

Reducing ANN-SNN Conversion Error through Residual Membrane Potential

Zecheng Hao¹, Tong Bu^{1,2}, Jianhao Ding¹, Tiejun Huang^{1,2}, Zhaofei Yu^{1,2*}

¹ School of Computer Science, Peking University

² Institute for Artificial Intelligence, Peking University

zechenghao@pku.edu.cn, putong30@pku.edu.cn, djh01998@stu.pku.edu.cn, {tjhuang,yuzf12}@pku.edu.cn

Abstract

Spiking Neural Networks (SNNs) have received extensive academic attention due to the unique properties of low power consumption and high-speed computing on neuromorphic chips. Among various training methods of SNNs, ANN-SNN conversion has shown the equivalent level of performance as ANNs on large-scale datasets. However, unevenness error, which refers to the deviation caused by different temporal sequences of spike arrival on activation layers, has not been effectively resolved and seriously suffers the performance of SNNs under the condition of short time-steps. In this paper, we make a detailed analysis of unevenness error and divide it into four categories. We point out that the case of the ANN output being zero while the SNN output being larger than zero accounts for the largest percentage. Based on this, we theoretically prove the sufficient and necessary conditions of this case and propose an optimization strategy based on residual membrane potential to reduce unevenness error. The experimental results show that the proposed method achieves state-of-the-art performance on CIFAR-10, CIFAR-100, and ImageNet datasets. For example, we reach top-1 accuracy of 64.32% on ImageNet with 10-steps. To the best of our knowledge, this is the first time ANN-SNN conversion can simultaneously achieve high accuracy and ultra-low-latency on the complex dataset. Code is available at https://github.com/hzc1208/ANN2SNN_SRP.

Introduction

Spiking Neural Networks (SNNs), known as the third generation of artificial neural networks (Maass 1997), have shown distinctive properties and remarkable advantages of temporal information processing capability and high biological plausibility (Roy, Jaiswal, and Panda 2019). As each neuron delivers binary spikes only when the membrane potential reaches the threshold, the calculation in SNNs has the property of high-sparsity and multiplication-free, which brings unique advantages of low power consumption on neuromorphic chips (Schemmel et al. 2010; Furber et al. 2012; Merolla et al. 2014; Davies et al. 2018; Pei et al. 2019; De-Bole et al. 2019). Despite these, it remains challenging to train high-performance and low-latency SNNs.

Generally, two main approaches are proposed to train deep SNNs: (1) backpropagation with surrogate gradient (Neftci, Mostafa, and Zenke 2019; Kim, Kim, and Kim 2020), (2) ANN-SNN conversion (Cao, Chen, and Khosla 2015; Rueckauer et al. 2016). Unlike the surrogate gradient-based learning method that requires more GPU computing than ANN training, ANN-SNN conversion directly trains a source ANN and then converts it to an SNN by replacing the ReLU activation with IF neurons, which is regarded as the most effective way to train deep SNNs.

Although ANN-SNN conversion can achieve comparable performance as ANNs on large-scale datasets, it often requires large time-steps to match the firing rates of SNNs to the activation value of ANNs. There is still a performance gap between ANNs and SNNs under the condition of low latency. Many researchers have made great efforts to eliminate these errors to achieve high-performance converted SNNs with low latency (Han, Srinivasan, and Roy 2020; Ding et al. 2021; Ho and Chang 2021; Deng and Gu 2021; Li et al. 2021a; Li, Ma, and Furber 2022). However, unevenness error (Bu et al. 2022b), which refers to the deviation caused by different temporal sequences of spike arrival on activation layers, has not been effectively resolved and seriously suffers the performance of converted SNNs.

In this paper, by analyzing and deducing the distribution of unevenness error, we propose an optimization strategy based on residual membrane potential (SRP), which can decrease unevenness error and improve network performance effectively under low latency. Our main contributions are summarized as follows:

- 1 We divide unevenness error into four cases according to the outputs of ANNs and SNNs. We systematically analyze the distribution of unevenness error under these four cases and point out that the case of the ANN output being zero while the SNN output being larger than zero accounts for the largest percentage.
- 2 We theoretically establish the mathematical relationship between residual membrane potential and the specific case of unevenness error, and propose an optimization strategy based on residual membrane potential to reduce unevenness error.
- 3 We demonstrate the effectiveness of the proposed method on CIFAR-10/100, and ImageNet. Our method outper-

*Corresponding author

forms previous state-of-the-art and shows remarkable advantages on all tested datasets and network structures.

- 4 We show that our method is compatible with other ANN-SNN conversion methods and remarkably improves the performance when the time-steps are small.

Related Work

Due to the non-differentiable property of the spike firing mechanism, the training of SNNs is not as easy as ANNs. In recent years, two mainstream training methods have been proposed and widely used in obtaining deep SNNs.

Supervised learning of SNNs. To overcome the non-differentiable problem, smoothing and surrogate gradient methods (Huh and Sejnowski 2018; Jin, Zhang, and Li 2018; Wu et al. 2018; Shrestha and Orchard 2018; Zenke and Ganguli 2018; Bellec et al. 2018; Neftci, Mostafa, and Zenke 2019) have been proposed. These models maintain the network architecture unchanged in forward propagation and replace the Heaviside function of membrane potential with a differentiable function in back-propagation. On this basis, many researchers began to improve the performance of SNNs by analyzing the surrogate gradients (Zenke and Vogels 2021; Li et al. 2021b), designing appropriate network structures (Zheng et al. 2021; Fang et al. 2021) and loss functions (Deng et al. 2022; Guo et al. 2022), and using trainable membrane time constant (Fang et al. 2021). Event-driven learning is another type of learning method, which makes better use of the temporal information in SNNs based on the differentiation of spike firing time. SpikeProp (Bohte, Kok, and La Poutre 2002) was the first event-driven supervised learning algorithm for SNNs, which aimed to use linear smoothing approximation to overcome the non-differentiable problem of the relative membrane potential in spike firing time. Kheradpisheh et al. (2020) set the gradient as -1 to train shallow networks. To improve temporal learning precision, Zhang et al. (2020) attempted to retain inter-neuron and intra-neuron dependencies. By calculating the presynaptic and postsynaptic spike firing time, Mostafa et al. (2017) and Zhou et al. (2021) constructed new continuous activation function, which directly avoids non-differentiable problem. Kim et al. (2020) combined rate-coding and temporal-coding together in the back-propagation of SNNs. Despite these, the activation-based methods consume a lot of memory and computation, while the event-driven methods are limited to shallow networks.

ANN-SNN Conversion. ANN-SNN Conversion aims to map the parameters of the pretrained ANNs to SNNs and reduce the performance loss as much as possible. Cao et al. (2015) firstly trained ANNs with the ReLU activation function and then replaced the activation layers with spiking neurons. As the firing rate is not precise enough to match the output of respective activation layers in ANNs, researchers have proposed many optimization methods to reduce conversion error. Diehl et al. (2015) attempted to narrow the gap between ANNs and SNNs by scaling and normalizing weights. Rueckauer et al. (2017) proposed the “reset-by-subtraction” mechanism to retain the temporal information, which was also adopted in the following work (Han, Srinivasan, and Roy 2020; Deng and Gu 2021; Bu et al. 2022b).

