Neural Computing and Applications (2024) 36:8531–8547
https://doi.org/10.1007/s00521-024-09529-w

ORIGINAL ARTICLE

MSAT: biologically inspired multistage adaptive threshold for conversion of spiking neural networks

Xiang He[1,2,3] · Yang Li[1,2,3] · Dongcheng Zhao[1,2] · Qingqun Kong[1,2,3,4] · Yi Zeng[1,2,3,4,5]

Received: 6 July 2023 / Accepted: 18 January 2024 / Published online: 25 February 2024

Abstract

Spiking neural networks (SNNs) can do inference with low power consumption due to their spike sparsity. Although SNNs can be combined with neuromorphic hardware to achieve efficient inference, they are often difficult to train directly due to discrete non-differentiable spikes. As an alternative, ANN-SNN conversion is an efficient way to achieve deep SNNs by converting well-trained artificial neural networks (ANNs). However, the existing methods commonly use constant threshold for conversion. A high constant threshold value prevents neurons from rapidly delivering spikes to deeper layers and causes high time delay. In addition, the same response for different inputs may result in information loss during the information transmission. Inspired by the biological adaptive threshold mechanism, we propose a multistage adaptive threshold (MSAT) method to alleviate this problem. Instead of using a single, constant value, the threshold is adjusted in multistages, adapting to each neuron's firing history and input properties. Specifically, for each neuron, the dynamic threshold is positively correlated with the average membrane potential and negatively correlated with the rate of depolarization. The adaptation to membrane potential and input allows a timely adjustment of the threshold to fire spikes faster and transmit more information. Moreover, we analyze the spikes of inactivated neurons error, which is pervasive in early time steps. We also propose spike confidence accordingly to measure confidence about the neurons that correctly deliver spikes. Such spike confidence in early time steps is used to determine whether to elicit the spike to alleviate the spikes of inactivated neurons error. Combined with the proposed methods, we examine the performance on CIFAR-10, CIFAR-100, and ImageNet datasets. We also conduct sentiment classification and speech recognition experiments on the IDBM and Google speech commands datasets, respectively. Experiments show that our methods can achieve near-lossless and lower latency ANN-SNN conversion. In summary, we build a biologically inspired multistage adaptive threshold for converted SNN, with comparable performance to state-of-the-art methods while improving energy efficiency.

Keywords Spiking neural networks · ANN-SNN conversion · Multistage adaptive threshold · Spike confidence · Low latency

1 Introduction

Artificial neural networks (ANNs) have been widely used in speech recognition, image processing, and other fields [1–3]. However, with network structures becoming more complex, they often require large amounts of computational resources. In addition, the current ANNs computing paradigm differs from the human brain's mechanism, which communicates by transmitting sparse spike trains. Spiking neural networks (SNNs) work similarly to the brains and transmit the spike sequence to the postsynaptic neurons. The sparsity of spikes and the event-based computational paradigm enables SNN to have low power consumption characteristics.

SNNs have the potential for efficient inference when combined with neuromorphic hardware [4–6]. In addition, SNNs exhibit inherent efficiency in processing temporal and spatial data due to their diverse coding mechanisms and event-driven characteristics. However, because the spikes are not differentiable, it is challenging to train SNNs directly. To solve this problem, some algorithms based on surrogate gradients (SG) [7–16] and spike-timing-dependent plasticity (STDP) [17–20] have been proposed. However, it is still challenging to train deeper SNNs with

Xiang He and Yang Li have contributed equally to this work.

Extended author information available on the last page of the article

complex network structures from scratch, resulting in a massive gap in performance between SNNs and ANNs [9, 16, 21, 22]. To bridge the performance gap between SNNs and ANNs, methods of converting well-trained ANNs to SNNs have been proposed. The fundamental principle of such conversion is that firing rates of spiking neurons could approximate their counterparts activation (ReLU) in ANNs with sufficient time steps [23].

Although there has been a lot of work dedicated to achieving efficient SNNs by converting ANNs [23–30], they all face the same challenge: that is, the large time delay of the converted SNNs and the loss of information during the conversion process. For the first challenge, i.e., the large time delay, we argue that this is due to the fact that in most existing conversion schemes, the thresholds of neurons in the same layer are identical and equal to the maximum activation value of that layer in the ANN. In this case, spatially, the neuron needs to accumulate more membrane potential to exceed a preset threshold and deliver the spike, which makes the deeper neuron need to wait longer to receive the spike from the prior layer, and temporally, the neuron delivers the same information when firing the spike at different time steps, regardless of how much the membrane potential exceeds the threshold, causing an information loss. While in the biological counterpart of the spiking neuron, the spiking thresholds exhibit large variability, which has a significant impact on the transmission of input information to spiking neurons [31]. The threshold value is not fixed even for the same neuron. The threshold variability can be considered as an inherent characteristic of biological neurons, which exhibits self-adaptation to the membrane potential over a short time scale [32–36]. It also enhances coincidence detection properties of cortical neurons [32, 36] as mentioned in Fontaine et al. [31]. Therefore, from the biological plausibility perspective, the different threshold across neurons and time steps is a genuine feature of neurons and profoundly impacts the input coding.

For the second challenge, i.e., the conversion error due to the loss of information during the conversion process, we find that this is due to a class of errors that are prevalent in networks, which we define as spikes of inactivated neurons error, and the analysis and resolution of which is often neglected by previous work. The purpose of this paper is to overcome and mitigate these two challenges. First, to reduce the time delay and obtain SNNs with low latency, motivated by the above biological mechanisms, we propose a multistage adaptive threshold (MSAT) for ANN-SNN conversion to effectively take advantage of the spatial information brought by different neurons in the same layer and the temporal information brought by the accumulation of membrane potentials over time steps. "Multi-stage" implies that the process of threshold updating occurs in multiple continuous time steps. This multistage process ensures a more fine-grained and adaptive approach to spike firing that more closely mimics biological neuronal systems' complex, dynamic nature than a constant threshold. It allows for more efficient and timely threshold adjustments, allowing neurons to fire spikes rapidly and transmit more information. For each neuron, the dynamic threshold is related to the firing history, i.e., positively correlated with the preceding membrane potential, and to the input characteristics, i.e., negatively correlated with the rate of depolarization. Then, for the loss of information in the conversion process, we analyze the conversion error layer-to-layer and show that the spikes of inactivated neurons (SIN), i.e., the misfired spikes which are due to the transient characteristics of the spikes, are generated in the early time steps and cannot be eliminated in later time steps, thus causing degradation of accuracy. According to statistical analysis of the error introduced by SIN in each layer, we propose spike confidence to measure the confidence about the correct neurons rather than misfired neurons and use spike confidence only in early time steps to determine whether to elicit the spike. Finally, we conduct experiments on standard object classification benchmarks and non-visual domains such as natural language processing and speech to validate the effectiveness and universality of the proposed method.

Our major contributions can be summarized as follows:

- Inspired by widely existing mechanisms in the nervous system, we propose the multistage adaptive threshold (MSAT) for converted SNNs. It adjusts the threshold value based on the network's activity and firing history. This allows neurons to generate spikes more promptly, reducing the time delay for spike propagation to deeper layers.
- We formulate the layer-wise conversion error and derive the spikes of inactivated neurons error. Accordingly, we propose spike confidence to alleviate SIN error. Each spike is given a confidence according to the statistical SIN error, and the spike confidence determines whether to fire a spike rather than fire directly.
- We conduct comprehensive experiments and ablation studies to show that the multistage adaptive threshold and spike confidence measurement facilitate faster transmission of information and reduce information loss in spike trains. Experimental results show that our proposed method has a comparable performance with the state of the art methods but consumes less energy.

2 Related work

The conversion methods use the well-trained ANN and map it to an equivalent SNN. The studies begin with Pérez-Carrasco et al. [37], then Diehl et al. [23] propose the conditions that SNN conversion should satisfy, namely the firing rate in integrate-and-fire (IF) neuron [38] can approximate ReLU activation value. A mathematical statement of the feasibility of conversion is expressed firstly in Rueckauer et al. [39]. Since Sengupta et al. [24], the converted SNNs have begun going deeper and doing classification tasks in larger datasets. However, performance degradation is still the main problem. A more efficient reset method named soft-reset, which uses a reset by subtraction mechanism, is used in SNN conversion [39–41] encode activation value of neurons in the ANN as time-to-first-spike in the converted SNN. Although the performance of the converted SNN is improved, the cost of high time delays also comes. To reduce the high latency of the conversion SNN, some researchers propose more efficient ANN activation functions or SNN neuron models, such as Ding et al. [42] propose Rate Norm Layer to replace the ReLU activation function in ANN training for better performance. Yu et al. [25] construct the deep SNN with double-threshold and the augmented spike. Li and Zeng [27] use burst mechanism and propose LIPooling to solve the conversion error caused by the MaxPooling layer. Some other works focus more on the correspondence between ANN and SNN to reduce the conversion error from the perspective of analyzing the conversion error. For example, Bu et al. [26, 30] consider membrane potential initialization and quantization of the activation function to obtain faster SNN. Deng and Gu [43] divide conversion into floor and clip error from a new quantization perspective, Li et al. [28] further optimize the conversion error. Meng et al. [44] analyze deviation error and convert very deep ANNs to SNNs. The "deviation error" is similar to SIN error, different from prior work, we will then analyze this statistically to mitigate this error at the early time step. Nonetheless, most previous works have been limited to the fixed threshold and ignored utilizing the individual neurons to transmit information effectively. In summary, compared to a constant threshold, a multistage threshold is more fine-grained but has not been explored in depth for deep spiking neural networks. Only a few existing studies have studied the dynamic threshold rules within SNNs. In the realm of converted SNNs, Li et al. [28, Kim et al. [29] take threshold variation into consideration while they still use the two-stage or heuristic method for SNNs, which require careful design or extensive search. Wu et al. [45] also use a similar dynamic threshold in converted SNNs, but it is not comprehensive enough in terms of the biological perspective. In the realm of SNN for robot control, Ding et al. [46] exploit the biological dynamic threshold mechanism, which is similar to our approach. Nevertheless, the focus of the two works is different: Ding et al. [46] focus on achieving a steady-state SNN for specific tasks such as robot obstacle avoidance and continuous control. In converted SNNs, however, we are more concerned with the timely transmission of information, i.e., the variable thresholds and the variation itself improve network performance. Our work aims to contribute a bioinspired multistage threshold for converted SNNs to reduce latency while analyzing and reducing SIN errors that have received less attention in other work.

