Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Improving stability and performance of spiking neural networks through enhancing temporal consistency

Dongcheng Zhao [a,1], Guobin Shen [a,c,1], Yiting Dong [a,c], Yang Li [a,d], Yi Zeng [a,b,c,d,*]

[a] Brain-inspired Cognitive Intelligence Lab, Institute of Automation, Chinese Academy of Sciences, China
[b] Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Chinese Academy of Sciences, China
[c] School of Future Technology, University of Chinese Academy of Sciences, China
[d] School of Artificial Intelligence, University of Chinese Academy of Sciences, China

ARTICLE INFO

Keywords:
Spiking neural networks
Enhancing temporal consistency
Low latency

ABSTRACT

Spiking neural networks have gained significant attention due to their brain-like information processing capabilities. The use of surrogate gradients has made it possible to train spiking neural networks with backpropagation, leading to impressive performance in various tasks. However, spiking neural networks trained with backpropagation typically approximate actual labels using the average output, often necessitating a larger simulation timestep to enhance the network's performance. This delay constraint poses a challenge to the further advancement of spiking neural networks. Current training algorithms tend to overlook the differences in output distribution at various timesteps. Particularly for neuromorphic datasets, inputs at different timesteps can cause inconsistencies in output distribution, leading to a significant deviation from the optimal direction when combining optimization directions from different moments. To tackle this issue, we have designed a method to enhance the temporal consistency of outputs at different timesteps. We have conducted experiments on static datasets such as CIFAR10, CIFAR100, and ImageNet. The results demonstrate that our algorithm can achieve comparable performance to other optimal SNN algorithms. Notably, our algorithm has achieved state-of-the-art performance on neuromorphic datasets DVS-CIFAR10 and N-Caltech101, and can achieve superior performance in the test phase with timestep $T = 1$.

1. Introduction

Spiking neural networks (SNNs), inspired by biological neural systems, exhibit unique advantages in processing spatiotemporal data. SNNs represent and transmit information through sparse, discrete spike sequences, which not only improves energy efficiency but also allows for better integration with neuromorphic chips, attracting the attention of researchers from various fields [1,2]. However, the binary information transmission method, similar to the human brain, possesses non-differentiable characteristics. It makes applying the backpropagation algorithm to directly train SNNs challenging, posing a significant obstacle in training deep SNNs.

Some researchers have attempted to incorporate brain-inspired learning principles into the modeling process of SNNs, such as spike-timing-dependent plasticity (STDP) [3,4] and short-term plasticity [5]. Although these methods have addressed the training issues of SNNs to some extent, they still have limitations in dealing with more complex structures and tasks. Currently, there are two main strategies for achieving high-performance deep SNNs: one is the conversion-based approach [6], and the other is the backpropagation (BP)-based approach [7,8]. The conversion-based method uses the activations of a pre-trained artificial neural network (ANN) as the firing rates of the SNN, allowing for a nearly lossless conversion from ANNs to SNNs. However, since this method requires firing rates to accurately simulate activation values in ANNs, it necessitates longer simulation timesteps. As well as, it cannot fully exploit the spatiotemporal information processing capabilities of SNNs. The BP-based training algorithm employs the approximate gradient of the spike firing function as a substitute for the actual gradient (surrogate gradient). This method has demonstrated superior performance across various datasets and has become the most widely adopted algorithm in the SNN domain.

Compared to ANNs, this imprecise gradient can lead to gradient vanishing or explosion when applied directly to more complex architectures. In contrast to ANNs that output results directly, SNNs need to accumulate outputs over multiple timesteps to make accurate judgments. However, this requires SNNs to use longer simulation timesteps

[*] Corresponding author at: Brain-inspired Cognitive Intelligence Lab, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
E-mail address: yi.zeng@ia.ac.cn (Y. Zeng).
[1] Dongcheng Zhao, Guobin Shen contributed equally to this work.

https://doi.org/10.1016/j.patcog.2024.111094
Received 18 September 2023; Received in revised form 29 August 2024; Accepted 20 October 2024
Available online 26 October 2024

to model more precise information, resulting in greater computational resources when deploying on edge devices. The latency caused by large simulation timestep has also become an important factor hindering the development of SNNs. When SNNs receive inputs at different timesteps, especially for neuromorphic data with rich spatiotemporal characteristics, different output patterns are generated at different timesteps. Currently, the training of spiking neural networks based on the backpropagation algorithm mainly optimizes the difference between the average membrane potential or spike firing rates of the output layer and the accurate labels. Then, it uses the output at different timesteps to guide network training in the backward process, which does not consider the difference in output distribution under different timesteps. During the training process, the optimization directions of inconsistent outputs at different moments conflict with each other, thus preventing the network from being effectively optimized. During testing, the variance in the distribution at different moments can result in the network's test results being offset by incorrect inference results at certain moments, thereby reducing overall performance. Therefore, this paper proposes a new method to enhance the temporal consistency (ETC) at different timesteps during the training process, making the training of spiking neural networks more stable. At the same time, we have verified the method on multiple datasets, and the results show that the method has achieved outstanding performance on all datasets. The main contributions of this paper are as follows:

- Through theoretical analysis, this paper reveals the limitations of existing backpropagation methods in the training of SNNs, especially in dealing with the differences in output distribution between different moments.
- We propose a novel method that enhances the temporal consistency across different moments, which improves the stability of SNN training and significantly improve the performance of SNNs under low latency.
- To validate the superiority of our proposed method, we have conducted experiments on the static datasets CIFAR10, CIFAR100, and ImageNet and the neuromorphic datasets DVS-CIFAR10, N-Caltech101 and UCF101-DVS. The results show that our algorithm achieves the best performance on neuromorphic datasets and delivers competitive performance compared to other state-of-the-art algorithms on static datasets.

2. Related work

Researchers have primarily sought to enhance BP-based SNNs through two key avenues. First, significant efforts have been directed towards improving the information transmission capabilities of spiking neural networks. A number of studies [9,10] have introduced learnable parameters, such as membrane potential constants and thresholds, into spiking neurons, thereby enabling the construction of various types of spiking neurons. This strategy significantly enhances the adaptability of neural networks to diverse information types. Furthermore, normalization techniques such as NeuNorm [11], tdBN [12], and TEBN [8] have been integrated into deep spiking neural networks, which effectively mitigate issues related to gradient vanishing and explosion, ultimately improving network performance and stability. Additionally, approaches inspired by knowledge distillation [13,14] have been employed to train SNNs, leveraging ANNs to enhance the training process of SNNs. In the works by [15,16], various attention mechanisms were introduced across different dimensions to guide SNNs in transmitting more task-relevant information, as a result of this enhancing their overall performance and adaptability.

Another researchers attempted to improve the structure of SNNs. Drawing inspiration from brain-inspired structures, LISNN [17] introduced lateral connections, while BackEISNN [18] incorporated self-feedback connections to enhance SNN's information processing capabilities. Furthermore, SewResNet [19] designed a more suitable residual module for SNNs, providing a simple method for training deep SNNs. Spikformer [20] combined self attention with SNNs, constructing a high-performance SNN structure. AutoSNN [21] and NASSNN [22] employed the neural architecture search technique to design more optimal structures for SNNs.

