O_{\text{H}} \times O_{\text{W}} \tag{A4}$$

where C_{out} is the output channel; O_{H} and O_{W} are the height and width of output feature. In Equation (A4), $C_{\text{in}} \times K_{\text{H}} \times K_{\text{W}}$ is the fan-in of one neuron and $C_{\text{out}} \times O_{\text{H}} \times O_{\text{W}}$ is the number of neurons in one neuron layer. It is worth noted the relationship between $f_{\text{in,conv}}$ and the FLOPS of convolution is $2 \times f_{\text{in,conv}} = \text{FLOPS}_{\text{conv}}$. For the linear layer, $f_{\text{in,fc}}$ is the number of input neurons of N_{in}. The pre-synaptic spikes of the linear layer $N_{\text{pre,fc}}$ is:

$$N_{\text{pre,fc}} = N_{\text{in}} \times N_{\text{out}} \tag{A5}$$

where N_{in} and N_{out} are the numbers of input and output features of the linear layer. After calculating the pre-synaptic and post-synaptic spikes, the total spikes is equal to the sum of pre-synaptic and post-synaptic spikes:

$$\text{\#total-spikes} = N_{\text{pre}} + N_{\text{post}} \tag{A6}$$

A3. Proof of Equivalence:

A3.1. Preliminaries

Notations definition. For reader-friendly, we provide the notations used in the following proof in Table A2.

ST-BIF Neuron Model The definition of the ST-BIF neuron model is:

$$\begin{aligned} V_t &= V_{t-1} + V_t^{\text{in}} - V_{\text{thr}} \cdot \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \\ S_t &= S_{t-1} + \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \\ \Theta(V, V_{\text{thr}}, S) &= \begin{cases} 1; & V \geq V_{\text{thr}} \ \& \ S < S_{\text{max}} \\ 0; & \text{other} \\ -1; & V < 0 \ \& \ S > S_{\text{min}} \end{cases} \end{aligned} \tag{A7}$$

where the first equation of Equation (A7) depicts the membrane potential updating in ST-BIF neuron. The membrane potential at time-step t equals to the membrane potential at the prior time-step $t-1$ adding the potential V_t^{in} caused by the input charge at t time-step, then subtract the potential of the fired spike. The fired spike is recorded by the spike tracer defined in the second equation of Equation (A7). The firing behavior of ST-BIF neuron depends on the spike decision function Θ in the third equation.

Equilibrium State Assume the external stimulate (*e.g.*, input and bias) are applied to SNN from $T = 0$ to T_{off}, we define the **equilibrium state** of SNN as the status where neurons of entire SNN are static (*e.g.*, no further activities of neuron firing and membrane update). The time-step that SNN enters the equilibrium state is noted as T_{eq}.

A3.2. The Equivalence between Quantized Function and ST-BIF$^+$ Neuron

Lemma A3.1. *After entering the equilibrium state at T_{eq}, the accumulated output spikes of one ST-BIF neuron can be derived as a closed-form equation of quantization function:*

$$V^{\text{out}} = V_{\text{thr}} \cdot \text{clip}(\text{floor}(\frac{V^{\text{in}} + V_{t=0}}{V_{\text{thr}}}), S_{\text{min}}, S_{\text{max}}) \tag{A8}$$

where $V^{in} = \sum_{t=0}^{T_{\text{eq}}} V_t^{in}$ is the accumulated input until T_{eq}, and $V_{t=0}$ denotes the initial membrane potential.

Table A1. Experiments with Swin-Transformer Tiny on A2S methods and corresponding digital transformers.