Besides, many works focused on improving the performance by adopting various mechanism to dynamically adjust the threshold (Rueckauer et al. 2017; Sengupta et al. 2019; Han, Srinivasan, and Roy 2020; Ding et al. 2021; Stöckl and Maass 2021; Ho and Chang 2021; Wu et al. 2021). Kim et al. (2020) pointed out that lossless conversion can be realized when the time-step is sufficiently long. Therefore, recent works focused on improving performance under the condition of short time-steps by optimizing the weights, biases (Deng and Gu 2021; Li et al. 2021a) and initial membrane potential (Bu et al. 2022a), utilizing burst spikes (Li and Zeng 2022) and memory function (Wang et al. 2022), and designing quantization clip-floor-shift activation function for ANNs (Bu et al. 2022b). Although these methods can achieve high performance with few ($4-8$) time-steps on the CIFAR-10/100 datasets, they usually take large number of time-steps (usually > 16) to achieve comparable performance as ANNs on complex dataset like ImageNet, due to the effects of unevenness error. This paper aims to reduce unevenness error and obtain high-accuracy and ultra-low-latency SNNs on complex dataset.

Preliminary

Neuron Model

ANN neurons. For ANNs, the output \mathbf{a}^l of neurons in layer l is realized by a linear weighting and a nonlinear mapping:

$$\mathbf{a}^l = f(\mathbf{W}^l \mathbf{a}^{l-1}). \quad (1)$$

where \mathbf{W}^l refers to the weights between layer l and layer $l-1$. $f(\cdot)$ is the nonlinear activation function, which is often set as ReLU function.

SNN neurons. For SNNs, we consider the commonly used Integrate-and-Fire (IF) model (Diehl et al. 2015; Sengupta et al. 2019; Han, Srinivasan, and Roy 2020; Deng and Gu 2021; Bu et al. 2022b), the dynamics of which can be described by:

$$\mathbf{v}^l(t) = \mathbf{v}^l(t-1) + \mathbf{W}^l \theta^{l-1} \mathbf{s}^{l-1}(t) - \theta^l \mathbf{s}^l(t). \quad (2)$$

where $\mathbf{v}^l(t)$ denotes the membrane potential of neurons in layer l at time-step t , \mathbf{W}^l refers to the weights between layer l and layer $l-1$, and θ^l is the firing threshold of neurons in layer l . $\mathbf{s}^l(t)$ denotes the binary output spikes of neurons in layer l at time-step t , which is defined as:

$$\mathbf{s}^l(t) = H(\mathbf{u}^l(t) - \theta^l). \quad (3)$$

where $\mathbf{u}^l(t) = \mathbf{v}^l(t-1) + \mathbf{W}^l \theta^{l-1} \mathbf{s}^{l-1}(t)$ denotes the membrane potential of neurons before the trigger of a spike at time-step t , $H(\cdot)$ is the Heaviside step function. The output spike $\mathbf{s}^l(t)$ equals 1 if the membrane potential $\mathbf{u}^l(t)$ is larger than the threshold θ^l and otherwise 0. Note that in Eq. (2), we use the “reset-by-subtraction” mechanism (Rueckauer et al. 2017) to reduce information loss, which means the membrane potential $\mathbf{v}^l(t)$ is subtracted by the threshold value θ^l if the neuron fires.

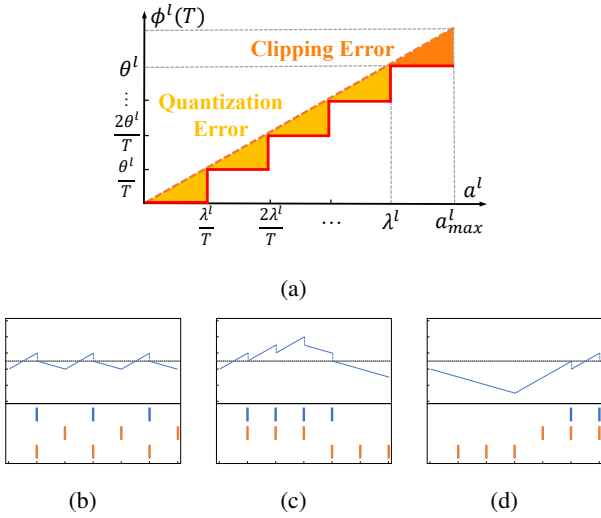


Figure 1: ANN-SNN conversion error includes: (a) Clipping error and Quantization error, (b)-(d) Unevenness error.

ANN-SNN Conversion

Relate ANN to SNN. The idea of ANN-SNN conversion is to relate the ReLU activation of analog neurons in ANNs to the firing rate (or postsynaptic potential) of spiking neurons in SNNs. Specifically, by summing Eq. (2) from time-step 1 to T and dividing T on both sides, we have:

$$\frac{v^l(T) - v^l(0)}{T} = \frac{\sum_{t=1}^T \mathbf{W}^l \theta^{l-1} s^{l-1}(t)}{T} - \frac{\sum_{t=1}^T \theta^l s^l(t)}{T}. \quad (4)$$

By using $\phi^l(T) = \frac{\sum_{t=1}^T \theta^l s^l(t)}{T}$ to denote the average postsynaptic potential, we will find the linear relationship between $\phi^l(T)$ and $\phi^{l-1}(T)$:

$$\phi^l(T) = \mathbf{W}^l \phi^{l-1}(T) - \frac{v^l(T) - v^l(0)}{T}. \quad (5)$$

From Eqs. (1) and (5), we note that if we can map the activation value a^l of analog neurons in ANNs to $\phi^l(T)$ of IF neurons in SNNs, then we will be able to train a source ANN with back-propagation (BP) and convert it to an SNN by replacing the ReLU activation with IF neurons, which is the core idea of ANN-SNN conversion. As Eqs. (1) and (5) are not exactly equal, there exists conversion error in general.

ANN-SNN conversion error. There are three main errors in ANN-SNN conversion (Bu et al. 2022b), including clipping error, quantization error, and unevenness error, which cause the performance gap between ANNs and SNNs. Specifically,

- **Clipping error** denotes the error caused by different value ranges of ANNs and SNNs. For ANNs, the output a^l belongs to a real number interval $[0, a_{max}^l]$ with a_{max}^l denoting the maximum value of a^l . For SNNs, the output $\phi^l(T)$ belongs to a finite discrete set $S_T =$

$$\left\{ \frac{\theta^l i}{T} \mid i \in [0, T] \wedge i \in \mathbb{N} \right\} \text{ due to } \phi^l(T) = \frac{\sum_{t=1}^T \theta^l s^l(t)}{T}.$$

As illustrated in Fig. 1(a), if we set λ^l as an actual threshold in ANNs to map the maximum value θ^l in S_T of SNNs, a^l can be mapped to $\phi^l(T)$ by $\phi^l(T) = \text{clip}\left(\frac{\theta^l}{T} \left\lfloor \frac{a^l T}{\lambda^l} \right\rfloor, 0, \theta^l\right)$ with $\lfloor \cdot \rfloor$ denoting the floor function. Then the output $a^l \in [\lambda^l, a_{max}^l]$ of the ANNs will be mapped to the same value θ^l , which will cause the so-called clipping error.

- **Quantization error** is generated when mapping the continuous value of a^l in ANNs to the discrete value of $\phi^l(T)$ in SNNs. As shown in Fig. 1(a), when $a^l \in \left[\frac{k\lambda^l}{T}, \frac{(k+1)\lambda^l}{T}\right)$ ($k = 0, 1, \dots, T-1$), the corresponding value of $\phi^l(T)$ will always be $\frac{k\theta^l}{T}$.
- **Unevenness error** refers to the deviation between a^l and $\phi^l(T)$ due to different temporal sequences of spike arrival on activation layers. Ideally, we expect the timing of receive spikes from the previous layer to be even. However, in reality, the spiking timing will be uneven when the spikes deliver to the deep layer, which will cause more spikes or fewer spikes than expected. An example is illustrated in Fig. 1(b)-(d), in which two presynaptic neurons in layer $l-1$ are connected to a postsynaptic neuron in layer l . We assume $\mathbf{W}^l = [2 \ -1]$, $\mathbf{a}^{l-1} = \phi^{l-1}(T) = [0.5 \ 0.5]^T$, $\theta^l = \lambda^l = 1$, $T = 6$, then we can get that the output a^l in ANN is $a^l = \text{ReLU}(\mathbf{W}^l \mathbf{a}^{l-1}) = 0.5$. Under the ideal situation (Fig. 1(b)), $\phi^l(T) = \frac{\sum_{t=1}^T \theta^l s^l(t)}{T} = \frac{3}{6} = 0.5$, which is exactly the same as ANN output. However, when the input spike timing changes (Fig. 1(c)-(d)), $\phi^l(T)$ may become larger or smaller than a^l in fact.

