Machine Intelligence Research
www.mi-research.net

20(3), June 2023, 435-446
DOI: 10.1007/s11633-022-1388-2

Symmetric-threshold ReLU for Fast and Nearly Lossless ANN-SNN Conversion

Jianing Han[1] Ziming Wang[1] Jiangrong Shen[1] Huajin Tang[1,2]

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China

[2] Zhejiang Lab, Hangzhou 311121, China

Abstract: The artificial neural network-spiking neural network (ANN-SNN) conversion, as an efficient algorithm for deep SNNs training, promotes the performance of shallow SNNs, and expands the application in various tasks. However, the existing conversion methods still face the problem of large conversion error within low conversion time steps. In this paper, a heuristic symmetric-threshold rectified linear unit (stReLU) activation function for ANNs is proposed, based on the intrinsically different responses between the integrate-and-fire (IF) neurons in SNNs and the activation functions in ANNs. The negative threshold in stReLU can guarantee the conversion of negative activations, and the symmetric thresholds enable positive error to offset negative error between activation value and spike firing rate, thus reducing the conversion error from ANNs to SNNs. The lossless conversion from ANNs with stReLU to SNNs is demonstrated by theoretical formulation. By contrasting stReLU with asymmetric-threshold LeakyReLU and threshold ReLU, the effectiveness of symmetric thresholds is further explored. The results show that ANNs with stReLU can decrease the conversion error and achieve nearly lossless conversion based on the MNIST, Fashion-MNIST, and CIFAR10 datasets, with 6× to 250 speedup compared with other methods. Moreover, the comparison of energy consumption between ANNs and SNNs indicates that this novel conversion algorithm can also significantly reduce energy consumption.

Keywords: Symmetric-threshold rectified linear unit (stReLU), deep spiking neural networks, artificial neural network-spiking neural network (ANN-SNN) conversion, lossless conversion, double thresholds.

Citation: J. Han, Z. Wang, J. Shen, H. Tang. Symmetric-threshold ReLU for fast and nearly lossless ANN-SNN conversion. *Machine Intelligence Research*, vol.20, no.3, pp.435–446, 2023. http://doi.org/10.1007/s11633-022-1388-2

1 Introduction

Artificial neural networks (ANNs)[1], such as convolutional neural networks[2] and recurrent neural networks[3], have achieved remarkable progress in various large-scale tasks[4–6]. However, powerful efficiency comes at the expense of high energy consumption. Compared with ANNs, spiking neural networks (SNNs)[7, 8] employ the spike to transmit information, resembling the information processing mechanism of human brain, which overcomes the problem of high power consumption in ANNs. SNNs can save several orders of magnitude of energy when deployed on specialized neuromorphic hardwares compared to ANNs[9, 10]. Existing training methods of SNNs generally include direct training and indirect training. In terms of direct training, both unsupervised learning algorithms, e.g., spike-timing-dependent plasticity (STDP)[11], and supervised learning algorithms such as spatio-temporal credit assignment (STCA)[12], spatio-temporal back-propagation (STBP)[13] for SNNs which are constrained to relatively shallow structures and limited performance on large-scale datasets. The indirect training methods exploit the knowledge from ANNs to improve the efficiency of SNNs, and thus have advantages in the feature extraction and structure scalability.

Indirect training methods for SNNs mainly refer to the conversion methods from ANNs to SNNs. This approach trains an ANN utilizing standard backpropagation methods, and then transfers the parameters to SNNs with an equivalent architecture as shown in Fig. 1. The conversion algorithm is not limited by the depth of networks and can be scaled up to deep structures to implement complex tasks. Cao et al.[14] found that the firing rate of neurons in SNNs can approximate the activations of neurons in ANNs within enough time steps, which is the foundation of ANN-SNN conversion. However, the performance loss during ANN-SNN conversion has been the bottleneck problem to be solved. Diehl et al.[15] thought that the two main sources of conversion error are under activation (Fig. 1(b)) caused by postsynaptic neurons that do not accumulate enough membrane potential for spike firing, and over activation (Fig. 1(b)) caused by postsynaptic neurons that integrate too many presynaptic spikes at a given time, leading their membrane potential to be-

Research Article
Manuscript received on July 24, 2022; accepted on October 28, 2022; published online on March 31, 2023
Recommended by Associate Editor Cheng-Lin Liu
Colored figures are available in the online version at https://link.springer.com/journal/11633

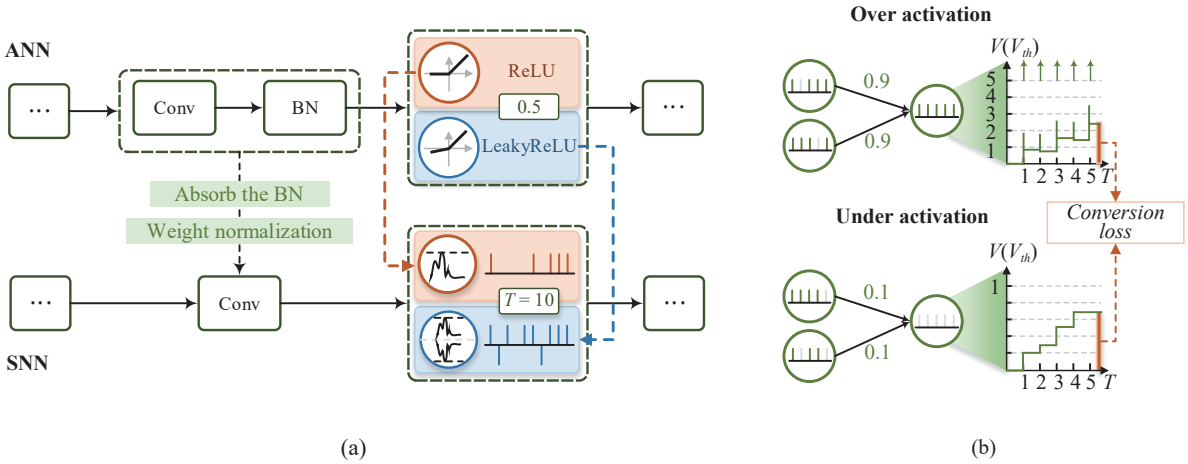

Fig. 1 Train an equivalent ANN, and then transfer the parameters to the SNN. (a) In the standard ANN-SNN conversion process, batch normalization layers should be absorbed into convolution layers, followed by weight normalization. ReLU and LeakyReLU are commonly used activation functions. (b) Spike patterns (left) and detailed membrane potential curves (right) of two main sources of conversion error[16].

come much higher than the threshold, and eventually leaving the residual membrane potential. After that, researchers have been focusing on improving the conversion efficiency, i.e., decreasing the conversion error.

Different methods have been introduced to decrease the conversion error. Many methods for weight normalization have been presented by researchers to address the problem of over activation. Diehl et al.[15] proposed model-based normalization and data-based normalization to scale the weights, which either utilizes the maximum positive activations or the maximum weights. However, max value normalization is prone to be influenced by singular outlier samples. Rueckauer et al.[17] proposed a robust percentile algorithm that selects the p-th maximum activation as the scale factor. Sengupta et al.[18] adopted the maximum SNN activations rather than ANN activations to realize Spike-norm. Instead of layer-wise weight normalization, Kim et al.[19] introduced channel-wise weight normalization based on the significantly different firing rate distribution of each channel. Nevertheless, the range of ANN activations is constrained to [0, 1] with weight normalization, which dramatically minimizes the conversion error while significantly increasing latency to achieve lossless conversion.