Method	Category	ImageNet		Cifar-100		Cifar-10		Cifar-10-DVS	
		Acc/#T	Power(W)	Acc/#T	Power(W)	Acc/#T	Power(W)	Acc/#T	Power(W)
MST	QANN	80.51/1	20.810	88.72/1	20.730	98.14/1	20.730	88.98/1	20.940
MST	SNN	78.51/512	8.528	86.91/256	8.286	97.27/256	8.304	88.12/512	8.188
SpikeZIP-TF(ours)	QANN	80.70/1	20.170	87.94/1	20.170	98.38/1	20.170	90.50/1	20.280
SpikeZIP-TF(ours)	SNN	**80.74/64**	**1.363**	**87.91/32**	**1.428**	**98.45/32**	**1.422**	**90.40/64**	**1.317**

Notation	Description
V_t	potential of neuron membrane at time-step t
V_{thr}	threshold voltage for neuron to fire a spike
$V^{\text{in}}, V^{\text{out}}$	input or output voltage of neuron
T_{eq}	time-step that neuron enters equilibrium state
T_{off}	time-steps when input and bias are turned off
S_t	spike tracer at time-step t
$S_{\text{max}}/S_{\text{min}}$	maximum/minimum value in spike tracer
$\text{clip}(x, \alpha_{\text{min}}, \alpha_{\text{max}})$	clip function that limits x between α_{min} and α_{max}
$\text{floor}(x)$	floor function that round down x
$\Theta(V, V_{\text{thr}}, S)$	output spike decision function of ST-BIF neuron
$\boldsymbol{Q}_t, \boldsymbol{K}_t$	spiking Query and spiking Key matrix in SESA.
$\boldsymbol{S}_{Q,t}, \boldsymbol{S}_{K,t}$	accumulated spike trains of query and key.

Table A2. Summary of mathematical notations used in the proof.

Proof. Starting from the first equation in Equation (A7), the membrane potential can be calculated without using the recursive form by summing over simulated time T:

$$V_T - V_0 = \sum_{t=1}^{T} V_t^{\text{in}} - V_{\text{thr}} \cdot \sum_{t=1}^{T} \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \tag{A9}$$

We sum the spike tracer S_t in Equation (A7) over the inference time-steps T, which is described as:

$$S_T - S_0 = \sum_{t=1}^{T} \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \tag{A10}$$

where $S_0 = 0$ is the default setting. By substituting Equation (A10) into Equation (A9), Equation (A9) is simplified as:

$$V_T = (\sum_{t=1}^{T} V_t^{\text{in}} + V_0) - V_{\text{thr}} \cdot S_T \tag{A11}$$

Then, we divide both sides of Equation (A11) by V_{thr}. With additional simple transformation, we get:

$$S_T = \frac{(\sum_{t=1}^{T} V_t^{\text{in}} + V_0 - V_T)}{V_{\text{thr}}} \tag{A12}$$

Hereby, we discuss three cases about S_T in Equation (A12) as follows.

Case 1. $S_{\text{min}} \leq (\sum_{t=1}^{T} V_t^{\text{in}} + V_0)/V_{\text{thr}} \leq S_{\text{max}}$: When $T \geq T_{\text{eq}}$, according to the definition of equilibrium state, the membrane potential of the ST-BIF neuron is insufficient to fire a spike, which means $V_T < V_{\text{thr}}$. Since S_T is an integer in Equation (A12), based on the definition of round down function (e.g., floor), S_T can be rewritten as:

$$S_T = \text{floor}(\frac{\sum_{t=1}^{T} V_t^{\text{in}} + V_0}{V_{\text{thr}}}) \tag{A13}$$

where the error caused by the rounding (down) of the membrane potential in Equation (A13) is equal to V_T. Equation (A13) represents the discretization part in the quantized-ReLU (Q-ReLU) function.

Case 2. $(\sum_{t=1}^{T} V_t^{\text{in}} + V_0)/V_{\text{thr}} > S_{\text{max}}$: According to the firing decision function Θ, the ST-BIF neuron fires positive spikes until spike tracer $S_T = S_{\text{max}}$, then the S_T becomes static:

$$S_T = S_{\text{max}} \tag{A14}$$

In virtue of setting the upper bound of S_T, we successfully limit the accumulated output in the ST-BIF neuron to S_{max}, which corresponds to the clipping upper bound in Q-ReLU.

