

# SpikingResformer: Bridging ResNet and Vision Transformer in Spiking Neural Networks

Xinyu Shi<sup>1,2</sup>, Zecheng Hao<sup>2</sup>, Zhaofei Yu<sup>1,2\*</sup>

<sup>1</sup> Institute for Artificial Intelligence, Peking University

<sup>2</sup> School of Computer Science, Peking University

## Abstract

The remarkable success of Vision Transformers in Artificial Neural Networks (ANNs) has led to a growing interest in incorporating the self-attention mechanism and transformer-based architecture into Spiking Neural Networks (SNNs). While existing methods propose spiking self-attention mechanisms that are compatible with SNNs, they lack reasonable scaling methods, and the overall architectures proposed by these methods suffer from a bottleneck in effectively extracting local features. To address these challenges, we propose a novel spiking self-attention mechanism named Dual Spike Self-Attention (DSSA) with a reasonable scaling method. Based on DSSA, we propose a novel spiking Vision Transformer architecture called SpikingResformer, which combines the ResNet-based multi-stage architecture with our proposed DSSA to improve both performance and energy efficiency while reducing parameters. Experimental results show that SpikingResformer achieves higher accuracy with fewer parameters and lower energy consumption than other spiking Vision Transformer counterparts. Notably, our SpikingResformer-L achieves 79.40% top-1 accuracy on ImageNet with 4 time-steps, which is the state-of-the-art result in the SNN field. Codes are available at <https://github.com/xyshi2000/SpikingResformer>

## 1. Introduction

Spiking Neural Networks (SNNs), considered as the third generation of Artificial Neural Networks (ANNs) [28], have garnered significant attention due to their notable advantages such as low power consumption, biological plausibility, and event-driven characteristics that are compatible with neuromorphic hardware. In comparison to ANNs, SNNs exhibit an energy-saving advantage when deployed on neuromorphic hardware [11, 30, 33], and have become popular in the field of neuromorphic computing in recent years [34].

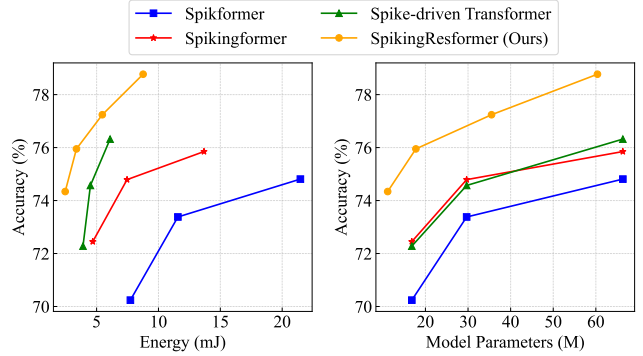


Figure 1. Comparison of Top-1 accuracy on ImageNet with respect to energy consumption per image for inference (left) and the number of parameters (right). The input size is  $224 \times 224$ .

However, the performance of existing SNNs still lags behind that of ANNs, particularly on challenging vision tasks, which limits further application of SNNs.

Researchers make great efforts to enhance the performance of SNNs. Recently, inspired by the remarkable achievements of vision transformers [8, 27] in ANNs, some attempts have been made to introduce transformer-based architecture into SNNs. The main challenge in incorporating transformer structures into SNNs lies in designing a self-attention mechanism suitable for SNNs. The vanilla self-attention mechanism in ANNs relies on float-point matrix multiplication and softmax operations. However, these operations involve float-point multiplication, division, and exponentiation, which do not comply with the spike-driven nature of SNNs. Moreover, commonly used components in ANN Transformers, such as layer normalization (LN) and GELU activation, are not directly applicable in SNNs. Therefore, the introduction of transformer architectures into SNNs necessitates circumventing these operations and catering to the unique requirements of SNNs.

Some work has attempted to partially preserve floating-point operations [38], or run an SNN and an ANN Transformer in parallel [49]. Although these approaches en-

\*Corresponding author: yuzf12@pku.edu.cn

hance performance, they do not fully address the incompatibility between vanilla self-attention and SNNs. There are also works that propose spiking self-attention mechanisms and spiking Vision Transformer architectures, which are entirely based on synaptic operations [46] and fully spike-driven [42, 45]. These works completely resolve the incompatibility of the self-attention mechanism with SNNs, significantly outperforming existing spiking convolutional neural networks. However, these approaches have certain limitations. Specifically, the spiking transformer architectures of these methods exhibit a bottleneck in extracting local features. They employ a shallow convolutional network before the Transformer encoder to extract local features and reduce the size of feature maps. However, the effectiveness of such a shallow network is limited compared to Transformer encoders. Replacing this shallow network with a Transformer encoder is not feasible. This is because their spiking self-attention mechanisms lack reasonable scaling methods, making them suitable only for small feature maps. Designing proper scaling factors for these mechanisms is challenging as the input currents to neurons, which generate self-attention, do not possess simple mean and variance forms. Thus, there is a pressing need to design a new spiking self-attention mechanism with a reasonable scaling method that can effectively handle large feature maps.

To address these problems, we propose a novel spiking self-attention mechanism, named Dual Spike Self-Attention (DSSA). It produces spiking self-attention via Dual Spike Transformation, which is fully spike-driven, compatible with SNNs, and eliminates the need for direct spike multiplications. In addition, we detail the scaling method in DSSA, enabling it to adapt to feature maps of arbitrary scales. Building upon DSSA, we propose SpikingResformer, a novel spiking Vision Transformer architecture. This architecture combines the ResNet-based multi-stage design with our proposed spiking self-attention mechanism. Experimental results show that our proposed SpikingResformer significantly outperforms the performance of existing spiking Vision Transformers with fewer parameters and less energy consumption. The main contributions of this paper can be summarized as follows:

- We propose the Dual Spike Self-Attention (DSSA), a novel spiking self-attention mechanism. It produces spiking self-attention via Dual Spike Transformation, which is fully spike-driven and compatible with SNNs.
- We detail the scaling factors employed in DSSA, enabling DSSA to handle feature maps of arbitrary scales.
- We propose the SpikingResformer architecture, which combines the ResNet-based multi-stage architecture with our proposed DSSA.
- Experimental results show that the proposed SpikingResformer significantly outperforms other spiking Vision Transformer counterparts with fewer param-

eters and lower energy consumption. Notably, our SpikingResformer-L achieves up to 79.40% top-1 accuracy on ImageNet.

## 2. Related Work

**Spiking Convolutional Neural Networks.** Spiking convolutional neural networks (SCNNs) have been extensively developed due to the remarkable success of surrogate gradient learning [31, 43] and are widely used in handling challenging vision tasks, including object recognition [1, 20], detection [17, 36], and segmentation [18, 32]. To improve the performance of SCNNs on these challenging tasks, researchers have dedicated great efforts to exploring training methods [6, 12, 21, 29, 39, 41, 47, 48] and ANN-to-SNN conversion techniques [2, 3, 5, 7, 13–16, 22, 24, 35, 40]. Moreover, many deep spiking convolutional architectures [9, 14, 21, 35, 44] have been proposed to achieve high performance. The success of SpikingVGG [21, 35] demonstrates that SCNNs can achieve comparable performance to ANNs in recognition tasks. SpikingResNet [14, 44] further explores SCNNs with residual structure and achieves deeper SCNN with ResNet-based architecture. Moreover, SEW ResNet [9] meticulously analyzes the identity mapping in directly-trained spiking residual networks and successfully trains a 152-layer SCNN directly. These architectures leverage large-scale SNNs with numerous layers and demonstrate superior performance on various tasks.

**Spiking Vision Transformers.** Spikformer [46] is the first directly-trained spiking vision transformer with a pure SNN architecture. It introduces a spiking self-attention mechanism that eliminates multiplication by activating Query, Key, and Value with spiking neurons and replacing softmax with spiking neurons. In addition, it replaces layer normalization and GELU activation in the Transformer with batch normalization and spiking neurons. Based on Spikformer, Spikingformer [45] achieves a purely spike-driven Vision Transformer by modifying the residual connection paradigm. Spike-driven Transformer [42] proposes a spike-driven self-attention mechanism with linear complexity, effectively reducing energy consumption. However, all of these efforts employ a shallow convolutional network to pre-extract local information to form a sequence of patches and lack proper scaling methods.

## 3. Preliminary

We describe the dynamics of the Leaky Integrate-and-Fire (LIF) neuron used in this paper by the following discrete-time model:

$$v_i[t] = u_i[t] + \frac{1}{\tau}(I_i[t] - (u_i[t] - u_{\text{rest}})), \quad (1)$$

$$s_i[t] = H(v_i[t] - u_{\text{th}}), \quad (2)$$

$$u_i[t+1] = s_i[t]u_{\text{rest}} + (1 - s_i[t])v_i[t]. \quad (3)$$

Eq. (1) describes the charging process. Here  $u_i[t]$  and  $v_i[t]$  denote the membrane potential of  $i$ -th postsynaptic neuron before and after charging.  $\tau$  is the membrane time constant.  $I_i[t]$  denotes the input current. In general,  $I_i[t] = \sum_j w_{i,j} s_j[t]$ , where  $s_j[t] \in \{0, 1\}$  represents the spike output of the  $j$ -th presynaptic neuron at time-step  $t$ , and  $w_{i,j}$  represents the weight of the corresponding synaptic connection from neuron  $j$  to neuron  $i$ . Eq. (2) describes the firing process, where  $H(\cdot)$  is the Heaviside function,  $s_i[t] \in \{0, 1\}$  is the spike output of the spiking neuron,  $u_{th}$  denotes the firing threshold. Eq. (3) describes the resetting process, with  $u_{rest}$  denoting the resting potential.

For the sake of simplicity and clarity in subsequent sections, we represent the spiking neuron model as follows:

$$\mathbf{S} = \text{SN}(\mathbf{I}), \quad (4)$$

where  $\text{SN}(\cdot)$  denotes the spiking neuron layer, omitting the dynamic processes within the neuron,  $\mathbf{I} \in \mathbb{R}^{T \times n}$  is the input current, where  $T$  is the time step and  $n$  is the number of neurons,  $\mathbf{S} \in \{0, 1\}^{T \times n}$  is the corresponding spike output.

## 4. Dual Spike Self-Attention

This section first revisits the vanilla self-attention (VSA) mechanism commonly used in ANNs and analyzes why VSA is not suitable for SNNs. Then, we propose dual spike self-attention (DSSA), specifically designed for compatibility. We further discuss the significance of the scaling factor in DSSA and the spike-driven characteristic of DSSA.

