SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2020, Vol. 63 122402:1–122402:19
https://doi.org/10.1007/s11432-019-1468-0

Deterministic conversion rule for CNNs to efficient spiking convolutional neural networks

Xu YANG[1,2], Zhongxing ZHANG[1,2], Wenping ZHU[1,2], Shuangming YU[1,2], Liyuan LIU[1,2] & Nanjian WU[1,2,3*]

[1]*State Key Laboratory of Superlattices and Microstructures, Institute of Semiconductors, Chinese Academy of Sciences, Beijing* 100083, *China;*
[2]*Center of Materials Science and Optoelectronics Engineering, University of Chinese Academy of Sciences, Beijing* 100049, *China;*
[3]*Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Beijing* 100083, *China*

Received 28 February 2019/Revised 7 May 2019/Accepted 8 July 2019/Published online 15 January 2020

Abstract This paper proposes a general conversion theory to reveal the relations between convolutional neural network (CNN) and spiking convolutional neural network (spiking CNN) from structure to information processing. Based on the conversion theory and the statistical features of the activations distribution in CNN, we establish a deterministic conversion rule to convert CNNs into spiking CNNs with definite conversion procedure and the optimal setting of all parameters. Included in conversion rule, we propose a novel "n-scaling" weight mapping method to realize high-accuracy, low-latency and power efficient object classification on hardware. For the first time, the minimum dynamic range of spiking neuron's membrane potential is studied to help to balance the trade-off between representation range and precise of the data type adopted by dedicated hardware when spiking CNNs run on it. The simulation results demonstrate that the converted spiking CNNs perform well on MNIST, SVHN and CIFAR-10 datasets. The accuracy loss over three datasets is no more than 0.4%. 39% of processing time is shortened at best, and less power consumption is benefited from lower latency achieved by our conversion rule. Furthermore, the results of noise robustness experiments indicate that spiking CNN inherits the robustness from its corresponding CNN.

Keywords convolutional neural networks (CNN), spiking neural networks (SNN), image classification, conversion rule, noise robustness, neuromorphic hardware

Citation Yang X, Zhang Z X, Zhu W P, et al. Deterministic conversion rule for CNNs to efficient spiking convolutional neural networks. Sci China Inf Sci, 2020, 63(2): 122402, https://doi.org/10.1007/s11432-019-1468-0

1 Introduction

Object classification is one of the most important branches in computer vision and has been widely applied in many fields, such as autonomous vehicles, drones, security and remote monitoring. Recently, deep neural networks continuously push the state-of-the-art classification accuracy to incredible levels. However, the inferences of large networks always require high-performance data center and large energy cost. As a candidate for solving this problem, spiking neural networks (SNNs) attract many researches because they can achieve ultra-low power classification on neuromorphic hardware. But, how to improve the classification accuracy of SNNs is a challenging issue. In this paper, we propose a deterministic conversion rule to convert convolutional neural networks (CNNs) into efficient spiking convolutional

* Corresponding author (email: nanjian@red.semi.ac.cn)

neural networks (spiking CNNs) for realizing high-performance and power efficient object classification on hardware.

Recently, many distinctive improvements have been achieved by deep learning based CNNs with complex and deeper network architectures [1–4], which proves that CNNs and its training rule are mature enough. So many researches [5–9] focus on accelerating the inference phase of CNN with dedicated hardware to better use it in real life. But some researchers [10, 11] concerned that the power consumption of CNN accelerators is still relatively high. Its power is mainly consumed in data transfer and numerous multipliers. To reduce the consumption of data transfer, specified processing elements with local memory and novel dataflow processing method were adopted by many CNN accelerators [7, 8, 12]. However, these efforts do nothing helpful to the power consumption of numerous multipliers and dense multiplication. To avoid multiplication, binary/ternary neural networks were proposed. Some hardwares dedicated for them show ultra-low power but suffer from unacceptable accuracy loss [10, 13].

SNN is a biological realistic neural network, which mimic the spike-based communication and computation mechanism of the human brain. Brain-like neurons used in SNNs avoid multiplication so that the power consumption caused by numerous multipliers is removed totally, and many neuromorphic platforms show several orders of magnitude improvements in the energy efficiency than some CNN accelerators [14–18]. For example, the SNN-based TrueNorth chip only consumes 63 mW for object classification [14]. The event-driven architecture adopted by TrueNorth focuses their computational effort only on currently active parts, so it naturally fits for realizing SNNs efficiently because of their sparse spike [19, 20]. But, SNNs are lack of mature training rule. Though the supervised [21, 22] or unsupervised [23, 24] spike-timing-dependent plasticity (STDP), Tempotron [25] or other brain-like learning rules and their optimizations [26, 27] were proposed to train SNNs, the classification accuracy achieved by SNNs are still lower than that of CNNs.

Recently, a number of studies have shown that spiking CNN, which is a kind of SNNs, can be converted from CNN and have potential to overcome the disadvantages mentioned above [28–31]. Cao et al. [28] first proposed the method for converting a deep CNN into spiking CNN. They trained CNN to realize high accuracy first, then all well-trained weights of CNN map to that of spiking CNN directly so that the learning issue for deep SNNs is solved. Spiking CNN combines the high accuracy of CNN and spike-based computing paradigm of SNN, so it has potential to realize high classification accuracy in ultra-low power neuromorphic hardware. Though their method is verified functional in two standard datasets, the setting of firing threshold and minimum membrane potential of spiking neuron still rely on hand-tuning and numerous simulations. Besides, there exist a relatively large classification accuracy loss, about 2%, when mapping CNN into spiking CNN.

In order to avoid the hand-tuning, Diehl et al. [29] proposed the weight normalization method. This method normalizes the weights of CNN before mapping into that of spiking CNN, which avoids either too high or too low firing of spiking neuron. Nearly lossless conversion is achieved, but severe classification latency is caused due to over pessimistic scaling factors they choose. For further narrowing the accuracy gap, Rueckauer et al. [31] proposed a modified spiking neuron with "reset by subtraction". And they achieved the state-of-the-art classification accuracy with the use of many tools, like converting biases, analog input to first layer, spiking softmax and spiking max-pooling. But hardware efficiency becomes worse because external spike input, multipliers, exponential function unit and finite impulse response filter need to be added to realize those tools in hardware, respectively. In addition, there is no research about how to determine the minimum dynamic range of spiking neuron's membrane potential. And we find that a dramatically accuracy drop occur if spiking CNNs are quantized into fixed-point number without a careful study of this parameter.

Therefore, some challenging issues are still left and summed up as follows. (1) A general and efficient conversion rule for converting CNNs into spiking CNNs without any hand-tuning about parameters is necessary. (2) The optimal method of weight mapping and all parameters setting must be determined and included in conversion rule to realize high-accuracy and low-latency classification with converted spiking CNNs, and the conversion rule should be hardware-friendly to some extent. (3) There is no noise robustness evaluations for converted spiking CNNs. To solve these issues, we study the setting method of

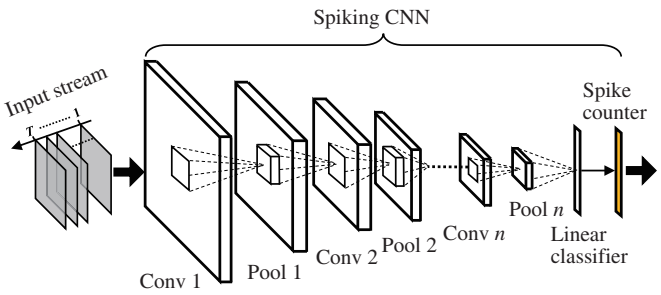

Figure 1 (Color online) Typical network architecture of spiking CNN.

all parameters in spiking CNN based on a general conversion theory we proposed and statistical features in CNN. The optimal parameters setting method are determined and included in conversion rule to convert CNNs into high-accuracy and low-latency spiking CNNs. Besides, hardware implementation is taken into consideration when determining the optimal setting method of all parameters, so our conversion rule is hardware-friendly enough towards dedicated hardware even with limited computing resources. Furthermore, we evaluate the accuracy loss caused by additive white Gaussian noise and salt-and-pepper noise.

The remainder of this paper is organized as follows. Section 2 describes spiking CNN and compares it with CNN in detail. Section 3 describes the proposed general conversion theory, the optimal parameters determination, and the complete set of conversion rule we established. In Section 4, some experimental results and analysis are given. Finally, Section 5 concludes this paper.

2 Spiking convolutional neural networks

This section describes the spiking CNN and compares it with the CNN used for conversion from three different levels: the network architecture, the neuron model, and the information processing flow. Through the comparison and analysis, the conversion issues are figured out in the end.

2.1 Network architecture

Figure 1 shows the network architecture of a typical spiking CNN. It consists of convolution layers, pooling layers, a fully connected linear classifier, and a spike counter. The convolution layers and pooling layers are arranged alternately. Each convolution layer extracts features through spiking convolution. Then, pooling layer combines the outputs of neurons cluster in one feature map into the input of one neuron in the next layer. For each neuron in the first layer, their input is a temporal spike train in the length of T time steps with time step size dt. The presence and absence of spike are represented as the binary logics "1" and "0", respectively. All input spike trains are combined together and become the input stream of T binary images. T binary images are input into the network successively with the time step dt. The linear classifier outputs several spike trains which correspond to different class channels, respectively. The spike counter counts the number of spikes on each class channel during the presentation of input stream, and the class with the maximum number of spikes is the classification result. The series of binary images can be obtained by a spike generator [28] or a biomorphic digital image sensor [32].

CNNs are trained to achieve high accuracy with its fine-tuned network, then spiking CNNs inherit CNNs' architecture with well-trained weights directly. Therefore, the architecture of CNN used for conversion is exactly same as spiking CNN except for the spike counter. The input and output of CNN are image raw data and classification result in real-value form, respectively. The entire network is trained with stochastic gradient descent method through error back propagation. Maximum and average pooling is commonly used in CNN. While average pooling can be seen as a special convolution with a kernel of uniform weights which sum to 1 and implemented in existing neuromorphic hardware more efficiently. As a result, we use average pooling only in spiking CNN and the CNN used for conversion.