Case 3. $(\sum_{t=1}^{T} V_t^{\text{in}} + V_0)/V_{\text{thr}} < S_{\text{min}}$: Similar to Case 2, before the T_{eq}, the ST-BIF neuron fires negative spikes until spike tracer $S_T = S_{\text{min}}$, then S_T is fixed.

$$S_T = S_{\text{min}} \tag{A15}$$

Then, we leverage the clip function to combine Equation (A13), Equation (A14) and Equation (A15), then we can derive:

$$S_T = \text{clip}(\text{floor}(\frac{\sum_{t=1}^{T} V_t^{\text{in}} + V_0}{V_{\text{thr}}}), S_{\text{min}}, S_{\text{max}}) \tag{A16}$$

The total output of an ST-BIF neuron can be defined as:

$$V^{\text{out}} = \sum_{t=1}^{T} V_t^{\text{out}} = V_{\text{thr}} \cdot \sum_{t=1}^{T} \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1}) \tag{A17}$$

where $\sum_{t=1}^{T} \Theta(V_{t-1} + V_t^{\text{in}}, V_{\text{thr}}, S_{t-1})$ denotes the number of total output spikes of an ST-BIF neuron. Equation (A17) shows the accumulated output of the ST-BIF neuron is equal to the number of total output spikes scaled by the firing threshold V_{thr}. We substitute Equation (A10) in Equation (A17) and get:

$$V^{\text{out}} = V_{\text{thr}} \cdot S_T \tag{A18}$$

Then, we further substitute Equation (A16) in Equation (A18):

$$V_{\text{out}} = V_{\text{thr}} \cdot \text{clip}(\text{floor}(\frac{\sum_{t=1}^{T} V_t^{\text{in}} + V_0}{V_{\text{thr}}}), 0, S_{\max}) \quad \text{(A19)}$$

Proof complete.

□

A3.3. The Equivalence of Spike-Equivalent Self-Attention (SESA)

Dynamic Model of SESA. Before we prove the equivalence between the SESA and quantized self-attention shown in Figure 3, we introduce the dynamic model of SESA during the SNN inference:

$$\begin{aligned} \boldsymbol{S}_{\text{Q},t} &= \boldsymbol{S}_{\text{Q},t-1} + \boldsymbol{Q}_t, \quad \boldsymbol{S}_{\text{K},t} = \boldsymbol{S}_{\text{K},t-1} + \boldsymbol{K}_t \\ \boldsymbol{O}_t &= \boldsymbol{S}_{\text{Q},t} \cdot \boldsymbol{K}_t^T + \boldsymbol{Q}_t \cdot \boldsymbol{S}_{\text{K},t}^T - \boldsymbol{Q}_t \cdot \boldsymbol{K}_t^T \end{aligned} \quad \text{(A20)}$$

where O_t is the output of SESA at t time-step, other notations are summarized in Table A2. As shown in Figure 4, we calculate the output of SESA at t time-step (the green dotted frame) by doing three matrix multiplication.

Lemma A3.2. *After entering the equilibrium state $T \geq T_{\text{eq}}$, the accumulated output of Spiking-Equivalent Self-Attention (SESA) equals the output of Quantized Self-Attention (QSA) with same input:*

$$\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{O}_t = \boldsymbol{O}_q \quad \text{(A21)}$$

Where $\boldsymbol{O}_t$ and $\boldsymbol{O}_q$ are the accumulated output of SESA and the output of QSA.

