Neurocomputing 469 (2022) 189–197

Contents lists available at ScienceDirect

Neurocomputing

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

An adaptive threshold mechanism for accurate and efficient deep spiking convolutional neural networks

Yunhua Chen [a],[*], Yingchao Mai [a], Ren Feng [a], Jinsheng Xiao [b],[*]

[a] School of Computers, Guangdong University of Technology, Guangzhou, China
[b] School of Electronic Information, Wuhan Universigy, Wuhan, China

A R T I C L E   I N F O

Article history:
Received 28 August 2020
Revised 20 September 2021
Accepted 24 October 2021
Available online 27 October 2021
Communicated by Zidong Wang

Keywords:
Spiking convolutional neural networks
Conversion from CNN to SNN
Approximation error
Ratio-of-threshold-to-weights

A B S T R A C T

Spiking neural networks(SNNs) can potentially offer an efficient way of performing inference because the neurons in the networks are sparsely activated and computations are event-driven. SNNs with higher accuracy can be obtained by converting deep convolutional neural networks(CNNs) into spiking CNNs. However, there is always a performance loss between CNN and its spiking equivalents, because approximation error occurs in the conversion from the continuous-valued CNNs to the sparsely firing, event-driven SNNs. In this paper, the differences between analog neurons and spiking neurons in neuron models and activities are analyzed, the impact of the balance between weight and threshold on the approximation error is clarified, and an adaptive threshold mechanism for improved balance between weight and threshold of SNNs is proposed. In this method, the threshold can be dynamically adjusted adapting to the input data, which makes it possible to obtain as small a threshold as possible while distinguishing inputs, so as to generate sufficient firing to drive higher layers and consequently can achieve better classification. The SNN with the adaptive threshold mechanism outperforms most of the recently proposed SNNs on CIFAR10 in terms of accuracy, accuracy loss and network latency, and achieved state-of-the-art results on CIFAR100.

1. Introduction

In the past a few years, thanks to the development of deep learning algorithms, deep convolutional neural networks(CNNs) have achieved many remarkable results in cognitive tasks [1–3]. However, their success largely depends on high-performance computing hardware such as GPUs. The demanding for substantial computational and energy cost limits their application in circumstance with limited computing resources and power consumption.

Spike neural network(SNN) is one of the solution that can balance performance and computational cost. SNNs transmit information by spike trains with precise time, the neurons in the networks are sparsely activated and computations are event-driven, which is different from analogy neural networks that process continuous analog values without a notion of time in a synchronous way. When an SNN is deployed on a neuromorphic computing platform, it will be expected to obtain an ultra-low-power cognitive system [4,5].

SNNs have remarkable advantages in information processing speed and power consumption, however, because the internal state variables of neurons do not satisfy the continuously differentiable requirement, it is difficult to be trained. To solve this problem, some algorithms based on the rules of gradient descent [6–9] and spike-time dependent plasticity (STDP) [10–14] were proposed, which had partly solved the problem of training SNNs. However, it is still difficult to train deeper SNNs with complex network structures, which results in a remaining of huge gap of recognition performance between SNNs and CNNs in complex recognition tasks.

To narrow the performance gap between SNNs and CNNs, methods of converting CNNs to SNNs had been proposed. In these methods, a CNN is firstly trained using the standard stochastic gradient descent and back propagation algorithm, and then the trained weights are mapped to an SNN with the same structure as the CNN. Inference is performed on the converted SNNs [15,16].

Although an SNN based on conversion has a closer accuracy to its CNN equivalents, there is still an accuracy loss, the main reason is that the difference in the information processing mechanism between analog and spiking neurons will cause an approximate error in the conversion process. The accuracy loss increases with

* Corresponding authors.
E-mail addresses: yhchen@gdut.edu.cn (Y. Chen), xiaojs@whu.edu.cn (J. Xiao).

https://doi.org/10.1016/j.neucom.2021.10.080
0925-2312/© 2021 Elsevier B.V. All rights reserved.

the increase of network size and complexity of classification tasks. To solve this problem, many researches have been conducted to compensate for approximation error caused by conversion, which can be divided into three main categories: compensate for approximation error via adjusting parameters [17,16,18]; compensate for approximation error via replacing rate-based spike coding with other coding schemes, such as temporal coding and phase coding [19–21]; compensate for approximation error via modeling activation functions [22–24].

Despite the success of these methods, performance gap between SNNs and CNNs still exists, especially encountered with more complex tasks. In this paper, the approximation error in the conversion caused by the difference between the neuron models and working mechanisms of CNNs and SNNs are analyzed. An adaptive threshold mechanism for improved balance between weight and threshold is proposed. In this method, the threshold is dynamically adjusted adapting to the input data of the converted SNN, which makes it possible to obtain as small a threshold as possible while distinguishing inputs, so as to generate sufficient firing to drive higher layers and consequently can achieve better classification.

The experimental results on the CIFAR-10 and CIFAR-100 data sets show that the classification accuracy, accuracy loss and network latency of the SNN with the adaptive threshold mechanism proposed in this paper are largely improved compared with the other methods.

Our major contribution can be summarized as:

- The impact of the balance between weight and threshold on the approximation error is clarified via analyzing the approximation error between analog neurons and spiking neurons.
- An adaptive threshold mechanism for improved balance between weight and threshold of an SNN is proposed. In this method, the threshold adjustment is added to the SNN inference process to adaptively obtain the optimal threshold of a single test sample, which can obtain a smaller threshold, thereby can generate sufficient firing to drive higher layers and consequently can achieve better classification.
- Extensive comparative experiments have been conducted to verify the performance of the proposed method.

The rest of this paper is organized as follows. Section 2 introduces related works, Section 3 analyses the approximation error caused by differences in neuron models and activities, and the impact of the balance between weight and threshold on the approximation error. Sections 4 introduces the proposed adaptive threshold mechanism. Experimental results and analysis are provided in Section 5 and the conclusion is drawn in Section 6.

2. Related works

2.1. Compensate for approximation error via adjusting parameters

The earliest study began with the work of Pérez-Carrasco [25], who proposed to convert the analogy calculation unit into a neuron model with leakage and refractory periods to process input from event-based sensors. On this basis, Cao et al. [26] tailored the structure of a CNN to adapt to an SNN, used the rectified linear units(ReLU) as the activation to train the CNN. The trained weights were used for the deep SNN with the same structure, and the Integrate & Fire(IF) model was used as the spiking neuron model, thus a 10-layer converted deep SNN was generated, which has achieved better results on the benchmark data set MNIST.