3 Conversion errors analysis

In this section, we first review the neuron models of ANNs and SNNs. Then, we analyze the conversion errors with constant threshold and adaptive threshold separately. The symbols used in this paper are summarized in Table 1.

3.1 Neuron model

In ANN, the activation value a_i^l (after ReLU) of neuron i in layer l can be computed as

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

where $l \in \{1, \cdots, L\}$ indicates l-th layer in a network with L layers; W_{ij}^l is the weight connection between neuron i in layer l and neuron j in layer $l-1$; b_i^l means neuron i bias in layer l and is constant all the time; we omit the bias for convenience in the following description. The number of neurons in layer l is M^l, and activation value a_i^l starts from $l = 0$ and $a^0 = x$ for the direct input.

In SNN, one of the most widely adopted models is integrate-and-fire (IF) neuron, the membrane potential before firing at time step t, $\mathbf{V}_i^l(t)$ is a sum of the membrane potential at time step $t-1$ and input in current time step. When $\mathbf{V}_i^l(t)$ exceeds a certain threshold, it emits an output spike and resets the membrane potential. Another way of membrane potential update is through a soft-reset mechanism, which subtracts the threshold in $\mathbf{V}_i^l(t)$ rather than resetting the membrane potential to V_{reset}. We used $V_i^l(t)$ to denote membrane potential after firing, then the mathematical form is as follows:

Table 1 Summary of notations in this paper

Symbol	Definition	Symbol	Definition
l	Layer index	M^l	Neuron numbers in layer l
i	Neuron index	W	Weight connection
t	Current time step	a	ANN activation value
T	Total time step	$\mathbf{V}_i^l(t)$	Membrane potential before firing
V_{th}	Constant threshold	$V_i^l(t)$	Membrane potential after firing
$V_{th}(t)$	Dynamic threshold	p	Spike confidence
$\Theta_{t,i}^l$	Spike of i-th in layer l at time step t	e	Conversion error in total time step
c	Spike filter	E	Early time step

$$V_i^l(t) = V_i^l(t-1) + V_{th}^{l-1}\sum_{j=0}^{M^{l-1}} W_{ij}^l \Theta_{t,j}^{l-1} - V_{th}^l \Theta_{t,i}^l, \tag{2}$$

where $\Theta_{t,i}^l$ is the elicited spike of neuron i in layer l at time step t. Here, we use $\Theta_{t,i}^l$ to represent $\Theta_i^l(t)$ for simplicity.

$$\Theta_{t,i}^l = \mathcal{H}\left(V_i^l(t-1) + z_i^l(t) - V_{th}^l\right), \quad \mathcal{H}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{else.} \end{cases} \tag{3}$$

where $z_i^l(t)$ is the input of neuron i in layer l and time t:

$$z_i^l(t) = V_{th}^{l-1}\sum_{j=0}^{M^{l-1}} W_{ij}^l \Theta_{t,j}^{l-1}. \tag{4}$$

3.2 Constant threshold conversion error

The main idea in ANN-SNN conversion is using firing rate $r_i^l(t)$ to approximate activation value a_i^l. Here, we give an analytical explanation for approximation process.

The conversion error comes from two-part: converting ANN to SNN directly, resulting in quantization error and clip error $e_{i,QC}^l$; the other is transient characteristic of neurons and irregular elicited spike, resulting in spikes of inactivated neurons [47] error $e_{i,SIN}^l$, as shown in Fig. 1. In the subsequent error analysis, we assume that the firing rate of the preceding layer fully represents the activation value, which means $r_i^{l-1} = a_i^{l-1}$. Then, the conversion error in layer l could be the following equation:

$$e_i^l = r_i^l - a_i^l = e_{i,QC}^l + e_{i,\text{SIN}}^l. \tag{5}$$

3.2.1 Quantization and clip error

For Eq. 2, it can be written in the following iterable form

$$V_i^l(t) - V_i^l(t-1) = V_{th}^{l-1}\sum_{j=0}^{M^{l-1}} W_{ij}^l \Theta_{t,j}^{l-1} - V_{th}^l \Theta_{t,i}^l, \tag{6}$$

By accumulating Eq. 6 in time steps from 0 to T and dividing T simultaneously on the left and right sides of the equation, we have:

$$\frac{V_i^l(T) - V_i^l(0)}{T} = \frac{\sum_{j=0}^{M^{l-1}} W_{ij}^l \sum_{t=0}^{T} V_{th}^{l-1}\Theta_{t,j}^{l-1}}{T} - \frac{\sum_{t=0}^{T} V_{th}^l \Theta_{t,i}^l}{T} \tag{7}$$

under the condition that the initial membrane potential $V_i^l(0)$ is zero, we can derive the firing rate $r_i^l(T)$ relationship layer by layer,

$$r_i^l(T) = \sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1}(T) - \frac{V_i^l(T)}{T}, \tag{8}$$

where $r_i^l(T) = \sum_{t=0}^{T} V_{th}^l \Theta_{t,i}^l / T$. When V_{th}^l is larger than the maximum activation value, $V_i^l(T)$ will be less than V_{th}^l. Thus, the residual membrane potential cannot be output, which is why information transmission suffers a loss. Because of the discreteness of the time steps, the mean firing rate can be described as a step floor function and cannot precisely approximate the continuous ReLU function, known as quantization error or flooring error. For example, as shown in Fig. 1, activation value 0.42 should be accurately represented by 42 spikes within 100 time steps, while 4.2 spikes cannot be achieved in discrete 10 time steps because the number of spikes should be an integer. This can also be represented by making a variation to Eq. 8

$$\frac{T}{V_{th}^l} r_i^l(T) = \frac{\sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1}(T) T}{V_{th}^l} - \frac{V_i^l(T)}{V_{th}^l}, \tag{9}$$

The left term in Eq. 9 is the number of fired spikes, i.e., $\frac{T}{V_{th}^l} r_j^l(T) = \sum_{t=0}^{T} \Theta_{t,j}^l$ is an integer. In the case where the residual membrane potential $V_i^l(T)$ is less than the threshold V_{th}^l, the $\frac{V_i^l(T)}{V_{th}^l}$ value ranges from 0 to 1. Therefore, the right hand side of the formula should take a round-down for the first term, i.e.,

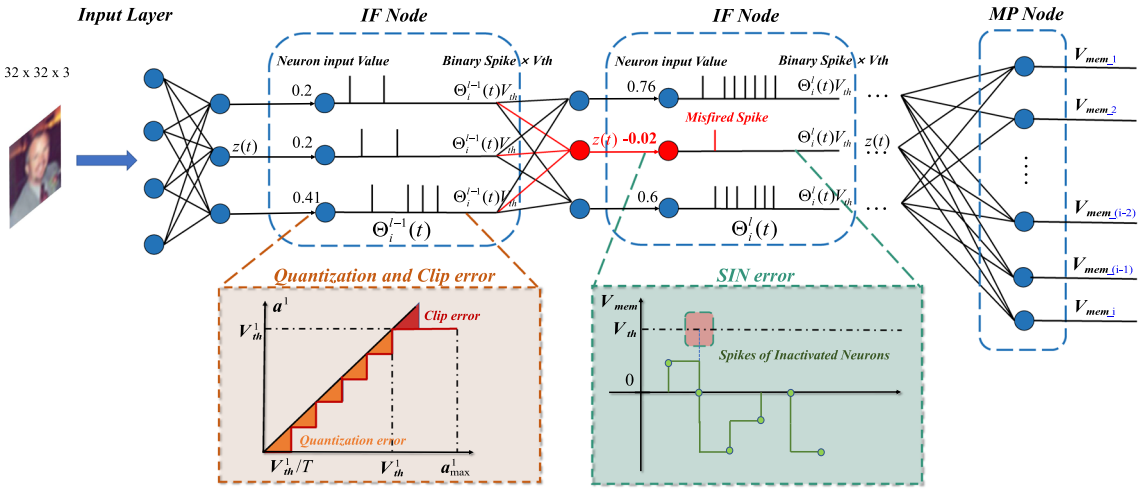


Fig. 1 A multilayer spiking neural network within 10 time steps for demonstrating conversion error. The red box shows the quantization error and clip error, caused by the discrete and insufficient time steps. The green box shows the spikes of inactivated neurons error, caused by the dynamic transients of the neurons. The MP nodes in the last layer are IF neurons that only accumulate membrane potential without firing, using membrane potential as the network output

$$r_i^l(T) = \frac{V_{th}^l}{T}\left\lfloor \frac{\sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1} T}{V_{th}^l} \right\rfloor, \tag{10}$$

where r_j^{l-1} means $r_j^{l-1}(T)$ for simplifying the statement and $\sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1}$ is a_i^l, and $\lfloor \cdot \rfloor$ denotes the floor function.