However, the works above all used the average membrane potential or spike firing rate at the output layer for prediction without considering the impact of the output distribution at different timesteps on performance. Reducing the differences in SNN output distribution at various timesteps is crucial for constructing stable, high-performance SNNs. While some studies have considered the performance of SNNs at each time step, TET [7] employs actual labels to constrain the output distribution at every moment. This constraint is stringent for each timestep, neglecting the relationship between the output distributions across different timesteps, limiting the model's generalization capability. TKS [23] introduces a more flexible approach by using outputs from correctly classified timesteps as teacher signals to guide the training at every timestep. However, this method requires labels to determine which moments are accurate, overlooking the valuable information provided by incorrectly classified timesteps. Our study makes full use of the outputs between different moments to guide each other, significantly utilizing the temporal information of SNNs.

3. Method

In this section, we first introduce the spiking neurons used and theoretically analyze the issue of uneven distribution at different timesteps. Ultimately, we introduce the enhancing temporal consistency constraint to standardize the output distribution at different timesteps. The whole training pipeline of our method is shown in Fig. 1.

3.1. Spiking neuron model

The leaky integrate-and-fire (LIF) neuron, as the most commonly used neuron model in deep spiking neural networks, describes the complex dynamics of biological neurons with a relatively simple differential equation as shown in Eq. (1):

$$\begin{aligned} \tau_m \frac{dV}{dt} &= -V + RI \quad V \le V_{th} \\ S &= H(V - V_{th}) \end{aligned} \tag{1}$$

V represents the membrane potential, $\tau_m = RC$ is the membrane potential time constant, $I = \sum_j W_{ij} S_j$ denotes the input current obtained by aggregating presynaptic spikes, W denotes the connection weight from pre-synaptic to post-synaptic neurons, and H refers to the step function for spike emission. When the membrane potential surpasses the threshold V_{th}, the neuron emits a spike S and resets to the resting potential V_r.

In this study, we set V_{th} to 0.5, τ_m to 2, R to 1, and the resting potential V_r to 0. By using the first-order Euler method, we obtain the discretized representation of Eq. (1) as shown in Eq. (2):

$$V_{t+1} = (1 - \frac{1}{\tau_m})V_t + \frac{1}{\tau_m} I_t \tag{2}$$

In order to use the backpropagation algorithm for network training, we employ surrogate gradients to approximate the gradients of the spike firing function, as follows:

$$\frac{\partial H}{\partial V_t} = \begin{cases} 0, & |V_t - V_{th}| > \frac{1}{a} \\ -a^2|V_t - V_{th}| + a, & |V_t - V_{th}| \le \frac{1}{a} \end{cases} \tag{3}$$

a is a hyperparameter used to control the shape of the surrogate gradient. In this study, we set a to 2.

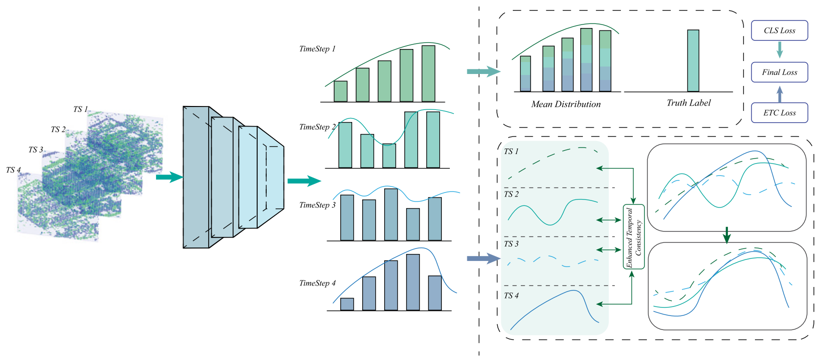

Fig. 1. The whole training pipeline of our model. SNNs receive input data at various timesteps and generate corresponding outputs at each timestep. The ETC constraint helps ensure output distribution consistency across different timesteps.

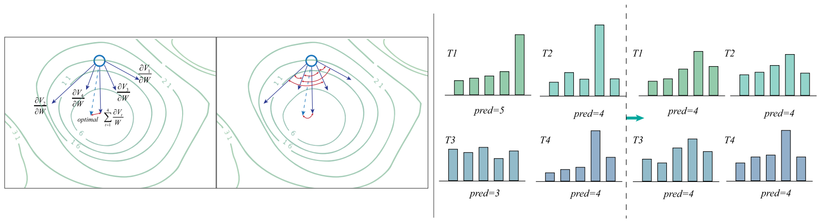

Fig. 2. The training and inference procedure compared our method with the traditional methods. For traditional methods, consistency in the distribution across different timesteps can cause discrepancies in the optimization direction and lead to misjudgments during inference.

3.2. The training and inference procedures of SNNs

For the traning and inference of SNNs, we adopt the commonly used direct input encoding strategy to encode the input. Meanwhile, the output for each timestep is derived from the undecayed membrane potential of the final layer. Unlike traditional artificial neural networks, SNNs adjust their weights during training based on outputs at multiple timesteps. During the inference of the network, the final output is calculated by weighting the outputs at various timesteps. For deep SNNs, the most widely used approach is to employ the average membrane potential of the neurons in the last layer as the final output, resulting in more efficient and accurate inference.

We represent the average membrane potential of the last layer as $O_{\text{mean}} = \frac{1}{T}\sum_{t=1}^{T} V_t$, where T is the total number of timesteps and V_t is the membrane potential at timestep t. The softmax function is applied to convert the average membrane potential into output probabilities $P_{\text{mean}} = \text{softmax}(O_{\text{mean}})$. During the network training process, we use the cross-entropy loss to minimize the difference between the actual outputs P_{mean} and expected outputs y as shown in Eq. (4).

$$L_{CE} = -\sum_{i=1}^{C} y_i \cdot \log(P_{\text{mean}}) \tag{4}$$

C is the number of categories. By applying the chain rule, we can compute the gradient of the loss function concerning the network parameters W.

$$\begin{aligned}\frac{\partial L_{CE}}{\partial W} &= \frac{\partial L_{CE}}{\partial O_{\text{mean}}}\frac{\partial O_{\text{mean}}}{\partial V_t}\frac{\partial V_t}{\partial W} = \frac{1}{T}\sum_{t=1}^{T}(P_{\text{mean}} - y)\frac{\partial V_t}{\partial W} \\ &= (P_{\text{mean}} - y)\frac{1}{T}\sum_{t=1}^{T}\frac{\partial V_t}{\partial W}\end{aligned} \tag{5}$$