Diehl et al. [17] studied the impact of the threshold on network performance, proposed a weight normalization method based on the maximum activation value of neurons in each layer, in which the weights are scaled to reduced the approximate error caused by unreasonable response of neurons.

The method was further improved by Rueckauer et al. [16], who proposed a robust weight normalization method based on abnormality of the reference value, and realized max pooling, softmax, batch normalization and bias, and achieved nearly lossless conversion based on 9-layer CNN.

Sengupta et al. [18] concluded that the weight normalization should be based on the maximum spike input of each layer in the converted deep SNN rather than the maximum activation of CNN, proposed a threshold optimization method named Spike-Norm. Unlike the above methods adjusting thresholds based on the CNN activations, Spike-Norm adjusts the thresholds based on the maximum inputs of each layer in the converted SNN, which reduced the accuracy loss of converted deep SNN on more complex classification tasks.

2.2. Compensate for approximation error via changing spike coding schemes

Lagorce and Siloni et al. [27,28] added temporal information into spike coding to form temporal-spatial features i.e. time surface, to reduce power consumption and network latency. However, these methods are tailored for neuromorphic visual sensors and have no equivalents in conventional deep learning or computer vision.

Kim et al. [19] replaced rate-based coding with phase coding and assigned weights to different spikes. Correspondingly, the total number of spikes and network latency are reduced, but phase coding produces more severe noises. To solve this problem, Rueckauer et al. [29] used time to first spike(TTFS) coding scheme with higher computational efficiency, and proposed a neuron model with bio-plausibility lower than IF model for this scheme. Each neuron was stipulated to fire only one spike so as to reduce the number of spikes and power consumption, but the accuracy decreased accordingly.

Chen et al. [20] proposed the concept of multi-strength spike. The spike strength is no longer just 0 or 1, but 1.5 or 2, to increase the spatial information of spike coding, to solve the problem that network latency of the SNN increases as the number of network layer increases. Correspondingly, the coding scheme for network input was adjusted accordingly, and some invalid or non-contributing neurons was removed using the network dynamic pruning technology to achieve sparsity and lower power consumption.

Park et al. [21] used burst spiking to achieve the dynamic firing threshold, which will change dynamically with the presence or absence of spikes, making it easier for subsequent inputs to fire spikes, and the hybrid spike coding schemes containing 4 modes of pairwise combination had been proposed, among which the *real + burst* combination has been proved to be better than the other combinations.

2.3. Compensate for approximation error via modeling activation functions

In the aforementioned methods, the ReLU function used in CNN training can be regarded as an approximation of the Integrate & Fire (IF) model with an absolute refractory period of 0, thus has good consistency with the IF model. However, when an SNN adopts a spiking neuron model with higher bio-plausibility, the CNN trained based on ReLU and its spiking equivalent obtained by conversion will have a larger approximation error. For this reason, some researchers attempted to propose activation functions having better consistency with the spiking neuron model to reduce the approximation error caused by conversion [22–24].

3. Analysis of approximation error brought by differences in neuron models and activities

As mentioned above, when we convert a CNN to its spiking equivalent, there will be an approximation error, and the error comes from many aspects, which will affect the matching degree of firing rates of spiking neurons and activations of analog neurons, thereby results in a performance loss in SNN. Here we mainly analyze the approximate errors caused by the difference of neuron models and activities between the spiking neurons and analog neurons.

3.1. The approximation error caused by difference in neuron models

The activation of an analog neuron based on the most widely uses activation function ReLU can be formalized as:

$$a_i^l := max\left(0, \sum_{j=1}^{M^{l-1}} W_{ij}^l a_j^{l-1} + b_i^l\right). \tag{1}$$

where $\boldsymbol{W^l}$ represents the weight matrix between layer $l-1$ and layer l, and M^{l-1} represents the number of neurons in layer $l-1$. a_j^{l-1} and a_i^l represent the activation value of neuron j in layer $l-1$ and neuron i in layer l, and b_i^l is the bias of neuron i. A neuron calculates its input based on the outputs of the previous layer, and only transmits information when the input is positive.

As one of the most widely used neuron models in SNNs, the IF model integrates the post-synaptic potentials (PSPs) generated by the input current at time t, which can be described as:

$$z_i^l(t) := V_{th}\left(\sum_{j=1}^{M^{l-1}} W_{ij}^l \Theta_{t,j}^{l-1} + b_i^l\right). \tag{2}$$

Where $\Theta_{t,j}^{l-1}$ is a step function indicating the spiking activity of neuron j at time t, as follows.

$$\Theta_{t,i}^l := \Theta(V_i^l(t) - V_{th}), with\ \Theta(x) = \begin{cases} 1 & if\ x \geqslant 0 \\ 0 & else \end{cases} \tag{3}$$

The IF neuron updates its membrane potential $V(t)$ in each time step as:

$$V_i^l(t) = V_i^l(t-1) + z_i^l(t). \tag{4}$$

Since the IF model does not contain leakage terms, when $z(t)$ is positive, the firing rate will increase as the positive input increases, and when the input $z(t)$ is negative, the output of the IF neuron will be zero since it can never cross the potential V_{th}, from this point of view, ReLU can approximate the firing rate of an IF neuron. However, this approximation is not strictly mathematically established, therefore there is always an approximation error between ReLU and IF neuron, which leads to the loss of performance after conversion.

3.2. The approximation error caused by difference in neuron activities

As can be seen from the above analysis, information is transmitted between neurons through spikes in SNNs. Neurons will accumulate membrane potential after receiving input, once the membrane potential $V(t)$ exceeds a threshold V_{th}, the neuron will fire a spike and reset the membrane potential.

Still taking the IF neuron as an example, first, suppose the input data is encoded as a constant input current $z_i^1 > 0$, the input to first-layer neurons is related to the ANN activation values via $z_i^1 = V_{th} \times a_i^1$. The CNN activations can be related to the SNN spike rates corresponding to different membrane potential reset mechanisms as follows [16]:

$$r_i^1(t) = \begin{cases} a_i^1 r_{max} \times \frac{V_{th}}{V_{th}+\zeta_i^1} - \frac{V_i^1(t)}{t\times(V_{th}+\zeta_i^1)} & \text{reset to zero} \\ a_i^1 r_{max} - \frac{V_i^1(t)}{t\times V_{th}} & \text{reset by subtraction} \end{cases}. \tag{5}$$

where a_i represents the activations in CNN, r_{max} represents the maximum firing rate of the neurons, V_{th} and $V_i^1(t)$ are the firing threshold and membrane potential of neuron i, and ζ_i^1 is the portion that the membrane potential exceeds the threshold.

