Spatial-Temporal Self-Attention for Asynchronous Spiking Neural Networks

Yuchen Wang, **Kexin Shi**, **Chengzhuo Lu**, **Yuguo Liu**, **Malu Zhang*** and **Hong Qu**
School of Computer Science and Engineering,
University of Electronic Science and Technology of China
yuchenwang@std.uestc.edu.cn, kexinshi@std.uestc.edu.cn, 2019270101012@std.uestc.edu.cn,
liuyuguo@std.uestc.edu.cn, maluzhang@uestc.edu.cn, hongqu@uestc.edu.cn

Abstract

The brain-inspired spiking neural networks (SNNs) are receiving increasing attention due to their asynchronous event-driven characteristics and low power consumption. As attention mechanisms recently become an indispensable part of sequence dependence modeling, the combination of SNNs and attention mechanisms holds great potential for energy-efficient and high-performance computing paradigms. However, the existing works cannot benefit from both temporal-wise attention and the asynchronous characteristic of SNNs. To fully leverage the advantages of both SNNs and attention mechanisms, we propose an SNNs-based spatial-temporal self-attention (STSA) mechanism, which calculates the feature dependence across the time and space domains without destroying the asynchronous transmission properties of SNNs. To further improve the performance, we also propose a spatial-temporal relative position bias (STRPB) for STSA to consider the spatiotemporal position of spikes. Based on the STSA and STRPB, we construct a spatial-temporal spiking Transformer framework, named STS-Transformer, which is powerful and enables SNNs to work in an asynchronous event-driven manner. Extensive experiments are conducted on popular neuromorphic datasets and speech datasets, including DVS128 Gesture, CIFAR10-DVS, and Google Speech Commands, and our experimental results can outperform other state-of-the-art models.

1 Introduction

The spiking neural networks (SNNs) are regarded as the third generation of neural networks [Maass, 1997]. Over the past decade, SNNs have been getting rising attention because they can mimic the dynamics of biological neurons from a microscopic perspective. Different from the conventional artificial neural networks (ANNs), the neuron in SNNs will emit a spike when accumulated membrane potential exceeds a firing threshold, and the information transmission in SNNs totally relies on discrete, sparse, and asynchronous spikes. Compared with ANNs, SNNs can provide low latency and low power alternatives for two reasons. Firstly, because the spike is binary information, it can use accumulate (AC) operations for calculation during forward propagation, avoiding energy-hungry multiply-accumulate (MAC) operations. The second is its capability of event-driven calculation, i.e., the membrane potential is only updated when spikes arrive, avoiding the calculation of useless zero values. The low power consumption and event-driven computing superiority of SNNs make them more suitable for deployment on emerging neuromorphic chips, such as TrueNorth [Merolla *et al.*, 2014], Loihi [Davies *et al.*, 2018], or Tianjic [Pei *et al.*, 2019].

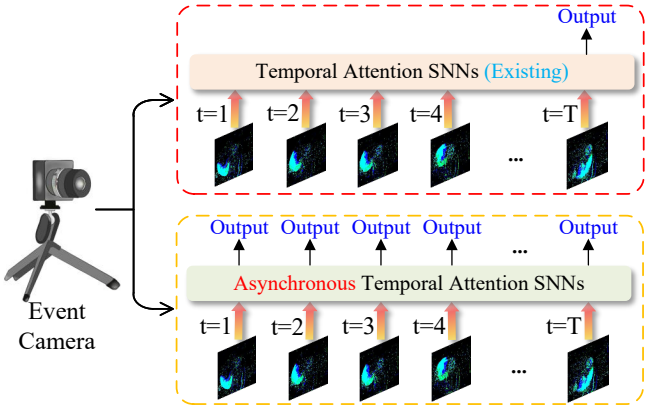


Figure 1: The existing temporal attention SNNs needs to wait for a data time duration T before getting the output, which destroys the SNN asynchronous computing capability. The asynchronous temporal attention SNNs can get output at any time step.

Currently, the most popular SNN learning algorithms are unsupervised spike-timing-dependent plasticity (STDP) learning [Diehl and Cook, 2015; Kheradpisheh *et al.*, 2018], ANN-SNN conversion [Diehl *et al.*, 2015; Rueckauer *et al.*, 2017; Wang *et al.*, 2022] and supervised surrogate gradient learning [Wu *et al.*, 2018; Neftci *et al.*, 2019]. Although STDP is an algorithm inspired by synaptic plasticity in biological neural networks, it is poor at complex datasets and restricted to shallow SNNs. ANN-SNN methods convert a pre-trained ANN to its SNN counterpart, while the limited-time steps will result in an evident accuracy drop. The sur-

*Contact Author

rogate gradient learning replaces the non-differentiable part during gradient backpropagation so that the mature backpropagation through time (BPTT) algorithm can be used to train SNNs, which makes it possible to inspire the design of SNN models with the latest technology of ANNs. Some conspicuous backbone networks, such as ResNet [He *et al.*, 2016], YOLO [Redmon and Farhadi, 2018], or graph neural network [Defferrard *et al.*, 2016], have inspired the structure of SNNs and achieved remarkable results [Fang *et al.*, 2021a; Chakraborty *et al.*, 2021; Xu *et al.*, 2021].