As a result, researchers have set their sights on reducing the number of time steps. Some works attempt to fit the transformed SNNs to ANNs. Li et al.[20] adjusted the parameters of SNNs, such as weights, bias, and initial membrane potential, to decrease the conversion error layer by layer while reducing the time steps required to achieve lossless conversion. The original ANNs are also directly altered to be fundamentally closer to the SNNs. By modifying the rectified linear unit (ReLU) function, clamped and quantized (CQ)[21] training of ANNs is performed to constrain the input and activation values, thereby reducing the gap between the integrate-and-fire (IF) neurons of SNNs and the activation functions of ANNs. Ding et al.[22] replaced the ReLU activation function with rate norm layer. The quantization clip-floor-shift activation function was introduced to replace ReLU by Bu et al.[23] to achieve ultra-low latency SNNs. Other researchers consider the loss of temporal dimension information during conversion and apply temporal coding in the converted SNNs, such as time-to-first-spiking coding[24], phase coding[25] and burst spikes[26]. These methods, however, are limited to the ReLU activation function in ANNs and are unable to transform negative activations into spikes, hindering the properties of ANNs to promote the development of SNNs. Therefore, Yu et al.[27] transferred ANNs with leaky rectified linear unit (LeakyReLU) to SNNs based on theoretical formulas. TerMapping, a double-threshold conversion model, was proposed to convert negative activations. To reduce transfer latency, they also proposed AugMapping, which uses extra spike coefficients to carry additional information including both polarity and the number of spikes fired at a time step. However, AugMapping requires additional memory to store the spike coefficients, as it replaces the energy cost with memory consumption. Hence, the fast and inherently low-power conversion method from ANNs to SNNs needs to be explored to improve the conversion efficiency.

In this paper, we propose a brand new symmetric-threshold ReLU activation function for fast and nearly lossless ANN-SNN conversion. The proposed activation function, whose thresholds corresponding to the positive and negative thresholds of converted SNNs, is employed during the training process of ANNs. The contributions of this paper are as follows:

1) A heuristic symmetric-threshold ReLU (stReLU) activation function is presented to perform the conversion from ANNs to SNNs based on the intrinsic difference between the activations of ANNs and the spiking rate of IF neurons in SNNs. Detailed theoretical formulas are provided to demonstrate its efficiency in reducing conversion error.

2) Based on the stReLU, the asymmetric-threshold LeakyReLU (atLeakyReLU) and threshold ReLU (tReLU) are compared to suggest the effectiveness of symmetric thresholds.

3) We evaluate the conversion method on various networks and datasets. The experimental results show that the proposed method can reduce the conversion error within a few time steps and achieve nearly lossless conversion at lower energy costs.

2 Preliminaries

2.1 Activation function in ANNs

ReLU assigns excellent nonlinear properties and is one of the most widely utilized activation functions in ANNs. Assuming the output of layer l is $\boldsymbol{z}^l$, the activations can be formulated by (1).

$$\boldsymbol{z}^l = f(\boldsymbol{W}^l \boldsymbol{z}^{l-1} + \boldsymbol{b}^l) \tag{1}$$

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases} \tag{2}$$

where $\boldsymbol{W}^l$ denotes the connection weights from the neurons in layer $l-1$ to the neurons in layer l, and b^l is the corresonding bias. However, the negative inputs to ReLU have no gradient as α is set to zero, leading neurons to fail to learn. Yu et al.[27] introduced the LeakyReLU[28, 29] activation function instead of the ReLU activation function in ANNs. LeakyReLU is a variant of ReLU that can solve the dying ReLU problem, because LeakyReLU sets the slope to nonzero when the input is negative.

2.2 Neuron model in SNNs

IF[30, 31] neuron models are used in converted SNNs. The membrane potential update of the IF neuron contains two phases. The first phase is membrane potential accumulation, where postsynaptic neurons accumulate spike inputs from presynaptic neurons, as represented in (3).

$$\boldsymbol{V}^l_{temp}(t) = \boldsymbol{V}^l(t-1) + \boldsymbol{W}^l \boldsymbol{o}^{l-1}(t) V^{l-1}_{th} + \boldsymbol{b}^l \tag{3}$$

where $\boldsymbol{V}^l(t)$ and $\boldsymbol{V}^l_{temp}(t)$ represent the membrane potential and instantaneous membrane potential of neurons in layer l at time t, respectively. $\boldsymbol{o}^l(t)$ indicates whether neurons in layer l fire spikes at time t. If $\boldsymbol{V}^l_{temp}(t)$ exceeds the threshold, $\boldsymbol{o}^l(t)$ equals $\mathbf{1}$, otherwise it equals $\mathbf{0}$.

The second stage is spike firing. When the membrane potential $\boldsymbol{V}^l_{temp}(t)$ exceeds the threshold V_{th}, there are two common ways to adjust the membrane potential, including reset by zero and reset by subtraction. We use the latter to subtract the threshold V_{th} from the current membrane potential at the time it exceeds V_{th}. The formula is as follows:

$$\boldsymbol{V}^l(t) = \begin{cases} \boldsymbol{V}^l_{temp}(t) - V^l_{th}, & \text{if } \boldsymbol{V}^l_{temp}(t) \geq V^l_{th} \\ \boldsymbol{V}^l_{temp}(t), & \text{if } \boldsymbol{V}^l_{temp}(t) < V^l_{th}. \end{cases} \tag{4}$$

Equating (3) and (4) yields (5), which represents the membrane potential of postsynaptic neurons at time t.

$$\boldsymbol{V}^l(t) = \boldsymbol{V}^l(t-1) + \boldsymbol{W}^l \boldsymbol{o}^{l-1}(t) V^{l-1}_{th} + \boldsymbol{b}^l - \boldsymbol{o}^l(t) V^l_{th}. \tag{5}$$

3 Method

In this section, we first introduce the proposed stReLU activation function used in ANN training. Then, the conversion method from ANNs to SNNs is described in detail. Next, the conversion error is analysed by formulating the network behavior with the proposed stReLU. Finally, to verify the effectiveness of stReLU, other extended activation functions based on stReLU are introduced.

3.1 Symmetric-threshold ReLU

The main purpose of TerMapping and AugMapping[27] is to retain the negative input of the activation function. However, since the slope for negative input of LeakyReLU in ANNs often takes a value of 0.01 or less, the positive and negative thresholds for spike firing of the converted SNNs are not equal. The asymmetric thresholds will result in a series of issues. The intrinsic reason is that when applying the subtracting mechanism as the reset operation, asymmetric thresholds provide two separate quantization factors, resulting in different updates in membrane potential when positive and negative spikes are firing. This makes the model unstable and even uncontrollable, limiting the effectiveness of the conversion from ANNs to SNNs.

Therefore, we focus on the distinction between the activation function of ANNs and the spike operation of SNNs. The spike mechanism in SNNs is essentially a quantization and clip operation that involves membrane potential accumulation and spike firing. Hence, this paper employs a novel activation function for ANNs in conversion named stReLU. stReLU has symmetric thresholds that correspond to the positive and negative thresholds of the IF neuron model, allowing the converted SNNs to retain negative inputs while avoiding a series of issues caused by asymmetric thresholds.

