Spikeformer: A Novel Architecture for Training High-Performance Low-Latency Spiking Neural Network

Yudong Li, Yunlin Lei, Xu Yang*

School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
{loyd, leiyunlin, pyro_yangxu}@bit.edu.cn

Abstract

Spiking neural networks (SNNs) have made great progress on both performance and efficiency over the last few years, but their unique working pattern makes it hard to train a high-performance low-latency SNN. Thus the development of SNNs still lags behind traditional artificial neural networks (ANNs). To compensate this gap, many extraordinary works have been proposed. Nevertheless, these works are mainly based on the same kind of network structure (i.e. CNN) and their performance is worse than their ANN counterparts, which limits the applications of SNNs. To this end, we propose a novel Transformer-based SNN, termed "Spikeformer", which outperforms its ANN counterpart on both static dataset and neuromorphic dataset and may be an alternative architecture to CNN for training high-performance SNNs. First, to deal with the problem of "data hungry" and the unstable training period exhibited in the vanilla model, we design the Convolutional Tokenizer (CT) module, which improves the accuracy of the original model on DVS-Gesture by more than 16%. Besides, in order to better incorporate the attention mechanism inside Transformer and the spatio-temporal information inherent to SNN, we adopt spatio-temporal attention (STA) instead of spatial-wise or temporal-wise attention. With our proposed method, we achieve competitive or state-of-the-art (SOTA) SNN performance on DVS-CIFAR10, DVS-Gesture, and ImageNet datasets with the least simulation time steps (i.e. low latency). Remarkably, our Spikeformer outperforms other SNNs on ImageNet by a large margin (i.e. more than 5%) and even outperforms its ANN counterpart by 3.1% and 2.2% on DVS-Gesture and ImageNet respectively, indicating that Spikeformer is a promising architecture for training large-scale SNNs and may be more suitable for SNNs compared to CNN. We believe that this work shall keep the development of SNNs in step with ANNs as much as possible. Code will be publicly available.

Introduction

Inspired by biological neural functionality, spiking neural network (SNN) is known as a promising bionic model with low-power computation and has gained considerable attention in recent years. As the third generation of artificial neural network (ANN), SNN is computationally more powerful than other neural networks with regard to the number of neurons that are needed (Maass 1997). When embedded on neuromorphic hardware such as TrueNorth (DeBole et al. 2019) and Loihi (Davies et al. 2018), SNN shows great potential to process spatio-temporal information effectively with low energy consumption due to its intrinsic spatio-temporal characteristic and event-driven spike communication mechanism. Nevertheless, owing to the lack of appropriate learning algorithm, its performance is not as good as traditional neural networks.

To tackle this problem, there are two main routes to train a SNN with high performance. The first is the ANN-to-SNN method, which obtains a SNN by simulating a pre-trained ANN model's behavior (Hu, Tang, and Pan 2018; Han, Srinivasan, and Roy 2020; Sengupta et al. 2019). With adequate simulation steps, the conversion method can achieve almost lossless accuracy compared to its ANN counterpart. However, extremely high latency is often required to achieve satisfying accuracy and limits the practical applications of SNNs. The other approach is the surrogate gredient (SG) method (Neftci, Mostafa, and Zenke 2019). To handle the non-differentiability of the spiking mechanism, the SG method utilizes surrogate gradients to approximate the gradients of the non-differentiable activation function on the backpropagation process. This method can directly train SNNs with low latency, but it cannot achieve high performance comparable to leading ANNs. Overall, SNNs obtained by aforementioned methods suffer from either low performance or high latency which hinders the development of training large-scale SNN models.

To deal with it, there are several methods and models proposed to train large-scale SNNs, such as tdBN (Zheng et al. 2021) and SEW-ResNet (Fang et al. 2021a). These methods and models are mainly based on convolutional neural networks (CNNs), such as ResNet (He et al. 2016) and VGG (Simonyan and Zisserman 2015), which dominate the field of computer vision for years. However, in the field of computer vision, CNNs are gradually taken place by Vision Transformers (ViT) because of their excellent capabilities at capturing long-range dependencies. With the emergence of Transformer (Vaswani et al. 2017) and ViT (Dosovitskiy et al. 2020), they have been the dominant approaches in sequence modeling problems such as language modeling and video understanding. Actually, the tasks of SNNs (e.g. DVS-Gesture) can be viewed as sequence modeling problems as well. Hence, it's natural to integrate SNN with inherent spatio-temporal information into ViT with spatio-temporal

*Corresponding author

attention (e.g. TimeSformer). However, due to the "data hungry" property and the substandard optimizability of ViT, the vanilla architecture suffers from poor performance (82.29% on DVS-Gesture) and the training accuracy even does not converge when integrated with spiking neurons.

To this end, we design the Convolutional Tokenizer (CT) module, which is a SNN-oriented convolutional block. With this module, we improve the top-1 accuracy of the original model on DVS-Gesture from 82.29% to 95.83%. And its SNN counterpart manages to converge and even obtains better result (i.e. 98.96% on DVS-Gesture). To our best knowledge, this is the first work to introduce spatio-temporal attention (STA) into SNNs and directly train a Transformer-based SNN with high performance and low latency. Although there are a few works utilizing attention mechanism in SNNs to improve performance, they are mainly studying either spatial-wise attention (Cannici et al. 2019; Xie et al. 2016; Kundu et al. 2021) or temporal-wise attention (Yao et al. 2021) separately. On the contrary, we integrate spatial and temporal attention into SNNs at the same time and achieve state-of-the-art SNN performance on DVS-Gesture, DVS-CIFAR10 and ImageNet datasets with the least simulation time steps (i.e. low latency). Prior to this work, (Mueller et al. 2021) obtain a Transformer-based SNN by conversion method. But their method requires relatively long time steps and they only experiment on simple tasks, while ours demands much fewer simulation time steps and achieves remarkable result on the challenging ImageNet dataset. On ImageNet, our Spikeformer obtains 7%/5% improvement compared to the SOTA SNN models trained by SG/conversion method and even outperforms its ANN counterpart by 2.2% for the first time.