Attention mechanisms have recently become an indispensable part of sequence modeling in various tasks, allowing the modeling of dependencies without regard to their distance in the sequences to achieve high performance. There have been a lot of works trying to combine the attention mechanism with SNNs, which can be divided into (1) spatial attention for SNNs [Zhou *et al.*, 2022] and (2) temporal attention for SNNs [Yao *et al.*, 2021; Zhu *et al.*, 2022; Yao *et al.*, 2022]. The former models all pairwise interactions between all patches in a single time step, ignoring the dependence of information on different time steps of the spike train. The latter focuses on using the attention mechanism to capture feature dependencies at different times. This type of work calculates the attention score in consideration of the information from all time steps, and this straightforward combination destroys the SNN asynchronous computing capability, as shown in Fig. 1. This approach prevents SNN from giving real-time prediction because it must wait T time steps before giving a prediction, where T is the time duration of a single data.

In this work, we try to take full advantage of both asynchronous SNNs and attention mechanisms. Unlike the existing attention for SNNs that need to make a trade between temporal-wise attention and asynchronous event-driven computing, we propose an SNN-based spatial-temporal self-attention (STSA) that can benefit from both of them. Subsequently, because the pure attention mechanism is unable to capture the sequence order, we further propose a spatial-temporal relative position bias (STRPB) for STSA to improve the performance. Finally, we deliver an effective and efficient SNN-based Transformer framework based on the above and conduct experiments on challenging stream datasets. The contributions of this work are summarized as follows:

- We propose a spatial-temporal self-attention (STSA) mechanism for SNNs to capture feature dependence from both the time domain and space domain, while still maintaining the asynchronous information transmission capability of SNNs.
- To further improve the performance, we propose a spatial-temporal relative position bias (STRPB) to infuse the spatiotemporal position of spikes into STSA.
- We develop a powerful spatial-temporal spiking transformer (STS-Transformer) framework based on STSA and STRPB. To verify the performance of the STS-Transformer, we conduct experiments on popular steam datasets and achieve state-of-the-art performance. Codes are available at https://github.com/ppppps/STSA_4_Asyn_SNN.

2 Related Work

2.1 Supervised Learning of Deep SNNs

In recent years, a lot of learning methods and deep SNN models have been developed with excellent performance [Tavanaei *et al.*, 2019; Zhang *et al.*, 2022; Luo *et al.*, 2022; Zhang *et al.*, 2021]. Wu *et al.* [2018] first proposed a spatio-temporal backpropagation (STBP) algorithm to train high-performance SNNs. Zheng *et al.* [2021] further proposed the threshold-dependent batch normalization (tdBN) method, extending the network from a shallow structure (< 10 layers) to a very deep structure (50 layers) for the first time. However, different from the conventional deep networks, the deepening of the SNN causes severe performance degradation. Fang *et al.* [2021a] analyzed this problem and proposed a SEW-ResNet, so that the SNNs can still enjoy the benefits of increasing the number of network layers. Hu *et al.* [2021] also noticed this degradation problem and proposed several alternative residual blocks. Beyond the structure of the network, researchers have also tried to further improve the performance of deep SNNs from other aspects. Fang *et al.* [2021b] proposed a Parametric Leaky Integrate and Fire (PLIF) neuron that makes deep SNNs more robust to the initial values and can converge faster than SNNs made of ordinary LIF neurons. In order to solve the problem of momentum loss in SNN during gradient descent, Deng *et al.* [2022] proposed the TET-Loss that makes the training process converge into flatter minima, resulting in a more generalized model. Although there have been many works on deep SNNs, the basic network module that can bring SNNs' superiority into full play is still ongoing research.

2.2 Attention Spiking Neural Networks

Since attention-based deep neural networks predominate in various tasks [Devlin *et al.*, 2018; Liu *et al.*, 2021], researchers have begun to replicate this successful experience in the SNN field. Inspired by the SE-Net [Hu *et al.*, 2018], Yao *et al.* [2021] put the attention mechanism into SNNs for the first time to filter out the irrelevant spike trains for the decision. However, this model can only identify the importance of information in the time domain, and all information at the same time step will get the same attention weight. Similarly, Zhou *et al.* [2022] proposed a spiking neural network with self-attention only in the spatial domain, ignoring the information interaction at different time steps. Zhu *et al.* [2022] further proposed a temporal and channel joint attention mechanism for SNNs and achieved good results on neuromorphic datasets, while SNNs lost the capability of asynchronous reasoning. Furthermore, Yao *et al.* [2022] proposed multi-dimensional attention for SNNs and surpassed the model focused on single-dimensional information. Although a lot of attention SNNs have been carried out, how to embrace both the asynchronous transmission of SNN and temporal-wise attention still requires further endeavor.

3 Preliminaries

3.1 Neuromorphic Vision Data

A piece of data containing N events collected by an event camera is often composed of a set $E = \{\mathbf{e}_i \mid i =$

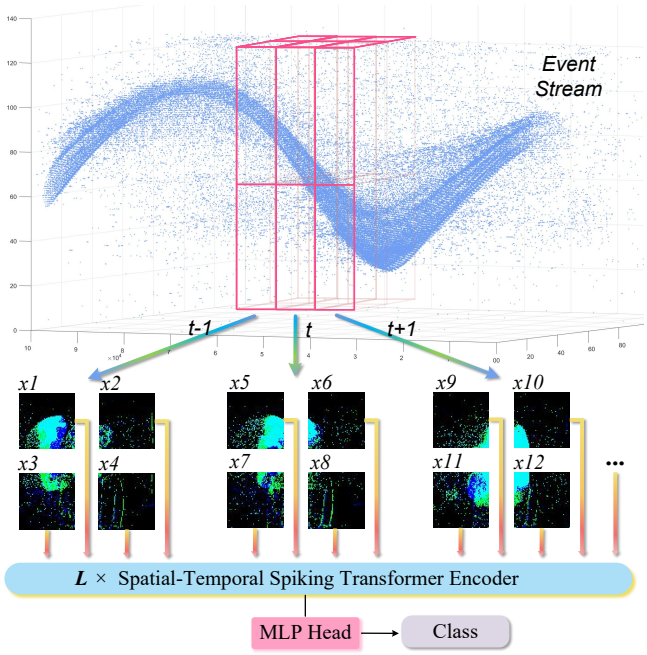


Figure 2: The overall architecture of our proposed STS-Transformer. This example shows three frames at time $t-1$, t, and $t+1$, and each frame is spatially divided into 4 patches.