Forward propagation of stReLU. We define the forward propagation of stReLU as (6) where V_{th} is a custom bound. When the input of the activation function exceeds the bound, it will be clipped to the bound.

$$\boldsymbol{z}^l = \text{stReLU}(\boldsymbol{W}^l \boldsymbol{z}^{l-1} + \boldsymbol{b}^l) \tag{6}$$

$$\text{stReLU}(x) = \begin{cases} V_{th}, & \text{if } x \geq V_{th} \\ x, & \text{if } -V_{th} < x < V_{th} \\ -V_{th}, & \text{if } x \leq -V_{th}. \end{cases} \tag{7}$$

When ignoring bias, further derivation of (7) yields (8).

$$\boldsymbol{z}^l = \text{clip}(\boldsymbol{W}^l \boldsymbol{z}^{l-1}, -V_{th}^l, V_{th}^l) \tag{8}$$

$\text{clip}(x, a, b)$ means setting the upper bound of x to a and the lower bound to b.

Backward propagation of stReLU. The partial derivative δ^l of the loss L with respect to z^l can be formulated by (9).

$$\begin{aligned} \delta^l = \frac{\partial L}{\partial \boldsymbol{z}^l} = \frac{\partial L}{\partial \boldsymbol{z}^{l+1}} \times \frac{\partial \boldsymbol{z}^{l+1}}{\partial \boldsymbol{z}^l} = \\ \delta^{l+1} \times \frac{\partial \text{stReLU}\left(\boldsymbol{z}^l\right)}{\partial \boldsymbol{z}^l} = \\ \begin{cases} \delta^{l+1}, & \text{if } -V_{th} < \boldsymbol{z}^l < V_{th} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{9}$$

3.2 Conversion from ANNs to SNNs

During the conversion process, convolution layers, fully-connected layers, and average pooling layers in converted SNNs are identical to ANNs when converting ANNs to SNNs. The conversion from the activation function to the IF neuron model, as shown in Fig. 2, is the most critical aspect and the main source of conversion error. The relationship between stReLU output and IF spiking rate is deduced, indicating that stReLU can implement lossless conversion.

When neurons fire spikes, we consider a soft-reset mechanism to reduce membrane potential by V_{th}^l after each spike, the membrane potential of neurons in layer l at time T can be given as (10).

$$\boldsymbol{V}^l(T) = \boldsymbol{V}^l(0) + \sum_{t=1}^{T} \boldsymbol{W}^l \boldsymbol{o}^{l-1}(t) V_{th}^{l-1} - \sum_{t=1}^{T} \boldsymbol{o}^l(t) V_{th}^l. \tag{10}$$

The meaning of $\boldsymbol{V}^l(t)$ is the same as Section 2. $\boldsymbol{V}^l(0)$ is the initial membrane potential, and is set to be 0 in the experiment. At each moment, each neuron has three spike firing states including positive spike, negative spike, and no spike fired. Since then, $\sum_{t=1}^{T} \boldsymbol{o}^l(t) V_{th}^l$ can be simplified to $\boldsymbol{N}^l V_{th}^l$, where $\boldsymbol{N}^l$ indicates the number of spikes of neurons in layer l and equals the number of positive spikes minus the number of negative spikes.

Then, we assume the residual membrane potential $\boldsymbol{V}^l(T)$ lies in $[-\frac{V_{th}^l}{2}, \frac{V_{th}^l}{2})$.

$$-\frac{V_{th}^l}{2} \leq \sum_{t=1}^{T} \boldsymbol{W}^l \boldsymbol{o}^{l-1}(t) V_{th}^{l-1} - \boldsymbol{N}^l V_{th}^l < \frac{V_{th}^l}{2} \tag{11}$$

further simplification,

$$-\frac{V_{th}^l}{2} \leq T \boldsymbol{W}^l \bar{\boldsymbol{o}}^{l-1}(t) V_{th}^{l-1} - \boldsymbol{N}^l V_{th}^l < \frac{V_{th}^l}{2} \tag{12}$$

where $\bar{\boldsymbol{o}}^{l-1} = (1/T) \sum_{t=1}^{T} \boldsymbol{o}^{l-1}$. We can further get

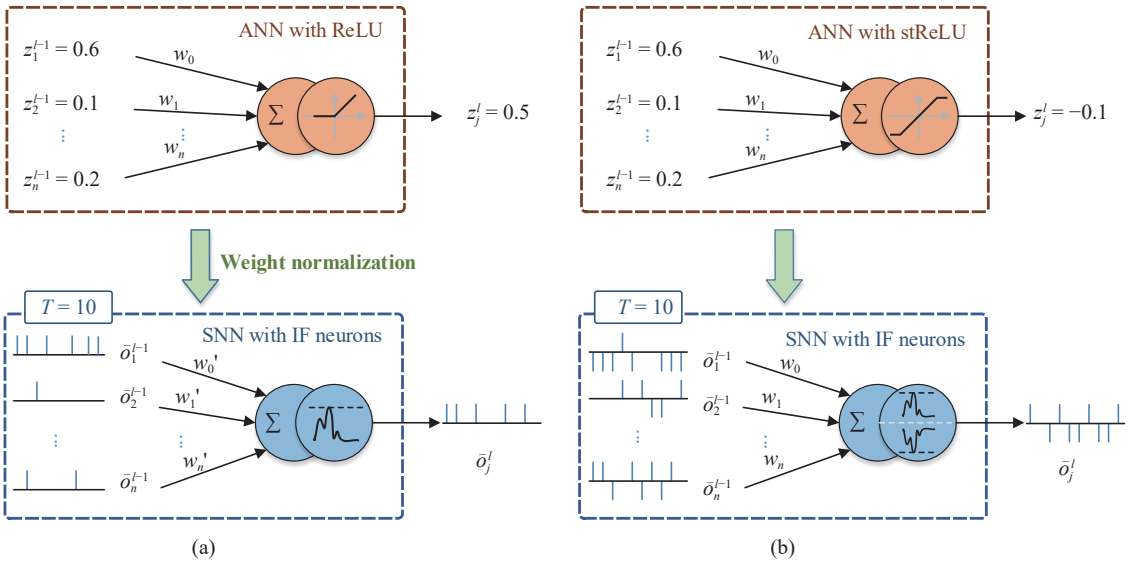

Fig. 2 Conversion process from activations of activation functions to spike firing rates of IF neural models. (a) Standard ANN-SNN conversion. (b) ANN-SNN conversion of ANNs with stReLU. Weight normalization is necessary for ANNs with ReLU conversion but not for ANNs with stReLU, and ANNs with stReLU can convert negative activations to negative spikes. In the case of double thresholds, IF neurons have three states at each time step: 0, 1 and −1, which represent no spike, positive spike, and negative spike, respectively. Positive and negative spikes cancel out when calculating the spike firing rate of IF neural in T time steps. Taking the postsynaptic IF neuron in (b) as an example, $\bar{o}_j^l = (4 + (-5))/10 = -0.1$.

$$\frac{T\boldsymbol{W}^l\bar{\boldsymbol{o}}^{l-1}(t)V_{th}^{l-1}}{V_{th}^l}-\frac{1}{2}<\boldsymbol{N}^l\leq\frac{T\boldsymbol{W}^l\bar{\boldsymbol{o}}^{l-1}(t)V_{th}^{l-1}}{V_{th}^l}+\frac{1}{2} \tag{13}$$

that is

$$\boldsymbol{N}^l=\text{round}\left(\frac{T\boldsymbol{W}^l\bar{\boldsymbol{o}}^{l-1}(t)V_{th}^{l-1}}{V_{th}^l}\right) \tag{14}$$

round$(\cdot)$ means rounding x up to the nearest integer.