As previously mentioned, there are inherent dependencies in the membrane potentials of SNNs across different time steps. Thus, we can get:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\frac{\partial V_t}{\partial W} &= \frac{1}{T}[\frac{\partial V_1}{\partial W} + (\frac{\partial V_2^D}{\partial W} + \frac{\partial V_2}{\partial V_1}\frac{\partial V_1}{\partial W}) + \cdots \\ &\quad + (\frac{\partial V_T^D}{\partial W} + \frac{\partial V_T}{\partial V_{T-1}}\frac{\partial V_{T-1}^D}{\partial W} + \cdots + \prod_{t=2}^{T}\frac{\partial V_t}{\partial V_{t-1}}\frac{\partial V_1}{\partial W})]\end{aligned} \tag{6}$$

where V_t^D means detach the gradient of V_t through the temporal dimension. Since the output layer of the network utilizes a membrane potential without decay, that is $\frac{\partial V_t}{\partial V_{t-1}} = 1$, Eq. (6) can be reformulated as:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{\partial V_t}{\partial W} = \frac{1}{T}[\frac{\partial V_1^D}{\partial W} + (\frac{\partial V_2^D}{\partial W} + \frac{\partial V_1^D}{\partial W}) + \cdots + \sum_{t=1}^{T}\frac{\partial V_t^D}{\partial W}] \tag{7}$$

Then, Eq. (5) can be formulated as

$$\frac{\partial L_{CE}}{\partial W} = (P_{\text{mean}} - y)\frac{1}{T}[\frac{\partial V_1^D}{\partial W} + (\frac{\partial V_2^D}{\partial W} + \frac{\partial V_1^D}{\partial W}) + \cdots + \sum_{t=1}^{T}\frac{\partial V_t^D}{\partial W}] \tag{8}$$

For ANNs, we use O to denote the final output and P to represent the probability distribution after the softmax operation. Using the same loss function, we can also obtain the gradient of the loss concerning the network parameters:

$$\frac{\partial L_{CE}}{\partial W} = \frac{\partial L_{CE}}{\partial O}\frac{\partial O}{\partial W} = [P - y]\frac{\partial O}{\partial W} \tag{9}$$

As shown in Eqs. (8) and (9), the final output determines the partial derivative of the weight. In minimizing the loss, the output O of the ANN will gradually approach the actual label. At this time, using the partial derivative of O concerning W will more accurately control the optimization direction of the ANN. However, the optimization direction of the weights in SNN depends on the direction of the partial derivative of the output V_t concerning the weight at each moment. SNN optimizes the distance between the average membrane potential and the actual label. As shown in Fig. 2, when the optimization directions at different moments are inconsistent or even significantly different, there will be a severe mismatch, which significantly interferes with the optimization direction of the spiking neural network. Simultaneously, as shown on the right side of Fig. 2, during the inference phase of the network, inconsistencies in the distribution at different timesteps can lead to overall network results being skewed by erroneous results, thereby leading to a decline in performance.

3.3. Enhancing temporal consistency

As discussed above, the performance degradation of SNNs is due to the mismatch of output distributions at different timesteps. In order to enhance the consistency between different timesteps, we propose an enhancing temporal consistency constraint, aiming to make the distributions at each timestep as similar as possible. First, we define the output probability distribution at each timestep P_t^i as shown in Eq. (10):

$$P_t^i(V_t;\tau) = softmax(V_t^i;\tau) = \frac{exp(V_t^i/\tau)}{\sum_j exp(V_t^j/\tau)} \tag{10}$$

The temperature parameter τ controls the smoothness of the model's output distribution, which is more conducive to learning the relationships between different categories [24]. Here we set $\tau = 4$. After obtaining the output distributions at different timesteps, we aim to minimize the distribution gap between the output P_t at time t and the outputs at other timesteps, here we use the Kullback–Leibler (KL) divergence. The loss function, as shown in Eq. (11), includes a τ^2 factor to balance the influence of the temperature parameter on the gradient, thereby ensuring stable training dynamics.

$$\begin{aligned} L_{ETC}^t &= \frac{\tau^2}{T-1}\sum_{m=1,m\neq t}^{T} KL(P_m \parallel P_t) = \frac{1}{T-1}\sum_{m=1,m\neq t}^{T}\sum_{i=1}^{C} P_m^i log\frac{P_m^i}{P_t^i} \\ &= \frac{1}{T-1}\sum_{m=1,m\neq t}^{T}\sum_{i=1}^{C}(P_m^i log P_m^i - P_m^i log P_t^i) \end{aligned} \tag{11}$$

To avoid model collapse, we do not propagate the gradient through P_m. As a result, the final loss can be represented as shown in Eq. (12):

$$L_{ETC} = -\frac{\tau^2}{T}\frac{1}{T-1}\sum_{t=1}^{T}\sum_{m=1,m\neq t}^{T}\sum_{i=1}^{C} P_m^i log P_t^i \tag{12}$$

Thus, the final loss function can be written as a dynamic combination of cross-entropy loss and ETC loss, as shown in Eq. (13).

$$L_{all} = L_{CE} + \lambda L_{ETC} \tag{13}$$

λ is the weight constraint term to control the influence of ETC loss on overall loss. The entire training procedure of the algorithm is depicted in Algorithm 1. Here we set $\lambda = 1$. We can obtain the partial derivative of the total loss concerning the weights $\frac{\partial L_{all}}{\partial W}$:

$$\begin{aligned} \frac{\partial L_{all}}{\partial W} &= \frac{\partial L_{CE}}{\partial W} + \lambda\frac{\partial L_{ETC}}{\partial W} \\ &= \frac{1}{T}\sum_{t=1}^{T}[P_{\text{mean}} - y]\frac{\partial V_t}{\partial W} + \lambda\tau^2\frac{1}{T}\frac{1}{T-1}\sum_{t=1}^{T}\sum_{m=1,m\neq t}^{T}(P_t - P_m)\frac{\partial V_t}{\partial W} \end{aligned} \tag{14}$$

As shown in Eq. (14), the first term ensures that the average output is close to the actual target, while the second term ensures the consistency of the output distribution at each timestep. Combining these two loss terms can make the output at each moment as accurate as possible, thus better guiding the optimization direction of the output for the weights at each moment. By using the output of different timesteps to approximate each other and correcting erroneous prediction moments, this combination of loss functions can also prevent overconfident predictions, thereby further improving the model's generalization ability on the test dataset. Moreover, this constraint can be considered a self-distillation process of the model. It uses the dark knowledge of the model's output at other timesteps to provide soft labels for each timestep, optimizing the output at each timestep and fully utilizing the temporal information of SNNs.

4. Experiments

We develop the SNN code based on the open-source framework BrainCog with NVIDIA A100 graphic processing unit (GPU), employing the AdamW optimizer with a weight decay setting 0.0001. The learning rate is set to 0.001 with the cosine annealing strategy. The batch size is set to 128. In our experiments, we set the total training epochs to 600. In order to demonstrate the superiority of our algorithm, we conduct experiments on multiple datasets, including static datasets such as CIFAR10, CIFAR100, and ImageNet, as well as neuromorphic datasets like DVS-CIFAR10 and N-Caltech101. The experiments are repeated five times randomly. We have reported the mean and standard deviation of the corresponding performance.