### 4.1. Vanilla Self-Attention

The vanilla self-attention in Transformer [37] can be formulated as follows:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{V} = \mathbf{X}\mathbf{W}_V, \quad (5)$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right) \mathbf{V} \quad (6)$$

Here  $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$  denote Query, Key and Value, respectively.  $n$  is the number of patches,  $d$  is the embedding dimension. We assume that  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$  have the same embedding dimension.  $\mathbf{X} \in \mathbb{R}^{n \times d}$  is the input of self-attention block,  $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$  are the weights of the linear transformations corresponding to  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ , respectively.

The vanilla self-attention commonly used in ANNs is not suitable for SNNs due to the following two types of operations involved: 1) the float-point matrix multiplication of  $\mathbf{Q}$  and  $\mathbf{K}$ , as well as between the attention map and  $\mathbf{V}$ ; 2) the softmax function, which contains exponentiation and division operations. These operations rely on float-point multiplication, division, and exponentiation operations, which are not compatible with the restrictions of SNNs.

### 4.2. Dual Spike Self-Attention

To introduce the self-attention mechanism into SNNs and efficiently handle the multi-scale feature maps, we propose a novel spiking self-attention mechanism, named Dual Spike Self-Attention (DSSA). DSSA only utilizes Dual Spike Transformation (DST), thereby eliminating the need for the float-point matrix multiplication and softmax function. We first define the DST as follows:

$$\text{DST}(\mathbf{X}, \mathbf{Y}; f(\cdot)) = \mathbf{X}f(\mathbf{Y}) = \mathbf{X}\mathbf{Y}\mathbf{W}, \quad (7)$$

$$\text{DST}_T(\mathbf{X}, \mathbf{Y}; f(\cdot)) = \mathbf{X}f(\mathbf{Y})^T = \mathbf{X}\mathbf{W}^T\mathbf{Y}^T. \quad (8)$$

In Eq. (7),  $\mathbf{X} \in \{0, 1\}^{T \times p \times m}$  and  $\mathbf{Y} \in \{0, 1\}^{T \times m \times q}$  represent the dual spike inputs.  $T$  is the time steps,  $p, m, q$  denote arbitrary dimensions. Here  $f(\cdot)$  is a generalized linear transformation on  $\mathbf{Y}$ , with  $\mathbf{W} \in \mathbb{R}^{q \times q}$  denoting its weight matrix. It represents any operation that can be equated to a linear transformation, including convolution, batch normalization (ignoring bias), etc. A detailed discussion on the equivalence of convolution operations to linear transformations can be found in the supplementary. Similarly, in Eq. (8), we have  $\mathbf{Y} \in \{0, 1\}^{T \times q \times m}$ ,  $\mathbf{W} \in \mathbb{R}^{m \times m}$ . Since  $\mathbf{X}$  and  $\mathbf{Y}$  are both spike matrices, all matrix multiplications are equivalent to the summation of weights. Consequently, the DST avoids floating-point multiplication, making it compatible with SNNs. Further discussion of the compatibility and the spike-driven characteristic of DST can be found in Sec. 4.4. Based on DST, the attention map in DSSA can be formulated as follows:

$$\text{AttnMap}(\mathbf{X}) = \text{SN}(\text{DST}_T(\mathbf{X}, \mathbf{X}; f(\cdot)) * c_1), \quad (9)$$

$$f(\mathbf{X}) = \text{BN}(\text{Conv}_p(\mathbf{X})), \quad (10)$$

where  $\mathbf{X} \in \{0, 1\}^{T \times HW \times d}$  is the spike input, with  $H$  and  $W$  denoting the spatial height and width of the input,  $d$  denoting the embedding dimension.  $\text{BN}(\cdot)$  refers to the batch normalization layer,  $\text{Conv}_p(\cdot)$  denotes a  $p \times p$  convolution with a stride of  $p$ , and  $c_1$  is the scaling factor. Since the convolution operation is equivalent to a generalized linear transformation, and batch normalization can be absorbed into the convolution (ignoring bias),  $\text{BN}(\text{Conv}_p(\cdot))$  can be viewed as a generalized linear transformation. Here we use the  $p \times p$  convolution with a stride of  $p$  to reduce the spatial size to handle the multi-scale feature map and reduce the computational overhead. In DSSA, there is no need for the softmax function since the spiking neuron layer inherently generates a binary attention map composed of spikes. Each spike  $s_{ij}$  in this spiking attention map signifies attention between patch  $i$  and patch  $j$ . We believe that such a spiking attention map is more interpretable compared to the attention map in ANN activated with softmax. With the spiking attention map, the DSSA can be formulated as follows:

$$\text{DSSA}(\mathbf{X}) = \text{SN}(\text{DST}(\text{AttnMap}(\mathbf{X}), \mathbf{X}; f(\cdot)) * c_2), \quad (11)$$

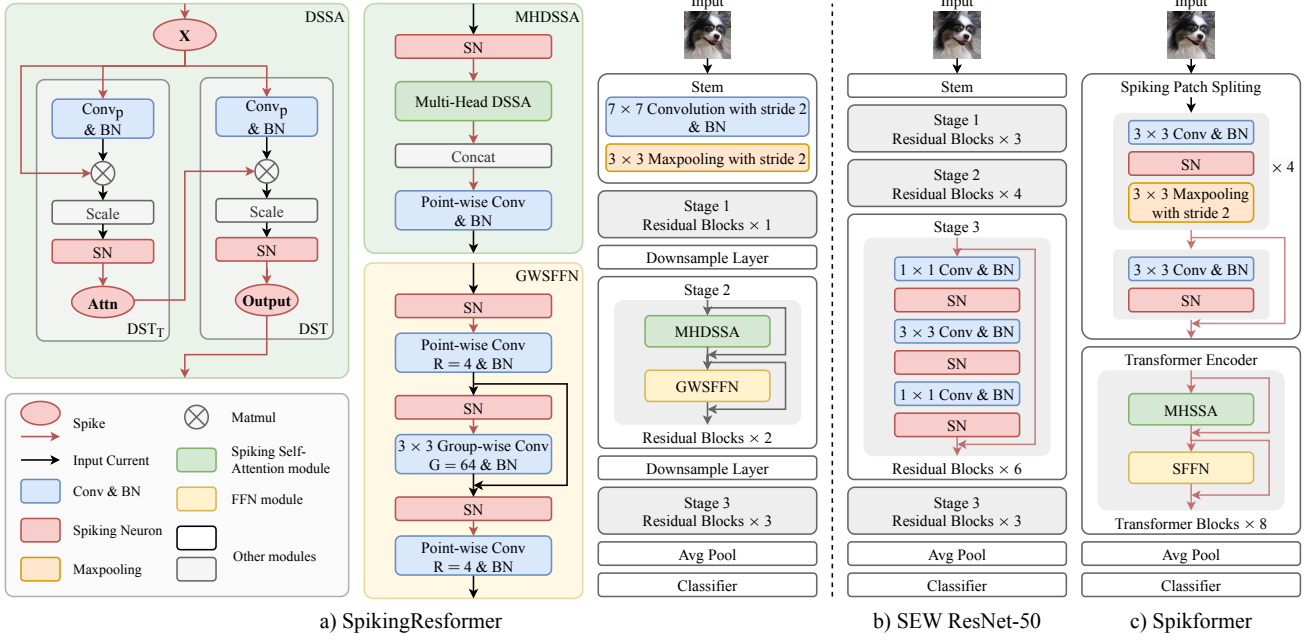


Figure 2. **Left:** Architecture of SpikingResformer and components including Dual Spike Self-Attention (DSSA), Multi-Head DSSA (MHDSSA), and Group-Wise Spiking Feed-Forward Network (GWSFFN). **Right:** Architecture of SEW ResNet-50 and Spikformer.

where  $c_2$  is the second scaling factor. Since the form of DSSA is quite different from VSA and existing spiking self-attention mechanisms, we further discuss how DSSA achieves self-attention in the supplementary.

### 4.3. Scaling Factors in DSSA

In the vanilla self-attention mechanism [37], the product of matrices  $\mathbf{Q}$  and  $\mathbf{K}$  in Eq. (6) is scaled by a factor of  $1/\sqrt{d}$  before applying the softmax operation. This scaling is necessary because the magnitude of  $\mathbf{QK}^T$  grows with the embedding dimension  $d_k$ , which can result in gradient vanishing issues after the softmax operation. Formally, assume that all the elements in  $\mathbf{Q}$  and  $\mathbf{K}$  are independent random variables with a mean of 0 and a variance of 1, then each element in their product  $\mathbf{QK}^T$  has mean 0 and variance  $d$ . By multiplying the product with a factor of  $1/\sqrt{d}$ , the variance of the product is scaled to 1.

While DSSA does not employ the softmax function, the surrogate gradient also suffers gradient vanishes without scaling. However, directly using the scaling factor of VSA is not feasible. Due to the spike-based nature of the input in DST and the attention map in DSSA, we cannot assume that they possess a mean of 0 and a variance of 1. Therefore, the scaling factor values in DSSA should be different from those used in VSA. In the following theorem, we present the mean and variance of DST.

**Theorem 1** (Mean and variance of DST). *Given spike input  $\mathbf{X} \in \{0, 1\}^{T \times p \times m}$ ,  $\mathbf{Y} \in \{0, 1\}^{T \times m \times q}$  and linear*

*transformation  $f(\cdot)$  with weight matrix  $\mathbf{W} \in \mathbb{R}^{q \times q}$ ,  $\mathbf{I} \in \mathbb{R}^{T \times p \times q}$  is the output of DST,  $\mathbf{I} = \text{DST}(\mathbf{X}, \mathbf{Y}; f(\cdot))$ . Assume that all elements in  $\mathbf{X}$  and  $\mathbf{Y}$  are independent random variables,  $x_{i_x, j_x}[t]$  in  $\mathbf{X}$  subject to Bernoulli distribution  $x_{i_x, j_x}[t] \sim B(f_x)$ , and the output of linear transformation  $f(\mathbf{Y})$  has mean 0 and variance 1, we have  $E(I_{i_I, j_I}[t]) = 0$ ,  $\text{Var}(I_{i_I, j_I}[t]) = f_x m$ . Similarly, for  $\mathbf{I} = \text{DST}_T(\mathbf{X}, \mathbf{Y}; f(\cdot))$  and  $\mathbf{Y} \in \{0, 1\}^{T \times q \times m}$ ,  $\mathbf{W} \in \mathbb{R}^{m \times m}$ , we also have  $E(I_{i_I, j_I}[t]) = 0$ ,  $\text{Var}(I_{i_I, j_I}[t]) = f_x m$ .*

The proof of Theorem 1 can be found in the supplementary. According to Theorem 1, we have  $c_1 = 1/\sqrt{f_X d}$ ,  $c_2 = 1/\sqrt{f_{\text{Attn}} HW/p^2}$ . Here  $f_X$  and  $f_{\text{Attn}}$  are the average firing rate of  $\mathbf{X}$  and spiking attention map, respectively.