We summarize our contributions as follows:

- To deal with the "data hungry" property and substandard optimizability of ViT, we design the CT module and gain essential improvement of accuracy (over 16%) on DVS-Gesture.
- We integrate STA into SNN to better utilize the spatio-temporal information inside the spiking neurons and achieve high performance with low latency.
- We propose a directly trained Transformer-based SNN, which may be an alternative scheme to CNN for training large-scale SNNs.
- With extensive experiments, we demonstrate the robustness of Spikeformer in terms of depth and time steps and achieve competitive or state-of-the-art (SOTA) SNN performance with low latency on DVS-CIFAR10, DVS-Gesture, and ImageNet datasets.

Related Work

Large-scale SNN model In the past few years, as SNNs have attracted increasing attention, lots of extraordinary works on training large-scale SNN models have sprung up. (Hu, Tang, and Pan 2018) are the first to build a SNN deeper than 100 layers by conversion method. But it requires hundreds or thousands of time steps, which results in high latency. It is not until (Zheng et al. 2021) propose the tdBN method that we can extend directly-trained SNNs from fewer than 10 layers to 50 layers. They effectively alleviate gradient vanishing or explosion and balance the threshold and input on each neuron to get appropriate firing rates with a modified batch normalization method. (Fang et al. 2021a) further analyze the gradient and identity map issues in SNNs, alter the connection of residual block, and propose SEW ResNet, which extends the depth to 152 layers. These large-scale SNNs are mainly based on CNNs, while Transformer-based architectures have gained great success in various domains (e.g. NLP, CV).

Transformer Since the proposal of Transformer (Vaswani et al. 2017), it has been a dominant approach in NLP due to its extraordinary capabilities at capturing long-range dependencies among words. (Dosovitskiy et al. 2020) are the first to apply it in computer vision and achieve remarkable results, which sets off a wave of research on ViT. (Dosovitskiy et al. 2020) note that ViT lacks inductive biases inherent to CNN, and therefore it requires more data to gain superior results. This issue has subsequently led to a series of studies to follow (Liu et al. 2021; Mehta and Rastegari 2021; Lee, Lee, and Song 2021; Hassani et al. 2021; Cao, Yu, and Wu 2022). (Hassani et al. 2021) introduce inductive biases into ViT with a convolutional head and successfully alleviate "data hungry". Moreover, (Xiao et al. 2021) propose that ViT exhibits substandard optimizability due to the patchify stem of ViT. They empirically demonstrate that early convolution can increase optimization stability of ViT and also improve peak performance. Despite the rapid development of Transformers in the field of ANNs, they are rarely used in SNNs due to the "data hungry" property and the substandard optimizability. (Mueller et al. 2021) convert a pre-trained Transformer to SNN, but this method requires relatively long time steps and has limited usages. Based on these observations, we propose the CT module and successfully train a high-performance Transformer-based SNN with low latency.

Attention mechanism in SNN The attention mechanism enables the model to pay more attention to the most informative components of input. It has been applied in SNN as spatial-wise attention (Cannici et al. 2019; Xie et al. 2016; Kundu et al. 2021) or temporal-wise attention (Yao et al. 2021), but these works either capture dependencies inside frames or utilize the statistical characteristics of the frames input at different time steps, which may lose information in spatial or temporal domain. Thus we integrate both spatial and temporal attention into SNN in a divided way and achieve state-of-the-art results.

Method

Spiking Neuron Model

In general, spiking neurons can be clssified into two classes by their data transmission form. One is the spike-based neuron, such as LIF (Burkitt 2006) and PLIF (Fang et al. 2021b), which uses spike streams for communication. When embedded on neuromorphic hardware, this kind of neurons will skip computation if they haven't received any input spike, i.e. event-driven computation. Thus they can process

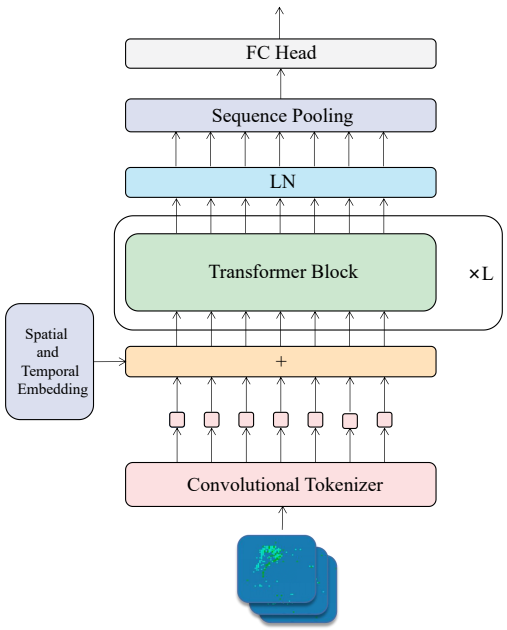


Figure 1: An overview of Spikeformer.

spatio-temporal information in an energy-saving way. The other is the analog-based neuron, such as LIAF (Wu et al. 2021), which uses the dynamic characteristics of spiking neurons but transmits analog values. This kind of neurons possesses strong representational capabilities due to their inherent spatio-temporal feature, but they cannot skip computation, which makes them cost more energy than spike-based neurons. Without loss of generality, we experiment with both types of neurons. Formally, We use the following equations to describe the dynamics of spiking neurons,

$$H[t] = V[t-1] + \frac{1}{\tau}(X[t] - (V[t-1] - V_{reset})) \quad (1)$$

$$S[t] = \Theta(H[t] - V_{th}), \quad (2)$$

$$V[t] = H[t](1 - S[t]) + V_{reset}S[t], \quad (3)$$

where $H[t]$ and $V[t]$ denote the membrane potential after integrating input and after the trigger of a spike at time step t, respectively. $X[t]$ is the input at time-step t, and $S[t]$ is the output spike at time-step t. The input propagates to the next layer is $S[t]$ for LIF model and $ReLU(H[t])$ for LIAF model. $\Theta(x)$ is the Heaviside step function, which is defined by $\Theta(x) = 1$ for $x > 0$, otherwise $\Theta(x) = 0$. V_{th} and V_{reset} denote firing threshold and reset potential, respectively. τ represents the membrane time constant. When τ is learnable, we have PLIF model (Fang et al. 2021b) and PLIAF model. In this paper, the surrogate method is used in the backpropagation process, and the surrogate function is the same as (Fang et al. 2021a). The details are presented in **Supplementary Material A**.