$\bar{\boldsymbol{o}}^l$ equals to the spike rate $\boldsymbol{N}^l/T$ of neurons in layer l. Since at most T positive spikes or T negative spikes can be fired in T time steps, we clip $\bar{\boldsymbol{o}}^l$ to $[-1,1]$.

$$\bar{\boldsymbol{o}}^l=\text{clip}\left(\frac{\text{round}(TW^l\bar{\boldsymbol{o}}^{l-1}V_{th}^{l-1}/V_{th}^l)}{T},-1,1\right). \tag{15}$$

3.2.1 Conversion error

In this section, we formulate a mathematical derivation to demonstrate that the stReLU can minimize the squared conversion error.

The conversion error between stReLU and IF is first defined as

$$\boldsymbol{e}^l=\boldsymbol{z}^l-\bar{\boldsymbol{o}}^l. \tag{16}$$

Then, the expectation of the squared conversion error of layer l is $\text{E}_{\boldsymbol{z}^l}[\boldsymbol{z}^l-\bar{\boldsymbol{o}}^l]^2$. By shifting $\boldsymbol{z}^l$ or $\bar{\boldsymbol{o}}^l$, the systematic bias between them can be further optimized[32]. Next, we demonstrate that stReLU can naturally reach the minimal $\text{E}_{\boldsymbol{z}^l}[\boldsymbol{z}^l-\bar{\boldsymbol{o}}^l]^2$ without shifting.

We fix $\bar{\boldsymbol{o}}^l$ and shift $\boldsymbol{z}^l$ by $\Delta\boldsymbol{z}$. Taking (15) and (8) into (16) and assuming $V_{th}=1$, when the input of layer l in ANN is equal to SNN, (17) can be obtained.

$$\begin{aligned}&\text{E}_{\boldsymbol{z}^l}[(\boldsymbol{z}^l+\Delta\boldsymbol{z})-\bar{\boldsymbol{o}}^l]^2=\\&\text{E}_{\boldsymbol{z}^l}\left[(\boldsymbol{z}^l+\Delta\boldsymbol{z})-\frac{\text{round}(T\boldsymbol{z}^l)}{T}\right]^2.\end{aligned} \tag{17}$$

Then, we assume that $\boldsymbol{z}^l$ is uniformly distributed within each small interval $[I_t,I_{t+1}]$ with the probability density function p_t $(t=0,1,\cdots,2T-1)$, where $I_t=(t-1)/T$[23]. Let

$$g(\boldsymbol{z}^l)=\boldsymbol{z}^l-\text{round}(T\boldsymbol{z}^l)/T. \tag{18}$$

Substituting (18) into (17), $\text{E}_{\boldsymbol{z}^l}[(\boldsymbol{z}^l+\Delta\boldsymbol{z})-\bar{\boldsymbol{o}}^l]^2$ can be simplified to

$$\begin{aligned}&\text{E}_{\boldsymbol{z}^l}[(\boldsymbol{z}^l+\Delta\boldsymbol{z})-\bar{\boldsymbol{o}}^l]^2=\\&\text{E}_{\boldsymbol{z}^l}[\Delta\boldsymbol{z}+g(\boldsymbol{z}^l)]^2=\\&(\Delta\boldsymbol{z}+\text{E}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)])^2+\text{D}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]\end{aligned} \tag{19}$$

where $\text{D}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]$ represents the variance of $g(\boldsymbol{z}^l)$. When $\Delta\boldsymbol{z}=-\text{E}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]$, $\text{E}_{\boldsymbol{z}^l}[(\boldsymbol{z}^l+\Delta\boldsymbol{z})-\bar{\boldsymbol{o}}^l]^2$ is minimal. Then, we give the following derivations to demonstrate $\text{E}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]=0$.

$$\begin{aligned}\text{E}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]=\text{E}_{\boldsymbol{z}^l}[\boldsymbol{z}^l-\frac{\text{round}(T\boldsymbol{z}^l)}{T}]=\\\int_{\frac{(t-T)}{T}}^{\frac{(2t-2T+1)}{2T}}p_t(\boldsymbol{z}^l-\frac{(t-T)}{T})\text{d}\boldsymbol{z}^l+\\\int_{\frac{(2t-2T+1)}{2T}}^{\frac{(t-T+1)}{T}}p_t(\frac{(t-T+1)}{T}-\boldsymbol{z}^l)\text{d}\boldsymbol{z}^l=\\\Delta e^-+\Delta e^+=0.\end{aligned} \tag{20}$$

As shown in Fig. 3, Δe^+ represents the error when the spike firing rate of the IF neuron is larger than the activation of stReLU, while Δe^- represents the error when the spike firing rate is less than the activation. Δe^+ can be offset against nearby Δe^- when calculating $\text{E}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]$. Therefore, when $\Delta\boldsymbol{z}=0$, the expected squared conversion error $\text{E}_{\boldsymbol{z}^l}[(\boldsymbol{z}^l+\Delta\boldsymbol{z})-\bar{\boldsymbol{o}}^l]^2$ is minimal.

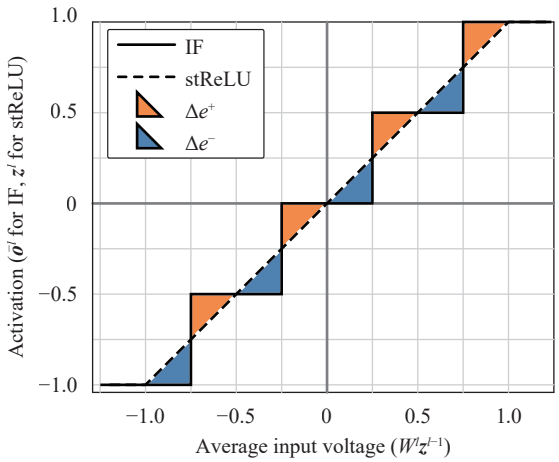

Fig. 3 Conversion error between stReLU in ANNs and IF in SNNs ($V_{th}=1, T=2$). Δe^+ can be offset against nearby Δe^- when calculating the expectation of conversion error $\text{E}_{\boldsymbol{z}^l}[g(\boldsymbol{z}^l)]$.

Through the above derivation, we demonstrate that stReLU naturally guarantees the minimal $\text{E}_{\boldsymbol{z}^l}[\boldsymbol{z}^l-\bar{\boldsymbol{o}}^l]^2$ without an additional shift operation. Furthermore, according to Lemma 1 in Li et al.[20], the expected squared conversion error of the last layer is minimal. Therefore, the application of stReLU can achieve lossless conversion from ANNs to SNNs.

3.3 Other threshold activation functions

The thresholds are added to the LeakyReLU and ReLU activation functions to highlight the significance of symmetric thresholds and negative activations, respectively.