### 4.4. Spike-driven Characteristic of DSSA

The spike-driven characteristic is important for SNNs, i.e., the computation is sparsely triggered with spikes and requires only synaptic operations. Previous work has made a great effort in achieving spike-driven Transformers [42, 45]. In this subsection, we delve into the spike-driven characteristic of DSSA and prove that DSSA is spike-driven. We first give the formal definition of the spike-driven characteristic.

**Definition 1.** *A spiking neural network is spike-driven if the input currents of all neurons satisfy the following form:*

$$I_i[t] = \sum_j w_{i,j} s_j[t] = \sum_{j, s_j[t] \neq 0} w_{i,j}, \quad (12)$$



where  $I_i[t]$  is the input current of the  $i$ -th postsynaptic neuron at time step  $t$ ,  $s_j[t] \in \{0, 1\}$  is the spike output of the  $j$ -th presynaptic neuron,  $w_{i,j}$  is the weight of the synaptic connection from neuron  $j$  to neuron  $i$ .

This definition reveals the nature of the spike-driven characteristic, i.e., the accumulation of input current is sparsely triggered by spikes emitted from presynaptic neurons. It is evident that commonly used linear layers and convolution layers satisfy this definition.

The DSSA has only two spiking neuron layers, including the spiking attention map layer and the output layer. Both of these layers receive input currents derived from DST. Thus, we only need to validate that the DST satisfies the form in Definition 1. We first validate the DST<sub>T</sub> in Eq. (8)

$$\mathbf{I} = \mathbf{X}\mathbf{W}^T\mathbf{Y}^T, \quad (13)$$

$$I_{i,j}[t] = \sum_{k=1}^m \sum_{l=1}^m x_{i,k}[t] w_{l,k} y_{j,l}[t] = \sum_{\substack{k,l \\ (x_{i,k}[t] \wedge y_{j,l}[t]) \neq 0}} w_{l,k}. \quad (14)$$

Slightly different from Definition 1, the spike input here is the logical AND of dual spikes. This can be viewed as a synaptic operation requiring dual spikes to trigger, which is the reason it is called dual spike transformation. Similar to the DST<sub>T</sub>, the DST in Eq. (7) can also be formulated as:

$$\mathbf{I} = \mathbf{X}\mathbf{Y}\mathbf{W}, \quad (15)$$

$$I_{i,j}[t] = \sum_{k=1}^m \sum_{l=1}^q x_{i,k}[t] y_{k,l}[t] w_{l,j} = \sum_{\substack{k,l \\ (x_{i,k}[t] \wedge y_{k,l}[t]) \neq 0}} w_{l,j}. \quad (16)$$

Thus, DSSA is spike-driven.

## 5. SpikingResformer

In this section, we first introduce the overall architecture of the proposed SpikingResformer and compare it with existing SEW ResNet [9] and Spikformer [46]. Then we detail the design of the spiking resformer block.

### 5.1. Overall Architecture

The overall pipeline of SEW ResNet [9], Spikformer [46], and the proposed SpikingResformer are shown in Fig. 2. As shown in Fig. 2, Spikformer employs a Spiking Patch Splitting (SPS) module to project an image to  $d$  dimensional feature with reduced spatial size. However, the local information can only be poorly modeled by this shallow spiking convolutional module and it only generates single-scale feature maps. On the other hand, SEW ResNet is much

deeper than the SPS in Spikformer and have a greater capability in extracting multi-scale feature with the multi-stage architecture, but lacks the global self-attention mechanism which helps extract global information. To overcome the limitations of these architectures while exploiting their advantages, we propose SpikingResformer, which combines the ResNet-based architecture and the spiking self-attention mechanism.

The details of the overall architecture of SpikingResformer series are listed in Tab. 1. Similar to the ResNet-based SNNs [9, 14], our model starts with a stem architecture consisting of a  $7 \times 7$  convolution and a  $3 \times 3$  max pooling to pre-extract localized features and employs a multi-stage backbone to generate multi-scale feature maps. In each stage, multiple spiking Resformer blocks are stacked sequentially. Each spiking Resformer block consists of two modules, named Multi-Head Dual Spike Self-Attention (MHDSSA) block and Group-Wise Spiking Feed-Forward Network (GWSFFN). A downsample layer is applied before each stage (except the first stage) to reduce the size of the feature maps and project them to a higher dimension ( $2 \times$  downsampling of resolution with  $2 \times$  enlargement of dimension). Finally, the model ends with a global average pooling layer and a classification layer.

### 5.2. Spiking Resformer Block

As illustrated in Fig. 2, the spiking Resformer block consists of a Multi-Head Dual Spike Self-Attention (MHDSSA) module and a Group-Wise Spiking Feed-Forward Network (GWSFFN). We first introduce the two modules and then derive the form of the spiking Resformer block.

**Multi-Head Convolutional Spiking Self-Attention.** In Sec. 4, we propose the single-head form of DSSA. It can be easily extended to the multi-head DSSA (MHDSSA) following a similar approach to the vanilla Transformer [37]. In MHDSSA, we first split the results of linear transformation in DST into  $h$  heads, then perform DSSA on each head and concatenate them together. Finally, we use point-wise convolution to project the concatenated features thus fusing the features in different heads. The MHDSSA can be formulated as follows:

$$\text{MHDSSA}(\mathbf{X}) = \text{BN}(\text{Conv}_1([\text{DSSA}_i(\text{SN}(\mathbf{X}))]_{i=1}^h)), \quad (17)$$

where  $[\dots]$  denotes the concatenate operation,  $\text{Conv}_1$  denotes the point-wise convolution.

**Group-Wise Spiking Feed-Forward Network.** The spiking feed-forward network (SFFN) proposed in previous spiking vision transformers is composed of two linear layers with batch normalization and spiking neuron activation [42, 46]. Moreover, the expansion ratio is usually set to 4, i.e., the first layer raises the dimension by a factor of 4 while the second layer reduces to the original dimension.

Table 1. Architectures of SpikingResformer series. The output size corresponds to the input size of  $224 \times 224$ .  $D_i$  and  $H_i$  are the embedding dimension and number of heads of MHDSSA in stage  $i$ , respectively.  $p_i$  denotes that the MHDSSA in stage  $i$  uses  $p_i \times p_i$  convolution in DST.  $R_i$  and  $G_i$  denote the expansion ratio and embedding dimension per group of GWSFFN in stage  $i$ , respectively.

Stage	Output Size	Layer Name	SpikingResformer-Ti	SpikingResformer-S	SpikingResformer-M	SpikingResformer-L
Stem	$56 \times 56$	Stem	Conv $7 \times 7$ , stride 2, Maxpooling $3 \times 3$ , stride 2			
Stage 1	$56 \times 56$	MHDSSA	$\begin{bmatrix} D_1 = 64 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$	$\begin{bmatrix} D_1 = 64 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$	$\begin{bmatrix} D_1 = 64 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$	$\begin{bmatrix} D_1 = 128 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$
		GWSFFN	$\begin{bmatrix} D_1 = 64 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$	$\begin{bmatrix} D_1 = 64 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$	$\begin{bmatrix} D_1 = 64 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$	$\begin{bmatrix} D_1 = 128 \\ H_1 = 1, p_1 = 4 \\ R_1 = 4, G_1 = 64 \end{bmatrix} \times 1$
Stage 2	$28 \times 28$	Downsample	Conv $3 \times 3$ , 192, stride 2	Conv $3 \times 3$ , 256, stride 2	Conv $3 \times 3$ , 384, stride 2	Conv $3 \times 3$ , 512, stride 2
		MHDSSA	$\begin{bmatrix} D_2 = 192 \\ H_2 = 3, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$	$\begin{bmatrix} D_2 = 256 \\ H_2 = 4, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$	$\begin{bmatrix} D_2 = 384 \\ H_2 = 6, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$	$\begin{bmatrix} D_2 = 512 \\ H_2 = 8, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$
Stage 3	$14 \times 14$	GWSFFN	$\begin{bmatrix} D_2 = 192 \\ H_2 = 3, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$	$\begin{bmatrix} D_2 = 256 \\ H_2 = 4, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$	$\begin{bmatrix} D_2 = 384 \\ H_2 = 6, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$	$\begin{bmatrix} D_2 = 512 \\ H_2 = 8, p_2 = 2 \\ R_2 = 4, G_2 = 64 \end{bmatrix} \times 2$
		Downsample	Conv $3 \times 3$ , 384, stride 2	Conv $3 \times 3$ , 512, stride 2	Conv $3 \times 3$ , 768, stride 2	Conv $3 \times 3$ , 1024, stride 2
Stage 3	$14 \times 14$	MHDSSA	$\begin{bmatrix} D_3 = 384 \\ H_3 = 6, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$	$\begin{bmatrix} D_3 = 512 \\ H_3 = 8, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$	$\begin{bmatrix} D_3 = 768 \\ H_3 = 12, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$	$\begin{bmatrix} D_3 = 1024 \\ H_3 = 16, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$
		GWSFFN	$\begin{bmatrix} D_3 = 384 \\ H_3 = 6, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$	$\begin{bmatrix} D_3 = 512 \\ H_3 = 8, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$	$\begin{bmatrix} D_3 = 768 \\ H_3 = 12, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$	$\begin{bmatrix} D_3 = 1024 \\ H_3 = 16, p_3 = 1 \\ R_3 = 4, G_3 = 64 \end{bmatrix} \times 3$
Classifier	$1 \times 1$	Linear	1000-FC			

Based on SFFN, we insert a  $3 \times 3$  convolution layer with the residual connection between two linear layers to enable SFFN to extract local features. Since the dimension of the hidden features between the two linear layers is expanded by a factor of 4 compared to the input, in order to reduce the number of parameters and computational overhead, we use group-wise convolution and set every 64 channels as 1 group. We also employ the spike-driven design in [42, 45]. The group-wise spiking feed-forward network (GWSFFN) can be formulated as follows:

$$\text{FFL}_i(\mathbf{X}) = \text{BN}(\text{Conv}_1(\text{SN}(\mathbf{X}))), \quad i = 1, 2, \quad (18)$$

$$\text{GWL}(\mathbf{X}) = \text{BN}(\text{GWConv}(\text{SN}(\mathbf{X}))) + \mathbf{X}, \quad (19)$$

$$\text{GWSFFN}(\mathbf{X}) = \text{FFL}_2(\text{GWL}(\text{FFL}_1(\mathbf{X}))). \quad (20)$$

Here  $\text{FFL}_i(\cdot)$ ,  $i = 1, 2$  denote the feed-forward layers,  $\text{Conv}_1(\cdot)$  is point-wise convolution ( $1 \times 1$  convolution), which equal to the linear transformation.  $\text{GWL}(\cdot)$  denotes group-wise convolution layer,  $\text{GWConv}(\cdot)$  denotes the group-wise convolution.