4.1. Results on the CIFAR10 dataset

CIFAR10 is a dataset encompassing 10 categories, with each image having a size of 32 × 32. The dataset comprises 50,000 training images and 10,000 test images. To ensure the comparability of our results with previous studies, we first adopt the ResNet-19 architecture, setting our simulation time to 4. As Table 1 indicates, our approach achieves an accuracy of 95.87%. Our ETC sets a new performance benchmark, surpassing the TET with simulation length 6 by 1.3% . When compared to TEBN, our method marks a 0.2% improvement. Moreover, to verify the generalization and efficiency of our method, we conduct experiments with the SEW-ResNet18 structure [19], which has fewer parameters. We undertake comparative experiments with a simulation length of 2,4,6,8. Notably, at the simulation length of 6, our model achieves 95.73%, surpassing all other algorithms.

4.2. Results on the CIFAR100 dataset

Further, we conduct experiments on the CIFAR100 dataset. The size and quantity of CIFAR100 are consistent with CIFAR10, encompassing 100 classes. As shown in Table 2, we initially chose ResNet-19 as our benchmark network for testing. The results indicate that at a simulation length of 4, our method achieves a performance of 79.47%, surpassing the current best performances. Notably, compared to the TET algorithm, even with a shorter simulation length, our performance still exceeds theirs by approximately 5%. Compared to other recently proposed optimal algorithms, our approach outperforms GLIF by 1.4% and TEBN by 0.8%. Furthermore, we carry out experimental validation on the SEW-ResNet-18 network structure. The results are encouraging, demonstrating that our method shows comparable performance with the current state-of-the-art even on a more minor network architecture.

Algorithm 1 Improving Stability and Performance of Spiking Neural Networks through Enhancing Temporal Consistency

```
Require:
    Training dataset D = $\{(x_i, y_i)\}_{i=1}^N$, SNN Model $F$, Simulation Length $T$, Temperature Parameter $\tau$
Ensure:
    A more stable and high performance spiking neural network model;
1: for each mini-batch training data $D_i = \{x_i, y_i\}$ do
2:     Encode the inputs $x_i$ as $\{x_i^1, \dots, x_i^t, \dots, x_i^T\}$;
3:     Compute the SNN output as $V_t = f(x_i^t)$, and get the final output as $O_{mean} = \frac{1}{T}\sum_{T=1}^{T} V_t$;
4:     Compute the output probability distribution at each timestep $P_t^i$ as shown in Eq. (10);
5:     Compute the ETC loss as shown in Eq. (12);
6:     Compute the final loss using Eq. (13);
7:     Update weights with surrogate gradient-based backpropagation algorithm;
8: end for
```

Table 1
Compare with existing works on CIFAR10 dataset.

Dataset	Model	Architecture	Step	Accuracy
CIFAR10	Diet-SNN [25]	ResNet-20	10	92.54
	NeuNorm [11]	CIFARNet	12	90.53
	BPSTA [26]	7-layer-CNN	8	92.15
	NASSNN [22]	NAS	5	92.73
	AutoSNN [21]	NAS	16	93.15
	tdBN [12]	ResNet-19	6	93.16
	PLIF [9]	PLIFNet	8	93.5
	TET [7]	ResNet-19	6	94.50
	GLIF [27]	ResNet-19	6	95.03
	TKS [23]	ResNet-19	4	95.3
	Rec-Dis [28]	ResNet-19	6	95.55
	TEBN [8]	ResNet-19	6	95.60
	Our Method	SEW-ResNet-18	2	94.65 ± 0.08
		SEW-ResNet-18	4	95.4 ± 0.07
		SEW-ResNet-18	6	95.73 ± 0.02
		SEW-ResNet-18	8	95.84 ± 0.03
		ResNet-19	4	**95.87** ± **0.04**

Table 2
Compare with existing works on CIFAR100 dataset.

Dataset	Model	Architecture	Step	Accuracy
CIFAR100	[29]	ResNet-18	8	75.67
	[30]	VGG-11	125	67.87
	Diet-SNN [25]	ResNet-20	5	64.07
	BPSTA [26]	ResNet34	8	69.32
	AutoSNN [21]	NAS	16	69.16
	NASSNN [22]	NAS	5	73.04
	TET [7]	ResNet-19	6	74.72
	Rec-Dis [28]	ResNet-19	4	74.10
	TKS [23]	ResNet-19	4	76.2
	GLIF [27]	ResNet-19	6	77.35
	TEBN [8]	ResNet-19	6	78.76
	Our Method	SEW-ResNet-18	2	75.96 ± 0.24
		SEW-ResNet-18	4	77.65 ± 0.13
		SEW-ResNet-18	6	78.25 ± 0.11
		SEW-ResNet-18	8	78.32 ± 0.07
		ResNet-19	4	**79.47** ± **0.22**

Table 3
Compare with existing works on ImageNet dataset.

Dataset	Model	Architecture	Step	Accuracy
ImageNet	[29]	ResNet-34	16	59.35
	[30]	ResNet-34	250	61.48
	tdBN [12]	Spiking-ResNet-34	6	63.72
	SEW [19]	SEW-ResNet-34	4	67.04
	Rec-Dis [28]	ResNet-34	6	67.33
	TET [7]	SEW-ResNet-34	4	68.00
	TEBN [8]	SEW-ResNet-34	4	68.28
	GLIF [27]	ResNet-34	6	69.09
	TKS [23]	SEW-ResNet-34	4	69.6
	Our Method	SEW-ResNet-18	4	63.70
		SEW-ResNet-34	4	68.54
		SEW-ResNet-34	6	**69.64**

4.3. Results on the ImageNet dataset

We conduct extensive validation on the ImageNet dataset. Compared to CIFAR10 and CIFAR100, ImageNet is more intricate, aggregating over a million images across 1,000 categories. As presented in Table 3, our experiments are conducted on the SEW-ResNet-18 and SEW-ResNet-34 architectures. When compared with TET and TEBN, our performance consistently surpassed theirs. With a simulation duration set to 4, our methodology marks a 1.5% accuracy improvement over the original SEW. Furthermore, when juxtaposed with the GLIF approach, our results showcase a lead of 0.6%. All of these findings underline the superior efficacy of our algorithm.

4.4. Results on DVS-CIFAR10 dataset

Compared to static datasets, neuromorphic datasets reveal richer spatiotemporal features, thereby better highlighting the advantages of spiking neural networks. The DVS-CIFAR10 dataset transforms the 10,000 frame-based images from CIFAR10 into 10,000 event streams using a Dynamic Vision Sensor (DVS). In our study, we first resize the input samples to a fixed size of 48×48. We adopt the VGGSNN structure (64C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-AP2-512C3-512C3-AP2-FC) used in TET. As shown in Table 4, compared to other superior SNN algorithms, our model has achieved the best performance. In comparison with Rec-Dis, our model has shown a 13% improvement. When compared to the TET algorithm using the same

Table 4
Compare with existing works on DVS-CIFAR10 dataset.