Suppose the voltage threshold is set smaller than the maximum activation value. In that case, when the $\mathbf{V}_i^l(t)$ exceeds the threshold, the emitted spike will not transmit effective information to distinguish membrane potential above the threshold, which is known as clip error. Setting the threshold to maximum activation value can avoid this but suffers a notable latency.

Considering the clip operation, the actual firing rate can be expressed as

$$r_i^l(T) = \text{clip}\left(\frac{V_{th}^l}{T}\left\lfloor \frac{\sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1} T}{V_{th}^l} \right\rfloor, 0, V_{th}^l\right), \tag{11}$$

the $\text{clip}(x, 0, V_{th}^l)$ is a mathematical operation that constrains x value within 0 and V_{th}^l.

Equation 11 shows that even assuming the same inputs in the previous layer, the average firing rate of the SNN still differs from the activation values in the ANN due to discrete quantization and clip, thus generating quantization and clip errors $e_{i,QC}^l$.

3.2.2 Spikes of inactivated neurons error

Inactivated neurons refer to the neurons whose activation value counterparts in ANN are negative. Theoretically, they should not fire spike in all time steps to achieve the ReLU function that filters the negative value. However, there will be a fraction of inactivated neurons that generate misfired spikes due to the transient characteristics of the spikes. We use $\mathcal{R}$ to denote this type of neuron, then it can be expressed as $\mathcal{R} = \left\{ i \middle| \sum_{t=0}^{T} \Theta_{t,i}^l > 0,\ a_i^l(t) < 0 \right\}$. The spikes mistakenly fired by neurons in $\mathcal{R}$, which we define as spikes of inactivated neurons (SINs). The SIN in the i_{th} neuron of layer l, SIN_i^l can be expressed by the formula as

$$\text{SIN}_i^l = \sum_{t=0}^{T} \Theta_{t,i}^l, \quad i \in \mathcal{R}. \tag{12}$$

Once neurons in $\mathcal{R}$ elicit spikes, the corresponding mean firing rate r_i^l will be larger than zero. As shown in Fig. 1, the adjacent weights of the red neuron are respectively 0.5, 0.5, and -0.5, then the simulation activation value is -0.01, meaning that no spike should have been fired. A spike fired by mistake results in SIN error. The SIN error of i_{th} neuron of layer l, i.e., $e_{i,\text{SIN}}^l$ can be expressed as

$$e_{i,\text{SIN}}^l = \frac{\sum_{j=0}^{M^{l+1}} \sum_{t=0}^{T} W_{ji}^{l+1} \Theta_{t,i}^l}{T} - \mathbf{0}, \quad i \in \mathcal{R}. \tag{13}$$

Recall that W_{ji}^{l+1} represents the weight connection between neuron i in layer l and neuron j in layer $l+1$. Thus, Eq. 13 represents that the spikes of inactivated neuron from the current layer are passed on to the next layer by multiplying the weights. And it would have been expected that the spikes should not have been fired.

Since SIN mean undesired error existed in spike, SIN lead to inaccurate coding of signals and degradation of the accuracy. By observing outputs of each neuron at different time steps, we find that the neurons with SIN usually fire at an early stage and then keep silent at a later stage during

the conversion process. We count neurons with SIN proportion in each layer, as shown in Fig. 2. It shows that neurons with SIN are prevalent and take a larger proportion in the deeper layer. Moreover, Most SIN appear in the early time steps. For example, for VGG16 on CIFAR-100 with a total of 256 time steps, the proportion of neurons with SIN in 32 time steps is approximately equal to the proportion of 256 time steps. In other words, most of the SIN in the entire 256 time steps occur within 32 time steps, which means that 32 time steps are representative enough to resolve the SIN error. This is why we choose to process SIN at an early time step.

3.3 Adaptive threshold optimization error

In this part, we demonstrate that when the threshold is a function of time step t, spike trains are able to transmit the equivalent or more efficient information than the constant threshold.

With the threshold varied with time step, Eq. 2 can be rewritten as follows:

$$V_i^l(t) = V_i^l(t-1) + \sum_{j=0}^{M^{l-1}} V_{th,j}^{l-1}(t) W_{ij}^l \Theta_{t,j}^{l-1} - V_{th,i}^l(t)\Theta_{t,i}^l. \tag{14}$$

The firing rate $r_i^l(T)$ during time step T is computed as $\sum_{t'=0}^{T} V_{th,i}^l(t')\Theta_{t',i}^l/T$, which means Eq. 8, the firing rate relationship in the higher layer still exists. On this basis, the residual neuron membrane potential $V_i^l(T)$ can be appropriately adjusted, so the spike information could be more efficient and the conversion latency could be shorten. The optimization target in threshold adaptation is

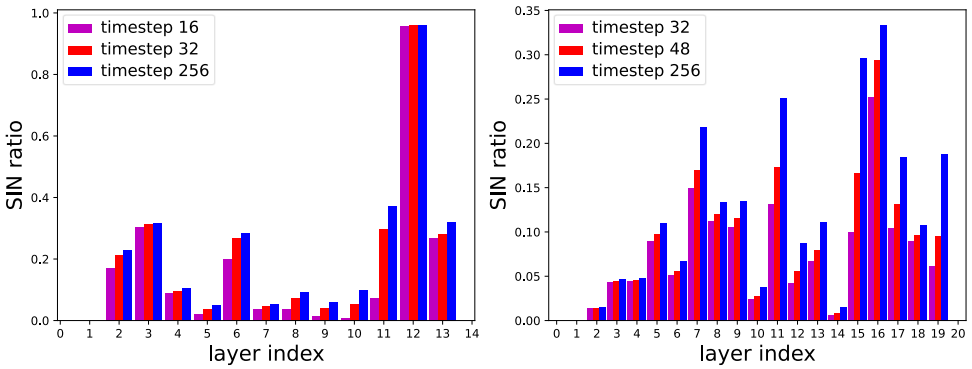
$$\begin{aligned} &\min_{V_{th,i}^l(T)} (\mathbf{o} - \mathbf{o}'), \\ &s.t. \quad \mathbf{o} = \frac{V_{th}^l}{T} \mathrm{clip}\left(\left\lfloor \frac{\sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1} T}{V_{th}^l} \right\rfloor, 0, T \right) \\ &\quad + \frac{\sum_{j\in\mathcal{R}} \sum_{t=0}^{T} W_{ij}^l \Theta_{t,j}^l}{T}, \\ &\mathbf{o}' = \mathrm{ReLU}\left(\sum_{j=0}^{M^{l-1}} W_{ij}^l r_j^{l-1} \right). \end{aligned} \tag{15}$$

There is no closed-form solution to the above problem, Li et al. [28] use grid search to heuristically find the final solution. A trivial solution is to make $V_{th,i}^l = \sum_j^{M^{l-1}} W_{ij}^l r_j^{l-1}$, which means the voltage threshold equals to input value for each neuron. With this solution, ANN can be converted to SNN only one time step. However, such SNN elicits spikes every time step and loses spike sparsity, so it is unreasonable and makes no sense, thus should not be used. In the next section, we will give a biologically rational method.

4 Methods

4.1 Multistage adaptive threshold

Based on the above analysis, the adaptive threshold, which is multistage and varies with inference time, could better facilitate information from different time steps. For each neuron, the adaptive threshold varies with firing history and input properties. The adaptive threshold $V_{th,i}^l(t+1)$ at time step $t+1$ can be described as



(a) VGG16 on CIFAR-100 *(b) ResNet20 on CIFAR-100*

Fig. 2 Neurons with SIN ratio in each layer of VGG16 and ResNet20 in CIFAR-100. Taking the total simulation time steps T=256 as an example, it can be seen that the proportion of neurons with SIN in the total neurons of this layer gradually increases with time step. More importantly, neurons with SIN in each layer are mainly activated in early time steps. In addition, neurons in deeper layers have a higher proportion of SINs

$$V^l_{th,i}(t+1) = S\Big(\tau_{mp} V^l_{th\_mp,i}(t) + \tau_{rd} V^l_{th\_rd,i}(t+1)\Big), \tag{16}$$

where τ_{mp} and τ_{rd} are the time constant of the dynamic tracking threshold $V^l_{th\_mp,i}(t)$ and dynamic evoked threshold $V^l_{th\_rd,i}(t+1)$ separately. S stands for sigmoid function. Specifically, the dynamic tracking threshold is positively correlated with the average preceding membrane potential and the dynamic evoked threshold is negatively correlated with the rate of depolarization. The first term in Eq. 16 involves computing $V^l_{th\_mp,i}(t)$, which is an intermediate result for the threshold calculation. The adaptive threshold calculation at time step $t+1$ depends on the result of $V^l_{th\_mp,i}(t)$ at the previous time step t. The second term $V^l_{th\_rd,i}(t+1)$ is another intermediate result for the threshold. Finally, the two intermediate results are weighted and combined to obtain the final $V^l_{th,i}(t+1)$.