Eliminating conversion error. The clipping and quantization errors can be eliminated by modifying the activation function of source ANNs. Specifically, the trainable clipping layer technique has been used to directly map the trainable upper bound of ANNs to the threshold of SNNs, so that the clipping error is zero (Ho and Chang 2021). Besides, the quantization clip-floor activation function was proposed to replace the ReLU activation function in source ANNs (Bu et al. 2022b), which can better approximate the activation function of SNNs and eliminate quantization error. Despite this, it is still challenging to reduce unevenness error, which seriously hampers the performance of SNNs under the condition of low time latency. Therefore, this paper aims to address unevenness error and explore high-performance ANN-SNN conversion with ultra-low latency.

Methods

In this section, we make a detailed analysis of unevenness error and divide it into four situations, then we count out the distribution of unevenness error and point out that the first case plays a more important role among the four kinds of errors. Based on this, we show the residual membrane potential is essential to ANN-SNN conversion and propose the optimization strategy to reduce unevenness error.

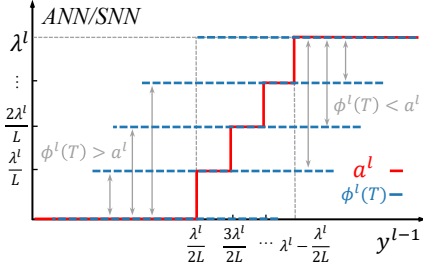


Figure 2: Four cases of unevenness error.

Dividing Unevenness Error into Four Situations

Here we divide unevenness error into different cases to analyze. We suppose that the ANN and SNN receive the same input to layer l , that is, $\mathbf{a}^{l-1} = \phi^{l-1}(T)$, and then analyze the error between the ANN and SNN in layer l . For simplicity, we use $\mathbf{y}^{l-1} = \mathbf{W}^l \mathbf{a}^{l-1} = \mathbf{W}^l \phi^{l-1}(T)$ to substitute the weighted input to layer l for both ANN and SNN. The conversion error is defined as the output of converted SNN subtracting the output of source ANN, which can be simplified according to Eqs. (1) and (5):

$$\begin{aligned} \mathbf{Error}^l &= \phi^l(T) - \mathbf{a}^l \\ &= \mathbf{y}^{l-1} - \frac{\mathbf{v}^l(T) - \mathbf{v}^l(0)}{T} - f(\mathbf{y}^{l-1}). \end{aligned} \quad (6)$$

In order to facilitate the analysis of unevenness error, we consider using the quantization clip-floor-shift (QCFS) function proposed in (Bu et al. 2022b) to replace the ReLU activation function in source ANNs, that is:

$$f(\mathbf{y}^{l-1}) = \lambda^l \text{clip} \left(\frac{1}{L} \left\lfloor \frac{\mathbf{y}^{l-1} L}{\lambda^l} + \frac{1}{2} \right\rfloor, 0, 1 \right). \quad (7)$$

where L is the quantization step of ANNs, λ^l represents the trainable threshold in l -th layer of ANNs and is generally mapped to the threshold of SNN, that is, $\theta^l = \lambda^l$. The QCFS activation function better approximates the activation function of SNNs, and therefore could eliminate clipping and quantization errors (see Appendix for the detailed proof). Thus, the error (\mathbf{Error}^l) here is caused by unevenness error.

For further analysis, we compare the output \mathbf{a}^l of ANN and $\phi^l(T)$ of SNN in layer l when receiving the same input \mathbf{y}^{l-1} . As shown in Fig. 2, according to the weighted input \mathbf{y}^{l-1} to layer l and the sign (positive and negative) of \mathbf{Error}^l , unevenness error can be divided into the following four categories:

- **Case 1:** $\mathbf{a}^l = 0, \phi^l(T) > \mathbf{a}^l$, which means that the input $\mathbf{y}^{l-1} < \lambda^l/2L$, the output of ANN equals 0 but the output of SNN is larger than 0 (fires more spikes as expected).
- **Case 2:** $0 < \mathbf{a}^l < \lambda^l, \phi^l(T) > \mathbf{a}^l$, which means that the input $\lambda^l/2L \leq \mathbf{y}^{l-1} < \lambda^l - \lambda^l/2L$, the output of SNN is larger than that of ANN (fires more spikes as expected).
- **Case 3:** $0 < \mathbf{a}^l < \lambda^l, \phi^l(T) < \mathbf{a}^l$, which means that the input $\lambda^l/2L \leq \mathbf{y}^{l-1} < \lambda^l - \lambda^l/2L$, the output of SNN is smaller than that of ANN (fires fewer spikes as expected).

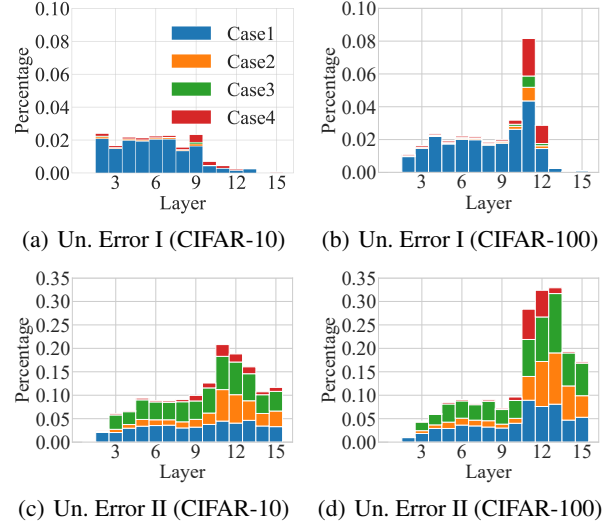


Figure 3: The distribution of Unevenness Error I/II in each layer of VGG-16. (a)-(b): Unevenness Error I, (c)-(d) Unevenness Error II.

- **Case 4:** $\mathbf{a}^l = \lambda^l, \phi^l(T) < \mathbf{a}^l$, which means that input $\mathbf{y}^{l-1} \geq \lambda^l - \lambda^l/2L$, the output of ANN equals λ^l but the output of SNN is smaller than λ^l (fires fewer spikes as expected).

Note that here we use the scalars \mathbf{a}^l and $\phi^l(T)$ to represent the outputs of two corresponding neurons in ANN and SNN, respectively.

Analyze the Distribution of Unevenness Error

The discussion above of unevenness error (\mathbf{Error}^l) in layer l is based on the assumption that $\mathbf{a}^{l-1} = \phi^{l-1}(T)$. However, in actual situations, \mathbf{a}^{l-1} is not always equal to $\phi^{l-1}(T)$, as unevenness error existed before the l -th layer has already caused the deviation between the respective output of ANNs and SNNs. Therefore, we attempt to give a more detailed discussion about unevenness error here.

Definition 1. Unevenness Error I: $\mathbf{a}^{l-1} = \phi^{l-1}(T) \wedge \mathbf{a}^l \neq \phi^l(T)$, this condition is consistent with the original definition of unevenness error and represents the error completely generated by the l -th layer.