When the slope for the negative input of LeakyReLU is set to α, its threshold equals $-1/\alpha$[27]. As a result, the proposed atLeakyReLU is obtained by setting the negative threshold of LeakyReLU to $-1/\alpha$ and the positive

threshold to 1, which can be formulated by (21).

$$\text{atLeakyReLU}(x)=\begin{cases}1, & \text{if } x\geq 1\\ x, & \text{if } 0\leq x<1\\ \alpha x, & \text{if } -\dfrac{1}{\alpha^2}<x<0\\ -\dfrac{1}{\alpha}, & \text{if } x\leq -\dfrac{1}{\alpha^2}.\end{cases}\tag{21}$$

Threshold ReLU (tReLU)[32] has only a positive threshold, and the activation value is clipped to the threshold when it exceeds the threshold. tReLU is defined as (22).

$$\text{tReLU}(x)=\begin{cases}1, & \text{if } x\geq 1\\ x, & \text{if } 0<x<1\\ 0, & \text{if } x\leq 0.\end{cases}\tag{22}$$

The three threshold activation functions are shown in Fig. 4.

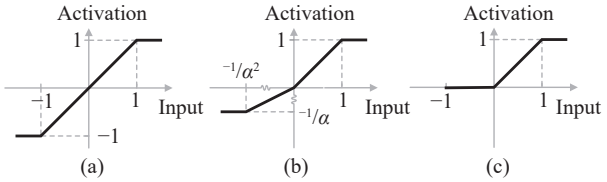

Fig. 4 Comparison of threshold activation functions. (a) Symmetric-threshold ReLU; (b) Asymmetrical-threshold Leaky-ReLU; (c) Threshold ReLU.

4 Experiments

4.1 Experimental setup

We use the mainstream framework PyTorch[33] to implement the whole conversion method. To verify the efficiency of the conversion method for reducing the time steps, we conduct experiments on MNIST[1], Fashion-MNIST[34] and CIFAR10[35] with six different architectures (Arc.) in Table 1, including fully connected networks, convolutional networks and more complex VGG networks. Single digits indicate the numbers of neurons in the fully connected layer, and c and p represent the convolutional layer and the pooling layer, respectively. For example, 32c5 indicates that the convolutional layer has 32 feature maps, and the kernel size of each feature map is 5×5. p2 means that the kernel size of the pooling layer is 2×2.

Dataset. MNIST is a handwritten digital dataset labelled from 0 to 9, each of which is a 28×28 grayscale image, with 60 000 training samples and 10 000 test samples. FashionMNIST is a dataset of clothes with the same sample numbers and image size as MNIST. Each image in CIFAR10 is a 32×32 RGB color image, and it contains 50 000 training samples and 10 000 test samples containing ten categories. MNIST and Fashion-MNIST are not preprocessed. On CIFAR10, we employ autoaugment[37] and cutout[38] for data augmentation. We fill four zero-pixel padding around the original 32×32 pixel image, then crop it to its original size at random, flip it horizontally with a probability of 0.5, followed by normalization.

Table 1 Experimental network configurations[27]

Dataset	Network Arc.	Topology
MNIST	Net1	1200-1200-10
	Net2	12c5-p2-64c5-p2-10
Fashion-MNIST	Net3	6400-10
	Net4	32c5-p2-64c5-p2-1024-10
CIFAR10	Net5	128c3-128c3-p2-256c3-256c3 p2-512c3-512c3-p2-1024-10
	Net6	VGG16[36]

Training. In all networks in Table 1, we utilize the stReLU activation function with bound of ±1. Among them, MNIST is trained using the simple fully connected network Net1 and the convolutional network Net2, Fashion-MNIST is trained using Net3 and Net4, and CIFAR10 is trained using the more complicated VGG networks Net5 and Net6. During training, we use the Adam optimizer with a dynamic learning rate for 300–400 epochs until the networks converge. We add a batch normalization layer in front of each activation function layer in the feature extraction section to improve the training accuracy for VGG networks.

Inference. IF neurons are employed in SNNs during inference, and the positive and negative thresholds are the same as in ANNs. To reduce the conversion error, we do not use the spiking mechanism in the input layer and output layer. The image pixel values are directly fed into the network without encoding the images in the first layer, which avoids information loss during the encoding process. For the last layer, the spiking neurons integrate the spiking input of presynaptic neurons, and then fire multiple spikes instead of a binary spike.

4.2 Results

4.2.1 Accuracy comparison with other methods

In this section, we compare our method with existing methods using VGG16 architecture including data-based normalization (DataNorm)[15], spike-based normalization (SpikeNorm)[18], CQ[21], residual membrane potential SNN (RMP-SNN)[39] and rate norm layer (RNL)[22], through the classification task on CIFAR10, and the result of DataNorm is from Yu et al.[27] By comparing the time steps required for the SNNs to converge and the conversion error relative to ANN accuracy (Acc.) during the ANN-SNN conversion, the performance of different meth-

Table 2 Comparison of the conversion error on CIFAR-10 dataset

Method	ANN Acc.	Error	Latency
DataNorm[15]	91.03%	5.08%	–
SpikeNorm[18]	91.70%	0.16%	2 500
CQ[21]	91.77%	0.00%	1 000
RMP-SNN[39]	93.63%	0.00%	1 536
RNL[22]	92.86%	–0.04%	–
This work	93.71%	0.00%	**147**

ods is evaluated. As shown in Table 2, ANNs with stReLU can achieve lossless conversion within 147 time steps.

In addition, we compare our algorithm with TerMapping, which also considers the importance of negative activations. Table 3 lists the ANN accuracy, SNN accuracy, latency, and final conversion error obtained by different conversion methods with the same network architectures on the same datasets. All six networks achieve lossless conversion in a few time steps, as shown in Table 3. The simple fully connected architectures Net1 and Net3 require only a few time steps to achieve the accuracy of ANNs, and SNNs even outperform ANNs. For convolution architectures Net2 and Net4, fewer than 100 time steps are needed to achieve lossless conversion, which is a significant improvement over TerMapping. For more complex tasks on CIFAR10, Net5 and Net6 with stReLU achieve lossless conversion within acceptable time steps, and even higher accuracy than ANNs.

Table 3 Comparison of our algorithm with TerMapping

Network Arc.	ANN Acc.	SNN Acc.	Latency	Error
MNIST				
Net1 (TerMapping)	98.77%	98.77%	750	0.00%
Net1 (Ours)	98.29%	98.29%	**3**	0.00%
Net2 (TerMapping)	99.35%	99.35%	750	0.00%
Net2 (Ours)	99.29%	99.29%	**11**	0.00%
FashionMNIST				
Net3 (TerMapping)	90.18%	90.18%	1 500	0.00%
Net3 (Ours)	89.90%	89.92%	**11**	**–0.02%**
Net4 (TerMapping)	92.11%	92.11%	2 900	0.00%
Net4 (Ours)	90.83%	90.83%	**79**	0.00%
CIFAR10				
Net5 (TerMapping)	94.13%	93.75%	2 800	0.40%
Net5 (Ours)	92.67%	92.69%	**79**	**–0.02%**
Net6 (TerMapping)	93.42%	92.30%	4 400	1.20%
Net6 (Ours)	93.71%	93.71%	**147**	**0.00%**

We further visualize the images of the conversion error of Net1-Net4 as a function of time steps in Fig. 5, and the accuracy of Net5-Net6 as a function of time steps in Fig. 6. The conversion error of TerMapping begins to exhibit a substantial downwards trend after approximately 8 time steps for Net1-Net3, whereas stReLU has a low error at the first time step. For Net4, stReLU performs worse than Net1-Net3 at the first time step, but still surpasses the TerMapping results. For more challenging CIFAR10 with deeper networks Net6, the stReLU has high accuracy from the beginning, whereas the TerMapping accuracy starts to improve after 128 time steps.