Model Architecture

An overview of Spikeformer is depicted in Fig. 1. Compared to the standard ViT, we use a Convolutional Tokenizer (CT) module to process a series of 2D frames and then reshape them into a sequence of flattened token embeddings instead of the original patchify stem. Furthermore, to better utilize the spatio-temporal information and achieve a trade-off between accuracy and computational complexity, we adopt divided Space-Time Attention (Bertasius, Wang, and Torresani 2021) in Transformer block. Finally, as it is non-trivial to deal with class token in divided Space-Time Attention and the class token may lose information when discarding embedings (Beyer, Zhai, and Kolesnikov 2022), we pool the sequential based information from the last transformer block using a sequential pooling method (Hassani et al. 2021), which may be a better choice than global average-pooling as it can learn to assign more weights to more informative tokens. We would elaborate each component in the following subsections.

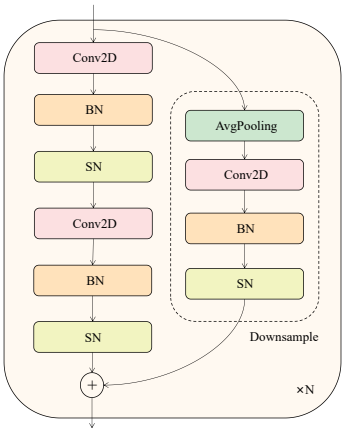


Figure 2: Illustration of Convolutional Tokenizer. Note that the part enclosed by the dotted line only appears when downsampling.

Convolutional Tokenizer In order to alleviate "data hungry" and help stabilize the training period, we propose the CT module to introduce inductive biases into Spikeformer. The architecture of CT module is illustrated in Fig. 2. Given a series of input frames $x \in \mathbb{R}^{T\times H\times W\times C}$:

$$Conv(x) = SN(BN(Conv2d(x))), \quad (4)$$

$$CT_d(x) = Conv(Conv(x)) + Conv(AvgPool(x)), \quad (5)$$

$$CT(x) = Conv(Conv(x)) + x, \quad (6)$$

where Conv2d and BN denote convolutional layers and batch normalization layers, which will process each time step separately, and SN denotes spiking neurons that will integrate input and output spike trains. $CT_d(\cdot)$ and $CT(\cdot)$ depict downsample module and normal module respectively. Multiple downsample modules and normal modules are stacked together to encode input frames and output spike trains to the following blocks. The design of the CT module mainly follows the architecture of SEW block (Fang et al. 2021a) as it is proven to implement identity mapping and is beneficial for training deep spiking neural networks. But

in the downsample module, we use a 2×2 average pooling layer with a stride of 2 and a stride-1 Conv1 $\times$ 1 instead of a stride-2 Conv1 $\times$ 1 in the shortcut connection in order not to discard information (He et al. 2019). And this minimal modification will not influence the functionality of identity mapping, the theoretical analysis and further description are presented in **Supplementary Material A**.

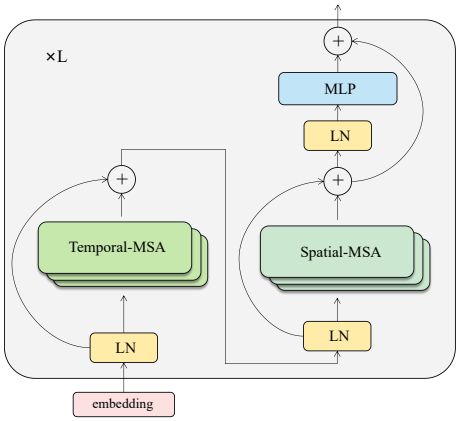


Figure 3: Illustration of Transformer Block.

Transformer Block Our Spikeformer contains L encoding blocks and the architecture of transformer block is shown in Fig. 3. The output of the last CT module $\hat{x} \in \mathbb{R}^{T \times H' \times W' \times D}$ is flattened into vectors $\hat{a}_{(p,t)} \in \mathbb{R}^D$, with $p = 1, ..., N$ denoting spatial locations and $t = 1, ..., T$ denoting temporal locations. $N = H' \times W'$ and D depicts the dimensionality of tokens. To encode the spatio-temporal position of each token, we add a learnable positional embedding $e^{pos}_{(p,t)} \in \mathbb{R}^D$ to each token:

$$z^{(0)}_{(p,t)} = \hat{a}_{(p,t)} + e^{pos}_{(p,t)} \tag{7}$$

Each encoding block l will compute a query/key/value vector:

$$q^{(l,h)}_{(p,t)} = LN(z^{(l)}_{(p,t)}) W^{(l,h)}_Q \in \mathbb{R}^{D_h} \tag{8}$$

$$k^{(l,h)}_{(p,t)} = LN(z^{(l)}_{(p,t)}) W^{(l,h)}_K \in \mathbb{R}^{D_h} \tag{9}$$

$$v^{(l,h)}_{(p,t)} = LN(z^{(l)}_{(p,t)}) W^{(l,h)}_V \in \mathbb{R}^{D_h} \tag{10}$$

where $LN(\cdot)$ denotes LayerNorm (Ba, Kiros, and Hinton 2016), $h = 1, ..., H$ is the index over multiple attention heads and $D_h = D/H$ is the dimensionality of each query/key/value vector. In Spikeformer, the attention mechanism is divided into spatial attention and temporal attention:

$$h^{(l,h)space}_{(p,t)} = \sum_{p'=1}^{N} SM(\frac{q^{(l,h)^T}_{(p,t)}}{\sqrt{D_h}} \left[\{k^{(l,h)}_{(p',t)}\}_{p'=1,...,N}\right]) v^{(l,h)}_{(p',t)}, \tag{11}$$

$$h^{(l,h)time}_{(p,t)} = \sum_{t'=1}^{T} SM(\frac{q^{(l,h)^T}_{(p,t)}}{\sqrt{D_h}} \left[\{k^{(l,h)}_{(p,t')}\}_{t'=1,...,T}\right]) v^{(l,h)}_{(p,t')}, \tag{12}$$

$$s^{(l)}_{(p,t)} = Concat(h^{(l,1)}_{(p,t)}, ..., h^{(l,H)}_{(p,t)}) \cdot W_O, \tag{13}$$

where $SM(\cdot)$ denotes the softmax activation function, $W_O \in \mathbb{R}^{H \cdot D_h \times D}$ is the projection matrix, and s is the output of temporal attention (TMSA) or spatial attention (SMSA). Thus a transformer block in Spikeformer can be formulated as:

$$z'_l = TMSA(LN(z_{l-1})) + z_{l-1} \tag{14}$$

$$z''_l = SMSA(LN(z'_l)) + z'_l \tag{15}$$

$$z_l = FC(SN(FC(LN(z''_l)))) + z''_l \tag{16}$$

with $l = 1, ..., L$ depicting an index over blocks.