**Spiking Resformer Block.** With the MHDSSA module and GWSFFN above, the spiking resformer block can be formulated as:

$$\mathbf{Y}_i = \text{MHDSSA}(\mathbf{X}_i) + \mathbf{X}_i, \quad (21)$$

$$\mathbf{X}_{i+1} = \text{GWSFFN}(\mathbf{Y}_i) + \mathbf{Y}_i. \quad (22)$$

where  $\mathbf{Y}_i$  denotes the output features of MHDSSA module in the  $i$ -th spiking resformer block.

## 6. Experiments

In this section, we first evaluate the performance and energy efficiency of SpikingResformer on the ImageNet classification task. Then, we perform ablation experiments on key

components in SpikingResformer. Finally, we evaluate the transfer learning ability of SpikingResformer.

### 6.1. ImageNet Classification

ImageNet [4] is one of the most typical static image datasets widely used for image classification. For a fair comparison, we generally follow the data augmentation strategy and training setup in [42]. More details of the experimental setup can be found in the supplementary.

**Results.** Our experimental results are listed in Tab. 2. For comparison, we also list the results of existing spiking convolutional networks and spiking Vision Transformers. As shown in Tab. 2, SpikingResformer achieves higher accuracy, fewer parameters, and lower energy consumption at the same time compared to existing methods. For instance, SpikingResformer-Ti achieves 74.34% accuracy with only 11.14M parameters and 2.73G SOPs (2.46mJ), outperforming Spike-driven Transformer-8-384 by 2.06%, saving 5.67M parameters and 1.44mJ energy. SpikingResformer-M achieves 77.24% accuracy with 35.52M parameters and 6.07G SOPs (5.46mJ), outperforming Spike-driven Transformer-8-768 by 0.92% while saving 30.82M parameters. Particularly, the SpikingResformer-L achieves up to 79.40% accuracy when the input size is enlarged to  $288 \times 288$ , which is the state-of-the-art result in SNN field.

### 6.2. Ablation Study

We perform ablation experiments on key components in SpikingResformer, including the multi-stage architecture, the group-wise convolution layer in GWSFFN, and our proposed spiking self-attention mechanism. The ablation experiments are conducted on the ImageNet100 dataset, which is a subset of the ImageNet dataset and consists of 100 categories from the ImageNet dataset. The experimen-

Table 2. Evaluation on ImageNet. SOPs denotes the average synaptic operations of an image inference on ImageNet validation data. Energy is the estimation of energy consumption same as [42, 46]. The default input resolution for training and inference is  $224 \times 224$ . † means the input is enlarged to  $288 \times 288$  in inference. - means the data is not provided in the original paper.

Method	Type	Architecture	T	Param (M)	SOPs (G)	Energy (mJ)	Top-1 Acc. (%)
Spiking ResNet [14]	ANN-to-SNN	ResNet-34	350	21.79	65.28	59.30	71.61
		ResNet-50	350	25.56	78.29	70.93	72.75
STBP-tdBN [44]	Direct Training	Spiking ResNet-34	6	21.79	6.50	6.39	63.72
TET [6]	Direct Training	Spiking ResNet-34	6	21.79	-	-	64.79
		SEW ResNet-34	4	21.79	-	-	68.00
SEW ResNet [9]	Direct Training	SEW ResNet-34	4	21.79	3.88	4.04	67.04
		SEW ResNet-50	4	25.56	4.83	4.89	67.78
		SEW ResNet-101	4	44.55	9.30	8.91	68.76
		SEW ResNet-152	4	60.19	13.72	12.89	69.26
Spikformer [46]	Direct Training	Spikformer-8-384	4	16.81	6.82	7.73	70.24
		Spikformer-8-512	4	29.68	11.09	11.58	73.38
		Spikformer-8-768	4	66.34	22.09	21.48	74.81
Spikingformer [45]	Direct Training	Spikingformer-8-384	4	16.81	-	4.69	72.45
		Spikingformer-8-512	4	29.68	-	7.46	74.79
		Spikingformer-8-768	4	66.34	-	13.68	75.85
Spike-driven Transformer [42]	Direct Training	Spike-driven Transformer-8-384	4	16.81	-	3.90	72.28
		Spike-driven Transformer-8-512	4	29.68	-	4.50	74.57
		Spike-driven Transformer-8-768	4	66.34	-/-	6.09/-	76.32/77.07†
SpikingResformer (Ours)	Direct Training	SpikingResformer-Ti	4	<b>11.14</b>	<b>2.73/4.71†</b>	<b>2.46/4.24†</b>	<b>74.34/75.57†</b>
		SpikingResformer-S	4	<b>17.76</b>	<b>3.74/6.40†</b>	<b>3.37/5.76†</b>	<b>75.95/76.90†</b>
		SpikingResformer-M	4	<b>35.52</b>	<b>6.07/10.24†</b>	<b>5.46/9.22†</b>	<b>77.24/78.06†</b>
		SpikingResformer-L	4	<b>60.38</b>	<b>9.74/16.40†</b>	<b>8.76/14.76†</b>	<b>78.77/79.40†</b>

Table 3. Ablation Study on ImageNet100 dataset. The number of parameters for all variants is comparable to SpikingResformer-S.

Model	SOPs (G)	Energy (mJ)	Acc. (%)
SpikingResformer-S	2.43	2.18	88.06
w/o multi-stage architecture	1.84	1.66	85.32
w/o group-wise convolution	2.37	2.13	84.64
w/o DSSA		not converge	
w/o $p \times p$ convolution	4.34	3.91	86.14
w/o scaling		not converge	

tal setup basically follows the one in Sec. 6.1. Detailed settings are listed in the supplementary.

**Multi-Stage Architecture.** To verify the effectiveness of the multi-stage architecture, we replace it with the Spikingformer-based architecture, while the structure of the spiking Resformer block remains unchanged. We adjust the embedding dimensions to make the model parameters comparable to SpikingResformer-S. As shown in Tab. 3, SpikingResformer-S outperforms the variant without multi-stage architecture by 2.74%, demonstrating the effectiveness of multi-stage architecture.

**Group-Wise Convolution Layer.** In comparison to the point-wise SFFN, GWSFFN employs a  $3 \times 3$  group-wise convolution layer between the two linear layers. We

evaluate its effect by removing the group-wise convolution layer and increasing the dimension to keep the number of parameters constant. As demonstrated in Tab. 3, SpikingResformer-S achieves 3.42% higher accuracy in contrast to the variant without the group-wise convolution layer. This difference highlights the benefits of the group-wise convolution layer in GWSFFN.

**Dual Spike Self-Attention.** To validate the effectiveness of DSSA, we first replace DSSA with existing spiking self-attention mechanisms, including Spiking Self-Attention (SSA) in Spikformer and Spike-Driven Self-Attention (SDSA) in Spike-driven Transformer. However, neither SSA nor SDSA converges. We believe this is because SSA and SDSA do not fit the multi-scale input. To further validate the key factors that help DSSA fit for the multi-scale input, we conduct two more sets of experiments. One set replaces all  $p \times p$  convolutions with  $1 \times 1$  convolutions in DST to validate the effect of reducing spatial size. The other removes the scaling factor or replaces our proposed scaling factors with  $1/\sqrt{d}$  to validate the effect of scaling. As a result, the first group converges but only achieves 86.14% accuracy with a higher energy consumption of 3.91mJ, while the second set does not converge. This shows that the scaling factor is the key to convergence. We believe that both SSA and SDSA lack proper scaling meth-

Table 4. Transfer learning results on CIFAR10, CIFAR100, CIFAR10-DVS, DVSGesture datasets.

Method	Type	CIFAR10		CIFAR100		CIFAR10-DVS		DVSGesture	
		T	Acc.	T	Acc.	T	Acc.	T	Acc.
STBP-tdBN [44]	Direct Training	6	93.16	-	-	10	67.8	40	96.87
PLIF [10]	Direct Training	8	93.50	-	-	20	74.8	20	97.57
Dspike [25]	Direct Training	6	94.25	6	74.24	10	75.4	-	-
Spikformer [46]	Direct Training	4	95.19	4	77.86	16	80.6	16	97.9
Spikingformer [45]	Direct Training	4	95.61	4	79.09	16	81.3	16	98.3
Spike-driven Transformer [42]	Direct Training	4	95.6	4	78.4	16	80.0	16	<b>99.3</b>
Spikformer [46]	Transfer Learning	4	97.03	4	83.83	-	-	-	-
<b>SpikingResformer (Ours)</b>	Transfer Learning	4	<b>97.40</b>	4	<b>85.98</b>	10	<b>84.8</b>	10	93.4

ods, thus do not apply to multi-stage architecture. SSA employs the same scaling factor as vanilla self-attention, i.e.,  $1/\sqrt{d}$ . However, this does not apply to spike matrix multiplication, since spikes do not have mean 0 and variance 1. SDSA has no scaling method, which may be feasible when the feature map is small but not for multi-scale inputs.

### 6.3. Transfer Learning

High transfer learning ability is a key advantage of Vision Transformer. We evaluate the transfer learning ability of the proposed SpikingResformer on static dataset CIFAR10 and CIFAR100 [19] and neuromorphic dataset CIFAR10-DVS [23] and DVSGesture [1] by fine-tuning the models pre-trained on ImageNet. Among the existing spiking Vision Transformers, only Spikformer provides transfer learning results, and only on the static image datasets CIFAR10 and CIFAR100. Thus, we also compared our results with the direct training results for a comprehensive comparison. Tab. 4 lists the highest accuracy achieved by these methods. The experimental setup and a more detailed comparison can be found in the supplementary.