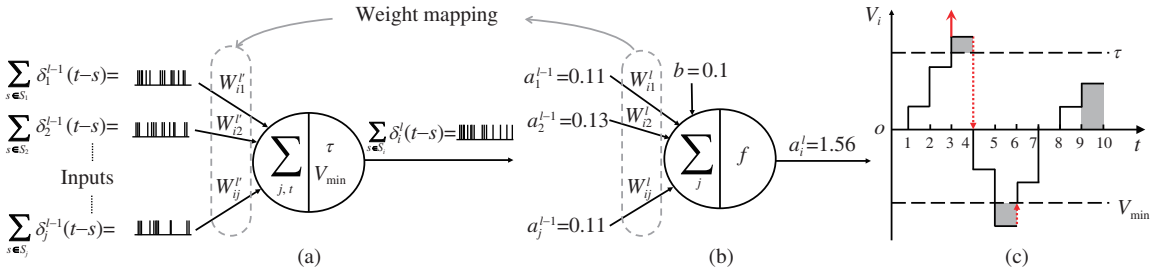

Figure 2 (Color online) Basic computing unit models. (a) IF neuron in spiking CNN, (b) CNN neuron, and (c) the dynamics of membrane potential of IF neuron in time discrete description.

2.2 Neuron model

Integrate-and-fire (IF) neuron is one of spiking neurons commonly used in spiking CNNs. It can realize high efficient spike computing in digital circuits and has already been widely supported by existing neuromorphic platforms [14–17]. Figure 2(a) shows the IF neuron model adopted in this paper. Information generated by IF neuron is in the form of spike train that can be described as comb function $\sum \delta(\cdot)$. Each IF neuron integrates the weighted inputs from the binary input stream or the presynaptic neurons, and updates its membrane potential $V_i(t)$ at each time t as follows:

$$V_i(t) = V_i(t-1) + \sum_{j \in \Gamma_j} w_{ij} \sum_{s \in S_j} \delta_j(t-s), \tag{1}$$

where w_{ij} indicates the synapse strength between neuron i and presynaptic neuron j. $S_j = \{s_j^1, s_j^2, \ldots\}$ contains the firing times of the neuron j, and Γ_j contains all presynaptic neurons connected to neuron i. Figure 2(c) shows the dynamics of membrane potential with time step t. If the membrane potential is greater than firing threshold τ, the neuron fires and outputs one spike, then membrane potential resets to zero commonly. If membrane potential is lower than $V_{\min}$, it will be reset to $V_{\min}$.

Figure 2(b) shows the CNN neuron model with parameters: weights and bias. In this paper, we adopt ReLU function as the activation function to speed up training process [1]. Due to the lack of mature learning rule, the synapse strengths of IF neurons in spiking CNN are mapped from that of the well-trained CNN neuron. Comparing these two neuron models, it is clear that IF neuron has two additional parameters: firing threshold τ and $V_{\min}$, but bias is removed. Furthermore, they are activated by different methods: the ReLU function and the firing operation, respectively. The differences are obvious while the link of weight mapping is strong, so the conversion from CNN neuron to IF neuron becomes complex.

2.3 Information representation and processing flow

The information representation in two networks are different. In spiking CNN, both the input of network and the information transfer between neurons are represented as spike train. On the other hand, in CNN, a raw image is fed into network and processed layer by layer in real-value form directly. Therefore, after conversion, the firing rate of spike train in spiking CNNs should be equivalent to the amplitude information of real-value in CNNs. The processing flow for both networks mainly includes three kinds of operations: convolution, average pooling and activation, which are all carried out by their neurons. In spiking CNN, the information processing of IF neuron includes two main steps: integration and firing. At each time step t, the tth binary image is fed into network. Each IF neuron integrates the sum of the weights with spike inputs in membrane potential. Then, if the membrane potential is greater than firing threshold, IF neuron fires and outputs one spike. These two steps repeat T times, and the information carried by spike train is transferred layer by layer. Therefore, it is obvious that only accumulations (ACC) are needed for the whole processing flow. In CNN, the information processing of neuron includes two steps: multiply-accumulation (MAC) and activation. After feeding a raw image into CNN, each neuron performs MAC operation to the inputs and weights pointwisely. Then, the result is activated through

Table 1 Comparison between CNN and spiking CNN

			CNN	Spiking CNN
Network architecture	Number of convolution layers		n	n
	Number of pooling layers		n	n
	Additional layers		×	Spike counter
	Input		An image	Series of binary map
	Output		Real-value	Num of spikes
Neuron model	Weights		W_{ij}	W'_{ij}
	Firing threshold (τ)		×	∘
	$V_{\min}$		×	∘
	Bias		∘	×
	Notion of time		×	∘
Processing flow	Information domain		Real-value	Spike (rate coding)
	Operation	Convolution	MAC	ACC
		Average pooling		
		Activation	ReLU	Firing operation

ReLU function and transferred to the next layer. Without the notion of time, the information flows through the whole network only once.

2.4 Conversion issues

Table 1 summarizes all the links and differences between two networks. Comparing with CNN, there are additional layers and more neuron parameters in spiking CNN. The information processing flow for two networks is totally different. These obvious differences increase the complexity of the conversion process from CNNs into spiking CNNs. Ideally, if information is copied perfectly from real-value domain to spike domain, spiking CNN can realize same classification accuracy as CNN. Therefore, to establish the conversion rule for converting CNNs into efficient spiking CNNs with low latency and near lossless classification accuracy, there are still some issues left: (1) we need a general conversion theory to reveal the relations between CNN and spiking CNN; (2) we must find an effective weight scaling method and determine the optimal parameters for spiking CNN to balance the classification accuracy and processing latency well; and (3) we should study the effect of membrane potential range limited by τ and $V_{\min}$ on classification accuracy.

3 The conversion rule

In this section, we put forward a general conversion theory. Then, based on the conversion theory and the statistical features of activations distribution in CNN, we determine the optimal parameters of spiking CNNs for realizing high-accuracy and low-latency classification. Finally, we establish a deterministic conversion rule to guide the conversion procedure from CNN into spiking CNN with optimal parameters setting.

3.1 Conversion theory

We assume that there is a one-to-one correspondence between the IF neuron in spiking CNN and the neuron in CNN. For the network with L layers, $W^l, l \in \{1, \ldots, L\}$ denotes the weight matrix connecting neurons between layer $l-1$ and l. In CNN, the ReLU activation of the neuron i in layer l is

$$a_i^l = \max\left(0, \sum_{j=1}^{J^{l-1}} W_{ij}^l a_j^{l-1}\right), \tag{2}$$

where J^{l-1} indicates the number of neurons in layer $l-1$ connected to the neuron i in layer l. Eq. (2) starts with a_j^0 which represents the pixel value of input image for CNN. In spiking CNN, each IF neuron

integrates the sum of the weighted input spike at each time step t:

$$Z_i^l(t) = \sum_{j=1}^{J^{l-1}} W_{ij}^l \sum_{s\in S_j} \delta_j^{l-1}(t-s), \tag{3}$$

and accumulates it into its membrane potential V_i^l according to (1). Since every input pattern is presented for T times with time step $\mathrm{d}t$, the firing rate of each IF neuron over a period of time $T\mathrm{d}t$ is defined as

$$r_i^l(T) = \frac{N_i^l(T)}{T\mathrm{d}t} = \frac{\sum_{t=1}^{T}\sum_{s\in S_i}\delta_i^l(t-s)}{T} r_{\max}, \tag{4}$$

where $N_i^l(T)$ is the total number of spikes emitted by neuron i during $T\mathrm{d}t$. The time step $\mathrm{d}t$ is the minimum time unit for once spike computing, which is always decided by circuits design or configuration of neuromorphic hardware. So $r_{\max} = 1/\mathrm{d}t$ (Hz) is the highest firing rate. The total residual membrane potential of each IF neuron within $T\mathrm{d}t$ is the part of membrane potential that does not contribute to generate spike outputs, and it can be computed as

$$\varepsilon_i^l(T) = \sum_{t=1}^{T} Z_i^l(t) - N_i^l(T)\tau^l, \tag{5}$$

where τ^l is the threshold of all IF neurons in layer l. Total residual $\varepsilon_i^l(T)$ includes three parts shown in Figure 2(c) with shadow, and it can be written in detail as

$$\varepsilon_i^l(T) = \sum_{s\in S_i}(V_i^l(s) - \tau^l) + \sum_{u\in U_i}(V_i^l(u) - V_{\min}^l) + V_i^l(T), \tag{6}$$

where $V_{\min}^l$ is the minimum membrane potential of all IF neurons in layer l and $U_i = \{u_i^1, u_i^2, \ldots\}$ represents the time of resetting membrane potential when it is below $V_{\min}^l$. Therefore, the three terms in the right of (6) are the total residual membrane potential after firing and $V_{\min}$-resetting, and membrane potential left at the last time step T, respectively.