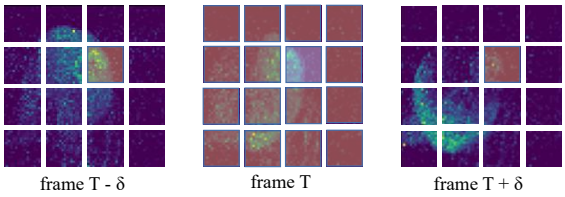


Figure 4: Illustration of Divided Space-Time Attention scheme. We denote in white the query patch and show in orange its self-attention space-time neighborhood. Patches without color are not used for self-attention computation of the white patch.

Intuitively, the divided Space-Time attention can be illustrated by Fig. 4. The query token in frame T first computes its attention scores with other tokens that are at the same location in other frames as temporal attention. And then it computes its attention scores with other tokens that are in the same frame as spatial attention. Not only can this scheme reduce memory cost and computational complexity, but it also achieves satisfying accuracy.

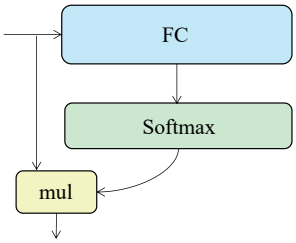


Figure 5: Illustration of Sequence Pooling.

Sequence Pooling In order to fully utilize the information in the outputs of the last transformer block, instead of discarding most of them (i.e. class token), we adopt Sequence Pooling to compute the weighted sum of the outputs and send the result to the classifier, as shown in Fig. 5. Given the normalized output tensor of the last transformer block

Model	#Layers	#Heads	Ratio	Dim
Spikeformer-2	2	2	1	128
Spikeformer-4	4	2	1	128
Spikeformer-7	7	4	2	256
Spikeformer-7L	7	8	3	512

Table 1: Details of Spikeformer model variants. Ratio denotes the MLP expansion ratio.

$z^l \in \mathbb{R}^{T\times N\times D}$, we feed z^l to a fully connected (FC) layer, apply softmax activation, and transpose it:

$$W = SM(FC(z^l))^T. \tag{17}$$

The FC layer is equivalent to a matrix $W_{FC} \in \mathbb{R}^{D\times 1}$ and thus $W \in \mathbb{R}^{T\times 1\times N}$. Then we multiply z^l by W:

$$\hat{z} = W \times z^l \in \mathbb{R}^{T\times 1\times D}, \tag{18}$$

squeeze the second dimension and get $z \in \mathbb{R}^{T\times D}$. Ultimately, we send the weighted sum $\hat{z}$ to the classifier with a minimal loss of information.

Experiment

We evaluate our models on static dataset (ImageNet) and neuromorphic datasets (DVS-CIFAR10, DVS-Gesture). PLIAF neuron model and PLIF neuron model are adopted for static dataset and neuromorphic datasets, respectively. Extensive ablation experiments reveal the cruciality of CT module and the robustness of our method in terms of depth and time steps. And then we compare our results with state-of-the-art methods and some concurrent works to demonstrate the superiority of our method. The details about experimental setup and training strategy are presented in **Supplementary Material B**.

Model Variants We base Spikeformer configurations on those used for CCT (Hassani et al. 2021), as summarized in Table 1. We use brief notation to indicate model variants and the CT stem: for instance, Spikeformer-7/5×2×3 means the Spikeformer-7 variant using $2 \times 3 = 6$ CT modules with kernel size 5 and every 2 CT modules have 1 downsample module. Note that every downsample CT module will reduce the number of tokens by 4, which will be further discussed in the ablation studies of CT module. Details of the model architecture can be found in **Supplementary Material C**.

Datasets We verify our method on three popular datasets, which consist of both static dataset and neuromorphic datasets. The first is DVS-Gesture (Amir et al. 2017), which contains 1342 records in the training set and 288 examples for testing captured by DVS cameras. The second is DVS-CIFAR10 (Li et al. 2017), which is converted from the famous CIFAR10 dataset to its dynamic form by scanning each sample in front of DVS cameras. The last one is ImageNet (Deng et al. 2009), which is the most popular benchmark dataset used for large-scale image classification.

CT	SN	Accuracy(%)
✗	✗	82.29
✗	✓	N/A
✓	✗	95.83
✓	✓	**98.96**

Table 2: Ablation studies on the Convolutional Tokenizer (CT) module and spiking neurons (SN) on DVS-Gesture dataset. N/A denotes that the model does not converge.

Dataset	Tokens	Accuracy(%)
DVS-CIFAR10	4096	N/A
	1024	73.10
	256	**80.0**
	64	79.6
	16	75.4
	4	73.0
DVS-Gesture	4096	N/A
	1024	82.64
	256	**90.97**
	64	88.89
	16	87.50
	4	85.42

Table 3: Ablation on the CT architecture. Note that increasing the number of tokens to 4096 causes a GPU memory overflow.

Ablation Studies

We perform extensive ablation experiments to verify the importance of the CT module and draw a empirical rule to design the CT module. And then we conduct experiments with various time steps and layers of transformer block to further demonstrate the robustness of our method. Note that the ablation studies are not aimed at pushing SOTA results, so we adopt realtively simple training recipes.