As can be seen from the above equation, the spike rates $r_i^1(t)$ are not strictly proportional to the ANN activations, there is an approximation error term between them. Compared with the *reset to zero* mechanism, the *reset by subtraction* mechanism can avoid the approximation error caused by discarding the residual charge ζ, but it still contains an additive error term $\frac{V(t)}{t\times V_{th}}$. Actually, the error comes from the residual membrane potential of the neuron at the end of simulation. When the input stimulation stops, the membrane potentials will stop accumulating and are not great enough to make neurons generate spikes, which leads to the loss of information.

From the composition of the error term in the above equation, it can be seen that reducing the membrane potential or increasing the threshold can reduce the approximate error, but both tend to reduce the firing rate, which is not conducive to the deepening of the network layer. Because the membrane potential depends largely on the weights of the network, researchers seek for better ratio-of-threshold-to-weights to promote the deepening of an SNN while reducing the approximation error. Here we use RTTW as the abbreviation of ratio-of-threshold-to-weights for convenience.

3.3. The impact of the balance between weight and threshold on the approximation error

In this subsection, the spiking response of IF neurons and the ReLU function are compared to clarify the impact of the balance between weight and threshold on the approximation error.

The spiking response curve of the IF neuron model is shown in Fig. 1. It can be seen from this figure that when the input is small, the weighted sum of the input is less than the firing threshold, the neuron does not fire a spike. As the input increases, the weighted

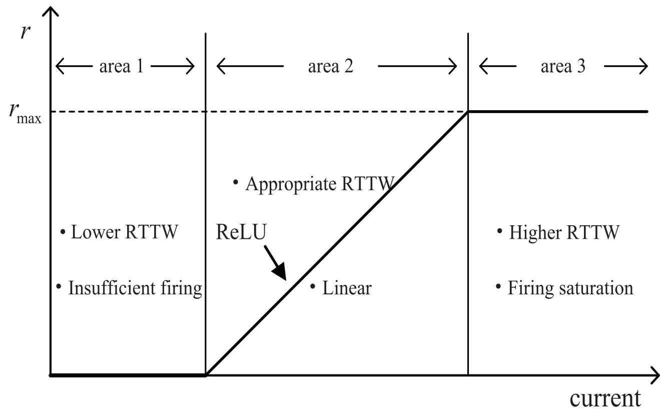


Fig. 1. Response curve of IF neuron.

sum of the input begin to exceed the threshold and the neuron begins to fire spikes. Since the IF neuron model does not contain leakage terms, the increase in the firing rate is linear with the input and is highly similar to the ReLU function. Finally, as the input continues to increase, the firing rate of the neuron remains at a constant value, i.e. the neuron will fire a spike within each time step and stays in a state of saturation.

It also can be seen from the above figure that higher RTTW may lead to insufficient activation as shown in area 1, and lower RTTW may lead to saturated firing as shown in area 3. There is a big difference between the spiking response curve and the ReLU function in these two areas that may cause a bigger approximation error during conversion. Therefore, it is necessary to set RTTW reasonably, so that the spiking response of the IF neurons can be corresponded to the spiking response curve in area 2, which can reduce the conversion approximation error caused by the difference of the neuron models and the neuron activities.

4. Methods

4.1. Basic neural operation and architectural constraints

The rectified linear unit (ReLU) is used as the CNN activation function, and the IF spiking neuron is used as the basic operation unit in the SNN. In this work, bias term is not used for both baseline analog VGG network and the converted SNN, this is not only because of the reasons mentioned in [18] that bias term will cause expanded parameter space exploration and more difficult conversion process, but also because when we reproduce existing research results, adding bias will often lead to the accuracy decrease of the converted SNNs. The average pooling operation is used in CNNs to avoid additional approximation errors caused by the conversion of pooling layers.

4.2. Input and output representation

In the methods of conversion-based spiking neural networks, rate-based coding is usually used to convert the magnitude of the analog values, such as the pixel intensity, into spike trains. There are several ways to implement rate based coding, and the effects of different implementations are different.

Poisson coding is a specific realization of rate-based coding, because the spike distribution after coding is similar to that in the biological brain, it has been widely used in spiking neural networks. In this encoding method, a random value is generated for each time step and compared with the corresponding analog value. If the generated random value is less than the analog value, a spike is fired. Which can ensure that the number of spikes input to the SNN is approximately proportional to the analog value of the input in CNN, however the random value may introduce additional noise and cause network performance degradation. For this reason, Cassidy et al. [30] proposed a uniform encoding method. The spike trains generated by the encoding method has a fixed interval between spikes. The larger the continuous value, the smaller the interval between spikes. Compared with Poisson coding, this coding method can better reflect the differences between pixels. However, limited by the temporal resolution and the length of the encoding window, this encoding method still has a small amount of information loss.

The encoding method proposed by Park et al. [21] interprets the input analog value as a constant current I, avoiding the difficulty of direct lossless encoding of analog data, and successfully encoding the analog data losslessly into spike trains. In this method, the membrane potential of neuron i in the first layer of the network can be expressed as:

$$V_i(t) = V_i(t-1) + I_i \qquad (6)$$

Neuron i adds a constant charge value to the membrane potential at each time step. When the membrane potential reaches the firing threshold, the neuron fire a spike, in this way, the encoding process is transferred to the first layer of the network. Due to its lossless characteristics and biological plausibility, this method has replaced Poisson coding and has been applied to many newly proposed deep SNNs. Therefore, we use this method to encode the input data. Corresponding to this encoding method, inference is based on the cumulative membrane potential of neurons at the output layer of the network over a given time-window.