**Static Image Datasets.** As shown in Tab. 4, SpikingResformer achieves 97.40% accuracy on CIFAR10 dataset and 85.98% accuracy on CIFAR100 dataset, which is the state-of-the-art result, outperforming the transfer learning results of Spikformer by 0.37% on CIFAR10 and 2.15% on CIFAR100. Compared to direct training methods, SpikingResformer obtained from transfer learning has significantly better performance. For example, SpikingResformer outperforms Spikingformer by 6.89% on CIFAR100 dataset, demonstrating the advantage of transfer learning.

**Neuromorphic Datasets.** Neuromorphic datasets differ significantly from traditional static image datasets. The samples of the neuromorphic dataset consist of event streams instead of RGB images. As a result, there is a large gap between the source and target domain for models pre-trained on static image datasets. To bridge this gap, we stack the events over a period of time to form a frame. The RGB channels are replaced with positive events,

negative events, and the sum of events channels, respectively. As shown in Tab. 4, transfer learning results of SpikingResformer significantly outperform the direct training ones on CIFAR10-DVS. SpikingResformer achieves 84.8% accuracy on CIFAR10-DVS, outperforming Spikingformer by 3.5%. However, the transfer learning results on DVSGesture fail to achieve comparable performance to direct training. SpikingResformer only achieves 93.4% accuracy on DVSGesture, falling behind the state-of-the-art method Spike-driven Transformer by 5.9%. We believe that this is mainly due to the way CIFAR10-DVS is constructed differs from DVSGesture. CIFAR10-DVS is converted from CIFAR10, which does not contain temporal information. Thus, models pre-trained on static datasets can transfer to CIFAR10-DVS well. However, DVSGesture is directly created from human gestures using a dynamic vision sensor, thus containing rich temporal information. As a result, models pre-trained on static datasets do not transfer well to DVSGesture. We hope our exploration of transfer learning on neuromorphic datasets could pave the way for further transfer learning research of SNNs.

## 7. Conclusion

In this paper, we propose a novel spiking self-attention mechanism named Dual Spike Self-Attention (DSSA). It produces spiking self-attention via Dual Spike Transformation, which is fully spike-driven and compatible with SNNs. We detail the scaling factors in DSSA enabling it to handle feature maps of arbitrary scales. Based on DSSA, we propose SpikingResformer, which combines the ResNet-based multi-stage architecture with our proposed DSSA to achieve superior performance and energy efficiency with fewer parameters. Extensive experiments demonstrate the effectiveness and superiority of the proposed SpikingResformer.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (62176003, 62088102) and by the Beijing Nova Program (20230484362).



## References

- [1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017. 2, 8, 15
- [2] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11–20, 2022. 2
- [3] Tong Bu, Wei Fang, Jianhao Ding, PENG LIN DAI, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *Proceedings of the International Conference on Learning Representations*, pages 1–19, 2023. 2
- [4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6, 14
- [5] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *Proceedings of the International Conference on Learning Representations*, pages 1–14, 2021. 2
- [6] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *Proceedings of the International Conference on Learning Representations*, pages 1–15, 2022. 2, 7
- [7] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2328–2336, 2021. 2
- [8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, pages 1–22, 2021. 1
- [9] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 21056–21069, 2021. 2, 5, 7
- [10] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2661–2671, 2021. 8, 16
- [11] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spiNNaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014. 1
- [12] Yufei Guo, Yuhang Zhang, Yuanpei Chen, Weihang Peng, Xiaode Liu, Liwen Zhang, Xuhui Huang, and Zhe Ma. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19420–19430, 2023. 2
- [13] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ANN-SNN conversion error through residual membrane potential. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11–21, 2023. 2
- [14] Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2021. 2, 5, 7
- [15] Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):14546–14562, 2023.
- [16] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ANN-SNN conversion: Towards optimal mapping from activation values to firing rates. In *Proceedings of the International Conference on Machine Learning*, pages 14945–14974, 2023. 2
- [17] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-YOLO: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11270–11277, 2020. 2
- [18] Paul Kirkland, Gaetano Di Caterina, John Soraghan, and George Matich. Spikeseg: Spiking segmentation via STDP saliency mapping. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2020. 2
- [19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 8, 14
- [20] Yuxiang Lan, Yachao Zhang, Xu Ma, Yanyun Qu, and Yun Fu. Efficient converted spiking neural network for 3d and 2d classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9211–9220, 2023. 2
- [21] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14:119, 2020. 2
- [22] Chen Li, Edward Jones, and Steve Furber. Unleashing the potential of spiking neural networks by dynamic confidence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13350–13360, 2023. 2
- [23] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in Neuroscience*, 11:309, 2017. 8, 15
- [24] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the International Conference on Machine Learning*, pages 6316–6325, 2021. 2
- [25] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 23426–23439, 2021. 8, 16

- [26] Yuhang Li, Youngeun Kim, Hyoungeob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. In *Proceedings of the European Conference on Computer Vision*, pages 631–649, 2022. 1
- [27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1
- [28] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997. 1
- [29] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6166–6176, 2023. 2
- [30] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. 1
- [31] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. 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. 2
- [32] Kinjal Patel, Eric Hunsberger, Sean Batir, and Chris Elia-smith. A spiking neural network for image segmentation. *arXiv preprint arXiv:2106.08921*, pages 1–25, 2021. 2
- [33] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019. 1
- [34] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*, pages 1–88, 2017. 1
- [35] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 13:95, 2019. 2
- [36] Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6555–6565, 2023. 2
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:1–11, 2017. 3, 4, 5
- [38] Xiao Wang, Zongzhen Wu, Yao Rong, Lin Zhu, Bo Jiang, Jin Tang, and Yonghong Tian. SSTFormer: Bridging spiking neural network and memory support transformer for frame-event based recognition. *arXiv preprint arXiv:2308.04369*, pages 1–12, 2023. 1
- [39] Wenjie Wei, Malu Zhang, Hong Qu, Ammar Belatreche, Jian Zhang, and Hong Chen. Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven backpropagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10552–10562, 2023. 2
- [40] Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7824–7840, 2021. 2
- [41] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7886–7895, 2023. 2
- [42] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in neural information processing systems*, pages 1–20, 2023. 2, 4, 5, 6, 7, 8, 12, 14, 16
- [43] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021. 2
- [44] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11062–11070, 2021. 2, 7, 8, 16
- [45] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Han Zhang, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Spiking-former: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, pages 1–16, 2023. 2, 4, 6, 7, 8, 16
- [46] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, YAN Shuicheng, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *Proceedings of the International Conference on Learning Representations*, pages 1–17, 2023. 2, 5, 7, 8, 12, 16
- [47] Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. In *Advances in Neural Information Processing Systems*, pages 30528–30541, 2022. 2
- [48] Yaoyu Zhu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Zhaofei Yu. Exploring loss functions for time-based training strategy in spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 65366–65379, 2023. 2
- [49] Shihao Zou, Yuxuan Mu, Xinxin Zuo, Sen Wang, and Li Cheng. Event-based human pose tracking by spiking spatiotemporal transformer. *arXiv preprint arXiv:2303.09681*, pages 1–12, 2023. 1

# SpikingResformer: Bridging ResNet and Vision Transformer in Spiking Neural Networks

## Supplementary Material

### A. Proof of Theorem 1

**Lemma 1. (Expectation and variance of the product of independent random variables)** *Given two independent random variables  $a$  and  $b$  with expectation and variance, we have*

$$E(ab) = E(a)E(b), \quad (S1)$$

$$\text{Var}(ab) = \text{Var}(a)\text{Var}(b) + \text{Var}(a)E(b)^2 + \text{Var}(b)E(a)^2. \quad (S2)$$

*Proof.* The expectation of  $ab$  can be formulated as:

$$E(ab) = E(a)E(b) + \text{Cov}(a, b). \quad (S3)$$

Since the random variables  $a$  and  $b$  are independent of each other, the covariance  $\text{Cov}(a, b) = 0$ . Thus, we have

$$E(ab) = E(a)E(b) + 0 = E(a)E(b). \quad (S4)$$

Using the above conclusion and the definition of variance, we have

$$\begin{aligned} \text{Var}(ab) &= E((ab - E(ab))^2) \\ &= E(a^2b^2) - E(ab)^2 \\ &= E(a^2)E(b^2) - E(a)^2E(b)^2 \\ &= (\text{Var}(a) + E(a)^2)(\text{Var}(b) + E(b)^2) - E(a)^2E(b)^2 \\ &= \text{Var}(a)\text{Var}(b) + \text{Var}(a)E(b)^2 + \text{Var}(b)E(a)^2. \end{aligned} \quad (S5)$$

□

**Lemma 2. (Expectation and variance of the sum of independent random variables)** *Given independent random variables  $a_1, a_2, \dots, a_n$  with expectation and variance, we have*

$$E\left(\sum_{i=1}^n a_i\right) = \sum_{i=1}^n E(a_i), \quad (S6)$$

$$\text{Var}\left(\sum_{i=1}^n a_i\right) = \sum_{i=1}^n \text{Var}(a_i). \quad (S7)$$

*Proof.* Considering first the case of two independent random variables  $a_i$  and  $a_j$  where  $i \neq j$ , the covariance  $\text{Cov}(a_i, a_j) = 0$ , we have

$$E(a_i + a_j) = E(a_i) + E(a_j), \quad (S8)$$

$$\begin{aligned} \text{Var}(a_i + a_j) &= \text{Var}(a_i) + \text{Var}(a_j) + 2\text{Cov}(a_i, a_j) \\ &= \text{Var}(a_i) + \text{Var}(a_j) \end{aligned} \quad (S9)$$

This can be simply generalized to the case of  $n$  random variables as:

$$E\left(\sum_{i=1}^n a_i\right) = \sum_{i=1}^n E(a_i), \quad (S10)$$

$$\begin{aligned} \text{Var}\left(\sum_{i=1}^n a_i\right) &= \sum_{i=1}^n \text{Var}(a_i) + \sum_{1 \leq i, j \leq n, i \neq j} \text{Cov}(a_i, a_j) \\ &= \sum_{i=1}^n \text{Var}(a_i). \end{aligned} \quad (S11)$$

□

With Lemma 1 and Lemma 2, we prove the Theorem 1 in the main text.