CT Module When using the patchify stem, the ANN version performs poorly (82.29%) on DVS-Gesture. There is a large gap between the training accuracy (100%) and the testing accuracy (82.29%). The model is severely overfitted because of the lack of inductive biases. And the training accuracy of the SNN version even does not converge because of the substandard optimizability caused by the patchify stem (Xiao et al. 2021). To deal with it, we propose CT module to help stabilize the training period, alleviate "data hungry" and make the model generalize better. In Table 2, we can observe that CT module greatly boosts the accuracy. With CT module, we obtain over 10% improvement for ANN version and successfully train the SNN version (i.e. Spikeformer), which further enlarges the gap to more than 16%. These results illustrate the indispensability of the CT module.

Moreover, the design of the CT module is crucial for the performance of Spikeformer. The more the downsample

Work	Model	Time Steps	Neuron	Accuracy(%)
(Shen, Zhao, and Zeng 2022)	ResNet-18	10	PLIF	96.75
(Fang et al. 2021b)	5Conv, 2Fc	20	PLIF	97.57
(Fang et al. 2021a)	7B-Net	16	PLIF	97.92
(Yao et al. 2021)	TA-SNN	60	LIF	95.48
			LIAF	98.61
This work	Spikeformer-7/5×1×3	**16**	PLIF	**98.96**

Table 4: Comparison with the SOTA methods on DVS-Gesture dataset.

blocks, the fewer the number of tokens. And the number of tokens determines the granularity of the feature map, which will influence the performance of transformer blocks (Dosovitskiy et al. 2020). From Table 3, we note that Spikeformer performs best with around 200 tokens. Thus, we follow this rule to design our models in the following experiments.

Depth And Time Steps In addition to the CT module, we also conduct depth analysis on transformer blocks. We experiment with three model variants on DVS-Gesture and DVS-CIFAR10 datasets, as shown in Table 5. Table 5 shows that the accuracy on both datasets consistently increases with the deepening of transformer blocks. And even with shallow architecture (i.e. Spikeformer-2), our model still performs well.

Dataset	Model	Acc.(%)
DVS-Gesture	Spikeformer-7/5×1×3	**98.96**
	Spikeformer-4/5×1×3	97.22
	Spikeformer-2/5×1×3	96.88
DVS-CIFAR10	Spikeformer-7/3×2×3	**80.0**
	Spikeformer-4/3×2×3	78.6
	Spikeformer-2/3×2×3	77.9

Table 5: Ablation on the depth of the transformer blocks.

To further demonstrate the robustness of our method, we evaluate our model with different time steps, as shown in Table 6. On DVS-CIFAR10 dataset, our method does not suffer from severe degradation even with extremely low time steps. We attribute this phenomenon to the dataset itself. DVS-CIFAR10 is collected by scanning static images with DVS cameras. Hence, this dataset inherently does not contain much temporal information. Enlarging the time steps is equivalent to making the network vote more times. The performance on DVS-CIFAR10 is more relevant to the ability to capture spatial relationships. Rather, DVS-Gesture dataset contains abundant temporal information because it is a collection of moving gestures performed by different individuals. Thus, its performance is more sensitive to the simulation time steps. The performance on DVS-Gesture is more relevant to the ability to capture temporal relationships. And our method achieves superior results on both datasets with various time steps, which demonstrates the strong spatio-temporal modeling capabilities of our method.

DVS-Gesture		DVS-CIFAR10	
T	Acc.(%)	T	Acc.(%)
16	**98.96**	4	**78.8**
12	97.22	3	77.9
8	95.14	2	77.7
4	93.75	1	77.6

Table 6: Ablation on time steps.

Comparison With SOTA Methods

Following the "200 tokens" rule, we adopt three different architectures. Since the input resolution of DVS-Gesture and DVS-CIFAR10 is 128×128, we downsample the input frames for 3 times, which results in 256 tokens. And for ImageNet with an input size of 224×224, we downsample the input for 4 times, which results in 196 tokens. Because DVS-Gesture dataset contains rich temporal information, it needs more time steps to gain better performance and takes up more memory. To balance the memory cost, we adopt Spikeformer-7/5×1×3 for DVS-Gesture and Spikeformer-7/3×2×3 for DVS-CIFAR10. But the original Spikeformer-7 variant is too simple for ImageNet dataset, we adopt Spikeformer-7L/3×2×4 for ImageNet and compare these methods with SOTA methods and concurrent works.

DVS-Gesture In Table 4, we compare our method with SOTA methods on DVS-Gesture. (Fang et al. 2021a) achieve 97.92% top-1 accuracy with 16 time steps, while we obtain 98.96% with the same time steps. Compared to (Yao et al. 2021), we achieve better result with much lower latency (i.e. less than one-third of their simulation time steps) and spike-based neurons (i.e. lower energy consumption). Since the performance on DVS-Gesture tends to saturate, the capacity of our method is not fully demonstrated. Thus we further conduct experiments on the more challenging DVS-CIFAR10 dataset.

DVS-CIFAR10 Our method achieves remarkable result on DVS-CIFAR10, as shown in Table 7. With the same time steps as (Fang et al. 2021a), our method outperforms their results by a large margin (i.e. 16.6%) and even achieves better performance than (Meng et al. 2022), who use five times as many time steps as ours and ten times more parameters. Our method gains comparable results to (Li et al. 2022) and (Shen, Zhao, and Zeng 2022) with fewer time

Work	Model	Time Steps	Neuron	Parameters(M)	Accuracy(%)
(Fang et al. 2021b)	4Conv, 2Fc	20	PLIF	17.4	74.8
(Fang et al. 2021a)	Wide-7B-Net	4, 8, 16	PLIF	1.19	64.8, 70.2, 74.4
(Yao et al. 2021)	TA-SNN	10	LIF	2.10	71.1
			LIAF	2.10	72.0
(Li et al. 2021)	ResNet-18	10	LIF	11.69	75.4
(Meng et al. 2022)	VGG-11	20	LIF	132.86	77.3
(Shen, Zhao, and Zeng 2022)	ResNet-18	10	PLIF	11.69	81.45
(Li et al. 2022)	VGG-11	10	LIF	132.86	81.7
This work	Spikeformer-7/3×2×3	**4**	PLIF	**9.28**	**81.4**

Table 7: Comparison with the SOTA methods on DVS-CIFAR10 dataset.