By comparison with other methods, the results show that the converted SNNs converge faster with our algorithm, which achieves higher accuracy within fewer time steps, with 6× to 250× speedup. The experimental results show that our algorithm has certain advantages in the existing ANN-SNN conversion algorithms. The application of stReLU restricts the activations of ANN to [–1, 1], thereby reducing the conversion loss caused by over activation, making the expectation of squared conversion error in each layer reach minimal, and achieving lossless conversion within limited time steps.

Weight normalization is a common technique for reducing conversion error in existing conversion algorithms, but it is typically challenging to determine the scale factor of normalization, and weight normalization requires statistics on the weights of all network layers for the entire dataset to calculate the scale factor, which will significantly increase the computational cost. In contrast, stReLU can modify the weights without normalization, balance over activation and under activation, and therefore reduce conversion error caused by residual membrane potential, as well as decrease the inference delay.

4.2.2 Ablation study

Inspired by the negative activations used in [27], which has been experimentally demonstrated to be efficient for improving the performance of the conversion method, stReLU is introduced to retain negative activations in this paper. We count the numbers of positive and negative activations in different layers of Net6 based on 1 024 samples. As shown in Fig. 7, the relative distribution of zero activations and positive activations for tReLU is similar to the relative distribution of positive and negative activations for stReLU. This indicates that the proposal of the stReLU activation function has little effect on the relative distribution of activations. Instead, it is the activation function itself that contributes most to reducing conversion error.

Afterwards, we evaluate the accuracy of the converted SNNs and the time steps required for conversion with these three activation functions to analyse how different threshold activation functions actually impact ANN-SNN conversion.

As illustrated in Table 4, compared with the activation function with asymmetric thresholds, the symmetric thresholds can minimize the expectation of the squared

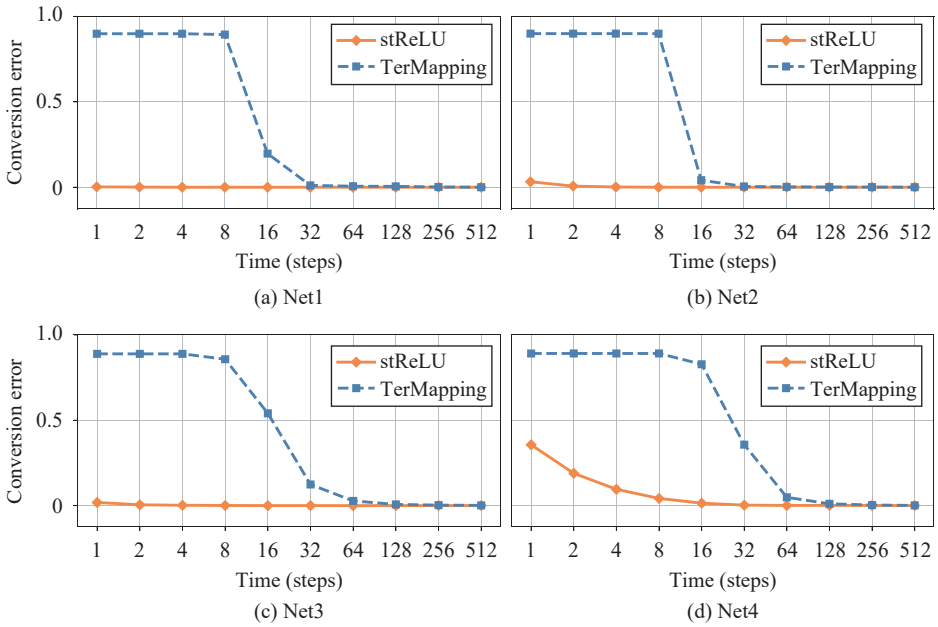

Fig. 5 Inference error versus the number of time steps of Net1-Net4. SNNs in our algorithm converge faster than TerMapping.

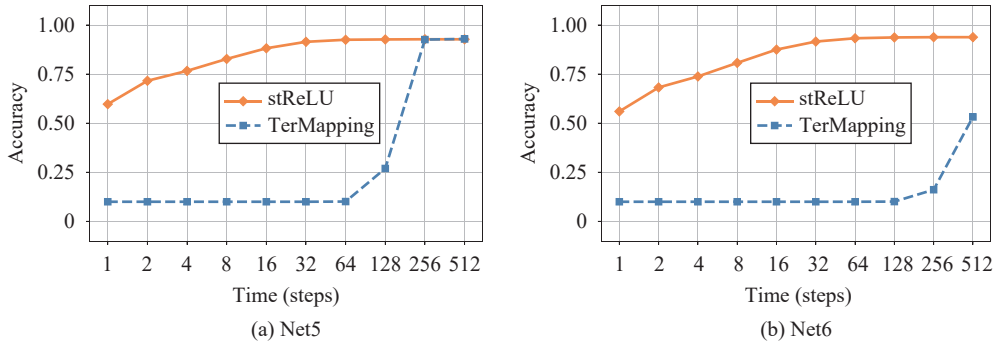

Fig. 6 Inference accuracy versus the number of time steps of Net5-Net6. SNNs in our algorithm converge faster than TerMapping.

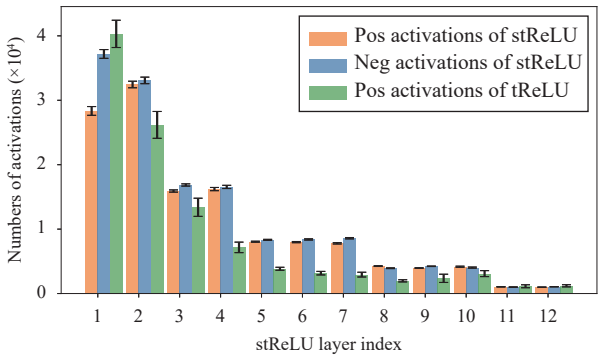

Fig. 7 Comparison of the numbers of positive and negative activations between stReLU and tReLU in each convolutional layer of Net6.

Table 4 Comparison of different activation functions

Network Arc.	AcFunction	ANN Acc.	SNN Acc.	Latency	Error
Net5	stReLU	92.67%	92.69%	**79**	**−0.02%**
	atLeakyReLU	92.78%	92.79%	373	−0.01%
	tReLU	93.03%	**93.05%**	756	**−0.02%**
Net6	stReLU	93.71%	**93.71%**	**147**	**0.00%**
	atLeakyReLU	93.28%	93.15%	808	0.14%
	tReLU	93.31%	93.31%	603	**0.00%**

conversion error in each layer, thus reducing the inference delay. Therefore, while the network structure becomes deeper, the ANN with stReLU can still achieve faster lossless conversion. Since the asymmetric thresholds will result in an asymmetric update of the membrane potential, the performance of atLeakyReLU is unpredictable in different models and may even degrade when the network structure is much deeper. It can be observed from Fig. 8 that the stReLU enables the converted SNN to obtain higher accuracy with fewer inference time steps than the other two threshold activation functions. In conclusion, the experimental results are consistent with our theoretical formulations.