*Proof.* Let us first consider the case of  $\text{DST}_T(\mathbf{X}, \mathbf{Y}; f(\cdot))$ . We denote the result of applying linear transformation  $f$  on  $\mathbf{Y}$  as  $\mathbf{Z} = f(\mathbf{Y}) = \mathbf{Y}\mathbf{W}$ . Thus, each element in the result of  $\text{DST}_T$  can be formulated as:

$$I_{i,j}[t] = \sum_{k=1}^m x_{i,k}[t]z_{j,k}[t] = \sum_{k=1}^m x_{i,k}[t] \left( \sum_{l=1}^m y_{j,l}[t]w_{l,k}[t] \right) \quad (S12)$$

Based on the assumption, each  $z_{j,k}[t]$  has a mean of 0 and variance of 1, and each  $x_{i,k}[t]$  subjects to Bernoulli distribution  $B(f_x)$ , thus we have  $E(x_{i,k}[t]) = f_x$  and  $\text{Var}(x_{i,k}[t]) = f_x(1 - f_x)$ . According to Lemma 1, we have

$$E(x_{i,k}[t]z_{j,k}[t]) = 0 \cdot f_x = 0, \quad (S13)$$

$$\begin{aligned} \text{Var}(x_{i,k}[t]z_{j,k}[t]) &= 1 \cdot f_x(1 - f_x) + 1 \cdot f_x^2 \\ &\quad + f_x(1 - f_x) \cdot 0^2 \\ &= f_x. \end{aligned} \quad (S14)$$

In addition, each  $z_{j,k}[t], k = 1, \dots, q$  consists of a set of  $y_{l,k}[t], l = 1, \dots, q$  that do not overlap each other. Thus  $z_{j,k}[t]$  can also be viewed as independent random variables. According to Lemma 2, we have

$$E(I_{i,j}[t]) = \sum_{k=1}^m 0 = 0, \quad (S15)$$

$$\text{Var}(I_{i,j}[t]) = \sum_{k=1}^m f_x = f_x m. \quad (S16)$$

Similar to  $\text{DST}_T$ , each element in the result of DST can be formulated as:

$$I_{i,j}[t] = \sum_{k=1}^m x_{i,k}[t] z_{k,j}[t] = \sum_{k=1}^m x_{i,k}[t] \left( \sum_{l=1}^q y_{k,l}[t] w_{l,j}[t] \right) \quad (\text{S17})$$

And we also have  $E(x_{i,k}[t] z_{k,j}[t]) = 0$  and  $\text{Var}(x_{i,k}[t] z_{k,j}[t]) = f_x$ . Thus, the expectation and variance of  $I_{i,j}[t]$  are also  $E(I_{i,j}[t]) = 0$  and  $\text{Var}(I_{i,j}[t]) = f_x m$ .  $\square$

**Further Discussion.** In Eq. (S15) and Eq. (S16), we treat  $z_{j,k}[t]$  as independent random variables, since each  $z_{j,k}[t]$ ,  $k = 1, \dots, q$  consists of a set of  $y_{l,k}[t]$ ,  $l = 1, \dots, q$  that do not overlap each other. It is true for the  $p \times p$  convolution with stride  $p$  used in this paper. However, this property does not hold for all generalized linear transformations. For example, the  $3 \times 3$  convolution with stride 1 leads to input overlap. For these operations, we need a stronger assumption that assuming  $z_{j,k}[t]$  are independent random variables.

## B. Scaling Factors in Existing Spiking Self-Attention Mechanisms

In the main text, we propose that existing spiking self-attention mechanisms lack reasonable scaling methods and design scaling factors for our DSSA. In this section, we use a similar approach to design scaling factors for these existing methods and thereby analyze the limitations of these methods.

**Scaling Factor in Spiking Self-Attention (SSA).** First, we try to design the scaling factor for the Spiking Self-Attention (SSA) in Spikformer [46]. The SSA can be formulated as follows:

$$\mathbf{Q} = \text{SN}(\text{BN}(\mathbf{X}\mathbf{W}_Q)), \quad (\text{S18})$$

$$\mathbf{K} = \text{SN}(\text{BN}(\mathbf{X}\mathbf{W}_K)), \quad (\text{S19})$$

$$\mathbf{V} = \text{SN}(\text{BN}(\mathbf{X}\mathbf{W}_V)), \quad (\text{S20})$$

$$\text{SSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{SN}(\mathbf{Q}\mathbf{K}^T \mathbf{V} * c), \quad (\text{S21})$$

where  $\mathbf{X} \in \mathbb{R}^{HW \times d}$  is the input,  $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d}$  are weight matrices,  $H$  and  $W$  are the height and width of input, respectively,  $d$  is the embedding dimension,  $c$  is the scaling factor. We denote  $\mathbf{I} = \mathbf{Q}\mathbf{K}^T \mathbf{V}$ , each element in  $\mathbf{I}$  can be formulated as:

$$I_{i,j}[t] = \sum_{r=1}^d \sum_{l=1}^{HW} q_{i,r}[t] k_{r,l}[t] v_{l,j}[t]. \quad (\text{S22})$$

Assume that all elements in  $\mathbf{Q}, \mathbf{K}$ , and  $\mathbf{V}$  are independent random variables,  $q_{i,q,j_q}[t]$  in  $\mathbf{Q}$  subject to Bernoulli distribution  $q_{i,q,j_q}[t] \sim B(f_Q)$ ,  $k_{i_k,j_k}[t]$  in  $\mathbf{K}$  subject to  $B(f_K)$ ,  $v_{i_v,j_v}[t]$  in  $\mathbf{V}$  subject to  $B(f_V)$ , respectively,  $f_Q, f_K$ , and

$f_V$  are the average firing rate of  $\mathbf{Q}, \mathbf{K}$ , and  $\mathbf{V}$ , respectively. We have  $E(I_{i,j}[t]) = HWdf_Qf_Kf_V$ .

However, the form of variance is complex. This is because the summation terms  $q_{i,r}[t]k_{r,l}[t]v_{l,j}[t]$  are not independent thus introducing a lot of covariance. The variance can be formulated as:

$$\begin{aligned} \text{Var}(I_{i,j}[t]) &= \sum_{r=1}^d \sum_{l=1}^{HW} \left( \text{Var}(q_{i,r}[t]k_{r,l}[t]v_{l,j}[t]) \right. \\ &\quad + \sum_{r' \neq r} \text{Cov}(q_{i,r}[t]k_{r,l}[t]v_{l,j}[t], q_{i,r'}[t]k_{r',l}[t]v_{l,j}[t]) \\ &\quad \left. + \sum_{l' \neq l} \text{Cov}(q_{i,r}[t]k_{r,l}[t]v_{l,j}[t], q_{i,r}[t]k_{r,l'}[t]v_{l',j}[t]) \right) \\ &= HWd \left( f_Qf_Kf_V(1-f_Q)(1-f_K)(1-f_V) \right. \\ &\quad + f_Qf_Kf_V^2(1-f_Q)(1-f_K) \\ &\quad + f_Qf_K^2f_V(1-f_Q)(1-f_V) \\ &\quad + f_Q^2f_Kf_V(1-f_K)(1-f_V) \\ &\quad + f_Qf_K^2f_V^2(1-f_Q) \\ &\quad + f_Q^2f_Kf_V^2(1-f_K) \\ &\quad + f_Q^2f_K^2f_V(1-f_V) \\ &\quad + (d-1)(f_Q^2f_K^2f_V - f_Q^2f_K^2f_V^2) \\ &\quad \left. + (HW-1)(f_Qf_K^2f_V^2 - f_Q^2f_K^2f_V^2) \right) \\ &= HWdf_Qf_Kf_V \left( 1 - (HW+d-1)f_Qf_Kf_V \right. \\ &\quad \left. + (d-1)f_Qf_K + (HW-1)f_Kf_V \right). \end{aligned} \quad (\text{S23})$$

As shown in Eq. (S23), this form is overly complex and lacks practicality. Thus it is difficult to design the scaling factor for SSA.

**Scaling Factor in Spike-driven Self-Attention (SDSA).** Next, we try to design the scaling factor for the Spike-driven Self-Attention (SDSA) in Spike-driven Transformer [42]. The SDSA can be formulated as follows:

$$\text{SDSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{SN}(\text{SUM}_c(\mathbf{Q} \otimes \mathbf{K})) \otimes \mathbf{V}, \quad (\text{S24})$$

where  $\otimes$  denotes Hadamard product,  $\text{SUM}_c$  represents the sum of each column,  $\mathbf{Q}, \mathbf{K}$ , and  $\mathbf{V}$  are the same as in Eq. (S18) to Eq. (S20). The original SDSA does not have a scaling factor. We believe that there should be a scaling factor before the spiking neuron layer and the Eq. (S24) should be reformulated as follows:

$$\text{SDSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{SN}(\text{SUM}_c(\mathbf{Q} \otimes \mathbf{K}) * c) \otimes \mathbf{V}, \quad (\text{S25})$$



Table S1. Further ablation study on ImageNet100 dataset. The number of parameters for all variants is comparable to SpikingResformer-S.

Model	Acc. (%)
SpikingResformer-S	88.06
Spike-driven Transformer-8-384 (w/o scaling)	83.06
Spike-driven Transformer-8-384 (with scaling)	83.96
SpikingResformer-S with SDSA (w/o scaling)	not-converge
SpikingResformer-S with SDSA (with scaling)	87.74

where  $c$  is the scaling factor. We denote  $\mathbf{I} = \text{SUM}_c(\mathbf{Q} \otimes \mathbf{K})$ , each element in  $\mathbf{I}$  can be formulated as:

$$I_j[t] = \sum_{i=1}^{HW} q_{i,j}[t] k_{i,j}[t]. \quad (\text{S26})$$

Following the same assumption in SSA, we have

$$\begin{aligned} \text{Var}(I_j[t]) &= \sum_{i=1}^{HW} \text{Var}(q_{i,j}[t] k_{i,j}[t]) \\ &= \sum_{i=1}^{HW} (f_Q f_K (1 - f_Q f_K)) \\ &= HW f_Q f_K (1 - f_Q f_K). \end{aligned} \quad (\text{S27})$$

Thus, the scaling factor  $c$  in SDSA should be  $c = 1/\sqrt{HW f_Q f_K (1 - f_Q f_K)}$ .