Method	Work	Model	T	Par.(M)	Accuracy(%)
ANN	(He et al, 2015)	ResNet-50	-	25.56	77.15
		ResNet-101	-	44.55	78.25
		ResNet-152	-	60.20	78.57
	(Beyer, Zhai, and Kolesnikov 2022)	ViT-S/16*	-	22.04	67.1
		ViT-S/16†	-		76.1
ANN-to-SNN	(Sengupta et al. 2019)	VGG-16	2500	138.36	69.96
	(Hu, Tang, and Pan 2018)	ResNet-50	350	25.56	72.75
	(Han, Srinivasan, and Roy 2020)	VGG-16	4096	138.36	73.09
Directly Training	(Zheng et al. 2021)	ResNet-34(large)	6	86.13	67.05
	(Fang et al. 2021a)	SEW ResNet-152	4	60.19	69.26
	(Li et al. 2021)	VGG-16	5	138.36	71.24
	(Meng et al. 2022)	PreAct-ResNet-18	50	11.69	67.74
	This work	Spikeformer-7L/3×2×4	**4**	**38.75**	**78.31**

Table 8: Comparison with the SOTA methods on ImageNet dataset.*The original version without additional data and strong data augmentation. †The original version without additional data but adopts strong data augmentation.

steps and parameters. These comparisons strongly support that our Spikeformer is equipped with distinguished spatio-temporal representational ability that can fully utilize limited spatio-temporal information to extract key information.

ImageNet To explore the potential of Spikeformer to be a large-scale SNN architecture, we conduct experiments on ImageNet dataset with PLIAF model, as shown in Table 8. Our Spikeformer outperforms other SNNs trained by SG method or conversion method by a large margin with low latency and medium parameters. Moreover, we even outperform ViT-S/16 that is not pretrained with additional data, which further demonstrates that our CT module do help alleviate "data hungry". To our best knowledge, this is the first time directly trained SNNs achieve comparable results to ANNs on ImageNet with fewer parameters. Given limited time steps and parameters, our model manages to focus on the most informative components of the input and possesses the capability to fully utilize parameters to learn robust representation. These results indicate that Spikeformer is a promising architecture for training large-scale SNNs.

Conclusion

In this paper, we introduce spatio-temporal attention (STA) into spiking neural network (SNN) and directly train a Transformer-based SNN (i.e. Spikeformer). Replacing the original patchify stem with CT module, we manage to alleviate "data hungry", stabilize the training period and obtain a high-performance low-latency SNN. Based on extensive ablation studies, we demonstrate the cruciality of CT module and the robustness of Spikeformer in terms of depth and time steps. Furthermore, we also propose an empirical rule to design CT module (i.e. the "200 tokens" rule). This module can be further developed into a delicately designed multi-stages network (e.g. ResNet) in future work. But in this paper, we adopt simple design as not to overwhelm the importance of transformer blocks, which may be a suboptimal choice. Ultimately, our Spikeformer gains competitive or state-of-the-art results on both neuromorphic datasets and large-scale static dataset with low latency, which demonstrates its superior capabilities to capture spatio-temporal information and the potential to be an alternative architecture to CNN for training large-scale SNNs. We believe this architecture will bring new insight into training large-scale SNNs and promote the development of SNNs.

References

Amir, A.; Taba, B.; Berg, D. J.; Melano, T.; McKinstry, J. L.; di Nolfo, C.; Nayak, T. K.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; Kusnitz, J. A.; DeBole, M. V.; Esser, S. K.; Delbrück, T.; Flickner, M.; and Modha, D. S. 2017. A Low Power, Fully Event-Based Gesture Recognition System. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7388–7397.

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

Bertasius, G.; Wang, H.; and Torresani, L. 2021. Is space-time attention all you need for video understanding? In *ICML*, volume 2, 4.

Beyer, L.; Zhai, X.; and Kolesnikov, A. 2022. Better plain ViT baselines for ImageNet-1k. *arXiv preprint arXiv:2205.01580*.

Burkitt, A. N. 2006. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biological cybernetics*, 95(1): 1–19.

Cannici, M.; Ciccone, M.; Romanoni, A.; and Matteucci, M. 2019. Attention mechanisms for object recognition with event-based cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1127–1136. IEEE.

Cao, Y.-H.; Yu, H.; and Wu, J. 2022. Training vision transformers with only 2040 images. *arXiv preprint arXiv:2201.10728*.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

DeBole, M. V.; Taba, B.; Amir, A.; Akopyan, F.; Andreopoulos, A.; Risk, W. P.; Kusnitz, J.; Otero, C. O.; Nayak, T. K.; Appuswamy, R.; et al. 2019. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5): 20–29.

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

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Fang, W.; Chen, Y.; Ding, J.; Chen, D.; Yu, Z.; Zhou, H.; Tian, Y.; and other contributors. 2020. SpikingJelly. https://github.com/fangwei123456/spikingjelly. Accessed: YYYY-MM-DD.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2021a. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 21056–21069.

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021b. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2661–2671.

Han, B.; Srinivasan, G.; and Roy, K. 2020. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13558–13567.

Hassani, A.; Walton, S.; Shah, N.; Abuduweili, A.; Li, J.; and Shi, H. 2021. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; and Li, M. 2019. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 558–567.

Hu, Y.; Tang, H.; and Pan, G. 2018. Spiking Deep Residual Networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. In *European conference on computer vision*, 646–661. Springer.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

Kundu, S.; Datta, G.; Pedram, M.; and Beerel, P. A. 2021. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3953–3962.

Lee, S. H.; Lee, S.; and Song, B. C. 2021. Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.13492*.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11: 309.

Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 23426–23439.

Li, Y.; Kim, Y.; Park, H.; Geller, T.; and Panda, P. 2022. Neuromorphic Data Augmentation for Training Spiking Neural Networks. *arXiv preprint arXiv:2203.06145*.

Liu, Y.; Sangineto, E.; Bi, W.; Sebe, N.; Lepri, B.; and Nadai, M. 2021. Efficient training of visual transformers with small datasets. *Advances in Neural Information Processing Systems*, 34: 23818–23830.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *International Conference on Learning Representations*.

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

Mehta, S.; and Rastegari, M. 2021. MobileViT: Lightweight, General-purpose, and Mobile-friendly Vision Transformer. In *International Conference on Learning Representations*.