Figure 3a gives the multistage adaptive threshold schematic diagram: when a neuron receives current input, its threshold will take average membrane potential and rate of depolarization into consideration, then the threshold will vary with these two factors. Finally, it gives a changed threshold as the adaptation to membrane potential and rate of depolarization.

4.1.1 Dynamic tracking threshold

The dynamic tracking threshold (DTT) reflects the spiking threshold that varies with firing history. In Fontaine et al. [31], the DTT is a similar first-order kinetic equation; here, we use the steady-state threshold for fitting our SNNs. Let us use $V^l_{m,i}(t)$ to denote the temporal running average membrane potential at time step t in layer l neuron i, then DTT can be described as follows

$$\begin{aligned} V^l_{th\_mp,i}(t) =& \alpha\Big(V^l_i(t) - V^l_{m,i}(t)\Big) \\ &+ V^l_T + k_a ln\left(1 + e^{\frac{V^l_i(t) - V^l_{m,i}(t)}{k_i}}\right) \end{aligned} \tag{17}$$

where α, k_a, k_i are all time constant, V^l_T is the parameter to optimize in biological model [31]. In our implementation, V^l_T is set to a fixed value, which we set to a negative constant threshold. $V^l_i(t)$ is residual membrane potential. These parameters above together determine the curvature. Dynamic tracking threshold $V^l_{th\_mp,i}(t)$ varies with membrane potential $V^l_i(t)$, which is a smooth curve because the third term in Eq. 17 introduces nonlinearity to the equation. As shown in Fig. 3a, the slopes of the DTT curve at the left and right sides of the inflection point are approximately α and $\frac{k_a}{k_i} + \alpha$, respectively.

Equation 17 shows that when the residual membrane potential increases, the threshold correspondingly increases and vice versa. The function of DTT is equivalent to a high-pass filter on the membrane potential and there are several benefits that come along with such DTT scheme; for example, small voltage fluctuation generated by a small current input will have no effect on the output spike if it is smaller than the amplitude of the threshold adaption. In the early time step, the neuron receives the synaptic current and adjusts the threshold according to the residual membrane potential, and if it receives a positive total input current but has not yet delivered a spike, the threshold is raised according to Eq. 17. Accordingly, the number of spikes decreases and the probability of spikes of inactivated neurons is reduced, thus alleviating the SIN error.

4.1.2 Dynamic evoked threshold

Dynamic evoked threshold (DET) is a concept that refers to the spiking threshold of a neuron, which changes based on the characteristics of the input it receives. Research conducted by Azouz and Gray [32] demonstrates that the voltage threshold of a neuron varies inversely with the preceding rate of depolarization, which is the change in membrane potential over time dV_m/dt. They presented this relationship by plotting a scatter graph. In the case of integrate-and-fire (IF) neurons, we use the variation of membrane potential before firing to express the preceding rate of depolarization thus DET can be expressed as

$$V^l_{th\_rd,i}(t+1) = \tau_{rd} e^{-\frac{\left(\mathbf{v}^l_i(t+1) - \mathbf{v}^l_i(t)\right)}{C}}, \tag{18}$$

where $\mathbf{V}^l_i(t)$ denotes membrane potential before firing and makes a distinction with $V^l_i(t)$, which denotes residual membrane potential after firing. C is the time constant representing the sensitivity to input membrane potential variation. The threshold decreases with input value exponentially. The equation for DET in IF neurons incorporates the time constant and exponential decay to calculate the threshold potential for the neuron at the next time step.

4.1.3 MSAT with DET and DTT

Algorithm 1 Conversion from ANN to SNN: Multistage adaptive threshold

Input: Pretrained ANN with L layer, training set
Parameter: inference time step T, spike confidence p, early time step E, threshold coefficient Cof,
Output: The multi-stage adaptive threshold SNN

Initialize threshold for all layers to zero
for s = 1 to # of samples **do**
  $a^l \leftarrow$ layer-wise activation value
  for $l = 1$ to L **do**
    $V_{th}^l \leftarrow max[V_{th}^l, max(a^l)]$
  end for
end for
for t = 1 to T **do**
  for $l = 1$ to L **do**
    spike filter $c^l \leftarrow Bernoulli(p[l])$
    for $j = 1$ to neuron number of layer l **do**
      compute DTT and DET as Eq. 17 Eq. 18
      Cof$[l][j] \leftarrow sigmoid(DTT + DET)$
      SNN.layer$[l].V_{th}[j] \leftarrow$ Cof$[l][j] * V_{th}^l$
      if fire spike and t < E **then**
        spike $= c^l *$ spike
      end if
    end for
  end for
end for

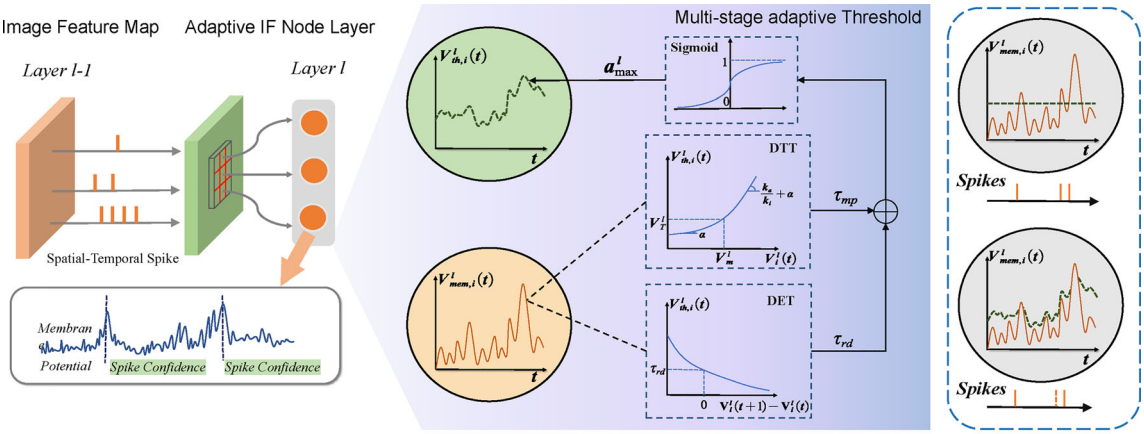
MSAT, as shown in Fig. 3a, shows that threshold-adapted neurons are insensitive to slow changes and selective to fast input variations with DET and DTT. In other words, adaptive thresholds filter out slow voltage fluctuations. The slow voltage fluctuations may come from the unexpected spikes of inactivated neurons, so it relieves the SIN error partly and reduces the total spike number thus promoting energy efficiency. On the other hand, the DET equation dynamically adjusts the spiking threshold based on the preceding rate of depolarization, which can be quantified using the membrane potential variation. Thus, the threshold variation by DET avoids endlessly increasing thresholds and reduces the fear of large quantization errors.

4.2 Spike confidence

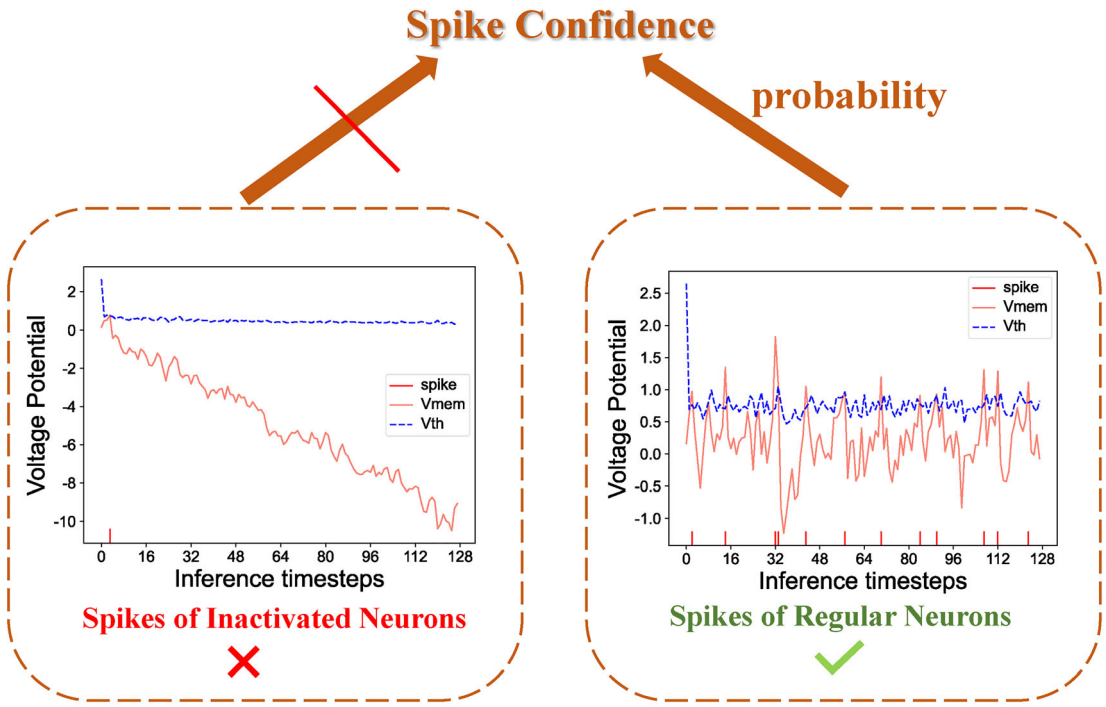
As shown in Fig. 2, the early fired spikes are not always reliable, and some parts of them are raised by SIN. Although dynamic tracking threshold, DTT can partly relieve this, neurons with SIN will not be distinguished from other normal neurons until they keep silent in longer time step. Inspired by image recognition and detection works [48, 49], we show how the SINs can be distinguished from normal neuron spikes with confidence, as shown in Fig. 3b shows. Within early time steps E, the spike confidence acts and every spike which should elicit uses this spike confidence to generate a spike filter to determine fire spike or not rather than fire directly. The value of E is determined based on the statistics of the SIN ratio. More specifically, computing the SIN ratio requires knowledge of the true activation value, i.e., for a given dataset, we run original ANN and converted SNN in parallel and then calculate SIN ratio from a batch samples in advance. Hence, the proposed method requires only one additional batch for original ANN network on a given dataset at a negligible additional computational cost. Spike confidence can be formulated as follows:

$$c_i^l \sim \text{Bernoulli}(p^l) \tag{19}$$

where p^l is spike confidence of neurons in layer l and the value of p^l equals to the opposite proportion of SIN in layer l. Random variable c_i^l, which is sampled from the Bernoulli distribution whose parameter is defined by the spike confidence, determines whether to fire a spike instead of firing directly. In this case, the accumulation of membrane potential becomes



(a) MSAT schematic diagram



(b) Spike confidence

Fig. 3 Illustration of MSAT and spike confidence. **a** MSAT is calculated from dynamic tracking threshold $V^l_{th\_mp,i}(t)$ and dynamic evoked threshold $V^l_{th\_rd,i}(t+1)$ and varies with inference time. The two neurons on the far right fire spikes under conditions of constant and dynamic thresholds, respectively. It is observable that the neuron with an adaptive threshold effectively filters out slow voltage fluctuations, with the dashed-line spikes representing the reduced spikes. **b** Spike confidence determines whether to fire a spike rather than fire directly to alleviate SIN error. Spike confidence is only implemented in the early stages

$$\begin{aligned}\tilde{\Theta}^l_{t,i} &= c^l_i * \Theta^l_{t,i},\\ V^l_i(t) &= V^l_i(t-1) + \sum_{j=0}^{M^{l-1}} V^l_{th,j}(t) W^l_{ij} \tilde{\Theta}^{l-1}_{t,j}\\ &\quad - V^l_{th,i}(t)\tilde{\Theta}^l_{t,i}.\end{aligned} \tag{20}$$

The MSAT and spike confidence ensure that all neurons adjust their thresholds and elicit spike according to the stimulus. The pseudocodes of multistage adaptive threshold are shown in Algorithm 1.

4.3 Overall conversion flowchart

We summarize the process of how trained ANNs are converted to SNNs in this section. To get SNN by converting ANN, it can be divided into a total of four steps, first obtaining the trained ANN, followed by copying the learned well learnable parameters from the ANN to the SNN. Unlike ANN, IF neurons are used in SNN instead of ReLU neurons in ANN, so the third step is that SNN uses neurons with MSAT and SC to fire spikes, and the output spikes are calculated by rate coding to get the firing rate. Finally, we get a converted SNN. In this paper, we use direct coding approach as in previous work [16] to deliver input directly to the network and allow the SNN to process and transmit the information more efficiently. The entire flow of the conversion is shown in Fig. 4 and the pseudocode of proposed method is shown in Algorithm 2. It is worth noting that the whole process does not require retraining of the network.

Algorithm 2 Conversion from ANN to SNN: PyTorch-style pseudocode

```
# IFNode: Integrate-and-Fire neuron in SNN
Class IFNode(nn.Module):
    def forward(self,input):
        integral(input) # accumulate membrane potential
        calculate_spike(input) # fire spike with MSAT and SC
        return self.threshold * self.spike

# get converted SNN
for layer in net.modules(): # net: a well-trained ANN model
    if isinstance(layer, ReLU):
        Replace_ReLU_with_IFNode(layer)

# SNN inference
for x, label in loader: # load a minibatch x with N samples
    net.reset() # reset SNN for each batch
    for t in step:
        pred += net(x)
    pred /= step # compute firing rate
    compue_accuracy(pred, label) # compute accuracy for minibatch
```

5 Experiment

Here, we conduct experiments on CIFAR-10 [50], CIFAR-100 [50], and ImageNet [51] datasets to demonstrate the effectiveness of our proposed method. To demonstrate the generalizability of the proposed method outside the visual task, we also perform sentiment classification and speech recognition experiments on the IDBM dataset [52] and the Google speech command dataset [53], respectively. We use data augmentation such as random horizontal flip, Cutout [54], and AutoAugment [55]. Our batch size is 128 and the total training epoch is 300. We use stochastic gradient descent (SGD) as the optimizer, with an initial learning rate of 0.1 and uses a cosine decay strategy. VGG16 [56], ResNet20, and ResNet34 [57] are used for ANN as in previous works for comparison. We used IF neurons for our experiments and our code implementation of deep SNNs is based on the open-source SNN framework BrainCog [58].

In the proposed adaptive thresholds, there are six hyperparameters and we give their meanings and values used in this paper in Table 2. Although hyperparameters are manually set, they do not vary with the dataset and setting these parameters is not difficult based on our experiments; our method works well for a wide value range of these parameters. Although these parameters can be set to trainable, we choose to manually set them for biological plausibility according to [31, 32].

5.1 Comparison with the state of the art

In order not to do too many restrictions on origin ANN, we choose to save the topology and use the original weights in

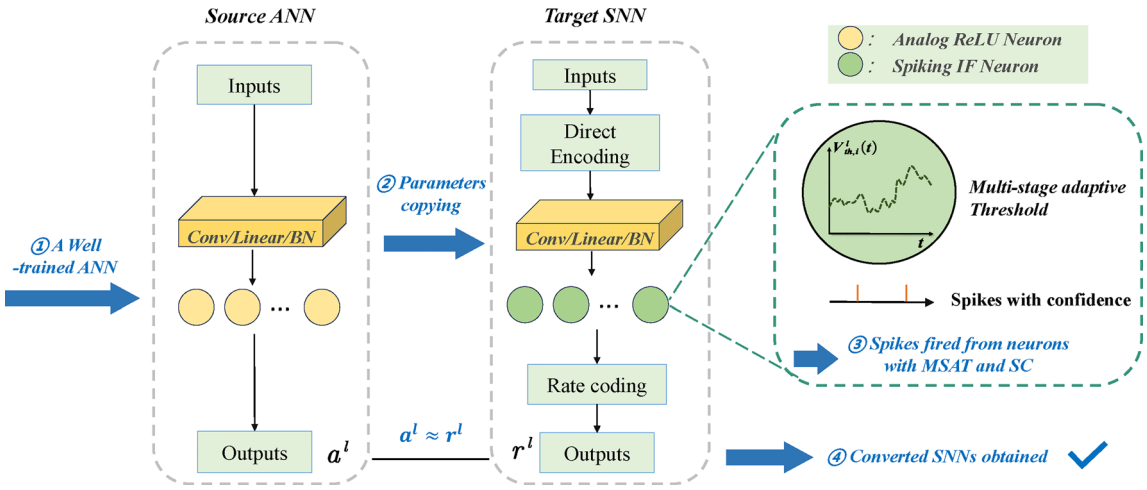


Fig. 4 The overall conversion flowchart with our methods. The three dashed boxes from left to right represent: the source ANN, the target SNN to be obtained, spikes fired by neuron with MSAT and SC. We use yellow and green neurons to represent ReLU neurons in ANN and IF neurons in SNN, respectively. Direct encoding delivers input directly to the network and the output value is obtained by rate coding, i.e., calculating the firing rate

Table 2 Summary of hyperparameters on different network

Symbol	Definition	VGG16	ResNet20	ResNet34
α	Left side slope	0.03	0.3	1.0
k_a	Right side slope hyperparameter	1	1	1.0
k_i	Right side slope hyperparameter	1.0	1.0	1.0
C	Input sensitivity	5.0	5.0	5.0
τ_{mp}	Coefficient of DTT	1	0.5	0.5
τ_{rd}	Coefficient of DET	1	0.5	0.5

Table 3 Experimental results on CIFAR-10 and CIFAR-100, DT means dynamic threshold