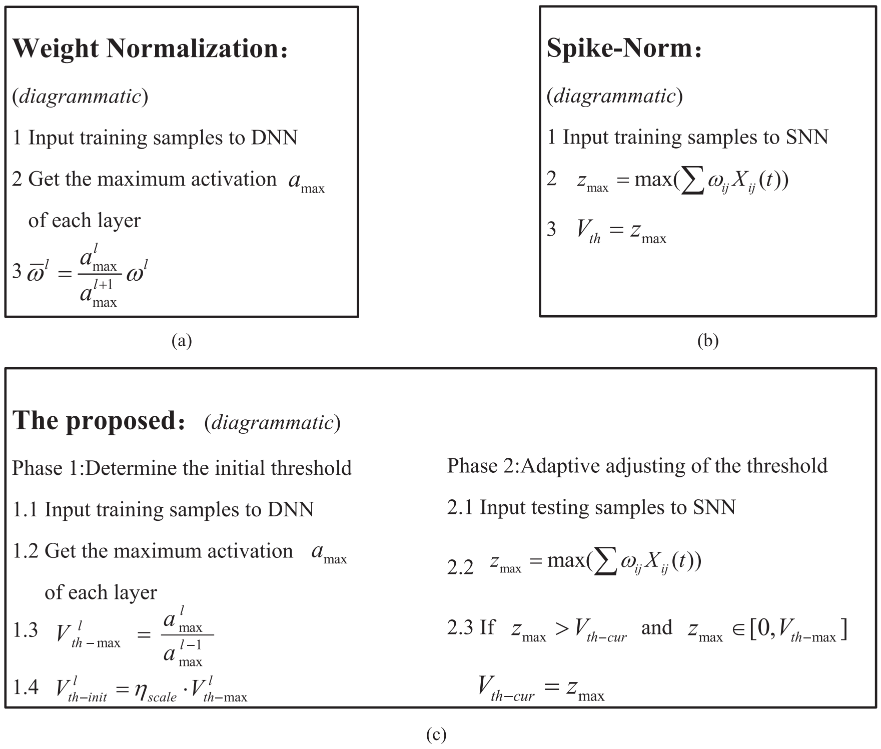
4.3. The adaptive threshold mechanism

As mentioned in Section 3, many researches reduce the approximation error by optimizing the value of RTTW. Weight Normalization [17] and Spike-Norm [18] are two outstanding representatives of them. In Weight Normalization, the weights and biases are scaled according to the maximum activation [17] or the 99.9th percentile of all the activations [16] of adjacent layers in CNN with the threshold unchanged. First, the maximum activation of each layer of the network is estimated through a certain number of training samples, and then the weights are scaled according to the maximum value of the adjacent layer, i.e., $\bar{w} = (a^{l}_{max}/a^{l+1}_{max})w$, as shown in Fig. 2(a). Such CNN-based weight adjusting, however, is not very suitable for the converted SNN. Hence, the Spike-Norm was proposed where the threshold of a particular neuron is set according to the max value of the PSPs in the corresponding layer, i.e., $z_{max} = max(\sum(w_i \cdot X_i(t)))$, as shown in Fig. 2(b).

Despite the success of these methods, it is still challenge to eliminate the approximation error between deeper CNNs and SNNs. Both of the above threshold adjustment methods are based on the maximum input or output value of a certain number of samples, in this way, the parameters is not optimal for a single sample, which may cause insufficient activation of some neurons. When the spike coding time window is small, there will be a larger approximation error.

Based on the above reasons, we propose an adaptive threshold adjustment method. Unlike the previous two methods that only adjust the weights on the CNN side or the SNN side, our method is divided into two phases: (1) determine of the initial threshold; (2) dynamic adjustment of the threshold, as shown in Fig. 2(c).

In the first stage, the initial threshold is determined according to the maximum activations of the adjacent layers. First, the maximum activation, which is greater than 99.9% of all the activations, of each layer of the network is estimated just as in the Weight Normalization method in [16]. Then the threshold is determined according to the maximum activations of adjacent layers. Since the threshold is obtained based on a large subset of training samples, it is generally larger. For the vast majority of input samples, even the maximum activation of units within a layer will lie far below a_{max} leading to insufficient firing within the layer to drive higher layers and subsequently worse classification results [16]. Therefore we decrease a_{max} by a scale factor η_{scale} to get the initial threshold, i.e., $V_{th-init} = \eta_{scale} \cdot a_{max}$. The pseudo-code for phase 1 of the algorithm is given below.



Weight Normalization:

(*diagrammatic*)

1 Input training samples to DNN

2 Get the maximum activation $a_{\max}$ of each layer

3 $\overline{\omega}^l = \frac{a^l_{\max}}{a^{l+1}_{\max}} \omega^l$

(a)

Spike-Norm:

(*diagrammatic*)

1 Input training samples to SNN

2 $z_{\max} = \max(\sum \omega_{ij} X_{ij}(t))$

3 $V_{th} = z_{\max}$

(b)

The proposed: (*diagrammatic*)

Phase 1:Determine the initial threshold

1.1 Input training samples to DNN

1.2 Get the maximum activation $a_{\max}$ of each layer

1.3 $V^l_{th-\max} = \frac{a^l_{\max}}{a^{l-1}_{\max}}$

1.4 $V^l_{th-init} = \eta_{scale} \cdot V^l_{th-\max}$

Phase 2:Adaptive adjusting of the threshold

2.1 Input testing samples to SNN

2.2 $z_{\max} = \max(\sum \omega_{ij} X_{ij}(t))$

2.3 If $z_{\max} > V_{th-cur}$ and $z_{\max} \in [0, V_{th-\max}]$

$V_{th-cur} = z_{\max}$

(c)

Fig. 2. A diagrammatic representation of the proposed method.

Algorithm 1: Determine the initial threshold (Phase 1)

```
Input: The convolutional neural network that has been trained
Output: Maximum threshold $V_{th-max}$ and initial threshold $V_{th-init}$ of each network layer in CNN
1: //The maximum threshold of each network layer is determined by data-based normalization, and its scaling is taken as the initial threshold
2: $previous\_factor = 1$
3: for $i \leftarrow 1$ to $\#net.layer$ do
4:    $max\_act = 0$;
5:    // Determine the maximum threshold for each layer
6:    for $j \leftarrow 1$ to $\#net.layer[i].neuron$ do
7:       $max\_act = max(max\_act, net.layer[i].neuron[j].output)$;
8:       $present\_factor = max\_act/previous\_factor$;
9:       $net.layer[i].V_{th-max} = 1 * present\_factor$;
10:        $previous\_factor = max\_act$;
11:    end for
12:    //Determine the initial threshold of each layer
13:    $net.layer[i].V_{th-init} = \eta_{scale} * net.layer[i].V_{th-max}$;
14: end for
```

In the second stage, the threshold is dynamically adjusted adapting to the specific input data. The maximum value of the weighted sum of PSPs $z_{max} = max(\sum(w_i \cdot X_i(t)))$ in each layer of the SNN will be calculated at each time step. If it is greater than the current threshold V_{th-cur}, replace V_{th-cur} with z_{max}. Compared with the Spike-Norm method, which adjusts the parameters based on a large number of training samples, this method adjusts the threshold for a single test sample. It can dynamically set the optimal threshold for each layer of neurons corresponding to the *current* sample during the inference process, and can distinguish the input difference while reducing the time for the membrane potential to cross the threshold, thus will result in firing rates that are more conducive to drive higher layers. The pseudo-code for phase 2 of the algorithm is given below.