Generally, the input image for CNN is normalized and then transformed into the input spike train for spiking CNN through a spike generator proposed by Cao et al. [28] commonly. Therefore, pixel value $a_j^0 \in [0, 1]$ and the firing rate of each input spike train can be written as

$$r_i^0 = a_i^0 r_{\max}. \tag{7}$$

Based on the total residual (5) and firing rate (4), we work out a recursive expression about the firing rate of the IF neurons in two connected layers. And the recursive expression is solved by inserting the firing rates of previous layers iteratively until the known firing rates of input spike trains listed in (7). The detailed calculations are given in Appendix A; a general description of relation between firing rate and real-value activation is obtained as

$$r_i^l(T) = \frac{a_i^l}{\tau^l\tau^{l-1}\cdots\tau^1} r_{\max} - \Delta\varepsilon_i^l, \tag{8}$$

where $\Delta\varepsilon_i^l$ is the approximation error:

$$\Delta\varepsilon_i^l = \frac{r_{\max}}{T}\left(\frac{\varepsilon_i^l(T)}{\tau^l} + \frac{1}{\tau^l\tau^{l-1}}\sum_{j=1}^{J^{l-1}} W_{ij}^l\varepsilon_j^{l-1}(T) + \cdots + \frac{1}{\tau^l\tau^{l-1}..\tau^1}\sum_{j=1}^{J^{l-1}} W_{ij}^l\cdots\sum_{j=1}^{J^1} W_{ij}^2\varepsilon_j^1(T)\right). \tag{9}$$

As expected, Eq. (8) reveals the relationship that the firing rates of IF neurons in spiking CNN approximate the real-value ReLU-activations of neurons in CNN. We regard the first term in the right of (8) as "ideal firing rate". Although the approximation error becomes greater with the network going deeper, the ideal firing rate is much greater than the approximation error when T is larger enough. So far,

the conversion theory is built and it is the first time that all parameters of spiking CNN are taken into consideration, including weight, firing threshold, and $V_{\min}$ especially. Besides, our theory is also suitable for the modified IF neuron with "reset by subtraction" [31], which does not have the lowest limitation $V_{\min}$ of membrane potential and their residual parts after firing and reset still remain in membrane potential. The total residual of modified IF neuron only contains the third term in (6). Therefore, the approximation relation derived from our theory can cover the previous theory proposed by [31] and is more general.

3.2 Linear weight mapping

We assume that the firing threshold of the neurons in layer l can be written as

$$\tau^l = \tau f^l, \tag{10}$$

where τ is the uniform firing threshold for neurons in all layers and f^l is a factor that is different for each layer. Putting (10) into (8) and unfold a_i^l with (2), we can get

$$r_i^l(T) = r_{\max}\frac{1}{(\tau)^l}\max\left(0, \sum_{j=1}^{J^{l-1}} W_{ij}^{l'} \max\left(0, \sum_{j=1}^{J^{l-2}} W_{ij}^{(l-1)'} \cdots \max\left(0, \sum_{j=1}^{J^0} W_{ij}^{1'} a_j^0\right)\cdots\right)\right) - \Delta\varepsilon_i^{l'} = \frac{a_i^{l'}}{(\tau)^l} r_{\max} - \Delta\varepsilon_i^{l'}, \tag{11}$$

where $W_{ij}^{l'} = W_{ij}^l / f^l$ is the f^l-scaled weight, and $a_i^{l'}$ is the activation convoluted by f^l-scaled weight. Also, Eq. (9) can be rewritten as

$$\Delta\varepsilon_i^{l'} = \frac{r_{\max}}{T}\left(\frac{\varepsilon_i^{l'}(T)}{\tau} + \frac{1}{(\tau)^2}\sum_{j=1}^{J^{l-1}} W_{ij}^{l'}\varepsilon_j^{(l-1)'}(T) + \cdots + \frac{1}{(\tau)^l}\sum_{j=1}^{J^{l-1}} W_{ij}^{l'} \ldots \sum_{j=1}^{J^1} W_{ij}^{2'}\varepsilon_j^{1'}(T)\right), \tag{12}$$

where $\varepsilon_j^{l'}(T) = \varepsilon_j^l(T)/f^l$. After introducing factor f^l, spiking CNN consists of the neurons with uniform firing threshold τ and scaled weights $W_{ij}^{l'}$. Comparing (8) and (11), we found that there is a balance between the weights and the firing threshold controlled by the factor f^l, which can be seen as the "scaling factor" for linear weights rescaling. And Eqs. (8), (9), (11) and (12) are in similar form, which demonstrate that there is not any other approximation errors introduced by weights rescaling.

According to (8) and (11), there are two methods to map weights and set firing thresholds, which are described in follows. The first method is based on (8). We can map the weight matrices of CNN into that of spiking CNN directly and set the firing thresholds τ^l for neurons in different layers to be different, respectively. The second method is based on (11). The firing threshold of all neurons are set as the uniform threshold τ and the weight matrix W^l is rescaled by scaling factor f^l that is different for each layers before mapping. For convenience, uniform threshold $\tau = 1$, so scaling factor f^l equals the firing thresholds τ^l in the first method according to (10). The first and second methods correspond to two proposed method: "threshold rescaling" [33] and "weight normalization" [29], respectively. These two methods for setting parameters seem different, but our theoretical analysis indicates that they are completely equivalent. To prevent the firing rate saturation, the scaling factor and firing threshold of first hidden layer are set as the maximum activation within layer, and for other layer l, f^l and τ^l are set as the ratio of the maximum activation within layer l and scale factor of previous layer. Therefore, the maximum activation of each layer is scaled to the maximum firing rate $r_{\max}$ in spike domain [29, 33].

When spiking CNNs are implemented in neuromorphic hardware platforms, for both mapping methods we mentioned in the previous paragraph, the firing threshold and weights of each neuron can be configured individualized if there are enough spike computing units to realize one-to-one mapping from neuron model to hardware. However, with network going deeper, computing units always need to be reused for decreasing cost. If we adopt "threshold rescaling" method, the τ^l needs to be configured differently and updated repeatedly for processing different layers, which is a burden on hardware. While, if we adopt "weight normalization" method, the linear weight mapping can been seen as a step in training process

before hardware implementation, so the update of firing threshold is avoided. Though "weight normalization" is more efficient, Rueckauer et al. [31] pointed out that serious latency for passing information to deeper layers is caused by its overly pessimistic scaling factors. Obviously, information transfers to deeper layers faster and more processing time is saved when neurons are in high firing rate, and approximation error has effects on classification accuracy. As shown in (11) and (12), if the scaling factor is too small, both the ideal firing rate and approximation error increase. The processing time is shortened but the classification accuracy becomes worse. On the contrary, if the scaling factor is too large, the processing time is lengthened while the classification accuracy becomes better. So it is clear that there is a trade-off between classification accuracy and processing time controlled by scaling factor. In the following part, we try to find out the optimal scaling factors that can achieve faster classification with lossless accuracy.

3.3 Scaling factor determination

The setting method of weights and threshold in our conversion rule is based on linear weight mapping method with uniform firing threshold $\tau = 1$. In order to optimize the accuracy-latency trade-off, we propose a robust "n-scaling" method to determine the optimal weights scaling factors. First we consider the setting of scaling factor in neuron level through our conversion theory, then determine the optimal scaling factor with the statistical features of activations distribution in CNN. Because each scaling factor f^l is used for rescaling the weight matrix of layer l only, the scaling factor for each layer can be set separately. If the weight matrices of previous $l-1$ layers have been rescaled by their scaling factor respectively but the weight matrix of current layer l not been scaled, the activations of neurons in layer l can be written as

$$a_i^{l''} = \frac{a_i^l}{f^{l-1}\cdots f^1 f^0} \tag{13}$$

with $f^0 = 1$. Assuming $f^l = n \cdot \max(a_i^{l''})$, Eq. (11) can be rewritten as

$$r_i^l(T) = \frac{a_i^{l''}}{n \cdot \max(a_i^{l''})} r_{\max} - \Delta\varepsilon_i^{l'}. \tag{14}$$

If n is set as 1, the method for setting scaling factor is the same as "weight normalization" with the shortcoming of serious latency. If n decreases, the firing rate increases and the latency is shortened. However, there are some neurons within layer l in CNN that their real-value activations $a_i^{l''}$ are higher than $n \cdot \max(a_i^{l''})$ when $n < 1$. As indicated in (14), the ideal firing rates of their corresponding IF neuron in spiking CNN are greater than $r_{\max}$. But they can only fire at saturated firing rate $r_{\max}$ actually, so the approximation error $\Delta\varepsilon_i^l$ increases. When the approximation error becomes larger than actual firing rate $r_i^l(T)$, we believe the approximation relationship between firing rate and real-value activation no longer holds. In order to prevent information distortion, the actual firing rate should be at least not lower than approximation error. Adding this limitation into (14), we can derive

$$2r_i^l(T) \geqslant \frac{a_i^{l''}}{n \cdot \max(a_i^{l''})} r_{\max}. \tag{15}$$

As analyzed above, in extreme case $a_i^{l''} = \max(a_i^{l''})$, the firing rate is saturated and equals to $r_{\max}$. Therefore, the range of n is worked out from (15), that is $n \in [0.5, 1]$. To realize the lowest latency, n should be set as the smallest value 0.5. Therefore, the scaling factor can be written as

$$f^l = 0.5 \cdot \frac{a_M^l}{f^{l-1}\cdots f^1 f^0}, \tag{16}$$

where a_M^l is the maximum activation of layer l within training set. So far, the above analysis is based on the conversion theory in neuron level only. However, it is still possible that most neurons within spiking CNN fire at saturated rate $r_{\max}$ with high level of approximation error, which may worsen classification accuracy.

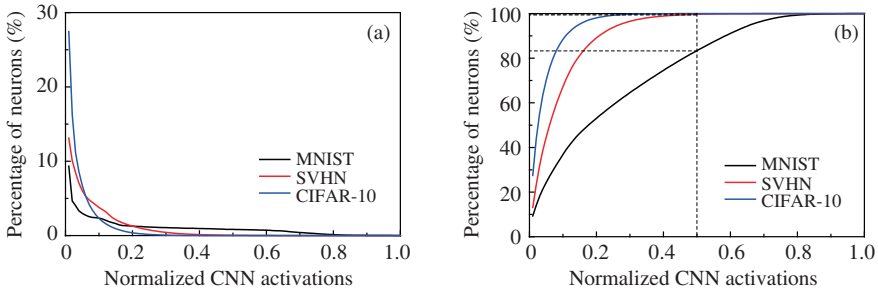

Figure 3 Activations distribution. (a) Distribution of normalized non-zero activations in the first convolution layer of CNN for MNIST, SVHN, and CIFAR-10 datasets respectively; (b) integral curve of activation distribution for three datasets.