Dataset	Model	Architecture	Step	Accuracy
DVS-CIFAR10	tdBN [12]	ResNet-19	10	67.8
	LIAF [31]	LIAF-Net	10	71.70
	LIAF [31]	LIAF-Net	10	70.40
	AutoSNN [21]	NAS	16	72.50
	BPSTA [26]	5-layer-CNN	16	78.95
	Rec-Dis [28]	ResNet-19	10	72.42
	TET [7]	VGGSNN	10	83.17
	TEBN [8]	VGGSNN	10	84.90
	TKS [23]	VGGSNN[a]	10	85.3
	Our Method	VGGSNN	10	**85.35 ± 0.40**
		VGGSNN[a]	10	**85.95 ± 0.43**

[a] Indicates using a model with tdBN [12].

Table 5
Compare with existing works on N-Caltech101 dataset.

Dataset	Model	Architecture	Step	Accuracy
N-Caltech101	ConvertSNN [32]	VGG11	20	55.0
	Dart [33]	N/A	N/A	66.8
	TCJA [15]	TCJAnet	14	78.5
	NDA [34]	ResNet-19	10	78.6
	EventMix [35]	ResNet-18	10	79.5
	TKS [23]	VGGSNN[a]	10	84.1
	Our Method	VGGSNN	10	**83.33 ± 0.41**
		VGGSNN[a]	10	**85.53 ± 0.09**

[a] Indicates using a model with tdBN [12].

structure, our model demonstrates a 2.8% performance increase, and compared to TEBN, our model also exhibits a 1% enhancement in performance.

4.5. Results on N-Caltech101 dataset

The N-Caltech101 dataset is a neuromorphic version of the Caltech101 dataset, encompassing 101 categories. We resize the dataset to 48 × 48 and conduct experiments using the VGGSNN network architecture. As shown in Table 5, we have achieved the accuracy of 85.53%, reaching state-of-the-art performance. Compared to the TCJA with the attention mechanism, our model surpasses it by 7%. In comparison with TKS, our model demonstrates a 1.4% improvement in performance. Meanwhile, our performance has significantly surpassed that of the EventMix and NDA algorithms, which use data augmentation techniques. Due to the characteristic of neuromorphic datasets where inputs at different times vary, the output distribution also changes. In such cases, introducing the method we propose in this paper can better standardize the membrane potential distribution at different timesteps, thereby significantly enhancing the network's performance.

4.6. Results on UCF101-DVS dataset

The UCF101-DVS dataset, a neuromorphic adaptation of the UCF101 action recognition dataset, presents a heightened challenge due to its intricate temporal dynamics and complex motion patterns. As shown in Table 6, our proposed ETC method achieved an accuracy of 61.3%, surpassing all benchmark models, including EventMix [35] and RM-Res-SNN-18 [36]. Notably, our method demonstrates a significant 5.0% improvement over TKS [23], underscoring its robustness in handling the temporal variability inherent to this dataset. These results further affirm the effectiveness of the ETC approach in enhancing network performance on datasets characterized by strong temporal features.

Table 6
Compare with existing works on UCF101-DVS dataset.

Dataset	Model	Accuracy
UCF101-DVS	TKS [23]	56.3
	Res-SNN-18 [19]	57.8
	RM-Res-SNN-18 [36]	58.5
	EventMix [35]	60.6
	ETC(Ours)	**61.3**

4.7. Evaluation on spiking transformer architecture

To further demonstrate the generalizability of the proposed ETC method, we extended our experiments to the Spikformer [20], a Spiking Transformer architecture. We evaluated its performance on three neuromorphic datasets: DVS-CIFAR10, N-Caltech101, and UCF101-DVS. As illustrated in Fig. 3, the ETC method consistently outperformed the baseline across all datasets. These results highlight the robustness and adaptability of the ETC method, affirming its effectiveness across diverse neural network architectures, including the Spiking Transformer.

5. Discussion

5.1. Ablation studies

To verify the effectiveness of our algorithm, we conduct ablation experiments on the DVS-CIFAR10 and N-Caltech101 datasets. We visualize the accuracy curves of the Base and ETC algorithms on two datasets. As shown in Fig. 4, after introducing the ETC module, there is a significant improvement in both the DVS-CIFAR10 and N-Caltech101 datasets. Without adding the ETC module, the accuracy of the N-Caltech101 dataset is only 78%. However, after adding the ETC module with $\lambda = 1$ and $\tau = 4$, the network performance improves to 83%, an increase of 5%. Under the same settings, the performance of the DVS-CIFAR10 dataset is also improved by 2.5%.

Further, we visualize the confusion matrix for the test dataset of the DVS-CIFAR10. Numbers from 0 to 9 represent airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, respectively. As shown

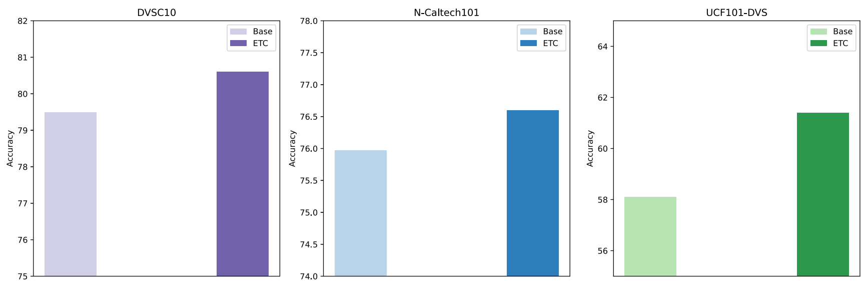

Fig. 3. Performance comparison of ETC and Base on DVS-CIFAR10, N-Caltech101 and UCF101-DVS datasets using spiking transformer architecture.

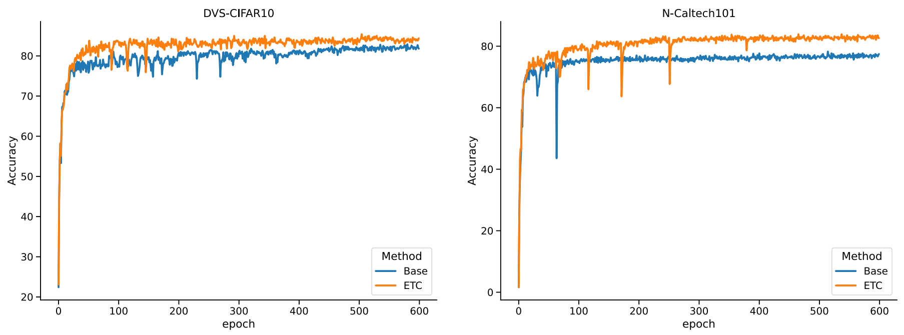

Fig. 4. Accuracy curves for the Base and ETC algorithms on the DVS-CIFAR10 (left) and N-Caltech101 (right) datasets.