Algorithm 2: Adaptive adjusting of the threshold (Phase 2)

```
Input: Input Possion Spike Trains$spikes$,Number of Time-Steps$\#timesteps$
Output: Threshold-balancing factors $V_{th-cur}[i]$ for each neural layer of the network
1: //Dynamic adjustment of threshold in simulation process of spiking neural network
2: $net.layer[1].input = spikes$;
3: //Set input of 1st layer equal to spike trains
4: for $t \leftarrow 1$ to $\#timesteps$ do
5:    $max\_input = 0$;
6:    // Set the threshold based on the input of the current layer
7:    for $i \leftarrow 1$ to $\#net.layer$ do
8:       //Forward pass spike trains for neurons in layer-i
9:       $net.layer[i]:forward(net.layer[i].input[t])$;
10:        $max\_input = max(0, net.layer[i].weight * net.layer[i].input[t])$;
11:        if $max_{input} > net.layer[i].V_{th-init}$ and $max_{input} < net.layer[i].V_{th-max}$ then
12:           $net.layer[i].V_{th-cur} = max_{input}$;
13:        end if
14:        //Record input spike-train for next layer
15:        $net.layer[i+1].input[t] = net.layer[i] : forward(net.layer[i].input[t])$;
16:    end for
17: end for
```

5. Experimental results and analysis

In the experiments, the SNN toolbox (SNN-TB) [16] was used to perform the CNN-SNN conversion.

The data sets. Two widely used data sets, CIFAR-10 and CIFAR-100, were selected for experiments. CIFAR-10 contains 60,000 color images of 10 categories, there are 6000 images for each category and the size of each image is 32×32 pixels. Choose 5000 images from each category for training, and the rest for testing.

CIFAR-100 contains 60,000 color images of 100 categories, which are divided into 20 super classes. There are 600 images for each category, each image has a fine label (the class it belongs to) and a coarse label (the super class it belongs to). Choose 500 images from each category for training, and the rest for testing.

Network structure and parameters. We use VGG-19 as the CNN, the only change to the network is to use average pooling instead of maximum pooling. The network training parameters are set as follows: The initial learning rate λ is set to be 0.1, the number of iterations is set to be 300, and every 80 iterations, the learning rate is updated according to the formula $\lambda = 0.1 \times \lambda$. The batch size is set to be 128 for CIFAR10 and 256 for CIFAR100, the momentum term is set to be 0.9, and the network layer weight is regularized according to the $L2$ rule and the attenuation term is set to be 0.0001. The cross-entropy loss function and stochastic gradient descent (SGD) algorithm were used for training. Data enhancement methods include: randomly flip the image horizontally, and randomly shift the image in the vertical and horizontal directions at a scale of 0.2, and finally normalize the image to the interval $[0, 1]$. The value of the pixel in the image is treated as a constant current to generate the spike trains corresponding to the pixel. The neuron model is IF, using *reset by subtraction* mechanism [16]. The resting membrane potential V_{rest} is set to be 0 mV, the simulation time T is set to be 400 ms for CIFAR10 and 1100 ms for CIFAR100, and each time step is 1 ms. The value of the scale factor used in the experiments is in the interval of [0.70,0.85].

The classification accuracy of the SNN. The trained VGG-19 has a classification accuracy of 93.93% on CIFAR10. After converting to its spiking equivalents using SNN-TB, the classification accuracy is only 31.09% on CIFAR10 without any threshold mechanism. When we applied the dynamic adaptive threshold mechanism described above in this paper, the classification accuracy quickly increased to more than 92.05% on CIFAR10.

The classification accuracy of SNN constructed by different methods on CIFAR10 and CIFAR100 are shown in Table 1 and Table 2, respectively. From which we can see that the conversion from CNN to SNN is lossless, it may be because the possible over fitting of ANN can be reduced by the threshold and discretization of converting to SNN, the accuracy of the converted SNN is slightly higher than that of ANN. The accuracy of our method on CIFAR10 is higher than all of the existing methods except for Chen's method [20]. Since the accuracy of the CNN in Chen's method is unknown, it is uncertain whether it is a lossless conversion. As can be seen from Table 2, our method achieves the highest accuracy and lossless conversion on the CIFAR100 data set. It should be noted that the accuracy of the SNN converted by the same trained ANN is fixed, however, due to the randomness of dropout, each time the ANN is trained, the accuracy will fluctuate slightly, and the accuracy of the converted SNN will also fluctuate slightly.

Convergence time. Unlike CNNs, which process data layer by layer in a synchronous way, SNNs process data in an event-based asynchronous way, i.e., as the spike trains continue to be input, the neurons in the output layer will gradually output spikes to realize the classification of input data. The time interval from spike input to spike output is the network latency. It is one of the important indicators to measure the performance of SNN and can be approximately expressed by the network convergence time defined by the following equation [16]:

$$C_T = \begin{cases} t \quad , \quad if \ \frac{|acc_t - acc_T|}{acc_T} < 0.02 \\ T \quad , \quad else \end{cases} \tag{7}$$

where acc_t represents the classification accuracy of the network at time t, and acc_T is the classification accuracy at the end of the simulation time window. The curve of classification accuracy over time is shown in Fig. 3. According to the convergence time defined by Eq. 7, the experimental results show that the classification accuracy on CIFAR10 can reach 92.10% after 122 steps, as shown in Fig. 3 (a), and it can reach 72.20% after 180 steps on CIFAR100, as shown in Fig. 3 (b).