Therefore, scaling factors need to be optimized with the IF neuron firing rates distribution of the whole spiking CNN. We infer the firing rates distribution through its corresponding activations distribution in CNN because of the approximation relation existed between two networks. Taking the first convolution layer as an example, its non-zero real-value activation distribution on MNIST, SVHN, and CIFAR-10 datasets is shown in Figure 3(a). For comparing the distribution pattern over three datasets intuitively, the activations have been normalized into $[0, 1]$ by dividing the maximum activations a_M^1 of first layer within training set of each dataset. We found that the activations of most neurons distribute in very low level, especially below $0.5a_M^1$. Calculating the area under curves that change with respect to the normalized activations for all three datasets, we obtain their integral curve of activation distribution, shown in Figure 3(b). So these integral curves describe that what percentage of neurons with activation below a specific value when this value changes from 0 to 1. Obtained by our statistic, the percentage of neurons with activation ranging from 0 to $0.5a_M^1$ is 83.37%, 99.66%, and 99.98% for MNIST, SVHN, and CIFAR-10 dataset, respectively shown in Figure 3(b). Based on above analysis about (14), if we set $f^l = 0.5a_M^l$, the same percentage of the neurons with unsaturated firing rate over three datasets can be inferred. To control the information distortion within an acceptable range, we suggest that more than 90% of activations should be less than the selected scaling factor f^l for every layer in CNN. Therefore, to satisfy this limitation, all weight scaling factor f^l should be optimized according to each layer's activation distribution of CNN. We found that modifying the scaling factor of first layer can improve the information distortion in higher layers. Usually, $f^1 = \lceil 0.5a_M^1 \rceil$ is a quick and efficient setting method to satisfy this limitation. After modification, the percentage of neurons with unsaturated firing rate in the first layer increases to 92.50%, 99.82%, and 99.99% for MNIST, SVHN, and CIFAR-10 dataset, respectively, and this limitation is also satisfied in higher layers.

3.4 $V_{\min}$ determination

Indicated in Figure 2(c), the dynamic range of membrane potential is limited by τ and $V_{\min}$. Firing threshold τ has been determined through "n-scaling" weight mapping method we propose in Subsection 3.3 by balancing higher accuracy and lower latency. While, $V_{\min} < 0$ commonly. When the membrane potential is not limited by $V_{\min}$, there is a lowest level that membrane potential can reach. If $V_{\min}$ is set as a value that more negative than that lowest membrane potential, it is difficult to trigger $V_{\min}$-resetting mechanism, so there is barely not residual membrane potential caused by resetting. And the second part of approximation error can be ignored according to (6), thereby improving the classification accuracy. When $V_{\min}$ become greater and getting closer to 0, the possibility that membrane potential reaches the limitation of $V_{\min}$ grows up, leading to greater residual membrane potential left after $V_{\min}$-resetting. As the result, approximation errors increase and classification accuracy becomes worse. So, there exists the greatest value that $V_{\min}$ can be set to ensure lossless accuracy, and the minimum dynamic range of membrane potential is also limited by this greatest value.

When data is represented in floating-point format, the representation range of floating-point numbers is much broader than the dynamic range of membrane potential so that $V_{\min}$ does not have to be set carefully, like when simulating in computers or running on dedicated hardware with floating-point arithmetic.

However, there are still many dedicated hardware platforms with numbers represented in fixed-point or integer [14,17,34,35]. Therefore, the trade-off between representation range and precise should be studied carefully and the minimum dynamic range of membrane potential should be determined to improve computing precision.

If the neurons in layer $l-1$, which are connected to the neurons in layer l through negative weight, fire at high frequency, the membrane potential of the neurons in layer l will decrease into a very low level. This indicates that $V^l_{\min}$ relates to the negative weight strengths within W^l and the firing rates of the neurons in layer $l-1$ both. Based on the approximation relationship we derive in Subsection 3.1, the firing rates of the neurons in spiking CNN approximate the activations of corresponding neurons in CNN, so $V^l_{\min}$ also can be inferred from CNN. Using W^l_{neg} to represent the weight matrix that only contains negative values of W^l in CNN, the activations in layer l are convoluted by matrix W^l_{neg} and the minimum convolution result m^l is obtained. Therefore, $V^l_{\min}$ in spiking CNN relates to m^l in CNN. But the weight matrix of CNN is scaled by scaling factor f^l before mapping to that of spiking CNN, so $V^l_{\min}$ should be equal to f^l-scaled m^l:

$$V^l_{\min} = \frac{m^l}{f^l f^{l-1} \cdots f^1}. \tag{17}$$

For avoiding complicated operations in the hardware, $V_{\min}$ for all neurons in spiking CNN is set to the same and equal to the minimum $V^l_{\min}$ among all layers. Therefore, the minimum dynamic range of membrane potential is obtained and the format of number representation is determined thereby. For example, the integer bit-width N of fixed-point membrane potential is $N = \min_N\{\max(\tau, |\min(V^l_{\min})|) < 2^N\}$. And, the $V_{\min}$-reset operation of the membrane potential can be replaced by the underflow reset operation in hardware actually.

3.5 Conversion rule

Based on the proposed general conversion theory and the statistical features of activations distribution in CNN, we establish a deterministic conversion rule to convert CNNs into spiking CNNs, which is shown as follows: (1) Train a ReLU-based CNN with zero biases and average-pooling through the error back propagation method. Obtain the well-trained weight matrix of each layer. (2) Based on the well-trained CNN and the training dataset, obtain the activations a^l_i and the maximum activation a^l_M of each layer. (3) Based on the activations a^l_i and the negative weight matrix W^l_{neg}, obtain the minimum result m^l. (4) Determine the scaling factor f^l for each layer by the method proposed in Subsection 3.3 and rescale the weight matrix of each layer by their scaling factor f^l, while the uniform firing threshold τ is set as 1. And (5) determine the $V_{\min}$ by the method proposed in Subsection 3.4. The pseudocode of conversion rule is shown in Appendix B.

Hereto, all conversion issues mentioned in Section 2 are solved and CNNs can be converted into spiking CNNs definitively by our conversion rule. Our method has the following advantages. Firstly, we proposed a general conversion theory for converting ReLU-based CNNs into IF neuron based spiking CNNs. The theory derives the approximation relations (8) and (9) without any preconditions of IF neuron reset mechanism nor the form of input spike train. And it is the first time that the approximation relations involve all parameters (even $V_{\min}$) of spiking CNN and give a theoretical support for parameters determination. Secondly, based on the conversion theory and the statistical features of activations distribution in CNN, we determined the optimal weight scaling factors to realize lossless classification accuracy and low latency classification with converted spiking CNNs. Finally, the parameters setting methods given by our conversion rule are hardware-friendly towards dedicated hardware with limited computing resources. We adopt "n-scaling" weight mapping method to maintain firing threshold of all layers as uniformed $\tau = 1$, which help to simplify hardware operations. And $V_{\min}$ is studied to determine the minimum dynamic range of membrane potential, which helps to optimize the trade-off between representation range and precision of fixed-point number. It guarantees the same performance as simulation when spiking CNNs are implemented on hardware which is not promised in previous researches. In summary, the conversion rule is an efficient method to convert CNNs into high-accuracy and low-latency spiking CNNs.

Figure 4 Image samples. 10 classes of objects from MNIST, SVHN and CIFAR-10 datasets are shown in different rows from top to bottom, respectively.

Table 2 Architecture of CNNs for MNIST, SVHN and CIFAR-10 dataset

MNIST	SVHN	CIFAR-10
Input 28×28 gray image	Input 32×32 RGB image	Input 24×24 RGB image
conv5-12, ReLU	conv5-32, ReLU	conv5-64, ReLU
avg-pool/2	avg-pool/2	avg-pool/2
conv5-64, ReLU	conv5-96, ReLU	conv5-64, ReLU
avg-pool/2	avg-pool/2	avg-pool/2
FC-10	conv3-64, ReLU	conv3-64, ReLU
	avg-pool/3	conv1-64, ReLU
	FC-10	

4 Experimental result

This section evaluates the performance of spiking CNNs that converted by our conversion rule. The classification accuracy and latency of converted spiking CNNs are assessed, and we evaluate the impact of noise on spiking CNNs' performance. Besides, we also evaluate the classification performance when computing with fixed-point number.

4.1 Experimental setup

We use three standard image datasets: MNIST, SVHN, and CIFAR-10 to evaluate the performance of the converted spiking CNNs. Figure 4 shows some image samples from three datasets. MNIST dataset consists of 70000 28×28 gray handwritten digit images in 10 classes. SVHN is a larger dataset that contains 630420 32×32 color street view house number images in 10 classes[1)] [36]. CIFAR-10 consists of 60000 32×32 color images in 10 classes that ranges from transportation to animals. Consequently, the classification task of CIFAR-10 is much more difficult than the other two datasets. The number of time step T is set as 100, 300, 400 for MNIST, SVHN, and CIFAR-10 dataset, respectively.

The architecture of CNNs used for conversion are listed in Table 2. The parameters of convolution layer are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The stride of average pooling is shown as "avg-pool/⟨stride⟩". And "FC-⟨number of classes⟩" indicates a fully connected layer with the output class number. In order to compare the performance with previous researches, the network architecture and training tricks used for MNIST and CIFAR-10 dataset are the same as [29] and [28], respectively. Data augmentation is used to enlarge CIFAR-10 dataset so that the original images are tailored into 24×24, which is the same as [28]. All the experiments are implemented in Matconvnet [37].

4.2 Accuracy and latency

4.2.1 *Spiking CNN based on conversion rule*

A high-accuracy CNN is the basis for conversion. We train CNN and choose the one with the highest accuracy on test set. The best accuracy achieved by CNNs for MNIST, SVHN, and CIFAR-10 datasets is 99.11%, 96.60%, and 80.81%, respectively. Then we convert CNNs into spiking CNNs by our conversion

1) We follow the method proposed by Goodfellow et al. [36] to build a validation set and training set.