To validate the effectiveness of this scaling factor, we conduct two sets of further ablation experiments. One set introduces our proposed scaling factor to the Spike-driven Transformer. The other replaces the DSSA in SpikingResformer with the SDSA with our proposed scaling factor. The  $p \times p$  convolutions and the GWSFFN remain unchanged. Experimental results are listed in Tab. S1. As shown in Tab. S1, the scaling factor successfully solves the non-converge problem, demonstrating the effectiveness of our proposed scaling factor and its necessity for multi-scale feature map inputs. Moreover, the scaling factor also improves the performance of SDSA with single-scale feature map inputs.

### C. Equivalence of Convolution to Linear Transformation

In this section, we discuss the equivalence of convolution to linear transformation. For ease of understanding, we first visualize a simple example, and then give a formal description of the equivalence. Since no dynamics in the temporal domain are involved here, we omit the time dimension.

Fig. S1 shows how a  $2 \times 2$  convolution with a stride of 2 on a  $4 \times 4$  input is equivalent to a linear transformation.

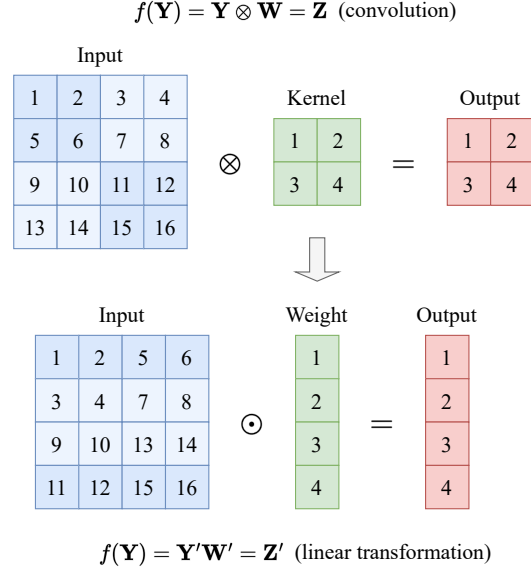


Figure S1. Diagram of the equivalence of convolution to linear transformation. **Top:**  $\text{Conv}_p(\cdot)$  on a  $4 \times 4$  input where  $p = 2$ ; **Bottom:** Its equivalent linear transformation.

In order to convert the convolution to its equivalent linear transformation, we first reshape the  $h \times w$  convolution kernel  $\mathbf{W}$  to a  $hw \times 1$  weight matrix  $\mathbf{W}'$ , where  $h$  and  $w$  are the height and width of the convolution kernel, respectively. Here  $h = w = 2$ . Then, we rewrite the  $H_{in} \times W_{in}$  input  $\mathbf{Y}$  to  $H_{out} W_{out} \times hw$  input  $\mathbf{Y}'$ , where  $H_{in}$  and  $W_{in}$  are the height and width of the input,  $H_{out}$  and  $W_{out}$  are the height and width of the output, respectively. Here  $H_{in} = W_{in} = 4$ ,  $H_{out} = W_{out} = 2$ . Each row in  $\mathbf{Y}'$  is a patch of input corresponding to an element in the output. By the above process, we convert the convolution to its equivalent linear transformation.

We give a formal description of the equivalence of convolution to linear transformation. Given an input  $\mathbf{Y} \in \{0, 1\}^{H_{in} \times W_{in} \times C_{in}}$  and a convolution kernel  $\mathbf{W} \in \mathbb{R}^{h \times w \times C_{out} \times C_{in}}$ , the convolution with stride  $p$  can be formulated as follows:

$$\mathbf{Z} = \mathbf{Y} \otimes \mathbf{W}, \quad (\text{S28})$$

$$\begin{aligned} & z_{i,j,c_{out}} \\ &= \sum_{k=1}^h \sum_{l=1}^w \sum_{c_{in}=1}^{C_{in}} (y_{(i-1)*p+k, (j-1)*p+l, c_{in}} \cdot w_{k,l,c_{out}, c_{in}}). \end{aligned} \quad (\text{S29})$$

Let  $g_Y$  be a mapping from  $\{0, 1\}^{H_{in} \times W_{in} \times C_{in}}$  to  $\{0, 1\}^{H_{out} W_{out} \times hw C_{in}}$  and  $g_W$  be a mapping from

$\mathbb{R}^{h \times w \times C_{out} \times C_{in}}$  to  $\mathbb{R}^{hwC_{in} \times C_{out}}$ , where

$$\mathbf{Y}' = g_Y(\mathbf{Y}),$$

$$s.t. \ y'_{i*W_{out}+j, c_{in}*hw+k*w+l} = y_{(i-1)*p+k, (j-1)*p+l, c_{in}}, \quad (S30)$$

$$\mathbf{W}' = g_W(\mathbf{W}),$$

$$s.t. \ w'_{c_{in}*hw+k*w+l, c_{out}} = w_{k, l, c_{out}, c_{in}}. \quad (S31)$$

The linear transformation of  $g_Y(\mathbf{Y})$  with weight matrix  $g_W(\mathbf{W})$  can be formulated as:

$$\mathbf{Z}' = \mathbf{Y}'\mathbf{W}' = g_Y(\mathbf{Y})g_W(\mathbf{W}), \quad (S32)$$

$$\begin{aligned} & z'_{i*W_{out}+j, c_{out}} \\ &= \sum_{k=1}^h \sum_{l=1}^w \sum_{c_{in}=1}^{C_{in}} \\ & \quad (y'_{i*W_{out}+j, c_{in}*hw+k*w+l} \cdot w'_{c_{in}*hw+k*w+l, c_{out}}) \\ &= \sum_{k=1}^h \sum_{l=1}^w \sum_{c_{in}=1}^{C_{in}} (y_{(i-1)*p+k, (j-1)*p+l, c_{in}} \cdot w_{k, l, c_{out}, c_{in}}) \\ &= z_{i, j, c_{out}}. \end{aligned} \quad (S33)$$

Thus, the convolution is equivalent to linear transformation.

## D. Self-Attention in DSSA

Dual Spike Self-Attention (DSSA) has no explicit Query, Key, and Value, which makes it quite different from the form of the Vanilla Self-Attention (VSA). In this section, we further discuss how DSSA achieves self-attention.

**Recall.** The DSSA can be formulated as follows:

$$\text{DSSA}(\mathbf{X}) = \text{SN}(\text{DST}(\text{AttnMap}(\mathbf{X}), \mathbf{X}; f(\cdot)) * c_2), \quad (S34)$$

$$\text{AttnMap}(\mathbf{X}) = \text{SN}(\text{DST}_T(\mathbf{X}, \mathbf{X}; f(\cdot)) * c_1), \quad (S35)$$

$$f(\mathbf{X}) = \text{BN}(\text{Conv}_p(\mathbf{X})). \quad (S36)$$

And the Dual Spike Transformation (DST) can be formulated as follows:

$$\text{DST}(\mathbf{X}, \mathbf{Y}; f(\cdot)) = \mathbf{X}f(\mathbf{Y}) = \mathbf{X}\mathbf{Y}\mathbf{W}, \quad (S37)$$

$$\text{DST}_T(\mathbf{X}, \mathbf{Y}; f(\cdot)) = \mathbf{X}f(\mathbf{Y})^T = \mathbf{X}\mathbf{W}^T\mathbf{Y}^T. \quad (S38)$$

DSSA achieves self-attention by the two DSTs. The first one,  $\text{DST}_T(\mathbf{X}, \mathbf{X}; f(\cdot))$ , produces the attention map. It computes the multiplicative attention of a pixel in the input  $\mathbf{X}$  and a  $p \times p$  patch of feature transformed by the  $p \times p$  convolution. The output of  $\text{DST}_T(\mathbf{X}, \mathbf{X}; f(\cdot))$  is then scaled and fed to the spiking neuron as the input current to generate the spiking attention map. The spiking attention map is a binary attention map consisting of spikes. Each spike  $s_{i,j}$  in this spiking attention map signifies attention between the patch  $i$  (pixel  $i$ ) and patch  $j$ . The second one,

$\text{DST}(\text{AttnMap}(\mathbf{X}), \mathbf{X}; f(\cdot))$ , produces the output feature. For each pixel, it computes the sum of features of patches that have attention to this pixel to form the output features. In this way, the first DST is similar to the product of  $\mathbf{Q}\mathbf{K}^T$  in the VSA, and the second DST is similar to the product of attention map and  $\mathbf{V}$  in the VSA.

## E. Experiment Details

**ImageNet Classification.** ImageNet [4] is a vast collection of static images and one of the most commonly used datasets in computer vision tasks. It consists of around 1.2 million high-resolution images, categorized into 1,000 distinct classes. Each class includes approximately 1,000 images, representing a diverse range of objects and scenes, making it an effective reflection of real-world scenarios.

For ImageNet classification experiments, we generally follow the data augmentation strategy and training setup in [42]. We use the standard preprocessing, i.e., data normalization, randomly crop and resize the input to  $224 \times 224$  during training, and set the input size to  $224 \times 224$  and  $288 \times 288$  for inference. We employ the standard data augmentation methods including random augmentation, mixup, cutmix, and label smoothing<sup>1</sup>, similar to [42]. We use the AdamW optimizer with a weight decay of 0.01. The batch size varies from 256 (SpikingResformer-Ti) to 128 (SpikingResformer-L) depending on the model size. We train the models for 320 epochs with a cosine-decay learning rate whose initial value varies from 0.001 (SpikingResformer-Ti) to 0.0005 (SpikingResformer-L).

Since the scaling factors in DSSA require the firing rate of input  $f_X$  and attention map  $f_{\text{Attn}}$ , we use an exponential moving average with a momentum of 0.999 to count the average firing rate during training, and use the average firing rate counted during training in inference. We used the same method to count the average firing rate in all subsequent experiments.

**Ablation Study.** All the ablation experiments are conducted on the ImageNet100 dataset. It is a subset of the ImageNet dataset consisting of 100 categories from the original ImageNet dataset. The experimental setup basically follows the ImageNet classification experiments. The weight decay is increased to 0.05 since the ImageNet100 is smaller and easy to overfit.

**Transfer Learning on Static Image Datasets.** We first perform transfer learning experiments on static image datasets CIFAR10 and CIFAR100 [19]. The CIFAR-10 dataset comprises 60,000 samples, divided into 10 categories with 6,000 samples in each category. Each group has 5,000 training samples and 1,000 testing samples. The images in the dataset are colored and have a resolution of  $32 \times 32$  pixels. On the other hand, the CIFAR-100 dataset is

<sup>1</sup>Implemented by [PyTorch Image Models](#)

an extension of the CIFAR-10 dataset, designed to provide a more challenging and diverse benchmark for image recognition algorithms. It contains 100 classes for classification, encompassing a broader range of objects and concepts than the CIFAR-10 dataset’s limited set of 10 classes.