Definition 2. Unevenness Error II: $\mathbf{a}^l \neq \phi^l(T)$ whether \mathbf{a}^{l-1} equals $\phi^{l-1}(T)$ or not. This condition reflects the cumulative effect of unevenness error.

For Unevenness Error II of the l -th layer, it can be considered as the joint effect of Unevenness Error I in the l -th layer (when $\mathbf{a}^{l-1} = \phi^{l-1}(T)$) and Unevenness Error II in the $l-1$ -th layer (when $\mathbf{a}^{l-1} \neq \phi^{l-1}(T)$). Therefore, Unevenness Error I is the essential reason that affects the performance of SNNs, which desires to be eliminated.

We further analyze the distribution of Unevenness Error I/II in four situations, and mine the main part of unevenness error. Here, we train the source ANNs with QCFS activation function (Eq. (7)) and VGG-16 structure on CIFAR-10/100

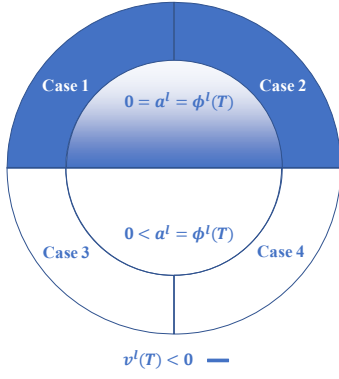


Figure 4: Illustration of all cases. The small concentric circle denotes that there is no error, while the circular ring denotes the four cases of unevenness error. The semicircle (blue) represents all the cases covered by $v^l(T) < 0$.

datasets and then convert them to SNNs. For unevenness Error I in layer l , we force the input \mathbf{a}^{l-1} to ANN to be equal to the input $\phi^{l-1}(T)$ to SNN, that is, $\mathbf{a}^{l-1} = \phi^{l-1}(T)$, and recalculate the output \mathbf{a}^l of ANN. Then we can compute the distribution of four kinds of Unevenness Error I by analyzing the values of \mathbf{a}^l and $\phi^l(T)$ of all neurons. For Unevenness Error II in layer l , we directly compute the distribution of four kinds of unevenness error by analyzing the values of \mathbf{a}^l and $\phi^l(T)$ of all neurons. The results are illustrated in Fig. 3, we have the following observation.

Observation 1. From Fig. 3(c)-3(d), we note that all the four cases mentioned above have a remarkable effect on Unevenness Error II. However, for Unevenness Error I that determines Unevenness Error II, we can find that **Case 1** ($a^l = 0, \phi^l(T) > a^l$) plays a more important role among four kinds of errors, which has a significant impact in almost every layer (Fig. 3(a)-3(b)). Therefore, our main goal is to optimize this case to alleviate the situation of firing additional spikes.

Optimization Strategy Based on Residual Membrane Potential (SRP)

Here we propose an optimization strategy to reduce unevenness error based on residual membrane potential. Our goal is to reduce Unevenness Error I of **Case 1** ($a^l = 0, \phi^l(T) > a^l$). However, in the practical application of SNNs, we cannot directly obtain the specific value of a^l , which becomes an obstacle to further determining the category of unevenness Error I. Fortunately, we find that we can compare a^l with $\phi^l(T)$ according to the value of residual membrane potential $v^l(T)$, and we have the following theorem:

Theorem 1. *Supposing that an ANN with QCFS activation (Eq. (7)) is converted to an SNN with $L = T$ and $\lambda^l = \theta^l$, and the ANN and SNN receive the same weighted input $\mathbf{y}^{l-1} = \mathbf{W}^l \mathbf{a}^{l-1} = \mathbf{W}^l \phi^{l-1}(T)$, then we will have the following conclusions:*

(i) If $a^l = 0, v^l(T) < 0$ is the sufficient condition of $\phi^l(T) \geq a^l$. In addition, $v^l(T) < 0$ is also the necessary condition of

Algorithm 1: The optimization strategy based on residual membrane potential (SRP)

Require: Time-step to calculate residual membrane potential τ ; Time-step to test dataset T ; Pretrained QCFS ANN model $f_{\text{ANN}}(\mathbf{W}, \lambda)$; Dataset D .

Ensure: SNN model $f_{\text{SNN}}(\mathbf{W}, \mathbf{v}, \theta)$.

```

1: for  $l = 1$  to  $f_{\text{ANN}}.\text{layers}$  do
2:    $f_{\text{SNN}}.\theta^l \leftarrow f_{\text{ANN}}.\lambda^l$ 
3:    $f_{\text{SNN}}.\mathbf{v}^l(0) \leftarrow \frac{1}{2} f_{\text{SNN}}.\theta^l$ 
4:    $f_{\text{SNN}}.\mathbf{W}^l \leftarrow f_{\text{ANN}}.\mathbf{W}^l$ 
5: end for
6: for length of Dataset  $D$  do
7:   Sample minibatch (data,label) from  $D$ 
8:   for  $t = 1$  to  $\tau$  do
9:      $f_{\text{SNN}}(\text{data})$ 
10:  end for
11:  for  $l = 1$  to  $f_{\text{SNN}}.\text{layers}$  do
12:     $f_{\text{SNN}}.\text{mask}^l \leftarrow (f_{\text{SNN}}.\mathbf{v}^l(\tau) \geq 0)$ 
13:     $f_{\text{SNN}}.\mathbf{v}^l(\tau) \leftarrow \frac{1}{2} f_{\text{SNN}}.\theta^l$ 
14:  end for
15:  for  $t = 1$  to  $T$  do
16:    for  $l = 1$  to  $f_{\text{SNN}}.\text{layers}$  do
17:      data  $\leftarrow f_{\text{SNN}}^l(\text{data}) \cdot f_{\text{SNN}}.\text{mask}^l$ 
18:    end for
19:  end for
20: end for
21: return  $f_{\text{SNN}}(\mathbf{W}, \mathbf{v}, \theta)$ 

```

$\phi^l(T) > a^l$.

(ii) If $a^l > 0, v^l(T) < 0$ is the sufficient and necessary condition of $\phi^l(T) > a^l$.

The proof is provided in the appendix. Theorem 1 implies that if $v^l(T) < 0$, we can infer that there is no error ($a^l = \phi^l(T)$) or there exist Unevenness Error I of **Case 1** and **Case 2** ($\phi^l(T) > a^l$), which is illustrated in the blue semicircle of Fig. 4. However, we are unable to determine whether it belongs to **Case 1** ($a^l = 0, \phi^l(T) > a^l$) or **Case 2** ($0 < a^l < \lambda^l, \phi^l(T) > a^l$). In fact, we have shown in Fig. 3(a)-3(b) that **Case 1** accounts for the largest percentage and **Case 2** can be approximately neglected. Thus we can conclude that when $v^l(T) < 0$, there exist mainly Unevenness Error I of **Case 1** or there is not any error.

Based on the above analysis, we can use the residual membrane potential to reduce unevenness error. If the residual membrane potential $v^l(T)$ is smaller than zero, we should set the corresponding output $\phi^l(T)$ to zero, so as to make $\phi^l(T) = a^l = 0$. In practice, we first train an ANN with QCFS activation and then convert it to an SNN. In the inference stage, we use a two-stage strategy. For the first stage, we readout residual membrane potential $v^l(\tau)$ with τ denoting the actual time-step we choose. If $v^l(\tau) < 0$ for a neuron, we set it to the dead neuron, which will not work in the next stage. For the second stage, we run the SNN and report the result for a given time-step T . We name our optimization strategy as SRP in the following content. The specific algorithm flow about SRP is shown in Algorithm 1.