Table 3 Classification accuracy of different types of SNNs on three datasets[a)]

Dataset	Network-type	Method	Accuracy
MNIST [38]	SNN	Balanced learning + STDP	98.64%
MNIST [39]	SNN	NormAD	98.17%
MNIST [31]	Converted spiking CNN	Modified IF neuron	99.44% (≈0%)
MNIST [29]	Converted spiking CNN	Weight Normalization	99.11% (0.04%)
MNIST	**Converted spiking CNN (this paper)**	**Conversion rule**	**99.09% (0.02%)**
SVHN [33]	Converted spiking CNN	Threshold rescaling	93.66% (2.27%)
SVHN [30]	Spiking CNN	Synaptic filter	93.92% (0.43%)
SVHN	**Converted spiking CNN (this paper)**	**Conversion rule**	**96.33% (0.27%)**
CIFAR-10 [28]	Spiking CNN	None	77.43% (1.66%)
CIFAR-10 [30]	Spiking CNN	Synaptic filter	83.54% (2.43%)
CIFAR-10	**Converted spiking CNN (this paper)**	**Conversion rule**	**80.41% (0.40%)**

a) Our results are shown with bold font.

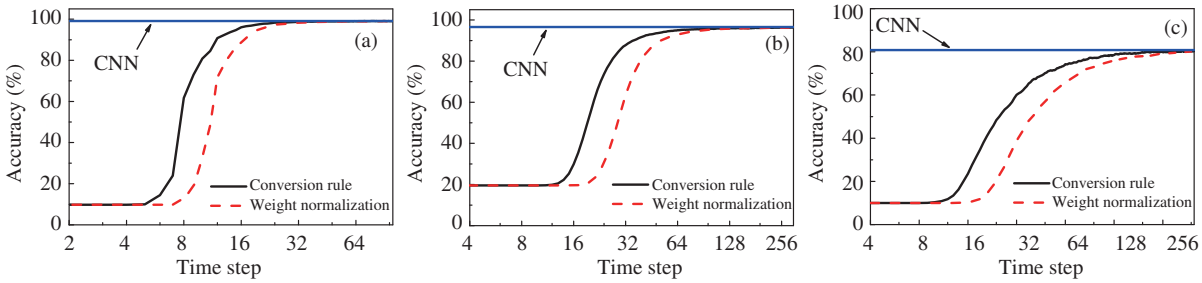

Figure 5 Classification accuracy of spiking CNNs over time step, for MNIST (a), SVHN (b), and CIFAR-10 dataset (c), respectively. "CNN" indicates the classification accuracy of CNN, which is independent of time step.

rule. The classification accuracy of converted spiking CNNs achieved 99.09%, 96.33%, and 80.41% for MNIST, SVHN, and CIFAR-10 datasets, respectively. We found that the accuracy loss is only 0.02%, 0.27%, and 0.40% for three datasets, respectively. Especially, the accuracy drop for CIFAR-10 dataset is much less than 1.66% reported by [28] and the accuracy we achieved is also higher. Table 3 [28–30,30,31, 33,38,39] compares the converted spiking CNNs with other types of SNNs. For converted spiking CNN, accuracy loss is listed along with accuracy in brackets. The results suggest that our conversion rule is much more efficient than brain-like learning rule [38,39]. Besides, we notice that state-of-the-art accuracy is realized with much more complex modified neuron models [30,31]. However, they sacrifice the hardware efficiency by adding additional operations or circuits to realize their modifications. Comparing with other methods, our conversion rule is a effective approach to achieve near lossless classification with converted spiking CNNs, and is more hardware-friendly. Furthermore, we evaluate the power consumption of our spiking CNNs by simulating the network operations and counting the total number of spikes generated during classification with the method proposed by Rueckauer et al. [31]. The spike numbers used by our method are about 1.23×10^7, 2.81×10^8, and 2.28×10^8 for MNIST, SVHN, and CIFAR-10 dataset, respectively. According to our statistics, the number of spikes used by spiking CNNs with our conversion rule is only 88%, 79%, and 62% of that used by weight normalization on MNIST, SVHN, and CIFAR-10 dataset, respectively, leading to the same degree of power consumption saved. So it suggests that higher power efficiency is benefited from our conversion rule.

Figure 5 shows the classification accuracy over time step for both spiking CNNs converted by our conversion rule and weight normalization proposed in [29]. The accuracy curves of the conversion rule based spiking CNNs rise earlier and converge to a stable accuracy faster than that of weight normalization based spiking CNNs for all three datasets. The accuracy achieved by our spiking CNNs is 99.03% at $T =$ 67 for MNIST, 96.28% at $T = 218$ for SVHN, and 80.03% at $T = 245$ for CIFAR-10, which is the stable accuracy achieved by weight normalization at $T = 100$, 300, 400, respectively. Therefore, 33%, 27%, and 39% of processing time is shorted for three datasets respectively, which is attributed to our conversion

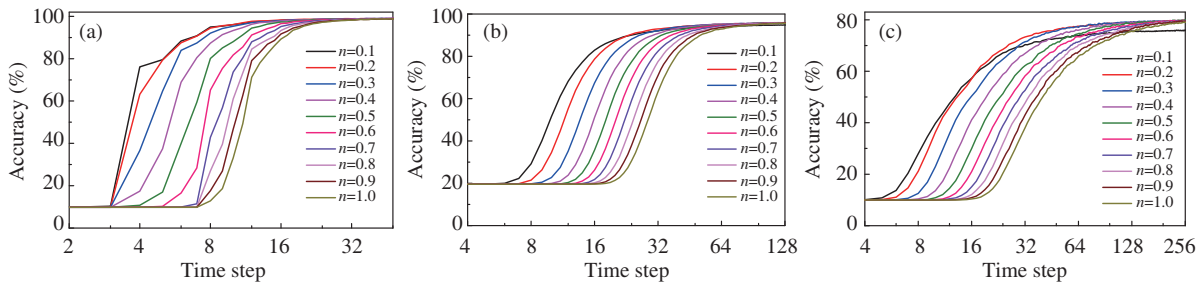

Figure 6 Dependence of latency on scaling factor for MNIST (a), SVHN (b), and CIFAR-10 dataset (c), respectively.

Table 4 Stable accuracy and approximation error $\Delta\varepsilon$ of spiking CNNs with different n-values on MNIST, SVHN, and CIFAR-10 dataset, respectively

n	MNIST		SVHN		CIFAR-10	
	Accuracy (%)	$\Delta\varepsilon$	Accuracy (%)	$\Delta\varepsilon$	Accuracy (%)	$\Delta\varepsilon$
0.1	98.75	0.582	95.13	0.100	76.10	0.278
0.2	99.06	0.138	96.04	0.035	79.42	0.079
0.3	99.12	0.066	96.10	0.021	79.82	0.040
0.4	99.12	0.042	96.19	0.015	80.08	0.026
0.5	99.13	0.031	96.30	0.012	80.14	0.019
0.6	99.10	0.024	96.33	0.010	80.38	0.014
0.7	99.13	0.020	96.30	0.009	80.46	0.012
0.8	99.10	0.017	96.35	0.007	80.53	0.010
0.9	99.15	0.015	96.43	0.007	80.55	0.008
1.0	99.13	0.013	96.31	0.006	80.29	0.007

rule. Furthermore, the stable accuracy of our spiking CNNs is always a little bit higher than that of the weight normalization based spiking CNNs because lower latency leads to more efficient convergence within same period of time. Therefore, these results demonstrate that our conversion rule provides an effective procedure to realize low-latency spiking CNN.

Besides, in order to evaluate whether the suggestion of $V_{\min}$ given by conversion rule in Subsection 3.4 is optimal, we carry out simulations to assess the classification accuracy loss when all data and computation are transformed from floating-point format into fixed-point format. During simulation, all weights and membrane potentials are represented in 16-bit fixed-point number when doing inference with spiking CNNs. And the network architecture of spiking CNN remains the same with that listed in Table 2 for each dataset. The simulation results are that spiking CNNs achieve 99.10%, 95.30%, and 80.41% accuracy on MNIST, SVHN, and CIFAR-10 dataset, respectively. Comparing with the accuracy we achieved in Table 3, except for 1.03% accuracy loss on SVHN dataset, there is no accuracy loss on the other two datasets. The two main reasons of resulting accuracy lose are analyzed in the following. Firstly, through statistic, we found that up to about 38% of weights closed to zero has been ignored for SVHN dataset when data is quantized into 16-bit fixed-point number. Secondly, the quantization errors behave like other additional errors within spiking CNN, and they combine together to form the approximation error.

4.2.2 *Dependence of accuracy and latency on scaling factor*

The scaling factor determined by our "n-scaling" weight mapping method can be represented by the value n, and we evaluate the performance of spiking CNNs with different n-values ranging from 0 to 1 with step = 0.1. Figure 6 shows latency of the converted spiking CNNs with different n-value. We found that the latency of spiking CNNs for all three datasets is improved with n decreasing, which are coincident with our conclusion derived from conversion theory. Their stable accuracy and approximation errors at the end of simulation[2)] are listed in Table 4. Because of the cumulative effect of approximation error, we use the

2) In order to converge to stable accuracy when n near 1, T is extended to 200, 400, and 600 for MNIST, SVHN, and CIFAR-10, respectively.

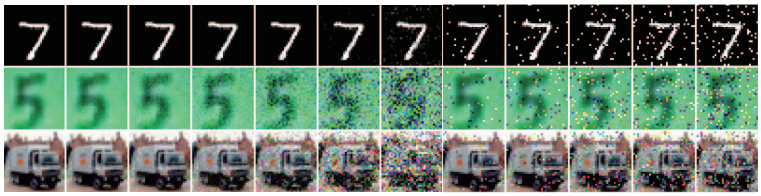

Figure 7 Noisy image samples. The noisy samples of MNIST, SVHN, and CIFAR-10 dataset are shown in different rows from top to bottom. The samples of each row listed from left to right are original images, AWGN distorted images with SNR = 30, 25, 20, 15, 10, 5 dB, noisy images with noise density = 3%, 6%, 9%, 12%, 15% of SP noise, respectively.