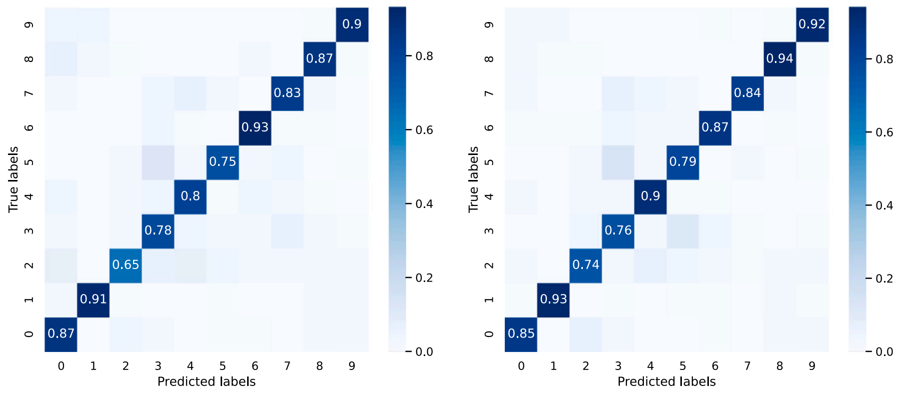

Fig. 5. Confuse Matrix for the Base and ETC algorithms on the DVS-CIFAR10 dataset.

in Fig. 5, the original algorithm often confuses between similar classes like bird and airplane, or cat and dog. However, after incorporating the ETC algorithm, there is a significant improvement in performance for these particular classes.

To further validate the robustness of our approach, we conducted experiments across various time steps T on the DVS-CIFAR10 dataset. As shown in Fig. 6, the ETC algorithm consistently outperforms the baseline at each tested time step, including $T = 2$, $T = 4$, $T = 6$, $T = 8$, $T = 10$, and $T = 12$. This demonstrates the stability and generalizability of the ETC algorithm, ensuring strong and reliable performance across different temporal configurations. Such consistent results underscore the robustness of our method and its potential to enhance spiking neural networks in dynamic environments.

5.2. Parameter sensitivity analysis

It is worth noting that the hyperparameter τ controls the smoothness of the output at different moments. If τ is too large, the output will be too smooth to show significant differences. Conversely, if τ is too small, the output will be overly confident and unable to reflect the relationships between different categories. Furthermore, the λ parameter controls the influence of the ETC module on the final loss. If λ is too large or too small, it cannot effectively guide the loss function to update the weights. To better illustrate the impact of these hyperparameters, we test the performance of the ETC algorithm under different hyperparameter settings. As shown in Fig. 7, when both λ and τ are very small, the performance of the network is extremely low. The accuracy for DVS-CIFAR10 is only 82.2%, while for N-Caltech101, it is merely 80.11%. With $\tau = 4$ and $\lambda = 2$, DVS-CIFAR10 achieves its highest accuracy. Meanwhile, when $\tau = 4$ and $\lambda = 1$, N-Caltech101 reaches its peak accuracy.

5.3. Temporal consistency verification

In addition to performance, latency is a crucial factor that constrains the development of SNNs. When the output distribution at each moment becomes more consistent, we can achieve higher accuracy in the testing phase using only a shorter simulation length. We have validated the results on the DVS-CIFAR10 and N-Caltech101 datasets. During the training phase, we use the simulation timestep T=10, while in the testing phase, we conduct separate experiments for different timesteps. As shown in Fig. 8, with a simulation step size of 1, the accuracy of

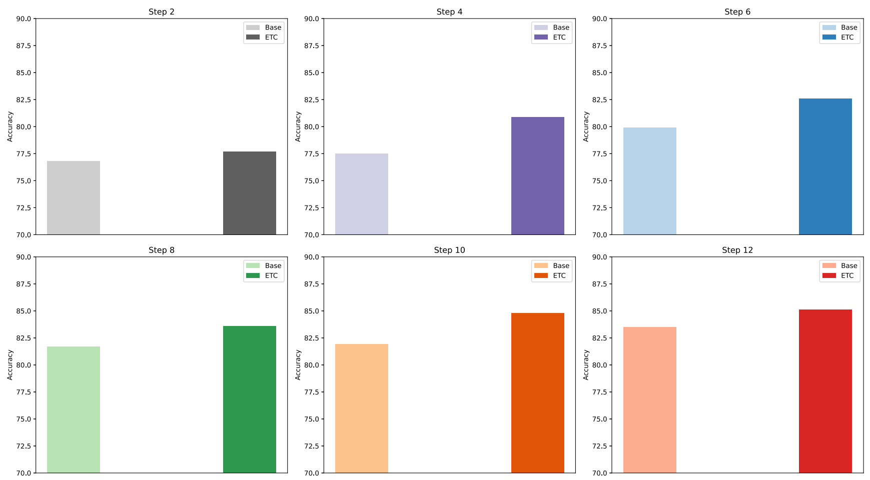

Fig. 6. Performance comparison of ETC and Base aross different timesteps on DVS-CIFAR10 dataset.

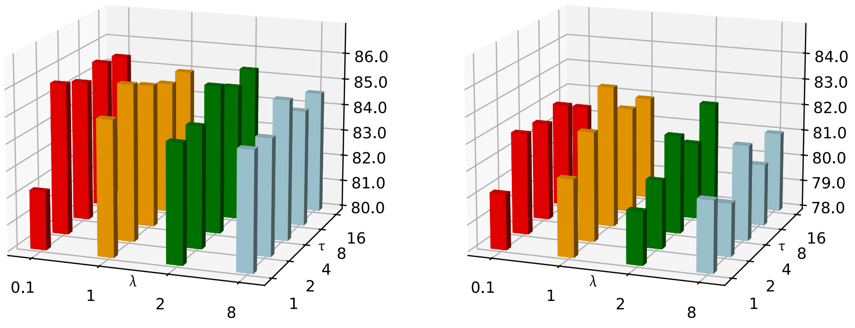

Fig. 7. The parameter sensitivity analysis for λ and τ on the DVS-CIFAR10 (left) and N-Caltech101 (right) datasets.

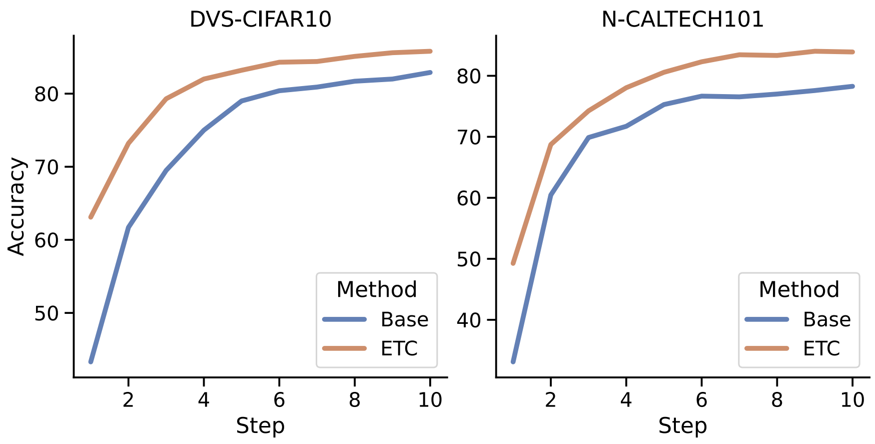

Fig. 8. Test accuracy in different timesteps on DVS-CIFAR10 and N-Caltech101, the model is trained at timestep 10.

the conventional algorithm is only 43.3% for the DVS-CIFAR10 dataset. In contrast, our ETC algorithm achieves an accuracy of 63.1%, an improvement of about 20%. At a step size of 5, the ETC algorithm already surpasses the performance of the traditional algorithm at a step size of 10, significantly reducing the network's latency. For the N-Caltech101 dataset, when the step size is 1, our algorithm has improved by 16% compared to the baseline. At a step size of 5, the accuracy of the ETC algorithm has reached 80.57%, which is 2% higher than the baseline at a step size of 10.