$0, 1, 2, ..., N\}$, and a single event $\mathbf{e}_i \in \mathbb{R}^{4\times1}$ is expressed as:

$$\mathbf{e}_i = [x_i, y_i, t_i, p_i], \tag{1}$$

where x_i, y_i represent the spatial coordinates, t_i is timestamp and p_i is the polarity of the event $\mathbf{e}_i$. The polarity p_i has only two values of $+1/-1$, respectively representing whether the event is caused by increasing or decreasing brightness.

We use the frame-based method to preprocess the event data, which transforms event streams to frame streams by aggregating events. Firstly, we set the length of the frame sequence as T, then divide the event stream data into T segments and aggregate the data to a frame in each segment. Particularly, the t-th frame $\mathbf{f}(t) \in \mathbb{R}^{2\times w\times h}$ can add 1 to the corresponding position according to the p_j, x_j and y_j of the event $\mathbf{e}_j$, where $j \in [\lfloor \frac{N}{T} \rfloor t, \lfloor \frac{N}{T} \rfloor (t+1)]$, w and h are the maximum width and height of event data respectively.

3.2 Spiking Neural Networks

Different from conventional neural networks, spiking neural networks simulate biological neural networks from a microscopic view. The Leaky Integrate and Fire (LIF) neuron is the most commonly used neuron in deep spiking neural networks since it has a simple mathematical form and can express the dynamic characteristics of biological neurons. The membrane potential $v_j^l(t)$ of LIF neuron j in layer l at time step t described by:

$$x_j^l(t) = \sum_i w_{ij} s_i^{l-1}(t) + b_j, \tag{2}$$

$$\tilde{v}_j^l(t) = \tau v_j^l(t-1) + x_j^l(t), \tag{3}$$

$$s_j^l(t) = g(\tilde{v}_j^l(t) - \theta), \tag{4}$$

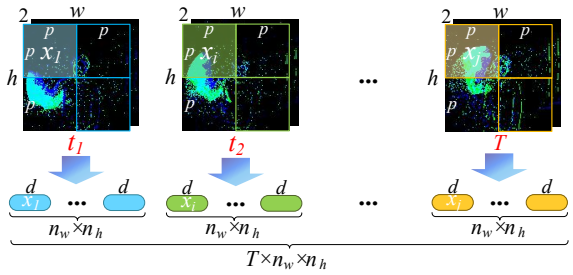


Figure 3: The tokenization method. There are T frames with a dimension of $2\times w\times h$ in the sequence obtained from the event stream, we map each patch of size $2\times p\times p$ to a vector of length d by convolutional stem. So a frame will convert to $n_w \times n_h$ vectors of length d, where n_w is $\lfloor \frac{w}{p} \rfloor$ and n_h is $\lfloor \frac{h}{p} \rfloor$.

$$v_j^l(t) = (1 - s_j^l(t))\tilde{v}_j^l(t) + s_j^l(t) v_{\text{rest}}, \tag{5}$$

where s_i^{l-1} is the spike from presynaptic neuron i and v_{rest} donates the resting potential. τ is a hyperparameter within $(0, 1)$ used to control membrane potential decay. When $\tilde{v}_j^l(t)$ exceeds firing threshold θ, neuron j will send a spike $s_j^l(t)$ to its postsynaptic neurons. After firing a spike, the membrane potential $v_j^l(t)$ will immediately return to resting potential.

During the forward propagation process, $g(x)$ in (4) is a heaviside step function, determined as:

$$g(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}, \tag{6}$$

which derivative $g'(x)$ is the Dirac function $\delta(x)$, given by:

$$\delta(x) = \begin{cases} \infty, & x = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

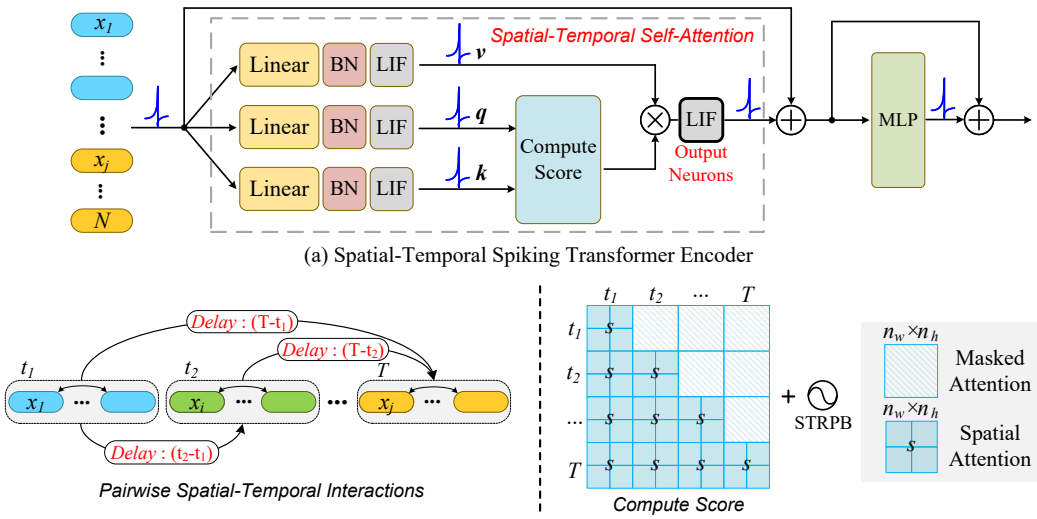
The integral of $\delta(x)$ is 1, but it cannot be used in backpropagation. So it is usually to use a function with an integral of 1 but a wider distribution, such as a rectangular, triangular, or sigmoid-shaped function surrogate $g'(x)$. There is currently no evidence to prove which surrogate function is optimal, and a simple triangular surrogate function is chosen in our work.