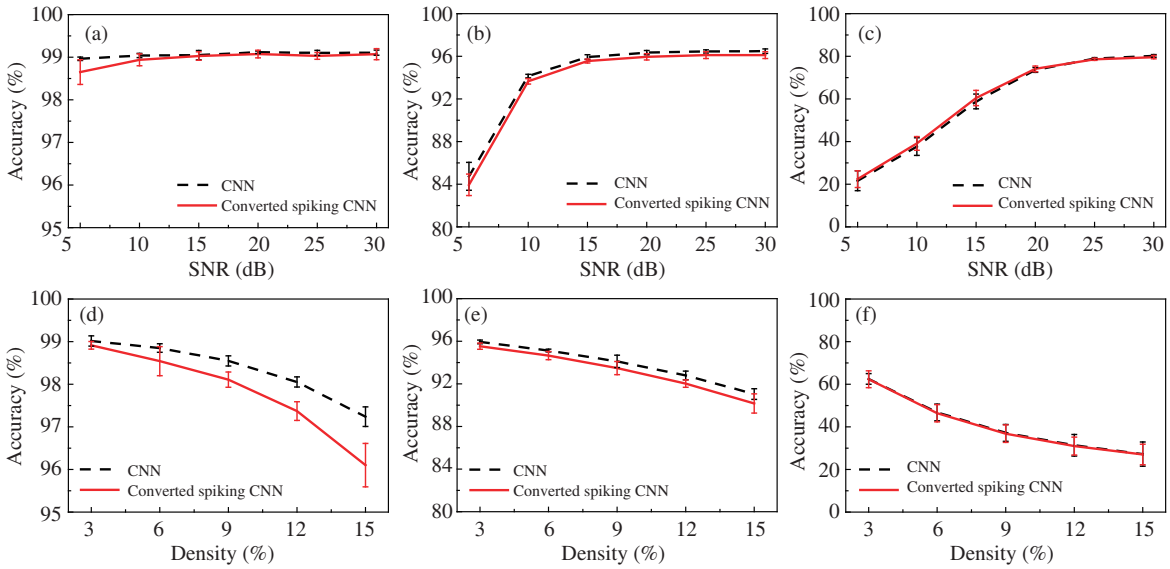

Figure 8 Classification accuracy of spiking CNNs and CNNs with different forms and degrees of noise. (a)–(c) show the accuracy when MNIST, SVHN, and CIFAR-10 dataset distorted by different levels of AWGN, respectively; (d)–(f) show the accuracy when MNIST, SVHN, and CIFAR-10 dataset distorted by different density of SP noise, respectively. The error bars indicate the accuracy range of five simulations.

average approximation errors of the neurons in the last layer to represent the approximation error of the whole network. For MNIST, the effect of approximation error is nearly neglectable because the network is shallow and the classification task is relatively simple. For other two datasets, the trends shown in Table 4 meet the analysis based on conversion theory that the approximation error increases and accuracy worsens when n becomes smaller. Over three datasets, the accuracy achieved with n-value closing to 1 is similar, but decreases sharply when n goes down. The value n determined by our conversion rule are 0.62, 0.55, and 0.57 for MINST, SVHN, and CIFAR-10 datasets, respectively, which are the turning points that the accuracy worsens sharply if n continue decreasing. Therefore, the results indicate that the scaling factors determined by our conversion rule are the optimal to realize near lossless classification accuracy with lowest latency.

4.3 Noise effect on accuracy

To assess the noise robustness of converted spiking CNN, we add two types of noise that are common in sensors: additive white Gaussian noise (AWGN) and salt-and-pepper (SP) noise into three datasets, and evaluate their impacts on the classification accuracy of spiking CNN. Some noisy samples are shown in Figure 7. The test set of each dataset is distorted by several degrees of AWGN with signal-to-noise ratio (SNR) = 5, 10, 15, 20, 25, 30 dB and SP noise with density = 3%, 6%, 9%, 12%, 15%. The classification accuracy with noisy images is shown in Figure 8, which are obtained by averaging five times simulation results.

The accuracy of spiking CNNs shows similar behavior with their corresponding CNNs, which prove

Table 5 Accuracy loss on three datasets when images are distorted by different degrees of AWGN and SP noise

ΔAcc	Additive white Gaussian noise (SNR)						Salt-and-pepper noise (density)				
	30	25	20	15	10	5	3%	6%	9%	12%	15%
MNIST (%)	0.03	0.07	0.04	0.02	0.10	0.31	0.10	0.31	0.44	0.68	1.14
SVHN (%)	0.37	0.35	0.39	0.37	0.49	0.80	0.41	0.47	0.63	0.76	0.89
CIFAR-10 (%)	0.61	0.39	−0.52	−1.63	−1.46	−0.79	0.18	0.37	0.35	0.36	0.19

that the converted spiking CNN inherits the noise robustness from CNN. As shown in Figure 8(c) and (f), the dramatically decreases in accuracy can be seen on CIFAR-10 dataset with two forms of noise because its original images are already blurrier and difficult for recognizing as the samples shown in the third row in Figure 7. For SVHN dataset, the maximum 12.17% and 5.37% of accuracy drop are caused by the AWGN and SP noise, respectively. For MNIST dataset, the AWGN and SP noise cause maximum 0.42% and 2.81% accuracy losses, respectively. Especially for MNIST, we achieve 98.11% and about 98.79% when objects are distorted by 9% and 4% of SP noise, which are much higher than 96.5% and 97.3% reported by [40]. Therefore, our converted spiking CNNs perform better noise robustness.

The accuracy loss between CNN and spiking CNN is defined as the accuracy of CNN subtracts that of spiking CNN, represented as ΔAcc. Table 5 shows the dependence of accuracy loss ΔAcc on different degrees of AWGN and SP noise over three datasets. Obviously, the accuracy loss caused by SP noise is always greater than that caused by AWGN. If the pixel noise is represented as Δn_i^0, then noisy pixel value is $a_i^0 + \Delta n_i^0$. According to (4) and (7), the total number of spikes produced by spike generator during $T\mathrm{d}t$ is $N_i^0(T) = T \cdot (a_i^0 + \Delta n_i^0)$. Due to the limited number of time step T, the value of $N_i^0(T)$ is an integer that approximately equals $T(a_i^0 + \Delta n_i^0)$ actually. If $|\Delta n_i^0| < 1/T$, $N_i^0(T)$ still approximately equals to Ta_i^0 and there is nothing effected by noise. Therefore, rate coding is a fault tolerant encoding method when the additive noise is less than the inverse of the number of total time step. When Δn_i^0 is great enough to distort the encoded firing rates of input spike trains, Δn_i^0 will pass through the whole network and it behaves like the total residual $\varepsilon_i^l(T)$ in (9) because they are both additive errors. So it is possible that the distortion caused by AWGN is able to offset the effect induced by total residual and improve the approximation error. Therefore, we can even notice a slight improvement on CIFAR-10 dataset in Table 5 when images are distorted by AWGN with SNR $<$ 20 dB. However, the SP noise distorted pixels are changed into the maximum or minimum real-value within sensors' dynamic range directly. After rate coding, the firing rates of distorted pixels are $r_{\max}$ or 0 without carrying any original image information. Therefore, comparing with AWGN, the distortion caused by SP noise is so severe that even a slight density of noise may result an obvious effect on accuracy.

5 Conclusion

This paper proposed a general conversion theory that clarifies the relations between CNN and spiking CNN from parameters to information processing for the first time. It is a theoretical support to set up the optimal converted spiking CNN. Though a similar theory has already been proposed by Rueckauer et.al [31], our theory does not have any preconditions and is more general for almost any converted spiking CNNs. Besides, the theory is also helpful for analyzing how additive errors transfer through the network and accumulate in neurons within each layer. Based on the theory and the statistical features of the activations distribution in CNN, we establish the deterministic conversion rule to convert arbitrary ReLU-based CNNs into equivalent spiking CNNs with IF neurons. The conversion rule guides the conversion procedure and gives the definite methods for setting all parameters of spiking CNN clearly. For the first time, our conversion rule gives a reliable method for determining the minimum range of membrane potential to optimize the precision of fixed-point computing. The spiking CNN converted by our conversion rule is optimized to realize high-accuracy and low-latency classification, and is hardware-friendly towards dedicated hardware with limited computing resources.

The simulation results demonstrated that the converted spiking CNNs perform well as high classifi-

cation accuracy and low latency on MNIST, SVHN and CIFAR-10 dataset. The maximum accuracy loss over three datasets is lower than 0.4%, and about 39% of processing time is shortened at best. And spiking CNNs show almost no performance loss when all data is quantized into 16-bit fixed-point number. Comparing with previous work [29], our converted spiking CNN saved 38% of the power at best. Therefore, there is a great potential for realizing high energy efficiency on neuromorphic hardware. Finally, we found that the converted spiking CNN is robust against AWGN and SP noise and it inherits the robustness from its corresponding CNN, which could open a door to practical vision application.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61704167, 61434004), Beijing Municipal Science and Technology Project (Grant No. Z181100008918009), Youth Innovation Promotion Association Program, Chinese Academy of Sciences (Grant No. 2016107), and Strategic Priority Research Program of Chinese Academy of Science (Grant No. XDB32050200).