Meanwhile, we have visualized the outputs at different timesteps in the final layer. As shown in Fig. 9, we visualize the distribution of the outputs at different timesteps for the samples in the DVS-CIFAR10 dataset. It can be observed that the output distributions of the traditional algorithm vary significantly at different timesteps. For instance, the lsample's output distribution is accurate in the early moments, but from the sixth moment onwards, the distribution begins to fluctuate. This led to a sample that should have been categorized as 8 being incorrectly predicted as 6, 8, 1, 8, 6. In contrast, the output distribution of the ETC algorithm is much more consistent. By enhancing the consistency of the temporal distribution, the ETC algorithm has dramatically improved the accuracy of the overall distribution of outputs at each moment. This allows us to achieve high-precision predictions in the testing phase with fewer simulation steps. This characteristic greatly facilitates the deployment of SNNs on various edge devices.

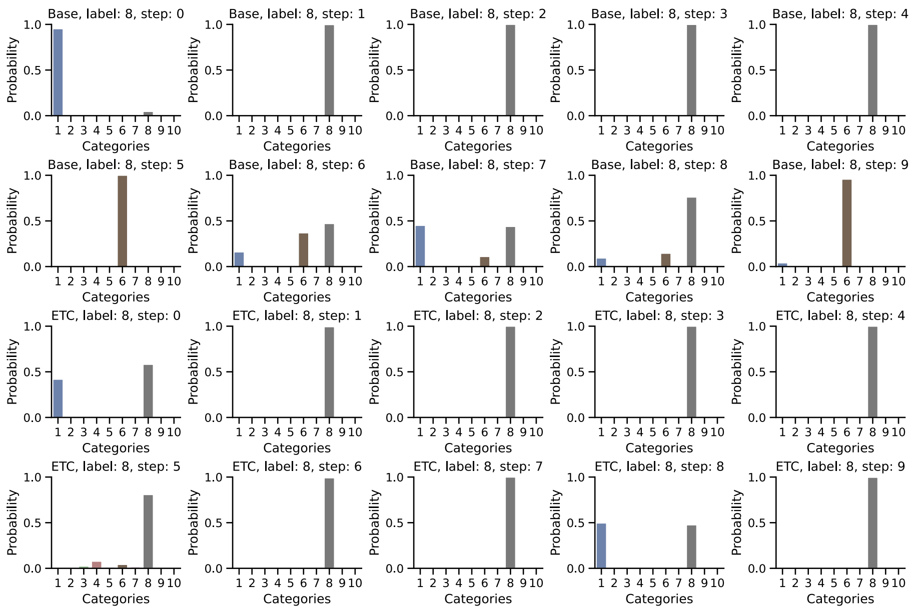

Fig. 9. The comparison between the base method and our method on output distribution at different timesteps for the sample in DVS-CIAFR10.

6. Conclusion

Spiking neural networks have emerged as a powerful paradigm, emulating the temporal dynamics of biological neural systems. Despite their promise, the challenge of aligning output distributions over varying timesteps has limited their full potential. Traditional approaches, which rely on longer simulation steps to average outputs for label approximation, have inadvertently introduced latency issues, constraining the deployment of SNNs in time-sensitive applications. In our research, we addressed this critical bottleneck by developing a method to enhance temporal consistency within SNNs. Unlike existing training algorithms that often neglect the impact of output distribution variations across different timesteps, our approach directly targets these discrepancies. By harmonizing the outputs across timesteps, our method not only improves performance but also reduces the need for extended simulation durations, thereby lowering latency. The efficacy of our method was validated across a spectrum of datasets, including both static and neuromorphic benchmarks. Looking ahead, while our approach has significantly improved temporal consistency, there are still areas for further refinement. Specifically, during the inference phase, the model does not yet fully leverage the information from different timesteps to decode a more convincing result. To address this, future work will focus on integrating biologically inspired mechanisms for encoding and decoding information [37,38], aiming to enhance the utilization of temporal information. By refining these processes, we believe SNNs can be further optimized for real-time processing, thereby pushing the boundaries of their performance in dynamic environments.

CRediT authorship contribution statement

Dongcheng Zhao: Conceptualization, Methodology, Software, Writing – original draft. **Guobin Shen:** Methodology, Software, Writing – review & editing. **Yiting Dong:** Visualization, Writing – review & editing. **Yang Li:** Visualization, Writing – review & editing. **Yi Zeng:** Conceptualization, Methodology, Supervision, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was financially supported by a funding from Institute of Automation, Chinese Academy of Sciences (Grant No. E411230101).

Data availability

Data will be made available on request.

References

[1] Y. Zeng, D. Zhao, F. Zhao, G. Shen, Y. Dong, E. Lu, Q. Zhang, Y. Sun, Q. Liang, Y. Zhao, et al., BrainCog: A spiking neural network based brain-inspired cognitive intelligence engine for brain-inspired AI and brain simulation, 2022, arXiv preprint arXiv:2207.08533.

[2] M. Zhang, J. Wang, J. Wu, A. Belatreche, B. Amornpaisannon, Z. Zhang, V.P.K. Miriyala, H. Qu, Y. Chua, T.E. Carlson, et al., Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks, IEEE Trans. Neural Netw. Learn. Syst. 33 (5) (2021) 1947–1958.

[3] P. Falez, P. Tirilly, I.M. Bilasco, P. Devienne, P. Boulet, Unsupervised visual feature learning with spike-timing-dependent plasticity: How far are we from traditional feature learning approaches? Pattern Recognit. 93 (2019) 418–429.

[4] Y. Dong, D. Zhao, Y. Li, Y. Zeng, An unsupervised STDP-based spiking neural network inspired by biologically plausible learning rules and connections, Neural Netw. 165 (2023) 799–808.

[5] Y. Zheng, L. Zheng, Z. Yu, B. Shi, Y. Tian, T. Huang, High-speed image reconstruction through short-term plasticity for spiking cameras, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 6358–6367.

[6] Y. Li, Y. Zeng, Efficient and accurate conversion of spiking neural network with burst spikes, 2022, arXiv preprint arXiv:2204.13271.

[7] S. Deng, Y. Li, S. Zhang, S. Gu, Temporal efficient training of spiking neural network via gradient re-weighting, in: International Conference on Learning Representations, 2022.

[8] C. Duan, J. Ding, S. Chen, Z. Yu, T. Huang, Temporal effective batch normalization in spiking neural networks, Adv. Neural Inf. Process. Syst. 35 (2022) 34377–34390.

[9] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, Y. Tian, Incorporating learnable membrane time constant to enhance learning of spiking neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2661–2671.