Average firing rate. The average firing rate is also one of the important indicators to measure the performance of SNNs. It can be defined as Eq. 8, where $N_{spike}(t)$ represents the total number of spikes generated by the network at time t, N_{neuron} is the number of neurons in the network, and T is the length of time window.

$$\text{Average firing rate} = \frac{\sum_{t=1}^{T} N_{spike}(t)}{N_{neuron} \times T} \tag{8}$$

On CIFAR10, the firing states of the neurons were recorded and visualized in units of network layer, as shown in Fig. 4. The firing states of neurons in layer 5 in the time window without and with adaptive threshold mechanism are shown in Fig. 4 (a) and Fig. 4 (b), respectively. The vertical axis corresponds to the neuron number, and the horizontal axis is the inference time. If neuron j fires a spike at time i, then draw a blue dot at coordinates (i, j). Fig. 4 (c) and Fig. 4 (d) demonstrate the distributions of the firing rates, where the firing rate on the horizontal axis is the normalized value of r/r_{max}.

From Fig. 4 (a) and Fig. 4 (b), we can see that the network firing rate is more sparse after applying the proposed adaptive threshold mechanism. From Fig. 4 (c) and Fig. 4 (d), we can see that the firing rate of neurons without an adaptive threshold presents a roughly uniform distribution, and after the adaptive threshold is applied, the number of neurons with a lower firing rate increases significantly.

Table 1
Accuracy of Spiking Neural Network on CIFAR-10.

Method	Construction method	Number of layers	acc. (%) CNN	acc. (%) SNN	acc. loss (%)	Network latency (ms)
Wu(2019) [32]	Directly trained	7	-	90.53	-	-
Lee(2020) [33]	based on SGD	Resnet-11	-	90.95	-	100(Aboutb)
Cao(2015) [26]	Converted from	4	79.12	77.43	1.69	400(Windowa)
Hunsberger(2016) [22]	trained CNN	8	85.97	83.54	2.52	200(Windowa)
Rueckauer(2017) [16]		9	91.91	90.85	1.06	200(Windowa)
Kim(2018) [19]		6	89.10	89.20	−0.1	117(Lastc)
Kim(2018) [19]		Resnet-20	91.40	91.40	0	-
Hu(2018) [34]		Resnet-44	92.85	92.37	0.48	-
Chen(2018) [20]		19	-	94.01	-	80(Aboutb)
Sengupta(2019) [18]		16	91.70	91.55	0.15	1000(Aboutb)
Park(2019) [21]		16	91.41	91.41	0	793(Lastc)
Ours		19	93.93	93.97	-0.04	122

a *Window* indicates that this article does not discuss network latency, we use the length of the simulation time to indicate the network latency. b *About* indicates that the network latency is not listed in the paper, but there is a latency curve. We roughly infer the latency time based on the latency curve. c *Last* indicates the time required for the network to identify the most difficult-to-categorize image in the data set, i.e., the longest time needed for a single image to be classified.

Table 2
Accuracy of Spiking Neural Network on CIFAR-100.

Method	Construction method	Number of layers	acc. (%) CNN	acc. (%) SNN	acc. loss (%)	Network latency (ms)
Kim(2018) [19]	'Converted from trained CNN	Resnet-32	66.10	66.20	−0.1	117(Lastc)
Kim(2018) [19]		plain-32	64.30	63.70	0.6	-
Hu(2018) [34]		Resnet-44	70.18	68.56	1.62	-
Park(2019) [21]		16	68.77	68.69	0.08	3000(Lastc)
Ours		19	73.57	73.58	-0.01	180

a *Window* indicates that this article does not discuss network latency, we use the length of the simulation time to indicate the network latency. b *About* indicates that the network latency is not listed in the paper, but there is a latency curve. We roughly infer the latency time based on the latency curve. c *Last* indicates the time required for the network to identify the most difficult-to-categorize image in the data set, i.e., the longest time needed for a single image to be classified.

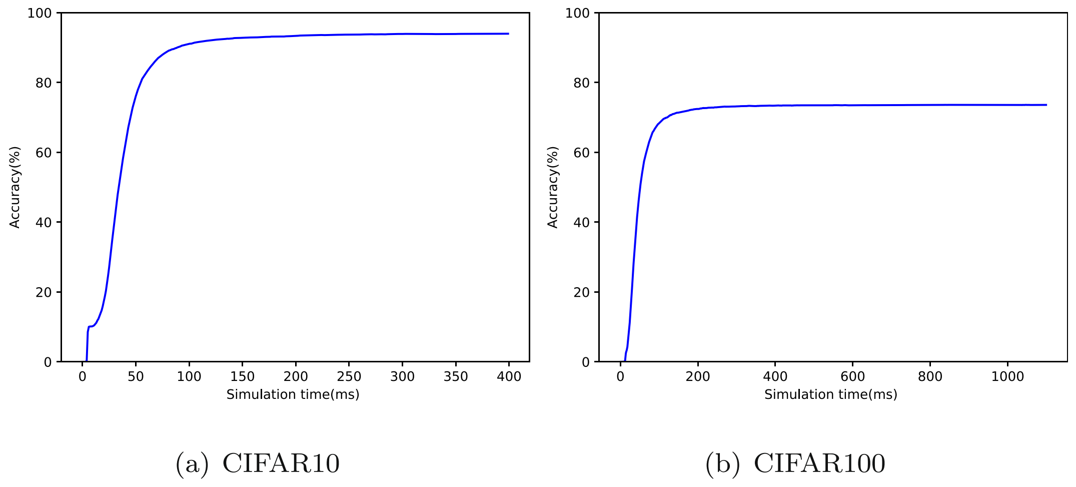


Fig. 3. The convergence time of the converted SNN on different data sets..