Proof. Firstly, according to the formula of QSA n Equation (4), the output of QSA $\boldsymbol{O}_q$ can be written:

$$\text{LHS} = \boldsymbol{O}_q = \boldsymbol{Q}_{\text{q}} \cdot \boldsymbol{K}_{\text{q}} \quad \text{(A22)}$$

where $\boldsymbol{Q}_{\text{q}}$, $\boldsymbol{K}_{\text{q}}$ are the query and key matrices in QSA, which are also the outputs of the quantized functions. In SESA, these quantized functions are replaced by the ST-BIF$^+$ neurons. By leveraging the Lemma A3.1, we build the relation between $\boldsymbol{Q}_{\text{q}}$, $\boldsymbol{K}_{\text{q}}$ and the spiking query $\boldsymbol{Q}_{\text{t}}$ and spiking key $\boldsymbol{K}_{\text{t}}$:

$$\text{LHS} = \boldsymbol{Q}_{\text{q}} \cdot \boldsymbol{K}_{\text{q}} = \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{Q}_t \cdot \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{K}_t \quad \text{(A23)}$$

where $\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{Q}_t \cdot \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{K}_t$ can be considered summing every element of a 2-D matrix up, whose element $A_{i,j} = \boldsymbol{Q}_i \cdot \boldsymbol{K}_j$. The procedure is also shown in the left brown matrix in Figure A1. Then, as illustrated in Figure A1, we decompose the sum operation into three parts: upper triangular, lower triangular, and diagonal. Therefore, we can rewrite the $\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{Q}_t \cdot \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{K}_t$ as:

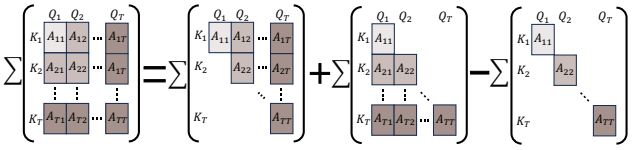


Figure A1. The Decomposition of Summing Up.

$$\begin{aligned} \text{LHS} &= \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{Q}_t \cdot \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{K}_t = \\ &\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{Q}_t \sum_{t_1=0}^{t} \boldsymbol{K}_{t_1}^T + \sum_{t=0}^{T_{\text{eq}}} \sum_{t_1=0}^{t} \boldsymbol{Q}_{t_1} \boldsymbol{K}_t^T - \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{Q}_t \boldsymbol{K}_t^T = \\ &\sum_{t=0}^{T_{\text{eq}}} (\boldsymbol{Q}_t \sum_{t_1=0}^{t} \boldsymbol{K}_{t_1}^T + \sum_{t_1=0}^{t} \boldsymbol{Q}_{t_1} \boldsymbol{K}_t^T - \boldsymbol{Q}_t \boldsymbol{K}_t^T) \end{aligned} \quad \text{(A24)}$$

where the $\sum_{t_1=0}^{t} \boldsymbol{Q}_{t_1}$ and $\sum_{t_1=0}^{t} \boldsymbol{K}_{t_1}^T$ are the accumulated spike trains of query and key at t time-step, which equal the $\boldsymbol{S}_{\text{Q},t}$ and $\boldsymbol{S}_{\text{K},t}$:

$$\begin{aligned} \text{LHS} &= \sum_{t=0}^{T_{\text{eq}}} (\boldsymbol{Q}_t \boldsymbol{S}_{\text{K},t}^T + \boldsymbol{S}_{\text{Q},t} \boldsymbol{K}_t^T - \boldsymbol{Q}_t \boldsymbol{K}_t^T) = \\ &\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{O}_t = \text{RHS} \end{aligned} \quad \text{(A25)}$$

Equation (A25) prove the output of QSA (LHS) equals the accumulated output of SESA (RHS). Proof complete. □

A3.4. The Equivalence of Spike-Softmax and Spike-LayerNorm.

Dynamic Model. The dynamic model of Spike-Softmax and Spike-LayerNorm at each time-step are inspired by the differential algorithm, which can be written as:

$$\begin{aligned} X_t &= X_{t-1} + x_t; \quad O_t = \sigma(X_t) \\ o_t &= O_t - O_{t-1} \end{aligned} \quad \text{(A26)}$$

where X_t is the accmulated input during t time-step, x_t is the input at t time-step, σ is the Softmax/LayerNorm function, o_t is the output of Spike-Softmax and Spike-LayerNorm at t time-step. Equation (A26) decompose the activation in Softmax and LayerNorm into multiple time-steps without changing the summation.