[10] J. Ding, B. Dong, F. Heide, Y. Ding, Y. Zhou, B. Yin, X. Yang, Biologically inspired dynamic thresholds for spiking neural networks, in: Advances in Neural Information Processing Systems, 2023.

[11] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, L. Shi, Direct training for spiking neural networks: Faster, larger, better, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, (01) 2019, pp. 1311–1318.

[12] H. Zheng, Y. Wu, L. Deng, Y. Hu, G. Li, Going deeper with directly-trained larger spiking neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, (12) 2021, pp. 11062–11070.

[13] Y. Guo, W. Peng, Y. Chen, L. Zhang, X. Liu, X. Huang, Z. Ma, Joint a-snn: Joint training of artificial and spiking neural networks via self-distillation and weight factorization, Pattern Recognit. 142 (2023) 109639.

[14] Q. Xu, Y. Li, J. Shen, J.K. Liu, H. Tang, G. Pan, Constructing deep spiking neural networks from artificial neural networks with knowledge distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 7886–7895.

[15] R.-J. Zhu, M. Zhang, Q. Zhao, H. Deng, Y. Duan, L.-J. Deng, Tcja-snn: Temporal-channel joint attention for spiking neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2024).

[16] M. Yao, G. Zhao, H. Zhang, Y. Hu, L. Deng, Y. Tian, B. Xu, G. Li, Attention spiking neural networks, IEEE Trans. Pattern Anal. Mach. Intell. (2023).

[17] X. Cheng, Y. Hao, J. Xu, B. Xu, LISNN: Improving spiking neural networks with lateral interactions for robust object recognition., in: IJCAI, 2020, pp. 1519–1525.

[18] D. Zhao, Y. Zeng, Y. Li, BackEISNN: A deep spiking neural network with adaptive self-feedback and balanced excitatory–inhibitory neurons, Neural Netw. 154 (2022) 68–77.

[19] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, Y. Tian, Deep residual learning in spiking neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 21056–21069.

[20] Z. Zhou, Y. Zhu, C. He, Y. Wang, Y. Shuicheng, Y. Tian, L. Yuan, Spikformer: When spiking neural network meets transformer, in: The Eleventh International Conference on Learning Representations, 2023.

[21] B. Na, J. Mok, S. Park, D. Lee, H. Choe, S. Yoon, Autosnn: towards energy-efficient spiking neural networks, in: International Conference on Machine Learning, PMLR, 2022, pp. 16253–16269.

[22] Y. Kim, Y. Li, H. Park, Y. Venkatesha, P. Panda, Neural architecture search for spiking neural networks, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV, Springer, 2022, pp. 36–56.

[23] Y. Dong, D. Zhao, Y. Zeng, Temporal knowledge sharing enable spiking neural network learning from past and future, IEEE Trans. Artif. Intell. (2024).

[24] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, arXiv preprint arXiv:1503.02531.

[25] N. Rathi, K. Roy, Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks, 2020, arXiv preprint arXiv:2008.03658.

[26] G. Shen, D. Zhao, Y. Zeng, Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks, Patterns (2022) 100522.

[27] X. Yao, F. Li, Z. Mo, J. Cheng, GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks, in: Advances in Neural Information Processing Systems.

[28] Y. Guo, X. Tong, Y. Chen, L. Zhang, X. Liu, Z. Ma, X. Huang, RecDis-SNN: Rectifying membrane potential distribution for directly training spiking neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 326–335.

[29] T. Bu, W. Fang, J. Ding, P. Dai, Z. Yu, T. Huang, Optimal ANN-snn conversion for high-accuracy and ultra-low-latency spiking neural networks, in: International Conference on Learning Representations, 2021.

[30] N. Rathi, G. Srinivasan, P. Panda, K. Roy, Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation, in: International Conference on Learning Representations, 2019.

[31] Z. Wu, H. Zhang, Y. Lin, G. Li, M. Wang, Y. Tang, Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[32] A. Kugele, T. Pfeil, M. Pfeiffer, E. Chicca, Efficient processing of spatio-temporal data streams with spiking neural networks, Front. Neurosci. 14 (2020) 439.

[33] B. Ramesh, H. Yang, G. Orchard, N.A. Le Thi, S. Zhang, C. Xiang, Dart: distribution aware retinal transform for event-based cameras, IEEE Trans. Pattern Anal. Mach. Intell. 42 (11) (2019) 2767–2780.

[34] Y. Li, Y. Kim, H. Park, T. Geller, P. Panda, Neuromorphic data augmentation for training spiking neural networks, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII, Springer, 2022, pp. 631–649.

[35] G. Shen, D. Zhao, Y. Zeng, Eventmix: An efficient data augmentation strategy for event-based learning, Inform. Sci. 644 (2023) 119170.

[36] M. Yao, H. Zhang, G. Zhao, X. Zhang, D. Wang, G. Cao, G. Li, Sparser spiking activity can be better: Feature refine-and-mask spiking neural network for event-based visual recognition, Neural Netw. 166 (2023) 410–423.

[37] E.T. Rolls, A. Treves, The neuronal encoding of information in the brain, Prog. Neurobiol. 95 (3) (2011) 448–490.

[38] R.Q. Quiroga, S. Panzeri, Principles of Neural Coding, CRC Press, 2013.

Dongcheng Zhao received the Bachelor degree from XiDian University, Xi'an, Shaanxi, China, in 2016 and Ph.D degree from University of Chinese Academy of Sciences, Beijing, China, in 2021. He is currently an assistant professor in the Brain-inspired Cognitive Intelligence Lab, Institute of Automation, Chinese Academy of Sciences (CASIA), China. His current research interests include learning algorithms in spiking neural networks, thalamus-cortex interaction, visual object tracking, etc.

Guobin Shen received Bachelor degree from Sun Yat-Sen University, Guangzhou, China, in 2021. He is currently an Ph.D. student in University of Chinese Academy of Sciences, Beijing, China. His interests include brain-inspired cognitive computation models and brain-inspired neural networks.

Yiting Dong received Bachelor degree from Beijing University of Technology, Beijing, China, in 2020. He is currently an PH.D student in University of Chinese Academy of Sciences, Beijing, China. His interests include brain-inspired intelligence and spiking neural networks.

Yang Li received the Bachelor degree from Harbin Engineering University, Harbin, China, in 2019. He is currently an PH.D student in University of Chinese Academy of Sciences, Beijing, China. His current research interests include spiking neural networks and brain-inspired cognitive computations.

Yi Zeng obtained his Bachelor degree in 2004 and Ph.D degree in 2010 from Beijing University of Technology, China. He is currently a Professor and Director in the Brain-inspired Cognitive Intelligence Lab, Institute of Automation, Chinese Academy of Sciences (CASIA), China. He is a Principal Investigator in the Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Chinese Academy of Sciences, China, and a Professor in the School of Future Technology, and School of Humanities, University of Chinese Academy of Sciences, China, and a Funding Director of Center for Long-term AI, China. His research interests include brain-inspired Artificial Intelligence, brain-inspired cognitive robotics, ethics and governance of Artificial Intelligence, etc.