Method	Arch	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64
RMP	VGG-16	93.63%	-	-	-	-	-	60.30%	90.35%
RTS	VGG-16	95.72%	-	-	-	-	-	76.24%	90.64%
TCL*	VGG-16	94.57%	-	-	-	-	-	93.64%	94.26%
SNNC-AP	VGG-16	95.72%	-	-	-	-	-	93.71%	95.14%
OPI	VGG-16	94.57%	-	-	-	90.96%	93.38%	94.20%	94.45%
QCFS	VGG-16	95.52%	88.41%	91.18%	93.96%	94.95%	95.40%	95.54%	95.55%
SRP($\tau = 4$)	VGG-16	95.52%	93.80%	94.47%	95.32%	95.52%	95.44%	95.42%	95.40%
RTS	ResNet-18	95.46%	-	-	-	-	-	84.06%	92.48%
SNNC-AP	ResNet-18	95.46%	-	-	-	-	-	94.78%	95.30%
OPI	ResNet-18	96.04%	-	-	-	75.44%	90.43%	94.82%	95.92%
QCFS	ResNet-18	95.64%	88.84%	91.75%	93.83%	95.04%	95.56%	95.67%	95.63%
SRP($\tau = 4$)	ResNet-18	95.64%	94.59%	95.06%	95.25%	95.60%	95.55%	95.55%	95.58%
TSC	ResNet-20	91.47%	-	-	-	-	-	-	69.38%
OPI	ResNet-20	92.74%	-	-	-	66.24%	87.22%	91.88%	92.57%
QCFS	ResNet-20	91.77%	62.43%	73.20%	83.75%	89.55%	91.62%	92.24%	92.35%
SRP($\tau = 4$)	ResNet-20	91.77%	86.37%	88.73%	90.51%	91.37%	91.64%	91.72%	91.80%

* Self-implementation results.

Table 1: Comparison between the proposed method and previous works on CIFAR-10 dataset

Experiments

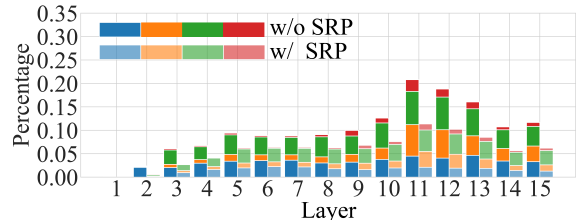
In this section, we evaluate the performance of our methods for image classification tasks on CIFAR-10 (LeCun et al. 1998), CIFAR-100 and ImageNet (Deng et al. 2009) datasets under the network architecture of ResNet-18, ResNet-20, ResNet-34 (He et al. 2016) and VGG-16 (Simonyan and Zisserman 2014). We compare our method with previous state-of-the-art ANN-SNN conversion methods, including RMP (Han, Srinivasan, and Roy 2020), TSC (Han and Kaushik 2020), RTS (Deng and Gu 2021), OPI (Bu et al. 2022a), SNNC-AP (Li et al. 2021a), TCL (Ho and Chang 2021), QCFS (Bu et al. 2022b) and SNM (Wang et al. 2022).

SRP Can Reduce Unevenness Error

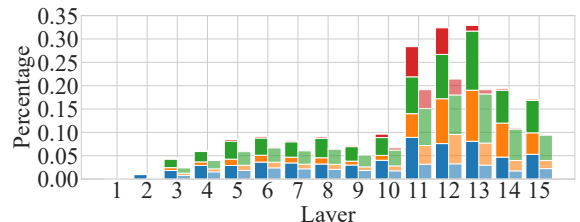
We first test whether the proposed SRP can reduce unevenness error. We train VGG-16 on the CIFAR-10/100 datasets and then convert it to SNNs with/without SRP. As Unevenness Error II can evaluate the overall performance of SNNs, we compare the distribution of Unevenness Error II before and after using SRP in each layer and show the results in Fig. 5. One can find that our proposed method can reduce the percentage of unevenness Error II that existed in each layer significantly. For VGG-16 on CIFAR-10, SRP can reduce the percentage of unevenness Error II that existed in last-5 layers by 9.44%, 8.60%, 7.55%, 5.06%, and 5.47%. For CIFAR-100, the corresponding decreased percentage is 9.21%, 10.96%, 13.77%, 8.34%, and 7.48%.

Comparison with the State-of-the-art

We compare our method with the state-of-the-art ANN-SNN conversion methods to validate the effectiveness of our method. For fair comparison, we use the performance of SRP on $T = t$ to compare with the performance of other state-of-the-arts on $T = t + \tau$.



(a) VGG-16, CIFAR-10



(b) VGG-16, CIFAR-100

Figure 5: Comparison of the distributions of Unevenness Error II before and after using SRP in each layer.

Tab. 1 shows the performance of our proposed method on the CIFAR-10 dataset. As CIFAR-10 is a relatively easy dataset, the advantages of SRP are not as remarkable as those on CIFAR-100 and ImageNet dataset. For VGG-16 architecture, we achieve 95.32% with 4 time-steps ($\tau = 4$), which is 4.36% higher than OPI and 0.37% higher than QCFS on $T = 8$. The accuracy of our method ($\tau = 4$) on $T = 4$ has outperformed the performance of other previous works on $T = 64$, which include RMP, TCL, RTS,

Method	Arch	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64
RTS	VGG-16	77.89%	-	-	-	-	-	7.64%	21.84%
SNNC-AP	VGG-16	77.89%	-	-	-	-	-	73.55%	76.64%
TCL*	VGG-16	76.32%	-	-	-	-	-	52.30%	71.17%
SNM	VGG-16	74.13%	-	-	-	-	-	71.80%	73.69%
OPI	VGG-16	76.31%	-	-	-	60.49%	70.72%	74.82%	75.97%
QCFS	VGG-16	76.28%	-	63.79%	69.62%	73.96%	76.24%	77.01%	77.10%
SRP ($\tau = 4$)	VGG-16	76.28%	71.52%	74.31%	75.42%	76.25%	76.42%	76.45%	76.37%
RMP	ResNet-20	68.72%	-	-	-	-	-	27.64%	46.91%
OPI	ResNet-20	70.43%	-	-	-	23.09%	52.34%	67.18%	69.96%
QCFS	ResNet-20	69.94%	-	19.96%	34.14%	55.37%	67.33%	69.82%	70.49%
SRP ($\tau = 4$)	ResNet-20	69.94%	46.48%	53.96%	59.34%	62.94%	64.71%	65.50%	65.82%

* Self-implementation results.

Table 2: Comparison between the proposed method and previous works on CIFAR-100 dataset

Method	Arch	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64
SNNC-AP	VGG-16	75.36%	-	-	-	-	-	63.64%	70.69%
SNM	VGG-16	73.18%	-	-	-	-	-	64.78%	71.50%
OPI	VGG-16	74.85%	-	-	-	6.25%	36.02%	64.70%	72.47%
QCFS	VGG-16	74.29%	-	-	-	19.12%	50.97%	68.47%	72.85%
SRP ($\tau = 14$)	VGG-16	74.29%	50.37%	61.37%	66.47%	68.37%	69.13%	69.35%	69.43%
SNNC-AP	ResNet-34	75.66%	-	-	-	-	-	64.54%	71.12%
QCFS	ResNet-34	74.32%	-	-	-	35.06%	59.35%	69.37%	72.35%
SRP ($\tau = 8$)	ResNet-34	74.32%	57.78%	64.32%	66.71%	67.62%	68.02%	68.40%	68.61%

Table 3: Comparison between the proposed method and previous works on ImageNet dataset

SNM and SNNC-AP. For ResNet-18, we obtain 95.25% after using 4 time-steps ($\tau = 4$), which is 19.81% higher than OPI and 0.21% higher than QCFS on $T = 8$ respectively. For ResNet-20, the performance of SRP ($\tau = 4$) on $T = 4$ can outperform OPI and QCFS on $T = 8$ with 24.27% and 0.96%.