References

1 Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, 2012. 1097–1105
2 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. ArXiv: 1409.1556
3 Szegedy C, Liu W, Jia Y Q, et al. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. 1–9
4 He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 770–778
5 Shi C, Yang J, Han Y, et al. A 1000 fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array processor and self-organizing map neural network. IEEE J Solid-State Circ, 2014, 49: 2067–2082
6 Yang J, Yang Y X, Chen Z, et al. A heterogeneous parallel processor for high-speed vision chip. IEEE Trans Circ Syst Video Technol, 2018, 28: 746–758
7 Du Z D, Fasthuber R, Chen T S, et al. ShiDianNao: shifting vision processing closer to the sensor. SIGARCH Comput Archit News, 2016, 43: 92–104
8 Chen Y H, Krishna T, Emer J S, et al. Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J Solid-State Circ, 2017, 52: 127–138
9 Wang J Q, Yang Y X, Liu L Y, et al. High-speed target tracking system based on multi-interconnection heterogeneous processor and multi-descriptor algorithm. Sci China Inf Sci, 2019, 62: 069401
10 Andri R, Cavigelli L, Rossi D, et al. YodaNN: an architecture for ultralow power binary-weight CNN acceleration. IEEE Trans Comput-Aided Des Integr Circ Syst, 2018, 37: 48–60
11 Genc H, Zu Y, Chin T W, et al. Flying IoT: toward low-power vision in the sky. IEEE Micro, 2017, 37: 40–51
12 Shin D, Lee J, Lee J, et al. 14.2 DNPU: an 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In: Proceedings of 2017 IEEE International Solid-State Circuits Conference (ISSCC). New York: IEEE, 2017. 240–241
13 Ando K, Ueyoshi K, Orimo K, et al. BRein memory: a single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 TOPS at 0.6 W. IEEE J Solid-State Circ, 2018, 53: 983–994
14 Merolla P A, Arthur J V, Alvarez-Icaza R, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science, 2014, 345: 668–673
15 Esser S K, Merolla P A, Arthur J V, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. Proc Nat Acad Sci, 2016, 113: 11441–11446
16 Benjamin B V, Gao P, McQuinn E, et al. Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. Proc IEEE, 2014, 102: 699–716
17 Shen J C, Ma D, Gu Z H, et al. Darwin: a neuromorphic hardware co-processor based on spiking neural networks. Sci China Inf Sci, 2016, 59: 023401
18 Wu N J. Neuromorphic vision chips. Sci China Inf Sci, 2018, 61: 060421
19 Farabet C, Paz R, Pérez-Carrasco J, et al. Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convNets for visual processing. Front Neurosci, 2012, 6: 32
20 O'Connor P, Neil D, Liu S C, et al. Real-time classification and sensor fusion with a spiking deep belief network. Front Neurosci, 2013, 7: 178
21 Brader J M, Senn W, Fusi S. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. Neural Comput, 2007, 19: 2881–2912
22 Beyeler M, Dutt N D, Krichmar J L. Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule. Neural Netw, 2013, 48: 109–124
23 Querlioz D, Bichler O, Dollfus P, et al. Immunity to device variations in a spiking neural network with memristive nanodevices. IEEE Trans Nanotechnol, 2013, 12: 288–295
24 Diehl P U, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Front Comput Neurosci, 2015, 9: 99
25 Zhao B, Ding R X, Chen S S, et al. Feedforward categorization on AER motion events using cortex-like features in a

spiking neural network. IEEE Trans Neural Netw Learn Syst, 2015, 26: 1963–1978
26 Morrison A, Aertsen A, Diesmann M. Spike-timing-dependent plasticity in balanced random networks. Neural Comput, 2007, 19: 1437–1467
27 Zheng N, Mazumder P. Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity. IEEE Trans Neural Netw Learn Syst, 2018, 29: 4287–4302
28 Cao Y, Chen Y, Khosla D. Spiking deep convolutional neural networks for energy-efficient object recognition. Int J Comput Vis, 2015, 113: 54–66
29 Diehl P U, Neil D, Binas J, et al. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: Proceedings of 2015 International Joint Conference on Neural Networks (IJCNN). New York: IEEE, 2015. 1–8
30 Hunsberger E, Eliasmith C. Training spiking deep networks for neuromorphic hardware. 2016. ArXiv: 1611.05141
31 Rueckauer B, Lungu I A, Hu Y, et al. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Front Neurosci, 2017, 11: 682
32 Culurciello E, Etienne-Cummings R, Boahen K A. A biomorphic digital image sensor. IEEE J Solid-State Circ, 2003, 38: 281–294
33 Xu Y, Tang H, Xing J, et al. Spike trains encoding and threshold rescaling method for deep spiking neural networks. In: Proceedings of 2017 IEEE Symposium Series on Computational Intelligence (SSCI). New York: IEEE, 2017. 1–6
34 Akopyan F, Sawada J, Cassidy A, et al. TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. IEEE Trans Comput-Aided Des Integr Circ Syst, 2015, 34: 1537–1557
35 Arthur J V, Merolla P A, Akopyan F, et al. Building block of a programmable neuromorphic substrate: a digital neurosynaptic core. In: Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN). New York: IEEE, 2012. 1–8
36 Goodfellow I J, Bulatov Y, Ibarz J, et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks. 2013. ArXiv: 1312.6082
37 Vedaldi A, Lenc K. Matconvnet: convolutional neural networks for matlab. In: Proceedings of the 23rd ACM International Conference on Multimedia. New YorK: ACM, 2015: 689–692
38 Zhang T L, Zeng Y, Zhao D C, et al. Brain-inspired balanced tuning for spiking neural networks. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018. 1653–1659
39 Kulkarni S R, Rajendran B. Spiking neural networks for handwritten digit recognition-supervised learning and network optimization. Neural Netw, 2018, 103: 118–127
40 Tavanaei A, Maida A S. Bio-inspired spiking convolutional neural network using layer-wise sparse coding and stdp learning. 2016. ArXiv: 1611.03000

Appendix A Proof of the conversion theory

We aim to derive the relation between the firing rate $r_i^l(T)$ of the IF neuron in spiking CNN and the real-value activation a_i^l of the corresponding neuron in CNN. Starting from the total residual shown in (5), the firing rate of neuron i in layer l over time $T\mathrm{d}t$ can be rewritten with (5), then Eq. (4) is transferred into

$$r_i^l(T) = \frac{N_i^l(T)}{T} r_{\max} = \frac{\sum_{t=1}^{T} Z_i^l(t) - \varepsilon_i^l(T)}{\tau^l T} r_{\max}, \tag{A1}$$

where $\sum_{t=1}^{T} Z_i^l(t)$ is the total weighted inputs received over time $T\mathrm{d}t$. Because the accumulation of weighted input spikes in time and spatial scale is not correlative, the accumulation of input spike over $T\mathrm{d}t$ can be calculate first. So $\sum_{t=1}^{T} Z_i^l(t)$ is transferred into the sum up of weighted total number of spikes emitted by presynaptic neurons:

$$\sum_{t=1}^{T} Z_i^l(t) = \sum_{t=1}^{T} \sum_{j=1}^{J^{l-1}} W_{ij}^l \sum_{s\in S_j} \delta_j^{l-1}(t-s) = \sum_{j=1}^{J^{l-1}} W_{ij}^l \sum_{t=1}^{T} \sum_{s\in S_j} \delta_j^{l-1}(t-s) = \sum_{j=1}^{J^{l-1}} W_{ij}^l N_j^{l-1}(T). \tag{A2}$$

Then, according to (4), the total number of spikes $N_j^{l-1}(T)$ can be rewritten into the form containing firing rate, like

$$N_j^{l-1}(T) = T \frac{r_j^{l-1}(T)}{r_{\max}}. \tag{A3}$$

Substituting (A3) into (A2), $\sum_{t=1}^{T} Z_i^l(t)$ are ultimately denoted as an equation containing the firing rate of the neurons in previous layer:

$$\sum_{t=1}^{T} Z_i^l(t) = \frac{T}{r_{\max}} \sum_{j=1}^{J^{l-1}} W_{ij}^l r_j^{l-1}(T). \tag{A4}$$

Replacing the $\sum_{t=1}^{T} Z_i^l(t)$ in (A1) with (A4) and rearranging, the relationship about firing rate of IF neurons in two connected layers can be gotten:

$$r_i^l(T) = \frac{1}{\tau^l} \sum_{j=1}^{J^{l-1}} W_{ij}^l r_j^{l-1}(T) - \frac{\varepsilon_i^l(T)}{\tau^l T} r_{\max}. \tag{A5}$$

Eq. (A5) is a recursive expression and it can be unfolded by inserting the firing rate of neurons in previous layers iteratively until the known firing rate of input spike trains r_i^0:

$$r_i^l(T)=\frac{1}{\tau^l}\sum_{j=1}^{J^{l-1}}W_{ij}^l\left(\frac{1}{\tau^{l-1}}\sum_{j=1}^{J^{l-2}}W_{ij}^{l-1}\cdots\left(\frac{1}{\tau^1}\sum_{j=1}^{J^0}W_{ij}^1r_j^0(T)-\frac{\varepsilon_i^1(T)}{\tau^1T}r_{\max}\right)\cdots-\frac{\varepsilon_i^{l-1}(T)}{\tau^{l-1}T}r_{\max}\right)-\frac{\varepsilon_i^l(T)}{\tau^lT}r_{\max}. \tag{A6}$$

According to (A5), the term within the innermost brackets of (A6) is the firing rate of ith IF neuron in layer $l=1$, which is solved by substituting (7) into (A6):

$$\frac{1}{\tau^1}\sum_{j=1}^{J^0}W_{ij}^1r_j^0(T)-\frac{\varepsilon_i^1(T)}{\tau^1T}r_{\max}=r_i^1(T)=\frac{1}{\tau^1}\sum_{j=1}^{J^0}W_{ij}^1a_j^0r_{\max}-\frac{\varepsilon_i^1(T)}{\tau^1T}r_{\max}. \tag{A7}$$