Method	Use DT	ANN	SNN	Loss	Step	ANN	SNN	Loss	Step
		VGG16, CIFAR-10				ResNet20, CIFAR-10			
RMP-SNN [40]	×	93.63%	93.63%	<0.01%	2048	91.47%	91.36%	0.11%	2048
TSC [59]	×	93.63%	93.63%	<0.01%	2048	91.47%	91.42%	0.05%	2048
Opt [43]	×	92.34%	92.24%	0.1%	128	93.61%	93.56%	0.05%	128
Calibration [28]	✓	95.72%	95.65%	0.07%	128	95.46%	95.42%	0.04%	128
Burst [27]	×	95.74%	95.75%	0.02%	256	96.56%	96.59%	−0.03%	<256
DTIF [45]	✓	92.37%	92.05%	0.32%	288	–	–	–	–
Ours	✓	95.45%	95.45%	**0.00%**	**112**	96.37%	96.36%	**0.01%**	**174**
		VGG16, CIFAR-100				ResNet20, CIFAR-100			
RMP-SNN [40]	×	71.22%	70.93%	0.29%	2048	68.72%	67.82%	0.9%	2048
TSC [59]	×	71.22%	70.97%	0.25%	2048	68.72%	68.18%	0.54%	2048
Opt [43]	×	70.49%	70.47%	0.02%	128	69.80%	69.49%	0.31%	128
Calibration [28]	✓	77.89%	77.79%	0.1%	>512	77.16%	77.29%	−0.13%	64
Burst [27]	×	78.49%	78.66%	−0.17%	128	80.69%	80.72%	−0.03%	<256
DTIF [45]	✓	67.45%	67.80%	−0.35%	184	–	–	–	–
Ours	✓	78.49%	78.50%	**−0.01%**	**224**	80.69%	80.70%	**−0.01%**	**252**

Bolded text signifies the highest result in comparison to others

Table 4 Performance of our method as compared to other dynamic threshold work at different time steps on the CIFAR-10 dataset with VGG16 architecture

Method	Network	T=16	T=32	T=64	T=128	T=256	ANN acc
Calibration	VGG16	–	93.71%	95.14%	95.65%	95.79%	95.72%
DTIF	VGG16	71.10%	86.37%	90.61%	91.56%	91.89%	92.37%
Ours	VGG16	**73.42%**	**93.87%**	**95.20%**	95.44%	95.50%	95.45%

Bolded text signifies the highest result in comparison to others

Table 5 Experimental results on ImageNet with ResNet34

Method	Use DT	ANN	SNN	Loss	Step
RMP-SNN	×	70.64%	69.89%	0.75%	>2048
TSC	×	70.64%	69.93%	0.71%	>2048
Opt	×	75.66%	75.44%	0.22%	>2048
Calibration	✓	75.66%	75.45%	0.21%	>2048
Burst	×	75.16%	74.94%	0.22%	256
Ours	✓	75.16%	74.93%	0.23%	2045

the target ANN for universal conversion without retraining ANN. In this way, we compare the state-of-the-art approaches including TSC [59], RMP-SNN [40], Opt [43], Calibration [28], Burst [27], DTIF [45]. As shown in Table 3, with the proposed method, all conversions are achieved with minimal accuracy degradation and shorter latency.

To demonstrate the difference in inference speed between our method and other dynamic threshold work, we compare our method with calibration and DTIF on CIFAR-10 dataset. The results are shown in Table 4. It can be seen that our method achieves best performance in the early time step, which is attributed to MSAT and spike confidence. It is worth noting that though calibration [28] also uses the dynamic threshold, our method does not need to search for the best threshold at each time step, and therefore results in less computational cost. Finally, we validate the robustness of our method on ImageNet. As shown in Table 5, we also achieve comparative performance with SOTA methods and nearly lossless conversion from the source ANN.

5.2 Performance on non-visual domains

Our work aims to contribute a biologically adaptive threshold system for ANN-SNN conversion, not limited to computer vision tasks and so MSAT is equally competent in other tasks. To show this, we further evaluate our method on non-visual domains, including natural language processing and speech recognition.

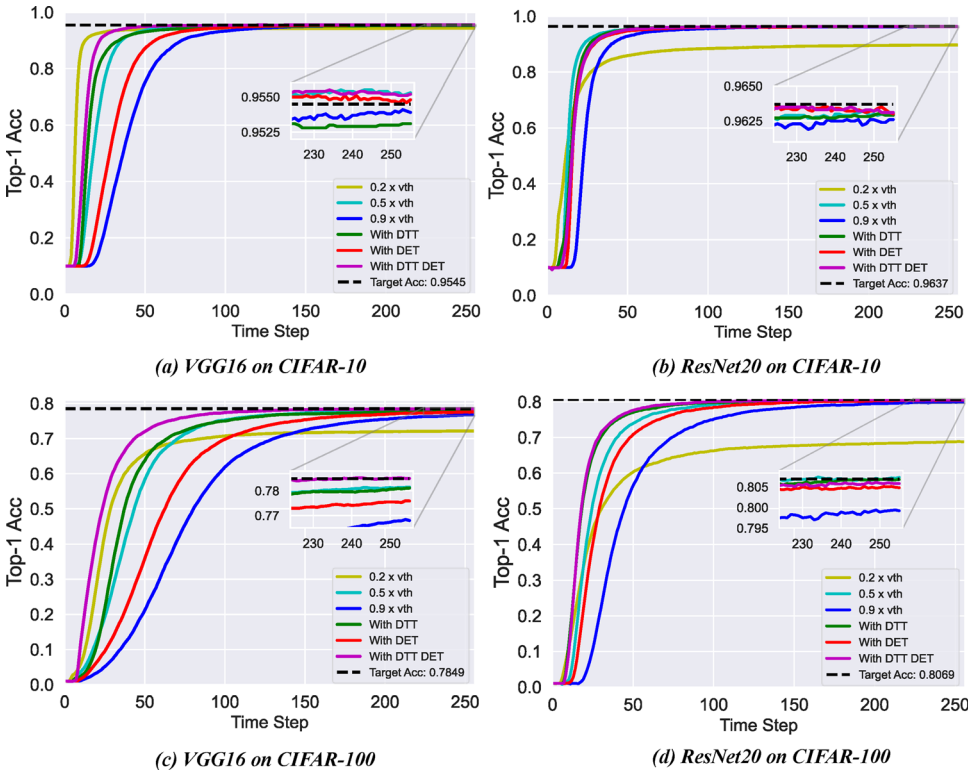
For natural language processing, we choose IMDB dataset for the sentiment classification task, and the network structure is three-layer bidirectional LSTM built by ourselves. For the speech processing, we evaluate the speech recognition task and select the Google speech command dataset. The dataset consisted of 105,829 utterances of 35 words, each stored as a one-second (or less) WAVE format file. The specific architecture is modeled after the M5 network architecture described in Dai et al. [60]. We report the performance of MSAT on these two tasks and compare it with RMP-SNN. The results can be seen in Table 6.

Compared to RMP-SNN, MSAT can achieve lossless conversion much faster, even in only half the time steps, for example, 7 steps on IDBM dataset for LSTM. On complex tasks, such as speech recognition with 35 classes, MSAT performs better at the same time step. These results are consistent with earlier analyses and validate the effectiveness of MSAT.

Table 6 Experimental results on non-visual task

Task	Dataset	Network	ANN accuracy	Method	Step	SNN accuracy
NLP	IDBM	LSTM	87.22%	RMP-SNN[1]	11	87.23%
Sentiment classification				MSAT	7	87.25%
Speech	Speech commands	M5	87.81%	RMP-SNN[1]	2048	86.83%
Speech recognition				MSAT	2048	87.33%

1. Our implemented



(a) VGG16 on CIFAR-10 *(b) ResNet20 on CIFAR-10*

(c) VGG16 on CIFAR-100 *(d) ResNet20 on CIFAR-100*

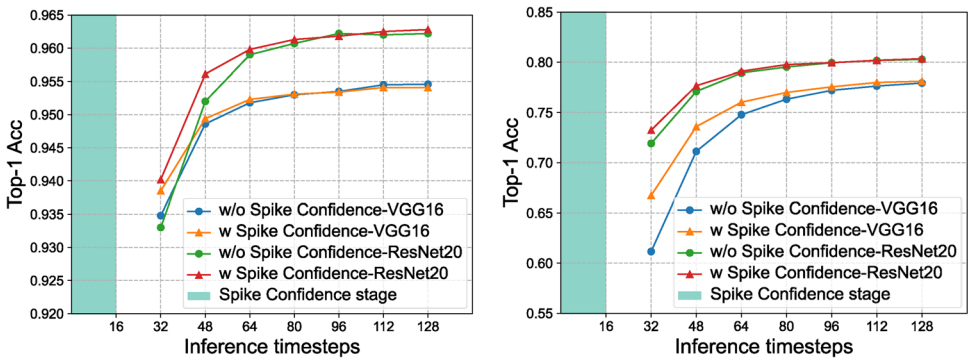
Fig. 5 The accuracy curves on CIFAR-10 and CIFAR-100 datasets

Table 7 Averaged Number of SIN (ANS) and classification accuracy on different networks and datasets.