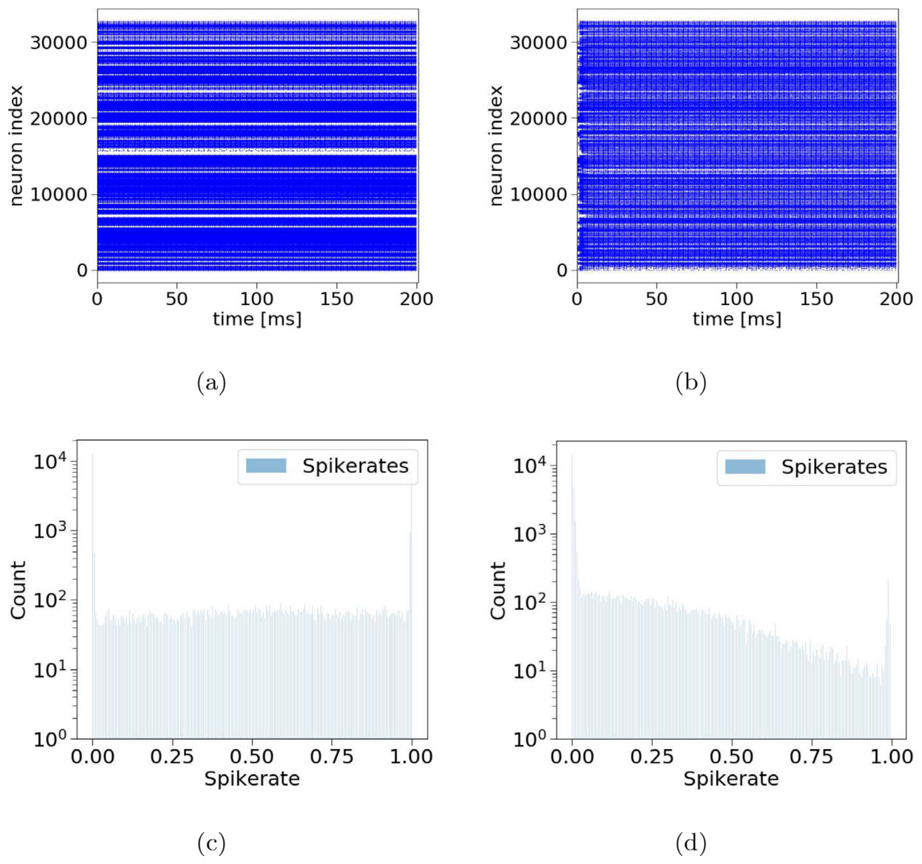


Fig. 4. The firing states comparison between fixed and dynamic adaptive threshold. (a) Firing state with fixed threshold. (b) Firing state with dynamic adaptive threshold. (c) Fire rate distribution with fixed threshold. (d) Fire rate distribution with dynamic adaptive threshold.

The average firing rates of the neurons with and without the adaptive threshold mechanism are 84 Hz and 122 Hz on CIFAR10 according to Eq. 8. After the adaptive threshold mechanism is applied, the firing rate of the network is reduced by about 31%, which shows the advantage of the proposed adaptive threshold mechanism in reducing the power consumption of the SNNs.

6. Conclusion

This work aims to reduce the performance loss caused by the approximation error exists in the conversion from CNNs to SNNs. The difference between analog neurons and spiking neurons in terms of neuron models and neuron activities, and the impact of the balance between weight and threshold on the approximation error are analyzed. On this basis, an adaptive threshold mechanism is proposed to make the neurons not only can distinguish differences between inputs, but also can reduce the accumulation time required for the membrane potential to reach the threshold. Unlike the existing threshold methods, such as Weight Normalization and Spike-Norm, which adjust the firing thresholds based on the training set. Our method adds threshold adjustment to the SNN inference process to adaptively obtain the optimal threshold of a single test sample, which results in smaller thresholds, thereby can generate sufficient firing to drive higher layers and consequently can achieve better classification. The proposed method outperforms most of the existing methods in terms of accuracy, accuracy loss, and network latency on the CIFAR-10 and CIFAR-100 data sets.

The adaptive threshold adjustment and optimization method proposed in this paper has successfully reduced the conversion error caused by the difference between analog and spiking neuron models, however the performance of SNN is also related to some other factors, such as the network structure and the spike coding method. Therefore, future research may focus on them.

The source code of this method has been uploaded to GitHub, for detailed information, please refer to the URL below: https://github.com/RPDS2020/Adaptively-adjusted-threshold-for-SCNN.git

CRediT authorship contribution statement

Yunhua Chen: Supervision, Conceptualization, Methodology, Software, Project administration, Writing - review & editing. **Yingcao Mai:** Software, Data curation, Writing - original draft, Writing - review & editing. **Ren Feng:** Validation, Investigation, Writing - review & editing. **Jinsheng Xiao:** Supervision, Funding acquisition.

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.

Acknowledgment

This work was supported by the Natural Science Foundation of Guangdong Province, China (No. 2021A1515012233) and the Open Project Fund of the Key Laboratory of Opto-electronic Information Processing, Chinese Academy of Sciences (No. OEIP-O-202009). We thank Center of Campus Network & Modern Educational Technology, Guangdong University of Technology, Guangdong, China for providing computational resources and technical support for this work.