4 Method

In this section, we first introduce the overall architecture of the STS-Transformer. Secondly, we introduce the tokenization method of the STS-Transformer. Then the details about how STSA works are presented. Finally, we illustrate the proposed STRPB.

4.1 Overall Architecture

The overall architecture of the STS-Transformer is shown in Fig. 2. For a given 3D event stream, we first artificially set a time window of size dt, and divide the event stream sequence into T segments of length dt. Then we aggregate events in each segment using the method in section 3.1, so that the data becomes a frame sequence $\mathbf{f} \in \mathbb{R}^{T\times2\times w\times h}$. Next, we divide each frame into patches of size p, and linearly map them into fixed-length tokens. The detailed tokenization method will be described in section 4.2. Next, the tokens will undergo information fusion through L STS-Transformer encoders with



(a) Spatial-Temporal Spiking Transformer Encoder

(b) Instruction of Spatial-Temporal Self-Attention for Asynchronous SNNs

Figure 4: (a) This figure shows the spatial-temporal spiking transformer encoder, while the proposed SNN-based STSA is located in the dotted box, and the neuron with a bold border is the output neuron of STSA. (b) The left half introduces the idea of STSA. And the right half introduces the parallel implementation of STSA, which slices the attention matrix $A \in \mathbb{R}^{M\times M}$ ($M = T \times n_w \times n_h$) into $T \times T$ regions according to the step size s ($s = n_w \times n_h$). The sliced block at position $[i, j]$ represents the similarity between tokens at different spatial positions of time i and time j.

SNNs-based STSA, which is the core innovation of our work. The calculation process of the encoder and the STSA mechanism will be introduced in detail in section 4.3. Finally, since we are verifying our model in the recognition task, an MLP is connected after the encoders to output the prediction result of the model.

4.2 Tokenization

We consider a straightforward method for mapping a frame sequence to a sequence of tokens $\mathbf{z} \in \mathbb{R}^{T\times n_w\times n_h\times d}$, where n_w equals to $\lfloor \frac{w}{p} \rfloor$, n_h equals to $\lfloor \frac{h}{p} \rfloor$, and p is patch size. Fig. 3 shows the specific tokenization method. For a single frame $\mathbf{f}(t)$, we regard it as 2D data for token generation, and finally aggregate the tokens of all time steps. Eventually, there will be $T \times n_w \times n_h$ tokens of length d entering the transformer encoder, and each transformer encoder models pairwise interactions between all spatial-temporal tokens. For the patchify stem that linearly maps the patch to the token, the method in Vision Transformer [Dosovitskiy *et al.*, 2020] can be regarded as a convolutional layer with a step size of 16 and a kernel size of 16. Since existing neuromorphic datasets are very small, in order to add the inductive bias of the model for the data, we replaced the patchify stem with a convolutional stem. The kernel size and step in the convolutional stem have been reduced, and the detailed settings will be introduced in the experimental section.

4.3 SNNs-Based Spatial-Temporal Self-Attention

The STSA is proposed to calculate the feature dependence of spike trains across the time and space domains without destroying the asynchronous transmission properties of SNNs. Together with an MLP block, the STSA constitutes the STS-Transformer encoder, shown in Fig. 4 (a). The STSA proposed in our work first generate spike trains $\mathbf{q}$, $\mathbf{k}$, $\mathbf{v}$, as:

$$\mathbf{q} = LIF(BN(\mathbf{z}^{l-1}W_Q)), \quad \mathbf{q} \in \mathbb{R}^{T\times n_w\times n_h\times d_q}, \tag{8}$$

$$\mathbf{k} = LIF(BN(\mathbf{z}^{l-1}W_K)), \quad \mathbf{k} \in \mathbb{R}^{T\times n_w\times n_h\times d_k}, \tag{9}$$

$$\mathbf{v} = LIF(BN(\mathbf{z}^{l-1}W_V)), \quad \mathbf{v} \in \mathbb{R}^{T\times n_w\times n_h\times d_v}. \tag{10}$$

Where BN represents the batch normalization operation, LIF represents the membrane potential accumulation and firing process of LIF neurons, i.e., Eq. (3) and Eq. (4). d_q, d_k and d_v are usually set to a same value.

After generating the $\mathbf{q}$, $\mathbf{k}$, and $\mathbf{v}$, STSA calculates the attention weights based on the pairwise similarity between the respective query $\mathbf{q}$ and key $\mathbf{k}$ of two elements of the sequence across space and time domains. The idea of the proposed STSA can be seen in the left half of Fig. 4 (b), each token can interact with all tokens at this moment and all tokens at the previous moment. We construct STSA by *synapses with delays* for asynchronous SNNs, the input $x_j(t)$ of the output neuron j ($j \in [0, n_w n_h - 1]$) of STSA (the neuron with a bold border in Fig. 4 (a)) can be written as follows:

$$x_j(t) = x_j^a + x_j^b + x_j^c + x_j^d. \tag{11}$$

The $x_j(t)$ consists of four parts: (a) input from the current moment at the current location, (b) input from the current moment at other locations, (c) input from the past moment at the current location, and (d) input from the past moment at other locations. We denote the input of these four parts as x_j^a, x_j^b, x_j^c, and x_j^d respectively, which are formulated by:

$$x_j^a = (\mathbf{q}_j(t) \cdot \mathbf{k}_j(t))\mathbf{v}_j(t), \tag{12}$$

$$x_j^b = \sum_{i=0}^{n_w n_h - 1} (\mathbf{q}_i(t) \cdot \mathbf{k}_j(t))\mathbf{v}_j(t), \quad i \neq j, \tag{13}$$

$$x_j^c = \sum_{\tau_d=0}^{t-1} (\mathbf{q}_j(t-\tau_d) \cdot \mathbf{k}_j(t))\mathbf{v}_j(t), \tag{14}$$

$$x_j^d = \sum_{\tau_d=0}^{t-1} \sum_{i=0}^{n_w n_h - 1} (\mathbf{q}_i(t-\tau_d) \cdot \mathbf{k}_j(t))\mathbf{v}_j(t), \quad i \neq j. \tag{15}$$

Where τ_d represents the synaptic delay, which is an integer in $[0, t-1]$. After receiving the input, the output neurons of the STSA will proceed to update the membrane potential through Eq. (3) $\sim$ Eq. (5).

To speed up the training process of the proposed STSA, we further develop *an equivalent parallel computation method*, which is shown in the right half of Fig. 4 (b). After calculating the attention weight matrix $\mathbf{A} \in \mathbb{R}^{M \times M}$ ($M = T \times n_w \times n_h$) by multiplying $\mathbf{q}$ and $\mathbf{k}^{\mathrm{T}}$, we slice the row and column of the matrix $\mathbf{A}$ $(T-1)$ times with step size s ($s = n_w \times n_h$), results in $\mathbf{A}$ becomes $T \times T$ blocks. Then we mask off the upper triangular blocks of the sliced $\mathbf{A}$, which corresponds to the shaded part in the figure, so that the weight of $\mathbf{v}$ obtained from future information will always be 0. There are two reasons why this is equivalent to the synaptic delay calculation method. The first point is that we will not perform the softmax operation after obtaining $\mathbf{q}\mathbf{k}^{\mathrm{T}}$, so intercepting part of the information will not change the attention score. The second point is that the information at different time steps is arranged chronologically in the $\mathbf{A}$ matrix. The block with index $[i, j]$ of the sliced matrix $\mathbf{A}$ indicates that tokens at time step i is used to query the similarity of tokens at time step j, so covering the upper triangular matrix can make earlier spikes invisible later spikes, and the lower triangular matrix allows later spikes can see earlier spikes.

The masked matrix is not the final weight score, we then add our proposed spatial-temporal relative position bias (STRPB) $\mathbf{b}$ to the weight, which will be introduced in detail in section 4.4. Therefore, the remaining calculation of STS-Transformer encoder can be expressed as follows:

$$\mathbf{A} = mask(\mathbf{q}\mathbf{k}^{\mathrm{T}}/c) + \mathbf{b}. \tag{16}$$

$$STSA(\mathbf{z}^{l-1}) = \mathbf{A}\mathbf{v}. \tag{17}$$

$$\tilde{\mathbf{z}}^l = LIF(STSA(\mathbf{z}^{l-1})) + \mathbf{z}^{l-1}. \tag{18}$$

$$\mathbf{z}^l = MLP(\tilde{\mathbf{z}}^l) + \tilde{\mathbf{z}}^l. \tag{19}$$

c is the scaling factor of attention weight, which is set to a fixed value.

4.4 Relative Position Bias Across Space and Time Domain

Because the attention mechanisms cannot directly obtain the location information between tokens, many previous works have shown that it can be helpful to add a relative position bias in self-attention computation.

In order to incorporate the spatiotemporal location information of spikes in STSA, we propose a spatial-temporal relative position bias (STRPB) $\mathbf{b} \in \mathbb{Z}^{M \times M}$, where M is also equal to $T \times n_w \times n_h$. We use the coordinate $[t, x, y]$ of the patch to represent its position, the coordinate of the starting position is $[0, 0, 0]$, and the coordinate of the final position is $[T, n_w, n_h]$. Therefore, there are $2T-1$ relative positions of tokens in the time domain, and the range is from $-T+1$ to $T-1$. Similarly, there are $(2n_w - 1) \times (2n_h - 1)$ relative positions in space domain. and the range is from $[-2n_w + 1, -2n_h + 1]$ to $[2n_w - 1, 2n_h - 1]$. Therefore, we can create a spatiotemporal relative position table $\mathbf{o} \in \mathbb{Z}^{(2T-1)\times(2n_w-1)\times(2n_h-1)}$, and according to the position of tokens, fill the value in $\mathbf{o}$ into $\mathbf{b}$.

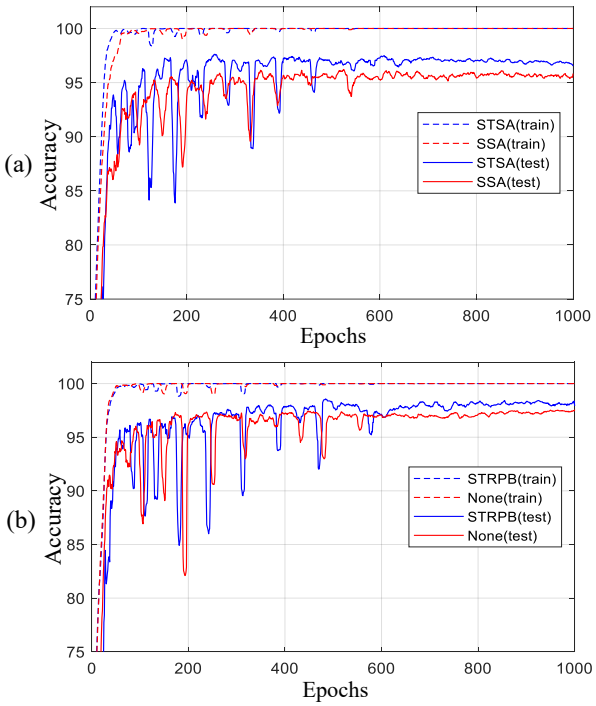


Figure 5: The results of ablation experiments.