We finetune the SpikingResformer-Ti and SpikingResformer-S pretrained in ImageNet classification on these datasets. We first replace the 1000-FC classifier layer with a randomly initialized 10-FC (CIFAR10) or 100-FC (CIFAR100) layer. We finetune the model for 100 epochs with an initial learning rate of  $1 \times 10^{-4}$  and cosine-decay to  $1 \times 10^{-5}$ . The batch size is set to 128. We employ data augmentation methods including random augmentation, mixup, and label smoothing. We use the AdamW optimizer with a weight decay of 0.01.

**Transfer Learning on Neuromorphic Datasets.** We also perform transfer learning experiments on neuromorphic dataset CIFAR10-DVS [23] and DVSGesture [1]. The CIFAR10-DVS dataset [23] is created by converting the static images in CIFAR10. This is done by moving the images and capturing the movement using a dynamic vision sensor. The CIFAR10-DVS dataset consists of 10,000 samples, with 1,000 samples per category. Each sample is an event stream with a spatial size of  $128 \times 128$ . It is worth noting that the CIFAR10-DVS dataset does not have predefined training and test sets. In our experiments, we select the first 900 samples of each category for training and the last 100 for testing.

The DVSGesture [1] dataset is created by directly capturing the human gestures using the DVS128 dynamic vision sensor. It has 1,342 instances of 11 hand and arm gestures. These gestures were grouped in 122 trials, performed by 29 subjects under 3 different lighting conditions. The dataset includes hand waving, arm rotations, air guitar, etc.

We use the following preprocessing procedure. Firstly, we divide the event stream into ten slices, each of which contains an equal number of events. Next, for each slice, we stack the events into a single frame consisting of three channels. These channels represent positive events, negative events, and all events. Finally, we use this frame as the input for that particular time step. In this way, we use a time step of 10 for these datasets.

We use the data augmentation technique proposed in [26]. Other settings follow the experiments of transfer learning on static image datasets.

## F. Further Comparison with ANN Version

We compare SpikingResformer with its ANN version, called Resformer in the following. To construct the ANN version of SpikingResformer, we replace the spiking neurons in SpikingResformer with ReLU activation, and replace the neurons in the attention map with Softmax function. For a fair comparison, other modules and the over-

Table S2. Further comparison with ANN version SpikingResformer. † means the input is enlarged to  $288 \times 288$  in inference.

Model	T	Param (M)	OPs (G)	Energy (mJ)	Top-1 Acc. (%)
Resformer-Ti	1	11.14	4.07	18.72	78.37
SpikingResformer-Ti	4	11.14	2.73/4.71 <sup>†</sup>	2.46/4.24 <sup>†</sup>	74.34/75.57 <sup>†</sup>
SpikingResformer-S	4	17.76	3.74/6.40 <sup>†</sup>	3.37/5.76 <sup>†</sup>	75.95/76.90 <sup>†</sup>
SpikingResformer-M	4	35.52	6.07/10.24 <sup>†</sup>	5.46/9.22 <sup>†</sup>	77.24/78.06 <sup>†</sup>
SpikingResformer-L	4	60.38	9.74/16.40 <sup>†</sup>	8.76/14.76 <sup>†</sup>	78.77/79.40 <sup>†</sup>

all architecture remain unchanged. In addition, the experimental setup of Resformer is the same as SpikingResformer. As shown in Tab. S2, the Resformer-Ti achieves higher accuracy (78.37% vs. 74.34%) but consumes significantly more energy (18.72mJ vs. 2.46mJ), even surpassing the energy consumption of SpikingResformer-L (8.76mJ, 78.77% acc.), demonstrating the energy efficiency advantage of SNNs.

## G. Further Comparison of Direct Training Results

In the main text, we perform only transfer learning experiments on small-scale datasets including CIFAR10, CIFAR100, CIFAR10-DVS, and DVSGesture. In this section, we perform direct training experiments on these datasets. To adapt to the small-size inputs, we replace the  $7 \times 7$  convolution with a  $3 \times 3$  convolution and remove the  $3 \times 3$  max pooling. Other modules remain unchanged. In addition, the minimum SpikingResformer-Ti is still too large for the DVSGesture dataset. Therefore, we halved the number of channels in each layer to avoid overfitting, thus constructing SpikingResformer-XTi. As shown in Tab. S3 and Tab. S4, our method is consistently effective when directly trained on small-scale datasets. For instance, SpikingResformer-Ti achieves 96.24% accuracy when directly trained on CIFAR10 dataset and achieves 79.28% accuracy on CIFAR100 dataset, outperforming state-of-the-art method Spikingformer-4-384 by 0.63% on CIFAR10 dataset and 0.19% on CIFAR100 dataset. For neuromorphic datasets, SpikingResformer-Ti achieves 81.5% accuracy on CIFAR10-DVS dataset, outperforming Spikingformer-2-256 by 0.2%. SpikingResformer-XTi achieves 98.6% accuracy on DVSGesture dataset, outperforming Spikingformer-2-256 by 0.3%, and competitive with 99.3% accuracy of Spike-driven Transformer-2-256.

## H. Detailed Comparison of Transfer Learning Results

**Static Image Datasets.** As shown in Tab. S3, SpikingResformer outperforms other transfer learning methods on CIFAR10 and CIFAR100 datasets with fewer parameters. The SpikingResformer-Ti achieves 84.53% accu-

Table S3. Detailed comparison on static datasets.

Method	Type	Architecture	#Param (M)	T	Top-1 Acc. (%)	
					CIFAR10	CIFAR100
STBP-tdBN [44]	Direct Training	ResNet-19	12.54	2	92.34	-
				4	92.92	-
				6	93.16	-
PLIF [10]	Direct Training	6 Conv, 2 FC	36.71	8	93.50	-
Dspike [25]	Direct Training	ResNet-18	11.21	2	93.13	71.68
				4	93.66	73.35
				6	94.25	74.24
Spikformer [46]	Direct Training	Spikformer-4-256	4.13	4	93.94	75.96
		Spikformer-2-384	5.74	4	94.80	76.95
		Spikformer-4-384	9.28	4	95.19	77.86
Spikingformer [45]	Direct Training	Spikingformer-4-256	4.13	4	94.77	77.43
		Spikingformer-2-384	5.74	4	95.22	78.34
		Spikingformer-4-384	9.28	4	95.61	79.09
Spike-driven Transformer [42]	Direct Training	Spike-driven Transformer-2-512	10.21	4	95.6	78.4
<b>SpikingResformer (Ours)</b>	Direct Training	SpikingResformer-Ti*	10.79	4	96.24	79.28
Spikformer [46]	Transfer Learning	Spikformer-4-384	9.28	4	95.54	79.96
		Spikformer-8-384	16.36	4	96.64	82.09
		Spikformer-8-512	29.08	4	97.03	83.83
<b>SpikingResformer (Ours)</b>	Transfer Learning	SpikingResformer-Ti	10.76	4	97.02	84.53
		SpikingResformer-S	17.25	4	<b>97.40</b>	<b>85.98</b>

\* The stem structure differs from the original SpikingResformer-Ti in the main text.

Table S4. Detailed comparison on neuromorphic datasets.

Method	Type	Architecture	#Param (M)	T	Top-1 Acc. (%)	
					CIFAR10-DVS	DVSGesture
STBP-tdBN [44]	Direct Training	ResNet-19	12.54	10	67.8	-
		ResNet-17	1.40	40	-	96.87
PLIF [10]	Direct Training	5 Conv, 2 FC	17.22	20	74.8	-
		6 Conv, 2 FC	1.69	20	-	97.57
Dspike [25]	Direct Training	ResNet-18	11.21	10	75.4	-
Spikformer [46]	Direct Training	Spikformer-2-256	2.55	10	78.6	95.8
				16	80.6	97.9
Spikingformer [45]	Direct Training	Spikingformer-2-256	2.55	10	79.9	96.2
				16	81.3	98.3
Spike-driven Transformer [42]	Direct Training	Spike-driven Transformer-2-256	2.55	16	80.0	<b>99.3</b>
<b>SpikingResformer (Ours)</b>	Direct Training	SpikingResformer-Ti*	10.79	10	81.5	-
		SpikingResformer-XTi*	2.71	16	-	98.6
<b>SpikingResformer (Ours)</b>	Transfer Learning	SpikingResformer-Ti	10.76	10	84.7	93.4
		SpikingResformer-S	17.25	10	<b>84.8</b>	93.4

\* The stem structure differs from the original SpikingResformer-Ti in the main text.

racy on the CIFAR100 dataset, outperforming Spikformer-8-512 by 0.7% with only 11.14M parameters. Moreover, the SpikingResformer-S achieves 85.98% accuracy on the CIFAR100 dataset, which is the state-of-the-art result and outperforms Spikformer-8-512 by 2.15%. Compared to direct training methods, the SpikingResformer ob-

tained from transfer learning has significantly higher performance. The SpikingResformer-Ti outperforms the Spiking Transformer-2-512 by 6.1% with a comparable number of parameters. This demonstrates the advantage of transfer learning.

**Neuromorphic Datasets.** Since the existing spiking vi-



sion transformer does not perform transfer learning experiments on neuromorphic datasets, we mainly compare with direct training methods. However, since the size of the models trained directly on the CIFAR10-DVS and DVSGesture is typically much smaller than the models pre-trained on ImageNet, we are not able to compare them to models with comparable parameters. As shown in Tab. S4, the SpikingResformer obtained from transfer learning has significantly higher performance on CIFAR10-DVS. The SpikingResformer-Ti achieves 84.7% accuracy on CIFAR10-DVS, outperforming Spiking Transformer-2-256 by 4.7% and outperforming Spikingformer-2-256 by 3.4%. However, the transfer learning results on DVSGesture fail to achieve comparable performance to direct training. SpikingResformer only achieves 93.4% accuracy on DVSGesture, falling behind the state-of-the-art method Spike-driven Transformer by 5.9%. We believe that this is mainly due to the way CIFAR10-DVS is constructed differs from DVSGesture. CIFAR10-DVS is converted from CIFAR10 using a dynamic vision sensor, which does not contain temporal information. Thus, models pre-trained on static datasets can transfer to CIFAR10-DVS well. However, DVSGesture is directly created from human gestures, which contain rich temporal information. As a result, models pre-trained on static datasets do not transfer well to DVSGesture.