Meng, Q.; Xiao, M.; Yan, S.; Wang, Y.; Lin, Z.; and Luo, Z.-Q. 2022. Training High-Performance Low-Latency Spiking Neural Networks by Differentiation on Spike Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12444–12453.

Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; et al. 2018. Mixed Precision Training. In *International Conference on Learning Representations*.

Mueller, E.; Studenyak, V.; Auge, D.; and Knoll, A. 2021. Spiking Transformer Networks: A Rate Coded Approach for Processing Sequential Data. In *2021 7th International Conference on Systems and Informatics (ICSAI)*, 1–5. IEEE.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. 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.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience*, 13: 95.

Shen, G.; Zhao, D.; and Zeng, Y. 2022. EventMix: An Efficient Augmentation Strategy for Event-Based Data. *arXiv preprint arXiv:2205.12054*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

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

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wu, Z.; Zhang, H.; Lin, Y.; Li, G.; Wang, M.; and Tang, Y. 2021. Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*.

Xiao, T.; Singh, M.; Mintun, E.; Darrell, T.; Dollár, P.; and Girshick, R. 2021. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 34: 30392–30400.

Xie, X.; Qu, H.; Yi, Z.; and Kurths, J. 2016. Efficient training of supervised spiking neural network via accurate synaptic-efficiency adjustment method. *IEEE transactions on neural networks and learning systems*, 28(6): 1411–1424.

Yao, M.; Gao, H.; Zhao, G.; Wang, D.; Lin, Y.; Yang, Z.; and Li, G. 2021. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10221–10230.

Zenke, F.; and Vogels, T. P. 2021. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4): 899–925.

Zhang, H.; Dana, K.; Shi, J.; Zhang, Z.; Wang, X.; Tyagi, A.; and Agrawal, A. 2018. Context encoding for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 7151–7160.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11062–11070.

Supplementary Material

A Methods and Theoretical Analysis

Methods

For all experiments, we adopt the same surrogate gradient function to approximate the gradients of the non-differentiable activation function on the backpropagation process. The function can be formulated as follows:

$$\sigma(x) = \frac{1}{\pi}\arctan\left(\frac{\pi}{2}\alpha x\right) + \frac{1}{2}, \tag{19}$$

$$\sigma'(x) = \frac{\alpha}{2(1+(\frac{\pi}{2}\alpha x)^2)}, \tag{20}$$

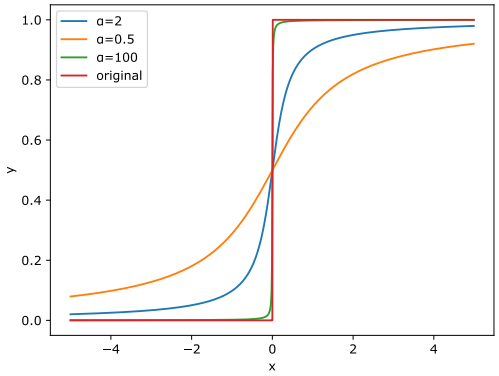
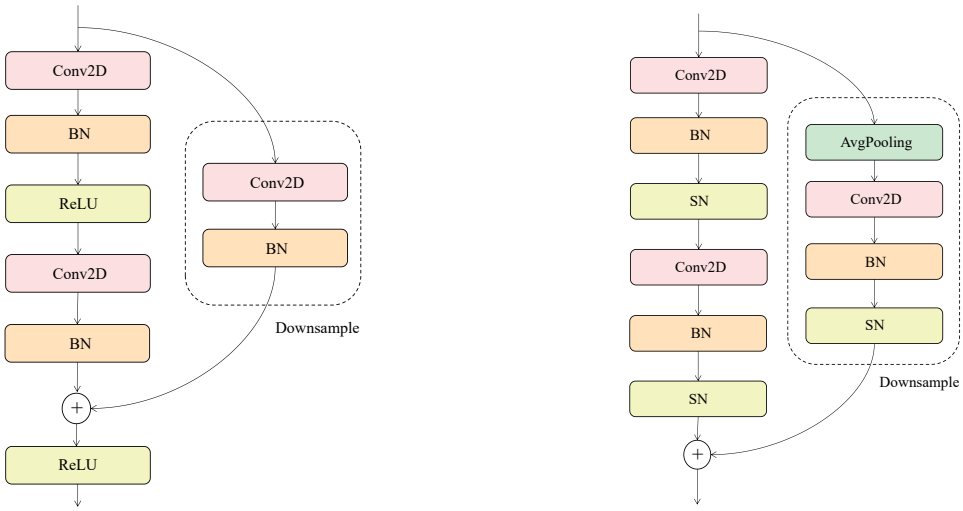


Figure 6: Visualization of the surrogate function and the original activation function.

where α is the slope parameter controlling how steep the function is. This parameter can be further set as a learnable



(a) Original block in ResNet

(b) block in CT module

Figure 7: Comparison of the original block in ResNet and the block in CT module.

parameter, but this is out of the scope of this paper. We set $\alpha = 2$, $V_{reset} = 0$ and $V_{th} = 1$ for all neurons. We detach $S[t]$ in Eq. (2) in the backward computational graph to improve performance (Zenke and Vogels 2021). The original activation function and surrogate gradient function are visualized in Fig. 6. Note that with the increase of the value of α, the surrogate function becomes steeper. The activation function works fine during the forward propagation, but the surrogate function takes its place in the computational graph. And thus, we manage to backpropagate the gradients of the non-differentiable activation function.

We adopt mixed precision training (Micikevicius et al. 2018) to accelerate training and save memory, but it may result in slightly lower accuracy. For ImageNet, we train our model on multi-GPU and adopt sync-BN technique (Zhang et al. 2018) to reduce the influence of small batch size. Additionally, we adopt Droppath (Huang et al. 2016) regularization with hyperparameter set to 0.1 for all experiments.