5 Experiments

In order to verify the effectiveness of the proposed method, we conduct object recognition experiments on neuromorphic vision datasets DVS128 Gesture [Amir *et al.*, 2017] and CIFAR10-DVS [Li *et al.*, 2017], and speech recognition experiments on Google Speech Commands V1 and Google Speech Commands V2 [Warden, 2018]. We first ablate the STSA and STRPB in section 5.2. Then, we compare the proposed STS-Transformer with the other state-of-the-art models on the above four datasets in section 5.3.

5.1 Experimental Setup

Datasets

The IBM DVS128 Gesture dataset is captured by DVS128 event cameras[1], with 11 hand gesture classes from 29 subjects under 3 illumination conditions. It contains a total of 1,464 samples, and the owner of the dataset divides 1,176 of them into the training set and 288 into the test set. CIFAR10-DVS

[1]https://inivation.com/wp-content/uploads/2019/08/DVS128.pdf

Datasets	Methods	Spikes	Asyns	Net. Arch.	T	Acc@1(%)
DVS128 Gesture	SLAYER[Shrestha and Orchard, 2018]	✓	✓	8 layers SCNN	1600	93.64
	DECOLLE[Kaiser *et al.*, 2020]	✓	✓	8 layers SCNN	500	95.5
	LIAF-Net[Wu *et al.*, 2021]	✗	✓	5 layers SCNN	60	97.56
	TA-SNN[Yao *et al.*, 2021]	✗	✗	5 layers SCNN	60	98.61
	tdBN[Zheng *et al.*, 2021]	✓	✓	ResNet-17	40	96.87
	SEW-ResNet[Fang *et al.*, 2021a]	✓	✓	9 layers SCNN	16	97.92
	PLIF[Fang *et al.*, 2021b]	✓	✓	7 layers SCNN	20	97.57
	Spikformer[Zhou *et al.*, 2022]	✓	✓	Spikformer-4-256	10	96.9
	Ours	✓	✓	STS-Transformer-1-256	10	$97.22_{\pm 0}$
	Ours	✓	✓	STS-Transformer-1-256	16	$98.38_{\pm 0.165}$
	Ours	✓	✓	STS-Transformer-2-256	10	$\mathbf{97.33}_{\pm 0.160}$
	Ours	✓	✓	STS-Transformer-2-256	16	$\mathbf{98.72}_{\pm 0.160}$
CIFAR10-DVS	NeuNorm[Wu *et al.*, 2019]	✓	✓	7 layers SCNN	Unknown	60.5
	LIAF-Net[Wu *et al.*, 2021]	✗	✓	7 layers SCNN	10	70.4
	TA-SNN[Yao *et al.*, 2021]	✗	✗	7 layers SCNN	10	72.0
	tdBN[Zheng *et al.*, 2021]	✓	✓	ResNet-19	10	67.8
	SEW-ResNet[Fang *et al.*, 2021a]	✓	✓	10 layers SCNN	16	74.4
	Dspike[Li *et al.*, 2021]	✓	✓	ResNet-18	10	75.4
	PLIF[Fang *et al.*, 2021b]	✓	✓	6 layers SCNN	20	74.8
	Spikformer[Zhou *et al.*, 2022]	✓	✓	Spikformer-4-256	10	78.9
	Ours	✓	✓	STS-Transformer-1-256	10	$78.28_{\pm 0.116}$
	Ours	✓	✓	STS-Transformer-1-256	16	$79.60_{\pm 0.171}$
	Ours	✓	✓	STS-Transformer-2-256	10	$\mathbf{78.96}_{\pm 0.048}$
	Ours	✓	✓	STS-Transformer-2-256	16	$\mathbf{79.93}_{\pm 0.132}$

Table 1: Comparison of the inference top-1 accuracy with other works on the neuromorphic vision datasets.

is one of the larger visual neuromorphic datasets, it contains 10,000 event streams converted from 10,000 frame-based images using a dynamic vision sensor. The dataset contains 10 classes, each class has 1,000 samples. In general, researchers divide the first 900 samples of each category into the training set and the remaining 100 samples into the test set. We also used this 9:1 division ratio in our experiments.

The Google Speech Commands (GSC) consists of wav files of a duration of 1 second and a sampling rate of 16kHz. There are V1 and V2 versions of the dataset, the former has 30 categories by 1,881 speakers and the latter has 35 by 2,618 speakers. To use the GSC, we only recognize 12 classes, which include 10 commands which are "yes", "no", "up", "down", "left", "right", "on", "off", "stop" and "go", and another two additional classes namely silence and an unknown class. The unknown class is composed of other words and the silence class is from background noise by splitting the long wav files into 1s clips. To balance samples between classes, we randomly select 1500 samples per command and split them into the training sets and test sets at a ratio of 8:2.