4.2.3 Energy efficiency

According to previous studies[40–42], combining SNNs

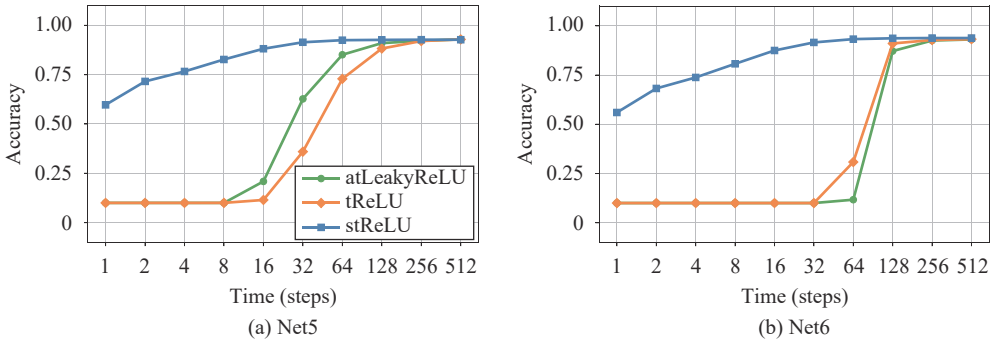

Fig. 8 Conversion accuracy of different activation functions versus the number of time steps. The conversion error of stReLU converges faster than atLeakyReLU and tReLU.

with event-driven neuromorphic platforms can further exploit the benefits of SNNs in terms of low power consumption. Only neurons that fire spikes are considered during inference, drastically reducing the amount of computation. Moreover, in ANNs, an operation involves a floating-point multiplication and a floating-point addition operation, known as the multiply accumulate (MAC) operation, whereas in SNNs, it refers to a floating-point addition operation. The power consumption of addition operations in hardware is substantially lower than that of multiplication operations, hence deploying SNNs on neuromorphic computing systems will greatly reduce power consumption. As a result, we exclusively compute the power consumption of ANNs and SNNs on Net1–Net6 (Table 5).

In Table 5, OP_{ANN} denotes the number of MAC operations in ANNs[43]. OP_{SNN} is the number of synaptic operations (SOP)[44, 45] within T time steps, when the conversion error is less than 1%. The energy cost for 32-bit ANN MAC operation is $4.6pJ$[46] while the energy cost for SOP is $77fJ$[47]. $Energy_{ANN}$ and $Energy_{SNN}$ represent the energy cost in ANNs and SNNs, respectively, which can be formulated by (23) and (24).

$$Energy_{ANN} = 4.6 \times OP_{ANN} \tag{23}$$

$$Energy_{SNN} = 0.077 \times OP_{SNN}. \tag{24}$$

Table 5 shows that, SNNs with stReLU only cost 43.43% of the energy of ANNs in tasks on CIFAR10, and nearly 99% of the energy can be saved in shallow networks. Compared with most ANNs, SNNs with stReLU can decrease energy usage, even in deep neural networks.

5 Conclusions

Based on the intrinsically different responses of the activation functions in ANNs and IF neuron models in SNNs, this paper proposes a novel symmetric-threshold ReLU activation function for ANN-SNN conversion, which not only retains the negative activations but also achieves lossless conversion within relatively few time steps. The effect of stReLU on the conversion error is demonstrated by detailed theoretical formulations. Based on the above analysis, the asymmetric-threshold LeakyReLU and threshold ReLU are explored to improve the training performance of ANNs. Furthermore, we conduct experiments on various network structures and datasets, which shows that our algorithm can significantly reduce the conversion error, accelerate the convergence of converted SNNs, and achieve higher accuracy within fewer time steps compared with other algorithms that use ReLU or LeakyReLU. The efficiency of SNNs in reducing energy consumption is further indicated by calculating the energy consumption of ANNs and SNNs. We will continue to focus on finding activation functions that are more suitable for conversion, and alleviate the restrictions on ANNs while achieving lossless conversion from ANNs to SNNs.

Table 5 Comparison between ANNs and SNNs in terms of operation number and energy consumption

Net Arc.	OP_{ANN} ($\times 10^6$)	OP_{SNN} ($\times 10^3$)	$Energy_{ANN}$ ($\times 10^6 pJ$)	$Energy_{SNN}$ ($\times 10^5 pJ$)	$\frac{Energy_{SNN}}{Energy_{ANN}}$
Net1	2.39	1.14	11.01	0.88	0.80%
Net2	1.41	1.02	6.49	0.78	1.20%
Net3	5.08	2.91	23.38	2.24	0.96%
Net4	4.80	23.51	22.06	18.10	8.20%
Net5	610.08	14 807.00	2 806.39	11 401.39	40.63%
Net6	313.48	8 132.60	1 442.00	6 262.10	43.43%

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2020AAA0105900), National Natural Science Foundation of China (No. 62236007), and Zhejiang Lab, China (No. 2021KC0AC01).

References

[1] Y. LeCun, Y. Bengio, G. Hinton. Deep learning. *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.

[2] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[3] W. Zaremba, I. Sutskever, O. Vinyals. Recurrent neural network regularization. [Online], Available: https://arxiv.org/abs/1409.2329, 2014.

[4] Y. J. Zhang, Z. F. Yu, J. K. Liu, T. J. Huang. Neural decoding of visual information across different neural recording modalities and approaches. *Machine Intelligence Research*, vol. 19, no. 5, pp. 350–365, 2022. DOI: 10.1007/s11633-022-1335-2.

[5] Y. Wu, D. H. Wang, X. T. Lu, F. Yang, M. Yao, W. S. Dong, J. B. Shi, G. Q. Li. Efficient visual recognition: A survey on recent advances and brain-inspired methodologies. *Machine Intelligence Research*, vol. 19, no. 5, pp. 366–411, 2022. DOI: 10.1007/s11633-022-1340-5.

[6] R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, USA, pp. 580–587, 2014. DOI: 10.1109/CVPR.2014.81.

[7] W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997. DOI: 10.1016/S0893-6080(97)00011-7.

[8] Q. Xu, J. R. Shen, X. M. Ran, H. J. Tang, G. Pan, J. K. Liu. Robust transcoding sensory information with neural spikes. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 5, pp. 1935–1946, 2022. DOI: 10.1109/TNNLS.2021.3107449.

[9] K. Roy, A. Jaiswal, P. Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, vol. 575, no. 7784, pp. 607–617, 2019. DOI: 10.1038/s41586-019-1677-2.

[10] J. Pei, L. Deng, S. Song, M. G. Zhao, Y. H. Zhang, S. Wu, G. R. Wang, Z. Zou, Z. Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. J. Wu, Z. Y. Yang, C. Ma, G. Q. Li, W. T. Han, H. L. Li, H. Q. Wu, R. Zhao, Y. Xie, L. P. Shi. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, vol. 572, no. 7767, pp. 106–111, 2019. DOI: 10.1038/s41586-019-1424-8.

[11] P. U. Diehl, M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, vol. 9, Article number 99, 2015. DOI: 10.3389/fncom.2015.00099.

[12] P. J. Gu, R. Xiao, G. Pan, H. J. Tang. STCA: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, pp. 1366–1372, 2019.

[13] Y. J. Wu, L. Deng, G. Q. Li, J. Zhu, L. P. Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, vol. 12, Article number 331, 2018. DOI: 10.3389/fnins.2018.00331.

[14] Y. Q. Cao, Y. Chen, D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015. DOI: 10.1007/s11263-014-0788-3.

[15] 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*, IEEE, Killarney, Ireland, pp. 1–8, 2015. DOI: 10.1109/IJCNN.2015.7280696.

[16] Z. M. Wang, S. Lian, Y. H. Zhang, X. X. Cui, R. Yan, H. J. Tang. Towards lossless ANN-SNN conversion under ultra-low latency with dual-phase optimization. [Online], Available: https://arxiv.org/abs/2205.07473, 2022.