References

[1] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, https://doi.org/10.1038/nature14539.
[2] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision(ICCV), 2015, pp. 1026–1034.
[3] J. Xiao, M. Shen, J. Lei, J. Zhou, R. Klette, H. Sui, Single image dehazing based on learning of haze layers, Neurocomputing 389 (2020) 108–122, https://doi.org/10.1016/j.neucom.2020.01.007, URL: https://www.sciencedirect.com/science/article/pii/S092523122030028X.
[4] M. Pfeiffer, T. Pfeil, Deep learning with spiking neurons: Opportunities and challenges, Front. Neurosci. 12 (2018) 774, https://doi.org/10.3389/fnins.2018.00774.
[5] A. Tavanaei, M. Ghodrati, S.R. Kheradpisheh, T. Masquelier, A. Maida, Deep learning in spiking neural networks, Neural Networks 111 (2019) 47–63, https://doi.org/10.1016/j.neunet.2018.12.002.
[6] J. Li, T. Delbruck, M. Pfeiffer, Training deep spiking neural networks using backpropagation, Front. Neurosci. 10 (508) (2016) 1–13, https://doi.org/10.3389/fnins.2016.00508.
[7] E.O. Neftci, C. Augustine, S. Paul, G. Detorakis, Event-driven random back-propagation: Enabling neuromorphic deep learning machines, Front. Neurosci. 11 (2017) 324, https://doi.org/10.3389/fnins.2017.00324.
[8] Y. Wu, L. Deng, G. Li, J. Zhu, L. Shi, Spatio-temporal backpropagation for training high-performance spiking neural networks, Front. Neurosci. 12 (331) (2018) 1–12, https://doi.org/10.3389/fnins.2018.00331.
[9] Y. Jin, W. Zhang, P. Li, Hybrid macro/micro level backpropagation for training deep spiking neural networks, in: Advances in Neural Information Processing Systems 31(ANIPS), Curran Associates Inc, 2018, pp. 7005–7015.
[10] A. Tavanaei, A.S. Maida, Multi-layer unsupervised learning in a spiking convolutional neural network, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2023–2030.
[11] A. Tavanaei, Z. Kirby, A.S. Maida, Training spiking convnets by stdp and gradient descent, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.
[12] A. Tavanaei, A. Maida, Bp-stdp: Approximating backpropagation using spike timing dependent plasticity, Neurocomputing 330 (2019) 39–47, https://doi.org/10.1016/j.neucom.2018.11.014.
[13] S.R. Kheradpisheh, M. Ganjtabesh, S.J. Thorpe, T. Masquelier, Stdp-based spiking deep convolutional neural networks for object recognition, Neural Networks 99 (2018) 56–67.
[14] J.C. Thiele, B. Olivier, D. Antoine, Event-based, timescale invariant unsupervised online deep learning with stdp, Front. Comput. Neurosc. 12 (2018) 1–14.
[15] D. Neil, M. Pfeiffer, S.-C. Liu, Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks, ACM, New York, NY, USA, 2016, pp. 293–298, https://doi.org/10.1145/2851613.2851724.
[16] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, S.-C. Liu, Conversion of continuous-valued deep networks to efficient event-driven networks for image classification, Front. Neurosci. 11 (2017) 682, https://doi.org/10.3389/fnins.2017.00682.
[17] P.U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in: Proceedings of International Joint Conference on Neural Networks (IJCNN), IEEE, 2015, pp. 1–8.
[18] A. Sengupta, Y. Ye, R. Wang, C. Liu, K. Roy, Going deeper in spiking neural networks: Vgg and residual architectures, Front. Neurosci. 13 (95) (2019) 1–10, https://doi.org/10.3389/fnins.2019.00095.
[19] K. Jaehyun, K. Heesu, H. Subin, L. Jinho, C. Kiyoung, Deep neural networks with weighted spikes, Neurocomputing 311 (2018) 373–386.
[20] R. Chen, H. Ma, S. Xie, P. Guo, P. Li, D. Wang, Fast and efficient deep sparse multi-strength spiking neural networks with dynamic pruning, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8. doi:10.1109/IJCNN.2018.8489339.
[21] S. Park, S. Kim, H. Choe, S. Yoon, Fast and efficient information transmission with burst spikes in deep spiking neural networks, in: 2019 56th ACM/IEEE Design Automation Conference (DAC), 2019, pp. 1–6.
[22] E. Hunsberger, C. Eliasmith, Training spiking deep networks for neuromorphic hardware, arXiv preprint arXiv:1611.05141.
[23] Liu Q., Chen Y. , Furber S., Noisy softplus: an activation function that enables snns to be trained as anns, arXiv preprint arXiv:1706.03609..
[24] Y. Chen, Y. Mai, J. Xiao, L. Zhang, Improving the antinoise ability of dnns via a bio-inspired noise adaptive activation funciton rand softplus, Neural Comput. 31 (6) (2019) 1215–1233, https://doi.org/10.1162/neco_a_01192.
[25] J.A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, B. Linares-Barranco, Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing-application to feedforward convnets, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2706–2719, https://doi.org/10.1109/TPAMI.2013.71.
[26] Y. Cao, Y. Chen, D. Khosla, Spiking deep convolutional neural networks for energy-efficient object recognition, Int. J. Computer Vision(IJCV) 113 (1) (2015) 54–66.
[27] X. Lagorce, G. Orchard, F. Galluppi, B.E. Shi, R.B. Benosman, Hots, A hierarchy of event-based time-surfaces for pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 39 (7) (2017) 1346–1359.

[28] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, R. Benosman, Hats: Histograms of averaged time surfaces for robust event-based object classification, Computer Vision and Pattern Recognition (CVPR) (2018) 1731–1740, https://doi.org/10.1109/CVPR.2018.00186.
[29] B. Rueckauer, S. Liu, Conversion of analog to spiking neural networks using sparse temporal coding, in: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1–5. doi:10.1109/ISCAS.2018.8351295.
[30] A.S. Cassidy, P. Merolla, J.V. Arthur, S.K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T.M. Wong, V. Feldman, A. Amir, D.B. Rubin, F. Akopyan, E. McQuinn, W.P. Risk, D.S. Modha, Cognitive computing building block, in: A versatile and efficient digital neuron model for neurosynaptic cores, in: The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–10, https://doi.org/10.1109/IJCNN.2013.6707077.
[32] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, L. Shi, Direct training for spiking neural networks: Faster, larger, better, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019, pp. 1311–1318. doi:10.1609/aaai.v33i01.33011311.
[33] C. Lee, S.S. Sarwar, P. Panda, G. Srinivasan, K. Roy, Enabling spike-based backpropagation for training deep neural network architectures, Front. Neurosci. 14 (2020) 119, https://doi.org/10.3389/fnins.2020.00119.
[34] Y. Hu, H. Tang, Y. Wang, G. Pan, Spiking deep residual network, ArXiv abs/1805.01352.

Yunhua Chen received the B.Eng. degree in Computer Software from Wuhan University of Surveying and Mapping, China, in 2000, the M.Eng. degree in Computer Application Technology from Wuhan University, China, in 2003, and the Ph.D. degree in Control Theory and Control Engineering from Guangdong University of Technology, China, in 2013. She was a visiting scholar in the APT group led by Professor Steve Furber at the University of Manchester from March 2016 to March 2017, and a member of International Neural Network Society (2019-2020). She is currently an associated professor with the school of Computers, Guangdong University of Technology, Guangzhou, China. Her research interests include neuromorphic computing, computer vision and machine learning.

Yingchao Mai recieved the B.Eng. degree in Information Engineering and the M.Eng. degree in Computer Technology from Guangdong University of Technology, Guangzhou, China, in 2017 and 2020 respectively. He is now a part-time research assistant at Guangdong University of Technology. His research interests include neuromorphic computing and computer vision. He is the recipient of Excellent Graduate Student of Guangdong University of Technology, China National Graduate Scholarship.

Ren Feng received the B.S. degree in Information and Computing Science from Changchun University of Science and Technology in 2019. He is currently a post-graduate student of Guangdong University of Technology. His research primarily focuses on spiking neural networks.

Jinsheng Xiao is an IEEE member and an associate professor in Information and Communication Engineering at the Electronic Information School, Wuhan University, China. He received his bachelors degree in computational mathematics at Wuhan University, China, in 1996 and PhD degree in mathematics from Wuhan University, China, in 2001. He has authored or co-authored more than 50 scientific articles in journals, books, and conference proceedings. His work focuses on image and video processing, computer vision, and image and video enhancement.