Implementation Details

The initialization of network parameters employs Kaiming normal initialization [He *et al.*, 2015], then we adopt AdamW [Kingma and Ba, 2014] optimizer to optimize the network parameters. The initial learning rate is set to 0.01 and we use a cosine learning rate decay schedule. We adopt the loss function of temporal efficient training [Deng *et al.*, 2022] and the L2 penalty with a value of $1e^{-4}$ is also added. In the tokenization process, we used four 3×3 convolutional layers in the convolutional stem. A max-pooling layer with a step size of 2 is followed by each convolutional layer to divide the original image into 16×16 patches. The LIF neurons of the constructed SNNs adopted a uniform setting, their firing threshold was set to 1, and the decay coefficient τ was set to 0.5.

5.2 Ablation Study

In this section, we will verify that the proposed STSA and STRPB proposed in our work can help SNNs improve prediction accuracy. The results here are the top-1 accuracy of DVS128 Gesture. And the simulation length T is set to 16.

Effect of STSA

We first study the effect of the proposed SNNs-based spatial-temporal self-attention, shown in Fig. 5 (a). We used a network model that only fuses spatial information as a comparison, named SNNs-based spatial self-attention, denoted as SSA in the figure. SSA is similar to the spiking self-attention proposed in [Zhou *et al.*, 2022]. It does not calculate the binary association of patches at different times, and independently calculates the pairwise association of all tokens at the current time at each moment. For STSA and SSA, we have adopted a consistent implementation method. We use one encoder block in our implementation and used a single-head structure. The batch size of both training and testing is 32, and the number of epochs is set to 1000. In the figure, we use blue lines to show the result of STSA, and red lines to show the result of SSA. It can be seen in the figure that due to the

large initial learning rate with the cosine descend schedule, in the first half of training (epoch<500), the accuracy of the test set is prone to shocks. When the epoch is greater than 500, the prediction results of the model on the test set are relatively stable. At this time, it can be clearly seen that our STSA has a great advantage over SSA, with an average performance improvement of about 1.5%. Furthermore, from the prediction results on the training set (dashed line), it is obvious that STSA converges faster than SSA. Such results prove that jointly consider the pairwise relationship of different tokens across the temporal domain and spatial domain better than only considering the relationship in the spatial domain.

Effect of STRPB

We next test the effectiveness of spatial-temporal relative position bias, and the ablation experimental results are shown in Fig. 5 (b). We still use the same implementation settings as the STSA ablation experiments. As a comparison, we do not add STRPB when calculating the attention matrix, and the experimental results are shown in red lines in the figure. The experimental results prove that since the STSA structure itself cannot perceive the relative position of tokens across the temporal domain and spatial domain, the addition of position information is helpful for model learning.

5.3 Comparison with Other Methods

In this section, we conduct experiments on popular datasets with spatial and temporal information, including visual neuromorphic datasets and audio datasets. We will report the results of the experiments separately below.

Neuromorphic Vision Datasets Classifiction

We first conduct experiments on the neuromorphic vision dataset **DVS128 Gesture** and **CIFAR10-DVS**, the results are shown in Table 1. After the event stream is converted to a sequence of frames of size 128 $\times$ 128, for DVS128 Gesture we apply the transformation to the training set which randomly rolls $[-5, 5]$ pixels along the x-axis and y-axis of the frame, while for CIFAR10-DVS we use random flip in addition to random rolling. We use some of the most common simulation lengths T which is 10 or 16, and calculate the three runs' average accuracy to compare with other state-of-the-art works. The number of blocks is set to 1 and 2 to experiment with models of different sizes. Our model trained on the DVS128 Gesture dataset can reach top-1 accuracy of 97.33% with 10 time steps and can achieve top-1 accuracy of 98.72% with 16-time steps. While our model trained on CIFAR10-DVS can achieve top-1 accuracy of 78.96% with 10 time steps and reach top-1 accuracy of 79.93% with 16-time steps. Notably, our model can achieve good results in experiments even when only one block is used. At the same time, our model only needs 10 time steps to achieve similar results as other works with more simulation length, indicating that our model has higher efficiency.

Audio Datasets

We then conduct experiments on audio datasets **Google Speech Commands V1** and **Google Speech Commands V2**, and the results are shown in the Tabel 2. In order to process audio data, we first use Mel Frequency Cepstrum Coefficient (MFCC) to preprocess the data. We set the sampling frequency to 16kHz, the number of Mel filter detectors to 128, and the Fast Fourier Transform (FFT) parameter to 400. The length of Hamming Window is set to 250, the window step is set to 150. Then we resize the frame output by MFCC to 128 $\times$ 128. Since the frame itself contains the spatiotemporal information, we repeatedly input the frame into the network T times, and T is set to 4. As the speech data is relatively simple, we set the token length to 256 and only use one block. It is obvious that our model's performance can far exceed the existing SNN models both on GSC V1 and V2. In the table, we also added some representative works of ANN in the comparison. It can be seen that our model can also surpass the popular ANN models on the Google Speech Commands V1 dataset.

Methods	Net. Type	Acc.(%)
Google Speech Commands V1		
ConvNet [2018]	ANN	87.9
Attention RNN[2018]	ANN	95.6
Residual SNN [2018]	ANN	94.1
NLIP SNN [2021]	SNN	87.9
E2E SNN [2022]	SNN	92.2
Ours	SNN	**96.74**
Google Speech Commands V2		
Attention RNN [2018]	ANN	96.9
E2E SNN [2022]	SNN	92.9
Ours	SNN	**95.18**

Table 2: Comparison the inference top-1 accuracy with other works on the Google Speech Commands dataset.