Case 1: If $\sum_{j=1}^{J^0}W_{ij}^1a_j^0\geqslant 0$, the term in the right of equation (A7) can be rewritten as the form that contains ReLU function, and according to (2), it transfer into

$$r_i^1(T)=\frac{1}{\tau^1}\sum_{j=1}^{J^0}W_{ij}^1a_j^0r_{\max}-\frac{\varepsilon_i^1(T)}{\tau^1T}r_{\max}=\frac{r_{\max}}{\tau^1}\left(\max\left(0,\sum_{j=1}^{J^0}W_{ij}^1a_j^0\right)-\frac{\varepsilon_i^1(T)}{T}\right)=\frac{r_{\max}}{\tau^1}\left(a_i^1-\frac{\varepsilon_i^1(T)}{T}\right). \tag{A8}$$

Case 2: If $\sum_{j=1}^{J^0}W_{ij}^1a_j^0<0$, then $\max(0,\sum_{j=1}^{J^0}W_{ij}^1a_j^0)=0$. And Eq. (A7) can be written as

$$r_i^1(T)=\frac{1}{\tau^1}\sum_{j=1}^{J^0}W_{ij}^1a_j^0r_{\max}-\frac{\varepsilon_i^1(T)}{\tau^1T}r_{\max}=\frac{r_{\max}}{\tau^1}\left(\sum_{j=1}^{J^0}W_{ij}^1a_j^0(T)-\frac{\varepsilon_{ia}^1(T)}{T}\right)-\frac{\varepsilon_{ib}^1(T)}{\tau^1T}r_{\max}, \tag{A9}$$

where $\sum_{j=1}^{J^0}W_{ij}^1a_j^0(T)-\varepsilon_{ia}^1(T)/T=0$ and $\varepsilon_i^1(T)=\varepsilon_{ia}^1(T)+\varepsilon_{ib}^1(T)$. Therefore, Eq. (A9) can be rewritten as the form which also contains ReLU function, and according to (2), it transfer into

$$r_i^1(T)=\frac{1}{\tau^1}\sum_{j=1}^{J^0}W_{ij}^1a_j^0r_{\max}-\frac{\varepsilon_i^1(T)}{\tau^1T}r_{\max}=\frac{r_{\max}}{\tau^1}\left(\max\left(0,\sum_{j=1}^{J^0}W_{ij}^1a_j^0\right)-\frac{\varepsilon_{ib}^1(T)}{T}\right)=\frac{r_{\max}}{\tau^1}\left(a_i^1-\frac{\varepsilon_{ib}^1(T)}{T}\right). \tag{A10}$$

According to (A7) and (A9), ε_{ia}^1 can be solved, and based on (4) and (A2), it can be further rewritten as

$$\varepsilon_{ia}^1=\frac{T}{r_{\max}}\sum_{j=1}^{J^0}W_{ij}^1r_j^0(T)=\frac{T}{r_{\max}}\sum_{j=1}^{J^0}W_{ij}^1\frac{N_j^0(T)}{T\mathrm{d}t}, \tag{A11}$$

and ε_{ia}^1 equals

$$\varepsilon_{ia}^1=\sum_{t=1}^{T}Z_i^1(t). \tag{A12}$$

For ε_{ib}^1, it acts like the total residual membrane potential. When $\sum_{j=1}^{J^0}W_{ij}^1a_j^0<0$, the result of input pixels' value a_j^0 convoluted by W_{ij}^1 after ReLU function becomes the activation a_i^1 of ith neuron in first hidden layer, which equals to zero. So, the firing rate r_i^1 of corresponding IF neuron in spiking CNN should be equal to zero too. In this case, according to (A10), ε_{ib} is the part of membrane potential used to generate spikes, thereby causing approximation error:

$$r_i^1(T)=\frac{N_i^1(T)}{Tdt}=-\frac{\varepsilon_{ib}^1(T)}{\tau^1T}r_{\max}, \tag{A13}$$

and ε_{ib}^1 equals

$$\varepsilon_{ib}^1(T)=-N_i^1(T)\tau^1. \tag{A14}$$

Comparing the results of (A8) and (A10), we found that they are in similar form that the activation of corresponding neuron in CNN along with an additional error. When putting the result of (A8) or (A10) into (A6) during each iteration, the general form of the innermost term is listed in the following, which is similar with the cases we discuss above when $l=1$. So the general result for arbitrary layer l is

$$r_i^l(T)=\frac{1}{\tau^l}\sum_{j=1}^{J^{l-1}}W_{ij}^la_j^{l-1}(T)r_{\max}-\frac{\varepsilon_i^l(T)}{\tau^lT}r_{\max}=\begin{cases}\dfrac{r_{\max}}{\tau^l}\left(a_i^l-\dfrac{\varepsilon_i^l(T)}{T}\right), & \text{if } \sum_{j=1}^{J^{l-1}}W_{ij}^la_j^{l-1}\geqslant 0, \quad (A15)\\ \dfrac{r_{\max}}{\tau^l}\left(a_i^l-\dfrac{\varepsilon_{ib}^l(T)}{T}\right), & \text{if } \sum_{j=1}^{J^{l-1}}W_{ij}^la_j^{l-1}<0, \quad (A16)\end{cases}$$

where $\varepsilon_{ib}^l=-N_i^l(T)\tau^l$. Solving the recursive expression by inserting (A15) and (A16) iteratively, we finally obtain a general description of the relationship between firing rate $r_i^l(T)$ and real-value activation a_i^l:

$$r_i^l(T)=\frac{a_i^l}{\tau^l\tau^{l-1}\cdots\tau^1}r_{\max}-\Delta\varepsilon_i^l, \tag{A17}$$

where $\Delta\varepsilon_i^l$ is the approximation error.

Case 1: When $\sum_{j=1}^{J^{l-1}} W_{ij}^l a_j^{l-1} \geqslant 0$ for all layers $l = \{1, 2, \ldots, L\}$, approximation error equals to

$$\Delta\varepsilon_i^l = \frac{r_{\max}}{T}\left(\frac{\varepsilon_i^l(T)}{\tau^l} + \frac{1}{\tau^l\tau^{l-1}}\sum_{j=1}^{J^{l-1}} W_{ij}^l\varepsilon_j^{l-1}(T) + \cdots + \frac{1}{\tau^l\tau^{l-1}..\tau^1}\sum_{j=1}^{J^{l-1}} W_{ij}^l\cdots\sum_{j=1}^{J^1} W_{ij}^2\varepsilon_j^1(T)\right). \tag{A18}$$

Case 2: When $\sum_{j=1}^{J^{l-1}} W_{ij}^l a_j^{l-1} < 0$ for some layers l, approximation error equals to

$$\Delta\varepsilon_i^l = \frac{r_{\max}}{T}\left(\frac{\varepsilon_i^l(T)}{\tau^l} + \frac{1}{\tau^l\tau^{l-1}}\sum_{j=1}^{J^{l-1}} W_{ij}^l\varepsilon_{jb}^{l-1}(T) + \cdots + \frac{1}{\tau^l\tau^{l-1}..\tau^1}\sum_{j=1}^{J^{l-1}} W_{ij}^l\cdots\sum_{j=1}^{J^1} W_{ij}^2\varepsilon_{jb}^1(T)\right). \tag{A19}$$

Though there are some $\varepsilon_{jb}^l(T)$ existed in (A19), the result of two above cases are in same form. So, we adopt the result shown in (A18) as the approximation error $\Delta\varepsilon_i^l$ with the total residual membrane potential $\varepsilon_i^l(T)$ of each IF neuron within $T\mathrm{d}t$ equaling:

$$\varepsilon_i^l(T) = \begin{cases} \sum\limits_{s\in S_i}(V_i^l(s) - \tau^l) + \sum\limits_{u\in U_i}(V_i^l(u) - V_{\min}^l) + V_i^l(T), & \text{(A20)} \\ -N_i^l(T)\tau^l. & \text{(A21)} \end{cases}$$

And the final relation between the firing rate $r_i^l(T)$ of the IF neuron in spiking CNN and the real-value activation a_i^l of the corresponding neuron in CNN is described as (8) and (9).

Appendix B Pseudocode of conversion rule

Algorithm B1 Conversion rules. L is the number of layers. $\mathrm{conv}(W, a)$ is a function which represents that activation map a is convoluted by weight matrix W. $\mathrm{avg\_pool}(a)$ is a function which represents that activation map a is pooling with the average value. $\mathrm{layer}(l).\mathrm{type}$ indicates the type of layer l.

Data: Weight matrix W_{ij}^l from a well-trained CNN with biases fixed to 0. Normalized training set as input a^0.

Result: Obtain parameters f^l, scaled weights $W_{ij}^{l'}$ and $V_{\min}$.

$f^0 = 1$;
for $l = 1$ to L **do**
  for all ij **do**
    if $W_{ij}^l < 0$ **then**
      $W\_\mathrm{neg}_{ij}^l \leftarrow W_{ij}^l$;
    else
      $W\_\mathrm{neg}_{ij}^l \leftarrow 0$;
    end
  end
  if $\mathrm{layer}(l).\mathrm{type} ==$ 'conv' **then**
    $a^l \leftarrow \max(\mathrm{conv}(W^l, a^{l-1}), 0)$;
    $a\_\mathrm{neg}^l \leftarrow \mathrm{conv}(W\_\mathrm{neg}^l, a^{l-1})$;
    $M^l \leftarrow \max(a^l)$;
    $m^l \leftarrow \min(a\_\mathrm{neg}^l)$;
    $f^l \leftarrow \frac{M^l}{\prod_{n=0}^{l-1} f^n} \cdot 0.5$;
    $V_{\min}^l \leftarrow \frac{m^l}{\prod_{n=1}^{l} f^n}$;
  end
  if $\mathrm{layer}(l).\mathrm{type} ==$ 'pool' **then**
    $a^l \leftarrow \mathrm{avg\_pool}(a^{l-1})$;
    $f^l \leftarrow 1$;
    $V_{\min}^l \leftarrow 0$;
  end
end
$f^1 \leftarrow \lceil f^1 \rceil$;
$V_{\min} \leftarrow \min(V_{\min}^l)$;
for $l = 1$ to L **do**
  for all ij **do**
    $W_{ij}^{l'} = W_{ij}^l / f^l$;
  end
end