Tab. 2 reports the results on the CIFAR-100 dataset. For low latency inference ($T < 16$), our model outperforms all the other methods with the same time-step setting. For VGG-16, we achieve 75.42% top-1 accuracy with 4 time-steps ($\tau = 4$), whereas the methods of OPI and QCFS reach 60.49% and 73.96% accuracy at the end of 8 time-steps. Moreover, we achieve an accuracy of 71.52% using only 1 time-step. For ResNet-20, the proposed method achieve 59.34% top-1 accuracy with 4 time-steps ($\tau = 4$), which is 36.25% higher than OPI and 3.97% higher than QCFS, respectively.

We further evaluate whether our method can be generalized to the large-scale dataset. Here we test the performance of our method on the ImageNet dataset and report the results in Tab. 3. One can find that our method shows its superiority more obviously in the large-scale dataset. For VGG-16, SRP achieves the accuracy of 61.37% with 2 time-steps ($\tau = 14$), whereas the methods of OPI and QCFS reach 36.02% and 50.97% accuracy at the end of 16 time-steps. For ResNet-34, we obtain 67.62% accuracy with 8 time-steps ($\tau = 8$),

which outperforms QCFS (59.35% when $T = 16$) by 8.27% accuracy. All these results demonstrate that our method outperforms the previous conversion methods.

The Effect of Parameter τ

Here we test the effects of parameter τ in our SRP. There is a trade-off between performance and latency. Larger τ can make a more accurate estimation of the residual membrane potential, thereby increasing the performance of SNNs. However, these advantages come with the price of long latency. If τ is set much smaller than the quantization step L in the QCFS activation function of ANN, the precondition $\mathbf{y}^{l-1} = \mathbf{W}^l \mathbf{a}^{l-1} = \mathbf{W}^l \phi^{l-1}(T)$ in Theorem 1 will be violated, which will decrease the performance of SRP.

For a more detailed analysis, we train a source ANN on CIFAR-100/ImageNet dataset and then convert it to SNNs with different τ . The performance of converted SNNs is shown in Fig. 6 and Tab. 4. For CIFAR-100, the quantization step L is set to 4 on VGG-16 and 8 on ResNet 20. For ImageNet, the quantization step L is set to 16 on VGG-16 and 8 on ResNet-34. One can find that as τ increases, the accuracy of SNNs generally improves. When τ is nearly equal to L , the performance of SNNs stabilizes gradually, which implies that a too large τ will increase latency but cannot improve the accuracy remarkably. Besides, the results demonstrate that despite a significant change in τ , our SRP method

Method	Arch	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64
Baseline	VGG-16	76.28%	57.50%	63.79%	69.62%	73.96%	76.24%	77.01%	77.10%
SRP($\tau = 2$)	VGG-16	76.28%	68.55%	71.24%	73.21%	74.16%	74.91%	75.13%	75.26%
SRP($\tau = 4$)	VGG-16	76.28%	71.52%	74.31%	75.42%	76.25%	76.42%	76.45%	76.37%
SRP($\tau = 8$)	VGG-16	76.28%	72.15%	75.20%	76.25%	76.66%	77.03%	77.14%	77.15%
SRP($\tau = 16$)	VGG-16	76.28%	72.97%	75.66%	76.55%	77.03%	77.08%	77.08%	77.07%
Baseline	ResNet-20	69.94%	13.19%	19.96%	34.14%	55.37%	67.33%	69.82%	70.49%
SRP($\tau = 2$)	ResNet-20	69.94%	30.99%	39.55%	47.72%	55.13%	59.41%	60.76%	61.80%
SRP($\tau = 4$)	ResNet-20	69.94%	46.48%	53.96%	59.34%	62.94%	64.71%	65.50%	65.82%
SRP($\tau = 8$)	ResNet-20	69.94%	55.29%	63.07%	66.44%	68.22%	68.93%	68.95%	68.96%
SRP($\tau = 16$)	ResNet-20	69.94%	58.32%	65.16%	68.07%	69.50%	69.27%	69.47%	69.49%

Table 4: Comparison about different τ of SRP on CIFAR-100 dataset

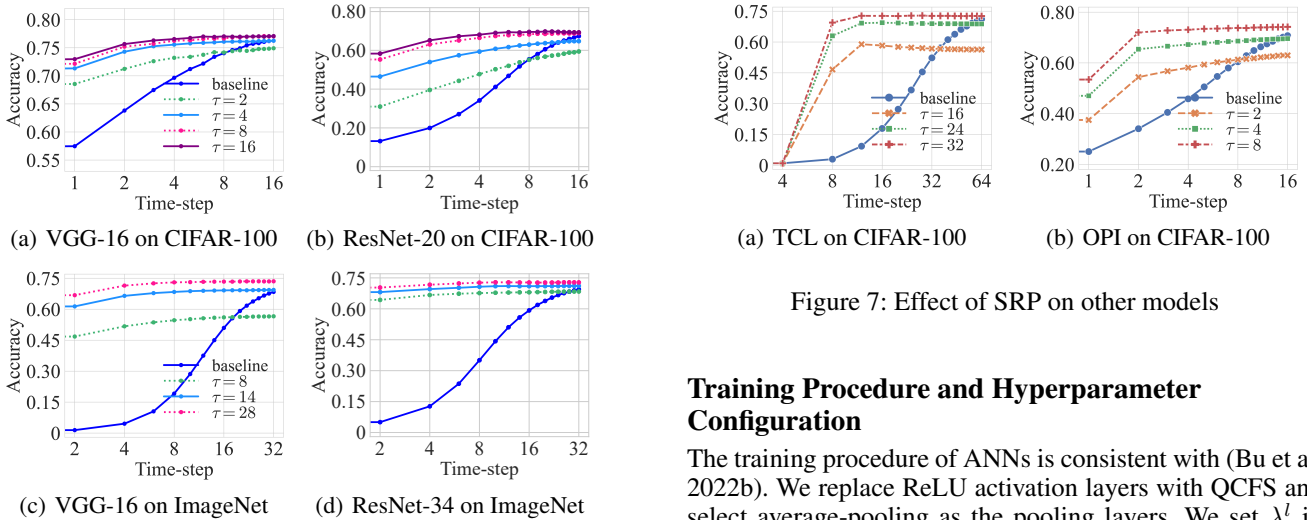


Figure 6: The effect of parameter τ

still maintains good generalization capabilities.

In addition, for the tasks with different allowable time-steps, we can set different τ to improve the performance of SNNs. Specifically, we consider using relatively small τ under low time latency and use larger τ for a longer time-step.

Apply SRP to Other ANN-SNN Models

In this section, we experimentally prove that SRP can be generalized to other conversion methods. To verify this, we consider TCL (Ho and Chang 2021) and OPI (Bu et al. 2022a) as the basic conversion method and then add SRP to those methods. As shown in Fig. 7, we can find that the performance of both TCL and OPI increases after using SRP. Specifically, for TCL model, we achieve 63.03% with 8 time-steps ($\tau = 24$), which outperforms baseline (52.30% when $T = 32$) with 10.73% accuracy. For OPI model, we obtain 67.24% with 4 time-steps ($\tau = 4$), which is 6.79% higher than baseline (60.45% when $T = 8$). These results show that SRP can apply to other conversion models and remarkably improve classification accuracy.