6 Conclusion

In this paper, we found that the existing attention mechanism conflicts with the asynchronous transmission characteristics of SNN, and proposed a new spiking attention mechanism to help SNN obtain the pairwise correlation of data from both temporal and spatial aspects without losing the asynchronous transmission capability of SNNs. In particular, we propose a spike spatial-temporal self-attention (STSA) module that allows information at different moments to interact through synapses with time delays. Furthermore, in order to improve the speed of network training, we propose a parallel computing method for STSA based on the masked matrix. After that, since STSA cannot actively perceive the spatio-temporal location relationship of different patches, we propose spatial-temporal relative position bias (STRPB) to add location information to tokens to improve the model performance. Finally, we conduct experiments on challenging time series datasets, and our model can achieve state-of-the-art performance on both visual and speech data.

Acknowledgments

This work was supported by the National Science Foundation of China under Grant 62236007 and 61976043 and 62106038, and in part by the Sichuan Science and

Technology Program under Grants 2022YFG0313 and 2023YFG0259.

References

[Amir *et al.*, 2017] 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 *Proc. of CVPR*, 2017.

[Chakraborty *et al.*, 2021] Biswadeep Chakraborty, Xueyuan She, and Saibal Mukhopadhyay. A fully spiking hybrid neural network for energy-efficient object detection. *IEEE Transactions on Image Processing*, 2021.

[Davies *et al.*, 2018] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 2018.

[De Andrade *et al.*, 2018] Douglas Coimbra De Andrade, Sabato Leo, Martin Loesener Da Silva Viana, and Christoph Bernkopf. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929*, 2018.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Proc. of NeurIPS*, 2016.

[Deng *et al.*, 2022] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Diehl and Cook, 2015] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 2015.

[Diehl *et al.*, 2015] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proc. of IJCNN*, 2015.

[Dosovitskiy *et al.*, 2020] 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. *arXiv preprint arXiv:2010.11929*, 2020.

[Fang *et al.*, 2021a] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Proc. of NeurIPS*, 2021.

[Fang *et al.*, 2021b] 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 *Proc. of ICCV*, 2021.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. of ICCV*, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. of CVPR*, 2018.

[Hu *et al.*, 2021] Yifan Hu, Yujie Wu, Lei Deng, and Guoqi Li. Advancing deep residual learning by solving the crux of degradation in spiking neural networks. *arXiv preprint arXiv:2201.07209*, 2021.

[Jansson, 2018] Patrick Jansson. Single-word speech recognition with convolutional neural networks on raw waveforms. 2018.

[Kaiser *et al.*, 2020] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 2020.

[Kheradpisheh *et al.*, 2018] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 2018.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Li *et al.*, 2017] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 2017.

[Li *et al.*, 2021] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Proc. of NeurIPS*, 2021.

[Liu *et al.*, 2021] 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 *Proc. of ICCV*, 2021.

[Luo *et al.*, 2022] Xiaoling Luo, Hong Qu, Yuchen Wang, Zhang Yi, Jilun Zhang, and Malu Zhang. Supervised learning in multilayer spiking neural networks with spike temporal error backpropagation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[Maass, 1997] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 1997.

[Merolla *et al.*, 2014] 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*, 2014.

[Neftci *et al.*, 2019] 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*, 2019.

[Pei *et al.*, 2019] 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*, 2019.

[Pellegrini *et al.*, 2021] Thomas Pellegrini, Romain Zimmer, and Timothée Masquelier. Low-activity supervised convolutional spiking neural networks applied to speech commands recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021.

[Redmon and Farhadi, 2018] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[Rueckauer *et al.*, 2017] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 2017.

[Shrestha and Orchard, 2018] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Proc. of NeurIPS*, 2018.

[Tang and Lin, 2018] Raphael Tang and Jimmy Lin. Deep residual learning for small-footprint keyword spotting. In *Proc. of ICASSP*, 2018.

[Tavanaei *et al.*, 2019] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 2019.

[Wang *et al.*, 2022] Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *Proc. of IJCAI*, 2022.

[Warden, 2018] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[Wu *et al.*, 2018] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 2018.

[Wu *et al.*, 2019] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proc. of AAAI*, 2019.

[Wu *et al.*, 2021] Zhenzhi Wu, Hehui Zhang, Yihan Lin, Guoqi Li, Meng Wang, and Ye Tang. Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[Xu *et al.*, 2021] Mingkun Xu, Yujie Wu, Lei Deng, Faqiang Liu, Guoqi Li, and Jing Pei. Exploiting spiking dynamics with spatial-temporal feature normalization in graph learning. In *Proc. of IJCAI*, 2021.

[Yang *et al.*, 2022] Qu Yang, Qi Liu, and Haizhou Li. Deep residual spiking neural network for keyword spotting in low-resource settings. *Proc. Interspeech 2022*, 2022.

[Yao *et al.*, 2021] Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *Proc. of ICCV*, 2021.

[Yao *et al.*, 2022] Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *arXiv preprint arXiv:2209.13929*, 2022.

[Zhang *et al.*, 2021] Yun Zhang, Hong Qu, Xiaoling Luo, Yi Chen, Yuchen Wang, Malu Zhang, and Zefang Li. A new recursive least squares-based learning algorithm for spiking neurons. *Neural Networks*, 138:110–125, 2021.

[Zhang *et al.*, 2022] Duzhen Zhang, Tielin Zhang, Shuncheng Jia, Qingyu Wang, and Bo Xu. Recent advances and new frontiers in spiking neural networks. In *Proc. of IJCAI*, 2022.

[Zheng *et al.*, 2021] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proc. of AAAI*, 2021.

[Zhou *et al.*, 2022] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

[Zhu *et al.*, 2022] Rui-Jie Zhu, Qihang Zhao, Tianjing Zhang, Haoyu Deng, Yule Duan, Malu Zhang, and Liang-Jian Deng. Tcja-snn: Temporal-channel joint attention for spiking neural networks. *arXiv preprint arXiv:2206.10177*, 2022.