Theoretical Analysis

The comparison of the original block in ResNet (He et al. 2016) and the block in CT module is shown in Fig. 7. As discussed by (Fang et al. 2021a), the original design does not fit into SNN. Take the basic block without downsampling as an example:

$$X^{l+1} = ReLU(F^l(X^l) + X^l), \tag{21}$$

$$X^{l+1} = SN(F^l(X^l) + X^l), \tag{22}$$

where X^l and $F^l(\cdot)$ denote the input and the residual mapping in the l^{th} block. Eq. (21) and Eq. (22) are the ANN version and SNN version respectively. Assuming that the blocks are zero-initialized (i.e. all weights equal to zero), we have:

$$X^{l+1} = ReLU(X^l), \tag{23}$$

$$X^{l+1} = SN(X^l). \tag{24}$$

In Eq. (23), X^l has been processed by the $ReLU$ function of the $l-1^{th}$ block, and thus $X^{l+1} = X^l$, which is identity mapping. As for Eq. (24), due to the leaky property of most spiking neurons, it is hard to implement identity mapping unless we change the neural dynamics equation Eq. (1) into:

$$H[t] = V[t-1] + X[t], \tag{25}$$

and assign a small enough value to V_{th}, which enables the neurons to output a spike as soon as they receive a spike. But this compromise limits the applications of SNNs, so the original design of ResNet is not suitable for SNNs. To deal with it, (Fang et al. 2021a) propose SEW block to achieve identity mapping. We adopt ADD as element-wise operation and obtain:

$$X^{l+1} = SN(F^l(X^l)) + X^l. \tag{26}$$

Spikes are propagating through these blocks. With zero-initialized, it can easily implement identity mapping. Because the $ReLU$ function will not shrink input, this architecture does not fit into its ANN counterpart. But the output of spiking neurons is 0/1, so every time there is a downsample block, the input will be no greater than 2, which effectively tackles this issue. Besides, we modify the shortcut of downsample block as stride-2 convolutional layer with kernel size 1 may lose information (He et al. 2019). So we use a 2×2 average pooling layer with a stride of 2 to downsample the feature map and a stride-1 convolutional layer with kernel size 1 to expand the dimension. The average pooling layer integrates inputs and transfers them to the following layers. It will not hinder the transfer of information and have a minimal impact on identity mapping. Further discussions and experiments can be found in (Fang et al. 2021a).

	Spikeformer-4/5×1×3	Spikeformer-7/5×1×3	Spikeformer-7/3×2×3	Spikeformer-7L/3×2×3
block1	$\begin{pmatrix} 5\times5, 32 \\ 5\times5, 32 \end{pmatrix} \times 1$	$\begin{pmatrix} 5\times5, 64 \\ 5\times5, 64 \end{pmatrix} \times 1$	$\begin{pmatrix} 3\times3, 64 \\ 3\times3, 64 \end{pmatrix} \times 2$	$\begin{pmatrix} 3\times3, 64 \\ 3\times3, 64 \end{pmatrix} \times 2$
block2	$\begin{pmatrix} 5\times5, 64 \\ 5\times5, 64 \end{pmatrix} \times 1$	$\begin{pmatrix} 5\times5, 128 \\ 5\times5, 128 \end{pmatrix} \times 1$	$\begin{pmatrix} 3\times3, 128 \\ 3\times3, 128 \end{pmatrix} \times 2$	$\begin{pmatrix} 3\times3, 128 \\ 3\times3, 128 \end{pmatrix} \times 2$
block3	$\begin{pmatrix} 5\times5, 128 \\ 5\times5, 128 \end{pmatrix} \times 1$	$\begin{pmatrix} 5\times5, 256 \\ 5\times5, 256 \end{pmatrix} \times 1$	$\begin{pmatrix} 3\times3, 256 \\ 3\times3, 256 \end{pmatrix} \times 2$	$\begin{pmatrix} 3\times3, 256 \\ 3\times3, 256 \end{pmatrix} \times 2$
block4				$\begin{pmatrix} 3\times3, 512 \\ 3\times3, 512 \end{pmatrix} \times 2$

Table 9: Architectures of the CT module.

B Details of Experiments

DVS-Gesture

The input resolution of DVS-Gesture is 128×128. We use the same AER data pre-processing method as (Fang et al. 2021b) and do not adopt any data augmentation for this dataset. We train our model using Adam (Kingma and Ba 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 16, and apply a weight decay of 0.00015. We warm up for 20 epochs and train the model for a total of 150 epochs with a learning rate of 0.001.

DVS-CIFAR10

The input resolution and the AER data pre-processing method of DVS-CIFAR10 are the same as DVS-Gesture. Because DVS-CIFAR10 is more challenging than DVS-Gesture, we adopt additional data augmentation. First, we horizontally flip the input frames with a possibility of 0.5. Then we sample two values a and b from an integer uniform distribution $U(-5, 5)$, horizontally move the input frames for a pixels, and vertically move the input frames for b pixels. When a/b is positive, the frames are moved upwards/rightwards, and vice versa. Finally, we randomly select two values l and h from an integer uniform distribution $U(1, 16)$ as length and height, mask off an area of the frames with the selected l, and w, and adopt label-smoothing regularization (Szegedy et al. 2016) with the hyperparameter set to 0.14. We train our model using Adam (Kingma and Ba 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 32, and apply a weight decay of 0.0001. We warm up for 25 epochs and train the model for a total of 600 epochs with an initial learning rate of 0.001. Every 192 epochs, we decay the learning rate from lr to $0.1lr$.

ImageNet

We first randomly crop the images to 224×224 and horizontally flip the images with a possibility of 0.5. Auto-Augment (Cubuk et al. 2019) and label-smoothing regularization with the hyperparameter set to 0.1 are adopted for further augmentation. We train our model using SGD with a momentum of 0.9, a total batch size of 160 on 8 GPUs, and apply a weight decay of 0.0001. We warm up for 5 epochs and train the model for a total of 130 epochs with an initial learning rate of 0.01 and a cosine learning rate decay scheduler (Loshchilov and Hutter 2017). Besides, we manually decay the learning rate by 0.1 when the training accuracy tends to plateau (at epoch $\{94, 115, 118\}$ respectively).

C Model Architecture

The detailed architectures of CT module are shown in Table 9. The first convolutional layer of each block will downsample the input with a stride of 2 and each block will double the dimension until it reaches the dimension of transformer block.

D Reproducibility

Our implementations are based on PyTorch (Paszke et al. 2019) and SpikingJelly (Fang et al. 2020). Our code will be publicly available, and for reproducibility, we use the identical seed in all experiments and will detail our configurations in our code repository.