[17] B. Rueckauer, I. A. Lungu, Y. H. Hu, M. Pfeiffer, S. C. Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, vol. 11, Article number 682, 2017. DOI: 10.3389/fnins.2017.00682.

[18] A. Sengupta, Y. T. Ye, R. Wang, C. Liu, K. Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, vol. 13, Article number 95, 2019. DOI: 10.3389/fnins.2019.00095.

[19] S. Kim, S. Park, B. Na, S. Yoon. Spiking-YOLO: Spiking neural network for energy-efficient object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 11270–11277, 2020. DOI: 10.1609/aaai.v34i07.6787.

[20] Y. H. Li, S. K. Deng, X. Dong, R. H. Gong, S. Gu. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6316–6325, 2021.

[21] Z. L. Yan, J. Zhou, W. F. Wong. Near lossless transfer learning for spiking neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10577–10584, 2021. DOI: 10.1609/aaai.v35i12.17265.

[22] J. H. Ding, Z. F. Yu, Y. H. Tian, T. J. Huang. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 2328–2336, 2021.

[23] T. Bu, W. Fang, J. H. Ding, P. L. Dai, Z. F. Yu, T. J. Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[24] B. Rueckauer, S. C. Liu. Conversion of analog to spiking neural networks using sparse temporal coding. In *Proceedings of IEEE International Symposium on Circuits and Systems*, Florence, Italy, 2018. DOI: 10.1109/ISCAS.2018.8351295.

[25] Y. Li, D. C. Zhao, Y. Zeng. BSNN: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons. *Frontiers in Neuroscience*, vol. 16, Article number 991851, 2022. DOI: 10.3389/fnins.2022.991851.

[26] Y. Li, Y. Zeng. Efficient and accurate conversion of spiking neural network with burst spikes. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, Vienna, Austria, pp. 2485–2491, 2022.

[27] Q. Yu, C. X. Ma, S. M. Song, G. Y. Zhang, J. W. Dang, K. C. Tan. Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1714–1726, 2022. DOI: 10.1109/TNNLS.2020.3043415.

[28] B. Xu, N. Y. Wang, T. Q. Chen, M. Li. Empirical evaluation of rectified activations in convolutional network. [Online], Available: https://arxiv.org/abs/1505.00853, 2015.

[29] A. L. Maas, A. Y. Hannun, A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, USA, vol. 30, Article number 3, 2013.

[30] Y. H. Liu, X. J. Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of Computational Neuroscience*, vol. 10, no. 1, pp. 25–45, 2001. DOI: 10.1023/A:1008916026143.

[31] M. Barbi, S. Chillemi, A. Di Garbo, L. Reale. Stochastic resonance in a sinusoidally forced LIF model with noisy threshold. *Biosystems*, vol. 71, no. 1–2, pp. 23–28, 2003. DOI: 10.1016/S0303-2647(03)00106-0.

[32] S. K. Deng, S. Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. M. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. J. Bai, S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information processing Systems*, Vancouver, Canada, vol. 32, Article number 721, 2019.

[34] H. Xiao, K. Rasul, R. Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. [Online], Available: https://arxiv.org/abs/1708.07747, 2017.

[35] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. University of Toronto, Canada, Technical Report TR-2009, 2009.

[36] K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition. [Online], Available: https://arxiv.org/abs/1409.1556, 2014.

[37] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, Q. V. Le. AutoAugment: Learning augmentation strategies from data. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Long Beach, USA, pp. 113–123, 2019. DOI: 10.1109/CVPR.2019.00020.

[38] T. DeVries, G. W. Taylor. Improved regularization of convolutional neural networks with cutout. [Online], Available: https://arxiv.org/abs/1708.04552, 2017.

[39] B. Han, G. Srinivasan, K. Roy. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Seattle, USA, pp. 13555–13564, 2020. DOI: 10.1109/CVPR42600.2020.01357.

[40] J. R. Shen, Y. Zhao, J. K. Liu, Y. M. Wang. HybridSNN: Combining bio-machine strengths by boosting adaptive spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, to be published. DOI: 10.1109/TNNLS.2021.3131356.

[41] D. Roy, I. Chakraborty, K. Roy. Scaling deep spiking neural networks with binary stochastic activations. In *Proceedings of IEEE International Conference on Cognitive Computing*, Milan, Italy, pp. 50–58, 2019. DOI: 10.1109/ICCC.2019.00020.

[42] L. Deng, Y. J. Wu, X. Hu, L. Liang, Y. F. Ding, G. Q. Li, G. S. Zhao, P. Li, Y. Xie. Rethinking the performance comparison between SNNS and ANNS. *Neural Networks*, vol. 121, pp. 294–307, 2020. DOI: 10.1016/j.neunet.2019.09.005.

[43] N. Rathi, K. Roy. DIET-SNN: Direct Input encoding with leakage and threshold optimization in deep spiking neural networks. [Online], Available: https://arxiv.org/abs/2008.03658, 2020.

[44] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, D. S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, vol. 345, no. 6197, pp. 668–673, 2014. DOI: 10.1126/science.1254642.

[45] J. B. Wu, E. Yılmaz, M. L. Zhang, H. Z. Li, K. C. Tan. Deep spiking neural networks for large vocabulary automatic speech recognition. *Frontiers in Neuroscience*, vol. 14, Article number 199, 2020. DOI: 10.3389/fnins.2020.00199.

[46] M. Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *Proceedings of IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, USA, pp. 10–14, 2014. DOI: 10.1109/ISSCC.2014.6757323.

[47] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, G. Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers in Neuroscience*, vol. 9, Article number 141, 2015. DOI: 10.3389/fnins.2015.00141.

Jianing Han received the B. Eng. degree in Internet of things engineering from Taiyuan University of Technology, China in 2021. Currently, she is a master student in computer technology at College of Computer Science and Technology, Zhejiang University, China.

Her research interests include learning algorithms in deep spiking neural networks and neural coding.

E-mail: jnhan@zju.edu.cn

ORCID iD: 0000-0002-5097-7894

Ziming Wang received the B.Sc. degree in computer science from Sichuan University, China in 2020. Currently, he is a Ph.D. degree candidate in computer science and technology at Department of Computer Science, Zhejiang University, China.

His research interests include neuromorphic computing, machine learning, and model compression.

E-mail: zi_ming_wang@zju.edu.cn

Jiangrong Shen received the Ph.D. degree in computer science and technology from College of Computer Science and Technology, Zhejiang University, China in 2022. She is currently a postdoctoral fellow at Zhejiang University, China. She studied as the honorary visiting scholar with University of Leicester, UK in 2019.

Her research interests include neuromorphic computing, cyborg intelligence and neural computation.

E-mail: jrshen@zju.edu.cn (Corresponding author)
ORCID iD: 0000-0003-3683-3779

Huajin Tang received the Ph.D. degree in computing intelligence from National University of Singapore, Singapore in 2005. He is currently a professor with Zhejiang University, China.

He received the 2016 IEEE Outstanding *Transactions on Neural Networks and Learning Systems* (TNNLS) Paper Award. He has served as an Associate Editor for the *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cognitive and Developmental Systems*, and *Frontiers in Neuromorphic Engineering*. His research work on brain GPS has been reported by MIT Technology Review in 2015.

His research interests include neuromorphic computing, neuromorphic hardware and cognitive systems, and robotic cognition.

E-mail: htang@zju.edu.cn