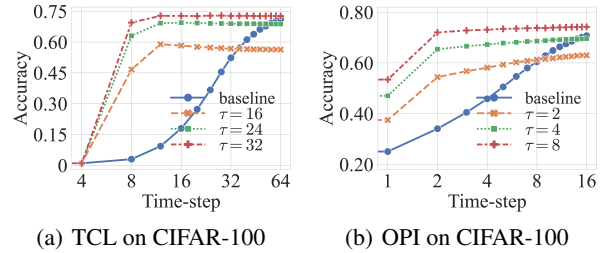


Figure 7: Effect of SRP on other models

Training Procedure and Hyperparameter Configuration

The training procedure of ANNs is consistent with (Bu et al. 2022b). We replace ReLU activation layers with QCFS and select average-pooling as the pooling layers. We set λ^l in each activation layer as trainable threshold and $v^l(0) = \frac{1}{2}\theta^l$ in IF Neuron. For CIFAR-10, we set $L = 4$ on all network architectures. For CIFAR-100, we set $L = 4$ for VGG-16 and $L = 8$ for ResNet-20. For ImageNet, we set $L = 16$ for VGG-16 and $L = 8$ for ResNet-34.

We use Stochastic Gradient Descent (Bottou 2012) as our training optimizer and cosine decay scheduler (Loshchilov and Hutter 2017) to adjust the learning rate. We set the momentum parameter as 0.9. The initial learning rate is set as 0.1 (CIFAR-10/ImageNet) or 0.02 (CIFAR-100) and weight decay is set as 5×10^{-4} (CIFAR-10/100) or 1×10^{-4} (ImageNet). In addition, we also use common data normalization and data enhancement techniques (DeVries and Taylor 2017; D Cubuk et al. 2019; Li et al. 2021a) for all datasets. During the procedure of adopting SRP, for CIFAR-10/100, we recommend $\tau = 4$ for all network architecture. For ImageNet, we recommend $\tau = 14$ for VGG-16 and $\tau = 8$ for ResNet-34.

Conclusion

In this paper, we systematically analyze the cases and distribution of unevenness error and propose an optimization strategy based on residual membrane potential to reduce unevenness error. By introducing the temporal information

of SNNs, our method achieves state-of-the-art accuracy on CIFAR-10, CIFAR-100, and ImageNet datasets, with fewer time-steps. Our method can alleviate the performance gap between ANNS and SNN, and promote the practical application of SNNs on neuromorphic chips.

Appendix

Eliminating Clipping and Quantization Errors

In the main text, we claim that the QCFS activation function (Eq. (7)) better approximates the activation function of SNNs and could eliminate clipping and quantization errors. Here we give the detailed proof.

Eliminating Clipping error. Since the QCFS activation function of Eq. (7) includes the clip operation, we can know that the value range of \mathbf{a}^l is $[0, \lambda^l]$. Therefore, the interval $[\lambda^l, a_{max}^l]$ has been clipped out and the clipping error is completely eliminated.

Eliminating Quantization error. According to (Bu et al. 2022b), if we take no account of unevenness error, we can consider that the timing of receive spikes is even, then we will have:

$$\phi^l(T) = \text{clip} \left(\frac{\theta^l}{T} \left\lfloor \frac{\mathbf{y}^{l-1}T + \mathbf{v}^l(0)}{\theta^l} \right\rfloor, 0, \theta^l \right). \quad (8)$$

where we use $\mathbf{y}^{l-1} = \mathbf{W}^l \mathbf{a}^{l-1} = \mathbf{W}^l \phi^{l-1}(T)$ to substitute the weighted input to layer l for both ANN and SNN. When we use QCFS function, we generally set $\lambda^l = \theta^l, \mathbf{v}^l(0) = \theta^l/2$. As we do not consider unevenness error here and clipping error has already been eliminated, it is reasonable to believe that the deviation between \mathbf{a}^l and $\phi^l(T)$ here is mainly caused by quantization error. Therefore, we use $\mathbf{a}^l - \phi^l(T)$ to represent the degree of quantization error. When $\mathbf{y}^{l-1} \in [0, \theta^l], \forall T, L$ we can derive the following conclusion:

$$\begin{aligned} & \mathbb{E}(\mathbf{a}^l - \phi^l(T)) \\ &= \mathbb{E} \left(\frac{\lambda^l}{L} \left\lfloor \frac{\mathbf{y}^{l-1}L}{\lambda^l} + \frac{1}{2} \right\rfloor - \frac{\theta^l}{T} \left\lfloor \frac{\mathbf{y}^{l-1}T + \mathbf{v}^l(0)}{\theta^l} \right\rfloor \right) \\ &= 0. \end{aligned} \quad (9)$$

The conclusion in Eq. (9) has already been proved by (Bu et al. 2022b). From the perspective of mathematical expectation, we have proved that the QCFS function can eliminate quantization error effectively.

Proof of Theorem

Theorem 1. *Supposing that an ANN with QCFS activation (Eq. (7)) is converted to an SNN with $L = T$ and $\lambda^l = \theta^l$, and the ANN and SNN receive the same weighted input $\mathbf{y}^{l-1} = \mathbf{W}^l \mathbf{a}^{l-1} = \mathbf{W}^l \phi^{l-1}(T)$, then we will have the following conclusions:*

(i) *If $a^l = 0, v^l(T) < 0$ is the sufficient condition of $\phi^l(T) \geq a^l$. In addition, $v^l(T) < 0$ is also the necessary condition of $\phi^l(T) > a^l$.*

(ii) *If $a^l > 0, v^l(T) < 0$ is the sufficient and necessary condition of $\phi^l(T) > a^l$.*

Proof. The proof of necessity in (i). By combining Eq.(5), $\mathbf{v}^l(0) = \frac{\theta^l}{2}$ and $\mathbf{y}^{l-1} = \mathbf{W}^l \phi^{l-1}(T)$, we will have

$$\phi^l(T) = \mathbf{y}^{l-1} - \frac{1}{T} \left(\mathbf{v}^l(T) - \frac{\theta^l}{2} \right). \quad (10)$$

On the one hand, when $a^l = 0$, according to the preconditions $L = T, \lambda^l = \theta^l$ mentioned in Theorem 1 and the definition of QCFS activation function in Eq.(7), we can know that $\mathbf{y}^{l-1} < \frac{\theta^l}{2T}$. On the other hand, as $\phi^l(T) > a^l = 0$, then we will have

$$\phi^l(T) = \mathbf{y}^{l-1} - \frac{1}{T} \left(\mathbf{v}^l(T) - \frac{\theta^l}{2} \right) \geq \frac{\theta^l}{T}, \mathbf{y}^{l-1} < \frac{\theta^l}{2T}. \quad (11)$$

According to Eq. (11), $\frac{1}{T}(\mathbf{v}^l(T) - \frac{\theta^l}{2}) < -\frac{\theta^l}{2T}$, then we can derive $\mathbf{v}^l(T) < 0$.

The proof of sufficiency in (i). As $\phi^l(T)$ belongs to a finite discrete set $S_T = \{\frac{\theta^l i}{T} | i \in [0, T] \wedge i \in \mathbb{N}\}$, $\phi^l(T) \geq 0$ at any time. If $a^l = 0, \phi^l(T) \geq a^l$.

The proof of necessity in (ii). For $a^l > 0$, we can set $a^l = \frac{k\theta^l}{T}, k \in \mathbb{N} \wedge 0 < k < T$, then we can have $\mathbf{y}^{l-1} \in [\frac{\theta^l}{T}(k - \frac{1}{2}), \frac{\theta^l}{T}(k + \frac{1}{2})]$. When $a^l < \phi^l(T)$, from Eq.(10) we can have

$$\begin{aligned} \phi^l(T) &= \mathbf{y}^{l-1} - \frac{1}{T} \left(\mathbf{v}^l(T) - \frac{\theta^l}{2} \right) \geq \frac{(k+1)\theta^l}{T}, \\ \mathbf{y}^{l-1} &\in \left[\frac{\theta^l}{T}(k - \frac{1}{2}), \frac{\theta^l}{T}(k + \frac{1}{2}) \right). \end{aligned} \quad (12)$$

From Eq. (12), it is obvious that $\frac{1}{T}(\mathbf{v}^l(T) - \frac{\theta^l}{2}) < -\frac{\theta^l}{2T}$, which means that $\mathbf{v}^l(T) < 0$.

