# 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*

$$\mathrm{E}(ab) = \mathrm{E}(a)\mathrm{E}(b), \tag{S1}$$

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

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

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

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

$$\mathrm{E}(ab) = \mathrm{E}(a)\mathrm{E}(b) + 0 = \mathrm{E}(a)\mathrm{E}(b). \tag{S4}$$

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

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

□

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

$$\mathrm{E}(\sum_{i=1}^{n} a_i) = \sum_{i=1}^{n} \mathrm{E}(a_i), \tag{S6}$$

$$\mathrm{Var}(\sum_{i=1}^{n} a_i) = \sum_{i=1}^{n} \mathrm{Var}(a_i). \tag{S7}$$

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

$$\mathrm{E}(a_i + a_j) = \mathrm{E}(a_i) + \mathrm{E}(a_j), \tag{S8}$$

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

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

$$\mathrm{E}(\sum_{i=1}^{n} a_i) = \sum_{i=1}^{n} \mathrm{E}(a_i), \tag{S10}$$

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

□

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

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

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

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 $\mathrm{E}(x_{i,k}[t]) = f_x$ and $\mathrm{Var}(x_{i,k}[t]) = f_x(1 - f_x)$. According to Lemma 1, we have

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

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

In addition, each $z_{j,k}[t], k = 1, \ldots, q$ consists of a set of $y_{l,k}[t], l = 1, \ldots, q$ 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

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

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

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

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

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

**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 [11]. The SSA can be formulated as follows:

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

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

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

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

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 $\mathbf{I}=\mathbf{Q}\mathbf{K}^{\mathrm{T}}\mathbf{V}$, each element in $\mathbf{I}$ can be formulated as:

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

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 $\mathrm{E}(I_{i,j}[t])=HWdf_Qf_Kf_V$.

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:

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

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 [8]. The SDSA can be formulated as follows:

$$\mathrm{SDSA}(\mathbf{Q},\mathbf{K},\mathbf{V})=\mathrm{SN}(\mathrm{SUM_c}(\mathbf{Q}\otimes\mathbf{K}))\otimes\mathbf{V}, \tag{S24}$$

where $\otimes$ denotes Hadamard product, $\mathrm{SUM_c}$ 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:

$$\mathrm{SDSA}(\mathbf{Q},\mathbf{K},\mathbf{V})=\mathrm{SN}(\mathrm{SUM_c}(\mathbf{Q}\otimes\mathbf{K})*c)\otimes\mathbf{V}, \tag{S25}$$

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 $\mathbf{I} = \mathrm{SUM_c}(\mathbf{Q} \otimes \mathbf{K})$, each element in $\mathbf{I}$ can be formulated as:

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

Following the same assumption in SSA, we have

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

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 2×2 convolution with a stride of 2 on a 4×4 input is equivalent to a linear transformation.

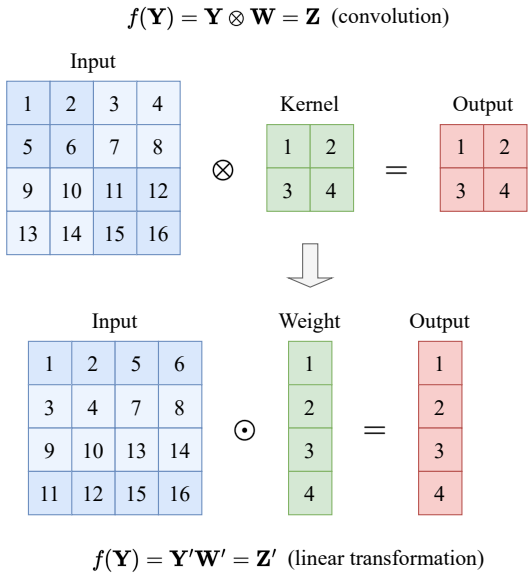


Figure S1. Diagram of the equivalence of convolution to linear transformation. **Top**: $\mathrm{Conv_p}(\cdot)$ on a 4×4 input where $p = 2$; **Bottom**: Its equivalent linear transformation.

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:

$$\mathbf{Z} = \mathbf{Y} \otimes \mathbf{W}, \tag{S28}$$

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

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

$$\begin{aligned}&\mathbf{Y}'=g_Y(\mathbf{Y}),\\ &s.t.\ y'_{i*W_{out}+j,c_{in}*hw+k*w+l}=y_{(i-1)*p+k,(j-1)*p+l,c_{in}},\end{aligned} \tag{S30}$$

$$\begin{aligned}&\mathbf{W}'=g_W(\mathbf{W}),\\ &s.t.\ w'_{c_{in}*hw+k*w+l,c_{out}}=w_{k,l,c_{out},c_{in}}.\end{aligned} \tag{S31}$$

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

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

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

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:

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

$$\mathrm{AttnMap}(\mathbf{X})=\mathrm{SN}(\mathrm{DST}_{\mathrm{T}}(\mathbf{X},\mathbf{X};f(\cdot))*c_1), \tag{S35}$$

$$f(\mathbf{X})=\mathrm{BN}(\mathrm{Conv}_{\mathrm{p}}(\mathbf{X})). \tag{S36}$$

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

$$\mathrm{DST}(\mathbf{X},\mathbf{Y};f(\cdot))=\mathbf{X}f(\mathbf{Y})=\mathbf{XYW}, \tag{S37}$$

$$\mathrm{DST}_{\mathrm{T}}(\mathbf{X},\mathbf{Y};f(\cdot))=\mathbf{X}f(\mathbf{Y})^{\mathrm{T}}=\mathbf{X}\mathbf{W}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}. \tag{S38}$$

DSSA achieves self-attention by the two DSTs. The first one, $\mathrm{DST}_{\mathrm{T}}(\mathbf{X},\mathbf{X};f(\cdot))$, 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 $\mathrm{DST}_{\mathrm{T}}(\mathbf{X},\mathbf{X};f(\cdot))$ 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, $\mathrm{DST}(\mathrm{AttnMap}(\mathbf{X}),\mathbf{X};f(\cdot))$, 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 $\mathbf{QK}^{\mathrm{T}}$ 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 [2] 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 [8]. We use the standard preprocessing, i.e., data normalization, randomly crop and resize the input to 224×224 during traning, and set the input size to 224×224 and 288×288 for inferince. We employ the standard data augmentation methods including random augmentation, mixup, cutmix, and label smoothing[1], similar to [8]. 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 $f_{Attn}$, 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 [4]. 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 32×32 pixels. On the other hand, the CIFAR-100 dataset is

[1] Implemented by PyTorch Image Models

Table S2. Detailed comparison on static datasets.

| Method | Type | Architecture | #Param (M) | T | Top-1 Acc. (%) | |
|---|---|---|---|---|---|---|
| | | | | | CIFAR10 | CIFAR100 |
| STBP-tdBN [9] | Direct Training | ResNet-19 | 12.54 | 2 | 92.34 | - |
| | | | | 4 | 92.92 | - |
| | | | | 6 | 93.16 | - |
| PLIF [3] | Direct Training | 6 Conv, 2 FC | 36.71 | 8 | 93.50 | - |
| Dspike [6] | Direct Training | ResNet-18 | 11.21 | 2 | 93.13 | 71.68 |
| | | | | 4 | 93.66 | 73.35 |
| | | | | 6 | 94.25 | 74.24 |
| Spikformer [11] | 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 [10] | 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 [8] | Direct Training | Spike-driven Transformer-2-512 | 10.21 | 4 | 95.6 | 78.4 |
| Spikformer [11] | 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 |
| **SpikingResformer (Ours)** | Transfer Learning | SpikingResformer-Ti | 10.76 | 4 | 97.02 | 84.53 |
| | | SpikingResformer-S | 17.25 | 4 | **97.40** | **85.98** |

Table S3. Detailed comparison on neuromorphic datasets.

| Method | Type | Archtecture | #Param (M) | T | Top-1 Acc. (%) | |
|---|---|---|---|---|---|---|
| | | | | | CIFAR10-DVS | DVSGesture |
| STBP-tdBN [9] | Direct Training | ResNet-19 | 12.54 | 10 | 67.8 | - |
| | | ResNet-17 | 1.40 | 40 | - | 96.87 |
| PLIF [3] | Direct Training | 5 Conv, 2 FC | 17.22 | 20 | 74.8 | - |
| | | 6 Conv, 2 FC | 1.69 | 20 | - | 97.57 |
| Dspike [6] | Direct Training | ResNet-18 | 11.21 | 10 | 75.4 | - |
| Spikformer [11] | Direct Training | Spikformer-2-256 | 2.55 | 10 | 78.6 | 95.8 |
| | | | | 16 | 80.6 | 97.9 |
| Spikingformer [10] | Direct Training | Spikingformer-2-256 | 2.55 | 10 | 79.9 | 96.2 |
| | | | | 16 | 81.3 | 98.3 |
| Spike-driven Transformer [8] | Direct Training | Spike-driven Transformer-2-256 | 2.55 | 16 | 80.0 | **99.3** |
| **SpikingResformer (Ours)** | Transfer Learning | SpikingResformer-Ti | 10.76 | 10 | 84.7 | 93.4 |
| | | SpikingResformer-S | 17.25 | 10 | **84.8** | 93.4 |

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 [5] and DVSGesture [1]. The CIFAR10-DVS dataset [5] 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 sam-

ples, with 1,000 samples per category. Each sample is an event stream with a spatial size of 128×128. 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 [7]. Other settings follow the experiments of transfer learning on static image datasets.

## F. Detailed Comparison of Transfer Learning Results

**Static Image Datasets.** As shown in Tab. S2, SpikingResformer outperforms other transfer learning methods on CIFAR10 and CIFAR100 datasets with fewer parameters. The SpikingResformer-Ti achieves 84.53% accuracy 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 obtained 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 vision 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. S3, 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.

## 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. 5, 6

[2] 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. 4

[3] 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. 5

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[5] 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. 5

[6] 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. 5

[7] Yuhang Li, Youngeun Kim, Hyoungseob 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. 6

[8] 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

[9] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural net-

works. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11062–11070, 2021. 5

[10] Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Han Zhang, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*, pages 1–16, 2023. 5

[11] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, 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