Metric	VGG16		ResNet20	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
w/o SC ANS	2.92	4.467	1.972	6.309
w/ SC ANS	**2.343**$_{-0.577}$	**2.514**$_{-1.953}$	**1.558**$_{-0.414}$	**3.063**$_{-3.246}$
w/o SC Acc (T=32)	93.48	61.16	93.30	71.92
w/ SC Acc (T=32)	**93.85**$_{+0.37}$	**66.73**$_{+5.57}$	**94.02**$_{+0.72}$	**73.25**$_{+1.33}$

The subscripts mean the decreased ANS and increased accuracy compared to the baseline in 1st and 3rd rows

Bolded text signifies the highest result in comparison to others



(a) Spike Confidence on CIFAR-10 *(b) Spike Confidence on CIFAR-100*

Fig. 6 Classification accuracy improves with spike confidence. Spike confidence is used within the time steps of the green area

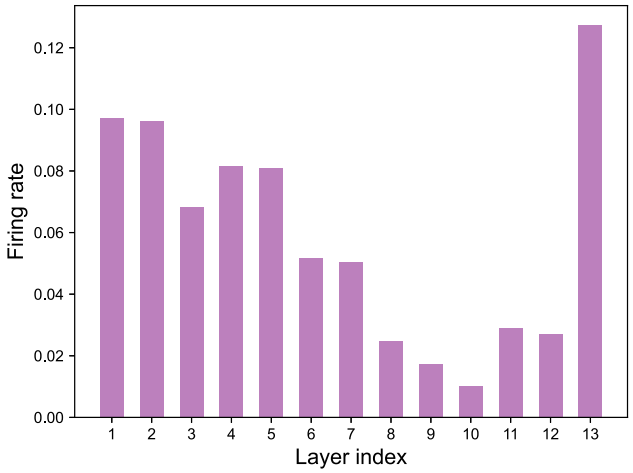


Fig. 7 Firing rate in each layer of VGG16 network on CIFAR-100 dataset

5.3 Ablation study

In our method, V_{th}^{l} is chosen as the maximum activation value and as the initial value of the multistage adaptive threshold. In Fig. 5, the dotted lines indicate the target ANN accuracy, from which we observe the DTT, DET, and fixed threshold impact on classification accuracy. It is intuitive that with either DTT or DET, we could achieve faster convergence rate, as shown in Fig. 5b, d. Instead, with the heuristic method, it is hard to achieve high accuracy and fast inference speed simultaneously. For example, $0.2V_{th}$ in Fig. 5a, b (yellow lines) lie above other color lines in early stage and have a shorter latency. However, yellow lines do not appear in the zoomed-in box and cannot touch dotted lines, which means that the converted SNN threshold with $0.2V_{th}$ suffers from larger performance degradation. Although a manually selected threshold might achieve a satisfactory result, it needs lots of computation to search or carefully design for an appropriate threshold.

The adaptive threshold MSAT with both DTT and DET could have a faster inference to achieve the target ANN accuracy. This shows that the MSAT could find an appropriate threshold at each time step to reduce the quantization error while not importing larger clip error.

Moreover, we validate the effect of spike confidence on accuracy in early time steps. In our experiment, we apply spike confidence to the last IF neuron layer because the last layer is directly responsible for calculating results. The spike confidence stage holds 16 time steps. As we mentioned before, SIN take a large proportion in each layer, so a large part of the error comes from SIN besides quantization and clip error. We use averaged number of SIN (ANS) over each neuron and the exclusion of spike confidence (SC) in both ANS and classification accuracy as baseline. Table 7 shows that spike confidence could reduce the averaged number of SIN and improve classification accuracy. This verifies that SNN in the early time steps could elicit fewer error spikes and benefit from spike confidence.

Figure 6 demonstrates the impact of adding spike confidence on performance. The improvement on CIFAR-10 is not as significant as on CIFAR-100, mainly because the SIN proportion in CIFAR-10 is already quite smaller than that in CIFAR-100. To show this, we statistic the neurons with SIN ratio in the last layer for CIFAR-10-VGG16, CIFAR-10-ResNet20, CIFAR-100-VGG16, and CIFAR-100-ResNet20, respectively. The ratio is 0.059, 0.063, 0.617, and 0.506, respectively, which means that SIN error is a very small part in CIFAR-10 whose influence on performance degradation is not as significant as in CIFAR-100. This also explains the different degrees of performance improvement on the two datasets by spike confidence.

5.4 Energy efficiency and sparsity

Although a realistic calculation and simulation on a neuromorphic chip for energy consumption analysis with proposed method is out of the scope of our work, we provide a theoretical estimation to demonstrate that the proposed method is energy efficient. Since the computation in SNN depends on the number of spikes, the energy consumption of the SNN is positively correlated with the firing rate of the neurons. Therefore, firstly, we show the firing rate of each layer of the converted SNN network obtained with the proposed method. We chose the VGG16 on CIFAR-100 dataset with time step T=64, and the firing rates of neurons in each layer are shown in Fig. 7. It can be observed that the average firing rate of the spikes is 0.0585, with the maximum and minimum spike firing rates in each layer being 0.010 and 0.127, respectively. This demonstrates the sparsity of spike activity.

In ANNs, each computational operation includes one floating-point (FP) multiplication and one FP addition (MAC). In contrast, in SNNs, each operation only involves one FP addition due to the binary nature of spikes. To quantitatively demonstrate the energy efficiency of our method, in the same way as with Rathi and Roy [61], we first calculate the number of layer l computational operations in the ANN.

Table 8 Comparison of different methods for energy consumption

Method	Mean firing rate	OP^1_{ANN}	$\sum_{l=2}^{L} OP^l_{\text{ANN}}$	$\sum_{l=2}^{L} OP^l_{\text{SNN}}$	Energy ratio$_{\text{SNN/ANN}}$
Burst [27]	0.00703	1.77M	311.48M	1486.79M	0.934
Ours	**0.00585**	1.77M	311.48M	**1137.53M**	**0.716**

Bolded text signifies the highest result in comparison to others

$$OP^l_{\text{ANN}} = \begin{cases} k_{\text{h}} \times k_{\text{w}} \times c_{\text{in}} \\ \quad \times c_{\text{out}} \times w_{\text{out}} \times h_{\text{out}}, & \text{if convolution layer} \\ f_{\text{in}} \times f_{\text{out}}, & \text{if fully connected layer} \end{cases} \tag{21}$$

Next we calculate the number of computational operations in the l_{th} layer of the SNN. In our implementation, the first layer of the SNN is directly encoding the input, so the energy consumption in this layer is still calculated as 4.6 PJ per operation. The calculation formula can be written as.

$$OP^l_{\text{SNN}} = \begin{cases} OP^1_{\text{ANN}}, & \text{if } l = 1 \\ \frac{\sum_{j=0}^{M^{l-1}} \sum_{t=0}^{T} \Theta^l_{t,j}}{M^l} \times OP^l_{\text{ANN}}, & \text{if} l > 1 \end{cases} \tag{22}$$

we adopt the results from Horowitz [62]: in 45 nm CMOS technology, the energy cost per computation in ANNs, i.e., a 32-bit floating-point MAC, is 4.6 pJ; on the other hand, the computation in SNNs is more energy-efficient, with a 32-bit floating-point AC consuming only 0.9 pJ.

Therefore, the energy consumption ratio of SNN to ANN can be written as:

$$\begin{aligned} \text{Energy Ratio}_{\text{SNN/ANN}} &= \frac{\sum_l \text{Energy}^l_{\text{SNN}}}{\sum_l \text{Energy}^l_{\text{ANN}}} \\ &= \frac{4.6OP^1_{\text{ANN}} + \sum_{l=2}^{L} 0.9OP^l_{\text{SNN}}}{\sum_{l=1}^{L} 4.6OP^1_{\text{ANN}}} \end{aligned} \tag{23}$$

By incorporating the per-layer spike firing frequencies from Fig. 7 into our calculations, we find that SNN consumes 0.716 energy compared to the target ANN, which implies that the SNN has a lower power consumption than the ANN. Table 8 shows our result. Furthermore, we reproduced the method of Li and Zeng [27]. Under the same conditions, the energy consumption ratio of SNN to ANN is 0.934. The reduced power consumption can be attributed to our MSAT and SC methods.

6 Conclusion

This paper proposes a biologically multistage adaptive threshold into ANN-SNN conversion and demonstrates its advantages in terms of accuracy-latency trade-off. In addition, we focus on the SIN error that occupies a large proportion in the early period and accounts for accuracy degradation but is rarely explored in existing works. Through statistical analysis of SIN errors, we propose to mitigate SIN errors by using spike confidence. Our experiments show that the converted SNN with MSAT and spike confidence has comparable accuracy to the state of the art while with lower latency and power consumption. Good performance on classification tasks in non-visual domains also demonstrates the generality of the proposed approach.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Grant No. 62372453).