The proof of sufficiency in (ii). When $a^l = \theta^l, \mathbf{y}^{l-1} \geq \frac{\theta^l}{T}(T - \frac{1}{2})$. By combining $\mathbf{v}^l(T) < 0$ with $\mathbf{y}^{l-1} \geq \frac{\theta^l}{T}(T - \frac{1}{2})$, from Eq.(10) we can have $\phi^l(T) > \theta^l$, which is impossible. Therefore, a^l must be smaller than θ^l .

When $0 < a^l < \theta^l$, By combining $\mathbf{v}^l(T) < 0$ with $a^l = \frac{k\theta^l}{T}, \mathbf{y}^{l-1} \in [\frac{\theta^l}{T}(k - \frac{1}{2}), \frac{\theta^l}{T}(k + \frac{1}{2})]$, we will have:

$$\begin{aligned} \phi^l(T) &= \mathbf{y}^{l-1} - \frac{1}{T} \left(\mathbf{v}^l(T) - \frac{\theta^l}{2} \right) \\ &> \mathbf{y}^{l-1} + \frac{\theta^l}{2T} \geq \frac{k\theta^l}{T} = a^l. \end{aligned} \quad (13)$$

□

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 62176003 and No. 62088102.

References

Bellec, G.; Salaj, D.; Subramoney, A.; Legenstein, R.; and Maass, W. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*.

- Bohte, S. M.; Kok, J. N.; and La Poutre, H. 2002. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4): 17–37.
- Bottou, L. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, 421–436. Springer.
- Bu, T.; Ding, J.; Yu, Z.; and Huang, T. 2022a. Optimized Potential Initialization for Low-latency Spiking Neural Networks. In *AAAI Conference on Artificial Intelligence*.
- Bu, T.; Fang, W.; Ding, J.; Dai, P.; Yu, Z.; and Huang, T. 2022b. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In *International Conference on Learning Representations*.
- Cao, Y.; Chen, Y.; and Khosla, D. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1): 54–66.
- D Cubuk, E.; et al. 2019. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1): 82–99.
- DeBole, M. V.; Taba, B.; Amir, A.; Akopyan, F.; Andreopoulos, A.; Risk, W. P.; Kusnitz, J.; Otero, C. O.; Nayak, T. K.; Appuswamy, R.; et al. 2019. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5): 20–29.
- Deng, J.; Socher, R.; Li, L.; Li, K.; and Li, F. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Deng, S.; and Gu, S. 2021. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*.
- Deng, S.; Li, Y.; Zhang, S.; and Gu, S. 2022. Temporal Efficient Training of Spiking Neural Network via Gradient Reweighting. *arXiv preprint arXiv:2202.11946*.
- DeVries, T.; and Taylor, G. W. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.
- Diehl, P. U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.-C.; and Pfeiffer, M. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, 1–8. IEEE.
- Ding, J.; Yu, Z.; Tian, Y.; and Huang, T. 2021. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *International Joint Conference on Artificial Intelligence*, 2328–2336.
- Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2661–2671.
- Furber, S. B.; Lester, D. R.; Plana, L. A.; Garside, J. D.; Painkras, E.; Temple, S.; and Brown, A. D. 2012. Overview of the spiNNaker system architecture. *IEEE Transactions on Computers*, 62(12): 2454–2467.
- Guo, Y.; Tong, X.; Chen, Y.; Zhang, L.; Liu, X.; Ma, Z.; and Huang, X. 2022. RecDis-SNN: Rectifying Membrane Potential Distribution for Directly Training Spiking Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 326–335.
- Han, B.; and Kaushik, R. 2020. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, 388–404.
- Han, B.; Srinivasan, G.; and Roy, K. 2020. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 13558–13567.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.
- Ho, N.-D.; and Chang, I.-J. 2021. TCL: an ANN-to-SNN conversion with trainable clipping layers. In *ACM/IEEE Design Automation Conference (DAC)*, 793–798. IEEE.
- Huh, D.; and Sejnowski, T. J. 2018. Gradient Descent for Spiking Neural Networks. In *Advances in Neural Information Processing Systems*.
- Jin, Y.; Zhang, W.; and Li, P. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *Advances in Neural Information Processing Systems*.
- Kheradpisheh, S. R.; and Masquelier, T. 2020. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(06): 2050027.
- Kim, J.; Kim, K.; and Kim, J.-J. 2020. Unifying Activation- and Timing-based Learning Rules for Spiking Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 19534–19544.
- Kim, S.; Park, S.; Na, B.; and Yoon, S. 2020. Spiking-YOLO: spiking neural network for energy-efficient object detection. In *AAAI Conference on Artificial Intelligence*.
- LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.
- Li, C.; Ma, L.; and Furber, S. 2022. Quantization Framework for Fast Spiking Neural Networks. *Frontiers in Neuroscience*, 16.
- Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021a. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, 6316–6325.
- Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021b. Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks. In *Advances in Neural Information Processing Systems*, 23426–23439.
- Li, Y.; and Zeng, Y. 2022. Efficient and Accurate Conversion of Spiking Neural Network with Burst Spikes. In *International Joint Conference on Artificial Intelligence*.

- Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.
- Maass, W. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9): 1659–1671.
- Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197): 668–673.
- Mostafa, H. 2017. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7): 3227–3235.
- Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6): 51–63.
- Pei, J.; Deng, L.; Song, S.; Zhao, M.; Zhang, Y.; Wu, S.; Wang, G.; Zou, Z.; Wu, Z.; He, W.; et al. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767): 106–111.
- Roy, K.; Jaiswal, A.; and Panda, P. 2019. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784): 607–617.
- Rueckauer, B.; Lungu, I.-A.; Hu, Y.; and Pfeiffer, M. 2016. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*.
- Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; and Liu, S.-C. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11: 682.
- Schemmel, J.; Brüderle, D.; Gröbl, A.; Hock, M.; Meier, K.; and Millner, S. 2010. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *IEEE International Symposium on Circuits and Systems*, 1947–1950. IEEE.
- Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 13: 95.
- Shrestha, S. B.; and Orchard, G. 2018. SLAYER: Spike Layer Error Reassignment in Time. In *Advances in Neural Information Processing Systems*.
- Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Stöckl, C.; and Maass, W. 2021. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 3(3): 230–238.
- Wang, Y.; Zhang, M.; Chen, Y.; and Qu, H. 2022. Signed Neuron with Memory: Towards Simple, Accurate and High-Efficient ANN-SNN Conversion. In *International Joint Conference on Artificial Intelligence*.
- Wu, J.; Chua, Y.; Zhang, M.; Li, G.; Li, H.; and Tan, K. C. 2021. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.
- Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12: 331.
- Zenke, F.; and Ganguli, S. 2018. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6): 1514–1541.
- Zenke, F.; and Vogels, T. P. 2021. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4): 899–925.
- Zhang, W.; and Li, P. 2020. Temporal spike sequence learning via backpropagation for deep spiking neural networks. In *Advances in Neural Information Processing Systems*, 12022–12033.
- Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going deeper With directly-trained larger spiking neural networks. In *AAAI Conference on Artificial Intelligence*, 11062–11070.
- Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S. T.; and Sanyal, A. 2021. Temporal-coded deep spiking neural network with easy training and robust performance. In *AAAI Conference on Artificial Intelligence*, 11143–11151.