Lemma A3.3. *If the accumulated input in SNN equals to the input in QANN, the accumulated output of Spike-Softmax/Spike-LayerNorm equals the output of Soft-*

Table A3. The hyperparameter of end-to-end finetuning with ViT-S ReLU on ImageNet. ∗: (Loshchilov & Hutter, 2017). ⋆: (Chen et al., 2020). †: (Clark et al., 2020; Bao et al., 2022). ‡: (Goyal et al., 2017). ▽ : (Loshchilov & Hutter, 2016). △: (Cubuk et al., 2020). ▲: (Szegedy et al., 2016). ▼: (Zhang et al., 2017). ♢: (Yun et al., 2019). ♦: (Huang et al., 2016).

config	value (ViT-S ReLU))
optimizer	AdamW ∗
optimizer momentum	β_1, β_2 = 0.9, 0.99 ⋆
base learning rate	1e-4
weight decay	0.05
layer-wise lr decay †	0.65
GPUs	4
batch_size	64/256
warmup epochs ‡	5
training epochs	100
learning rate schedule	cosine decay ▽
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5) △
label smoothing ▲	0.1
mixup ▼	0.8
cutmix ♢	1.0
drop path ♦	0.1

max/LayerNorm after entering equilibruim state:

$$\sum_{t=0}^{T_{\text{eq}}} o_t = o_q; \quad \text{s.t.} \sum_{t=0}^{T_{\text{eq}}} x_t = x_q \tag{A27}$$

where x_q and o_q the input and output of Softmax (LayerNorm).

Proof. In Equation (A26), summing up the X_t through time, we have $X_{T_{\text{eq}}} = \sum_{t=0}^{T_{\text{eq}}} x_t = x_q$. Similarly, summing up the o_t over time, we also have $\sum_{t=0}^{T_{\text{eq}}} o_t = O_{T_{\text{eq}}}$. Combine the two condition, we have:

$$\begin{aligned} &\text{LHS} = o_q = \sigma(x_q) = \sigma(\sum\nolimits_{t=0}^{T_{\text{eq}}} x_t) = \\ &\sigma(X_{T_{\text{eq}}}) = O_{T_{\text{eq}}} = \sum\nolimits_{t=0}^{T_{\text{eq}}} o_t = \text{RHS} \end{aligned} \tag{A28}$$

Proof complete. □

A4. Implementation Details

A4.1. Implementation Details on ImageNet

The hyperparameter of our end-to-end finetuning with ViT-S ReLU, ViT-B ReLU and ViT-L ReLU on ImageNet are tabulated in Table A3, Table A4 and Table A5. Then we follow the hyperparameter in Table A6, Table A7 and Table A8 to conduct 32-Level Quantization-Aware-Training (QAT) on ViT-S ReLU, ViT-B ReLU and ViT-L ReLU obtained above, and achieve QViT-S-32Level, QViT-B-32Level and QViT-L-32Level in Table 5. Finally we apply our SpikeZIP-TF on QViT-S-32Level, QViT-B-32Level and QViT-L-32Level to achieve corresponding SViT-S-32Level, SViT-B-32Level and SViT-L-32Level in Table 5.

Table A4. The hyperparameter of end-to-end finetuning with ViT-B ReLU on ImageNet.

config	value (ViT-B ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1.66e-4
weight decay	0.05
layer-wise lr decay	0.65
GPUs	8
batch_size	96/768
warmup epochs	5
training epochs	100
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.1

A4.2. Implementation Details on CIFAR10/100

The hyperparameter of our end-to-end finetuning with ViT-S ReLU on CIFAR10/100 are tabulated in Table A9. Then we follow the hyperparameter in Table A10 to conduct 8-Level and 16-Level Quantization-Aware-Training (QAT) on ANN ViT-S ReLU obtained above, and achieve QViT-S-8Level and QViT-S-16Level in Table 4. Finally we apply our SpikeZIP-TF on QViT-S-8Level and QViT-S-16Level to achieve corresponding 16 time-steps SViT-S and 32 time-steps SViT-S in Table 4.