Data availability The datasets analyzed during the current study are publicly available. CIFAR-10 dataset and CIFAR-100 dataset: https://www.cs.toronto.edu/kriz/cifar.html. ImageNet dataset: https://www.image-net.org/download.php Brain-Cog framework: https://github.com/BrainCog-X/Brain-Cog

Code availability The code used in this study is available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
2. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H (2018) State-of-the-art in artificial neural network applications: a survey. Heliyon 4(11):00938
3. Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53:5455–5516
4. Akopyan F, Sawada J, Cassidy A, Alvarez-Icaza R, Arthur J, Merolla P, Imam N, Nakamura Y, Datta P, Nam G-J et al (2015) Truenorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. IEEE Trans Comput Aided Des Integr Circuits Syst 34(10):1537–1557
5. Davies M, Srinivasa N, Lin T-H, Chinya G, Cao Y, Choday SH, Dimou G, Joshi P, Imam N, Jain S et al (2018) Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro 38(1):82–99

6. Pei J, Deng L, Song S, Zhao M, Zhang Y, Wu S, Wang G, Zou Z, Wu Z, He W et al (2019) Towards artificial general intelligence with hybrid tianjic chip architecture. Nature 572(7767):106–111
7. Lee JH, Delbruck T, Pfeiffer M (2016) Training deep spiking neural networks using backpropagation. Front Neurosci 10:508
8. Wu Y, Deng L, Li G, Zhu J, Shi L (2018) Spatio-temporal backpropagation for training high-performance spiking neural networks. Front Neurosci 12:331
9. Wu Y, Deng L, Li G, Zhu J, Xie Y, Shi L (2019) Direct training for spiking neural networks: faster, larger, better. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 1311–1318
10. Zhang W, Li P (2020) Temporal spike sequence learning via backpropagation for deep spiking neural networks. Adv Neural Inf Process Syst 33:12022–12033
11. Shen G, Zhao D, Zeng Y (2022) Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks. Patterns 3(6):100522
12. Li Y, Guo Y, Zhang S, Deng S, Hai Y, Gu S (2021) Differentiable spike: rethinking gradient-descent for training spiking neural networks. Adv Neural Inf Process Syst 34:23426–23439
13. Wu Z, Zhang H, Lin Y, Li G, Wang M, Tang Y (2021) Liaf-net: leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. IEEE Trans Neural Netw Learn Syst 33(11):6249–6262
14. Fang W, Yu Z, Chen Y, Huang T, Masquelier T, Tian Y (2021) Deep residual learning in spiking neural networks. Adv Neural Inform Process Syst 34:21056–69
15. Duan C, Ding J, Chen S, Yu Z, Huang T (2022) Temporal effective batch normalization in spiking neural networks. Adv Neural Inform Process Syst 35:34377–90
16. Deng S, Li Y, Zhang S, Gu S (2022) Temporal efficient training of spiking neural network via gradient re-weighting. arXiv preprint arXiv:2202.11946
17. Caporale N, Dan Y et al (2008) Spike timing-dependent plasticity: a hebbian learning rule. Annu Rev Neurosci 31(1):25–46
18. Diehl PU, Cook M (2015) Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Front Comput Neurosci 9:99
19. Hao Y, Huang X, Dong M, Xu B (2020) A biologically plausible supervised learning method for spiking neural networks using the symmetric stdp rule. Neural Netw 121:387–395
20. Zhao D, Zeng Y, Zhang T, Shi M, Zhao F (2020) Glsnn: a multi-layer spiking neural network based on global feedback alignment and local stdp plasticity. Front Comput Neurosci 14:576841
21. Datta G, Liu Z, Beerel PA (2022) Hoyer regularizer is all you need for ultra low-latency spiking neural networks. arXiv preprint arXiv:2212.10170
22. Lien H-H, Chang T-S (2022) Sparse compressed spiking neural network accelerator for object detection. IEEE Trans Circuits Syst I Regul Pap 69(5):2060–2069
23. Diehl PU, Neil D, Binas J, Cook M, Liu SC, Pfeiffer M (2015) Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: 2015 International joint conference on neural networks (IJCNN), pp 1–8. IEEE
24. Sengupta A, Ye Y, Wang R, Liu C, Roy K (2019) Going deeper in spiking neural networks: Vgg and residual architectures. Front Neurosci 13:95
25. Yu Q, Ma C, Song S, Zhang G, Dang J, Tan KC (2021) Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes. IEEE Trans Neural Netw Learn Syst 33(4):1714–1726
26. Bu T, Fang W, Ding J, Dai P, Yu Z, Huang T (2021) Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In: International conference on learning representations
27. Li Y, Zeng Y (2022) Efficient and accurate conversion of spiking neural network with burst spikes. arXiv preprint arXiv:2204.13271
28. Li Y, Deng S, Dong X, Gong R, Gu S (2021) A free lunch from ann: towards efficient, accurate spiking neural networks calibration. In: International conference on machine learning, pp 6316–6325. PMLR
29. Kim S, Park S, Na B, Kim J, Yoon S (2020) Towards fast and accurate object detection in bio-inspired spiking neural networks through bayesian optimization. IEEE Access 9:2633–2643
30. Bu T, Ding J, Yu Z, Huang T (2022) Optimized potential initialization for low-latency spiking neural networks. arXiv preprint arXiv:2202.01440
31. Fontaine B, Peña JL, Brette R (2014) Spike-threshold adaptation predicted by membrane potential dynamics in vivo. PLoS Comput Biol 10(4):1003560
32. Azouz R, Gray CM (2000) Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons in vivo. Proc Natl Acad Sci 97(14):8110–8115
33. Henze D, Buzsáki G (2001) Action potential threshold of hippocampal pyramidal cells in vivo is increased by recent spiking activity. Neuroscience 105(1):121–130
34. Azouz R, Gray CM (2003) Adaptive coincidence detection and dynamic gain control in visual cortical neurons in vivo. Neuron 37(3):513–523
35. Pena JL, Konishi M (2002) From postsynaptic potentials to spikes in the genesis of auditory spatial receptive fields. J Neurosci 22(13):5652–5658
36. Wilent WB, Contreras D (2005) Stimulus-dependent changes in spike threshold enhance feature selectivity in rat barrel cortex neurons. J Neurosci 25(11):2983–2991
37. Pérez-Carrasco JA, Zhao B, Serrano C, Acha B, Serrano-Gotarredona T, Chen S, Linares-Barranco B (2013) 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):2706–2719
38. Burkitt AN (2006) A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. Biol Cybern 95(1):1–19
39. Rueckauer B, Lungu I-A, Hu Y, Pfeiffer M, Liu S-C (2017) Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Front Neurosci 11:682
40. Han B, Srinivasan G, Roy K (2020) Rmp-snn: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 13558–13567
41. Rueckauer B, Liu SC (2018) Conversion of analog to spiking neural networks using sparse temporal coding. In: 2018 IEEE International symposium on circuits and systems (ISCAS), pp 1–5. IEEE
42. Ding J, Yu Z, Tian Y, Huang T (2021) Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. arXiv preprint arXiv:2105.11654
43. Deng S, Gu S (2021) Optimal conversion of conventional artificial neural networks to spiking neural networks. arXiv preprint arXiv:2103.00476
44. Meng Q, Yan S, Xiao M, Wang Y, Lin Z, Luo Z-Q (2022) Training much deeper spiking neural networks with a small number of time-steps. Neural Netw 153:254–268
45. Wu X, Zhao Y, Song Y, Jiang Y, Bai Y, Li X, Zhou Y, Yang X, Hao Q (2023) Dynamic threshold integrate and fire neuron model for low latency spiking neural networks. Neurocomputing 544:126247

46. Ding J, Dong B, Heide F, Ding Y, Zhou Y, Yin B, Yang X (2022) Biologically inspired dynamic thresholds for spiking neural networks. Adv Neural Inf Process Syst 35:6090–6103
47. Li Y, Zeng Y, Zhao D (2021) Bsnn: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons. arXiv preprint arXiv:2105.12917
48. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: Single shot multibox detector. In: European conference on computer vision, pp 21–37. Springer
49. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 779–788
50. Krizhevsky A, Hinton G, et al (2009) Learning multiple layers of features from tiny images
51. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
52. Maas A, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, pp 142–150
53. Warden P (2018) Speech commands: a dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209
54. DeVries T, Taylor GW (2017) Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552
55. Cubuk ED, Zoph B, Mane D, Vasudevan V, Le QV (2019) Autoaugment: learning augmentation strategies from data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 113–123
56. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
57. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 770–778
58. Zeng Y, Zhao D, Zhao F, Shen G, Dong Y, Lu E, Zhang Q, Sun Y, Liang Q, Zhao Y, et al (2022) Braincog: a spiking neural network based brain-inspired cognitive intelligence engine for brain-inspired ai and brain simulation. arXiv preprint arXiv:2207.08533
59. Han B, Roy K (2020) Deep spiking neural network: Energy efficiency through time based coding. In: European conference on computer vision, pp 388–404. Springer
60. Dai W, Dai C, Qu S, Li J, Das S (2017) Very deep convolutional neural networks for raw waveforms. In: 2017 IEEE International conference on acoustics, speech and signal processing (ICASSP), pp 421–425. IEEE
61. Rathi N, Roy K (2020) Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. arXiv preprint arXiv:2008.03658
62. Horowitz M (2014) 1.1 computing's energy problem (and what we can do about it). In: 2014 IEEE International solid-state circuits conference digest of technical papers (ISSCC), pp 10–14. IEEE

Authors and Affiliations

Xiang He[1,2,3] · Yang Li[1,2,3] · Dongcheng Zhao[1,2] · Qingqun Kong[1,2,3,4] · Yi Zeng[1,2,3,4,5]

✉ Qingqun Kong
qingqun.kong@ia.ac.cn

✉ Yi Zeng
yi.zeng@ia.ac.cn

Xiang He
hexiang2021@ia.ac.cn

Yang Li
liyang2019@ia.ac.cn

Dongcheng Zhao
zhaodongcheng2016@ia.ac.cn

1 Brain-inspired Cognitive Intelligence Lab, Chinese Academy of Sciences, Beijing, China

2 Center for Long-term Artificial Intelligence, Beijing, China

3 School of Artificial Intelligence, Chinese Academy of Sciences, Beijing, China

4 School of Future Technology, Chinese Academy of Sciences, Beijing, China

5 Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Chinese Academy of Sciences, Shanghai, China