A4.3. Implementation Details on CIFAR10-DVS

The hyperparameter of our end-to-end finetuning with ViT-S ReLU on CIFAR10-DVS are tabulated in Table A11. Then we follow the hyperparameter in Table A12 to conduct 16-Level and 32-Level Quantization-Aware-Training (QAT) on ANN ViT-S ReLU obtained above, and achieve QViT-S-16Level and QViT-S-32Level in Table 4. Finally we apply our SpikeZIP-TF on QViT-S-16Level and QViT-S-32Level to achieve corresponding 32 time-steps SViT-S and 64 time-steps SViT-S in Table 4.

Table A5. The hyperparameter of end-to-end finetuning with ViT-L ReLU on ImageNet.

config	value (ViT-L ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1.67e-3
weight decay	0.05
layer-wise lr decay	0.75
GPUs	8
batch_size	24/192
warmup epochs	5
training epochs	50
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.2

Table A6. The hyperparameter of Quantization-Aware-Training with ViT-S ReLU on ImageNet.

config	value (ViT-S ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1.5e-4
weight decay	0.05
layer-wise lr decay	0.65
GPUs	4
batch_size	64/256
warmup epochs	5
training epochs	100
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.1
quantization level	32

Table A7. The hyperparameter of Quantization-Aware-Training with ViT-B ReLU on ImageNet.

config	value (ViT-B ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1.5e-4
weight decay	0.001
layer-wise lr decay	0.65
GPUs	6
batch_size	96/576
warmup epochs	5
training epochs	50
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.05
quantization level	32

Table A8. The hyperparameter of Quantization-Aware-Training with ViT-L ReLU on ImageNet.

config	value (ViT-L ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1.6e-3
weight decay	0.0005
layer-wise lr decay	0.75
GPUs	8
batch_size	20/160
warmup epochs	5
training epochs	50
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.05
quantization level	32

Table A9. The hyperparameter of end-to-end finetuning with ViT-S ReLU on CIFAR10/100.

config	value (ViT-S ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1e-4
weight decay	0.05
layer-wise lr decay	0.65
GPUs	8
batch_size	192/1536
warmup epochs	5
training epochs	100
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.1

Table A10. The hyperparameter of Quantization-Aware-Training with ViT-S ReLU on CIFAR10/100.

config	value (ViT-S ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	1.5e-4
weight decay	0.05
layer-wise lr decay	0.65
GPUs	8
batch_size	128/1024
warmup epochs	5
training epochs	300
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.1
quantization level	8, 16

Table A11. The hyperparameter of end-to-end finetuning with ViT-S ReLU on CIFAR10-DVS.

config	value (ViT-S ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	2e-4
weight decay	0.05
layer-wise lr decay	0.65
GPUs	4
batch_size	192/768
warmup epochs	5
training epochs	300
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.1

Table A12. The hyperparameter of Quantization-Aware-Training with ViT-S ReLU on CIFAR10-DVS.

config	value (ViT-S ReLU))
optimizer	AdamW
optimizer momentum	β_1, β_2 = 0.9, 0.99
base learning rate	2.25e-4
weight decay	0.05
layer-wise lr decay	0.65
GPUs	4
batch_size	92/368
warmup epochs	5
training epochs	300
learning rate schedule	cosine decay
distillation weight	1.0
distillation temp.	2.0
augmentation	RandomAug (9, 0.5)
label smoothing	0.1
mixup	0.8
cutmix	1.0
drop path	0.1
quantization level	16, 32